MASTER THESIS NO. 2022:16

College of Information Technology

Department of Information Systems and Security

AUTHENTICATED KEY ESTABLISHMENT PROTOCOL FOR CONSTRAINED SMART HEALTHCARE SYSTEMS BASED ON PHYSICAL UNCLONABLE FUNCTION

Abdalla Saleh Khalifa Elkushli

Sensor Level
Gateway Level
Medical Server Level
Inertial sensor
ECG sensor
PPG sensor
Motion sensors
WWW

April 2022

United Arab Emirates University

College of Information Technology

Department of Information Systems and Security

AUTHENTICATED KEY ESTABLISHMENT PROTOCOL FOR CONSTRAINED SMART HEALTHCARE SYSTEMS BASED ON PHYSICAL UNCLONABLE FUNCTION

Abdalla Saleh Khalifa Elkushli

This thesis is submitted in partial fulfilment of the requirements for the degree of Master of Science in Information Security

April 2022

United Arab Emirates University Master Thesis
2022: 16

Cover: Securing smart healthcare systems

(Photo: By Abdalla Saleh Khalifa Elkushli)

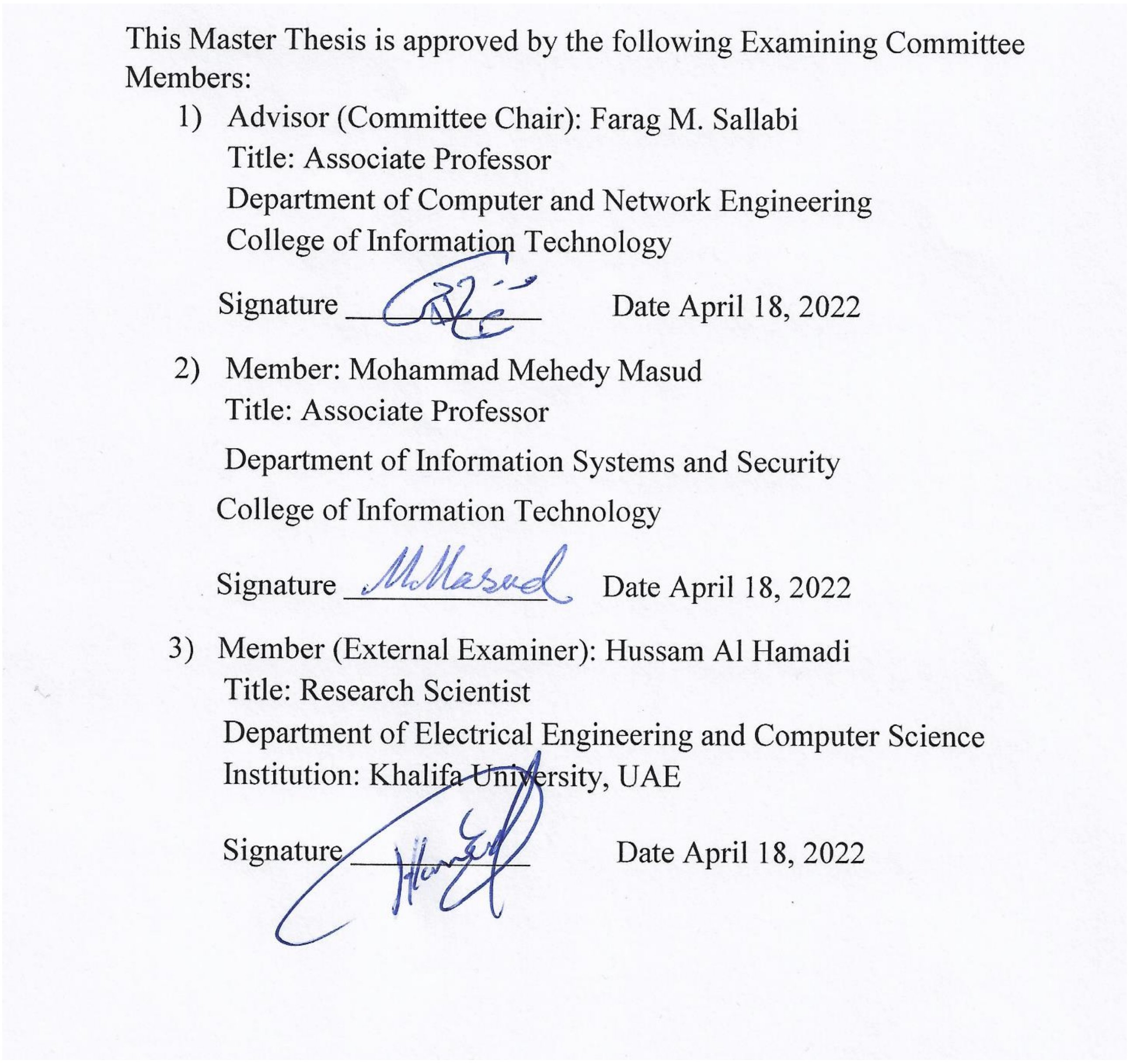
Declaration of Original Work

I, Abdalla Saleh Khalifa Elkushli, the undersigned, a graduate student at the United Arab Emirates University (UAEU), and the author of this thesis entitled "*Authenticated Key Establishment Protocol for Constrained Smart Healthcare Systems Based on Physical Unclonable Function*", hereby, solemnly declare that this thesis is my own original research work that has been done and prepared by me under the supervision of Dr. Farag Sallabi, in the College of Information Technology at UAEU. This work has not previously formed the basis for the award of any academic degree, diploma or a similar title at this or any other university. Any materials borrowed from other sources (whether published or unpublished) and relied upon or included in my thesis have been properly cited and acknowledged in accordance with appropriate academic conventions. I further declare that there is no potential conflict of interest with respect to the research, data collection, authorship, presentation and/or publication of this thesis.

Student's Signature: [illegible]

Date: 24/4/2022

Approval of the Master Thesis



This Master Thesis is approved by the following Examining Committee Members:

1) Advisor (Committee Chair): Farag M. Sallabi
   Title: Associate Professor
   Department of Computer and Network Engineering
   College of Information Technology

   Signature [illegible] Date April 18, 2022

2) Member: Mohammad Mehedy Masud
   Title: Associate Professor
   Department of Information Systems and Security
   College of Information Technology

   Signature MMasud Date April 18, 2022

3) Member (External Examiner): Hussam Al Hamadi
   Title: Research Scientist
   Department of Electrical Engineering and Computer Science
   Institution: Khalifa University, UAE

   Signature [illegible] Date April 18, 2022

This Master Thesis is accepted by:

Dean of the College of Information Technology: Professor Taieb Znati

Signature Taieb Znati

Date 13/06/2022

Dean of the College of Graduate Studies: Professor Ali Al-Marzouqi

Signature Ali Hassan

Date 13/06/2022

Abstract

Smart healthcare systems are one of the critical applications of the internet of things. They benefit many categories of the population and provide significant improvement to healthcare services. Smart healthcare systems are also susceptible to many threats and exploits because they run without supervision for long periods of time and communicate via open channels. Moreover, in many implementations, healthcare sensor nodes are implanted or miniaturized and are resource-constrained. The potential risks on patients/individuals' life from the threats necessitate that securing the connections in these systems is of utmost importance. This thesis provides a solution to secure end-to-end communications in such systems by proposing an authenticated key establishment protocol. The main objective of the protocol is to examine how physical unclonable functions could be utilized as a lightweight root of trust. The protocol's design is based on rigid security requirements and inspired by the vulnerability of physical unclonable function to machine learning modeling attacks as well as the use of a ratchet technique. The proposed protocol verification and analysis revealed that it is a suitable candidate for resource-constrained smart healthcare systems. The proposed protocol's design also has an impact on other important aspects such as anonymity of sensor nodes and gateway-lose scenario.

Keywords: Authenticated key establishment, Perfect forward secrecy, PUF, Root of trust, Smart healthcare systems, Resource-constrained.

Title and Abstract (in Arabic)

بروتوكول مصادقة وتأسيس مفتاح تشفير لأنظمة الرعاية الصحية الذكية المقيدة بالاعتماد على دالة مادية غير قابلة للاستنساخ

الملخص

تعد أنظمة الرعاية الصحية الذكية أحد أهم تطبيقات إنترنت الأشياء. هذه الأنظمة تفيد العديد من فئات المجتمع وتوفر تحسينات ملحوظة لخدمات الرعاية الصحية. أنظمة الرعاية الصحية الذكية تكون عرضة للعديد من التهديدات والاختراقات الأمنية لأنها تعمل دون إشراف لفترات طويلة من الوقت، كما أن عملية التواصل فيها عبر قنوات عامة غير آمنة. بالإضافة إلى ذلك، في العديد من تطبيقات هذه الأنظمة، عقد الاستشعار تكون مزروعة داخل الجسم أو ذا حجم مصغر، ومقيدة الموارد. تحتم علينا المخاطر المحتملة من هذه التهديدات على حياة المرضى أو الأفراد أن نعطي تأمين الاتصالات في هذه الأنظمة أهمية قصوى. هذه الأطروحة توفر حلاً لمثل هذه الأنظمة يقوم بتأمين قنوات الاتصال فيها، من الطرف الى الطرف، وذلك باقتراح تصميم ابداعي لبروتوكول مصادقة وتأسيس مفتاح تشفير. الهدف الرئيسي للبروتوكول هو دراسة كيفية استخدام الدالة المادية الغير قابلة للاستنساخ كأساس للثقة ذو أعباء قليلة على النظام. لإنجاز الأهداف المرجوة، صمم البروتوكول حسب متطلبات أمنية صارمة، كما تمت الاستفادة من ضعف الدالة المادية غير القابلة للنسخ أمام هجوم النمذجة باستخدام التعلم الآلي بالإضافة إلى استخدام آلية السقاطة. كشفت عمليتا التحقق والتحليل للبروتوكول المقترح أنه مرشح مناسب لأنظمة الرعاية الصحية الذكية ذات الموارد المحدودة. تصميم البروتوكول المقترح له أيضًا تأثير على جوانب مهمة أخرى مثل إخفاء هوية عقد الاستشعار وسيناريو فقدان الجهاز الوسيط.

مفاهيم البحث الرئيسية: المصادقة وتأسيس مفتاح تشفير، السرية التامة الأمامية، الدالة المادية الغير قابلة للاستنساخ، أساس الثقة، أنظمة الرعاية الصحية الذكية، محدودة الموارد.

Acknowledgements

I would like to express my sincere gratitude and appreciation to my advisor Dr. Farag Sallabi for his continuous support, assistance, and patience. His immense knowledge, motivation, and enthusiasm were key during my research thesis preparation, implementation and writing.

Special thanks to Prof. Mohamed Adel Serhani for his assistance, understanding, and continuous support during the period of my research thesis. Also, I would like to thank the thesis examination committee members, Dr Muhammed Mehedy Masud and Dr Hussam Al Hammadi of Khalifa University for their encouragement and insightful feedback.

In addition, I would like to thank all members of the College of IT, academics and staff for their guidance and help during my studies at UAEU.

Dedication

To my parents and family

Table of Contents

List of Tables

List of Figures

List of Abbreviations

AKE	Authenticated Key Establishment
AVISPA	Automated Validation of Internet Security Protocols and Applications
CA	Certification Authority
CRP	Challenge-Response Pair
DoSL	Denial of Sleep
DY	Dolev-Yao
EA	Evolution Strategies
ECC	Elliptic Curve Cryptography
ES	Evolution Strategies
FS	Forward Secrecy
GA	Genetic Algorithm
HD	Hamming Distance
HLPSL	High Level Protocol Specification Language
IC	Integrated Circuit
ID	Identity/Identification
ID-PKC	Identity-Based Public Key Cryptography
IoT	Internet of Things
LR	Logistic Regression
MAC	Message Authentication Code
MITM	Man in the Middle
ML-MA	Machine Learning Modeling attack
NVM	Non-Volatile Memory
PDA	Personal Digital Assistant
PFS	Perfect Forward Secrecy

PRNG	Pseudo Random Number Generator
PUF	Physical Unclonable Function
RoT	Root of Trust
SCA	Side Channel Attack
SHS	Smart Healthcare Systems
SVM	Support Vector Machine
TPKC	Traditional Public Key Cryptography
TPM	Trusted Platform Module
TTP	Trusted Third-Party
WBAN	Wireless Body Area Network
WMSN	Wireless Medical Sensor Network

Chapter 1: Introduction

1.1 Overview

Smart Healthcare Systems (SHS) are one of the critical applications of the Internet of Things (IoT). SHS enables medical doctors and other healthcare professionals to monitor and even control the health status of patients for an extended period, without the need for their presence at the clinic or hospital, and with minimal human intervention. SHS benefits many categories of the population, such as people with chronic diseases or the elderly. Smart healthcare systems provide significant improvement to healthcare services. They save human labor time and cut costs without sacrificing comfort.

A typical SHS consists of sensors, actuators, personal assistant devices connected to the backend system for patient's health monitoring and control. Sensors and actuators can be deployed on or implanted inside an individual's body. These nodes continuously collect physiological data such as heart rate and body temperature and send it to the healthcare provider. The data is transmitted through different devices and networks in the system before it reaches its final aggregation point. The collected data could be used by medical professionals to perform real-time patient monitoring and diagnosis or saved for later analysis by other parties to improve both patient experience and healthcare quality.

While smart healthcare systems save lives and improve quality of life, they also impose risks. Because SHS usually runs without supervision for long periods of time and communicating messages and data between different devices are sent via open channels, they are susceptible to many threats and exploits such as Denial of Sleep (DoSL) and node impersonation (Kumar & Lee, 2012; Sun et al., 2019). These threats have widened the attack surface and increased the security burden. The threats on SHS could have a serious impact if realized. In addition to conventional consequences of cyber-attacks such as data leakage, financial losses and violation of privacy, attacks on SHS also have the potential to cause direct physical harm as well as jeopardize human life. Moreover, SHS faces many challenges and constraints. In many implementations, healthcare nodes are implanted or miniaturized (Hassija et al., 2021). These nodes are designed with a limited power source, memory, processing, and bandwidth. They are also constrained in their operational environment.

The potential risks of SHS on patients/individuals' life necessitate that measures such as security to protect them are of utmost importance. Security measures should be applied at hardware, such as nodes and other devices in the system, data, and communication level. Typically, this is accomplished by satisfying security requirements such as authentication, confidentiality, integrity, and access control.

In general, the initial step of achieving security in any modern communication system is establishing a secure connection over open and insecure channels, such as the internet. The secure channel is expressed by securely setting the secret key/s and other parameters that are going to be used for that session and authenticate involved parties, in the same session run. This is accomplished with Authentication and Key Establishment (AKE) process. Authenticated key establishment is the basis of security and is either a prerequisite or included in any security solution. The techniques and building blocks of AKE protocols are mainly based on cryptography.

In fact, authenticated key establishment is not possible without an existing pre-set secure channel between communicating parties. Pre-set secure channels are established during the setup/initialization stage. They characterize genuine identities and associated secret parameters for participating entities. Pre-set secure channel could be represented in many ways, such as the availability of certificates, a shared key between participant entities, or with hardware means. These means also provide a Root of Trust (RoT) for the system.

The root of trust is any hardware or software component (or function) of a participating device/entity in a security system that establishes the foundation for the chain of trust (Verbauwhede & Schaumont, 2007). RoT performs and supports initial security-critical functions such as protecting long-term cryptographic keys and authenticating entities. The RoT element must be secure by design, protected and extremely hard to mutate. For critical systems such as SHS, the ideal implementation would be in hardware. Hardware RoT could be embedded within some system circuitry or as a standalone security module on a chip. RoT also is highly related to the robustness and efficiency of security protocols. Accordingly, it should be considered as early as the design and manufacturing process of nodes, not only during protocols design.

Traditional RoT solutions such as Trusted Platform Module (TPM) standard and public key certificates are ideal for securing resource-rich nodes or the network and application layer devices of the SHS architecture. However, they are not suitable for resource-constrained nodes such as implanted sensors. This is because they usually rely on complex and computation-intensive cryptographic algorithms. Furthermore, hardware RoT needs to store the identity and keys on some sort of protected non-volatile memory (NVM). This implementation requires more power sources, which are scarce in constrained nodes. All these factors, in turn, add more complexity, area, and cost to the constrained nodes, which ultimately will hinder the realization of ROT-based security solutions for SHS.

There is another emerging technology that appears to agree with the requirements and terms of RoT, and yet could be applicable to constrained nodes in SHS. This technology is called Physical Unclonable Function (PUF). PUF is a hardware element that could be integrated with any electronic device. PUF is a function that is generated from the uniqueness and irreproducibility of the microstructure or circuitry of devices during the manufacturing stage, thus, acting as the device biometrics. The main application of PUF is the intrinsic identification of Integrated Circuits (IC). Integrating PUF technology in constrained nodes can provide hardware RoT and hence could provide a robust basis of security for SHS with constrained nodes. The unique input-to-output mapping (challenge-response mechanism) of a PUF instance in a node can be utilized to authenticate the node as well as assist in generating secret keys and other parameters.

1.2 Statement of the Problem

End-to-end connections (sensor/actuator-gateway-backend server) in smart healthcare systems pass through different network segments of different service providers. Therefore, providing secure and reliable connections become imperative and a challenge. Authenticated key establishment is the first step towards securing these active channels. The last connection between the gateway and the sensors is the weakest link in cybersecurity as it involves the most resource-constrained devices (Alladi et al., 2021;

Kompara et al., 2019). Several ongoing research is investigating the security challenges of such connection but may not provide a suitable solution for resource-constrained SHS implementation (Banerjee et al., 2019; Fotouhi et al., 2020; Gope et al., 2019; Kompara et al., 2019).

More specifically, protocols that are based on shared keys as a RoT element don't imply good security properties and lack Perfect Forward Secrecy (PFS) property. Moreover, protocols that are based on public-key cryptography are not lightweight. Additionally, previous work that endeavors to use PUF was mainly focused on authentication only and lack network infrastructure support. Their performance impact also is not evident. PUF implementation also still an emerging field of research. It faces many challenges and is susceptible to new threats, such as machine learning modeling attacks (ML-MA) (Rührmair et al., 2010; Ruhrmair & Solter, 2014).

This work is based on the hypothesis that an authenticated key establishment protocol design that is based on the lowest abstraction of security, i.e., node identity, and coupled with the use of a lightweight hardware RoT, i.e., PUF, will enhance security and reduce the computation and storage overhead. Since AKE is the cornerstone of security, such a design is also likely to enhance other important aspects of SHS. Evaluating this hypothesis is aligned with the answer to the following research questions:

- How could PUF be leveraged in an AKE protocol design for constrained SHS?
- What enhancements would such a design bring to security and efficiency in these systems?

1.3 Research Objectives and Contribution

The objective of this work is to propose an improved authenticated key establishment protocol that is more suitable and efficient for Smart Healthcare Systems (SHS) with resource-constrained nodes. The proposed protocol aims to solve the mentioned issues with the previous protocols by employing PUF in the nodes as an intrinsic identity and the RoT. Specifically, the proposed protocol: (1) provides better security leveraging the vulnerability of PUF to ML-MA and ratchet technique (Goutsos, 2020); (2) does not require the storage of any secrets on nodes (secret free); (3) and utilizes lightweight cryptographic primitives. Our contribution to achieving these goals is as follows:

- Design an enhanced authenticated key establishment protocol for SHS with solid requirements that satisfy the given objectives.
- Implement the proposed protocol using the AVISPA tool, which mainly simulates and validates the proposed protocol. The tool also verifies the security of the protocol against different attack models.
- Analyze the protocol and prove its correctness (against the requirements).
- Evaluate and compare the performance of the proposed protocol against previous protocols, considering functional features as well as computation costs.

1.4 Relevant Literature

In order to understand the proposed protocol better, a short review of background topics and the techniques are provided in this chapter. Furthermore, a comprehensive analysis of previous related work from the literature is included, which highlights the issues with previous protocols.

1.4.1 Identification and Authentication

Security starts as early as the identification of communicating entities. Entity IDs also play an essential role in securing devices and preventing many attacks. Although sometimes identification and entity authentication are treated as synonyms, they are considered separate but fundamentally related topics. Identification is a significantly weaker concept than authentication. It is merely stating or claiming an identity, without necessarily presenting any credible proof. Identification is not expressly a security mechanism since it doesn’t accomplish any meaningful security objectives. However, it still has very useful features (Maes, 2013):

1. Identification is considered a necessary prerequisite for entity authentication and hence an integral part of entity authentication and other security techniques.

2. In certain situations, after authentication for the first time, identification only is sufficient to fulfill re-authentication since the authentication conditions are met.

3. In some applications without strict security goals, such as non-critical closed systems, identification only can be sufficient.

Identities can be either assigned or inherent, depending on their identifying features. Assigned features of an entity could be a barcode that is printed on its surface or a unique serial number that is either printed or retrieved in the bootstrapping process. Inherent features are the specific characteristics of an entity that arises in its creation process. The best analogy to describe the difference between both identification techniques is with human beings, where we make a distinction between a person's fingerprints, which are inherent, and his/her name, which is assigned after birth. Table 1 highlights other characteristics and differences between the two types of identity (Maes, 2013).

Table 1: Characteristics of assigned and inherent identities

Criteria	Assigned identities	Inherent identities
Uniqueness	A unique identity needs to be generated before it is assigned to an entity. To ensure high probability uniqueness, a state needs to be kept by the provisioning party (i.e., counter).	Uniqueness results naturally from the creation process of inner circuits of the entities.
Memory requirements	Permanent physical changes (or at least non-volatile) need to be made for each entity during the assignment process. These changes should be compatible with the entity's construction and induce extra cost.	Additional physical memory capabilities are not required.
Assignment process	The process of writing an identity is intrusive and adds many after-manufacturing processes.	The process of reading an identity (enrollment) is less intrusive and much faster and more reliable.
Cost	Relatively easy and cheap to produce, even after entity manufacturing.	Harder to produce, costly, and sometimes impractical after entity manufacturing.

While inherent identities possess many practical advantages, however, there is no direct control over the values denoted by the identifiers. This is considered an issue if a meaning to an identifier value is needed to be assigned (e.g., a serial number that is based on when an entity is created). Another undesirable aspect of most inherent identities is their fuzzy nature. In general, the fuzzy random behavior of an inherent identity means that its responses are not entirely and uniformly distributed. The same responses are also not "perfectly" reproducible, when measured many times (Maes, 2013).

Authentication is the process of verifying and proving an identity. In information security, authentication can be related to entities or to data. In the latter, it is called message authentication. Besides validating identities, entity authentication also assures that the participating entity is actively present during the authentication process (Menezes et al., 1996). The types of authentication procedures are:

- One-way / Unilateral authentication: two parties wishing to communicate with each other, one party authenticates itself to the other.
- Two-way / Mutual authentication: both entities authenticate each other.
- Via Trusted Third-Party (TTP) authentication systems.

Historically, weak entity authentication was considered as the only security measure, where assigned IDs with passwords were enough to secure communications between computers. In modern security, this approach does not satisfy most security requirements. Strong authentication is needed to provide freshness and assurance of relevant participants. Authentication is the basis of security. Authentication protocols could be classified into two categories, non-cryptographic (challenge-response), including conventional password, biometrics, and CAPCHA, and cryptographic. Research in this field has been – and still – going for decades. In recent protocols, a hybrid approach of both cryptographic and non-cryptographic types of authentication is included in the design (Stallings & Brown, 2015).

In this work, the proposed protocol is based on inherent identity (provided by employing PUF) rather than just the assigned identity of nodes. Even though entity authentication is particularly considered in this thesis, and is implied whenever we talk about authentication, message authentication is also needed to verify the integrity of exchanged messages during protocol run, this prevents many attacks such as message tampering attacks.

1.4.2 Key Establishment and Management

Although authentication is the initial step of accomplishing security in any communication protocol, however, authenticating participant entities alone is not adequate, setting up other secret parameters is needed to secure the connection between them. The notions “Establishing a secure channel” or “Starting up a secure session” denotes authentication and key establishment (AKE). Authentication and key establishment are the fundamental building blocks for securing network communications over open and insecure channels, such as WLAN and the internet. AKE is performed at the beginning of each session. The end goal of AKE is to securely establish the secret key/s that are going to be used for that session - where involved parties have been authenticated in the same session run.

Authenticated key establishment is the basis for any security protocol and is either a prerequisite or included in any security solution. The significance of AKE is stated by the fact that security algorithms and protocols cannot perform their function unless a session key has been established securely and all parties know with whom they share this key. The main requirements of any AKE protocol are (Boyd et al., 2020):

- Each party should be able to determine the identity of other participants in the protocol.
- All participants should be able to construct the same session key.
- Any other party cannot predict the session key.

Practically, establishing authenticated session keys (i.e., secure channel) is not possible without the availability of pre-set secure channels. All protocols adhere to one of these modes to establish a new session key (Boyd et al., 2020):

- All parties already share a secret cryptographic key

- A trusted online server must be accessible: each party must share a key with the trusted online server. To transfer information between the parties, it might be necessary to pass it through a chain of other trusted online servers.
- A trusted offline server: All parties have private-public key pair, where public keys are certified from a certification authority (CA).
- Secure physical means for key establishment. PUF is considered one of the solutions under this category.

Modern security solutions, including authenticated key establishment are based on cryptography. Cryptography, in turn, is based on keys and secret values. According to Kerckhoff’s principle, a cryptosystem should be secure even if everything about the system is public knowledge, except the keys. If the keys used in a system are not secure, the whole system is not secure, even if secure and robust techniques and algorithms are used.

Key management is a more general notion than the key establishment. It starts as early as the registration and initialization procedure of entities (setup). Once initial and secret keys have been established, they should be managed to provide secure communication. Key management consists of many services such as establishment, update, storage, backup, recovery, and revocation of cryptographic keys. Although it is one of the most important aspects of cryptographic systems, key management is most often neglected (Masdari et al., 2017). This work mainly focuses on the initial step of key management, i.e., setup phase and the key establishment procedure during the protocol run.

1.4.3 Physical Unclonable Function (PUF)

PUF is defined as an embodied function in the physical structure of a device that maps input challenges to output responses. PUF is non-deterministic and varies for each instance (Gassend et al., 2002), as opposed to a mathematical function that produces a fixed output for the same input and is deterministic in nature. The idea of PUF was first proposed by Pappu et al. (2002). Their concept was based on an optical principle of operation. Following this work, the concept of silicon PUF was introduced by Gassend et al. (2002) where the authors argued that a complex integrated circuit could be regarded as silicon PUF. The silicon PUF exploits the fundamental and random variations in CMOS circuits as a result of the manufacturing process. In their study, a technique to identify and authenticate individual ICs were also described.

The complex statistical variations of the circuits embedded in devices can be used to map a set of challenges to an equivalent set of responses. Each instance of a silicon PUF and its mapping should be different from other instances. The set of challenge-response pairs (CRPs) for a silicon PUF can be defined as (Ci, Ri), where i = 1…n. In general, challenges could be described as a k-bit inputs. The challenges actually control the behavior of a PUF, where corresponding responses produced based on challenges applied. As shown in Figure 1, when a challenge C is applied to two distinct PUFs (X and Y), the respective responses R1 and R2 are generated, where response R1 ≠ response R2.

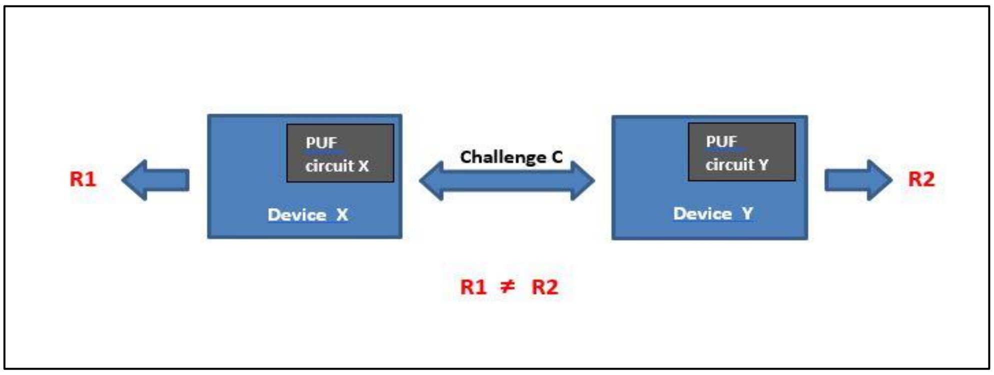


Figure 1: Challenge-response mapping

The uniqueness of responses act like electronic biometrics, which distinctively identifies each PUF instance. Silicon PUF eliminates the need for storing secret keys in the memory, unlike the conventional method of IC security (TPM). This way, secret keys are only generated when required, by applying a challenge. This implementation provides intrinsic, random, and secure features for devices embedded with PUF which make it a promising technology that could replace current security solutions.

1.4.3.1 Variants of PUF

Silicon PUFs can be classified into three categories, according to their challenge-response properties, each with their own ideal applications (Guajardo et al., 2007; Lim et al., 2005):

1. Strong PUFs: Strong PUFs are PUFs with a vast number of CRPs. The number of CRPs increases exponentially as the number of bit challenges rises. The PUF interface is directly accessible without a protection mechanism and the challenge-response pairs could be collected using partial measurement.
2. Weak PUFs: PUFs with a small and limited number of CRPs. Weak PUFs also have fixed challenges.
3. Controlled PUFs: An enhanced version of strong PUFs. The PUF interface is not directly accessible and is protected by a logic processing unit. Different techniques of protection such as hash function, obfuscation and permutation is used (Gassend et al., 2008). Challenges and responses of the strong PUF are processed by the protection function before being handled to or output from PUF. A model-building attack is one of the probable attacks on strong PUFs, where the adversary builds a numerical model of the PUF by measuring an adequate number of CRPs. The introduction of extra pre and post-processing steps for the controlled PUFs increases the level of difficulty to measure and collect the CRPs. Hence, it reduces its vulnerability to model-building attacks.

Based on the above, the main distinction between strong and weak PUFs is the number of CRPs. Because strong PUFs support a larger number of CRPs, they could be utilized to provide authentication, particularly, to protocols that use challenge-response mechanism. On the other hand, the limited number of CRPs of weak PUFs means that these CRPs should be kept secret. This make weak PUFs more suited for secret key storage and generation for cryptographic operation such as symmetric encryption and Message Authentication Code (MAC) (Lim et al., 2005).

1.4.3.2 Quality Metrics of PUF

Uniqueness: It is the ability of uniquely distinguishing one PUF instance from other instances. The Hamming Distance (HD) is used to measure this uniqueness. The Inter_HD is the HD between any two binary strings of the same length is the number of locations at which their corresponding bits are diverse. If the same challenge C is applied to two chips, i and j (where i≠ j), and n-bit responses are generated, $R_i(n)$ and $R_j(n)$, respectively, the Inter_HD between k chips is defined as in Equation 1 (Maiti et al., 2013).

$$\text{Inter\_HD} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{\text{HD}\left(R_i(n), R_j(n)\right)}{n} \times 100\% \qquad \text{(Equation 1)}$$

To uniquely identify a PUF from other instances of PUFs of a similar type with high probability, it is a desirable feature to have The PUF responses randomly distributed, where the Inter_HD is centered close to 50%.

Reliability: This is determined by the consistency of the PUF responses, given the same challenge at various ambient temperatures and supply voltage oscillations. Intra HD is used to evaluate the reliability of a PUF instance. For a single chip, represented as i, it has a challenge C and an n-bit reference response $R_i(n)$ at default room temperature and a reference supply voltage. If the same challenge C is applied to the chip i at the different conditions to generate the n-bit response, $R'_{i,j}(n)$, the average Intra_HD form samples is defined as in Equation 2 (Maiti et al., 2013).

$$\text{Intra\_HD} = \frac{1}{m} \sum_{j=1}^{m} \frac{\text{HD}\left(R_i(n), R'_{i,j}(n)\right)}{n} \times 100\% \qquad \text{(Equation 2)}$$

From the Intra_HD value produced, the reliability of a PUF instance can be defined as in Equation 3.

$$\text{Reliability} = 100\% - \text{Intra\_HD} \qquad \text{(Equation 3)}$$

From the Equation 3, a small Intra_HD is desired to achieve high reliability.

Uniformity: It is the ratio of 0's and 1's in the response bits of a PUF. The uniformity of a PUF characterizes the randomness of its response. Ideally, the value of PUF uniformity should be around 50%. The Hamming Weight (HW) is used to measure the uniformity in PUF. It evaluates the number of `1' bits in the binary sequence as described in Equation 4 (Maiti et al., 2013), where $r_{i,j}$ is the jth binary bit of an n-bit response from chip i.

$$(\text{Uniformity})_i = \frac{1}{n}\sum_{j=1}^{n} r_{i,j} \times 100\% \qquad \text{(Equation 4)}$$

Uniqueness and uniformity quality metrics are independent parameters. For k chips of a similar type of PUF with an n-bit response from each chip, the average uniformity can be close to the ideal value of 50%, but that does not necessarily mean 50% uniqueness. For example, a number of chips k of the worst PUF could generate k similar n-bit responses, which have a well-balanced distribution of 0's and 1's in their n-bit responses. Furthermore, similar type of PUF k chips could achieve uniqueness around the ideal value of 50%, whereas the average uniformity is not necessarily at 50%. For example, it is possible that a number of the k chips of the PUF could generate all 1's or all 0's in their corresponding responses.

1.4.3.3 Attacks on PUF

PUFs are exposed to many types of attacks. The main category of attacks are invasive and non-invasive, as shown in Figure 2. Invasive attacks involve physical modification of the PUF to evaluate it and gain deeper knowledge on its implementation. Invasive attacks typically affect weak PUFs due to its limited challenges. On the other hand, non-invasive attacks invisibly collect challenge-response pairs without any physical modification to the PUFs. Non-invasive attacks usually threaten strong PUFs because they have a huge number of challenges. In order for this kind of attack to succeed, data (CRPs) must be gathered and then evaluated. The next section describes non-invasive attacks that are closely related to our work.

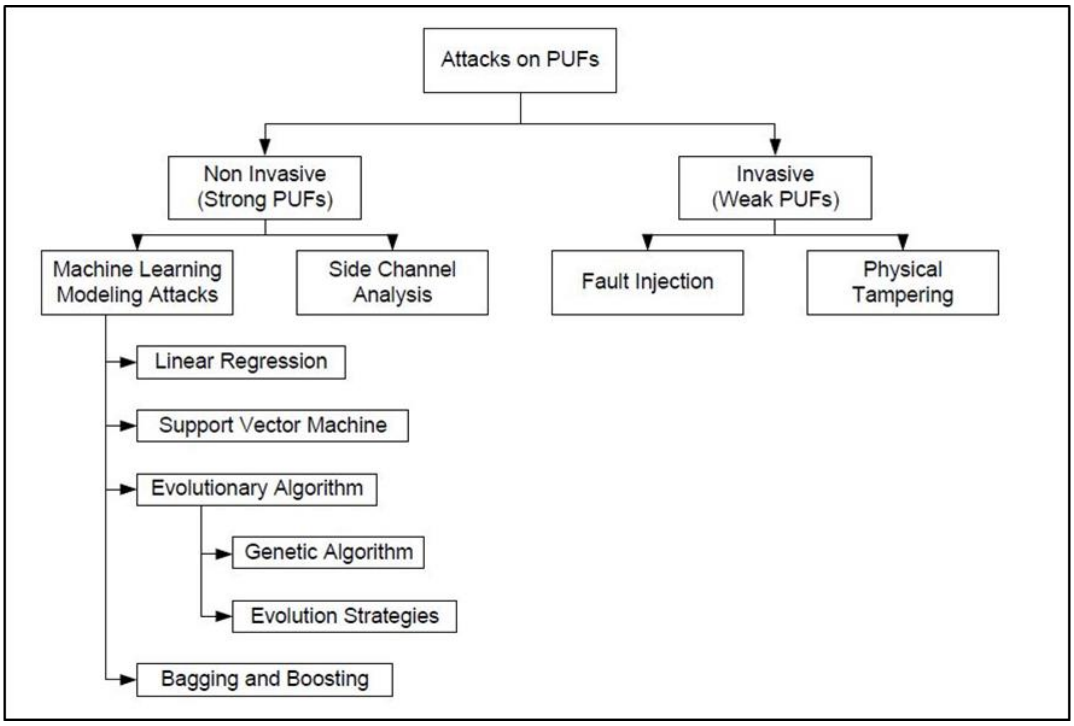


Figure 2: PUF attacks classification

The most significant non-invasive attack that threatens strong PUFs is known as machine learning modeling attack (ML-MA). This attack was first introduced by Rührmair et al. (2010). In this attack, an adversary collects vast number of CRPs from the strong PUF. Next, the adversary infers the behavior of the PUF (challenge-response relationship) on the unknown CRPs by combining numerical method of analysis along with the internal characteristics of that specific PUF. The impact of this attack is surprisingly immense since most strong PUFs, including enhanced versions of arbiter PUF are vulnerable to this type of attack. Commonly, there are three effective machine learning algorithms, namely logistic regression (LR), Support Vector Machine (SVM), and evolutionary algorithms (EA), such as evolution strategies (ES) and genetic algorithm (GA), that are used to execute modeling attacks.

LR is different from linear regression, where it outputs a probability between 0 and 1 instead of producing ±1 output. The SVM algorithm is a tool that utilizes the optimal margin between vectors to set the best hyper plane. This method requires computing the distance of input vectors from the hyperplane. As for the GA algorithm, it handles integer and binary string solutions by simulating biological evolution using models such as reproduction, mutation, and selection. ES algorithm is used to generate population heuristically by adapting the previously produced population to certain environmental conditions (Ruhrmair & Solter, 2014). The data set processed is randomized to avoid them from separated linearly. The final resulting model must be parameterized to ensure the data set (CRPs) is reliable.

Another attack that is concerned with strong PUF is known as side-channel analysis (SCA). The attacker performs SCA by observing the non-functional metrics of the PUF element, such as the power consumption or timing parameters to extract information for developing ML-MA. In general, SCA attack on PUF-based systems is challenging as it involves attacking the main PUF component/circuitry embedded in the device . Since SCA alone is hard to perform on PUF element, some researchers proposed combining ML-MA with SCA to improve the attack results (Karakoyunlu & Sunar, 2010).

1.4.4 Related Literature

In recent years, many protocols that provide authenticated and key exchange solutions to SHS in different settings have been proposed. Amin et al. (2018) proposed a protocol for patient monitoring system utilizing wireless medical sensor networks (WMSN), the protocol provides mutual authentication as well as user anonymity. It was proved that their protocol is resilient against relevant and known attacks, lightweight and is suitable for healthcare applications. Additionally, the session key was constructed by all three entities. One-way function and symmetric encryption were employed for efficiency.

Despite their claim of robust security, Amin et al.'s scheme (2018) still has security issues. The protocol was analyzed and found susceptible to off-line guessing and the de-synchronization attacks by authors of (Wu et al., 2018). It was also proven prone to stolen mobile attacks, secret key exposure and de-synchronization attacks by (Jiang et al., 2017). Both authors of (Jiang et al., 2017; Wu et al., 2018) proposed an improved protocol that withstands these attacks and is more efficient in terms of computation cost. Interestingly, quadratic residues were used by (Jiang et al., 2017) to overcome the mentioned weaknesses in (Amin et al., 2018).

Alternatively, the work of (Kompara et al., 2019) proposed a new authentication and key agreement protocol for WBAN that solved the issues of anonymity and traceability of nodes as well as sensor node capture attack. Their solution came with a cost, to attain a temporary node id and not reveal the real node id, extra parameters needed to be saved on the node. Furthermore, the intermediary node (IN), such as smartphone was excluded in their scheme, this, in turn, might add more overhead on the nodes.

Recently, Fotouhi et al. (2020) designed a lightweight two-factor authentication protocol that is resilient to key compromise, impersonation, and denial of sleep attacks. Additionally, they claim that the protocol achieves perfect forward security. Their scheme also utilized a new hash chain technique and introduced revocation and invalidation of users to prevent any unauthorized access in case of their password or other parameters involved in the authentication process are compromised.

The fact that all previous protocols are based on shared keys stored on the device as a pre-set secure channel make them inadequate for addressing many security issues. For example, perfect forward secrecy is hard to achieve with such schemes. In addition, security features such as anonymity were often come with the cost of extra temporary parameters saved on nodes. Moreover, many added security features such as two-factor user authentication were implemented on the server-side of the network, and it did not have any effect on the security on end nodes or intermediary devices, where it is needed the most.

In the IEEE 802.15.6 communication standard (WBAN) security is initiated with the security association procedure. During this process, the node and hub are identified to each other, a pre-shared/new key is activated, and a pairwise temporal key is generated. To achieve this goal, the nodes and hub negotiate which one of the four key agreement protocols (defined in the standard) to be applied. All four protocols are based on public-key cryptography. The security of these four protocols were challenged by (Toorani, 2015). The analysis showed that all the protocols do not provide forward secrecy and vulnerable to KCI attack. The researchers also hinted that the security mechanism in the standard does not show any indication for privacy.

Alternatively, other scholars such as in (He et al., 2017; Shen et al., 2018) opt to use a different approach. The authors used elliptic curve cryptography (ECC) and identity-based public-key cryptography (ID-PKC), respectively to avoid the need for certificates and the modular exponentiation in traditional public-key cryptography (TPKC) while providing robust security. Although ECC and ID-PKC are considered better alternatives - in terms of computation - than TPKC and might suit intermediary devices in the network such as gateway, it would not be a suitable solution for constrained nodes.

The proposed protocol aims to design an efficient AKE protocol using PUF. PUF implementation is rather challenging and is still considered an emerging field of research. There have been many attempts to use PUF for security protocols. Previous work such as (Goutsos, 2020; Yanambaka et al., 2019; Yilmaz et al., 2018) mainly consider authentication. In these protocols, different designs were approached. For example, Yilmaz et al. (2018) protocol aims for general IoT device without any consideration for resource constraints. The protocol also assumes that a PUF model for the device is stored in the verifier's database along with its ID and MAC address. This scheme also consider a lightweight secret key encryption to obfuscate the challenge-response relationship.

In (Yanambaka et al., 2019), the design assumes PUF is embedded in every device participating in the protocol, including the server. During the node enrollment phase, the response of initial challenge generated in the server is fed to the PUF instance in the node as the challenge. The response is then fed back to the server which uses it to get a new response from its PUF instance. The server stores this relationship as the initial CRP and use it for authentication during protocol run. The CRP is renewed after each session. The work of Goutsos (2020) presented a CRP ratcheting protocol that is based on PUF. The protocol also provides only authentication for node to node communication scenario. The strength of the protocol lies in renewing authentication secrets by blending parameters from both participating nodes to create a secure link between them. The protocol implementation also supports de-synchronization recovery.

On the other hand, some recent end-to-end protocols utilize PUF in AKE (Aman et al., 2017; Banerjee et al., 2019; Gope et al., 2019). Gope et al. (2019) suggested real-time information exchange security protocol for industrial WSN. Both sensor and end user device is assumed to be embedded with PUF. In this scheme, most protocol computation are carried out in the gateway which is considered as a trusted third party. This protocol relied on the storage of a database of CRPs for each PUF instance in the gateway.

Alternatively, Alladi et al. (2021) opt to store the initial CRP on the edge server side. The node’s CRP is renewed in each session run and is transmitted securely to the server via a secure channel. This channel is established during setup stage utilizing another PUF instance in the gateway.

Chapter 2: Methodology and Design

In this section, the approach used to realize the solution to our research problem is demonstrated. First, the system architecture with the main entities of the protocol is depicted. The adversary capabilities that pose threats to the proposed protocol are also stated. Second, the requirements that the protocol should satisfy to achieve our objectives are defined. Lastly, the practical steps taken to reach to final protocol design are described. This includes exploration of key types and hierarchy, followed by an analysis of how and where PUF could be utilized.

2.1 Network Architecture

The smart healthcare system network architecture is shown in Figure 3. The architecture consists of three main levels, namely, sensor level, gateway level, and the medical server level. The sensor level constitutes a Wireless Body Area Network (WBAN), where sensors are implanted inside the patient's body. The sensor nodes are resource-constrained in terms of processing power, storage, data rate, energy, and communications range. In the second level, a gateway connects the nodes to the server. Both gateway and nodes are within close proximity, usually the body of the patient or nearby IoT nodes. The gateway could be a smartphone or a proprietary Personal Digital Assistant (PDA) provided by the healthcare provider. The gateway is assumed to have fair resources to pre-process the packets received from the sensor nodes.

The third level is the backend server that resides at the edge computing of the healthcare provider. In addition to receiving the data from the gateway and process them, the server performs the protocol initialization and setting up parameters needed for the protocol run (setup phase). It is assumed that the server is placed on a secure site, trusted and with unlimited resources.

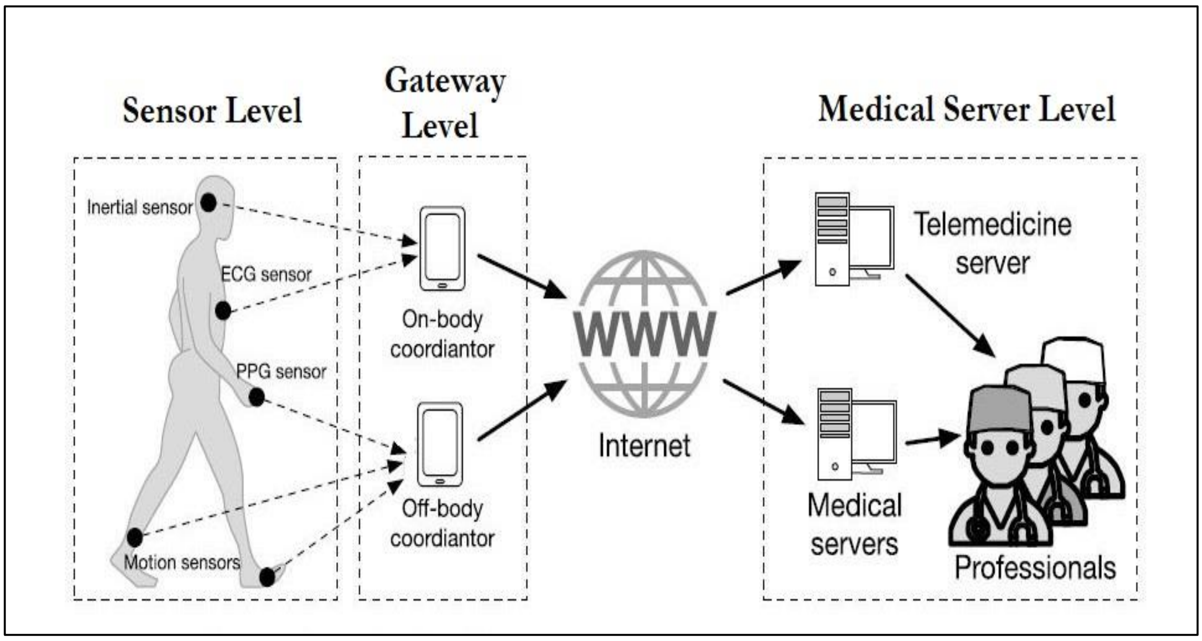


Figure 3: Network architecture

2.2 Adversary/Attack Model

In the network architecture, the communication channels between nodes and gateway, and between server and gateway are considered insecure and untrustworthy. Therefore, the proposed protocol messages are susceptible to many attacks where an adversary can have control over all these messages. The widely used Dolev-Yao model (Dolev & Yao, 1983) is employed to assess the attacker capabilities and their impact on the security of the proposed protocol. The model assumes that the adversary can read, record, alter, forge, delay, redirect, and delete messages. The adversary also can replay past or current messages and inject new messages between communicating parties. Another popular model for evaluating the security of key-exchange protocols known as the CK-adversary model (Canetti & Krawczyk, 2001) is also considered. Under this model, an adversary can manipulate messages as in the DY model, as well as the ability to exploit extra information such as the session keys, private keys, and session state.

2.3 Requirements of the Proposed Protocol

The literature on secure protocol design and the analysis of the previous work (Section 1.4.4) reveal that addressing the mentioned issues needs rigid requirements during design stage. Thus, the proposed protocol needs to satisfy the following requirements to accomplish the wanted objectives:

2.3.1 Essential Requirements

1. Strong entity authentication: The authenticating party should have a fresh assurance of who is the other participating (authenticated) party.
2. Mutual authentication: Both communicating entities should prove their identities to each other. This is a crucial security property and a countermeasure to prevent many attacks.
3. Session key establishment/agreement
    a. Key freshness: The session key should be newly generated and all involved entities should be able to verify its freshness.
    b. Key authentication: The session key should be known only to intended parties.
    c. Key integrity: Inputs to session key computation function should be transferred to other participants in a verifiable manner to assure that they have not been modified.
    d. Key confirmation: Each participant of the protocol should have an assurance that the generated session key is a good key and the other participants possess the same session key.
4. Resistant to known attacks: The following list presents core attacks that are a possible threat to AKE protocols. An adversary could also leverage them as part of or prerequisite to other attacks. The proposed protocol should have countermeasure mechanisms to prevent the success of such attacks:
    a. Node impersonation attack.
    b. Message tampering attack.
    c. Replay attack.
    d. De-synchronization attack.
    e. Privileged insider attack.
    f. Man in the middle attack (MITM).

2.3.2 Enhanced Security Requirements

1. Forward secrecy (FS): FS signifies that future secrecy mistakes should not threat past secrets and is a desirable security property in AKE protocols. The security of session keys that have been previously established should not be affected even if a participant entity's private/secret key has been compromised. On the other hand, if the established session keys remained secure even with compromising all private long-term keys of participant entities in the protocol, this denotes perfect forward secrecy (PFS) (Toorani, 2015).
2. Machine learning modeling attack resistant (PUF related).
3. Lightweight (low overhead).

2.4 Key Hierarchy

Theoretically, satisfying the requirements should lead to a secure protocol design. However, key management is practically the main challenge to achieve security goals. This is because AKE protocols are based on cryptography, which is based on keys and secrets. Failing this task usually results in vulnerabilities in the protocol implementation, which comes in a form of exploiting - or even the likelihood of - some secret parameters that the protocol's security depends on. For this reason, key management should be considered carefully at each stage in the design. The first practical step in the proposed protocol design is to conceptualize distinct types of keys used in the protocol and how they relate to each other. Table 2 summarizes key types and their hierarchy.

Table 2 helps in better understanding how various levels of keys and secrets are related to AKE protocols. It also highlights that the key hierarchy in fact represents secure channels and the interaction of an AKE protocol. The task of exhibiting a one-fits-all generic high-level protocol is challenging. This is because the building blocks and steps differ considerably from protocol implementation to another, for instance, whether symmetric or asymmetric encryption is employed.

Table 2: Key types and hierarchy

Level	Description	Example	Uses	Occurrence
0	Trusted key	CA key RoT	Chain of trust Authority	-
	Node enrollment -- Gateway registration -- Level 1 secret generation			Setup (once)
1	Secured identity and/or secret key of entities	Pre-set secure channel (Private public key pair or shared secret key)	Authentication MAC (integrity) Construct lower-level keys	-
2	Other secrets	Random values Ephemeral keys Pseudo/Temporary ID Other computed secrets	Freshness Extra secrecy Anonymity/Privacy Verification/Integrity	Setup (each session run)
3	Session key	Agreed-on session key Distributed session key	Encryption of transmitted data	Goal of AKE protocol

2.5 How and Where PUF Could be Utilized?

Essentially, a PUF instance uniquely identifies the device it is embedded in, which intuitively makes it a good candidate for providing authentication for that device. However, the classical implementation of PUF for authentication does not provide adequate security properties. First, the authenticator (gateway or server) must store a vast database of challenge-response pairs (CRPs) for each authenticating entity (node) in the system. Second, this approach is vulnerable to modeling attacks using machine learning techniques. The PUF's vulnerability is due to the linearity of its output (follows Gaussian distribution model). Lastly, this approach does not suit three participants' system model.

Alternatively, PUF could be leveraged differently in designing a security protocol such as generating random values in the node, a secret function between the gateway and the node, or as an ephemeral key (to provide PFS). To use PUF as a pseudo-random number generator (PRNG) in the sensor node, the challenge C could act as a seed in the generation process and the response RS as the random value generated. Most PUF types such as arbiter provide outstanding randomness and their output is uniquely and uniformly distributed.

In protocols that use a shared key to establish the pre-set secure channel between gateway and node, the constrained node such as implantable probably stores the key for an extended period (years). If this key is exposed in the gateway the security of the system will be compromised because it is hard to change the shared key in the node. PUF could substitute the shared key between the gateway and the node. A pair of challenge-response is used for this task. The challenge C could be used to identify a node while its response RS acts as the shared key for node, without the need for any key storage. The challenge-response pair (CRP) could be easily designed to be updated in each protocol run. However, this method – again – requires the storage of a vast database for each node in the gateway and is not secure. The gateway could be lost, stolen, or compromised, which results in losing the secure channel with the node and jeopardizing the entire system.

The key to enhance PUF implementation in AKE protocols is hiding the relationship between challenges Cs and responses RSs while abolishing the need for storage of a database of CRPs. Generally, this implies obfuscating all responses and storing the minimal CRPs on the gateway, yet, satisfying security requirements of the protocol. Practically, this is a challenging task, especially for the proposed network architecture. Considering any solution would be highly correlated to addressing the following questions:

- How is the session key is going to be constructed?
- How are the secrets constructing the session key going to be transported and authenticated from one participant to another?

The session key could be constructed as a function of contributions (inputs) from all protocol participants (key agreement). These inputs could be random values freshly generated by each participant during session run. Other parameters such as identities could also be considered in the function for added obfuscation. More importantly, no one participant should be able to predetermine the resulting session key. To transport the session key inputs securely from one participant to another, the pre-set secure channel (shared secret key) is usually used to encrypt the messages containing these key inputs.

The encrypted messages also should be authenticated and verified at the receiving end to assure their integrity (Section 2.3.1). This could be achieved by setting-up verification parameters with the secret private keys of the participants.

The vulnerability of PUF to ML-MA (Section 1.4.3.3) could be leveraged to build a model for each node so that responses RSs could be extracted in other participants than the node. According to (Ruhrmair & Solter, 2014), a ML-MA on 128-bit arbiter PUF yielded a 99.9% prediction accuracy rate for only 39200 CRPs. The training time was as little as 2.10 seconds. Considering a model for PUF would consequently raise the following questions:

- Where the PUF model would be stored? In the gateway or server?
- What parameters need to be stored?

The PUF model would be better suited in the server as it is assumed to be physically more secure than the gateway. In this case, the gateway should have knowledge of the CRPs used as well as their transport and update mechanism in each session run. In fact, the gateway participation plays a crucial part in the security of the protocol as the intermediary device, yet, is the most vulnerable. This suggests that minimum secrets should be stored in the gateway while maintaining a secure relationship between the server and the node. The strength of the proposed approach lies in the integration between the PUF instance in the node with the PUF model in the server with the minimum intervention of the gateway.

To solve this issue a ratchet technique is introduced in the design of the proposed protocol. The technique is built on ”ratcheting” or refreshing the protocol secrets where the refreshment process is one-way only (Goutsos, 2020). In this technique, local state that has been established in previous protocol session is renewed with each fresh session run. This results in the inability to derive previous secrets from future ones and creates a trust chain. This technique is used in previous work, such as in (Goutsos & Bystrov, 2019; Poettering & Rösler, 2018). The proposed protocol is based on combining ratchet with PUF to provide the desired requirements.

Chapter 3: Protocol Implementation and Analysis

The proposed protocol is conducted in two phases, setup phase and authenticated key establishment phase. In the setup phase, a secure channel – represented by shared secrets and verification parameters – is established between the participants of the protocol and is executed once. This phase is further divided into two stages, the node enrollment and gateway registration. The authenticated key establishment phase is the main protocol and is executed in each session run. The setup phase of the protocol is performed over a secure channel and in a trusted environment, whereas the protocol run phase assumes untrusted and insecure channel. The notations used in the proposed protocol are illustrated in Table 3.

Table 3: Protocol notation

Notation	Description
V, VID	The healthcare provider server and its identity
Gj, GIDj	The jth gateway and its identity
Ni, NIDi	The ith node and its identity
Vsec	Server secret key
Hvsec	Masked server secret key
GLKj	Gateway long term secret key
SKvg	Registration token of gateway
RegN	Registration Verifier (server-node)
Ai	Masked SKvg (server side)
RegG	Registration Verifier (gateway)
Bi	Masked SKvg (Gateway side)
Di	Masked RegG
Rv, Rg, Rn	Random value generated by server, gateway and node, respectively
Kses	Session key
Ci, Rsi	Challenge and response
Cn, RSn	New challenge and response
SNi, SNn	Dynamic shared secret between Gj and Ni
$\oplus$	Bitwise XOR function
h ()	one-way hash function
PUF ()	Physically unclonable function

3.1 Setup Phase

3.1.1 Node Enrollment

The goal of this stage is enrolling the node Ni to the healthcare system by the server V. The server V controls the assignment of Ni to a specific gateway Gj. For this reason, the presence of both Ni and Gj at the same time during both stages of the setup phase is assumed. The node enrollment stage is performed in an offline mode and through a secure and trusted channel. The procedure begins with the node Ni identifying itself to the server V by sending its legitimate identity NIDi (assuming that Ni has an assigned identity). After receiving NIDi, the server V generates random challenges C1, C2 …Ci and sends them to the node Ni.

In each time the node Ni receives a challenge Ci, it extracts its corresponding response RSi from its PUF instance and sends it to the server V. It is suggested that this process repeats for about 40,000 times (enough challenge-response pairs to build a model for the PUF with 99.9% prediction accuracy) (Ruhrmair & Solter, 2014). This stage concludes with the server V generates a PUF model for the node Ni and stores it along with NIDi in its database. Figure 4 demonstrates the steps carried out in this stage.

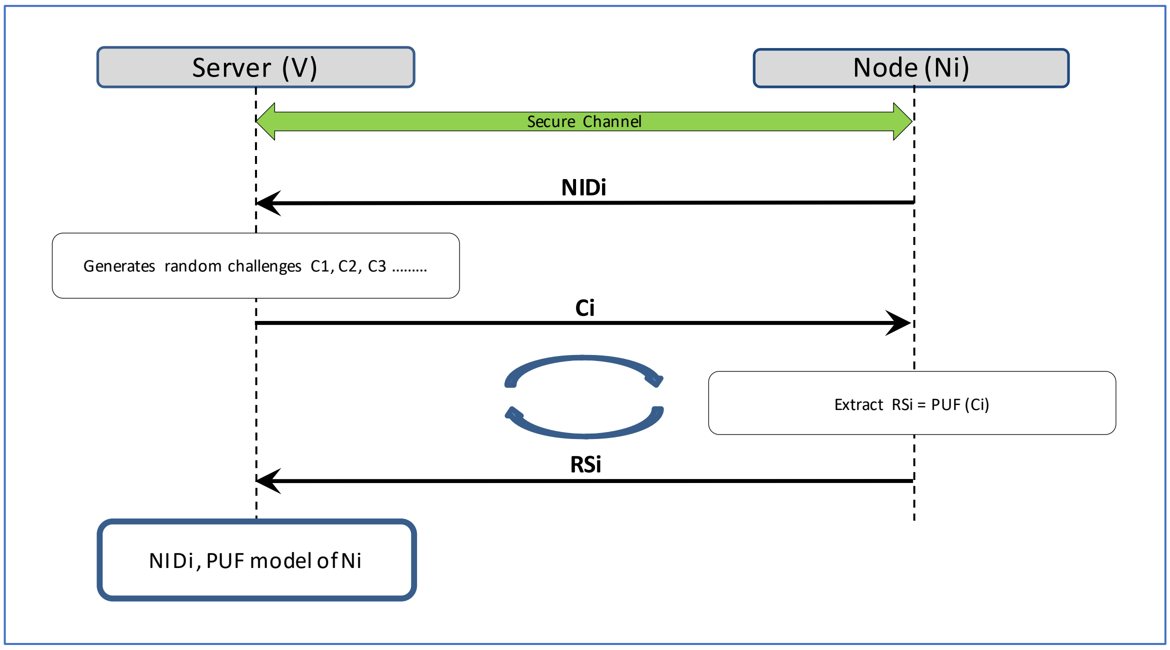


Figure 4: Node enrollment process

3.1.2 Gateway Registration

In this stage, one of the main goals is registering the gateway Gj to the healthcare system by the server V. More importantly, the server V creates a secure channel between all participating entities in the protocol (V, Gj, and Ni) where it computes needed secrets, and initializes identifiers and verification parameters. This stage is executed after the node Ni is enrolled in the system by the server V and server V has knowledge of NIDi and its associated PUF model. It is performed in an offline mode and through a secure and trusted channel. Figure 5 illustrates the procedures for realizing the goals of this stage. The detailed steps are presented as follows:

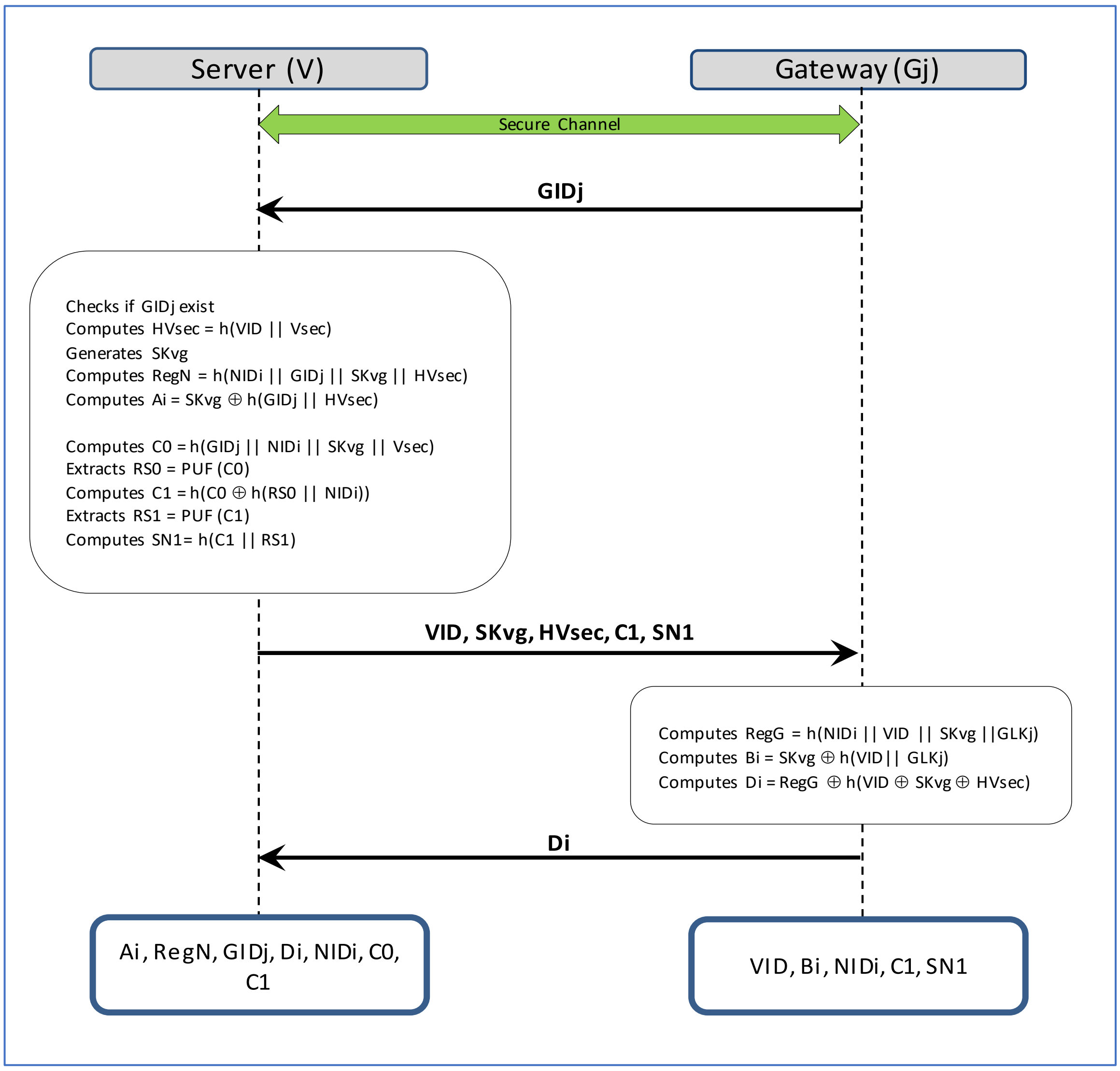

Server (V)
Gateway (Gj)
Secure Channel
GIDj
Checks if GIDj exist
Computes HVsec = h(VID || Vsec)
Generates SKvg
Computes RegN = h(NIDi || GIDj || SKvg || HVsec)
Computes Ai = SKvg ⊕ h(GIDj || HVsec)
Computes C0 = h(GIDj || NIDi || SKvg || Vsec)
Extracts RS0 = PUF (C0)
Computes C1 = h(C0 ⊕ h(RS0 || NIDi))
Extracts RS1 = PUF (C1)
Computes SN1= h(C1 || RS1)
VID, SKvg, HVsec, C1, SN1
Computes RegG = h(NIDi || VID || SKvg ||GLKj)
Computes Bi = SKvg ⊕ h(VID|| GLKj)
Computes Di = RegG ⊕ h(VID ⊕ SKvg ⊕ HVsec)
Di
Ai, RegN, GIDj, Di, NIDi, C0, C1
VID, Bi, NIDi, C1, SN1

Figure 5: Gateway registration process

Step 1: The gateway Gj initiates this stage by sending its identity GIDj to the healthcare server V as it is needed to be included in later computation.

Step 2: Upon receiving GIDj, the server V checks if this gateway is already registered in its database - as the gateway Gj could be registered previously with different node Ni. Next, the server V computes a masked version of its secret Vsec as HVsec = (VID || Vsec) and generates a random secret SKvg to be shared with the gateway Gj. After that, it computes registration value for the node Ni as RegN = h(NIDi || GIDj || SKvg || HVsec) to be used in later verification. Finally, a masked version of SKvg is computed as Ai. Both HVsec and GIDj take part in the masking process.

Step 3: First, the server V computes C0 = h(GIDj || NIDi || SKvg || Vsec). Then, it extracts the response RS0 of C0 from the PUF model for Ni. Next, it computes the challenge C1 as h(C0 ⊕ h(RS0 || NIDi)); extracts its response RS1 from the PUF model of Ni; and computes the shared secret between the gateway and the node SN1. Lastly, the server V sends the parameters VID, Ai, HVsec (from step 2), C1, and SN1 to the gateway Gj.

Step 4: After receiving the server V’s message, the gateway Gj computes registration verifier parameter RegG as h(NIDi || VID || SKvg || GLKj). Then, it hides the shared secret SKvg by masking it with h(VID || GLKj) and save it as Bi. After that, it masks RegG with h(VID ⊕ SKvg ⊕HVsec) as Di. Finally, the gateway sends Di to the server V.

Step 5: The server V stores Ai, RegN, GIDj, Di, NIDi, C0, and C1, while the gateway Gj stores VID, Bi, NIDi, C1, and SN1.

In the proposed protocol, the challenge C0 (Step 3) is considered as a reference identifier for the node Ni in the system and is kept secret. C0 is constructed as a function of parameters from all the protocol participants such as the server V’s private key (Vsec) and shared secret between V and the gateway Gj (SKvg). On the other hand, the challenge C1 is considered as initial sudo-identity and is constructed as a function of only the node Ni’s characteristics such as assigned identity NIDi and the reference challenge C0. C1 is also used as a function in constructing next challenge Ci during protocol run. C1 and SN1 play a key role in the ratchet design. C0 also could be utilized in a fallback mechanism to retrieve previous Ci.

At the end of this phase (setup), two secured channels are established. The first channel is between the server V and the gateway Gj. This channel is proved by RegN and RegG to verify the authentication process (constructed with the private keys of V and Gj, Vsec and GLKj, respectively), and SKvg as a pre-set shred secret for encryption. The second channel is between the gateway Gj and the node Ni. This channel is proved by C1 for authentication provision and SN1 as a dynamic shared secret. Both Ci and SNi are a function of the PUF response RSi which considered the root of trust for the proposed protocol. The server V verifies the newly generated Ci during the last step of the protocol run (next phase).

3.2 Authenticated Key Establishment Phase (Protocol Run)

This is the main phase of the protocol that will run continuously to provide a secure channel to SHS. Its goal is to secure the connection between the server V (could be represented by registered application process or a user and is referred to as “authority”) and enrolled node Ni, via the intermediary gateway Gj. To fulfil the goals, all participants mutually authenticate each other, as well as contribute to (agree on) session key construction. A summary of this phase is provided in Figure 6 and the steps are described as follows:

Step 1: After choosing the desired Ni and its associated Gj, the authority on the server V initiates the process (and authenticates itself) by entering VID and Vsec to get the masked secret value HVsec*. Then, server V retrieves SKvg* from Ai and computes RegN* using h(NIDi || GIDj || SKvg || HVsec*). After that, server V verifies whether RegN* (computed) value is equal to RegN (stored). If RegN* is not valid, the server V aborts the login process, otherwise, the server proceeds with the following operations: generates random value Rv; retrieves RegG from Di and h(VID ⊕ SKvg ⊕ HVsec); masks Rv as P1 by h(SKvg || GIDj) ⊕ Rv; and computes the verification parameter P2 by h(VID || NIDi || RegG || Rv). Finally, server V sends message M1 to Gj comprising GIDj, C1, P1, and P2.

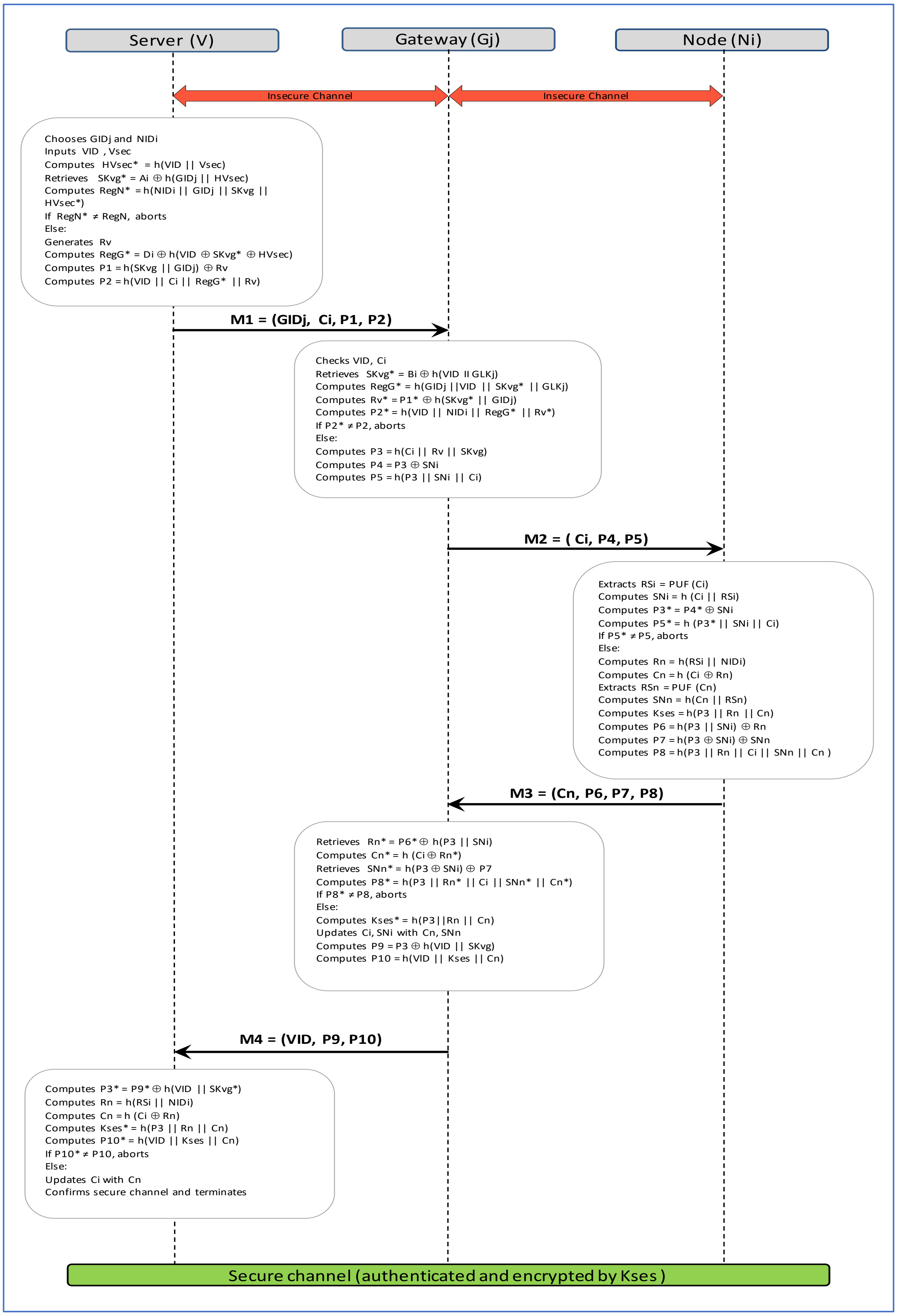

Server (V)
Gateway (Gj)
Node (Ni)
Insecure Channel
Insecure Channel
Chooses GIDj and NIDi
Inputs VID , Vsec
Computes HVsec* = h(VID || Vsec)
Retrieves SKvg* = Ai ⊕ h(GIDj || HVsec)
Computes RegN* = h(NIDi || GIDj || SKvg || HVsec*)
If RegN* ≠ RegN, aborts
Else:
Generates Rv
Computes RegG* = Di ⊕ h(VID ⊕ SKvg* ⊕ HVsec)
Computes P1 = h(SKvg || GIDj) ⊕ Rv
Computes P2 = h(VID || Ci || RegG* || Rv)
M1 = (GIDj, Ci, P1, P2)
Checks VID, Ci
Retrieves SKvg* = Bi ⊕ h(VID II GLKj)
Computes RegG* = h(GIDj ||VID || SKvg* || GLKj)
Computes Rv* = P1* ⊕ h(SKvg* || GIDj)
Computes P2* = h(VID || NIDi || RegG* || Rv*)
If P2* ≠ P2, aborts
Else:
Computes P3 = h(Ci || Rv || SKvg)
Computes P4 = P3 ⊕ SNi
Computes P5 = h(P3 || SNi || Ci)
M2 = (Ci, P4, P5)
Extracts RSi = PUF (Ci)
Computes SNi = h (Ci || RSi)
Computes P3* = P4* ⊕ SNi
Computes P5* = h (P3* || SNi || Ci)
If P5* ≠ P5, aborts
Else:
Computes Rn = h(RSi || NIDi)
Computes Cn = h (Ci ⊕ Rn)
Extracts RSn = PUF (Cn)
Computes SNn = h(Cn || RSn)
Computes Kses = h(P3 || Rn || Cn)
Computes P6 = h(P3 || SNi) ⊕ Rn
Computes P7 = h(P3 ⊕ SNi) ⊕ SNn
Computes P8 = h(P3 || Rn || Ci || SNn || Cn)
M3 = (Cn, P6, P7, P8)
Retrieves Rn* = P6* ⊕ h(P3 || SNi)
Computes Cn* = h (Ci ⊕ Rn*)
Retrieves SNn* = h(P3 ⊕ SNi) ⊕ P7
Computes P8* = h(P3 || Rn* || Ci || SNn* || Cn*)
If P8* ≠ P8, aborts
Else:
Computes Kses* = h(P3||Rn || Cn)
Updates Ci, SNi with Cn, SNn
Computes P9 = P3 ⊕ h(VID || SKvg)
Computes P10 = h(VID || Kses || Cn)
M4 = (VID, P9, P10)
Computes P3* = P9* ⊕ h(VID || SKvg*)
Computes Rn = h(RSi || NIDi)
Computes Cn = h (Ci ⊕ Rn)
Computes Kses* = h(P3 || Rn || Cn)
Computes P10* = h(VID || Kses || Cn)
If P10* ≠ P10, aborts
Else:
Updates Ci with Cn
Confirms secure channel and terminates
Secure channel (authenticated and encrypted by Kses)

Figure 6: Authenticated key establishment phase (Protocol run)

Step 2: Upon receiving the login message M1, the gateway Gj checks VID and Ci against its stored credentials. If there is a match, it proceeds as follows: retrieves SKvg* form Bi and h(VID || GLKj); computes RegG from h(GIDj || VID || SKvg* || GLKj); retrieves Rv* form P1 and h(SKvg* || GIDj); and computes verification parameter P2* = h(VID || NIDi || RegG* || Rv*). Gj verifies that P2* equals received P2. If P2* ≠ P2 the protocol aborts, else, Gj produces Rg by computing P3 = h(Ci || Rv || SKvg). Gj also masks P3 with SN1 as P4, computes P5 = h(P3 || SNi || Ci) for verification, forms message M2 = C2, P4, and P5, and sends it to the node Ni.

Step 3: When the node Ni receives M2, it verifies its legitimacy and authenticity first. To carry out this task Ni does the following: retrieves RSi from the PUF of Ci; computes SNi = h (Ci || RSi); retrieves P3* by masking P4 and h(SNi); and computes verification value P5* = h (P3* || SNi || Ci). Then, Ni verifies whether P5* is equal to the received P5. If P5* is not valid, the protocol aborts, otherwise, Ni performs the following: generates Rn = h(RSi || NIDi); computes the new challenge Cn as h(Ci || Rn); retrieves RSn from the PUF of Cn; and computes session key Kses from h(P3 || Rn || Cn). After that, it masks Rn with h(P3 || SNi) as P6, SNn with h(P3 ⊕ SNi) as P7, and computes verification value P8 = h(P3 || Rn || Ci || SNn || Cn). Lastly, Ni forms the message M3 = (C2, P6, P7, P8) and sends it back to the gateway Gj.

Step 4: Upon receiving the message M3, Gj first retrieves Rn* by xoring P6 with h(P3 || SNi), computes Cn* = h (Ci ⊕ Rn*), and retrieves SNn* by xoring P7 with h(P3 ⊕ SN1). Then, it computes the verification parameter P8* = h(P3 || Rn* || Ci || SNn* || Cn*). If P8* ≠ received P8 holds, the protocol aborts, else, Gj proceeds with computing the session key Kses = h(P3 || Rn || Cn) and updating Ci and SNi with Cn and SNn. After that, Gj masks P3 with h(VID || SKvg) as P9 and computes the verification parameter P10 = h(VlD || Kses || Cn). Finally, Gj constructs message M4 comprising VID, P19, and P10, and sends it to the server V.

Step 5: After the server V receives M4 from Gj, it first verifies its legitimacy and authenticity. To accomplish this, V retrieves P3* form P9* ⊕ h(VID || SKvg*), computes Rn = h(RSi || NIDi) and Cn = h (Ci ⊕ Rn) with aid of Ni PUF model. Then V constructs the session key Kses* = h(P3 || Rn || Cn) and forms the verification message P10* = h(VlD

|| Kses* || Cn*) using previously retrieved and computed parameters . If P12* ≠ P12, the protocol aborts, else, V updates Ci with Cn in its database, confirms secure channel establishment, and terminates.

3.3 Protocol Simulation

3.3.1 Overview

To verify that the proposed protocol will function as required and is secure against the threats in the attack model (Section 2.2), a formal verification tool is required. Formal verification tools simulate and validate security protocols. Automated Validation of Internet Security Protocols and Applications (AVISPA) is one of the verification tools that is extensively used and supported by the research community (Armando et al., 2005; Viganò, 2006). AVISPA tool uses model checking and state exploration approach. This tool is used to evaluate the functionality and security of the proposed protocol. The reason for choosing AVISPA over other tools such as ProVerif and Scyther are:

- AVISPA is actually a back-end framework of four well-known tools. These tools are: OFMC (On-the-fly Model-Checker), CL-AtSe (Constraint-Logic-based Attack Searcher), SATMC (SAT-based Model-Checker) and TA4SP (Tree Automata-based Protocol Analyzer) (Viganò, 2006).
- AVISPA can analyze small, medium-scale protocols as well as large scale internet security protocols. It also detects attacks against a protocol for infinite as well as predefined number of sessions.
- AVISPA can analyze many algebraic properties such as Exclusive-Or and exponentiation natively, as opposed to some other tools that either cannot deal with these properties or introduce an add-on to achieve that.

AVISPA tool provides a sophisticated language called high level protocol specification language (HLPSL) to implement a variety of security protocols and describe their security features. HLPSL is a role-based language. Each participant in the protocol plays a role. There is also a session and an environment section that connects all the roles and the way they interact. The goal section sets the security objectives of the protocol.

3.3.2 Simulation Process

1. The proposed protocol design was translated to HLPSL language. The HLPSL script written as text and then saved in a .hlpsl file, as detailed in appendix. A summary of the HLPSL language types used in the protocol is as follows:

   agent: to represent the participant/entity of the protocol.

   symmetric_key: defines the private-key of the agents.

   function: defines functions for the whole domain of the message and is helpful in modeling cryptographic functions. For example, a hash function is a function of type *hash_func.*

   channel: defines the communication channel between the agents. It is used to send/receive messages to/from another agent.

   played_by: to specify the role played by each agent.

   local: declares local variables for agents. For example, to declare variable *State* as *nat* (a natural number) to an agent that indicates its local state. The initial value of the variable should be set to zero in the *init* section.

   text: to indicate a type of *text* for the message and is considered to be fresh. For example, a nonce.

   nat : represents natural numbers in non-message contexts.

   const: declares a global constant for the roles.

   protocol_id: declares constants as protocol identifiers. For example, indicating a secret between different agents which to be measured later.

   Snd/Rcv: defined as of type channel (dy) for communication between agents.

   P = | > *Q*: represents the immediate transition of an event *P* that is related to an action *Q*.

witness(A, B, id, T): declares authentication property T of the agent *A* by the agent *B* on the protocol identifier *id* in the goal section. The statement *authentication_on id* must be present in the goal section.

request(B, A id, T): denotes strong authentication property of *A* by *B* on T on the protocol identifier *id* in the goal section.

secret(T, idt; A, B): indicates that T is a secret and known for only *A* and *B*. The identifier *idt* measures the secrecy of *E* in the goal section. The *statement secrecy_of idt* must be presented in the goal section.

2. An interface tool called security protocol animator (SPAN) which is installed in a Linux operating system environment converts the protocol HLPSL script using the HLPSL2IF translator into a lower level specification called Intermediate Format (IF). The IF specification is then fed to the back-ends. Each back-end provides the results of the protocol analysis in an Output Format (OF). This process is illustrated in Figure 7.

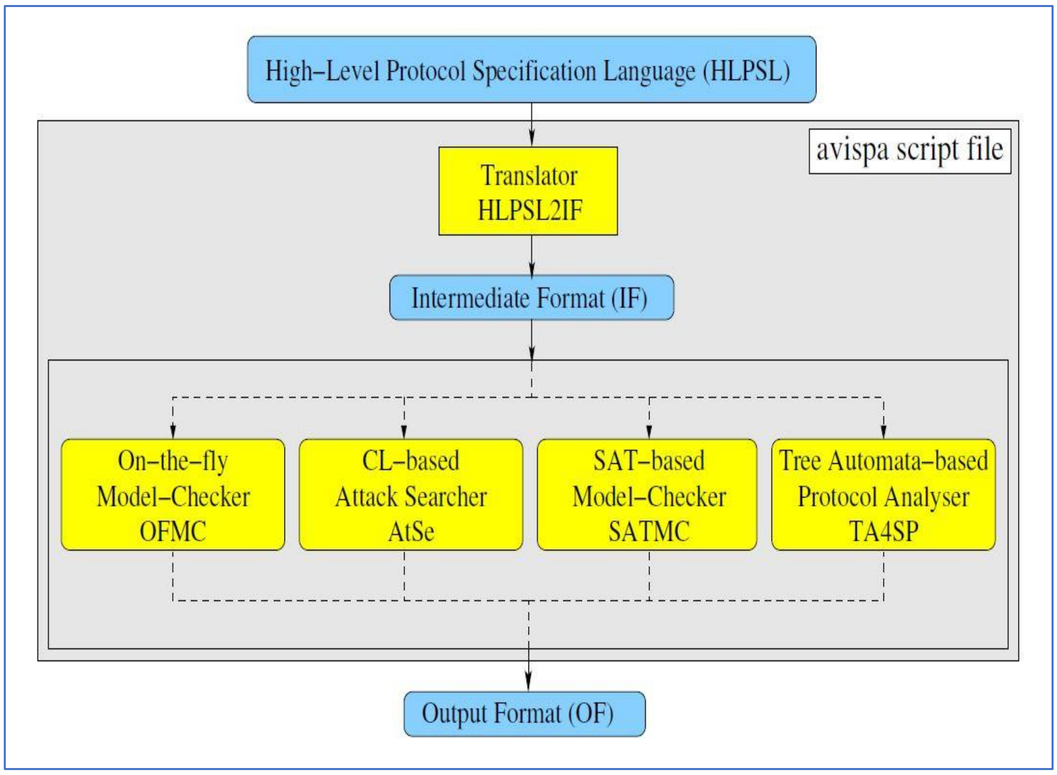
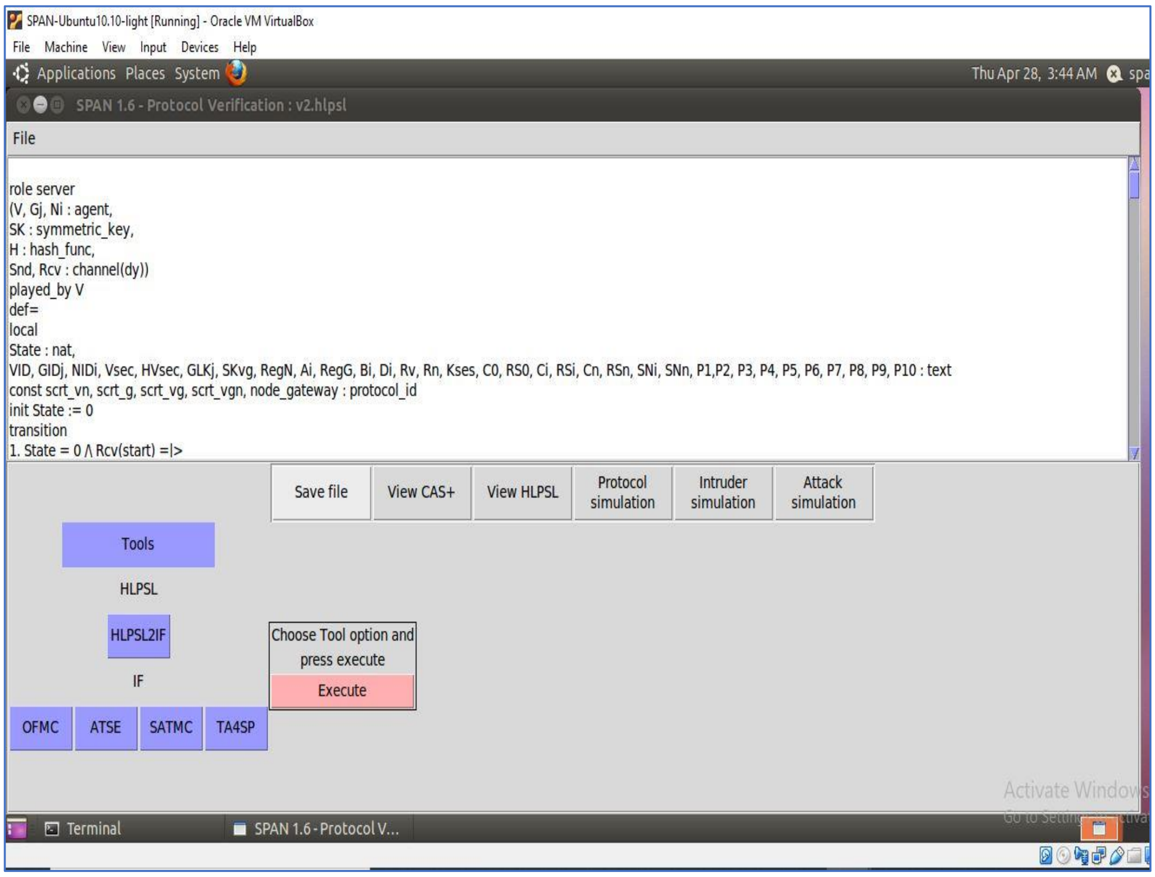

High–Level Protocol Specification Language (HLPSL)
avispa script file
Translator HLPSL2IF
Intermediate Format (IF)
On–the–fly Model–Checker OFMC
CL–based Attack Searcher AtSe
SAT–based Model–Checker SATMC
Tree Automata–based Protocol Analyser TA4SP
Output Format (OF)

Figure 7: Simulation process

3. When the SPAN tool executes an HLPSL specification file, the chosen back-end analyzes the security of the protocol and provides a descriptive output. The output results describes whether the protocol is secure or has weaknesses against certain attacks(s). Figure 8 shows a screenshot of the proposed protocol's HLPSL file opened with SPAN.



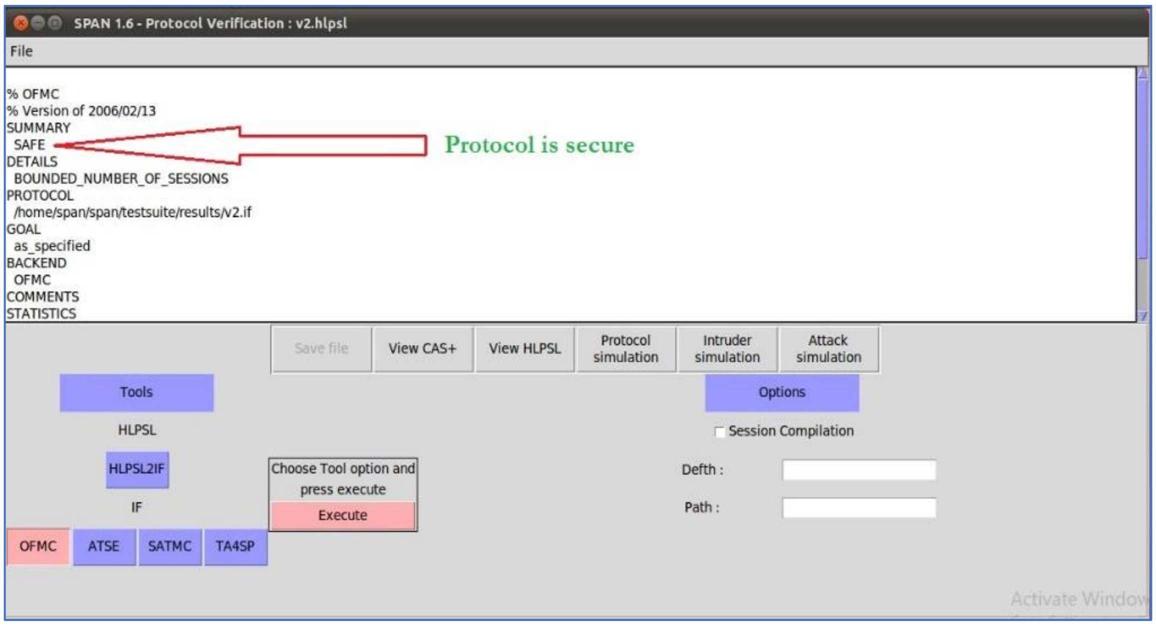
```
role server
(V, Gj, Ni : agent,
SK : symmetric_key,
H : hash_func,
Snd, Rcv : channel(dy))
played_by V
def=
local
State : nat,
VID, GIDj, NIDi, Vsec, HVsec, GLKj, SKvg, RegN, Ai, RegG, Bi, Di, Rv, Rn, Kses, C0, RS0, Ci, RSi, Cn, RSn, SNi, SNn, P1,P2, P3, P4, P5, P6, P7, P8, P9, P10 : text
const scrt_vn, scrt_g, scrt_vg, scrt_vgn, node_gateway : protocol_id
init State := 0
transition
1. State = 0 /\ Rcv(start) =|>
```

SPAN-Ubuntu10.10-light [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Applications Places System
Thu Apr 28, 3:44 AM
SPAN 1.6 - Protocol Verification : v2.hlpsl
File
Save file
View CAS+
View HLPSL
Protocol simulation
Intruder simulation
Attack simulation
Tools
HLPSL
HLPSL2IF
IF
OFMC
ATSE
SATMC
TA4SP
Choose Tool option and press execute
Execute
Terminal
SPAN 1.6 - Protocol V...

Figure 8: SPAN tool

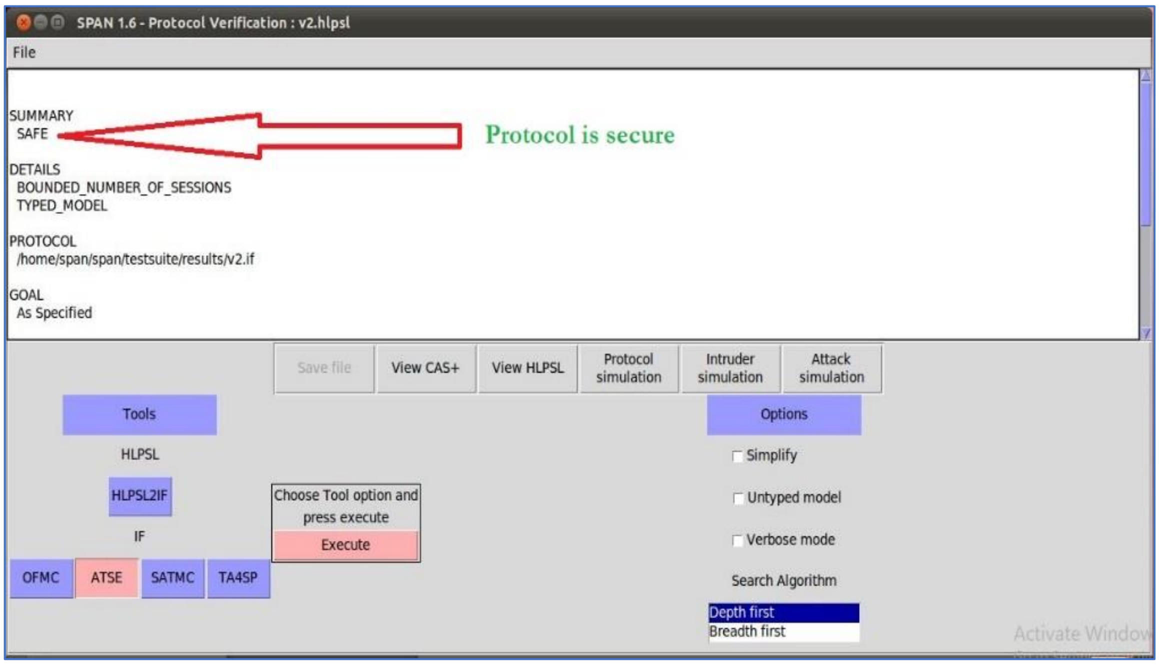
3.3.3 Simulation Results

The proposed protocol was verified by model checkers OFMC and CL-AtSe back-ends and the simulation results are presented in Figure 9 and 10. Both results on the model checkers OFMC and CL- AtSe showed that the proposed protocol is "SAFE" which concludes that the protocol is secure against known attacks. The simulation against the other two model checkers SATMC and TA4SP are not applicable since these back-ends do not support the XOR operation.



```
% OFMC
% Version of 2006/02/13
SUMMARY
 SAFE
DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
 /home/span/span/testsuite/results/v2.if
GOAL
 as_specified
BACKEND
 OFMC
COMMENTS
STATISTICS
```

Figure 9: Simulation results of OFMC



```
SUMMARY
 SAFE

DETAILS
 BOUNDED_NUMBER_OF_SESSIONS
 TYPED_MODEL

PROTOCOL
 /home/span/span/testsuite/results/v2.if

GOAL
 As Specified
```

Figure 10: Simulation results of CL-AtSe

3.4 Discussion and Analysis

In this section, the proposed protocol is analyzed and its validity is discussed. As the protocol is designed with rigid requirements in mind (described in Section 3.3), the analysis details how these requirements are satisfied in the proposed protocol. The analysis focuses on the gateway and the node as well as the exchanged messages between them during protocol run. The server involvement in the protocol will be discussed only as needed. The performance of the proposed protocol against other work is also analyzed.

3.4.1 Proof of Mutual Strong Entity Authentication

The proposed protocol achieves strong entity authentication by using nonces (random challenges) in a challenge-response mechanism to provide freshness and cryptographically bind entity's identity to the corresponding authentication messages. The nonce challenges are represented by Rv, Rg, and Rn. Response is denoted by the messages contain the encrypted nonce received from the authenticating party along with its identity, as a fresh proof of its current participation. In the proposer protocol nonces are considered instead of timestamps or counters (sequence numbers). This is because timestamps and counters require synchronization and keeping state which must be maintained securely. They are also more prone to de-synchronization attacks and might add overhead to constrained nodes.

For example, when the gateway Gj wants to authenticate the node Ni, it initiates the authentication process by sending Rg (masked with SNi as P4) in message M2 as the challenge (step 2). The node Ni responds by sending back Rg and its identity Ci - cryptographically protected by newly generated shared key SNn - represented by parameter P8, with the message M3 (step 3). This way, the protocol provides strong entity authentication of Ni to Gj, where: Gj is convinced that the other party is Ni; Gj is assured that Ni participates in the current protocol run; and Gj knows that Ni is intentionally communicating with it. Similarly, the gateway Gj authenticates itself to the node Ni with Cn, Rn and verification parameter P8.

3.4.2 Proof of Session Key Security

Key freshness: The proposed protocol establishes the session key Kses with participating entities' contributions, i.e., key agreement. Each entity contributes with a random value that it freshly generated. Kses is computed as a function of these random values. The function used is a one-way function. In particular, the protocol computes Kses as a function of P3 (includes both Rv and Rg) which is generated in the gateway G and Rn that is constructed from the PUF in the node Ni. In the protocol, this is denoted by Kses = h(P3 || Rn || Cn). This way, each participant in the protocol is confident that Kses is fresh.

Key authentication: In order to be able to construct the session key Kses, V, Gj, and Ni should receive other participants' contributions (Rv, Rg, or Rn) from each other in a secure manner. Ni receives P3 from Gj in an encrypted form (protected with dynamic shared key SNi). Likewise, Gj receives Rn from Ni encrypted with SNi and Rv from V encrypted with shared key between them SKvg. Since SNi is known to only Ni and Gj, they can both assured the authenticity of the exchanged random values. Similarly, as SKvg is a secret that is shared between only Gj and V, they can be confident that Rv is authentic.

Key integrity: The protocol verifies the integrity of all inputs to the session key Kses function (Rv, P3, Rn, and Cn) after their transport between the participants. The verification process ensures that sent messages are not tampered with or modified by an adversary. The protocol accomplishes this task by validating the received verification parameter against the computed one. The verification parameters are generated with a one-way function that is not possible to reverse.

For example, the gateway Gj computes verification parameter P5 = h(P3 || SNi || Ci) that includes both Rv and Rg and sends it with message M2. When the node Ni receives M2, it computes P5 as well after retrieving other needed parameters. If received parameter P5 is equivalent to the one computed in Ni that proves the message M2 has not been altered. Similarly, the node Ni computes verification parameter P8 = h(Rn || Ci || SNn || Cn) that includes Rn and Cn and sends it with M3. The gateway Gj verifies the received parameter P8 against the computed one.

Key confirmation: In the proposed protocol, all participants can confirm that the established session key Kses is a good key (fresh and authentic) since they could ensure the freshness and authenticity of the received Kses contributions. Furthermore, the server V - the initiator of the protocol and final data aggregation point – can confirm that the node Ni (data collector) possess the same session key Kses. This is achieved by inclusion of the computed Kses in the gateway and the new challenge Cn in the verification parameter P10. The server V is assured that the node Ni is mutually believe in the session key Kses and wants to communicate with it using Kses when it validates the integrity of P10.

3.4.3 Proof of Resistance to Core Attacks

The proposed protocol prevents node impersonation attack with a dynamic pseudo identity mechanism where the identity of the node Ni changes not only for each session, but even in each message. The pseudo identity is represented with challenges Ci and Cn. The challenges are also cryptographically bound in the verification parameters P5 and P8. Furthermore, even if the adversary succeeded in masquerading as the node Ni to the gateway Gj, this attempt would fail when the server V verifies that the new challenge Cn received from Gj is actually not valid.

Replay attack also is not possible with the proposed protocol because of the freshness element in the messages exchanged, particularly between the gateway Gj and the node Ni. Freshness is provided by the newly generated random nonces in each session Rv, Rg (P3), and Rn. The sending party generates the nonce and sends it - encrypted - to the other party. The receiving party returns it back cryptographically bound to the message.

Preventing message tampering assures the integrity of the messages sent form one participant to another. In the proposed protocol, this is realized by the verification parameters P2, P5, P8, and P10. In each step in the protocol, sending entity computes a verification parameter for the message it intends to send and includes it with the message itself. The recipient entity verifies that the message was not modified during transmission by computing the same verification parameter and comparing it with the received one. The function used to create the verification parameters is a one-way function that is easy to compute but hard to manipulate.

As for de-synchronization attack, it is more common with protocols that use time stamps or a counter as a freshness element or when certain parameters need to be updated in each session run. This attack is not applicable in the proposed protocol as nonces are used instead. However, the attacker could try to manipulate the pseudo identity Ci to impact the sequence of the protocol as Ci get updated in each session run and plays significant role in the protocol. The attacker would not gain anything as manipulating Ci has no direct effect on the node Ni and could be easily detected in the verification process and the server V, similar to the countermeasure technique to prevent node impersonation.

Since the communication channel is insecure and exposed, the proposed protocol is prone to many more attacks. Nevertheless, most of the advanced attacks primarily rely on the previous core attacks to succeed. The proof of protocol's resistance against the core attacks infers that it is also secure against more advanced attacks such as privileged insider attack and man in the middle attacks.

3.4.4 Proof of Perfect Forward Secrecy (PFS)

The proposed protocol’s design prevents an adversary from obtaining past session keys even upon compromising all the secret long-term keys. In the proposed protocol, the adversary has the ability to read exchanged messages between any two participants. Assuming that the adversary acquired previous messages M2 and M3, then, at some point, compromised the secrets Vsec, SKvg, GLKj, and SNi, the adversary cannot obtain or compute any previous session key Kses. This is due to the following: First, the adversary could decrypt the collected messages with the possessed secret keys and get Rv and Rg (P3), but not Rn. Rn is transported from the node to the gateway encrypted with the node secret SNi. SNi is dynamic and updated in each session run. Second, none of the compromised long-term keys participate in the generation or transportation new SNi (SNn). Furthermore, SNn is constructed as a function of the response RSn which is never revealed in the messages and the function is irreversible.

3.4.5 Proof of Machine Learning Modeling Attack (ML-MA) Resistant

The design and security of the proposed protocol depend on the challenge-response relationship of the PUF instance in the node Ni. An adversary could utilize ML-MA to disclose this relationship to break the security of the protocol. The success of ML-MA depends on collecting a vast number of challenge-response pairs to construct a model for the PUF. This is not possible in the proposed protocol. While the adversary could easily collect challenges Ci (sent in the clear), the responses RSi are never revealed - in any form - in the protocol messages or the gateway, to be collected. The worst-case scenario is that the adversary is able to collect Ci, SNi pairs - by any means - and attempts to use machine learning techniques to extract RSi from SNi. The adversary would not be able to achieve his/her goal because even though SNi is in fact a function of RSi, the function is only one-way.

3.4.6 Performance Analysis and Comparison

In this section, the performance of the proposed protocol is evaluated in terms of computation costs as well as advanced security features. To demonstrate the advantages of the proposed protocol, it was compared with some of the recently proposed protocols from the literature, such as protocols of Fotouhi et al. (2020), Kompara et al. (2019), Gope et al. (2019), and Banerjee et al. (2019).

All the protocols comprise of pseudo random number generator (PRNG), bitwise XOR, and one-way function such as hash operations. The proposed protocol and protocols (Banerjee et al., 2019; Gope et al., 2019) also carry out PUF operation as well. The PRNG and XOR operations are assumed to have very low overhead and are omitted from the computation cost calculations. Based on the work of Banerjee et al. (2019), the PUF and hash operations (p and h) in the node are reported with computation time of 0.43 and 1.37 milliseconds (ms), respectively, while the hash operation in the gateway is evaluated at 0.68 milliseconds. The computation cost for both the node and gateway is calculated as the computation time multiplied by the number of occurrences of that operation. Table 4 shows the computation cost comparison.

Table 4: Computation cost comparison

Protocol	**Node**		**Gateway**		**Total**
	Operations	Comp cost (ms)	Operations	Comp cost (ms)	Comp cost (ms)
Fotouhi et al. (2020)	7h	9.59	17h	11.56	21.15
Kompara et al. (2019)	3h	4.11	5h	3.4	7.51
Gope et al. (2019)	4h + 2P	6.34	9h	6.12	12.46
Banerjee et al. (2019)	6h + 2P	9.08	8h	5.44	14.52
Proposed protocol	9h + 2P	13.19	10h	6.8	19.99

Although all compared protocols achieve the required basic AKE security goals, their computation time to accomplish this task differs considerably. The proposed protocol has the highest computation cost in the node (13.19 ms) and the second highest in total (19.99 ms) among the other protocols. This computation overhead – especially in the node – is due to the advanced security features in the protocol, such as providing PFS, adaptability to network changes (offline scenarios), and compromised gateway recovery, which are not provided by the other protocols. Table 5 compares the proposed protocol's advanced security features against the other selected protocols.

Table 5: Advanced features comparison

Protocol	**PUF**	**PFS**	**Network adaptability**	**Compromised or lost gateway**	**Node anonymity**
Fotouhi et al. (2020)	×	✓	×	×	✓
Kompara et al. (2019)	×	×	×	×	×
Gope et al. (2019)	✓	✓	×	×	✓
Banerjee et al. (2019)	✓	✓	×	×	✓
Proposed protocol	✓	✓	✓	✓	✓

The mentioned features are crucial for any AKE protocol and its security. In fact, these features are all concerned with the gateway. In all compared protocols, the gateway acts like a trusted intermediary device, yet is the most vulnerable entity in the protocol. This is because the gateway stores private and secret keys as well as other secret parameters for the system. In addition, if the patient lost the gateway, the connection to the PUF will be lost forever. This is because the CRPs are mostly stored on the gateway. To collect the CRPs, the PUF circuit usually investigated directly (extracting responses from sent challenges) only once, during the setup phase, then there should be a mechanism preventing this for the security of the system. In conclusion, securing the gateway from such scenarios necessitate the increase in computation overhead.

Chapter 4: Conclusion

This thesis aimed to provide a solution to secure end-to-end connection for smart healthcare systems with resource-constrained nodes such as implanted sensors. The proposed solution was designing an enhanced authenticated key establishment (AKE) protocol based on physical unclonable function as the root of trust and with rigid security requirements. The protocol was simulated and validated by AVISPA tool as well as informally verified by analyzing it against the requirements. The formal and informal analysis of the protocol proved the hypothesis. PUF implementation in AKE protocols not only enhances security but also could be leveraged in other features such as anonymity and adaptability for network changes.

4.1 Research Implications

Besides the achievement of its goals, the protocol's design revealed other features that could be a significant impact on other important aspects. First, most protocols assume online connection between the node Ni and the server V via the gateway Gj. A patient utilizing a smart healthcare system where he/she is equipped with implanted sensor node could lose the gateway-server connection (internet) for many reasons. The implication of losing such a connection is that the gateway does not receive the random value generated in the server Rv nor be able to send the updated node secret Cn to it. The server is also not aware of the random values generated it the gateway Rg during the disconnection. In this scenario, the proposed protocol's design allows for a countermeasure mechanism to be easily applied to it to retain its functionality and with the same level of security.

The gateway Gj could reuse the last known Rv as a seed in computing a new Rv in the gateway Gj in such a way that it could be recomputed later in the server V. One suggestion is using a counter. Another recommendation is using Rv as an input to an irreversible function such as h(Rv). Since Rg is constructed as a function of Rv, it could be easily reconstructed back in the server V. One important aspect here is that the server should have a mechanism in place to trace back the modifications.

Another impact of the proposed design is node anonymity, which is a desirable property that assures the privacy of patients and their data. In the proposed protocol, the challenge Ci changes randomly for each session run. In this case, Ci is possibly an ideal candidate to provide anonymity since it represents a pseudonym of the node Ni. Moreover, while Ci is known to the node and the server (computed) even in offline mode, it is transported and updated in the gateway Gi. An alias of the gateway's identity GIDj could be also constructed from Ci, for example, as $h(Ci \oplus GIDj)$.

4.2 Research Limitation and Future Work

The main limitation of the proposed protocol is the assumption of an ideal PUF scenario where responses are 100% reproducible. In fact, the fuzziness of PUF output is one the issues that hinder utilizing PUF more. A generic (arbiter) PUF also was assumed without regard to any specific PUF construction type. In the future, the protocol could be implemented empirically on hardware and the PUF programed in a field programable gate array (FPGA) to verify the applicability and the performance of the protocol in a more realistic scenario. Further improvements to the protocol might include performance optimization as well as more analysis with advanced formal security proof methods such as real-or-random (RoR).

References

Alladi, T., Chamola, V., & Naren (2021). HARCI: A two-way authentication protocol for three entity healthcare IoT networks. *IEEE Journal on Selected Areas in Communications*, *39*(2), 361-369.

Aman, M. N., Chua, K. C., & Sikdar, B. (2017). Mutual authentication in IoT systems using physical unclonable functions. *IEEE Internet of Things Journal*, 4(5), 1327-1340.

Amin, R., Islam, S. H., Biswas, G. P., Khan, M. K., & Kumar, N. (2018). A robust and anonymous patient monitoring system using wireless medical sensor networks. *Future Generation Computer Systems*, *80*, 483-495.

Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuéllar, J., Drielsma, P. H., Heám, P. C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., & Vigneron, L. (2005, July). The AVISPA tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification* (pp. 281-285). Springer, Berlin, Heidelberg.

Banerjee, S., Odelu, V., Das, A. K., Chattopadhyay, S., Rodrigues, J. J., & Park, Y. (2019). Physically secure lightweight anonymous user authentication protocol for internet of things using physically unclonable functions. *IEEE Access*, 7, 85627-85644.

Boyd, C., Mathuria, A., & Stebila, D. (2020). *Protocols for authentication and key establishment* (Vol. 2). Heidelberg: Springer.

Canetti, R., & Krawczyk, H. (2001, May). Analysis of key-exchange protocols and their use for building secure channels. In *International Conference on the Theory and Applications of Cryptographic Techniques* (pp. 453-474). Springer, Berlin, Heidelberg.

Dolev, D., & Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on information theory*, 29(2), 198-208.

Fotouhi, M., Bayat, M., Das, A. K., Far, H. A. N., Pournaghi, S. M., & Doostari, M. A. (2020). A lightweight and secure two-factor authentication scheme for wireless body area networks in healthcare IoT. *Computer Networks*, 177, 107333. DOI: 10.1016/j.comnet.2020.107333.

Gassend, B., Clarke, D., Van Dijk, M., & Devadas, S. (2002, November). Silicon physical random functions. In *Proceedings of the 9th ACM conference on Computer and communications security* (pp. 148-160).

Gassend, B., Dijk, M. V., Clarke, D., Torlak, E., Devadas, S., &Tuyls, P. (2008). Controlled physical random functions and applications. *ACM Transactions on Information and System Security (TISSEC)*, 10(4), 1-22.

Gope, P., Das, A. K., Kumar, N., & Cheng, Y. (2019). Lightweight and physically secure anonymous mutual authentication protocol for real-time data access in industrial wireless sensor networks. *IEEE transactions on industrial informatics*, *15*(9), 4957-4968.

Goutsos, K. (2020). *Physical Unclonability Framework for the Internet of Things*, Doctoral dissertation, Newcastle University. Newcastle, UK.

Goutsos, K., & Bystrov, A. (2019, August). Lightweight PUF-based Continuous Authentication Protocol. In *2019 International Conference on Computing, Electronics & Communications Engineering (iCCECE)* (pp. 229-234). IEEE.

Guajardo, J., Kumar, S. S., Schrijen, G. J., &Tuyls, P. (2007, September). FPGA intrinsic PUFs and their use for IP protection. In *International workshop on cryptographic hardware and embedded system*s (pp. 63-80). Springer, Berlin, Heidelberg.

Hassija, V., Chamola, V., Bajpai, B. C., & Zeadally, S. (2021). Security issues in implantable medical devices: Fact or fiction? *Sustainable Cities and Society*, *66*, 102552. DOI: 10.1016/j.scs.2020.102552.

He, D., Zeadally, S., Kumar, N., & Lee, J. H. (2016). Anonymous authentication for wireless body area networks with provable security. *IEEE Systems Journal*, 11(4), 2590-2601.

Jiang, Q., Ma, J., Yang, C., Ma, X., Shen, J., & Chaudhry, S. A. (2017). Efficient end-to-end authentication protocol for wearable health monitoring systems. *Computers & Electrical Engineering*, 63, 182-195.

Karakoyunlu, D., & Sunar, B. (2010, December). Differential template attacks on PUF enabled cryptographic devices. In *2010 IEEE International Workshop on Information Forensics and Security* (pp. 1-6). IEEE.

Kompara, M., Islam, S. H., & Hölbl, M. (2019). A robust and efficient mutual authentication and key agreement scheme with untraceability for WBANs. *Computer Networks*, 148, 196-213.

Kumar, P., & Lee, H. J. (2012). Security issues in healthcare applications using wireless medical sensor networks: A survey. *sensors*, *12*(1), 55-91.

Lim, D., Lee, J. W., Gassend, B., Suh, G. E., Van Dijk, M., & Devadas, S. (2005). Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10), 1200-1205.

Maes, R. (2013). PUF-based entity identification and authentication. In *Physically unclonable functions* (pp. 117-141). Springer, Berlin, Heidelberg.

Maiti, A., Gunreddy, V., &Schaumont, P. (2013). A systematic method to evaluate and compare the performance of physical unclonable functions. In *Embedded systems design with FPGAs* (pp. 245-267). Springer, New York, NY.

Masdari, M., Ahmadzadeh, S., & Bidaki, M. (2017). Key management in wireless Body Area Network: Challenges and issues. *Journal of Network and Computer Applications*, *91*, 36-51.

Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of applied cryptography*. CRC press.

Pappu, R., Recht, B., Taylor, J., &Gershenfeld, N. (2002). Physical one-way functions. *Science*, 297(5589), 2026-2030.

Poettering, B., & Rösler, P. (2018, August). Towards bidirectional ratcheted key exchange. In *Annual International Cryptology Conference* (pp. 3-32). Springer, Cham.

Ruhrmair, U., & Solter, J. (2014). PUF modeling attacks: An introduction and overview. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1-6).

Rührmair, U., Sehnke, F., Sölter, J., Dror, G., Devadas, S., &Schmidhuber, J. (2010, October). Modeling attacks on physical unclonable functions. *In Proceedings of the 17th ACM conference on Computer and communications security* (pp. 237-249).

Shen, J., Chang, S., Shen, J., Liu, Q., & Sun, X. (2018). A lightweight multi-layer authentication protocol for wireless body area networks. *Future Generation Computer Systems*, 78, 956-963.

Stallings, W., Brown, L. (2015). *Computer security: principles and practice* (pp. 838-0). Upper Saddle River, NJ, USA: Pearson Education.

Sun, Y., Lo, F. P. W., & Lo, B. (2019). Security and privacy for the internet of medical things enabled healthcare systems: A survey. *IEEE Access*, *7*, 183339-183355.

Toorani, M. (2015, January). On vulnerabilities of the security association in the IEEE 802.15. 6 standard. In *International conference on financial cryptography and data security* (pp. 245-260). Springer, Berlin, Heidelberg.

Verbauwhede, I., & Schaumont, P. (2007, April). Design methods for security and trust. In 2007 Design, Automation & Test in *Europe Conference & Exhibition* (pp. 1-6). IEEE.

Viganò, L. (2006). Automated security protocol analysis with the AVISPA tool. *Electronic Notes in Theoretical Computer Science*, 155, 61-86.

Wu, F., Li, X., Sangaiah, A. K., Xu, L., Kumari, S., Wu, L., & Shen, J. (2018). A lightweight and robust two-factor authentication scheme for personalized healthcare systems using wireless medical sensor networks. *Future Generation Computer Systems*, 82, 727-737.

Yanambaka, V. P., Mohanty, S. P., Kougianos, E., & Puthal, D. (2019). PMsec: Physical unclonable function-based robust and lightweight authentication in the Internet of Medical Things. *IEEE Transactions on Consumer Electronics*, *65*(3), 388-397.

Yilmaz, Y., Gunn, S. R., & Halak, B. (2018, July). Lightweight PUF-based authentication protocol for IoT devices. In *2018 IEEE 3rd international verification and security workshop (IVSW)* (pp. 38-43). IEEE.

Appendix

HLPSL code

```
role server

(V, Gj, Ni : agent,

SK : symmetric_key,

H : hash_func,

Snd, Rcv : channel(dy))

played_by V

def=

local

State : nat,

VID, GIDj, NIDi, Vsec, HVsec, GLKj, SKvg, RegN, Ai, RegG, Bi, Di, Rv, Rn, Kses,
C0, RS0, Ci, RSi, Cn, RSn, SNi, SNn, P1,P2, P3, P4, P5, P6, P7, P8, P9, P10 : text

const scrt_vn, scrt_g, scrt_vg, scrt_vgn, node_gateway : protocol_id

init State := 0

transition

1. State = 0 /\ Rcv(start) =|>

State' := 1 /\ HVsec' := H(VID.Vsec)

/\ secret({Vsec}, scrt_vn, {V, Ni})

/\ SKvg' := new()

/\ secret ({SKvg}, scrt_vg, {V, Gj})

/\ RegN' := h(NIDi.GIDj.SKvg.HVsec)

/\ Ai' := xor(SKvg,H(GIDj.HVsec))

/\ C0' := H(GIDj.NIDi.SKvg.Vsec)

/\ RS0' := H(xor(C0,Vsec))

/\ Ci' := H(xor(C0,H(RS0.NIDi)))

/\ RSi' := H(xor(Ci,Vsec))
/\ SNi' := H(Ci.RSi)
/\ secret ({SNi}, scrt_vgn, {V, Gj, Ni})
/\ Snd({VID.SKvg.HVsec.Ci.SNi}_SK)
2. State = 1 /\ Rcv({Di}_SK)=|>
State':= 2 /\ HVsec' := H(VID.Vsec)
/\ SKvg' := xor(Ai,H(GIDj.HVsec))
/\ Rv' := new()
/\ RegG' := xor(Di,H(xor(VID,xor(SKvg,HVsec))))
/\ P1' := xor(Rv,H(SKvg.GIDj))
/\ P2' := H(VID.Ci.RegG.Rv)
/\ Snd(GIDj.Ci.P1.P2)
3. State = 2 /\ Rcv(VID,P9,P10) =|>
State':= 3 /\ P3' := xor(P9,H(VID.SKvg))
/\ RSi' := H(xor(Ci,Vsec))
/\ Rn' := H(RSi.NIDi)
/\ Cn' := H(xor(Ci,Rn))
/\ Kses' := H(P3.Rn.Cn)
/\ Ci' := Cn
end role

role gateway
(V, Gj, Ni : agent,
SK : symmetric_key,
H : hash_func,
Snd, Rcv : channel(dy))
played_by Gj
def=
local
State : nat,
VID, GIDj, NIDi, Vsec, HVsec, GLKj, SKvg, RegN, Ai, RegG, Bi, Di, Rv, Rn, Kses,
Ci, Rsi, Cn, RSn, SNi, SNn, P1,P2, P3, P4, P5, P6, P7, P8, P9, P10 : text

const scrt_vn, scrt_g, scrt_vg, scrt_vgn, node_gateway : protocol_id

init State := 0

transition

1. State = 0 /\ Rcv({VID.SKvg.HVsec.Ci.SNi}_SK) =|>

State' := 1 /\ RegG' := H(NIDi.VID.SKvg.GLKj)

/\ secret({GLKj}, scrt_g, {Gj})

/\ Bi' := xor(SKvg,H(VID.GLKj))

/\ Di' := xor(RegG,H(xor(VID,xor(SKvg,HVsec))))

/\ Snd({Di}_SK)

2. State = 1 /\ Rcv(GIDj.Ci.P1.P2) =|>

State' := 2 /\ SKvg' := xor(Bi,H(VID.GLKj))

/\ RegG' := H(GIDj.VID.SKvg.GLKj)

/\ Rv' := xor(P1,H(SKvg.GIDj))

/\ P3' := H(Ci.Rv.SKvg)

/\ P4' := xor(P3,SNi)

/\ P5' := H(P3.SNi.Ci)

/\ witness(Gj,Ni,node_gateway,P3)

/\ Snd(Ci.P4.P5)

3. State = 2 /\ Rcv(Cn.P6.P7.P8) =|>

State' := 3

/\ Rn' := xor(P6,H(P3.SNi))

/\ Cn' := H(xor(Ci,Rn))

/\ SNn' := xor(H(xor(P3,SNi)),P7)

/\ Kses' := H(P3.Rn.Cn)

/\ Ci' := Cn

/\ SNi' := SNn

/\ P9' := xor(P3,H(VID.SKvg))
/\ P10' := (VID.Kses.Cn)
/\ Snd(VID.P9.P10)
end role

role sensornode
(V, Gj, Ni : agent,
SK : symmetric_key,
H : hash_func,
Snd, Rcv : channel(dy))
played_by Ni
def=
local
State : nat,
VID, GIDj, NIDi, Vsec, HVsec, GLKj, SKvg, RegN, Ai, RegG, Bi, Di, Rv, Rn, Kses,
Ci, RSi, Cn, RSn, SNi, SNn, P1,P2, P3, P4, P5, P6, P7, P8, P9, P10 : text
const scrt_vn, scrt_g, scrt_vg, scrt_vgn, node_gateway : protocol_id
init State := 0
transition
1. State = 0 /\ Rcv(Ci.P4.P5) =|>
State' := 1  /\ RSi' := H(xor(Ci,Vsec))
/\ SNi' := H(Ci.RSi)
/\ P3' := xor(P4,SNi)
/\ Rn' := H(RSi.NIDi)
/\ Cn' := H(xor(Ci,Rn))
/\ RSn' := H(xor(Cn,Vsec))
/\ SNn' := H(Cn.RSn)
/\ Kses' := H(P3.Rn.Cn)
/\ P6' := xor(H(P3.SNi),Rn)
/\ P7' := xor(H(xor(P3,SNi)),SNn)
/\ P8' := H(P3.Rn.Ci.SNn.Cn )
/\ request(Ni,Gj,node_gateway,P3)
/\ Snd(P7, P8)
end role

role session
(V, Gj, Ni : agent,
SK : symmetric_key,
H : hash_func)
def=
local
SVch, RVch, SNch, RNch, SGch, RGch : channel(dy)
composition
server(V,Gj,Ni,SK,H,SVch,RVch)
/\ gateway(V,Gj,Ni,SK,H,SGch,RGch)
/\ sensornode(V,Gj,Ni,SK,H,SNch,RNch)
end role

role environment()
def=
const v, gj, ni : agent,
sk : symmetric_key,
h : hash_func,
scrt_vn, scrt_g, scrt_vg, scrt_vgn, node_gateway : protocol_id
intruder_knowledge = {v,gj,ni,h}
composition
session(v,gj,ni,sk,h)
end role
```

```
goal
secrecy_of scrt_vn
secrecy_of scrt_g
secrecy_of scrt_vg
secrecy_of scrt_vgn
authentication_on node_gateway
end goal

environment()
```

UAE UNIVERSITY MASTER THESIS NO. 2022:16

This thesis proposes an Authenticated Key Establishment (AKE) protocol to secure smart healthcare systems with constrained sensors such as implantable. The protocol utilizes Physical Unclonable Function (PUF) and ratchet technique to satisfy rigid security requirements and investigate some security issues that were not addressed in previous protocols.

Abdalla Elkushli received his MSc in Information Security from the Department of Information Systems and Security, College of Information Technology at UAE University, UAE. He received his BSc from the School of Computing and Information Science, Faculty of Science and Engineering, Anglia Ruskin University, UK.