8-2021

Lightweight mutual authentication and privacy preservation schemes for IOT systems.

Samah Mansour
University of Louisville

LIGHTWEIGHT MUTUAL AUTHENTICATION AND PRIVACY PRESERVATION SCHEMES FOR IOT SYSTEMS

By

Samah Mansour

M.S., Grand Valley State University, USA, 2017

PhD., University of Louisville, USA 2006

M.A., University of Louisville, USA 2004

B.A., Ain Shams University, Egypt 1999

A Dissertation

Submitted to the Faculty of the

J.B. Speed School of Engineering of the University of Louisville

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in Computer Science and Engineering

Department of Computer Engineering and Computer

Science University of Louisville

Louisville, Kentucky

August 2021

LIGHTWEIGHT MUTUAL AUTHENTICATION AND PRIVACY PRESERVATION SCHEMES FOR IOT SYSTEMS

By

Samah Mansour

M.S., Grand Valley State University, USA, 2017

PhD., University of Louisville, USA 2006

M.A., University of Louisville, USA 2004

B.A., Ain Shams University, Egypt 1999

A Dissertation Approved On

July 23, 2021

by the following Dissertation Committee:

_____

Dr. Adrian P. Lauf, Dissertation Director

_____

Dr. Adel Elmaghraby

_____

Dr. Mehmed Kantardzic

_____

Dr. Michael Losavio

DEDICATION

This dissertation is dedicated to parents Mr. Sayed Mansour and Mrs. Afaf Aly, my husband Dr. Mostafa El-Said, and my sons Adam El-Said and Ian El-Said, who always supported me and also encouraged me to pursue my Ph.D

ACKNOWLEDGMENTS

I would like to recognize my advisor, Dr. Adrian P. Lauf, for his mentorship. He provided me ample guidance and encouragement as I navigated my way toward this dissertation topic. Under his direction, I was given the creative freedom to grow and build the skills necessary to succeed in the future as a scholar and researcher. I am immensely grateful for all his efforts.

I would also like to express my appreciation for the assistance provided by my committee members: Mehmed Kantardzic, Adel Elmaghraby, and Michael Losavio. They all offered various guidance and suggestions to improve my work and to ensure my success. Their insightfulness and wisdom ultimately helped to refine my work into its present form.

ABSTRACT

LIGHTWEIGHT MUTUAL AUTHENTICATION AND PRIVACY PRESERVATION SCHEMES FOR IOT SYSTEMS

Samah Mansour

July 10, 2021

Internet of Things (IoT) presents a holistic and transformative approach for providing services in different domains. IoT creates an atmosphere of interaction between humans and the surrounding physical world through various technologies such as sensors, actuators, and the cloud. Theoretically, when everything is connected, everything is at risk. The rapid growth of IoT with the heterogeneous devices that are connected to the Internet generates new challenges in protecting and preserving user's privacy and ensuring the security of our lives. IoT systems face considerable challenges in deploying robust authentication protocols because some of the IoT devices are resource-constrained with limited computation and storage capabilities to implement the currently available authentication mechanism that employs computationally expensive functions. The limited capabilities of IoT devices raise significant security and privacy concerns, such as ensuring personal information confidentiality and integrity and establishing end-to-end authentication and secret key generation between the communicating device to guarantee secure communication among the communicating devices.

The ubiquity nature of the IoT device provides adversaries more attack surfaces which can lead to tragic consequences that can negatively impact our everyday connected lives. According to [1], authentication and privacy protection are essential security requirements. Therefore, there is a critical need to address these rising security and privacy concerns to ensure IoT systems' safety. This dissertation identifies gaps in the literature and presents new mutual authentication and privacy preservation schemes that fit the needs of resource-constrained devices to improve IoT security and privacy against common attacks. This research enhances IoT security and privacy by introducing lightweight mutual authentication and privacy preservation schemes for IoT based on hardware biometrics using PUF, Chained hash PUF, dynamic identities, and user's static and continuous biometrics. The communicating parties can anonymously communicate and mutually authenticate each other and locally establish a session key using dynamic identities to ensure the user's unlinkability and untraceability. Furthermore, virtual domain segregation is implemented to apply security policies between nodes. The chained-hash PUF mechanism technique is implemented as a way to verify the sender's identity.

At first, this dissertation presents a framework called "A Lightweight Mutual Authentication and Privacy-Preservation framework for IoT Systems" and this framework is considered the foundation of all presented schemes. The proposed framework integrates software and hardware-based security approaches that satisfy the NIST IoT security requirements for data protection and device identification. Also, this dissertation presents an architecture called "PUF Hierarchal Distributed Architecture" (PHDA), which is used to perform the device name resolution.

Based on the proposed framework and PUF architecture, three lightweight privacy-preserving and mutual authentication schemes are presented. The Three different schemes are introduced to accommodate both stationary and mobile IoT devices as well as local and distributed nodes. The first scheme is designed for the smart homes domain, where the IoT devices are stationary, and the controller node is local. In this scheme, there is direct communication between the IoT nodes and the controller node. Establishing mutual authentication does not require the cloud service's involvement to reduce the system latency and offload the cloud traffic. The second scheme is designed for the industrial IoT domain and used smart poultry farms as a use case of the Industrial IoT (IIoT) domain. In the second scheme, the IoT devices are stationary, and the controller nodes are hierarchical and distributed, supported by machine-to-machine (M2M) communication. The third scheme is designed for smart cities and used IoV fleet vehicles as a use case of the smart cities domain. During the roaming service, the mutual authentication process between a vehicle and the distributed controller nodes represented by the Roadside Units (RSUs) is completed through the cloud service that stores all vehicle's security credentials. After that, when a vehicle moves to the proximity of a new RSU under the same administrative authority of the most recently visited RSU, the two RSUs can cooperate to verify the vehicle's legitimacy. Also, the third scheme supports driver static and continuous authentication as a driver monitoring system for the sake of both road and driver safety.

The security of the proposed schemes is evaluated and simulated using two different methods: security analysis and performance analysis. The security analysis is implemented through formal security analysis and informal security analysis. The formal

analysis uses the Burrows–Abadi–Needham logic (BAN) and model-checking using the automated validation of Internet security protocols and applications (AVISPA) toolkit. The informal security analysis is completed by: (1) investigating the robustness of the proposed schemes against the well-known security attacks and analyze its satisfaction with the main security properties; and (2) comparing the proposed schemes with the other existing authentication schemes considering their resistance to the well-known attacks and their satisfaction with the main security requirements. Both the formal and informal security analyses complement each other.

The performance evaluation is conducted by analyzing and comparing the overhead and efficiency of the proposed schemes with other related schemes from the literature. The results showed that the proposed schemes achieve all security goals and, simultaneously, efficiently and satisfy the needs of the resource-constrained IoT devices.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION AND DISSERTATION OVERVIEW

The rapid increase in using the Internet means that more than 49% of the world population is reachable at the click of a button, providing people with economic and social opportunities. Internet of Things (IoT) is an old and new term at the same time. Kevin Ashton mentioned this term in 1999 during a presentation at Proctor & Gamble to link the idea of radio frequency identification (RFID) to the new topic of the Internet [1]. Internet of Things (IoT) is an emerging technology that is experiencing continuous growth. IoT creates a small and smart interconnected world that enables human-to-devise communication and device-to-device communication. IoT established an intelligent environment in which there is a constant exchange of data and services. It is expected that more than 21 billion devices will be interconnected by 2025 [2]. The predicted exponential growth of IoT depends partially on both Moore's law and Koomey's law. Moore's law states that the number of transistors on a chip doubles approximately every two years [3]. As a result, powerful computers can be developed on the same sized chip.

Koomey's law explains that the number of computations per kilowatt-hour roughly doubles every one and a half years [4]. Kevin Ashton describes that these two laws enabled us to create powerful and energy-efficient computers. Those two laws are telling us that we can perform the same amount of computations on a smaller chip while consuming less energy. This means that computers become more computationally effective as they become more energy efficient. The output is a small, powerful, and energy-efficient computer that allows us to provide more advanced services with less chip size at lower energy consumption. [5].

There is no universally accepted definition for IoT. The main idea of IoT is the connection of many objects to communication and the exchange of data as well as enabling people to communicate with these objects. IEEE's definition of IoT is: "Internet of Things envisions a self-configuring, adaptive, complex network that interconnects' things' to the Internet through the use of standard communication protocols. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration" [6].

According to the IEEE IoT definition [6], the IoT system consists of three main layers: (1) a front layer that consists of the front-end devices such as sensors that sense and gather data about the environment (2) a back layer that consists of back-end computing and storage devices which provides analytics and intelligence, (3) a network layer that provides the communication infrastructure which connects front-end sensors to back end servers.

The Telecommunication Standardization Sector [7] defined IoT as "… a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies".

According to the ITU, the physical world refers to anything in our physical world that can be sensed, actuated, and connected. The virtual world refers to data and information that can be collected, processed, stored, and accessed. By integrating both the physical and virtual world, we can use a diverse collection of smart objects to collect,

process, and store data from any environment to make decisions and take actions at any time.

In IoT, classical computing and communication devices as well as a wide range of other gadgets that we use in our daily life, are connected in different domains such as smart homes, health care, wearable devices, smart city, and others. Consequently, A large number of devices will be connected. Those devices will have smart capabilities to collect, analyze and even make decisions without any human interaction. Cisco forecasts that the IoT market will be $14.4 trillion by 2022, with the majority invested in improving customer experience. Other areas of investment are reducing the time-to-market ($3T), cost reduction strategies ($2.5T), improving supply chain and logistics ($2.7T), and increasing employee productivity ($2.5T). Moreover, Cisco found that 50% of IoT activity is in manufacturing, transformation, smart cities, and consumer markets [8] as presented in figure 1.

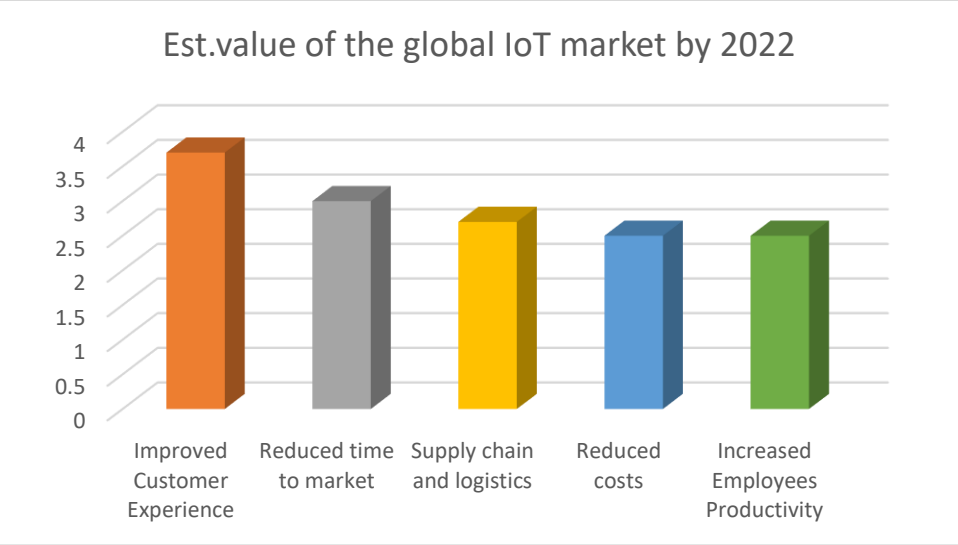


Figure 1: Cisco Forecast of the IoT Growth

Analysts at McKinsey institutes [9] estimate that the annual economic impact of IoT will grow from $3.9 trillion to $11.1 trillion by 2025. The growth will be in different domains such as manufacturing, smart cities, retail environments, cars, and the human body, as shown in figure 2[9].

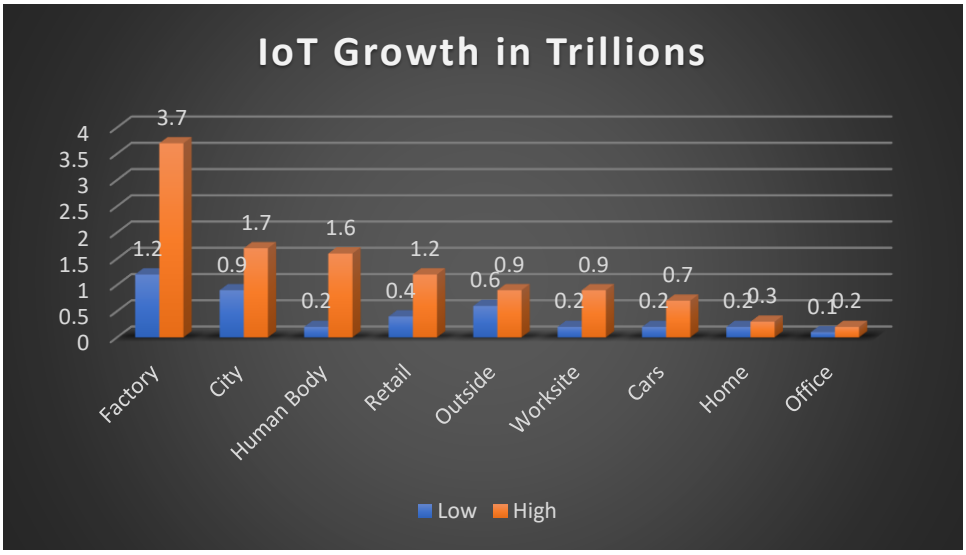


Figure 2: McKensey Forecast to IoT Growth Per Domain

IoT technology is deployed in different domains such as surveillance devices in smart cities, patient monitoring devices in health care, connected cars in vehicular networks, wearable devices, and smart homes. IoT systems have a heterogeneous mix of diverse devices, networking and communication protocols, and design methodologies, making them complex subjects for designing a security framework. Therefore, security is an essential requirement in the IoT environment, especially authentication, which is of high interest given the damage that could happen from a malicious unauthenticated device in an IoT system.

1.1 Challenges

The communication among IoT devices over the Internet and on local networks needs to be secured to gain user trust. The implementation of the security system in IoT requires the building of a Root of Trust (ROT). ROT can be defined as an element of a system that offers services to verify the achievement of security-related goals such as confidentiality, messages integrity, and authentication of the sending and receiving devices [10]

Security and privacy are key challenges to make the IoT ecosystem. Typically, IoT devices are characterized by their constrained resources, consisting of a small amount of memory and computational power, which are not architecturally designed to employ robust security techniques [11]. Besides, IoT devices are often deployed in open and public places, which may cause them to be vulnerable to physical and cloning attacks. We cannot deal with IoT in an ad-hoc manner using reactive approaches and being on the defensive side. Instead, a proactive approach is required by being on the offensive side.

Also, any security solution designed for IoT devices must be computationally efficient and able to detect any security violation of the IoT devices without sacrificing security.

The traditional security and protection techniques, including currently available cryptographic solutions, secure protocols, and privacy preservation techniques, cannot be re-implemented to secure resource-constrained devices. IoT systems must be used to ease our lives and not harm our security and privacy. Therefore, any new solution must be resource-efficient to fit devices with limited processing power, memory, and communication bandwidth and to preserve user's privacy.

1.2 Problem Definition

IoT devices are pervasive in our everyday life. IoT spending is now forecast to pass the $1 trillion projected market in 2022, reaching $1.1 trillion in 2023, for an annual growth rate of 13%. Both smart homes and connected vehicles are estimated to be the second-largest source of IoT spending [12]. Although there is a growth in the IoT market, there are still concerns about the security and privacy of such systems. Because IoT devices can monitor every aspect of people's lives, the user still has legitimate privacy concerns. Moreover, companies worry about reputational damage from data getting into the wrong hands, and governments fear security risks.

IoT security is different for several reasons. First, IoT devices are characterized by their constrained resources consisting of a small amount of memory and computational power, which are not architecturally designed to support robust security techniques. Second, IoT systems have a heterogeneous mix of diverse devices and networking protocols, which makes it complex to design a security framework given

raised scalability and interoperability issues. Third, IoT devices are connected to various components, such as actuators or monitoring systems. The collected data by those devices is vital for the functioning of the operation of the whole system. Fourth, IoT nodes store a secret key in non-volatile memory (NVM), flash memory, or battery-operated static random-access memory (SRAM). It has been proven that physical attacks such as invasive, semi-invasive, or side-channel attacks and software attacks such as malware can expose the key and lead to security breaches [13].

The IoT system can be subject to two main types of attacks: (i) attacks against the IoT devices such as physical attacks on the device; (ii) attacks against the communications such as a man in the middle attack that may lead to listening, modifying, injecting, or re-routing the message. Also, the adversary can either compromise existing IoT devices or inject malicious devices that can be used to launch different types of attacks. Therefore, appropriate countermeasures must be taken to secure the IoT systems from those attacks.

Traditional password-based or secret-key-based authentication schemes, in which a shared secret is the only authentication factor, are not enough for addressing the security problems, especially if the IoT devices are mobile devices. Many IoT devices have weak passwords using manufacturer default passwords, making them vulnerable to botnets, such as the Mirai IoT botnet [14]. Also, an adversary with physical access to an IoT device can launch various physical or side-channel attacks to acquire the device's secret key, thus compromising the device and the entire system. Moreover, hackers can connect rogue devices to IoT networks using fake or multiple identities without being caught. Furthermore, IoT devices may rely on user biometrics as another factor of authentication.

However, the biometric template affects the user's privacy if a hacking activity compromises it. Therefore, protecting the user's privacy is a necessity for biometrics-based authentication systems.

Privacy preservation is another major security component. The lack of privacy preservation may allow the attacker to track and identify the user or the IoT device identify, leading to a privacy breach. The lack of protecting users or IoT devices in smart homes and health care will allow the adversary to identify the user's lifestyle and infer sensitive information, which can potentially be life-threatening to the users. Therefore, anonymity, unlinkability, undetectability, unobservability, and pseudonymity are important characteristics to ensure privacy preservation and prevent attackers from obtaining the user's real identity or the IoT device [15].

Two of the core NIST requirements for securable IoT devices are device identification and data protection. Security, privacy, and trust are three main elements that must be satisfied in any IoT application [16]. Device's authentication, data confidentiality, data integrity, trust management, and privacy are key challenges in designing a secure IoT.

Device authentication is the process of verifying the device's identity. Without strong authentication, attackers can capture sensitive data and execute malicious actions. Data integrity is the process of maintaining and ensuring the originality, accuracy, and consistency of the data. Trust management guarantees trust in the devices and ensures that data has been handled and processed in compliance with user needs and rights. Privacy is essential to ensure that the user's data and credentials are preserved.

Ensuring security and privacy in the IoT systems can be achieved using: (1) encryption that makes sensitive data useless to the adversary and (2) authentication techniques that reject malicious devices that can cause harmful attacks.

There is a need for new robust authentication techniques to avoid the problems mentioned above and, at the same time, can work on resource constraint devices. In this dissertation, different IoT authentication architectures are presented that ensure security and privacy preservation and present computational efficiency solutions to fit the nature of the IoT devices.

1.3 Solution Approach

To address the aforementioned issues, authentication techniques that can deal with a mix of diverse and resource constraint devices and protect users' security and privacy should be developed. Unlike traditional Internet, a one-size-fits-all solution is not applicable in the IoT ecosystem because the IoT domains and the use cases are different. Consequently, the requirements are different. Therefore, more specialized solutions need to be designed and developed.

The main goal of this dissertation is the design and development of new secure mutual authentication and key exchange schemes for secure communication in the IoT systems for both stationary and mobile IoT nodes. The proposed schemes are built on six main foundations: (i) implementation of mutual authentication; (ii), the use of Physically Unclonable Functions (PUFs) as the root of trust (iii) verification of hardware counterfeiting; (iv) use of lightweight cryptosystem; (v) assurance of data integrity, confidentiality and privacy preservation; and (vi) support system scalability.

1.4 Main Contributions

Designing lightweight authentication schemes for different IoT domains is necessary. Transmitting user's data in a secure environment is crucial to preserve the user's privacy and security. Our proposed schemes depend on modern lightweight cryptography, PUF hardware's intrinsic security primitive, and user biometric features.

Cryptography techniques are used to generate a shared key, establish an encrypted and secure channel of communication between the devices with the IoT systems, and authenticate the devices. PUF is used to create a secure and robust authentication scheme for devices and protect devices from counterfeiting along with the different cryptographic protocols. Furthermore, both static and dynamic user biometric features are used to complete initial and ongoing user authentication.

The main contributions are as follows of this work are as follows:

- The development of PUF Hierarchal Distributed Architecture (PHDA) for device name resolution to support system scalability and protect the IoT systems from being counterfeited.
- Implement the Root of Trust (ROT) scheme in the IoT environment through the utilization of PUF to ensure authenticity by using the challenge-response pair protocol. Also, PUF is used to derive device-dependent security keys.
- The development of a zero-knowledge crypto solution known as " Chained Hash PUF" to maintain node tracking. It is considered the first of its kind technique in the IoT authentication research field.

- The development of a Lightweight Mutual Authentication and Privacy Preservation Framework for IoT (LMAP^{2}A). The framework is developed to be domain-independent for generalized usage in IoT environments. LMAP^{2}A integrates both software and hardware-based security approaches. Also, LMAP^{2}A applies the "defense in depth" concept by using multiple layers of security countermeasures through an IoT system to provide redundancy in case a security countermeasure fails, or a vulnerability is successfully exploited.

- Based on the proposed framework, three lightweight authentication schemes were developed to support three different IoT systems. The three different schemes are designed to support different node mobility and distribution requirements.

- Detailed security and performance evaluation of the presented three schemes.

1.5 Dissertation Outlines

The dissertation is organized as follows:

- Chapter 2 provides a detailed overview of IoT security, authentication, and hardware security. Also, the chapter discusses related work on authentication and key agreement for IoT.

- Chapter 3 presents the foundation framework that is called “Lightweight Mutual Authentication and Privacy Preservation Architecture for IoT (LMAP^{2}A)”. This framework is considered the backbone and the road map of the schemes that are presented in the following three chapters. This chapter also presents a PUF Hierarchal Distributed Architecture (PHDA), which is used to perform the IoT device name resolution.

- Chapter 4: presents the first authentication scheme titled “Lightweight Privacy Preservation and Mutual Authentication Scheme for Smart Homes Using Physical Unclonable Functions.” All devices in the proposed smart home use case are stationary and within range.
- Chapter 5: presents the second authentication scheme titled “M2M Distributed Multi-Layer Lightweight Mutual Authentication and Key Agreement Scheme Using Chained Hash PUF in Industrial IoT System”. The chapter presents the concept of the chained hash PUF and how it is used to authenticate IoT devices. The poultry farm network model is multi-layer in nature, and the devices are distributed and heterogeneous.
- Chapter 6: describes the third authentication scheme called “Three-Factor Authentication and Privacy Preservation Scheme Using User and Device Biometrics for IoV System.” All devices and users in the IoV network model are mobile and require frequent authentication.
- Chapter 7: concludes the dissertation by discussing the contributions of the research and outlining future work.

CHAPTER II

BACKGROUND AND LITERATURE REVIEW

To better present the contribution of this dissertation, an in-depth analysis of IoT and the available solutions of IoT security are discussed. This chapter provides background information about the Internet of Things, integrity, confidentiality, authentication, privacy, and trust. Also, the use of a physical unclonable function (PUF) as a hardware fingerprint is studied. Moreover, the use of fog computing as support to IoT resource-constraint devices in data processing and information delivery is discussed.

2.1 Why IoT Security is Different

Internet of Things (IoT) is experiencing continuous growth. Not only classical computing and communication devices are connected, but also a whole range of other gadgets that are used in our daily life in different domains such as smart homes, health care, wearable devices, smart city, and others. Consequently, tens and even hundreds of billions of devices will be connected. In the IoT ecosystem, objects are connected via the Internet without any human intervention. IoT enables the transfer and sharing of data among living and nonliving objects to achieve specific goals. Those devices will have smart capabilities to collect, analyze, and even make decisions without any human interaction.

As described in [7], five main characteristics differentiate the IoT devices from any other device: (i) Interconnectivity that indicates that all IoT devices can be connected to the global information and communication infrastructure. IoT is based on the idea of being able to interconnect everything ;(ii) heterogeneity where IoT systems have a mix of various devices and networking protocols but can still interact with each other through different networks. This feature is considered one of the main challenges in the IoT ecosystem. ;(iii) dynamic changes where the devices can be in different states such as connected, disconnected, waking up, and sleeping. The devices may change their state dynamically which will, in turn, change the number of devices;(iv) limited resources where the IoT devices experience limited CPU capabilities, memory, and power resources; (v) large scale where the number of IoT devices grows exponentially and will be larger than the number of devices in the current Internet. Most of the communication will be device-to-device instead of human-to-device, and (vi) IoT devices are often deployed unattended in open and public places which may cause them to be vulnerable to physical and cloning attacks

The exponential growth of IoT is challenging from both security and technical perspectives due to the heterogeneous mix of devices and communication protocols. At least two challenges need to be carefully tackled: (1) security and privacy preservation requirements to ensure a safe IoT environment; (2) the heterogonous nature of IoT that makes one-size-fits-all security schemes unfeasible and inapplicable. When everything is connected, multiple security threats will arise and put confidentiality, integrity, availability, authenticity, privacy, and trust in risk.

2.2 Security Goals

Information Security refers to the processes and methodologies which are designed and implemented to protect data, information, and systems. Information security means protecting data, information, and systems from unauthorized access, use, disclosure, disruption, modification, or destruction. National Institute of Standards and Technology (NIST) and the IETF defined security metrics as a tool to facilitate decision-making and improve the systems' performance. Security metrics are used to measure the security level because what we cannot measure, we cannot improve. Security metrics can be used to identify the strength and weaknesses of the implemented security system. NIST and IETF agree on a set of security goals that are necessary to achieve information security. Those objectives are confidentiality, integrity, authenticity, availability, and accountability of all messages [17]. The authors in [18] added auditability, trustworthiness, non-repudiation, and privacy as additional security requirements.

- **Confidentiality**: the process of ensuring that data will be disclosed only to authorized users or systems. This applies to data in storage, during processing, and while in transit. Confidentiality is crucial for IoT devices because they might handle critical personal information. For example, unauthorized access to user's data may lead to life-threatening situations.

- **Integrity**: is the process of ensuring that the data has not to be altered or modified. The modification includes writing, changing the status, deleting, creating, delaying, and replaying the messages. Integrity is important to provide a

reliable service. If a modification has occurred, it must be detected. The compromisation of integrity can lead to serious consequences.

- **Availability**: A system should always be available to legitimate users and deliver prompt and reliable service. Availability is essential to ensure that devices are available for collecting data and prevents service interruptions.

- **Scalability**: The IoT system must be able to accept and register new IoT devices. The IoT scheme must be able to adapt to new technologies and incorporate the needed modifications [19].

- **Privacy**: It refers to the protection of sensitive data. Privacy implementation requires securing end-to-end communication. Therefore, the transmission process of the collected data should ensure that the data is not being tampered by an adversary. Furthermore, the collected data must be stored in a secure environment.

- **Authenticity**: It is the process of verifying the device's or user's identity. Through authentication, we can verify the origin of a message, the date of the message, and message content. There are two classes of authentication: entity authentication and data authentication. Data authentication implies data integrity.

- **Non-repudiation** is the process that prevents both the sender and the receiver from denying previous communication or actions.

2.3 Authentication

Authentication can be viewed as the first line of defense by ensuring the enforcement of security measures. Authentication requires a handshake process that can be done before an authorization is granted. Device authentication is the process of verifying the device identity. Without strong authentication, an attacker can capture sensitive data and execute malicious actions. Deploying robust authentication protocols becomes one of the first steps to ensure the security of the whole system.

Authentication needs to be established to build secure communication between the IoT devices before transferring any data. Traditional authentication techniques depend on using one of the well-known authentication factors to identify users, such as a secret that a user knows or a token that a user has. On the other hand, modern authentication techniques combine several authentication factors to authenticate users or devices in different ways, such as two factors authentication and multi-factor authentication that use two or more independent channels for authentication. Two or more factor authentication processes aim to create a layered defense system that combines two or more independent authentication factors to make it hard and even impossible to gain access to a target. The two-factor authentication techniques have been widely used in recent years to ensure secure authentication in systems well as providing an extra layer of protection and increase user trust in the system. The authentication process can take two different formats: machine-to-machine authentication and user-to-machine authentication.

2.4 Cryptographic Systems

One of the main techniques to achieve security goals is the implementation of cryptographic algorithms. There are two main cryptographic algorithms: (1) symmetric cryptography; (2) asymmetric cryptography.

2.4.1 Symmetric Cryptography

In symmetric cryptography, both the sender and receiver share the same secret key (K) to encrypt and decrypt data [20]. When the shared secret key encrypts the message, it needs to be decrypted by the same key. A symmetric key cipher f is used to encrypt a plaintext message m and generates a ciphertext $c = f_K(m)$ to be sent. At the receiver side, an inverse cipher f^{-1} is used to retrieve the plaintext $m = f^{-1} K(c)$, as illustrated in figure 3.

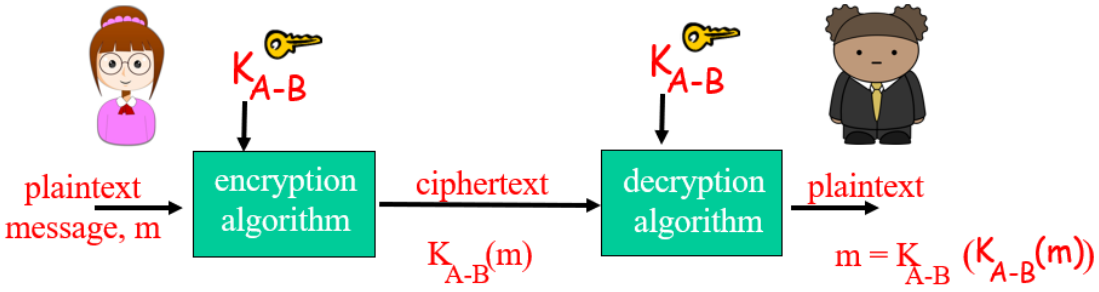


Figure 3: Symmetric Encryption Process

One of the fast and secure symmetric cryptography algorithms is the Advanced Encryption Standard (AES). AES is a famous cipher that NIST has standardized to replace DES and Triple DES [20]. AES supports key lengths of 128, 192, or 256 bits, and the block length is 128 bits. AES does not require much memory, making it suitable for resource constraint IoT devices [21].

One of the main properties of symmetric cryptography that can be viewed as a drawback is the need to establish a secure channel of communication between the sender

and the receiver before exchanging the shared secret key between the two sides of the communication. This is known as the key distribution problem [20]. The storage of the shared secret key can introduce vulnerabilities to the secure system. A compromised device will reveal all stored secret keys, and consequently, all communication on this device would be accessible.

Although the problems mentioned above are universal of symmetric cryptography, it is still the best tool for encryption. Several techniques can be implemented to overcome the shortcomings. One of the techniques is called Diffie Hellman key exchange techniques. This technique will allow parties to establish a shared secret key over an insecure channel of communication. Diffie Hellman can be used in conjunction with asymmetric cryptography.

2.4.2 Asymmetric Cryptography

Unlike symmetric cryptography, asymmetric cryptography, or what is also called public-key cryptography (PCKS), allows users to communicate securely without the need for a shared secret key. PCKS is considered more computationally expensive compared to symmetric key cryptography. Both the sender and the receiver each have two keys called the public key and the private key. The private keys are kept secret, while the public keys are public and widely distributed. The receiver's public key is used for encryption by the sender, and the receiver's private key is used for decryption by the

receiver, as illustrated in figure 4. When a sender encrypts a message using the receiver's public key, only the receiver can decrypt the message using his private key.

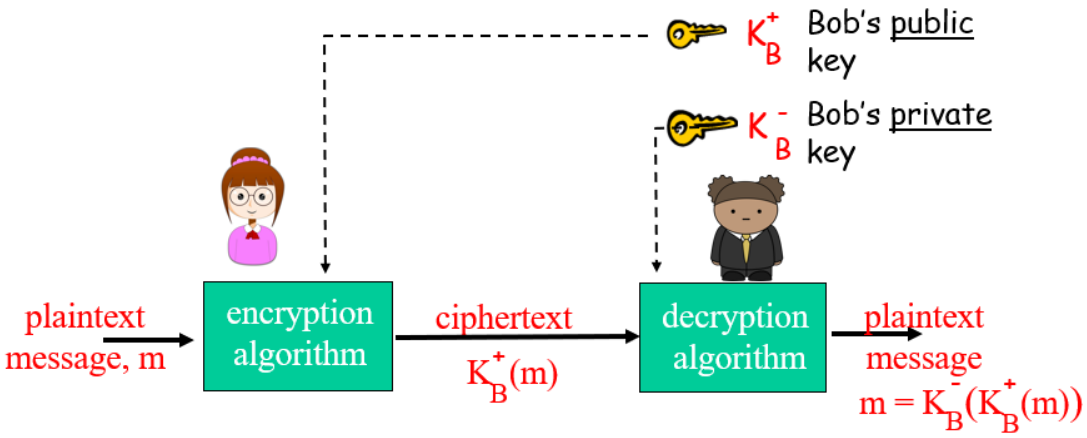


Figure 4: The Process of Asymmetric Encryption

Besides encryption, PKCS can be used for key establishment and authentication and non-repudiation [20]. Key establishment features in PKCS can be used to overcome the key establishment issue in symmetric cryptography, where the two parties need to communicate through a secure channel to exchange the shared secret key. By using PKCS, the two sides can securely establish a shared key. Non-repudiation is the technique that prevents any party from denying its actions. Non-repudiation and authentication can be achieved through the use of a digital signature that employs PKCS. The sender encrypts a message with his private key, and the receiver decrypts the message using the sender's public key; the receiver is assured that the sender generated the message, provided the public key is verified and trusted or extremely signed by a trusted certifying authority.

Because PKCS is more computationally expensive than symmetric cryptography, it cannot be used for encrypting the ongoing communication during the session duration. Instead, symmetric cryptography is used. PKC can be used for the key establishment

stage, and then symmetric cryptography can be used to encrypt and decrypt all the ongoing communication during the session. The key establishment can be divided into a key transport protocol and a key agreement protocol [22]. One party creates the shared secret key in the key transport mechanism and securely transfers it to the other party using the receiver's public key to encrypt the shared secret key. An example of a key transport mechanism is RSA.

On the other side, the shared secret key derivation process in the key agreement mechanism is completed through the contribution of the two parties. This process causes the two sides to influence the generated key and have the same key. An example of a key agreement protocol is Diffie Hellman key exchange

2.4.2.1 Diffie Hellman Key Exchange

The Diffie-Hellman Key Exchange (DHKE) was the first PKC protocol that is based on the Discrete Log Problem (DLP) [20]. The definition of DLP, as given in [20], is presented in figure 5. When we select large numbers, computing the discrete logarithm to identify the x value becomes a very time-consuming and challenging task [20].

Given in the finite cyclic group Z_p where p is the prime number with a finite set of integers $i = 0, 1, \ldots, p-1$ and a primitive element $g \in Z_p$ and another element $h \in Z_p$. The DLP is the problem of determining the integer $1 \leq x \leq p-1$ such that:

$$g^x \equiv h \bmod_p$$

Figure 5: Discrete Log Problem (DLP)

DHKE consists of two main phases: (a) an initialization phase and (b) a key-agreement phase. During the initialization phase, the two parties need to agree upon a set

of parameters called domain parameters, as presented in figure 6. These domain parameters are required to generate the secret key. By using the domain parameters, the two sides can generate the same secret key over an insecure channel using the key agreement phase.

Diffie Hellman Initialization Phase

1. Select a large prime number P
2. select a positive integer g, such that g is a generator modulo p
3. publish both P and g

Figure 6: Initialization Phase Diffie-Hellman protocol

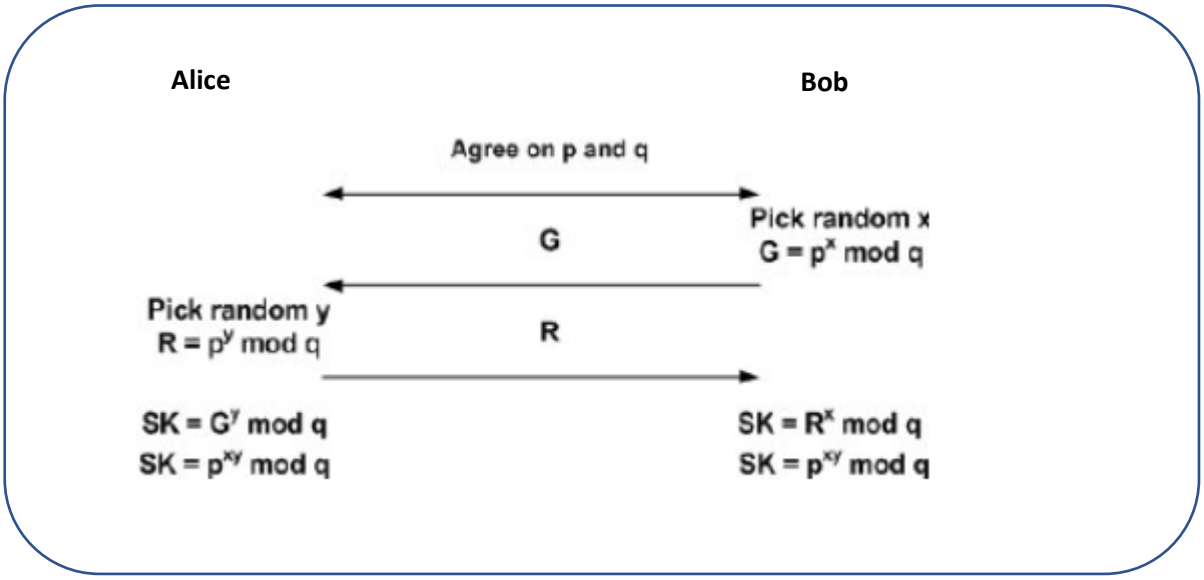


Figure 7: Key Agreement Process Diffie-Hellman protocol

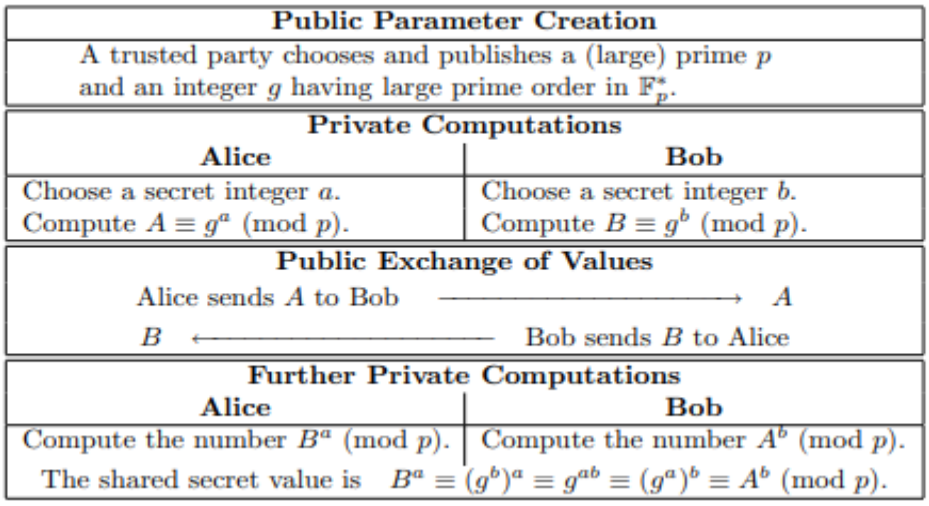


Figure 8: Key Agreement Phase Diffie-Hellman protocol

During the key-agreement phase, both participants contribute to the key establishment. Figures 7 and 8 show the key agreement process and phases of the Diffie Hellman protocol. During the first step of the key agreement phase, Alice and Bob agree on a large prime p and a nonzero integer g modulo p. Alice and Bob make the values of p and g public knowledge. The next step is for Alice to pick a secret integer y that she does not reveal to anyone, and Bob picks an integer x that he keeps secret. Bob and Alice use their secret integers to compute their public keys. As a next step, both Alice and Bob exchange their public keys. Alice sends R to Bob, and Bob sends G to Alice. Finally, Bob and Alice use their secret integers to compute the secret key.

2.4.2.2 Elliptic Curve Cryptography (ECC)

Elliptic Curve Cryptography (ECC) is a public key algorithm that was introduced in 1987 by [23]and by [24]. ECC is considered lightweight compared to RSA because it can achieve the same level of security with much fewer arithmetic operations [20]. For example, 1024-bit RSA corresponds to 160-bit in ECC. Therefore, ECC can be used in

resource-constraint devices. The high security of ECC is based on the computation of the ECC discrete logarithm problem (ECDLP). Solving this problem cannot be done effortlessly [22]. Also, the Elliptic Curve Diffie-Hellman (ECDH) protocol can be used to generate the secret key in a secure environment.

ECC is based on the ECDLP, which is based on the DLP. This makes DHKE suitable for ECC. As described in [20], the elliptic curve over Z_p, $p > 3$, is the set of all pairs $(x, y) \in Z_p$ which fulfill:

$$y^2 \equiv x^3 + ax + b \bmod p$$

together with an imaginary point of infinity θ, where $a, b \in Z_p$ and the condition $4a^3 + 27b^2 \neq 0 \bmod p$. ECDLP is defined by [19] as described in figure 9.

Given is an elliptic curve E. We consider a primitive element P and another element T. The ECDLP problem is finding the integer d, where $1 \leq d \leq E$, such that

$$\underbrace{P + P + \ldots + P}_{d \text{ times}} = dP = T$$

where E is the number of points on the curve

Figure 9: Elliptic Curve Discrete Logarithm Problem (ECDLP)

ECDH key agreement protocol is based on the ECDLP. ECDH allows two parties to establish a shared secret key over an insecure channel, where each of the parties has an elliptic curve public-private key pair [19]. In the beginning, the two parties agree on the ECC domain parameters that are presented in figure 10.

1. p: Field that the curve defined over
2. G: the generator or the primitive root
3. a, b: values define the curve
4. n: prime order of G
5. h: cofactor

Figure 10: Elliptic Curve Diffie Hellman Domain Parameters

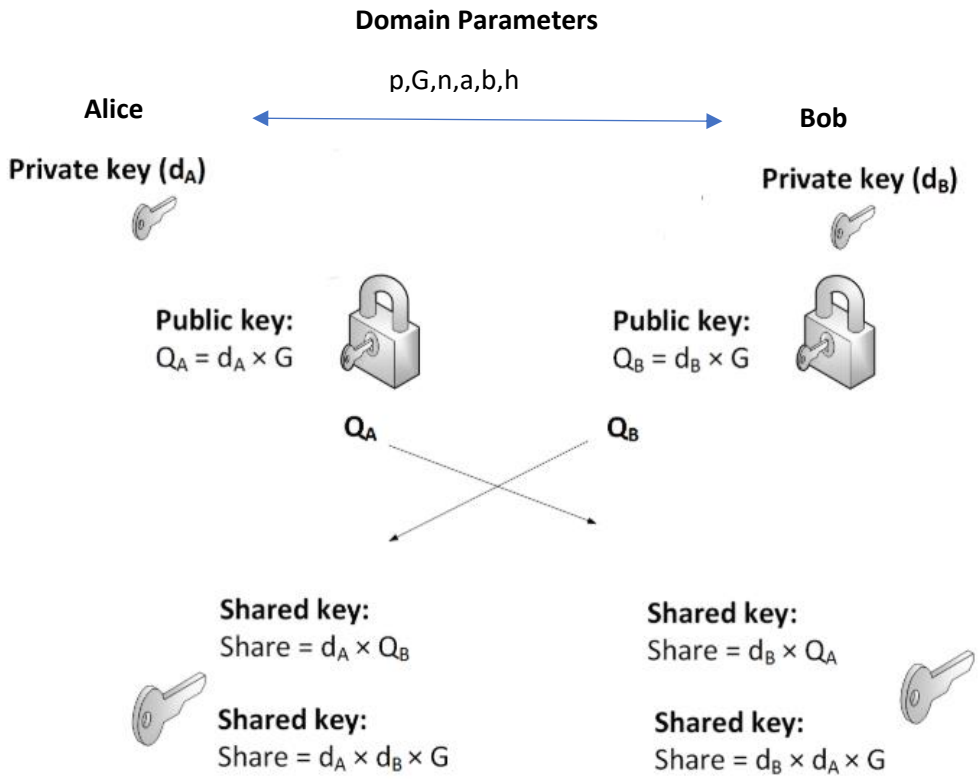


Figure 11: Elliptic Curve Diffie Hellman Key Agreement Protocol

Figure 11 describes the ECDH protocol. First, both Alice and Bob select a private key that is a random integer d between {1,……n-1}. Second, each side will calculate its public keys, which are QA for Alice and QB for Bob. Third, Alice and Bob exchange their public keys over an insecure channel of communication. Last, each side calculates

the secret key using its private key and the other side's public key. Once the two sides have the same secret key, they can use symmetric encryption for all subsequent ongoing communication.

Because one of the key features of the IoT ecosystem is handling resource-constrained devices, ECC is preferred over RSA for the following reasons: (1) It uses a shorter encryption key which in turn uses less memory, less storage, less power, and fewer CPU resources; (2) ECC is based on the computation of the discrete logarithm problem. Therefore, it is highly secured; (3) Zigbee Networking and Wi-Fi WPA 3 Standards specify ECDSA and ECDH as the algorithm of choice, and (4) The US government publishes a set of standards algorithms approved for use in non-defense applications called Suite B Cryptography. Currently, this standard includes only ECC for authentication and key management. RSA has been completely removed.

Table 1 presents a comparison of RSA and ECC algorithms for the key size. Column 1 shows the security level of that particular row. The security level is a measure of the strength that a cryptographic algorithm achieves. The security level is usually expressed in bits, where n-bit security means that the attacker would have to perform 2^n operations to break the cryptographic cipher. Column 2 shows the public key size of RSA to achieve the corresponding security level. Column 3 shows the public key size of ECC to accomplish the corresponding security level. Column 4 displays the key size ratio between RSA ECC.

Table 1: RSA Versus ECC Key size

	Minimum Size of Public Keys		Key Size Ratio
Security Level in bits	**RSA**	**ECC**	**ECC to RSA**
80	1024	160-223	1:6
112	2048	224-255	1:9
128	3072	256-383	1:12
192	7680	384-511	1:20
256	15360	512+	1:30

The current common security requirement is to achieve a 128-bit security level. As stated in table 1, an ECC key size of 256 can accomplish a 128-bit security level compared to the RSA key size of 3072. This indicated that the ECC key size is 12 times smaller than the RSA key size. This means that RSA is more computationally expensive than ECC, and it requires more storage, more power, and more CPU Resources. Therefore, ECC is considered the potential algorithm for resource-constrained IoT devices. Because the main focus of this dissertation is IoT constrained devices, we will adopt ECC as the chosen Cryptographic algorithm.

2.4.3 Message Authentication Code

A message authentication code (MAC) is a block of a few bytes that is used to authenticate a message. The receiver can check this block and ensure that the message is coming from the right sender and that an adversary hasn't modified it. A specific type of MAC is called HMAC. HMAC stands for Hash-based MAC. HMAC is a special type of message authentication code involving a cryptographic hash function and a secret cryptographic key [25]. This scheme extends the error detection capability in verifying data integrity as well as message authentication. HMAC can work with any cryptographic hash function such as MD5, SHA-1, SHA-256, or SHA-512, combined with a shared secret key [26]. Once the HMAC hash is calculated, the message must be sent alongside the HMAC hash. In HMAC, the presence of a shared secret helps to establish authenticity because it is generated during a key exchange process that requires the participation of the two communication parties. Only those two parties know what the secret key is. Then, they can verify that the message is from a legitimate source when the hash matches.

Figure 12 describes how HMAC is working. First, the sender and the receiver share a secret key. Second, the sender creates the message and calculates the HMAC hash for the message where the message and the key will be used as inputs to the algorithm. Third, the sender will send the message alongside the HMAC hash. Fourth, the receiver will calculate the HMAC hash. Fifth, the receiver will verify the message's integrity and authenticity by comparing the received HMAC hash with the computed HMAC hash. If the two values are the same, the receiver ensures that the message has not been corrupted or modified.

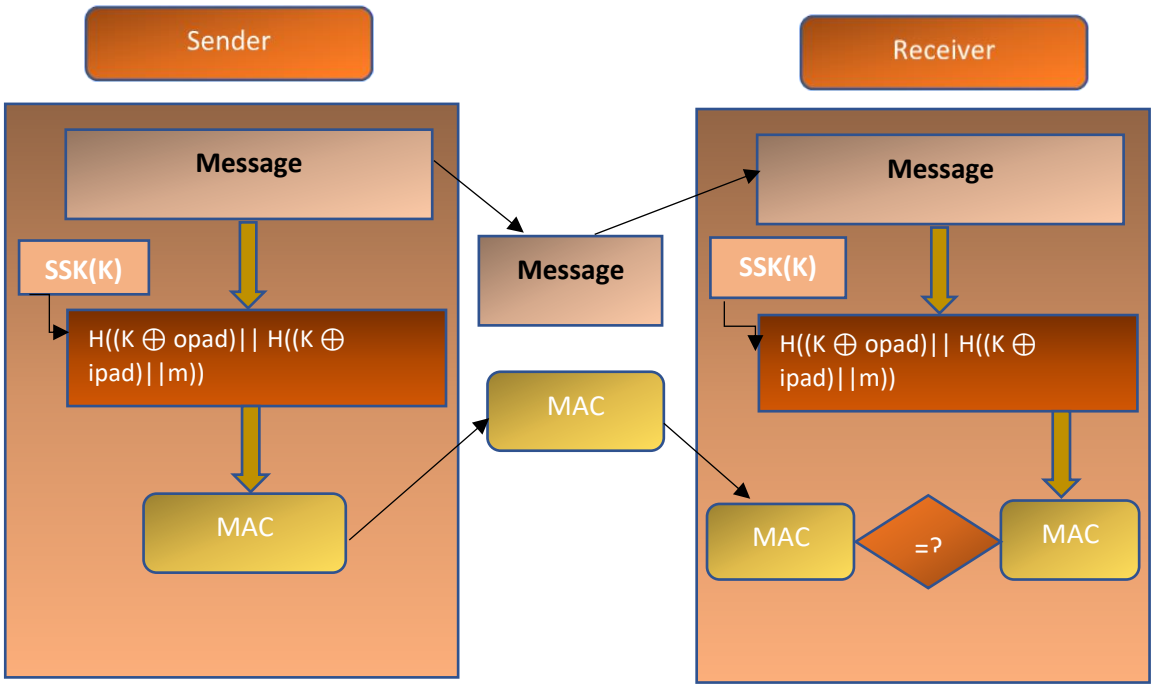


Figure 12 Hash-based MAC

2.4.4 Physical Unclonable Function (PUF) Technology

2.4.4.1 Concept of PUFs

With the quadratic growth of IoT devices, security and privacy become essential requirements [27]. Most of the currently available security solutions depend on the implementation of cryptography. The assumption is that the devices can securely store the secret key and keep it away from an adversary. According to [28] and [29], this assumption is not practical because physical attacks such as invasive, semi-invasive, or side cha attacks can lead to key exposure and security breaches. Also, [29] highlighted that some devices could be resource-constrained or do not have a nonvolatile memory to store the secret key.

An alternative solution is a hardware security primitive known as Physical Unclonable Function (PUF). The PUF concept was first introduced by [30] in 2000. The authors proposed the idea of the mismatch in silicon devices for integrated circuit (IC) identification. The authors in [31] introduced Physical On-Way Functions. Then, [32] proposed a Silicon Physical Random Functions and presented it as a PUF. PUF is new hardware-based security primitive. A PUF is a function embedded in a physical device to extract a secret from a complex physical system. The key idea behind PUF is the exploitation of the "random physical disorder" or the "manufacturing variation" of the silicon chips [33]. This manufacturing variation can not be controlled during the fabrication process, and at the same time, it can not be refabricated intentionally. PUF can be described as a function that returns a value called the response. This value can be treated as a unique signature or a unique fingerprint of an integrated circuit for a given challenge. Hence, we can define PUF as a Physical challenge-response procedure to extract the signature of an integrated circuit. A PUF is similar to human fingerprints because it produces a specific device signature to authenticate the device. PUF can be used for device authentication and to protect from devices counterfeiting [34]. Also, the unique secret key generated by the PUF can be used in different domains [35].

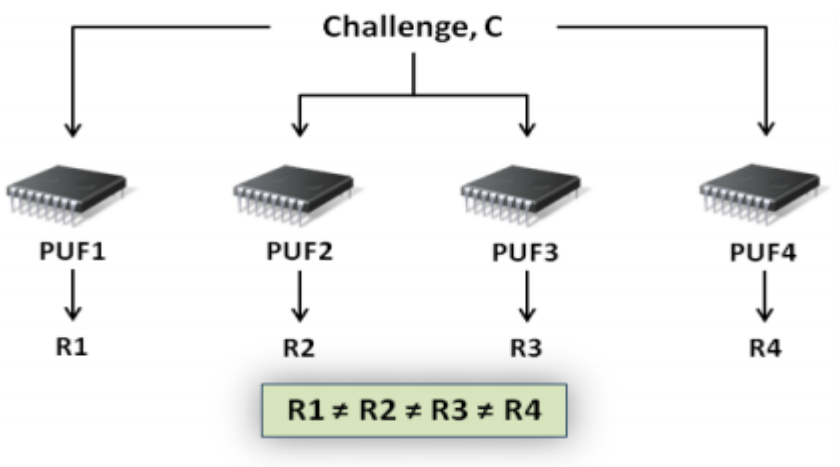


Figure 13: PUF Challenge-Response Uniqueness

2.4.4.2 Properties and Parameters of PUFs

Based on the PUF definition, we can extract two main properties of a PUF:(1) It is **unpredictable**. A PUF is unpredictable because an attacker who can use a limited and fixed amount of resources can only extract a small amount of information from the PUF's secret response. (2) It is **unclonable**. This means that it is hard to produce two identical PUFs from two different IC as described in figures 13 and 14 . PUFs take advantage of the circuit characteristics related to the uncontrollable random variation in the manufacturing process. Therefore, the less control present during the circuit's manufacturing process, the harder the reproduction of an identical PUF. The author in [36] added four additional properties which have been identified from multiple proposed PUF definitions. These properties are:

1. **Low cost**. This means that the measurement circuit should be easy to implement and low cost. From a theoretical point of view, the PUF response should be easy to evaluate/produce.

2. **Unique.** The PUF response is extracted from the unique identity of the physical entity. The set of challenge-response pairs should be sufficient to identify a PUF among a given population uniquely.

3. **Reproducible**. This property distinguishes PUFs from True Random Number Generators (TRNGs). The PUF response should be reproducible when introducing the same challenge, even when it is under different environmental conditions.

4. **Secure**. When an invasive physical attack is performed, PUFs should produce an error response when asked.

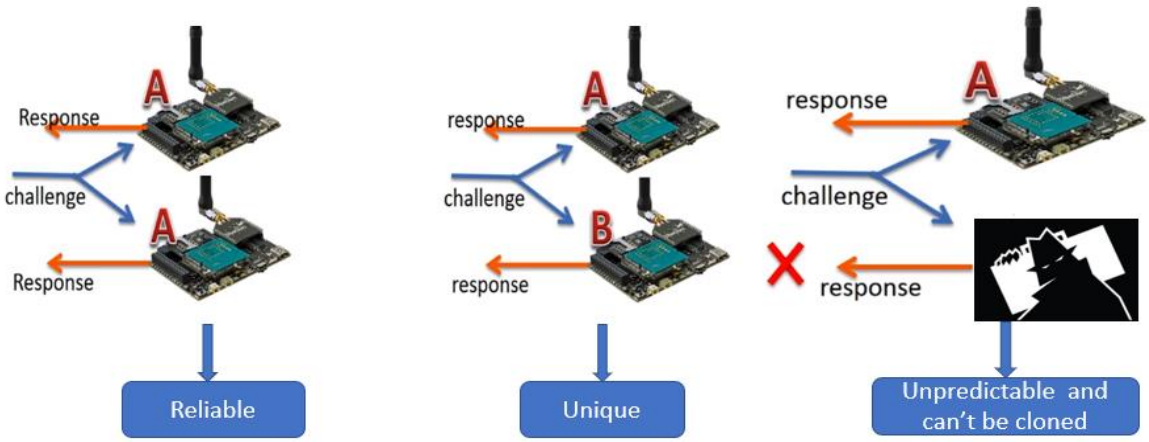


Figure 14: Main properties of PUF

The authors in [37] proposed the construction of PUF-FSM as a controlled strong PUF without the need for error correction codes and helper data by only using error-free responses, which are fed in a finite state machine. This type of PUF is implemented in our proposed schemes.

2.4.4.3 PUF Applications

There are four main applications that can use PUF.

2.4.4.3.1 Low-cost Device Authentication

The researcher in [38] proposed a low-cost device authentication based on a challenge-response protocol. The proposed protocol can also be applied to IoT resource-constrained devices. The authentication process ensures that an adversary cannot obtain the PUF output that is used for authentication. PUFs are expected to have exponential numbers of challenge-response pairs. This implies that the challenge can only be used for only one time, which can overcome the man-in-the-middle attack.

2.4.4.3.2 Cryptographic Key Generation

The authors in [39] presented a PUF based process to generate a reliable cryptographic key. The cryptographic key generation process is divided into two stages, which are initialization and regeneration of the PUF response. Figure 15 described the cryptographic key generation process using PUFs. The generated c key can be used with different cryptographic algorithms such as Advanced Encryption Standard (AES), Rivest Shamir and Adleman (RSA) or Elliptic Curve Cryptography (ECC) and others

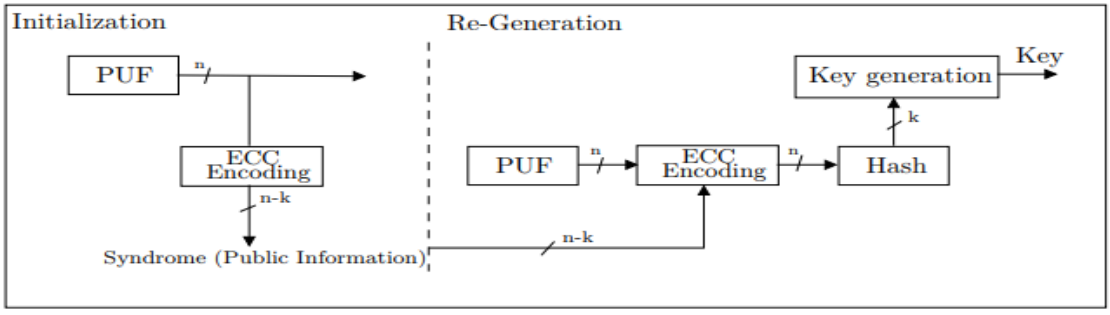


Figure 15: Cryptographic key generation using PUFs

2.4.4.3.3 Intellectual Property (IP) Protection

Counterfeit and Intellectual Property (IP) theft are two related key problems in the IoT ecosystem. Most of the counterfeit products are made from stolen Intellectual Property. While counterfeiting is the process of creating cloned fake copies of IoT devices, Intellectual Property is the process of unauthorized use of software in a fake device in violation of the law. Because PUF responses are unpredictable and unclonable, the PUF can be used as a hardware fingerprint to protect the IoT devices against counterfeiting.

2.4.4.4 PUF Classification

There are two PUF classifications from the security perspective. The two classes are strong PUF and weak PUF. The distinction between the two types of PUF depends on the number of challenge-response pairs. A PUF is called strong when its challenge is large, such as the arbiter PUF. On the other hand, Weak PUFs structures have one challenge, such as SRAM PUFs. According to [33], a weak PUF has two main features: (1) few challenges: A Weak PUF has only a few and fixed challenges; (2) Access-restricted responses: the challenge-response interface needs to be restricted. In weak PUF, it is assumed that the attacker cannot access PUF's responses. The authors of [40] [41] defined the main features of strong PUF. The main features of strong PUF are (1) Many challenges: Strong PUF has a large number of possible CRPs; (2) Unpredictability: even if an adversary knows a large number of CRPs, he cannot predict unknown CRPs.

2.4.4.4.1 Delay based PUFs – Arbiter PUF

The arbiter PUF is a delay-based silicon PUF. The arbiter PUF structure comprises two parallel, identical, and controllable delay paths and an arbiter circuit at the end. It is composed of a sequence of switch components. The simplest way to implement them is with a pair of 2-to-1 multiplexers. Each one interconnects two input ports to two output ports with different configurations depending on the applied control bit (0 or 1). A delay-based PUF exploits the random variations in delays of wires and gates on silicon. Ideally, the delay between the red and blue lines should be 0 if they are symmetrically laid out. In practice, the variation in the manufacturing process will introduce a random delay between the two paths.

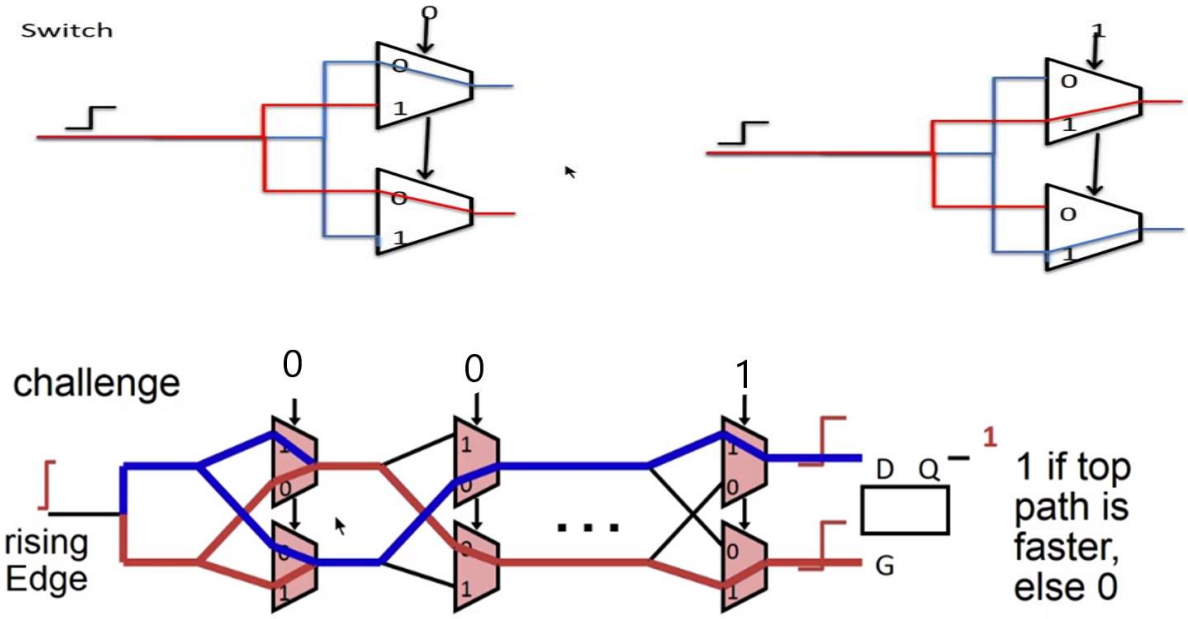


Figure 16: PUF Design

Figure 16 shows how the arbiter PUF works. The idea is to introduce an edge and then make a race between the two paths and sample the top signal and the bottom one at the end. Then, the arbiter circuit outputs a binary value to indicate which one of the two

paths is faster or slower. Because the two parallel paths are identical, the delay difference between them is minor. Hence, two scenarios are possible to get the arbiter circuit decision:

1. Even when designed identically, the delays of the two paths may not be equal. This is due to the random silicon process variation. This random difference between the high and the low path will determine the output of the arbiter circuit and then the PUF response. And, because the random silicon process variation is device specific, the arbiter output will be device specific.

2. It is also possible that even with the random process variation between the two designed paths, the delay difference cannot be detectable by the arbiter circuit. Therefore, the introduced edge will simultaneously reach the two inputs, so the arbiter circuit produces a metastable state. Then, after a short random time, the arbiter will output a value that is independent of the paths' race.

The authors in [37] proposed the construction of PUF-FSM as a controlled strong PUF without the need for error correction codes and helper data by only using error-free responses, which are fed in a finite state machine. This type of arbiter PUF is implemented in my proposed architecture.

2.5 Fog Computing

Cloud computing enables IoT devices to provide data storage, cost-effective, and on-demand services. However, [42] and [43] stated that the cloud-based IoT services have some issues such as latency, lack of mobility support and security, and location

awareness. For example, sensor and actuators systems, and monitoring systems are too latency-sensitive to be deployed on the cloud. Also, IoT devices produce a large amount of confidential and sensitive data. At the same time, most of the IoT devices are resource-constrained. One way to overcome the latency issue and to offload the security operations from the IoT device is the use of a more resourceful entity such as a fog-based node. Fog computing technology, also known as fog networking or fogging, was first introduced by Cisco in 2014 as an extension of cloud computing that would work on the edge of the end-users network [43] [44]. Fog computing puts most of the communication, storage, and management at the edge of a network rather than on the centralized cloud, which in turn will improve the service and reduce the latency that will be interpreted as better service to the user.

There is no standard architecture for fog computing [45]. According to [46], fog computing can be divided into cloud-fog-edge-device architecture and fog-device architecture, as shown in Figure 17.

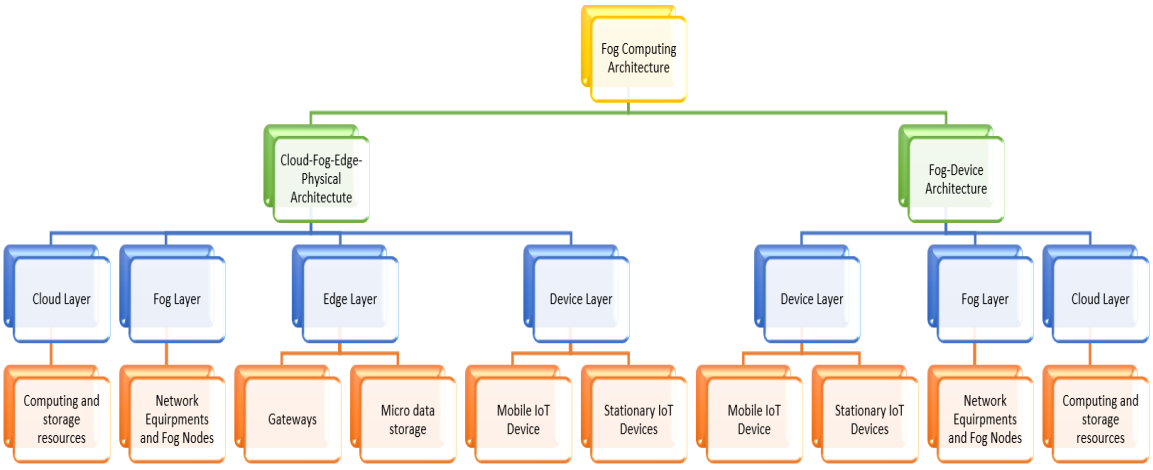


Figure 17: Fog computing architecture

2.5.1 Cloud-Fog- Edge-Device Architecture

As presented in figure 17, the Cloud-Fog-Edge-Device architecture has four layers, which are the cloud layer, fog layer, edge layer, and device layer. By moving down from the cloud layer to the other three layers, the storage and computing capabilities decrease, and at the same time, the latency will decrease too. The cloud layer consists of computing and permanent storage devices. The fog layer provides data processing and temporary data storage near the IoT device. The use of the fog nodes can support the mobility and the heterogeneity of the IoT device as well as reduce the latency and provide service to a larger number of devices [47]. The edge layer provides local gateways that can be used to process data locally instead of sending it to the fog or the cloud. The device layer consists of both mobile and stationary IoT devices. These devices can be anything equipped with different capacities of storage, communication, processing, and computational capabilities.

2.5.2 Cloud-Fog-Device Architecture

In cloud-fog-device architecture, the cloud layer consists of computing and permanent storage devices. Fog nodes provide different services to the IoT devices without the intervention of the cloud layer. The implementation of this architecture depends on the IoT application. This, in turn, depends on the needs and requirements of IoT applications.

In summary, the main goal of fog computing is to improve the quality of provided service to users and reduce the traffic to the cloud. Data processing occurs in a gateway to minimize the amount of the transmitted data to the cloud. According to [48], fog computing performs short-term analytics at the edge while the cloud performs long-term analytics.

2.6 Biometric Authentication

Biometric features have become a key tool to authenticate mobile IoT devices. Biometric identification allows the user to use physical features instead of human made features to be authenticated and access systems. According to a survey by Javelin Strategy & Research, in 2014, $16 billion was stolen by 12.7 million people who were victims of identity theft in the US only [49]. The verification and authentication are performed based on three different techniques: (1) something we know, such as a password; (2) something we have such as tokens; or (3) something we are such as physiological and behavioral characteristics. The authentication process may employ one or more of the techniques mentioned above. There are two types of biometric identification, physiological and behavioral [50]. Physiological identification is based on

direct human body features such as the face, fingerprint, or iris. Behavioral features are based on indirect human features that are measured through feature extraction and machine learning. Examples of behavioral biometrics are signature, keystroke dynamics, gait, and voice [51].

The authors of [52] presented the benefits of using biometric authentication in the IoT applications as follows:

1. **Accurate information**: the biometric features are more precise and secured compared to password-based or token-based security. The use of biometric features does not require the user to remember any security credentials. Consequently, no adversary can steal or guess the security credential such as a password or a token. Therefore, biometric authentication can be a long-term security solution.
2. **Accountability**: A user who is using a service and access it by using his biometric features cannot deny using the service.
3. **Time-saving**: Biometric authentication is fast to execute compared to traditional security techniques.
4. **User-friendly systems**: it is easy for users to install biometric systems into their devices. By 2025 there will be 1200 million people aged 60, and above and by 2050 they will reach 2000 million. Therefore, the use of biometric authentication will be more suitable and user-friendly for those who will use IoT devices.

5. **Convenience**: It is a convenient security technique because users do not need to remember a password or keep any identity cards for verification.

According to [53], authentication mechanisms can be divided into static and dynamic authentication methods. Static authentication techniques authenticate users but do not monitor post-authentication sessions to detect if it is the same user accessing the system or not [54]. Dynamic authentication techniques monitor a system during the lifetime of a session to detect if it is the same user accessing the system or not [55]. The third proposed architecture will employ both static and dynamic biometric authentication to identify and verify users.

2.7 Summary

This chapter presents the basis for this work, and it influences every aspect of this research. The chapter discusses why IoT security is different and presents the main security goals. The chapter illustrates the different cryptographic techniques. Also, the chapter introduces the PUF concept, including its properties, applications, and classification. Furthermore, the chapter discusses the fog computing concept and its different architecture. Finally, the chapter reviews biometric authentication and its main advantages. In the subsequent chapters, it is assumed that readers have the necessary background knowledge.

CHAPTER III

A LIGHTWEIGHT MUTUAL AUTHENTICATION AND PRIVACY-PRESERVATION FRAMEWORK FOR IOT SYSTEMS

This chapter introduces an approach to authenticate IoT devices. This chapter presents how the proposed approach addresses the IoT security and privacy preservation challenges through a Lightweight Mutual Authentication and Privacy Preservation Framework for IoT ($LMAP^2A$). This chapter also presents a PUF Hierarchal Distributed Architecture (PHDA), which can be used to perform the IoT device name resolution. Additionally, this chapter describes how the $LMAP^2A$ framework achieves both the security and privacy goals. Finally, this chapter discusses the evaluation process to assess the security and performance of the schemes that will be designed based on the proposed framework.

3.1 Security Goals to Build an Effective Authentication scheme

To build a secure authentication schemes and evaluate other existing ones, a set of security goals must be defined and used as metrics to measure the authentication architecture's robustness. The following metrics should be taken into account when developing the IoT:

1. **Mutual authentication**: This means that the two communicating parties should authenticate each other. We must not assume a point of trust.

2. **Multiple authentication levels**: Multiple levels of authentication are required, each with different credentials. This feature is vital to implement the defense-in-depth concept [56]. Defense-in-depth requires using multiple layers of security countermeasures through an IoT system to provide redundancy if a security countermeasure fails or a vulnerability is successfully exploited.
3. **Protection against well-known attacks**: Security countermeasures should be in place to mitigate against well-known attacks such as man-in-the-middle attacks, replay attacks, eavesdropping, and brute force attacks.
4. **Scalability**: The architecture must be able to accommodate new devices and adapt and incorporate new technologies. According to [19], IoT architecture should be able to expand incrementally to support newly added devices or growing data volumes.
5. **Anonymity**: Anonymity is an essential feature to protect both a user's and device's privacy. The IoT devices can communicate anonymously with either fog or the cloud nodes without exchanging their real identities. It is essential to guarantee that the IoT devices' IDs and messages can only be traced by a trusted fog node or the Cloud. Anonymity ensures the IoT device's untraceability and unlinkability. The IoT device uses a pseudonym when transmitting data. Consequently, the adversary cannot trace back the device's identity from the pseudonym.
6. **Counterfeiting resistance**: The IoT devices should be secured from being cloned and replaced with fake devices.

3.2 PUF Hierarchal Distributed Architecture (PHDA)

We present a PUF architecture that can support system scalability, protect the IoT systems from the counterfeited devices and protect devices' privacy. To ensure system

scalability and flexibility, we propose PHDA as a device name resolution to store and retrieve the devices' Challenge-Response Pairs (CRPs). The proposed architecture is presented in figure 18. It stores the PUF data using a 3-tier architecture employing a simple naming scheme. The first node at the top of the hierarchy is the centralized Grand hub that maintains a reference to whom should be queried (Parent PUF Nodes) to obtain a reference to who may know how to retrieve the PUF data for a certain IoT device in a specific domain. The Parent PUF nodes maintain a reference to the service provider container (Child PUF Node) that should be queried next to return the PUF CRPs for a specific IoT device. The parent PUF nodes are organized based on different specific categories of the IoT devices (e.g., medical equipment, home equipment, etc.) The Child PUF nodes are organized based on different service providers. The Cloud or the fog should communicate initially with the Grand PUF servers to retrieve the CRPs of the IoT node. Moreover, the proposed architecture supports Over the Air (OTA) authentication and assist in overcoming the counterfeiting problem.

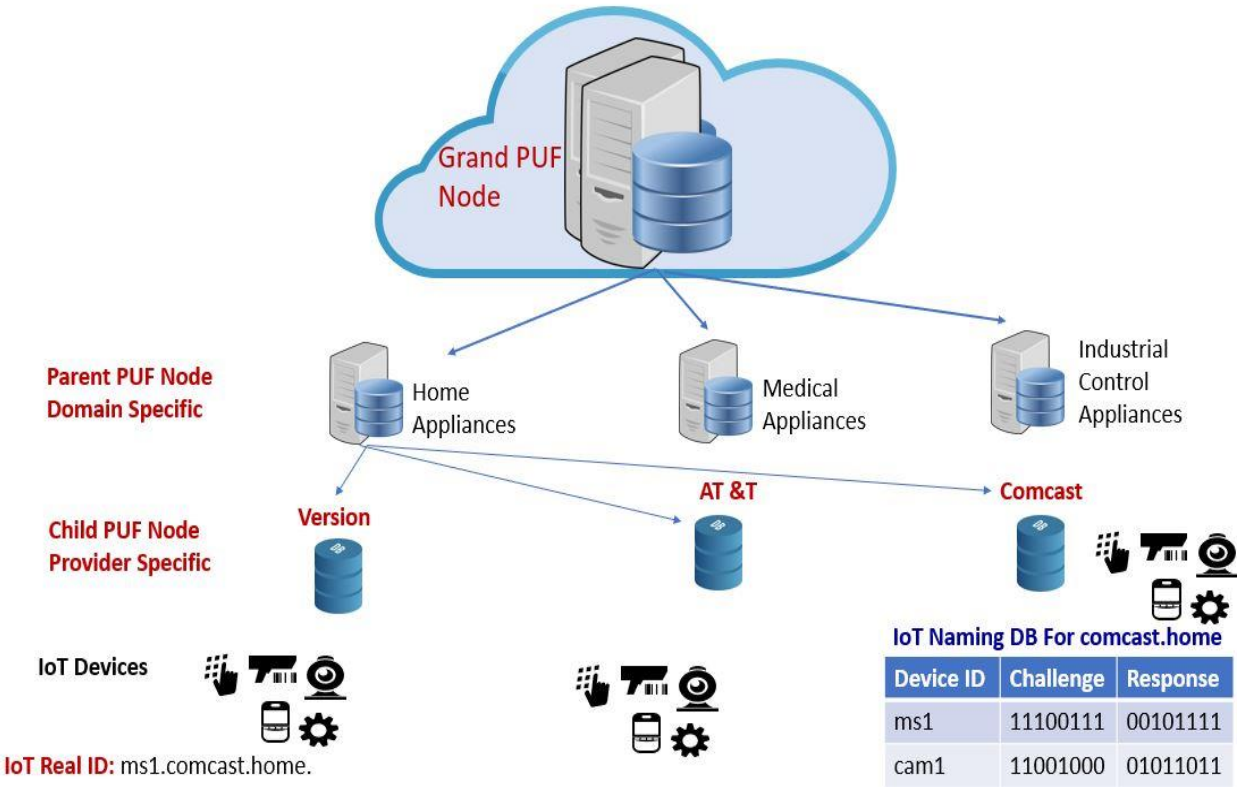


Figure 18: PUF Hierarchal Distributed Architecture

3.3 Lightweight Mutual Authentication and Privacy Preservation Framework for IoT (LMAP^{2}A)

With the exponential growth of the IoT field, IoT devices' lightweight nature makes security a significant concern. It highlights the need for designing schemes that can fit the resource-constrained nature of the IoT devices and satisfy new security and privacy requirements. When an IoT device and controller node need to interact securely, they need to authenticate each other. In some cases, both the IoT node and the controller node may need the cloud service's involvement to facilitate the authentication process. Furthermore, to support the mobility of the IoT device, there is a need to allow the

controller nodes under the same administrative authority to interact with each other and cooperate to verify the authenticity of the mobile IoT devices.

This work aims to present an frameework that can establish a secure connection between a resource-constrained device and a resource-rich controller node. We propose a secure, lightweight mutual authentication, key exchange, and privacy preservation framework for the IoT systems. The proposed framework integrates software and hardware-based security approaches that satisfy the NIST IoT security requirements for data protection and device identification [16]. We call this framework A Lightweight Mutual Authentication and Privacy Preservation Architecture for IoT ($LMAP^2A$). The proposed framework can authenticate both stationary and mobile IoT devices and authenticate both local and distributed controller nodes. The controller nodes build and manage trust relationships with other controller nodes. We assume that the granularity and requirements of the IoT devices may vary depending on the IoT domain.

$LMAP^2A$ is developed to be domain independent for generalized usage in IoT environments. Based on the proposed framework, three lightweight authentication schemes are developed to support three different IoT systems. The three different schemes are designed to support different node mobility and distribution requirements.

A use case scenario is designed to demonstrate each scheme. The first use case is the smart home domain, where the IoT devices are stationary and the controller node is local. In this use case, there is direct communication between the two parties. Establishing mutual authentication does not require the cloud service's involvement to reduce the system latency and offload the cloud traffic as the authentication data is stored locally on the controller unit in a secure database. The second use case is smart poultry

farms, an example of the Industrial IoT (IIoT) domain. In the second use case, the IoT devices are stationary, and the controller nodes are hierarchical and distributed, supported by machine-to-machine (M2M) communication. The third use case is the internet of vehicles (IoV) fleet vehicles as part of the smart city domain. During the roaming service, the mutual authentication process between a vehicle and the distributed controller nodes that are represented by the Road Side Units (RSUs) will be completed through the cloud service to accommodate the mobility of the vehicle as it is going to hand off from an RSU coverage to a different one depending on its speed. To avoid the unnecessary repetition of the authentication phase, the authentication process will be handled by the cloud service that stores all vehicle's security credentials. After that, when a vehicle moves to the proximity of a new RSU under the same administrative authority of the most recently visited RSU, the two RSUs can cooperate to verify the vehicle's legitimacy. Also, the third use case supports driver static and continuous authentication as a driver monitoring system for the sake of both road and driver safety.

The proposed framework consists of four main phases: the enrollment phase, the registration phase, the mutual authentication phase, and the key agreement phase.

3.3.1 LMAP^{2}A Phases

3.3.1.1 Device Enrollment Phase

The enrollment phase is completed at the manufacturer's site before the devices are shipped to the service provider or user. First, the manufacturer will load the devices with a real device ID. The manufacturer will then load the device with information and a list of the supported cryptosystem techniques. Second, a large number of Challenge-

Response Pairs (CRPs) associated with the device's PUF are stored in a database that is hosted on the manufacturer's Cloud, as shown in figures 19 and 20. The manufacturer sends random challenges to the device. Then the device processes the challenges using PUF and replies with the corresponding responses. The CRPs and devices' real IDs are stored in a highly secured environment. Lastly, the manufacturer stores the CRPs in the database and the real ID of the device. The manufacturer can then repeat this procedure as much as needed.

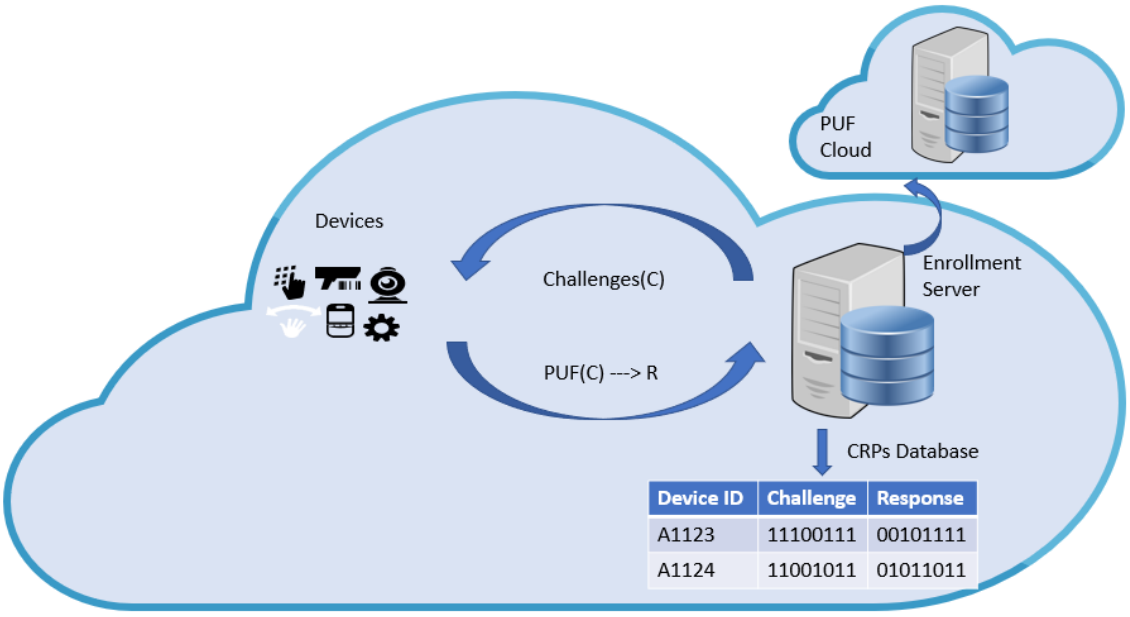


Figure 19: Device Enrollment Phase

Enrollment of a device

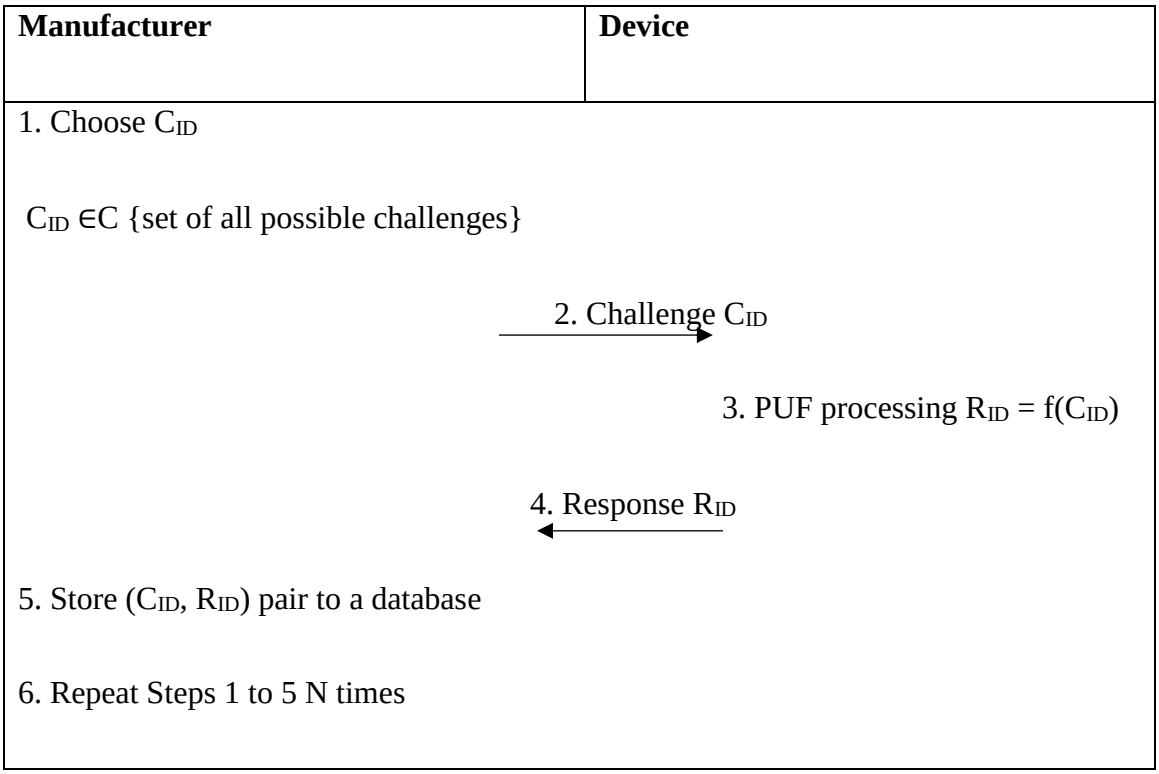


Figure 20: Device Enrollment Process

3.3.1.2 Registration Phase

During this phase, the communicating devices are assigned their Alias IDs, getting the Alias ID and Real ID of the other party. The communicating parties will communicate using their pseudonym IDs. Because the dynamic pseudonym IDs will be changed every authentication session, they will offer the anonymity of the sender's ID, receiver's ID, and the sender-receiver relationship, in addition to device untraceability and end-to-end flow untraceability. Other authentication parameters will be calculated as needed. All these data are loaded into the device's memory through a secured channel. The controller node saves this data in its database and forwards it to the cloud service if needed.

3.3.1.3 Mutual Authentication Phase

During this phase, the communicating parties securely verify each other's identities that will be sent as part of a hash function or an encrypted message. This phase can be completed through direct communication in case the communicating parties are stationary. Also, this phase can be achieved through the cloud service when the IoT device is mobile and is trying to communicate with a distributed controller node. Different lightweight cryptographic primitives such as ECDH, one-way hash function, AES, PUF, chained hash PUF, and Exclusive-OR are utilized in this phase.

3.3.1.4 Key agreement phase

During this phase, the two parties agree on a symmetric session key to encrypt the data exchanged. Once the mutual authentication is completed, the two sides will generate the new shared secret key (*ssk*) that will be derived from the shared secret parameters as well as the PUF challenges and their corresponding responses. AES will be used as a symmetric encryption algorithm to encrypt and decrypt all the subsequent data that will be sent between the communicating parties.

3.3.2 Components of the Architecture

LMAP^{2}A consists of four layers: the physical layer, edge layer, fog layer, and cloud layer.

3.3.2.1 Physical layer

This layer consists of IoT devices that can be resource-constrained devices or general-purpose computing devices. All IoT nodes are equipped with a PUF, and any attempt to tamper with the PUF will change the device's behavior and render the device useless.

3.3.2.2 Edge Layer

The edge layer consists of devices that have more resources than the IoT devices. The edge devices support different communication protocols and facilitate the communication between the IoT devices and the fog nodes. The edge nodes are within one/two-hop distance from the IoT nodes. Edge resources have varying ranges of computational capabilities, from highly capable devices such as mini data centers to less capable such as tablets or smartphones. Edge layer resources are assumed to have device-to-device connectivity within the layer and reliable connectivity to fog layer.

3.3.2.3 Fog Layer

The Fog layer resides on top of the edge. It consists of networking devices such as routers and switches with high computing capabilities. The fog layer acts as an interconnecting link between the IoT devices or the edge layer on one side and the cloud on the other side. The fog layer can perform protocol conversion and provides other services such as data aggregation, filtering, and dimensionality reduction. The devices on the fog layer can also act as a local repository to temporarily store sensors' generated data, provide local processing capability, and perform data cleaning, aggregation, analysis, and interpretation techniques because it has a local database.

3.3.2.4 Cloud Layer

The cloud layer resides on top of the fog layer. It is a consolidation of devices such as servers with the highest computing and storage capabilities. The cloud layer in the LMAP^{2}A consists of the PUF cloud, where the PUF challenges and responses (CRPs) of the devices are stored, and the service provider cloud.

3.3.3 How LMAP^{2}A Address Security Challenges

Identifying the security goals serves as a first step to ensure the implementation of an efficient and lightweight mutual authentication security operation that can fit the IoT devices' resource-constrained nature. The following section covers the main characteristics of LMAP^{2}A that satisfy the security goals. All the security characteristics of LMAP^{2}A will be inherited to the proposed schemes. LMAP^{2}A employs different cryptographic techniques, as presented in figure 21.

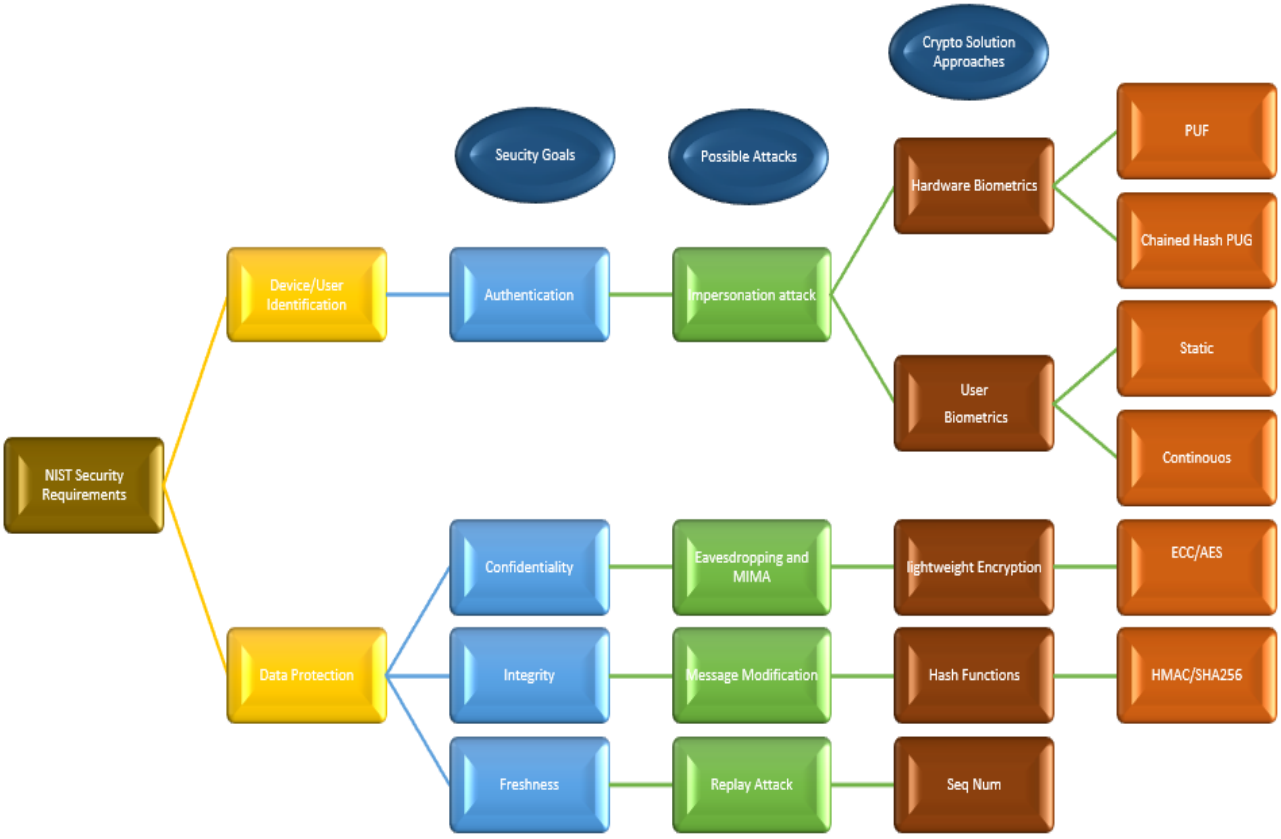


Figure 21: LMAP^{2}A Security Approaches

1. **Device authentication**: device authentication is achieved using physical identifier. The physical identifier is applied through Physical Unclonable Function (PUF) technology that acts as a hardware fingerprint. A delay-based

arbiter PUF-is used for authentication and secret key generation. An arbiter PUF is used as the root of trust in our architecture. From the security perspective, the employment of PUF in the proposed framework is considered one of the key assets.

2. **Key-length Security Level**: based on the current security needs, the employed security level is AES 128-bits. Therefore, cryptographic algorithms are chosen to satisfy this security level.

3. **No storage of shared secret keys locally**: the proposed framework requires that the secret keys are not stored on the resource-constrained IoT devices.

4. **No use of a trusted third party**: The proposed framework does not require the use of a digital certificate that requires the involvement of a third party during the mutual authentication process between the IoT node and the fog node.

5. **No sharing of Shared Symmetric Key (*SSK*) over the network**: the proposed framework does not require an exchange of shared secret symmetric keys over the network to be resistant against side-channel attacks. All shared secret keys are generated locally on the devices.

6. **Confidentiality**: To ensure the confidentiality of the data and to be sure that only authorized users access it, the communication is encrypted. Symmetric cryptography can be used to achieve confidentiality. However, symmetric cryptography requires the two parties to share the same key, and this can be achieved by a key-establishment protocol. AES is used as a symmetric

cryptography technique. Elliptic curve Diffie Hellman (ECHD) is used for key establishment. ECDH is selected because it is lightweight and requires less computation to fit the nature of the resource-constrained IoT devices. ECDH is an acceptable standard and has low-overhead properties.

7. **Data freshness**: It is achieved using timestamp and nonce to ensure the message's freshness.

8. **Data integrity**: It is achieved using either one-way function SHA-256 to ensure the message has not been altered or modified.

9. **Privacy and anonymity**: The proposed framework uses both a real ID and an Alias ID for each device. The nodes' real IDs will never be released in a cleartext format. Also, the real IDs are transferred over the network in an encrypted format. Furthermore, the IoT Alias ID is updated for every authentication session. Accordingly, the adversary will not be able to monitor the activities of the IoT devices.

10. **Server Impersonation**: to ensure the robustness of the proposed framework against server impersonation attacks, the server authenticates itself to the IoT node before conducting any further steps in the authentication process.

11. **Defense in Depth**: the proposed framework implements the defense in depth concept by applying the multi-factor authentication technique.

12. **Heterogeneity**: LMAP^{2}A supports different security configurations that can be implemented in different circumstances to meet the diverse needs of the IoT devices. The configuration options support both stationary and mobile IoT

devices as well as local and distributed controller nodes. Also, the configuration supports the integration of the user's static and continuous biometrics as another authentication feature. The framework configuration includes multiple alternatives for cryptography strength and key lifetime, the number of authentication factors, and session keys usage. For instance, in a high-risk environment, short-term keys can be employed to limit the damage when an IoT node is compromised.

13. **Scalability**: The scalability here refers to the framework's ability to expand incrementally to support the newly added devices and growing data volumes. The proposed approach addresses the first problem by introducing the PHDA to store and retrieve the CRPs of the IoT devices, which can support Over the Air (OTA) registration on the fly to accommodate new devices to be admitted to the system. Also, the proposed framework allows the authenticated IoT devices to move from one controller node to another and be authenticated to the new controller node with the help of the most recently visited controller node.

3.4 Evaluation Process

There is strong agreement among the IoT cybersecurity research community on how to evaluate and compare the effectiveness of the authentication frameworks using both security analysis and performance analysis [57], [58], [59], [60], [61], [62] and [63]. We plan to use and extend this approach to measure the effectiveness of the proposed schemes. Figure 22 provides a roadmap for the evaluation process as follows:

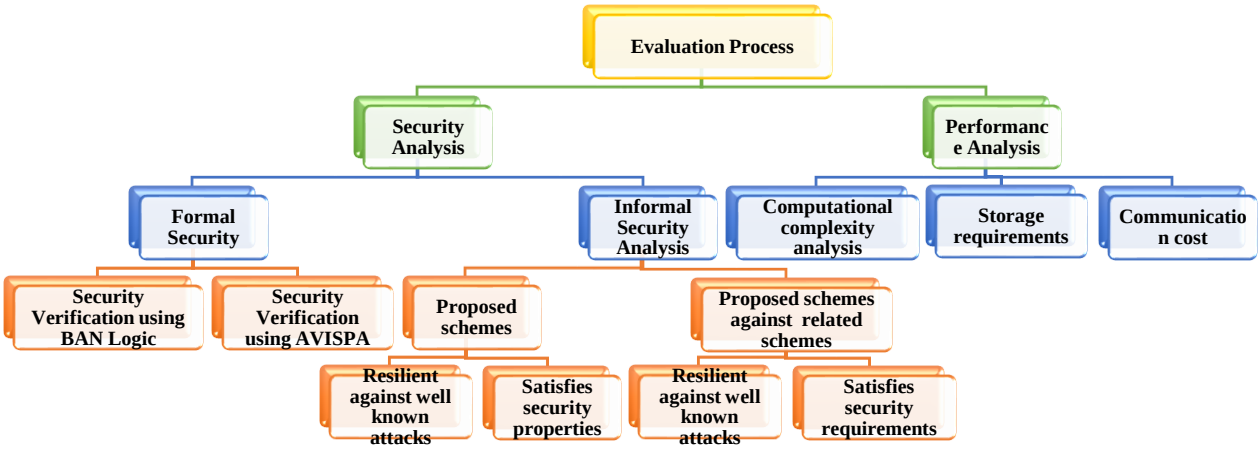


Figure 22: Proposed Framework Evaluation Process

3.4.1 Phase 1: Security Analysis

The security analysis will be conducted using both *formal* and *informal* evaluations. The *formal security* evaluation assesses the proposed schemes' resistance to some of the well-known attacks using two approaches: Burrows–Abadi–Needham (BAN) logic and a well-known rule-based simulation called AVIPSA. To confirm and demonstrate the results of the *formal security* evaluation, we will conduct *informal security* evaluation to analyze in details the strength of the proposed scheme against well-known attacks using the best practice approach. The *informal security* analysis will use Dolev–Yao's threat model as a foundation. In addition, during the *informal security* evaluation, we compare the proposed schemes with the other existing authentication schemes considering their

resistance to the well-known attacks and their satisfaction with the main security requirements.

3.4.1.1 Informal Security Analysis

Informal security analysis is completed to investigate the robustness of the proposed framework against the well-known security attacks and analyze its satisfaction with the main security properties. Dolev–Yao's threat model [64] is used to identify any security issues in any of the presented protocols. It is a powerful adversarial model that is widely accepted as the standard by which cryptographic protocols should be evaluated. The threat model is based on the following two assumptions:

- Cryptography is secure:
    1. The adversary is not able to decrypt a message without the key.
    2. The adversary can't solve the private key pairing of a public key.
    3. The adversary is not able to compute HMAC without the key.
    4. The adversary is not able to guess an encryption key.
    5. The adversary cannot guess a random number that is chosen as part of a security protocol or a random number.
- The adversary can do the following:
    1. Obtain any message passing through the network.
    2. Act as a legitimate user of the network where he can initiate a conversation with any other user.

3. Construct and deconstruct messages.
4. Become the receiver to any sender and be able to read, store, and block every message in transit.
5. Encrypt/decrypt if the encryption/decryption key is known.
6. Manipulate any message.
7. Send messages to any entity by impersonating any other entity.

To evaluate the security of the proposed schemes, we should assume that the adversaries are everywhere in the network. Based on the Dolev-Yao threat model, those adversaries may eavesdrop, manipulate, inject, alter, duplicate, or re-route messages. Therefore, the following security attacks have been considered for examination:

1. **Impersonation Attack**: The adversary uses a trusted user's identity and authentication credentials to get access and send malicious messages to other entities [64].
2. **Replay Attack**: is a form of network attack in which a message is intercepted and maliciously repeated or delayed. A malicious node may repeat the data continuously, which will cause irregular and malicious behavior in the network [65]
3. **Man-in-the-Middle Attack**: is a type of attack where the attacker inserts him/herself into the conversation and tampers with the communication between two parties. The attacker impersonates one or both parties and gains access to the exchanged information between the two parties. This attack takes advantage of

the authentication process's weakness and modifies the communication between the communicating entities.

4. **Eavesdropping Attack**: This is a passive attack. The attacker successfully accesses some secret or confidential information by listening to the communication between the entities [65].

5. **Denial of Service (DoS) Attack**: it is a type of attack where a malicious node keeps sending messages to the receiver node and negatively consumes the network's bandwidth to negatively impact service availability [65]. In summary, the adversary overloads the system with too many requests causing it to crash eventually.

6. **Brute-force attack**: An attacker performs an exhaustive search where he or she uses a trial-and-error method to explore all possible passwords of the user [66].

7. **Side-channel attack**: This type of attack enables an adversary to extract secret keys maintained in a security system. By observing the system's timer, the amount of power consumed by the system to respond to various queries, the adversary can figure out how the encryption algorithm is implemented [67].

8. **Modeling attack**: It is an attack where the adversary collects a large number PUF CRPs and uses a regression algorithm to build a model and predict responses for new challenges [68].

The next step in the *informal security analysis* is to *compare* the proposed schemes with the other existing authentication schemes considering their resistance to the

above-mentioned well-known attacks and their satisfaction with the main security requirements.

3.4.1.2 Formal Security Analysis

The formal security analysis will use a theoretical analysis technique called Burrows–Abadi–Needham (BAN) and a simulation-based engine called AVISPA.

3.4.1.2.1 Burrows–Abadi–Needham Logic

BAN logic has been widely used for the formal evaluation and verification of authentication protocols and to provide proof of the correctness of any authentication protocol [69]. Therefore, this dissertation employs the widely accepted BAN logic to prove that the proposed authentication protocols provide secure mutual authentication between the IoT node and the fog node. The following paragraph presents a summarized introduction about the essential symbols and rules of BAN logic.

3.4.1.2.1.1 BAN symbols and rules

BAN logic is based on a set of postulates and assumptions. BAN uses three objects: principles, encryption keys, and logic formulas. The main notations used in BAN are described as follows:

• P|≡X: P believes the statement X.

• #(X):X is fresh.

• P|⇒X: P has jurisdiction over the statement X.

• P ◅ X: P sees the statement X.

- P|~X: P once said the statement X.
- (X, Y): X or Y is one part of the formula (X, Y).

- $\{X\}_k$: The formula X is encoded/encrypted using the key K.
- $<X>_Y$: X combined with the formula Y.
- $P \overset{K}{\longleftrightarrow} Q$: K is a secret parameter shared between P and Q

3.4.1.2.1.2 BAN logic's rules

The following commonly used BAN logic rules are utilized to prove that the authentication scheme ensures secure mutual authentication and key agreement:

- Message-meaning rule: If P believes that the K is shared with Q and P sees X combined with K, then P believes Q said X.

$$\frac{P| \equiv P \overset{K}{\longleftrightarrow} Q, P \lhd <X>_y}{P| \equiv Q| \sim X}$$

- Nonce-verification rule: If P believes X is fresh, and P believes Q once said X, then P believes Q believes X.

$$\frac{P| \equiv \#(X), P| \equiv Q \sim X}{P| \equiv Q| \equiv X}$$

- Freshness- Conjunction rule: If P believes that X is fresh, then P believes that (X, Y) is fresh.

$$\frac{P| \equiv \#(X)}{P| \equiv \#(X,Y)}$$

- Jurisdiction rule: If P believes that Q has jurisdiction over X and P believes Q believes X, then P believes X.

$$\frac{P| \equiv Q \Rightarrow X, P| \equiv Q| \equiv X}{P| \equiv X}$$

3.4.1.2.2 AVISPA Tool

Authors in [70] introduced Automated Validation of Internet Security Protocols and Applications (AVISPA) to validate and assess the Internet Security Protocols and Applications. AVISPA is widely accepted as a validation simulation in the research community [71]. Also, the authors in [71] evaluated the specifications of several industrial-scale security protocols that are currently being drafted or standardized. In addition, the researchers in [57], [58], [62], and [63] used AVISPA to validate the security features of their protocols. AVISPA is a role-oriented language where each agent plays a distinct role during the execution of the given protocol. We will implement the proposed framework using HLPSL (High-Level Protocols Specifications Language) and then present the simulation results. HLPSL's semantics is based on Lamport's Temporal Logic of Actions (TLA). HLPSL is used to verify security properties such as data secrecy and authentication in message exchanges between agents. HLPSL provides a separate section to define the security properties, called the goal section.

This tool uses a Dolev Yao attacker model presented in section 3.4.1.1, giving the attacker complete control over the network. In general, it can be stated that this threat model has full control over the network. AVISPA verifies the confidentiality, integrity, authentication of communication and data origin, anonymity, non-repudiation, and key-management properties of a given protocol or communication exchange. This tool is a web-based, platform-independent realization and is compatible with standard operating systems.

The protocol is determined, whether SAFE or not, based on predefined goals. The safety of the protocol is evaluated based on the Dolev-Yao threat model. HLPSL specifications are automatically translated into a lower language called the Intermediate

Format (IF) using the HLPSL2IF translator. These translations' main goal and designing the IF language is to offer and serve as an adequate input to the various back-end of the AVISPA toolkit.

AVISPA integrates different back-ends implementing a variety of automatic analysis techniques for protocol falsification by finding an attack on the input protocol and abstraction-based verification methods both for finite and infinite numbers of sessions. AVISPA has the following four back-end tools:

• OFMC Model Checker: The On-the-Fly Model-Checker (OFMC) incorporates several symbolic techniques and algebraic properties to explore the state space in a demand-driven way.

- CL-AtSe Model Checker: The Constraint-Logic-based Attack Searcher (CL-AtSe) translates any security protocol specification written as a transition relation in IF language into a set of constraints that are effectively used to discover any possible attack on the protocol.
- SATMC Model Checker: SAT-based Model checker (SATMC) constructs a propositional formula based on the Transitional state obtained from the IF specification. The propositional formula represents any violation of the security properties, which can be translated into an attack.
- TA4SPModelChecker: The Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) identifies the vulnerability of a protocol or predicts the protocol correctness by accurate estimation of the intruder's capabilities.

The architecture of the AVISPA Tool is presented in figure 23. A user interacts with the tool by specifying a security problem - a protocol paired with a security property that the protocol is expected to achieve in the High-Level Protocol Specification Language (HLPSL). The HLPSL is an expressive, modular, role-based, formal language for modeling communication and security protocols. HLPSL specifications are automatically translated into the IF by the HLPSL2IF translator. The Intermediate Format (IF) is a lower-level language at a lower abstraction level. These translations, in turn, serve as input to the various back-ends that are analysis tools of the AVISPA toolset.

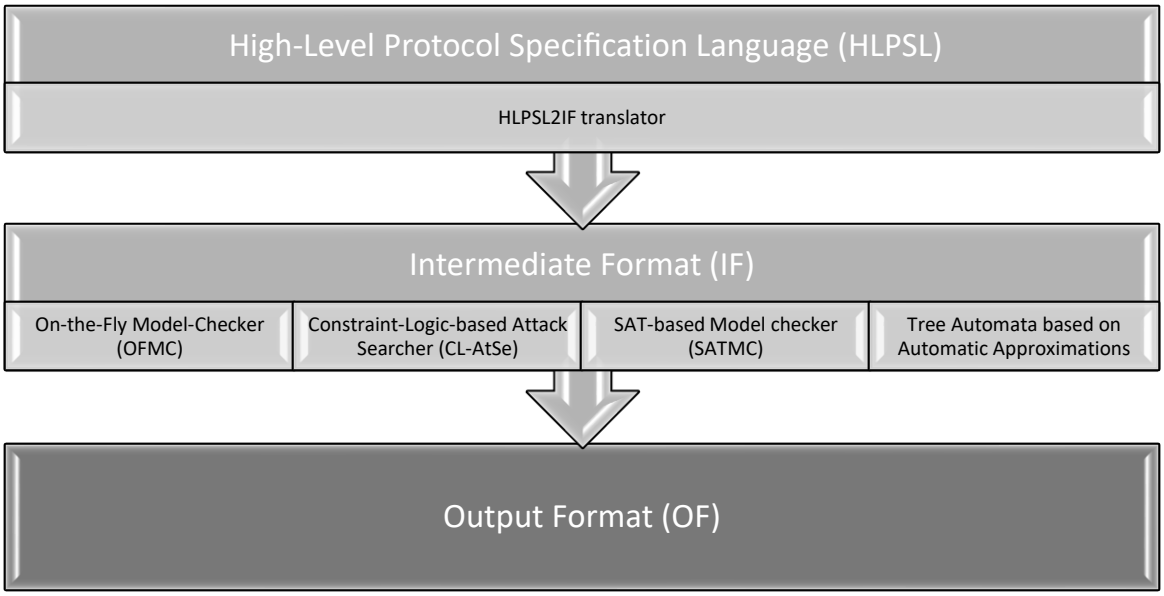


Figure 23 AVISPA Architecture

The entities involved in the communication process are modeled as roles with their message exchanges. Apart from the initiator and the receiver, the environment and the session of the protocol are also roles in HLPSL. The roles can be basic or composed depending on if they are constituent of one agent or more. A session is created between the roles to achieve the required message exchanges. The session also considers an

intruder in the process. Finally, the environment is created for all the sessions simultaneously.

3.4.2 Phase 2: Performance Analysis

The proposed framework implement the performance evalution to analyze the proposed schemes and compare them with other related protocols regarding computational complexity, communication cost and storage requirements. The computational complexity accounts for the execution time of the crypto operations such as one-way hash function SHA-256, xor function, HMAC, PUF response generation, performing ECC, encryption, and decryption using ECC and AES. The communication cost computes the size of the message in bits. The message size varies depending on the message content. Finally, the storage requirements calculate the needed storage space on the communicating parties.

3.5 Summary

This chapter presents the proposed framework and the proposed PUF distributed architecture. The different phases of the proposed framework are described. The evaluation process and the employed tools that will be implemented to evaluate the proposed work are presented in detail. The next chapter presents a new secure, lightweight mutual authentication scheme for stationary and mobile IoT nodes.

CHAPTER IV

LIGHTWEIGHT PRIVACY PRESERVATION AND MUTUAL AUTHENTICATION PROTOCOL FOR SMART HOMES USING PHYSICAL UNCLONABLE FUNCTIONS

With the exponential growth of IoT technology, there is an increasing demand for secure IoT authentication protocols. Most of the currently published protocols depend on computationally expensive mechanisms such as public-key encryption. In contrast, most IoT devices are resource-constrained, which may not handle such security approaches.

This chapter proposes a secure, lightweight mutual authentication, key exchange, and a privacy preservation protocol for IoT smart homes and similar environments where both the IoT device and the controller node are stationary and within range. The presented protocol is called Lightweight Privacy Preservation and Mutual Authentication Protocol for Smart Homes Using Physical Unclonable Functions. This protocol adopts the cloud-fog-device architecture.

The IoT node communicates and establishes a session with the controller node using dynamic anonymous identity, and the two sides agree on a symmetric key in an unlinkable manner. The protocol also sets up virtual domain segregation based on the IoT device's type to apply the same service agreement level and enforce the same security

policies on similar devices. We also evaluate the proposed protocol by using different measures, namely, security and performance evaluation. The security evaluation will be both formal and informal. The formal analysis uses the Burrows–Abadi–Needham logic (BAN), the automated validation of internet security protocols and applications (AVISPA) toolkit.

Furthermore, the protocol efficiency is evaluated and compared with other related protocols. Through the informal analysis, we will assess the proposed protocol against well-known security attacks. We also validate the efficiency of the proposed authentication mechanism in terms of storage requirements, computational cost, and communication overhead.

In Section 4.1, we first introduce the motivation of the work. Section 4.2 presents the related work and the other existing protocols. Section 4.3 demonstrates the use of case application and proceeds with defining the requirements that act as guidelines during the proposed protocol design. In addition, the section discusses the IoT virtual domain segregation process. Section 4.4 presents the proposed protocol. Section 4.5 illustrates the protocol evaluation process, and finally, section 4.6 summarizes the chapter.

4.1 Motivation

There are two types of security threats, which are passive attacks and active attacks. In passive attacks, the adversary tries to collect data and learn about the system without affecting its resources. This type of attack can take the form of eavesdropping or traffic analysis. Through eavesdropping, the adversary intercepts the ongoing communication without consent from the authorized parties. Through traffic analysis, the adversary

monitors traffic patterns to deduce helpful information for later use. Both attacks are hard to detect, and therefore we focus on preventing them.

On the other hand, in active attacks, the adversary tries to alter system resources or affect their operations. The attackers may try to modify the data stream or introduce fake data to the system. The most common active attacks are man-in-the-middle attacks, replay attacks, message modification, and denial of service. In a man-in-the-middle attack, the adversary tries to be a trusted node to gain privileges. A replay attack allows the adversary to passively capture the messages and retransmit them to produce an unauthorized effect. The main goal of the denial-of-service attack is to interrupt the availability of the system [72].

4.2 Related Work

In [73], the authors propose an efficient IoT device authentication protocol that employs the keyed hash algorithm. The proposed protocol avoids the use of a certificate authority to fit the protocol to constrained applications. The proposed protocol increases efficiency by minimizing the number of message exchanges. The results show that the proposed authentication protocol improves the security level and reduces devices' resource consumption.

The author of [74] presents PUF-based algorithms for resource-constrained IoT devices. The author uses the PUF-based algorithm for device enrollment, authentication, encryption, decryption, and digital signature generation. The author highlights the impact of the environmental variations under which the IoT devices will be operating and suggests using error correction codes to overcome PUF aging. The researcher concludes

that the implementation of PUF based elliptic curve protocols over elliptic curves are ideal in the IoT environment. Also, he states that elliptic curve cryptography provides security with low computational and storage requirements. Finally, the author argues that PUF provides low-cost tamper resistance for IoT devices.

The authors in [75] propose a lightweight mutual authentication scheme for IoT devices. A block cipher algorithm is employed in this scheme to conduct the mutual authentication between the IoT device and the server. The protocol requires a secret key to be initially stored in both the IoT device and the server. A session key is dynamically generated using this secret key. The authentication mechanism is a typical application of encryption/decryption cryptography algorithms in authenticating IoT devices. This, in turn, makes the IoT device vulnerable to being compromised by an adversary who has remote access to the IoT device and uses a side-channel attack technique.

In [76], the authors present an IoT authentication protocol based on PUF. Initially, the IoT device registers its credentials, including the device serial number and fingerprint generated using a PUF. Then, the gateway proves the device's authenticity by matching the device's PUF fingerprint with what is stored in the gateway's repository. The IoT device and the gateway share an initial secret key; the IoT device stores the secret key in its non-volatile memory. The gateway stores the secret key in its repository. The initial secret key is used to encrypt and decrypt the initial message processes. The proposed authentication protocol requires that the IoT device stores an initial cryptographic key in its secure non-volatile memory to set up an initial connection with the gateway. This proposed protocol suffers from the same vulnerability that is previously presented in [75],

where storing an initial secret in IoT devices can be compromised by an intruder using a firmware side-channel attack to retrieve the secret key.

The authors in [77] introduce a mutual authentication scheme for an IoT system. In the proposed scheme, authentication occurs between the IoT device and the server. The scheme is based on hashing feature extraction and asymmetric cryptography. The authors improve IoT security and decreased computation and communication costs by combining one-way hashing using SHA1 and feature extraction. This combination helps avoid any collision attacks. The authors conduct a security analysis and conclude that the proposed scheme is secure for application and consumes low computation and memory resources.

The authors in [78] propose an IoT authentication scheme for a smart home system. The presented protocol uses elliptic curve cryptography (ECC) for authenticating IoT devices. The proposed protocol uses a Wi-Fi gateway to complete the initial system configuration, authenticate IoT devices, and provide a means for the user to set up, access, and control the system through an Android-based mobile device. The protocol facilitates the authentication between a user and an IoT device via the home gateway. The presented protocol does not support user traceability and user anonymity.

The authors of [79] propose a mutual authentication protocol using public-key encryption and the IoT device's capability to calculate a predesigned functional operation. The server and the IoT device store their private key and each others' public key. The proposed protocol has two main weaknesses. First, the IoT device and the server store their long-term private key, which an adversary can extract by conducting firmware attack techniques. Second, the proposed protocol employs public cryptography, which is

considered computationally expensive and resource-consuming for resource-constrained IoT devices.

The authors in [80] present a Linear Feedback Shift Register (LFSR)-based PUF. The proposed protocol is based on generating a response to a challenge, then feeding the response to another PUF as a challenge. The two PUF responses are connected using the LFSR. The length of the response can be increased by increasing the number of states of LFSR without any changes in the hardware. The proposed protocol's main weaknesses are the complexity of the proposed design and the absence of security analysis.

The proposed protocol in [81] presents a secure authentication method using zero-knowledge proof for a smart home environment. The presented protocol does not require any secret key during the authentication process between the IoT device and the home gateway. Alternatively, the gateway provides the IoT node with a number to act as a token during the authentication phase. This token can be used later by the IoT device to prove its authenticity to the gateway.

The author of [82] proposes a two-factor one-time password (OTP) technique based on a lightweight identity-based ECC scheme. The proposed protocol achieves both efficiency and security because (1) the Key Distribution Center (KDC) does not require any key storage, and (2) it does not store the private and public keys of the other devices. This presented protocol consumed a small amount of resources.

In [83], the authors develop a secure and efficient architecture for authentication and authorization of resource-constrained IoT devices. The proposed architecture aims to maintain the security, privacy, and accuracy of patient data. The presented architecture shifts the authentication task from the remote end-user to the distributed smart e-health to

accommodate the medical sensors' resource-constrained nature. The proposed architecture is based on a certificate-based DTLS handshake protocol. The architecture is tested via a healthcare prototype structure based on Pandaboard, Texas Instruments (TI) SmartRF06 board and WiSMotes. The CC2538 module integrated into the TI board acts as a smart gateway, and the WiSMotes act as medical sensor nodes. The effectiveness of denial-of-service (DoS) attacks is mitigated through the distributed nature of the proposed architecture. The results indicate that the proposed architecture minimized the communication burden by 26% and reduced communications latency by 16% compared to the delegation-based architecture.

Authors in [84] introduce a lightweight and secure session key establishment scheme for smart home environments. Both the control unit and the sensors use a token to establish trust and a secret session key. The short authentication token ensures mutual authentication between the IoT resource-constrained device and the control unit. The researchers use the AVISPA tool to verify the authentication and confidentiality attributes. Also, the proposed scheme's security analysis proved that the proposed scheme is secure against the Dolev-Yao attack model. The performance and efficiency of the proposed scheme are evaluated using a testbed. The results indicated that the proposed scheme achieved security goals as expected.

Authors in [85] present a machine-to-machine mutual authentication protocol that relies on asymmetric cryptography for resource-constrained IoT nodes in the smart home domain. The proposed protocol enables the IoT nodes to authenticate each other and generate a dynamic shared secret using a pseudo-random generator in every session without any human intervention. The researchers use SPAN/AVISPA toolkit as a security

evaluation tool. The results indicate that the introduced protocol is resilient against well-known attacks, such as man-in-the-middle attacks, replay attacks, and eavesdropping attacks. However, using the pseudo-random number generator to generate the shared session key does not provide complete randomness to the key generation process. Additionally, the use of asymmetric cryptography does not fit the resource-constrained nature of smart home IoT devices.

In [86], the authors introduce a new authentication scheme to authenticate resource-constrained IoT devices in smart homes. The proposed protocol uses PUF and Physical Key Generation (PKG) as an alternative to public-key cryptography. The system generates a secure key based on the PUF, producing unique physical parameters for each IoT device. The presented protocol provides secrecy and authenticity against attackers.

The authors of [87] introduce a PUF based mutual authentication scheme between the IoT devices and the gateway using the CRPs stored inside the gateway. The protocol enables the users to authenticate themselves with the gateway to communicate with the IoT device using a session key generated between them. Timestamp data are used to ensure security against replay attacks.

The authors in [57] present a lightweight mutual authentication protocol for IoT. The researchers use physically unclonable functions (PUFs) and XOR functions to provide security. First, the proposed authentication protocol sends the same device ID in a hash format for every authentication session. As a result, the adversary can trace the activities of the IoT device. Therefore, this protocol does not protect the privacy of the IoT device. Second, during the setup phase, the server retrieves the IoT CRP's from the manufacturer’s cloud and stores them locally on its memory. This action may lead to

different problems. First, the protocol is not resistant to the stolen-verifier attack [61], where the adversary can obtain verification information stored in the server. Second, if an adversary hacks the server, the adversary could access all the CRP's, leading to server impersonation attacks. Third, the adversary can also use the retrieved CRP's to conduct modeling attacks by using the available challenges and responses to build a model and predict the responses of new challenges. Fourth, storing all the CRP's on the server's local storage will not support system scalability because it will require vast storage space. On another side, the presented protocol supports mutual authentication but does not support multiple authentication sessions concurrently.

The authors in [59] propose a PUF based secure communication and key generation protocol for IoT. The researchers use the PUF to generate the public key. The presented protocol has multiple weaknesses. First, the protocol sends the device's real ID in cleartext, which does not preserve its privacy. Also, it does not protect anonymity and untraceability security features. Second, the protocol does not authenticate the server first, leading to server impersonation attacks. Consequently, the IoT node will calculate the responses and calculate its private and public keys without verifying the server's authenticity. Third, during the enrollment phase, the server sends challenges to the IoT node that generates the corresponding responses. Those CRPs are stored on the server. This technique does not protect the system from enrolling fake devices. The protocol does not have any mechanism to ensure that the IoT device is an authentic device and not a cloned or a fake device. Therefore, this protocol does not protect the IoT system against counterfeiting. Fourth, the presented protocol does not support IoT system scalability. The server stores all the CRPs of the IoT nodes, which requires considerable storage space on the server. This, in turn, may

hinder the system's scalability. Fifth, the protocol is not resilient against a stolen-verifier attack where the adversary can obtain the CRPs that are stored on the server's storage. Sixth, the protocol is not resistant to replay attacks because the time stamp has never been updated, and therefore, there is no way to ensure the message's freshness.

The author in [60] proposes a mutual authentication protocol using PUF, ECC, and simple XOR functions. The presented protocol has multiple weaknesses. The author in [60] shares three weaknesses with [59]: (1) the protocol sends the device's real ID in clear text, which does not preserve the device privacy and also does not protect anonymity and untraceability security features; (2) the CRPs are generated and collected during the enrollment phase. Those CRPs are stored on the server. This technique does not protect the system from counterfeiting, and (3) the protocol is not resistant against the replay attack because the timestamp is a constant value during the authentication process. Therefore, there is no way to ensure the message's freshness.

Besides, the presented protocol is subject to a brute force attack. The adversary can capture the IoT node's first message that contains the node ID and the TS in cleartext. The server calculates a value called h_{11} = h (C1 $\oplus$ C2 $\oplus$ TS), a value called C'_2 = $C_2 \oplus h(R_1 \oplus TS \oplus D_2)$, along the same C_1 in cleartext. The adversary can brute force h_{11} to get the value of C_2. Once the adversary succeeds in getting the value of C_2, he can XOR C_2 with C'_2 to get h ($R_1 \oplus TS \oplus D_2$). Because the adversary already has D_2, and TS, he can conduct a brute force attack to find R_1. This significant weakness will corrupt the authentication process and allow the adversary to collect enough CRPs to build a model that can be used to predict responses for new challenges.

The authors in [61] introduce a lightweight and privacy-preserving two-factor authentication protocol for IoT devices. The authors use PUFs as one of the authentication techniques and an XOR function for data encryption. The presented protocol protects the device's privacy and ensures the anonymity of the devices. The researchers use the reverse fuzzy extractor to eliminate the issue that occurred because of noise during the operation of the PUF. During the setup phase, the server sends the IoT device the PUF challenges, and the IoT device responds with the corresponding PUF responses. Because this process is completed during the setup phase, there is no way to verify the IoT device's originality, making this protocol vulnerable to IoT device counterfeiting. Also, the proposed protocol stores a long-term secret key that is used as the first authentication factor for proving the legitimacy of the IoT device on the device's non-volatile memory. This makes the IoT device vulnerable to being compromised by an adversary who can use firmware attack techniques to retrieve the secret key.

Authors in [62] present a mutual authentication protocol for IoT devices using physically unclonable functions. The researchers use PUFs and message authentication code (MAC) to build the authentication protocol. The presented protocol is a lightweight protocol, but it uses the real identity of the IoT device while completing the authentication phase. As a result, the adversary can monitor the activity pattern of the IoT device. Therefore, the protocol does not protect the privacy of the IoT device.

In [63], the authors introduce a lightweight mutual two-factor authentication scheme using PUF and hashing algorithms to secure authentication and session key agreement between the IoT device and the server. The researchers conduct a formal analysis using BAN logic to validate the protocol's security. They also conduct informal analysis and

evaluate their proposed protocol against different types of attacks. The researchers concluded that the proposed protocol fits the needs of resource-constrained IoT devices. The presented protocol is a lightweight protocol, but it uses the IoT device's real identity during the authentication phase, which does not protect the IoT device's anonymity and untraceability.

Based on the work presented above, there are common limitations that are shared among most of the presented schemes. First, in the schemes presented in [59,60,61], the CRPs are generated and collected during the setup phase and outside the manufacturing process. This does not allow the opportunity to discover counterfeited devices from original devices. Second, using PUF generates considerable interest in terms of authentication. One of the main drawbacks that is clear in many of the protocols mentioned above [57,59,60] is storing challenge and response pairs on the server-side. Many of the presented protocols store the CRPs on the server storage, which can be an obstacle to system scalability. This is a significant problem with much of the literature regarding resource-constrained IoT devices. Third, in some of the presented protocols [57, 59, 60,62,63], the researchers overlook the IoT devices' privacy, which does not guarantee the anonymity and privacy of the IoT device. Fourth, the proposed protocol [61] requires storing a secret key hash on the IoT device's memory, which can be retrieved using side-channel attacks.

4.3 IoT Smart Home Network Model and Security Goals

4.3.1 Smart Homes Network Model

The use case presented in this section is illustrated in figure 24. It presents a resource-rich Intelligent Gateway (IG) node that communicates with resource-constrained IoT devices (N) in an untrusted field. The IG is the central device and is aware of all the nodes within its network. It is also responsible for managing the data about each node and starting and maintaining the network. Both the IoT device and the fog node that is represented by the controller node are stationary. The communication between the IoT device and the controller node is subject to both active and passive attacks. In passive attacks, the adversary can eavesdrop on the communication. In active attacks, the intruder may replay, modify, or delete specific communication messages.

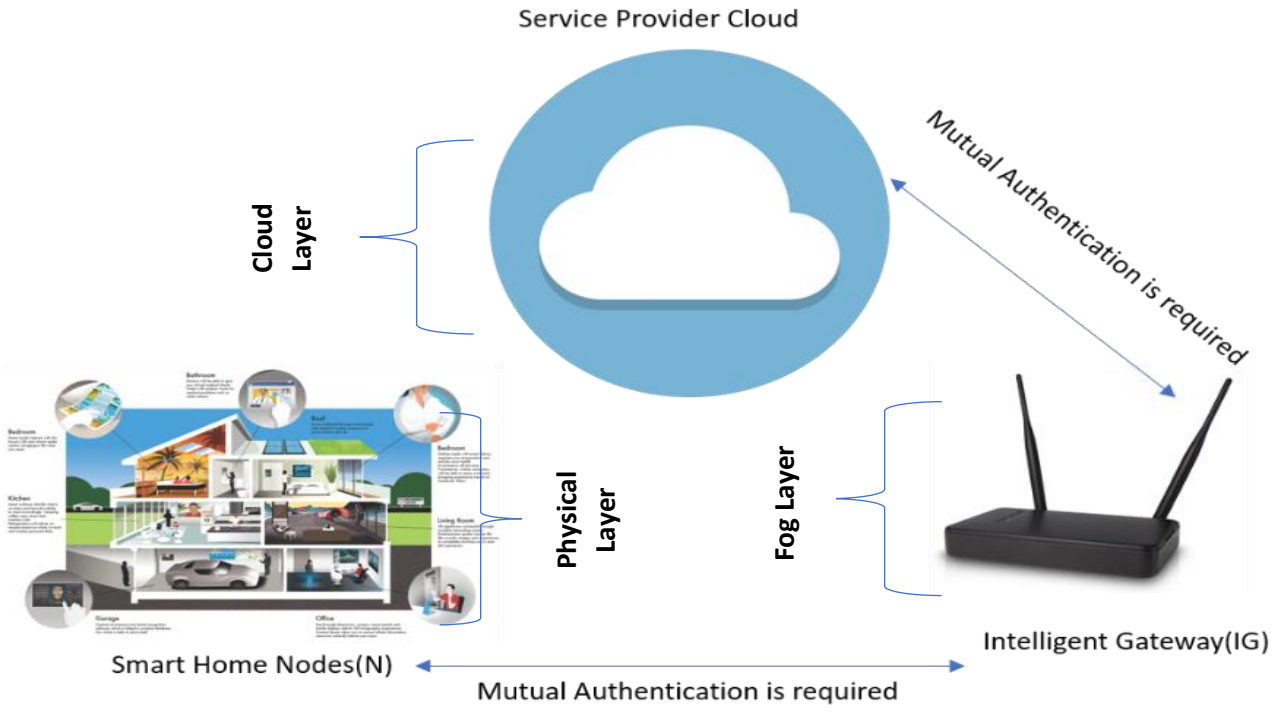


Figure 24: IoT Smart Home Network

Establishing a secure communication between an IoT device (N) and an intelligent gateway (IG) node requires a secure protocol. Authentication, key agreement, and access control are the four critical components of the scheme that we present in this chapter, as demonstrated in figure 25.

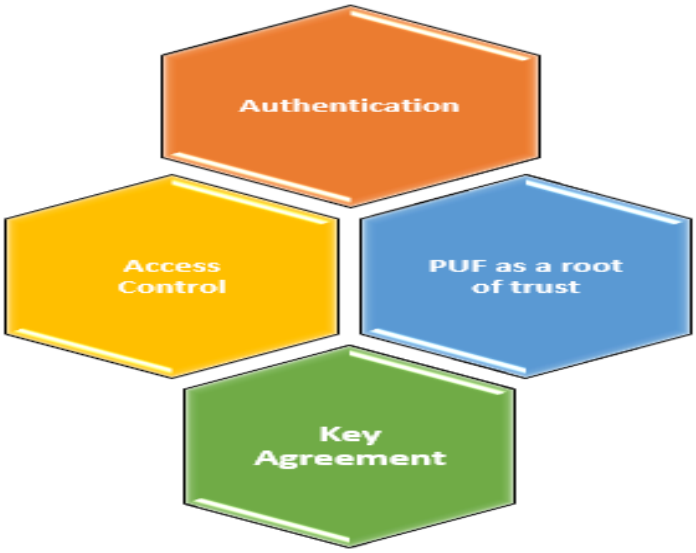


Figure 25: Key Components for Secure Communication

Figure 26 describes the proposed scheme. The protocol consists of four participants: the IoT node (N), the Intelligent gateway node (IG), the service provider cloud, and the manufacturer cloud. The local IoT network includes IoT nodes and an intelligent gateway (IG) that acts as the central node. The IG node oversees starting and maintaining the network, and it is aware of all the devices' alias identities and real identities. IoT devices can be resource-constrained devices or general-purpose computing devices. All IoT nodes and the IG are equipped with a PUF, and any attempt to tamper with the PUF will change the device's behavior and render the device useless. The PUF hardware

security primitive is used as the root-of-trust, which provides device authenticity. The IG simply is a secured node with more resources than the IoT device.

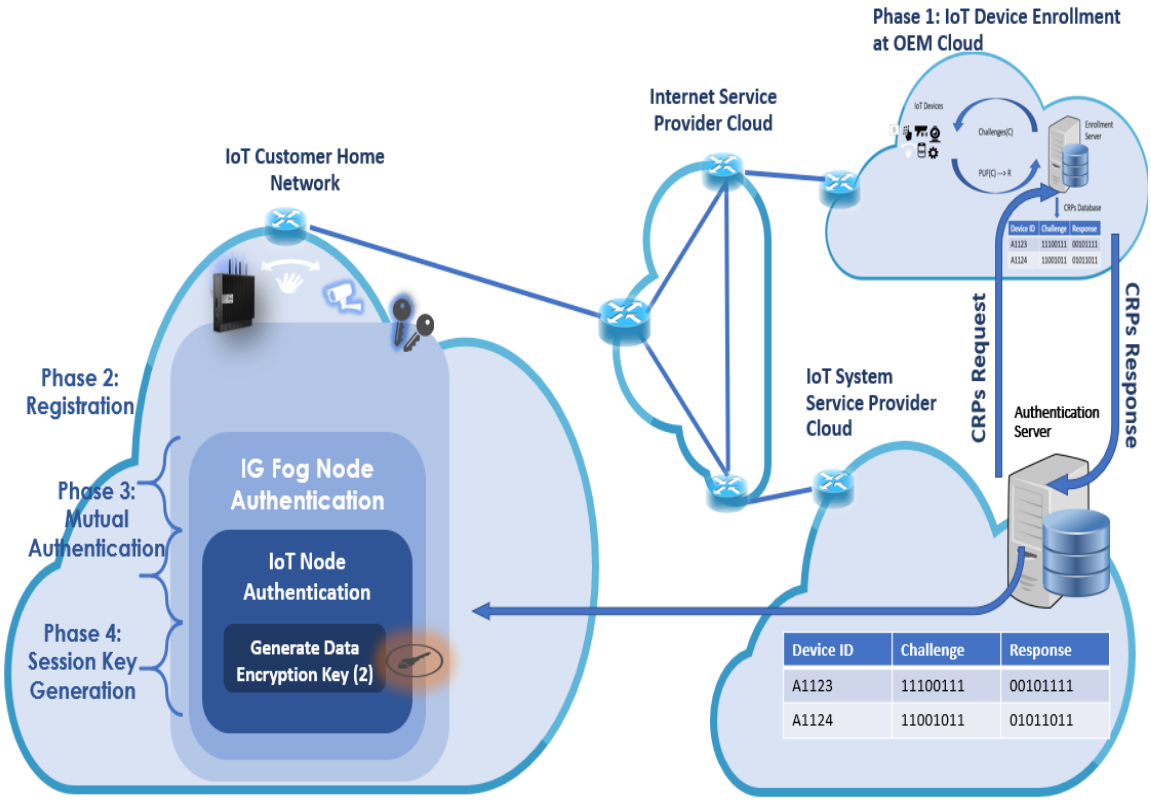


Figure 26: The Proposed Protocol Overview

The IG supports different communication protocols and acts as a communication point between the IoT nodes and the local Internet. It receives data from various sensors, performs protocol conversion, and provides other services such as data aggregation, filtering, and dimensionality reduction. The IG will support different wireless protocols and facilitate inter-device communication. The IG also acts as a local repository to temporarily store sensors' generated data, provides local processing capability, and performs data cleaning, aggregation, analysis, and interpretation techniques because it has a local database. The IoT nodes communicate directly with the IG. A trust relationship and

a secure communication link exist between the IG and the service provider cloud environment. The manufacturer assigns the IoT devices and the IGs real IDs that are stored in the device's memory. Also, the service provider gives the IoT nodes and the IG alias IDs.

The proposed scheme ensures data confidentiality by encrypting the exchanged messages using the AES algorithm with a 128 bits key length as recommended by NIST [16]. In addition, a one-way cryptographic hash function with a 256 bits length is used to validate and ensure the integrity of the data transmitted between the two devices.

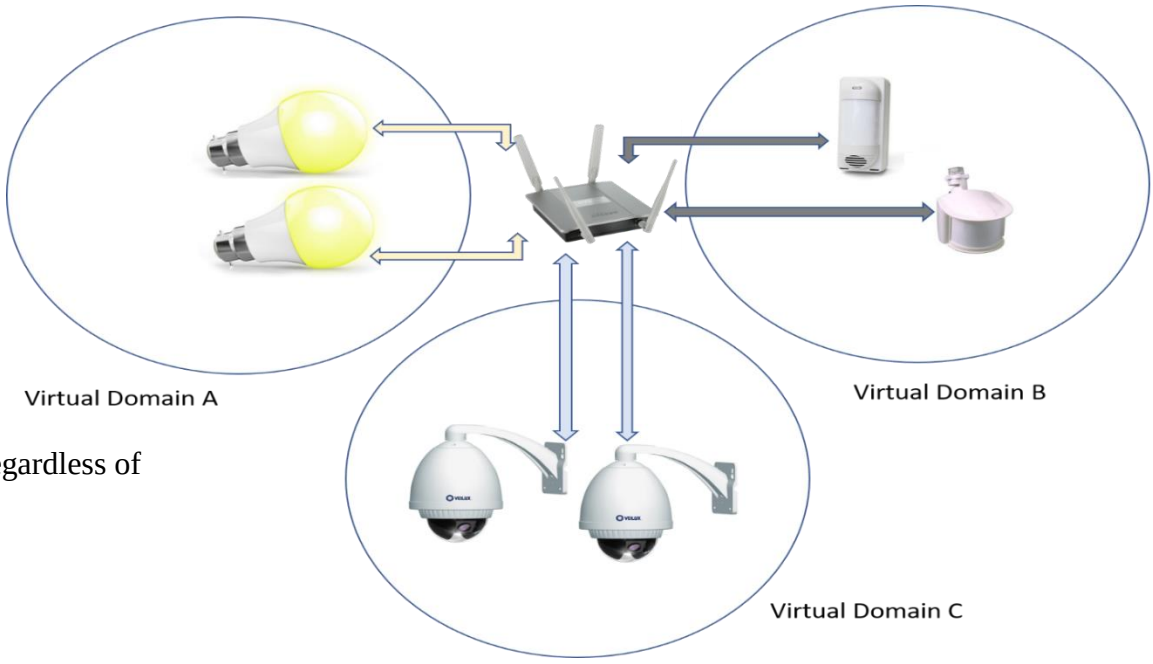
The IoT node will be given two different options for generating a shared secret session key during the authentication process to add more flexibility to the proposed protocol and accommodate IoT nodes with diverse computational resources. The IoT node will select the option that is suitable to its resources and the security requirements.

The proposed protocol supports IoT devices' dynamic addition to the network without causing any changes in the network's present security states. Adding a new IoT device is controlled by the capabilities and the available resources of the IG, which is assumed to be a powerful device because it stores separate information for each IoT device in the network.

4.3.2 IoT Virtual Domain Segregation

To apply security policies that fit the needs of the different IoT devices, the proposed protocol uses virtual segregation among the IoT devices. The IoT network is virtually partitioned based on the type of the IoT device. The segregation process will help the IG to control the communication process in the IoT network.

Virtual segregation is an essential factor to secure the IoT network so that the same policies and security rules can be applied on the IoT devices of the same type



regardless of

Figure 27: Virtual Domain Segregation Based on Device Type

where they are located. For example, as in figure 27, all motion sensors are in the same virtual domain. The same is for the glass breaking sensors and lighting sensors. Each IoT device is assigned a virtual domain ID(V_{ID}) and Type ID(T_{ID}) during the registration phase.

4.4 Proposed Scheme: Lightweight Privacy Preservation and Mutual Authentication Protocol for Smart Homes Using Physical Unclonable Functions

This section presents a lightweight mutual authentication and key agreement protocol for stationary IoT devices. The abstract notations used to describe the authentication protocol are listed in Table 2. The proposed protocol is based on LMAP^{2}A

architecture that is presented in chapter 3. An IoT node (N) and an Intelligent gateway (IG) are responsible for ensuring secure, anonymous mutual authentication and key exchange in the proposed protocol. The proposed protocol consists of four phases: the enrollment phase, registration phase, mutual authentication phase, and the key generation phase. Depending on how the IoT node will choose to generate the session key, the key generation phase can be executed in two different ways.

The enrollment phase will be completed during manufacturing. The Authoritative entity will be in charge of completing the registration phase. Both the mutual authentication and the key generation phases will be conducted between the IoT device and the IG without any human involvement. The two parties are responsible for ensuring secure, anonymous mutual authentication and key generation. The relationship between the IoT devices and the IG is based on the client-server topology. It is assumed that the two communication parties can have direct communication with each other.

4.4.1 Enrollment Phase

The enrollment phase is completed during the manufacturing process. The manufacturers will assign a real ID to every IoT device and every intelligent gateway. Also, the manufacturers will collect a set of CRPs for each IoT device and each intelligent gateway during the enrollment phase. The manufacturers will test all the CRPs under different temperature and voltage conditions and consider the aging effects to keep only the error-free CRPs [[37]. Then, the manufacturers will load the CRPs of the IoT devices and the intelligent gateways to the cloud of their service providers. The service provider containers are part of the PUF architecture that is presented in Chapter 3.

Table 2: Notations Used in the Proposed Scheme

Notation	**Description**
Authoritative Entity	AE
IoT node	N
Intelligent Gateway	IG
N_{IDR}	IoT Node Real ID
N_{IDA}	IoT node Alias ID
IG_{IDR}	Intelligent Gateway Real ID
IG_{IDA}	The intelligent Gateway Alias ID
OTP	One-time password
VD_{IDN}	IoT node virtual domain ID
T_{IDN}	IoT node Type ID
RV_N	The random value generated by the IoT node
RV_{IG}	The random value generated by the IG
Auth1,Auth2, and Auth3	Authentication parameters
TS_N	Timestamp generated by the IoT node
TS_{IG}	Timestamp generated by the IG
PUF	Physical unclonable function
C	PUF Challenge
R	PUF Response
Pub_N	IoT device public key
$Priv_N$	IoT device private key
Pub_{IG}	Intelligent gateway public key
$Priv_{IG}$	Intelligent gateway private key
OTPnew	New OTP
N_{IDANew}	N new alias ID

SSK	Shared secret session key
{}_	Encryption
H	Hash function
⊕	Bitwise xor operation

4.4.2 Registration Phase

The main goal of this phase is to register the IoT device. Before operation in the field, the Authoritative Entity (AE) will complete multiple tasks, as described in figure 28. First, The AE will assign the IoT node (N) an Alias identity (N_{IDA}), virtual domain ID(VD_{IDN}), and type ID (T_{IDN}). N_{IDA} will be a fake ID that the IoT node will use to communicate with the IG. N_{IDA} will be dynamically changed in every authentication. The goal of the N_{IDA} is to protect and ensure the unlinkability and untraceability of the IoT devices.

Additionally, the N_{IDA} supports the anonymity of the node in each session. VD_{IDN} refers to the virtual domain that N belongs to, and the T_{IDN} indicates the type of the IoT device. Then the AE will use N's different ID's and the real ID of the IG to generate three authentication parameters, which are $Auth_1$ and $Auth_2$, and $Auth_3$ using a one-way hash function and XOR operations.

Second, the AE will use a random number generator to generate a one-time password (OTP) between N and the IG. Once N is registered, both N and IG will generate the OTP during every authentication session without any human involvement.

Third, the AE will retrieve CRP (C_{IG}, R_{IG}) of the IG from the PUF cloud. Fourth, the AE will insert and store N_{IDR}, N_{IDA}, the VD_{IDN}, the T_{IDN}, and the OTP in the IG's database and store $Auth_1$, $Auth_2$, $Auth_3$, OTP, N_{IDA}, R_{IG}, and IG_{IDA} in N's memory.

Step 1: The AE assigns an Alias ID to N
Step 2: The AE assigns N to its appropriate virtual domain and generates a VD_{IDN} to the N.
Step 3: The AE generates and assigns a T_{IDN} to each N based on its type
Step 4: AE communicates with the PUF cloud to retrieve a CRP (C_{IG}, R_{IG} of the IG.
Step 5: The AE calculates the three parameters using the VD_{IDN}, T_{IDN}, and the R_{IG}: $Auth_1$, $Auth_2$, and $Auth_3$ as follows:

$Auth_1$: H (N_{IDR} || IG_{IDR}||R_{IG})

$Auth_2$: H(Auth1 ||IG_{IDR} ||VD_{IDN})

$Auth_3$: H (Auth2|| IG_{IDR} ||T_{IDN})

The VD_{IDN}, the T_{IDN}, and the R_{IG} are only used to create $Auth_1$ $Auth_2$ and $Auth_3$. Both the VD_{IDN} and the T_{IDN} are only stored on the IG database. The IoT node does not know anything about those two IDs and the R_{IG}. The C_{IG} is only stored on the IG database, and R_{IG} is unknown to both the IoT node and the IG because the IG will generate it on every authentication session. Using these two IDs to generate the authentication parameters makes it almost impossible for an adversary to identify any IoT node's VD_{IDN} and the T_{IDN}. Also, using the R_{IG} makes it impossible for an adversary to impersonate the IG.
Step 6: The AE generates a one-time token (OTT) that will be used as part of a one-time password (OTP) using a random number generator
Step 7: The AE stores the $Auth_1$, $Auth_2$, $Auth_3$, N_{IDA}, IG_{IDA}, and OTP in the IoT device database.
Step 8: The AE inserts the N_{IDR}, N_{IDA}, IG_{IDA}, Pub_{IG}, VD_{IDN}, OTP, C_{IG}, and T_{IDN} of the IoT node in the IG database.

Figure 28: Registration Phase

The real identities of both the IoT node and the IG represent secure parameters between the IoT node and the IG. The N_{IDR} and the IG_{IDR} will never be transmitted in a plain text format and will be used during the authentication process between the IoT node and the IG.

4.4.3 Mutual Authentication Phase

This phase includes implementing the two-factors authentication process and the agreement on the technique of generating the secret session key (*ssk*) between the IoT

node and the IG. The *ssk* will be generated either by employing a hash function or using the Elliptic Curve Diffie Hellman Key exchange protocol.

This phase includes implementing the two-factor mutual authentication using the PUF, one-way hash function, bitwise XOR operation, and symmetric encryption. The security of exchanged messages in this phase is ensured by encrypting the messages using the AES algorithm with a 128-bit key length. Besides, a one-way cryptographic hash function with a length of 256 bits is used to validate and ensure the integrity of the transmitted data between N and the IG.

During this phase, the two sides negotiate the session key generation technique. The two sides will agree to either generate the session key by using a one-way hash function or by using Elliptic Curve Diffie Hellman Key Exchange Protocol (ECDH). The ECDH Key Exchange is a key-agreement protocol that can be used for deriving a shared secret key [20]. Figure 29 describes the ECDH key exchange process.

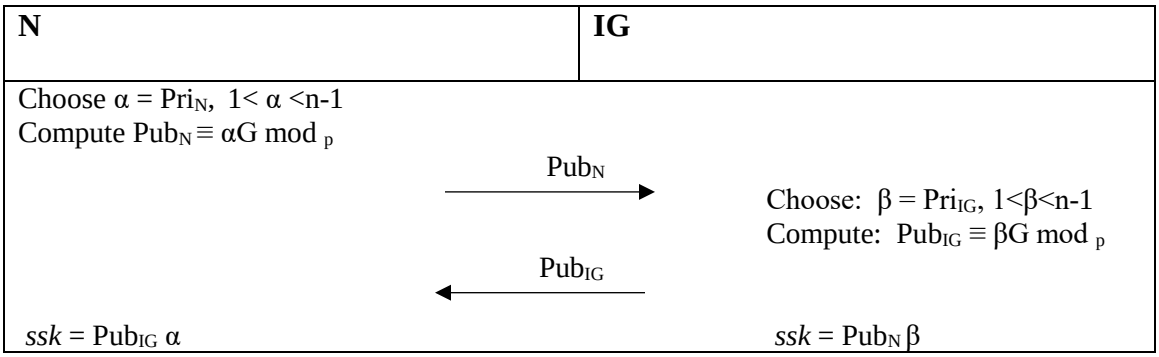


Figure 29: Elliptic Curve Diffie Hellman Key Exchange Protocol

As presented in figure 30, this phase starts when N prepares the connection request message that will be sent to the IG. The preparation process of this message includes the following steps:

1. N generates a new TS_{N1} to avoid replay attacks
2. N Computes NX_1
    a. $NX_1 = H (Auth_3 || OTP)$
3. N selects a random parameter RV_N and then calculates NX_2
    a. $NX_2 = XOR (RV_{N1,} NX_1)$
4. N computes $S_N = H (N_{IDR} || NX_1 || TS_{N1} || RV_{N\,1})$
5. N sends a connection request message.

The connection request includes $N_{IDA,}$ $TS_{N1,}$ X_2, and $S_{N,}$ as shown in 4.1.

$$N \longrightarrow IG\ Conn\text{-}Req\ (N_{IDA}, TS_{N1}, NX_2, S_N) \qquad (4.1)$$

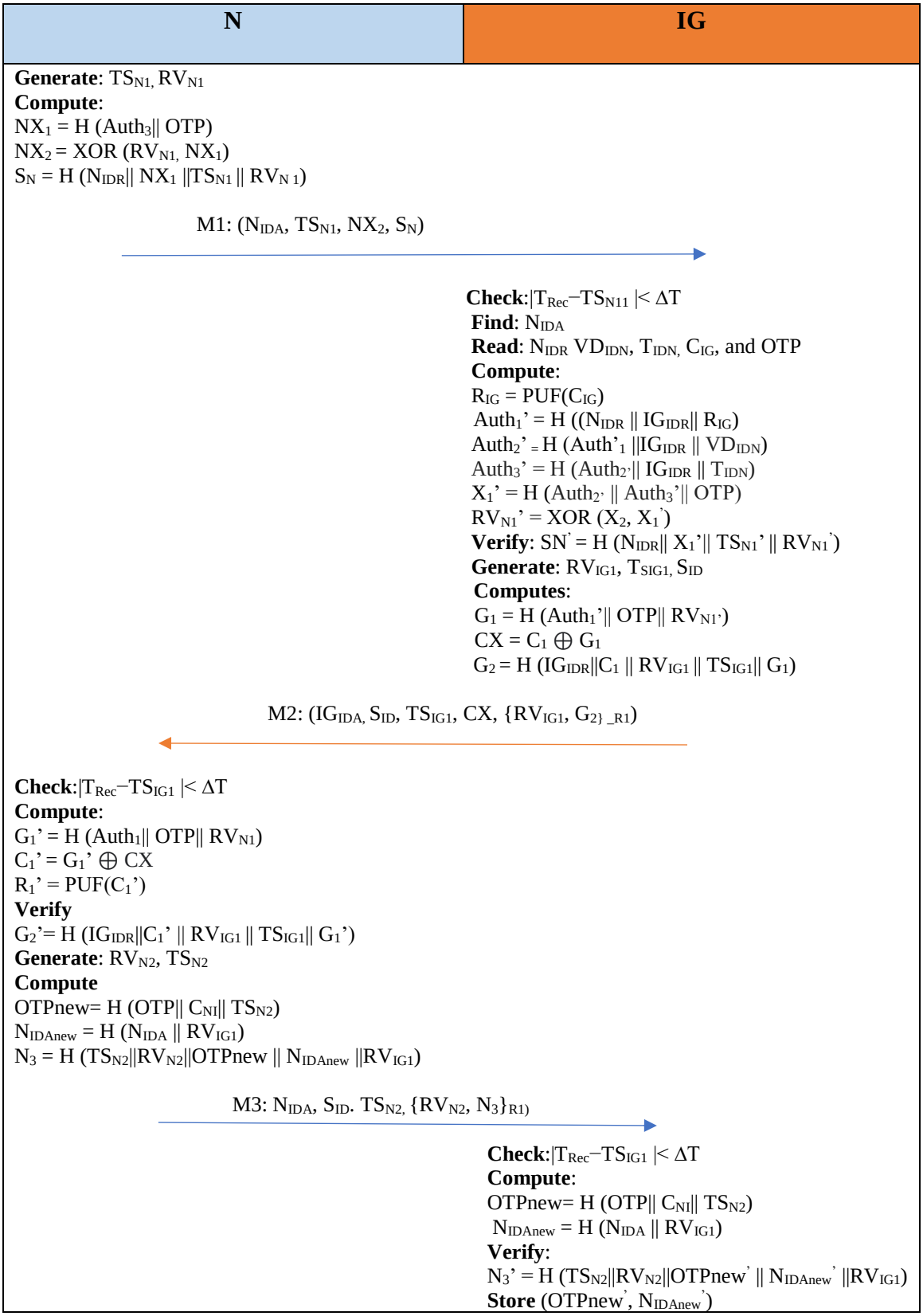


Figure 30: The Proposed protocol authentication process between N and IG

Once the IG receives the connection request, it will complete the following steps:

1. IG checks the validity of the received timestamp $|T_{Rec} - TS_{N11}| < \Delta T$. T_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. IG retrieves from its database the N_{IDR} that corresponds to its N_{IDA}. If the IG did not find the N_{IDA} in its database, it will ignore and drop the connection request.
3. Once the IG finds the N_{IDR}, it will retrieve it along with the VD_{IDN}, $T_{IDN,}$ C_{IG}, and OTP of N.
4. The IG passes the challenge C_{IG} to its PUF function and generates a response R_{IG}
5. The IG uses the generated R_{IG} to compute $Auth_{1'}$ by applying a one-way hash function, as described in 4.2. Based on the calculated $Auth_{1'}$, the IG calculates $Auth_{2}'$ as presented in 4.3. Also, the IG calculates $Auth_{3}'$ by using $Auth_{2}'$ as presented in 4.4.

$$Auth_{1}' = H((N_{IDR} \| IG_{IDR} \| R_{IG}) \quad (4.2)$$

$$Auth_{2}' = H(Auth'_{1} \| IG_{IDR} \| VD_{IDN}) \quad (4.3)$$

$$Auth_{3}' = H(Auth_{2'} \| IG_{IDR} \| T_{IDN}) \quad (4.4)$$

6. The IG computes $X_{1}' = H(Auth_{2'} \| Auth_{3}' \| OTP)$

7. The IG computes the RV_{N1}' = XOR (X_2, X_1')

8. The IG uses the calculated X_1', TS_{N1}', N_{IDR}, and the RV_{N1}' to generate S_N', which is a combination of $N_{IDR,}$ X_1', TS_{N1}', the RV_{N1}' using a one-way hash function as shown in figure 31. If the computed value is equal to the received value, the IG accepts the connection request; otherwise, it will drop it.

$SN^{'} = H (N_{IDR}|| X_1' || TS_{N1}' || RV_{N1}^{'})$
If $SN = SN^{'}$ Then
    The IG will accept the connection request
Else
    The IG will drop the connection request

Figure 31: IoT Node Message Verification

Once the IG verifies the IoT node, the IG communicates with the Grand PUF node to retrieve the PUF challenge (C_{NI}) and their corresponding PUF responses (R_{NI}) through a secure communication channel. The IG will generate a random value $RV_{IG,}$ a session ID to distinguish one session from the rest of the running sessions simultaneously, and a timestamp TS_{IG1}. The preparation process of the connection response message includes the following steps:

1. IG selects a random parameter ($RV_{IG1)}$
2. IG generates a new timestamp (TS_{IG1})
3. IG generates a session ID(S_{ID})
4. IG computers $G_1 = H (Auth_1' || OTP || RV_{N1'})$
5. IG calculates $CX = C_{NI} \oplus G_1$
6. IG calculates $G_2 = H (IG_{IDR}||C_{NI} || RV_{IG1} || TS_{IG1}|| G_1)$

7. IG sends a connection response message

The connection response includes $IG_{IDA,}$ session ID(S_{ID}), TS_{IG1}, RV_{IG1}, CX, and G_2, as shown in 4.5. The message is encrypted by R_{NI} because it is a secured parameter between the N and the IG

$$IG \longrightarrow N \text{ Conn-Res } (IG_{IDA,} S_{ID}, TS_{IG1}, CX, \{RV_{IG1,} G_{2}\}_{R1}) \quad (4.5)$$

Once N receives the connection response, it completes the following steps:

1. N checks the validity of the received timestamp $|TS_{Rec} - TS_{IG1}| < \Delta T$. T_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. N retrieves from its database IG_{IDR} that corresponds to its IG_{IDA}. If N did not find the IG_{IDA} in its database, it will ignore and drop the connection response
8. N computes $G_1' = H (Auth_1 || OTP || RV_{N1})$
3. N calculates $C_{NI}' = G_1' \oplus CX$
4. N computes $R_{NI}' = PUF(C_{NI}')$
5. Then N will use R_{NI}' to decrypt the message to retrieve RV_{IG1} and G_2
6. As presented in figure 32, N calculates G_2'using a one-way hash function and compares the generated G_2' with the received G_2. If the two values are the same, N will accept the connection response and authenticate the IG; otherwise, it will drop it.

G_2'= H (IG_{IDR}||C_{NI}' || RV_{IG1} || TS_{IG1}|| G_1')
If $G_2 = G_2$'Then
Then N will accept the connection response.
Else
N will drop the connection response

Figure 32: Evaluating the IG Authentication Parameter

After ensuring the correctness of the G_2' and verifying the integrity of the sent message, N completes the following steps:

1. N generates the TS_{N2}
2. N generates the RV_{N2}
3. N computes new OTP(OTPnew) = H (OTP|| C_{NI}|| TS_{N2})
4. N computes a new alias ID(N_{IDAnew}) = H (N_{IDA} || RV_{IG1})
5. N computes N_3 = H (TS_{N2}||RV_{N2}||OTPnew || N_{IDAnew} ||RV_{IG1})
6. N selects its technique to generate the session key. N will send either T1 or T2.
    a. If N decides to use a hash function to generate the session key, it will send to the IG the hash function that it will use.
    b. If N decides to use ECDH to generate the session key, N derives its private and public keys using the ECC public domain parameters. N selects a random secret key and calculates the corresponding public key as G^{α}.
7. N sends to the IG a message containing RV_{N2}, TS_{N2}, and Nthree. Also, N will send either the hash function as shown in 4.6a or the public key as shown in 4.6b that will be used to generate the session key. The message will be encrypted by the R_{NI}

$$N \longrightarrow IG \quad N_{IDA}, S_{ID}. TS_{N2,} \{RV_{N2}, N_3, SHA256\}_{R1)} \qquad (4.6.a)$$

$$N \longrightarrow IG \quad N_{IDA}, S_{ID}, TS_{N2,} \{RV_{N2}, N_{3,} G^{\alpha}\}_{R1)} \qquad (4.6.b)$$

Upon receiving N's message, IG completes the steps involved in this process, as listed below.

1. IG verifies the $TS_{N2.}$. IG checks the validity of the received timestamp $|TS_{Rec} - TS_{N2}| < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. IG decrypts the message using the R_1.
3. IG computes new OTP(OTPnew) = H (OTP|| C_{NI}|| TS_{N2}')
4. IG computes a new alias ID(N_{IDAnew}) = H (N_{IDA} || RV_{IG1})
5. IG computes the N_3, which combines TS_{N2}, RV_{N2}, OTPnew, $N_{IDAnew,}$ and RV_{IG1}. If the calculated N_3 equals the received N_3, the IG will accept the message and authenticate N; otherwise, it will end the session as presented in figure 33. Also, the IG will store OTPnew and N_{IDAnew} in its memory for the next authentication session.

N_3' = H (TS_{N2}'||RV_{N2}'||*ssk*'|| OTPnew' || N_{IDAnew}' ||RV_{IG1})
If N3' = N_3 Then
Integrity and authenticity are proved, and The IoT node will be authenticated
Else
The IG node will end the session

Figure 33: IoT Node Authentication Process

4.4.4 Key Generation Phase

Once the mutual authentication is completed, the two sides will generate a shared secret session key (*ssk*) using the technique they agreed upon during the authentication phase.

Suppose the two sides agreed on using the hash function to generate the *ssk*. In that case, the two sides will generate the *ssk* locally by hashing a combination of the PUF responses and the generated random values as presented in 4.9. and figure 34 The uniqueness of the PUF responses ensures the uniqueness of the generated *ssk*.

$$ssk = H\left(R_{NI}^{'} \,||\, C_{NI} || \, RV_{N1}^{'} || RV_{IG1} || \, RV_{N2}^{'}\right) \qquad (4.9)$$

If the two sides decided to use ECHD to generate the *ssk*, the two sides would generate the *ssk* as presented in figure 35. AES will be used as a symmetric encryption algorithm to encrypt and decrypt all the subsequent data that will be sent between N and the IG.

If the key generation phase gets disrupted and the connection gets terminated for any reason, the IoT device and the IG will have to repeat the entire mutual authentication steps.

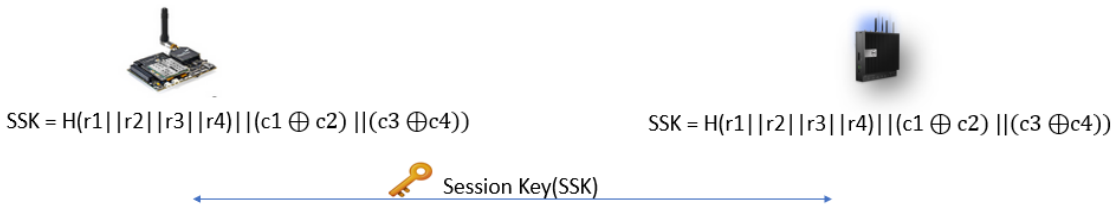


Figure 34: The Process of Generating *ssk* Using Hash Function and CRPs

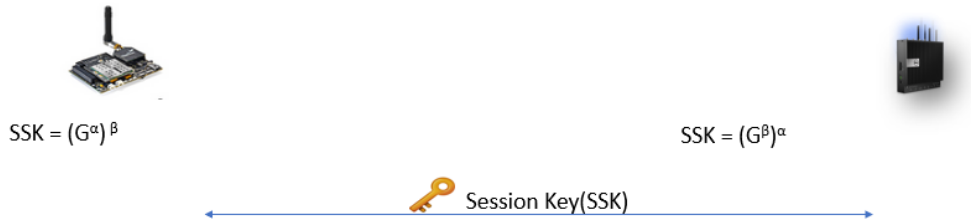


Figure 35: The Process of Generating *ssk* Using ECDH

The algorithm for the proposed mutual authentication mechanism is shown in detailed steps in figure 36.

Algorithm 1 The mutual authentication between IoT device and the IG

Input:

An IoT device with real identity (N_{IDR}), alias identity (N_{IDA}), IG real identity (IG_{IDR}), IG alias identity (IG_{IDR}), Authentication parameters ($Auth_1$, $Auth_2$, and $Auth_3$), G, and one time password (OTP).

IG with with real identity (N_{IDR}), alias identity (N_{IDA}), IG real identity (IG_{IDR}), IG alias identity (IG_{IDR}), the virtual domain ID(V_{IDA}) and type ID (T_{ID}) of the IoT node, G, PUF challenge (C_{IG}), and one time password (OTP).

Output:

Mutual authentication between the IoT device(N) and the IG.

Begin

1. The IoT device generates a random nonce $RV_{N1,}$ a timestamp $TS_{N1,}$ computes the (XN_1) H ($Auth_3$, OTP), NX_2 , xor(RV_{N1},NX_1), and (SN) H(XN_1,TS_{N1},RV_{N1},N_{IDR}).
2. IoT device sends (N_{IDA}, TS_{N1}, XN_2, SN) message to the server.
3. **If** (the IG finds N_{IDA} in its repository) **then**
4. The IG retrieves N_{IDR}, OTP, V_{IDA}, C_{IG}, and T_{IDA} that belongs to the N_{IDA} from its repository to its memory.IG passes the challenge C_{IG} to its PUF function and generates a $R_{IG.}$
5. The IG calculates $Auth_1$', $Auth_2$', and $Auth_3$'. Then the IG computes NX_1, retrieves the RV_{N1} from XORing NX_1 and NX_2 and finally calculates SN' H (NX_1, TS_{N1}, RV_{N1}, N_{IDR}) message.

6. **If** (the calculated hash message in step 6 matches the hash message that was In step 2) **then**
7. The IG communicates with the Grand PUF node to a CRP, generates a timestamp $TS_{IG1,}$ generates a random nonce RV_{IG1}, calculates the hash value (G_1) H ($Auth_1$, OTP, RV_{N1}) , computes (CX) xor(C_1,G_1) and generates (G_2) H ($G_1C_1TS_{IG1}$, RV_{IG1}, IG_{IDR});
8. The IG sends $IG_{IDA,}$ S_{ID}, TS_{IG1}, CX, {RV_{IG1}, $G_{2\}R1}$ message encrypted by R_1 to N;
9. **else**

   Go to step 24;

end if

10. **else**

    Go to step 24;

11. **end if**
12. N verifies the time stamp, calculates G1' to retrieves C_1 from CX. N applies the PUF function to calculate R1 for C1. N uses R1 to decrypt the received message and retrieves RV_{IG1} and G_2. N generates G2' H (G_1, C_1, TS_{IG1}, RV_{IG1}, IG_{IDR});
13. **If** (the calculated hash message in step 12 matches the hash message that was sent in step 9) **then**

    The authenticity of the IG is verified

14. N generates a timestamp TS_{N2}, a random nonce RV_{N2}, a private key($PRIV_N$) and calculates its corresponding public key(Pub_N) using the ECC algorithm.
15. N computes new OTP(OTPnew) = H (OTP|| C_{NI}|| TS_{N2})
16. N computes a new alias ID(N_{IDAnew}) = H (N_{IDA} || RV_{IG1})
17. N computes N_3 = H (TS_{N2}||RV_{N2}||OTPnew || N_{IDAnew} ||RV_{IG1})
18. N sends $N_{IDA,}$ S_{ID}, TS_{N2}, {RV_{N2}, Pub_N, $N_{3\}R1}$ message encrypted by R_1 to IG;
19. **else**

    Go to step 24.

end if

20. The IG verifies the TSN2, decrypts the message using R1 and generates (N_3') H (TS_{N2}'||RV_{N2}'||*ssk*'|| OTPnew' || N_{IDAnew}' ||RV_{IG1})
21. **If** (the calculated hash message in step 20 matches the hash message that was sent in step 18 **then**

The authenticity of the IoT device is verified.

22. Mutual authentication between the IoT device and the server is established.
23. **else**

Go to step 24.

End if

24. **Stop** (terminates the connection).

End

Figure 36: Algorithm 1 The mutual authentication between IoT device and the IG

4.5 Evaluation Process

The following section presents a detailed security and performance analysis of the proposed protocol and compares the analysis results with other related schemes presented in the literature.

4.5.1 Security Validation of the Proposed Protocol

In this section, we conducted both formal and informal security analysis. The formal evaluation uses two different approaches. The first formal evaluation approach is simulation-based by using the AVISPA tool to ensure that the proposed scheme is secure against active and passive attacks such as replay attacks and man-in-the-middle attacks. The second evaluation approach theorem proving-based uses Burrows, Abadi, and Needham (BAN) belief logic of authentication to confirm that the authenticated participants share the session key and the generated one-time password (OTP) securely in the proposed scheme. The BAN logic can be used as an illustrative method of the essential concepts of an authentication protocol and as a basic verification tool for a security protocol. After completing the formal evaluation, we conduct an informal

security evaluation by examining the proposed protocol against the well-known attacks and prove that the proposed protocol satisfies the main security properties.

4.5.1.1 Formal Security Evaluation

4.5.1.1.1 Simulation-based Security Verification Using AVISPA

The proposed protocol is simulated and tested using the AVISPA software, a widely accepted tool for automatically validating the protocols' security features. The messages involved in the cryptography and security of the authentication process are taken into consideration.

4.5.1.1.1.1 Simulation Overview

The abstract notations used to describe the authentication protocol and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 3. Also, table 4 shows the symbols that are used in AVISPA HLPSL scripting.

Table 3: Abstract Notation and AVISPA HLPSL Scripting Variables/Functions for Protocol Specification

Notation	**Description**
AE	AE
N	N
IG	G
S_{ID}	SIDNIG
N_{IDR}	NIDR

N_{IDA}	NIDA
IG_{IDR}	IGIDR
IG_{IDA}	IGIDA
$Auth_1$, $Auth_2$,$Auth_3$	T,X,Y
TS_{IG}	TSoneIG,TstwoIG
TS_N	TsoneN,TstwoN
RV_{N1}	Na
RV_{N2}	Naa
RV_{IG1}	Nb
T_{IDN}	TIDN
VD_{IDN}	VDIDN
R_1, R_2	Rone,Rtwo
R_{IG}	RG
C_1, C_2	Cone,Ctwo
CX	CX
N_1, N_2, N_3	None,Ntwo,Nthree
G_1, G_2	Gone,Gtwo

Pub_N	Exp (GX,Pn)
Pub_{IG}	IG_pub
N_{IDAnew}	NnewIDA
OTP_{New}	NOTP
$\oplus$	XOR
H	H

Table 4: Symbols for Protocol Specification

Symbol	**Meaning**
‘	The variable is locally computed by the agent.
/\	Logical AND
{}	Encryption

The main goals of the simulation are as follows:

1. Goal 1: The secrecy_of secNIDR represents that the NIDR is permanently kept secret, known to only (N and IG).
2. Goal 2: The secrecy_of secOTP represents that OTP is kept secret to (N, IG) only.
3. Goal 3: The secrecy_of secNa represents that Na is kept secret to (N, IG) only.

4. Goal 4: The secrecy_of secNb represents that Nb is kept secret to (N, IG) only.
5. Goal 5: The secrecy_of secIGIDR represents that the IGIDR is permanently kept secret, known to only (N and IG).
6. Goal 6: The secrecy_of secRone represents that Rone is kept secret to (N, IG) only.
7. Goal 7: The secrecy_of secCone represents that Cone is kept secret to (N, IG) only.
8. Goal 8: The secrecy_of secNaa represents that the secret key Naa is permanently kept secret, known to only (N and IG).
9. Goal 9: The secrecy_of secNOPT represents that NOPT is permanently kept secret, known to only (N and IG).
10. Goal 10: The secrecy_of secRG represents that RG is permanently kept secret, known to only (N and IG).
11. Authentication Property 1: The authentication_on Nb represents that IG generates Nb. If N securely receives Nb through a message, it authenticates IG.
12. Authentication Property 2: The authentication_on Na represents that N generates Na. If IG securely receives Na through a message, it authenticates N.

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with

their message exchanges. There are four defined roles: that are: (1) role_N that is played by the IoT node; (2) role _IG that is played by the Intelligent gateway; (3) session, where the session role and all its declarations are defined, and (4) environment, which instantiates all agents, variables, and functions. When the agent locally computes the variable, it is marked as primed (’). Also, the symbol /\ presents the conjunction (logical AND). Notice that { }_denotes the encryption where the encryption key is identified following a _ .

Figure A_1 in Appendix A shows the role of the Node N played by N. N is aware of the N and IG agents in the protocol, and its alias identity and the IG alias identity. Also, it is aware of its own real identity and the IG real identity that should be kept secured. Furthermore, N is aware of the authentication parameters T, X, Y, the OTP generated by the AE, the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model.

At the first state, “state 2,” N receives a start message “Rcv(start)” as a signal to begin the protocol run. All local variables are declared under the local section. N generates new random values (TSoneN and Na) and computes None, Ntwo, and SN. The computation process of the None, Ntwo, and SN follows the presented protocol description. N sends (NIDA, TSoneN, Ntwo, SN) to the IG.

At the second transition, “State 4”, N receives the (IGIDA, SIDNIG, TSoneIG, {CX, Gtwo} R_1) message from the IG encrypted by R_1. The computation TG, Gone, Cone, and Gtwo follows the description of our protocol. At this transition, if the Gtwo appears as expected by N, N authenticates the IG. After authenticating the IG, N generates TStwoN and Naa. Also, N computes NOTP, NnewIDA and Nthree. The

computation process of Nthree follows the presented protocol description. Based on the negotiation agreement regarding the algorithm that will be used to generate the session key, N will generate its public key and send it to the IG as part of the message. The “end role” at the end of the N role denotes the end of the role Node played by N.

Figure A_2 in Appendix A shows the role of the Intelligent Gateway played by IG. The IG is aware of all agents in the protocol N and IG, its alias and real identities (IGIDA, IGIDR), the alias and real identities (NIDA, NIDR) of the N node. IG knows its public key (IG_pub), the OTP, the virtual domain of the IoT device VDIDN, the type of the IoT device TIDN, the challenge (CG), the hash function H (·), and the send/receive channels Snd/RcvAll local variables are defined under the local section.

At the first transition, “State 1”, IG receives the (NIDA, TSoneN, Ntwo, SNG) message which is sent by N. IG uses the N’ alias ID to retrieve its NIDR, VDIDN, TIDN, OTP, and CG. Once the IG verifies SNG, it will generate a fresh value Nb and a TSoneIG. IG will compute TG, Gone, CX, and Gtwo according to the description of the introduced scheme. Then the IG will send to N (IGIDA, SIDNID, TSoneIG, CX, {Nb, Gtwo} $_{R1}$) encrypted by R_1. At the second transition, IG receives (SIDNIG.TStwoN' {Naa'.exp(GX,Pn').Nthree'}$_{R1}$. The IG decrypts the message using R_1. Then it will compute NnewIDA and NOPT and verify Nthree in accordance with the description of the presented protocol. The “end role” at the end of the IG role denotes the end of the role Node played by IG.

Figure A_3 in Appendix A shows the session role where the two agents’ roles are invoked, and all the session parameters are defined. First, all known constant parameters and their declarations are presented. The predefined constants are T, X, Y, NIDA, NIDR,

IGIDA, IGIDR, GX, CG, OTP, VDIDN, and TIDN. IG_pub is defined as a public key, and H is defined as the hash function. In the composition section, each agent's role is invoked along with its constant values and functions. For example, The G role is invoked with N and G as agents, NIDA, NIDR, IGIDA, IGIDR, VDIDN, TIDN, GX, CG, and OTP and as predefined constants, IG_pub as a public key, H as the hash function, and SND2 and RCV2 as the send and receive channels for G. Similarly, N role is invoked in the same manner. The “end role” at the end of the session role indicates the end of this role.

Figure A_4 in Appendix A shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants n, and g are instantiated as agents representing agents N, and G. The ig_pub instantiates the public key IG_pub. The constants t,x,y, nida, nidr , igida , igidr, vdidn, tidn,otp, gx and cg instantiates T, X, Y, NIDA, NIDR, IGIDA, IGIDR, VDIDN, TIDN, OTP, GX and CG, respectively. The function h instantiates the hash function H. The protocol identifiers secNIDR,secOTP, secNa,secIGIDR, secNb, secRone, secCone ,secNaa,secNOTP,secRG, na, and nb are also instantiated and defined. In the intruder knowledge section, all relevant values that the intruder is assumed to know before the execution are provided. The attacker is assumed to know n and g. He/ she is also assumed to know ig_pub, gx, and the hash h. The session is instantiated with n, g, n,g,t,x,y, nida, igida, nidr, igidr, gx, otp, cg,ig_pub, and h instances in the composition section. The “end role” at the end of the environment role denotes the end of this role.

Figure A_5 in Appendix A shows the simulation goals which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator

is dictated to check the secrecy NIDR, IGIDR, T, X,Na,Nb,Rone,Cone, Naa, RG and NOTP at different states using ‘secrecy_of secNIDR’, ‘secrecy_of secOTP’, ‘secrecy_of secNa’, ‘secrecy_of secNb, ‘secrecy_of secIGIDR’, ‘secrecy_of secRone’, ‘secrecy_of secCone’, ‘secrecy_of secNaa’, ‘secrecy_of secNOTP’, and ‘secrecy_of secRG’. The authentication is checked using ‘authentication_on na and ‘authentication_on nb’. The “end role” at the end of the goal section denotes the end of this role.

4.5.1.1.1.2 Simulation results

The simulation results of the proposed scheme are based on OFMC and CL-AtSe, which are considered two widely accepted backend model checkers to check and verify that the protocol meets the specified security goals [71]. These backends help in the automatic execution analysis of the security protocols. OFMC and CL-AtSe were used because they support the bitwise XOR operation. OFMC and CL-Atse are multipurpose automatic model analyzers for testing how efficient the security of the cryptographic protocols. OFMC and CL-Atse receive an input a protocol described by a set of rules generated by the AVISPA compiler and decide If an attack exists based on the Dolev-Yao intruder threat model [88]. A security protocol animator (SPAN) is used to build a Message Sequence Chart (MSC) of the protocol execution from the outlined HLPSL specification. Moreover, SPAN automatically creates attacks on HLPSL specifications using the well-known “Dolev-Yao” intruder model. The SPAN protocol simulation’s MSC corresponding to our HLPSL specification is shown in Figure 37.

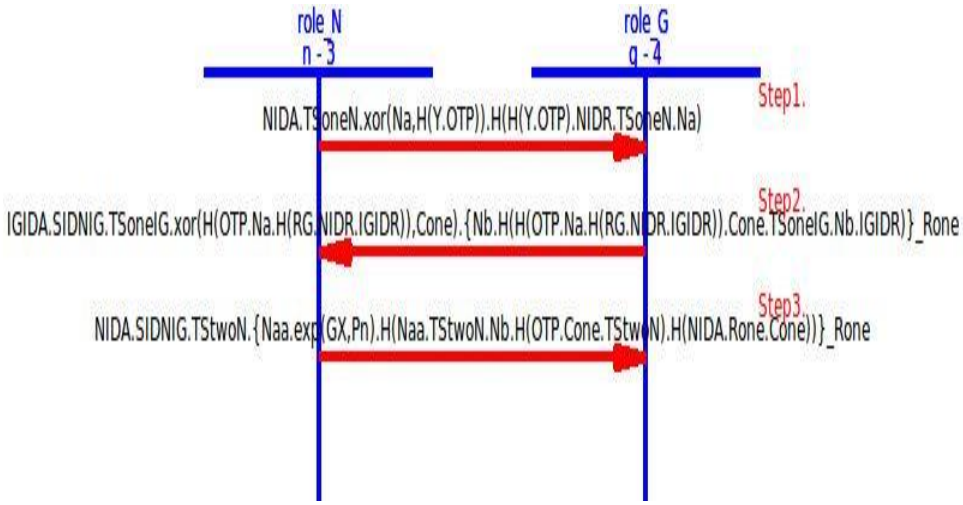


Figure 37: Snapshot of the protocol simulation in AVISPA

In AVISPA tool, the security properties such as authentication, integrity, and secrecy are specified in a separate section. Therefore, when SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN generates an attack trace if an attack is found, and it will consider the protocol unsafe. The presented protocol's simulation results are achieved by the OFMC back-end checker and the CL-AtSe back-end checker. Figure 38 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure 39 presents the OFMC back-end checker report shows that the protocol is SAFE, thus meeting the defined security goals. In summary, we can conclude that the proposed protocol is secure.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/N_G_Diss.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS
 Analysed   : 1 states
 Reachable  : 1 states
 Translation: 0.06 seconds
 Computation: 0.00 seconds
```

Figure 38: CL-AtSe Summary Report

```
% OFMC
% Version of 2006/02/13
SUMMARY
SAFE
DETAILS
BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
/home/span/span/testsuite/results/N_G_Diss.if
GOAL
 as_specified
BACKEND
OFMC
COMMENTS
STATISTICS
parseTime: 0.00s
searchTime: 0.15s
visitedNodes: 5 nodes
depth: 2 plies
```

Figure 39: OFMC Summary Report

4.5.1.1.2 Formal Proof Based on BAN Logic

This section uses BAN logic to verify the legitimacy of the shared secret session key (*ssk*) and the new OTP (OTPnew) shared between N and IG. To ensure the security of the proposed protocol under BAN logic, it needs to satisfy a set of security goals. The following section defines the main goals of the analysis of the presented authentication scheme.

4.5.1.1.2.1 Goals Identification

Goal 1: The N and the IG want to establish a secret shared session key (*ssk*) that is believed by both, respectively.

G 1_1: N believes that the IG believes that the *ssk* is a securely shared parameter between N and IG.

$$N| \equiv IG \mid \equiv (N \overset{ssk}{\longleftrightarrow} IG)$$

G1_2: N believes that *ssk* is a securely shared parameter between N and IG.

$$N| (N \overset{ssk}{\longleftrightarrow} IG)$$

G1_3: IG believes that the N believes that the *ssk* is a securely shared parameter between N and IG.

$$IG| \equiv N \mid \equiv (N \overset{ssk}{\longleftrightarrow} IG)$$

G1_4: IG believes that *ssk* is a securely shared parameter between N and IG.

$$IG| \equiv (N \overset{ssk}{\longleftrightarrow} IG)$$

Goal 2: OTPnew is well protected and believed by N.

G2_1: N believes that the IG believes that the OTPnew is a securely shared parameter between N and IG.

$$N| \equiv IG \mid \equiv (N \overset{OTPnew}{\longleftrightarrow} IG)$$

G2_2: N believes that OTPnew is a securely shared parameter between N and IG.

$$N| \equiv (N \overset{OTPnew}{\longleftrightarrow} IG)$$

G2_3: IG believes that the N believes that the OTPnew is a securely shared parameter between N and IG.

$$IG| \equiv N \mid \equiv (N \overset{OTPnew}{\longleftrightarrow} IG)$$

G2_4: IG believes that OTPnew is a securely shared parameter between N and IG.

$$IG| \equiv (N \overset{OTPnew}{\longleftrightarrow} IG)$$

4.5.1.1.2.2 Messages Idealization

First, we transfer all transmitted messages into idealized form as follows:

M1: $N \longrightarrow IG: <N_{IDR}, NX_1, TS_{N1}, RV_{N1}>_{N \overset{OTP}{\longleftrightarrow} IG}$

M2: $IG \longrightarrow N: \{IG_{IDR}, C_1, C_2\ G_1, RV_{IG1}, TS_{IG1}\}_{R1}$

M3: $N \longrightarrow IG: \{N_{IDR}, TS_{N2}, RV_{N2}, G^{\alpha}\}_{R1}$

4.5.1.1.2.3 Main Assumptions

The second step to be completed is to define some assumptions as initiative promises. The fundamental assumptions of the presented authentication scheme are as follows:

P1: IG believes that OTP is a securely shared parameter between N and IG

$$IG| \equiv (N \xleftrightarrow{OTP} IG)$$

P2: IG believes N believes that OTP is a securely shared parameter between N and IG

$$IG \mid \equiv N| \equiv (N \xleftrightarrow{OTP} IG)$$

P3: N believes that OTP is a secure shared parameter between N and IG

$$N| \equiv (N \xleftrightarrow{OTP} IG)$$

P4: N believes IG believes that OTP is a securely shared parameter between N and IG

$$N \mid \equiv IG| \equiv (N \xleftrightarrow{OTP} IG)$$

P5: IG believes that R_1 is a securely shared parameter between N and IG

$$IG| \equiv (N \xleftrightarrow{R_1} IG)$$

P6: IG believes N believes that R_1 is a securely shared parameter between N and IG

$$IG \mid \equiv N| \equiv (N \xleftrightarrow{R_1} IG)$$

P7: N believes that R_1 is a securely shared parameter between N and IG

$$N| \equiv (N \xleftrightarrow{R_1} IG)$$

P8: N believes IG believes that R_1 is a securely shared parameter between N and IG

$$N \mid \equiv IG \mid \equiv (N \overset{R_1}{\longleftrightarrow} IG)$$

P9: IG believes TS_{N1} is fresh.

$$IG \mid \equiv \#(TS_{N1})$$

P10: IG believes TS_{N2} is fresh.

$$IG \mid \equiv \#(TS_{N2})$$

P11: IG believes RV_{N1} is fresh.

$$IG \mid \equiv \#(RV_{N1})$$

P12: IG believes that N believes RV_{N1}.

$$IG \mid \equiv N \mid \equiv \ RV_{N1}$$

P13: IG believes that N has jurisdiction over RV_{N1}. That is, N is an authority and believes RV_{N1}.

$$IG \mid \equiv N \ \Rightarrow RV_{N1}$$

P14: IG believes RV_{N2} is fresh.

$$IG \mid \equiv \#(RV_{N2})$$

P15: IG believes that N believes RV_{N2}.

$$IG \mid \equiv N \mid \equiv \ RV_{N2}$$

P16: IG believes that N has jurisdiction over RV_{N2}. That is, N is an authority and believes RV_{N2}.

$$IG \mid \equiv N \Rightarrow RV_{N2}$$

P17: N believes TS_{IG1} is fresh.

$$N \mid \equiv \#(TS_{IG1})$$

P18: N believes RV_{IG1} is fresh.

$$N \mid \equiv \#(RV_{IG1})$$

P19: N believes that IG believes RV_{IG1}.

$$N \mid \equiv IG \mid \equiv RV_{IG1}$$

P20: N believes that N has jurisdiction over RV_{IG1}. That is, IG is an authority and believes RV_{IG1}.

$$N \mid \equiv IG \Rightarrow RV_{IG1}$$

P21: IG believes N_{IDR} is a securely shared parameter between N and IG.

$$IG \mid \equiv (N \overset{N_{IDR}}{\longleftrightarrow} IG)$$

P22: IG believes N believes that N_{IDR} is a securely shared parameter between N and IG.

$$IG \mid \equiv N \mid \equiv (N \overset{N_{IDR}}{\longleftrightarrow} IG)$$

P23: N believes IG_{IDR} is a securely shared parameter between N and IG.

$$N \mid \equiv (N \overset{IG_{IDR}}{\longleftrightarrow} IG)$$

P24: N believes IG believes that IG_{IDR} is a securely shared parameter between N and IG.

$$N \mid \equiv IG \mid \equiv (N \overset{IG_{IDR}}{\longleftrightarrow} IG)$$

P 25: IG believes α is fresh

$$IG \equiv \#(\alpha)$$

P26: IG believes that N believes α.

$$IG \mid \equiv N \mid \equiv \alpha$$

P 27: IG believes that N has jurisdiction over α. That is, N is an authority and believes α.

$$IG \mid \equiv N \Rightarrow \alpha$$

P28: N believes that IG believes C_1

$$N \mid \equiv IG \mid \equiv C_1$$

P29: N believes that IG has jurisdiction over C_1

$$N \mid \equiv IG \Rightarrow C_1$$

P30: IG believes N believes that *ssk* is a securely shared parameter between N and IG

$$IG \mid \equiv N \mid \equiv (N \overset{ssk}{\longleftrightarrow} IG)$$

P31: N believes IG believes that *ssk* is a securely shared parameter between N and IG

ssk

$$N \mid \equiv IG \mid \equiv (N \overset{OTP}{\longleftrightarrow} IG)$$

P32: N believes that IG believes OPTnew

$$N \mid \equiv IG \mid \equiv OTPnew$$

P33: IG believes that N believes OPTnew

$$IG \mid \equiv N \mid \equiv OTPnew$$

4.5.1.1.2.4 Analysis of the Authentication Protocol

We then prove that the proposed protocol achieves the security goals based on the idealized form of the messages, assumptions, and BAN logic rules. The proposed authentication scheme analysis is shown below to prove that the protocol achieves mutual authentication between N and IG.

According to M_1:

$$V_1\text{: IG} \triangleleft :<N_{IDR},NX_1,TS_{N1},RV_{N1}> \overset{OTP}{\longleftrightarrow}$$

According to P1, P21, and Rule 1, we derive the following

$$\text{V2: } \frac{IG \mid \equiv N \overset{OTP}{\longleftrightarrow} IG, IG \triangleleft (N_{IRD},NX_1,TS_{N1},RV_{N1}) \; N \overset{OTP}{\longleftrightarrow} IG}{IG \mid \equiv N \mid \sim (N_{IRD},NX_1,TS_{N1},RV_{N1})}$$

According to P9, P11, and rule 3, we derive the following

$$\text{V3: } \frac{IG \mid \equiv \#(TS_{N1} \; and \; RV_{N1})}{IG \mid \equiv \#(N_{IRD},NX_1,TS_{N1},RV_{N1}) \mid}$$

According to P2, P22, V2, V3, and rule 2, we derive the following

V4:

$$\frac{IG|\equiv\#(\mathrm{N}_{IRD},\mathrm{NX}_1,\mathrm{TS}_{N1},\mathrm{RV}_{N1}),\mathrm{IG}|\equiv\mathrm{N}\sim(\mathrm{N}_{IRD},\mathrm{NX}_1,\mathrm{TS}_{N1},\mathrm{RV}_{N1})}{IG|\equiv\mathrm{N}|\equiv(\mathrm{N}_{IRD},\mathrm{NX}_1,\mathrm{TS}_{N1},\mathrm{RV}_{N1})}$$

According to V4, P12, P13, and rule 4, we derive the following:

$$\text{V5: } \frac{\mathrm{IG}\,|\equiv\mathrm{N}|\Rightarrow,\mathrm{RV}_{N1}\,,\mathrm{IG}\,|\equiv\,\mathrm{N}\,|\equiv\,,\mathrm{RV}_{N1}}{\mathrm{IG}|\equiv\,,\mathrm{RV}_{N1}}$$

According to M_2:

$$\text{V}_6\text{: N} \triangleleft \{\mathrm{IG}_{\mathrm{IDR}},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{\mathrm{IG1}},\mathrm{TS}_{\mathrm{IG1}}\}_{\mathrm{R1}}$$

According to P7, P23, and Rule 1, we derive the following

V7:

$$\frac{N|\equiv\mathrm{IG}\overset{\mathrm{R1}}{\leftrightarrow}\mathrm{N},\mathrm{IG}\triangleleft(\mathrm{IG}_{IDR},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{IG1},\mathrm{TS}_{IG1})\;N\overset{R1}{\leftrightarrow}IG}{N|\equiv\mathrm{IG}\,|\sim\,(\,\mathrm{IG}_{IDR},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{IG1},\mathrm{TS}_{IG1})}$$

According to P17, P18, and rule 3, we derive the following

$$\text{V8: } \frac{N|\equiv\#(\mathrm{RV}_{IG1},\mathrm{TS}_{IG1})}{N|\equiv\#(\mathrm{IG}_{IDR},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{IG1},\mathrm{TS}_{IG1})|}$$

According to P8, P24, V7, V8, and rule 2, we derive the following

V9:

$$\frac{N|\equiv\#(\mathrm{IG}_{IDR},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{IG1},\mathrm{TS}_{IG1}),\mathrm{N}|\equiv\mathrm{IG}\sim(\mathrm{IG}_{IDR},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{IG1},\mathrm{TS}_{IG1}\,)}{N|\equiv\mathrm{IG}|\equiv(\mathrm{IG}_{IDR},\mathrm{C}_1,\mathrm{G}_1,\mathrm{RV}_{IG1},\mathrm{TS}_{IG1})}$$

According to P19, P20, P28, P29, V9, and rule 4, we derive the following:

$$\text{V10: } \frac{N \mid\equiv \text{IG}\mid\Rightarrow \text{RV}_{IG1},\text{C}_1,\ \ \text{N} \mid\equiv \text{IG} \mid\equiv \text{RV}_{IG1},\text{C}_1,\text{C}_2,}{\text{N}\mid\equiv\ \text{RV}_{IG1},\text{C}_1,\text{C}_2,}$$

According to M_3, we get

$$\text{V11: IG} \triangleleft \{\text{N}_{\text{IDR}}, \text{TS}_{\text{N2}}, \text{RV}_{\text{N2}}, \text{G}^{\alpha}\}_{\text{R1}}$$

According to P5, P21, and Rule 1, we derive the following

V12:

$$\frac{\text{IG}\mid\equiv \text{N} \overset{\text{R1}}{\longleftrightarrow} \text{IG},\text{IG} \lhd (N_{IDR},\text{TS}_{N2},\text{RV}_{N2},\text{G}\alpha) \overset{N,R1}{\longleftrightarrow} IG}{IG\mid\equiv \text{N} \mid\sim (N_{IDR},\text{TS}_{N2},\text{RV}_{N2},\text{G}\alpha)}$$

According to P10, P14, P25, and rule 3, we derive the following

$$\text{V13: } \frac{IG\mid\equiv \#(\text{TS}_{N2},\text{RV}_{N2},\alpha)}{IG\mid\equiv \#(N_{IDR},\text{TS}_{N2},\text{RV}_{N2},\text{G}\alpha)}$$

According to P6, P22, V12, V13, and rule 2, we derive the following

V14:

$$\frac{IG\mid\equiv \#(N_{IDR},\text{TS}_{N2},\text{RV}_{N2},\text{G}\alpha),\text{IG}\mid\equiv \text{N}\sim(\text{I}N_{IDR},\text{TS}_{N2},\text{RV}_{N2},\text{G}\alpha)}{IG\mid\equiv \text{N}\mid\equiv (N_{IDR},\text{TS}_{N2},\text{RV}_{N2},\text{G}\alpha)}$$

According to P15, P16, P26, P27, V14, and rule 4, we derive the following:

$$\text{V15: } \frac{IG \mid\equiv \text{N}\mid \Rightarrow \text{RV}_{N2},\alpha,\text{IG} \mid\equiv \text{N} \mid\equiv \text{RV}_{N2},\text{R}_2,\alpha}{N\mid\equiv\ \text{RV}_{N2},\alpha}$$

If both N and the IG agreed on generating the *ssk* using the hash function:

By combining V9 and Rule 2, we get

$$V16:\ N \mid \equiv IG \mid (N \overset{ssk}{\longleftrightarrow} IG) \qquad (Goal\ 1\_1)$$

From assumption P31, V10, V16, and Rule 4, we get:

$$V17:\ N \mid \equiv (N \overset{ssk}{\longleftrightarrow} IG) \qquad (Goal\ 1\_2)$$

By combining V4, V14, and Rule 2, we get

$$V18:\ IG \mid \equiv N \mid \equiv (N \overset{ssk}{\longleftrightarrow} IG) \qquad (Goal\ 1\_3)$$

From assumption P30, V5, V15, V18, and Rule 4, we get:

$$V19:\ IG \mid \equiv (N \overset{ssk}{\longleftrightarrow} IG) \qquad (Goal\ 1\_4)$$

By combining V9 and Rule 2, we can derive

$$V20:\ N \mid \equiv IG \mid \equiv (N \overset{OTPnew}{\longleftrightarrow} IG) \qquad (Goal\ 2\ \_1)$$

From assumptions P4, P32, V10, V20, and Rule 4, we derive

$$V21:\ N \mid \equiv (N \overset{OTPnew}{\longleftrightarrow} IG) \qquad (Goal\ 2\_2)$$

By combining V14 and Rule 2, we can derive

$$V22:\ IG \mid \equiv N \mid \equiv (N \overset{OTPnew}{\longleftrightarrow} IG) \qquad (Goal\ 2\ \_3)$$

From assumptions P2, P33, V15, V22, and Rule 4, we derive

$$V23:\ IG \mid \equiv (N \overset{OTPnew}{\longleftrightarrow} IG) \qquad (Goal\ 2\_4)$$

Thus, by achieving all the goals mentioned above, we demonstrate the validity of the proposed scheme. we formally prove our protocol correctness as a secure mechanism to achieve mutual authentication and key agreement between the IoT node and the IG.

4.5.1.2 Informal Security Analysis

This section examines the security of the proposed protocol against well-known security attacks and satisfies the main security properties. The security of the protocol is explored against various known attacks. This section will provide brief descriptions of different types of attacks against IoT systems. A description of how the proposed protocol successfully resisted the well-known attacks and achieved the proposed security properties is presented.

4.5.1.2.1 Security attack Scenarios

Table 5 presents a short definition of the well-known threats that an attacker can launch on an IoT system and the main security properties that can ensure the systems' security

Table 5: A Summary of the Well-known Attacks

Attack name	**Attack definition**
Man-in-the-middle	An attack where an adversary can intercept the traffic between two communication parties and possibly modifies the communicated information.
Forward/forward secrecy	This is a security feature to ensure that even if the most recent key is hacked, a minimal amount of sensitive data is exposed, and no future keys will not be affected
Anonymity and Unlinkability properties	Anonymity: is a feature to ensure the privacy of the device

	Unlinkability: is a property that does not enable an adversary to determine whether two events occurring in the system are related or not.
A brute-force attack	It is an attack where the adversary can try every possible combination of letters, numbers, and symbols to guess a secret key or a password.
Replay attack	A network attack where legitimate data transmission is maliciously repeated or delayed to disrupt communications.
Eavesdropping attack	A type of passive attack where the adversary sniffs sensitive data from insecure communication networks.
Impersonating/spoofing attacks	An attack where an adversary pretends to be a trusted node in an IoT system.
Modeling attack	An attack that employs machine learning prediction algorithms to predict unknown CRPs from the known ones.
Physical attack	A hardware attack where an adversary physically accesses the hardware's semiconductor to read the stored secrets on it or discover its structure [89]
IoT device counterfeiting	Imitating real IoT devices with the intent to steal, destroy, or replace the original devices.

4.5.1.2.2 Security Properties Assessment

The following section presents a detailed security analysis of the security properties of the proposed protocol. It demonstrates how the proposed protocol satisfies

the security requirements for mutual authentication, session key agreement and resists various kinds of known attacks. Then, a comparison with other related protocols is presented.

4.5.1.2.2.1 Mutual Authentication

Before sending the first message, N chooses a random number (RV_{N1}) and computes NX_1, NX_2, and SN. For the IG to retrieve RVN1 and compute $Auth_2$ and $Auth_3$, the IG needs to compute $Auth_1$. The computation of $Auth_1$ requires that the IG passes the stored PUF challenge to the PUF function to compute the PUF response (R_{IG}). An adversary can't generate $Auth_1 = H ((N_{IDR} || IG_{IDR} || R_{IG})$ without knowing the N_{IDR} of N and the IG_{IDR} of the IG and the PUF response (R_{IG}). Also, the adversary cannot verify $Auth_2 = H (Auth_1 ||IG_{IDR}|| VD_{IDN})$ and $Auth_3 = H (Auth_2|| IG_{IDR}||T_{IDN})$ without having the VD_{IDN} and T_{IDN} that are securely stored and known only by the IG.

N and IG authenticate each other by verifying the correctness of $G_2 = H (IG_{IDR}||C_1' ||RV_{IG1} || TS_{IG1}|| G_1')$ and $N_3 = H (R_2 || RV_{N2}|| TS_{N2} || RV_{IG1}')$. The adversary cannot generate G_2 without having $G_1 = H (Auth_1|| OTP|| RV_{N1})$. Due to the PUF nature that is resilient to prediction and replication, the adversary cannot guess or generate R_{IG} to construct $Auth_1$ that is an element of G_1. Also, the adversary does not know the OTP that is considered a secured shared parameter between N and the IG. Additionally, the adversary cannot generate N_3 without having R_2, which can be generated only by N. In summary, the adversary cannot generate either G_2 or N_3 without knowing R_{IG}. As a result, the proposed scheme can achieve mutual authentication between N and IG.

4.5.1.2.2.2 Session key agreement and forward/backward security

According to the proposed scheme, N and the IG can agree on generating the session key in one of two different ways. The selection of the key generation technique will depend on the IoT node's capabilities and what it can support. During the session key agreement phase, both N and the IG locally generate the session key *ssk* in one of the following two ways.

By using the first technique, the session key $ssk = (G^{\alpha})^{\beta}$ is established between N and the IG using the ECDH algorithm to protect future communication. The secrecy of the *ssk* depends on the secrecy of β and α. The D-H key agreement scheme is based on ECDH; therefore, the disclosed session key will not cause the compromise of any future session key. The forward/backward security property's objective is to ensure that any past or future session keys will not be affected when any session key *ssk* is exposed. Even if an adversary obtains *ssk* of a session, he/she cannot compute any of the past and future session keys by using the disclosed *ssk* because the *ssk* is protected by the randomization of the β and α and the discrete logarithm problem of the Elliptic curve algorithm. As a result, the proposed scheme achieves session key security.

By using the second technique, the session key $ssk = h(R_1||C_1||RV_{N1}||RV_{N2}||RVI_{G1})$ is established between N and the IG to protect future communication. In the proposed scheme, *ssk* is generated locally. The secrecy of the *ssk* depends on the secrecy of RV_{N1}, RV_{N2}, RVI_{G1}, C_1 and R_1. Because RV_{N1}, RV_{N2} and RVI_{G1} are randomly selected for every new authentication session and R_1 cannot be replicated or predicted because it is unique for each N due to the nature of the PUF, the disclosed *ssk* will not cause the compromise of any future *ssk*. The forward/backward security property's objective is to ensure that any past or future shared secret keys will not be affected when any *ssk* is

exposed. Even if an adversary obtains *ssk* of a session, he/she cannot compute any of the past and future shared secret keys by using the disclosed *ssk* because the *ssk* is protected by the randomization of R_1|| C_1|| RV_{N1}|| RV_{N2} ||RVI_{G1}. As a result, the proposed scheme achieves the security of *ssk*.

4.5.1.2.2.3 Anonymity Unlinkability, and Untraceability Properties

For fully protected N privacy, strong anonymity with unlinkability is required. In the proposed protocol, the N's real identity N_{IDR} is not transmitted during all phases in clear text format. Therefore, even if the adversary eavesdrops on all communication messages, it is impossible to obtain the IoT node's real ID. In addition, the new alias ID of the IoT node N_{IDA}New = H (N_{IDA} || R_1||C_1). Because C_1 and R_1 are fresh in each session, the attacker cannot link any two different N_{IDA}'s to the same N and cannot trace a given IoT device to N's messages. By using a new N_{IDA} for each authentication session, the adversary will not be able to decide whether these authentication messages are from the same N or not. This means that device N cannot be linked to different sessions. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the devices by intercepting messages.

4.5.1.2.2.4 Stolen database attack

On the IoT node side, if any IoT node is compromised and the adversary is able to steal $Auth_1$, $Auth_2$, $Auth_3$, N_{IDR}, and the OTP from the IoT node's database to impersonate the IoT node, the fake node will fail the CRP verification process. On the IG side, if the IG is compromised and the adversary is able to steal NIDR, IG_{IDR}, VD_{IDN}, T_{IDN}, and OTP from the IG database to impersonate the IG, the fake IG will fail to calculate the R_{IG}.

4.5.1.2.2.5 Brute Force Attack

The first optional technique to generate the session key *ssk* is done locally by both the IoT node and the IG using the ECDH algorithm. Due to the discrete logarithm problem of the ECDH, guessing the session key is a time-consuming and challenging task. The probability of guessing is so negligible that the attacker will fail to guess the correct session key, given that this session key changes in every session.

The optional technique to generate the session key *ssk* is generated locally by both the IoT node and the IG using RV_{N2} and the generated nonce values from the two sides. All the *ssk* components are transferred in an encrypted format either by using a bitwise XOR function or by using R_1. Also, the session key's secrecy depends on the security of the communication between the IoT node and the IG and the response's uniqueness. Because no other IoT node can duplicate the responses, the adversary cannot obtain it from the protocol. The probability of guessing the session key is so negligible that the adversary will fail to guess the correct parameters given that this session key changes in every session.

4.5.1.2.2.6 Replay attack

The proposed protocol overcame the replay attack by using timestamps. The timestamp TS is generated by the sender node and then inserted in a hash function to ensure the adversary cannot replace it. Furthermore, Each CRP is used only one time to provide security against replay attacks. Hence, the protocol protects against replay attacks.

4.5.1.2.2.7 Eavesdropping attack

During the authentication phase, the adversary can intercept messages transmitted between N and the IG. All the intercepted messages will be useless because they are sent in an encrypted format using a one-way hash function and symmetric encryption. For the attacker to decrypt the messages, he/she needs to know R_1 that can only be generated by N. The only two parameters that are sent in clear text are the Alias identities and session ID. The Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The adversary will not be able to link the message to a particular device because the proposed protocol uses alias identities that vary in every session. Therefore, the proposed protocol protects against eavesdropping attacks.

4.5.1.2.2.8 Impersonation attack

In this attack, when the intruder eavesdrops on the messages transmitted from N to IG or vice versa, he/she can use the intercepted information for malicious actions, such as impersonating the IoT device or the IG and sending fabricated messages.

4.5.1.2.2.8.1 IG impersonation

This attack will not succeed for several reasons. First, each IG has an Alias ID and Real ID. The real ID is known only for IoT devices. The real ID of the IG will never be released in clear text format. Also, for the IG to compute $Auth_1$, it needs to compute the PUF response (RIG), which is one of the Auth1 parameters. Due to the PUF nature, R_{IG} cannot be predicted or replicated by another device. The IoT device will verify the

IG's real ID and G_2= H (IG_{IDR}||C_1 || RV_{IG1} || TS_{IG1}|| G_1) before proceeding with the authentication process.

4.5.1.2.2.8.2 IoT Node Impersonation

To impersonate the IoT node, an adversary should intercept the messages exchanged in the previous sessions. However, in the proposed scheme, the adversary cannot produce valid messages without having R_1 that is used to encrypt the second, third, and fourth authentication messages. Also, this attack will fail for numerous reasons. First, the alias ID of the IoT node changes every authentication session. Second, the fake IoT node will not be able to generate legitimate responses because the PUF responses are unique for each device.

From another perspective, even if the adversary succeeded in compromising an IoT node, the adversary's further attacks using the compromised node only affect the communication related to that node. Because each IoT node has its own secret keys, the adversary cannot derive other non-compromised IoT nodes' keys without knowing those nodes' random information. Therefore, further attacks will not affect other communications. As a result, the proposed scheme is resistant to IoT node impersonation attacks.

4.5.1.2.2.9 Data Modification attack

As discussed in the replay attack, eavesdropping, and impersonation attacks, the data modification attack is defeated because the real identity of IoT node NIDR, the timestamp TS, the symmetric key, the one-way hash function, the nonce values RV_{N1}, $RV_{N2,}$ and $RV_{IG1,}$ and the CRPs are unknown to the attacker. Therefore, the man-in-the-middle attack is prevented. The IoT device and the IG do not exchange sensitive data

such as real identities, nonce values, PUF responses R in plaintext over the insecure communication channel. This supports data confidentiality. The only exchanged data in plaintext during authentication are the alias identities, the timestamps (TS_{N1}, TS_{N2}, TS_{IG1}), and hashed messages. In such a setting, a data modification attack will not benefit from any captured data. Furthermore, if an adversary modifies the transferred data, the intended receiver will detect the data modification by matching the hashed messages and drop the connection.

4.5.1.2.2.10 Modeling Attacks

This is an attack where the adversary collects a large number PUF CRPs and uses a regression algorithm to build a model and predict responses for new challenges. All the PUF challenges and the responses are transferred in an encrypted format. Therefore, the adversary will not be able to collect enough useful CRPs to build a model and predict responses to new challenges.

4.5.1.2.2.11 Physical Attack

Any attempt to tamper with the IoT device will change the PUF embedded chip's behavior and, consequently, renders the PUF useless.

4.5.1.2.2.12 IoT Device Counterfeit/Cloning

Both the IG and the IoT device are authenticated using the PUF. PUF responses are treated as hardware fingerprints that cannot be duplicated or cloned. Therefore, by the use of the PUF, the IoT devices and the IG cannot be cloned.

4.5.1.2.3 Comparison of Security Features

We compare the proposed protocol's security features with other related authentication and key agreement schemes [57,59,60,61,62,63]. Table 6 shows the comparison results. The table indicates that the proposed protocol is secure against all the imperative security threats and accomplishes diverse security features.

Two of the proposed protocols [57,59] require secret key storage, which may compromise the secret key. Consequently, the adversary can exploit the compromised key to conduct different types of attacks and possibly reconstruct the secret key. Also, many of the proposed schemes, such as [57] and [60-63], used the IoT node's real identity to exchange messages with the server, which is against protecting the anonymity and unlinkability of the IoT devices. Furthermore, several proposed schemes such as [61], [60],[59], and [62] did not protect their schemes against replay attack.

Table 6: Security feature comparison of the proposed scheme with other related Mutual authentication and key agreement schemes.

Security Property	**[57]**	**[61]**	**[60]**	**[59]**	**[62]**	**[63]**	**Proposed Protocol**
Resilience to IG the Impersonation Attack	No	No	No	No	No	Yes	Yes
Resilience to IoT the Impersonation Attack	Yes	Yes	Yes	No	Yes	Yes	Yes
Resilience to Replay Attack	Yes	No	No	No	No	Yes	Yes
Resilience to Device Counterfeit	Yes	No	Yes	Yes	Yes	Yes	Yes

Resilience to Modeling Attack	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Resilience to the Data Modification Attack	Yes	Yes	Yes	No	Yes	Yes	Yes
Anonymity and Untraceability	No	Yes	No	No	No	No	Yes
Requires Key Storage	yes	Yes	No	No	No	No	No
Resilience to brute force attack	Yes	No	No	Yes	Yes	Yes	Yes
Resilience to Physical attacks	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Two-factor authentication	No	Yes	No	No	No	No	Yes

On the other hand, the proposed protocol supports multiple essential security features: First, the proposed protocol does not require any secret key storage or secret key sharing over the network. Second, each IoT device maintains two authentication factors by using OTP and PUF to prove its legitimacy to the IG. Third, all secret keys are computed locally on both the IoT node and the IG. Fourth, the proposed protocol does not store any CRPs on the IG, but they are frequently retrieved from the PUF cloud. Fifth, the IoT devices use a one-time alias identity for each authentication session in the proposed protocol. Therefore, it will be difficult for an outside adversary to comprehend

the activities of the IoT devices. Sixth, the proposed protocol overcomes the noise issue in PUF operation by using error-free CRPs [[37] and [90]. From table 6, we can conclude that the proposed protocol can support all the desired security properties, which are essential for IoT devices' security.

4.5.2 Performance Analysis

In this section, we present a performance analysis of the proposed protocol. The performance analysis evaluates the storage requirements. Also, we analyze the preset protocol's overhead and efficiency in terms of computational complexity and communication overhead, and storage constraints. We then compare our mutual authentication mechanism to the most relevant protocols in the literature proposed by [57], [59], [60],[61], [62], and [63].

4.5.2.1 Storage requirements

Table 7: Storage Cost of the Proposed Scheme

Node	**Storage cost (in bits)**
N	128 + 256 * 3 + 8* 3 + 256 = 1024 + 64 = **1176 bits**
IG	128 + 8 * 5 +256 + 128 = **552 bits**

In the proposed protocol, each IoT node must store its real identity N_{IDR} alias identity N_{IDA}, the real identity of the IG_{IDR}, alias identity IG_{IDA}, authentication parameters $Auth_1$, $Auth_2$ and $Auth_3$, one-time password OTP. We use SHA-256 as an example of hash function. By applying these settings, we obtain $|N_{IDA}| = 128$ bits, $|Auth_1| = |Auth_2| = |Auth_3| = 256$ bits, while $|N_{IDR}| = |IG_{IDR}| = |IG_{IDA}| = 8$ bits and $|OTP| = 256$ bits. On the other hand, IG is required to store the tuple N_{IDA}, N_{IDR}, IG_{IDA}, IG_{IDR}, VD_{IDN}, T_{IDN}, OTP and C_{IG}. By applying these settings, we obtain $|N_{IDA}| = 128$ bits, while $|N_{IDR}| = |IG_{IDA}| = |IG_{IDR}| = |VD_{IDN}| = |T_{IDN}| = 8$ bits $|OTP| = 256$ bits and the $C_{IG} = 128$ bits. Table 7 summarizes the storge cost of the proposed scheme.

4.5.2.2 Computational Complexity Analysis

We compare the computational complexity of the proposed protocol with other related protocols [57,59–63]. We only focus on comparing the authentication phase because the key generation phase is executed differently from one protocol to another. Because the time for executing a bitwise XOR operation is negligible, we do not consider XOR operations for computational cost analysis. To analyze the performance of the proposed protocol with respect to other presented protocols in the literature, we conduct simulations of the cryptographic operations using Dell Inspiron Laptop with Intel Core i7, dual-core 2.7 GHz CPU, and 8 GB RAM to act as the IG. We use a Raspberry Pi 4 Model B with 64-bit quad-core cortex A-72 processor and 1 GB RAM to simulate an IoT device. The simulations used PyCryptodome cryptographic and Fastecdsa libraries in Python. For these results, we considered the 128-bit arbiter PUF for PUF operation. The fuzzy extraction's execution time is almost the same as the ECC point multiplication and the execution time for modular exponentiation is double the time of the execution of the

ECC point multiplication [58]. Table 8 presents the used notation and the execution time of each operation.

Table 8: Execution Time of the Cryptographic Operations

Operation	Computation Time on IoT	Computation Time on IG
H: time for executing a one-way hash function	0.002 ms	.001 ms
FE: time for executing a fuzzy extractor	5 ms	4 ms
EM: time for executing an ECC point multiplication	5 ms	4 ms
EXP: time for a modular exponentiation	10	8
HMAC: time for executing the HMAC	2.7 ms	1.5 ms
MAC	2.9 ms	1.23 ms
ENC: Time for Executing (AES 128 Encryption)	0.18 ms	.14 ms
DEC: Time for executing (AES-CBC Decryption)	0.18 ms	.14 ms

PUF: Time for executing PUF (128-bit Arbiter)	.12 ms	.12 ms

Table 9 and figures 40, 41, and 42 summarize the computational complexity comparison between the proposed protocol and the other related protocols presented in the literature. Table 9 lists the number of cryptographic operations of each type for each protocol. Figure 40 displays the number of cryptographic operations of each type that are completed on the IoT side. Figure 41 depicts the number of cryptographic operations of each type that are completed on the IG side. Figure 42 demonstrates the total number of cryptographic operations of each type that are complete for each protocol. As depicted from the table and the figures, the proposed protocol used more hash functions than most of the other schemes and used AES 128 symmetric encryption to avoid computationally expensive cryptographic operations such as MAC and HMAC while achieving the same security goals. On the IoT side, all the protocols used the same number of PUF. Our proposed protocol is the only one that used PUF on the IG side to protect the protocol from IG impersonation

Table 9: Comparison of computational costs for the authentication phase of the proposed scheme and other related protocols.

Entity	**[57]**	**[59]**	**[60]**	**[61]**	**[62]**	**[63]**	**Proposed Scheme Using a hash function**	**Proposed Scheme Using ECC**
IoT	$8T_H+$ $2T_{PUF}$	$3T_H+$ $2T_{PUF}+$ $2T_{EM}$	$5T_H+$ $2T_{PUF}+$ $3T_{EM}+$ $1T_{ENC}$	$3T_H+$ $2T_{PUF}+$ $1T_{FE\ Gen}$	$2T_H+$ $3TMAC+$ $1T_{ENC+}$ $1T_{DEC+}$ $2T_{PUF}+$	$2T_{PUF}+$ 3_{HMAC}	$7T_H$ $+1T_{PUF}+1$ T_{ENC+} $1T_{DEC}$	$7T_{H+}$ $1T_{PUF}+1$ T_{ENC+} $1T_{DEC+}$ $1T_{EM}$

IoT Cost	**.26 ms**	**11.20 ms**	**16.39 ms**	**5.25ms**	**9.3ms**	**8.34 ms**	**.49ms**	**5.49 ms**
IG	$9T_H+$ $2T_{EXP}$	$6T_H+$	$5T_H$ $+2T_{EM+}$ 1 DEC	$5T_H+$ $1T_{FE\ Rec}$	$2T_H+$ $3TMAC+$ $1T_{ENC+}$	3_{HMAC}	$10T_H$ $+1T_{PUF}+$ $1T_{ENC+}$ $1T_{DEC}$	$10T_H$ $+1T_{PUF}+1$ T_{ENC+} $1T_{DEC}$
IG Cost	**16.01 ms**	**.006 ms**	**8.16 ms**	**4.01ms**	**3.83ms**	**4.5ms**	**.41ms**	**.41ms**
Total Cost	**16.27 ms**	**11.21 ms**	**24.55 ms**	**9.26ms**	**13.13ms**	**12.84ms**	**.90ms**	**5.9 ms**

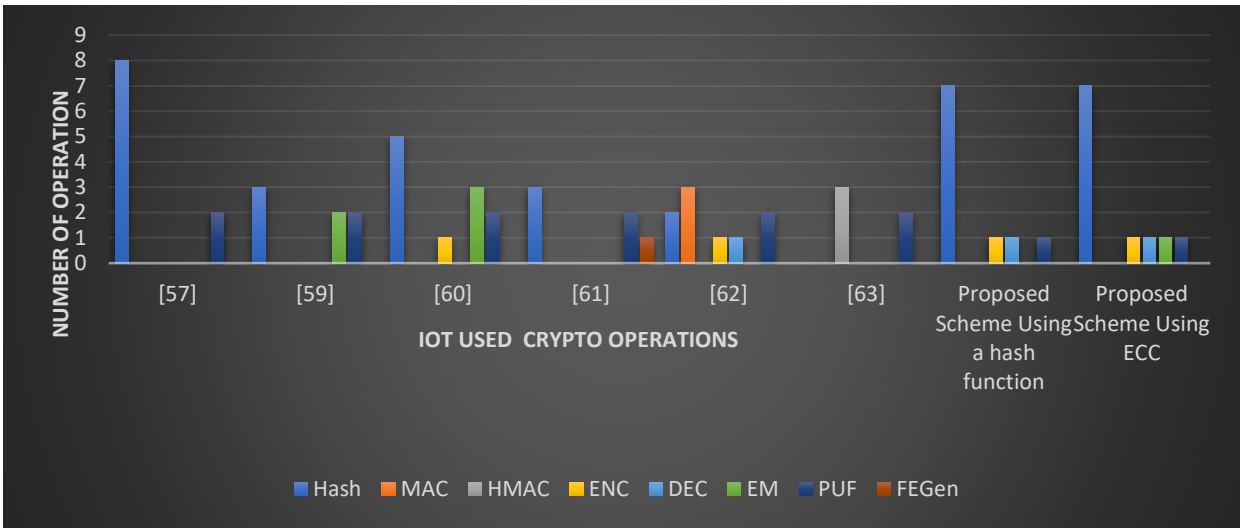


Figure 40: Comparison of the number of the IoT crypto operations across the protocols

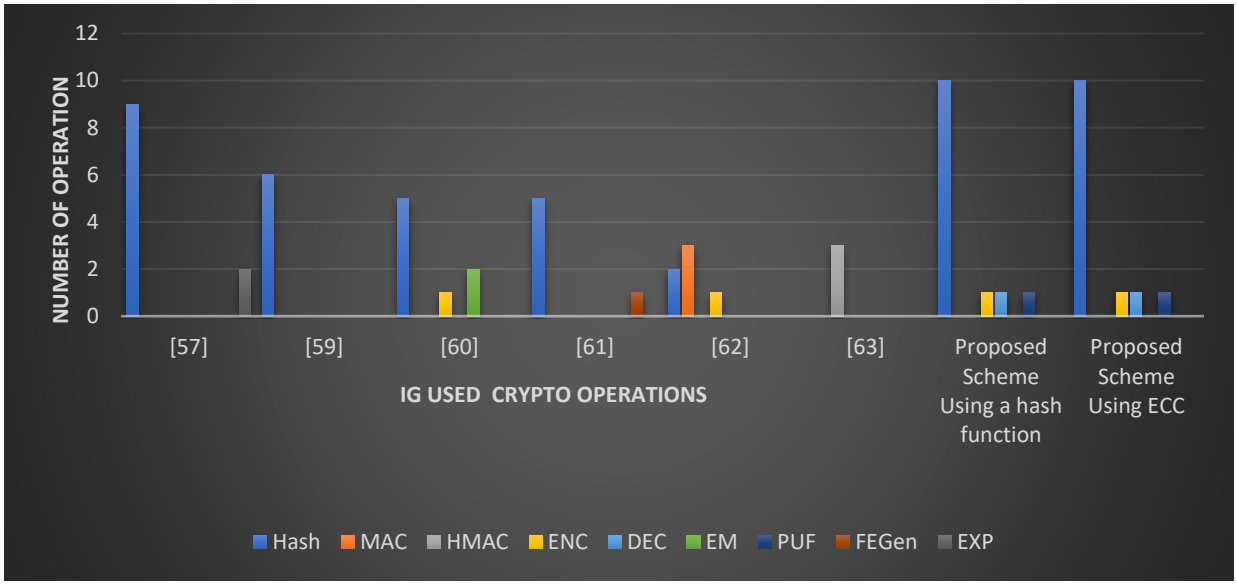


Figure 41: Comparison of the number of IG crypto operations across the protocols

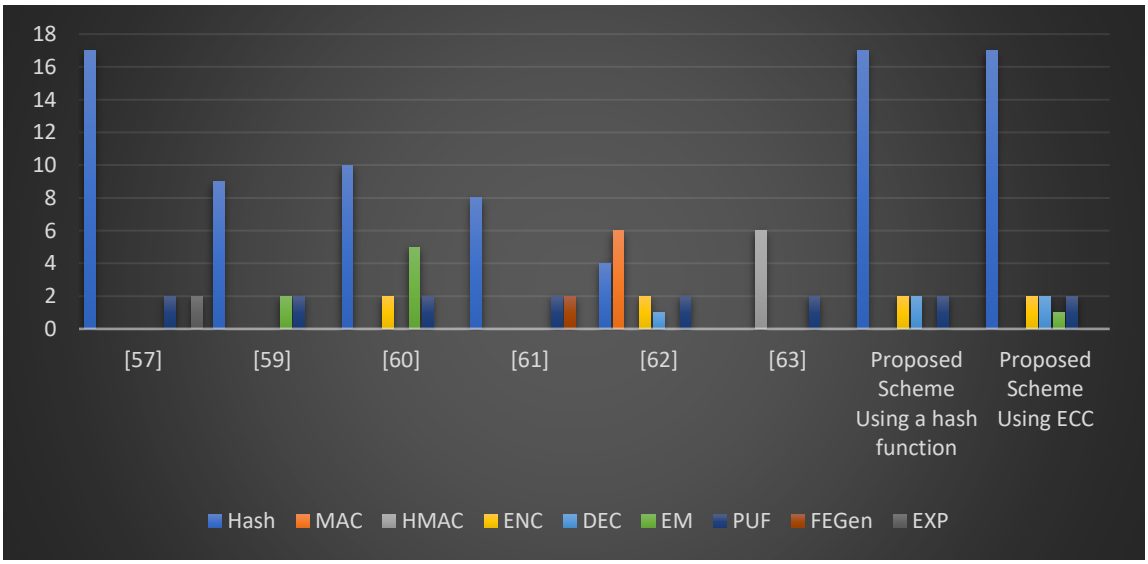


Figure 42: Comparison of the total number of crypto operations across the protocols

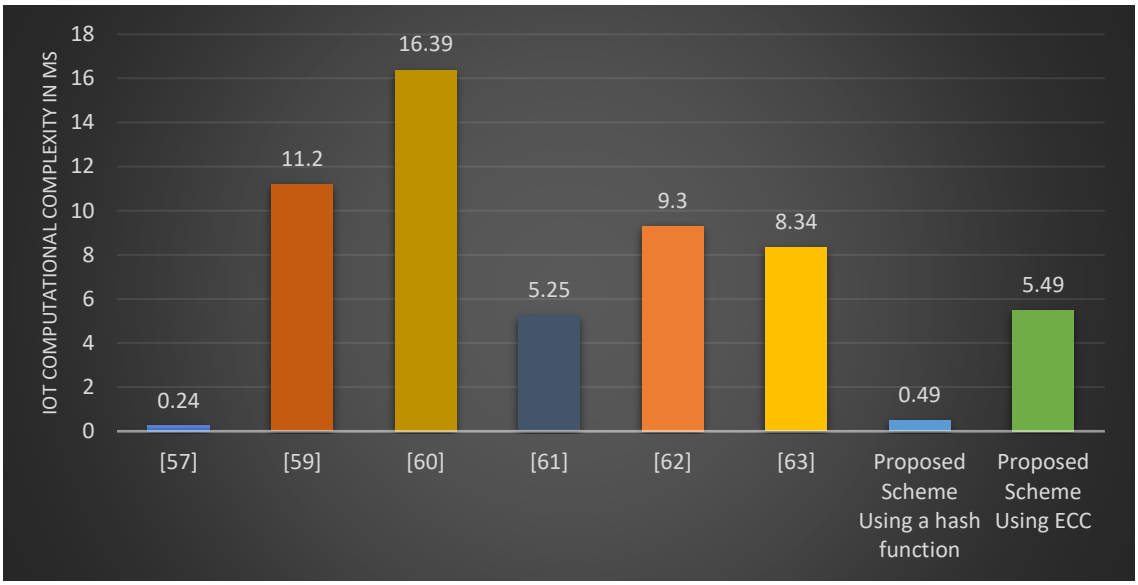


Figure 43: The IoT computational time across the different protocols

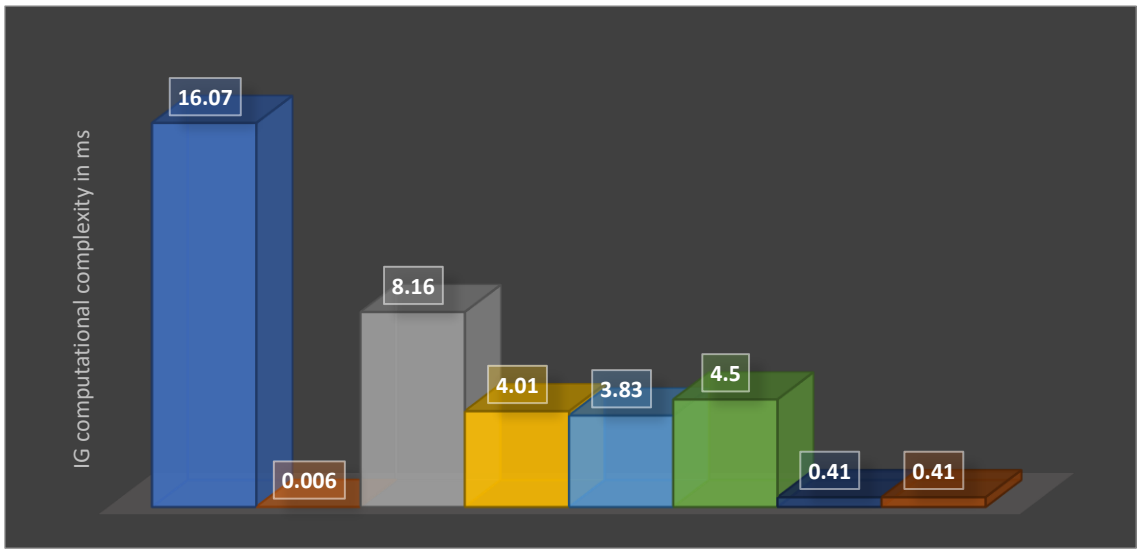


Figure 44: IG computational time across the different protocols

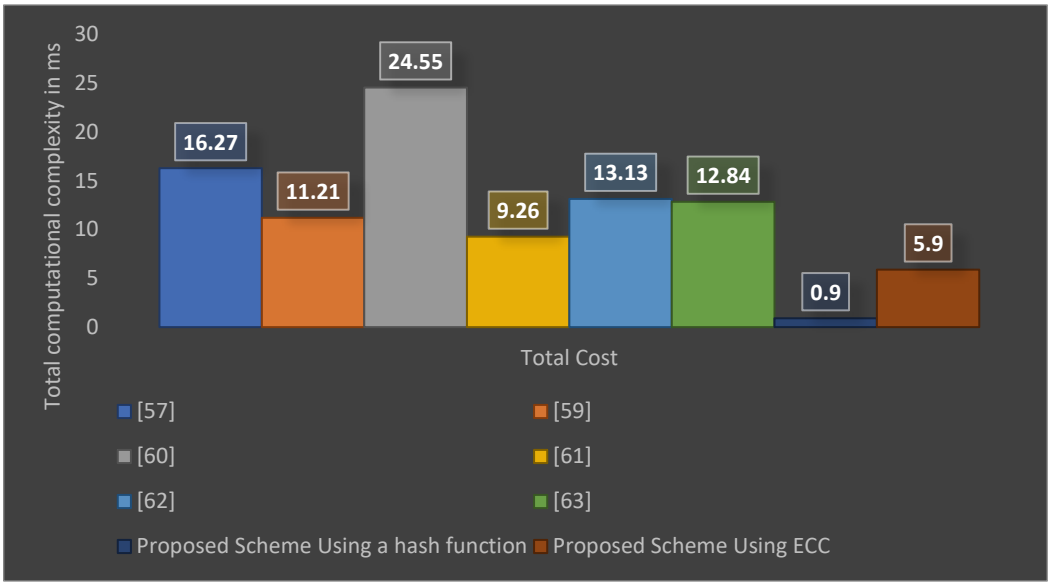


Figure 45: Total computational time across the different protocols

Table 9 and figures 43,44, and 45 demonstrate the cost comparison of the presented protocol with other similar protocols from the literature. Table 9 and figure 45 present the computational cost of each protocol. Figure 43 displays the computation cost of each protocol for the IoT side, and figure 44 depicts the computational cost of each scheme for the IG side. The results indicate that the proposed protocol has the lowest computational cost for the IoT side compared to [59,60,62, and 63]. These results prove that our proposed protocol is suitable for IoT resource-constrained devices. Although our proposed protocol has a higher computational cost on the IoT side than [57] and [61], the presented protocol ensured data confidentiality and user anonymity which are essential security goals.

From table 9 and figure 45, it is observed that while [57], [59],[60],[61],[62] and [63] take approximately 16.27, 11.21, 24.55, 9.26, 13.13, and 12.840, the proposed protocol takes only either .90 or 5.9. The protocol in [60] performs the worst due to the

higher number of ECC point operations involved. Also, the protocol in [57] has a bad performance due to the use of modular exponentiation. Furthermore, both protocols [62 and 63] have high computational cost due to using many MAC and HMAC functions that are computationally heavy. In summary, our proposed protocol is more efficient than other related protocols and has a higher security level than the rest of the protocols. The proposed protocol achieved the main security goals of confidentiality, integrity, and authenticity besides accommodating IoT devices that are diverse in their capabilities.

4.5.2.3 Communicational Cost

The following section analyzes the communication cost of the proposed protocol. To reduce network congestion and to provide fast message transmission, the communication costs of the protocol should be as low as possible. For the communication cost analysis, we evaluate the communication cost in terms of the size of the message in bits. Then, we compare the proposed protocol to the other related protocols. Table 10 presents a summary of the sizes of the message parameters. Table 11 lists the message parameters that are communicated between the server and the IoT device along with their sizes.

Table 10: Size of the Message Parameters

Message Parameters	**Size in Bits**
ID of N [[57], [62], [63]	128
ID of server [63]	8
S_{ID} [57], [62]	8

Nonces [61], [63]	128
CRP (C, R) [61], [62]	128
HMAC [63]	256
Hash Function [57], [62]	256
Timestamp (TS) [63]	48
ECC [60], [59]	256
MAC [62]	256

- **Message 1**: In the transmission (N → IG), N sends the tuple, N_{IDA}, TS_{N1}, NX_2, S_N Therefore, the size of this tuple is 128 + 48 + 128+256 = 560. bits.
- **Message 2**: In the transmission (IG → N), IG sends the tuple, $IG_{IDA,}$ S_{ID}, TS_{IG1}, CX {RV_{IG1}, $G_{2\}\ R1.}$ Therefore, the size of this tuple is 8 +8 + 48 + 128+128+256 = 576 bits.
- **Message 3_1**: In case the two parties decided to use the hash function to generate the session key. In the transmission (N → IG), N sends the tuple, N_{IDA}, S_{ID}. TS_{N2}{RV_{N2}, N_3, SHA256} $_{R1)}$ Therefore, the size of this tuple is 128 + 8 + 48+128 +256 = 568. bits.
- **Message 3_2**: In case the two parties decided to use ECC to generate the session key. In the transmission (N → IG), N sends the tuple, N_{IDA}, S_{ID}. $TS_{N2,}${RV_{N2}, N_3, G^{α} }$_{R1)}$ Therefore, the size of this tuple is 128 + 8 + 48+128 +256 = 824 bits.

Table 11: Communication Cost Comparison (bits)

Message Number	[57]	[59]	[60]	[61]	[62]	[63]	Proposed scheme_ hash	Proposed scheme_ ECC
M_1: N→ S	256 bits	128 bits	176 bits	256 bits	256 bits	432 bits	560 bits	560 bits
M_2: S→ N	512 bits	304 bits	640 bits	512 bits	768 bits	432 bits	576 bits	576 bits
M_3: N→S	688 bits	1024 bits	512 bits	640 bits	768 bits	304 bits	568 bits	824 bits
M_4: S→ N	384 bits	1024 bits	1024 bits	----	----	480 bits	---	---
M_5: N→S	640	------		----	---			
Total	**2480 bits**	**2480 bits**	**2352 bits**	**1408 bits**	**1792 bits**	**1648 bits**	**1704 Bits**	**1960 bits**

The total communication costs of [57], [59], [60], [61], [62], and [63] are 2480,2480,2352,1408,1792, and 1648 bits, respectively. If the proposed protocol uses a hash function to generate a session key, the total communication cost is 1704. Still, if the two communicating sides decide to use ECC to generate the session key, the total communication cost is 1960 bits. Compared to the other presented protocols that used ECC to generate the session key [59 and 60], our proposed protocol has a lower communication cost and, at the same time, achieved all the required security goals. While if the two communicating parties decided to use the hash function, the communication cost of my proposed protocol is less than [57 and 62], while it is higher than [61 and 63]. As given in table 6, protocols in [57], [61], [62], and [63] are incapable of achieving all the security goals.

4.6 Summary

This chapter presents a two-factor mutual authentication protocol for IoT systems between an IG and an IoT device. The proposed protocol introduces a mechanism for secure session key establishment. The protocol provides confidentiality, integrity, anonymity, unlinkability, and untraceability capabilities while achieving mutual authentication between the IoT and the IG devices. The proposed protocol is assessed using both security and performance analysis. The security analysis used both formal and informal techniques. The formal evaluation employed used the BAN logic and AVISPA simulation to verify the proposed protocol's security. The results showed that the proposed scheme is safe. The informal security analysis also showed that the presented protocol is resistant to common attacks and satisfies the main security properties. Furthermore, we evaluated the protocol's efficiency in terms of storage requirements, communication, and computational cost and compared it with other related protocols. The presented protocol achieves the required low computational complexity for resource-constrained IoT devices.

CHAPTER V

M2M DISTRIBUTED MULTI-LAYER LIGHTWEIGHT MUTUAL AUTHENTICATION AND KEY AGREEMENT SCHEME USING CHAINED HASH PUF IN INDUSTRIAL IOT SYSTEM

The emergence of Industry 4.0 and the Internet revolution changed the manufacturing atmosphere and created an integrated environment between the virtual world and the physical world. Although the Internet made the world smaller, there is still a gap between the physical and cyber worlds. With the rise of the IoT, all objects in the cyber world and the physical world have become interconnected. This integration can be viewed as a physical–digital-physical information chain that happens through three simple steps: physical to digital, digital to digital, and digital to physical. During the first step, data is collected from the surrounding environment through IoT sensors. In the second step, the collected data is processed and analyzed. The third stage consists of transmitting the decisions to the physical world through actuators.

With the emergence of Industrial IoT(IIoT), machine-to-machine (M2M) communication is essential to build IIoT environments that enable heterogeneous devices such as sensors, actuators, and gateways to communicate without any human intervention through a wireless or wired link. [91]. These IoT devices have a connection to the Internet either directly or through another device. Their connection to the Internet makes them

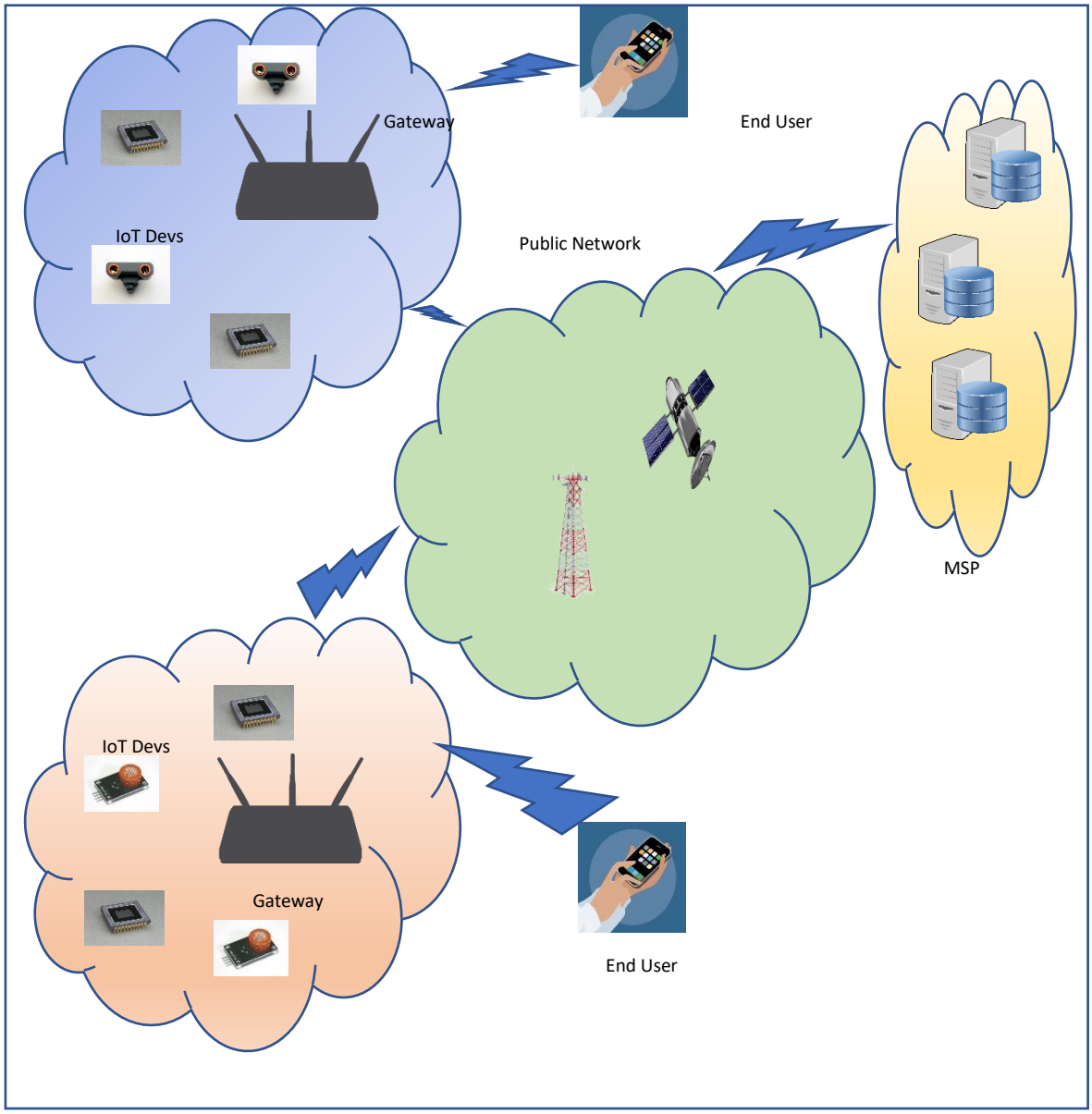


Figure 46: M2M Networks in the IoT Environment

available any time and from everywhere in the world to their users. M2M communication is a network where smart devices communicate through wireless and wired technologies. According to [92], the M2M communication system consists of three interconnected layers as presented in figure 46: (1) a M2M area layer that includes a M2M area network with M2M gateways; (2) a communication network layer that includes wired/wireless

networks; and (3) an application services layer consisting of the end-users and required applications. However, this connectivity makes them accessible to adversaries who can attack a device to access sensitive data or use actuators to perform harmful actions that can damage the system [93] and [94]. IIoT devices produce a massive amount of confidential and sensitive data. Several cryptographic mechanisms can be applied to protect the IoT devices from cyber-attacks, but they are not suitable for the resource constraint nature of the IoT devices. One way to solve this problem is to offload some security-related operations from resource-constrained IoT devices to more resource-rich IoT devices. In IIoT, devices' authentication is crucial to achieving a trusted communication among the communicating devices. However, most of the currently employed M2M authentication protocols in the IIoT domain are based on asymmetric cryptography with a high computational cost. Consequently, those authentication schemes are not the best suitable solution for resource-constrained IoT devices, leading to many security issues in the IIoT environment.

This chapter introduces lightweight privacy-preserving M2M mutual authentication and key agreement scheme. The presented protocol is called M2M Distributed Multi-Layer Lightweight Mutual Authentication and Key Agreement Scheme Using Chained Hash PUF in Industrial IoT System. The proposed scheme enables the devices to verify each others' identities and perform secure, anonymous authentication while the real identities stay secret. It will also allow the devices to locally generate shared secret keys to ensure the confidentiality and integrity of the exchanged data. The proposed scheme utilizes PUF as a hardware security approach and the chained hash concept. The scheme also employs an access control method and implements virtual

domain segregation to protect the IoT devices from any arbitrary access by other unauthorized devices. The proposed scheme employs lightweight operations of XOR and a one-way hash function; thus, the scheme does not have a high impact on the device's computational and battery resources. It only requires between 3 and 4 messages to be exchanged between the communicating parties for the mutual authentication and key agreement. We evaluate the proposed scheme by using different measures, namely, security and performance evaluation. The security evaluation will be both a formal and informal security evaluation. The formal analysis used the Burrows–Abadi–Needham logic (BAN) as a formal validation of the proposed authentication scheme and the automated validation of the internet security protocols and applications (AVISPA) toolkit.

Furthermore, the scheme efficiency is evaluated and compared with other related schemes. Through the informal analysis, we will assess the proposed scheme against well-known security attacks. Furthermore, we also validate the efficiency of the proposed scheme and compares its performance in terms of computational cost and communication overhead with other related schemes.

In Section 5.1, we first introduce the motivation of the work. Section 5.2 discusses the security and functional requirements for M2M communication. Section 5.3 presents the security vulnerabilities and potential threats in M2M communication. Section 5.4 presents the related work and the other existing protocols. Section 5.5 demonstrates the network model and the security goals. Section 5.6 presents the proposed protocol. Section 5.7 illustrates the protocol evaluation process, and finally, section 5.8 summarizes the chapter.

5.1 Motivation

The implementation of IIoT can lead to many benefits to the industry, such as monitoring and optimizing the supply chains. Also, the use of IIoT can lead to early detection of machine failure, which can prevent production delay, equipment damage, or injuries to workers [95]. IIoT showed a significant interest in implementing M2M communication in wireless networks. Some characteristics of M2M communication within the IoT ecosystem are as follows: [96]

1. **Heterogeneity of devices**: IoT devices are different due to their functionalities and applications.
2. **Device coexistence and collaboration**: the different IoT devices communicate with each other anytime.
3. **Diverse networks and networking standards**: Because the IoT ecosystem consists of heterogeneous devices, they can use various communication standards such as cellular systems, WiFi, Bluetooth, Zigbee, or others to facilitate communication between devices.
4. **Device limitations**: Most IoT devices are resource-constrained with limited capabilities in terms of battery life, memory, and processing power.
5. **Multihop communication**: Most IoT devices are equipped with limited transceiver systems. Therefore, they will only be capable of short-range transmissions and will need to route information over multiple hops.

The emergence of M2M significantly impacts peer-to-peer networks and is expected to influence the IoT. Most of the current IoT domains require higher data rates,

low latency, and better quality of service (QoS) from the wireless networks [91] and [97]. With the exponential increase in IoT devices, the implementation of M2M technology becomes necessary to provide better services. M2M communication can improve wireless networks' performance by eliminating the need for a central node, reducing the overall communication latency, and offloading central node traffic [98]. Furthermore, M2M communication provides autonomous and energy-efficient systems.

In M2M communication, IoT devices can take different forms of communication. The communication can be through a single hup or multiple hops. Also, communication can be intra-domain or inter-domain. Single-hop communication is where there is direct communication between source and destination without any other device involvement. On the other hand, multi-hop communication is where an intermediate device exists as a relay or an access point between the source and the destination devices. Intra-domain communication is where the interaction will be among the devices within the same network. In contrast, inter-domain is referring to communication among the devices of different networks.

The communication systems employ nonstandard hard-wired communication technologies among communication and industrial devices [99]. On the other hand, the evolution of the IIoT, the industrial environment saw changes where the heterogeneous communication technologies employ wireless standards to assure communication, compatibility, and remote operation and control of the different devices through the Internet. Given that IIoT sensors are designed to be resource-constrained, these sensors tend to have some security drawbacks.

To meet the industrial requirements, smarter sensors capable of more complex operations are developed. However, there are still several challenges facing IoT sensors. First, there is still a gap between the security requirements and the currently implemented security solutions. According to [100], few of the currently available solutions adopt the security by design approach, which provides a new attack surface for the attackers. The attacker can cause physical damage, threaten human life, manipulate the final product by compromising the production processes or increasing the used resources. Second, machines operating on a production floor typically last for several decades, and it is not always economically feasible to replace the old equipment with the latest technology. Therefore, finding solutions that can ensure security to modern manufacturing technology and the old systems is essential. Third, another major issue is the authenticity of the IIoT device and identifying the counterfeited devices.

One of the main challenges in M2M communication is security because of such systems' dynamic nature [101]. While designing M2M systems, we should keep in mind how to prevent threats and vulnerabilities rather than curing them. Unfortunately, most M2M research focuses on other topics such as interference management, route discovery, and resource optimization rather than security. Security also is an essential issue in wireless networks because of the exchanged information over the network that can be eavesdropped on, modified, or corrupted by an adversary. M2M communications face several security threats that can affect privacy, authentication, confidentiality, integrity, network availability, and QoS.

Most of the currently employed cryptographic primitives are not suitable for the resource-constrained nature of IIoT devices, which negatively impacts the ability to

conduct authentication, identification, integrity, and verification of those devices. Most of the proposed authentication protocols employ public-key cryptography [102], [103], [104], [105], [106]. The long key length and the computational complexity of such cryptosystems make them hard to be implemented in resource-constrained devices due to their limited memory and limited power supply [107], [108]. Because identification and authentication are the cornerstones of providing security, new lightweight authentication approaches and securing those resource-constrained devices are necessary.

5.2 Security and functional requirements for M2M communications

To ensure the effectiveness of the M2M communication, there are security and functional requirements that need to be satisfied.

5.2.1 Security requirements

The communication channels between two IoT nodes (M2M) or an IoT node and a gateway are subject to several attacks. Because most IoT domains require a high level of security, several security measures need to be addressed and satisfied. Table 12 presents the main security requirements in the M2M communication

Table 12: Main security requirements in the M2M communication

Security Requirement	Description
Confidentiality	This requirement prevents an adversary from disclosing transmitted data and ensures that only the authorized entities can read the exchanged data in a M2M communications system.

Integrity	This requirement ensures that the data has not been modified or altered by an adversary. Data integrity requirement is crucial in M2M communication systems because unauthorized data modification can lead to severe consequences.
Availability	The availability requirement ensures that the M2M application systems can access the service anytime and is always available.
Authentication	Authentication is one of the main requirements to ensure the security of the M2M communication. Through authentication, the device can verify each other's identity and authenticity before exchanging sensitive information.
Anonymity	It is essential to secure the device's anonymity and unlinkability to ensure data privacy when an attacker illegally exposes sensitive information.
Access control	It is vital to limit data access and communicate with other devices to the authorized devices.
Non-repudiation	This requirement ensures that no device can deny sending a message that it originated.
Data freshness	This feature implies that received messages are fresh and are not a replayed old messages sent by an adversary.

5.2.2 Functional requirements

Besides the security requirements, there are functional requirements that can ensure the quality and applicability of the authentication protocols. Table 13 presents the main functional requirements of M2M authentication protocols

Table 13: Main functional requirements of M2M authentication protocols

Feature	Description
Scalability	This feature ensures the ability of the system to add new nodes easily
Efficiency	This feature ensures communication efficiency by minimizing the number of the exchanged messages between the communicating parties.
Low computational cost	The M2M systems need to employ lightweight cryptographic primitives to support the resource-constrained nature of the IoT devices.
Storage needs	The authentication protocols must request minimum storage on the IoT nodes.
A distributed scheme	IoT nodes should communicate, if applicable, directly and not require a

	central server's support to execute the authentication phase.

5.3 Security vulnerabilities and potential threats in M2M communication

M2M networks consist of a heterogeneous mix of devices ranges from resource-constrained devices to resource-rich devices. Most of the resource-constraint devices are IoT nodes. The resource-rich devices are gateways in charge of managing the connection among the IoT devices on one side and between the internal and external networks on the other side. M2M networks consist of many IoT nodes and one or more gateways. IoT nodes are simple devices equipped with some specific sensing technology to collect real-time data and transmit the collected data to the gateway in a single-hop or multi-hop format. Once the gateways receive the IoT node data, they can either process the received data or send it to a back-end server for processing and storage [95],[98], [105], [106].

Table 14 presents a description of the main security threats in M2M networks, the potential consequences, and the possible countermeasure.

Table 14: Main security threats in M2M networks, the potential consequences, and the possible countermeasure

Security Threat	Potential Consequence	Countermeasure
Eavesdropping: is a type of attack where the adversary listens to the exchanged messages between the communicating parties to learn things about the network and try to find a way to be inside it.	1. They are exposing sensitive and private data.	1. Using secure communication links that implement modern cryptographic primitives. 2. Establish security association between the communicating parties after performing mutual authentication between them.
Hacking the long term key of the IoT node/gateway	1. device impersonation	1. Store the long-term keys in a Hardware Security Module (HSM) located inside the devices and is tamper-resistance. This makes it impossible for attackers to discover

		the values of long-term keys. 2. Limiting the lifetime of the session key.
Modifying/replacing long term keys in IoT/gateway	DoS attack	1. Store the long-term keys in a HSM located inside the devices and is tamper-resistance. 2. Allowing the modification of stored sensitive data and long-term key after completing strong mutual authentication.
Modifying messages between entities	Loss of message integrity	1. Use of modern cryptographic primitives to secure message's confidentiality and integrity 2. Limiting the lifetime of the session keys

		3. Establish a security association between the communicating entities.
Replaying sniffed message to entities	1. Unauthorized access to the system 2. Impersonating the IoT node/gateway	1. Use of a timestamp or sequence number to ensure message freshness
Physical attack where the attacker tries to take control of a device and then extract any information from it, instead of destroying it	1. Impersonating the legitimate devices 2. Gain root-level access on the device and compromise the device, which allows the attacker to move freely and attack other devices to bring the whole system down.	1. The use of hardware-based security provides robustness because it makes it hard for the attacker to alter or replicate the device's physical features.

5.4 Related Work

The following section introduces an analysis of some of the presented authentication protocols in the literature. The authors in [109] propose a user authentication and key agreement protocol for heterogeneous ad hoc wireless sensor networks for achieving energy efficiency, user anonymity, and mutual authentication. The researchers used hash functions and XOR operations to implement the authentication process. They present a heterogeneous environment with two or more types of nodes: resource-constrained sensor nodes and more powerful gateways. The researchers state that their proposed protocol ensures high security and performance features. However, the presented protocol has multiple weaknesses. First, both the user and the sensor are using fixed masked IDs, which can lead to a lack of user anonymity and linkability and traceability attacks. Second, the authentication process does not require any user input; therefore, stealing the smart card can lead to a user impersonation attack. Third, the proposed protocol does not support two-factor authentication. Fourth, suppose a sensor node became compromised; in that case, an attacker can monitor this sensor node and obtain the session key shared between another sensor node and the user who has ever connected to this compromised sensor node.

Authors in [110] present a modified version of the protocol presented by [109]. The researchers propose a remote user authentication and key agreement protocol to access IoT nodes that are part of the wireless sensor networks where the network consists of multiple gateway nodes. The proposed protocol has multiple phases: the system setup phase, user registration phase, login phase, authentication phase, password update phase, and dynamic node addition phase. The researchers assume that each sensor node is assigned to the nearest gateway based on minimum distance. The proposed protocol used

the one-way hash and the XOR as lightweight cryptographic functions. The proposed protocol used a long-time session key, which can make it subject to session key disclosure. Also, the protocol transmits the sensors' real identities in clear text, which does not support the anonymity and unlinkability security features.

Authors in [111] present a lightweight authentication protocol that relies on digital signatures. The proposed authentication protocol presents a lightweight signature solution using Hierarchical Identity Based Signature (HIBS). It consists of five phases; two setup phases, extraction phase, signing phase, and verification phase. The presented protocol uses an identity-based signature to generate signing and verification keys. The identity-based signature uses the user's identity to generate the signing and verification keys. The protocol consists of three layers. The first layer is called the Root Private Key Generator (Root PKG) and is located in the cloud. The second layer is called sub-PKGs, which act as agents and are located in the fog. The third layer is users' layer, which consists of end-users. Each sub-PKG is responsible for the public and private keys generation of its registered mobile users. The use of the sub PKG reduces the overheads on the Root PKG. The researchers compared their work to other related protocols in light of the key distribution method, key generation method, and security attack model. The results indicate that the presented protocol is resilient to multiple attacks such as linking signature, replay attack, key eavesdropping attack, and signature forgery attack.

The authors in [91] introduce a proxy-based key establishment protocol for IoT using the Diffie Hellman (D.H.) algorithm. The proposed protocol allows any two unknown, resource-constrained devices to initiate secure end-to-end (E2E) communication. The constrained devices should maintain secured connections with the

neighbor nodes that are less resource-constrained devices in the local networks in which they are deployed. The less constrained devices act as proxies and are performing expensive cryptographic operations on behalf of the resource-constrained node. The proposed protocol aims to allow two resource-constrained devices located into two different networks to generate a shared secret key. The proxies into the two different networks will support the resource-constrained devices in computing the shared secret key. The researchers assumed that the proxies of the two different networks could securely communicate with each other. The researchers also assumed that the resource-constrained IoT nodes have a list of the neighbor proxies and their corresponding pre-shared keys for authentication. Although the proposed protocol is proposed for resource-constrained devices, it requires the resource-constrained devices to store multiple pre-shared keys of the different proxies. Also, the protocol requires storing the pre-shared key and reusing them for each authentication session which may lead to a side-channel attack.

The authors in [93] introduce a lightweight authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks. The proposal relies on XOR and hash functions. The researcher put in their consideration five main factors: (1) ensuring user anonymity, (2) requiring no complex computations, (3) performing mutual authentication, (4) providing a user-friendly scheme, and (5) ensuring the correctness of the session key. The proposed protocol consists of six phases: predeployment phase, registration phase, login phase, authentication phase, password-change phase, and dynamic node addition phase. The proposed protocol requires lots of computations, and it is subject to node capture attacks.

The authors in [94] propose a group authentication and key agreement (GROUP-AKA) protocol designed for M2M communication. The authors successfully prevent the DoS attack, but the protocol does not handle the privacy preservation problem. Furthermore, the proposed protocol is computationally expensive because it employs a public cryptology approach.

In [97], the authors proposed a lightweight mutual authentication and key agreement protocol for M2M communications in IIoT environment between resource-constrained sensors and a router, including a trusted platform module (TPM). The protocol is based on hash and XOR operations. The authors assume that the proposed protocol is characterized by low computational cost, low communication cost, and low storage overhead. The authors in [98] demonstrate that the proposed protocol by [97] is vulnerable against DoS, and router impersonation attacks. Also, they prove that the adversary could trace the smart sensor by eavesdropping on only one session. Furthermore, the researchers present how an untrusted smart sensor can obtain the router's secret key and the session key that another sensor established with the router.

The authors in [100] present a smart card-based authentication scheme for heterogeneous ad hoc wireless sensor networks. They introduce two versions of the proposed protocol. The first version of the protocol is called P1. P1 has four phases: predeployment phase, registration phase, authentication phase, and password changing phase. The researchers claimed that P1 is lightweight but does not provide perfect forward secrecy. Therefore, they introduce the second version of the protocol called P2. The researchers mentioned that P2 could guarantee perfect forward secrecy. They explain that P2 is an advanced version of P1 and shares with P1 the same procedures in the

predeployment, registration, and password changing phases. P1 is lightweight because it mainly uses XOR and one-way hash function, but P2 uses public-key cryptography to make it more computationally intensive. The two versions of the proposed protocol do not achieve a proper mutual authentication as the user's identity is not verified by the sensor. Therefore, the protocol is executed even if the user's inputs are wrong. The proposed protocol does not support the unlinkability security feature where the sensor I.D. (SID) is static and sent over the network in clear text format. Finally, the validity of the user's identity is not checked by any party. Therefore, the authentication phase will still be executed even if the user inputs a wrong identity.

The authors in [102] introduce an authentication, authorization, and accounting (AAA) system for IIoT. The proposed protocol is based on the Next Generation Access Control (NGAC) standard that provides access control in different environments. The proposed system provides access control and security to heterogeneous IIoT devices in a local cloud. The proposed authentication mechanism employs X.509 certificates. The use of the digital certificate makes the proposed solution not suitable for resource-constrained devices. According to [103], using the X.509 certificate and the public-key certificate management causes performance bottlenecks and makes the proposed scheme not applicable to resource-constrained IoT devices.

Authors in [104] present a mutual authentication protocol between an IoT node and a fog node using the HMAC and XOR function. The two sides have a fixed shared secret key. The proposed protocols used challenge-response negotiation. The presented protocol is vulnerable to various types of security attacks and violates security requirements. First, the protocol exchanges messages between the IoT node and the fog

node using their real identities, which do not preserve the user's privacy. Consequently, the proposed protocol does not provide anonymity and unlinkability, and the adversary can trace the devices by intercepting messages. The first message that the fog node sent to the IoT node contains $K \oplus C_A \oplus C_B$ where the K is the fixed share secret key, $C_{.A.}$ is the nonce value that is generated by the IoT node and was sent to the fog node in clear text format. $C_{.B.}$ is the nonce value that was generated by the fog node. Once the IoT node receives the message from the fog node, it increments the timestamp by one and calculates a new HMAC using the K, the ID of the IoT node, and C_B. The adversary can intercept the messages and conduct a brute force attack by guessing the C_B and the K. Then, the adversary can verify his guess by recomputing the XOR values using the guessed $C_{B.}$ and K. Also, the adversary can intercept the last message that was sent from the IoT node to the fog node and recompute the HMAC by using the guessed K and C_B. If the recalculated HMAC value is equal to the intercepted HMAC value, this will ensure that the guessed K and $C_{B.}$ are correct. Then the adversary can use the guessed key to decrypt all subsequent messages.

The authors in [105] introduce an efficient authentication scheme for M2M networks in IoT-enabled cyber-physical systems. The researchers present four different protocols to achieve four different authentication tasks. The four main elements of the presented protocol are gateways, mobile users, IoT devices connected to the gateway, and IoT devices that are not connected to the gateways but connected to the IoT devices that are connected to the gateway. The authentication protocols are achieved used symmetric cryptography and a one-way hash function. The first protocol enables the mobile user to authenticate with any gateway mutually. The second protocol enables the mobile user to

mutually authenticate with any of the IoT nodes that are within the domain of this gateway the user is authenticated to in protocol 1. The third protocol allows the IoT nodes to mutually authenticate each other, given the condition that at least one of the IoT nodes is connected to the gateway. The fourth protocol allows IoT nodes that are not connected to the gateway to mutually authenticate each other with the IoT node connected to the gateway. Although the four protocols are lightweight, all of them require the storage of long-term shared secret keys, which can make the system vulnerable to several attacks, such as side-channel attacks, to obtain the secret key. Second, the different devices exchange messages using their real identities that make the protocols violate privacy preservation and do not satisfy the untracability and unlikability security features. Finally, the proposed protocols have unnecessary steps that can be avoided to reduce the computational complexity and the computational cost. For example, the mobile user can send a message directly to the gateway requesting to perform mutual authentication with one of the IoT devices instead of sending the message to the IoT node. Then the node forwards the message to the gateway.

The authors in [106] propose three new lightweight, efficient authentication protocols for IoT-based healthcare applications. The first protocol is an improvement of the protocol proposed by [105]. The second protocol is the M2C (machine to the cloud) mutual authentication protocol that is based on a one-way hash function. The third protocol is M2M mutual authentication protocol using Elliptic Curve Cryptography ECC. Although the researcher claimed that the M2M authentication protocol is efficient and suitable for resource-constrained devices, it does not fit the resource-constrained devices. The third protocol used digital signatures and ECC, which are computationally expensive

and consume a lot of power from IoT devices. Also, the exchanged messages did not include any IDs. Therefore, it is not clear how the recipient can identify the sender and verify each other's identities. The protocols cannot mitigate against a replay attack.

The authors in [107] propose an Identity-Based Cryptography (IBC) without Key-Escrow (AIBCwKE) authentication scheme. The scheme covers most known attacks; however, the scheme requires IoT devices and mobile devices to perform computationally expensive cryptographic operations. The IoT nodes perform multiple bilinear pairing operations on elliptic curves in addition to several point multiplications and exponentiations. These computational requirements are not suitable for resource-constrained devices.

Authors in [112] propose a new Token-Based Lightweight User Authentication (TBLUA) for IoT devices. The presented protocol consists of the following phases: (i) Offline smart device and gateway registration, (ii) User reservation, (iii) Token distribution between the gateway and smart devices, and (iv) Login and Authentication. The proposed protocol is secure against replay attacks, user impersonation attacks, and password guessing attacks. Also, it satisfies the user anonymity as well as perfect forward secrecy features. On the other hand, similar to many other authentication schemes, it requires storing long-time security keys and does not ensure against devices counterfeiting.

The authors in [95] introduce an authentication protocol for resource-constrained IIoT devices. The proposed protocol is based on simple operations such as XOR, addition, subtraction, and a hash function. The protocol consists of two phases: (1) the registration phase, where IoT node and gateway exchange secrets that they will later use

to prove their identities, and (2) the mutual authentication phase, where the authentication and session key generation process is completed. The protocol is resilient against replay, impersonation, tracking, man-in-the-middle attacks, and identity guessing attacks. Also, it satisfies multiple security requirements such as perfect forward secrecy, device anonymity, mutual authentication, and data integrity. The only limitation of this protocol is storing several long-term secret keys on the IoT node.

Based on the work presented above, there are common limitations that are shared among most of them. First, many suggested protocols used computationally expensive cryptographic operations that do not fit the IoT devices' resource-constrained nature. Second, most of the proposed protocols did not focus on ensuring the originality of the IoT devices and secure the system from integrating counterfeited devices. However, this is considered a major concern in the IIoT domain. Third, many of the proposed protocols do not implement multiple authentication levels, which is the core of the defense-in-depth concept. Fourth, in some of the presented protocols, the researchers overlooked the IoT devices' privacy, which does not guarantee the IoT device's anonymity and privacy. Fifth, many of the proposed protocols require storing long-term secret keys on the IoT device's memory, which can be retrieved using side-channel attacks. Sixth, most of the proposed protocols did not account for gateways load, leading to slowing down the system response and not responding to authentication requests promptly. This will, in turn, negatively impact the system's scalability. Seventh, most of the presented protocols may suffer from losing the authentication service entirely due to the design's single point failure.

5.5 Industrial IoT (IIoT) network model and security goals

This section will present a smart poultry farm network model and introduce the proposed authentication scheme's security mechanisms and components. Then, the threat model and the involved attack surface will be discussed.

5.5.1 Smart poultry farm network model

One of the industries that can make good use of IoT technology is poultry farms. In such an environment, it is essential to monitor the environmental quality to ensure the healthy growth of the chicken and, at the same time, to reduce chicken loss. Therefore, there is a need for a smart monitoring system. The environmental parameters can be measured by using wireless sensors. Collected data can be transmitted through a gateway to the employer's control station to monitor the farm's operation and production. The use of IoT sensors can be a solution to build a smart poultry farm. The implementation of an automated real-time monitoring system of the environmental parameters can: (1) increase productivity and at the same time reduce cost and time; and (2) facilitate the data analysis process that can help in making faster and better decisions. However, IoT and smart communication technologies introduce a vast exposure to cybersecurity threats and vulnerabilities in smart poultry farming environments. Such threats can disturb the operation and the production of the farm. This chapter introduces a distributed lightweight mutual authentication and key generation protocol that fits the poultry farms' dynamic and distributed cyber-physical nature to ensure the system's security.

In the proposed scheme, an IoT node (N) mini gateways (MG), Intermediate gateway (IG), and central gateway (CG) are responsible for ensuring secure, anonymous

mutual authentication and key generation. The proposed scheme is based on a challenge-response technique and is developed as a secure and efficient mutual authentication scheme without requiring high computational and communication costs. The use of symmetric cryptography may generate security issues due to key exchanges over public channels. To solve this issue, the proposed scheme does not exchange keys over the network and does not use the devices' real identities, and consequently, it is not affected by these problems. The proposed scheme enables a M2M communication, which provides the chance for devices inside the IIoT network to do data offloading and allows the IoT devices that are away from their gateways to get authenticated and communicate with them through other IoT devices.

Figure 47 illustrates the network model of the farm. The cage is divided into three levels (Level 1, Level 2, and Level 3). The three levels are identical. Each level has a dedicated gateway called the Intermediate gateway (IG). The three IGs are connected to a central gateway that is in charge of the whole cage. Each level is divided into three zones (Zone A, Zone B, and Zone C). The three zones are identical. Each zone is controlled by a mini-gateway (MG). All MGs can communicate with each other to exchange data. Each zone is divided into multiple virtual domains where the IoT devices are located. Each virtual domain has the more powerful IoT nodes closer to the MG. There are four main players: the central gateway (CG), the intermediate gateways, the mini-gateways, and IoT nodes(N).

- **The central gateway (CG):** CG is a central device for starting and maintaining the network. It is aware of all the devices' real identities. The CG achieves multiple tasks: (1) it is in charge of authenticating the intermediate gateways; (2) it supports the mini-gateways while authenticating the IoT devices; and (3) it is in charge of

sending the collected data from the intermediate gateways to the cloud. The CG also acts as a local repository to temporarily store the data generated from the whole cage and provides a local processing capability, and performs data cleaning, aggregation, analysis, and interpretation techniques because it has a local database. Then it will send the collected data to the cloud.

- **Intermediate Gateways (IG)**: the intermediate gateway controls starting and maintaining the level-based network. The IGs achieve multiple tasks: (1) they are in charge of authenticating the mini-gateways; (2) The IG also acts as a local repository to temporarily store the data generated from its level and provides a local processing capability and performs data cleaning, aggregation, analysis, and interpretation techniques because it has a local database. The IGs store the received data in buffers and forward them to the central gateway.
- **Mini-gateways (MGs):** MGs are less powerful than IGs. Each MG is in charge of a zone and acts as a connection point between the IG and the IoT devices. MGs are used to offload the IG and distribute the authentication process among multiple MGs under the authority of the IG. The MGs will be in charge of authenticating all the IoT nodes. Each MG acts as a local repository to temporarily store the data generated from its zone and provides a local processing capability, and performs data cleaning, aggregation, analysis, and interpretation techniques because it has a local database. The MGs store the received data in buffers and forward them to their IGs., which aggregate the information of different MGs and redirect it to the central gateway.
- **IoT nodes**: Those are the limited resources devices that are used to collect and send data to its MG. The IoT nodes are not allowed to communicate directly with the IG

or the CG. If either the CG or the MG receives any message from any IoT node, it will drop it. Some IoT nodes are connected directly to the MG, while others are connected indirectly through other connected IoT nodes to MG. The IoT devices include temperature sensors, water sensors, CO_2 $sensor_s$, NH_2 sensors, static pressure sensors, and actuators.

The presented scheme consists of four main layers: the physical, edge, fog, and cloud layers. The bottom layer is the physical layer consisting of sensors and actuators distributed in the cage. These devices include different types of sensors such as temperature sensors and water sensors, and actuators to take actions. The sensors are responsible for sensing and collecting data from the surrounding environment. The collected data helps in actuating other devices such as the ventilation system. The sensors collect real-time data about water, temperature, CO2 level, or NH2 level, which can be sent to the edge, fog, or cloud systems to provide recommendations and enable automation. For example, data collected by the temperature sensors are processed at the edge to determine the amount of air or heat that is needed in the cage, optimize the airing or the heating schedule, and support the farm operation.

The next layer up is the edge layer that consists of the mini gateways. Each virtual domain has its own mini gateway that provides communication among smart objects and acts as the link between the smart device and the intermediate gateway. The MG in the edge layer oversees conducting real-time data cleaning, aggregation, analysis, and making decisions within its virtual domain.

The third layer up is the fog layer that consists of the intermediate gateways and the central gateway. The fog layer oversees local real-time data cleaning, aggregation,

analysis, and making decisions. This layer's integration reduces the computation and data analysis load off the centralized cloud layer. The fog layer consists of two sub-layers. The bottom sublayer consists of the intermediate gateways. Each Intermediate gateway oversees the cleaning, aggregating, and analyzing the data of its level. The top sublayer consists of the central gateway. The central gateway aggregate and analyze the data of the whole cage. Security monitoring, device failure prediction, and anomaly detection systems can be deployed for real-time monitoring of abnormal events and classifying these events as malicious or benign.

The cloud layer is generally virtualized in data centers and communicates with the other layers using the Internet. The network model presented in figure 47 has two clouds. The first one is the cloud of the service provider, where the collected data that has been sent through the edge layer will be stored on Distributed File System (DFS). This stored data will be used to conduct further analysis and to mine knowledge. The second cloud is the PUF cloud, where the manufacturers store the CRPs of their produced devices. The CRPs will be used to authenticate the devices and ensure the originality of the devices to protect the farms from integrating counterfeited devices.

The IoT nodes such as sensors and actuators represent the sensing layer that is used to sense/control the industrial world and acquire data. The MGs, IGs, and the CG represent the aggregation layer responsible for collecting and processing the data and then sending information to the cloud. All MGs can communicate directly with each other. Each IoT node can communicate directly with its MG but can't communicate with either the IG or the CG. A trust relationship and a secure communication link exist between the CG. and the service provider cloud environment.

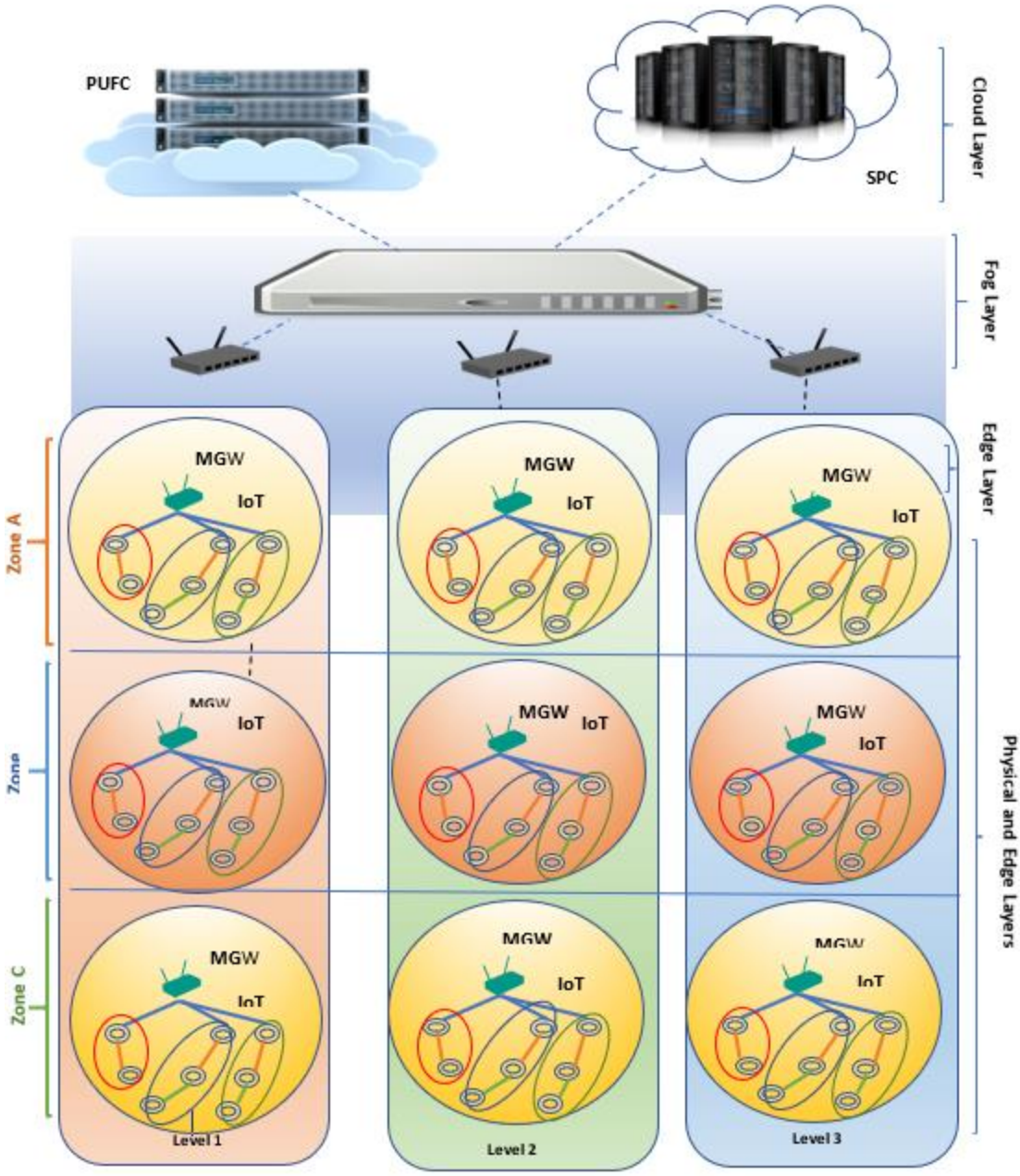


Figure 47: Smart poultry farm network model

The manufacturer assigns the IoT nodes real IDs that are stored on their memory. The service provider gives the IoT nodes Alias IDs. The Alias ID will be changed every authentication session. The goal of the alias ID is to protect the device's anonymity and ensure the device's untraceability.

All devices are stationary. The communication between any two devices is subject to both active and passive attacks. In passive attacks, the adversary can eavesdrop on the communication. In active attacks, the intruder may replay, modify, or delete specific communication messages. Moreover, IoT devices can be destroyed or stolen. The message header containing the device Alias ID is sent in a clear text while the payload is encrypted and authenticated. A one-way cryptographic hash function with a 256 bits length is used to validate and ensure the integrity of the data transmitted between the two devices.

The number of IoT devices can vary throughout the lifetime of the network. Moreover, because the addition of new IoT devices to the network is highly likely, the proposed scheme accommodates system scalability. The proposed scheme supports the dynamic addition of IoT devices to the network without changing the network's security states. The dynamic addition of new IoT devices is a shared responsibility between the CG and the MG., which are assumed to be powerful devices.

5.5.2 Automated access control and privacy

The nature of the multi-layered architecture of the poultry farms highlights the chances of cyber threats challenges. Therefore, an access control solution needs to be integrated to support smart poultry farms' dynamic nature. Efficient access control and privacy protection are necessary to protect sensitive data [113], [114]. Therefore, an automated access control model based on virtual domain segregation of IoT network and device type is introduced.

5.5.2.1 IoT virtual domains segregation

The proposed scheme creates virtual segregation among the IoT devices. Virtual segregation's primary goals are to (1) improve security through applying security policies that fit the different IoT devices' needs and increase the data's confidentiality;(2) improve the network performance through eliminating the unnecessary traffic; and (3) improve and ease the management of the network.

The IoT network is virtually partitioned based on the type of IoT device. Each virtual domain is connected to its MG to manage and control that virtual domain. If an MG of a specific virtual domain goes down, its neighbor gateway that is located at the same level would temporarily take over the connection management. At the same time, it will notify the IG that its neighbor MG is out of service. All MGs within the same level periodically exchange heartbeats signals to ensure their availability over time and share a common database that lists those sensors under the authority of each MG. The IoT devices are organized in a multi-tier architecture where the top layer (L1) houses the most powerful IoT nodes within the MG range. The lower layer (L2) includes the IoT nodes that are distant and out of range from the MG but within their peers' range sitting at the top layer. The L1 IoT nodes will be used as a relay between the L2 IoT nodes and the MG. Each virtual domain has a unique virtual Domain ID (VD_{ID}) and a unique Group address (G_{AD}). The G_{AD} will equal the hardware address of the MG. Each device has a typeID (T_{ID}) that is shared among all devices with the same type.

5.5.3 Challenge-response mechanism based on PUF and chained hash PUF

The proposed scheme achieves mutual authentication by implementing a PUF challenge-response and chained hash PUF value authentication mechanism. The idea

behind using the chained hash PUF value is to create a type of tracking system that stores a value related to interaction over time between the IoT node and the MG. The chained hash PUF value is utilized to present a historical factor for authenticating the IoT node. Employing both the PUF and the chained hash PUF mechanisms to generate a cumulative value from all previous authentication sessions ensures mutual authentication. The two-way challenge/response authentication technique allows the MG to check the authenticity of the IoT node, and at the same time, enables the IoT node to ensure that it is not communicating with a malicious MG. The use of the PUF proves the originality of the devices and protects the IoT ecosystem from impersonation attacks. The use of the cumulative chained hash PUF value depends on the ability of the IoT node and MG to show proof of knowledge of past chained hash PUF values.

In the proposed scheme, during the registration phase, both the sending IoT node and the MG hash the first PUF response R_{H1} along with the real ID of the IoT node and produce a hashed value named C_{HX}. Then, for all subsequent authentication sessions, the IoT node hashes the previously-stored chained C_{HX} along with the new PUF response (Rx) to generate a new chained hash CHx_N. On the other side, the MG will apply the same technique to ensure the authenticity of the received hashed value upon receiving the data. First, the MG retrieves the Rx value from its database. Second, the MG hashes the retrieved Rx with the previously stored chained CH_{x-1}. Then, the MG compares the computed value $CH_{XN}' = h (R_X, CH_{x-1})$ with the one that was received CH_{XN} from the IoT node. If it matches, it will authenticate the sender IoT node N. Once the authentication is completed; the two sides will store the CH_{XN} in their databases and use the CHx_{-1} along with the exchanged random values to generate the shared session key(*ssk*).

The proposed mechanism chains the blocks of hashed response values together by hashing the new hash value with previously generated hash values and the $real_{ID}$ of the IoT node. Thus, the mechanism looks like a blockchain technology at first glance, but it does not utilize blockchains with its overheads. Because the R_X values cannot be predicted or replicated due to the nature of the PUF, the adversary can't predict the chained hash value.

5.5.4 Security Design Goals

To ensure the security and efficiency of communication in the IIoT domain, the security goals that are presented in figure 48 need to be achieved in the proposed protocol.

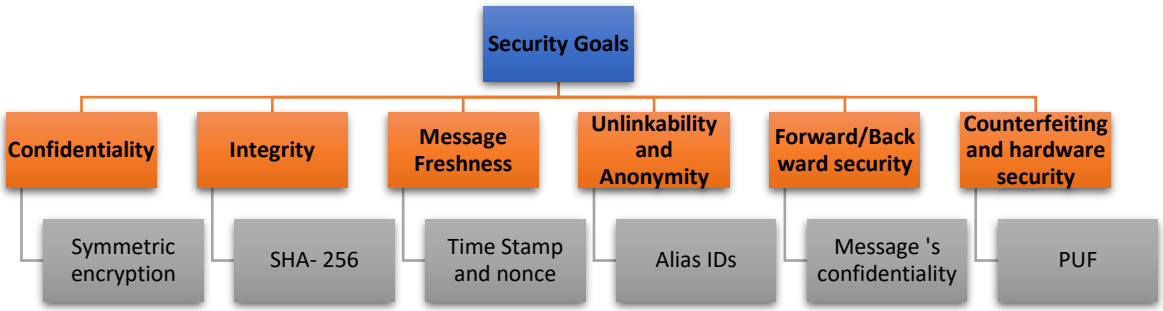


Figure 48: Security Design Goals

- **Confidentiality**: This is achieved through symmetric encryption.
- **Resistance to data modification attack**: An adversary cannot tamper with the communication data to break its integrality: This is achieved by using a strong hash function such as SHA-256.
- **Anonymity and unlinkability**: The sensors' identity information cannot be exposed to anyone outside its level. This is achieved through using an Alias ID/ pseudonym that will be changed for every authentication session. The adversary cannot trace back the device's real ID from the pseudonym. Using pseudonyms also ensures that when an IoT device uses network resources multiple times, it will be hard to link these uses together by an adversary.
- **Resistance to the Replay Attack**: An adversary cannot reuse the previously exchanged valid information to steal sensitive data. This attack can be avoided by ensuring the message's freshness by using timestamps and nonce values.
- **Forward/backward security**: This is satisfied by ensuring the confidentiality of all messages.
- **Counterfeiting and hardware security**: This is achieved through using the PUF
- **Mutual Authentication**: Each communicating pair should authenticate each other to avoid potential malicious attacks.
- **Resistance to the Impersonation Attack**: An adversary pretends to be a registered entity. The implementation of the PUF concept can avoid this attack.

- **High Efficiency**: To encounter the communication demands of the resource-constrained IoT devices in industrial systems, the proposed schems should have low-computation cost and communication overhead.

5.5.5 Threat Model

The Dolev–Yao (DY) threat model [64] is applied in the presented protocol. In the DY threat model, any two communicating parties (in the context, IoT node and MG, MG and IG, and IG and CG) communicate with each other via insecure public channels. During the interaction process, an adversary can eavesdrop, intercept, and modify the transmitted data of both parties. The service provider cloud and the PUF cloud are assumed to be fully trusted in the presented scheme.

The following assumptions about security properties and adversary abilities are made.

1. The used communication channel during the registration process is secure.

2. The one-way hash function is collision-resistant.

3. The gateways and IoT nodes have protection against tampering.

4. Replay attack: An adversary can capture messages from old authentication sessions and replay them in the current session.

5.Message modification attack: the adversary can tamper with intercepted messages.

7. Injection attack: the adversary can send fake messages.

6. Impersonation attack: The adversary can pretend to be a legitimate gateway or sensor node.

5.6 Proposed Scheme: M2M Distributed Multi-Layer Lightweight Mutual Authentication and Key Agreement Scheme Using Chained Hash PUF in Industrial IoT System

This work focuses on resource-constrained IoT devices that are not suitable for traditional security methods to protect communication with the gateway and IoT devices. Deploying a massive number of IoT devices in the IIoT environment may consume a considerable amount of energy. Therefore, there is a need to optimize the communication tasks among the IoT devices to reduce the consumed energy [115]. The proposed scheme aims to achieve low computing-resource usage by reducing computational complexity and the computational cost of IoT devices, which will reduce the energy cost of the IIoT system and extend the battery life of the IoT devices.

Because most IoT devices are resource-constrained, we use lightweight cryptographic operations during exchanged authentication messages. To achieve this goal, a lightweight mutual authentication and key agreement scheme for M2M communication between a CG and IG, IG and MG, and finally between N and MG are proposed in this section.

The proposed scheme consists of four main phases: the enrollment phase, registration phase, mutual authentication phase, and key agreement phase. The enrollment phase will be completed during manufacturing. The Authoritative entity (AE) will oversee completing the registration phase. Both the mutual authentication and the key generation phases will be completed between CG and IG, IG and MG, and N and the MG without any human involvement. Every two parties are responsible for ensuring a secure, anonymous mutual authentication and key generation process. The relationship between CG and IG,

IG and MG, and N and MG is based on infrastructure communication mode. The relationship among the different MGs and the relationship among the IoT nodes within the same virtual domain is based on peer-to-peer topology. The mutual authentication and the key generation phases are broken down as follows:

1. Mutual authentication and key agreement between CG and IG.
2. Mutual authentication and key agreement between IG and MG.
3. Mutual authentication and key agreement between two different MGs.
4. Mutual authentication between MG and N.

The proposed scheme allows any two devices in the M2M network to authenticate each other and exchange data mutually. Moreover, the IoT nodes are not necessarily required to be directly connected to their mini gateway at the time of authentication. It is important to highlight that the authenticated devices must be within the same virtual domain. Any communication between any two IoT nodes from two different virtual domains must be through the mini gateway of each of them. The proposed scheme utilizes a one-way hash function, PUF, chained hash PUF and simple XOR function between any two mutually authenticated devices.

Furthermore, IoT nodes use Alias IDs (pseudonyms) instead of their real IDs during the communication process. Alias IDs support the anonymity of senders' IDs, receivers' IDs, and the sender-receiver relationship. The abstract notations used to describe the proposed authentication scheme are listed in table 15. The scheme phases are presented below.

5.6.1 Enrollment Phase

The enrollment phase is completed during the manufacturing process. The manufacturer will assign a real ID to every IoT device and gateways. All devices are equipped with a PUF, and any attempt to tamper with the PUF will change the device's behavior and render the device useless and must be replaced. The PUF hardware security primitive is used as the root-of-trust, which provides device authenticity.

The manufacturer will also collect a set of CRPs for each device during the enrollment phase. The manufacturer will test all the CRPs under different temperature and voltage conditions and consider the aging effects to keep only the error-free CRPs [37]. Then, the manufacturer will load the CRPs of the devices to the cloud of their service provider. The service provider container is part of the PUF architecture that is presented in chapter 2. The main goals of PUF are to protect the devices from counterfeiting and generate a relevant, unique key for the device. The uniqueness of the generated key in the network would be guaranteed based on the diversity and the manufacturing variations that generate the PUF. The generated PUF responses will be used to generate the symmetric keys.

Table 15: Notations used in the scheme

Notation	**Description**
Authoritative Entity	AE
IoT node	N
Mini-Gateway	MG
Intermediate Gateway	IG
Central Gateway	CG

N_{IDR}	IoT Node Real ID
N_{IDA}	IoT node Alias ID
MG_{IDR}	Mini-Gateway Real ID
MG_{IDA}	Mini-Gateway Alias ID
IG_{IDR}	Intermediate Gateway Real ID
IG_{IDA}	Intermediate Gateway Alias ID
CG_{IDR}	Central Gateway Real ID
CG_{IDA}	Central Gateway Alias ID
OTP	One-time password
Token	TK
MSK	Master secret key
GID	Group ID
T_{IDN}	IoT node Type ID
RV	Random values
S1, S2, and S3	Authentication parameters
TS	Timestamp generated by the IoT node
PUF	Physical unclonable function
C	Challenges
R	Response
H	Hash function

5.6.2 Registration Phase

In this phase, the Authoritative Entity (AE) securely registers CG, IG, MG, and N. Before operation in the field, the AE will complete multiple tasks, as described in figure 49. First, the AE will assign alias IDs to the CG, IGs, MGs, and N nodes. The alias IDs will be fake IDs that devices will use to communicate with each other. Alias IDs will act as pseudonym IDs and will be dynamically changed in every authentication session. The

goal of the Alias ID goal is to protect and ensure the unlinkability and untraceability of the device and support the anonymity of each device in each session. Second, the AE will retrieve CRPs (C_{CG}, $R_{CG,}$ C_{IG}, $R_{IG,}$ C_{CMG}, R_{MG}) of the CG, IG, and MG, respectively, from the PUF cloud

Third, AE will use a random number generator to generate three random values to be used as for one-time passwords (OTP) between CG and each IG. There will be a unique OTP between the CG and each IG. Also, the AE will generate a master secret key (MSK_{CI}) to be only known to the CG and the three IGs. Once the IG is registered, IG and CG will generate the OTP during every authentication session without any human involvement. Then, the AE will insert and store CG_{IDR}, $CG_{IDA,}$ IG_{IDR}, $IG_{IDA,}$ MSK_{CI}, and OTP_{CI} in the IG's database and store CG_{IDR}, $CG_{IDA,}$ IG_{IDR}, $IG_{IDA,}$ MSK_{CI}, and OTP_{CI} of each IG in CG' memory.

Fourth, AE will use a random number generator to generate three random values to be used as one-time passwords (OTP) between the IG of each level and its MGs. There will be a unique OTP between the IG and each MG within the same level. Also, The AE will generate a master secret key (MSK_{IM}) to be only known to the IG and the MGs inside the same level. Once MGs are registered, both IG and MG will generate the OTP during every authentication session without any human involvement. The AE will insert and store MG_{IDR}, $MG_{IDA,}$ MSK_{IM}, and OTP_{IM} in the MG's database and store IG_{IDR}, $IG_{IDA,}$ MG_{IDR}, $MG_{IDA,}$ MSK_{IM}, and OTP_{IM} of each MG in IG' memory.

Fifth, AE will assign the IoT node Group ID(G_{IDN}), and type ID (T_{IDN}). The G_{IDN} refers to the virtual domain that N belongs to, and the T_{IDN} indicates the type of the IoT device. Second, the AE will use a random number generator to generate a one-time

token (OTT) to be used as part of a one-time password (OTP) between N and the MG. Once N is registered, both N and MG will generate the OTT during every authentication session without any human involvement. Then the AE will use N's different IDs and the

CG-IG Registration

Step1: AE uses a random number generator to generate a random value to be used as a one-time password (OTP_{Ci}) between CG and each IG_i.
Step 2: AE generates a master secret key (MSK_{CI}) only known to the CG and its IGs.
Step 3: AE communicates with the PUF cloud to retrieve a CRP (C_{CG}, R_{CG}) of the CG and CRP (C_{IGi}, R_{IGi}) of the IG_i
Step 4: The AE calculates the authentication parameters as follows:

$CGA = H (CG_{IDR} \| IGi_{IDR} \| R_{CG})$

$CIAi = H (H (CG_{IDR} \| IGi_{IDR} \| R_{IGi})$

Step 5: AE inserts and stores CG_{IDR}, $CG_{IDA,}$ IG_{IDR}, $IG_{IDA,}$ MSK_{CI}, CGA, C_{IGi}, and OTP_{Ci} in the IG's database. The MSK_{CI} is stored in an encrypted format using the $R_{IGi.}$ The IGi only knows the $C_{IGi.}$ R_{IGi} is unknown to the IGi. To calculate CIAi and decrypt the OTP_{Ci} and the MSK_{CI}, the IGi needs to feed the C_{IGi} to the PUF function to compute the R_{IGi}.
Step 6: The AE inserts and stores CG_{IDR}, $CG_{IDA,}$ IGi_{IDR}, $IGi_{IDA,}$ MSK_{CI}, CIAi, C_{CG}, and OTP_{Ci} in CG' memory. The MSK_{CI} is stored in an encrypted format using the $R_{CG.}$ The CG only knows the $C_{CG.}$ R_{CG} is unknown to the CG. To calculate CGA and decrypt the OTP_{Ci} and the MSK_{CI}, the CG needs to feed the C_{CG} to the PUF function to compute the R_{CG}.

IG-MG Registration
Step1: AE uses a random number generator to generate a random value to be used as a one-time password (OTP_{IM}) between IG and the MGi.
Step 2: AE generates a master secret key (MSK_{IM}) to be only known to the IG and the three MGs that are within the same level.
Step 3: AE communicates with the PUF cloud to retrieve a CRP (C_{MGi}, R_{MGi}) of the MGi.
Step 4: The AE calculates the following authentication parameter as follows:

$MGAi = H (IGi_{IDR} \| MGi_{IDR} \| R_{MGi})$

$IMAi = H (IGi_{IDR} \| MGi_{IDR} \| R_{IGi})$

Step 5: AE inserts and stores MG_{IDR}, $MG_{IDA,}$ MSK_{IM}, MGAi, and OTP_{IM} in the IG's database. The MSK_{IM} is stored in an encrypted format using the $R_{IGi.}$ The IGi only knows the $C_{IGi.}$ R_{IGi} is unknown to the IGi. To calculate IMAi, and decrypt the MSK_{IM} and OTP_{IM}, the IGi needs to feed the C_{IGi} to the PUF function to compute the R_{IGi}.
Step 6: AE inserts and stores IG_{IDR}, $IG_{IDA,}$ MG_{IDR}, $MG_{IDA,}$ MSK_{IM}, IMAi,$C_{MGi,}$ and OTP_{IM} in the MGi's memory. The MSK_{IM} is stored in an encrypted format using the $R_{MGi.}$ The MGi only knows the $C_{MGi.}$ R_{MGi} is unknown to the MGi. To calculate MGAi

and decrypt the MSK_{IM} and OTP_{IM}, the MGi needs to feed the C_{MGi} to the PUF function to compute the R_{MGi}.

MG-N Registration

Step 1: The AE assigns N to its appropriate virtual domain and generates a G_{IDN} to the N.

Step 3: The AE generates and assign a T_{IDN} to each N based on its type

Step 4: The AE calculates the three parameters: S_1, S_2, and S_3 as follows:

S_1: h (N_{IDR} || MG_{IDR} ||R_{MGi})

S_2: h (S_1 ||MG_{IDR} || G_{IDN})

S3: h (S_2|| MG_{IDR} ||T_{IDN})

$G_{IDN,}$ T_{IDN} and R_{MGi} are only used to create S_1, $S_{2,}$ and S_3. Both the G_{IDN} and the T_{IDN} are only stored on the MGi. The IoT node does not know anything about those two IDs and the R_{MGi}. The C_{MGi} is only stored on the MGi database. R_{MGi} is unknown to both the IoT node and the MGi because the MGi will compute R_{MGi} on every authentication session. Using G_{IDN} and T_{IDN} to generate the authentication parameters makes it almost impossible for an adversary to be able to identify the G_{IDN} and the T_{IDN} of any IoT node. Also, using the R_{MGi} makes is impossible for an adversary to impersonate the MGi.

Step 5: AE generates a one-time token (OTT) that will be used as part of a one-time password (OTP) using a random number generator

Step 6: AE stores the S_1, S_2, S_3, N_{IDA}, MG_{IDA}, MG_{IDR}, and OTT in the IoT device database.

Step 7: AE inserts the N_{IDR}, N_{IDA}, $G_{IDN,}$ T_{IDN}, and OTT of the IoT node in the MG database.

Figure 49: Registration Phase

MG's real ID to generate three different authentication parameters, which are S1 and S2, and S3, using a one-way hash function and XOR operations. Then, the AE will insert and store N_{IDR}, $N_{IDA,}$ the $MG_{IDA,}$ G_{IDN}, T_{IDN}, and the OTT in the MG's database and store S_1, S_2, S_3, OTT, N_{IDA}, $MG_{IDR,}$ and MG_{IDA} in N' memory.

The real identities of the devices represent a secure parameter between the devices within the network. The real IDs will never be transmitted in a plain text format and will be used during the authentication process between every two devices.

5.6.3 Mutual authentication and key generation phases

This phase will cover three different protocols. The protocol depends on the relationship between every two nodes. Figure 50 presents the different types of relationships that we have in the poultry farm IoT network. We have a parent-child relationship, and this is presented as the relationship between a gateway and the other gateway that is above it. The two examples of the parent-child relationship are the CG-IG relationship and the IG-MG relationship. The second type of relationship is the child-child relationship. This type of relationship is between two gateways at the same level. The two examples in the presented network are IG-IG relationship and MG-MG relationship. The third type of relationship is between a gateway and IoT.

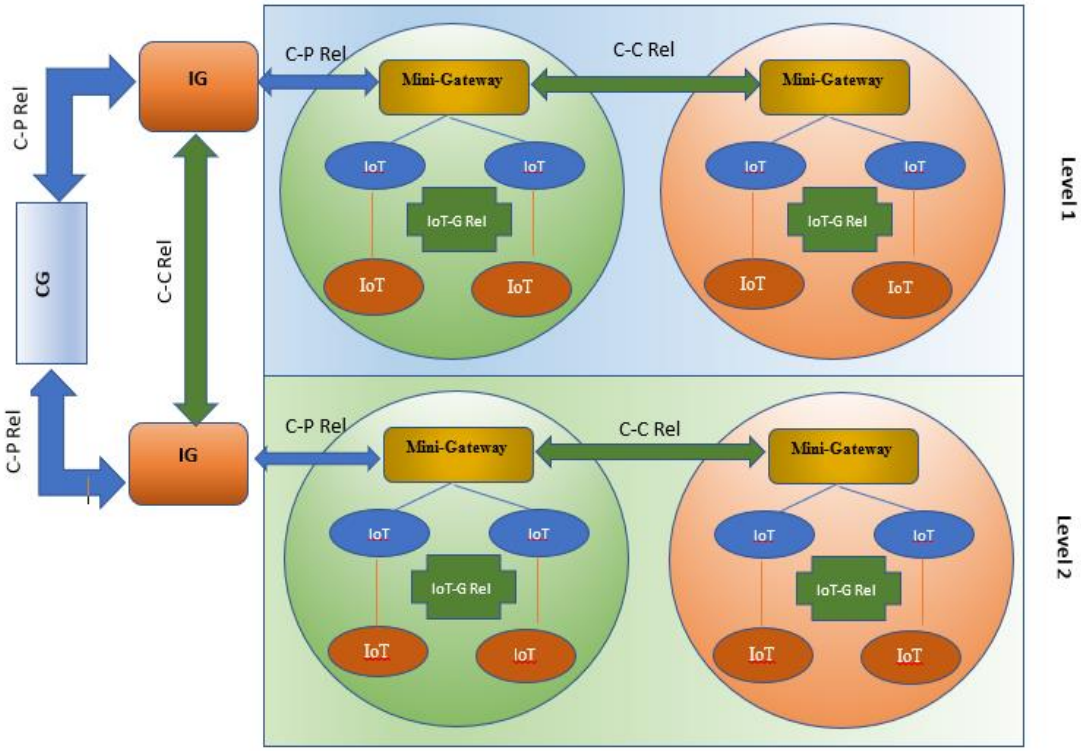


Figure 50: Different types of relationships in the poultry farm network

5.6.3.1 Protocol 1: parent gateway-child gateway authentication and key generation

This protocol presents the mutual authentication process between parent gateway (PG) and child gateway (ChG) and establishes a session key for data transfer. As presented in figure 51, this process starts when ChG prepares the authentication request message that will be sent to the PG. Also, the algorithm for the proposed parent gateway-child gateway mutual authentication mechanism is shown in detailed steps in figure 52. The steps involved in this process are listed below.

Child Gateway (ChG)	Parent Gateway (PG)
Generate: TS_{ChG1}, RV_{ChG1} **Compute**: $X_1 = H(OTP \| Auth_P)$ $X_2 = X_1 \oplus (RV_{ChG1})$ $X_3 = H(ChG_{IDR} \| TS_{ChG1} \| RV_{ChG1} \| X_1)$	
M1: $(ChG_{IDA}, TS_{ChG1}, X_2, X_3)$ →	
	Check: $\|T_{Rec} - TS_{ChG1}\| < \Delta T$ **Find**: ChG_{IDA} **Read**: ChG_{IDR} C_p, OTP **Compute**: $R_P = PUF(C_P)$ $Auth_P' = H(PG_{IDR} \| ChG_{IDR} \| R_P)$ $X_{1'} = H(OTP \| Auth_P')$ $RV'_{ChG1} = X_2 \oplus X'_1$ **Verify**: $X_3' = H(ChG_{IDR} \| TS_{ChG1} \| RV_{ChG1}' \| X_1')$ **Generate**: $TS_{PG1}, RV_{PG1}, S_{ID}$ **Computes**: $Yx = H(OTP \| Auth_c)$ $Y_1 = RV_{PG1} \oplus H(Yx \| RV_{ChG1})$ $Y_2 = H(PG_{IDR} \| TS_{PG1} \| RV_{PG1} \| RV_{ChG1} \| Yx)$
← M2: $(PG_{IDA}, S_{ID}, TS_{PG1} Y_1, Y_2)$	
Check: $\|T_{Rec} - TS_{PG1}\| < \Delta T$ **Read**: C_C, OTP **Compute**: $R_C = PUF(C_C)$ $Auth_C' = H(PG_{IDR} \| ChG_{IDR} \| R_C)$ $Y_{X'} = H(OTP \| Auth_C')$	

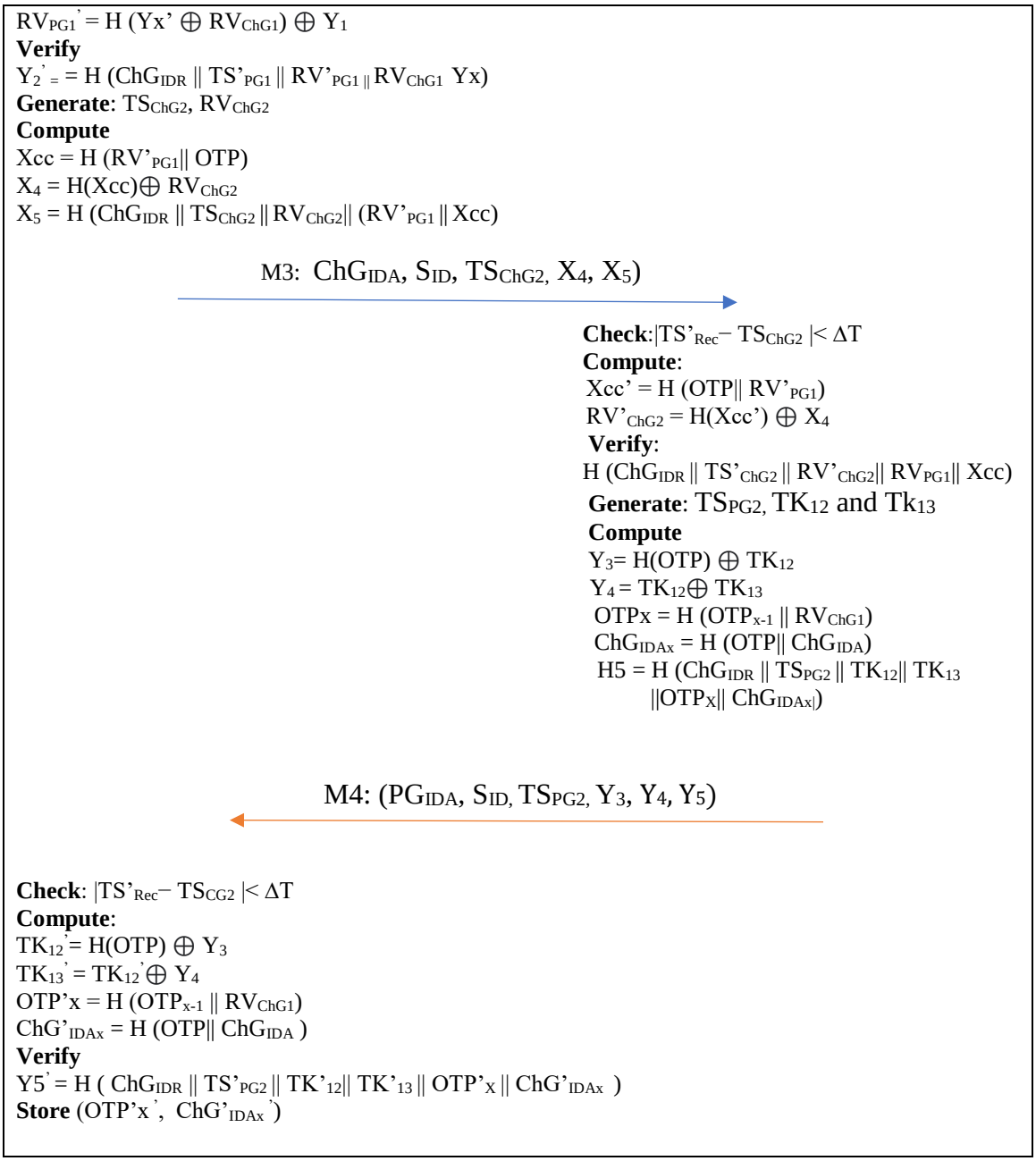


Figure 51: The authentication process between PG and CG

5.6.3.1.1 Step 1: Interaction Request

1. ChG generates a new TS_{ChG1} to avoid replay attacks
2. ChG computes X_1
   a. $X_1 = H(OTP \| Auth_P)$

3. ChG selects a random parameter RV_{ChG1} and then calculates X_2

   a. $X_2 = X_1 \oplus (RV_{ChG1})$

4. ChG computes $X_3 = H (ChG_{IDR} \| TS_{ChG1} \| RV_{ChG1} \| X_1)$

5. ChG sends the message to the PG.

The message includes ChG_{IDA}, TS_{ChG1}, X_2, and X_3, as shown in 5.1.

$$ChG \longrightarrow PG\ M_1(ChG_{IDA}, TS_{ChG1}, X_2, X_3) \qquad (5.1)$$

Algorithm 2 The mutual authentication between PG device and the ChG

Input:

Child gateway (ChG) device with real identity (PG_{IDR}), alias identity (PG_{IDA}), IG real identity (ChG_{IDR}), IG alias identity (ChG_{IDR}), master secret key (MSK), PUF Challenge (C_{ChG}), the Authp authentication parameter, and one time password (OTP)

PG device with real identity of the PG (PG_{IDR}), alias identity of the PG (PG_{IDA}), ChG real identity (ChG_{IDR}), IG alias identity (ChG_{IDR}), master secret key (MSK), one-time password (OTP), PUF challenge (Cp), and the authentication parameter $Auth_c$

Output:

Mutual authentication between the CG and the IG

Begin

1. The ChG device generates a random nonce RV_{ChG1}, a timestamp TS_{ChG1}, $X_1 = H (OTP \| Auth_P)$ $X_2 = X_1 \oplus (RV_{ChG1})$, and $X_3 = H (ChG_{IDR} \| TS_{ChG1} \| RV_{ChG1} \| X_1)$
2. **ChG** device sends $<ChG_{IDA}, TS_{ChG1}, X_2, X_3>$ message to the PG.
3. **If** (the PG finds ChG_{IDA} in its repository) **then**
4. The PG retrieves ChG_{IDR}, OTP, and C_p from its repository to its memory;
5. The PG passes the challenge Cp to its PUF function and generates a response RP_G
6. The PG calculates $Authp'$. Then the PG computes $X_{1'}$, retrieves the RV'_{ChG1} from XORing X_1 and X_2 and finally calculates $X_3' = H (ChG_{IDR} \| TS_{ChG1} \| RV_{ChG1} \| X_1)$

7. **If** (the calculated hash message in step 6 matches the hash message that was sent in step 2) **then**
8. The PG generates a timestamp TS_{P1}, generates a random nonce RV_{P1}, calculates the hash value $Yx = H(OTP \| Auth_C)$, $Y_1 = RV_{PG1} \oplus H(Yx \| RV_{ChG1})$, and $Y_2 = H(PG_{IDR} \| TS_{PG1} \| RV_{PG1} \| RV_{ChG1} \| Yx)$
9. The PG sends PG_{IDA}, S_{ID}, TS_{P1}, TS_{PG1} Y_1, and Y_2 message ChG.
10. **else**
11. Go to step 42.
12. **end if**
13. **else**
14. Go to step 42.
15. **end if**
16. ChG verifies the time stamp and retrieves C_C, OTP. ChG computes $R_C = PUF(C_C)$, $Auth_C' = H(PG_{IDR} \| ChG_{IDR} \| R_C)$, and retrieves RV_{ChG1}. ChG generates $Y_2' = H(ChG_{IDR} \| TS'_{PG1} \| RV'_{PG1} \| RV_{ChG1}, Yx)$
17. **If** (the calculated hash message in step 16 matches the hash message that was sent in step 9)
18. **then**
19. The authenticity of the PG is verified
20. ChG generates a timestamp TS_{ChG2} and a random nonce RV_{ChG2}, ChG $Xcc = H(RV'_{PG1} \| OTP)$, $X_4 = H(Xcc) \oplus RV_{ChG2}$, and $X_5 = H(ChG_{IDR} \| TS_{ChG2} \| RV_{ChG2} \| (RV'_{PG1} \| Xcc)$
21. ChG sends $<ChG_{IDA}, S_{ID}, TS_{ChG2}, X_4, X_{5>}$ message to PG.
22. **else**
23. Go to step 42.
24. **end if**
25. The PG verifies the TS_{ChG2}, PG computes $Xcc' = H(OTP \| RV'_{PG1})$ and retrieving RV'_{ChG2} by xoring = $H(Xcc')$ and X_4. Also, PG generates $X_5' = H(ChG_{IDR} \| TS_{ChG2} \| RV_{ChG2} \| (RV'_{PG1} \| Xcc)$
26. **If** (the calculated hash message in step 25 matches the hash message that was sent in step 21
27. **then**
28. The authenticity of the IG device is verified.
29. PG generates a timestamp TS_{PG2} TK_{12} and Tk_{13}. Also, PG computes $Y_3 = H(OTP) \oplus TK_{12}$, $Y_4 = TK_{12} \oplus TK_{13}$, $OTPx = H(OTP_{x-1} \| RV_{ChG1})$,

ChG_{IDAx} = H (OTP|| ChG_{IDA}), and Y5 = H (ChG_{IDR} || TS_{PG2} || TK_{12}|| TK_{13} ||OTP_X|| $ChG_{IDAx|}$)

30. PG sends < PG_{IDA}, $S_{ID,}$ $TS_{PG2,}$ Y_3, Y_4, Y_5> message to ChG.
31. , ChG Xcc = H (RV'_{PG1}|| OTP), X_4 = H(Xcc)⊕ $RV_{ChG2,}$ and X_5 = H (ChG_{IDR} || TS_{ChG2} || RV_{ChG2}|| (RV'_{PG1} || Xcc)
32. ChG <ChG_{IDA}, S_{ID}, $TS_{ChG2,}$ X_4, $X_{5>}$ message to PG.
33. **else**
34. Go to step 42.
35. **end if**
36. ChG verifies the $TS_{PG2.}$ Also, ChG computes TK_{12}'= H(OTP) ⊕ $Y_{3,}$ TK_{13}' = TK_{12}'⊕ Y_4, OTP'x = H (OTP_{x-1} || RV_{ChG1}), ChG'_{IDAx} = H (OTP|| ChG_{IDA}). Finally, ChG generates $Y5'$ = H (ChG_{IDR} || TS'_{PG2} || TK'_{12}|| TK'_{13} || OTP'_X || ChG'_{IDAx})
37. **If** (the calculated hash message in step 36 matches the hash message that was sent in step 32
38. **then**
39. ChG stores OTP'x', ChG'_{IDAx} in its database
40. **else**
41. Go to step 42.

End if

42. **Stop** (terminates the connection)

End

Figure 52: Algorithm 1 The mutual authentication between PG device and the ChG

5.6.3.1.2 Step 2: Parent Gateway Response

Once the PG receives the connection request, it will complete the following steps:

1. PG checks the validity of the received timestamp | TS'_{Rec}− TS_{ChG1} |< ΔT. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.

2. PG retrieves from its database the ChG_{IDR} that corresponds to its $ChGG_{IDA}$. If the PG did not find the IG_{IDA} in its database, it will ignore and drop the received message.
3. PG will retrieve the C_p and pass it to the PUF function to compute the R_P
   a. $R_P = PUF(C_p)$
4. PG computes $Auth_P' = H (PG_{IDR} || ChG_{IDR} || R_P)$
5. PG computes the $X_1' = H (OTP || Auth_P')$
6. PG computes the $RV'_{ChG1} = X_2 \oplus X'_1$
7. The PG uses the calculated X_1', TS_{ChG1}', ChG_{IDR}, and the RV'_{ChG1} to verify X_3', which is a combination of ChG_{IDR}, X_1', TS_{ChG1}', and RV'_{ChG1} using a one-way hash function as shown in figure 53. If the computed value is equal to the received value, the PG accepts the connection request; otherwise, it will drop it.

$X_3' = H (ChG_{IDR} || TS_{ChG1} || RV_{ChG1}' || X_1')$
If $X_3 = X_3'$ Then
    The PG will accept the interaction request
Else
    The PG will drop the interaction request

Figure 53: Verify the interaction request parameters

Once the PG accepts the interaction request, it generates a random value RV_{PG1} and time stamp TS_{PG1}. Also, the PG generates a session ID. The session ID aims to distinguish one session from the rest of the running sessions simultaneously. The PG prepares and sends an interaction response to the ChG. The preparation process of the connection response message includes the following steps:

1. PG generates a new TS_{PG1}
2. PG Calculates Yx
    a. $Yx = H (OTP \| Auth_c)$
3. PG selects a random parameter RV_{PG1} and then calculates Y_1
    a. $Y_1 = RV_{PG1} \oplus H (Yx \| RV_{ChG1})$
4. PG generates a session ID(S_{ID})
5. PG calculates $Y_2 = H (PG_{IDR} \| TS_{PG1} \| RV_{PG1} \| RV_{ChG1} \| Yx)$
6. PG sends an interaction response message

The interaction response includes $PG_{IDA,}$ session ID(S_{ID}), $TS_{PG1,}$ Y_1, and Y_2, as shown in 5.2.

$$PG \longrightarrow ChG\ Conn\text{-}Res\ (PG_{IDA}, S_{ID,} TS_{PG1,} Y_{1,} Y_2) \qquad (5.2)$$

5.6.3.1.3 Step 3: Parent Gateway Authentication

Once ChG receives the connection response, it completes the following steps:

1. ChG checks the validity of the received timestamp $| TS'_{Rec} - TS_{PG1} | < \Delta T$. TS'_{Rec} is when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. ChG will retrieve the C_C and pass it to the PUF function to compute the R_C
    a. $R_C = PUF(C_C)$

3. ChG computes $Auth_C' = H (PG_{IDR} || ChG_{IDR} || R_C)$
4. ChG computes the $Y_{X'} = H (OTP || Auth_C')$
5. ChG calculates the $RV_{PG1}' = H (Yx' \oplus RV_{ChG1}) \oplus Y_1$
6. As presented in figure 54, ChG calculates Y_2' using a one-way hash function and compares the generated Y_2' to the received Y_2. If the two values are the same, ChG will accept the interaction response and authenticate the PG.; otherwise, it will drop it.

$Y_2' = H (ChG_{IDR} || TS'_{PG1} || RV'_{PG1} || RV_{ChG1.} Yx)$
If $Y_2 = Y_2'$ Then
Then ChG will accept the connection response.
Else
ChG will drop the connection response

Figure 54: Evaluate the PG authentication parameter

Once the ChG authenticates the PG, it generates a random value RV_{ChG2} and time stamp TS_{CHG2}. The ChG prepares and sends another message to the PG. The preparation process of the message includes the following steps:

1. ChG generates a new TS_{ChG2}
2. ChG computers Xcc
    a. $Xcc = H (RV'_{PG1}|| OTP)$
3. ChG selects a random parameter RV_{ChG2} and then calculates X_4
    a. $X_4 = H(Xcc) \oplus RV_{ChG2}$
4. ChG calculates X_5
    a. $X_5 = H (ChG_{IDR} || TS_{ChG2} || RV_{ChG2}|| (RV'_{PG1} || Xcc)$
5. ChG sends the message to the PG

The message includes $ChG_{IDA,}$ session ID, $TS_{ChG2,}$ X_4, and X_5, as shown in 5.3.

$$ChG \longrightarrow PG\ (ChG_{IDA}, S_{ID}, TS_{ChG2,}\ X_4, X_5) \qquad (5.3)$$

5.6.3.1.4 Step 4: Child Gateway Authentication

Once the PG receives the message, it will complete the following steps:

1. PG checks the validity of the received timestamp $| TS'_{Rec} - TS_{ChG2} | < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. PG computes $Xcc' = H\ (OTP || RV'_{PG1})$
3. PG computers $RV'_{ChG2} = H(Xcc') \oplus X_4$
4. The PG uses the $TS'_{ChG2,}$ Xcc', RV_{PG1} and, RV'_{ChG2} to generate X_5' using a one-way hash function as shown in figure 55. If the computed value is equal to the received value, the PG accepts the message and authenticates the ChG; otherwise, it will drop the connection.

$X5^{'} = H\ (ChG_{IDR} || TS'_{ChG2} || RV'_{ChG2} || RV_{PG1} || Xcc)$
If $X_5 = X_5^{'}$ Then
    The PG will accept the message
Else
    The PG will drop the connection

Figure 55: Evaluate the ChG Authentication Parameter

Once the PG accepts the message, it will generate a new timestamp TS_{PG2}. Also, the PG will generate two news tokens that the ChG can use to complete a mutual authentication with the other two ChGs. The PG will also compute a new OTP and a new alias ID for the ChG to be used for the next authentication session. The PG prepares and sends a message to the ChG. The preparation process of the message includes the following steps:

1. PG generates a new TS_{PG2}
2. PG generates two new tokens TK_{12} and $Tk_{13,}$ and then calculates and then calculates Y_3 and Y_4
    a. $Y_3 = H(OTP) \oplus TK_{12}$
    b. $Y_4 = TK_{12} \oplus TK_{13}$
3. PG calculate a new OTP
    a. $OTPx = H (OTP_{x-1} || RV_{ChG1})$
4. PG calculate a new Alias ID for the ChG
    a. $ChG_{IDAx} = H (OTP|| ChG_{IDA})$
5. PG calculates $Y5 = H (ChG_{IDR} || TS_{PG2} || TK_{12}|| TK_{13} ||OTPx|| ChG_{IDAx|})$
6. PG sends the message to the ChG

The message includes $PG_{IDA,}$ $S_{ID,}$ $TS_{PG2,}$ Y_3, Y_4, and Y_5, as shown in 5.4.

$$PG \longrightarrow ChG\ M4\ (PG_{IDA}, S_{ID,} TS_{PG2,} Y_3, Y_4, Y_5) \qquad (5.4)$$

Once the ChG receives the message, it will complete the following steps:

1. ChG checks the validity of the received timestamp $| TS'_{Rec} - TS_{CG2} | < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. ChG computes the TK_{12} and TK_{13}
    a. $TK_{12}' = H(OTP) \oplus Y_3$
    b. $TK_{13}' = TK_{12}' \oplus Y_4$
3. ChG computes the $OTP'x = H (OTP_{x-1} || RV_{ChG1})$
4. ChG computes the $ChG'_{IDAx} = H (OTP|| ChG_{IDA})$
5. The ChG uses $TS'_{PG2,}$ TK'_{12}, TK'_{13}, OTP'x and ChG'_{IDAx} to generate Y_5' using a one-way hash function as shown in figure 56. If the computed value is equal to the received value, the ChG accepts the message, saves the tokens in its database, and updates its database with the new OTP and the new Alias ID.

$Y5' = H (ChG_{IDR} || TS'_{PG2} || TK'_{12}|| TK'_{13} || OTP'_X || ChG'_{IDAx})$
If $Y_5 = Y_8'$ Then
    The ChG will accept the message and update its database
Else
    The ChG will drop the connection

Figure 56: Evaluate the PG received message

5.6.3.1.5 Key generation phase between PG and ChG

Once the mutual authentication is completed, the two sides will generate a shared secret session key (*ssk*) that will be used to encrypt all the subsequent communication between the PG and the ChG. The *ssk* will be a combination of the generated random

values as presented in 5.5. SHA 256 will be used to generate the *ssk*. AES 128 bits will be used to encrypt and decrypt the messages.

$$ssk = H\ (RV_{ChG1} || RV_{ChG2} || RV_{PG1} \| OTP) \quad (5.5)$$

5.6.3.2 Protocol 2: child gateway-child gateway authentication and key generation

This protocol presents the mutual authentication process between any two nodes that inherit the child-child relationship. The two-child gateways will complete mutual authentication and establishes a session key for data transfer. As presented in figure 57, this process starts when ChG_1 prepares the interaction request message that will be sent to the ChG_2. In the case the MG-MG mutual authentication, the two gateways must be inside the same level. Each child node knows the MSK. In the case of the intermediate gateways, all the IGs share the same MSK with their parent gateway, which is the CG. In the case of the mini gateways. The mini gateways within the same level share the MSK with their parent gateways, the IG. Also, each child gateway knows the tokens that it received from its parent gateway to use during the mutual authentication with another child gateway. Furthermore, each child gateway knows its real and alias IDs and the other child gateways' real and alias IDs that they are eligible to authenticate with them mutually. The algorithm for the proposed Child Gateway -Child Gateway mutual authentication mechanism is shown in detailed steps in figure 59. The steps involved in this process are listed below.

5.6.3.2.1 Step 1: Interaction Request

1. ChG_1 generates a new TS_{ChG_11}
2. ChG_1 calcilates Rj

a. $Rj = H (TK|| ChG\_2_{IDR})$

3. ChG_1 selects a random parameter RV_{ChG_11} and then calculates T1

   a. $T1 = RV_{ChG\_11} \oplus Rj$

4. ChG_1 computes $T2 = H (ChG\_1_{IDR} || TS_{ChG\_11} || RV_{ChG\_11} ||Rj)$

5. ChG_1 sends an interaction request message.

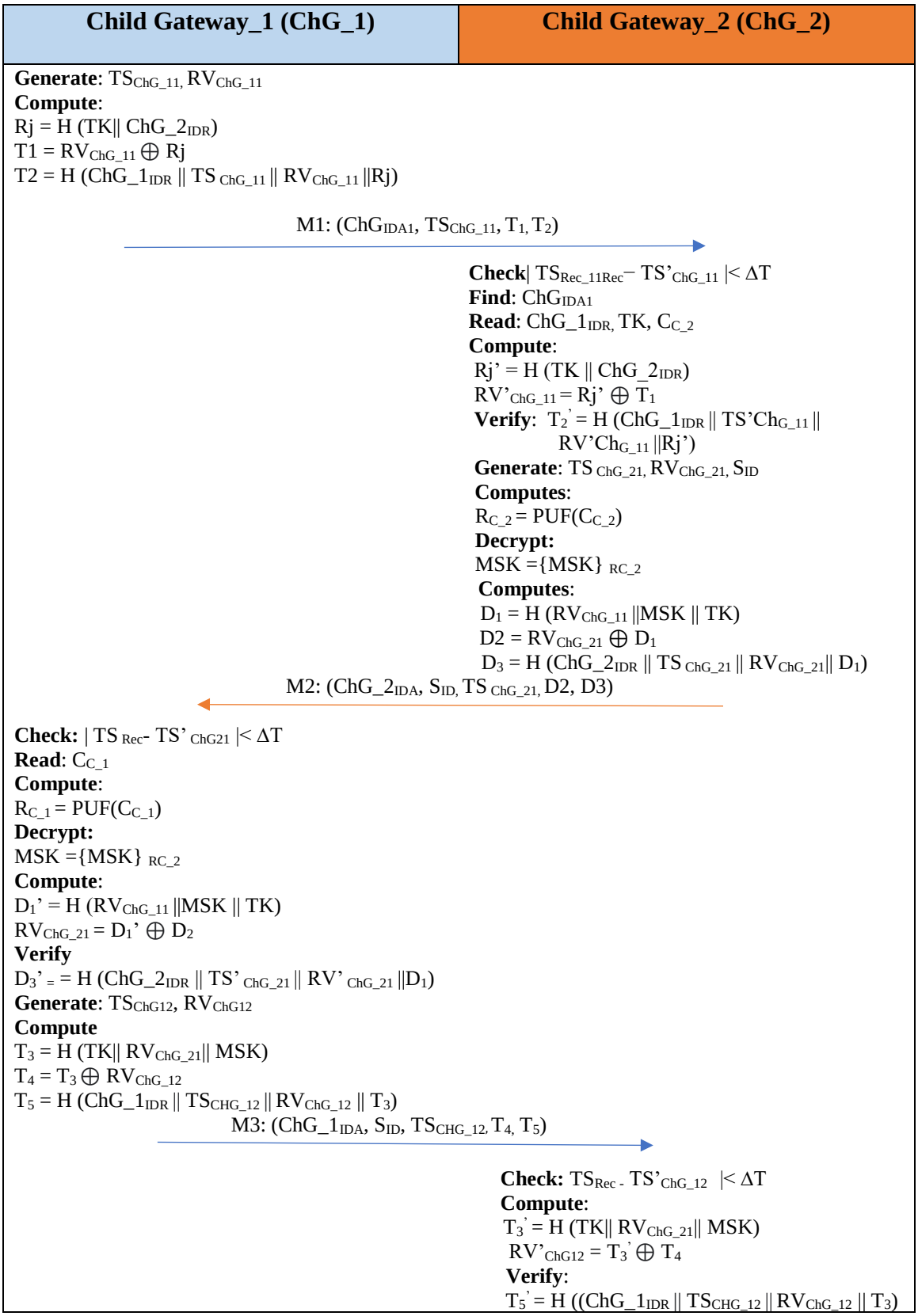


Figure 57: The authentication process between ChG_1 and ChG_2

The connection request includes ChG_1_{IDA}, TS_{ChG_11}, T_1, and T_2, as shown in 5.7.

$$ChG\_1 \longrightarrow ChG\_2\ Conn\text{-}Req\ (ChG_{IDA}, TS_{ChG\_11}, T_1, T_2) \quad (5.7)$$

Once the ChG_2 receives the interaction request, it will complete the following steps:

1. ChG_2 checks the validity of the received timestamp $| TS_{Rec\_11Rec} - TS'_{ChG\_11} | < \Delta T$. $TS_{Rec\_}$ is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. ChG_2 retrieves from its database the ChG_1_{IDR} that corresponds to its ChG_1_{IDA}. If the ChG_2 did not find the ChG_1_{IDA} in its database, it will ignore and drop the interaction request.
3. Once the ChG_2 finds the ChG_1_{IDR}, it will retrieve it along with the token (TK) of ChG_1.
4. ChG_2 computes $Rj' = H (TK \| ChG\_2_{IDR})$
5. ChG_2 computes the $RV'_{ChG\_11} = Rj' \oplus T_1$
6. The ChG_2 uses $ChG_{IDR,}$ $TS'_{ChG\_11,}$ Rj', and the RV_{ChG_11} using a one-way hash function as shown in figure 58 to verify T_2'. If the computed value is equal to the received value, the ChG_2 accepts the interaction request; otherwise, it will drop it.

$T_2^{'} = H (ChG\_1_{IDR} || TS'Ch_{G\_11} || RV'Ch_{G\_11} ||Rj')$
If $T_2 = T_2^{'}$ Then
The ChG_2 will accept the interaction request
Else
The ChG_2 will drop the interaction request

Figure 58: Evaluate the ChG interaction request parameters

Algorithm 3 The mutual authentication between IG device and the MG

Input:

First Child gateway (ChG1) device with real identity (ChG_{IDR1}), alias identity (ChG_{IDA1}), ChG_2 real identity (ChG_{IDR2}), IG alias identity (ChG_{IDR2}), master secret key (MSK), PUF Challenge (C_{ChG1}), and the TK.

Second Child gateway (ChG2) with real identity (ChG_{IDR1}), alias identity (ChG_{IDA1}), ChG_2 real identity (ChG_{IDR2}), IG alias identity (ChG_{IDR2}), master secret key (MSK), PUF Challenge (C_{ChG2}), and the TK.

Output:

Mutual authentication between the ChG1 and the ChG2

Begin

1. The ChG1 device generates a random nonce $RV_{ChG\_11\ and}$ a timestamp $TS_{ChG\_11.}$ ChG1 computes Rj = H (TK|| ChG_2_{IDR}), T1 = $RV_{ChG\_11} \oplus$ Rj, and T2 = H (ChG_1_{IDR} || TS_{ChG_11} || RV_{ChG_11} ||Rj)
2. ChG1 device sends < ChG_{IDA1}, TS_{ChG_11}, T_1, T_2 > message to the ChG2.
3. **If** (the ChG2 finds ChG_{IDA1} in its repository)
4. **then**
5. The ChG2 verifies the timestamp TS_{ChG_11} and retrieves $ChG\_1_{IDR,}$ TK, C_{C_2} from its repository to its memory.
6. The ChG2 computes Rj' = H (TK || ChG_2_{IDR}) and retrieves RV'_{ChG_11} from xoring Rj' and T_1. Then ChG2 generates $T_2^{'}$ = H (ChG_1_{IDR} || $TS'Ch_{G\_11}$ || $RV'Ch_{G\_11}$ ||Rj')
7. **If** (the calculated hash message in step 6 matches the hash message that was in step 2) **then**
8. The ChG2 generates TS_{ChG_21} and $RV_{ChG\_21,}$ S_{ID}
9. The ChG2 passes the challenge C_{C_2} to its PUF function and generates a

response $R_{C\_2.}$ Then the ChG2 decrypts the MSK using $R_{C\_2.}$

10. The ChG2 computes $D_1 = H (RV_{ChG\_11} \| MSK \| TK)$, $D2 = RV_{ChG\_21} \oplus D_1$, $D_3 = H (ChG\_2_{IDR} \| TS_{ChG\_21} \| RV_{ChG\_21} \| D_1)$
11. ChG2 device sends $< ChG\_2_{IDA}, S_{ID,} TS_{ChG\_21,} D2, D3>$ message to the ChG1.
12. **else**
13. Go to step 34.
14. **end if**
15. **else**
16. Go to step 34.
17. **end if**
18. ChG1 verifies the timestamp and retrieves C_{C_1}. The ChG1 passes the challenge C_{C_1} to its PUF function and generates a response $R_{C\_1.}$ ChG1 computes $D_1' = H (RV_{ChG\_11} \| MSK \| TK)$ and retrieves RV_{ChG_21} from xoring D2 and D_1. Then ChG1 generates $D_3' = H (ChG\_2_{IDR} \| TS'_{ChG\_21} \| RV'_{ChG\_21} \| D_1)$
19. **If** (the calculated hash message in step 18 matches the hash message that was sent in step 11)

    then
20. The authenticity of the ChG2 device is verified.
21. ChG1 generates a time stamp TS_{ChG12} and a nonce value RV_{ChG12}
22. ChG1 computes $T_3 = H (TK \| RV_{ChG\_21} \| MSK)$, $T_4 = T_3 \oplus RV_{ChG\_12}$, and $T_5 = H (ChG\_1_{IDR} \| TS_{CHG\_12} \| RV_{ChG\_12} \| T_3)$
23. ChG1 device sends $< ChG\_1_{IDA}, S_{ID}, TS_{CHG\_12,} T_{4,} T_5>$ message to the ChG2
24. **else**
25. Go to step 34.
26. **end if**
27. The ChG2 verifies the timestamp $TS_{CHG\_12.}$ It computes $T_3' = H (TK \| RV_{ChG\_21} \| MSK)$ and retrieves RV_{ChG_12} by XORing = T_3 and $T_{4.}$ Also, ChG2 generates $T_5' = H ((ChG\_1_{IDR} \| TS_{CHG\_12} \| RV_{ChG\_12} \| T_3)$
28. **If** (the calculated hash message in step 27 matches the hash message that was sent in step 23
29. **then**
30. The authenticity of the ChG1 device is verified.

31. **else**
32. Go to step 34.
33. **end if**
34. **Stop** (terminates the connection)

End

Figure 59: Algorithm 3 The mutual authentication between ChG1 device and the ChG2

5.6.3.2.2 Step 2: ChG_2 Response

Once the ChG_2 accepts the interaction request, it will retrieve its stored PUF challenge (C_{C_2}) and feed it to the PUF function to calculate the PUF response (R_{C_2}). Then ChG_2 will use R_{C_2} to decrypt the stored MSK. Also, it generates a random value RV_{ChG_21} and time stamp TS_{ChG_21}. Furthermore, the ChG_2 generates a session ID. The session ID aims to distinguish one session from the rest of the running sessions simultaneously. The ChG_2 prepares and sends a connection response to the ChG_1. The preparation process of the connection response message includes the following steps:

1. ChG_2 retrieves its C_{C_2} from its database and computes R_{C_2}
    a. $R_{C\_2} = PUF(C_{C\_2})$
2. ChG_2 decrypt the MSK using R_{C_2}
3. ChG_2 calculates D_1
    a. $D_1 = H (RV_{ChG\_11} || MSK || TK)$
4. ChG_2 generates a new $TS_{ChG\text{-}21}$
5. ChG_2 selects a random parameter RV_{ChG_21} and then calculates D_2
    a. $D2 = RV_{ChG\_21} \oplus D_1$
6. ChG_2 generates a session ID (S_{ID})

7. ChG_2 calculates $D_3 = H (ChG\_2_{IDR} \| TS_{ChG\_21} \| RV_{ChG\_21} \| D_1)$
8. ChG_2 sends the message to ChG_1

The message includes $ChG\_2_{IDA,}$ session ID (S_{ID}), D_2, and D_3, as shown in 5.6. The timestamps present information about when an event occurred, which makes this value important.

$$ChG\_2 \longrightarrow ChG\_1 (ChG\_2_{IDA}, S_{ID,} D2, D3) \qquad (5.8)$$

5.6.3.2.3 Step 3: ChG_2 Authentication

Once ChG_1 receives the connection response, it completes the following steps:

1. ChG_1 checks the validity of the received timestamp $| TS_{Rec} - TS'_{ChG21} | < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. ChG_1 retrieves its C_{C_1} from its database and computes R_{C_1}
    a. $R_{C\_1} = PUF(C_{C\_1})$
3. ChG_1 decrypt the MSK using R_{C_1}
4. ChG_1 computes $D_1' = H(RV_{ChG\_11} \| MSK \| TK)$
5. ChG_1 calculates the $RV_{ChG\_21} = D_1' \oplus D_2$
6. As presented in figure 60, ChG calculates D_3' using a one-way hash function and compares the generated $D_3'$with the received D_3. If the two values are the same, ChG_1 will authenticate ChG_2; otherwise, it will drop the message.

D_3' = = H (ChG_2_{IDR} || TS' $_{ChG\_21}$ || RV' $_{ChG\_21}$ ||D_1)
If $D_3 = D_3$' Then
Then ChG_1 will Authenticate ChG_2
Else
ChG will drop the message

Figure 60: Evaluate the ChG_2 authentication parameter

Once the ChG_1 authenticate ChG_2, it generates a random value RV_{ChG_12} and time stamp TS_{ChG_12}. The ChG_1 prepares and sends another message to the ChG_2. The preparation process of the message includes the following steps:

1. ChG_1 generates a new TS_{ChG_12}
2. ChG_1 calculates T_3
    a. $T_3 = H (TK|| RV_{ChG\_21}|| MSK)$
3. ChG_1 generate a new random value RV_{ChG_12} and computers T_4
    a. $T_4 = T_3 \oplus RV_{ChG\_12}$
4. ChG_1 calculates T_5
    a. $T_5 = H (ChG\_1_{IDR} || TS_{CHG\_12} || RV_{ChG\_12} || T_3)$
5. ChG_1 sends the message to the ChG_2

The message includes $ChG\_1_{IDA,}$ S_{ID}, TS_{ChG_21}, T_4, and T_5, as shown in 5.9.

$$ChG\_1 \longrightarrow ChG\_2 (ChG\_1_{IDA}, S_{ID}, TS_{CHG\_12} T_{4,} T_5) \quad (5.9)$$

5.6.3.2.4 Step 4: ChG_1 Authentication

Once the ChG_2 receives the message, it will complete the following steps:

1. ChG_2 checks the validity of the received timestamp $TS_{Rec} - TS'_{ChG\_12} | < \Delta T$. TS'_{ChG_12} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
2. ChG_2 computes $T_3' = H (TK || RV_{ChG\_21} || MSK)$
3. ChG_2 computers $RV'_{ChG12} = T_3' \oplus T_4$
4. The ChG_2 uses $TS'_{CHG\_12,}$ the $RV'_{ChG12,}$ and T_3' to generate T5' using a one-way hash function and XOR function as shown in figure 61. If the computed value is equal to the received value, the ChG_2 accepts the message and authenticate ChG_1; otherwise, it will drop the connection.

$T_5' = H (ChG\_1_{IDR} || TS_{CHG\_12} || RV_{ChG\_12} || T_3)$
If $T_5 = T_5'$, Then
The ChG_2 will accept the message and authenticate ChG_1
Else
The ChG_2 will drop the connection

Figure 61: Evaluate the ChG_1 authentication parameter

5.6.3.2.5 Key Generation Phase

Once the mutual authentication is completed, the two sides will generate a shared secret session key (*ssk*) that will be used to encrypt all the subsequent communication between ChG_1 and ChG_2. The *ssk* will be a combination of the generated random values and the TK as presented in 5.10. SHA 256 will be used to generate the *ssk*. AES 128 bits will be used to encrypt and decrypt the messages.

$$ssk = H (RV_{ChG\_11} || RV_{ChG\_21} || RV_{ChG\_12} || TK) \qquad (5.10)$$

5.6.3.3 Protocol 3: mini gateway–IoT node authentication and key generation

This protocol presents the mutual authentication process between a mini-gateway (MG) and an IoT node(N). The two devices will complete mutual authentication and establish a session key for data communication. As presented in figure 62, this process starts when N_i prepares the connection request message that will be sent to the MG. All IoT nodes inside the same virtual domain share a group ID (G_{ID}). Also, all IoT nodes with the same type share a typeID (T_{ID}). Each IoT node stores three authentication parameters that will be used during the authentication process. Each IoT node knows its real and alias IDs, its MG's real and alias IDs, the OTT, the three authentication parameters, and the chained hash PUF value. The steps involved in this process are listed below.

As presented in figure 62, this phase starts when N_W prepares the connection request message that will be sent to the MG. The steps involved in this process are listed below.

5.6.3.3.1 Step 1: Interaction Request

1. N generates a new TS_{N1} to avoid replay attacks
2. N computes X_1
    a. $X_1 = H(S_2 || S_3)$

IoT(N)	Mini Gateway (MG)
Generate: TS_{N1}, RV_{N1} **Compute**: $X_1 = H(S_2 \| S_3)$ $X_2 = RV_{N1} \oplus OTT$ $SN = H(N_{IDR} \| TS_{N1} \| RV_{N1} \| X_1)$	
M1: $(N_{IDA}, TS_{N1}, X_2, SN)$ →	
	Check: $\|TS_{Rec} - TS'_{N1}\| < \Delta T$ **Find**: N_{IDR} **Read**: G_{IDN}, T_{IDN}, C_{MG}, and OTT **Compute**: $R_{MG} = PUF(C_{MG})$ $S'_1 = H((N_{IDR} \| MG_{IDR} \| R_{MG})$ $S'_2 = H(S'_1 \| MG_{IDR} \| G_{IDN})$ $S'_3 = H(S_2 \| MG_{IDR} \| T_{IDN})$ $X_1' = H(S_2' \| S_3')$ $RV'_{N1} = OTT' \oplus X_2$ **Verify**: $SN' = H(N_{IDR} \| TS'_{N1,} \| RV'_{N1} \| X_1)$ **Generate**: $TS_{MG1,}$ $RV_{1MG,}$ S_{ID} **Computes**: $Y_1 = H(S'_1 \| S'_2)$ $Y_2 = RV_{MG1} \oplus OTT$ $C' = C \oplus H(RV_{MG1})$ $Y_3 = H(MG_{IDR} \| TS_{MG1} \| RV_{N1} \| RV_{MG1} \| C \| Y_1)$
← M2: $(MG_{IDA}, S_{ID,} TS_{MG1,} Y_2, C', Y_3)$	
Check: $\|TS'_{Rec} - TS'_{MG1}\| < \Delta T$ **Compute**: $Y_1' = H(S'_1 \| S'_2)$ $RV'_{MG1} = Y_2 \oplus OTT$ $C_X = C' \oplus H(RV'_{MG1})$ **Verify** $Y3' = H(MG_{IDR} \| TS_{MG1} \| RV_{N1} \| RV_{MG1} \| C \| Y_1)$ **Generate**: $TS_{N2,}$ RV_{N2} **Read**: CHX^i **Compute** $Ri = PUF(C')$ $CHX^{i+1} = H(R^i \| CHX^i)$ $C^{i+1} = H(RV'_{MG1} \| RV_{N1})$ $R_{i+1} = PUF(C^{i+1})$ $X_3 = RV_{N2} \oplus CHX^{i+1}$ $Rx = H(RV_{N2} \| CHX^{i+1}) \oplus R^{i+1}$ $OTT_{new} = H(OTP_{new-1} \| C' \| R_i)$ $N_{IDAnew} = H(RV_{MG1} \| NW_{IDAnew-1})$ $X_4 = H(N_{IDR} \| TS_{N2} \| RV_{N2} \| CHX^{i+1} \| R^{i+1} \| C^{i+1} \| OTT_{new} \| N_{IDAnew})$	
M3: $(N_{IDA,} TS_{N2}, X_{3,} Rx, X_4)$ →	
	Check: $\|TS'_{Rec} - TS'_{N2}\| < \Delta T$

Read: R'_i and the CHX'_i
Compute:
$CHX'^{i+1} = H(R'_i \| CHX'i)$
$C'^{i+1} = H(RV_{MG1} \| RV'_{N1})$
$RV'_{N2} = X_3 \oplus CHX^{i+1}$
$R^{i+1'} = Rx \oplus H(RV'_{N2} \| CHX^{i+1})$
$OTT_{new} = H(OTP_{new-1} \| C' \| R_i)$
$N_{IDAnew} = H(RV_{MG1} \| NW_{IDAnew-1})$
Verify:
$X_4' = X_4 = H(N_{IDR} \| TS'_{N2} \| RV'_{N2} \| CHX'^{i+1} \| R'x \| C'^{i+1} \| OTT_{new} \| N_{IDAnew})$
Store: CHX^{i+1}, OTT_{new}, N_{IDAnew}, C'^{i+1}, R^{i+1}

Figure 62: MG-N mutual authentication process

3. N selects a random parameter RV_{N1} and then calculates X_2
4. $X_2 = RV_{N1} \oplus OTT$
5. N computes $SN = H(N_{IDR} \| TS_{N1} \| RV_{N1} \| X_1)$
6. N sends a connection request message.

The connection request includes N_{IDA}, TS_{N1}, X_2, and SN, as shown in 5.11.

$$N \longrightarrow MG\ (N_{IDA}, TSN_1, X_2, SN) \qquad (5.11)$$

5.6.3.3.2 Step 2: mini-gateway response

Once the MG receives the connection request, it will complete the following steps:

1. MG checks the validity of the received timestamp $|TS_{Rec} - TS'_{N1}| < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.
2. MG retrieves from its database the N_{IDR} that corresponds to its N_{IDA}. If the MG did not find the N_{IDA} in its database, it will ignore and drop the connection request.

3. Once the MG finds the N_{IDR}, it will retrieve it along with the G_{IDN}, T_{IDN}, C_{MG}, and OTT of N.

4. MG passes the challenge C_{MG} to its PUF function and generates a response R_{MG}

    a. $R_{MG} = PUF(C_{MG})$

5. MG uses the generated R_{MG} to compute $S_{1'}$ by applying a one-way hash function. Based on the calculated $Auth_{1'}$, the IG calculates $Auth_2'$ as presented in 5.12. Also, the MG calculates S_2' and S_3'as presented in 5.13 and 5.14.

$$S'_1 = H ((N_{IDR} || MG_{IDR} || R_{MG}) \quad (5.12)$$

$$S'_2 = H (S'_1 || MG_{IDR} || G_{IDN}) \quad (5.13)$$

$$S'_3 = H (S_{2'} || MG_{IDR} || T_{IDN}) \quad (5.14)$$

6. MG computes $X_1^{'} = H (S_2^{'} || S_3^{'})$

7. MG computes the $RV'_{N1} = OTT \oplus X_2$

8. The MG uses N_{IDR}, TS_{N1}, RV_{N1}, and X_1 to generate using a one-way hash function, as shown in figure 63. If the computed value is equal to the received value, the MG accepts the connection request; otherwise, it will drop it.

$SN' = H (N_{IDR} || TS'_{N1,} || RV'_{N1} || X_1)$
If SN = SN" Then
    The MG will accept the connection request
Else
    The MG will drop the connection request

Figure 63: Evaluate the IoT node interaction request parameter

Once the MG accepts the message, it generates a timestamp and a random value RV_{MG}. Also, the MG generates a session ID. The session ID aims to distinguish one

session from the rest of the running sessions simultaneously. Furthermore, the MG will retrieve a challenge(C) from its database. The MG prepares and sends a connection response to the IoT node. The preparation process of the connection response message includes the following steps:

1. MG calculates Y_1
    a. $Y_1 = H(S'_1 || S'_2)$
2. MG selects a random parameter RV_{1MG} and then calculates Y_2
    a. $Y_2 = RV_{MG1} \oplus OTT$
3. MG generates a new TS_{MG1}
4. MG generates a session ID(S_{ID})
5. $C' = C \oplus H(RV_{MG1})$
6. MG calculates Y_3

    $Y_3 = H(MG_{IDR} || TS_{MG1} || RV_{N1} || RV_{MG1} || C || Y_1)$
7. MG sends a connection response message

The connection response includes $MG_{IDA,}$ session ID(S_{ID}), TS_{MG1}, Y_2, C', and Y_3, as shown in 5.15.

$$MG \longrightarrow N_W \text{ Conn-Res } (MG_{IDA}, S_{ID}\, TS_{MG1}, Y_2, C', Y_3) \qquad (5.15)$$

5.6.3.3.3 Step 3: mini-gateway authentication

Once N receives the connection response, it completes the following steps:

1. N checks the validity of the received timestamp $|TS'_{Rec} - TS'_{MG1}| < \Delta T$. S'_{MGRec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and when the message is received is higher than the expected maximum transmission delay.
2. N calculates $Y_1' = H(S'_1 || S'_2)$
3. N calculates the $RV'_{MG1} = Y_2 \oplus OTT$
4. N calculates the $C_X = C' \oplus H(RV'_{MG1})$
5. As presented in figure 64, N calculates Y_3' using a one-way hash function and compares the generated Y_3' with the received Y_3. If the two values are the same, N will authenticate MG; otherwise, it will drop it.

$Y3' = H(MG_{IDR} || TS_{MG1} || RV_{N1} || RV_{MG1} || C || Y_1)$

If $Y3' = Y3$ Then
Then N will accept the connection response and authenticate MG
Else
N will drop the connection response

Figure 64: Evaluate the MG authentication parameter

Once N accepts the received message and authenticates MG, it will complete the following steps

1. N will input the received challenge Cx to its PUF and obtains the response Ri.
    a. $Ri = PUF(C')$
2. N will retrieve the chained hash response $s(CHX^i)$ from its memory and calculate the new value of the chained has responses (CHX^{i+1}) as follows

a. $CHX^{i+1} = H (R^i || CHX^i)$

b. N will compute a new challenge (C^{i+1}). Then N will input the C^{i+1} into a PUF function to obtains a new CRP for the next authentication session where:

- $C^{i+1} = H (RV'_{MG1} || RV_{N1})$
- $R_{i+1} = PUF (C^{i+1})$

3. N generates a new TS_{N2} to avoid replay attacks
4. N selects a random parameter RV_{N2} and then calculates X_3

   a. $X_3 = RV_{N2} \oplus CHX^{i+1}$

5. N calculates $Rx = H (RV_{N2} || CHX^{i+1}) \oplus R^{i+1}$
6. N generate a new OTT_{new}

   a. $OTT_{new} = H (OTT_{new-1} || C' || R_i)$

7. N calculate a new Alias ID for the N

   a. $N_{IDAnew} = H (RV_{MG1} || NW_{IDAnew-1})$

8. N calculate X_4

   a. $X_4 = H (N_{IDR} || TS_{N2} || RV_{N2} || CHX^{i+1} || R^{i+1} || C^{i+1} || OTT_{new} || N_{IDAnew})$

9. N sends the message to the MG

The message includes $N_{IDA,}$ $S_{ID,}$ TS'_{N2}, $RV'_{N2,}$ $X_{3,}$ Rx, and $X_{4,}$ as shown in 5.16.

$$N \longrightarrow MG (N_{IDA,} TS_{N2}, X_{3,} Rx, X_4) \qquad (5.16)$$

5.6.3.3.4 Step 4: IoT node authentication

Once MG receives the message, it completes the following steps:

1. MG checks the validity of the received timestamp $|TS'_{Rec} - TS'_{N2}| < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.
2. MG will retrieve the R'_i and the CHX'_i from its databased and computes CHX'^{i+1}
    a. $CHX'^{i+1} = H(R'_i \,||\, CHX'i)$
3. MG calculates the new challenge $C'^{i+1} = H(RV_{MG1} || RV'_{N1})$
4. MG calculates the $RV'_{N2} = X_3 \oplus CHX^{i+1}$
5. MG computes the $R^{i+1'} = Rx \oplus H(RV'_{N2} || CHX^{i+1})$
6. MG computes $OTT_{new} = H(OTT_{new-1} || C' \,||\, R_i)$
7. MG computes $N_{IDAnew} = H(RV_{MG1} || NW_{IDAnew-1})$

As presented in figure 65, MG calculates X_4' using a one-way hash function and compares the generated X_4' with the received X_4. MG will authenticate N; otherwise, it will drop it if the two values are the same.

$X_4' = X_4 = H(N_{IDR} || TS'_{N2} || RV'_{N2} \,||\, CHX'^{i+1} \,||\, R'x \,||\, C'^{i+1} || OTT_{new} \,||\, N_{IDAnew})$
If $X_4' = X_4$
Then
Then MG will authenticate the N.
Else
MG will drop the connection response

Figure 65: Evaluate the IG authentication parameter

After authenticating the N, the MG will store the new CRP (C'^{i+1}, R'^{i+1}) in its database for the next authentication session. Also, The MG will override the old CHX^i with the new chained hash response value CHX^{i+1}. Then the MG will store the new OTT and the new alias ID for the N. The algorithm for the proposed IoT- mini gateway mutual authentication mechanism is shown in detailed steps in figure 66.

Algorithm 4 The mutual authentication between IoT device and the MG

Input:

An IoT device with real identity (N_{IDR}), alias identity (N_{IDA}), MG real identity (MG_{IDR}), MG alias identity (IG_{IDR}), Authentication parameters (S_1, S_2, and S_3), and one-time token (OTT)

MG with real identity (N_{IDR}), alias identity (N_{IDA}), MG real identity (MG_{IDR}), MG alias identity (MG_{IDR}), the virtual domain Group ID(G_{ID}) and type ID (T_{ID}) of the IoT node, PUF challenge (C_{MG}), and one-time token (OTT)

Output:

Mutual authentication between the IoT device(N) and the MG

Begin

1. The IoT device generates a random nonce RV_{N1}, a timestamp $TS_{N1,}$ computes the $X_1 = H(S_2 \| S_3)$, $X_2 = RV_{N1} \oplus OTT$, and $SN = H(N_{IDR} \| TS_{N1} \| RV_{N1} \| X_1)$
2. IoT device sends (N_{IDA}, TS_{N1}, X_2, SN) message to the MG.
3. **If** (the MG finds N_{IDA} in its repository)
4. **then**
5. MG verifies the timestamp TS_{N1} and retrieves N_{IDR}, OTT, G_{ID}, CM_G, and T_{ID} that belongs to the N_{IDA} from its repository to its memory.
6. The MG passes the challenge C_{MG} to its PUF function and generates a response $_{MG.}$ The MG calculates S_1', S_2', and S_3'. Then the MG computes $X_{1'}$, retrieves the RV_{N1} from XORing OTT and NX_2 and finally calculates $SN' = H(N_{IDR} \| TS_{N1} \| RV_{N1} \| X_1)$ message
7. **If** (the calculated hash message in step 7 matches the hash message that was

sent in step 2)

8. **then**

9. MG generates a timestamp TS_{MG1}, generates a random nonce RV_{MG1},

10. retrieves the challenge (Ci) from its database. Then, MG calculates

$Y_1 = H(S'_1 || S'_2)$, $Y_2 = RV_{MG1} \oplus OTT$, $C' = C \oplus H(RV_{MG1})$ and

$Y_3 = H(MG_{IDR} || TS_{MG1} || RV_{N1} || RV_{MG1} || C || Y_1)$

11. The MG sends $< MG_{IDA}, S_{ID}, TS_{MG1}, Y_2, C', Y_3 >$ message to N.

12. **else**

13. Go to step 34.

14. **end if**

15. **else**

16. Go to step 34.

17. **end if**

18. N verifies the time stamp, calculates $Y_1' = H(S'_1 || S'_2)$ and retrieves RV_{MG1} from xoring OTT and Y_2 and C from xoring C' and $H(RV_{MG1})$. Then, N generates $Y3' = H(MG_{IDR} || TS_{MG1} || RV_{N1} || RV_{MG1} || C || Y_1)$.

Also, N computes C_1 from CX. N applies the PUF function to calculate R1 for C1.

19. **If** (the calculated hash message in step 18 matches the hash message that was sent in step 11)

20. **then**

21. **The authenticity of the MG is verified**

22. N generates a timestamp TS_{N2}, a random nonce RV_{N2}, and retrieves CHX^i. N computes $Ri = PUF(C')$, $CHX^{i+1} = H(R^i || CHX^i)$, $C^{i+1} = H(RV'_{MG1} || RV_{N1})$, $R_{i+1} = PUF(C^{i+1})$, $X_3 = RV_{N2} \oplus CHX^{i+1}$, $Rx = H(RV_{N2} || CHX^{i+1}) \oplus R^{i+1}$, $OTT_{new} = H(OTP_{new-1} || C' || R_i)$, $N_{IDAnew} = H(RV_{MG1} || NW_{IDAnew-1})$, and $X_4 = H(N_{IDR} || TS_{N2} || RV_{N2} || CHX^{i+1} || R^{i+1} || C^{i+1} || OTT_{new} || N_{IDAnew})$

23. N sends $< N_{IDA}, TS_{N2}, X_3, Rx, X_4 >$ message to MG.

24. **else**

25. Go to step 34.

26. **end if**

27. The MG verifies the TSN2 and retrieves R'_i and the CHX'_i. N computes $CHX'^{i+1} = H(R'_i || CHX'i)$, $C'^{i+1} = H(RV_{MG1} || RV'_{N1})$, $RV'_{N2} = X_3 \oplus CHX^{i+1}$, $R^{i+1'} = Rx \oplus H(RV'_{N2} || CHX^{i+1})$, $OTT_{new} = H(OTP_{new-1} || C' || R_i$, $N_{IDAnew} = H(RV_{MG1} ||$

$NW_{IDAnew-1}$). Then N generates $X_4' = X_4 = H (N_{IDR} || TS'_{N2} || RV'_{N2} || CHX'^{i+1} || R'x || C'^{i+1} || OTT_{new} || N_{IDAnew})$

28. **If** (the calculated hash message in step 27 matches the hash message that was sent in step 23)
29. **then**
30. **The authenticity of the IoT device is verified.**
31. **else**
32. Go to step 34.
33. **End if**
34. **Stop** (terminates the connection)
35. **End**

Figure 66: Algorithm 4 the mutual authentication between IoT device and the MG

5.6.3.3.5 Key generation phase

Once the mutual authentication is completed, the two sides will generate a shared secret session key (SK) that will be used to encrypt all the subsequent communication between N and MG. The two sides will generate the *ssk* locally by hashing a combination of the PUF responses and the generated random values, as presented in 5.17. The uniqueness of the PUF responses ensures the uniqueness of the generated *ssk*.

$$ssk = H (RV_{N1} || R_i || RV_{N2} || RV'_{MG1}) \qquad (5.17)$$

5.7 Evaluation Process

The following section presents a detailed security and performance analysis of the proposed scheme and compares the analysis results with other related schemes presented in the literature.

5.7.1 Security validation of the proposed scheme

In this section, we conduct both formal and informal security analysis. The formal evaluation uses two different approaches. The first formal evaluation approach is simulation-based using AVISPA tool to ensure that the proposed scheme is secure against active and passive attacks such as replay attacks and man-in-the-middle attacks. The second evaluation approach theorem proving-based to perform the logical verification using BAN logic to confirm that the authenticated participants share the secret parameters securely in the proposed protocols. The BAN logic is used as an illustrative method of the essential concepts of an authentication protocol and as a basic verification tool for a security protocol. After completing the formal evaluation, we examine the proposed protocols against the well-known attacks and prove that the proposed protocols satisfy the main security properties.

5.7.1.1 Formal Security Evaluation

5.7.1.1.1 Security Verification Using AVISPA

5.7.1.1.1.1 Protocol 1: parent gateway-child gateway mutual authentication

The abstract notations used to describe the authentication protocol and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 16.

Table 16: Abstract notation and AVISPA HLPSL scripting variables/functions for protocol specification

Notation	**Description**
PG	Parent gateway
ChG	Child gatetway

ChG_{IDR}	CIDR
ChG_{IDA}	CIDA
PG_{IDR}	PIDR
PG_{IDA}	PIDA
MSK	MSK
OTP	OTP
Xc	Xc
RV_{ChG1}	RVCone
X1	Xone
X2	Xtwo
Yx	Yx
Y1	Yone
Y2	Ytwo
RV_{PG1}	RVPone
Xcc	Xcc
X3	Xthree
X4	Xfour

RV_{ChG2}	RVCtwo
TK_{11}, TK_{12}	TKone, TKtwo
YxP	YxP
Y3	Ythree
Y4	Yfour
YP	YP
OTPx	OTPnew
ChG_{IDAx}	CIDAnew
Sk	SK
TS_{PG}	TSoneP,TstwoP
TS_{ChG}	TsoneC,TstwoC
$\oplus$	XOR
H	H

The main goals of the simulation are as follows:

13. **Goal 1**: The secrecy_of secRVCone represents that RVCone is kept secret to (C, P) only.
14. **Goal 2**: The secrecy_of secRVCtwo represents that RVCtwo is kept secret to (C, P) only.

15. **Goal 3**: The secrecy_of secRVPone represents that RVPone is kept secret to (C, P) only
16. **Goal 4**: The secrecy_of secOTP represents that OTP is kept secret to (C, P) only
17. **Goal 5**: The secrecy_of secTKone represents that TKone is kept secret to (C, P) only.
18. **Goal 6**: The secrecy_of secTKtwo represents that TKtwo is kept secret to (C, P) only.
19. **Goal 7**: The secrecy_of secOTPnew represents that the secret key secOTPnew is permanently kept secret, known to only (C and P).
20. **Goal 8**: The secrecy_of secCIDR represents that the NIDR is permanently kept secret, known to only (C and P).
21. **Goal 9**: The secrecy_of secPIDR represents that the IGIDR is permanently kept secret, known to only (C and P).
22. **Goal 10**: The secrecy_of secCIDAnew represents that CIDAnew is permanently kept secret, known to only (C and P).
23. **Authentication Property 1**: The authentication_on RVPone represents that P generates RVPone. If N securely receives RVPone through a message, it authenticates P.
24. **Authentication Property 2**: The authentication_on rvcone represents that C generates RVCone. If P securely receives rvpone through a message, it authenticates C.

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with their message exchanges. There are four defined roles: that are: (1) role_C that is played by the child gateway; (2) role _P that is played by the parent gateway; (3) session, where the session role and all its declarations are defined, (4) and environment, which instantiates all agents, variables, and functions.

Figure B_1 in Appendix B presents the role of the child gateway that is played by C. C is aware of the C and P agents in the protocol, its alias identity (CIDA), and the P alias identity (PIDA). Also, it is aware of its own real identity (CIDR) and P real identity (PIDR) that should be kept secured. Furthermore, C is aware of MSK and OTP that are generated by the parent gateway, its PUF challenge (CPC), the parent gateway authentication parameter (PAU), the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model. At the first state "state 2," C receives a start message "Rcv(start)" as a signal to begin the protocol run. The keyword 'Played_by C' denotes that the role of the child gateway is played by agent C. All employed local variables in this role are defined under the local section. C generates new random values (TSoneC and RVCone) and computes Xc, Xone, and Xtwo. The computation process of the Xc, Xone, and Xtwo follows the presented protocol description. C sends CIDA, TSoneC, Xone, Xtwo to P.

At the second transition, "State 4", C receives the (PIDA, SIDNIG, TSoneP, Yone, Ytwo) message from the P. The computation of Yx, Yone, and Ytwo follows the description of the protocol. At this transition, if the Ytwo appears as expected by C and C correctly verified RVPone, then C authenticates P. Then C generates new random values

(TStwoC and RVCtwo) and computes Xcc, Xthree, and Xfour. The computation process of the Xcc, Xthree, and Xfour follows the presented protocol description. C sends CIDA, TStwoC, Xthree, Xfour to P.

At the third transition, “State 6”, C received the (PIDA, SIDNIG, TStwoP, Ythree, Yfour, YPx) from P. C calculates and verifies the received Tokens, new OTP, and New Alias ID. After completing the verification process, C will store the tokens, its new alias ID and the new NOTP on its own memory. C will use the new NOPT during the next authentication session. The “end role” at the end of the C role denotes the end of the role Node played by C.

Figure B_2 in Appendix B shows the role of the Parent Gateway played by P. is aware of all agents in the protocol (C and P), its alias and real identities (PIDA, PIDR), the Alias and real identities (CIDA, CIDR) of the C node. P knows the OTP, the MSK, its PUF challenge (CPP), C authentication parameter (CAU), the hash function H(·), and the send/receive channels Snd/Rcv.

At the first transition, “State 1”, P receives the (CNIDA, TSoneC, Xone, Xtwo) message which was sent by C. The extraction and computation of RVCone, Xc, Xone, and Xtwo follow the presented scheme's description. Once the P verifies Xtwo, it generates a fresh value RVPone and a TSoneP and computes Yx, Yone, and Ytwo. Then the P sends to C (PIDA, SIDNID, TSoneP, Yone, Ytwo). The computation of Yone and Ytwo follows the description of the introduced scheme.

At the second transition, “State 3”, P received the (CIDA, SIDNIG, TStwoC,Xthree, Xfour) from the C. The computation of the Xthree and Xfour followed

the description of the presented protocol. At this transition, if the Xfour is proved to be valid by P, P will authenticate C. Consequently, P will generate a new alias ID for C and a new OTP. The computations of NnewIDA and OPTnew follows the description of the presented protocol. Also, the P generates two tokens that act as a one-time secret code that C can use to authenticate its peer Cs. P will then send to C (PIDA, SIDNIG, TStwoP, Ythree, Yfour, and YP). The "end role" at the end of the P role denotes the end of the role Node played by P.

Figure B_3 in Appendix B demonstrates the session role where all the two agents' roles are invoked and all the session parameters are defined. Both the parent gateway and the child gateway roles are invoked with C and P as agents. CIDA, PIDA, CIDR, PIDR, OTP, MSK, CPC, PAU, CPP, CAU are predefined as constants. H is defined as the hash function. SND1, RCV1, SND2, and RCV2 are defined as send and receive channels for C and P. The "end role" at the end of the session role indicates the end of this role.

Figure B_4 in Appendix B shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants, c, and p are instantiated as agents representing agents C and P. The constants cida, pida, cidr, pidr, otp, msk, cpp,cau,cpc, and pau instantiates CIDA, PIDA, CIDR, PIDR, OTP, MSK, CPP,CAU,CPC,and PAU respectively. The function h instantiates the hash function H. The protocol identifiers are secNIDR, secPIDR, secOTP, secMSK, secRVoneC, secRVtwoC, secRVoneP, secTKone, secTKtwo, secOTPnew,secCIDAnew, secYxP, rvcone and rvpone are also instantiated and defined. In the intruder knowledge section, all relevant values that the intruder is assumed to know before the execution are provided. The attacker is assumed to know c and p. He/ she is also assumed to know the

hash h. The session is instantiated with c, p, cida, cidr, pida,pidr,otp, msk, cpp,cau,cpc,pau and h instances in the composition section. The “end role” at the end of the environment role denotes the end of this role.

Figure B_5 in Appendix B presents the simulation goals which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator is dictated to check the secrecy CIDR, PIDR, OTP,MSK,RVoneC,RvtwoC, RVoneP, TKone, TKtwo, OTPnew, and CIDAnew at different states using secrecy_of secCIDR’, ‘secrecy_of secPIDR’, ‘secrecy_of secRVoneC’, ‘secrecy_of secRVtwoC’, ‘secrecy_of secRVoneP’, ‘secrecy_of secOTP’, ‘secrecy_of secTKone, secrecy_of secTKtwo, ‘secrecy_of secOTPnew, and ‘secrecy_of secCIDAnew’. The authentication is checked using ‘authentication_on rvonec and ‘authentication_on rvonep’. The “end role” at the end of the goal section denotes the end of this role.

5.7.1.1.1.1.1 Protocol 1: parent gateway-child gateway simulation results

A security protocol animator (SPAN) is used to build a Message Sequence Chart (MSC) of the protocol execution from the outlined HLPSL specification. Moreover, SPAN automatically creates attacks on HLPSL specifications using the well-known “Dolev-Yao” intruder model. The SPAN protocol simulation’s MSC corresponding to the HLPSL specification is shown in figure 67.

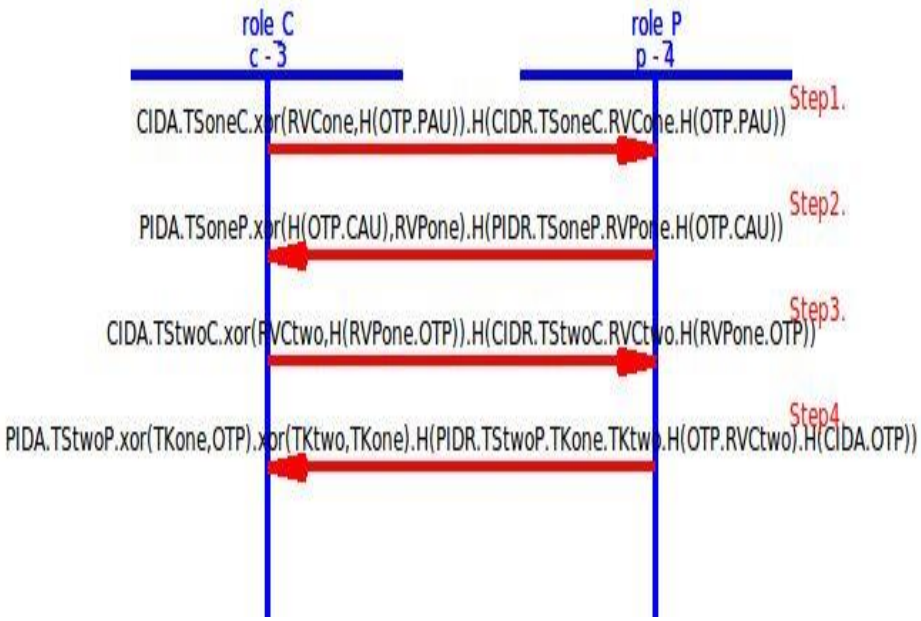


Figure 67: Snapshot of the protocol simulation in AVISPA

In the AVISPA tool, the security properties such as authentication, integrity, and secrecy are specified in a separate section. Therefore, when SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN generates the attack trace if any attack is found, and it will consider the protocol unsafe. The presented protocol's simulation results are achieved by the OFMC back-end checker and the CL-AtSe back-end checker. Figure 68 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure 69 presents the OFMC back-end checker report shows that the protocol is SAFE, thus meeting the defined security goals. In summary, we can conclude that the proposed protocol is secure.

```
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL
PROTOCOL
 /home/span/span/testsuite/results/Protocol_2_P_C_final_m.if
GOAL
 As Specified
BACKEND
 CL-AtSe
STATISTICS
 Analysed   : 1 states
 Reachable  : 1 states
 Translation: 0.06 seconds
 Computation: 0.00 seconds
```

Figure 68: CL-AtSe summary report

```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Protocol_2_P_C_final_m.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.14s
 visitedNodes: 3 nodes
 depth: 2 plies
```

Figure 69: OFMC summary report

5.7.1.1.1.2 Protocol 2: child gateway-child gateway mutual authentication

The abstract notations used to describe the authentication protocol and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 17.

Table 17: Abstract notation and AVISPA HLPSL scripting variables/functions for protocol specification

Notation	**Description**
Cone	First child gateway
Ctwo	Second child gateway
ChG_1_{IDR}	ConeIDR
ChG_1_{IDA}	ConeIDA
ChG_2_{IDR}	CtwoIDR
ChG_2_{IDA}	CtwoIDA
MSK	MSK
TK	TK
Rj	Rj
RV_{ChG11}	RCone
T1	Tone
T2	Ttwo
T3	Tthree
D1	Done
D2	Dtwo

D3	Dthree
$RVCh_{G21}$	RCCone
T4	Tfour
T5	Tfive
RV_{ChG21}	RCCone
TK	TK
TS_{ChG11}, TS_{ChG12}	TSoCone,TStCone
TS_{ChG21}	TSoCtwo
$\oplus$	XOR
H	H

The main goals of the simulation are as follows:

1. **Goal 1**: The secrecy_of secRCone represents that RCone is kept secret to (Cone, Ctwo) only.
2. **Goal 2**: The secrecy_of secRCCone represents that RCCone is kept secret to (Cone, Ctwo) only.
3. **Goal 3**: The secrecy_of secRCtwo represents that RCtwo is kept secret to ((Cone, Ctwo) only

4. **Goal 4**: The secrecy_of secTK represents that TK is kept secret to ((Cone, Ctwo) only
5. **Goal 5**: The secrecy_of secMSK represents that MSK is kept secret to ((Cone, Ctwo) only.
6. **Goal 6**: The secrecy_of secConeIDR represents that the secConeIDR is permanently kept secret, known to only (Cone and Ctwo).
7. **Goal 7**: The secrecy_of secCtwoIDR represents that the secCtwoIDR is permanently kept secret, known to only (Cone and Ctwo).
8. **Authentication Property 1**: The authentication_on RCone represents that ChG_1 generates RCone. If ChG_2 proves that it securely receives and successfully verifies RCone, ChG_1 authenticate ChG_2.
9. **Authentication Property 2**: The authentication_on RCCone represents that ChG_2 generates RCCone. If ChG_1 proves that it securely receives and successfully verifies RCone, ChG_2 authenticate ChG_1

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with their message exchanges. There are four defined roles: that are: (1) role_Cone that is played by the first child gateway; (2) role_Ctwo that is played by the second child gateway; (3) session, where the session role and all its declarations are defined, (4) and environment, which instantiates all agents, variables, and functions.

As presented figure B_6 in Appendix B, the role of the first child gateway is played by C_{one}. C_{one} is aware of the C_{one} and C_{two} agents in the protocol, and its alias

identity (ConeIDA), and the C_{two} alias identity (CtwoIDA). Also, it is aware of its own real identity (ConeIDR) and the C_{two} real identity (ConeIDR) that should be kept secured. Furthermore, C_{one} is aware of the master secret key (MSK) and one-time token (TK) that are generated by the parent gateway, the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model.

At the first state "state 2," Cone receives a start message "Rcv(start)" as a signal to begin the protocol run. The keyword 'Played_by C_{one}' denotes that agent C_{one} plays the role of the child gateway. All employed local variables in this role are defined under the local section. Cone generates new random values (TSoCone and RCone) and computes Rj, Tone, and Ttwo. The computation process of the Rj, Tone, and Ttwo follows the presented protocol description. C_{one} sends (ConeIDA, Tone, Ttwo) to the C_{two}.

At the second transition, "State 4", C_{one} receives the (CtwoIDA, SIDNIG, Dtwo, Dthree) message from the C_{two}. At this transition, if the Dtwo appears as expected by C_{one} and C_{one} correctly verified Dtwo and RCone, then C_{one} authenticates C_{two}. Then C_{one} generates new random values (TStCone and RCtwo) and computes Tthree, Tfour, and Tfive. The computation process of the Tthree, Tfour, and Tfive follows the presented protocol description. C_{one} sends (ConeIDA, Tfour, Tfive) to the C_{two}. The "end role" at the end of the C_{one} role denotes the end of the role Node played by C_{one}.

Figure B_7 in Appendix B shows the role of the second child Gateway played by C_{two}. C_{two} is aware of all agents in the protocol (C_{one} and C_{two}), its alias and real identities (CtwoIDA, CtwoIDR), the Alias and real identities (ConeIDA, ConeIDR) of the C_{one}

node. C_{two} knows the TK, the MSK, the hash function H (·), and the send/receive channels Snd/Rcv.

At the first transition, "State 1", C_{two} receives the (ConeIDA, Tone, Ttwo) message which was sent by C_{one}. Once the C_{two} verifies Ttwo, it generates a fresh value RCCone and a TSoCtwo and computes Done, Dtwo, and Dthree. Then the C_{two} will send to C_{one} (CtwoIDA, SIDNID, Dtwo, Dthree). At this transition, if the Tfive appears as expected by C_{two} and C_{two} correctly verified Tfive and RCCone, then C_{two} authenticates C_{one}. The "end role" at the end of the C_{two} role denotes the end of the role Node played by C_{two}

Figure B_8 in Appendix B shows the session role where the two agents' roles are invoked, and all the session parameters are defined. First, all known constant parameters and their declarations are presented. The predefined constants are ConeIDR, CtwoIDA, CtwoIDR, ConeIDA, TK, and MSK as predefined constants, and H as the hash function. A send and receive channel is assigned to each agent under the local section. The "end role" at the end of the session role indicates the end of this role.

Figure B_9 in Appendix B shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants, cone, and ctwo are instantiated as agents representing agents Cone and Ctwo. The constants coneida, coneidr, ctwoida, ctwoidr, tk and msk instantiates ConeIDA, ConeCIDR, CtwoIDA, CtwoIDR, TK, and MSK, respectively. The function h instantiates the hash function H. The protocol identifiers are secConeIDR, secCtwoIDR, secTK, secMSK, secRCone, secRCCone, secRCtwo, rcone and rccone are also instantiated and defined. In the intruder knowledge section, all relevant values that the intruder is assumed

to know before the execution are provided. The attacker is assumed to know cone and ctwo. He/ she is also assumed to know the hash h. The session is instantiated with cone, ctwo, coneida, coneidr, ctwoida, ctwoidr,tk, msk, and h instances in the composition section. The “end role” at the end of the environment role denotes the end of this role.

Figure B_10 in Appendix B shows the simulation goals which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator is dictated to check the secrecy ConeIDR, CtwoIDR, TK, MSK, RCCone, RCtwo, and RCCone, at different states using secrecy_of secConeIDR’, ‘secrecy_of secCtwoIDR’, ‘secrecy_of secRCone’, ‘secrecy_of secRCtwo’, ‘secrecy_of secRCCone’, ‘secrecy_of secMSK’, and ‘secrecy_of secTK. The authentication is checked using ‘authentication_on rcone’ and ‘authentication_on rccone’. The “end role” at the end of the goal section denotes the end of this role.

5.7.1.1.1.2.1 Simulation results

The SPAN protocol simulation’s MSC corresponding to our HLPSL specification is shown in Figure 70. In the AVISPA tool, security properties such as authentication, integrity, and secrecy are specified in a separate section. Therefore, when SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN generates the attack trace if an attack is found, and it considers the protocol unsafe. The presented protocol's simulation results are achieved by the OFMC back-end checker and the CL-AtSe back-end checker. Figure 71 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure. 72 presents the OFMC back-end checker report shows that the protocol is SAFE, thus

meeting the defined security goals. In summary, we can conclude that the proposed protocol is secure.

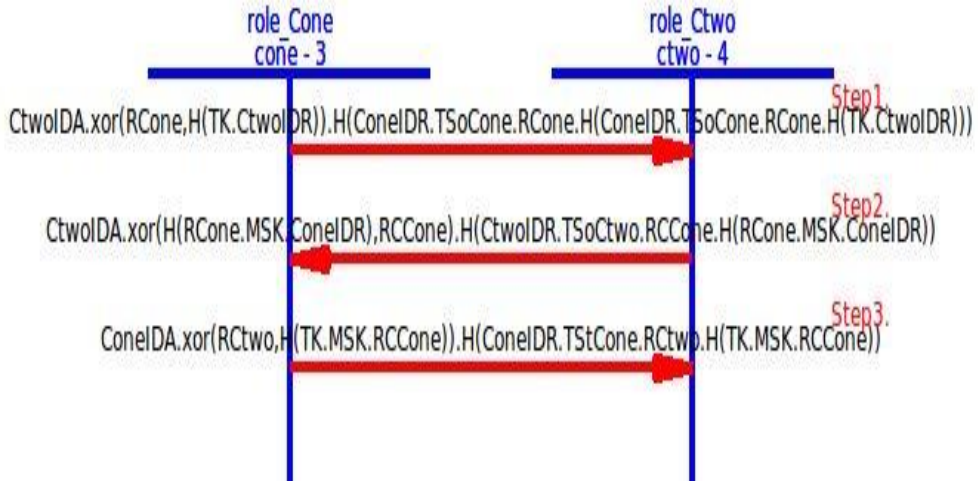


Figure 70: Snapshot of the protocol simulation in AVISPA

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/span/span/testsuite/results/Protocol_2_Cone_Ctwo_final_m.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 1 states
  Reachable  : 1 states
  Translation: 0.03 seconds
  Computation: 0.00 seconds
```

Figure 71: CL-AtSe summary report

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/Protocol_2_Cone_Ctwo_final_m.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.08s
  visitedNodes: 5 nodes
  depth: 2 plies
```

Figure 72: OFMC summary report

5.7.1.1.1.3 Protocol 3: mini-gateway-IoT node mutual authentication

The abstract notations used to describe the authentication protocol and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 18.

Table 18: Abstract notation and AVISPA HLPSL scripting variables/functions for protocol specification

Notation	**Description**
N	N
MG	MG
S_{ID}	SIDNIG
N_{IDR}	NIDR

N_{IDA}	NIDA
MG_{IDR}	MGIDR
MG_{IDA}	MGIDA
S_1, S_2, S_3	T, X, Y
X_1	Xone
X_2	Xtwo
X_3	Xthree
X_4	Xfour
Y_1	Yone
Y_2	Ytwo
Y_3	Ythree
Y_4	Yfour
RX	RX
Sk	Skk
TS_{MG1}	TSoneMG
TS_{N1}, TS_{N2}	TsoneN,TstwoN,
RV_{N1}	Na

RV_{N2}	Naa
RVM_G	Nb
Ri	Rone
Ri+1	Rtwo
C	Cone
Ci+1	Ctwo
C’	CMG
CHX	CHX
CHXi+1	CHXnew
$\oplus$	XOR
H	H
N_{IDAnew}	NnewIDA
OTP_N	NOTP

The main goals of the simulation are as follows:

1. **Goal 1**: The secrecy_of secNa represents that Na is kept secret to (N, MG) only.
2. **Goal 2**: The secrecy_of secNb represents that Nb is kept secret to (N, MG) only

3. **Goal 3**: The secrecy_of secNaa represents that Naa is kept secret to (N, MG) only
4. **Goal 4**: The secrecy_of secOTT represents that OTP is kept secret to (N, MG) only
5. **Goal 5**: The secrecy_of secNIDR represents that the NIDR is permanently kept secret, known to only (N and MG).
6. **Goal 6**: The secrecy_of secMGIDR represents that the MGIDR is permanently kept secret, known to only (N and MG).
7. **Goal 7**: The secrecy_of secX represents that X is permanently kept secret, known to only (N and MG).
8. **Goal 8**: The secrecy_of secY represents that Y is permanently kept secret, known to only (N and MG).
9. **Goal 9**: The secrecy_of secCone represents that Cone is permanently kept secret, known to only (N and MG).
10. **Goal 10**: The secrecy_of secRone represents that Rone is permanently kept secret, known to only (N and MG).
11. **Goal 11**: The secrecy_of secCHXnew represents that CHXnew is permanently kept secret, known to only (N and MG).
12. **Goal 12**: The secrecy_of secRtwo represents that Rtwo is permanently kept secret, known to only (N and MG).
13. **Goal 13**: The secrecy_of secNOTT represents that NOTP is permanently kept secret, known to only (N and MG).

14. **Goal 14**: The secrecy_of secNnewIDA represents that NnewIDA is permanently kept secret, known to only (N and MG).
15. **Authentication Property 1**: The authentication_on na represents that N generates Na. If MG securely receives Na through a message, it authenticates N.
16. **Authentication Property 2**: The authentication_on Nb represents that MG generates Nb. If N securely receives Nb through a message, it authenticates MG.

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with their message exchanges. There are four defined roles: that are: (1) role_N that is played by the IoT node; (2) role _MG that is played by the mini-gateway gateway; (3) session, where the session role and all its declarations are defined, (4) and environment, which instantiates all agents, variables, and functions.

Figure B_11 in Appendix B presents the role of node N. Node N is aware of the N and MG agents in the protocol, its alias identity (NIDA), and the MG alias identity (MGIDA). Also, it is aware of its own real identity (NIDR) and the MG real identity (MGIDR) that should be kept secured. Furthermore, N is aware of the three authentication parameters (S_1, S_2, $S_{3)}$, one-time token (OTP), the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model. At the first state “state 2,” N receives a start message “Rcv(start)” as a signal to begin the protocol run. All local variables are declared under the local section. At the first transition, “State 2”, N generates new random values

(TSoneN and Na) and computes Xone, Xtwo, and SN. The computation process of the Xone,Xtwo, and SN follows the presented protocol description. N sends (NIDA, TSoneN, Xtwo, SN) to the MG.

At the second transition, “State 4”, N receives the (MGIDA, SIDNIG, TSoneMG, Ytwo, CMG, Ythree) message from the MG. At this transition, if the Ythree appears as expected by N, then N authenticates the MG. After authenticating the MG, N generates a new challenge CTwo and calculates the corresponding response Rtwo. Also, N generates a new OTT (NOTT) and a new Alias ID (NnewIDA) to be used in the next authentication session. Then N will store its new NnewIDA, and the new NOTT on its own memory. The “end role” at the end of the N role denotes the end of the role Node played by N.

Figure B_12 in Appendix B shows the role of the mini gateway played by MG. The MG is aware of all agents in the protocol (N and MG), its alias and real identities (MGIDA, MGIDR), the Alias and real identities (NIDA, NIDR) of the N node. MG knows the OTT, the virtual domain ID of N(GID), the typeID of N(TID), and the hash function H (·), and the send/receive channels Snd/Rcv. The MG does not know S_1, S_2, or S_3 parameters that will be locally computed later using N's received authentication parameters. All local variables are defined under the local section. At the first transition, “State 1”, MG receives the (NIDA, TSoneN, Xtwo, SNG) message which was sent by N in role node played by N. MG uses the N’ alias ID to retrieve its real ID, GID, TID, and its OTT. MG verify the received SNG’. Once the MG verifies SNG, MG generates a fresh value Nb and a TSoneMG. Also, MG retrieved N’s PUF challenge (C_{one}) from its memory. Then, MG computes Yone, Ytwo, Ythree, and CMG. Then the MG will send to

N (MGIDA, SIDNID, TSoneMG, Ytwo, CMG, Ythree). The computation for Ythree follows the description of the introduced scheme.

At the second transition, “State 3”, MG received the (NIDA, SIDNIG, TSTwoN, Xthree, RX, Xfour) from N. The computation of the Xfour followed the description of the presented protocol. At this transition, if the Xfour is proved to be valid by MG, IG will authenticate the N. Consequently, the MG stores the new C, new R, new alias ID for N, and a new OTT in its database for the next authentication session. The computation of NnewIDA and NOTT follows the description of the presented protocol. The “end role” at the end of the IG role denotes the end of the role Node played by IG.

Figure B_13 in Appendix B shows the session role where the two agents’ roles are invoked, and all the session parameters are defined. First, all known constant parameters and their declarations are presented. The predefined constants are NIDR, MGIDA, MGIDR, NIDA, OTT, T, X, Y, and CHX as predefined constants, and H as the hash function. A send and receive channel is assigned to each agent under the local section. The “end role” at the end of the session role indicates the end of this role.

Figure B_14 in Appendix B shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants n and mg are instantiated as agents representing agents N and MG. The constants t,x,y, nida, nidr , mgida , mgidr, ott,and chx instantiates T, X, Y, NIDA, NIDR, IGIDA, IGIDR, OTT, and CHX, respectively. The function h instantiates the hash function H. The protocol identifiers are secNIDR, secMGIDR, secOTT, secCHXnew, secNnewIDA, secNOTP, secX,secY,secRtwo, secNa, secNb, secRtwo, secNaa, secRone, secCone, na, and nb are also instantiated and defined. In the intruder knowledge section, all relevant

values that the intruder is assumed to know before the execution are provided. The attacker is assumed to know n and mg. He/ she is also assumed to know the hash h. The session is instantiated with n, mg, nida, nidr, mgida,mgidr,t,x,y, ott,chx, and h instances in the composition section. The “end role” at the end of the environment role

denotes the end of this role.

Figure B_15 in Appendix B shows the simulation goals which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator is dictated to check the secrecy NIDR, MGIDR, X,Y,Na,Nb,Naa,OTT, Rone,Rtwo,Cone,CHXnew,NOTP,NnewIDA, and OTT at different states using secrecy_of secNIDR’, ‘secrecy_of secMGIDR’, ‘secrecy_of secX’, ‘secrecy_of secY’, ‘secrecy_of secCone’ ,‘secrecy_of secRone’, ‘secrecy_of secRtwo’, ‘secrecy_of secNa’, ‘secrecy_of secNb’, ‘secrecy_of secNaa’, ‘secrecy_of secOTT’, ‘secrecy_of secChXnew’, ‘secrecy_of secNnewIDA’ and ‘secrecy_of secNOTT’. The authentication is checked using ‘authentication_on na’ and ‘authentication_on nb’. The “end role” at the end of the goal section denotes the end of this role.

5.7.1.1.1.3.1 Simulation results

The SPAN protocol simulation's MSC corresponding to the HLPSL specification is shown in figure 73. Figure 74 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure. 75 presents the OFMC back-end checker report shows that the protocol is SAFE, thus meeting the defined security goals. In summary, we can conclude that the proposed SMART protocol is secure.

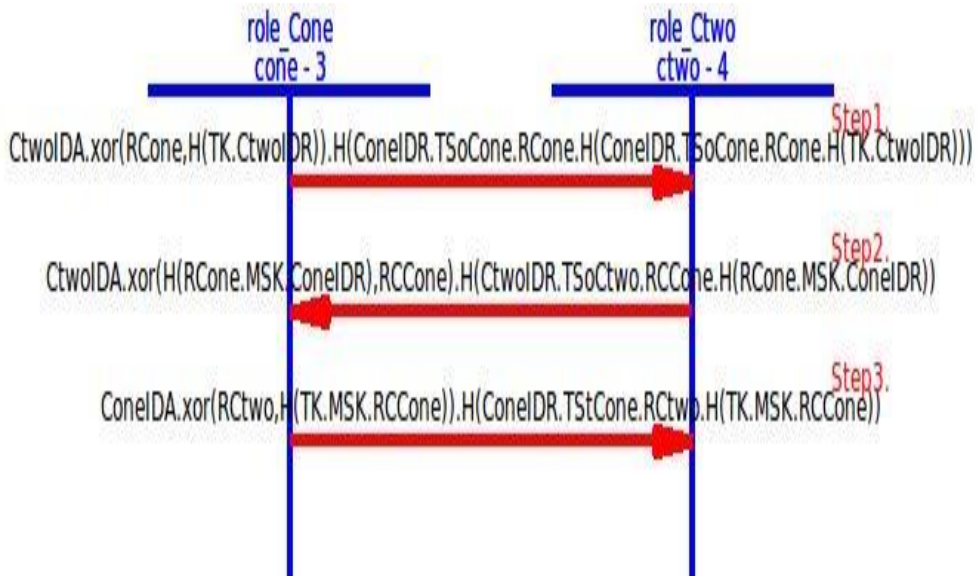


Figure 73: Snapshot of the protocol simulation in AVISPA

```
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL
PROTOCOL
 /home/span/span/testsuite/results/Protocol_2_Cone_Ctwo_final_m.if
GOAL
 As Specified
BACKEND
 CL-AtSe
STATISTICS
 Analysed   : 1 states
 Reachable  : 1 states
 Translation: 0.03 seconds
 Computation: 0.00 seconds
```

Figure 74: CL-AtSe summary report

```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/Protocol_2_Cone_Ctwo_final_m.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.08s
 visitedNodes: 5 nodes
 depth: 2 plies
```

Figure 75: OFMC summary report

5.7.1.1.2 Formal proof Based on BAN logic

5.7.1.1.2.1 Protocol 1: Parent gateway-child gateway mutual authentication

In this section, we use BAN logic to verify the legitimacy of the session key (*ssk*), the TK_{12}, TK_{13}, and OTPx that are shared between the communicating entities C and P. To ensure the security of the proposed protocol under BAN logic, it needs to satisfy a set of security goals. The following section defines the main goals of the analysis of the presented authentication scheme.

5.7.1.1.2.1.1 Protocol 1 Goals Identification

Goal 1: The C and the P want to establish a shared secret key (*ssk*) key that is believed by each other.

G1_1: P believes that the C believes that the *ssk* is a securely shared parameter between C and P.

$$P| \equiv C| \equiv (C \overset{ssk}{\longleftrightarrow} P)$$

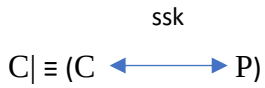
G1_2: P believes that *ssk* is a securely shared parameter between C and P.

$$P| \equiv (C \overset{ssk}{\longleftrightarrow} P)$$

G1_3: C believes that the P believes that the *ssk* is a securely shared parameter between C and P.

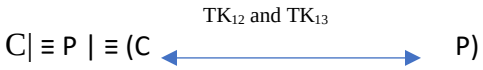
$$C| \equiv P| \equiv (C \overset{ssk}{\longleftrightarrow} P)$$

G1_4: C believes that *ssk* is a securely shared parameter between C and P.

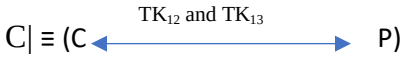


Goal 2: TK_{12} and TK_{13} are well protected and believed by C.

G2_1: C believes that the P believes that the TK_{12} and TK_{13} are securely shared parameters between C and P.

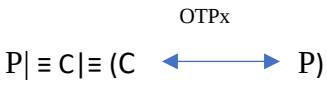


G2_2: C believes that TK_{12} and TK_{13} are securely shared parameters between C and P.

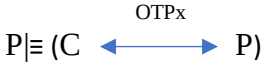


Goal 3: P and C want to generate a secret one-time password (OTPx) that is believed by each other.

G3_1: P believes that C believes that the OTPx is a securely shared parameter between P and C.



G3_2: P believes that OTPx is a securely shared parameter between C and P.



G3_3: C believes that the P believes that the OTPx is a securely shared parameter between C and P.

$$C| \equiv P \mid \equiv (C \overset{OTPx}{\longleftrightarrow} P)$$

G3_4: C believes that OTPx is a securely shared parameter between C and P.

$$C| \equiv (C \overset{OTPx}{\longleftrightarrow} P)$$

5.7.1.1.2.1.2 Protocol 1 messages idealization

First, we transfer all transmitted messages into idealized form as follows:

M1: $C \longrightarrow P:(ChG_{IDR}, TS_{CHG1}, RV_{CHG1})_{X1}$

M2: $P \longrightarrow C:(PG_{IDR}, TS_{PG1}, RV_{PG1})_{YX}$

M3: $C \longrightarrow P:(ChG_{IDR}, TS_{CHG2}, RV_{CHG2})_{XCC}$

M4: $P \longrightarrow C:(PG_{IDR}, TS_{PG2}, TK_{12}, TK_{13})_{OTP}$

5.7.1.1.2.1.3 Protocol 1 main assumptions

The second step to be completed is to define some assumptions as the initiative promises. The fundamental assumptions of the presented authentication scheme are as follows:

P1: P believes that X_1 is a secure shared parameter between C and P

$$P| \equiv (C \overset{X1}{\longleftrightarrow} P)$$

P2: P believes C believes that that X_1 is a secure shared parameter between C and P

$$P \mid \equiv C \mid \equiv (C \overset{X1}{\longleftrightarrow} P)$$

P3: C believes that Yx is a secure shared parameter between C and P

$$C \mid \equiv (C \overset{Yx}{\longleftrightarrow} P)$$

P4: C believes P believes that Yx is a secure shared parameter between C and P

$$C \mid \equiv P \mid \equiv (C \overset{Yx}{\longleftrightarrow} P)$$

P5: P believes that Xcc is a secure shared parameter between C and P

$$P \mid \equiv (C \overset{Xcc}{\longleftrightarrow} P)$$

P6: P believes C believes that Xcc is a secure shared parameter between C and P

$$P \mid \equiv C \mid \equiv (C \overset{Xcc}{\longleftrightarrow} P)$$

P7: C believes that OTP is a secure shared parameter between C and P

$$C \mid \equiv (C \overset{OTP}{\longleftrightarrow} P)$$

P8: C believes P believes that OTP is a secure shared parameter between C and P

$$C \mid \equiv P \mid \equiv (C \overset{OTP}{\longleftrightarrow} P)$$

P9: P believes that OTP is a secure shared parameter between C and P

$$P \mid \equiv (C \overset{OTP}{\longleftrightarrow} P)$$

P10: P believes P believes that OTP is a secure shared parameter between C and P

$$P \mid \equiv C \mid \equiv (C \overset{OTP}{\longleftrightarrow} P)$$

P 11: P believes ChG_{IDR} is a secure shared parameter between C and P.

$$P \mid \equiv (C \overset{ChG_{IDR}}{\longleftrightarrow} P)$$

P 12: P believes C believes that ChG_{IDR} is a secure shared parameter between C and P.

$$P \mid \equiv C \mid \equiv (C \overset{ChG_{IDR}}{\longleftrightarrow} P)$$

P13: C believes PG_{IDR} is a secure shared parameter between C and P.

$$C \mid \equiv (C \overset{PG_{IDR}}{\longleftrightarrow} P)$$

P14: C believes P believes that PG_{IDR} is a secure shared parameter between C and P.

$$C \mid \equiv P \mid \equiv (C \overset{PG_{IDR}}{\longleftrightarrow} P)$$

P 15: P believes TS_{ChG1} is fresh.

$$P \mid \equiv \#(TS_{ChG1})$$

P 16: P believes TS_{ChG2} is fresh.

$$P \mid \equiv \#(TS_{ChG2})$$

P 17: C believes TS_{PG1} is fresh.

$$C \mid \equiv \#(TS_{PG1})$$

P 18: C believes TS_{PG2} is fresh.

$$C| \equiv \#(TS_{PG2})$$

P 19: P believes RV_{ChG1} is fresh.

$$P| \equiv \#(RV_{ChG1})$$

P20: P believes that C believes RV_{ChG1}.

$$P| \equiv C | \equiv \ RV_{ChG1}$$

P21: P believes that C has jurisdiction over RV_{ChG1}. That is, C is an authority and believes RV_{ChG1}.

$$P| \equiv C \Rightarrow RV_{ChG1}$$

P 22: P believes RV_{ChG2} is fresh.

$$P| \equiv \#(RV_{ChG2})$$

P23: P believes that C believes RV_{ChG2}.

$$P| \equiv C | \equiv \ RV_{ChG2}$$

P24: P believes that C has jurisdiction over RV_{ChG2}. That is, C is an authority and believes RV_{ChG2}.

$$P| \equiv C \Rightarrow RV_{ChG2}$$

P 25: C believes RV_{PG1} is fresh.

$$C| \equiv \#(RV_{PG1})$$

P26: C believes that P believes RV_{PG1}.

$$C| \equiv P| \equiv \ RV_{PG1}$$

P27: C believes that P has jurisdiction over RV_{PG1}. That is, P is an authority and believes RV_{PG1}.

$$C| \equiv P \Rightarrow RV_{PG1}$$

P28: P believes C believes that SK is a secure shared parameter between C and P.

$$P| \equiv C| \equiv \ (C \overset{ssk}{\longleftrightarrow} P)$$

P 29: C believes P believes that SK is a secure shared parameter between C and P.

$$C| \equiv P| \equiv (C \overset{ssk}{\longleftrightarrow} P)$$

P 30: C believes that P believes TK_{12}.

$$C| \equiv P| \equiv TK_{12}$$

P31: C believes that P has jurisdiction over TK_{12}, which is securely shared parameter between C and P. That is, P is an authority and believes TK_{12}.

$$C| \equiv P \Rightarrow (C \overset{TK_{12}}{\longleftrightarrow} P)$$

P 32: C believes that P believes TK_{13}.

$$C| \equiv P| \equiv TK_{13}$$

P33: C believes that P has jurisdiction over TK_{13}, which is securely shared parameter between C and P. That is, P is an authority and believes TK_{13}.

$$C| \equiv P \Rightarrow (C \overset{TK_{13}}{\longleftrightarrow} P)$$

P34: C believes that P believes OPTx.

$$C| \equiv P | \equiv OTPx$$

P35: P believes that C believes OPTx.

$$P| \equiv C | \equiv OTPx$$

5.7.1.1.2.1.4 Analysis of the authentication protocol

We then prove that the proposed protocol achieves the security goals based on the idealized form of the messages, assumptions, and BAN logic rules. The proposed authentication scheme analysis is shown below to prove that the protocol achieves mutual authentication between C and P.

According to M_1:

$$V_1: P \triangleleft (ChG_{IDR}, TS_{ChG1}, RV_{ChG1})_{X1}$$

According to P1, P11, and Rule 1, we derive the following:

$$\text{V2: } \frac{P|\equiv C \overset{X1}{\leftrightarrow} P, P \triangleleft (ChG_{IDR}, TS_{ChG1}, RV_{ChG1}) \quad C \overset{X1}{\leftrightarrow} P}{P|\equiv C |\sim (ChG_{IDR}, TS_{ChG1}, RV_{ChG1})}$$

According to P15, P19, and rule 3, we derive the following:

$$\text{V3: } \frac{P|\equiv \#(TS_{ChG1} \; and \; RV_{ChG1})}{P|\equiv \#(ChG_{IDR}, TS_{ChG1}, RV_{ChG1})|}$$

According to P2, P12, V2, V3, and rule 2, we derive the following:

V4:

$$\frac{P|\equiv\#(ChG_{IDR},TS_{ChG1},RV_{ChG1}),\mathrm{P}|\equiv\mathrm{C}\sim(ChG_{IDR},TS_{ChG1},RV_{ChG1})}{P|\equiv\mathrm{C}|\equiv(ChG_{IDR},TS_{ChG1},RV_{ChG1})}$$

According to V4, P20, P21, and rule 4, we derive the following:

$$\text{V5: } \frac{\mathrm{P}\ |\equiv\mathrm{C}|\Rightarrow RV_{ChG1}\ ,\mathrm{P}\ |\equiv\ \mathrm{C}\ |\equiv RV_{ChG1}}{\mathrm{P}|\equiv\ RV_{ChG1}}$$

According to M_2:

V_6: C (P_{IDR}, TS_{PG1}, RV_{PG1}) $_{Yx}$

According to P3, P13 and Rule 1, we derive the following:

$$\text{V7: } \frac{C|\equiv\mathrm{P}\ \underset{\longleftrightarrow}{Yx}\ C\ ,\mathrm{P}\lhd(\mathrm{P}_{IDR},TS_{PG1},RV_{PG1})\ \underset{\longleftrightarrow}{C\ Yx}\ p}{C|\equiv\mathrm{P}\ |\sim\ (\mathrm{P}_{IDR},TS_{PG1},RV_{PG1})}$$

According to P17, P25, and rule 3, we derive the following:

$$\text{V8: } \frac{C|\equiv\#(TS_{PG1}\text{and }\ RV_{PG1})}{C|\equiv\#(\mathrm{P}_{IDR},TS_{PG1},RV_{PG1})|}$$

According to P4, P14, V7, V8 and rule 2, we derive the following:

$$\text{V9: } \frac{C|\equiv\#(\mathrm{P}_{IDR},TS_{PG1},RV_{PG1}),\mathrm{C}|\equiv\mathrm{P}\sim(\mathrm{P}_{IDR},TS_{PG1},RV_{PG1})}{C|\equiv\mathrm{P}|\equiv(\mathrm{P}_{IDR},TS_{PG1},RV_{PG1})}$$

According to P26, P27, V9, and rule 4, we derive the following:

$$\text{V10: } \frac{C \mid\equiv \text{P}\mid\Rightarrow RV_{PG1} \ ,\text{C} \mid\equiv \text{P} \mid\equiv RV_{PG1}}{\text{C}\mid\equiv \ RV_{PG1}}$$

According to M_3:

$$V_{11}\text{: P (ChG}_{IDR}\text{, TS}_{ChG2}\text{, RV}_{ChG2}\text{) Xcc}$$

According to P5, P11, and Rule 1, we derive the following:

$$\text{V12: } \frac{P\mid\equiv \text{C} \ \text{XCC} \quad \text{P} \ ,\text{P} \lhd (ChG_{IDR},TS_{ChG2},RV_{ChG2}) \ C \ XCC \quad P}{P\mid\equiv \text{C} \mid\sim \ (ChG_{IDR},TS_{ChG2},RV_{ChG2})}$$

According to P16, P22, and rule 3, we derive the following:

$$\text{V13: } \frac{P\mid\equiv \#(TS_{ChG2} \ and \ RV_{ChG2})}{P\mid\equiv \#(ChG_{IDR},TS_{ChG2},RV_{ChG2})\mid}$$

According to P6, P12, V12, V13, and rule 2, we derive the following:

V14:

$$\frac{P\mid\equiv \#(ChG_{IDR},TS_{ChG2},RV_{ChG2}),\text{P}\mid\equiv \text{C}\sim (ChG_{IDR},TS_{ChG2},RV_{ChG2})}{P\mid\equiv \text{C}\mid\equiv (ChG_{IDR},TS_{ChG2},RV_{ChG2})}$$

According to V14, P23, P24, and rule 4, we derive the following:

$$\text{V15: } \frac{\text{P} \mid\equiv \text{C}\mid \Rightarrow RV_{ChG2} \ ,\text{P} \mid\equiv \text{C} \mid\equiv RV_{ChG2}}{\text{P}\mid\equiv \ RV_{ChG2}}$$

According to M_4:

V_{16}: C (P_{IDR}, TS_{PG2}, TK_{12}, TK_{13}) OTP

According to P7, P13, V16 and Rule 1, we derive the following:

$$\text{V17: } \frac{C|\equiv P \overset{OTP}{\longleftrightarrow} C, P \triangleleft (P_{IDR}, TS_{PG2}, TK_1, TK_2) \quad C \overset{OTP}{\longleftrightarrow} P}{C|\equiv P|\sim (P_{IDR}, TS_{PG2}, TK_{12}, TK_{13})}$$

According to P18 and rule 3, we derive the following:

$$\text{V18: } \frac{C|\equiv \#(TS_{PG2})}{C|\equiv \#(P_{IDR}, TS_{PG2}, TK_1, TK_2)|}$$

According to P8, P14, V17, V18 and rule 2, we derive the following:

$$\text{V19: } \frac{C|\equiv \#(P_{IDR}, TS_{PG2}, TK_{12}, TK_{13}), C|\equiv P\sim(P_{IDR}, TS_{PG2}, TK_{12}, TK_{13})}{C|\equiv P|\equiv(P_{IDR}, TS_{PG2}, TK_{12}, TK_{13})}$$

According to P30, P31, P32, P33, V19, and rule 4, we derive the following:

$$\text{V20: } \frac{C|\equiv P|\Rightarrow TK_{12}, TK_{13}, C|\equiv P|\equiv TK_{12}, TK_{13}}{C|\equiv TK_{12}, TK_{13}}$$

As ssk = H (RV_{ChG1}|| RV_{ChG2} || RV_{PG1} || OTP) and in combination with P9, V4, V14 we can derive

V21: $P|\equiv C|\equiv (C \overset{ssk}{\longleftrightarrow} P)$ (Goal 1_1)

As ssk = H (RV_{ChG1}|| RV_{ChG2} || RV_{PG1} || OTP) and in combination with P10, P28, V5, V15 and V21

V22: $P|\equiv (C \overset{ssk}{\longleftrightarrow} P)$ (Goal 1_2)

As ssk = H (RV_{ChG1}|| RV_{ChG2} || RV_{PG1} || OTP) and in combination with P7, V9, we can derive

$$V23: \ C|\equiv P|\equiv (C \overset{ssk}{\longleftrightarrow} P) \qquad \text{(Goal 1\_3)}$$

As ssk = H (RV_{ChG1}|| RV_{ChG2} || RV_{PG1} || OTP) and combining with P8, P29, V10, and V23

$$V24: \ C|\equiv (C \overset{ssk}{\longleftrightarrow} P) \qquad \text{(Goal 1\_4)}$$

Based on V20, we can derive

$$V25: \ C|\equiv P|\equiv (C \overset{NOTP,\ TK_1,\ TK_2}{\longleftrightarrow} P) \qquad \text{(Goal 2\_1)}$$

Based on P30, P32, V20, and V25, we derive the following

$$V26: \ C|\equiv (C \overset{NOTP,\ TK_1,\ TK_2}{\longleftrightarrow} P) \qquad \text{(Goal 2\_2)}$$

As OTPx = H (OTP || RV_{ChG1}) and in combination with P9 and V4 we can derive

$$V27: \ P|\equiv C|\equiv (C \overset{OTPx}{\longleftrightarrow} P) \qquad \text{(Goal 3\_1)}$$

As OTPx = H (OTP || RV_{ChG1}) and in combination with P10, P35, V5, V15 and V27

$$V28: \ P|\equiv (C \overset{OTPx}{\longleftrightarrow} P) \qquad \text{(Goal 3\_2)}$$

As OTPx = H (OTP || RV_{ChG1}) and in combination with P7 we can derive

$$V29: \ C|\equiv P|\equiv (C \overset{OTPx}{\longleftrightarrow} P) \qquad \text{(Goal 3\_3)}$$

As OTPx = H (OTP || RV_{ChG1}) and combining with P8, P34, and V29

$$V30: C|\equiv (C \overset{OTPx}{\longleftrightarrow} P) \qquad \text{(Goal 3\_4)}$$

Therefore, the above logic proves that the proposed protocol achieves the goals 1_1 to 1_4, goals 2-1 to 2-2, and goals 3_1 to 3_4 successfully. Thus, we prove that the shared secret key *ssk*, OTPx, TK_{12}, and TK_{13} are trusted by both C and the P. Based on the above results, the achievement of goals 1_1 to 1_4, 2_1 to 2_2, and 3_1 to 3_4 proves that the proposed protocol achieves the mutual authentication and the *ssk*, OTPx, TK_{12}, and TK_{13} are securely shared between C and P.

5.7.1.1.2.2 Protocol 2: Child gateway-child gateway mutual authentication

5.7.1.1.2.2.1 Protocol 2: Goals identification

Goal 1: The ChG_1 and the ChG_2 want to establish a shared secret key (*ssk*) key that is believed by each other.

G1_1: ChG_2 believes that the ChG_1 believes that the *ssk* is a securely shared parameter between ChG_1 and ChG_2.

$$ChG\_2|\equiv ChG\_1|\equiv (ChG\_1 \overset{ssk}{\longleftrightarrow} ChG\_2)$$

G1_2: ChG_2 believes that *ssk* is a securely shared parameter between ChG_1 and ChG_2.

$$ChG\_2|\equiv (ChG\_1 \overset{ssk}{\longleftrightarrow} ChG\_2)$$

G1_3: ChG_1 believes that the ChG_2 believes that the *ssk* is a securely shared parameter between ChG_1 and ChG_2.

$$ChG\_1|\equiv ChG\_2|\equiv (ChG\_1 \overset{ssk}{\longleftrightarrow} ChG\_2)$$

G1_4: ChG_1 believes that *ssk* is a securely shared parameter between ChG_1 and ChG_2.

$$ChG\_1|\equiv (ChG\_1 \overset{ssk}{\longleftrightarrow} ChG\_2)$$

5.7.1.1.2.2.2 Protocol 2: message idealization

First, we transferred all transmitted messages into idealized form as follows.

M1: $ChG\_1 \longrightarrow ChG\_2:(ChG_{IDR\_1}, TK, TS_{ChG11}, RV_{ChG11})_{Rj}$

M2: $ChG\_2 \longrightarrow ChG\_1:(ChG_{IDR\_2}, TK, MSK, TS_{ChG21}, RV_{ChG21})_{D1}$

M3: $ChG\_1 \longrightarrow ChG\_2:(ChG_{IDR\_1}, TK, MSK, TS_{ChG12}, RV_{ChG12})_{T3}$

5.7.1.1.2.2.3 Protocol 2 main assumptions

The second step to be completed is to define some assumptions as of the initiative promises. The fundamental assumptions of the presented authentication protocol are as follows:

P1: ChG_2 believes that Rj is a secure shared parameter between ChG_2 and ChG_1

$$ChG\_2| \equiv (ChG\_1 \overset{Ri}{\longleftrightarrow} ChG\_2)$$

P2: ChG_2 believes ChG_1 believes that Rj is a secure shared parameter between ChG_1 and ChG_2

$$\overset{Rj}{}$$

$$ChG\_2 \mid \equiv ChG\_1 \mid \equiv (ChG\_1 \longleftrightarrow ChG\_2)$$

P3: ChG_1 believes that D_1 is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_1 \mid \equiv (ChG\_1 \overset{D_1}{\longleftrightarrow} ChG\_2)$$

P4: ChG_1 believes ChG_2 believes that D_1 is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_1 \mid \equiv ChG\_2 \mid \equiv (ChG\_1 \overset{D_1}{\longleftrightarrow} ChG\_2)$$

P5: ChG_2 believes that T3 is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_2 \mid \equiv (ChG\_1 \overset{T_3}{\longleftrightarrow} ChG\_2)$$

P6: ChG_2 believes ChG_1 believes that T_3 is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_2 \mid \equiv ChG\_1 \mid \equiv (ChG\_1 \overset{T_3}{\longleftrightarrow} ChG\_2)$$

P 7: ChG_2 believes TS_{ChG11} is fresh.

$$ChG\_2 \mid \equiv \#(TS_{ChG11})$$

P 8: ChG_2 believes TS_{ChG12} is fresh.

$$ChG\_2 \mid \equiv \#(TS_{ChG12})$$

P 9: ChG_1 believes TS_{ChG21} is fresh.

$$ChG\_1 \mid \equiv \#(TS_{ChG21})$$

P 10: ChG_2 believes RV_{ChG11} is fresh.

$$ChG\_2 \mid \equiv \#(RV_{ChG11})$$

P11: ChG_2 believes that ChG_1 believes RV_{ChG11}.

$$ChG\_2 \mid \equiv ChG\_1 \mid \equiv \ RV_{ChG11}$$

P12: ChG_2 believes that ChG_1 has jurisdiction over RV_{ChG11}. That is, ChG_1 is an authority and believes RV_{ChG11}.

$$ChG\_2 \mid \equiv ChG\_1 \ \Rightarrow RV_{ChG11}$$

P 13: ChG_2 believes RV_{ChG12} is fresh.

$$ChG\_2 \mid \equiv \#(RV_{ChG12})$$

P14: ChG_2 believes that ChG_1 believes RV_{ChG12}.

$$ChG\_2 \mid \equiv ChG\_1 \mid \equiv \ RV_{ChG12}$$

P15: ChG_2 believes that ChG_1 has jurisdiction over RV_{ChG12}. That is, ChG_1 is an authority and believes RV_{ChG12}.

$$ChG\_2 \mid \equiv ChG\_1 \Rightarrow RV_{ChG12}$$

P 16: ChG_1 believes RV_{ChG21} is fresh.

$$ChG\_1 \mid \equiv \#(RV_{ChG21})$$

P17: ChG_1 believes that ChG_2 believes RV_{ChG21}.

$$ChG\_1 \mid \equiv ChG\_2 \mid \equiv \ RV_{ChG21}$$

P18: ChG_1 believes that ChG_2 has jurisdiction over RV_{PG1}. That is, ChG_2 is an authority and believes RV_{ChG21}.

$$ChG\_1 \mid \equiv ChG\_2 \quad \Rightarrow RV_{ChG21}$$

P19: ChG_1 believes that TK is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_1 \mid \equiv \ ChG\_1 \overset{TK}{\longleftrightarrow} ChG\_2)$$

P20: ChG_1 believes ChG_2 believes that TK is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_1 \mid \equiv ChG\_2 \mid \equiv (ChG\_1 \overset{TK}{\longleftrightarrow} ChG\_2)$$

P21: ChG_2 believes that TK is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_2 \mid \equiv \ ChG\_1 \overset{TK}{\longleftrightarrow} ChG\_2)$$

P22: ChG_2 believes ChG_1 believes that TK is a secure shared parameter between ChG_1 and ChG_2

$$ChG\_2 \mid \equiv ChG\_1 \mid \equiv (ChG\_1 \overset{TK}{\longleftrightarrow} ChG\_2)$$

P 23: ChG_2 believes ChG_1 believes that *ssk* is a secure shared parameter between ChG_2 and ChG_1

$$ChG\_2 \mid \equiv ChG\_1 \mid \equiv (ChG\_1 \overset{ssk}{\longleftrightarrow} ChG\_2)$$

P 24: ChG_1 believes ChG_2 believes that *ssk* is a secure shared parameter between ChG_1and ChG_2

$$ChG\_1 \mid \equiv ChG\_2 \mid \equiv (ChG\_1 \overset{ssk}{\longleftrightarrow} ChG\_2)$$

5.7.1.1.2.2.4 Protocol 2 Analysis of the authentication protocol

We then prove that the proposed protocol achieves the security goals based on the idealized form of the messages, assumptions, and BAN logic rules. The proposed authentication protocol analysis is shown below to prove that the protocol achieves mutual authentication between ChG_1 and ChG_2.

According to M_1:

$$V_1: \text{ChG\_2} \triangleleft (\text{ChG}_{\text{IDR\_1}}, \text{TK}, \text{TS}_{\text{ChG11}}, \text{RV}_{\text{ChG11}})$$

According to P1 and Rule 1, we derive the following

V2:

$$\frac{\text{ChG\_2}|\equiv \text{ChG\_1} \overset{Rj}{\leftrightarrow} \text{ChG\_2}, \text{ChG\_2} \triangleleft (ChG_{IDR}, TK, TS_{ChG1}, RV_{ChG1}) \; \text{ChG\_1} \overset{Rj}{\leftrightarrow} \text{ChG\_2}}{\text{ChG\_2}|\equiv \text{ChG\_1}|\sim (ChG_{IDR\_1}, TK, TS_{ChG1}1, RV_{ChG11})}$$

According to P7, P10, and rule 3, we derive the following

$$\text{V3: } \frac{\text{ChG\_2}|\equiv \#(TS_{ChG11} \; and \; RV_{ChG11})}{\text{ChG\_2}|\equiv \#(ChG_{IDR\_1}, TK, TS_{ChG1}1, RV_{ChG11})|}$$

According to P2, V2, V3, and rule 2, we derive the following

V4:

$$\frac{\text{ChG\_2}|\equiv \#(ChG_{IDR_1}, \text{TK}, TS_{ChG11}, RV_{ChG11}), \text{ChG\_2}|\equiv \text{ChG\_1} \sim (ChG_{IDR\_1}, TK, TS_{ChG11}, RV_{ChG11})}{\text{ChG\_2}|\equiv \text{ChG\_1}|\equiv (ChG_{IDR\_1}, TK, TS_{ChG11}, RV_{ChG11})}$$

According to P11, P12, and rule 4, we derive the following:

$$\text{V5: } \frac{\text{ChG\_2} \; |\equiv \text{ChG\_1}| \Rightarrow RV_{ChG11}, \text{ChG\_2} \; |\equiv \text{ChG\_1} \; |\equiv RV_{ChG11}}{\text{ChG\_2}|\equiv \; RV_{ChG11}}$$

According to M_2:

V_6: ChG_1 (ChG_{IDR_2}, TK, MSK, TS_{ChG21}, RV_{ChG21}) D1

According to P3 and Rule 1, we derive the following

V7:

$$\frac{\text{ChG\_1} \mid\equiv \text{ChG\_2} \overset{D1}{\leftrightarrow} \text{ChG\_1}, \text{ChG\_2} \triangleleft (ChG_{IDR\_2}, TS_{ChG21}, RV_{ChG21}) \; ChG\_2 \overset{D1}{\leftrightarrow} ChG\_1}{ChG\_1 \mid\equiv ChG\_2 \mid\sim (ChG_{IDR\_2}, TK, MSK, TS_{ChG21}, RV_{ChG21})}$$

According to P9, P16, and rule 3, we derive the following

$$\text{V8: } \frac{\text{ChG\_1} \mid\equiv \#(TS_{ChG21} \text{ and } RV_{ChG21})}{\text{ChG\_1} \mid\equiv \#(ChG_{IDR\_2}, TS_{ChG21}, RV_{ChG21}) \mid}$$

According to P4, V7, V8 and rule 2, we derive the following

V9:

$$\frac{\text{ChG\_1} \mid\equiv \#(ChG_{IDR\_2}, Tk, MSK, TS_{ChG21}, RV_{ChG21}), \text{ChG\_1} \mid\equiv \text{ChG\_2} \sim (ChG_{IDR\_2}, Tk, MSK, TS_{ChG21}, RV_{ChG21})}{\text{ChG\_1} \mid\equiv \text{ChG\_2} \mid\equiv (ChG_{IDR\_2}, TK, MSK, TS_{ChG21}, RV_{ChG21})}$$

According to P17, P18, V9, and rule 4, we derive the following:

$$\text{V10: } \frac{\text{ChG\_1} \mid\equiv \text{ChG\_2} \mid\Rightarrow RV_{ChG21}, \text{ChG\_1} \mid\equiv \text{ChG\_2} \mid\equiv RV_{ChG21}}{\text{ChG\_1} \mid\equiv RV_{ChG21}}$$

According to M_3:

V_{11}: ChG_2 $\triangleleft$ (ChG_{IDR_12},Tk,MSK, TS_{ChG12}, RV_{ChG12})

According to P5 and Rule 1, we derive the following

V12:

$$\frac{\text{ChG\_2}|\equiv \text{ChG\_1} \overset{T3}{\longleftrightarrow} \text{ChG\_2}, \text{ChG\_2} \lhd (ChG_{IDR}, \text{TK}, \text{MSK}, TS_{ChG12}, RV_{ChG12}) \; \text{ChG\_1} \overset{T3}{\longleftrightarrow} \text{ChG\_2}}{\text{ChG\_2}|\equiv \text{ChG\_1}\,|\sim\, (ChG_{IDR\_1} TK, MSK, „TS_{ChG12}1, RV_{ChG11})}$$

According to P8, P13, and rule 3, we derive the following

$$\text{V13: } \frac{\text{ChG\_2}|\equiv \#(TS_{ChG12}\ and\ RV_{ChG12})}{\text{ChG\_2}|\equiv \#(ChG_{IDR\_1}, TS_{ChG12}, RV_{ChG12})|}$$

According to V12, V13, and rule 2, we derive the following

V14:

$$\frac{\text{ChG\_2}|\equiv \#(ChG_{IDR_1}, TK, MSK, TS_{ChG12}, RV_{ChG12}), \text{ChG\_2}|\equiv \text{ChG\_1} \sim (ChG_{IDR\_1}, TK, MSK, TS_{ChG12}, RV_{ChG12})}{\text{ChG\_2}|\equiv \text{ChG\_1}|\equiv (ChG_{IDR\_1}, TK, MSK, TS_{ChG12}, RV_{ChG12})}$$

According to P14, P15, V14, and rule 4, we derive the following:

$$\text{V15: } \frac{\text{ChG\_2}\ |\equiv \text{ChG\_1}| \Rightarrow RV_{ChG12}, \text{ChG\_2}\ |\equiv\ \text{ChG\_1}\ |\equiv RV_{ChG12}}{\text{ChG\_2}|\equiv\ RV_{ChG12}}$$

As *ssk* = H (RV_{ChG11}|| RV_{ChG12} || RV_{ChG21}|| TK) and in combination with V4, V14 and P21 we can derive

V16: ChG_2|≡ChG_1|≡(ChG_2 $\overset{ssk}{\longleftrightarrow}$ ChG_1) (Goal 1_1)

As *ssk* = H (RV_{ChG11}|| RV_{ChG12} || RV_{ChG21}|| TK) and in combination with P22, P23, V5, V15 and V16.

V17: ChG_2|≡ (ChG_1 $\overset{ssk}{\longleftrightarrow}$ ChG_2) (Goal 1_2)

As *ssk* = H (RV_{ChG11}|| RV_{ChG12} || RV_{ChG21}|| TK) and in combination with P19 and V9, we can derive

ssk

V18: ChG_1| ≡ ChG_2|≡ (ChG_1 $\longleftrightarrow$ ChG_1) (Goal 1_3)

As *ssk* = H (RV_{ChG1}|| RV_{ChG2} || RV_{PG1}) and combining with P20, P24, V10, and V18.

V19: ChG_1|≡ (ChG_1 $\overset{ssk}{\longleftrightarrow}$ ChG_2) (Goal 1_4)

Therefore, the above logic proves that the proposed protocol achieves the goals 1_1 to 1_4 successfully. Thus, we prove that the shared secret key *ssk* is trusted by both ChG_1 and the ChG_2. Based on the above results, the achievement of goals 1_1 to 1_4 proves that the proposed protocol achieves mutual authentication, and the session key *ssk* is securely shared between ChG_1 and ChG_2.

5.7.1.1.2.3 Protocol 3: mini-gateway _IoT node mutual authentication

5.7.1.1.2.3.1 Protocol 3 goals identification

Goal 1: C^{i+1}, R^i, and R^{i+1} are well protected and believed by MG.

G1_1: MG believes that the N believes that the C^{i+1}, $R^{i,}$ and R^{i+1} are securely shared parameters between N and MG.

MG| ≡ N | ≡ (N $\overset{R^i, R^{i+1}, Ci^{+1}}{\longleftrightarrow}$ MG)

G1_2: MG believes that C^{i+1}, R^i and R^{i+1} are securely shared parameters between N and MG.

MG| ≡ (N $\overset{R^i, R^{i+1}, Ci^{+1}}{\longleftrightarrow}$ MG)

Goal 2: The N and the MG want to establish a shared secret key (*ssk*) key that is believed by each other.

G2_1: MG believes that the N believes that the *ssk* is a securely shared parameter between N and MG.

$$MG| \equiv N| \equiv (N \overset{ssk}{\longleftrightarrow} MG)$$

G2_2: MG believes that *ssk* is a securely shared parameter between N and MG.

$$MG| \equiv (N \overset{ssk}{\longleftrightarrow} MG)$$

G2_3: N believes that the MG believes that the *ssk* is a securely shared parameter between N and MG.

$$N| \equiv MG \mid \equiv (N \overset{ssk}{\longleftrightarrow} MG)$$

G2_4: N believes that *ssk* is a securely shared parameter between N and MG.

$$N| \equiv (N \overset{ssk}{\longleftrightarrow} MG)$$

5.7.1.1.2.3.2 Protocol 3 Messages Idealization

First, we transfer all transmitted messages into idealized form as follows:

M1: $N \longrightarrow MG:(N_{IDR}, S2, S3, TS_{N1}, RV_{N1})_{OTT}$

M2: $MG \longrightarrow N:(MG_{IDR}, S_1, S_2, RV_{MG1}, TS_{MG}, C)_{OTT}$

M 3: $N \longrightarrow MG:(N_{IDR}, TS_{N2}, RV_{N2}, NOTP, NnewIDA)_{CHXi+1}$

5.7.1.1.2.3.3 Protocol 3 main assumptions

The second step to be completed is to define some assumptions as the initiative promises. The fundamental assumptions of the presented authentication scheme are as follows:

P1: MG believes that OTT is a secure shared parameter between N and MG

$$MG| \equiv (N \overset{OTT}{\longleftrightarrow} MG)$$

P2: MG believes N believes that OTT is a secure shared parameter between N and MG.

$$MG | \equiv N| \equiv (N \overset{OTT}{\longleftrightarrow} MG)$$

P3: N believes that OTT is a secure shared parameter between N and MG.

$$N| \equiv (N \overset{OTT}{\longleftrightarrow} MG)$$

P4: N believes MG believes that OTT is a secure shared parameter between N and MG.

$$N | \equiv MG| \equiv (N \overset{OTT}{\longleftrightarrow} MG)$$

P5: MG believes that CHX^{i+1} is a secure shared parameter between N and MG.

$$MG| \equiv (N \overset{CHX^{i+1}}{\longleftrightarrow} MG)$$

P6: MG believes N believes that CHX^{i+1} is a secure shared parameter between N and MG.

$$MG | \equiv N| \equiv (N \overset{CHX^{i+1}}{\longleftrightarrow} MG)$$

$$MG \mid \equiv N \mid \equiv (N \longleftrightarrow MG)$$

P7: MG believes TS_{N1} is fresh.

$$MG \mid \equiv \#(TS_{N1})$$

P8: MG believes TS_{N2} is fresh.

$$MG \mid \equiv \#(TS_{N2})$$

P9: N believes TS_{MG1} is fresh.

$$N \mid \equiv \#(TS_{MG1})$$

P10: MG believes RV_{N1} is fresh.

$$MG \mid \equiv \#(RV_{N1})$$

P11: MG believes that N believes RV_{N1}.

$$MG \mid \equiv N \mid \equiv \ RV_{N1}$$

P12: MG believes that N has jurisdiction over RV_{N1}. That is, N is an authority and believes RV_{N1}.

$$MG \mid \equiv N \ \Rightarrow RV_{N1}$$

P13: MG believes RV_{N2} is fresh.

$$MG \mid \equiv \#(RV_{N2})$$

P14: MG believes that N believes RV_{N2}.

$$MG \mid \equiv N \mid \equiv \ RV_{N2}$$

P15: MG believes that N has jurisdiction over RV_{N2}. That is, N is an authority and believes RV_{N2}.

$$MG \mid \equiv N \Rightarrow RV_{N2}$$

P16: N believes RV_{MG1} is fresh.

$$N \mid \equiv \#(RV_{MG1})$$

P17: N believes that MG believes RV_{MG1}.

$$N \mid \equiv MG \mid \equiv RV_{MG1}$$

P18: N believes that MG has jurisdiction over RV_{MG1}. That is, MG is an authority and believes RV_{MG1}.

$$N \mid \equiv MG \Rightarrow RV_{MG1}$$

P19: MG believes that N has jurisdiction over R^i

$$MG \mid \equiv N \Rightarrow R^i$$

P20: MG believes that N believes R^i

$$MG \mid \equiv N \mid \equiv R^i$$

P21: MG believes that N has a jurisdiction over R^{i+1}

$$MG \mid \equiv N \Rightarrow R^{i+1}$$

P22: MG believes that N believes R^{i+1}

$$MG \mid \equiv N \mid \equiv R^{i+1}$$

P23: MG believes that N has a jurisdiction over C^{i+1}

$$MG| \equiv N \Rightarrow C^{i+1}$$

P24: MG believes that N believes C^{i+1}

$$MG| \equiv N | \equiv C^{i+1}$$

P25: MG believes N believes that *ssk* is a secure shared parameter between N and MG

$$MG | \equiv N| \equiv (N \overset{ssk}{\longleftrightarrow} MG)$$

P26: N believes MG believes that *ssk* is a secure shared parameter between N and MG

$$N | \equiv MG| \equiv (N \overset{ssk}{\longleftrightarrow} MG)$$

5.7.1.1.2.3.4 Protocol 3 analysis of the authentication protocol

We then prove that the proposed protocol achieves the security goals based on the idealized form of the messages, assumptions, and BAN logic rules. The proposed authentication protocol analysis is shown below to prove that the protocol achieves mutual authentication between N and MG.

According to M_1:

$$V_1: IG \triangleleft (N_{IDR}, S2,S3,TS_{N1},RV_{N1})_{OTP}$$

According to P1 and Rule 1, we derive the following

$$V2: \frac{MG| \equiv N \overset{OTT}{\longleftrightarrow} MG, MG \lhd (N_{IDR},S_2,S_3,TS_{N1},RV_{n1}) \; N \overset{OTT}{\longleftrightarrow} MG}{MG| \equiv N |\sim ((N_{IDR},S_2,S_3,TS_{N1},RV_{n1})}$$

According to P7, P10, and rule 3, we derive the following

$$\text{V3: } \frac{MG|\equiv\#(TS_{N1}\ and\ RV_{N1})}{MG|\equiv\#|(N_{IDR},S_2,S_3,TS_{N1},RV_{n1})}$$

According to V2, V3, and rule 2, we derive the following

$$\text{V4: } \frac{MG|\equiv\#(N_{IDR},S_2,S_3,TS_{N1},RV_{n1}),\mathrm{MG}|\equiv\mathrm{N}\sim(N_{IDR},S_2,S_3,TS_{N1},RV_{n1})}{MG|\equiv\mathrm{N}|\equiv(N_{IDR},S_2,S_3,TS_{N1},RV_{n1})}$$

According to P11, P12, V4 and rule 4, we derive the following:

$$\text{V5: } \frac{\mathrm{MG}\ |\equiv\mathrm{N}|\Rightarrow RV_{n1}\ ,\mathrm{MG}\ |\equiv\ \mathrm{N}\ |\equiv\ RV_{n1}}{\mathrm{MG}|\equiv\ RV_{n1}}$$

According to M_2, we get

$$\text{V6: } \mathrm{N}\ \triangleleft\ (\mathrm{MG_{IDR}},\ \mathrm{S_1},\ \mathrm{S_2},\ \mathrm{RV_{MG1}},\ \mathrm{TS_{MG}},\ \mathrm{C})_{\ Y2}$$

According to P3 Rule 1, we derive the following

$$\text{V7: } \frac{N|\equiv\mathrm{N}\ \overset{OTT}{\longleftrightarrow}\ \mathrm{MG}\ ,\mathrm{N}\ \triangleleft(\mathrm{MG}_{IDR},TS_{MG1},S_1,S_2,RV_{MG1},\mathrm{C})\ N\ \overset{OTT}{\longleftrightarrow}\ MG}{N|\equiv\mathrm{MG}\ |\sim\ (\mathrm{MG}_{IDR},TS_{MG1},S_1,S_2,RV_{MG1},\mathrm{C})}$$

According to P9, P16 and rule 3, we derive the following

$$\text{V8: } \frac{N|\equiv\#(TS_{MG1},RV_{MG1})}{N|\equiv\#(\mathrm{MG}_{IDR},TS_{MG1},S_1,S_2,RV_{MG1},\mathrm{C})}$$

According to V7, V8, and rule 2, we derive the following

V9:

$$\frac{N|\equiv\#(\mathrm{MG}_{IDR},TS_{MG1},S_1,S_2,RV_{MG1},\mathrm{C}),\mathrm{N}|\equiv\mathrm{MG}\sim(\mathrm{MG}_{IDR},TS_{MG1},S_1,S_2,RV_{MG1},\mathrm{C})}{N|\equiv\mathrm{MG}|\equiv(\mathrm{MG}_{IDR},TS_{MG1},S_1,S_2,RV_{MG1},\mathrm{C})}$$

According to P17, P18 and rule 4, we derive the following:

$$\text{V10: } \frac{N \mid \text{MG} \mid \Rightarrow RV_{MG1}, \text{N} \mid \equiv \text{MG} \mid \equiv RV_{MG1}}{N \mid \equiv \ RV_{MG1}}$$

According to M_3:

$$V_{11}\ \text{IG} \triangleleft (N_{IDR}, TS_{N2}, RV_{N2}, \text{OTPN}, \text{NnewIDA}, C^{i+1}, R^{i}, Ri^{+1})_{CHXi^{+1}}$$

According to P5 and Rule 1, we derive the following

V12:

$$\frac{MG \mid \equiv \text{N} \ \text{CHXi+1 MG}, \text{MG} \triangleleft (N_{IDR}, NOTP, NnewIDA, TS_{N2}, RV_{n2}, \ N\ C^{i+1}, R^{i}, R^{i+1}\ MG)_{N\ CHXi+1}\ \ MG}{MG \mid \equiv \text{N} \mid \sim ((N_{IDR}, NOTP, NnewIDA, TS_{N2}, RV_{n2}, N\ C^{i+1}, R^{i}, R^{i+1}\ MG)}$$

According to P8, P13 and rule 3, we derive the following

$$\text{V13: } \frac{MG \mid \equiv \#(TS_{N2}\ and\ RV_{N2})}{MG \mid \equiv \#\mid (N_{IDR}, NOTP, NnewIDA, TS_{N2}, RV_{n2}, N\ C^{i+1}, R^{i}, R^{i+1}\ MG)}$$

According to V12, V13, and rule 2, we derive the following:

V14

$$\frac{MG \mid \equiv \#(N_{IDR}, NOTP, NnewIDA, TS_{N2}, RV_{n2}, N\ C^{i+1}, R^{i}, R^{i+1}\ MG), \text{MG} \mid \equiv \text{N} \sim (N_{IDR}, NOTP, NnewIDA, TS_{N2}, RV_{n2}\ N\ C^{i+1}, R^{i}, R^{i+1}\ MG)}{MG \mid \equiv \text{N} \mid \equiv (N_{IDR}, NOTP, NnewIDA, TS_{N2}, RV_{n2}, N\ C^{i+1}, R^{i}, R^{i+1}\ MG)}$$

According to P14, P15, P19, P20, P21, P22, P23, P24, V14, and rule 4, we derive the following:

$$\text{V15: } \frac{\text{MG} \mid \equiv \text{N} \mid \Rightarrow RV_{n2}, C^{i+1}, R^{i}, R^{i+1}, \text{MG} \mid \equiv \text{N} \mid \equiv RV_{n2}, C^{i+1}, R^{i}, R^{i+1}}{\text{MG} \mid \equiv \ RV_{n2}, C^{i+1}, R^{i}, R^{i+1}}$$

Based on V15, we can derive

$$C^{i+1}, R^{i}, R^{i+1}$$

$$\text{V16: } \text{MG} \mid \equiv \mid \text{N} \equiv (\text{N} \longleftrightarrow \text{MG}) \qquad \text{(Goal 1\_1)}$$

Based on P20, P22, P24, V15, and V16, we derive the following

C^{i+1},R^{i},R^{i+1}

V17: $MG|\equiv (N \longleftrightarrow MG)$ (Goal 1_2)

As session key *ssk* = h ($(RV_{N1}|| RV_{N2} || RV_{MG1} || Ri)$ and in combination with V9, we can derive

ssk

V18: $N| \equiv MG|\equiv (N \longleftrightarrow MG)$ (Goal 2_1)

As session key *ssk* = h ($(RV_{N1}|| RV_{N2} || RV_{MG1} || Ri)$ and combining with P26, V10 and V18

ssk

V19: $N|\equiv (N \longleftrightarrow MG)$ (Goal 2_2)

As session key *ssk* = h ($(RV_{N1}|| RV_{N2} || RV_{MG1} || Ri)$ and combining with V4 and V14, we can derive

ssk

V20: $IG \equiv N|\equiv (N \longleftrightarrow MG)$ (Goal 2_3)

As session key *ssk* = h ($(RV_{N1}|| RV_{N2} || RV_{MG1} || Ri)$ and combining with P25, V5, V15, and V20.

ssk

V21: $MG|\equiv (N \longleftrightarrow MG)$ (Goal 2_4)

Hence, the above logic proves that the proposed protocol achieves Goals 1_1 - 1_2 and 2_1 - 2_4 successfully. In other words, the proposed protocol achieves mutual authentication. Also, it has been proved that C^{i+1}, R^{i}, $R^{i+1,}$ and *ssk* are securely shared between the communicating parties.

In summary, by achieving all the goals mentioned above, we demonstrate the validity of the proposed protocols. We proved that the presented three protocols are secure mechanisms to perform mutual authentication and secret key generation to secure the communication between the involved parties.

5.7.1.2 Informal Security Analysis

This section examines that the proposed protocols are secure against well-known security attacks and satisfy the main security properties. The security of the protocols is explored against various known attacks. This section will provide brief descriptions of different types of attacks against the smart poultry farming IoT systems. A description of how the proposed protocols successfully resisted the well-known attacks and achieved the proposed security properties will be presented.

5.7.1.2.1 Security Properties Assessment

The following section presents a detailed security analysis of the security properties of the three proposed protocols. It demonstrates how the proposed protocols satisfy the security requirements for mutual authentication and session key agreement and resist various known attacks. Then, a comparison with other related schemes is presented.

5.7.1.2.1.1 Protocol 1: parent gateway-child gateway mutual authentication

####### 5.7.1.2.1.1.1 Mutual authentication

During the mutual authentication process, PG and the ChG authenticate each other by verifying the correctness of the X_3, X_5, Y_2, and Y_5. Also, the two sides verify the freshness of the RV_{ChG1}, RV_{ChG2}, RV_{PG1} and generating the *ssk*. The *ssk* is generated by

using the OTP that is known only to PG and the ChG, the nonce values RV_{ChG1} and RV_{ChG2} generated by ChG, and the nonce value (RV_{PG1}) generated by the PG. An adversary can't generate $X_1 = H (OTP \| Auth_P)$ without knowing the ChG_{IDR} of ChG, PG_{IDR} of the PG, and OTP, which are considered secret shared parameters between ChG and PG well as knowing Authp that is only known to ChG. Also, the adversary can't generate $Yx = H (OTP \| Auth_C)$ without knowing the ChG_{IDR} of ChG, PG_{IDR} of the PG, and OTP, which are considered secret shared parameters between ChG and PG well as knowing Authc that is only known to PG. Furthermore,the adversary cannot verify $X3 = H (ChG_{IDR} \| TS_{ChG1} \| RV_{ChG1} \| X_1)$, $Y_2 = H(PG_{IDR} \| TS_{PG1} \| RV_{PG1} \| RV_{ChG1} \| Yx)$, $X_5 = H(ChG_{IDR} \| TS_{ChG2} \| RV_{ChG2} \| (RV'_{PG1} \| Xcc)$, and $Y5 = H (ChG_{IDR} \| TS_{PG2} \| TK_{12} \| TK_{13} \| OTP_X \| ChG_{IDAx|}$ without knowing the PUF responses for both ChG and PG,OTP, PG_{IDR},ChG_{IDR}, $RV_{ChG1,}$ RV_{ChG2}, and RV_{PG1}. As a result, the proposed scheme can achieve mutual authentication between PG and the ChG.

5.7.1.2.1.1.2 Session key agreement and forward /backward security

At the end of the mutual authentication phase, the shared secret key *ssk* is established between ChG and the PG. The *ssk* is a combination of the OTP, RV_{ChG1}, $RV_{ChG2,}$ and RV_{PG1}. The goal of the *ssk* is to protect all the communication between PG and ChG. The *ssk* will be generated locally. The secrecy of the *ssk* depends on the secrecy of OTP, RV_{ChG1}, $RV_{ChG2,}$ and RV_{PG1}. Because all those parameters are randomly selected for every new authentication session, the disclosure of the *ssk* will not cause the compromise of any future *ssk*. The forward/backward security property's objective is to ensure that any past or future shared secret keys will not be affected when any *ssk* is exposed. Even if an adversary obtains *ssk* of a session, he/she can't compute any of the

past and future shared secret keys by using the disclosed *ssk* because the *ssk* is protected by the randomization of the OTP, RV_{ChG1}, $RV_{ChG2,}$ and RV_{PG1}. As a result, the proposed scheme achieves the security of *ssk*.

5.7.1.2.1.1.3 Anonymity unlinkability, and untraceability properties

For fully protected ChG and PG privacy, strong anonymity with unlinkability is required. In the proposed protocol, the ChG's and PG's real identities ChG_{IDR} and PG_{IDR} are not transmitted during all phases in clear text format. Therefore, even if the adversary eavesdrops on all communication messages, it is impossible to obtain real identities. In addition, the new alias ID of the ChG node ChGnewIDA = h (ChG_{IDA}. OTP). Because OTP is fresh in each session, the attacker cannot link any two different ChG_{IDA}'s to the same ChG and cannot trace a given ChG_{IDA} to ChG_{IDR}'s messages. By using a new ChG_{IDA} for each authentication session, the adversary will not be able to decide whether these authentication messages are from the same ChG or not. This means that ChG cannot be linked to different sessions. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the devices by intercepting messages.

5.7.1.2.1.1.4 Node stolen database attack

Due to the nature of the PUF, if an adversary can compromise a locally stored database at one of the communicating parties, he/she will fail to prove himself/herself because the PUF response can't be replicated or predicted. Besides, the secret keys are different every session, which overcomes the issue of having any secret key exposed. Also, the lifetime of the *ssk*'s parameters is only one authentication session, and then they

will be destroyed. Furthermore, the proposed protocol is resilient against any device's full comptonization by employing node behavioral analysis where the node transaction, including traffic type, timing, and frequency, will be monitored and cross-compared to similar devices.

5.7.1.2.1.1.5 Brute force attack

The shared secret key *ssk* is generated locally by both the ChG node and the PG using the random secret parameters OTP, RV_{ChG1}, $RV_{ChG2,}$ and RV_{PG1}. Because this *ssk* depends on random parameters, the adversary cannot obtain it from the protocol. The probability of guessing is so negligible that the adversary will fail to guess the shared secret key's correct parameters, given that this *ssk* changes in every session.

5.7.1.2.1.1.6 Replay attack

The proposed protocol overcame the replay attack by using timestamps and nonce values. The timestamp TS and the nonce value are generated by the sender node and then inserted in the transmitted message in an encrypted format to ensure the adversary cannot replace it. Furthermore, the OTP is used only one time to ensure security against replay attacks. Hence, the protocol protects against the replay attack.

5.7.1.2.1.1.7 Eavesdropping attack

During the authentication phase, the adversary can intercept the messages transmitted between PG and $ChG_{\_1}$. All the intercepted messages will be useless because they are sent encrypted using the XOR and one-way hash function. For the attacker to verify the received parameters, he/she needs to know the PUF responses, the real

identities of the communicating parties, and the OTP that are protected and out of the adversary's reach. The only two parameters that are sent in clear text are the Alias identities and session ID. The Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The adversary will not link the message to a particular device because the proposed protocol uses alias identities that vary in every session. Therefore, the proposed protocol protects against eavesdropping attacks.

5.7.1.2.1.1.8 Impersonation attack

In this attack, when the intruder eavesdrops on the messages transmitted from ChG to PG or vice versa, he/she can use the intercepted information for malicious actions, such as impersonating the ChG device or the PG and sending fabricated messages.

This attack will not succeed for several reasons. First, each device has an Alias ID and Real ID. The real IDs are known only for the communicating parties. The Real IDs will never be released in clear text format. The two sides will verify the real ID of each other before proceeding with the authentication process. Furthermore, without the knowledge of the OTP and the PUF response, the adversary will not be able to generate any of the different authentication parameters to prove itself to the other side.

In the current protocol, if an attacker impersonates GhG and sends (ChG_{IDA} , $TSone_{ChG}$, X_2, X_3) or (ChG_{IDA} , $TSone_{ChG}$, X_4, X_5)which might be previously eavesdropped during the transmission between GhG and PG, to PG. This attack will fail for certain reasons. First, the temporary identity of GhG_{IDA} changes in every session.

Second, both X_3 and X_5 are protected using the message parameters' one-way hash function, which includes the timestamp, OTP, and $GhG_{IDR.}$

On the other side, if the attacker impersonates PG and sends (PG_{IDA}, $TSone_{PG1}$, Y_1, Y_2) or (PG_{IDA}, $TSone_{PG2}$, Y_3, Y_4, Y5) which might be previously eavesdropped during the transmission between ChG and PG, to ChG. This attack will fail because both Y_2 and Y_5 are protected using the secret parameters' one-way hash function, which includes the timestamp, OTP, and PG_{IDR}. Finally, without the knowledge of the OTP, RV_{ChG1}, $RV_{ChG2,}$ and RV_{PG1}, the attacker will not be able to generate the same SK. Thus, the protocol protects against the impersonation attack.

5.7.1.2.1.1.9 Man-in-the-middle attack

As discussed in the replay attack, eavesdropping, and impersonation attacks, the man-in-the-middle attack is defeated because the communication parties' real identities and the secret parameters OTP and the PUF responses are unknown to the attacker. Therefore, the man-in-the-middle attack is prevented.

5.7.1.2.1.2 Protocol 2: child gateway-child gateway mutual authentication

5.7.1.2.1.2.1 Mutual authentication

During the mutual authentication process, ChG_1 and ChG_2 authenticate each other by verifying the correctness of the T_2, T_5, and D_3. Also, the two sides verify the freshness of the RV_{ChG_11}, RV_{ChG_12}, and RV_{ChG_21} and generate the SK. The *ssk* is generated by using the TK known only to ChG_1 and the ChG_2, the nonce values RVChG_11 and RVChG_12 generated by ChG_1 nonce value (RV_{ChG_21}) generated by the ChG_2. An adversary can't generate $T_{2,}$ T_5, and D_3 without knowing the $ChG_{1IDR,}$ ChG_{2IDR}, TK, and MSK. Because the MSK is stored on each side in an encrypted format

using the PUF response of that side as an encryption key, both ChG_1 and the ChG_2 need to compute their PUF responses to retrieve MSK. As a result, the proposed scheme can achieve mutual authentication between ChG_1 and ChG_2.

5.7.1.2.1.2.2 Session key agreement and forward/backward security

After completing the mutual authentication phase, the shared secret key *ssk* is established between ChG_1 and ChG_2. The *ssk* is a combination of the TK, RV_{ChG11}, $RV_{ChG12,}$ and RV_{ChG21}. The goal of the *ssk* is to protect all the communication between ChG_1 and ChG_2. The *ssk* is generated locally. The secrecy of the *ssk* depends on the secrecy of the TK, RV_{ChG11}, $RV_{ChG12,}$ and RV_{ChG21}. Because all those parameters are randomly generated for every new authentication session, the disclosure of the *ssk* will not cause the compromise of any future *ssk*. Even if an adversary obtains *ssk* of a session, he/she can't compute any of the past or future shared secret keys by using the disclosed *ssk* because the *ssk* is protected by the randomization of the TK, RV_{ChG11}, $RV_{ChG12,}$ and RV_{ChG21}. As a result, the proposed scheme achieves the security of *ssk*.

5.7.1.2.1.2.3 Anonymity unlinkability, and untraceability properties

For fully protected ChG_1 and ChG_2 privacy, strong anonymity with unlinkability is required. In the proposed protocol, the ChG's and ChG's real identities $ChG_{\_1IDR}$ and $ChG_{\_2IDR}$ are not transmitted during all phases in clear text format. Therefore, even if the adversary eavesdrops on all communication messages, it is impossible to obtain real identities. In addition, a new alias ID will be assigned to each ChG for every authentication session as presented in protocol 1. By using a new ChG_{IDA} for each authentication session, the adversary will not be able to decide whether these authentication messages are from the same ChG or not. This means that ChG cannot be

linked to different sessions. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the devices by intercepting messages.

5.7.1.2.1.2.4 Node stolen database attack

If the $ChG_{\_1}$ or the $ChG_{\_2}$ is hacked, the MSK is encrypted by the PUF response of the ChG. For the adversary to retrieve the MSK, he/she needs to calculate the PUF response, which is impossible due to the nature of the PUF that can't be predicted or replicated. From another perspective, the *ssk* is different every session, which overcomes the issue of having any secret key exposed. Furthermore, the proposed protocol is resilient against any device's full compromising by employing node behavioral analysis where the node transaction, including traffic type, timing, and frequency, will be monitored and cross-compared to similar devices.

5.7.1.2.1.2.5 Brute force attack

The shared secret key *ssk* is generated locally by both the $ChG_{\_1}$ node and the $ChG_{\_2}$ using the random secret parameters TK, RV_{ChG11}, $RV_{ChG12,}$ and RV_{ChG21}. Because this *ssk* depends on random parameters, the adversary cannot obtain it from the protocol. The probability of guessing is so negligible that the adversary will fail to guess the shared secret key's correct parameters, given that this *ssk* changes in every session.

5.7.1.2.1.2.6 Replay attack

The proposed protocol overcame the replay attack by using timestamps and nonce values. The timestamp TS and nonce values are generated by the sender node and then inserted in the transmitted message so that it ensures the attacker cannot replace it. Before trusting the received TS, the receiver verifies the TS's integrity by recalculating the

received hash value that includes secret parameters between the two communicating parties. The attacker cannot reconstruct the hash value because he/she does not know the TK, MSK, and the real identities of the communicating parties. Therefore, any replay attack attempts will fail. Hence, the protocol protects against replay attack.

5.7.1.2.1.2.7 Eavesdropping attack

During the authentication phase, the adversary can intercept the messages transmitted between $ChG_{\_1}$ and $ChG_{\_2}$. All the intercepted messages will be useless because they are sent in an encrypted format using the XOR function and one-way hash function. For the attacker to verify the received parameters, he/she needs to know the TK and MSK, which are composed of random values and are protected and out of the adversary's reach. The TK is changed every authentication session, and the PUF response encrypts the MSK. The only two parameters that are sent in clear text are the Alias identities and session ID. The Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The adversary will not link the message to a particular device because the proposed protocol uses alias identities that change in every session. Therefore, the proposed protocol protects against eavesdropping attacks.

5.7.1.2.1.2.8 Impersonation attack

In this attack, when the intruder eavesdrops on the messages transmitted from $ChG_{\_1}$ to $ChG_{\_2}$ or vice versa, he/she can use the intercepted information for malicious actions, such as impersonating the $ChG_{\_1}$ device or the $ChG_{\_2}$ and sending fabricated messages.

This attack will not succeed for several reasons. First, each device has an Alias ID and Real ID. The real IDs are known only for the communicating parties. The Real IDs will never be released in clear text format. The two sides will verify the real ID of each other before proceeding with the authentication process. Furthermore, without the knowledge of the TK and the MSK, the adversary will not be able to generate any of the authentication parameters to prove himself/herself to the other side.

In the current protocol, if an attacker impersonates $GhG_{\_1}$ and sends (ChG_{IDA}, TS_{ChG_11}, $T_{1,}$ T_2) or (ChG_1_{IDA}, S_{ID}, $TS_{CHG\_12,}$ T_4, T_5), which might be previously eavesdropped during the transmission between $GhG_{\_1}$ and $ChG_{\_2}$, to $ChG_{\_2}$, this attack will fail for certain reasons. First, the temporary identity of GhG_{IDA} changes in every session. Second, for T1, T2, T4, and T5, the adversary will not be able to reconstruct them without knowing the TK, MSK, the real identity of the receiving node, and its own real identity.

On the other side, if the attacker impersonates $GhG_{\_2}$ and sends (ChG_2_{IDA}, S_{ID} ,D2, D3), which might be previously eavesdropped on during the transmission between $ChG_{\_1}$ and $ChG_{\_2}$, to $ChG_{\_1}$, this attack will fail because D3 is protected using the one-way hash function of the message parameters. D_2 message includes TK and $RV_{ChG,}$ which are different for every authentication session, and MSK that is encrypted by $GhG_{\_2}$ PUF response. Finally, without knowing the TK, RV_{ChG11}, $RV_{ChG12,}$ and RV_{CHG21}, the attacker will not be able to generate the same *ssk*. Thus, the protocol protects against the impersonation attack.

5.7.1.2.1.2.9 Data modification attacks

Assuming that M1 < ChG_{IDA1}, TS_{ChG_11}, T_1, T_2 > has been tampered by an adversary M1* < ChG_{IDA1}^{*}, $TS_{ChG\_11}^{*}$, T_1^{*}, T_2^{*} >. After receiving the message, the adversary will first need to have TK and ChG_2_{IDR}, which are secure parameters to compute Rj <H (TK|| ChG_2_{IDR})> and T_1< $RV_{ChG\_11} \oplus$ Rj>. Furthermore, without having all secret parameters, the adversary will not be able to generate a legitimate T2 = H <ChG_1_{IDR} || TS_{ChG_11} || RV_{ChG_11} ||Rj>, that is, if the message has been tampered, the ChG_1 is considered to be illegal, and the session will be terminated immediately by the ChG_2.

Now suppose the message M2 < ChG_2_{IDA}, S_{ID}, TS_{ChG12}, D2, D3> has been tampered with by an adversary and sent to V $M2^{*}$ < $ChG\_2_{IDA}^{*}$, S_{ID}^{*}, TS_{ChG12}^{*}, D2*, $D3^{*}$ >. After receiving the message, the adversary will first need to compute <D_1 = H (RV_{ChG_11} ||MSK || TK)> and <D2 = $RV_{ChG\_21} \oplus D_1$> using TK, MSK, and the random generated value received from ChG_1. Because each ChG_2 has a PUF chip and each chip is unique, the adversary will not be able to replicate or predict the response(R) of the real ChG_2. Consequently, the adversary will not be able to generate R and decrypt the MSK to compute D_1. Also, the TK is secret and is only known to the ChG_1 and ChG_2, the adversary will not correctly calculate D_1 and D_2. Finally, without having all above-mentioned secret parameters, the adversary will not be able to compute <D_3 = H (ChG_2_{IDR} || TS_{ChG_21} || RV_{ChG_21}|| D_1)> to pass the ChG_2 authentication.

As for the message M3 <ChG_1_{IDA}, S_{ID}, TS_{CHG12}, T_4, T_5)> suppose the message M3 was tempered by an adversary. Because ChG_1 has a PUF chip and each chip is unique, the adversary will not be able to replicate or predict the response(R) of the real ChG_2. Consequently, the adversary will not be able to generate R and decrypt the MSK

to compute T_3. Furthermore, also the TK is secret and is only known to the ChG_1 and ChG_2; the adversary will not correctly calculate T_3 and T4. Furthermore, the adversary will not be able to construct a valid T5and send a legitimate message to ChG_2 that can pass the authenticate verification. So, o after receiving the message $M3^*$ and verifying it, ChG_2 will terminate the session with ChG_1. In summary, the proposed protocol can resist all message modification attacks.

5.7.1.2.1.2.10 Man-in-the-middle attack

As discussed in the replay attack, eavesdropping, and impersonation attacks, the man-in-the-middle attack is defeated because the communication parties' real identities and the secret parameters TK and MSK are unknown to the attacker. Therefore, the man-in-the-middle attack is prevented.

5.7.1.2.1.3 Protocol 3: mini-gateway _IoT node mutual authentication

5.7.1.2.1.3.1 Mutual authentication

N and the MG authenticate each other during the mutual authentication by verifying the SN, Y_3, and X_4. An adversary cannot generate $S_1 = h (N_{IDR} || MG_{IDR}|| R_{MG})$ without knowing the N_{IDR} of N and the MG_{IDR} of the MG and compute the PUF response (R_{MG}). Also, the adversary cannot verify $S_2 = h (S_1 ||MG_{IDR}) \oplus G_{IDN}$ and $S_3 = h ((S_2|| MG_{IDR}) \oplus T_{IDN}$ without knowing the G_{IDN} and T_{IDN}. Furthermore, the adversary will not be able to generate Y3 = H ($MG_{IDR} || TS_{MG1} || RV_{N1} || RV_{MG1} ||C||Y_1$) without having S_1, S_2 OTT, $MG_{IDR,} || RV_{MG1}$ and RV_{N1}. Also, the adversary cannot generate X_4 without having the R that can only be generated by N and having CHX, which is considered a

secure parameter between N and MG. As a result, the proposed scheme can achieve mutual authentication between N and MG.

5.7.1.2.1.3.2 Session key agreement and forward/backward security

After successfully completing the mutual authentication phase, the shared secret key *ssk* is established between N and the MG. The *ssk* is a combination of RV_{N1}, RV_{N2}, RV_{MG1}, and Ri. The *ssk* will be generated locally. The secrecy of the *ssk* depends on the secrecy of RV_{N1}, RV_{N2}, RV_{MG1}, and Ri. Because all those parameters are randomly selected for every new authentication session, the disclosed *ssk* will not cause the compromise of any future *ssk*. Even if an adversary obtains the *ssk* of a session, he/she can't compute any of the past or future shared secret keys by using the disclosed *ssk*. The *ssk* is protected by the uniqueness of the Ri where each CRP will be used only one time. The corresponding responses can't be replicated or predicted because they are unique for each N because of the nature of the PUF. Second: the randomization of the RV_{N1}, RV_{N2}, and RV_{MG1}. As a result, the proposed scheme achieves the security of *ssk*.

5.7.1.2.1.3.3 Anonymity unlinkability and untraceability properties

Protecting the privacy of N requires strong anonymity with unlinkability. The N's real identity NIDR is not transmitted during all phases in clear text format in the proposed protocol. Therefore, even if the adversary eavesdrops on all communication messages, it is impossible to obtain the IoT node's real ID. In addition, the new alias ID of the IoT node $NIDA^2 = h(NIDA^1.RV_{MG})$. Because $RV_{MG}1$ is fresh in each session, the attacker cannot link any two different N_{IDA}'s to the same N and cannot trace a given IoT device to N's messages. By using a new N_{IDA} for each authentication session, the adversary will not be able to decide whether these authentication messages are from the

same N or not. This means that device N cannot be linked to different sessions. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the devices by intercepting messages.

5.7.1.2.1.3.4 Node stolen database attack

If any IoT node is compromised and the adversary can steal S_1, S_2, S_3, N_{IDR}, CHX, and the OTT from the IoT node's database to impersonate the IoT node, the fake node will fail the CRP verification process. The proposed protocol is resilient against a full compromising of the IoT node by employing node behavioral analysis where the node transaction, including traffic type, timing, and frequency, will be monitored and cross-compared to the previous data from the same virtual domain. Also, if an MG was compromised and the adversary can steal its database, the adversary will fail to generate R_{MG} to compute S1 that is needed for N to authenticate the MG.

5.7.1.2.1.3.5 Brute force attack

The shared secret key *ssk* is generated locally by both the IoT node and the MG using the random secret parameters RV_{N1}, RV_{N2}, RV_{MG1}, and Ri. Because this *ssk* depends on random parameters, the adversary cannot obtain it from the protocol. The probability of guessing is so negligible that the adversary will fail to guess the shared secret key's correct parameters, given that this *ssk* changes every session.

5.7.1.2.1.3.6 Replay attack

The sender node generates the timestamp TS and is then inserted in the transmitted message to ensure the attacker cannot replace it. Before trusting the received TS, the receiver verifies the TS's integrity by recalculating the received hash value that includes secret parameters between the two communicating parties. Furthermore, Each

CRP is used only one time to ensure security against replay attacks. Hence, the proposed protocol protects against the replay attack

5.7.1.2.1.3.6 Eavesdropping attack

During the authentication phase, the adversary can intercept messages transmitted between N and the MG. All the intercepted messages will be useless because they are sent in an encrypted format using XOR and a one-way hash function. To verify the received parameters, the attackers need to know the R_{MG}, OTT, CHX, N_{IDR}, M_{GIDR}, and Ri, which are composed of random values and are protected and out of the adversary's reach. The only two parameters that are sent in clear text are the Alias identities and session ID. the Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The adversary will not be able to link the message to a particular device because the proposed protocol uses alias identities that change in every session. Therefore, the proposed protocol protects against eavesdropping attack.

5.7.1.2.1.3.7 Impersonation attack

In this attack, when the intruder eavesdrops on the messages transmitted from N to MG or vice versa, he/she can use the intercepted information for malicious actions, such as impersonating the IoT device or the MG and sending fabricated messages.

5.7.1.2.1.3.7.1 MG impersonation

This attack will not succeed for several reasons. First, Each MG has an Alias ID and Real ID. The real ID is known only for IoT devices. The MG's Real ID will never be

released in Clear Text format. The IoT device will verify the MG's Real ID before proceeding with the authentication process. Second, the proposed protocol uses the Y_3 value to authenticate the MG node. Without the knowledge of S_1, S_2 OTT, $MG_{IDR,}$ RV_{MG1}, and RV_{N1}, the attacker cannot construct a valid Y_3. To compute S_1, the adversary needs to have the PUF response R_{MG} that the MG can only generate.

5.7.1.2.1.3.7.2 IoT node impersonation

To impersonate the IoT node, an adversary should intercept the messages exchanged in the previous sessions. This attack will fail for numerous reasons. First, the Alias ID of the IoT node changes every authentication session. Second, the fake IoT node will not be able to generate legitimate responses because the PUF responses are unique for each device. Third, X_4 is protected using the one-way hash function of the message, including CHX, Ri, and TS. Thus, the protocol protects against the impersonation attack.

From another perspective, even if the adversary succeeded in compromising an IoT node, the adversary's further attacks using the compromised node only affect the communication related to that node. Because each IoT node has its own secret keys, the adversary can't derive other non-compromised IoT nodes' keys without knowing those nodes' random information. Therefore, further attacks will not affect other communications. As a result, the proposed scheme is resistant to IoT node impersonation attacks.

5.7.1.2.1.3.8 Data modification attacks

Assuming that M1 < N_{IDA}, TS_{N1}, X_2, SN > has been tampered by an adversary M1* < $N_{IDA}{}^{*,}$ $TS_{N1}{}^{*,}$ $X_2{}^{*,}$ SN * >. After receiving the message, the adversary will first need to have S_2, S_3, OTT, and $N_{IDR,}$ which are secure parameters to compute X_1 <H (S_2

$||S_3$) and $X_2 < RV_{N1} \oplus OTT>$. Furthermore, without having all secret parameters, the adversary will not be able to generate a legitimate $SN = < H (N_{IDR}||TS_{N1} || RV_{N1}|| X_1)>$, that is, if the message has been tampered, N is considered to be illegal, and the session will be terminated immediately by the MG.

Now suppose the adversary needs to modify M2 $<MG_{IDA}, S_{ID,} TS_{MG1,} Y_2, C', Y_3 >$ to be $M2^* < MG_{IDA}{}^*, S_{ID}{}^*, TS_{MG1}{}^*, Y_2{}^*, C'^*, Y_3{}^* >$. After receiving the message, the adversary will first need to compute $<Y_1 = H (S'_1|| S'_2) >$ and $<Y2 = RV_{MG1} \oplus OTT>$ using $S'_1 S'_2$ and OTT. To compute S1, the adversary needs to have the PUF response (R_{MG}). Because the MG has a PUF chip and each chip is unique, the adversary will not be able to replicate or predict the response (R_{MG}) of the real ChG_2. Consequently, the adversary will not be able to generate R to compute $S_{1.}$ Also, the OTT is secret and is only known to the N and MG, and the adversary will not correctly calculate Y_1 and Y_2. Finally, without having ll above mentioned secret parameters, the adversary will not be able to compute $<Y_3 = H(MG_{IDR} || TS_{MG1} || RV_{N1} || RV_{MG1} ||C||Y_1)>$ to pass the N authentication.

As for the message $M3 < N_{IDA,} TS_{N2}, X_{3,} Rx, X_4)>$ suppose an adversary tempered the message M3. Because N has a PUF chip and each chip is unique, the adversary will not be able to replicate or predict the response (Ri) to compute CHX^{+1} of the real N. Also, the OTT is secret and is only known to N and MG. Therefore, the adversary will not correctly calculate X_3 and $Rx_{.}$ Furthermore, the adversary will not be able to construct a valid X_4 and send a legitimate message to MG that can pass the authenticate verification. So, o after receiving the message $M3^*$ and verifying it, MG will terminate the session with N. In summary, the proposed protocol can resist all message modification attacks.

5.7.1.2.1.3.9 Man-in-the-middle attack

As discussed in the replay attack, eavesdropping, and impersonation attacks, the man-in-the-middle attack is defeated because of the employment of the real identity of IoT node N_{IDR}, the timestamp TS, the one-way hash function, the nonce values generated by both the IoT and MG, and the CRPs that are unknown to the attacker. Moreover, the chained-hash value CHX is protected using a one-way hash function. Therefore, the man-in-the-middle attack is prevented.

5.7.1.2.1.3.10 Modeling attacks

It is an attack where the adversary collects a large number of PUF CRPs and uses a regression algorithm to build a model and predict responses for new challenges. All the challenges and their corresponding responses are transferred in an encrypted format by using a a one-way hash function. Therefore, the adversary will not be able to collect enough useful CRPs to build a model and predict responses to new challenges.

5.7.1.2.1.3.11 Physical attack

Any attempt to tamper with the IoT device will change the PUF embedded chip's behavior and, consequently, renders the PUF useless.

5.7.1.2.1.3.12 IoT Device Counterfeit/Cloning

The IoT device is authenticated through the use of the PUF. PUF responses are treated as hardware fingerprints that cannot be duplicated or cloned. Therefore, by the use of the PUF, IoT devices cannot be cloned.

5.7.1.2.2 Comparison of Security Features

We compare the proposed MG-IoT protocol security features with other related authentication and key agreement protocols [97,104,105,106,95]. Table 19 shows the comparison results. The table indicates that the proposed protocol is secure against all the imperative security threats and accomplishes diverse security features. The protocols [104-105] used the IoT node real identity to exchange messages with the server, which is against protecting the anonymity and the unlinkability of the IoT devices. Furthermore, the researchers in [106] do not defend their protocol against the replay attack. Also, none of the proposed protocols protects against IoT device counterfeiting, database stealing, or physical attacks.

On the other side, the proposed Mg-IoT protocol supports multiple essential security features: First, the proposed protocol does not require any secret key storage or secret key sharing over the network. Second, all secret keys are computed locally on all the communicating parties. Third, the mini-gateway and the IoT device use their one-time alias identity for each authentication session in the proposed protocol. Therefore, it will be difficult for an outside adversary to comprehend the activities of the IoT devices. Fourth, the proposed protocol used the PUF to protect the IoT device against counterfeiting and used the PUF responses as an encryption key to protect the device database if the device got compromised. From table 19, we can conclude that the proposed protocol in this chapter can support all the desired security properties, which are essential for IoT devices' security.

Table 19: Security feature comparison of the proposed mini-gateway _IoT node protocol with other related mutual authentication and key agreement schemes.

Security Property	[97]	[104]	[105]	[106]	[95]	Proposed protocol
Resilience to IG the Impersonation Attack	No	No	Yes	Yes	Yes	Yes
Resilience to IoT the Impersonation Attack	Yes	No	Yes	Yes	Yes	Yes
Resilience to Replay Attack	Yes	Yes	Yes	No	Yes	Yes
Resilience to Device Counterfeit	No	No	No	No	No	Yes
Resilience to Modeling Attack	NA	NA	NA	NA	NA	Yes
Anonymity and Untraceability	Yes	No	No	Yes	Yes	Yes
Resilience to Physical attacks	No	No	No	No	No	Yes
Resilience to Database stealing	No	No	No	No	No	Yes
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	Yes

5.7.2 Performance Analysis

In this section, we present a performance analysis of the proposed protocol. The performance analysis evaluates the storage requirements. Also, we analyze the presented protocol's overhead and efficiency in terms of computational complexity and communication overhead and compare its computational complexity and communication cost with the most relevant protocol in the literature proposed by [97], [104],[105], [106] and [95].

5.7.2.1 Storage requirements

Table 20: Storage cost of the proposed scheme

Node	**Storage cost (in bits)**
N	128+ 256 * 5 + 8 * 3 = **1432 b**
MG	128*3+ 256 * 2 + 8 * 5 = **936 b**

Table 20 presents the storage requirements for the proposed protocol. The calculation considered only the authentication protocol between the IoT node and the mini gateway. In the proposed protocol, each IoT node must store its real identity N_{IDR} alias identity N_{IDA}, the real identity of the MG_{IDR}, alias identity MG_{IDA}, authentication parameters S1, S2 and S3, Chained hash CHX,and one-time token OTT. We use SHA-1 and SHA-2 as two examples of hash function, and the output of SHA-1 is 160 bits and SHA-2 is 256 bits. By applying these settings, we obtain $|N_{IDA}| = 128$ bit. $|S_1| = |S_2| = |S_3| = |CHX| = |OTT| = 256$ bits and $|N_{IDR}| = |MG_{IDR}| = |MG_{IDA}| = 8$ bits.

On the other hand, MG is required to store the tuple N_{IDA}, N_{IDR}, MG_{IDA}, MG_{IDR}, G_{IDN}, T_{IDN}, OTT, C and R. By applying these settings, we obtain $|N_{IDA}| = 128$. $|N_{IDR}| = |MG_{IDR}| = |G_{IDN}| = |MG_{IDA}|$ $|T_{IDN}| = 8$ bits, $|C| = |R| = 128$ bits, and $|CHX| = |OTP| = 256$ bits.

5.7.2.2 Computational Complexity Analysis

We compare the computation cost of the proposed scheme with other related schemes [97], [104],[105], [106] and [95]. We only focus on comparing the mutual authentication phase between the MG and the IoT. Because the time for executing a bitwise XOR operation is negligible, we do not consider XOR operations for

computational cost analysis. To analyze the performance of the proposed protocol with respect to other presented protocols in the literature, we conducted simulations of the cryptographic operations using Dell Inspiron Laptop with Intel Core i7, dual-core 2.7 GHz CPU, and 8 GB RAM to act as the IG. we used a Raspberry Pi 4 Model B with 64-bit quad-core cortex A-72 processor and 1 GB RAM to simulate an IoT device. The simulations used PyCryptodome cryptographic and Fastecdsa libraries in Python. For these results, we considered the 128-bit arbiter PUF for PUF operation. The fuzzy extraction's execution time is almost the same as the ECC point multiplication and the execution time for modular exponentiation is double the time of the execution of the ECC point multiplication [58]. Table 21 presents the used notation and the execution time of each operation.

Table 21: Execution time of the cryptographic operations

Operation	**Computation Time on IoT**	**Computation Time on MG**
H: time for executing a one-way hash function	0.002 ms	.001 ms
F: time for executing a fuzzy extractor	5 ms	4 ms
EM: time for executing an ECC point multiplication	5 ms	4 ms

EXP: time for a modular exponentiation	10 ms	8 ms
HMAC: time for executing the HMAC	2.7 ms	1.5 ms
ENC: Time for Executing (AES-CBC Encryption)	0.18 ms	.14 ms
DEC: Time for executing (AES-CBC Decryption)	0.18 ms	.14 ms
PUF: Time for executing PUF (128-bit Arbiter)	.12 ms	.12 ms

Table 22: Comparison of computation costs for the authentication phase of the proposed scheme and other related schemes.

Entity	[97]	[104]	[105]	[106]	[95]	Proposed Protocol
IoT	$7T_h$	$1T_{HMAC}$	$3T_h + 2T_{TSE} + T_{TSD}$	$2T_{TP} + 3T_h$	8^{Th}	$10T_h + 2T_{PUF}$
CosT	7*.002 = **.014 ms**	1* 2.7 = **2.7ms**	3*.002 + .18*2 +.18 = **.55 ms**	2*5 + 3*.002 = **10.006 ms**	8*.002 = **.016 ms**	10 * .002 +.12 *2 = **.26 ms**
MG	$8T_h$	$1T_{HMAC}$	$3T_h + T_{TSE} + 2T_{TSD}$	$2T_{TP} + 3T_h$	8^{Th}	$13T_h + T_{PUF}$
Cost	8 *.001 = **.008 ms**	1 * 1.5 = 1.5 **ms**	3*.001 +2*.14 +.14 = **.45 ms**	2* 4 + 3*.001= **8.003 ms**	8 *.001 = **.008 ms**	13*.001 +.1 = **.13 ms**
Total Cost	**.022 ms**	**4.2 ms**	**1 ms**	**18.009 ms**	**.024 ms**	**.39 ms**

Table 22 summarizes the computational cost comparison between the proposed protocol and the other related protocols presented in the literature. The total run time of the proposed protocol is .39 ms. When comparing the proposed protocol with other related protocols, we found that the proposed protocol is more efficient than [104] that used HMAC to complete the mutual authentication process, and [105] that used symmetric encryption. Furthermore, the proposed proposal is more efficient than [106] that used scaler multiplication. Although the proposed protocol's computable cost is more than [97 and 95], the proposed protocol does not require any secret key storage and is secure against database stealing. One of the main strengths of the proposed protocol is securing the IoT devices from physical attacks and counterfeiting by employing the PUF and the CHX.

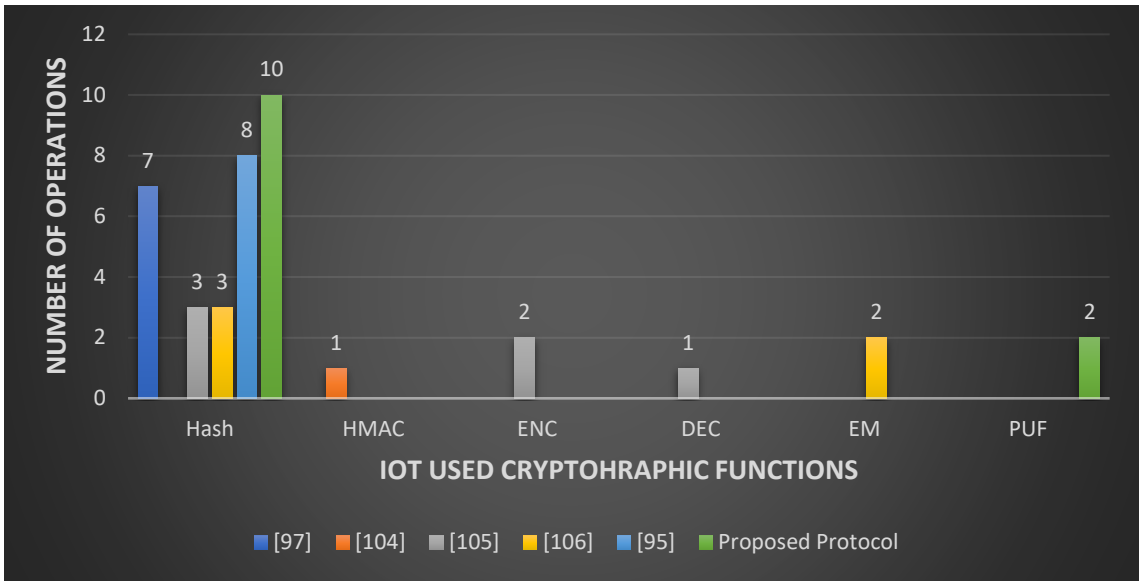


Figure 76: Comparison of the number of IoT cryto operations across the protocols

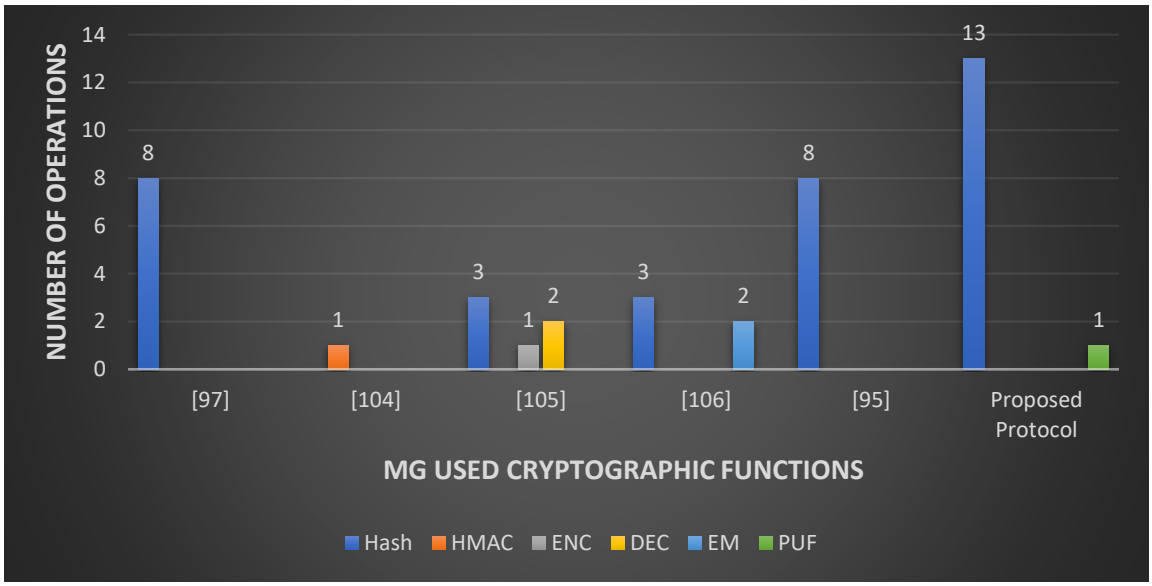


Figure 77: Comparison of the number of the MG crypt operations across the protocols

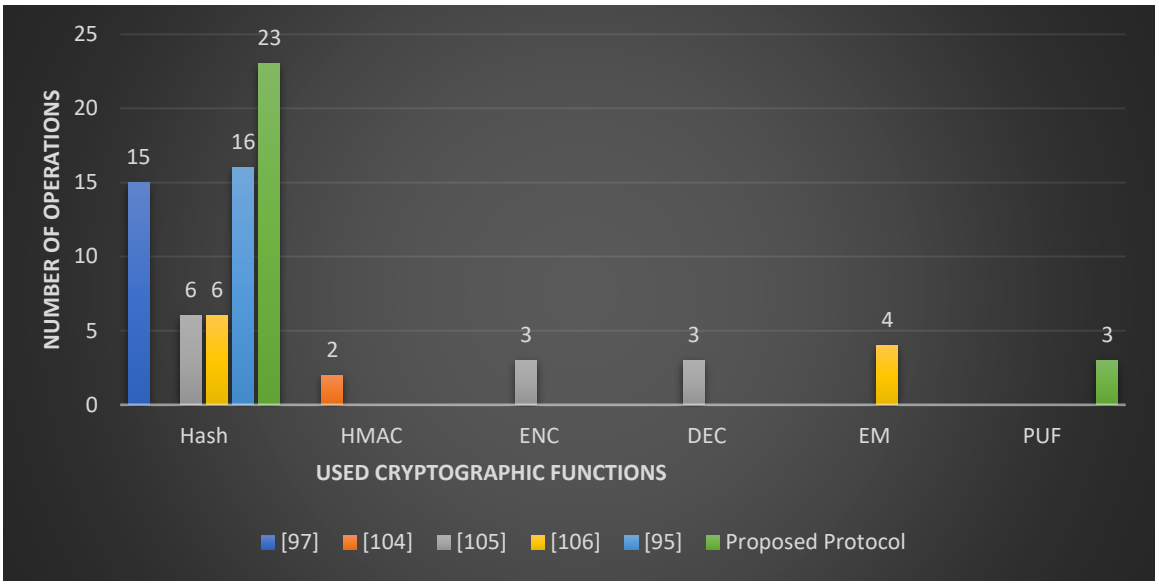


Figure 78: Comparison of the total number of crypto operations across the protocols

Table 22 and figures 76,77 and 78 summarize the computational complexity comparison between the proposed protocol and the other related protocols presented in the literature. Table 22 listed out the number of operations of each type for each protocol considered. Figure 76 displays the number of authentication operations of each type that are completed on the IoT side. Figure 77 depicts the number of authentication operations of each type that are completed on the MG side. Figure 79 demonstrates the total number of cryptographic operations of each type that are complete for each protocol. As depicted from the table and the figures, the proposed protocol used more hash functions than most of the other protocols to avoid computationally expensive cryptographic operations such as ECC and HMAC and, at the same time, achieve the same security goals. The proposed protocol is the only protocol that used PUF as a hardware fingerprint to secure the devices from counterfeiting and avoid storing long-term security keys on the communicating devices. The proposed protocol is the only one that used PUF on the MG

side to protect the protocol from MG impersonation and secure the MG database to be stolen in case the device is compromised.

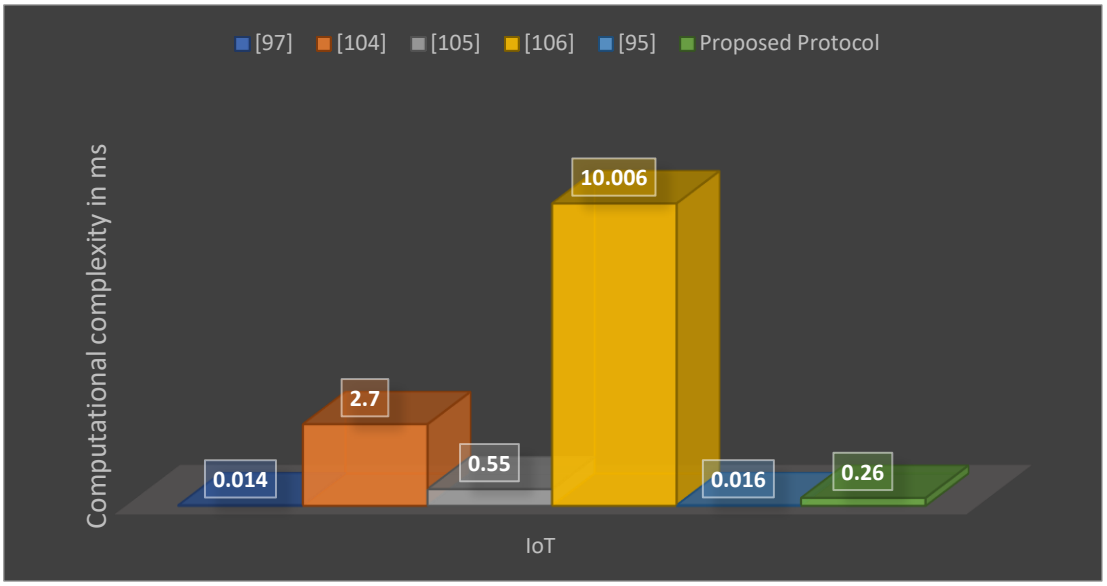


Figure 79: The IoT computational time across the different protocols

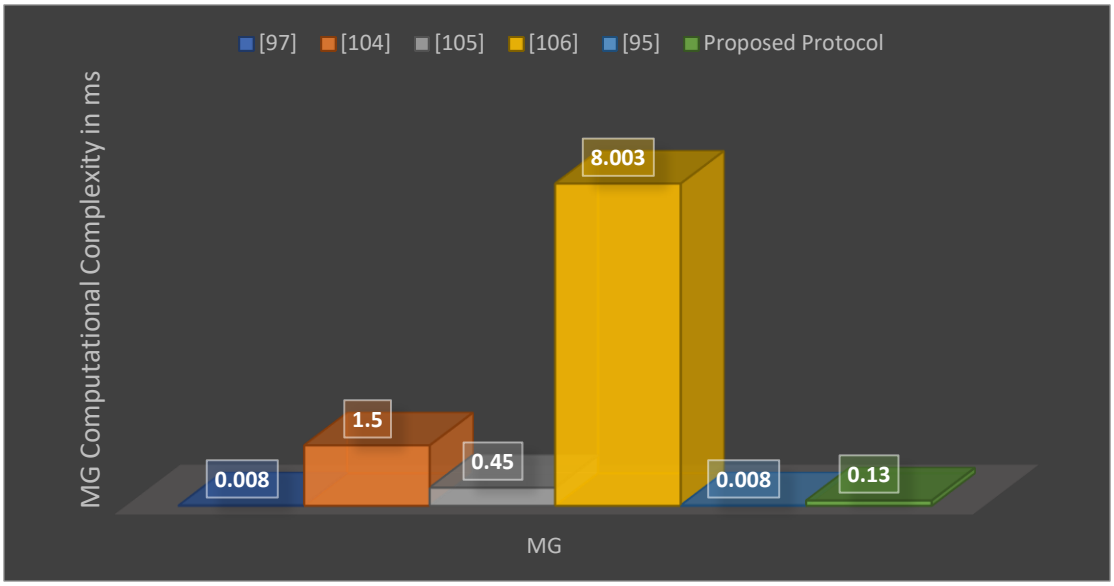


Figure 80: MG computational time across the different protocols

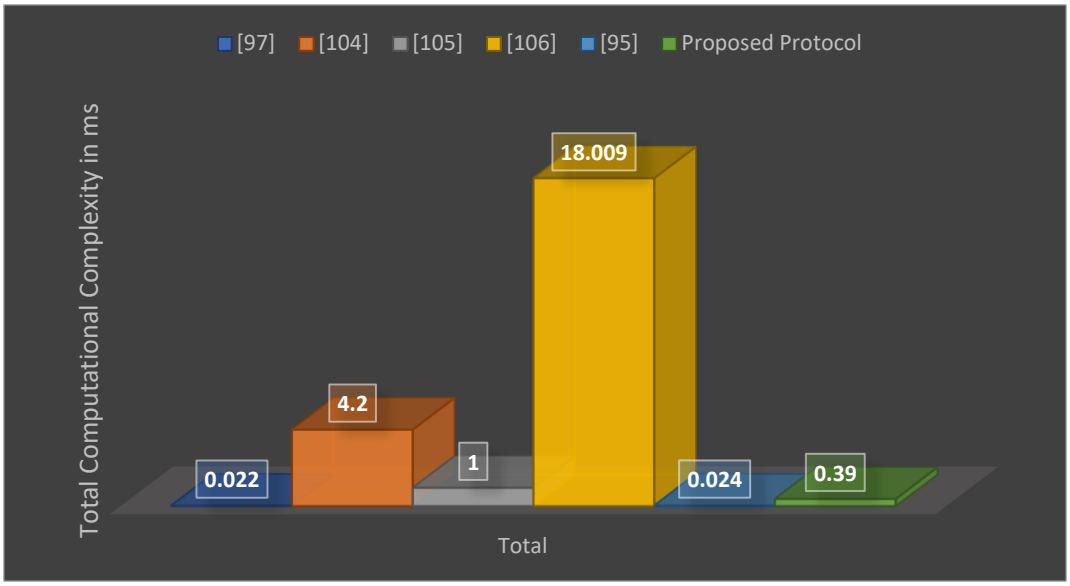


Figure 81: Total computational time across the different protocols

Table 22 and figures 79, 80 and 81 demonstrate the cost comparison of the authentication protocol with other similar protocols. Table 22 and figure 81 present the total computational cost of each protocol. Figure 79 displays the computation cost of each protocol for the IoT side, and figure 80 depicts the computational cost of each protocol for the MG side. The results indicate that the proposed protocol has a lower computational cost for the IoT side than [104,105, and 106]. Although the proposed protocol is higher than [97 and 95], the proposed protocol is resilient against impersonation attacks and does not require long-term secret key storage. The above results prove that the proposed protocol is suitable for IoT resource-constrained devices.

From table 22 and figure 81, we observe that while [97], [104],[105],[106], and [95] take approximately 0.022, 4.2, 1, 18.009, and 0.024 ms, the proposed protocol takes only .39 ms. The protocol in [106] performs the worst due to the high number of ECC point operations involved. Also, the protocol in [104] has a bad performance due to the

use of HMAC, which is computationally expensive. Furthermore, protocol [105] has a high computational cost due to multiple encryption and decryption calculations.

We can conclude that the proposed protocol has a higher security level than the rest of the protocols. The proposed protocol achieved the main security goals of confidentiality, integrity, and authenticity besides accommodating IoT devices that are diverse in their capabilities.

5.7.2.3 Communication cost analysis

The following section analyzes the communication cost of the proposed MG-IoT protocol. To reduce network congestion and provide fast message transmission, the communication costs of the protocol should be as low as possible. For the communication cost analysis, we evaluate the communication cost in terms of the size of the message in bits. Then, we compare the proposed protocol to the other related protocols. Table 23 presents a summary of the sizes of the message parameters. Table 24 lists the message parameters that are communicated between the server and the IoT device, along with their sizes.

Table 23: Size of the message parameters

Message Parameters	**Size in Bits**
ID of N [[57], [62], [63]	128
ID of server [63]	8
S_{ID} [57], [62]	8
Nonces [61], [63]	128

CRP (C, R) [61], [62]	128
HMAC [63]	256
Hash Function [57], [62]	256
Timestamp (TS) [63]	48
ECC [60], [59]	256
MAC [62]	256

- **Message 1**: In the transmission (N → MG), N sends the tuple, N_{IDA}, TS_{N1}, X_2, $S_{N.}$ Therefore, the size of this tuple is 128 + 48 + 128+256 = **560 bits**.
- **Message 2**: In the transmission (MG 1→ N), IG sends the tuple, $MG_{IDA,}$ S_{ID}, TS_{MG1}, Y_2, C', $Y_{3.}$ Therefore, the size of this tuple is 8 +8 + 48 + 128+128+256 = **576 bits**.
- **Message 3**: In the transmission (N → MG), N sends the tuple, N_{IDA}, S_{ID}. $TS_{N2,}$ X_3, R_X, $X_{4.}$ Therefore, the size of this tuple is 128 + 8 + 48+128 +128+256+256 = **952. bits**.

Table 24: Communication cost comparison (bits)

Message Number	**[97]**	**[104]**	**[105]**	**[106]**	**[95]**	**Proposed scheme**
M_1: N→ S	640 bits	136 bits	136 bits	128 bits	560 bits	560 bits
M_2: S→ N	512 bits	568 bits	296 bits	512 bits	560 bits	576 bits

M_3: N→S	128 bits	256 bits	672 bits	256 bits	304 bits	568 bits
M_4: S→ N	------	------	536 bits	----	304 bits	---
M_5: N→S	------	------	144 bits	----	---	
Total	**1280 bits**	**960 bits**	**2056 bit**	**896 bits**	**1728 bits**	**1824 bits**

Table 24 presents the communication cost of the proposed protocol and the comparison schemes. Four protocols have less communication cost than the proposal, which are [97,104,106 and 95] with differences of 30%, 47%, 51% and 5% respectively. The proposed protocol requires sending only three messages to achieve mutual authentication, and they contain timestamps, XOR-ciphered random numbers, and hash functions. Sending this information prevents attacks such as data modification, replay, and impersonation [28]. The protocols that have less communication cost do not send all this information, and some of them used real identities of the IoT nodes during the communication process; consequently, they have been found vulnerable to some of the attacks, as can be seen in the above-presented comparison. The proposed protocol has a lower communication cost than [105] and a reasonable cost compared to [95]. Therefore, even if the proposal is not the protocol with the lowest communication cost, it is suitable for IIoT. At the same time, it achieves more security properties than the protocols with less cost.

5.8 Summary

This chapter presents a mutual authentication scheme for M2M communication in IIoT. The proposed scheme introduces a mechanism for secure session key establishment. the proposed scheme targets to have low computational complexity and low computational cost to be suitable for resource-constrained IIoT devices. To achieve the lightweight feature, the scheme is based on the lightweight operations XOR and hash functions. The scheme provides confidentiality, integrity, anonymity, unlinkability, and untraceability capabilities while achieving mutual authentication between the communicating devices. The proposed scheme is assessed using both security and performance analysis. The security analysis used both formal and informal techniques. The formal evaluation employed used the BAN logic and AVISAP simulation to verify the proposed protocol's security. The informal security analysis also showed that the presented scheme is resistant to common attacks and satisfies the main security properties such as confidentiality, integrity, mutual authentication, perfect forward and backward secrecy. Furthermore, we evaluated the MG-IoT protocol's efficiency in terms of storage requirements, communication cost, and computational complexity and compared it with other related protocols. The results proved that the proposed protocol has a higher attack resistance compared to the other protocols.

In summary, the presented protocol achieves the required low computational complexity for resource-constrained IIoT devices. Furthermore, the proposed protocol's high security and low computational complexity allow resource-constrained IIoT devices to implement a security service, protect data privacy, and prevent attacks such as device

impersonation, data modification, and man-in-the-middle attacks that can interrupt the system operations.

CHAPTER VI
THREE-FACTOR AUTHENTICATION AND PRIVACY PRESERVATION SCHEME USING USER DEVICE BIOMETRICS FOR IOV SYSTEM

With the rise of the smart city concept and the growing demands for smart vehicles, the Internet of Vehicles (IoV) emerged as new technology. IoV is an emerging concept in intelligent transportation systems (ITS). IoV is considered an extension of the Vehicle-to-Vehicle (V2V) communication network. IoV is connected in an adhoc networking environment that employs each vehicle in the network as a node, called Vehicular Ad Hoc Network (VANET). The goal of IoV is to enhance the existing capabilities of VANETs by integrating them with the Internet of Things (IoT). The emergence of the IoV in the transportation system is related to several factors such as the large network scale, compatibility with personal devices, the reliability of the Internet, and the high-performance processing capabilities. The IoV will lead to the reduction of accidents, levels of pollution, and traffic congestion. IoV is also essential for autonomous vehicles because it will allow them to communicate with other vehicles surrounding them instantaneously.

Despite the promising aspects of the IoV, it is subject to several security threats because sensitive data is transmitted through an insecure channel in the IoV-based smart city environment. It can be a rich environment for passive and active attacks where the adversary can intercept, modify, or delete messages during the communication

process. The driver needs to determine whether the received message is authentic or not. In IoV, vehicles' authentication is crucial to achieve a trusted communication among the communicating vehicles. However, most of the currently employed authentication protocols in the IoV domain are based on asymmetric cryptography with a high computational cost.

This chapter introduces an efficient and privacy-preserving IoV mutual authentication and key agreement scheme. The presented protocol is called Three-Factor Authentication and Privacy Preservation Scheme Using User and Device Biometrics for IoV System. The proposed scheme enables the communicating devices to verify each other's identities and perform secure, anonymous authentication while the real identities stay secret. It will also allow the communicating parties to locally generate shared secret keys to ensure the confidentiality and integrity of the exchanged data. The proposed scheme utilizes PUF as a hardware security approach and the chained hash PUF concept. The proposed scheme employs lightweight operations of XOR and a one-way hash function; thus, the scheme does not significantly impact the device's computational and battery resources. We evaluate the proposed scheme by using different measures, namely, security and performance evaluation. The security evaluation will be both formal and informal security evaluation. The formal analysis used the Burrows–Abadi–Needham logic (BAN), the automated validation of internet security protocols and applications (AVISPA) toolkit.

Furthermore, the scheme efficiency is evaluated and compared with other related schemes. Through the informal analysis, we will assess the proposed scheme against well-known security attacks. Furthermore, we also validate that the proposed

authentication mechanism is efficient in storage requirements, computational cost, and communication overhead.

In Section 6.1, we first introduce the motivation of the work. Section 6.2 discusses the security aspects of the IoV. Section 6.3 presents the security threats and potential attacks in IoV communication. Section 6.4 presents the related work and the other existing protocols. Section 6.5 demonstrates the network model and the security goals. Section 6.6 presents the proposed protocols. Section 6.7 illustrates the protocol evaluation process, and finally, section 6.8 summarizes the chapter.

6.1 Motivation

Every year, approximately 1.3 million people die, and more than 7 million people are injured in around 8 million traffic accidents. People waste more than 90 billion hours because of traffic problems (accidents and traffic jams), causing a loss of 2% of the global gross domestic product, and vehicular travel generates 220 million metric tons of carbon equivalent [116]. The cost of personal transportation in cars is about $3 trillion per year in the United States, and 40% of this cost is related to crashes, parking, roads, traffic services, and pollution [117]. The authors in [118] stated that the number of all vehicles, whether commercial or passenger, used worldwide is more than one billion, and it is expected by 2035 to be around 2 billion [119].

The technological advances in the last few years led to the emergence of "Smart Cities". Improving road safety, traffic monitoring, and passengers' comfort are the main goals of designing new intelligent transportation systems (ITSs) in smart cities. One of the main goals of the Intelligent Transportation System (ITS) is tackling the above-

mentioned issues by providing smart mechanisms that are efficient, accessible, and secure.

Modern vehicles need to integrate smart sensors, powerful computational units, IP-based connectivity to the Internet and communicate with other vehicles and other devices in the surrounding environment. According to [120], the implementation of the IoV requires the integration of intelligent devices, capable processors with powerful computing and communication capabilities, internal sensors, external sensors such as cameras, location tracking, sensors to detect the physical, mental and emotional condition of the driver, and actuators to take actions to create an intelligent system that can support the needs of smart cities and the ITS.

IoV can be defined as a network of vehicles equipped with sensors, software, and technologies that enable exchanging information between the car and its surroundings through different communication media to connect and exchange data over the Internet according to agreed standards. IoV is composed of three fundamental components: 1) the inter-vehicular network; 2) the intravehicular network; and 3) vehicular mobile Internet [121]. IoV supports six types of network communication, as presented in figure 82:

Vehicle-to-Vehicle (V2V): An ad hoc communication enables each vehicle to contact its neighbor vehicles directly. This type of communication is used to exchange information about the speed and position of surrounding vehicles.

Vehicle-to-Roadside unit (V2R): An ad hoc communication facilitates exchanging information between vehicles and roadside units (RSU). RSUs are used as data storage servers. Unlike V2V, V2R allows long-distance communication. Besides, RSUs can act as an intermediate hop between the vehicle and the destination.

Vehicle-to-Infrastructure (V2I): An ad hoc communication where the vehicle can connect to the Internet to get different internet services.

Vehicle to Personal Devices **(V2P)**: refers to the interaction between vehicles and personal devices such as tablets and smartphones. Personal devices can be related to the driver, passengers, cyclists, or pedestrians. This kind of communication can be used to create awareness for vulnerable users, road users or connect the vehicle with other devices to share files such as music or video streaming.

Vehicle-to-Sensor (V2S): this can also be called intra-vehicle communication. The main goal of this type of communication is to monitor the vehicle's internal performance, such as speed, tire pressure, and oil pressure, through the On-Board Units (OBUs).

Vehicle-to-everything (V2X): this communication allows the vehicle to communicate with everything in the surrounding environment.

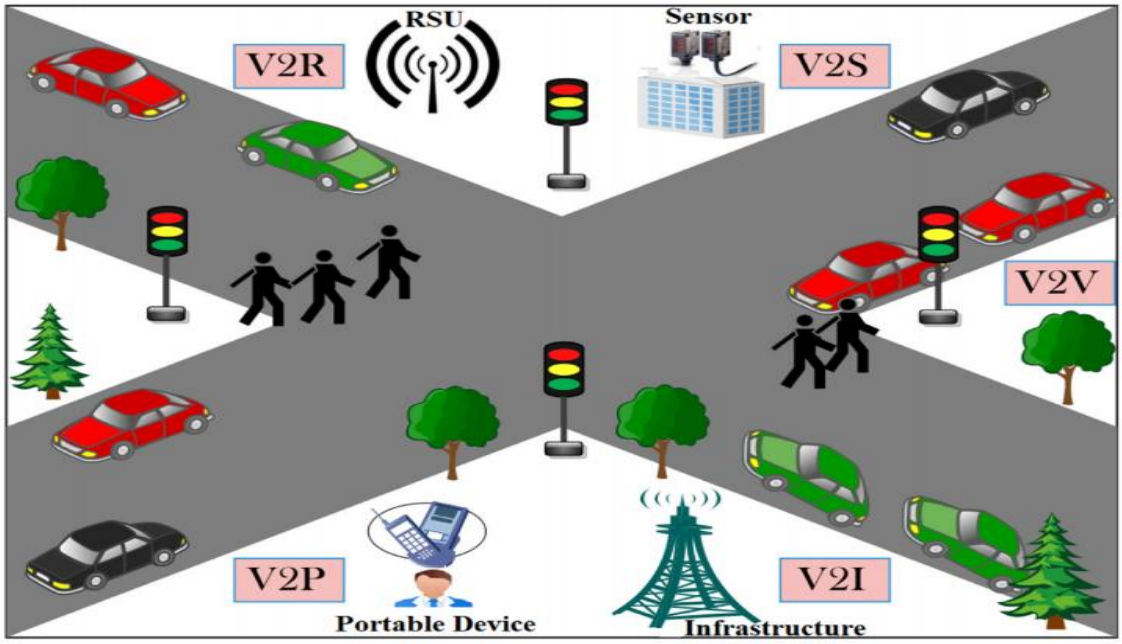


Figure 82: The IoV environment with five communication components

Based on the above-mentioned different types of communication that the IoV supports, IoV enables vehicles to connect to the Internet, constructing an interconnected cluster of vehicles that can exchange data about traffic, road safety, and others [122]. Furthermore, IoV facilitates communication with drivers, passengers, pedestrians, cyclists for entertainment and safety goals using different communication standards.

IoV will add individual, societal and environmental benefits as described below [120]:

Individual Level: The implementation of IoV reduces the vehicle running cost by lowering insurance rates, reducing operation costs, and minimizing the time spent in traffic, which in turn will increase productivity and reduce fatal accidents

Societal Level: IoV supports the smart city evolution by lowering road operational costs, reducing the number of accidents, and providing real-time traffic updates that can better control congestion through traffic management and road network optimization.

Environmental Level: reducing traffic and controlling congestion through traffic management will reduce the production of CO_2, which will positively impact the green environment.

Because IoV is one of the main components of the ITS, the security of the IoV is an essential factor. The IoV domain vehicles are equipped with the onboard unit (OBU) and application unit (AU) to communicate with the other vehicles. These units help exchange messages between vehicles and infrastructure, such as Internet-based roadside units (RSUs). The vehicles collect sensitive data about traffic jams, accidents, and weather conditions. The collected data is sent to the nearest access points for further action. This data can be transferred to other vehicles to avoid traffic jams and take an

alternative route in severe road conditions or accidents. Because most of the IoV ecosystem's communication is wireless, it is subject to different security risks. In IoV, most of the channels the vehicles use for communication are not secure, and therefore, the adversary can eavesdrop and tamper their messages.

On another perspective, transferring a driver's data to a trusted authority (TA) can lead to a security threat such as data eavesdropping, data modification, or data fabrication [123], which can negatively impact the driver's privacy, data confidentiality, and data integrity. Therefore, a secure authentication protocol is essential for IoV to establish trust in the system. Developing secure and efficient mutual authentication schemes and establish trust are major challenges of an IoV domain. There is limited work on developing lightweight mutual authentication protocols for the IoV domain. Most of the available work depends on heavy cryptographical methodologies to authenticate and secure the communication between the communicating parties. This chapter's main goals are to (1) present a lightweight mutual authentication protocol for IoV that preserves user privacy and (2) establish a mechanism to continually verify that the received data is coming from the authenticated driver. The proposed authentication scheme should detect and discover whenever someone else different from the authorized driver is driving the car during the lifetime of the authentication session to maintain the same level of trust among the authenticated entities and ensure the security of the IoV ecosystem.

6.2 Security aspects of IoV

According to [124], IoV integrates diverse devices, technologies, services, and standards, which will increase the need to secure exchanged data and communication. The authors in [125] argued that the heterogeneity of the IoV domain has many security

vulnerabilities. The operation of vehicles in an open and vulnerable environment makes the IoV vulnerable to cyberattacks and raises serious problems during V2V, V2I, V2R, and V2P communication. An adversary can exploit existing vulnerabilities, eavesdrop, manipulate, or delete vehicular data streams with destructive effects. Suppose the adversary succeeds in taking control of the vehicle. In that case, he/she can control the vehicle's brakes, heating system, or turning on or off the car, which in turn can present significant harm to road safety, including driver, passenger, pedestrian, and road infrastructure [126] and [127]. Therefore, it is a necessity to ensure security a high priority for the IoV.

IoV has several security requirements that must be addressed, and security solutions must be implemented to ensure user security and privacy. Table 25 presents the main security requirements, the potential attacks that threaten each requirement, and the mitigation techniques.

Table 25: IoV security requirements

The Security Requirement	**The definition**
Data Integrity	The sent and the received data are original and have not to be modified or altered [128]
Data Confidentiality	Data need to be protected to ensure the secrecy of the data between the parties participating in the IoV [129] and [130].

Authentication	To ensure that communication is occurring between the legitimate nodes. The identities of the vehicles must be verified [131] and [132]
Access Control	Vehicles must be able to only access data and data they are eligible to gain access to. Each participating node/vehicle has been assigned different roles and privileges to access the network [133].
Non-repudiation	It is preventing a vehicle from denying sending data to another vehicle. [134].
Availability	It is ensuring that the system is always available to service the vehicles and users. Also, to ensure the ongoing communication between the vehicles under different conditions [132].
Anti-jamming	A technique that can be employed to prevent malicious vehicles from sending fake/interfering messages to the surrounding vehicles to interrupt communication among vehicles [135].

Message Freshness	Enable the vehicle to verify that the received messages are fresh and are not replayed by an adversary.
Privacy	The vehicle's driving route driving is related to people's privacy, and therefore it must stay untraceable and get compromised by unauthorized access [136].

6.3 Security threats and attacks in IoV domain

IoV is exposed to different types of attacks and threats. To enhance the security of IoV, it is vital to understand the existing attacks in IoV. In this section, several security attacks in the IoV domain are discussed in the following section and presented in figure 83.

Sybill attack: The adversary creates some fake vehicles that use fake ids around the victim vehicles and sends fake messages such as traffic jamming messages, wrong directions, or false positions to misguide and deceive the surrounding innocent vehicles. consequently, the whole network is disturbed, and it is risky for users' lives [123], [137], and [138]

Denial of service (DoS) attack: This attack threatens the availability security requirement. It aims to prevent legitimate and innocent users from using network services and resources [122] by making them unavailable. It overloads the communication channel so that no communication occurs among the authentic vehicles,

which is a severe problem. Communication is vital in safety applications because timely information is required to avoid accidents [139]. The DoS attack can take different forms, such as (1) Communication channel jamming, which will hinder users from the network by jamming the communication channel. In this type of DoS attack, the adversary will intentionally use a high-power transmitter to continuously transmit a signal at a higher power level on the same channel frequency as the target. The target can be: (1) communication between vehicles or between vehicles and RSUs; (2) Network overloading where the adversary sends fake traffic to other vehicles or RSUs in the network to keep the other node busy and prevent them from performing their tasks and eventually the performance of the network is reduced; or (3) Packet dropping in which the adversary makes the information unavailable to other nodes by dropping the packets which carry essential information, so communication is affected.

Node impersonation attack This attack is a violation of authentication in a network. In IoV domain, each vehicle has a unique identity to identify itself, and this unique identity is useful in case of any unsafe situations. In a node impersonation attack, the adversary may use the id of an authenticated node to send malicious messages to other nodes on the network. In such a case, the innocent vehicles assume that a message is received from a legitimate node; thus, confidential information is violated [139].

Man in the middle (MITM) attack: This attack violates data integrity and privacy goal. In this attack, the adversary position himself/herself between the legitimate communicating parties. As a result, the adversary may eavesdrop on their communication, modify the sniffed message, or inject fake messages. At the same time, two communicating parties assume that they are directly communicating with each other.

Consequently, the message's integrity and authenticity are negatively impacted, and the network security is compromised [123] and [133].

Replay attack: this type of attack repeatedly broadcasting the already sent message by the adversary to deceive the other vehicles in the network by dropping the priority messages from the queue. The system performance would be affected by frequent replaying, and bandwidth cost also increases [136].

Brute force attack: session key/symmetric keys play a vital role in cryptographic algorithms in securing the information. In this attack, the adversary tries all possible combinations and already existed dictionaries to steal sensitive information such as passwords. The attacker makes multiple attempts as the brute force approach takes time in decoding the encrypted information [122].

Eavesdropping attack: In this attack, the attacker acts as a passive illegitimate listener to the exchange message between the communicating parties to collect private, confidential data of the drivers or the passenger to use it against their privacy without even letting them know.

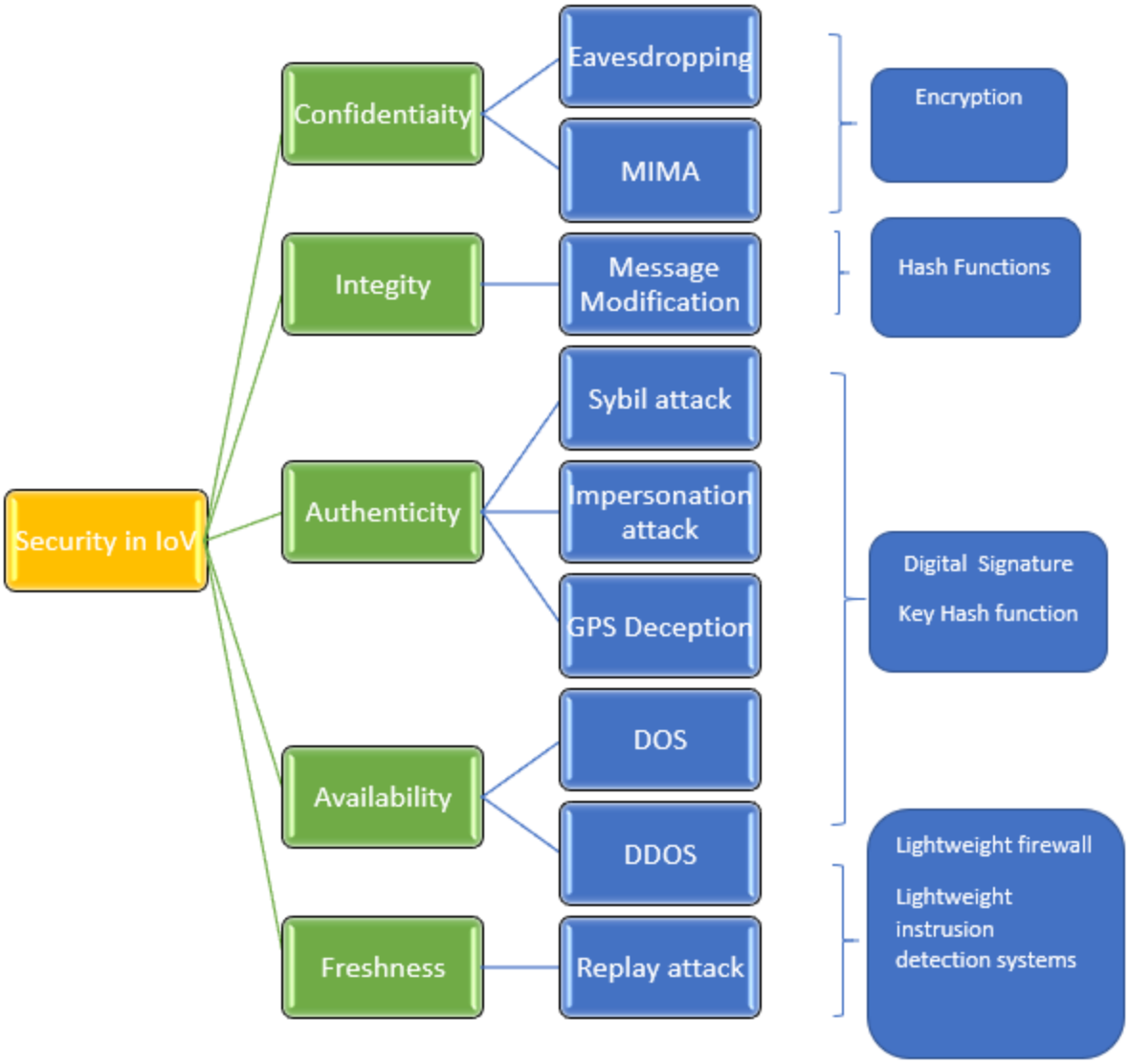


Figure 83: Security threats and attacks in IoV domain

6.4 Related Work

Authenticating vehicles in the IoV domain is a security requirement that has been studied by several researchers in conjunction with achieving other security requirements. The authors in [140] present the possible vulnerabilities and threats of connected vehicles from three different aspects. The first aspect is the vehicle itself that consists of three main elements: electronic control units (ECUs), in-vehicle networks, and the gateway of

standardized public communications such as 3G/4G, Wi-Fi, and Bluetooth. The second aspect is the connection between the mobile device to the vehicle. The third aspect is the communication technologies used by the vehicle to communicate with the surrounding environment. Authors in [141] classified possible attacks of connected vehicles into physical attacks such as physical damage of the vehicle, close-proximity attacks to obfuscate sensor information and inject faulty data in ECUs within a range of approximately 10 meters, and remote attacks, including accessing ECUs and sensors via insecure communication paths over remote wireless access. The authors in [142] explained that connected vehicles are more vulnerable to cyber threats than traditional vehicles due to the communication between the connected vehicle and the surrounding external environment and the increased internal communications among the system's internal components.

The authors in [143] present a lightweight mutual authentication protocol for IoV using cryptographic operations. The presented protocol enables a vehicle and a server to establish a secret key to secure the ongoing communication while minimizing the computational cost associated with the process through a trusted authority server. The protocol consists of four main parties: the vehicle, the vehicle server, the trusted authority, and the registration authority. The protocol proved to be secure against impersonation attacks because the malicious vehicles cannot generate request messages as it involves passing through the second mandatory phase. Furthermore, the protocol is resilient against data modification, replay attacks, and password guessing attacks. On the other hand, the protocol does not clarify how the trusted authority can identify the

vehicles without using an ID. Also, the presented protocol uses a long-term secret key during the authentication process, which can be retrieved using side-channel attacks.

Authors in [144] proposed an authentication scheme that focused on authentication, privacy preservation, integrity, and non-repudiation. The researcher used Timed Efficient Stream Loss-tolerant Authentication (TESLA). The authors in [145] presented TESLA as a broadcast authentication scheme to provide V2V communication using ECDSA to sign the first packet that is sent by each vehicle. Because ECDSA is computationally expensive, the researchers in [144] used the Bloom filtering (B.) technique as an alternative to ECDSA. In the presented protocol, the RSUs are responsible for generating a group of pseudonyms for each vehicle. The vehicle rapidly changes its pseudonyms in fixed time slots ensuring privacy. The vehicles are grouped according to their similar characteristics like speed and location. The authentication of the vehicle is verified against the BF value. Any new vehicle that initiates a communication would request a BF value from the corresponding RSU and key from the vehicle that it needs to communicate with. The proposed scheme can protect the system against malicious nodes, preserve the user's privacy, and helps in reducing delay and latency.

Author [146] presents an authentication scheme resistant to secret key compromisation by presenting a key insulation concept. It has a Private Key Generation (PKG) algorithm that generates a set of keys for both the vehicles and RSUs. They also suggested adding to the vehicle an additional Tampered Proof Devices device (TPD) that acts as a helper. They assume that the TPD is physically secure but computationally limited, and its stored information can never be disclosed. The presented scheme is based

on elliptic curve cryptography (ECC). The scheme has four phases. The first phase is the initialization phase, where the PKG generates parameters of ECC. The PKG generates the private keys and calculates the corresponding public keys of TPD and RSUs. The second phase is the key generation phase, where the vehicles create their secret keys using the public keys of OBU and TPD and some randomized parameters. The third phase is the signing phase, where the OBU sends the signature to the RSU for verification. The fourth step is verification, where the RSU verifies the received signature from the vehicle for authentication. The proposed scheme is secure against various chosen plaintext attacks and forgery attacks as its private key is of two parts: one part is with TPD, and the other is with the vehicle itself. On the other side, it is computationally expensive due to the generation of multiple private and public keys and the use of digital signatures.

The authors in [147] present a two-level authentication key exchange scheme employing the Elliptic Curve Cryptography (ECC). During the first authentication level, the cluster head (CH) vehicle is verified and authenticated by the CA. Then the authenticated CH will be responsible for authenticating the vehicles within the cluster in the second level of authentication. The protocol consists of four different phases. The first phase is the pre-deployment of vehicles phase, where the CA assigns the vehicle the needed authentication parameters that will be verified during the authentication phase. The second phase is the cluster head verification, where the CA verifies and authenticates the CH vehicle. The third phase is the authentication and key agreement of the vehicles. During this phase, the cluster head verifies and authenticates the vehicles within the cluster. After completing the authentication process, the CH establishes a session key for their secure communications in the future. The last phase is Dynamic Vehicles Addition,

where new vehicles can be dynamically added to the cluster. The presented protocol is not clear how the CA and the CH identify the vehicles without receiving any vehicle ID. Also, the proposed protocol is very computationally expensive due to the extensive use of public cryptography and digital signature. Finally, the protocol depends on the CH to verify and authenticate all the vehicles that are within the cluster. This design will generate a significant overhead on the CH, and at the same time, it is vulnerable to a single-point failure. Moreover, the cluster head will always be changing depending on the vehicle speed and position; therefore, the CH will often be altered, reducing the authentication session lifetime as the new CH will be elected and a new authentication process will be conducted.

Authors in [148] introduced secure and efficient message authentication protocols for IoV communication such as V2V, V2S, V2R, V2I, and V2P. The proposed protocol consists of four phases: initial setup, registration, authentication, and communication. The presented protocol cannot resist various security threats such as secret key disclosure, MITM, and impersonation attacks and does not ensure authentication. The adversary can conduct an impersonation attack because he/she can masquerade as a legitimate vehicle. Also, the adversary can conduct a secret key disclosure attack by being able to extract the secret credentials {Za, Ua, Wa} stored in a smart card. Then, the adversary will be able to compute and retrieve the vehicle server's secret key KVS = Za $\oplus$ h(Ua||Wa), and random nonce pa = Aa $\oplus$ h (KVS ||T1). Consequently, the adversary can execute a secret key disclosure attack by calculating Ce = De $\oplus$ KVS $\oplus$ pa and masquerade as a legitimate vehicle and also can conduct a man-in-the-middle attack. Finally, because the adversary can obtain the VS's secret key KVS and symmetric key

between each entity. Then, the adversary can generate authentication request messages, verify response messages, and achieve message authentication with other entities successfully. Consequently, the proposed scheme does not ensure secure message authentication.

The researchers in [149] present an efficient message authentication protocol for IoV in a smart city environment, called IoV-SMAP. The proposed protocol consists of three phases that are initialization, registration, and authentication. The vehicle server registers all IoV entities in the communication system during the initialization phase and generates a secret master key (KV). The proposed protocol does not present a technique on how the IoV entities can verify the identities of each other. Therefore, the protocol can be subject to an impersonation attack and also a man in a middle attack.

Furthermore, by impersonating and acting as a legitimate vehicle, the adversary will be able to generate the session key between each entity. Then, the adversary can generate authentication request messages, verify response messages, and successfully achieve message authentication with other entities. Consequently, the proposed scheme does not ensure secure message authentication. The proposed IoV-SMAP can resist security drawbacks and provide user anonymity and mutual authentication.

The authors in [150] present a two-factor authentication and key agreement protocol for IoV by combining a password with the PUF. The proposed protocol preserves user's anonymity. The protocol consists of three parties: (1) the User (Ui) that receives data from the vehicle sensors and statistical data from the data center;(2) the Data Center (DC) that manages and stores data collected by vehicle sensors; (3) and the Vehicle Secret Key (Vsk) that is located in the vehicle to collect data from the

surrounding environment. The protocol consists of three phases that are system setup, registration, login, and authentication. During the setup phase, the DC generates its own public-private key pair (pk, sk) and publishes its pk to users and vehicles for encrypting the subsequent messages. During the registration phase, both the user and the vehicle sensor are registered to the DC. During the authentication phase, the DC authenticates both the user and the vehicle and generates a session key with each of them. On the proposed protocol, the user and the vehicle sensor do not authenticate the CD first before sending the PUF responses, which can lead to DC impersonation. Because the vehicle sensor generates its PUF challenge, there is no way to verify is the vehicle sensor is original or counterfeited.

The authors in [151] propose a platform for secure data sharing and storage in VANETs using blockchain. The proposed authentication scheme requires the vehicle to prove its ID before sharing any data. One of the main weaknesses of the proposed protocol is using blockchain that added overhead to the proposed protocol and did not support its scalability. Similarly, the researchers in [152] and [153] introduced other authentication and key management protocols that employ blockchain, which inherit the same weaknesses as [151].

The authors in [154] present an efficient protocol for mutual authentication in the IoV. The proposed protocol uses physical unclonable functions as part of the authentication process. The researchers claim that the proposed protocol Protects against physical and cloning attacks and preserves user's privacy. The proposed protocol consists of three phases that are: Device registration, authentication, and CRP update. Each vehicle needs to register itself with the trusted authority (TA) during the device

registration phase before it can become a part of the IoV. During the authentication phase, the vehicles are authenticated to the TA and the RSUs and generate a session key. Once the authentication process is completed, the authenticated vehicle can communicate with the RSUs using the session key. The third phase is the CRP update, where the TA can update the CRPs of the vehicles. The proposed protocol preserves the user's privacy. However, the proposed protocol does not support ensuring the device's originality before completing the registration process.

The authors in [155] present a cloud-centric three-factor authentication and key agreement protocol that integrates passwords, biometrics, and smart cards to ensure secure access to both cloud and Autonomous Vehicles (AVs). At the end of the authentication process, two session keys are negotiated. The first key is between the user and AV to support secure remote control of the AV, and the second key is negotiated between the mobile device and the cloud. The proposed protocol has three entities, including the user, the vehicle, and the cloud. The protocol consists of 6 phases: system setup, AV registration, user registration, user authentication, password, biometrics change, and smart card revocation. During the setup phase, the cloud generates and advertises the Elliptic Curve Cryptosystem (ECC) public parameters and generates an ID. In the AV registration phase, the cloud generates and distributes a secret key for each AV. In the user registration phase, the cloud issues a smart card storing the secret key to each user. During the user authentication phase, the authentication mechanism is employed to verify the user identity and build secure channels among the cloud, the AV, and the user. Although the protocol achieves its security goals, it is computationally expensive due to the extensive use of the ECC scaler multiplier and the Fuzzy extraction.

Therefore, this protocol does not support the required system efficiency and does not support system scalability. Furthermore, the protocol does not support physical security.

The authors in [156] present a physical unclonable function (PUF) based on three-factor authentication and key agreement protocol to ensure that the system is secure even if the user devices or sensors are compromised. The researchers used password, biometrics, and PUF as the three authentication parameters. The proposed protocol combined the user characteristic through the use of biometric and the device characteristic through the use of the PUF. The protocol consists of five stages: system setup, registration, login and authentication, password update, and biometric update. Although the proposed protocol is secure, it has several limitations: (1) it depends heavily on scaler multiplier for ECC and fuzzy extractor, which are computationally expensive; (2) although the proposed protocol used PUF as a hardware fingerprint, the proposed protocol can't verify the originality of the devices and sensors before completing the registration process which in turn does not protect the IoV ecosystem from the devices counterfeiting; and (3) the lack of efficient computation negatively impact the scalability of the proposed protocol.

The authors in [157] propose a lightweight and secure authentication and attestation scheme for attesting vehicles on the roads. The presented protocol proposed a security scheme using PUFs to perform a combined authentication and attestation protocol for the IoV network. The Roadside Units initially authenticate the Vehicle Onboard Units (OBUs) (RSUs), and then the attestation process is to be completed at the edge servers. The authors state that they used PUFs for generating on-the-fly secret information that can be used as part of the authentication process. The proposed protocol

consists of five main entities: the vehicle Vi, the RSU Rj, the edge server ES, the IoV cloud server IoV C, and the trusted authority TA. The RSUs are responsible for authenticating the vehicles. The proposed protocol consists of the initialization phase, Registration phase, Vehicle-RSU authentication phase, and Attestation phase. Although the proposed protocol is lightweight, it has several limitations. First, it uses the same pseudoID for both the vehicle and RSU in every authentication session. This, in turn, can lead to vehicle traceability and linkability. Second, the same challenge and response are used in every authentication session. If a hacker succeeded in retrieving the challenge and the response, they could impersonate the vehicle. Third, the authentication session can't be transferred from one RSU to another within the same cluster, limiting the protocol scalability.

Many of the above-presented protocols share the same common limitations. First, many suggested protocols used computationally expensive cryptographic operations that do not support system scalability and, at the same time, do not support devices with limited memory and computational capabilities. Second, most of the proposed protocols did not focus on the ensure the authenticity of the communicating devices and secure the system from integrating counterfeited devices. However, this is considered a significant concern in the IoV domain. Third, many proposed protocols do not implement multiple authentication levels, which is the core of the defense-in-depth concept. Fourth, in some of the presented protocols, the researchers overlooked the devices' privacy, which, in response, does not guarantee the anonymity and privacy of the IoT device and, at the same time, allows devices traceability. Fifth, many of the proposed protocols require storing long-term secret keys that can be retrieved using side-channel attacks. Sixth, some of the proposed protocols, such as

[157], use the same CRP for every authentication session and store the CRP on the vehicle's OBU. If the device is compromised, the adversary may obtain the CRP and impersonated the vehicle.

The available literature shows the necessity of using multi-factor authentication to implement the defense-in-depth concept. Also, it highlights the need for lightweight authentication protocols that fit the limited resources devices and support the system scalability. The proposed scheme in this chapter combines both the driver's biometric parameters and the PUF as unique identifiers to authenticate the vehicle and the driver. Using only one of the two mechanisms will lead to limitations in the presented scheme. Using the driver's biometric only will authenticate the driver, but the vehicle's authenticity will be questionable.

On the other hand, using only the PUF to authenticate the vehicle will authenticate the vehicle but will not enable us to verify the authenticity of the driver. Therefore, we propose combining both the vehicle and the driver biometric to implement a secure and robust authentication protocol that facilitates building a root of trust among all the communication parties. The proposed protocol will achieve three levels of authentication, namely, V2V, V2I, U2I.

6.5 Three-Factor Authentication and Privacy Preservation Scheme Using User and Device Biometrics for IoV System Network Model and Security Goals

We present an IoV network model and introduce the security mechanisms and components of the proposed scheme. Then, we will discuss the main security goals.

6.5.1 Network Model

Figure 84 presents the network model of the proposed scheme. We propose a four-layer architecture consisting of 6 main entities: the driver (D), the Vehicle (V), RSU, central RSU (CRSU), the vehicle owners cloud server (VOCS), and trusted authority (TA). The vehicles and drivers are at the lowest layer. The vehicles and drivers are connected wirelessly to the next layer, which consists of RSUs. The RSUs are connected to a central RSU that is located in the third layer. This connection may be wired or wireless. Finally, the CRSU is connected to both the service provider cloud server and the trusted authority (TA.) through a wired connection. There are several types of communication modes, such as Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I), user to infrastructure (U2I).

The responsibility of the TA is to perform the registration of V, RSUs, CRSUs, and the D prior to placement in the network. Each V has an OBU, which processes and stores all vehicles' information [158]. Vehicles are also equipped with sensors that sense and process surrounding information and send that information to the OBU of the vehicle [159]. Each vehicle is connected with the Internet and can also send and receive data using the Internet [160]. D-VOCS, RSU-CRSU, V-VOCS, and CRSU-TA communications occur through the Internet. V2V and V-RSU communication also occur through the Internet, but it is preferable to have dedicated short-range communications (DSRC) [161].

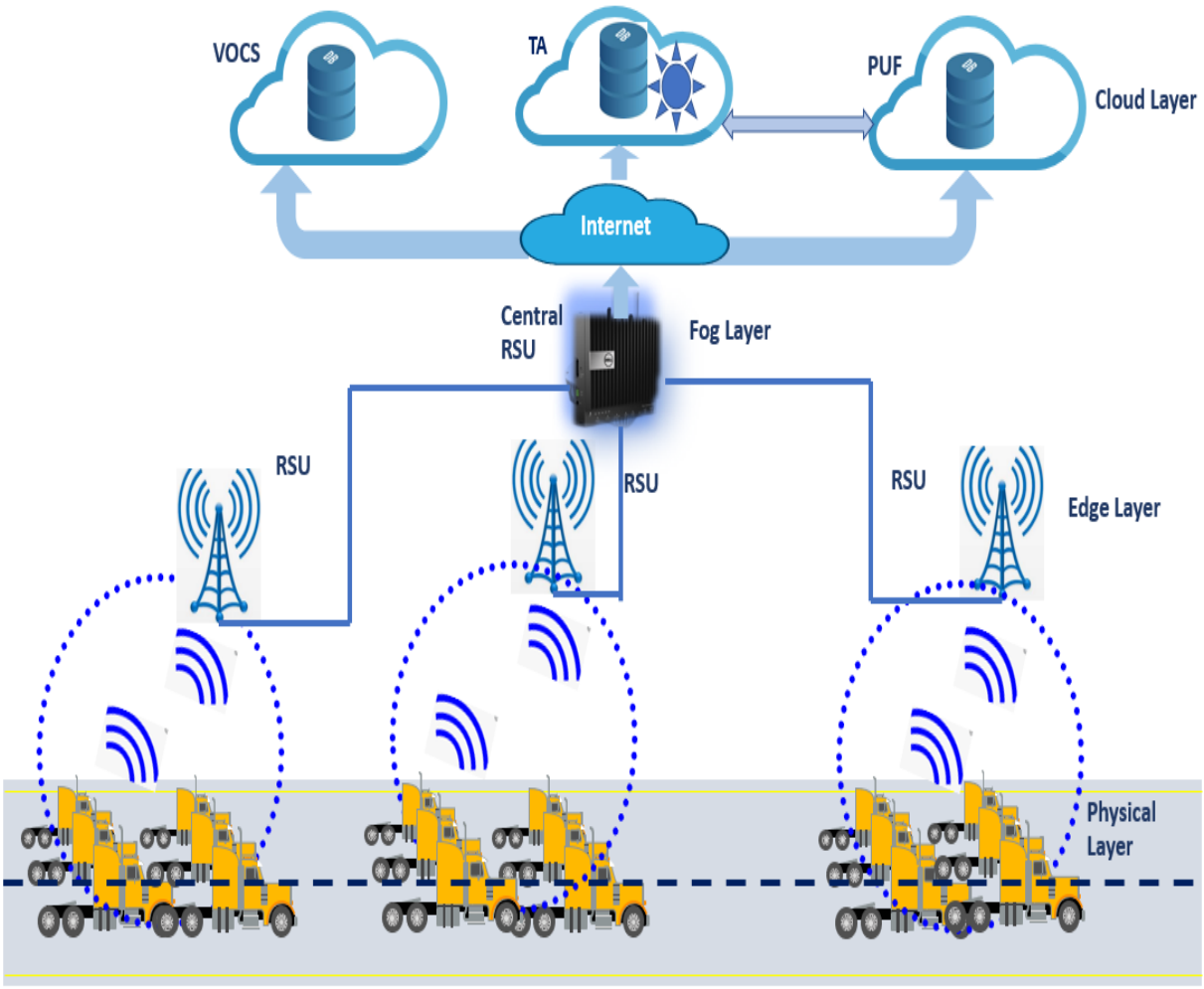


Figure 84: Proposed IoV network model

- **The Vehicle** is equipped with sensors, ECUs and in-vehicle networks, and onboard units (OBUs). The OBU is a device that is implanted in a vehicle, and it is used to send and receive data to other OBUs or RSUs. Furthermore, a PUF chip is embedded in the OBU and used as the vehicle fingerprint. By using OBUs, vehicles can communicate with each other as well as with the RSUs. The communication among them is based on the DSRC protocol [155]. Therefore, any attempt to tamper or separate the PUF from the OBU renders the PUF useless [162] and [163]

- **The Driver**: The system continuously monitors the driver's behavior to make sure that he/she is the authorized driver on the vehicle and he/she is in good health condition.
- **Roadside Unit (RSU)**: It is a stationary device located along the roads and at intersections. RSU gathers information about the road traffic and broadcasts it to the OBUs within the communication range. Also, an RSU can communicate with other RSUs and the CRSU to exchange messages related to road traffic through a secure channel.
- **Central RSU (CRSU)**: The CRSU is a device that is in charge of a group of RSUs, and it is considered the intermediate communication channel between the RSU and the T.A. The CRSU collects and aggregates the data from all its RSUs and sends it to the T.A. for further analysis and storage.
- **Vehicle Owner Cloud Server**: The VOCS is responsible for the vehicle's data storage and data analysis and the driver's collected data. The VOCS is in charge of registering the vehicle and the driver on the company level. The analyzed data results will help the organization monitor the driver's health and behaviors and allow the company to monitor the activities of the vehicle as an asset.
- **Trusted Authority (TA)**: The TA is responsible for data storage and data analysis. The TA collects real-time traffic information, road conditions, and environmental data from the CRSUs and analyzes them for situational awareness. All vehicles, RSUs, and CRSUs, need to register with the TA. The TA is assumed to be completely trustable, hard to compromise, and powerful. In addition, it has sufficient computation and storage capacity.

The proposed scheme consists of four main layers: the physical layer that consists of the vehicles and the driver, the edge layer consists of the RSUs, the fog layer that consists of the central RSUs, and the cloud layer that consists of the trusted authority (TA), the Vehicle Owner Cloud Server (VOCS), and the PUF cloud. The Vehicle on Layer I is equipped with different types of internal and external sensors collecting data in real-time. This work focuses on sensors that collect real-time data from the vehicle surrounding environment, such as radar, lidar, 360 camera, proximity sensors, and lane tracking sensors. The collected data is sent to the designated ECUs units for further processing and reporting. Data that has to be reported will be sent over by the appropriate ECU to the OBU unit to be transmitted to the RSU. The OBU has an implanted PUF chip that will be used during the authentication process. The driver's fingerprint will be collected via the in-vehicle fingerprint scanner. Additionally, the driver's biometric features such as EKG or ECG will be continuously collected by using a steering wheel-based sensor for real-time driver's verification.

The edge layer is a series of individual RSUs that are under the authority of a single Central RSU. RSUs collect real-time data from the vehicle and acts as a relay between the vehicle and the Central RSU. In addition, the RSU facilitates V2V communication. Furthermore, when a vehicle moves to the proximity of a new RSU, to verify the vehicle's authenticity, the most recently visited RSU will help the new RSU.

The fog layer houses the CRSUs, where the CRSU interconnect multiple individual RSUs. The CRSU plays a major role in facilitating vehicle communication that involves the following activities: (1) vehicle authentication; (2) handover between neighbor RSUs; (3) maintain authentication session and parameters during the vehicle

trips that may cross multiple RSUs. Furthermore, the CRSU will be in charge of data cleaning, aggregation, analysis, and decision-making. The main goal of this distributed architecture of the RSUs is to create a scalable architecture for vehicle systems that exhibit flexible admission control and avoiding single-point failure. Additionally, it enables the vehicle to be authenticated just once using its PUF when it is within the CRSU range. This, in turn, reduces the expected latency and the computational complexity of each new authentication session. The RSUs on the edge layer and CRSUs on the fog layer participate in local decision-making such as reroute vehicles to reduce traffic jams or avoid road constructions or predict the probability of accidents in a specific area. Based on local decision-making, the alarm system gives a warning message to the driver of that particular vehicle.

The cloud layer is generally virtualized in data centers and communicates with the other layers using the Internet. The architecture presented in figure 84 has three clouds. The first one is the cloud of the Vehicle Owner Cloud Server (VOCS.), where the collected data from the vehicles and the drivers will be stored and analyzed to be utilized as an organizational monitoring system. This stored data will be used to conduct further analysis and mine knowledge to help the organization make better decisions regarding their business. The VOSC is responsible for the registration of the vehicles. The VOCS is in charge of generating the communicating credentials for the vehicles, such as alias identities, OTP, and secret keys to be used during the authentication process. After storing the generated credentials in the vehicle's memory, the vehicles can be deployed in the field. The VOCS is also responsible for registering the driver and stores the template

of his/her biometrics to be used during the authentication process. The driver can also access the data of the deployed vehicles from the VOCS.

The second cloud is the cloud of the trusted authority (TA), where the collected data that has been sent through the fog layer will be stored on Distributed File System (DFS). This stored data will be used to conduct further analysis and to mine knowledge. The TA is responsible for registering various network communicating entities, including the vehicle, the RSU, and the CRSU. RSUs, CRSUs. TA collaborates in large-scale decision-making, such as traffic conditions for a whole city. The TA is in charge of generating the communicating credentials for the vehicles, such as alias identities, OTP, and other authentication parameters. The third cloud is the PUF cloud, where the manufacturers store the CRPs of the vehicles, RSUs, CRSUs, TA, and VOCS. The CRPs will be used to authenticate the vehicle as well as ensure the authenticity of the devices to protect the IoV ecosystem from integrating counterfeited devices.

The manufacturer assigns each vehicle a unique secret ID. The VOCS assigns the vehicle and the driver unique alias IDs. Also, the TA assigns the vehicle an alias ID. The Alias IDs will be changed every authentication session. The main goal of the alias ID is to protect the vehicle and the driver's anonymity and ensure their untraceability.

The communication between any two entities is subject to both active and passive attacks. In passive attacks, the adversary can eavesdrop on the communication. In active attacks, the intruder may replay, modify, or delete specific communication messages. The message header that contains the device Alias ID is sent in clear text while the payload is sent in an encrypted format. The number of vehicles can vary throughout the lifetime of the network. Because adding a new vehicle to the network is highly possible, the

proposed protocol accommodates system scalability. The proposed scheme supports the dynamic addition of vehicles to the IoV network without changing the network's security states.

6.5.2 Driver biometric and vehicle biometric

Ensuring the security of the IoV ecosystems requires ongoing authentication of the communicating entities and verification of the collected data from both the vehicle and the driver to establish trust. An authentication mechanism is needed to ensure the validity of the data before being used to analyze the collected data about the driver and the vehicle surrounding environment. Furthermore, there is no proof that the collected data is still related to the same driver during the authentication session once the driver got authenticated. To ensure safety and security, the fleet vehicle in the IoV ecosystem requires continuous monitoring of the driver. Therefore, the received data from the driver need to be verified and validated continuously to ensure that it is coming from the same driver during the whole authentication session [145] and [146]. The IoV system needs to ensure that the driver's received data belongs to the authenticated driver and also is obtained from the authenticated vehicle. This can be achieved by using user biometric and vehicle biometric that the PUF represents.

6.5.2.1 Biometric authentication techniques

6.5.2.1.1 Driver's biometric authentication

During an authentication session, the verification process can be performed based on one or more of the following three different factors: (1) something we know such as a password or unique identification value; (2) something we have such as token or secret keys; and (3) something we are such as biometric features like fingerprint, voice, face, or

electrocardiography (ECG). Biometric authentication is considered a decisive authentication factor compared to other authentication credentials because it is a unique identifier for each human and cannot be transferred or replicated [164]. Authentication mechanisms can be categorized as static and continuous authentication methods [165]. Static authentication can be treated as an initial authentication where the user can be authenticated to the system. Still, there is no post-authentication monitoring technique to ensure that the same user is initially authenticated [166].

On the other hand, continuous authentication mechanisms monitor a system during the lifetime of the authentication session to verify and ensure that it is the same user who accesses the system [167]. The proposed scheme employs both static and continuous authentication of the driver to ensure the same driver during the whole authentication session. In addition, the constant collection of the driver's biometric will be an effective method to monitor the driver's health condition. The proposed scheme will authenticate the driver using his/her biometric in the fingerprint format as the static biometric feature and the driver's biometric in the ECG or EKG format as the continuous biometric feature.

6.5.2.1.2 Vehicle biometric mechanism based on PUF and chained hash PUF authentication

The PUF technique uses the hardware properties of the vehicle for unique identification. The proposed scheme achieves mutual authentication by implementing a challenge-response authentication mechanism. In the proposed scheme, during the registration phase, both the vehicle and the TA hash the response of a PUF challenge and the TA master secret key to produce a hashed value named CHX. Then, for all

subsequent authentication sessions, the vehicle hashes the previously-stored chained CHX_{x-1} along with the new Rx, to generate a new chained hash CHXx. The TA retrieves the Rx value from its database and hash the retrieved R with the previously stored chained CHX_{x-1}. Then, it compares the resulting value (CHXx' = h (R'|| CHXx-1)) with the received CHXx'. If it matches, it will authenticate the vehicle node. Once the authentication is completed, the two communicating parties will store the CHXx in their databases and use the $CHXx_{-1}$ along with the exchanged random values to generate the shared session key(*ssk*).

During the driver authentication, the hash value of the combination of the vehicle Rx and the user static biometric will be calculated before being sent to the VOCS to authenticate the driver. This process aims to secure the driver's privacy and ensure the security of the biometric feature.

The proposed scheme chains the blocks of hashed response values together by hashing the new PUF response value with previously generated chained hash PUF values and the vehicle's real identity. Thus, my mechanism looks like a blockchain technology at first glance, but it does not utilize blockchains. Because the PUF response value cannot be predicted or replicated, the adversary can't predict the chained hash PUF value. The chained hash PUF value is used to present a historical factor for authenticating the vehicle and the driver. Employing the chained hash PUF mechanism and user biometric technique ensures mutual authentication through a challenge-response scheme which is essential in IoV ecosystem security. The two-way challenge/response authentication technique allows the communicating parties to check the device's authenticity and, at the same time, enables the communicating parties to ensure that they are not communicating

with a malicious node. The use of the PUF proves the device's identity and ensures its originality. The use of the cumulative chained hash PUF value proves the ability of the communicating parties to show proof of knowledge of past chained hash PUF value.

6.5.3 Security goals

Because security attacks on the vehicles in the IoV ecosystem may cause physical damage and loss of human lives, security requirements became vital in the IoV domain. Single-factor authentication is not sufficient to validate the security of the vehicles and users in the IoV domain. A 3-factor authentication protocol is a promising alternative to secure the IoV ecosystem. This chapter presents a three-factor authentication schemewith the following goals:

1. **Mutual authentication**: The goal of mutual authentication is to ensure that the communication is occurring only among the legitimate parties as well as ensuring that only the legitimate parties have access to the vehicles, the RSU, VOCS, and the TA. Therefore, the legitimate parties should be able to prove they are who/what they claim.
2. **Secret session key (*ssk*) perfect secrecy**: the *ssk* of one session should not be derived even if the adversary reveals the *ssk*s of previous sessions.
3. **User/vehicle anonymity**: the proposed scheme should be able to preserve user/vehicle anonymity. The adversary must not identify the user's or vehicle's real identity by capturing any communication message.
4. **User /vehicle untraceability**: the scheme should preserve user's and vehicle's privacy and ensure that an adversary can't trace the user's actions and status as well as vehicle location.

5. **Biometric privacy protection**: The leakage of a user's biometric template can cause serious privacy risks once being leaked. To preserve biometric privacy, no biometric template will be saved in cleartext on the VOCS.
6. **System scalability**: the scheme should be scalable and be able to accommodate the growing number of vehicles. This goal can be achieved by reducing the communication overhead and reducing the number of times a vehicle needs to authenticate to the RSUs.
7. **Low latency**: Because the IoV is dynamic and the vehicle movement is so frequent and fast, the proposed scheme needs to be efficient and not computationally expensive. One way to achieve this goal is by reducing the required data to complete the authentication process and employing lightweight cryptosystems.
8. **Counterfeiting and physical protection**: This goal can be achieved by using the PUF.

6.5.4 Network model assumptions

We make the following assumptions.

- Every vehicle is equipped with a PUF.
- The communication between a vehicle's OBU and PUF is considered secure.
- Every vehicle is equipped with a fingerprint scanner.
- Every vehicle is equipped with a smart steering wheel that can collect data about the driver’s health.

- Every vehicle is equipped with a keypad for the driver to insert his username and password.
- The communication between the vehicle's OBU and the fingerprint scanner is considered secure.
- The communication between the vehicle's OBU and the smart steering wheel is considered secure.
- The communication between CRSU and TA is considered to be secure.
- The communication between the RSUs and the CRSU is considered secure.

6.5.5 Threat model

Assume vehicle x sends an authentication request to the VOCS, an adversary S is able to replay, eavesdrop, tamper, and inject packets sent by a vehicle or by the driver. Moreover, S may gain physical access to vehicles and exploit physical attacks to retrieve stored secret data. The following assumptions about security properties and adversary abilities are made.

1. The used communication channel during the registration process is secure.

2. The one-way hash function is collision resistant.

3. The vehicle's OBU has protection against tampering.

4. Replay attack: An adversary can capture messages from old authentication sessions and replay them in the current session.

5. Eavesdropping attack: the adversary can eavesdrop on the communication channel between the vehicle and the infrastructure.

6. Message modification attack: the adversary can tamper with intercepted messages.

7. Injection attack: the adversary can send counterfeit messages.

8. Impersonation attack: The adversary can pretend to be a legitimate vehicle or a driver and sends a message to the infrastructure. Also, the adversary can impersonate the infrastructure and sends a message to the vehicle.

9. Physical attack: the adversary can conduct a physical attack on vehicles to extract stored secrets in memory. Any physical attack will tamper with the PUF and destroys it.

6.6 Proposed authentication scheme: Multi-Factor Authentication and Privacy Preservation Scheme Using User and Device Biometrics for IoV System

Fleet vehicles are a special type of vehicle that may share their data with more than one destination. On the road, the fleet vehicle is treated as any other regular vehicle that can collect data from the surrounding environment to support the intelligent transportation system. From another perspective, the fleet vehicles need to continuously communicate their data and the drivers' data with the vehicle owning company as a mechanism of fleet monitoring. This work focuses on authenticating the fleet vehicles in the IoV ecosystem to the road infrastructure and authenticating both the vehicle and the driver to the cloud server of the vehicle-owning company. The continuous growth of the number of fleet vehicles on the road highlights the need to optimize the communication tasks and the authentication process to reduce the consumed energy and support the system scalability. According to [168], a fleet monitoring system adds several benefits to businesses. First, it will allow businesses to collect real-time data about the driver's behaviors for driver safety and build trust between the drivers and their managers to increase drivers' retention. Second, the implementation of fleet monitoring allows businesses to track the vehicle to be able to identify the vehicle location at a given time.

Fleet tracking will provide businesses with real-time data about the vehicle operation that can reduce the vehicle cost and help businesses make better decisions about new vehicle acquisition. At the same time, an effective tracking system can ensure vehicle safety. Third, it provides the businesses with a dynamic and accurate Electronic Logging Device (ELD) compliant system to track the vehicle's hours of service to reduce the accidents that fatigued drivers can cause.

This work proposes a lightweight mutual authentication and key agreement scheme for communication between vehicle to road infrastructure and authentication and key agreement between the vehicle and user to the VOCS. Furthermore, to reduce the latency and support the system scalability, the proposed scheme will present how an RSU can communicate and collaborate with other RSU under the same CRSU to verify the legitimacy of a roaming vehicle's expedited authentication process. The proposed scheme aims to build trust among all communication parities while reducing the computational complexity and the computational cost, ensuring security and preserving privacy during the authentication process and all subsequent communication, and supporting system scalability.

The proposed scheme consists of four phases: the enrollment phase, registration phase, mutual authentication phase, and key agreement phase. The enrollment phase will be completed during manufacturing. The trusted authority (TA) will complete the registration phase when the vehicle communicates with the road infrastructure. The vehicle owner cloud server (VOCS) will complete the registration phase when the vehicle is communicating with the owning company monitoring system. The mutual authentication and the key agreement phases will be completed between V and TA, V and VOCS, and

Driver and VOCS without any human involvement. Every two parties are responsible for ensuring a secure, anonymous mutual authentication and key generation. V and TA, Driver and VOCS, and V and VOCS are based on the client-server topology. The relationship among the different vehicles is based on peer-to-peer topology. The mutual authentication and the key generation phases are broken down into other sub-phases

5. The mutual authentication and key agreement between V and TA
6. The mutual authentication and key agreement between V and VOCS
7. The mutual authentication and key agreement between D and VOCS

The proposed scheme utilizes a one-way hash function, simple XOR function or symmetric encryption between the communicating devices. Furthermore, both vehicles and drivers use Alias IDs (pseudonyms) instead of using their real IDs during the communication process. Alias IDs support the anonymity of senders' IDs, receivers' IDs, and the sender-receiver relationship. The abstract notations used to describe the proposed authentication scheme are listed in table 26. The scheme phases are presented below.

6.6.1 Enrollment Phase

The manufacturer will collect a set of CRPs for each vehicle, RSUs, CRSUs, and the cloud servers during the enrollment phase. The manufacturer will test all the CRPs under different temperature and voltage conditions and consider the aging effects to keep only the error-free CRPs. Then, the manufacturer will load the CRPs of the vehicle, the RSUs, the CRSUs and the servers to the cloud of their service providers. The service provider container is part of the PUF architecture that is presented in chapter 2. The main goals of PUF are to be used as the device biometric that is unique per device, protect the

device from having counterfeited parts, and generate a relevant, unique key for the device. The uniqueness of the generated key in the network would be guaranteed based on the diversity and the manufacturing variations that generate the PUF. The generated PUF responses will be used to generate the symmetric keys.

Table 26: Notations used in the scheme

Notation	**Description**
Trusted Authority	TA
Vehicle	Vi
Vehicle owner cloud server	VOCS
Driver	Di
V_{IDR}	Vehicle Real ID
V_{IDA}	Vehicle Alias ID
TA_{IDR}	Trusted Authority Real ID
TA_{IDA}	Trusted Authority Alias ID
$VOCS_{IDR}$	Vehicle Owner Cloud Server Real ID
$VOCS_{IDA}$	Vehicle Owner Cloud Server Alias ID
OTP	One Time password
OTT	One Time Token
MSK	Master Secret Key
PUF	Physical unclonable function
CHx	Chained hash PUF
C	Challenges
R	Response
V_{TA1}	Authentication parameter between V and TA
V_{CS1}, V_{CS2}	Authentication parameters between V and VOCS

VT_{ID}	Vehicle Type ID
RV	Random values
Ui	Username
PUi	User password
FP	User's fingerprint
BBU	User's behavioral biometric
Z_{U1}, Z_{U2}	Authentication parameters between D and VOCS
TS	Timestamp
H	Hash function

6.6.2 Registration Phase

6.6.2.1 Vehicle – TA registration process

Each vehicle needs to register itself with the TA before being deployed in the field and be part of the IoV system. The TA can be a state-wide entity where all vehicles need to be registered with this TA. The vehicle will send a registration request to the TA, including its ID. Once the TA receives the registration request, it will complete multiple tasks, as described in figures 85 and 86. First, the TA will assign alias IDs to each vehicle. The alias ID will be a fake ID that the vehicle will use to communicate with other communication parities in the IoV domain. Alias IDs will act as pseudonym IDs and will be dynamically changed in every authentication. The goal of the Alias ID goal is to protect and ensure the unlinkability and untraceability of the vehicle and to support the anonymity of each vehicle in each session.

Second, the TA will send the vehicle a PUF challenge to use the PUF function to calculate the corresponding PUF response. The TA will use the PUF response as a seed to calculate the chained hash PUF by calculating the hash value of the PUF response concatenated with the secret vehicle ID. Third, TA will use the PUF function to generate the PUF response (R_{TA}) of its challenge (C_{TA}). The challenge will be stored on the memory of the TA, and the response will be calculated for every authentication session. Fourth, TA will use a random number generator to generate a random value to be used as one-time passwords (OTP) between V and the TA. There will be a unique OTP between each vehicle and the TA. Finally, the TA will insert and store V_{IDR}, TA_{IDR}, $V_{IDA,}$ TA_{IDA}, OTP, and the $CHX_{x,}$ in the OBU of each vehicle.

Vehicle-TA Registration

Step1: The vehicle sends a registration request to the TA, including its ID (V_{IDR}).
Step 2: The TA searches its database to check if the vehicle is registered. If it is not registered, it sends a message to the vehicle, including a PUF challenge, so the response can be used as a seed to create the chained hash PUF
Step 3: Once the vehicle receives the challenge, it will use the PUF function to calculate the response and send it to the TA.
Step 4: Once the TA receives the response, it will verify the received response and compares it with the retrieved one from the PUF cloud. After the successful verification, the TA will generate a nonce (a) and will use the PUF function to generate the response of its challenge and complete the following calculations:

1. Calculates the chained hash (CHX_{xi})= H (Rv || V_{IDR})
2. Generates an OTP_{Vi}
3. R_{TA} = PUF(C_{TA})
4. Calculates an authentication parameter V_{TA1} = H (TA_{IDR} || MSK_{TA} || R_{TA}|| V_{IDR}). The vehicle does not know the $TA_{IDR,}$ $R_{TA,}$ nor $MSK_{TA.}$
5. Calculates an Alias ID for the vehicle V_{IDA} = H (R_1|| a || V_{IDR})

Finally, the TA stores the $V_{IDR,}$ $V_{IDA,}$ $C_{TA,}$ OTP_{Vi}. CHX_{xi} and the vehicle stores on its OBU the $V_{IDA,}$ OTP_{Vi}. CHX_{xi}, $V_{TA1.}$

Figure 85: The registration process between V and TA

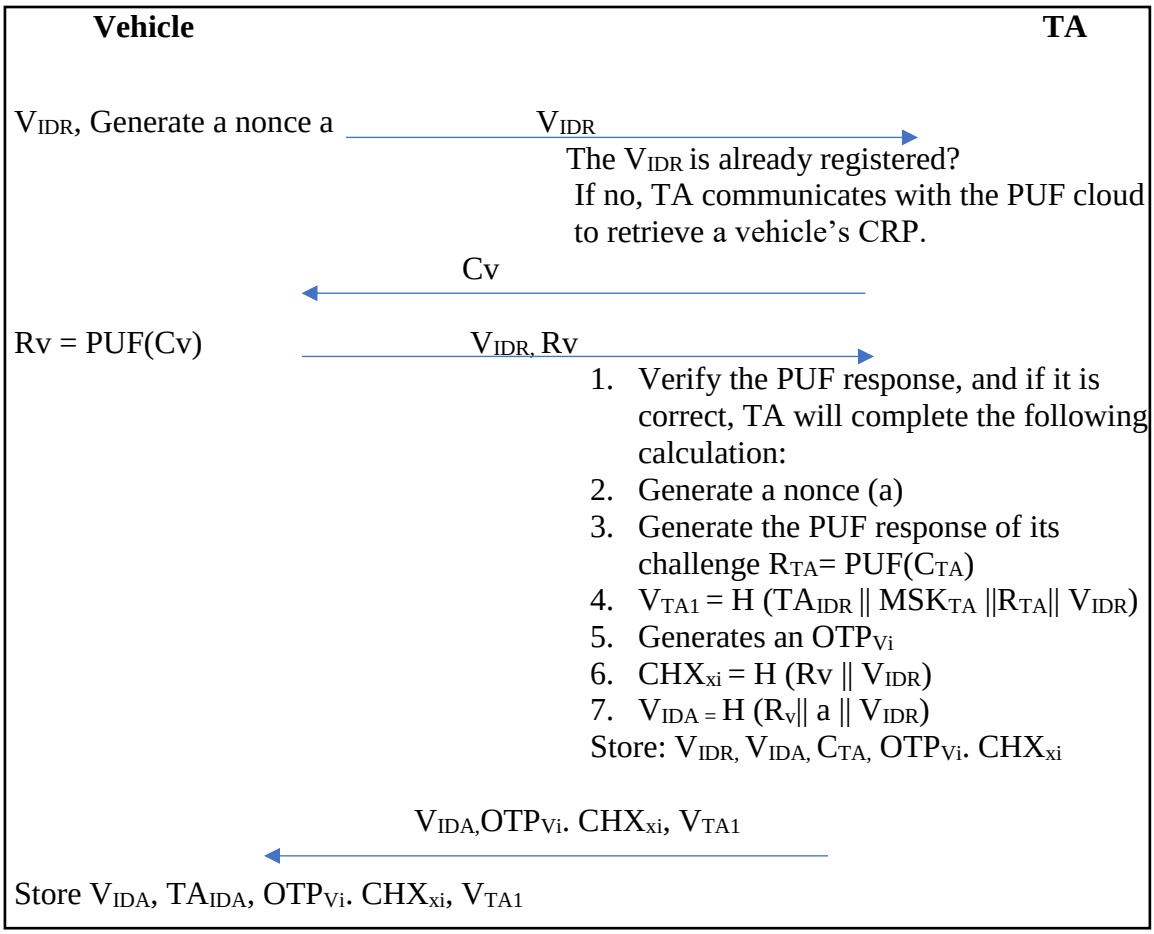


Figure 86: The Exchanged Messages between V and TA during the

6.6.2.2 Vehicle – Vehicle Owner Cloud Server (VOCS) registration process

Each vehicle needs to register itself with its VOCS before being deployed in the field. The VOCS is the owning company cloud server that is in charge of authenticating the vehicle and storing and analyzing the collected data from the vehicle. The vehicle will send a registration request to the VOCS, including its real ID. Once the VOCS receives the registration request, it will complete multiple as described in figure 87 and 88. First, the VOCS will assign an alias ID to each vehicle. The alias ID will be a fake ID that the vehicle will use to communicate with the VOCS. Alias IDs will act as pseudonym IDs

and will be dynamically changed in every authentication. The goal of the Alias ID goal is to protect and ensure the unlinkability and untraceability of the vehicle and to support the anonymity of each vehicle in each session.

Second, VOCS will use a random number generator to generate a nonce (b) to be used in generating the vehicle's alias ID and a random value to be used as a one-time token (OTT) between V and the VOCS. There will be a unique OTT between each vehicle and the VOCS. Third, the VOCS will generate a PUF response (R_{CS}) to its challenge. Fourth, VOCS will use vehicle real ID, its real ID, the R_{VOCS}, and its MSK to generate authentication parameters using a one-way hash function XOR operations. Finally, the VOCS will insert and store V_{IDR}, $V_{IDA,}$ $VOCS_{IDA}$, OTT, and authentication parameters V_{CS1} and $V_{CS2.}$

Vehicle-VOCS Registration

Step1: The vehicle will send a registration request to the VOSC, including its ID (V_{IDR}).
Step 2: VOSC searches its database to check if the vehicle is registered. If not registered, The VOCS will complete the following calculations:

1. Calculates an authentication parameter $V_{CS1} = H (VOCS_{IDR} || MSK_{CS} || R_{CS}|| V_{IDR})$. The vehicle does not know the $VOCS_{IDR,}$ R_{CS}, nor the $MSK_{CS.}$
2. The VOCS generates and assigns a VT_{IDN} to each vehicle based on its type.
3. Calculates an authentication parameter $V_{CS2} = H (V_{CS1} ||VT_{ID})$. The vehicle does not know the $VT_{ID.}$ The VT_{ID} is only used to create V_{CS2}. VT_{ID} is only stored on the VOCS. The vehicle does not know anything about the VT_{ID}. Using the $VOCS_{IDR}$, $R_{CS,}$ the $MSK_{CS,}$ and VT_{ID} to generate the authentication parameters makes it almost impossible for an adversary to identify the VT_{ID} of any vehicle or to identify the $VOCS_{IDR}$, R_{CS}, or the MSK_{CS} of the VOCS.
4. Generates a One-Time-Token OTT_{Vi}
5. Generates a nonce (b)
6. Calculates an Alias ID for the vehicle $V_{IDA} = H (b || V_{IDR})$.

Steps 3: Finally, the VOCS stores the $V_{IDR,}$ C_{CS}, $V_{IDA,}$ OTT_{Vi}. VT_{ID} and the vehicle stores on its OBU the $V_{IDA,}$ OTT_{Vi}, V_{CS1}, V_{CS2}

Figure 87: The registration process between V and VOCS

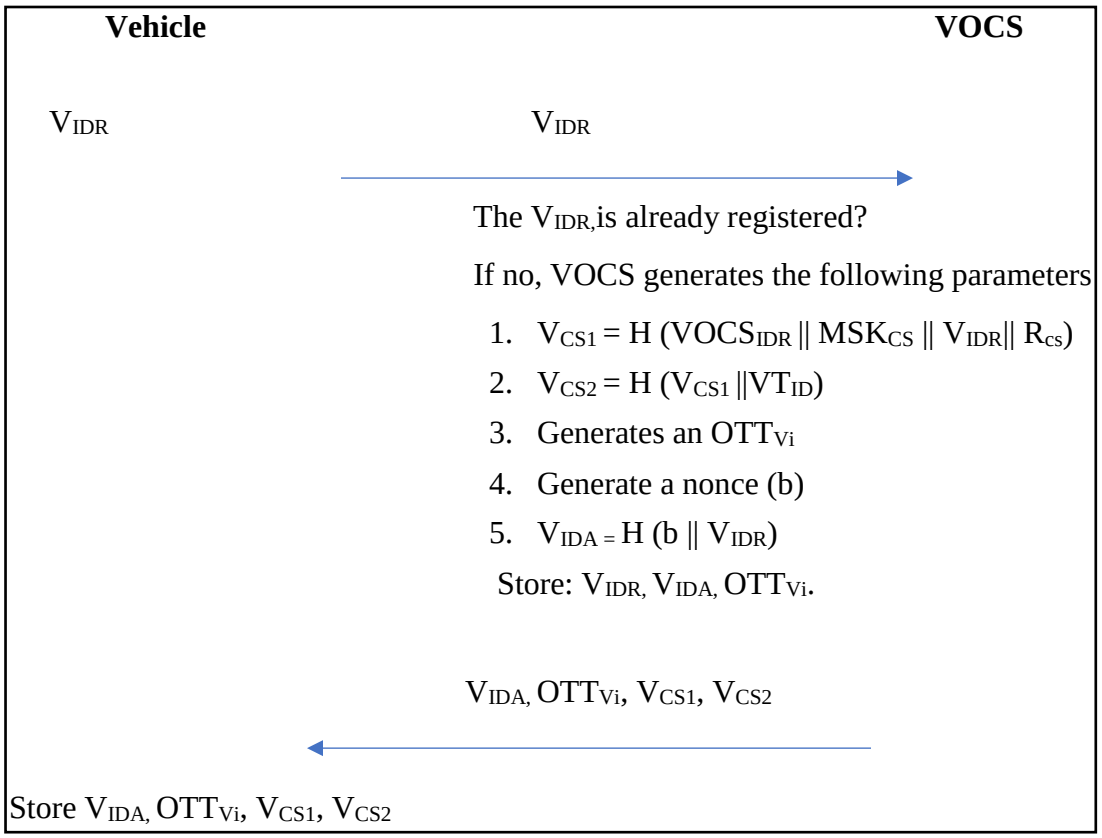


Figure 88: The Exchanged Messages between V and VOCS during the

6.6.2.3 Driver - VOCS Registration

Each driver needs to register himself/herself with its VOCS before driving any vehicle. The VOCS is the owning company cloud server that is in charge of authenticating the driver and storing and analyzing the collected data from the driver. During the registration process, the driver will complete the following tasks: (1) the driver will select a username and a password using the vehicle's touch screen; (2) the driver will use the vehicle fingerprint scanner to create a biometric template that will be used for static authentication; and (3) For the continuous authentication, we propose to use a built-in sensor in the vehicle steering wheel to collect the driver's ECG or EKG. The driver's ECG or EKG features are extracted to create a biometric template, and it will be used for the continuous authentication process. The driver computes the authentication

parameter $Z_{U1} = H (FP_U || PUi)$. VOCS will generate the PUF response (R_{UCS}) of its challenge (C_{UCS}) and computes the authentication parameter $Z_{U2} = H (Z_{u1} || R_{UCS})$. VOCS will store Ui and C_{UCS} in clear text and the E_{ID}, $BB_{Ui,}$ and Z_{U2} encrypted by the PUF response (R_{UCS}) into its database for later authentication. The registration process is presented in figures 89 and 90.

Driver-VOCS Registration

Step1: The driver will complete the following tasks:

1. Generate a username (U_i) and password PU_i) using the vehicle's touch screen.
2. Capture fingerprint features (FP_{Ui}) using a vehicle fingerprint scanner to create a biometric template that will be used for static authentication
3. Capture the blood pressure or the ECG features (BB_{Ui}) using a built-in sensor in the vehicle steering wheel to create a biometric template that will be used for continuous authentication.
4. computes $Z_{u1} = H (FP_U || PUi)$

Step 2: VOSC searches its database to check if the user is registered. If not registered, the VOCS will:

1. Generate the PUF response of its challenge $R_{UCS} = PUF(C_{Ucs})$.
2. Compute $Z_{u2} = H (Z_{u1} || R_{Ucs})$.
3. Store the $U_{i,}$ and C_{UCS} in cleartext. E_{ID}, $Z_{U2,}$ and BB_{Ui} in its database are encrypted by the PUF response for further comparison during the authentication process.

Figure 89: The registration process between D and VOCS

Driver	**VOCS**
Employee ID(E_{ID}) The driver will complete the following tasks: 1. Generate a username (U_i) 2. Generate a password PU_i) 3. Capture a fingerprint features (FP_U) 4. Capture the EKG or the ECG features (BB_{Ui}) 5. Comptes $Zu_1 = H (FP_U || PU_i)$	
E_{ID}, $U_{i,}$ Z_{U1} BB_{Ui} →	
	1. Generate a PUF response Rcs = PUF(C_{UCSU}) 2. Computes $Z_{U2} = H (Z_{U1} || R_{U}cs)$ 3. Stores: E_{ID}, $U_{i,}$ BB_{Ui}, Z_{U2}, C_{UCS}

Figure 90: The exchanged messages between D and VOCS during the registration process

6.6.3 Mutual authentication phase

This phase will cover two different protocols. The first protocol is to achieve mutual authentication between the vehicle and the TA. The second protocol is to achieve mutual authentication among vehicle, driver, and the VOCS.

6.6.3.1 Protocol 1: Vehicle-TA mutual authentication and key generation phases

Vehicle to TA mutual authentication process will be executed when a vehicle V_i wants to send the data that it collects from the surrounding environment to the TA with the help of the RSU as presented in figures 91 and 92 . However, the RSU can't validate the legitimacy of the V_i. Therefore, the RSU needs to interact with the TA to authenticate the V_i.

V	TA
Generate: $TS_{Vi1,}$ RV_{Vi1} **Compute**: $X_1 = RV_{Vi1} \oplus H(V_{TA1})$ $X_2 = H\ (RV_{Vi1} \| V_{TA1} \| TS_{Vi1} \| V_{IDR})$	
M1: $(V_{IDA}, TS_{Vi1}, X_{1,} X_2)$ →	
	Check$\| TS_{Rec\_11Rec} - TS'_{Vi1}\| < \Delta T$ **Find**: V_{IDA} **Read**: $V_{IDR,}$ OTP, C_{TA}, MSK_{TA} **Compute**: $R_{TA} = PUF(C_{TA})$ $V_{TA1}' = H\ (MSK_{TA} \| V_{IDR} \| R_{TA} \| TA_{IDR})$ $RV'_{Vi1} = H\ (V_{TA1}') \oplus X_1$ **Verify**: $X_2^{'} = H\ (RV_{Vi1} \| V_{TA1'} \| TS_{Vi1'} \| V_{IDR})$ **Generate**: $TS_{TA1,}$ $RV_{TA1,}$ S_{ID} **Retrieve**: Vi CRP, CHX_x **Computes**: $Y_1 = H\ (V_{TA1}' \| OTPVi)$ $Y_2 = RV_{TA1} \oplus Y_1$ $C' = H\ (CHX_x \| RV_{TA1)} \oplus C$ $Y_3 = H\ (C \| TS_{TA1} \| Y_1 \| RV_{TA1} \| RV_{Vi1'})$

M2: (TA_{IDA}, S_{ID}, TS_{TA1}, Y_2, C', Y_3)

Check: | TS_{Rec} - TS'_{TA1} |< ΔT

Compute:
$Y_1' = H(V_{TA1}' || OTPVi)$
$RV_{TA1}' = Y_1 \oplus Y_2$
$C'_X = C' \oplus H(H(CHX_x || RV_{TA1}'))$
Verify
$Y_3' = H(C || TS_{TA1} || Y_1' || RV_{TA1}' || RV_{Vi1}')$
Generate: TS_{Vi2}, RV_{Vi2}
Compute
$Ri = PUF(C'_X)$
$CHX_{x+1} = H(R^i || CHXx)$
$X_3 = RV_{Vi2} \oplus H(RV_{TA1} || CHX_{x+1})$
$X_4 = H(V_{IDR} || TS_{Vi2} || RV_{Vi2} || CHX_{x+1})$

M3: (V_{IDA}, S_{ID}, TS_{Vi2}, X_3, X_4)

Check: TS_{Rec} - TS'_{Vi2} |< ΔT
Compute:
$CHA_{x+1}' = H(R'_i || CHXx)$
$RV'_{vi2} = X_3 \oplus (RV_{TA1} || CHX_{x+1}')$
Verify:
$X_4' = H(V_{IDR} || TS_{Vi2}' || RV_{Vi2}' || CHX_{x+1}')$
Generate: TS_{TA2}, RV_{TA2}
Compute
$Y_4 = H(RV_{Vi2}' || R^i)$
$Y_5 = RV_{TA2} \oplus Y_4$
$OTP_{new} = H(OTP_{new-1} || RV_{TA2} || R_i)$
$V_{IDAnew} = H(C || V_{IDA})$
$Y_6 = H(V_{IDR} || Y_4 || TS_{TA2} || RV_{TA2} || OTP_{new} || V_{IDAnew})$

M4: (TA_{IDA}, S_{iD} TS_{TA2}, Y_5, Y_6)

Check: |TS'_{Rec} – TS_{TA2} |< ΔT
Compute:
$RV'_{TA2} = Y_4 \oplus H(RV_{Vi2}' || R^i)$
$OTP_{new}' = H(OTP_{new-1} || RV_{TA2}' || R_i)$
$V_{IDAnew}' = H(C'_X || V_{IDA})$
Verify:
$Y6' = H(V_{IDR} || Y_4' || TS_{TA2} || RV_{TA2} || OTP_{new}' || V_{IDAnew}')$
Store (OOTPnew','VIDAnew'')

Figure 91: V-TA mutual authentication process

Algorithm 6: The mutual authentication between V and TA

Input:

V with real identity (V_{IDR}), alias identityV_{IDA}), TA alias identity (TA_{IDR}), Authentication parameters (V_{TA1}), one-time password (OTPvi), and the chained hash PUF (CHX_{xi})

TA with real identity (V_{IDR}), alias identity (V_{IDA}), TA real identity (TA_{IDR}), TA alias identity (TA_{IDR}), PUF challenge (C_{TA}), and one time password (OTP), and the chained hash PUF (CHX_{xi}).

Output:

Mutual authentication between the vehicle (N) and the Trusted Authority (TA)

Begin

1. V generates a random nonce $RVvi_1$, a timestamp $TSvi_1$, computes $X_1 = RV_{Vi1} \oplus H(V_{TA1})$ and $X_2 = H(RV_{Vi1} \| V_{TA1} \| TS_{Vi1} \| V_{IDR})$.
2. V sends (V_{IDA}, $TSvi_1$, X_1, X_2) message to TA.
3. **If** (TA finds V_{IDA} in its repository)
4. **then**
5. TA retrieves V_{IDR}, OTP, and C_{TA} that belongs to the V_{IDA} from its repository to its memory.
6. The TA passes the challenge C_{TA} to its PUF function and generates a response $R_{TA.}$
7. TA calculates V_{TA1}. Then the TA retrieves $RVvi_1$ from XORing X_1 and $H(V_{TA1})$ and finally calculates $X_2' = H(RV_{Vi1} \| V_{TA1} \| TS_{Vi1} \| V_{IDR})$.
8. **If** (the calculated hash message in step 6 matches the hash message that was sent in step 2)
9. **then**
10. The TA communicates with the Grand PUF node to retrieve a CRP of the Vi, generates a timestamp TS_{TA1}, generates a random nonce RV_{TA1}, calculates $Y_1 = H(V_{TA1}' \| OTPVi)$, $Y_2 = RV_{TA1} \oplus Y_1$, $C' = H(CHX_x \| RV_{TA1}) \oplus C$, and $Y_3 = H(C \| TS_{TA1} \| Y_1 \| RV_{TA1} \| RV_{Vi1}')$
11. The TA sends $TA_{IDA,}$ S_{ID}, TS_{TA1}, Y_2, C', Y_3 to V.
12. **else**
13. Go to step 42.
14. **end if**
15. **else**
16. Go to step 42.
17. **end if**
18. V verifies the time stamp, calculates $Y_1' = H(V_{TA1}' \| OTPVi)$, retrieves RV_{TA1}' and $C'_{X.}$ V computes $Y_3' = H(C \| TS_{TA1} \| Y_1 \| RV_{TA1} \| RV_{Vi1}')$
19. **If** (the calculated hash message in step 18 matches the hash message that was sent in step 11)
20. **then**
21. **The authenticity of the TA is verified**
22. V generates a timestamp $TSvi_2$ and a random nonce $RVvi_{2.}$ V computes Ri =

PUF **(**C'_X), $CHX_{x+1} = H(R^i || CHXx)$, $X_3 = RV_{Vi2} \oplus H(RV_{TA1} || CHX_{x+1})$ and $X_4 = H(V_{IDR} || TS_{Vi2} || RV_{Vi2} || CHX_{x+1})$

23. V sends V_{IDA}, S_{ID}, $TSvi_2$, X_3, X_4 to TA.
24. **else**
25. Go to step 42.
26. **end if**
27. The TA verifies the TS_{vi2}, computes $CHX_{x+1}' = H(R'_i || CHXx)$, retrieves $RV'_{vi2} = X_3 \oplus (RV_{TA1} || CHX_{x+1}')$ and generates $X_4' = H(V_{IDR} || TS_{Vi2}' || RV_{Vi2}' || CHX_{x+1}')$
28. **If** (the calculated hash message in step 27 matches the hash message that was sent in step 23
29. **then**
30. **The authenticity of V is verified.**
31. TA generates TS_{TA2} and RV_{TA2}. Then TA computes $Y_4 = H(RV_{Vi2}' || R^i)$, $Y_5 = RV_{TA2} \oplus Y_4$, $OTP_{new} = H(OTP_{new-1} || RV_{TA2} || R_i)$, $V_{IDAnew} = H(C || V_{IDA1})$ and, $Y_6 = H(V_{IDR} || Y_4 || TS_{TA2} || RV_{TA2} || OTP_{new} || V_{IDAnew})$
32. TA sends TA_{IDA}, S_{iD} TS_{TA2}, Y_5, Y_6 to V.
33. **else**
34. Go to step 42.
35. **End if**
36. V verifies the TS_{TA2}, computes $RV'_{TA2} = Y_4 \oplus H(RV_{Vi2}' || R^i)$, $OTP_{new'} = H(OTP_{new-1} || RV_{TA2}' || R_i)$, $V_{IDAnew}' = H(C'_X || V_{IDA})$ and verifies $Y6' = H(V_{IDR} || Y_4' || TS_{TA2} || RV_{TA2} || OTP_{new}' || V_{IDAnew}')$
37. **If** (the calculated hash message in step 36 matches the hash message that was sent in step 32
38. **then**
39. **V stores the received data in its database**
40. **else**
41. Go to step 42
42. **Stop** (terminates the connection).
43. **End**

Figure 92: Algorithm of the mutual authentication algorithm between V and TA

This scheme consists of the following steps:

6.6.3.1.1 Step 1: Interaction Request

6. V_i generates a new TS_{Vi1} to avoid replay attacks
7. V_i selects a random parameter RV_{Vi1} and then calculates X_1
    a. $X_1 = RV_{Vi1} \oplus H(V_{TA1})$
8. V_i computes $X_2 = H(RV_{Vi1} || V_{TA1} || TS_{Vi1} || V_{IDR})$
9. V_i sends the authentication request to the TA.

The authentication request includes $V_{IDA,}$ $TS_{Vi1,}$ $X_{1,}$ and X_2, as shown in 6.1. The timestamp presents information about when an event occurred, which makes this value important.

$$V_i \longrightarrow \text{TA Authentication-Req } (V_{IDA}, TS_{Vi1,} X_{1,} X_2) \qquad (6.1)$$

6.6.3.1.2 Step 2: TA Response

Once the TA receives the authentication request, it will complete the following steps:

1. TA checks the validity of the received timestamp $| TS'_{Rec} - TS'_{Vi1} | < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.
2. TA retrieves from its database the V_{IDR} that corresponds to its V_{IDA}. If the TA did not find the V_{IDA} in its database, it would send a message to the vehicle to complete the registration process.
3. Once the TA finds the V_{IDA}, it will retrieve the C_{TA} of the V_i.
4. TA computes $R_{TA} = PUF(C_{TA})$
5. TA computes the $V_{TA1}' = H (MSK_{TA} \| V_{IDR} \| R_{TA} \| TA_{IDR})$
6. TA computes the $RV'_{Vi1} = H (V_{TA1}') \oplus X_1$
7. TA computes $X_2' = H (TS'_{Vi1} \| RV'_{Vi1} \| V_{IDR} \| V_{TA1}')$

   If the computed value of X_2' is equal to the received value, the TA accepts the authentication request; otherwise, it will drop it as shown in figure 95.

$X_2^{'}$ = H (TS'_{Vi1} || RV'_{Vi1} || V_{IDR} || V_{TA1}')
If $X_2 = X_2^{'}$ Then
The TA will accept the authentication request
Else
The TA will drop the authentication request

Figure 93: Evaluating the Vi Authentication Parameter

Once the TA accepts the authentication request, it will retrieve a Vi CRP from the PUF cloud and generates a random value RV_{TA1} and time stamp TS_{TA1}. Also, the TA generates a session ID. The session ID aims to distinguish one session from the rest of the running sessions simultaneously. The TA prepares and sends a connection response to the Vi. The preparation process of the connection response message includes the following steps:

1. TA retrieve a Vi CRP from the PUF cloud
2. TA generates a new TS_{TA1} to avoid replay attacks
3. TA calculates Y_1
    a. $Y_1 = H (V_{TA1}' ||OTPVi)$
4. TA generates a random parameter RV_{TA1} and then calculates Y_1
    a. $Y_2 = RV_{TA1} \oplus Y_1$
5. TA calculates C'
    a. $C' = H (CHX_x || RV_{TA1)} \oplus C$
6. TA calculates $Y_3 = H (C || TS_{TA1} || Y_1 ||RV_{TA1} || RV_{Vi1'})$
7. TA generates a session ID(S_{ID})
8. TA sends a connection response message

The connection response includes $TA_{IDA,}$ session ID (S_{ID}), $TS_{TA1,}$ C', Y_2, and Y_3, as shown in 6.2. The timestamps present information about when an event occurred, which makes this value important.

$$TA \longrightarrow Vi\ M2(TA_{IDA}, S_{ID,} TS_{TA1,} C', Y_{2,} Y_3) \qquad (6.2)$$

6.6.3.1.3 Step 3: TA authentication

Once the Vi receives M2, it will complete the following steps:

6. Vi checks the validity of the received timestamp $|TS_{Rec} - TS'_{TA1}| < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and when the message is received is higher than the expected maximum transmission delay.
7. Vi calculates $Y_1' = H(V_{TA1}' \| OTPVi)$
8. Vi calculates $RV_{TA1}' = Y_1 \oplus Y_2$
9. Vi calculates the $C'_X = C' \oplus H(H(CHX_x \| RV_{TA1}'))$
10. As presented in figure 94, Vi calculates Y_3' using a one-way hash function and compares the generated Y_3' with the received Y_3. If the two values are the same, Vi will authenticate TA; otherwise, it will drop it.

$Y_3' = H(C \| TS_{TA1} \| Y_1' \| RV_{TA1}' \| RV_{Vi1}')$
If
$Y_3' = Y_3$
Then
Then Vi will accept M2 and authenticate TA

Else Vi will drop M2

Figure 94: Evaluating the TA authentication parameter

Once Vi accepts the received message and authenticates TA, it will complete the following steps

1. Vi will input the received challenge Cx' to its PUF and obtains the response Ri.
2. Vi will retrieve the chained hash responses (CHX_x) from its memory and calculate the new value of the chained has responses (CHX_{x+1}) as follows
    a. $CHX_{x+1} = H(R^i \| CHXx)$
3. Vi generates a new TS_{Vi2} to avoid replay attacks
4. Vi selects a random parameter RV_{Vi2} and then calculates X_3
    a. $X_3 = RV_{Vi2} \oplus H(RV_{TA1} \| CHX_{x+1})$
5. Vi calculate X_4
    a. $X_4 = H(V_{IDR} \| TS_{Vi2} \| RV_{Vi2} \| CHX_{x+1})$
6. Vi sends the message to the TA

The message includes $V_{IDA,}$ $S_{ID,}$ TS_{Vi2}, X_3 and X_4 as shown in 6.3.

$$Vi \longrightarrow TA\ (V_{IDA,}\ TS_{Vi2},\ X_{3,}\ X_4) \qquad (6.3)$$

6.6.3.1.4 Step 4: Vi authentication

Once TA receives the message, it completes the following steps:

1. TA checks the validity of the received timestamp $|TS'_{Rec} - TS'_{vi2}| < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.
2. TA will retrieve the R'_i and the CHXx from its databased and computes CHX'^{i+1}
    a. $CHX_{x+1}' = H(R'_i \,||\, CHXx)$
3. TA calculates the $RV'_{vi2} = X_3 \oplus (RV_{TA1} || CHX_{x+1}')$
4. TA calculates $X_4' = H(V_{IDR} || TS_{Vi2}' || RV_{Vi2}' || CHX_{x+1}')$

As presented in figure 95, TA calculates X_4' using a one-way hash function and compares the generated X_4' with the received X_4. If the two values are the same, TA will authenticate Vi; otherwise, it will drop it.

$X_4' = X_4 = H(V_{IDR} || TS_{Vi2}' || RV_{Vi2}' || CHX_{x+1}')$
If $X_4' = X_4$ Then
Then TA will authenticate the Vi.
Else
TA will drop the message

Figure 95: Evaluating the V authentication parameter

After authenticating the Vi, the TA will override the old CHX_x with the new chained hash responses value CHX_{x+1}. Then the TA will generate a new OTP and the new alias ID for the Vi and store them on its database.

1. TA generates a new timestamp TS_{TA2}
2. TA calculates Y_4
    a. $Y_4 = H(RV_{Vi2}' || R^i)$

3. TA selects a random parameter RV_{TA2} and then calculates Y_5

    a. $Y_5 = RV_{TA2} \oplus Y_4$

4. TA computes $OTP_{new} = H (OTP_{new-1} || RV_{TA2} || R_i)$

5. TA computes $V_{IDAnew} = H (C || V_{IDA})$

6. TA calculates Y_6

    a. $Y_6 = H (V_{IDR} || Y_4 || TS_{TA2} || RV_{TA2} || OTP_{new} || V_{IDAnew})$

7. TA sends the message to the Vi. The message includes $TA_{IDA,}$ $S_{ID,}$ TS_{TA2}, Y_5, and Y_6, as shown in 6.4.

$$TA \longrightarrow Vi\ (TA_{IDA,}\ TS_{TA2},\ Y_{5,}\ Y_6) \qquad (6.4)$$

6.6.3.1.5 Step 5: New Parameters Verification

Once Vi receives the message, it completes the following steps:

1. Vi checks the validity of the received timestamp $|TS'_{Rec} - TS'_{TA2}| < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.

2. Vi calculates the $RV'_{TA2} = Y_4 \oplus H (RV_{Vi2}' || R^i)$

3. Vi computes $OTP_{new'} = H (OTP_{new-1} || RV_{TA2}' || R_i)$

4. Vi computes $V_{IDAnew}' = H (C'_X || V_{IDA})$

5. Vi calculates Y_6'

a. $Y6' = H (V_{IDR} || Y_4' || TS_{TA2} || RV_{TA2} || OTP_{new}' || V_{IDAnew}')$

As presented in figure 96, Vi calculates Y_6' using a one-way hash function and compares the generated Y_6' with the received Y_6. If the two values are the same, Vi will accept the message; otherwise, it will drop it.

$Y_6' = Y_6 = H (V_{IDR} || Y_4' || TS_{TA2} || RV_{TA2} || OTP_{new}' || V_{IDAnew}')$
If $Y_6' = Y_6$ Then
Then Vi will accept the message.
Else
Vi will drop the message

Figure 96: TA -V Parameters Verification

After verifying and accepting the message, the Vi, will store the $OTP_{new,}$ $V_{IDAnew,}$ in its database for the next authentication session.

6.6.3.1.6 Key generation

Once the mutual authentication is completed, the two sides generate a shared secret session key (*ssk*) using the technique they agreed upon during the authentication phase. The two sides generate the *ssk* locally by hashing a combination of PUF response and the generated random values as presented in 6.5. The uniqueness of the PUF responses ensures the uniqueness of the generated *ssk*.

$$ssk = H (Ri || (RV_{vi1} || RV_{TA2} || RV_{vi2} || RV'_{TA1}) \quad (6.5)$$

Then the TA will send the *ssk* of Vi to the CRSU through a secure channel. Once the CRSU received the key it will share it will all its RSU to communicate with Vi. By using the *ssk*, Vi can communicate with all RSUs that are within the same CRSU. However, when Vi enters an area of another CRSU, it has to authenticate itself with the TA again. Finally, the RSU that communicates with Vi will share the secret broadcast

key (SKB) with it that Vi can use to communicate with other vehicles within the same CRSU. The RSU will send the SK_B to Vi encrypted by the *ssk*.

6.6.3.2 Protocol 2: Vehicle– Driver-Vehicle Owner Cloud Server (V-D-VOCS) Authentication and Key Generation

This authentication phase consists of two phases. The first one handles the mutual authentication process between a vehicle (Vi) and a vehicle owner cloud server (VOCS). The second one focuses on the mutual authentication between the driver (Di) and the VOCS where the vehicle acts as a relay between Di and VOCS. The first phase must be completed first before starting on the second phase.

6.6.3.2.1 Phase one: Vehicle -Vehicle Owner Cloud Server Mutual Authentication

As presented in figures 97 and 98, this process starts when Vi prepares an authentication request message that will be sent to the VOCS. All vehicles of the same type share a typeID (VT_{ID}). Each vehicle stores two authentication parameters that will be used during the authentication process. Furthermore, all OBUs in the vehicles are equipped with PUF chips that will be used to build a root of trust. Each vehicle knows its Real and alias IDs, the alias ID of the VOCS, the OTT, and the two authentication parameters. The steps involved in this process are listed below.

V	VOCS

Generate: $TSvi_1$, $RVvi_1$
Compute:
$VX_1 = H(V_{CS2} \| OTT)$
$VX_2 = H(VX_1) \oplus RV_{Vi1}$
$VX_3 = H(V_{IDR} \| V_{CS1} \| TS_{Vi1} \| RV_{Vi1} \| OTT)$

M1: $(V_{IDA}, TSvi_1, VX_2, VX_3)$ →

Check: $|TS_{Rec} - TS'vi_1| < \Delta T$
Find: V_{IDA}
Read: V_{IDR}, VT_{ID}, C_{CS}, and OTT
Compute:
$R_{CS} = PUF(C_{CS})$
$V_{CS1} = H(VOCS_{IDR} \| MSK_{VOCS} \| V_{IDR} \| R_{cs})$.
$V_{CS2} = H(V_{CS1} \| VT_{ID})$
$VX_1' = H(V_{CS2'} \| OTT)$
$RV'_{Vi1} = VX_2 \oplus VX_1'$
Verify: $VX_3' = H(V_{IDR} \| V_{CS1}' \| TS_{Vi1}' \| RV_{Vi1}' \| OTT)$
Generate: TS_{CS1}, RV_{CS1}, S_{ID}
Computes:
$CSY_1 = H(V_{IDR} \| V_{CS1'} \| OTT)$
$CSY_2 = H(CSY_1) \oplus RV_{CS1}$
$CV = C \oplus H(CSY_1 \| RV_{CS1})$
$CSY_3 = H(VOCS_{IDR} \| CSY_1 \| TS_{CS1} \| RV_{Vi1} \| C \| V_{CS1} \| RV_{CS1})$

M2: $(VOCS_{IDA}, S_{ID}, TS_{CS1}, CSY_2, CV, CSY_3)$ ←

Check: $|TS'_{Rec} - TS'_{CS1}| < \Delta T$
Compute:
$CSY_1' = H(V_{IDR} \| V_{CS1'} \| OTT)$
$RV'_{CS1} = H(CSY_1) \oplus CSY_2$
$C' = CV \oplus H(RV'_{CS1} \| CSY_1')$
Verify
$CSY_3' = H(VOCS_{IDR} \| CSY_1' \| TS_{CS1}' \| RV_{Vi1} \| C' \| V_{CS1}' \| RV_{CS1}')$
Generate: $TSvi_2$, $RVvi_2$
Compute
$Ri = PUF(C')$
$C^{i+1} = H(RV'_{CS1} \| RVvi_1)$
$R_{i+1} = PUF(C^{i+1})$
$VX_4 = H(OTT \| RV_{CS1})$
$VX_5 = RV_{vi2} \oplus H(VX_4 \| V_{IDR})$
$VX_6 = H(VX_4 \| RV_{Vi2}) \oplus R^i$
$VX_7 = H(C' \| R') \oplus Ri+1$
$OTTnew = H(OTT \| RV_{Vi2} \| C)$
$V_{IDAnew} = H(V_{IDA} \| C \| OTT)$
$VX_8 = H(VX_4 \| RV_{Vi2} \| TS_{Vi2} \| Ri \| C^{i+1} \| R^{i+1} \| OTTnew \| V_{IDAnew})$

M3: $(V_{IDA}, S_{ID}, TS_{Vi2}, VX_5, VX_6, VX_7, VX_8)$ →

Check: $|TS'_{Rec} - TS'vi_2| < \Delta T$
Read: R'_i
Compute:
$VX_4' = H(RV_{CS1} \| OTT)$
$RV'_{VI2} = VX_5' \oplus H(VX_4' \| V_{IDR})$
$R_i' = VX_6' \oplus H(RV_{Vi2} \| VX_4')$
$C^{i+1} = H(RV'_{CS1} \| RV_{Vi1})$
$R^{i+1'} = VX_7' \oplus H(C' \| R_i')$
$OTTnew' = H(OTT \| RV_{Vi2} \| C)$
$V_{IDAnew'} = H(V_{IDA} \| C \| OTT)$
Verify:
$VX_8 = H(VX_4 \| RV_{Vi2} \| TS_{Vi2} \| Ri \| C^{i+1} \| R^{i+1} \| OTTnew \| V_{IDAnew})$
Store: $OTT_{new}, V_{IDAnew}, C'^{i+1}, R^{i+1}$

Figure 97: V-VOCS Mutual Authentication Process

6.6.3.2.1.1 Step 1: Interaction request

1. Vi generates a new TS_{Vi1} to avoid replay attacks
2. Vi computes $VX_1 = H(V_{CS2} \| OTT)$
3. Vi selects a random parameter RV_{Vi1} and then calculates VX_2
    a. $VX_2 = VX_1 \oplus RV_{Vi1}$
4. Vi computes $VX_3 = H(V_{IDR} \| V_{CS1} \| TS_{Vi1} \| RV_{Vi1} \| OTT)$
5. Vi sends a authentication request message.

The authentication request includes V_{IDA}, TS_{vV1}, VX_2, and VX_3 as shown in 6.6.

$$Vi \longrightarrow \text{VOCS Auth-Req } (V_{IDA}, TS_{Vi1}, VX_2, VX_3) \qquad (6.6)$$

Algorithm 6:_Protocol 2_ Phase 1: The mutual authentication between V-VOCS
Input: **Vehicle** with real identity (V_{IDR}), alias identity (V_{IDA}), VOCS real identity ($VOCS_{IDR}$), VOCS alias identity ($VOCS_{IDR}$), Authentication parameters (V_{CS1}, V_{CS2}), and one time token (OTT).

VOCS with with real identity (V_{IDR}), alias identity (V_{IDA}), VOCS real identity ($VOCS_{IDR}$), VOCS alias identity ($VOCS_{IDR}$), the vehicle type ID(VT_{ID}) Master secret key (MSKcs), PUF challenge (C_{CS}), and one-time token (OTT).

Output:

Mutual authentication between the vehicle (V) and the VOCS.

Begin

1. V generates a random nonce $RVvi_1$, a timestamp $TSvi_1$, computes $VX_2 = H(VX_1) \oplus RV_{Vi1}$ and $VX_3 = H(V_{IDR} || V_{CS1} || TS_{Vi1} || RV_{Vi1} || OTT)$
2. V sends (V_{IDA}, $TSvi_1$, VX_2, VX_3) message to the VOCS.
3. **If** (the VOCS finds V_{IDA} in its repository).
4. **then**
5. The VOCS retrieves V_{IDR}, OTT, C_{CS}, and VT_{ID} that belongs to the V_{IDA} from its repository to its memory. The VOCS passes the challenge C_{CS} to its PUF function and generates a response R_{CS}
6. The VOCS calculates V_{CS1}' and V_{CS2}'. Then the VOCS computes VX_1, retrieves the $RVvi_1$ from XORing $VX_2 \oplus VX_1$' and finally calculates VX_3' $H(V_{IDR} || V_{CS1} || TS_{Vi1} || RV_{Vi1} || OTT)$
7. **If** (the calculated hash message in step 7 matches the hash message that was sent in step 2)
8. **then**
9. The VOCS communicates with the Grand PUF node to retrieve a CRP of Vi, generates a timestamp TS_{CS1}, generates a random nonce RV_{CS1}, calculates $CSY_1 = H(V_{IDR} || V_{CS1'} || OTT)$, $CSY_2 = H(CSY_1) \oplus RV_{CS1}$, $CV = C \oplus H(CSY_1 || RV_{CS1})$, and $CSY_3 = H(VOCS_{IDR} || CSY_1 || TS_{CS1} || RV_{Vi1} || C || V_{CS1} || RV_{CS1})$.
10. The VOCS sends $VOCS_{IDA}$, S_{ID}, TS_{CS1}, CSY_2, CV, CSY_3 to Vi.
11. **else**

Go to step 29.

end if

12. **else**

Go to step 29.

end if

13. V verifies the time stamp, calculates CSY_1' to retrieves RV_{CS1}' and C' from CV. V generates CSY_3' $H(VOCS_{IDR} || CSY_1 || TS_{CS1} || RV_{Vi1} || C || V_{CS1} || RV_{CS1})$.
14. **If** (the calculated hash message in step 14 matches the hash message that was sent in step 11)
15. **then**

The authenticity of the VOCS is verified

16. V generates a timestamp $TSvi_2$ and a random nonce $RVvi_2$. V computes $Ri = PUF(C')$, $C^{i+1} = H(RV'_{CS1} \| RVvi_1)$, $R_{i+1} = PUF(C^{i+1})$, $VX_4 = H(OTT \| RV_{CS1})$, $VX_5 = RV_{vi2} \oplus H(VX_4 \| V_{IDR})$, $VX_6 = H(VX_4 \| RV_{Vi2}) \oplus R^i$, $VX_7 = H(C' \| R') \oplus Ri+1$, $OTTnew = H(OTT \| RV_{Vi2} \| C)$, $_{IDAnew} = H(V_{IDA} \| C \| OTT)$, $VX_8 = H(VX_4 \| RV_{Vi2} \| TS_{Vi2} \| Ri \| C^{i+1} \| R^{i+1} \| OTTnew \| V_{IDAnew})$
17. V sends V_{IDA}, S_{ID}, TS_{Vi2}, VX_5, VX_6, VX_7, VX_8 to VOCS.
18. **else**
19. Go to step 29
20. **end if**
21. V verifies the time stamp TS_{vi2}, calculates VX_4' to retrieves $RViv2'$, VX6, and VX_7. V generates $VX_8 = H(VX_4 \| RV_{Vi2} \| TS_{Vi2} \| Ri \| C^{i+1} \| R^{i+1} \| OTTnew \| V_{IDAnew})$
22. **If** (the calculated hash message in step 22 matches the hash message that was sent in step 18)
23. **then**
24. **The authenticity of V is verified.**
25. Mutual authentication between V and the VOCS is established.
26. **else**
27. Go to step 29
28. **End if**
29. **Stop** (terminates the connection).

End

Figure 98: The mutual authentication between V and the VOCS

6.6.3.2.1.2 Step 2: VOCS response

Once the VOCS receives the authentication request, it will complete the following steps:

1. VOCS checks the validity of the received timestamp $|TS_{Rec} - TS'_{Vi1}| < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.

2. VOCS retrieves from its database the V_{IDR} that corresponds to its V_{IDA}. If the VOCS did not find the N_{IDA} in its database, it will ignore and drop the authentication request.

Once the VOCS finds the V_{IDR}, it will retrieve it along with the $VT_{ID,}$ its PUF Challenge that relates to this Vi and OTT of Vi.

1. VOCS generates the PUF response R_{cs} = PUF(Ccs)
2. VOCS generates V_{CS1} = H ($VOCS_{IDR}$ || MSK_{VOCS} || V_{IDR}|| R_{cs}).
3. VOCS generates V_{CS2} = H (V_{CS1} || VT_{ID})
4. VOCS computes VX_1' = H ($V_{CS2'}$ || OTT)
5. VOCS computes the $RV'_{Vi1} = VX_2 \oplus VX_1'$

The VOCS uses the calculated V_{CS2}, $TS'_{Vi1,}$ OTT, and the RV'_{Vi1} to generate VX_3', a combination of $V_{IDR,}$ V_{CS3}, $TS'_{Vi1,}$ OTT, and the RV'_{Vi1} using a one-way hash function as shown in figure 99. If the computed value is equal to the received value, the VOCS accepts the authentication request; otherwise, it will drop it.

VX3' = H (V_{IDR}|| V_{CS1}'|| TS_{Vi1}' || RV_{Vi1}'|| OTT)
If VX_3 = VX_3' Then
The VOCS will accept the authentication request
Else
The VOCS will drop the connection request

Figure 99: Evaluating the Vi authentication request message

Once the VOCS accepts the message, it generates a timestamp and a random value RV_{CS1}. Also, the VOCS generates a session ID. The session ID aims to distinguish

one session from the rest of the running sessions simultaneously. Furthermore, the VOCS will retrieve a PUF CRP from the PUF cloud to authenticating the vehicle. The VOCS prepares and sends an authentication response to the Vi. The preparation process of the authentication response message includes the following steps:

1. VOCS generates a new TS_{CS1} to avoid replay attacks
2. VOCS calculates CSY_1
    a. $CSY_1 = H (V_{IDR} || V_{CS1'} || OTT)$
3. VOCS selects a random parameter RV_{CS1} and then calculates CSY_2
    a. $CSY_2 = H(CSY_1) \oplus RV_{CS1}$
4. VOCS generates a session ID(S_{ID})
5. $CV = C \oplus H (CSY_1 || RV_{CS1)}$
6. VOCS calculates CSY_3
7. $CSY_3 = H (VOCS_{IDR} || CSY_1 || TS_{CS1} || RV_{Vi1} || C || V_{CS1} || RV_{CS1})$
8. VOCS sends an authentication response message

The connection response includes $VOCS_{IDA,}$ session ID(S_{ID}), TS_{CS1}, CSY_2, CV, and CSY_3, as shown in 6.7.

$$VOCS \longrightarrow Vi \text{ Auth-Res } (VOCS_{IDA}, S_{ID,} TS_{Vi1,} CSY_2, CV, CSY_3) \quad (6.7)$$

6.6.3.2.1.3 Step 3: VOCS authentication

Once Vi receives the authentication response, it completes the following steps:

1. Vi checks the validity of the received timestamp $|TS'_{Rec} - TS'_{CS1}| < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.
2. Vi calculates $CSY_1' = H(V_{IDR} \| V_{CS1'} \| OTT)$
3. Vi calculates the $RV'_{CS1} = H(CSY_1) \oplus CSY_2$
4. Vi calculates the $C' = CV' \oplus H(RV'_{CS1} \| CSY_1')$
5. As presented in figure 100, Vi calculates CSY_3' using a one-way hash function and compares the generated CSY_3' with the received CSY_3. If the two values are the same, Vi will authenticate VOCS; otherwise, it will drop it.

$CSY3' = H(VOCS_{DR} \| CSY_2' \| TS_{CS1}' \| RV_{Vi1}' \| C' \| V_{CS1}' \| RV_{CS1}')$
If $CSY_3' = CSY_3$ Then
Then Vi will accept the authentication response and authenticate VOCS
Else
Vi will drop the authentication response

Figure 100: Evaluating the VOCS authentication parameter

Once Vi accepts the received message and authenticates VOCS, it will complete the following steps

1. Vi will input the received challenge C' to its PUF and obtains the response Ri.
    a. $Ri = PUF(C')$
2. Vi will compute the $H(RV'_{CS1} \| RV_{Vi1})$ to generate a new challenge (C^{i+1}). Then Vi will input the C^{i+1} into a PUF function to obtains a new R_{i+1} where:
    a. $C^{i+1} = H(RV'_{CS1} \| RV_{Vi1})$

b. R_{i+1} = PUF (C^{i+1}) to be used in the next authentication session.

3. Vi generates a new TS_{Vi2} to avoid replay attacks
4. Vi calculates VX_4 = H (OTT|| RV_{CS1})
5. Vi selects a random parameter RV_{Vi2} and then calculates VX_5

   a. VX_5= RV_{vi2} ⊕ H (VX_4|| V_{IDR})

6. Vi calculates VX_6

   a. VX_6 = H (VX_4|| RV_{Vi2}) ⊕ R^i

7. Vi calculates VX_7

   a. VX_7 = H (C'|| R') ⊕ Ri+1

8. Vi will generate a new OTT(OTTnew)

   OTTnew = H (OTT|| RV_{Vi2}|| C)

9. Vi will generate a new Alias ID (V_{IDAnew})

   a. V_{IDAnew} = H (V_{IDA} || C|| OTT)

10. Vi calculates VX_8

    a. VX_8 = H (VX_4 || RV_{Vi2} || TS_{Vi2} || Ri || C^{i+1} || R^{i+1} || OTTnew || V_{IDAnew})

11. Vi sends the message to the VOCS

The message includes $V_{IDA,}$ TS'$_{Vi2}$, VX_5, VX_6, VX_7, and VX_8 as shown in 6.8.

$$Vi \longrightarrow VOCS\ (V_{IDA,}\ TS_{Vi2},\ VX_{5,}\ VX_{6,}\ VX_{7,}\ VX_8) \quad (6.8)$$

6.6.3.2.1.4 Step 4: Vi authentication

Once VOCS receives the message, it completes the following steps:

1. VOCS checks the validity of the received timestamp $|TS'_{Rec} - TS'_{Vi2}| < \Delta T$. TS'_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message is dropped if the difference between the timestamp and the time when the message is received is higher than the expected maximum transmission delay.
2. VOCS calculates $VX_4' = H(RV_{CS1} \| OTT)$
3. VOCS calculates the $RV'_{VI2} = VX_5' \oplus H(VX_4' \| V_{IDR})$
4. VOCS computes the $R_i' = VX_6' \oplus H(RV_{Vi2} \| VX_4')$
5. VOCS computes $C^{i+1} = H(RV'_{CS1} \| RV_{Vi1})$
6. VOCS computes the $R^{i+1'} = VX_7' \oplus H(C' \| R_i')$
7. VOCS Computes $OTTnew' = H(OTT \| RV_{Vi2} \| C)$
8. VOCS computes $V_{IDAnew'} = H(V_{IDA} \| C \| OTT)$
9. VOCS compures $VX_8' = H(VX_4' \| RV_{Vi2'} \| TS_{Vi2} \| Ri' \| C^{i+'1} \| R^{i+1'} \| OTTnew' \| V_{IDAnew'})$

As presented in figure 101, VOCS calculates VX_8' using a one-way hash function and compares the generated VX_8' with the received VX_8. If the two values are the same, VOCS will authenticate Vi; otherwise, it will drop it.

$VX_8' = VX_8 = H(VX_4' \| RV_{Vi2'} \| TS_{Vi2} \| Ri' \| C^{i+'1} \| R^{i+1'} \| OTTnew' \| V_{IDAnew'})$
If $VX_8' = VX_8$
Then
Then VOCS will authenticate the Vi.
Else
VOCS will drop the connection response

Figure 101: Evaluate the Vi authentication parameter

After authenticating the Vi, the VOCS will store the new CRP, OTTnew, and V_{IDAnew} in its database for the next authentication session.

6.6.3.2.1.5 Key generation

Once the mutual authentication is completed, the two sides will generate a shared secret session key (*ssk*). The two sides will generate the *ssk* locally by combining the PUF responses and the generated random values. The uniqueness of the PUF responses ensures the uniqueness of the generated *ssk*.

$$ssk = H\ (R||\ (RV'_{VI1}||\ \ RV'_{VI2}\ ||RV_{CS1}))$$

6.6.3.2.2 Phase 2: Driver- Owner Cloud Server (VOCS) authentication

This sub-phase presents a mutual authentication process between a Driver (Di) and a vehicle owner cloud server (VOCS). The two devices will complete mutual authentication. The mutual authentication between Di and VOCS should be completed after completing the mutual authentication between the vehicle and the VOCS. The authentication between Di and VOCS will be completed through the vehicle, where it will act as a relay between the two ends. All the exchanged messages will be encrypted by the *ssk* between the vehicle and the VOCS. Also, the vehicle OTT and Ri will be used to authenticate the driver to the VOCS. Using some of the vehicle parameters ensures that the driver parameters are collected and sent through the authenticated vehicle.

As presented in figures 102 and 103, this phase starts when Vi prepares the authentication request message on behalf of Di that will be sent to the VOCS. The steps involved in this process are listed below.

6.6.3.2.2.1 Step 1: Authentication request

1. Di will insert his/her username (Ui) and the password (PUi).
2. Di will use the vehicle's fingerprint scanner to capture fingerprint biometric characteristics (FP)
3. The Ui, PUi, Su, and the FP will send to the vehicle's OBU through a secured channel

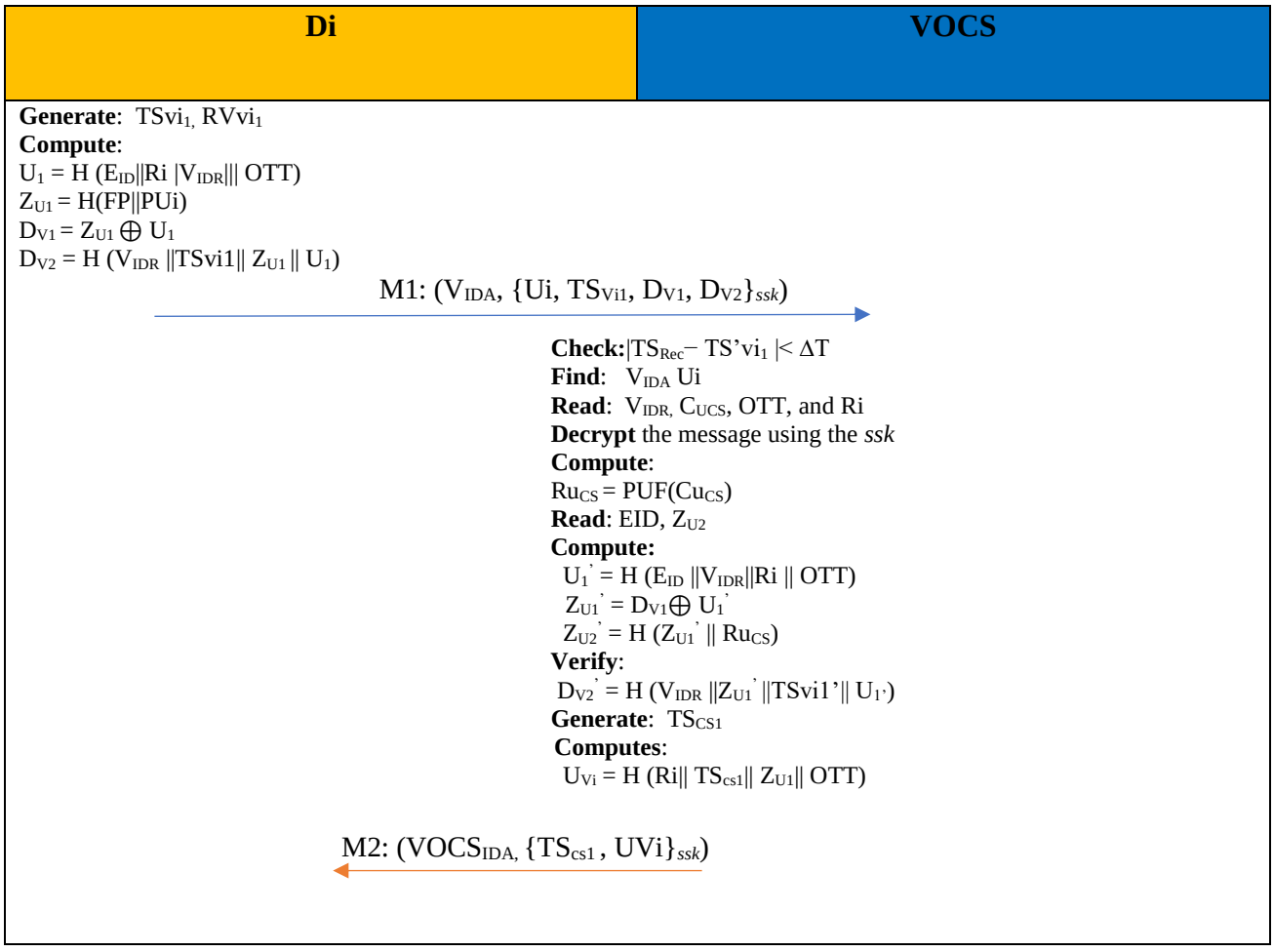


Figure 102: Di-VOCS mutual authentication process

Algorithm 7_Protocol 2_Phase 2: The mutual authentication between D and the VOCS

Input:
V with real identity (V_{IDR}), alias identity (V_{IDA}), VOCS real identity ($VOCS_{IDR}$), VOCS alias identity ($VOCS_{IDR}$), one-time-token (OTT), PUF response (Ri), and Authentication parameters (V_{CS1}, V_{CS2})
VOCS with with real identity (V_{IDR}), alias identity (V_{IDA}), VOCS real identity ($VOCS_{IDR}$), VOCS alias identity ($VOCS_{IDR}$), the vehicle type ID(VT_{ID}) Master secret key (MSKcs), PUF challenge (C_{CS}), and one time token (OTT)
Output:
Mutual authentication between Driver (Di) and the VOCS
Begin

1. D inserts the Employee ID(E_{ID}), username (Ui), password (PUi) and fingerprint (FP). V generates a timestamp TS_{vi1} and random nonce $RV_{vi1.}$ V computes computes $U_1 = H(E_{ID}||Ri\ |V_{IDR}||\ OTT)$, $Z_{U1} = H(FP||PUi)$, $D_{V1} = Z_{U1} \oplus U_1$, $D_{V2} = H(V_{IDR}\ ||TSvi1||\ Z_{U1}\ ||\ U_1)$
2. V sends V_{IDA}, $\{Ui, TS_{Vi1}, D_{V1}, D_{V2}\}_{ssk}$ to VOCS encrypted by the *ssk* between V and VOCS.
3. **If** (the VOCS finds V_{IDA} in its repository)
4. **then**
5. VOCS decrypts the message and retrieves the Ui. Then VOCS will retrieve the PUF challenge that corresponds to the Ui and feed it to the PUF function. VOCS uses the PUF response to decrypt the driverer's data, including E_{ID}, $Z_{U2,}$ and $BB_{Ui,}$ that are stored on its database.
6. The VOCS calculates $U_1^{'}$ and retrieves Z_{U1} from xoring U_1 with D_{V1}. Finally, VOCS calculates $D_{V2} = H(V_{IDR}\ ||TSvi1||\ Z_{U1}\ ||\ U_1)$
7. **If** (the calculated hash message in step 6 matches the hash message that was sent In step 2)
8. **then**
9. **The authenticity of the D is verified.**
10. VOCS communicates generates a timestamp TS_{CS1} and computes $U_{Vi} = H(Ri||\ TS_{cs1}||Z_{U1}||OTT)$
11. VOCS sends ($VOCS_{IDA,}$ $\{TS_{cs1}, UVi\}_{ssk}$) message encrypted by *ssk* to Vi
12. **else**
13. Go to step 21.
14. **end if**
15. **else**
16. Go to step 21.
17. **end if**
18. V verifies the time stamp, decrypts the message using *ssk* and computes $UVi^{'} = = H(Ri||\ TS_{cs1}||\ Z_{U1}||\ OTT)$
19. **If** (the calculated hash message in step 18 matches the hash message that was sent in step 11) **then**

The authenticity of the VOCS is verified

20. **else**
21. **Stop** (terminates the connection);
End

Figure 103: Mutual Authentication between D and VOCS

The Vi will complete the following steps:

1. Vi generates a new TS_{Vi1} to avoid replay attacks
2. Vi computes U_1
    a. $U_1 = H (E_{ID}||Ri |V_{IDR}||| OTT)$
3. Vi computes Z_{U1}
    a. $Z_{U1} = H(FP||PUi)$
4. Vi computes D_{V1}
    a. $D_{V1} = Z_{U1} \oplus U_1$
5. Vi computes D_{V2}
    a. $D_{V2} = H (V_{IDR} ||TSvi1|| Z_{U1} || U_1)$

1. Vi sends an authentication request message encrypted by the *ssk* between Vi and VOCS.

The authentication request includes $V_{IDA,}$ TS_{Vi1}, D_{V1}, and D_{V2} as shown in 6.9.

$$Vi \longrightarrow \text{VOCS Auth-Req } (V_{IDA}, \{Ui, TS_{Vi1}, D_{V1}, D_{V2}\}_{ssk}) \quad (6.9)$$

6.6.3.2.2.2 Step 2: VOCS response

Once the VOCS receives the authentication request, it will complete the following steps:

1. VOCS decrypts the message using *ssk*

2. VOCS retrieves from its database the V_{IDR} that corresponds to its V_{IDA}. If the VOCS did not find the V_{IDA} in its database, it will ignore and drop the authentication request.
3. VOCS checks the validity of the received timestamp $|TS_{Rec} - TS'_{Vi1}| < \Delta T$. TS_{Rec} is the time when the message is received, and ΔT is the maximum transmission delay. The message will be dropped if the difference between the timestamp and the time when the message is received higher than the expected maximum transmission delay.
4. Once the VOCS finds the V_{IDR}, OTT and Ri of Vi
5. VOCS retrieves its PUF challenge (Cu_{CS}) and computes the corresponding PUF response (Ru_{CS}) uses the R_{CS} to decrypt and retrieve E_{ID}, Z_{U2}
    a. $Ru_{CS} = PUF((Cu_{CS})$
6. VOCS computes $U_1' = H(E_{ID} || V_{IDR} || Ri || OTT)$
7. VOCS retrieves Z_{U1}'
    a. $Z_{U1}' = D_{V1} \oplus U_1'$
8. VOCS computes $Z_{U2}' = H(Z_{U1}' || Ru_{CS})$ and compares it with the stores Z_{U2}
9. VOCS calculates D_{V2}
    b. $D_{V2}' = H(V_{IDR} || Z_{U1}' || TSvi1' || U_{1'})$

If the computed value is equal to the received value, as shown in figure 104, the VOCS accepts the authentication request and authenticates the driver; otherwise, it will drop it.

$D_{V2}^{'} = H (V_{IDR} \| TS_{vi1}^{'} \| Z_{U1} \| U_1^{'})$
If $D_{V2}^{'} = D_{V2}$ Then
The VOCS will authenticate Di
Else
The VOCS will drop the authentication request

Figure 104: Evaluating the driver authentication parameter

Once the VOCS accepts the message, it generates a timestamp. The VOCS prepares and sends an authentication response to the Vi. The preparation process of the authentication response message includes the following steps:

1. VOCS generates a new TS_{CS1} to avoid replay attacks
2. VOCS generates UVi
3. $U_{Vi} = H (Ri \| TS_{cs1} \| Z_{U1} \| OTT)$
4. VOCS sends an authentication response message

The authentication response includes $VOCS_{IDA,}$ U_{Vi}, and TS_{CS1} encrypted by the *ssk,* as shown in 6.10.

$$VOCS \longrightarrow Vi \quad Auth\text{-}Res (VOCS_{IDA}, TS_{cs1,} UVi\}_{ssk}) \qquad (6.10)$$

Once the Vi receives the authentication response message, it will verify the response, Vi calculates UVi'. If the computed value is equal to the received value, as shown in figure 105, the Vi accepts the authentication response and ensures the authenticity of the VOCS.

$U_{Vi}^{'} = H (Ri \| TS_{cs1} \| Z_{U1} \| OTT)$
If $U_{Vi} = U_{Vi}^{'}$ Then
The VOCS will be verified
Else
The Vi will drop the authentication response

Figure 105: Evaluate the VOCS authentication response

Once the authentication phase is completed, the process of continuous biometric authentication will start. The vehicle will periodically collect the drivers' BB_{Ui} using the built-in sensor in the steering wheels and sends the driver's BB_{Ui} as part of its data. Every time the VOCS receives the driver's BB_{Ui}, it will compare it against the stored BPi template to ensure that he/she is the same driver, monitor their health condition, and monitor the driver's behaviors.

6.7 Evaluation Process

The following section presents a detailed security and performance analysis of the proposed scheme and compares the analysis results with other related schemes presented in the literature.

6.7.1 Security validation of the proposed scheme

In this section, we conduct both a formal and informal security analysis. The formal evaluation uses two different approaches: First, we validate the proposed scheme using the AVISPA tool to ensure that the proposed scheme is secure against active and passive attacks such as replay attacks and man-in-the-middle attacks. Second, we then perform the logical verification using BAN logic to confirm that the authenticated participants share the secret parameters securely in the proposed protocols. After completing the formal evaluation, we examine the proposed protocols against the well-known attacks and prove that the proposed protocols satisfy the main security properties.

6.7.1.1 Formal security evaluation

6.7.1.1.1 Simulation-based security verification using AVISPA

The proposed protocols are simulated and tested using the AVISPA software, a widely accepted tool for automatically validating the schemes' security features. The messages involved in the cryptography and security of the authentication process are taken into consideration.

6.7.1.1.1.1 Protocol 1: V-TA mutual authentication

6.7.1.1.1.1.1 Simulation overview

The abstract notations used to describe the authentication scheme and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 27.

Table 27: Abstract notation and AVISPA HLPSL scripting variables/functions for protocol specification

Notation	**Description**
TA	Trusted Authority
V	Vehicle
V_{IDR}	VIDR
V_{IDA}	VIDA
TA_{IDA}	TAIDA
VT_{A1}	VTA
RV_{vi1}	Na
OTPvi	OTP

TS_{vi1}	TSoneV
RV_{TA1}	Nb
C	Cone
CHXx	CHX
TS_{TA1}	TSoneTA
Ri	Rone
CHX_{x+1}	CHXnew
TS_{vi2}	TStwoV
RV_{vi2}	Naa
TS_{TA2}	TStwoTA
RV_{TA2}	Nbb
OTPnew	NOTP
V_{IDAnew}	NnewIDA
$\oplus$	XOR
H	H

The main goals of the simulation are as follows:

- **Goal 1**: The secrecy_of secNa represents that RVvi1 is kept secret to (V, TA) only.

- **Goal 2**: The secrecy_of secNaa represents that RVvi2 is kept secret to (V, TA) only.
- **Goal 3**: The secrecy_of secNb represents that RV_{TA1} is kept secret to ((V, TA) only.
- **Goal 4**: The secrecy_of secVTA represents that V_{TA1} is kept secret to (V, TA) only.
- **Goal 5**: The secrecy_of secCHX represents that CHXx is kept secret to ((V, TA) only.
- **Goal 6**: The secrecy_of secRone represents that Ri is kept secret to (V, TA) only.
- **Goal 7**: The secrecy_of $secC_{one}$ represents that C' is kept secret to (V, TA) only.
- **Goal 8**: The secrecy_of secNOTP represents that OTPnew is kept secret to (V, TA) only.
- **Goal 9**: The secrecy_of secNnewIDA represents that V_{IDAnew} is kept secret to (V, TA) only.
- **Goal 10**: The secrecy_of secVIDR represents that the V_{IDR} is kept secret to (V, TA) only.
- **Authentication Property 1**: The authentication_on nb represents that TA generates RV_{TA1}. If V securely receives RV_{TA1} through a message, it authenticates TA.

- **Authentication Property 2**: The authentication_on na represents that V generates RV_{vi1}. If TA securely receives RV_{vi1} through a message, it authenticates V.

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with their message exchanges. There are four defined roles: that are: (1) role_V that is played by the vehicle; (2) role _TA that is played by the trusted authority; (3) session, where the session role and all its declarations are defined, (4) and environment, which instantiates all agents, variables, and functions.

As presented in Figure C_1 in Appendix C, the role of the vehicle is played by V. V is aware of the Vi and TA agents in the protocol, and its alias identity and the TA alias identity. Also, it is aware of its own real identity and the CHX that should be kept secured. Furthermore, V is aware of the OTP that is generated by the TA, the VTA authentication parameter, the hash function H (·) and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model. The keyword 'Played_by V' denotes that the role of the vehicle is played by agent V. All employed local variables in this role are defined under the local section.

At the first state "state 2," V receives a start message "Rcv(start)" as a signal to begin the protocol run. V generates new random values (TSoneV and Na) and computes Xone and Xtwo. The computation process of the Xone and Xtwo follows the presented protocol description. V sends VIDA, TSoneV, Xone, and Xthree to the TA. At the second transition, "State 4", V receives the (TAIDA, SIDNIG, TSoneTA, Ytwo, CX and Ythree) message from the TA. The computation of Ytwo, CX, and Ythree follows the description

of our protocol. At this transition, if Ythree appears as expected by V and V correctly verified nb, V authenticates TA. Then V generates a timestamp (TStwoV), new random value (Naa), and computes CHxnew, Xthree, and Xfour. The computation process of Xthree, Xfour, and CHxnew follows the presented protocol description. V sends VIDA, TStwoV, Xthree, and Xfour to TA. At the third transition, “State 6”, V received the (TAIDA, SIDNIG, TStwoTA, Yfive, and Ysix) from the TA. V calculates and verifies the received new OTPnew, and NnewIDA by following the presented protocol's description. After completing the verification process, Vstores its new alias ID and the new OTP on its own memory for the next authentication session. The “end role” at the end of the V role denotes the end of the role played by V.

Figure C_2 in Appendix C shows the role of the trusted authority played by TA. The TA is aware of all agents in the protocol (V and TA), its alias and real identities (TAIDA, TAIDR), the Alias and real identities (VIDA, VIDR) of the vehicle. TA knows the OTP, the CHX, the hash function H (·), and the send/receive channels Snd/Rcv.

At the first transition, “State 1”, TA receives the (VIDA, TSoneV, Xone, and Xtwo) message which was sent by V. TA uses the alias ID of V to retrieve its real ID, its PUF challenge, the CHX, and OTP. TA uses the calculated VTA value to extract the V's random generated value. Furthermore, TA uses V’s real identity, TSoneV, Na, and the VTA to compute and verify the received Xtwo’. The computation and extraction of Na, Xone, and Xtwo follow the presented scheme's description. Once the TA verifies Xtwo, it generates a new value Nb and a TSoneT and computes Yone, Ytwo, CX and Ythree. Then the TA sends to V (TAIDA, SIDNID, TSoneT, Ytwo, CX,Ythree). The computation for Yone, Ytwo, CX, Ythree follows the description of the introduced

scheme. t the second transition, “State 3”, TA received the (VIDA, SIDNIG, TStwoV, Xthree, and Xfour) from the V. The computation of the Xthree and Xfour followed the description of the presented protocol. At this transition, if Xfour is proved to be valid by TA, TA will authenticate V. Consequently, the TA will generate a new alias ID for V and a new OTP. The computations of NnewIDA and NOPT will follow the description of the presented protocol. The TA will then send to V (TAIDA, SIDNIG, TStwoT, Yfive,Ysix). The “end role” at the end of the TA role denotes the end of the role played by TA.

Figure C_3 in Appendix C shows the session role where all the two agents’ roles are invoked, and all the session parameters are defined. Both the TA and V roles are invoked with TA and V as agents, VIDA, VIDR, TAIDA, OTP, VTA, and CHX are predefined as constants, H as the hash function, and SND1 and RCV1 as well as SND2 and RCV2 as the send and receive channels for V and TA. The “end role” at the end of the session role indicates the end of this role.

Figure C_4 in Appendix C shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants, v, and ta are instantiated as agents representing agents V and TA. The constants vida, vidr ,taida , otp,vta, and chx instantiates VIDA, VIDR, TAIDA, OTP,VTA, and CHX. The function h instantiates the hash function H. The protocol identifiers are secVIDR, secOTP, secNa, secNb,secNaa, $secC_{one}$, secRone, secNOTP,secVIDAnew, secCHXnew, secVTA, na, and nb are also instantiated and defined. In the intruder knowledge section, all relevant values that the intruder is assumed to know before the execution are provided. The attacker is assumed to know v and TA. He/ she is also assumed to know the hash h. The session is instantiated with v, ta, vida, vidr, taida, otp, chx, vta, and h instances in the

composition section. The “end role” at the end of the environment role denotes the end of this role.

Figure C_5 in Appendix C shows the simulation goals which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator is dictated to check the secrecy VIDR, OTP,Na,Nb, VTA,Cone,Rone,CHXnew,Naa,NOTP, and VnewIDA at different states using secrecy_of secVIDR’, ‘secrecy_of secOTP’, ‘secrecy_of secNa’, ‘secrecy_of secNb, ‘secrecy_of secVTA’, ‘secrecy_of secC_{one}’, ‘secrecy_of secRone, secrecy_of secCHXnew, ‘secrecy_of secNaa, ‘secrecy_of secNOTP’, and ‘secrecy_of secNnewIDA’. The authentication is checked using ‘authentication_on na’ and ‘authentication_on nb’. The “end role” at the end of the goal section denotes the end of this role.

6.7.1.1.1.1.2 Simulation results

The SPAN protocol simulation’s MSC corresponding to our HLPSL specification is shown in Figure. 106.

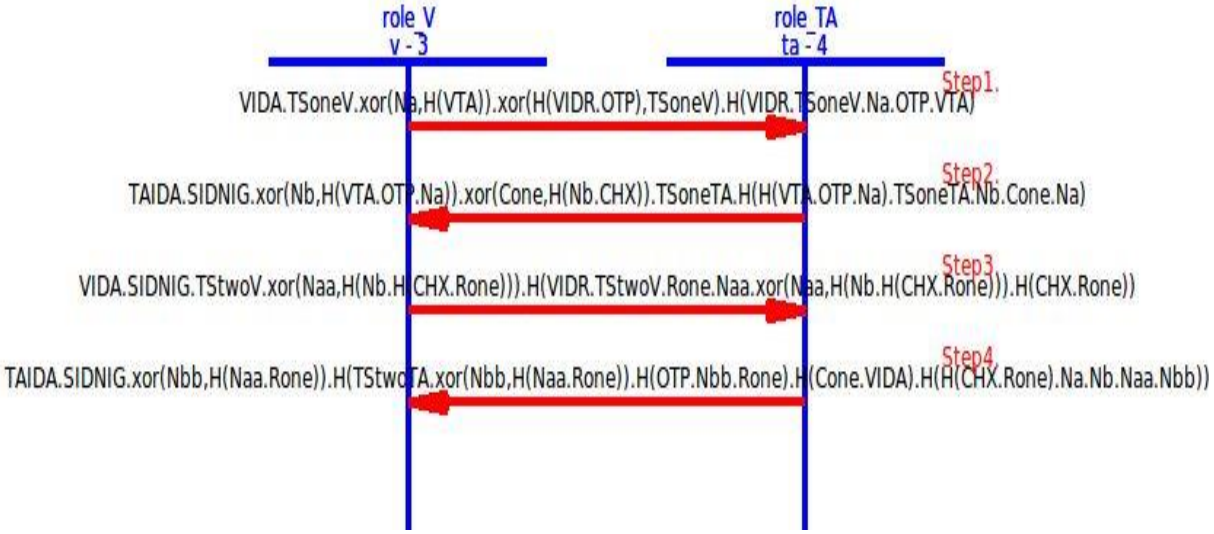


Figure 106: Snapshot of the protocol simulation in AVISPA

In AVISPA tool, the security properties such as authentication, integrity, and secrecy are specified in a separate section. Therefore, when SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN will generate the attack trace if an attack is found and considers the protocol unsafe. The presented protocol's simulation results are achieved by the OFMC back-end checker and the CL-AtSe back-end checker. Figure 107 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure 108 presents the OFMC back-end checker report shows that the protocol is SAFE, thus meeting the defined security goals. In summary, we can conclude that the proposed protocol is secure.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/V1_TA_trial.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS
 Analysed  : 1 states
 Reachable  : 1 states
 Translation: 0.06 seconds
 Computation: 0.00 seconds
```

Figure 107: CL-AtSe Summary Report

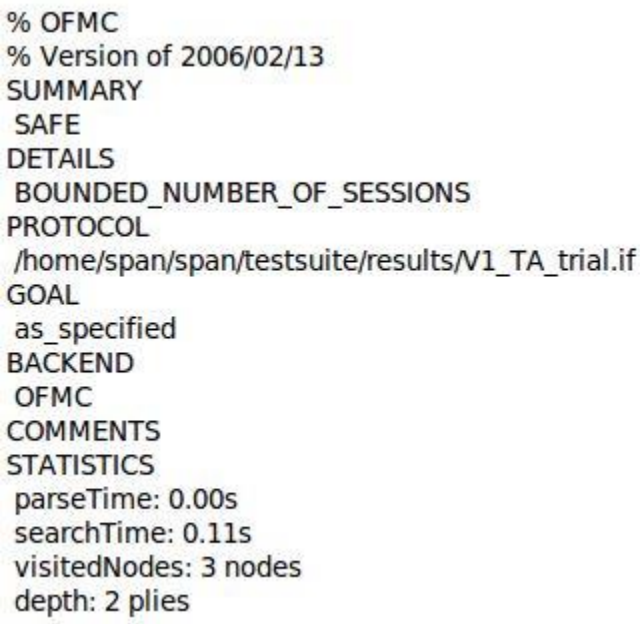


Figure 108: OFMC summary report

6.7.1.1.1.2 Protocol 2: Vehicle-Driver-VOCS authentication

6.7.1.1.1.2.1 Phase 1: Vehicle -VOCS mutual authentication

6.7.1.1.1.2.1.1 Simulation overview

The abstract notations used to describe the authentication process and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 28.

Table 28: Abstract notation and AVISPA HLPSL scripting variables/functions for protocol specification

Notation	**Description**
VOCS	CS
Vi	Vehicle
V_{IDR}	VIDR

$VOCS_{IDR}$	CSIDR
$VOCS_{IDA}$	CSIDA
MSK	MSK
OTT	OTT
VT_{ID}	TID
RV_{vi1}	Na
TS_{vi1}	TSoneV
V_{CS1}	VCS1
V_{CS2}	VCS2
TS_{CS1}	TSoneCS
RV_{CS1}	Nb
C	Cone
CV	CV
SIDNIG	S_{ID}
TS_{vi2}	TStwoV
RV_{vi2}	Naa
Ri	Rone

C^{i+1}	Ctwo
R^{i+1}	Rtwo
OTTnew	NOTT
V_{IDAnew}	VnewIDA
$\oplus$	XOR
H	H

The main goals of the simulation are as follows:

1. **Goal 1**: The secrecy_of secVIDR represents that the VIDR is permanently kept secret, known to only (V, VOCS)
2. **Goal 2**: The secrecy_of secOTT represents that OTT is kept secret to (V, CS) only.
3. **Goal 3**: The secrecy_of secVCS1 represents that V_{CS1} is kept secret to (V, CS) only.
4. **Goal 4**: The secrecy_of secVCS2 represents that V_{CS2} is kept secret to (V, CS) only.
5. **Goal 5**: The secrecy_of secNa represents that RVvi1 is kept secret to (V, CS) only.

6. **Goal 6**: The secrecy_of secNb represents that RV_{TA1} is kept secret to (V, CS) only.
7. **Goal 7**: The secrecy_of $secC_{one}$ represents that C is kept secret to ((V, CS) only.
8. **Goal 8**: The secrecy_of secRone represents that Ri is kept secret to ((V, CS) only
9. **Goal 9**: The secrecy_of secRtwo represents that Ri+1 is kept secret to ((V, CS) only.
10. **Goal 10**: The secrecy_of secNaa represents that RVvi2 is kept secret to (V, CS) only.
11. **Goal 11**: The secrecy_of secNOTT represents that OTTnew is kept secret to (V, CS) only.
12. **Goal 12**: The secrecy_of secVnewIDA represents that the secret key secVIDAnew is permanently kept secret, known to only (V, CS).
13. **Authentication Property 1**: The authentication_on Nb represents that CS generates Nb. If V securely receives Nb through a message, it authenticates CS.
14. **Authentication Property 2**: The authentication_on Na represents that V generates Na. If CS securely receives Na through a message, it authenticates V.

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with

their message exchanges. There are four defined roles: that are: (1) role_V that is played by the vehicle; (2) role _CS that is played by the vehicle owner cloud server; (3) session, where the session role and all its declarations are defined, (4) and environment, which instantiates all agents, variables, and functions.

As presented in Figure C_6 in Appendix C, the role of the vehicle is played by V. V is aware of the V and CS agents in the protocol, and its alias identity and the CS alias identity. Also, it is aware of its own real identity VIDR real identity that should be kept secured. Furthermore, V is aware of the VCS1, VCS2, OTT, the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model. The keyword 'Played_by V' denotes that the role of the vehicle is played by agent V. All employed local variables in this role are defined under the local section.

At the first state "state 2," V receives a start message "Rcv(start)" as a signal to begin the protocol run. V generates new random values (TSoneV and Na and computes VXone and VXtwo. The computation process of the VXone and VXtwo follows the presented protocol description. V sends VIDA, TSoneV, VXone, and VXtwo to the CS.

At the second transition, "State 4", V receives the CSIDA, SIDNIG, TSoneCS, Ytwo, CV and Ythree) message from the CS The computation of CSYtwo, CV, and CSYthree follows the description of our protocol. At this transition, if the CSYthree appears as expected by V and V correctly verified Nb, then V authenticates CS. Then V generates new random values TStwoV and Naa. Also, V generates a new challenge (Ctwo) and calculates its response (Rtwo), its new alias ID(VnewIDA), and a new OTT(NOTT). The computations of Ctwo, Rtwo,VnewIDA, NOTT, VXfour,Vxfive,

Vxsix, and Vxseven follow the description of the presented protocol. V sends (VIDA, TStwoV, VXfour, VXfive, Vxsix, and Vxseven) to CS. The “end role” at the end of the V role denotes the end of the role played by V.

Figure C_7 in Appendix C shows the role of the Vehicle owner cloud server played by CS. The CS is aware of all agents in the protocol (V and CS), its alias and real identities (CSIDA, CSIDR), the Alias and real identities (VIDA, VIDR) of the vehicle. CS knows the OTT, the TID, the hash function H (·), and the send/receive channels Snd/Rcv.

At the first transition, “State 1”, CS receives the (VIDA, TSoneV, VXone, and VXtwo) message, which V. CS sent uses the alias ID of V to retrieve its real ID, TID, and OTT and its own PUF challenge. CS uses the calculated VSC1 and VCS2 to extract the V's random generated value. Also, CS uses V’s real identity, TSoneV, Na, OTT, and the VCS1to compute and verify the received VXtwo’. The computation and extraction of Na, TSoneV, and VXtwo follow the presented scheme's description. Once the CS verifies VXtwo’, it generates a fresh value Nb and a TSoneCS and computes CSYone, CSYtwo, CV, and CSYthree. Then the CS sends to V (CSIDA, SIDNID, TSoneCS, CSYtwo, CV,CSYthree). The computation for CSYone, CSYtwo, CV, CSYthree follows the description of the introduced scheme.

At the second transition, “State 3”, CS received the (VIDA, SIDING, TStwoV, VXfour, VXfive, VXsix, and VXseven) message from the V. The computation of the VXfour, VXfive, VXsix, and VXseven follows the description of the presented protocol. At this transition, CS calculates and verifies the received new token (NOTT), New Alias ID (VnewIDA), new challenge (Ctwo), and new response (Rtwo) by following the

presented protocol's description. Suppose VXseven is proved to be valid by CS. In that case, CS authenticates V. The “end role” at the end of the CS role denotes the end of the role played by CS.

Figure C_8 in Appendix C shows the session role where the two agents’ roles are invoked and defined all the session parameters. Both the CS and V roles are invoked with CS and V as agents, VIDA, VIDR, CSIDA, VCS1, VCS2, OTT, and TID are predefined as constants, H as the hash function, and SND1 and RCV1 as well as SND2 and RCV2 as the send and receive channels for V and CS. The “end role” at the end of the session role indicates the end of this role.

Figure C_9 in Appendix C shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants, v, and cs are instantiated as agents representing agents V and CS. The constants vida, vidr, csida ,vcs1,vcs2, otp, and tid instantiates VIDA, VIDR, CSIDA, VCS1,VCS2,OTT, and TID, respectively. The function h instantiates the hash function H. The protocol identifiers are secVIDR, , secOTT, secNa, secNb,secNaa, secCone, secRone, secRtwo, secNaa, secNOTT,secVCS1, secVCS1, secVnewIDA, na and nb are also instantiated and defined. In the intruder knowledge section, all relevant values that the intruder is assumed to know before the execution are provided. The attacker is assumed to know v and cs. He/ she is also assumed to know the hash h. The session is instantiated with v, cs, vida, vidr, csida, vcs1, vcs2, ott, tid, and h instances in the composition section. The “end role” at the end of the environment role denotes the end of this role.

Figure C_10 in Appendix C shows the simulation goals, which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator

is dictated to check the secrecy VIDR, OTT,Na,VCS1,VCS2, Nb, Cone,Rone,Rtwo,Naa,NOTT, and VnewIDA at different states using secrecy_of secVIDR', 'secrecy_of secOTT', 'secrecy_of secNa', 'secrecy_of secVCS1', 'secrecy_of secVCS2', 'secrecy_of secNb, 'secrecy_of secCone', 'secrecy_of secRone, secrecy_of secRtwo, 'secrecy_of secNaa,'secrecy_of secNOTT', and 'secrecy_of secVnewIDA' . The authentication is checked using 'authentication_on na' and 'authentication_on nb'. The "end role" at the end of the goal section denotes the end of this role.

6.7.1.1.1.2.1.2 Simulation results

The SPAN protocol simulation's MSC corresponding to our HLPSL specification is shown in figure 109.

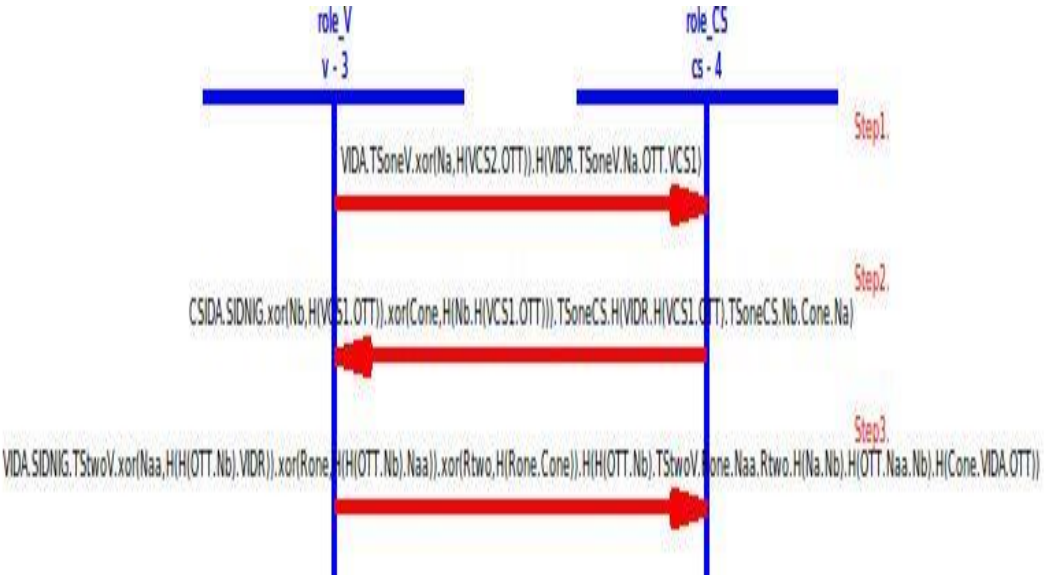


Figure 109: Snapshot of the Protocol Simulation in AVISPA

In AVISPA tool, the security properties such as authentication, integrity, and secrecy are specified in a separate section. Therefore, when SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN will generate the attack trace if an

attack is found, and it will consider the protocol unsafe. The presented protocol's simulation results are achieved by the OFMC back-end checker and the CL-AtSe back-end checker. Figure 110 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure. 111 presents the OFMC back-end checker report shows that the protocol is SAFE, thus meeting the defined security goals. In summary, we can conclude that the proposed protocol is secure.

```
SUMMARY
SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/V_VOCS_trial.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS

 Analysed   : 1 states
 Reachable  : 1 states
 Translation: 0.05 seconds
 Computation: 0.00 seconds
```

Figure 110 : CL-AtSe summary report

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/span/span/testsuite/results/V_VOCS_trial.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.11s
  visitedNodes: 3 nodes
  depth: 2 plies
```

Figure 111: OFMC summary report

6.7.1.1.1.2.2 Phase 2: Driver-VOCS Authentication

6.7.1.1.1.2.2.1 Simulation Overview

After completing the mutual authentication between the vehicle and the VOCS, the driver's authentication to the VOCS will be started. The vehicle will be used as the communication medium between the driver and the VOCS. The abstract notations used to describe the authentication protocol and the corresponding AVISPA HLPSL scripting variables/functions are presented in table 29.

Table 29: D-VOCS abstract notation and AVISPA HLPSL scripting variables/functions for protocol specification

VOCS	CS

Vi	Vehicle
V_{IDR}	VIDR
$VOCS_{IDR}$	CSIDR
$VOCS_{IDA}$	CSIDA
OTT	OTT
UI	Ui
PUi	PUI
Ssk	SK
VT_{ID}	TID
TS_{vi1}	TSoneV
V_{CS1}	VCS1
V_{CS2}	VCS2
TS_{CS1}	TSoneCS
Ri	Rone
$\oplus$	XOR
H	H

The main goals of the simulation are as follows:

1. **Goal 1**: The secrecy_of secVIDR represents that the VIDR is permanently kept secret, known to only (V, CS)
2. **Goal 2**: The secrecy_of secOTT represents that OTT is kept secret to (V, CS) only
3. **Goal 3**: The secrecy_of secRone represents that Ri is kept secret to ((V, CS) only
4. **Goal 4**: The secrecy_of secSk represents that SKis kept secret to (V, CS) only.
5. **Goal 5**: The secrecy_of secPUI represents that PUI is kept secret to (V, CS) only.
6. **Authentication Property 1**: The authentication_on Uone represents that the user generates Uone. If CS securely receives Uone through a message, it authenticates U.

To achieve the goals mentioned above, we wrote the HLPSL script for the protocol. The entities involved in the communication process are modeled as roles with their message exchanges. There are four defined roles: that are: (1) role_V that is played by the vehicle that represents the user; (2) role _CS that is played by the vehicle owner cloud server; (3) session, where the session role and all its declarations are defined, (4) and environment, which instantiates all agents, variables, and functions.

As presented in figure C_11 in Appendix C, the role of the vehicle is played by V. V is aware of the V and CS agents in the protocol, and its alias identity and the CS alias identity. Also, it is aware of its own real identity VIDR that should be kept secured.

Furthermore, V is aware of the VCS1, VCS2, OTT, UI, Rone, SIDIG, SK, the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model. The keyword 'Played_by V' denotes that the role of the vehicle is played by agent V. All employed local variables in this role are defined under the local section. At the first state "state 2," V receives a start message "Rcv(start)" as a signal to begin the protocol run. V generates a timestamp (TSoneV) and computes Uone, ZUone,Utwo, and Uthree. The computation process of the Uone, ZUone, Utwo, and Uthree follows the presented protocol description. V sends VIDA, Ui, TSoneV, Utwo, and Uthree to the CS. At the second transition, "State 4", V receives the CSIDA, SIDNIG, TSoneCS, and UV) message from the CS. At this transition, if the UV appears as expected by V and V correctly verified UV, this proves that the response came from the real CS, and the user got successfully authenticated to the CS. The "end role" at the end of the V role denotes the end of the role played by V.

Figure C_12 in Appendix C shows the role of the vehicle owner cloud server played by CS. The CS is aware of all agents in the protocol (V and CS), its alias and real identities (CSIDA, CSIDR), the Alias and real identities (VIDA, VIDR) of the vehicle. CS knows the OTT, the TID, UI, Rone, SIDING, SK, the hash function H (·), and the send/receive channels Snd/Rcv. The (dy) notation indicates that the channels are following the Dolev-Yao model. The keyword 'Played_by CS' denotes that agent CS plays the role of the vehicle owner cloud server. All employed local variables in this role are defined under the local section. At the first transition, "State 1", CS receives the (VIDA, TSoneV, UI, Utwo, Uthree) message from V. At this transition, Uthree is proved to be valid by CS, CS authenticates the user. Then, CS generates a fresh TSoneCS and

computes Uv. Then the CS sends to V (CSIDA, SIDNID, {TSoneCS, UV}$_{ssk}$). The computation for Uv, follows the description of the introduced scheme. The “end role” at the end of the CS role denotes the end of the role played by CS.

Figure C_13 in Appendix C shows the session role where the two agents’ roles are invoked and defined all the session parameters. Both the CS and V roles are invoked with CS and V as agents, CSIDA, VIDA, VIDR, CSIDR, VCS1, VCS2, OTT, UI, Rone, SIDNIG, TID, and SK are predefined as constants, H as the hash function, and SND1 and RCV1 as well as SND2 and RCV2 as the send and receive channels for V and CS. The “end role” at the end of the session role indicates the end of this role.

Figure C_14 in Appendix C shows the environment role. In this role, one or more sessions are instantiated. First, all constants are instantiated and defined. The constants, v, and cs are instantiated as agents representing agents V and CS. The constants csida, vida, vidr , csidr ,vcs1,vcs2, ott,ui, rone, siding, tid, and sk instantiates CSIDA,VIDA, VIDR, CSIDR, VCS1,VCS2,OTT,UI,Rone, SIDNIG, TID, and SK, respectively. The function h instantiates the hash function H. The protocol identifiers are secVIDR, secOTT, secRone, secSK, secPUI, and uone are also instantiated and defined. In the intruder knowledge section, all relevant values that the intruder is assumed to know before the execution are provided. The attacker is assumed to know v and cs. He/ she is also assumed to know the hash h. The session is instantiated with v, cs, csida, vida, vidr, csidr, vcs1, vcs2, ott, ui, rone, siding, tid, sk, and h instances in the composition section. The “end role” at the end of the environment role denotes the end of this role.

C_15 in Appendix C shows the simulation goals, which are declared under the “goal” keyword using the protocol identifiers declared as ‘protocol_id’. The simulator is

dictated to check the secrecy VIDR, OTT, Rone, SK, and PUI at different states using secrecy_of secVIDR', 'secrecy_of secOTT', 'secrecy_of secRone', 'secrecy_of secSK', and 'secrecy_of secPUI. The authentication is checked using 'authentication_on uone'. The "end role" at the end of the goal section denotes the end of this role.

6.7.1.1.1.2.2.2 Simulation Results

The SPAN protocol simulation's MSC corresponding to the HLPSL specification is shown in figure 112.

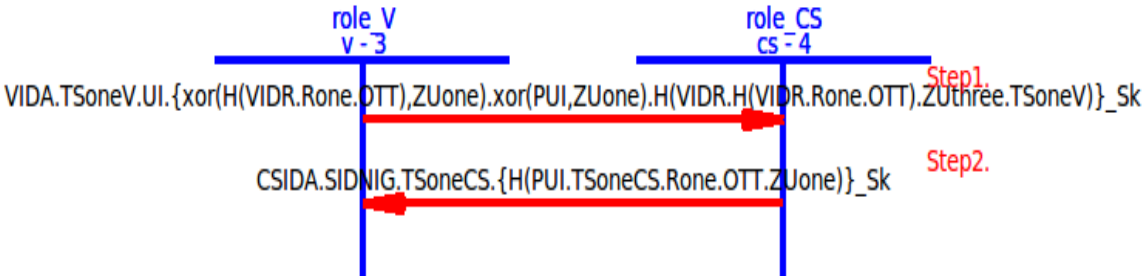


Figure 112: Snapshot of the protocol simulation in AVISPA

In AVISPA tool, the security properties such as authentication, integrity, and secrecy are specified in a separate section. Therefore, when SPAN is executed, it verifies if the protocol satisfies the specified properties. SPAN will generate the attack trace if an attack is found, and it will consider the protocol unsafe. The presented protocol's simulation results are achieved by the OFMC back-end checker and the CL-AtSe back-end checker. Figure 113 shows the CL-AtSe back-end checker report, which guarantees that the protocol is SAFE and satisfies all the specified security goals. Figure. 114 presents the OFMC back-end checker report shows that the protocol is SAFE, thus meeting the defined security goals. In summary, we can conclude that the proposed protocol is secure.

```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/User_Auth_trial.if

GOAL
 As Specified

BACKEND
 CL-AtSe

STATISTICS
 Analysed   : 4 states
 Reachable  : 2 states
 Translation: 0.02 seconds
 Computation: 0.00 seconds
```

Figure 113: CL-AtSe summary report

```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/User_Auth_trial.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
 parseTime: 0.00s
 searchTime: 0.05s
 visitedNodes: 3 nodes
 depth: 2 plies
```

Figure 114: OFMC summary report

6.7.1.1.2 Formal proof based on BAN logic

6.7.1.1.2.1 Protocol 1: V-TA authentication

In this section, we use BAN logic to verify the legitimacy of OTPnew and the session key (*ssk*) that are shared between V and TA that communicate in the presented protocol. To ensure the security of the proposed protocol under BAN logic, it needs to satisfy security goals. The following section defines the main goals of the analysis of the presented authentication scheme.

6.7.1.1.2.1.1 Goals identification

Goal 1: V and TA want to establish a shared secret key (*ssk*) that is believed by each other.

G1_1: TA believes that the V believes that the *ssk* is a securely shared parameter between V and TA.

$$TA| \equiv V| \equiv (V \overset{ssk}{\longleftrightarrow} TA)$$

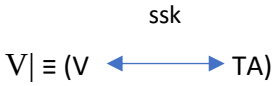
G1_2: TA believes that *ssk* is a securely shared parameter between V and TA.

$$TA| \equiv (V \overset{ssk}{\longleftrightarrow} TA)$$

G1_3: V believes that TA believes that the *ssk* is a securely shared parameter between V and TA.

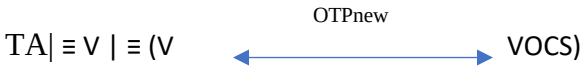
$$V| \equiv TA \mid \equiv (V \overset{ssk}{\longleftrightarrow} TA)$$

G1_4: V believes that the *ssk* is a securely shared parameter between V and TA.



Goal 2: V and TA want to generate a one-time password (OTPnew) that is believed by each other.

G2_1: TA believes that the V believes that the OTPnew is securely shared parameters between V and TA.



G2_2: TA believes that OTPnew is securely shared parameter between V and TA.

G2_3: V believes that the TA believes that the OTPnew is securely shared parameters between V and TA.

G2_4: V believes that OTPnew is securely shared parameter between V and TA.

$$V| \equiv (V \xleftrightarrow{OTPnew} TA)$$

6.7.1.1.2.1.2 Messages idealization

First, we transfer all transmitted messages into idealized form as follows.

M1: $V \longrightarrow TA:(V_{IDR}, TS_{v1}, RV_{v1})_{VTA1}$

M2: $TA \longrightarrow V:(RV_{TA1}, TS_{TA1}, OTP_{Vi}, C)_{Yone}$

M 3: $V \longrightarrow TA:(V_{IDR}, TS_{V2}, RV_{V2,} OTP_{Vi}, Ri)_{CHXi+1}$

M 4: $TA \longrightarrow V:(V_{IDR}, TS_{TA2}, RV_{TA2})_{Yfour}$

6.7.1.1.2.1.3 Main assumptions

The second step is to define some assumptions as initiative promises. The fundamental assumptions of the presented authentication scheme are as follows:

P1: TA believes that V_{TA} is a secure shared parameter between V and TA

$$TA| \equiv (V \xleftrightarrow{VTA} TA)$$

P2: TA believes V believes that V_{TA} is a secure shared parameter between V and TA.

$$TA \mid \equiv V| \equiv (V \xleftrightarrow{VTA} TA)$$

P3: V believes that Yone is a secure shared parameter between V and TA

$$V| \equiv (V \overset{Yone}{\longleftrightarrow} TA)$$

P4: V believes TA believes that Yone is a secure shared parameter between V and TA.

$$V| \equiv TA| \equiv (N \overset{Yonr}{\longleftrightarrow} MG)$$

P5: TA believes that CHX^{i+1} is a secure shared parameter between V and TA

$$TA| \equiv (V \overset{CHX^{i+1}}{\longleftrightarrow} TA)$$

P6: TA believes V believes that CHX^{i+1} is a secure shared parameter between V and TA.

$$TA| \equiv V| \equiv (V \overset{CHX^{i+1}}{\longleftrightarrow} TA)$$

P7: V believes that Yfour is a secure shared parameter between V and TA

$$V| \equiv (V \overset{Yfour}{\longleftrightarrow} TA)$$

P8: V believes TA believes that Yfour is a secure shared parameter between V and TA.

$$V| \equiv TA| \equiv (V \overset{Yfour}{\longleftrightarrow} TA)$$

P9: TA believes RV_{V1} is fresh.

$$TA| \equiv \#(RV_{V1})$$

P10: TA believes that V believes RV_{V1}.

$$TA \mid \equiv V \mid \equiv \ RV_{V1}$$

P11: TA believes that V has jurisdiction over RV_{V1}. That is, V is an authority and believes RV_{V1}.

$$TA \mid \equiv V \ \Rightarrow RV_{V1}$$

P12: TA believes TS_{V1} is fresh.

$$TA \mid \equiv \#(TS_{V1})$$

P13: TA believes RV_{V2} is fresh.

$$TA \mid \equiv \#(RV_{V2})$$

P14: TA believes that V believes RV_{V2}.

$$TA \mid \equiv V \mid \equiv \ RV_{V2}$$

P15: TA believes that V has jurisdiction over RV_{V2}. That is, V is an authority and believes RV_{V2}.

$$TA \mid \equiv V \ \Rightarrow RV_{V2}$$

P16: TA believes TS_{V2} is fresh.

$$TA \mid \equiv \#(TS_{V2})$$

P17: V believes RV_{TA1} is fresh.

$$V \mid \equiv \#(RV_{TA1})$$

P18: V believes that TA believes RV_{TA1}.

$$V \mid \equiv TA \mid \equiv \ RV_{TA1}$$

P19: V believes that TA has jurisdiction over RV_{TA1}. That is, TA is an authority and believes RV_{TA1}.

$$V \mid \equiv TA \ \Rightarrow RV_{TA1}$$

P20: V believes RV_{T2} is fresh.

$$V \mid \equiv \#(RV_{TA2})$$

P21: V believes that TA believes RV_{TA2}.

$$V \mid \equiv TA \mid \equiv \ RV_{TA2}$$

P22: V believes that TA has jurisdiction over RV_{TA2}. That is, TA is an authority and believes RV_{TA2}.

$$V \mid \equiv TA \ \Rightarrow RV_{TA2}$$

P23: V believes TS_{TA1} is fresh.

$$V \mid \equiv \#(TS_{TA1})$$

P24: V believes TS_{TA2} is fresh.

$$V \mid \equiv \#(TS_{TA2})$$

P25: TA believes V_{IDR} is a secure shared parameter between V and TA.

$$TA \mid \equiv (V \overset{V_{IDR}}{\longleftrightarrow} TA)$$

P26: TA believes V believes that V_{IDR} is a secure shared parameter between V and TA.

$$TA \mid \equiv V \mid \equiv (V \overset{V_{IDR}}{\longleftrightarrow} TA)$$

P27: TA believes that V believes R^i

$$TA \mid \equiv V \mid \equiv R^i$$

P28: TA believes that V has a jurisdiction over R^i

$$TA \mid \equiv V \Rightarrow R^i$$

P29: V believes TA believes that OTP is a secure shared parameter between V and TA.

$$V \mid \equiv TA \mid \equiv (V \overset{OTP}{\longleftrightarrow} MG)$$

P30: V believes that OTP is a secure shared parameter between V and TA

$$V \mid \equiv (V \overset{OTP}{\longleftrightarrow} TA)$$

P31: TA believes that OTP is a secure shared parameter between V and TA

$$TA \mid \equiv (V \overset{OTP}{\longleftrightarrow} TA)$$

P32: TA believes V believes that OTP is a secure shared parameter between V and TA.

$$TA \mid \equiv V \mid \equiv (V \overset{OTP}{\longleftrightarrow} TA)$$

P33: TA believes that CHX_{x+1} is a secure shared parameter between V and TA

$$TA| \equiv (V \xleftrightarrow{CHXx+1} TA)$$

P34: TA believes V believes that CHXx is a secure shared parameter between V and TA.

$$TA \mid \equiv V| \equiv (V \xleftrightarrow{CHXx+1} TA)$$

P35: TA believes V believes that *ssk* is a secure shared parameter between V and TA

$$TA \mid \equiv V| \equiv (V \xleftrightarrow{ssk} TA)$$

P36: V believes TA believes that *ssk* is a secure shared parameter between V and TA

$$V \mid \equiv TA| \equiv (V \xleftrightarrow{ssk} TA)$$

P37: TA believes V believes that OTPnew is a secure shared parameter between V and TA

$$TA \mid \equiv V| \equiv (V \xleftrightarrow{OTPnew} TA)$$

P38: V believes TA believes that OTPnew is a secure shared parameter between V and TA

$$V \mid \equiv TA| \equiv (V \xleftrightarrow{OTPnew} TA)$$

6.7.1.1.2.1.4 Analysis of the V-TA authentication protocol

We then prove that the proposed protocol achieves the security goals based on the idealized form of the messages, assumptions, and BAN logic rules. The proposed

authentication scheme analysis is shown below to prove that the protocol achieves mutual authentication between V and TA.

According to M_1:

$$V_1\text{: TA} \triangleleft :(V_{IDR}, TS_{v1}, RV_{v1})\ _{VTA1}$$

According to P1, P25 and Rule 1, we derive the following

V2:

$$\frac{TA|\equiv V \overset{\leftrightarrow}{VTA}\ TA, TA \lhd (V_{IDR}, , TS_{V1}, RV_{V1})\ V \overset{\leftrightarrow}{VTA}\ TA}{TA|\equiv V\ |\sim\ (\ (V_{IDR}, TS_{V1}, RV_{V1})}$$

According to P9, P12, and rule 3, we derive the following

$$\text{V3: } \frac{TA|\equiv \#(TS_{V1}\ and\ RV_{V1})}{TA|\equiv \#|(V_{IDR}, TS_{V1}, RV_{V1})}$$

According to P2, P26, V2, V3, and rule 2, we derive the following

$$\text{V4: } \frac{TA|\equiv \#(V_{IDR}, , TS_{V1}, RV_{V1}), TA|\equiv V \sim (V_{IDR}, TS_{V1}, RV_{V1})}{TA|\equiv V|\equiv (V_{IDR}, TS_{V1}, RV_{V1})}$$

According to V4, P10, P11, and rule 4, we derive the following:

$$\text{V5: } \frac{TA\ |\equiv V| \Rightarrow RV_{v1}\ , TA\ |\equiv\ V\ |\equiv RV_{v1}}{TA|\equiv\ RV_{v1}}$$

According to M_2, we get

$$V6\text{: } V \triangleleft\ (RV_{TA1}, TS_{TA1}, OTP_{Vi}, C)\ _{Yone}$$

According to P3, P 29 and Rule 1, we derive the following

V7:

$$\frac{V|\equiv V \overset{Yone}{\leftrightarrow} TA, V \triangleleft (TS_{TA1}, RV_{TA1}, V \overset{OTP}{\leftrightarrow} TA, C) \quad V \overset{Yone}{\leftrightarrow} TA}{V|\equiv TA|\sim (TS_{TA1}, RV_{TA1}, V \overset{OTP}{\leftrightarrow} TA, C)}$$

According to P17, P23 and rule 3, we derive the following

$$\text{V8: } \frac{V|\equiv \#(TS_{TA1}, RV_{TA1},)}{V|\equiv \#(TS_{TA1}, RV_{TA1}, V \overset{OTP}{\leftrightarrow} TA, C)}$$

According to P30, V7, V8, and rule 2, we derive the following

V9:

$$\frac{V|\equiv \#(TS_{TA1}, RV_{TA1}, V \overset{OTP}{\leftrightarrow} TA, C), V|\equiv TA\sim (TS_{TA1}, RV_{TA1}, V \overset{OTP}{\leftrightarrow} TA, C)}{V|\equiv TA|\equiv (TS_{TA1}, RV_{TA1}, V \overset{OTP}{\leftrightarrow} TA, C)}$$

According to V9, P18, P19 and rule 4, we derive the following:

$$\text{V10: } \frac{V|TA| \Rightarrow RV_{TA1}, V|\equiv TA|\equiv RV_{TA1}}{V|\equiv RV_{TA1}}$$

According to M_3:

V_{11} TA $\triangleleft$ (V_{IDR}, TS_{V2}, $RV_{V2,}$ OTP_{Vi} ,Ri) $_{CHXi+1}$

According to P5, P25, P27, P31 and Rule 1, we derive the following

V12:

$$\frac{TA|\equiv V \overset{CHXx+1}{\leftrightarrow} TA, TA \triangleleft (VIDR, TS_{V2}, RV_{V2}, V \overset{Ri}{\leftrightarrow} TA, V \overset{OTP}{\leftrightarrow} TA) \quad V \overset{CHAx+1}{\leftrightarrow} TA}{TA|\equiv V|\sim (VIDR, TS_{V2}, RV_{V2}, V \overset{Ri}{\leftrightarrow} TA, \ V \overset{OTP}{\leftrightarrow} TA)}$$

According to P13, P16, and rule 3, we derive the following

$$\text{V13: } \frac{TA|\equiv\#(TS_{V2}\ and\ RV_{V2})}{MTA|\equiv\#|(\text{VIDR},\text{TS}_{V2},RV_{V2},\text{V Ri TA},\text{V OTP TA})}$$

According to P6, P26, P32, V12, V13, and rule 2, we derive the following

V14:

$$\frac{TA|\equiv\#(\text{TS}_{V2},RV_{V2},\text{V Ri TA},\text{V OTP TA}),\text{V}|\equiv\text{TA}\sim(\text{TS}_{TA1},RV_{TA1},\text{V Ri TA},\text{V OTP TA})}{TA|\equiv\text{V}|\equiv(\text{TS}_{TA1},RV_{TA1},\text{V Ri TA},\text{V OTP TA})}$$

According to V14, P15, P28 and rule 4, we derive the following:

$$\text{V15: } \frac{\text{TA}\ |\equiv\text{V}|\Rightarrow RV_{v2}\ ,\text{Ri TA}\ |\equiv\ \text{V}\ |\equiv RV_{v2},Ri}{\text{TA}|\equiv\ RV_{v2},Ri}$$

According to M_4, we get

V16: $\text{V} \triangleleft (\text{V}_{\text{IDR}}, \text{TS}_{\text{TA2}},\text{RV}_{\text{TA2,}}\ \text{OTPvi})_{\text{Yfour}}$

According to P7, P29, and Rule 1, we derive the following

V17:

$$\frac{TA|\equiv\text{V}\ \ Yfour\ \ \text{TA},\text{V}\lhd(\text{VIDR},\text{TS}_{TA2},RV_{TA2},\ \ \text{V OTP TA})\ \ V\ \ Yfour\ TA}{V|\equiv\text{TA}\ |\sim\ (\ \text{VIDR},\text{TS}_{TA2},RV_{TA2},\text{V OTP TA})}$$

According to P20, P24 and rule 3, we derive the following

$$\text{V18: } \frac{V|\equiv\#(TS_{TA2,}\ RV_{TA2,})}{V|\equiv\#(\text{VIDR},\text{TS}_{TA2},RV_{TA2},\text{V OTP TA})}$$

According to P8, P30, V17, V18, and rule 2, we derive the following

V19:

$$\frac{V|\equiv\#(\text{VIDR},\text{TS}_{TA2},RV_{TA2},\text{V OTP TA}),\text{V}|\equiv\text{TA}\sim(\text{VIDR},\text{TS}_{TA2},RV_{TA2},\text{V OTP TA})}{V|\equiv\text{TA}|\equiv(\text{VIDR},\text{TS}_{TA2},RV_{TA2},\text{V OTP TA})}$$

According to V19, P21, P22 and rule 4, we derive the following:

$$\text{V20: } \frac{V\ |\text{TA}|\Rightarrow RV_{TA2},\text{V}\ |\equiv\ \text{TA}\ |\equiv\ RV_{TA2}}{V|\equiv\ RV_{TA2}}$$

As session key *ssk* = H (Ri || (RV_{vi1}|| RV_{TA2} ||RV_{vi2} || RV'_{TA1}) and in combination with P10, P14, V4 and V14, we can derive

$$\text{V21: TA}|\equiv\text{V}\ |\equiv(\text{V}\overset{ssk}{\longleftrightarrow}\text{TA}) \quad \text{(Goal 1\_1)}$$

As session key *ssk* = H (Ri || (RV_{vi1}|| RV_{TA2} ||RV_{vi2} || RV'_{TA1}) and combining with P27, P35 and V5, V15, V21

$$\text{V22: TA}|\equiv(\text{V}\overset{ssk}{\longleftrightarrow}\text{TA}) \quad \text{(Goal 1\_2)}$$

As session key *ssk* = H (Ri || (RV_{vi1}|| RV_{TA2} ||RV_{vi2} || RV'_{TA1}) and combining with P18, P21, V9 and V19 we can derive

$$\text{V23: V}\equiv\text{TA}|\equiv(\text{V}\overset{ssk}{\longleftrightarrow}\text{TA}) \quad \text{(Goal 1\_3)}$$

As session key *ssk* = H (Ri || (RV_{vi1}|| RV_{TA2} ||RV_{vi2} || RV'_{TA1}) and combining with P36, V10, V20 and V23.

$$\text{V24: V}|\equiv(\text{V}\overset{ssk}{\longleftrightarrow}\text{TA}) \quad \text{(Goal 1\_4)}$$

As OTPnew = H (OTP||RV_{TA2} || Ri) and in combination with P27, P31 and V14, we can derive

$$V25:\ TA|\equiv V\ |\equiv (V \overset{OPTnew}{\longleftrightarrow} TA) \quad (Goal\ 2\_1)$$

As OTPnew = H (OTP||RV_{TA2} || Ri) and combining with P32, P37, and V15, PV25

$$V26:\ TA|\equiv (V \overset{OTPnew}{\longleftrightarrow} TA) \quad (Goal\ 2\_2)$$

As OTPnew = H (OTP||RV_{TA2} || Ri) and combining with P21, P29 and V19 we can derive

$$V27:\ V\equiv TA|\equiv (V \overset{OTPnew}{\longleftrightarrow} TA) \quad (Goal\ 2\_3)$$

As OTPnew = H (OTP||RV_{TA2} || Ri) and combining with P30, P38, V20, and V27.

$$V28:\ V|\equiv (V \overset{OTPnew}{\longleftrightarrow} TA) \quad (Goal\ 2\_4)$$

Hence, the above logic proves that the proposed protocol successfully achieves Goals 1.1 to 1.4 and 2.1 to 2.4. In other words, the proposed protocol achieves mutual authentication and secure session key agreement between V and TA.

6.7.1.1.2.2 Protocol 2: Vehicle- Driver-VOCS authentication

6.7.1.1.2.2.1 Phase 1: V-VOCS

In this section, we use BAN logic to verify the legitimacy of the session key (*ssk*)**,** R^{i+1}**,** and OTTnew that are shared between V and VOCS, which communicate in the presented phase. To ensure the security of the proposed protocol under BAN logic, it needs to satisfy a set of security goals. The following section defines the main goals of the analysis of the presented authentication scheme.

6.7.1.1.2.2.1.1 Goals identification

Goal 1: R^{i+1} **is well protected and believed by VOCS.**

G1_1: VOCS believes that the V believes that the R^{i+1} is securely shared parameters between V and VOCS.

$$VOCS| \equiv V | \equiv (V \overset{R^{i+1}}{\longleftrightarrow} VOCS)$$

G1_2: VOCS believes that R^{i+1} is securely shared parameters between V and VOCS.

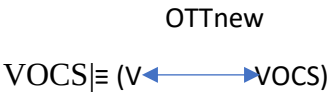
$$VOCS| \equiv (V \overset{R^{i+1}}{\longleftrightarrow} VOCS)$$

Goal 2: The V and the VOCS want to establish a One Time Token (OTT) key that each other believes.

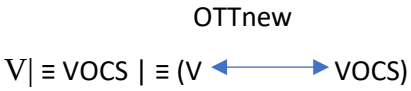
G2_1: VOCS believes that the V believes that the OTTNew is a securely shared parameter between V and VOCS.

OTTnew

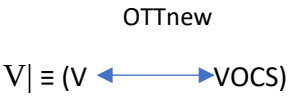
G2_2: VOCS believes that OTTnew is a securely shared parameter between V and VOCS.



G2_3: V believes that the VOCS believes that the OTTnew is a securely shared parameter between V and VOCS.

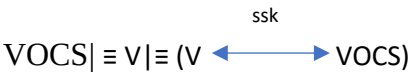


G2_4: V believes that OTTnew is a securely shared parameter between V and VOCS.

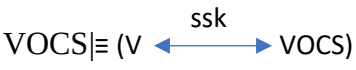


Goal 3: The V and the VOCS want to establish a shared secret key (*ssk*) that each other believes.

G3_1: VOCS believes that the V believes the *ssk* is a securely shared parameter between V and VOCS.



G3_2: VOCS believes that *ssk*is a securely shared parameter between V and VOCS.



G3_3: V believes that the VOCS believes that the *ssk* is a securely shared parameter between V and VOCS.

$$V| \equiv VOCS\ |\ \equiv (V \xleftrightarrow{} VOCS)$$

G3_4: V believes that *ssk* is a securely shared parameter between V and VOCS.

$$V| \equiv (V \xleftrightarrow{ssk} VOCS)$$

6.7.1.1.2.2.1.2 Messages idealization

First, we transferred all transmitted messages into idealized form as follows.

M1: $V \longrightarrow VOCS:(V_{IDR}, TS_{v1}, RV_{v1}, OTT, V_{CS1})_{VX1}$

M2: $VOCS \longrightarrow V:(V_{IDR}, RV_{TA1}, TS_{TA1}, C, OTT, V_{CS1})_{CSY1}$

M 3: $V \longrightarrow VOCS:(V_{IDR}, OTT, TS_{V2}, RV_{V2}, C^{i+1}\ Ri, R^{i+1})_{VXfour}$

6.7.1.1.2.2.1.3 Main assumptions

The second step to be completed is to define some assumptions as the initiative promises. The fundamental assumptions of the presented authentication scheme are as follows:

P1: VOCS believes that VX_1 is a secure shared parameter between V and VOCS

$$VOCS| \equiv (V \xleftrightarrow{VX1} VOCS)$$

P2: VOCS believes V believes that VX_1 is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv V \mid \equiv (V \overset{VX1}{\longleftrightarrow} VOCS)$$

P3: V believes that CSY1 is a secure shared parameter between V and VOCS

$$V \mid \equiv (V \overset{CSY1}{\longleftrightarrow} VOCS)$$

P4: V believes VOCS believes that CSY1 is a secure shared parameter between V and VOCS.

$$V \mid \equiv VOCS \mid \equiv (V \overset{CSY1}{\longleftrightarrow} VOCS)$$

P5: VOCS believes that VX_4 is a secure shared parameter between V and VOCS

$$VOCS \mid \equiv (V \overset{VX4}{\longleftrightarrow} VOCS)$$

P6: VOCS believes V believes that VX_4 is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv V \mid \equiv (V \overset{VX4}{\longleftrightarrow} VOCS)$$

P7: VOCS believes RV_{V1} is fresh.

$$VOCS \mid \equiv \#(RV_{V1})$$

P8: VOCS believes that V believes RV_{V1}.

$$VOCS \mid \equiv V \mid \equiv \ RV_{V1}$$

P9: VOCS believes that V has jurisdiction over RV_{V1}. That is, V is an authority and believes RV_{V1}.

$$VOCS \mid \equiv V \Rightarrow RV_{V1}$$

P10: VOCS believes RV_{V2} is fresh.

$$VOCS \mid \equiv \#(RV_{V2})$$

P11: VOCS believes that V believes RV_{V2}.

$$VOCS \mid \equiv V \mid \equiv RV_{V2}$$

P12: VOCS believes that V has jurisdiction over RV_{V2}. That is, V is an authority and believes RV_{V2}.

$$VOCS \mid \equiv V \Rightarrow RV_{V2}$$

P13: VOCS believes TS_{V1} is fresh.

$$VOCS \mid \equiv \#(TS_{V1})$$

P14: VOCS believes TS_{V2} is fresh.

$$VOCS \mid \equiv \#(TS_{V2})$$

P15: V believes RV_{VOCS1} is fresh.

$$V \mid \equiv \#(RV_{VOCS1})$$

P16: V believes that VOCS believes RV_{VOCS1}.

$$V \mid \equiv VOCS \mid \equiv RV_{VOCS1}$$

P17: V believes that VOCS has jurisdiction over RV_{VOCS1}. That is, VOCS is an authority and believes RV_{VOCS1}.

$$V \mid \equiv VOCS \Rightarrow RV_{VOCS1}$$

P18: V believes TS_{VOCS1} is fresh.

$$V \mid \equiv \#(TS_{VOCS1})$$

P19: VOCS believes V_{IDR} is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv (V \overset{V_{IDR}}{\longleftrightarrow} VOCS)$$

P20: VOCS believes V believes that V_{IDR} is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv V \mid \equiv (V \overset{V_{IDR}}{\longleftrightarrow} VOCS)$$

P21: VOCS believes that V believes R^i

$$VOCS \mid \equiv V \mid \equiv R^i$$

P22: VOCS believes that V has a jurisdiction over R^i

$$VOCS \mid \equiv V \Rightarrow R^i$$

P23: VOCS believes that V believes R^{i+1}

$$VOCS \mid \equiv V \mid \equiv R^{i+1}$$

P24: VOCS believes that V has a jurisdiction over R^{i+1}

$$VOCS| \equiv V \Rightarrow R^{i+1}$$

P25: V believes that VOCS believes C

$$V| \equiv VOCS | \equiv C$$

P26: V believes that VOCS has a jurisdiction over C

$$VOCS| \equiv V \Rightarrow C$$

P27: V believes that OTT is a secure shared parameter between V and VOCS

$$V| \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

P28: V believes VOCS believes that OTT is a secure shared parameter between V and VOCS.

$$V | \equiv VOCS| \equiv \overset{OTT}{\longleftrightarrow} (V \quad VOCS)$$

P29: VOCS believes that OTT is a secure shared parameter between V and VOCS

$$VOCS| \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

P30: VOCS believes V believes that OTT is a secure shared parameter between V and VOCS.

$$VOCS | \equiv V| \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

P31: VOCS believes V believes that *ssk* is a secure shared parameter between V and VOCS

$$VOCS \mid \equiv V \mid \equiv (V \overset{ssk}{\longleftrightarrow} VOCS)$$

P32: V believes VOCS believes that *ssk* is a secure shared parameter between V and VOCS

$$V \mid \equiv VOCS \mid \equiv (V \overset{ssk}{\longleftrightarrow} VOCS)$$

P33: VOCS believes V believes that OTTnew is a secure shared parameter between V and VOCS

$$VOCS \mid \equiv V \mid \equiv (V \overset{OTTnew}{\longleftrightarrow} VOCS)$$

P34: V believes VOCS believes that OTTnew is a secure shared parameter between V and VOCS

$$V \mid \equiv VOCS \mid \equiv (V \overset{OTTnew}{\longleftrightarrow} VOCS)$$

6.7.1.1.2.2.1.4 Analysis of the authentication V-VOCS phase

We then prove that the proposed protocol achieves the security goals based on the idealized form of the messages, assumptions, and BAN logic rules. The proposed authentication protocol analysis is shown below to prove that the protocol achieves mutual authentication between V and VOCS.

According to M_1:

$$V_1\text{: VOCS} \triangleleft :(:(V_{IDR}, TS_{v1}, RV_{v1}, \text{OTT}, V_{CS1})\ _{VX1}$$

According to P1, P19, P29, and Rule 1, we derive the following

V2:

$$\frac{VOCS|\equiv V \overset{\longleftrightarrow}{VX1}\ \text{VOCS}, \text{VOCS} \triangleleft (V_{IDR}, V\ OTT\ VOCS, TS_{V1}, V_{CS1}\ RV_{V1})\ V \overset{\longleftrightarrow}{VX1}\ VOCS}{VOCS|\equiv V|\sim (\ (V_{IDR}, V \overset{\longleftrightarrow}{OTT}\ VOCS, TS_{V1}, V_{CS1,} RV_{V1})}$$

According to P7, P13, and rule 3, we derive the following

$$\text{V3: } \frac{VOCS|\equiv \#(TS_{V1}\ and\ RV_{V1})}{VOCS|\equiv \#|(V_{IDR}, V\ \overset{\longleftrightarrow}{OTT}\ VOCS, TS_{V1}, V_{CS1,} RV_{V1})}$$

According to P2, P20, P30, V2, V3, and rule 2, we derive the following

V4:

$$\frac{VOCS|\equiv \#(V_{IDR}, V \overset{\longleftrightarrow}{OTT}\ VOCS, TS_{V1}, V_{CS1,} RV_{V1}), \text{VOCS}|\equiv V\sim (V_{IDR}, V\ OTT\ VOCS, TS_{V1}, V_{CS1}\ \overset{\longleftrightarrow}{RV_{V1}})}{VOCS|\equiv V|\equiv (V_{IDR}, V \overset{\longleftrightarrow}{OTT}\ VOCS, TS_{V1}, V_{CS1,} RV_{V1})}$$

According to V4, P8, P9, and rule 4, we derive the following:

$$\text{V5: } \frac{\text{VOCS}\ |\equiv V| \Rightarrow RV_{v1}\ , \text{VOCS}\ |\equiv V\ |\equiv RV_{v1}}{\text{VOCS}|\equiv\ RV_{v1}}$$

According to M_2, we get

V6: $V \triangleleft (V_{IDR}, RV_{CS1}, TS_{CS1}, C, OTT, V_{CS1})_{CSY1}$

According to P3, P27, and Rule 1, we derive the following

V7

$$\frac{V|\equiv V \overset{\text{CSYone VOCS}}{\longleftrightarrow} ,V \triangleleft (TS_{VOCS1}, RV_{VOCS1}, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, V_{CS1}, C)\ V \overset{\text{CSYone VOCS}}{\longleftrightarrow}}{V|\equiv VOCS\ |\sim (TS_{VOCS1}, RV_{VOCS1}, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, V_{CS1}, C)}$$

According to P15, P18 and rule 3, we derive the following

$$\text{V8: } \frac{V|\equiv \#(TS_{CS1}, RV_{CS1},)}{V|\equiv \#(TS_{CS1}, RV_{CS1}, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, V_{CS1}, C)}$$

According to P4, P16, P25, P28, V7, V8, and rule 2, we derive the following

V9:

$$\frac{V|\equiv \#(TS_{CS1}, RV_{CS1}, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, C), V|\equiv VOCS \sim (TS_{CS1}, RV_{CS1}, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, C)}{V|\equiv VOCS|\equiv (TS_{CS1}, RV_{CS1}, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, C)}$$

According to V9, P17, P26 and rule 4, we derive the following:

$$\text{V10: } \frac{V\ |VOCS| \Rightarrow RV_{CS1}, C \quad V\ |\equiv VOCS| \equiv RV_{CS1}, C}{V|\equiv\ RV_{CS1}, C}$$

According to M_3:

V_{11} VOCS $\triangleleft (V_{IDR}, OTT, TS_{V2}, RV_{V2}, Ri, R^{i+1})$ VXfour

According to P5, P19, P29, and Rule 1, we derive the following

V12:

$$\frac{VOCS|\equiv V \overset{\text{VXfour VOCS}}{\longleftrightarrow} ,VOCS \triangleleft (VIDR, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, TS_{V2}, RV_{V2}, R^1, R^{i+1})\ V \overset{\text{VXfour VOCS}}{\longleftrightarrow}}{VOCS|\equiv V\ |\sim (VIDR, V \overset{\text{OTT VOCS}}{\longleftrightarrow}, TS_{V2}, RV_{V2}, R^1, R^{i+1})}$$

According to P10, P14, and rule 3, we derive the following

$$\text{V13: } \frac{VOCS|\equiv\#(TS_{V2}\ and\ RV_{V2})}{TA|\equiv\#|(\text{VIDR, V} \overset{\text{OTT}}{\leftrightarrow} \text{VOCS,TS}_{V2},RV_{V2},R^1,R^{i+1})}$$

According to P11, P20, P21, P23, P30, V12, V13, and rule 2, we derive the following

V14:

$$\frac{VOCS|\equiv\#(\text{VIDR, V} \overset{\text{OTT}}{\leftrightarrow} \text{VOCS,TS}_{V2},RV_{V2},R^1,R^{i+1}),\text{V}|\equiv\text{VOCS}\sim(\text{VIDR, V} \overset{\text{OTT}}{\leftrightarrow} \text{VOCS,TS}_{V2},RV_{V2},R^1,R^{i+1})}{VOCS|\equiv\text{V}|\equiv(\text{VIDR, V} \overset{\text{OTT}}{\leftrightarrow} \text{VOCS,TS}_{V2},RV_{V2},R^1,R^{i+1})}$$

According to V14, P12, P22, P24, and rule 4, we derive the following:

$$\text{V15: } \frac{\text{VOCS }|\equiv\text{V}|\Rightarrow RV_{v2},R^1,R^{i+1},\text{VOCS }|\equiv\text{ V }|\equiv RV_{v2},R^1,R^{i+1}}{\text{VOCS}|\equiv\ RV_{v2,RV_{v2}},R^1,R^{i+1}}$$

Based on P21, P23, V15, we can derive

$$\text{V16: VOCS}|\equiv|\equiv(\text{V} \overset{R^i,R^{i+1}}{\leftrightarrow} \text{VOCS}) \quad \text{(Goal 1\_1)}$$

Based on P22, P24, V15, and V16, we derive the following

$$\text{V17: VOCS}|\equiv(\text{V} \overset{R^i,R^{i+1}}{\leftrightarrow} \text{VOCS}) \quad \text{(Goal 1\_2)}$$

As session key *ssk* = H (R^i|| (RV'_{VI1}|| RV'_{VI2} ||RV_{CS1}) and in combination with V4 and V14 we can derive

$$\text{V18: VOCS}|\equiv\text{V}|\equiv(\text{V} \overset{ssk}{\leftrightarrow} \text{VOCS}) \quad \text{(Goal 2\_1)}$$

As session key *ssk*= H (R^i|| (RV'_{VI1}|| RV'_{VI2} ||RV_{CS1}) and combining with P31, V18, and Rule 4

V19: VOCS|≡ (V $\overset{ssk}{\longleftrightarrow}$ VOCS) (Goal 2_2)

As session key *ssk* = H (R^i|| (RV'_{VI1}|| RV'_{VI2} ||RV_{CS1}) and combining with V9 we can derive

V20: V ≡VOCS|≡ (V $\overset{ssk}{\longleftrightarrow}$ VOCS) (Goal 2_3)

As session key SK H (R^i|| (RV'_{VI1}|| RV'_{VI2} ||RV_{VOCS1}) and combining with P32, V20, and Rule 4.

V21: V|≡ (V $\overset{ssk}{\longleftrightarrow}$ VOCS) (Goal 2_4)

As OTTnew = H (OTT|| RV_{Vi2}|| C) and in combination with V4, V14 we can derive

V22: VOCS| ≡ V |≡ (V $\overset{OTTnew}{\longleftrightarrow}$ VOCS) (Goal 3_1)

As OTTnew = H (OTT|| RV_{Vi2}|| C) and combining with P33, V22, and Rule 4, we can drive

$$V23: VOCS|\equiv (V \overset{OTTnew}{\longleftrightarrow} VOCS) \qquad \text{(Goal 3\_2)}$$

As OTTnew = H (OTT|| RV_{Vi2}|| C) and combining with V9 we can derive

$$V24: V|\equiv VOCS|\equiv (V \overset{OTTnew}{\longleftrightarrow} TA) \qquad \text{(Goal 3\_3)}$$

As OTTnew = H (OTT|| RV_{Vi2}|| C) and combining with P34, V24, and Rule 4 we can drive.

$$V25: V|\equiv (V \overset{OTTnew}{\longleftrightarrow} VOCS) \qquad \text{(Goal 3\_4)}$$

The above logic proves that the proposed protocol achieves Goals 1.1 to 1.2, 2.1 to 2.4, and 3.1-3.4 successfully. In other words, the proposed protocol achieves mutual authentication and secure session key agreement between V and VOCS. In summary, by achieving all the goals mentioned above, we demonstrate the validity of our proposed protocols. We proved that the presented three protocols are a secure mechanism to achieve mutual authentication and secret key generation to secure the communication between the involved parties.

6.7.1.1.2.2.2 Phase 2: D-VOCS authentication

In this section, we use BAN logic to verify the legitimacy of the authentication parameter Z_{U2} shared between driver and VOCS, which communicate in the presented protocol. To ensure the security of the proposed protocol under BAN logic, it needs to satisfy a set of security goals. The following section defines the main goals of the analysis of the presented authentication scheme.

6.7.1.1.2.2.2.1 Goals identification

Goal 1: Z_{U1} is well protected and believed by Di and VOCS.

G1_1: VOCS believes that the Di believes that the Z_{U1} is a securely shared parameter between Di and VOCS.

$$\text{VOCS}| \equiv \text{V and Di}| \equiv (\text{Di} \overset{Z_{U1}}{\longleftrightarrow} \text{VOCS})$$

G1_2: VOCS believes that Z_{U1} is a securely shared parameter between Di and VOCS.

$$\text{VOCS}| \equiv (\text{Di} \overset{Z_{U1}}{\longleftrightarrow} \text{VOCS})$$

G2_1: Di believes that the VOCS believes the UVi is a securely shared parameter between Di and VOCS.

$$\text{Di}| \equiv \text{VOCS} | \equiv (\text{Di} \overset{UVi}{\longleftrightarrow} \text{VOCS})$$

G2_2: Di believes that UVi is a securely shared parameter between Di and VOCS.

$$\text{Di}| \equiv (\text{Di} \overset{UVi}{\longleftrightarrow} \text{VOCS})$$

6.7.1.1.2.2.2.2 Messages idealization

First, we transferred all transmitted messages into idealized forms as follows:

$$\textbf{M1: Di} \xrightarrow{Z_{U1}} \textbf{VOCS:} (\{VIDR, TSvi, Ri, OTT, FP, PUi, Di \rightarrow VOCS\})_{ssk}$$

$$\textbf{M1: VOCS} \xrightarrow{UVi} \textbf{Di:} (\{TS_{CS1}, Di \rightarrow VOCS\})_{ssk}$$

6.7.1.1.2.2.2.3 Main assumption

The second step is to define some assumptions as initiative promises. The fundamental assumptions of the presented authentication scheme is as follows:

P1: VOCS believes that *ssk* is a secure shared parameter between V and VOCS

$$VOCS| \equiv (V \overset{ssk}{\longleftrightarrow} VOCS)$$

P2: VOCS believes V believes that *ssk* is a secure shared parameter between V and VOCS.

$$VOCS | \equiv V| \equiv (V \overset{ssk}{\longleftrightarrow} VOCS)$$

P3: V believes that *ssk* is a secure shared parameter between V and VOCS

$$V| \equiv (V \overset{ssk}{\longleftrightarrow} VOCS)$$

P4: V believes VOCS believes that *ssk* is a secure shared parameter between V and VOCS.

$$V \mid \equiv V \mid \equiv (V \longleftrightarrow VOCS)$$

P5: VOCS believes TS_{V1} is fresh.

$$VOCS \mid \equiv \#(TS_{V1})$$

P6: V believes TS_{CS1} is fresh.

$$V \mid \equiv \#(TS_{CS1})$$

P7: VOCS believes that Di believes Z_{U1}

$$VOCS \mid \equiv Di \mid \equiv Z_{U1}$$

P8: VOCS believes that Di has a jurisdiction over Z_{U1}

$$VOCS \mid \equiv Di \Rightarrow Z_{U1}$$

P9: Di believes that VOCS believes UVi

$$Di \mid \equiv VOCS \mid \equiv UVi$$

P10: Di believes that VOCS has a jurisdiction over UVi

$$Di \mid \equiv VOCS \Rightarrow UVi$$

P11: VOCS believes V_{IDR} is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv (V \overset{V_{IDR}}{\longleftrightarrow} VOCS)$$

P12: VOCS believes V believes that V_{IDR} is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv V \mid \equiv (V \overset{V_{IDR}}{\longleftrightarrow} VOCS)$$

P13: VOCS believes that Di believes FP

$$VOCS \mid \equiv Di \mid \equiv FP$$

P14: VOCS believes that Di has a jurisdiction over FP

$$VOCS \mid \equiv Di \Rightarrow FP$$

P15: VOCS believes that Di believes PUi

$$VOCS \mid \equiv Di \mid \equiv PUi$$

P16: VOCS believes that Di has a jurisdiction over PUi

$$VOCS \mid \equiv Di \Rightarrow PUi$$

P17: VOCS believes that V believes R^i

$$VOCS \mid \equiv V \mid \equiv R^i$$

P18: VOCS believes that V has a jurisdiction over R^i

$$VOCS \mid \equiv V \Rightarrow R^i$$

P19: VOCS believes that OTT is a secure shared parameter between V and VOCS

$$V \mid \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

P20: VOCS believes V believes that OTT is a secure shared parameter between V and VOCS.

$$V \mid \equiv VOCS \mid \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

P21: V believes that OTT is a secure shared parameter between V and VOCS

$$VOCS \mid \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

P22: V believes V believes that OTT is a secure shared parameter between V and VOCS.

$$VOCS \mid \equiv V \mid \equiv (V \overset{OTT}{\longleftrightarrow} VOCS)$$

6.7.1.1.2.2.2.4 Analysis of D-VOCS authentication phase

The third step is to start analyzing the authentication scheme to prove that the proposed scheme achieves mutual authentication between Di and VOCS.

According to M_1:

$$V_1: \ VOCS \triangleleft (\{VIDR, TSvi, U1, FP, PUi, \ Di \overset{Z_{U1}}{\longrightarrow} VOCS\})_{ssk}$$

According to P1, and Rule 1, we derive the following

V2:

$$\frac{VOCS \mid \equiv V \overset{ssk}{\longleftrightarrow} VOCS, VOCS \triangleleft (V_{IDR}, Di \overset{Z_{U1}}{\longleftrightarrow} VOCS, TS_{V1}, Ri, OTT, FP, PUi) \quad V \overset{ssk}{\longleftrightarrow} VOCS}{VOCS \mid \equiv V \mid \sim ((V_{IDR}, Di \overset{Z_{U1}}{\longleftrightarrow} VOCS, TS_{V1}, Ri, OTT, FP, PUi)}$$

According to P5 and rule 3, we derive the following

$$\text{V3: } \frac{VOCS|\equiv\#(TS_{V1})}{VOCS|\equiv\#|\left(V_{IDR}, Di\ Z_{U1}\ VOCS, TS_{V1}, \text{Ri,OTT,FP,PUi}\right)}$$

According to P2, P7,P12,P13,P15,P17,P20, V2,V3, and rule 2, we derive the following

V4:

$$\frac{VOCS|\equiv\#(V_{IDR}, Di\ Z_{U1}\ VOCS, TS_{V1}, \text{Ri,OTT,FP,PUi}), \text{VOCS}|\equiv \text{V and Di}\sim(V_{IDR}, Di\ Z_{U1}\ VOCS, TS_{V1}, \text{Ri,OTT,FP,PUi})}{VOCS|\equiv \text{V and Di}|\equiv(V_{IDR}, Di\ Z_{U1}\ VOCS, TS_{V1}, \text{Ri,OTT,FP,PUi})}$$

According to P18, V4 and rule 4, we derive the following:

$$\text{V5: } \frac{\text{VOCS }|\equiv \text{V}| \Rightarrow Ri\ , \text{VOCS }|\equiv \text{V }|\equiv \text{Ri}}{\text{VOCS}|\equiv\ \text{Ri}}$$

According to P8, P13, P16, V4 and rule 4, we derive the following:

$$\text{V6: } \frac{\text{VOCS }|\equiv \text{Di}|\Rightarrow Z_{U1}, FP, PUi\ \text{VOCS }|\equiv \text{V }|\equiv\ Z_{U1}, FP, PUi}{\text{VOCS}|\equiv\ \ Z_{U1}, FP, PUi}$$

According to M_2, we get

$$\text{V7: V} \triangleleft (\{\text{TS}_{\text{CS1}}, \text{Di} \xrightarrow{\text{UVi}} \text{VOCS}\})_{ssk}$$

According to P3, and Rule 1, we derive the following

$$\text{V8} \frac{V|\equiv \text{V}\ \ ssk\ \ \text{VOCS}\ , \text{V} \triangleleft (\text{TS}_{CS1}, \text{VOCS}\ UV_i\ \ \text{Di})\ \ V\ \ ssk\ \ VOCS}{V|\equiv \text{VOCS }|\sim\ (\text{TS}_{CS1}, \text{VOCS}\ UV_i\ \ \text{Di})}$$

According to P6 and rule 3, we derive the following

$$\text{V9: } \frac{V|\equiv\#(\mathrm{TS}_{CS1}\ )}{V|\equiv\#(\mathrm{TS}_{CS1},\mathrm{VOCS}\ \overset{UV_i}{\longleftrightarrow}\ \mathrm{Di})}$$

According to P4, P9 and rule 2, we derive the following

V10:

$$\frac{V|\equiv\#(\mathrm{TS}_{CS1},\mathrm{VOCS}\ \overset{UV_i}{\longleftrightarrow}\ \mathrm{Di}),\mathrm{V}|\equiv\mathrm{VOCS}\sim\mathrm{TS}_{CS1},\mathrm{VOCS}\ \overset{UV_i}{\longleftrightarrow}\ \mathrm{Di})}{V|\equiv\mathrm{VOCS}|\equiv(\mathrm{TS}_{CS1},\mathrm{VOCS}\ \overset{UV_i}{\longleftrightarrow}\ \mathrm{Di})}$$

According to V10, P10, and rule 4, we derive the following:

$$\text{V11: } \frac{V\ |\mathrm{VOCS}|\Rightarrow UV_i\ ,\mathrm{V}\ |\equiv\mathrm{VOCS}|\equiv UV_i}{V|\equiv\ UV_i}$$

Based on P7, V4, we can derive

V12: $\mathrm{VOCS}|\equiv \mathrm{V}\ |\equiv (\mathrm{V} \overset{Z_{U1}}{\longleftrightarrow} \mathrm{VOCS})$ (Goal 1_1)

Based on P8, V5, V6 and V12, we derive the following

V13: $\mathrm{VOCS}|\equiv (\mathrm{V\ and\ Di} \overset{Z_{U1}}{\longleftrightarrow} \mathrm{VOCS})$ (Goal 1_2)

Based on P9, V10, we can der

V14: $\mathrm{V}|\equiv \mathrm{VOCS}\ |\equiv (\mathrm{V} \overset{UVi}{\longleftrightarrow} \mathrm{VOCS})$ (Goal 2_1)

Based on P10, V11, and V14, we derive the following

V15: $\mathrm{V}|\equiv (\mathrm{V} \overset{UVi}{\longleftrightarrow} \mathrm{VOCS})$ (Goal 2_1)

The above logic proves that the proposed protocol achieves Goals 1.1 to 1.2 and 2.1 to 2.2 successfully. In other words, the proposed protocol achieves mutual authentication between Di and VOCS. In summary, by achieving all the goals mentioned above, we demonstrate the validity of the proposed protocol.

6.7.1.2 Informal security analysis

6.7.1.2.1 Security properties assessment

The following section presents a detailed security analysis of the security properties of the proposed scheme. It demonstrates how the proposed scheme satisfies the security requirements for mutual authentication, session key agreement and resists various kinds of known attacks. Then, a comparison with other related schemes is presented.

6.7.1.2.1.1 Protocol 1: V- TA mutual authentication

6.7.1.2.1.1.1 Mutual authentication

V and TA authenticate each other during the mutual authentication by generating and verifying the correctness X_2, X_4Y_3, Y_6, and generating the *ssk*. An adversary can't generate $V_{TA} = H (TA_{IDR} || MSK_{TA} || V_{IDR}||R_{TA})$ without knowing the V_{IDR} of V, the TA_{IDR}, MSK_{TA} of the TA and the PUF response (R_{TA}) of the TA. Furthermore, the adversary will not be able to generate X_2 without having V_{TA}, RV_{Vi1} $V_{TA1,}$ and V_{IDR} and can't generate X_4 without having V_{IDR}, RV_{Vi2} and CHX_x^{i+1}. The generation of CHX_x^{i+1} requires the generation of the PUF response (Ri) that can be generated only by V. Additionally, the adversary cannot generate Y_3 without having C, Y_1, RV_{TA1}, and $RV_{Vi1'}$. Y_1 consists of both V_{TA1}' and OTPVi. Furthermore, the adversary can't generate Y_6

without having RV_{TA2}, Ri, $RV_{vi2,}$ C, and OTPVi. In summary, V_{TA1}', OTPVi, CHXxi are considered secured parameters between V and TA. Ri is a value that can only be generated by V and R_{TA} can only be generated by the TA. As a result, the proposed scheme can achieve mutual authentication between V and TA.

6.7.1.2.1.1.2 Session key agreement and forward/backward security

After completing the mutual authentication phase, V and the TA establish the shared secret key ssk. The *ssk* is a combination of RV_{Vi1}, RV_{Vi2}, RV_{TA1}, and Ri. The *ssk* will be generated locally *ssk* = H (RV_{Vi1} || RV_{Vi2} || RV_{TA1} ||Ri). The secrecy of the *ssk* depends on the secrecy of RV_{Vi1}, RV_{Vi2}, RV_{TA1}, and Ri. Because all those parameters are randomly selected for every new authentication session, the disclosed *ssk* will not cause the compromise of any future *ssk*. The forward/backward security property's objective is to ensure that any past or future shared secret keys will not be affected when any *ssk* is exposed. Even if an adversary obtains *ssk* of a session, he/she can't compute any of the past. Future shared secret keys by using the disclosed *ssk* because the *ssk* is protected by: first, the uniqueness of the Ri where each CRP will be used only one time. The corresponding responses can't be replicated or predicted because they are unique for each V because of the nature of the PUF. Second: the randomization of the RV_{Vi1}, RV_{Vi2}, and RV_{TA1}. As a result, the proposed scheme achieves the security of *ssk*.

6.7.1.2.1.1.3 Anonymity unlinkability and untraceability properties

Anonymity is of paramount importance to the IoVs. The identities of the vehicles need to be hidden to avoid profiling. For fully protected V privacy, strong anonymity with unlinkability is required. The V's real identity VIDR is not transmitted during all phases in clear text format in the proposed protocol. Therefore, even if the adversary

eavesdrops on all communication messages, it is impossible to obtain the real identity of Vi. In addition, the new alias ID of the vehicle V_{IDAnew} = H (C|| V_{IDA}) cannot be regenerated without knowledge of C. Moreover, because C is fresh in each session, the attacker cannot link any two different V_{IDAnew}'s to the same Vi. By using a new V_{IDAnew} for each authentication session, the adversary will not be able to decide whether these authentication messages are from the same Vi or not. This means that Vi cannot be linked to different sessions. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the vehicles by intercepting messages.

6.7.1.2.1.1.4 Database attack

Suppose any Vi or the TA is compromised, and the adversary can steal VTA, VIDR CHX, and the OTP from the Vi's or the TA's database to impersonate the vehicle or the TA. In that case, the fake vehicle or the fake TA will fail the CRP verification process.

6.7.1.2.1.1.5 Brute force attack

The shared secret key *ssk* is generated locally by both the vehicle and the TA using the random secret parameters RV_{N1}, RV_{N2}, RV_{MG1}, and Ri. Because this *ssk* depends on random parameters, the adversary cannot obtain it from the protocol. The probability of guessing is so negligible that the adversary will fail to guess the shared secret key's correct parameters, given that this *ssk* changes in every session.

6.7.1.2.1.1.6 Replay attack

To avoid the replay attack where an adversary tries to delay or repeat previously transmitted packets, the proposed protocol uses fresh nonce values and TS each time in

each message during the authentication process. The sender node generates the timestamp TS and then inserts the transmitted message in an encrypted format to ensure the attacker cannot replace it. Before trusting the freshness of the message, the receiver first verifies the timeliness of the timestamp. If the set threshold ΔT is exceeded, the receiver will reject the request and terminate the current session. If it is within ΔT, the receiver will calculate the verification message based on the received timestamp and the other message parameters to verify if it is equal to the received message. If they are not equal, the receiver will still reject the current session request. Therefore, if an adversary replays an intercepted message from the previous session, the other party will detect that the message is outdated and immediately terminate the session. Therefore, the protocol can be protected from replay. On another side, each CRP of the V is used only one time to ensure security against replay attacks. Hence, the proposed protocol protects against the replay attack

6.7.1.2.1.1.6 Eavesdropping attack

During the authentication phase, the adversary can intercept messages transmitted between Vi and the TA. All the intercepted messages will be useless because they are sent in an encrypted format using XOR and a one-way hash function. For the attacker to verify the received parameters, he/she needs to know the OTP, CHX, V_{TA}, V_{IDR}, TA_{IDR}, $MSK_{TA,}$ $R_{TA,}$ and Ri, which are either composed of random values or are protected and out of the adversary's reach. The only two parameters that are sent in clear text are the Alias identities and session ID. The Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The adversary will not be able to link the message to a particular device

because the proposed protocol uses alias identities that change in every session. Therefore, the proposed protocol protects against eavesdropping attacks.

6.7.1.2.1.1.7 Impersonation attack

In this attack, when the intruder eavesdrops on the messages transmitted from Vi to TA or vice versa, he/she can use the intercepted information for malicious actions, such as impersonating the vehicle or the TA and sending fabricated messages.

6.7.1.2.1.1.7.1 TA impersonation

According to the authentication phase in the proposed protocol, if an adversary wants to impersonate to be the TA, he/she needs to construct a message M2 $<TA_{IDA}, S_{ID,}$ $TS_{TA1,}$ C', Y_2, $Y_3>$. To construct this message, the adversary needs to have MSK_{TA}, V_{IDR}, R_{TA}, TA_{IDR}, and the OTPvi, which are considered to be well protected, and the adversary can't compute the PUF response (R_{TA}) of the TA. Also, for the adversary to construct M4 $< TA_{IDA,}$ TS_{TA2}, $Y_{5,}$ $Y_6>$, he/she needs to be aware of the values of Ri,CHXx, and the $CHX_{x+1'}$ which are well protected too. Therefore, the message sent by the adversary cannot pass the verification of V, so the vehicle will immediately terminate the session with the TA. Consequently, the proposed protocol can resist TA impersonation attacks.

6.7.1.2.1.1.7.2 Vehicle impersonation

To impersonate the vehicle, an adversary should intercept the messages exchanged in the previous sessions. This attack will fail for numerous reasons. First, the Alias ID of the vehicle changes every authentication session. Second, due to PUF that is built into the vehicle and the characteristics of PUF, the adversary will not be able to generate legitimate responses. Because the PUF responses are unique for each device,

the adversary will not be able to replicate the response even if he/she has the challenge. Third, if an adversary wants to impersonate the vehicle, he/she needs to construct a message M1<V_{IDA}, TS_{Vi}, X_1, X_2>. In order to construct M1, the adversary needs to know V_{TA1}, and the $V_{IDR,}$ which are considered secured parameters and never be transmitted in clear text format. Also, for the adversary to construct M3 < $V_{IDA,}$ TS_{Vi2}, $X_{4,}$ $X_{5>.,}$ he/she needs to have R, ChX, and the $CHX^{+1,}$ which are well-protected. Furthermore, the message is sent in an encrypted format using The Xor function and the one-way hash function of the message. Finally, the adversary cannot derive the secret values $RV_{vi1,}$ $RV_{vi2,}$ $RV_{TA1,}$ RV_{TA2} to calculate the session key. Thus, our protocol protects against the impersonation attack.

From another perspective, even if the adversary succeeded in compromising a vehicle, the adversary's further attacks using the compromised vehicle only affect the communication related to that vehicle. Because each vehicle has its own secret key, the adversary can't derive other non-compromised vehicles' keys without knowing those vehicles' random information. Therefore, further attacks will not affect other communications. As a result, the proposed scheme is resistant to vehicle impersonation attacks.

6.7.1.2.1.1.8 Man-in-the-middle attack

Assuming that M1 < V_{IDA}, $TS_{Vi1,}$ $X_{1,}$ and X_2 > has been tampered by an adversary M1* < V_{IDA}^{*}, TS_{Vi1}^{*}, X_1^{*}, X_2^{*}>. After receiving the message, the adversary will first need to have $V_{TA1,}$ which is a secure parameter and requires the PUF response (R_{TA}) of the TA to compute X_1. Also, the adversary needs the V_{TA1} and V_{IDR} the to calculate X_2 < H (V_{IDR} || $RVvi_1$ || V_{TA1}|| TS_{Vi1}) >. Without all secret parameters, the adversary will not

generate a legitimate X_2; if the message has been tampered with, the vehicle is considered illegal, and the TA will terminate the session immediately.

Suppose the message M2 < TA_{IDA}, $S_{ID,}$ $TS_{TA1,}$ C', Y_2, Y_3 > has been tampered with an adversary and sent to V $M2^*$ < TA_{IDA}^*, S_{ID}^*, TS_{TA1}^*, C'^*, Y_2^*, Y_3^*>. After receiving the message, the adversary will first need to compute V_{TA1}< H (MSK_{TA} || V_{IDR} || TA_{IDR}|| R_{TA} > using its master secret key, the vehicle's real identity of the trusted authority, and the PUF response of the TA. Because these four parameters are secret and are only known to the TA, the adversary will not be able to correctly calculate V_{TA}. Consequently, it will be impossible to calculate the valid Y_3 to pass the vehicle verification.

As for the message M3 < $V_{IDA,}$ TS_{Vi2}, $X_{3,}$ X_4 > suppose an adversary tempered the message M3. Because each vehicle has a PUF chip and each chip is unique, the adversary will not be able to replicate or predict the response (Ri) of the real vehicle. Consequently, the adversary will not generate R and calculate the Chained hash PUF (CHX_{x+1}). Therefore, the adversary will not be able to construct a generate X_3 and X_4 and send a legitimate message to the TA that can pass the TA's vehicle authenticate. So, o after receiving the message $M3^*$ and verifying it, TA will terminate the session with the vehicle. Finally, assume M4 < $TA_{IDA,}$ TS_{TA2}, $Y_{5,}$ Y_6 > was tempered by an adversary $M4^*$ < TA_{IDA}^*, TS_{TA2}^*, Y_5^*, Y_6^*>. Because the adversary does have Ri, he/she will not be able to construct Y_4. In summary, the proposed protocol can resist all message modification attacks. The proposed protocol can resist all message modification attacks.

6.7.1.2.1.1.9 Modeling attacks

It is an attack where the adversary collects a large number PUF CRPs and uses a regression algorithm to build a model and predict responses for new challenges. All the challenges and the responses are transferred in an encrypted format by using a one-way hash function. Therefore, the adversary will not be able to collect enough useful CRPs to build a model and predict responses to new challenges.

6.7.1.2.1.1.10 Physical attack

According to the characteristics of PUF, any change or damage on the device with built-in PUF will render the PUF useless. Therefore, the physical attacks cannot obtain any useful information. We can conclude that the proposed protocol can ensure the physical security of the system.

6.7.1.2.1.1.11 Device counterfeit/cloning

Both the TA and vehicle are authenticated using the PUF. PUF responses are treated as hardware fingerprints that cannot be duplicated or cloned. Therefore, using the PUF, the vehicle and the TA cannot be cloned.

6.7.1.2.1.1.12 Session key agreement

After confirming the identity of both the vehicle and the TA, each side will generate the session key $ssk = H(Ri \,||\, RV_{vi1} \,||\, RV_{TA2}' \,||\, (RV_{vi2} \,||\, RV'_{TA1})$. In summary, the session key between V and TA is established in the proposed protocol.

6.7.1.2.1.2 Protocol 2: Vehicle-Driver-VOCS authentication

6.7.1.2.1.2.1 Phase 1: V- VOCS mutual authentication

6.7.1.2.1.2.1.1 Mutual authentication

V and the VOCS authenticate each other during the mutual authentication by generating verifying the correctness $VX_{3,}$ $VX_{8,}$ $CSY_{3.}$ An adversary cannot prepare the authentication request message without having $V_{CS1}$and V_{CS2} that are known only by the vehicle. Therefore, the adversary will not generate VX3 without having V_{IDR}, $V_{CS1,}$ OTTvi, which are considered secure parameters. The adversary also cannot generate VX_8 without having R^i that can be generated only by V. Additionally, the adversary cannot generate CSY_3 without computing $V_{CS1} = H (VOCS_{IDR} || MSK_{VOCS} || V_{IDR}||R_{CS})$ and $V_{CS2}= H (V_{CS1} || VT_{ID})$ because $VOCS_{IDR,}$ MSK_{VOCS}, and VT_{ID} are only known to the VOCS. The R_{CS} can only be generated by the VOCS. In summary, V_{CS2} and V_{CS2} are considered secure parameters that are only known by the vehicle and need to be computed by the VOCS. OTT is regarded as a secure parameter between V and VOCS. R^i and R^{i+1} are values that V can only generate. R_{CS} is a value that can be generated only by VOCS. As a result, the proposed scheme can achieve mutual authentication between V and VOCS.

6.7.1.2.1.2.1.2 Session key agreement and forward/backward security

After completing the mutual authentication phase, the shared secret key *ssk* is established between V and the VOCS. The *ssk* is a combination of RV_{Vi1}, RV_{Vi2}, RV_{CS1}, and Ri. The *ssk* will be generated locally *ssk* = H (RV_{Vi1} || RV_{Vi2} || RV_{CS1} ||Ri). The secrecy of the *ssk* depends on the secrecy of RV_{Vi1}, RV_{Vi2}, RV_{CS1}, and Ri. Because all those parameters are randomly selected for every new authentication session, the disclosed *ssk* will not cause the compromise of any future *ssk*. The forward/backward security property's objective is to ensure that any past or future shared secret keys will not

be affected when any *ssk* is exposed. Even if an adversary obtains *ssk* of a session, he/she cannot compute any of the past or future shared secret keys. The *ssk* keys are protected by: (1) uniqueness of the Ri where each CRP will be used only one time, and (2) the randomization of RV_{Vi1}, RV_{Vi2}, and RV_{VOCS1}. As a result, the proposed scheme achieves the security of *ssk.*

6.7.1.2.1.2.1.3 Anonymity unlinkability, and untraceability properties

Anonymity is essential to protect user privacy and security. The identities of the vehicles need to be hidden to avoid profiling. For fully protected V privacy, strong anonymity with unlinkability is required. In the proposed protocol, the V's real identity V_{IDR} is not transmitted during all phases in clear text format. Therefore, even if the adversary eavesdrops on all communication messages, it is impossible to obtain the real identity of Vi. In addition, the new alias ID of the vehicle $V_{IDAnew} = H (V_{IDA} || C|| OTT)$. Moreover, because C and OTT are fresh in each session, the attacker cannot link any two different V_{IDAnew}'s to the same Vi. By using a new V_{IDAnew} for each authentication session, the adversary will not be able to decide whether these authentication messages are from the same Vi or not. This means that Vi cannot be linked to different sessions. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the vehicles by intercepting messages.

6.7.1.2.1.2.1.4 Session key security

The shared secret key *ssk* is generated locally by both the vehicle and the VOCS using the random secret parameters RV_{V1}, RV_{V2}, RV_{CS1}, and Ri where $ssk = H (RV_{V1}|| RV_{V2}|| RV_{CS1}|| Ri)$. Because the *ssk* depends on random parameters, the adversary cannot obtain it from the protocol because the four random parameters are transferred in an

encrypted format. Also, the Ri cannot be predicted or replicated because it is unique per PUF chip. The probability of guessing it is so negligible that the adversary will fail to guess the shared secret key's correct parameters, given that this *ssk* changes in every session. Thus, the proposed protocol is secure to session key disclosure attacks.

6.7.1.2.1.2.1.5 Replay attack

During the replay attack, the adversary may attempt to delay or repeat previously transmitted authentication messages; the proposed protocol uses fresh nonce values and TS each time in each message during the authentication process. The sender node generates the timestamp TS and is then inserted in the transmitted message to ensure the attacker cannot replace it. Before trusting the freshness of the message, the receiver first verifies the timeliness of the timestamp. If the set threshold ΔT is exceeded, the receiver will reject the request and terminate the current session. If it is within ΔT, the receiver will calculate the verification message based on the received timestamp and the other message parameters to verify if it is equal to the received message. If they are not equal, the receiver will still reject the current session request. Therefore, if an adversary replays an intercepted message from the previous session, the other party will detect that the message is outdated and immediately terminate the session. Therefore, the protocol is protected from replay. On another side, each CRP is used only one time to ensure security against replay attacks. Hence, the protocol is protected against replay attacks.

6.7.1.2.1.2.1.6 Eavesdropping attack

During the authentication phase, the adversary can intercept messages transmitted between Vi and the VOCS. All the intercepted messages will be useless because they are sent in an encrypted format using XOR and a one-way hash function. For the attacker to

verify the received parameters, he/she needs to know the OTT, V_{CS1}, $V_{CS2,}$ V_{IDR}, $VOCS_{IDR}$, $MSK_{VOCS,}$ Ri, R_{CS}, and C that are either composed of random values or are protected and out of the adversary's reach. The only two parameters that are sent in clear text are the Alias identities and session ID. The Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The adversary will not be able to link the message to a particular device because the proposed protocol uses alias identities that change in every session. Therefore, the proposed protocol protects against eavesdropping attacks.

6.7.1.2.1.2.1.7 Impersonation attack

In this attack, the adversary eavesdrops on the messages transmitted from Vi to VOCS or vice versa, he/she can use the intercepted information and attempts to masquerade to conduct malicious actions, such as impersonating the vehicle or the VOCS and sending fabricated messages.

6.7.1.2.1.2.1.7.1 VOCS impersonation

According to the authentication phase in the proposed protocol, if an adversary wants to impersonate to be the VOCS, he/she needs to construct a message M2 < $VOCS_{IDA}$, $S_{ID,}$ TS_{Vi1}, CSY_2, CV, CSY_3>. The adversary needs to have MSKVOCS, VIDR, VTID, and the OTT, which are well-protected to construct this message. Also, due to PUF that is built into the VOCS and the characteristics of PUF, the adversary will not be able to generate a legitimate R_{CS} response. Therefore, the message sent by the adversary cannot pass the verification of V, so the vehicle will immediately terminate the session with the VOCS. Consequently, the proposed protocol can resist VOCS impersonation attacks.

6.7.1.2.1.2.1.7.2 Vehicle impersonation

To impersonate the vehicle, an adversary should intercept the messages exchanged in the previous sessions. This attack will fail for numerous reasons. First, the Alias ID of the vehicle changes every authentication session. Second, due to PUF that is built into the vehicle and the characteristics of PUF, the adversary will not be able to generate legitimate responses. Because the PUF responses are unique for each device, the adversary will not be able to replicate the response even if he/she has the challenge. Third, if an adversary wants to impersonate the vehicle, he/she needs to construct a message M1 < V_{IDA}, TS_{Vi1}, VX_2, VX_3 >. To construct M1, the adversary needs to have V_{CS1}, V_{CS2}, V_{IDR}, and OTT well-protected. Also, M3 < $V_{IDA,}$ TS_{Vi2}, $VX_{5,}$ $VX_{6,}$ $VX_{7,}$ VX_8>. To construct this message, the adversary needs to have Ri. The adversary will not be able to generate Ri due to the PUF characteristics. Furthermore, all messages are sent in an encrypted format using the XOR function and the one-way hash function. Thus, our protocol protects against the impersonation attack.

From another perspective, even if the adversary succeeded in compromising a vehicle, the adversary's further attacks using the compromised vehicle only affect the communication related to that vehicle. Because each vehicle has its own secret keys, the adversary can't derive other non-compromised vehicles' keys without knowing those vehicles' random information. Therefore, further attacks will not affect other communications. As a result, the proposed scheme is resistant to vehicle impersonation attacks.

6.7.1.2.1.2.1.8 Data modification attacks

Assuming that M1 < V_{IDA}, TS_{Vi1}, VX_2, VX_3 > has been tampered by an adversary M1* < V_{IDA}^*, TS_{Vi1}^*, VX_2^*, $_V X_3^*$ >. After receiving the message, the adversary will first need to have V_{CS1}, V_{CS2}, and OTTP, which are secure parameters to compute VX_1 < H (V_{CS2} || OTT)> and VX_2 < $VX_1 \oplus RV_{Vi1}$>. Furthermore, without having all secret parameters, the adversary will not be able to generate a legitimate VX3 < H (V_{IDR}|| V_{CS1}|| TS_{Vi1} || RV_{Vi1}|| OTT)) >. If the message has been tampered with, the vehicle is considered illegal, and the session will be terminated immediately by the VOCS.

Now suppose the message M2 < $VOCS_{IDA}$, S_{ID}, TS_{VOCS1}, CSY_2, CV, CSY_3> has been tampered by an adversary and sent to V M2* < $VOCS_{IDA}^*$, S_{ID}^*, TS_{VOCS}^*, CSY_2*, CV^*, CSY_3^* >. After receiving the message, the adversary will first need to compute V_{CS1} < H ($VOCS_{IDR}$ || MSK_{VOCS} || V_{IDR}||R_{CS} > and V_{CS2} <H (V_{CS1} || VT_{ID} > using its master secret key (MSK_{VOCS}), the real identity of the vehicle (V_{IDR}), the real identity of the VOCS ($VOCS_{IDR}$), the PUF response of VOCS(R_{CS}), and the vehicle type ID(VT_{ID}). Because these parameters are secret and are only known to the VOCS, the adversary will not correctly calculate V_{CS1} and V_{CS2}. Also, the adversary will not be able to generate R_{CS} and to get OTT. Consequently, it will be impossible to calculate a valid CV. Finally, without having all the above-mentioned secret parameters, the adversary will not compute CSY3 to pass the vehicle verification.

As for the message M3 < V_{IDA}, TS_{Vi2}, VX_5, VX_6, VX_7, VX_8> suppose the message M3 was tampered with by an adversary. Because each vehicle has a PUF chip and each chip is unique, the adversary will not be able to replicate or predict the response (Ri) of the real vehicle. Consequently, the adversary will not be able to generate R and compute

VX4. Therefore, the adversary will not be able to construct a valid VX8 and send a legitimate message to the VOCS that can pass the VOCS's vehicle authenticate. So, after receiving the message $M3^*$ and verifying it, VOCS will terminate the session with the vehicle. In summary, the proposed protocol can resist all message modification attacks.

6.7.1.2.1.2.1.9 Modeling attacks

It is an attack where the adversary collects a large number PUF CRPs and uses a regression algorithm to build a model and predict responses for new challenges. All the challenges and the responses are transferred in an encrypted format by using a one-way hash function. Therefore, the adversary will not collect enough useful CRPs to build a model and predict responses to new challenges.

6.7.1.2.1.2.1.10 Physical attack

According to the characteristics of PUF, any change or damage on the device with built-in PUF will render the PUF useless. Therefore, the physical attacks cannot obtain any useful information. We can conclude that the proposed protocol can ensure the physical security of the system.

6.7.1.2.1.2.1.11 Device counterfeit/cloning

Both the vehicle and the VOCS are authenticated using the PUF. PUF responses are treated as hardware fingerprints that cannot be duplicated or cloned. Therefore, by the use of the PUF, the vehicle and the VOCS cannot be cloned.

6.7.1.2.1.2.1.12 Session key agreement

At the end of the authentication phase, after confirming the identity of both the vehicle and the VOCS, each side will generate the session key $ssk = H(Ri || RV_{vi1}$

$RV_{CS1}||RV_{vi2}$) In summary, the session keys between V and VOCS is established in the proposed protocol.

6.7.1.2.1.2.2 Phase 2: Driver -VOCS mutual authentication

6.7.1.2.1.2.2.1 Mutual authentication

After completing the mutual authentication between V and VOCS completed, the mutual authentication between the user (U) and the VOCS will be started with the help of the V. User's authentication will use a three-factor authentication process by using the user's password and biometrics combined with the PUF of V. U and the VOCS authenticate each other during the mutual authentication by verifying the correctness D_{V3} and UVi. U will insert his/her username and password using the vehicle's touch screen and his/her fingerprint using the vehicle's fingerprint scanner. The collected data will be sent to the vehicle OBU through a secure channel. V will generate a timestamp, calculates U_1< (H (E_{ID}||Ri ||V_{IDR}|| OTT)>, and computes Z_{U1}<H (FP|| PUi>. Then, V calculates D_{V1} and D_{V2}. Then, V uses its shared secret key with the VOCS (*ssk*) to encrypt the identity and the verification message before sending it to the VOCS.

Once the VOCS receives the message, it will use the *ssk* to decrypt the received message. Then, VOCS will use the user ID and the vehicle ID to retrieve Ri, OTT, and V_{IDR} from its database, thereby calculating the verification message D_{V2}^{*} =< H (V_{IDR} ||TSvi1|| Z_{U1} || U_1)> to authenticate U's identity. Only the party with the secret values can retrieve ZU_1 from D_{v1}. Once the VOCS verify and authenticate the user, it will compute UVi < H (Ri|| TS_{Vocs1}|| Z_{U1}|| OTT)> and encrypt the verification message using its *ssk*

with the vehicle. When the vehicle receives the message, it will decrypt it using *ssk* and calculate the verification message $UVi^{*} < H (Ri|| TS_{Vocs1}|| Z_{U1}|| OTT)>$ to authenticate VOCS's identity. So, by verifying the correctness of UVi^{*}, U can confirm the identity of VOCS. Thus, U and VOCS complete the mutual authentication.

6.7.1.2.1.2.2.2 Resisting offline password guessing attacks

The password in the proposed protocol is included in $Z_{U1} = H (FP|| PUi)$ and $D_{V1} = Z_{U1} \oplus U_1$. $U_1< (H (E_{ID}||Ri || V_{IDR}||OTT)>$. The E_{ID} is a fixed secret parameter stored on the VOCS. The OTT is a secret parameter that is dynamically changing every authentication session. Furthermore, Ri, which is a PUF response, cannot be obtained and can't be predicted by the adversary due to the unique nature of PUF. Consequently, U_1 cannot be calculated to complete the rest of the computation process.

Moreover, the D_{V2} is protected by the *ssk* between the vehicle and the VOCS. In summary, the user's password has two shields of protection through the xor function and is encrypted by the *ssk*. Therefore, the adversary cannot guess the password of the user.

6.7.1.2.1.2.2.3 Anonymity unlinkability, and untraceability properties

Anonymity is essential to protect user privacy and security. The identity of the user needs to be hidden to avoid profiling. The proposed protocol implements strong anonymity with unlinkability to protect user's privacy. The user's real identity (E_{ID}) is transferred in hash and encrypted format using the *ssk*. Therefore, even if the adversary eavesdrops on all communication messages, it is impossible to obtain the user's EID or any other related data. Consequently, the proposed protocol provides anonymity and unlinkability, and the adversary cannot trace the user by intercepting messages.

6.7.1.2.1.2.2.4 Replay attack

The proposed protocol uses fresh values and TS each time in each message during the authentication process. The vehicle generates the timestamp TS and is then inserted in the transmitted message to ensure the adversary cannot replace it. Before trusting the freshness of the message, the receiver first verifies the timeliness of the timestamp. If the set threshold ΔT is exceeded, the receiver will reject the request and terminate the current session. If it is within ΔT, the receiver will calculate the verification message based on the received timestamp and the other message parameters to verify if it is equal to the received message. If they are not equal, the receiver will still reject the current session request. Therefore, if an adversary replays an intercepted message from the previous session, the other party will detect that the message is outdated and immediately terminate the session. Therefore, the protocol is protected from replay. On another side, each CRP and OTT are used only one time to ensure security against replay attacks. Hence, our protocol protects against replay attacks.

6.7.1.2.1.2.2.5 Eavesdropping attack

During the authentication phase, the adversary can intercept messages transmitted between Vi and the VOCS. All the intercepted messages will be useless because all of them are sent in an encrypted format using the shared secret key (*ssk*) between Vi and the VOCS. For the attacker to decrypt the message, he/she needs to guess the *ssk* that is fresh for every authentication session and is assumed to be protected and out of the adversary's reach. The only two parameters that are sent in clear text are the Alias identities and session ID. The Alias identities and the session ID do not pose any threat because this information is constructed from random parameters that change in every session. The

adversary will not be able to link the message to a particular user or vehicle because the proposed protocol uses alias identities that change in every session. Therefore, the proposed protocol protects against eavesdropping attacks.

6.7.1.2.1.2.2.6 Impersonation attack

In this attack, the adversary eavesdrops on the messages transmitted from Vi to VOCS or vice versa. He/she can use the intercepted information and attempts to masquerade to conduct malicious actions, such as impersonating the user, vehicle, or the VOCS and sending fabricated messages.

6.7.1.2.1.2.2.6.1 Driver impersonation

According to the authentication phase, if the adversary wants to impersonate to be the user (U), he/she needs to construct a message M1 =< V_{IDA}, $U_{I,}$ {TS_{Vi1}, D_{V1}, D_{V2}} $_{ssk}$ > to pass the VOCS's verification. The proposed protocol does not require storing a user's data either on a user's personal device or the vehicle. Because the adversary can't get the user's biometric or the user's password from the user side and can't get the Ri, OTT, or the V_{IDR} from the vehicle side, he/she will not be able to construct a valid U_1 = < H (E_{ID}||Ri || V_{IDR}||OTT)>, D_{V1} = < Z_{U1} $\oplus$ U_1>, and DV_2 < H (V_{IDR} || TSvi1|| Z_{U1} || U_1)>. Additionally, the adversary will not be able to predict the *ssk* that will be used to encrypt the user's authentication request message. Therefore, the adversary cannot impersonate the user.

6.7.1.2.1.2.2.6.2 VOCS impersonation

According to the authentication phase in the proposed protocol, if an adversary wants to impersonate to be the VOCS, he/she needs to construct a message M2 <

($VOCS_{IDA}$, TS_{Vocs1}, UVi} *ssk*) >. To construct this message, the adversary needs to have OTT, TS_{CS1}, Z_{U1}, and Ri, which are considered to be well protected. Also, the adversary needs to generate the PUF response (R_{CS}), which is considered to be impossible due to the nature of the PUF that can't be replicated or predicted. Furthermore, the adversary needs to know the *ssk* to encrypt the message before sending it out. Therefore, the message sent by the adversary cannot pass the verification of the user's vehicle, so the vehicle will immediately terminate the user's authentication session with the VOCS. Consequently, the proposed protocol can resist VOCS impersonation attacks.

6.7.1.2.1.2.2.6.3 Vehicle impersonation

Due to the PUF nature, the adversary can't obtain the response or even knows the "challenge" that the vehicle received from the VOCS during their authentication process. Also, if the adversary wants to impersonate the vehicle, he/she needs to know the OTT, the $V_{IDR,}$ and the *ssk* to encrypt the message before sending it to the VOCS. In fact, the adversary cannot derive the secret values, the session key (*ssk*) shared with Vi cannot be generated, nor can message M1 < V_{IDA}, $U_{I,}$ {TS_{Vi1}, D_{V1}, D_{V2}} $_{ssk}$ > be generated validly to communicate with VOCS. The protocol can resist impersonation attacks from the vehicle.

6.7.1.2.1.2.2.7 Data modification attacks

Assuming that M1 < V_{IDA}, $U_{I,}$ {TS_{Vi1}, D_{V1}, D_{V2}} $_{ssk}$ > has been tampered with an adversary M1* < V_{IDA}^{*}, U_{I}^{*} {, TS_{Vi1}^{*}, D_{V1}^{*}, D_{V2}^{*}} $_{ssk}$ >. After receiving the message, the adversary will first need to have the *ssk* to decrypt the message. Also, the adversary needs to know the OTT, Ri, $E_{ID,}$ and the $V_{IDR,}$ which are considered secure parameters to compute U_1 < H (E_{ID}||V_{IDR} || Ri || OTT)>. Additionally, the adversary needs to have the

user's username, password, R_{CS}, and fingerprint to construct Z_{U1} and $D_{V1.}$ Furthermore, without having all secret parameters, the adversary will not be able to generate a legitimate $D_{V2} < H (V_{IDR} || TSvi1|| Z_{U1} || U_1)>>$ that is, if the message has tampered with, the vehicle is considered to be illegal. The session will be terminated immediately by the VOCS.

Now suppose the message $M2 < VOCS_{IDA}, S_{ID}^{*}, TS_{Vocs1}, UVi\}_{ssk} >$ has been tampered with an adversary and sent to V $M2^{*} < VOCS_{IDA}^{*}, S_{ID}^{*}, TS_{Vocs1}^{*}, UVi^{*}\}_{ssk*} >$. After receiving the message, the adversary will first need to have *ssk* to decrypt the message. The *ssk* is assumed to be protected and secure. Then the adversary needs to compute $U_1 < H (E_{ID}||VI_{DR} || Ri || OTT)>$ Because these parameters are secret and are only known to VOCS and the vehicle, the adversary will not correctly calculate U_1. Also, it is impossible to replicate the user's fingerprint. Finally, without having all the above-mentioned secret parameters, the adversary will not be able to compute UVi to pass the vehicle verification. In summary, the proposed protocol can resist all message modification attacks.

6.7.1.2.1.2.2.8 Biometric privacy protection

In the proposed protocol, the user biometric samples are processed on the cloud side. There is no user biometric data stored or maintained on any user-related device or on the vehicle. The biometric templates of the continuous biometric are stored on the VOCS encrypted by Z_{U1}. The user biometric is transferred in encrypted format using both the xor function and the *ssk* between the Vi and the VOCS. This information does not endanger biometric privacy, even if it is leaked. Therefore, the proposed protocol supports biometric privacy protection.

6.7.1.2.1.2.2.9 Three-factor security

The proposed protocol employs three factors to achieve user authentication of the user: biometric, password, and PUF response. Additionally, the proposed protocol employs continuous biometric authentication that is considered another layer of ongoing user authentication.

6.7.1.2.2 Comparison of security features

This section compares the proposed scheme security features with other related authentication schemes from the literature [156],[155],[154],[150], and [157] Table 30 summarizes the comparison results of the proposed protocol with existing works against the imperative security threats and accomplishes diverse security features. From table 30, we can observe that the presented schemes in the literature satisfy most of the security properties. However, none of them protects against vehicle sensors counterfeiting. All of them depend on using CRPs that are generated outside the manufacturing process, so it does not provide any means to verify the originality of the sensors. Also, protocol [150] use the same alias ID for both the vehicle and RSU in every authentication session. This, in turn, can lead to vehicle traceability and linkability.

Furthermore, most of the proposed protocols [156, 155, 150, 157] implement computationally expensive cryptographic solutions such as elliptic curve cryptographic system (ECC) and fuzzy extractors to satisfy the security properties. Furthermore, the protocols [156,155, 150] depend on mobile devices and smart cards to authenticate users. The use of smart cards makes the user's sensitive data subject to being stolen or subject

to physical attacks that can corrupt the whole system. Moreover, none of the presented protocols introduced using continuous biometrics to regularly verify the user's identity to establish ongoing trust and ensure that the received data is coming from the same user initially authenticated.

Table 30: Security feature comparison of the proposed scheme with other related mutual authentication and key agreement schemes.

Security Property	**Proposed V-TA**	**Proposed V-D-VOCS**	**[156]**	**[155]**	**[154]**	**[150]**	**[157]**
Mutual Authentication	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Replay attack	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Perfect forward/backward secrecy	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Anonymity and untraceability	Yes	Yes	Yes	Yes	Yes	Yes	No
Resisting offline password guessing attacks	Yes	Yes	Yes	Yes	NA	Yes	NA
Resisting Data Modification attacks	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Resilience against TA/VOCS impersonation attacks	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Resilience against	Yes	Yes	Yes	Yes	Yes	Yes	No

vehicle impersonation attacks							
Resilience against User impersonation attacks	Yes	Yes	Yes	Yes	Yes	NA	NA
Physical and cloning attacks	Yes	Yes	Yes	No	Yes	Yes	Yes
Biometric privacy protection	Yes	Yes	Yes	Yes	Yes	Na	Na
Resisting Vehicle counterfeiting	Yes	Yes	No	No	No	No	No
Session key agreement	Yes	Yes	Yes	Yes	Yes	Yes	Yes

As presented in table 30, the proposed scheme supports all the essential security features: First, the proposed scheme does not require any secret key storage or secret key sharing over the network. Second, all secret keys are computed locally on all the communicating parties. Third, in the proposed scheme, the vehicles use their one-time alias identity for each authentication session. Therefore, it will be difficult for an outside adversary to comprehend the activities of the vehicle. Fourth, the proposed scheme uses the PUF to protect the vehicles and VOCS against counterfeiting. Fifth, the proposed scheme uses both static and continuous biometric for user authentication to ensure that the received data belongs to the correct driver during the entire session.

From table 30, we can conclude that the proposed scheme can support all the desired security properties, which are essential for IoV ecosystem security.

6.7.2 Performance Analysis

In this section, we present a performance analysis of the proposed scheme. The performance analysis evaluates the storage requirements. Also, we analyze the presented scheme's overhead and efficiency in terms of computational complexity and communication cost and compare its computational complexity and communication cost with the most relevant protocols in the literature proposed by [150], [154], [155] [156], and [157].

6.7.2.1 Storage requirements

Table 31: Storage cost of our scheme

Node	**Storage cost (in bits)**
V	128 + 8 * 3 +256 * 6= **1688 b**
TA	128 * 2 + 3 * 8 +2 * 256 = **792 b**
VOCS	128 * 3 + 8 * 5 + 160 + 256 * 2 = **1096 b**

In the proposed scheme, each vehicle must store its real identity V_{IDR} alias identity V_{IDA}, alias identities of $VOCS_{IDA}$ and TA_{IDA}, authentication parameters V_{TA1}, VCS_1, and V_{CS23}, Chained hash CHX, one-time password OTP, and one-time token (OTT). We use SHA-1 and SHA-2 as two examples of hash function, and the output of SHA-1 is 128 or 160 bits and SHA-2 is 256 bits. By applying these settings, we obtain $|V_{IDA}| = 128$, $|VOCS_{IDA}| = |TA_{IDA}| = |V_{IDR}| = 8$ bits, and $|OTP| = |OTT| = |V_{TA1}| = V_{CS1}| = |V_{CS2}|$ CHX = 256 bits. Meanwhile, TA is required to store the tuple V_{IDA}, V_{IDR},

TA_{IDA}, TA_{IDR}, OTP, and CHX. By applying these settings, we obtain $|V_{IDA}|$ = 128 bits, $|TA_{IDA}| = |V_{IDR}| = |TA_{IDR}|$ = 8 bits, |OTP| = |CHX| = 256 bits, and C_{TA}=128 bit. Finally, VOCS is required to store the tuple V_{IDA}, V_{IDR}, $VOCS_{IDA}$, $VOCS_{IDR}$, E_{ID}, $T_{ID,}$ $U_{i,}$ $OTT_{,}$ $_{ZU2}$ and,BP_{Ui}. By applying these settings, we obtain $|V_{IDA}| = |Ui|$ = 128 bits, $|VOCS_{IDA}| = |VOCS_{IDR}| = |V_{IDR}| = |E_{ID}| = |T_{ID}|$ = 8 bits, $|BP_{Ui}|$ = 160 bits, C_{cs} = 128 bits, and |OTT| = $|Z_{U2}|$ = 256 bits.

6.7.2.2 Computational complexity analysis

We compare the computation cost of the proposed scheme with other related schemes [150], [154], [155], [156], and [157]. we do not consider XOR operations for computational cost analysis because the time for executing a bitwise XOR operation is negligible. To analyze the performance of the proposed scheme with respect to other presented protocols in the literature, we conduct simulations of the cryptographic operations using dedicated hardware device including laptop computer and Raspberry Pi. Dell Inspiron Laptop with Intel Core i7, dual-core 2.7 GHz CPU, and 8 GB RAM to act as the server. We used a Raspberry Pi 4 Model B with 64-bit quad-core cortex A-72 processor and 1 GB RAM to simulate the vehicle. We used PyCrytodome cryptographic and Fastecdsa libraries in Python 3.6. For these results, we considered the 128-bit arbiter PUF for PUF operation. The fuzzy extraction's execution time is almost the same as the ECC point multiplication and the execution time for modular exponentiation is double the time of the execution of the ECC point multiplication [58]. Table 32 presents the used notation and the execution time of each operation.

Table 32: Execution Time of the cryptographic operations

Operation	**Computation Time on IoT**	**Computation Time on server**
H: time for executing a one-way hash function SHA (256)	0.002 ms	.001ms
F: time for executing a fuzzy extractor	5 ms	4 ms
EM: time for executing an ECC point multiplication	5 ms	4 ms
EA: time for executing an ECC point addition	5.5 ms	5 ms
EXP: time for a modular exponentiation	10 ms	8 ms
HMAC: time for executing the HMAC	2.7 ms	1.5 ms
ENC: Time for Executing (AES-CBC Encryption)	0.18 ms	.14 ms
DEC: Time for executing (AES-CBC Decryption)	0.18 ms	.14 ms

PUF: Time for executing PUF (128-bit Arbiter)	.12 ms	.12 ms

Table 33: Comparison of computation costs for the authentication phase of the proposed scheme and other related schemes

Entity	[150]	[154]	[155]	[156]	[157]	Proposed [V-TA]	Proposed [V-D-VOCS]
Vehicle	$2T_{PUF}$ + $13T_h$ + $2T_F$ + $1T_{ENC}$ + $1T_{DEC}$ + $1T_{EM}$	$2T_h$ + $1T_{ENC}$	$4T_h$ + $1T_{ENC}$	$1T_{PUF}$ + $12T_h$ + $1T_{EM}$	$1T_{EM}$+ $1T_{EA}$ + $2T_h$	$12T_h$ + $1T_{PUF}$	$14T_h$ + $2T_{PUF}$
Cost	2*.06 + 13*.002 +2*5+1*.18+1* .18 + 1*10 = **20.51 ms**	2*.002 + 1*.18 = **.182 ms**	4*.002 +1*.18 = **.188 ms**	1*.06+12 *.002+1* 5 = **5.08 ms**	1* 5 + 1*5.5 +2*.002= **10.50 ms**	12*.002 +.06 = **.086 ms**	14 * .002 +.06 *2 = **.148 ms**
Server	$9T_h$ + $2T_F$ + $1T_{ENC}$ + $1T_{DEC}$ + $2T_{EM}$	$4T_h$ + + $2T_{ENC}$ + $1T_{DEC}$	$5T_{EM}$ + $1T_{EA+}$ $1T_{ENC}$ + $1T_{DEC}$ + $12T_h$	$19T_h$ +$1T_{EM}$	$1T_{EM}$+ $1T_{EA}$ +$2T_h$	$13T_h$ + $1T_{PUF}$	$20T_h$ + $2T_{PUF}$ + $1T_{SE}$ + $1T_{SD}$
Cost	9 *.001 +2*4+1* .14+1*.14 +2*8 = **24.89 ms**	4 *.001 + 2 *.14 + 1*.14= **.424 ms**	5*4 + 1*5 + 1*.14+ 1*.14 +12*.001 = **25.29 ms**	19 *.001 +1*4= **4.019 ms**	1 * 4 + 1* 5 + 2*.001 = **9.002 ms**	13*.001 +.06= **.073 ms**	20*.001 + .06 *2 +.14+.14 = **.42 ms**
User	---------	---------	$5T_{EM}$ + $1T_{EA}$+ $1T_{ENC}$ + $9T_h$	$1T_{PUF}$ + $13T_h$ +$2T_{EM}$ +$1T_{ENC}$	-------	---------	$3T_h$ +$1T_{ENC}$ + $1T_{DEC}$
Cost	---------	--------	5*5 + 1*5.5 + 1*.18+	1*.06 + 13*.002 +2*5+1*	-------	--------	.002*3 + .18 + .18= **.37 ms**

			9*.002 = **30.70 ms**	.18= **10.27 ms**			
Total Cost	**44.80 ms**	**.606 ms**	**56.18 ms**	**19.37 MS**	**19.50 ms**	**.159ms**	**.94 ms**

Table 33 summarizes the computational complexity cost comparison between the proposed scheme and the other related vehicular networks schemes presented in the literature. The total run time of the V-TA scheme is .159 ms and the run time of V-D-VOCS is .94 ms. The results of the proposed scheme with other related protocols indicate that the V-TA protocol is more efficient than the other protocols. The run time of the V-D-VOCS is higher than [154] because it involves the authentication of the driver. The protocol in [155] performs the worst because it makes use of multiple ECC point multiplication and point addition which are computationally heavy. The proposed scheme is more efficient than [150] that used scalar multiplication and asymmetric encryption. Also, protocols [155],[156], and [157] used ECC, which increased the run time. The main strengths of my proposed scheme are the avoidance of using long-term secret keys and securing the communicating devices from physical attacks and counterfeiting by employing the PUF and the chained hash PUF.

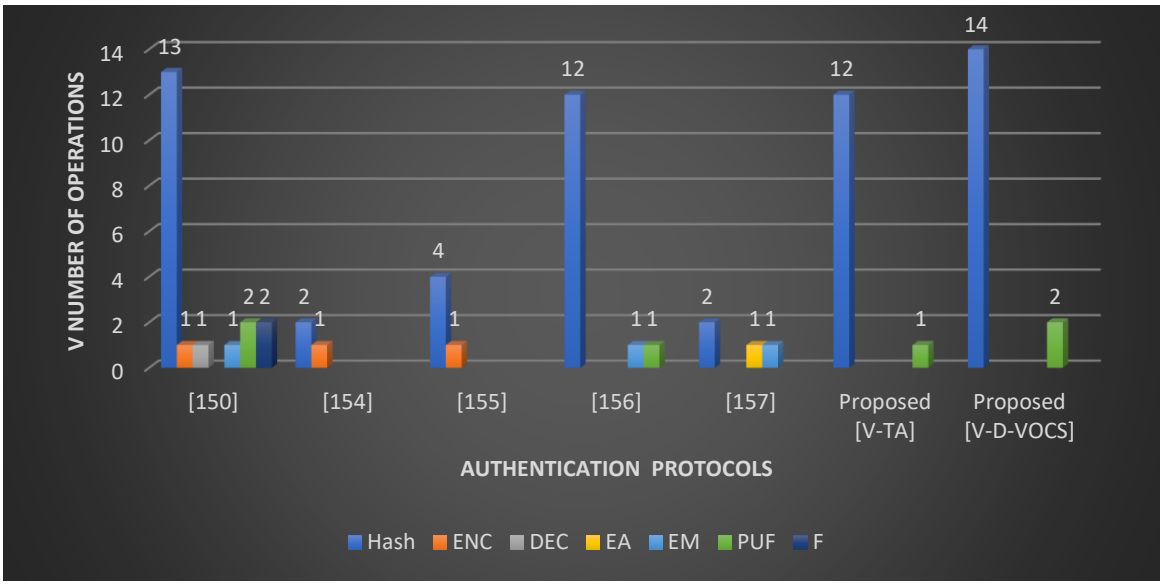


Figure 115: Comparison of the number of V crypto operations Across the schemes

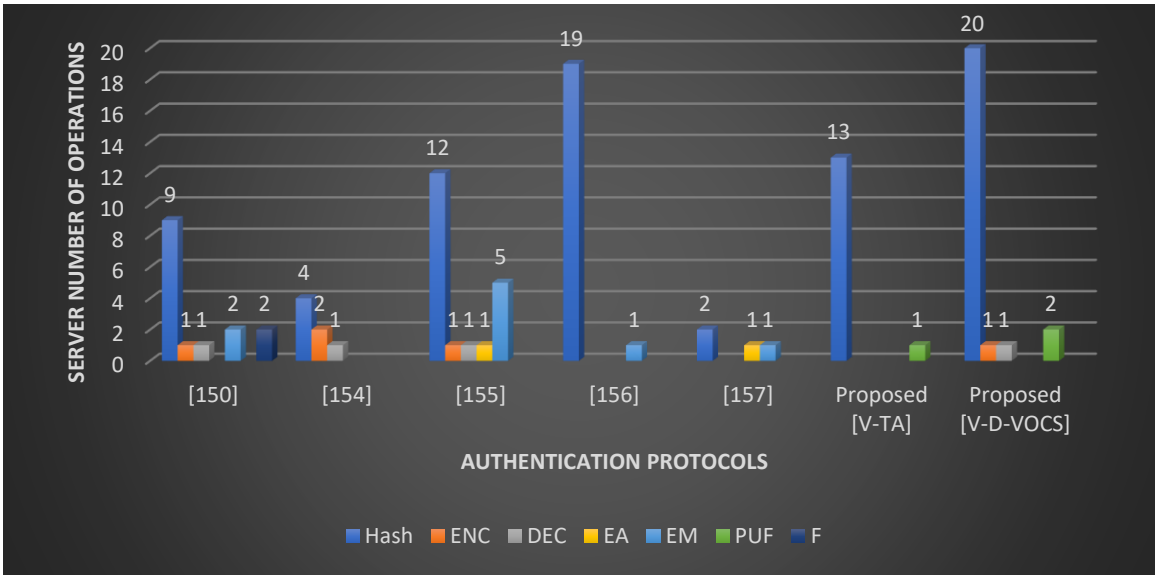


Figure 116: Comparison of the number of the server crypto operations across the schemes

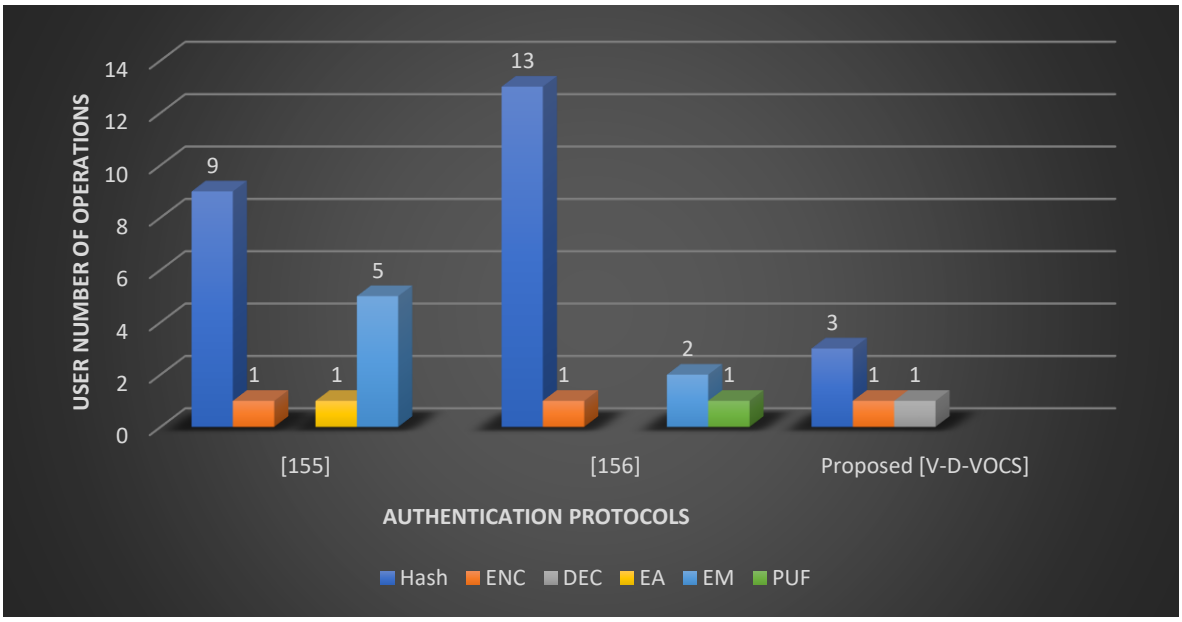


Figure 117 : Comparison of the number of the user crypto operations across the schemes

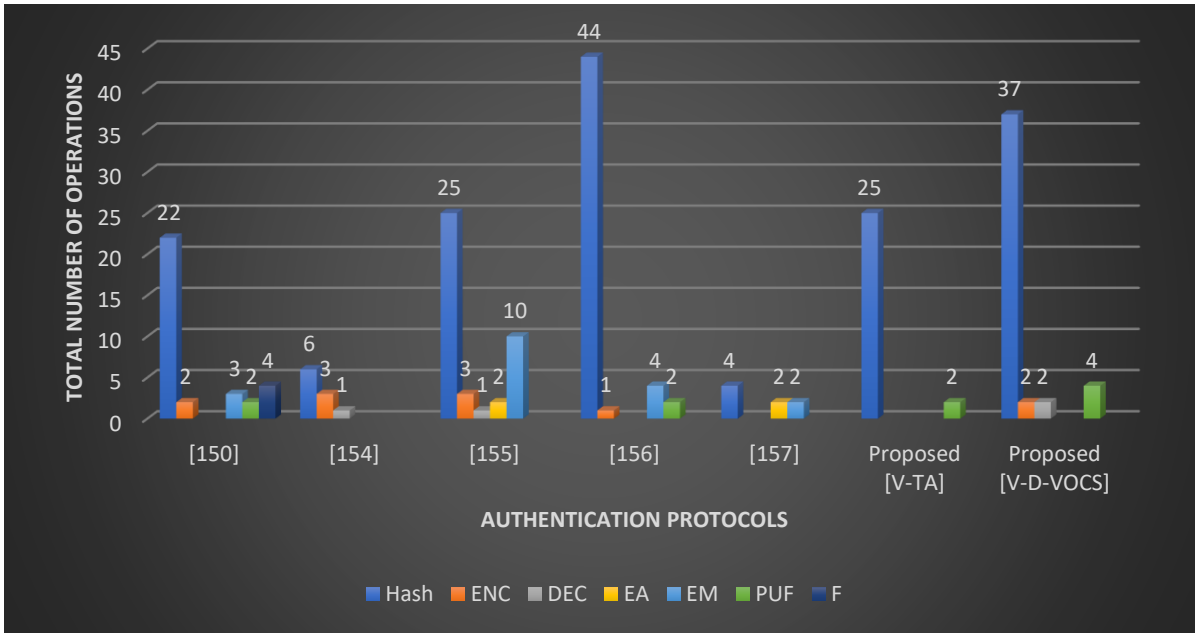
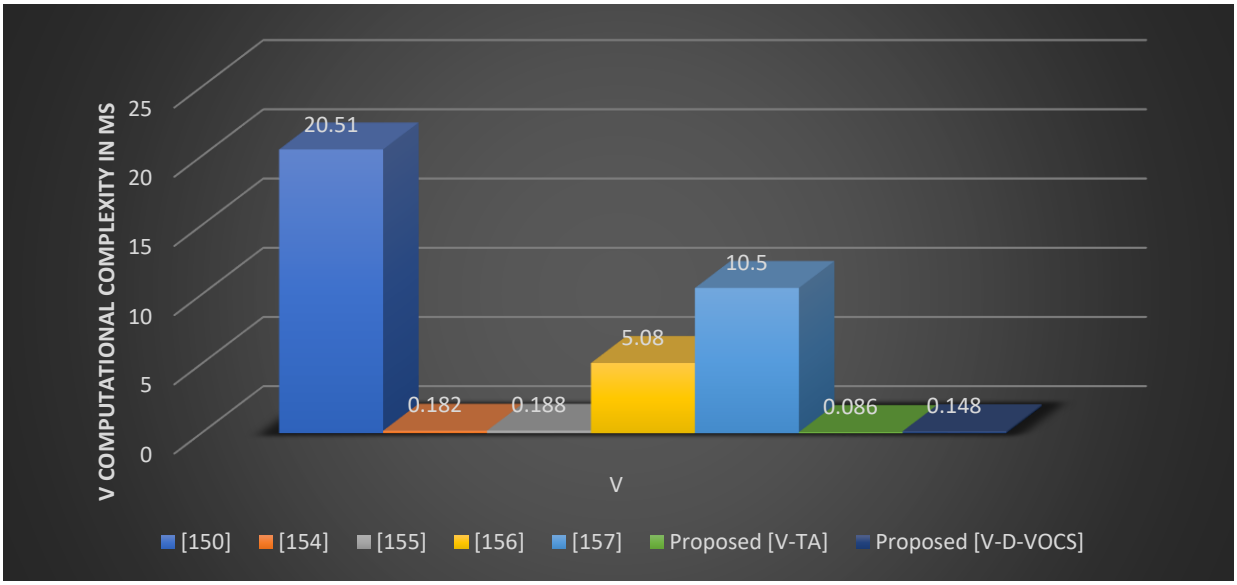


Figure 118: Comparison of the total number of crypto operations across the schemes

Figure 115 displays the number of authentication operations of each type that are completed on the V side. Figure 116 depicts the number of authentication operations of each type that are completed on the server-side. Figure 117 depicts the number of authentication operations of each type that are completed on the user side. Figure 118 demonstrates the total number of authentication operations of each type that are complete

for each scheme. As depicted from the table and the figures, our proposed scheme used more hash functions than most of the other schemes to avoid computationally expensive cryptographic operations such as ECC and HMAC and, at the same time, achieve the same security goals. The proposed scheme is the only protocol that used PUF as a hardware fingerprint on the server-side to secure the devices from counterfeiting and use the PUF response as an encryption key to avoiding storing long-term security keys on the communicating devices. Furthermore, using PUF on the server-side can protect the scheme from server impersonation and secure the server database to be stolen if the device got compromised.

Figure 119: V computational time across the different schemes



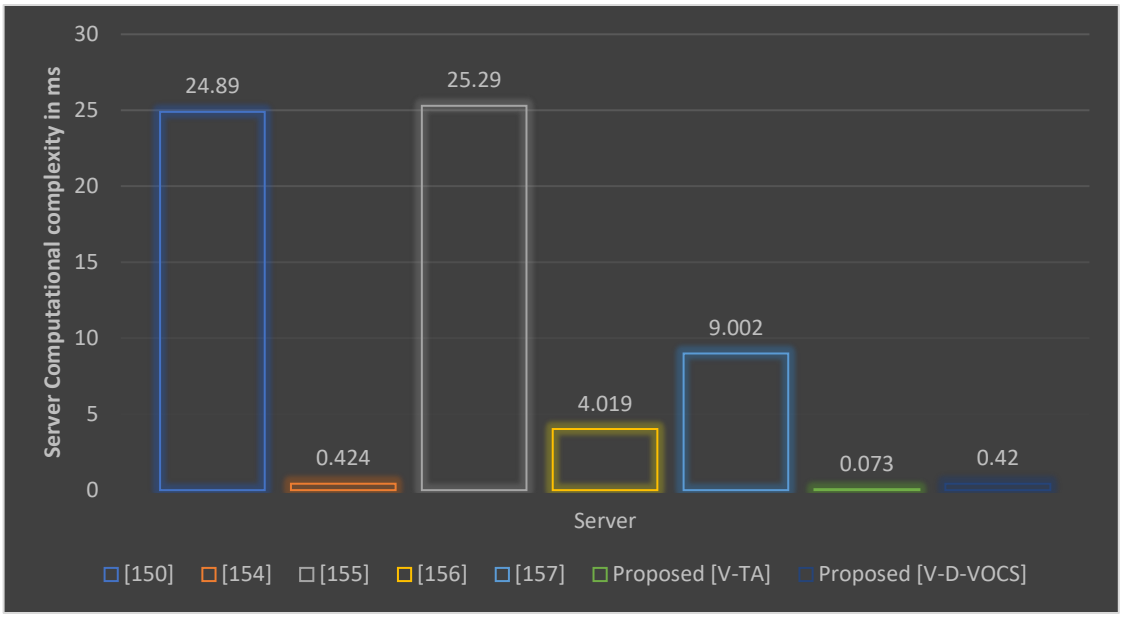


Figure 120: Server computational time across the different schemes

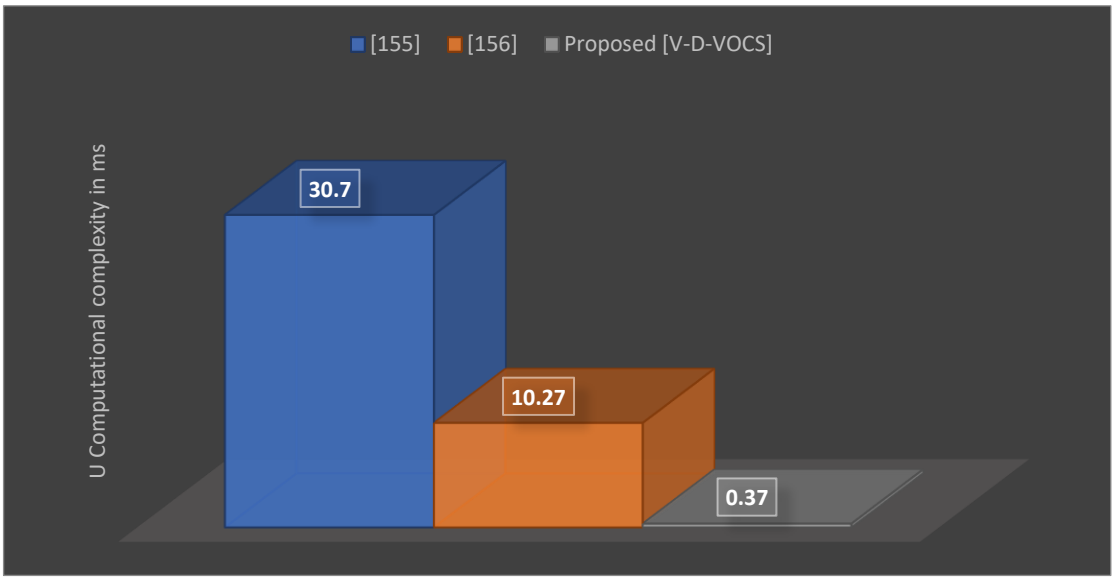


Figure 121 : U computational time across the different schemes

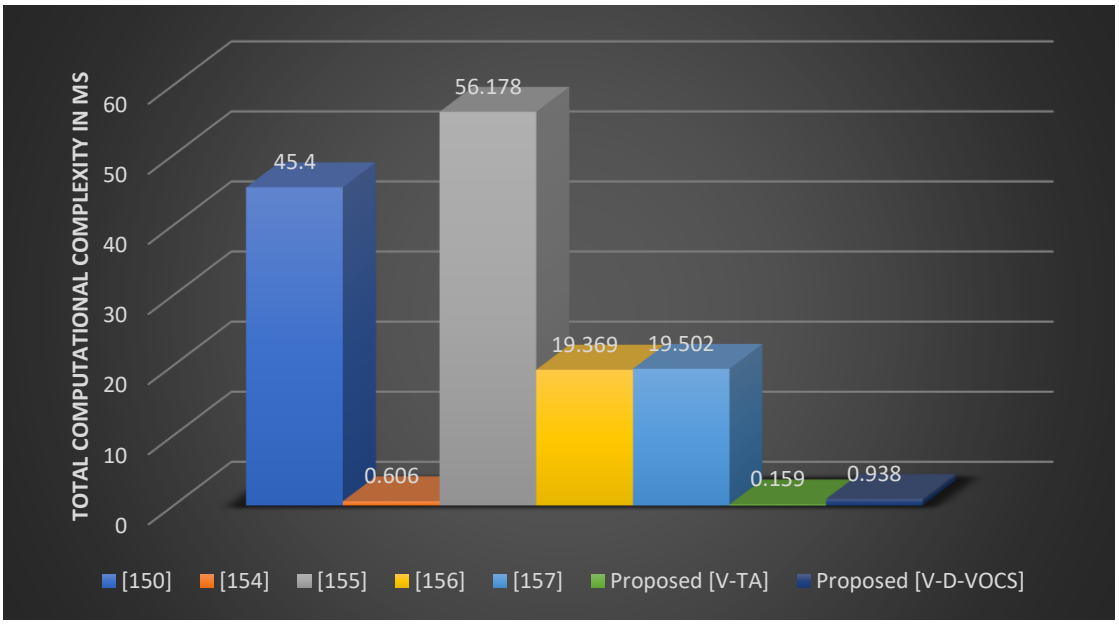


Figure 122: Total computational time across the different schemes

Figures 119, 120, 121, and 122 demonstrate the cost comparison of my authentication scheme with other similar schemes. Figure 122 presents the total computational cost of each scheme. Figure 119 displays the computation cost of each scheme for the V side, figure 120 depicts the computational cost of each scheme for the server-side, and figure 121 depicts the computational cost of each scheme for the user side. The results indicate that the V-TA and V-D-VOCS protocols have a lower computational cost for the V side compared to the rest of the protocols. On the server-side, although the V-D-VOCS protocol has almost the same computational cost as [154], our proposed protocol includes user authentication. Also, it preserves both the driver and the vehicle privacy by using different alias IDs for every authentication session. The V-D-VOCS protocol has the lowest computational cost on the user side because it depends on the hash function and the PUF instead of expensive cryptographic approaches such as ECC and modular exponentiation. The above results prove that the proposed scheme is suitable for IoV domain.

We can conclude that the proposed scheme has a higher security level than the rest of the protocols. The proposed scheme achieves the main security goals of confidentiality, integrity, and authenticity besides accommodating IoV devices that are diverse in their capabilities.

6.7.2.3 Communicational Cost

The following section analyzes the communication cost of the proposed scheme. To reduce network congestion and to provide fast message transmission, the communication cost of the scheme should be as low as possible. For the communication cost analysis, we evaluate the communication cost in terms of the size of the message in bits. Then, we compared our proposed scheme to the other related schemes. Table 34 presents a summary of the sizes of the message parameters. Table 35 lists the message parameters that are communicated among the communicating parties, along with their sizes.

Table 34: Main parameters of the messages and their sizes in bits

Message Parameters	**Size in Bits**
ID of N [[57], [62], [63]	128
ID of server [63]	8
S_{ID} [57], [62]	8
Nonce [61], [63]	128
CRP (C, R) [61], [62]	128

HMAC [63]	256
Hash Function [57], [62]	256
Timestamp (TS) [63]	48
ECC [60], [59]	256
MAC [62]	256

V-TA protocol

- **Message 1:** In the transmission (V → TA), V sends the tuple, V_{IDA}, TS_{vi1}, X_1, X_2 Therefore, the size of this tuple is 128 + 48 + 128+256 = **560. bits**.
- **Message 2**: In the transmission (TA → V), TA sends the tuple, $TA_{IDA,}$ S_{ID}, $TS_{TA1,}$ Y_2, C', $Y_{3.}$ Therefore, the size of this tuple is 8 +8 + 48 + 128+128+256 = **576 bits**.
- **Message 3**: In the transmission (V → TA), V sends the tuple, V_{IDA}, S_{ID}. $TS_{VI2,}$ X_3, $X_{4.}$ Therefore, the size of this tuple is 128 + 8 + 48+128 +256 = **568 bits**.
- **Message 4**: In the transmission (TA → V), TA sends the tuple, $TA_{IDA,}$ S_{iD} TS_{TA2}, $Y_{5,}$ $Y_{6.}$ Therefore, the size of this tuple is 128 + 8 + 48+128 +256 = **568. bits**.

V-D-VOCS protocol

- **Message 1:** In the transmission (V → VOCS), V sends the tuple, $V_{IDA,}$ $TSvi_{1,}$ $VX_{2,}$ $VX_{3.}$ Therefore, the size of this tuple is 128 + 48 + 128+256 = **560. bits**.
- **Message 2**: In the transmission (VOCS → V), TA sends the tuple, $VOCS_{IDA}$, $S_{ID,}$ $TS_{CS1,}$ CSY_2, CV, CSY_3 Therefore, the size of this tuple is 8 +8 + 48 + 128+128+256 = **576 bits**.

- **Message 3**: In the transmission (V → VOCS), V sends the tuple, $V_{IDA,}$ TS_{Vi2}, $VX_{5,}$ $VX_{6,}$ $VX_{7,}$ $VX_{8.}$ Therefore, the size of this tuple is 128+48+ 128+ 128+128+256 = **816. bits**.

Table 35: Communication cost comparison (bits)

Message Number	**[150]**	**[154]**	**[155]**	**[156]**	**[157]**	**V-TA Proposed scheme**	**V-D-VOCS Proposed scheme**
M_1:	384 bits	256 bits	512 bits	1072 bits	816 bits	560 bits	560 bits
M_2:	384 bits	512 bits	384 bits	1072 bits	816 bits	576 bits	576 bits
M_3:	384 bits	392 bits	384 bits	816 bits	768 bits	568 bits	824 bits
M_4:	512 bits	384 bits	768 bits	816 bits	----	448 bits	256 bits
M_5:	-------	392 bits		----	---		136 bits
M6:	-------	512 bits					
M7:	-------	256 bits					
Total	**1664 bits**	**2704 bits**	**2048 bits**	**3776 bits**	**1792 bits**	**2152 bits**	**2352 bits**

The total communication costs of [150], [154], [155],[156], and [157] are 1664,2704,2048,3776, and 1792 bits, respectively. The total communication cost of the proposed V-TA protocol is 2152 and 2352 for V-D-VOCS scheme. My two proposed protocols are less than [154 and 156]. My proposed protocols have lower communication

cost and, at the same time, achieved all the required security goals. Also, the proposed schemes have proper communication cost compared to [150], [155], and [157]. The V-TA protocol requires sending only four messages to achieve mutual authentication. The messages contain timestamps, XOR functions, ciphered random numbers, and hash functions that prevent attacks such as data modification, replay, and impersonation. The V-D-VOCS protocol requires three messages to achieve mutual authentication between the vehicle and the VOCS and two messages to achieve mutual authentication between the driver and the VOCS. Even if the protocols of my proposed scheme are not the protocols with the lowest communication cost, they have a reasonable cost for IoV, and at the same time, they achieve more security properties than the protocols with less cost.

6.8 Summary

In this chapter, we designed a "Multi-Factor Authentication, and Privacy Preservation Protocol Using Biometrics and Chained Hash PUF for IoV'' to solve security threats of the existing authentication schemes. The protocol enables a lightweight mutual authentication and an establishment of secret keys, which can be used to secure the exchanged messages between the communicating parties. The lightweight and the privacy-preserving properties are ensured as they are the essential characteristics of dynamic entities. Those properties are achieved through the use of PUF, chained hash PUF, hash functions, and XOR operations. The proposed protocol is lightweight with a lower computation cost and execution time compared with the competitive schemes. We completed an in-depth security analysis against different strong adversarial attacks and found that the proposed protocol is resistant to these types of attacks. The lightweight

nature allows easy implementation of the protocol in different entities of the IoV domain such as the vehicle's OBU, RSUs, infrastructure and sensors.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

The Internet of Things (IoT) plays an essential role in all aspects of our daily lives. It benefits different fields, including smart homes, healthcare, smart cities, agriculture and others. The IoT consists of billions of connected devices over the internet that are able to gather and exchange data using IoT nodes and controllers.

Adversaries are shifting their attention from traditional computers to IoT devices for malignant activities like exposing smart homeowner private information and/or to launch botnet attacks. Therefore, it is very critical to move fast to address the rising security and privacy concerns in IoT systems before severe disasters happen. Similar to traditional networks, the security of IoT networks depends mainly on how properly the authentication process is done and on how user's privacy is preserved. However, the IoT infrastructure faces challenges in implementing and operating strong authentication schemes because of the resource constrained nature of the IoT devices that have limited computational and storage capabilities.

The first contribution of this work is the introduction of PUF Hierarchal Distributed Architecture (PHDA). The main goals of PHDA are to support system scalability and to protect the system's devices from being counterfeited. The PHDA acts as a device name resolution to store and retrieve the Challenge-Response Pairs (CRPs) of

the IoT devices. PHDA stores the PUF data using a 3-tier architecture employing a simple naming scheme.

The second contribution of this work is the introduction of lightweight mutual authentication and privacy preservation protocol using PUF for smart home network model. The proposed protocol is based on using PUF as a hardware fingerprint to authenticate the communicating devices. Also, another aspect of the securing the system depends on employing network segregation based on the IoT device type to mitigate the threats. Moreover, the security and privacy preservation of the proposed protocol is implemented using dynamic identities and temporary secret session keys that change in every session and are exchanged in an unlinkable and untraceable manner.

The third contribution is the introduction of multi-layer distributed lightweight mutual authentication using chained hash PUF scheme for IIoT. A smart poultry farm is introduced as a network model. The design of the proposed scheme fits the dynamic and distributed cyber-physical nature of the poultry farms to ensure the system's security. The proposed scheme enables a M2M communication, which provides the chance for devices inside the IIoT network to do data offloading and allows the IoT devices that are away from their gateways to get authenticated and communicate with their gateways through other IoT devices. Access control solution is integrated to support smart poultry farms' dynamic nature. Therefore, an automated access control model based on virtual domain segregation of IoT network and device type is introduced. Also, the chained hash PUF value is used to create a type of tracking system between the IoT node and its gateway. The proposed scheme proves that it provides confidentiality, integrity, anonymity,

unlinkability, and untraceability properties while achieving mutual authentication between the communicating devices.

The fourth contribution of this work is the presentation of three factor mutual authentication and Privacy Preservation Scheme using user and device biometrics scheme for IoV System. The proposed scheme employed user's biometric to achieve static and continuous driver's authentication process to ensure that the received data belongs to correct driver and identifies the fabricated data. The proposed scheme introduces employing central RSUs (CRSUs) between the RSUs and the TA to reduce the number of times a vehicle needs to authenticate with the RSUs. Furthermore, the proposed scheme uses PUFs and chained hash PUF to perform authentication via a challenge-response mechanism. The proposed scheme can perform authentication and establish a shared secret session key without the need to store any long-term keys on the vehicle's OBU. The results prove that the proposed scheme is not only secure against different types of attacks but is also efficient enough for IoV.

The PUF as a hardware security mechanism is used for key authentication and for encrypting the data on the server side where the key can be generated on the fly and does not need be stored. The two main advantages of this approach are: (1) the elimination of the need to store long term secret keys; and (2) the hardness of predicting the keys due of the unclonable and unpredictable nature derived from the PUFs. Also, the three proposed schemes eliminate the need for key exchanges between device and the server over the network because the devices generate the key locally using the securely shared random values and the PUF challenges and responses.

Additionally, through an intensive formal and informal security analysis of our protocols using the BAN logic and AVISPA tool, the results indicate that the proposed schemes are resilient against well- known attacks. The proposed schemes achieved the key security properties such anonymity, unlinkability, untracability and system scalability with a limited performance overhead. Also, we conduct performance analysis to evaluate the computational complexity and the computational cost and compare them with other proposed protocols in the literature. The results prove that the proposed schemes are in general more efficient than recently proposed protocols.

7.2 Future Work

We plan to extend the proposed scheme titled “Lightweight Privacy Preservation and Mutual Authentication Scheme for Smart Homes Using Physical Unclonable Functions”, to authenticate mobile devices and users. Also, We plan to extend the proposed scheme to account for node-to-node connectivity that can take place within the same virtual domain (intradomain) or across different domains(interdomains). Furthermore, our extended study will explore vulnerabilities resulted from the new form of connectivity that involves node-to-node communication.

For the second scheme titled “M2M Distributed Multi-Layer Lightweight Mutual Authentication and Key Agreement Scheme Using Chained Hash PUF in Industrial IoT System”, we plan to enable an IoT device in one home network to communicate with an IoT device in another home network regardless of the underlying communication protocols. Also, we plan to extend the proposed scheme to facilitate in farm and cross farm authentication and communication between the IoT sensors on one side and the farmer/operators on the other side by examining the kind of operations the

farmer/operator need to complete that may require single level or multiple level access control based on the risk factor associated with the operation. Furthermore, in many farms the animals have sensors embedded, which require appropriate authorized access. These wearable and health monitoring devices are attached to livestock and collect sensitive data, which can be used by adversaries to control the animal or to send false data about the animal. we plan to investigate how to conduct continuous authentication between the animal and the control unit in order to implement an ongoing verification of the animals and to identify the fabricated data.

For the third scheme titled “Three-Factor Authentication and Privacy Preservation Scheme Using User and Device Biometrics for IoV System”, we plan to extend the work to implement machine learning based IDS to detect fake messages that are sent by authenticated malicious vehicles to alert authorities to revoke the shared secret keys and alert other vehicles. Also, we plan to explore more vehicular networking attacks and analyze them with the proposed scheme.

REFERENCES

[1] NIST, "NISTIR 8259," [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8259-draft.pdf.

[2] K.Ashton., "That‘internet ofthings' thing.," *RFID Journal,* pp. ,22(7): 97–114,, 2009.

[3] Norton, "The future of IoT: 10 predictions about the Internet of Things," 2011. [Online]. Available: https://us.norton.com/internetsecurity-iot-5-predictions-for-the-future-of-iot.html. [Accessed 10 Jan 2020].

[4] G. E. Moor, "Cramming More Components onto Integrated Circuits," *Electronics,* vol. 86, no. 1, pp. 114-117, 1998.

[5] S. B. M. S. a. H. W. ”. 2. J. G. Koomey, "Assesing Trends In The Elecritcal Efficiency Computation Over Time," IEEE Annals of the History of Computing, 2009.

[6] K. Ashtons, Interviewee, *The Internet of Thing.* [Interview]. 19 June 2014.

[7] *A Definition of the Internet of Things (IoT),* http://iot.ieee.org/definition.html.

[8] I. T. S. Sector, "Internet of Things Global Standards Initiative," 2012. [Online]. Available: http://handle.itu.int/11.1002/1000/11559-en?locatt=format:pdf&auth. [Accessed 5 Jan 2020].

[9] Cisco. [Online]. Available: http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoE_Economy.pdf.

[10] M. &. Company. [Online]. Available: https://www.mckinsey.com/industries/semiconductors/our-insights/whats-new-with-the-internet-of-things.

[11] H. J. S. [2] Van Tilborg, "Encyclopedia of Cryptography and Security," in *Science & Business Media*, Springer, 2014.

[12] A. Majeed, "Internet of Things (IoT): A verification framework," in *IEEE 7th Annu. Comput. Commun. Work. Conf*, 2017.

[13] C. M. T. Lead, *IoT Spending Projected To Pass $1 Trillion In 2022.*

[14] R. Anderson, Security engineering, Wiley, 2008.

[15] G. J. C.Kolias, "Ddos in the iot: Mirai and other botnets," *Computer,* vol. 50, no. 7, pp. 80-84, 2017.

[16] M. A. N. T. T. B. a. S. A. N. Abbas, "A mechanism for securing iot-enabled applications at the fog layer," *Journal of Sensor and Actuator Networks,* vol. 8, pp. 1-16, 2019.

[17] M.Abomharaetal, "Cyber security and the interne tof things: vulnerabilities,threats, intruders and attacks," *Journal of Cyber Security and Mobility, vol. 4, no. 1,* p. 65–88, 2015.

[18] W. a. B. L. Stallings, Computer Security: Principles and Practice, Pearson, 2015.

[19] D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, New York,: Springer, 2003.

[20] C. Paar and J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners, Springer Publishing Company, Incorporated, 2009.

[21] "Symmetric Key Cipher," [Online]. Available: https://www.sciencedirect.com/topics/computer-science/symmetric-key-cipher. [Accessed 5 12 2020].

[22] P. v. O. a. S. V. A. Menezes, "Key Establishment Protocols," in *The Handbook of Applied Cryptography,* CRC Press, 1996.

[23] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation,* vol. 48, no. 177, pp. 203-209, 1987.

[24] V. S. Miller, "Use of Elliptic Curves in Cryptography," *Springer Heidelberg: Springer,* pp. 417-426, 1986.

[25] J. M.Turner, "National Institute of Standard and Technology.The Keyed-Hash Message Authentication Code(HMAC)," 2008.

[26] R. M. H.Krawczyk, "Hmac: Keyed-hashing for message authentication," Network Working Group,RFC2104,Category: Informational, 1997.

[27] "Internet Security Threat Report," Symantec, 2019.

[28] I. V. J. Delvaux, "Side channel modeling attacks on 65nm arbiter PUFs exploiting CMOS device noise," *HOST,* 2013.

[29] S. Devadas, Interviewee, *Physical unclonable functions and secure processors.* [Interview]. 2009.

[30] K. L. a. W. Daasch., "IC identification circuit using device mismatch," *International Solid State Circuits Conference,* 2000.

[31] R. Pappu, *Physical One-Way Functions*, Phd thesis, MIT, 2001.

[32] B. L. P. Gassend, *Physical random functions,* MSc thesis, MIT, 2003.

[33] U. H. D. Rührmair, "Pufs at a glance," in *Proceedings of the conference on Design, Automation & Test in Europe*, 2014.

[34] V. v. d. L. a. P. Tuyls, "Anti-counterfeiting with hardware intrinsic security," in *Design, Automation Test in Europe Conference Exhibition*, 2013.

[35] V. v. d. L. a. A. Schaller, *Physically unclonable functions found in standard components of commercial devices,* http://www.intrinsic-id.com/ wp-content/uploads/2014/09/Unclonable-functions.pdf, 2013.

[36] R. Maes, Physically Unclonable Functions - Constructions, Properties and Applications, Springer, 2013.

[37] Y. Gao, H. Ma, S. Al-Sarawi, D. Abbott and D. Ranasinghe, "PUF-FSM: A Controlled Strong PUF," in *IEEETrans. Comput.-Aided Des. Integr. Circuits Syst*, 2017.

[38] G. E. S. a. S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *44th ACM/IEEE Design Automation Conference*, 2007.

[39] G. E. S. a. S. Devadas, " Physical unclonable functions for device authentication and secret key generation.," in *In Proceedings of the 44th annual Design Automation Conference,ACM*, 2007.

[40] J. S. F. S. U. Ruhrmair, *On the Foundations of Physical Unclonable Funtions,* Cryptology ePrint Archive, 2009.

[41] S. D. F. K. U. Ruhrmair, "Security based on Physical Unclonability and Disorder," in *In M. Tehranipoor and C. Wang (Editors): Introduction to Hardware Security and Trust*, Springer, 2011.

[42] S. Yi, Z. Hao, Z. Qin and Q. Li, "Fog computing: Platform and applications," in *Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Washington, DC, USA, 2015.

[43] R. Mahmud, R. Kotagiri and R. Buyya, " Fog computing: A taxonomy, survey and future directions. In Internet of Everything," *Springer,* pp. 103-130, 2018.

[44] Z. Maamar, T. Baker, M. Sellami, M. Asim, E. Ugljanin and N. Faci, "Cloud vs edge: Who serves the Internet-of-Things better?," *Internet Technol. Lett,* vol. 1, no. e66, 2018.

[45] R. Naha, S. Garg, D. Georgakopoulos, P. Jayaraman, L. Gao, Y. Xiang and R. Ranjan, " Fog Computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access,* vol. 6, p. 47980–48009, 2018.

[46] J. Ni, K. Zhang, X. Lin and X. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Commun. Surv. Tutor,* vol. 20, p. 601–628, 2017.

[47] A. Alrawais, A. Alhothaily, C. Hu and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," *IEEE Internet Comput.,* vol. 21, pp. 34-42, 2017.

[48] M. Alshahrani, *Secure and Lightweight Authentication Schemes for Internet of Things (IoT),* PhD Dissertation.

[49] R. Sen and S. Borle, "Estimating the contextual risk of data breach: an empirical approach," *Journal of Management InformationSystems,* vol. 32, no. 2, pp. 314-341, 2015.

[50] N. B. M.S. Obaidat, Security of e-Systems and Computer Networks, Cambridge University Press, 2007.

[51] I. Y.Nakkabi, "Improving mouse dynamics biometric performance using variance reduction via extractors with separate features," *IEEETransactionsonSystems, Man, and Cybernetics: Systems,* vol. 40, no. 6, pp. 1345-1353, 2010.

[52] S. P. R. T. M. D. G. a. S. D. M. S. Obaidat, "Biometric security and Internet of Things (IoT)," in *in BiometricBased Physical and Cybersecurity Systems,* Springer, 2019, p. 477–509.

[53] E. Alsolami, *An examination of keystroke dynamic for continuous authentication,* PhD thesis, Queensland University of Technology, 2012.

[54] F. R. Y. C. L. a. H. T. J. Liu, "Optimal combined intrusion detection and biometric based continuous authentication in high security mobile ad hoc networks," *IEEE Transactions* , vol. 8, no. 2, pp. 806-815.

[55] T. S. G. X. Y. K. a. R. R. R. H. C. Yap, "Physical access protection using continuous authentication," in *In technologies for homeland security, IEEE conference*, 2008.

[56] S. Wang, R. Shumba and W. and Kelly, "Security by Design: Defense-in-Depth IoT," *Journal of The Colloquium for Information System Security Education (CISSE)*, vol. 4, no. 2, pp. 1-15, 2017.

[57] M. Aman, K. Chua and B. Sikdar, "Light-Weight Mutual Authentication Protocol for IoT Systems," in *IEEE Global Communications Conference*, 2017.

[58] S. a. K. T. Shin, "A lightweight Three-Factor Authentication and Key Agreement Scheme in Wireless Sensor Networks for Smart Homes," *Sensors*, 2019.

[59] U. Chatterjee, R. Chakraborty and Mukhopadhyay, "A PUF-Based Secure Communication Protocol for IoT.," in *ACM Trans. Embed. Comput. Syst.*, 2017.

[60] A. Braeken, "PUF Based Authentication Protocol for IoT," *Symmetry* , vol. 10, 2018.

[61] P. a. S. B. .. Gope, "Lightweight and Privacy-Preserving Two-Factor Authentication Scheme for IoT Device," *IEEE Internet of Things Journal*, 2018.

[62] M. Aman, K. Chua and B. Sikdar, "Mutual Authentication in IoT Systems Using Physical Unclonable Functions," *IEEE Internet Things Journal*, vol. 4, p. 1327–1340, 2017.

[63] A. Mostafa, S. Lee and Y. and Peker, "Physical Unclonable Function and Hashing Are All You Need to Mutually Authenticate IoT Devices.," *Sensors*, vol. 20, no. 16, 2020.

[64] D. a. A.Yao, "On the security of public keyprotocols," *IEEETransactionson InformationTheory*, vol. 29, no. 2, p. 198–208, 1983.

[65] H. L. V. a. A. R. Cavalli, "Security attacks and solutions in vehicular ad hoc networks: a survey," *International journal on AdHoc networking systems (IJANS)*, vol. 4, no. 2, pp. 1-20, 2014.

[66] M. V. P. a. J. Anuradha, "Network security and types of attacks in network," *Procedia Computer Science*, vol. 48, no. 503 – 506, 2015.

[67] R. D. S. a. Y. P. S. Shunmuganathan, "Secure and efficient smart-card-based remote user authentication scheme for multiserver environment," *Canadian Journal of Electrical and Computer Engineering*, vol. 38, p. 20–30, 2015.

[68] D. B. a. D. Boneh, "Remote timing attacks are practical," *Computer Networks*, vol. 48, no. 5, p. 701 – 716, 2005.

[69] M. Burrows, M. Abadi and R. Needham, "A Logic of Authentication," in *ACM Trans. Comput*, 1990.

[70] "AVISPA (Automated Validation of Internet Security Protocols and Applications)," [Online]. Available: http: //www.avispa-project.org/.

[71] L. Viganò, "Automated security protocol analysis with the avispa tool," *Electronic Notes in Theoretical Computer Science,* vol. 155, p. 61–86, 2006.

[72] Y. S. S. Y. a. F. S. H. Ren, "Secure smart home: A voiceprint and internet based authentication system for remote accessing," in *2016 11th International Conference on Computer Science & Education*, 2016.

[73] D. L. J. K. a. I. a. J. s. S. Jang, "An Efficient Device Authentication Protocol Without Certification Authority for Internet of Things," *Wireless Personal Communication,* vol. 9, no. 4, 2016.

[74] J. Wallrabenstein, "Practical and Secure IoT Device Authentication Using Physical Unclonable Functions," in *IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, Vienna, Austria, 2016.

[75] J. Han and J. A. Kim, "Lightweight authentication mechanism between IoT," in *n Proceedings of the 2017 International Conference on (ICTC)*, Jeju, Korea, 2017.

[76] M. Mughal, X. Luo, Z. Mahmood and A. Ullah, "Physical Unclonable Function Based Authentication Scheme for Smart Devices in Internet of Things.," in *Proceedings of the IEEE International Conference on (SmartIoT)*, Xi'an, China, 2018.

[77] X. S. J. W. X. L. a. T. H. G. Zhao, "A novel mutual authentication scheme for Internet of Things," in *Proceedings of 2011 International Conference on Modelling, Identification and Control*, 2011.

[78] F. a. N.C.Vun., "Securing iot for smart home system," in *International SymposiumonConsumerElectronics(ISCE)*, 2015.

[79] N. T. a. I. J. Z. Alizai, "Improved IoT Device Authentication Scheme Using Device Capability and Digital Signatures," in *International Conference on Applied and Engineering Mathematics (ICAEM)*, Taxila, Pakistan, 2018.

[80] F. K. S. a. N. C. Vun, "Securing IotYfor smart home system," in *IEEE International Symposium on. IEEE Consumer Electronics (ISCE)*, 2015.

[81] G. P. a. B. K. a. M.-s. Jun, A Design of Secure Authentication Method Using Zero Knowledge Proof in Smart-Home Environment, 2016.

[82] M. V.Shivraj, "One time password authentication scheme based on elliptic curves for internet of things(IoT)," in *5 2015 th National Symposium on Information Technology: Towards New Smart World (NSITNSW),*.

[83] T. N. G. A.-M. R. E. N. S. V. J. I. H. T. Sanaz Rahimi Moosavi, "SEA: A Secure and Efficient Authentication and Authorization Architecture for IoT-Based Healthcare Using Smart Gateways," *Procedia Computer Science*, vol. 52, pp. 452-459, 2015.

[84] A. G. J. I. M. Y. a. M. S. P. Kumar, "Lightweight and Secure Session-Key Establishment Scheme in Smart Home Environments," *IEEE Sensors Journal*, vol. 16, no. 1, pp. 254-264, 2016.

[85] g. P.Wilson, *Inter-device authentication protocol for the internet of things, Master's thesis,The University of Victoria, Department of Electrical and Computer Engineerin*, 2017.

[86] J. Z. P. D. a. T. G. C. Huth, "Securing systems on the Internet of Things via physical properties of devices and communications," in *Annual IEEE Systems Conference (SysCon) Proceedings*, Vancouver, BC, Canada, , 2015.

[87] F. K. M. A. a. M. U. M. Jan, "A payload-based mutual authentication scheme for Internet of Things," *Future Gen. Comput. Syst*, vol. 92, p. 1028–1039, 2019.

[88] M. Turuani, "The CL-Atse Protocol Analyser," in *17th International Conference on Term Rewriting and Applications*, Seattle, WA, 2006.

[89] "Invasive Attacks," [Online]. Available: https://www.sec.ei.tum.de/en/research/invasive-attacks/. [Accessed 30 July 2020].

[90] W. B. a. D. E. H. X. Xu, "Using statistical models to improve the reliability of delay-based PUFs," in *in Proc. Symp. VLSI. IEEE*, 2016.

[91] A. B. P. K. A. G. a. M. Y. P. Porambage, "Proxy-based end-to-end key establishment protocol for the Internet of Things," in *IEEE International Conference on Communication Workshop (ICCW),*, 2015.

[92] M. M. a. Z. L. S. Chen, "An authentication scheme with identitybased cryptography for M2M security in cyber-physical systems," *Secur. Commun. Netw*, vol. 9, no. 10, p. 1146–1157, 2016.

[93] W. Tai, Y. Chang and W. Li, "An IoT notion–based authentication and key agreement scheme ensuring user anonymity for heterogeneous ad hoc wireless sensor networks," *Journal of Information Security and Applications*, p. 133–141, 2017.

[94] M. M. a. H. L. J. Cao, "Gbaam: group-based access authentication for mtc in lte networks," *Security and Communication Networks*, vol. 8, no. 17, p. 3282–329, 2015.

[95] E. Lara, L. Aguilar, M. A. s. b. o. Sanchez and J. A. García, "Lightweight Authentication Protocol for M2M Communications of Resource-Constrained Devices in Industrial Internet of Things," *Sensors*, vol. 20, no. 2, 2020.

[96] B. a. S. Zeadally, "Intelligent Device-to-Device Communication in the Internet of Things," *IEEE Systems Journal*, vol. 10, no. 3, pp. 1172-1182, 2016.

[97] A. Esfahani et al., " "A Lightweight Authentication Mechanism for M2M Communications in Industrial IoT Environment," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 288-296, 2019.

[98] S. Aghili and H. Mala, "Breaking a Lightweight M2M Authentication Protocol for Communications in IIoT Environment," Cryptology ePrint Archive, 2018.

[99] C. S. a. J. B. M. Tauber, "A lightweight authentication mechanism for m2m communications in industrial iot environment," *IEEE Internet of Things Journal*, 2017.

[100] C. Chang and H. Le, "A Provably Secure, Efficient, and Flexible Authentication Scheme for Ad hoc Wireless Sensor Networks," in *IEEE Trans. Wirel. Commun*, 2016.

[101] C. Chang and H. Le, " A Provably Secure, Efficient, and Flexible Authentication Scheme for Ad hoc Wireless Sensor Networks," in *IEEE Trans. Wirel. Commun*, 2016.

[102] K. Kolluru, C. Paniagua, J. van Deventer, J. Eliasson, J. Delsing and R. DeLong, " An AAA solution for securing industrial IoT devices using next generation access control.," in *In Proceedings IEEE Industrial Cyber-Physical Systems (ICPS)*, , 2018.

[103] Y. Zhang, R. Deng, D. Zheng, J. Li, P. Wu and J. Cao, "Efficient and Robust Certificateless Signature for Data Crowdsensing in Cloud-Assisted Industrial IoT," in *IEEE Trans. Ind. Inform*, 2019.

[104] A. E. H. A. S. Amine Erroutbi, "Secure and Lightweight HMAC Mutual Authentication Protocol for Communication between IoT Devices and Fog Nodes," in *IEEE International Smart Cities Conference (ISC2)*, 2019.

[105] S. K. D. Z. A. L. L. KM RENUKA, "Design of a Secure Password-Based Authentication Scheme for M2M Networks in IoT Enabled Cyber-Physical Systems," *IEEE Access,* 2019.

[106] F. C. A. B. M. B. O. C. E. S. D. Merabet, "New efficient M2C and M2M mutual authentication protocols for IoT-based healthcare applications," *Peer-to-Peer Networking and Applications,* vol. 13, no. 11, 2019.

[107] M. M. L. Shuo Chen, "An authentication scheme with identity-based cryptography for M2M security in cyber-physical systems," *Security and Communication Networks,* vol. 9, no. 10, pp. 1146-1157, 2016.

[108] O. R. M. B. M. A. M. S. M. S. a. C. G. 16. M. Dammak, "Token-Based Lightweight Authentication to Secure IoT Networks," in *16th IEEE Annual Consumer Communications & Networking*, 2019.

[109] M. 1. Turkanovi'c, B. Brumen and M. Hölbl, "A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion," *Ad Hoc Network,* p. 96–112, 2014.

[110] R. Amin and G. Biswas, "A secure light weight scheme for user authentication and key agreement in multi-gateway based wireless sensor networks," *Ad Hoc Netw,* p. 58–80, 2016.

[111] M. S. WESAM ALMOBAIDEEN, "LIGHTWEIGHT AUTHENTICATION FOR MOBILE USERS IN THE CONTEXT OF FOG COMPUTING," *International Journal of Advanced Computational Engineering and Networking,* vol. 6, no. 12, 2018.

[112] O. R. M. B. M. A. M. S. M. S. a. C. G. M. Dammak, "Token-Based Lightweight Authentication to Secure IoT Networks," in *6th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019.

[113] F. P. a. R. S. M. Gupta, "Object-tagged RBAC model for the Hadoop ecosystem," *Springer,* p. 63–81, 2017.

[114] M. A. M. G. T. F. P. a. J. C. C. F. M. Awaysheh, "Next-generation big data federation access control: A reference Model," *Future Generation Computer Systems*, vol. 108, pp. 726-741, 2019.

[115] K. Wang, Y. Wang, Y. Sun, S. Guo and J. G. Wu, "Industrial Internet of Things Architecture: An Energy-Efficient Perspective," *IEEE Commun. Mag,* p. 48–54, 2016.

[116] "Transportation cost and benefits analysis II—Vehicle costs, " Dept. Transp., Amer. Automobile Assoc., Heathrow, FL, USA, and Victoria Transp. Policy Inst., Victoria, BC, Canada, Tech. Rep.," 2015.

[117] A. Mai, "The Internet of Cars, Spawning New Business Models. CISCO," Oct 2012. [Online]. Available: http://www.gsma.com/ connectedliving/wp-content/uploads/2012/07/12-10-24-SCV-GSMACisco-Perspective-F.pdf. [Accessed 10 July 2020].

[118] 0. u. Organisation Internationale des Constructeurs d'Automobiles" (OICA). Number of passenger cars and commercial vehicles in use worldwide from 2006 to 2014 in (1, "Statista," 2014. [Online]. Available: http://www.statista. com/statistics/281134/number.

[119] Voelcker, "It's official: we now have one billion vehicles on the planet," 2011. [Online]. Available: http://www.greencarreports.com/news/1065070_itsofficial-we-nowhave-one-billion-vehicles-on-the-planet.

[120] S. Z. a. J. A. G.-I. J. Contreras-Castillo, "Internet of Vehicles: Architecture, Protocols, and Security," *IEEE Internet of Things Journal,* vol. 5, no. 5, pp. 3701-3709, 2018.

[121] S. Z. a. J. C.-C. J. A. Guerrero-Ibanez, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and Internet of Things technologies," *IEEE Wireless Commun,* vol. 22, no. 6, p. 122–128.

[122] I. W. R. S. Database, *World Bank Global Road Safety Facility, 2010, Cisco IBSG,* San Jose,, 2011.

[123] S. S. F. Sakiz, " A survey of attacks and detection mechanisms on intelligent transportation systems," *VANETs and IoV, Ad Hoc Netw,* vol. 61, p. 33–50, 2017.

[124] M. K. e. al., "System on chip and method for cryptography using a physically unclonable function". US Patent 8 750 502 B2, 22 March 2012.

[125] T. Zhang, "Securing Connected Vehicles: Challenges and Opportunities," in *CISCO Syst,* San Jose, CA, 2015.

[126] D. Yadron, "Hackers demonstrate how to take control of cars," in *Proc. Black Hat Security Conf,* Las Vegas, NV, 2015.

[127] J. Hickey, Interviewee, *Vice President, Vínsula.* [Interview]. Oct 2012.

[128] M. A. B. Mokhtar, "Survey on security issues in vehicular ad hoc networks," *Alex. Eng. J.,* vol. 54, no. 4, p. 1115–1126, 2015.

[129] R. A. R. M. K. M.H.M. Zaharuddin, " Technical comparison analysis of encryption algorithm on site-to-site IPSec VPN," in *International Conference on Computer Applications and Industrial Electronics, ICCAIE, IEEE*, 2010.

[130] M. A. B. C. C. Bernardini, " Security and privacy in vehicular communications: challenges and opportunities," *Veh. Commun,* vol. 10, p. 13–28, 2017.

[131] A. R. A. Daeinabi, "Detection of malicious vehicles (DMV) through monitoring in Vehicular Ad-Hoc Networks," *Multimed. Tools Appl,* vol. 66, no. 2, p. 325–338, 2013.

[132] J. B. S. W. J. Guo, " A group signature based secure and privacy preserving vehicular communication framework," in *Mobile Networking for Vehicular Environments, IEEE*, 2007.

[133] I. A. H. H. I.A. Sumra, "Behavior of attacker and some new possible attacks in Vehicular Ad hoc Network (VANET)," in *3rd International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), IEEE*, 2012.

[134] S. M. S.S. Tangade, "A survey on attacks, security and trust management solutions in VANETs," in *Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), IEEE*, 2013.

[135] N. M. Y. Qian, "Design of secure and application-oriented VANETs," in *Vehicular Technology Conference IEEE*, 2008.

[136] e. a. Y. Sun, "Attacks and countermeasures in the Internet of vehicles," *Ann. Telecommun,* vol. 72, no. 5-6, p. 283–295, 2017.

[137] B. D. G. Guette, "On the Sybil attack detection in VANET," in *EEE MoVeNet 2007*, Pisa, 2007.

[138] H. H. J.-l. b. A. M. I.A. Sumra, "Classes of Attacks in VANET," in *Saudi International Electronics, Communications and Photonics Conference (SIECPC), IEEE,* Riyadh, Saudi Arabia, 2011.

[139] e. a. M. Ghosh, "Distributed misbehavior detectio," in *Wireless Communications and Networking Conference, WCNC 2009. IEEE*, 2009.

[140] S. A. a. P. G. T. Bécsi, "Security issues and vulnerabilities in connected car systems," in *International Conference on Models and Technologies for Intelligent Transportation Systems*, 2015.

[141] X. Z. K. J. R. A. S. L. Y. Z. Y. X. ,. Y. Mahdi Dibaei, "Attacks and defences on intelligent connected vehicles: a survey," *Digital Communications and Networks,* vol. 6, no. 4, pp. 399-421, 2020.

[142] J. D. ,. a. R. K. Florian Sommer, "Survey and Classification of Automotive," *Information,* vol. 10, 2019.

[143] V. D. D. D. a. S. K. D. H. Vasudev, "A Lightweight Mutual Authentication Protocol for V2V Communication in Internet of Vehicles," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 6, pp. 6709-6717, 2020.

[144] W. H. H. C. Z. S. P. A. a. A. L. ". ,. v. 4. n. 4. p. ,. D. 2. S. Bao, "A lightweight authentication and privacy-preserving scheme for VANETs using TESLA and bloom filters," *ICT Express,* vol. 4, no. 4, p. 221–227, 2018.

[145] A. P. a. J. D. Tygar, " TESLA Broadcast Authentication. Boston," *Springer,* p. 29–53, 2003.

[146] S. L. M. X. S. D. a. X. W. Y. Zhou, "An efficient V2I authentication scheme for VANETs," *Mobile Inf. Syst,* vol. 2018, 2018.

[147] N. K. A. K. D. a. W. S. A. Dua, "Secure Message Communication Protocol Among Vehicles in Smart City," *IEEE Transactions on Vehicular Technology,* vol. 67, no. 5, pp. 4359-4373, 2018.

[148] D. D. a. A. V. V. H. Vasudev, "Secure message propagation protocols for IoVs communication components," *Comput. Electr. Eng,* vol. 82, pp. 1-15, 2020.

[149] J. L. K. P. A. K. D. A. Y. P. SUNGJIN YU, "IoV-SMAP: Secure and Efficient Message Authentication Protocol for IoV in Smart City Environment," *IEEE Access,* vol. 8, pp. 167875-167886, 2020.

[150] X. Z. N. Z. Y. T. X. M. a. J. M. ". Q. Jiang, "Two-Factor Authentication Protocol Using Physical Unclonable Function for Io," in *IEEE/CIC International Conference on Communications in China (ICCC)*, China, 2019.

[151] ". X. Zhang and X. Chen, "Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network," *IEEE Access,* vol. 7, p. 58241–58254, 2019.

[152] K. Y. L. L. K. Z. a. V. C. M. L. Z. Yang, "Blockchain-based decentralized trust management in vehicular networks," *IEEE Internet Things J,* vol. 6, no. 2, p. 1495–1505, 2019.

[153] H. C. Y. C. P. A. C. P. A. O. a. Z. S.] A. Lei, "Blockchain-based dynamic key management for heterogeneous intelligent transportation systems," *IEEE Internet Things J.*, vol. 4, no. 6, p. 1832–1843, 2017.

[154] U. J. a. B. S. M. N. Aman, "A Privacy-Preserving and Scalable Authentication Protocol for the Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1123-1139, 2021.

[155] N. Z. J. N. J. M. X. M. a. K. -K. R. C. Q. Jiang, "Unified Biometric Privacy Preserving Three-Factor Authentication and Key Agreement for Cloud-Assisted Autonomous Vehicles," *IEEE Transactions on Vehicular Technology,* vol. 69, no. 9, p. 9390, 2020.

[156] X. Z. N. Z. Y. T. X. M. J. M. Qi Jiang, "Three-factor authentication protocol using physical unclonable function for IoV," *Computer Communications,* vol. 173, pp. 45-55, 2021.

[157] S. C. V. C. M. G. Tejasvi Alladi, "A Lightweight Authentication and Attestation Scheme for In-Transit Vehicles in IoV Scenario," in *IEEE Transactions on Vehicular Technology*, 2020.

[158] M. W. e. al, "Design of lightweight authentication and key agreement protocol for vehicular ad hoc networks," *IEEE Access,* vol. 5, p. 14966–14980, 2017.

[159] N. K. A. K. D. a. W. S. A. Dua, "Secure message communication protocol among vehicles in smart city," *IEEE Trans. Veh. Technol,* vol. 67, no. 5, p. 4359–4373, 2018.

[160] C. J. Z. H. Y. R. a. L. H. J. Wang, "Internet of Vehicles: Sensing-aided transportation information collection and diffusion," *IEEE Trans. Veh. Technol,* vol. 67, no. 5, p. 3813–3825, 2018.

[161] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the United States," *Proc. IEEE,* vol. 99, no. 7, p. 1162–1182, 2011.

[162] B. S. K. C. C. a. A. A. M. N. Aman, "Low power data integrity in IoT systems," *IEEE Internet Things J,* vol. 5, no. 4, p. 3102–3113, 2018.

[163] M. K. e. al., "System on chip and method for cryptography using a physically unclonable function". USA Patent 8 750 502 B2, 22 March 2012.

[164] N. C. a. S. F. H. Saevanee, pp. 465-474, 2012.

[165] E. Alsolami, *An examination of keystroke dynamic for continuous authentication,* PhD thesis, Queensland University of Technology, 2012.

[166] F. R. Y. C. L. a. H. T. J. Liu, "Optimal combined intrusion detection and biometric based continuous authentication," *IEEE Transactions,* vol. 8, no. 2, pp. 806-815, 2009.

[167] T. S. G. X. Y. K. a. R. R. R. H. C. Yap, "Physical access protection using continuous authentication," in *In technologies for homeland security, IEEE conference*, 2008.

[168] Samsara, "Everything You Need to Know About Fleet Management," 2021. [Online]. Available: https://www.samsara.com/guides/what-is-fleet-management. [Accessed 5 Jan 2021].

[169] "pHash," [Online]. Available: https://www.phash.org/. [Accessed 5 11 2019].

[170] S. W. X. Z. a. W. W. Z. Tang, "Structural feature-based image hashing and similarity metric for tampering detection," *Fundam Inf,* vol. 106, no. 1, p. 75–91, 2011.

[171] D. O, "On the security of public key protocols," *IEEE Transactionson Information Theory,* vol. 29, no. 2, p. 198–208, 1983.

[172] M.Abomharaetal, "Cyber security and the interne tof things: vulnerabilities,threats, intruders and attacks," *Journal of Cyber Security and Mobility,vol. 4, no. 1,* p. 65–88, 2015.

[173] M.Abomharaetal, "Cyber security and the interne tof things: vulnerabilities,threats, intruders and attacks," *Journal of Cyber Security and Mobility,* vol. 4, no. 1, p. 65–88, 2015.

[174] W. S. a. L. Brown, Computer Security: Principles and Practice, Pearson, 2015.

APPENDIX A

```
role role_N

(N:agent,G:agent,T:text,X:text,Y:text,NIDA:text,NIDR:text,IGIDR:text,IGIDA:text,GX:text,O
TP:text, H:hash_func,SND,RCV:channel
 (dy))
 played_by N

 def=
        local

                State:nat,
                TSoneN:text,TStwoN:text,
                None:hash(text.text),
                TSoneIG:text,TStwoIG:text,
                SIDNIG:text,Pn:text,Pg:text,
                Rone:text,Rtwo:text,
                Cone:text,Ctwo:text,
                Gtwo:hash(message.text.text.text.text),
                Nthree:hash(text.text.text.message.message),
                SN:hash(message.text.text.text),
                SSK:hash(text.text.text.text.text),
                NOTP:hash(text.text.text),
                NnewIDA:hash(text.text.text),
                Gone:hash(text.text.message),
                Na:text,Nb:text,NBB:text,Naa:text,
              Gthree:hash(text.text.message.text),Ntwo:text,
                IG_pub:public_key

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
             =|> State' :=4
                                /\ TSoneN' :=new()
                                /\ Na' := new()
                                /\ None':= H(Y.OTP)
                                /\ Ntwo' := xor(Na',None')
                                /\ SN' :=H(None'.NIDR.TSoneN'.Na')
                                /\ SND(NIDA.TSoneN'.xor(Na',None').H(None'.NIDR.TSoneN'.Na'))
                                /\ secret(NIDR,secNIDR,{N,G})
```

```
                        /\ secret(OTP,secOTP,{N,G})
                        /\ secret(Na,secNa,{N,G})

          2. State=  4 /\
RCV(IGIDA.SIDNIG'.TSoneIG'.xor(Gone',Cone').{Nb'.Gtwo'}_Rone')
          =|> State' :=6

                        /\ TStwoN' :=new()
                        /\ Naa' :=new()
                        /\ Pn' :=new()
                        /\ NnewIDA' := H(NIDA.Rone.Cone)
                        /\ NOTP' := H(OTP.Cone'.TStwoN')
                        /\ Nthree' := H(Naa'.TStwoN'.Nb'.NOTP'.NnewIDA')
                        /\ SND(NIDA.SIDNIG.TStwoN'.{Naa'.exp(GX,Pn').Nthree'}_Rone')
                        /\ secret(Naa,secNaa,{N,G})
                        /\ request(N,G,na,Na)
                        /\ secret(NOTP,secNOTP,{G,N})
                        /\ witness(N,G,nb,Nb')
end role
```

Figure A_1: HLPSL code for role IoT Node played by N

```
role role_G
(G:agent,N:agent,IGIDA:text,NIDA:text,NIDR:text,IGIDR:text,GX:text,OTP:text,CG:text,I
G_pub:public_key,H:hash_func,SND,RCV:channel(dy))

played_by G

def=
        local
                State:nat,
                T:text,X:text,Y:text,
                TSoneN:text,TStwoN:text,
                TSoneIG:text,TStwoIG:text,
                SIDNIG:text,Pn:text,Pg:text,
                Rone:text,Rtwo:text,
                Cone:text,Ctwo:text,
                Gtwo:hash(message.text.text.text.text),
                Nthree:hash(text.text.text.message.message),
                SNG:hash(message.text.text.text),
                Ssk:hash(text.text.text.text.text),
                NOTP:hash(text.text.text),
                NnewIDA:hash(text.text.text),
                Gone:hash(text.text.message),
                Na:text,Nb:text,NAA:text,NB:text,Naa:text,CX:text,
                Gthree:hash(text.text.message.text),
                TG:hash(text.text.text),
```

```
        RG:text,Ntwo:text,None:hash(text.text)
    Init
        State:= 1
    transition

        1. State=1
            /\RCV(NIDA.TSoneN'.xor(Na',None').SNG')
        =|> State' :=3

            /\ Nb':= new()
            /\ TSoneIG' :=new()
          /\ SIDNIG' := new()
            /\ RG' := new()
            /\ Rone' :=new()
            /\ TG' := H(RG'.NIDR.IGIDR)
            /\ Gone' := H(OTP.Na'.TG')
            /\ Cone' :=new()
            /\ CX' := xor(Gone',Cone')
            /\ Gtwo' :=H(Gone'.Cone'.TSoneIG'.Nb'.IGIDR)
            /\ SND(IGIDA.SIDNIG'.TSoneIG'.xor(Gone',Cone').
                    {Nb'.H(Gone'.Cone'.TSoneIG'.Nb'.IGIDR)}_Rone')
            /\ secret(IGIDR,secIGIDR,{G,N})
            /\ secret(OTP,secOTP,{G,N})
            /\ secret(Nb,secNb,{G,N})
            /\ secret(Rone,secRone,{G,N})
            /\ secret(Cone,secCone,{G,N})
            /\ secret(RG,secRG,{G,N})
            /\ witness(G,N,na,Na')


        2. State=3 /\
RCV(NIDA.SIDNIG.TStwoN'.{Naa'.exp(GX,Pn').Nthree'}_Rone)
        =|> State' :=5

            /\request(G,N,nb,Nb)
end role
```

Figure A_2: HLPSL code for role Intelligent Gateway played by IG

```
role session(N:agent,G:agent,T:text,X:text,Y:text,NIDA:text,IGIDA:text,NIDR:text,
IGIDR:text,GX:text,OTP:text,CG:text,IG_pub:public_key,
H:hash_func)
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition

        role_G(G,N,IGIDA,NIDA,NIDR,IGIDR,GX,OTP,CG,IG_pub,H,SND2,RCV2)

        role_N(N,G,T,X,Y,NIDA,NIDR,IGIDR,IGIDA,GX,OTP,H,SND1,RCV1)/\
end role
```

Figure A_3: HLPSL code for role session

```
role environment()
def=
        const
n:agent,g:agent,t:text,x:text,y:text,nida:text,igida:text,nidr:text,igidr:text,gx:text,otp:tex
t,cg:text,ig_pub:public_key,h:hash_func,secNIDR, secOTP, secNa, secIGIDR, secNb,
secRone, secCone,secNaa,secNOTP,secRG,na,nb:protocol_id
intruder_knowledge = {n,g,gx,h,ig_pub}
        composition
                session(n,g,t,x,y,nida,igida,nidr,igidr,gx,otp,cg,ig_pub,h)
end role
```

Figure A_4: HLPSL code for role environment

```
Goal
        secrecy_of secNIDR
        secrecy_of secOTP
        secrecy_of secNa
        secrecy_of secNb
        secrecy_of secIGIDR
        secrecy_of secRone
        secrecy_of secCone
        secrecy_of secNaa
        secrecy_of secNOTP
        secrecy_of secRG
        authentication_on na
        authentication_on nb
end goal
```

FigureA_5: HLPSL code for goal

APPENDIX B

```
role
role_C(C:agent,P:agent,PIDA:text,CIDA:text,PIDR:text,CIDR:text,OTP:text,MSK:tex
t,CPC:text,PAU:text,H:hash_func,SND,RCV:channel(dy))
played_by C
def=
        local

State:nat,
                TSoneC:text,TStwoC:text,
                TSoneP:text,TStwoP:text,
                RVCone:text,RVCtwo:text,RVPone:text,
                YxP:hash(text.text.text),
                Xone:text,
                Xtwo: hash(text.text.text.message),
                Xthree:text,
                Xfour: hash(text.text.text.message),
                Yonex:text,
                Ytwox: hash(text.text.text.message),
                Ythree:text,
                Yfour:text,
                YPx:hash(text.text.text.text.message.message),
                TKone:text,TKtwo:text,
                OTPnew:hash(text.text),
                CIDAnew:hash(text.text),
                Xc:hash(text.text),
                Yx:hash(text.text),
                Xcc:hash(text.text),CAU:text

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
            =|> State' :=4 /\ TSoneC' :=new()
                                /\ RVCone' := new()
                                /\ Xc' := H(OTP.PAU)
                                /\ Xone' := xor(RVCone',H(OTP.PAU))
                                /\ Xtwo' := H(CIDR.TSoneC'.RVCone'.Xc')

                                 /\SND(CIDA.TSoneC'.xor(RVCone',H(OTP.PAU)).
                                        H(CIDR.TSoneC'.RVCone'.H (OTP.PAU)))
                                /\secret(CIDR,secCIDR,{C,P})
                                /\secret(OTP,secOTP,{C,P})
                                /\secret(RVCone,secRVCone,{C,P})
```

```
            2. State=   4 /\
RCV(PIDA.TSoneP'.xor(H(OTP.CAU'),RVPone').Ytwox')=|>
                State' :=6

                        /\ TStwoC' :=new()
                        /\ RVCtwo' :=new()
                        /\ Xcc' := H(RVPone'.OTP)
                        /\ Xthree' := xor(RVCtwo',H(RVPone'.OTP))
                        /\ Xfour' := H(CIDR.TStwoC'.RVCtwo'.Xcc')
                        /\ SND(CIDA.TStwoC'.xor(RVCtwo',H(RVPone'.OTP)).
                                H(CIDR.TStwoC'.RVCtwo'.H(RVPone'.OTP)))
                        /\ request(C,P,rvcone,RVCone)
                        /\witness(C,P,rvpone,RVPone')
                        /\ secret(RVCtwo,secRVCTwo,{C,P})
    1.  State=  6 /\
          RCV(PIDA.TStwoP'.xor(TKone',OTP).xor(TKtwo',TKone').YPx')=|>
              State' :=8
end role
```

Figure B_1: HLPSL code for role Child Gateway Node played by C

```
role role_P
(P:agent,C:agent,PIDA:text,CIDA:text,PIDR:text,CIDR:text,OTP:text,MSK:text,CPP:t
ext,CAU:text,H:hash_func,SND,RCV:channel(dy))
played_by P
def=
        local
                State:nat,
                TSoneC:text,TStwoC:text,
                TSoneP:text,TStwoP:text,
                RVCone:text,RVCtwo:text,RVPone:text,
                YxP:hash(text.text.text),
                Xone:text,
                XtwoP: hash(text.text.text.message),
                XthreeP:text,
                XfourP:hash(text.text.text.message),
                Yone:text,
                Ytwo: hash(text.text.text.message),
                Ythree:text,
                Yfour:text,
                YP:hash(text.text.text.text.message.message),
                TKone:text,TKtwo:text,
                OTPnew:hash(text.text),
                CIDAnew:hash(text.text),
                Xc:hash(text.text),
                Yx:hash(text.text),PAU:text,
```

```
                Sk:hash(text.text.text.message)


        init
                State:= 1
        transition

                1. State=1
                        /\RCV(CIDA.TSoneC'.xor(RVCone',H(OTP.PAU')).XtwoP')
           =|> State' :=3

                                /\ TSoneP' :=new()
                                /\ RVPone':= new()
                                /\ Yx':= H(OTP.CAU)
                                /\ Yone' := xor(H(OTP.CAU),RVPone')
                                  /\ Ytwo' := H(PIDR.TSoneP'.RVPone'.Yx')
                                /\
SND(PIDA.TSoneP'.xor(H(OTP.CAU),RVPone').H(PIDR.TSoneP'.RVPone'.
                                        H(OTP.CAU)))
                                /\ secret(PIDR,secPIDR,{P,C})
                                /\ secret(OTP,secOTP,{P,C})
                                /\ secret(RVPone,secRVPone,{P,C})
                                /\ witness(P,C,rvcone,RVCone')

                2. State= 3 /\
RCV(CIDA.TStwoC'.xor(RVCtwo',H(RVPone.OTP)).XfourP')
           =|> State' :=5
                                /\ TStwoP' :=new()
                                /\ TKone' := new()
                                /\ TKtwo' := new()
                                /\ Ythree' := xor(TKone',OTP)
                                /\ Yfour' := xor(TKtwo',TKone')
                                /\ OTPnew' := H(OTP.RVCtwo')
                                /\ CIDAnew' := H(CIDA.OTP)
                                /\ YP' :=
H(PIDR.TStwoP'.TKone'.TKtwo'.OTPnew'.CIDAnew')
                                    /\
SND(PIDA.TStwoP'.xor(TKone',OTP).xor(TKtwo',TKone').

H(PIDR.TStwoP'.TKone'.TKtwo'.H(OTP.RVCtwo').H (CIDA.OTP)))
                                 /\request(P,C,rvpone,RVPone)
                                /\secret(TKone,secTKone,{P,C})
                                /\secret(TKtwo,secTKtwo,{P,C})
                                /\secret(OTPnew,secOTPnew,{P,C})
                                /\secret(CIDAnew,secCIDAnew,{P,C})
                                /\secret(YxP,secYxP,{P,C})
```

```
end role
```

Figure B_2: HLPSL code for role Parent Gateway played by P

```
role
session(C:agent,P:agent,CIDA:text,PIDA:text,CIDR:text,PIDR:text,OTP:text,MSK:tex
t,CPC:text,PAU:text,CPP:text,CAU:text,H:hash_func)
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition

        role_C(C,P,PIDA,CIDA,PIDR,CIDR,OTP,MSK,CPC,PAU,H,SND1,RCV1)/\

        role_P(P,C,PIDA,CIDA,PIDR,CIDR,OTP,MSK,CPP,CAU,H,SND2,RCV2)

end role
```

Figure B_3: HLPSL code for role session

```
role environment()

def=

        const
c:agent,p:agent,cida:text,pida:text,cidr:text,pidr:text,otp:text,msk:text,cpp:text,cau:text,
cpc:text,pau:text,h:hash_func, secCIDR,
secOTP,secRVCone,secRVCTwo,secPIDR,secRVPone,secTKone,secTKtwo,secOTPn
ew,secCIDAnew,secYxP,rvcone,rvpone:protocol_id

intruder_knowledge = {c,p,h}

        composition

                session(c,p,cida,pida,cidr,pidr,otp,msk,cpp,cau,cpc,pau,h)
end role
```

Figure B_4 HLPSL code for role environment

```
Goal

        secrecy_of secCIDR
        secrecy_of secOTP
        secrecy_of secRVCone
        secrecy_of secRVPone
        secrecy_of secPIDR
        secrecy_of secRVCTwo
        secrecy_of secTKone
        secrecy_of secTKtwo
        secrecy_of secOTPnew
        secrecy_of secCIDAnew
        authentication_on rvcone
        authentication_on rvpone

end goal
```

Figure B_5: HLPSL code for goal

```
role
role_Cone(Cone:agent,Ctwo:agent,ConeIDA:text,CtwoIDA:text,ConeIDR:text,CtwoI
DR:text,TK:text,MSK:text,H:hash_func,SND,RCV:channel(dy))
played_by Cone

def=
        local

          State:nat,
                TSoCone:text,TStCone:text,
                TSoCtwo:text,TStCtwo:text,
                RCone:text,RCtwo:text,RCCone:text,RCCTwo:text,
                Tone:text,Tfour:text,
                Ttwo: hash(text.text.text.message),
                    Tthree: hash(text.text.text),
                Tfive: hash(text.text.text.message),
                Dtwo:text,
                Dthreex: hash(text.text.text.message),
                Done: hash(text.text.text),
                Rj:hash(text.text),
                Sk:hash(text.text.text.text)

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
         =|> State' :=4 /\ TSoCone' :=new()
```

```
                        /\ Rj' := H(TK.CtwoIDR)
                        /\ RCone' := new()
                        /\ Tone' := xor(RCone',Rj')
                          /\ Ttwo' := H(ConeIDR.TSoCone'.RCone'.Rj')


/\SND(CtwoIDA.xor(RCone',Rj').H(ConeIDR.TSoCone'.RCone'.

H(ConeIDR.TSoCone'.RCone'.H(TK.CtwoIDR))))
                        /\secret(ConeIDR,secConeIDR,{Cone,Ctwo})
                        /\secret(MSK,secMSK,{Cone,Ctwo})
                        /\secret(TK,secTK,{Cone,Ctwo})
                        /\secret(RCone,secRCone,{Cone,Ctwo})

          2. State=   4 /\
RCV(CtwoIDA.xor(H(RCone'.MSK.ConeIDR),RCCone').Dthreex')=|>
          State' :=6

                        /\ TStCone' :=new()
                             /\ RCtwo' :=new()
                        /\ Tthree' := H(TK.MSK.RCCone')
                        /\ Tfour' := xor(RCtwo',H(TK.MSK.RCCone'))
                        /\ Tfive' :=
H(ConeIDR.TStCone'.RCtwo'.H(TK.MSK.RCCone'))
                        /\ request(Cone,Ctwo,rcone,RCone)
                        /\ SND(ConeIDA.xor(RCtwo',H(TK.MSK.RCCone')).

H(ConeIDR.TStCone'.RCtwo'.H(TK.MSK.RCCone')))
                        /\ secret(RCtwo,secRCtwo,{Ctwo,Cone})
                        /\witness(Cone,Ctwo,rccone,RCCone')

end role
```

Figure B_6: HLPSL code for role first child gateway played by C_{one}

```
role role_Ctwo
(Cone:agent,Ctwo:agent,ConeIDA:text,CtwoIDA:text,ConeIDR:text,CtwoIDR:text,TK
:text,MSK:text,H:hash_func,SND,RCV:channel(dy))

played_by Ctwo

def=
        local
                State:nat,
                TSoCone:text,TStCone:text,
                TSoCtwo:text,TStCtwo:text,
                RCone:text,RCtwo:text,RCCone:text,RCCTwo:text,
```

```
                Tone:text,Tfour:text,
                Tthree: hash(text.text.text),
                Ttwox: hash(text.text.text.message),
                Tfive: hash(text.text.text.message),
                Dtwo:text,
                Dthree: hash(text.text.text.message),
                Done:hash(text.text.text),
                Rj:hash(text.text),
                Sk:hash(text.text.text.text)


        init
                State:= 1
        transition

                1. State=1
                        /\RCV(CtwoIDA.xor(RCone',Rj').Ttwox')
          =|> State' :=3

                        /\ TSoCtwo' :=new()
                        /\ Done' := H(RCone'.MSK.ConeIDR)
                        /\ RCCone':= new()
                        /\ Dtwo' := xor(Done',RCCone')
                        /\ Dthree' := H(CtwoIDR.TSoCtwo'.RCCone'.
                                        H(RCone'.MSK.ConeIDR))
                        /\ SND(CtwoIDA.xor(H(RCone'.MSK.ConeIDR),RCCone')

.H(CtwoIDR.TSoCtwo'.RCCone'.H(RCone'.MSK.ConeIDR)))
                        /\secret(CtwoIDR,secCtwoIDR,{Ctwo,Cone})
                        /\secret(MSK,secMSK,{Ctwo,Cone})
                        /\secret(RCCone,secRCCone,{Ctwo,Cone})
                        /\witness(Ctwo,Cone,rcone,RCone')
                2. State=3 /\ RCV(ConeIDA.xor(RCtwo',H(TK.MSK.RCCone')).Tfive')
           =|> State' :=5
                        /\ request(Ctwo,Cone,rccone,RCCone)
end role
```

Figure B_7: HLPSL code for role second child gateway played by C_{two}

```
role
session(Cone:agent,Ctwo:agent,ConeIDA:text,CtwoIDA:text,ConeIDR:text,CtwoIDR:
text,TK:text,MSK:text,H:hash_func)

def=
        local
                SND2, RCV2, SND1, RCV1: channel(dy)
        composition
```

```
        role_Cone(Cone,Ctwo,ConeIDA,CtwoIDA,ConeIDR,CtwoIDR,TK,MSK,H,S
ND1,RCV1)/\

        role_Ctwo(Cone,Ctwo,ConeIDA,CtwoIDA,ConeIDR,CtwoIDR,TK,MSK,H,S
ND2,RCV2)

end role
```

Figure B_8: HLPSL code for role session

```
role environment()

def=

        const
cone:agent,ctwo:agent,coneida:text,ctwoida:text,coneidr:text,ctwoidr:text,tk:text,msk:t
ext,h:hash_func,
        secConeIDR,secMSK,secTK,secRCone,secRCCone,secCtwoIDR,secRCtwo,rc
cone,rcone:protocol_id

intruder_knowledge = {cone,ctwo,h}

        composition

                session(cone,ctwo,coneida,ctwoida,coneidr,ctwoidr,tk,msk,h)

end role
```

Figure B_9: HLPSL code for role environment

```
Goal

        secrecy_of secConeIDR
        secrecy_of secMSK
        secrecy_of secRCone
        secrecy_of secRCtwo
        secrecy_of secTK
        secrecy_of secRCCone
        secrecy_of secCtwoIDR
        authentication_on rcone
        authentication_on rccone

end goal
```

Figure B_10: HLPSL code for security goals

```
role role_N
(N:agent,MG:agent,T:text,X:text,Y:text,NIDA:text,NIDR:text,MGIDR:text,MGIDA:te
xt,CHX:text,OTT:text,H:hash_func,SND,RCV:channel
 (dy))

 played_by N

 def=
        local

        State:nat,
                TSoneN:text,TStwoN:text,
                Xone:hash(text.text),
                TSoneMG:text,TStwoMG:text,RX:text,
                SIDNIG:text,
                Rone:text,Rtwo:text,
                Cone:text,Ctwo:text,CMG:text,
                CHXnew:hash(text.text),
                SN:hash(message.text.text.text),
                Yone:hash(text.text),
                Ythreex:hash(message.text.text.text.text),
                NOTT:hash(text.text.text),
                NnewIDA:hash(text.text.text),
                Xfour:hash(text.text.text.message.message.message.message.text.text),
                Na:text,Nb:text,Xtwo:text,Naa:text,Xthree:text,
                Ytwo:text

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
         =|> State' :=4 /\ TSoneN' :=new()
                                /\ Na' := new()
                                  /\ Xone':= H(X.Y)
                                /\ Xtwo' := xor(Na',OTT)
                                /\ SN' :=H(H(X.Y).NIDR.TSoneN'.Na')
                                /\
SND(NIDA.TSoneN'.xor(Na',H(X.Y)).H(H(X.Y).NIDR.TSoneN'.Na'))
                                /\ secret(NIDR,secNIDR,{N,MG})
                                /\ secret(OTT,secOTT,{N,MG})
                                /\ secret(Na,secNa,{N,MG})
                                /\ secret(X,secX,{N,MG})
                                /\ secret(Y,secY,{N,MG})

             2. State=   4 /\ RCV(MGIDA.SIDNIG'.TSoneMG'.xor(Nb',OTT)
```

```
                                .xor(Cone',Nb').TSoneMG'.Ythreex')
                            =|>  State' :=6

                             /\ TStwoN' :=new()
                               /\ Rone' := new()
                             /\ CHXnew' := H(CHX.Rone')
                             /\ Ctwo' := H(Na.Nb')
                             /\ Rtwo':=new()
                                  /\ Naa':=new()
                             /\ Xthree':=xor(Naa',CHXnew')
                             /\ RX' :=xor(Rtwo',H(Nb'.CHXnew'))
                             /\ NOTT' := H(OTT.Cone'.Rone')
                             /\ NnewIDA' := H(NIDA.Nb')
                             /\Xfour' := H(NIDR.TStwoN'.Rone'.H(CHX.Rone')

.H(OTT.Cone'.Rone').H(NIDA.Nb').H(Na.Nb').Rtwo'.Naa')
                                    /\SND(NIDA.SIDNIG'.TStwoN'.xor(Naa',H(CHX.Rone'))
                                          .xor(Rtwo',H(Nb'.H(CHX.Rone'))).Xfour')
                            /\secret(Rone,secRone,{N,MG})
                            /\secret(CHXnew,secCHXnew,{N,MG})
                            /\secret(Naa,secNaa,{N,MG})
                            /\secret(Rtwo,secRtwo,{N,MG})
                            /\secret(NOTT,secNOTT,{N,MG})
                            /\secret(NnewIDA,secNnewIDA,{N,MG})
                            /\request(N,MG,na,Na)
                            /\witness(N,MG,nb,Nb')
end role
```

Figure B_11: HLPSL code for role IoT Node played by N

```
role role_MG
MG:agent,N:agent,MGIDA:text,NIDA:text,NIDR:text,MGIDR:text,CHX:text,OTT:te
xt,H:hash_func,SND,RCV:channel(dy))
 played_by MG
 def=
        local
                State:nat,
                T:text,X:text,Y:text,RX:text,
                TSoneN:text,TStwoN:text,
                TSoneMG:text,TStwoMG:text,
                SIDNIG:text,
                Rone:text,Rtwo:text,
                Cone:text,Ctwo:text,CMG:text,
                CHXnew:hash(text.text),
                Ythree:hash(message.text.text.text.text),
                SNG: hash(message.text.text.text),
                Yone:hash(text.text),
```

```
                NOTT:hash(text.text.text),
                NnewIDA:hash(text.text.text),
                Xfour:hash(text.text.text.message.message.message.message.text.text),
                Na:text,Nb:text,Naa:text,Xthree:text,
                Ytwo:text
        init
                State:= 1
        transition
                1. State=1
                        /\RCV(NIDA.TSoneN'.xor(Na',OTT).SNG')
          =|> State' :=3
                                /\ Yone' :=H(T.X)
                                /\ Nb':= new()
                                  /\ Ytwo' :=xor(Nb',OTT)
                                /\ TSoneMG' :=new()
                                /\ Cone' := new()
                                /\ CMG' :=xor(Cone',Nb')
                                     /\ Ythree' := H(H(T.X).TSoneMG'.Nb'.Na'.Cone')
                                     /\ SIDNIG' := new()
                                /\
SND(MGIDA.SIDNIG'.TSoneMG'.xor(Nb',OTT).xor(Cone',Nb').TSoneMG'.H(H
                                          (T.X).TSoneMG'.Nb'.Na'.Cone'))
                                /\secret(MGIDR,secMGIDR,{MG,N})
                                /\secret(OTT,secOTT,{MG,N})
                                /\secret(Nb,secNb,{MG,N})
                                /\secret(Cone,secCone,{MG,N})
                                /\witness(MG,N,na,Na')
    1.  State=3
                          /\ RCV(NIDA.SIDNIG.TStwoN'.xor(Naa',H(CHX.Rone'))
                        .xor(Rtwo',H(Nb.H(CHX.Rone'))).H
(NIDR.TStwoN'.Rone'.H(CHX.Rone')
                                  .H(OTT.Cone.Rone').H(NIDA.Nb).H(Na.Nb).Rtwo'.Naa'))
            =|> State' :=5
                                /\request(MG,N,nb,Nb)

end role
```

Figure B_12: HLPSL code for mini-gateway node played by MG

```
role
session(N:agent,MG:agent,T:text,X:text,Y:text,NIDA:text,MGIDA:text,NIDR:text,M
GIDR:text,CHX:text,OTT:text,H:hash_func)

 def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition
```

```
        role_N(N,MG,T,X,Y,NIDA,NIDR,MGIDR,MGIDA,CHX,OTT,H,SND1,RCV
1)/\

        role_MG(MG,N,MGIDA,NIDA,NIDR,MGIDR,CHX,OTT,H,SND2,RCV2)

 end role
```

Figure B_13. HLPSL code for role session

```
role environment()

 def=
        const
n:agent,mg:agent,t:text,x:text,y:text,nida:text,mgida:text,nidr:text,mgidr:text,chx:text,o
tt:text,h:hash_func,

    secNIDR, secOTT, secNa,secX,
secY,secMGIDR,secNb,secCone,secRone,secCHXnew,secNaa,secRtwo,secNOTT,sec
NnewIDA,na,nb:protocol_id

 intruder_knowledge = {n,mg,h}

        composition

                session(n,mg,t,x,y,nida,mgida,nidr,mgidr,chx,ott,h)
end role
```

Figure B_14: HLPSL code for role environment

```
Goal
        secrecy_of secNIDR
        secrecy_of secOTT
        secrecy_of secNa
        secrecy_of secX
        secrecy_of secY
        secrecy_of secNb
        secrecy_of secMGIDR
        secrecy_of secCone
        secrecy_of secRone
        secrecy_of secCHXnew
        secrecy_of secNaa
        secrecy_of secRtwo
        secrecy_of secNOTT
        secrecy_of secNnewIDA
        authentication_on na
        authentication_on nb
```

```
end goal
```

Figure B_15: HLPSL code for goal

APPENDIX C

```
role
role_V(TA:agent,V:agent,TAIDA:text,VIDA:text,VIDR:text,VTA:text,CHX:text,OTP
:text,H:hash_func,SND,RCV:channel(dy))

played_by V

def=
        local

                State:nat,
                TSoneV:text,TStwoV:text,
                TSoneTA:text,TStwoTA:text,RX:text,
                SIDNIG:text,
                Rone:text,
                Cone:text,Ctwo:text,CMG:text,
                CHXnew:hash(text.text),
                Xtwo:hash(text.text.text.text.text),
                Yone:hash(text.text.text),
                Ythreex:hash(message.text.text.text.text),
                NOTP:hash(text.text.text),
                NnewIDA:hash(text.text),
                Xthree:text,
                Xfour:hash(text.text.text.text.text.message),
                Na:text,Nb:text,Xone:text,Naa:text,Nbb:text

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
         =|> State' :=4 /\ TSoneV' :=new()
                                /\ Na' :=new()
                                /\ Xone' :=xor(Na',H(VTA))
                                /\ Xtwo' :=H(VIDR.TSoneV'.Na'.OTP.VTA)
                                /\  SND(VIDA.TSoneV'.xor(Na',H(VTA))
                                   .H(VIDR.TSoneV'.Na'.OTP.VTA))
                                /\ secret(VIDR,secVIDR,{V,TA})
                                /\ secret(OTP,secOTP,{V,TA})
                                /\ secret(Na,secNa,{V,TA})
                                /\ secret(VTA,secVTA,{V,TA})

            2. State= 4 /\ RCV(TAIDA.SIDNIG'.xor(Nb',H(VTA.OTP.Na')).
                                xor(Cone',H(Nb'.CHX)).TSoneTA'.Ythreex')=|>
            State' :=6
```

```
                        /\ TStwoV' :=new()
                        /\ Rone' := new()
                        /\ CHXnew' := H(CHX.Rone')
                        /\ Naa':=new()
                        /\ Xthree':=xor(Naa',H(Nb'.CHXnew'))
                        /\Xfour' := H(VIDR.TStwoV'.Rone'.Naa'.
                                    xor(Naa',H(Nb'.CHXnew')).H(CHX.Rone'))
                        /\SND(VIDA.SIDNIG'.TStwoV'.xor(Naa',H(Nb'.CHXnew'))
                          .H(VIDR.TStwoV'.Rone'.Naa'.xor(Naa',H(Nb'.CHXnew'))
                          .H(CHX.Rone')))
                       /\secret(Rone,secRone,{V,TA})
                       /\secret(CHXnew,secCHXnew,{V,TA})
                       /\secret(Naa,secNaa,{V,TA})
                       /\request(V,TA,na,Na)
                       /\witness(V,TA,nb,Nb')

          3. State=   6 /\ RCV(TAIDA.SIDNIG.xor(Nbb',H(Naa'.Rone')).
                           H(TStwoTA'.xor(Nbb',H(Naa'.Rone'))
                          .H(OTP.Nbb'.Rone').H(Cone'.VIDA)))
        =|> State' :=8
end role
```

Figure C_1: HLPSL code for role Vin T_VA authentication protocol

```
role role_TA
(TA:agent,V:agent,TAIDA:text,VIDA:text,VIDR:text,VTA:text,CHX:text,OTP:text,H:
hash_func,SND,RCV:channel(dy))

played_by TA

def=
        local
                State:nat,
                TSoneV:text,TStwoV:text,
                TSoneTA:text,TStwoTA:text,
                SIDNIG:text,
                Rone:text,
                Cone:text,CMG:text,
                CHXnew:hash(text.text),
                Yfour:hash(text.text),
                Ysix:hash(text.text.message.message),
                Yone:hash(text.text.text),
                NOTP:hash(text.text.text),
                NnewIDA:hash(text.text),
                Na:text,Nb:text,Naa:text,Nbb:text,
                Ytwo:text,
```

```
                Ythree:hash(message.text.text.text.text),
                Yfive:text,

                State:= 1
        transition

                1. State=1
                        /\RCV(VIDA.TSoneV'.xor(Na',H(VTA)).
                                H(VIDR.TSoneV'.Na'.OTP.VTA))
          =|> State' :=3

                                /\ TSoneTA' :=new()
                                /\ Nb':= new()
                                /\ Yone' := H(VTA.OTP.Na')
                                /\ Ytwo' := xor(Nb',H(VTA.OTP.Na'))
                                /\ Cone' := new()
                                /\ CMG'  := xor(H(Nb'.CHX),Cone')
                                /\ Ythree' := H(H(VTA.OTP.Na').TSoneTA'.Nb'.Cone'.Na')
                                /\ SIDNIG' := new()
                                /\ SND(TAIDA.SIDNIG'.xor(Nb',H(VTA.OTP.Na'))

.xor(Cone',H(Nb'.CHX)).TSoneTA'.H(H(VTA.OTP.Na').
                                        TSoneTA'.Nb'.Cone'.Na'))
                                /\secret(OTP,secOTP,{TA,V})
                                /\secret(Nb,secNb,{TA,V})
                                /\secret(Cone,secCone,{TA,V})
                                /\witness(TA,V,na,Na')

                2. State=3 /\ RCV(VIDA.SIDNIG.TStwoV'

.xor(Naa',H(Nb'.CHXnew')).H(VIDR.TStwoV'.Rone'.Naa'
                                        .xor(Naa',H(Nb'.CHXnew')).H(CHX.Rone')))
          =|> State' :=5
                                /\ TStwoTA' :=new()
                                /\ Nbb' := new()
                                /\Yfour' := H(Naa'.Rone')
                                /\Yfive' := xor(Nbb',H(Naa'.Rone'))
                                /\NOTP' := H(OTP.Nbb'.Rone')
                                /\NnewIDA' := H(Cone.VIDA)
                                /\Ysix' := H(TStwoTA'.xor(Nbb',H(Naa'.Rone'))
                                        .H(OTP.Nbb'.Rone').H(Cone.VIDA)
                                        .H(H(CHX.Rone').Na.Nb.Naa'.Nbb'))
                                /\SND(TAIDA.SIDNIG.xor(Nbb',H(Naa'.Rone'))
                                        .H(TStwoTA'.xor(Nbb',H(Naa'.Rone'))
                                        .H(OTP.Nbb'.Rone').H(Cone.VIDA)))
                               /\request(TA,V,nb,Nb)
```

```
                    /\secret(NOTP,secNOTP,{V,TA})
                    /\secret(NnewIDA,secNnewIDA,{V,TA})
end role
```

Figure C-2: HLPSL code for role TA

```
role
session(V:agent,TA:agent,VIDA:text,TAIDA:text,VIDR:text,VTA:text,CHX:text,OTP
:text,H:hash_func)
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_V(TA,V,TAIDA,VIDA,VIDR,VTA,CHX,OTP,H,SND1,RCV1)/\
                role_TA(TA,V,TAIDA,VIDA,VIDR,VTA,CHX,OTP,H,SND2,RCV2)
end role
```

Figure C_3. HLPSL code for role session

```
role environment()

def=

        const
v:agent,ta:agent,vida:text,taida:text,vidr:text,vta:text,chx:text,otp:text,h:hash_func,
        secVIDR,secOTP,secNa,secX,secNb,secCone,secRone,secCHXnew,secNaa,se
cNOTP,secVTA,secNnewIDA,na,nb:protocol_id

intruder_knowledge = {v,ta,h}

        composition

                session(v,ta,vida,taida,vidr,vta,chx,otp,h)

end role
```

Figure C_4. HLPSL code for role environment

```
.Goal
        secrecy_of secVIDR
        secrecy_of secOTP
        secrecy_of secNa
        secrecy_of secVTA
        secrecy_of secNb
        secrecy_of secCone
        secrecy_of secRone
        secrecy_of secCHXnew
        secrecy_of secNaa
        secrecy_of secNOTP
```

```
        secrecy_of secNnewIDA
        authentication_on na
        authentication_on nb
end goal
```

Figure C_5: HLPSL code for goal

```
role role_V(CS:
agent,V:agent,CSIDA:text,VIDA:text,VIDR:text,VCS1:text,VCS2:text,OTT:text,H:ha
sh_func,SND,RCV:channel(dy))

played_by V

def=
        local

        State:nat,
                TSoneV:text,TStwoV:text,
                VXtwo:hash(text.text.text.text.text),
                TSoneCS:text,TStwoCS:text,RX:text,
                SIDNIG:text,
                Rone:text,Rtwo:text,
                Cone:text,Ctwo:text,
                VXthree:hash(text.text),
                Yone:hash(text.text.text),
                Ythreex:hash(message.text.text.text.text),
                NOTT:hash(text.text.text),
                VnewIDA:hash(text.text.text),
                VXfour:text,
                VXfive:text,VXsix:text,
                VXseven:hash(message.text.text.text.text.message.message.message),
                XCSYfive:hash(text.message.message.message),
                Na:text,Nb:text,Xone:text,Naa:text,
                VXone:text,
                Sk:hash(text.text.text.text.text)

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
         =|> State' :=4 /\ TSoneV' :=new()
                                /\ Na' :=new()
                                /\ VXone' :=xor(Na',H(VCS2.OTT))
                                /\ VXtwo' :=H(VIDR.TSoneV'.Na'.OTT.VCS1)
                                /\ SND(VIDA.TSoneV'.xor(Na',H(VCS2.OTT)).
                                        H(VIDR.TSoneV'.Na'.OTT.VCS1))
```

```
                        /\ secret(VIDR,secVIDR,{V,CS})
                        /\ secret(OTT,secOTT,{V,CS})
                        /\ secret(Na,secNa,{V,CS})
                        /\ secret(VCS1,secVCS1,{V,CS})
                        /\ secret(VCS2,secVCS2,{V,CS})

        2. State= 4 /\ RCV(CSIDA.SIDNIG'.xor(Nb',H(VCS1.OTT))
                           .xor(Cone',H(Nb'.H(VCS1.OTT))).TSoneCS'.H(VIDR.H
                           (VCS1.OTT).TSoneCS'.Nb'.Cone'.Na'))

        =|>   State' :=6

                        /\ TStwoV' :=new()
                        /\ Rone' := new()
                        /\ Ctwo' := H(Na'.Nb')
                        /\ Rtwo':=new()
                        /\ Naa':=new()
                        /\ VXthree' := H(OTT.Nb')
                        /\ VXfour':=xor(Naa',H(H(OTT.Nb').VIDR))
                        /\VXfive' :=xor(Rone',H(H(OTT.Nb').Naa'))
                        /\VXsix' := xor(Rtwo',H(Rone'.Cone'))
                        /\NOTT' := H(OTT.Naa'.Nb')
                        /\VnewIDA' := H(Cone'.VIDA.OTT)
                        /\VXseven' := H(H(OTT.Nb').
                                        TStwoV'.Rone'.Naa'.Rtwo'.
                                   H(Na'.Nb').H(OTT.Naa'.Nb').
                                   H(Cone'.VIDA.OTT))

                        /\ SND(VIDA.SIDNIG.TStwoV'
.xor(Naa',H(H(OTT.Nb').VIDR))
                                .xor(Rone',H(H(OTT.Nb').Naa')).xor(Rtwo',H
                                (Rone'.Cone')).H(H(OTT.Nb').

TStwoV'.Rone'.Naa'.Rtwo'.H(Na'.Nb').H(OTT.Naa'.Nb').H
                                (Cone'.VIDA.OTT)))
                        /\secret(Rone,secRone,{V,CS})
                        /\secret(Rtwo,secRtwo,{V,CS})
                        /\secret(Naa,secNaa,{V,CS})
                        /\secret(NOTT,secNOTT,{V,CS})
                        /\secret(VnewIDA,secVnewIDA,{V,CS})
                        /\request(V,CS,na,Na)
                        /\witness(V,CS,nb,Nb')
end role
```

Figure C_6: HLPSL code for V in V-VOCS authentication protocol

```
role role_CS
(CS:agent,V:agent,CSIDA:text,VIDA:text,VIDR:text,OTT:text,TID:text,H:hash_func,
SND,RCV:channel(dy))

played_by CS

def=
        local
                State:nat,
                VCS1:text,VCS2:text,
                TSoneV:text,TStwoV:text,
                TSoneCS:text,
                SIDNIG:text,
                Rone:text,Rtwo:text,
                Cone:text,CV:text,
                CSYone:hash(text.text),
                CSYtwo:text,
                CSYthree:hash(text.message.text.text.text.text),
                NOTT:hash(text.text.text),
                VnewIDA:hash(text.text.text),
                Na:text,Nb:text,Naa:text


        init
                State:= 1
        transition

                1. State=1
                        /\RCV(VIDA.TSoneV'

.xor(Na',H(VCS2'.OTT)).H(VIDR.TSoneV'.Na'.OTT.VCS1'))
            =|> State' :=3

                                /\ TSoneCS' := new()
                                /\ Nb':= new()
                                /\ CSYone' :=H(VCS1'.OTT)
                                 /\ CSYtwo' :=xor(Nb',H(VCS1'.OTT))
                                /\ Cone' :=new()
                                /\ CV' :=xor(Cone',H(Nb'.H(VCS1'.OTT)))
                                /\ CSYthree' :=
H(VIDR.H(VCS1'.OTT).TSoneCS'.Nb'.Cone'.Na')
                                /\ SIDNIG' := new()
                                /\ SND(CSIDA.SIDNIG'.xor(Nb',H(VCS1'.OTT))
                                        .xor(Cone',H(Nb'.H(VCS1'.OTT))).TSoneCS'
                                         .H(VIDR.H(VCS1'.OTT).TSoneCS'.Nb'.Cone'.Na'))
                                /\secret(OTT,secOTT,{CS,V})
```

```
                    /\secret(Nb,secNb,{CS,V})
                    /\secret(Cone,secCone,{CS,V})
                    /\witness(CS,V,na,Na')
          2. State=3 /\ RCV(VIDA.SIDNIG.TStwoV'
                              .xor(Naa',H(H(OTT.Nb').VIDR))
                              .xor(Rone',H(H(OTT.Nb).Naa'))
                              .xor(Rtwo',H(Rone'.Cone')).H(H(OTT.Nb').TStwoV'
                              .Rone'.Naa'.Rtwo'.H(Na'.Nb').H(OTT.Naa'.Nb')
                              .H(Cone'.VIDA.OTT)))
        =|> State' :=5
end role
```

Figure C_7: HLPSL code for V

```
role
session(V:agent,CS:agent,CSIDA:text,VIDA:text,VIDR:text,VCS1:text,VCS2:text,OT
T:text,TID:text,H:hash_func)
def=
        local
                SND2,RCV2,SND1,RCV1:channel(dy)
        composition
                role_V(CS,V,CSIDA,VIDA,VIDR,VCS1,VCS2,OTT,H,SND1,RCV1)/\
                role_CS(CS,V,CSIDA,VIDA,VIDR,OTT,TID,H,SND2,RCV2)

end role
```

Figure C_8: HLPSL code for session

```
role environment()

def=

        const
v:agent,cs:agent,csida:text,vida:text,vidr:text,vcs1:text,vcs2:text,ott:text,tid:text,h:hash
_func,
        secVIDR,secOTT,secNa,secNb,secCone,secRone,secRtwo,secNaa,secNOTT,se
cVCS1,secVCS2,secVnewIDA,na,nb:protocol_id

intruder_knowledge = {v,cs,h}

        composition

                session(v,cs,csida,vida,vidr,vcs1,vcs2,ott,tid,h)

end role
```

Figure C_9: HLPSL code for environment

```
Goal

        secrecy_of secVIDR
        secrecy_of secOTT
        secrecy_of secNa
        secrecy_of secVCS1
        secrecy_of secVCS2
        secrecy_of secNb
        secrecy_of secCone
        secrecy_of secRone
        secrecy_of secRtwo
        secrecy_of secNaa
        secrecy_of secNOTT
        secrecy_of secVnewIDA
        authentication_on na
        authentication_on nb
end goal
```

Figure C_10: HLPSL code for goals

```
role
role_U(CS:agent,V:agent,CSIDA:text,VIDA:text,VIDR:text,VCS1:text,VCS2:text,OTT:text,UI:text,Rone:text,SIDNIG:text,Sk:symmetric_key,H:hash_func,SND,RCV:channel(dy))

played_by V

def=
        local

                State:nat,
                TSoneV:text,
                Uone:hash(text.text.text),
                Utwo:text,
                Uthree:hash(text.message.message.text),
                TSoneCS:text,
                PUI:text,FP:text,
                ZUone:hash(text.text),
                Uv:hash(text.text.text.message)

        init
                State:= 2
        transition

                1. State= 2 /\ RCV(start)
         =|> State' :=4       /\ TSoneV' :=new()
```

```
                        /\ Uone' :=H(VIDR.Rone.OTT)
                        /\ FP' :=new()
                        /\ PUI' :=new()
                        /\ ZUone' :=H(FP'.PUI')
                        /\ Utwo' := xor(Uone',ZUone)
                        /\ Uthree' := H(VIDR.Uone'.ZUone'.TSoneV')

                        /\ SND(VIDA.TSoneV'.UI.
                              {xor(Uone',ZUone').
                               H(VIDR.Uone'.ZUone'.TSoneV')}_Sk)
                        /\ secret(VIDR,secVIDR,{V,CS})
                        /\ secret(OTT,secOTT,{V,CS})
                        /\ secret (PUI,secPUI,{V,CS})
                        /\ witness(V,CS,uone,Uone')

          2. State= 4 /\ RCV(CSIDA.SIDNIG.TSoneCS'.{Uv'}_Sk)

        =|>   State' :=6

end role
```

Figure C_11: HLPSL code for V

```
role role_CS
(CS:agent,V:agent,CSIDA:text,VIDA:text,VIDR:text,VCS1:text,VCS2:text,OTT:text,
UI:text,Rone:text,SIDNIG:text,Sk:symmetric_key,H:hash_func,SND,RCV:channel(dy
))

played_by CS

def=
        local
                State:nat,
                TSoneV:text,
                TSoneCS:text,
                Uone:hash(text.text.text),
                ZUone:hash(text.text),ZUthree:text,
                Uv:hash(text.text.text.message),
                PUI:text
        init
                State:= 1
        transition

                1. State=1
                        /\RCV(VIDA.TSoneV'.UI. {xor(Uone',ZUone').
                            H(VIDR.Uone'.ZUone'.TSoneV')}_Sk)
           =|> State' :=3
```

```
                    /\ TSoneCS' := new()
                    /\ Uv' := H(TSoneCS'.Rone.OTT.ZUone')

                    /\ SND(CSIDA.SIDNIG.TSoneCS'.
                          {H(TSoneCS'.Rone.OTT.ZUone')}_Sk)
                    /\request(CS,V,uone,Uone')


end role
```

Figure C_12: HLPSL code for CS

```
role
session(V:agent,CS:agent,CSIDA:text,VIDA:text,VIDR:text,VCS1:text,VCS2:text,OT
T:text,UI:text,Rone:text,SIDNIG:text, Sk:symmetric_key ,H:hash_func)

def=
        local
                SND2, RCV2, SND1, RCV1: channel (dy)
        composition

        role_V(CS,V,CSIDA,VIDA,VIDR,VCS1,VCS2,OTT,UI,Rone,SIDNIG,Sk,
H,SND1,RCV1)/\
                role_CS (CS, V, CSIDA, VIDA, VIDR, VCS1, VCS2, OTT, UI, Rone,
SIDNIG, Sk, H, SND2, RCV2)

end role
```

Figure C_13: HLPSL code for session

```
role environment()

def=
        const
v:agent,cs:agent,csida:text,vida:text,vidr:text,vcs1:text,vcs2:text,ott:text,ui:text,rone:te
xt,sidnig:text,sk:symmetric_key,h:hash_func,secVIDR,secOTT,secRone,secSk,secPUI,
uone:protocol_id

intruder_knowledge = {v,cs,h}

        composition

                session(v, cs,csida,vida,vidr,vcs1,vcs2,ott,ui,rone,sidnig,sk,h)

end role
```

Figure C-14: HLPSL code for environment

```
Goal
        secrecy_of secVIDR
        secrecy_of secOTT
        secrecy_of secRone
        secrecy_of secSk
        secrecy_of secPUI
        authentication_on uone
end goal
```

Figure C_15: HLPSL code for goals

CURRICULUM VITA

Samah S. Mansour, PhD

School of Computing and Information Systems

College of Engineering and Computing

Grand Valley State University

D-2-228 Mackinac Hall
1 Campus Drive

Allendale, MI 49418

Tel: (616) 331-3051
Fax: (616) 331-2106
Email: mansours@gvsu.edu

EDUCATION

2017-Aug 2020 **Ph.D. in Computer Science and Engineering**

University of Louisville, Louisville, Kentucky, USA

Academic Advisor: Adrian Lauf, PhD.

2013-2017 **M.S. in Computer Information Systems**

Grand Valley State University, College of Engineering and Computing,

Allendale, Michigan, USA

Thesis Title: A Comparative Study Among Mobile Forensics Tools for Android Based Smartphones

Academic Advisor: Andrew Kalafut, PhD.

2002-2006 **Ph.D. in Instructional Technology and Development**

University of Louisville, Louisville, Kentucky, USA

Dissertation title: Collaborative Virtual Environment: The Fourth Generation of Communication Media to Increase Social Interaction and Social Presence in E-Learning.

Academic Advisor: Carolyn Rude-Parkins, PhD.

2001-2002 **M.A. in Leadership and Administration of Higher Education**
University of Louisville, Louisville, Kentucky, USA

1996-2000 **Bachelor of Art**
Ain Shams University, Cairo, Egypt
Major: English.
Minor: German and Arabic.

ACADEMIC AWARDS

- 2008, Nominated to Marquis Who's Who Award
- 2006, Who's Who Among Students in American Universities and Colleges Award.
- 2006, Graduate Dean's Citation Award.
- 2006, Nominated to Hauchens prize for the best dissertation in the university, University of Louisville, KY.
- 2002, Graduated master degree with distinction, University of Louisville, KY.
- 1995-1999, Undergraduate scholarship, Ain Shams University, Cairo, Egypt.

APPOINTMENTS

- Aug 2020-Current, Assistant Professor, School of Computing
- Aug 2008-May 2020, Affiliate Faculty, School of Computing and Information Systems, Grand Valley State University.
- June 2010 - June 2011, Assistant Professor, School of Business, Canadian International College (CIC)
- Sep 2006- May 2007, Part-Time Faculty, Foundations & Technology Department and Department of Statistics, Grand Valley State University.
- Jan 2007-April 2007, Online Part-Time Faculty, Department of Leadership, Foundations & Human Resource Education, University of Louisville.
- Aug 2004-May 2006, Research Assistant, Department of Leadership, Foundations & Human Resource Education, University of Louisville.
- Aug 2003-Aug 2004, Instructor, Business Administration Department and Mathematics Department, Pennsylvania State University, New Kensington Campus.
- Aug 2003-Aug 2004, Technology Coach, Pennsylvania State University, New Kensington Campus.
- Jan 2002-Jul 2002, Part-Time Instructional Technology Specialist, Delphi Center for Teaching and Learning, University of Louisville.

- Nov 2001-July 2003, Program Assistant, Housing and Residence Life, University of Louisville.
- Sep 1999-Nov 2001, Substitute Teacher, Jefferson County Public School, Louisville, KY.

EMPLOYMENT

Aug 2020-Current	**Assistant Professor**

Grand Valley State University, School of Computing, Allendale, MI

- Teaching undergraduate and graduate courses.
- Conducting research in digital forensics, cyber security and data science.
- Involved in numerous professional, university, and school services.

Aug 2008-May 2020	**Affiliate Faculty**

Grand Valley State University, School of Computing and Information Systems, Allendale, MI

- Teaching undergraduate courses
- Conducting research in digital forensics, cyber security and data science
- Involved in numerous professional, university, and school services

April 2010-June 2011	**Assistant Professor**

Canadian International College, School of Business Technology, Cairo, Egypt

- Taught Undergraduate courses

Sep 2006 - May 2007	**Part-Time Faculty**

Grand Valley State University, Foundations & Technology Department,

Grand Rapids, MIGrand Valley State University, Department of

Statisitics, Allendale, MI

- Taught undergraduate and graduate courses.

Jan 2007-April 2007	**Online Part-Time Faculty**

University of Louisville, Department of Leadership, Foundations & Human Resource Education, Louisville, KY

- Taught online graduate courses.

Aug 2003– Aug 2004 **Instructor**

Pennsylvania State University, Business Administration Department, New Kensington, PA

Pennsylvania State University, Mathematics Department, New Kensington, PA

- Taught undergraduate courses.

Aug 2003-Aug 2004 **Technology Coach**

The Pennsylvania State University, New Kensington, PA

- Collaborated with the Nursing School professors to design and develop on-line quantitative statistics courses.
- Collaborated, coached and advised the NK professors on creating instruction that integrated technology into their pedagogy.
- Supported NK professors' development by tutoring and coaching professors in Microsoft Office, Adobe Acrobat, Angel, Macromedia Flash, Adobe Photoshop, and operating computer peripherals.

2004- 2006 **Research Assistant**

University of Louisville, Department of Leadership, Foundations & Human Resource Education, Louisville, KY

Worked with my advisor on the following research projects:

- Design and implementation of a virtual environment called *The Survival.*
- Design and implementation of a virtual environment *called Internet Teaching Lab.*
- Interactive and Independent Effect of Visual Fidelity and Behavioral Fidelity of Avatar on the Perception of Social Interaction and Social Presence *(Experimental Studies.)*
- Impact of Collaborative Virtual Environment as a Communication Medium *(Experimental Studies).*

Jan 2002-Jul 2002 **Part-Time Instructional Technology Specialist**

University of Louisville, Delphi Center for Teaching and Learning, Louisville, KY

- Collaborated with other instructional specialists, technicians, and computer programmers to design and develop online non-credit courses.

- Designed and developed PowerPoint presentations, Multimedia presentations using Macromedia Flash, web-based course materials, and instructional graphics.

Nov 2001-July 2003 Program Assistant

University of Louisville, Louisville, KY, Housing and Residence Life Department

- Developed and taught online training modules for full-time office staff, student assistants, resident directors, and resident assistants.
- Designed self-paced training modules for desk-staff students.
- Conducted face-to-face training seminars for the resident directors, resident assistants and desk-staff students.
- Helped the Unit Business Manager in creating and keeping updated electronic budgetary reports.

TEACHING

- **Grand Valley State Univ., School of Computing**
    - CIS 150: Introduction to Computing
    - CIS 231: Problem Solving using Spreadsheet
    - CIS 221: Excel concepts and Applications I
    - CIS 309: Teaching Computer Science
    - CIS 331: Data Analysis Tools and Techniques
    - CIS 335: Data Mining
    - CIS 321: Excel concepts and Applications II
    - CIS 615: Info. Security Principles

- **Grand Valley State Univ., Statistics Department**
    - STA 215: Introductory Applied Statistics

- **Grand Valley State Univ., Foundations & Technology Department**
    - ED 205: Computers in Education
    - EDG 619 - Curricular Integration of Educational Technology

- **The Pennsylvania State Univ., Business Administration Department.**
    - MIS 103: Microcomputer Applications in Business.

- **The Pennsylvania State Univ., Mathematics Department.**

- STAT 200: Elementary Statistics.

- **Canadian International College, School of Business Technology, Cairo, Egypt**

  - BTEC 107: Internet Fundamentals
  - BTEC 319: Business Strategies in IT
  - BTEC 105: PC Hardware Fundamentals
  - BTEC 311: Visual Basic

- **University of Louisville, Department of Leadership, Foundations & Human Resource Education**

  - ELFH 600 (Co-teaching): Introduction to Research Methods and Statistics (online course).
  - ELFH 664: Facilitating Change in Organizations (online course).

CURRICULUM DEVELOPMENT

- **Cybersecurity B.S. New Program. (May 2018 – Winter 2019)**
  - Contributed to the development of the prospectus and the proposal for the new Cybersecurity undergraduate major.

- **Cybersecurity M.S. New Program. (May 2018 – Winter 2019).**
  - Led the task force effort in developing prospectus and the proposal for the new Cybersecurity graduate program.

- **New B.S./M.S. Combined Degree in Cybersecurity**
  - Authored four new program B.S./M.S. proposals and study plans for cybersecurity, Computer Science and Cybersecurity, Information Systems and Cybersecurity, and Information Technology and Cybersecurity

B.S. Information Technology B.S. New Program (W-17)
  - Contributed to the development of the prospectus and the proposal for the new Information Technology major

- **Cyber Range Proposal Task Force**
  - This task force was charged with writing a proposal for GVSU to host a Michigan Cyber Range hub. We wrote the proposal, but the decision was made not to submit it.

- **Program Change Request for the Healthcare Information Systems Minor**
  - I wrote the proposal for a program change request

- The goal of the change was to include CIS 331 as an option for students who are seeking the HIS minor

- **CIS 280/ 221: Excel Concepts and Applications I (August 2019-Current)**
  - I wrote the proposal for a new Excel Concepts and Applications I course.

- **CIS 380/ 322: Excel Concepts and Applications I**
  - Currently, I am working on the development of this course

- **CIS 455/555-Applied Cryptography**
  - Dr. Andrew Kalafut and I wrote the proposal for a new Applied Cryptography course.

- **CIS 619-Data Analytics for Cybersecurity**
  - I wrote the proposal for a new Data Analytics for Cybersecurity course.

- **CIS 331: Data Analysis Tools and Techniques**
  - Dr. Jonathan Leidig, Dr. Greg *Schymik*, and I wrote a course change proposal in order to change the course title, description and content.
  - I developed the course material and started to teach this course from Fall 2018

- **CIS 231: Problem Solving Using Spreadsheet**
  - Changed the course structure and developed new material

PUBLICATIONS

- 2019, Mansour, S and Lauf, A. "Hardware Root of Trust for IoT Security In Smart Home Systems" (submitted). IEEE Consumer Communications & Networking Conference. 10-13 January 2020, Las Vegas, USA.

- 2018, Mansour, S "Social Media Analysis of Users' Responses to Terrorism Using Sentiment Analysis and Text Mining". Cyber Physical Systems and Deep Learning Conference, November 5-7 2018, Chicago, IL

- 2018, El-Said, M. M., Mansour, S., Bhuse, V, "*DSRC Based Sensor-Pooling Protocol for Connected Vehicles in Future Smart Cities*" The Cyber Physical Systems and Deep Learning Conference, November 5 - 7 2018, Chicago, Illinois, USA

- 2017, A Comparative Study Among Mobile Forensics Tools for Android Based Smartphones. M.Sc. Thesis, College of Engineering and Computing, Grand Valley State University

- 2017, El-Said, M. M., Arendsen, A., & Mansour, S. S. "*DSRC Performance Analysis in Foggy Environment for Intelligent Vehicles System*." The International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 5 Issue: 5, ISSN: 2321-8169, 2017, pp 807-812. (Work is done in 2016 and published after review in 2017)

- 2017, Mansour, S. "Terrorist Watcher: An Interactive Web-based Visual Analytical Tool of Terrorist's Personal Characteristics". International Journal of Data Mining & Knowledge Management Process (IJDKP), Volume 6, ISSN: 2230 - 9608[Online]; 2231 - 007X

- 2017, El-Said, M. M., Arendsen, A., & Mansour, S. M. A "*Lightweight Message Authentication Framework in the Intelligent Vehicles System*" The International Journal of Engineering And Science (IJES), Volume 06 - Issue 06, ISSN (e):2319 –1813 ISSN (p):2319 –1805, pp 33-39
- 2016, Mansour, S. "Cloud Computing and Digital Forensics Challenges". 7th Annual International Conference on ICT: Big Data, Cloud and Security (ICT-BDCS 2016)., Singapore.

- 2016, El-Said, M. M., Arendsen, A., & Mansour, S. S. "DSRC Performance Analysis in Foggy Environment for Intelligent Vehicles System." *The International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 4 Issue: 12, ISSN: 2321-8169, 2016*

- 2016, El-Said, M. M., Arendsen, A., & Mansour, S. M. A "Lightweight Message Authentication Framework in the Intelligent Vehicles System." *The International Journal of Engineering And Science (IJES), Volume 06 - Issue 12 ISSN: 2319 – 1813 ISBN: 2319 – 1805, 2016*

- 2015, El-Said M, Arendsen A and Mansour S "*Detect and Defend System on a Stick (D^2S^2) Against GPS Spoofing Attack*", The 14th Annual Security Conference is scheduled for May 19-21, 2015 Las Vegas, Nevada, USA

- 2012, Mansour S and El-Said M *"Building a Bi-Directional Bridge Between Social Presence and Interaction in Online Games"* 17th International Conference on Computer Games, (CGAMES 2012)

- 2011, Mansour S and El-Said M *"Building a Bi-Directional Bridge Between Social Presence and Interaction in Online Games"* 17th International Conference on Computer Games, (CGAMES 2012)

- 2010, Mansours, S. El-Said, M. and Bennett, L. "*Does the Use of Second Life Affect Students' Feeling of Social Presence in E-Learning?*" The 8th International Conference on Education and Information Systems, Technologies and Applications: EISTA 2010**, June 29th - July 2nd, 2010 – Orlando, Florida, USA**

- 2009, Mansour, S & Reynolds, J. Development of a Baccalaureate Major in Information Technology: Adding a Third Dimension to a Comprehensive Computing Program. *Proceedings of the 10th ACM conference on SIG-Information Technology Education, Fairfax, Virginia, USA*

- 2009, El-Said, M. & Mansour, S "*Game Based Learning Creating a Triangle of Success: Play, Interact and Learn*". International Journal of Intelligent Games & Simulation. (*Invited Paper*).

- 2009, Mansour, S., Rude-Parkins, C., & Bennett, L. An Empirical Study: Assessment of Students' Learning performance Using 3D Leaning Environments in Online Courses. *Journal of Systemics, Cybernetics and Informatics*

- 2008, Mansour, S., & El-Said, M. Multi-Player Role Playing Games: A Link between Fun and Learning. *The International Journal of Learning, 15 (11)* 229-240

- 2008, Mansour, S & El-Said, M. The Relationship between Educational Serious Games, Gender, and Students' Social Interaction. *WSEAS Transactions on Computers, 7(6)*640-649.

- 2008, Mansour, S., El-Said, M., & Nandigam, J. (2008). An Empirical Study to Measure Students Learning Performance Using Serious Games. *The 12th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia & Serious Games*

- 2008, Mansour, S., Rude-Parkins, C., & Bennett, L. (2008). *How the Use of Second Life Affects E-Learners' Perceptions of Social Interaction in Online Courses*. Proceedings of the 6th International Conference on Education and Information Systems, Technologies and Applications (EISTA). Orlando, Florida

- 2008, Mansour, S., & El-Said, M. (2008). The Impact of Multi-Players Serious Games on the Social Interaction among Online Students versus Face-to-Face Students. Proceedings of the 7th

International conference on Advances on Applied Computer and Applied Computational Science (WSEAS), Hangzhou, China

- 2007, S. Mansour, M. El-Said, C. Rude-Parkins, & J. Nandigam. The Interactive Effect of Avatar Visual Fidelity and Behavioral Fidelity in the Collaborative Virtual Reality Environment on the Perception of Social Interaction, *WSEAS Transactions on Communications*, 8 (5) 1501-1509.
- 2006, Mansour, S., El-Said, M., Rude-Parkins, C., & Nandigam, J. (2006). *The Interactive Effect of Avatar Visual Fidelity and Behavioral Fidelity in the Collaborative Virtual Reality Environment on the Perception of Social Interaction.* Proceedings of the 10th International conference of World Scientific and Engineering Academy and Society (WSEAS). Athens, Greece.

- 2006, Mansour, S., Rude-Parkins, C., & El-Said, M. *The Effect of the Type of Communication Medium on Learners' Perceptions of Social Interaction in E-Learning*. In E. Pearson & P. Bohman (Eds.), Proceedings of the 17th World Conference on Educational Multimedia, Hypermedia & Telecommunications (ED-MEDIA) 2006 ((pp. 1274-1281), Orlando, Florida.

- 2005, Mansour, S., Rude-Parkins, C., & El-Said M. *The Relationship between the Avatar's Behavioral Fidelity and Social Interaction in 3d Collaborative Learning-Based Games.* Proceedings of the 7th International Conference on Computer Games: AI and Mobile Systems (CGAIM 2005), Angoulem, France.

- 2005, El-Said, M. & Mansour, S. *A Hybrid Anthropomorphism Model To Enhance The Social Presence In 3D Virtual Multi-Players Games*. Proceedings of the 6th International Conference on Computer Games: AI and Mobile Systems (CGAIM 2005), Louisville, KY.

GRANTS

- 2019, CyberGen (will be submitted on Oct 25, 2019) *(*Joint collaboration with *Bhuse, V, El-Said, M., Kalafut, A., Wang, X*

- 2018, *Grand Valley State University Cyber Range Hub Site Initiative", Michigan Economic Development Corporation (MEDC), $259,800.00, (*Joint collaboration with *Leidig, P., El-Said, M., Kalafut, A., Bhuse, V. Wang, X* (we did not submit it)

- 2016, XRY Complete Digital Forensic Kit" Sponsored by the MSAB Incorporated, (Mobile Forensic Trial Kit - worth $7,990.00 provided by MSAB Corporate). (Joint collaboration with *El-Said, M.)*

- 2016, Digital Forensics for Mobile Devices. Sponsored by CSCE, Grand Valley State University, $400.00.

- 2013, A Novel Approach in the Delivery of Occupational Safety and Health Training for the Beverage Industry". MIOSHA (Michigan Occupational Safety and Health Administration. Funded $45,000.00. (Joint collaboration with Huizen, D.)

- 2010, Using Second Life for Developing Interactive Games to Teach Computer Science Programming Skills. $1,980.00 ((funded by the FTLC Grand Valley State Univ).

- 2009, Web Based Mobile Learning environment (M-WEB-Learning). 5,274.00 (funded by the FTLC Grand Valley State Univ)

- 2008, Immersive Education: using Second life to teach computing and information systems, 4,274.00 (funded by the FTLC Grand Valley State Univ) (Joint collaboration with Nandigam, J)

- 2008, Proactive Intrusion Detection System (IDS) using Baseline Recording, Analysis and Real time Monitoring $4,912.50 *(funded by* the FTLC Grand Valley State Univ) (Joint collaboration with El-Said, M)

- 2006, Learning by Doing: An Adapted Teaching Philosophy in the Wireless Courses, $1750 (funded by the FTLC, Grand Valley State Univ) (Joint collaboration with El- Said, M).

- 2006, Building an Interactive Collaborative Virtual Reality Teaching Environment, $2970 (funded by FTLC-Grand Valley State Univ) (Joint collaboration with El- Said, M.).

- 2005, Building an Interactive Constructivist Approach in Teaching the Next Generation of Autonomic Mobile Computing Networks, $1560 (funded by FTLC-Grand Valley State Univ) (Joint collaboration with Prof. El- Said,M.).

PROFESSIONAL ACTIVITIES

Reviewer for Conferences and Journals

- 10th International Conference on Society and Information Technologies: ICSIT 2019
- Complex Adaptive Systems conference 2018
- 8th International Conference on Society and Information Technologies: ICSIT 2017
- SIGITE/RIIT, 2016
- 7th International Conference on Society and Information Technologies: ICSIT 2016
- International Conference on Society and Information Technologies: ICSIT 2015
- The 19th International Computer Games Conference, AI, Animation, Interactive Multimedia, Virtual Worlds and Serious Games, (CGAMES 2014), Louisville, Kentucky. - International Conference on Society and Information Technologies: ICSIT 2014
- member of the ISTE SIG 1-to-1 Computing,2013
- International Conference on Society and Information Technologies: ICSIT 2013
- International Conference on Society and Information Technologies: ICSIT 2012
- International Conference on Society and Information Technologies: ICSIT 2011 o International Conference on Engineering and Meta-Engineering: ICEME 2011
- The 17th International Computer Games Conference: AI, Interactive Multimedia, Virtual Worlds and Serious Games (CGAMES'12)
- International Conference on Engineering and Meta-Engineering: ICIME 2010
- The 3rd International Multi-Conference on Engineering and Technological Innovation (IMETI 2010)
- International Conference on Society and Information Technologies: ICSIT 2010
- Journal of Computer Assisted Learning
- The International Journal of Learning

Professional Society Memberships

- Internet of Things, West Michigan (IoTWM), Member, Grand Rapids, MI, USA, (2017 - Present)
- West Michigan Cyber Security Consortium, Board Member, Grand Rapids, MI, USA, (2017 - Present).

ADVISING and MENTORING STUDENTS' PROJECTS

- **Honors Senior Project: Investigating the Sense of Belonging Among Various Super Smash Bros. Gaming Communities**
  - The goal of this research project was to analyze the perception of individuals within the Melee, Project M, and Smash 4 communities to see if there is any variation in the sense of belonging in each community and if so, why?

- **Data Parsing**

- This project was developed for the Little Sprouts pre-kindergarten schools as a collaboration with Take Flight Enterprises, LLC. The specific procedure was part of an overhaul of the schools' current records. The project work involves the following tasks: opening an Excel data file, importing the output file from the WebGUI, and running a parsing procedure.

- **Production Capacity**
  - This project focused on developing a simulation a tool that can be used in a real-world industry setting. The simulation tool calculates how much capacity is needed in the upcoming periods of the machine's running time window as well as it shows how much that machine's capacity will cost. One of the main features of this project was how to figure out the product's unit price, which depends on several factors such as: industry average price, industry average quantity sold, production capacity, beginning inventory, and ending inventory.

- **Improving Mental Math: The Method Behind the Program –**
  - The main objective of the game was to test the basic mental math skills of students. The program is allowing the teacher to set the range of numbers to be included in the test, the amount of questions, and a time limit. Then they would be able to administer the test to the class and monitor the results. Based on those results, the teacher could cater any review needed changed. The program allows the students to log in, save and send their scores to their teachers. This would allow the teacher to establish trends in the individual students score. With this, the teacher could then work on a specific range for each student and instead of using one quiz, each student could work on their own personalized quiz.A pilot study was conducted at Central Grand Rapids High School (CHS). Based on the students' input, the student changed some aspects of the project. The teacher in CHS decided to adopt and use the game in their classes.

- **Fractions Are Our Friends**
  - This game was designed to teach fractions to elementary school students. In order to validate the impact of the game on helping students to get a better understanding of the fraction concepts.
  - A crossover experimental design was used to conduct the experiment
  - The experiment was designed to answer the following questions:
    - Does the use of a game as a teaching tool impact the students' learning?
    - What is the impact of providing students with instant feedback on motivating students to answer the questions correctly?
    - Is there a relationship between the format of delivering the quiz (paper/game) and the students' inspiration to take the quiz?
    - What is the impact of using educational serious games on motivating students from low socioeconomically status and low performance level in math to learn mathematical complex concepts?

SERVICE

- **Grand Valley State University**

  - Women in Computing

- GVSU Engineering Engineer Day
- Revise the required courses for the IS and IT minor programs.

- **The Pennsylvania State University**

  - Spring 2004, Committee for internationalizing the curriculum.
  - Fall 2003-Spring 2004, Committee on Diversity.
  - Fall 2003-Spring 2004, Consultant for the design and development of online statistics courses for the School of Nursing.

- **The Housing and Residence Life Department, University of Louisville**

  - Spring 2003, Resident Director Search Committee.
  - Spring 2003, Student Life Awards Committee.
  - Fall 2002, Assistant Director For Administrative Services Search Committee.
  - Fall 2002, Associate Director for Facilities Search Committee.
  - Spring 2002, Director Search Committee.
  - Spring 2001-Spring 2003, Annual Resident Assistants Retreat Planning Committee.