Lightweight serverless protocols for the internet of things

Collins Mtita

THÈSE DE DOCTORAT CONJOINT TELECOM SUDPARIS et L'UNIVERSITE PIERRE ET MARIE CURIE

Spécialité : Informatique et Réseaux

École doctorale : Informatique, Télécommunications et Électronique de Paris

Présentée par

Collins MTITA

Pour obtenir le grade de
DOCTEUR DE TELECOM SUDPARIS

Les Protocoles de Sécurité Serverless Légers Pour l'Internet des Objets

Soutenue le 13 Juin 2016 devant le jury composé de :

Thèse No : 2016TELE0010

PhD THESIS TELECOM SUDPARIS IN PARTNERSHIP WITH PIERRE ET MARIE CURIE UNIVERSITY

Speciality : Informatics and Networks

Doctoral School : Informatique, Télécommunications et Électronique de Paris

Presented by

Collins MTITA

To obtain the degree of
DOCTOR OF TELECOM SUDPARIS

Lightweight Serverless Protocols for the Internet of Things

Presented on June 13, 2016 in front of the Jury composed by :

Thèse No : 2016TELE0010

Abstract

The most profound technologies are those that disappear.
They weave themselves into the fabric of everyday life
until they are indistinguishable from it.

Mark Weiser
The Computer for the Twenty-First Century,
Scientific American, 1991, pp. 66 - 75

THIS thesis addresses the security and privacy challenges relevant to the resource constrained devices in the era of pervasive computing. Pervasive computing, a term coined by Schechter [90] to describe the idea of computing services available anytime, anywhere and on demand, is characterized by seamless interactions between heterogeneous players in the Internet. This phenomenon allows intelligent chips, sensors or microcontrollers to be embedded into everyday objects to enable them generate, communicate and share information. Pervasive computing accelerates technological evolution by integrating small and resource constrained devices to the Internet arena, eventually opening doors to new services requiring seamless interactions and integrations with the existing technologies, infrastructures and services.

The information collected, stored and communicated by specialized pervasive devices is targeted for various uses such as monitoring, auditing, control or research. The nature of the information generated, stored and shared may require proper security and privacy guarantees. Towards that end, the classical security solutions are not ideal candidates to solve the security and privacy challenges in pervasive systems for two reasons. First, classical security protocols require a lot of resources from the host devices while most of the pervasive devices have very strict resource constraints. Second, most classical security solutions work in a connected mode, which requires constant communication between devices and centralized servers for authentication and authorization purposes. However, pervasive devices may be working in isolated areas with intermittent network coverage and connectivity. Thus, it is ideal to come up with alternative solutions suitable for heterogeneous pervasive devices to smoothly interact, authenticate and securely share information. One of the suitable alternative solutions is the *serverless protocols.*

The term *"serverless protocol"* refers to the mechanism of enabling centrally controlled devices to autonomously authenticate one another, or other heterogeneous devices, without an active participation of the centralized authentication or authorization servers. Serverless protocols prioritize on securing proximity communication between heterogeneous devices while optimizing on the little resources available.

In this thesis, we tackle the challenges of pervasive systems by proposing lightweight and efficient serverless protocols for authenticating heterogeneous pervasive devices during proximity communication. Our proposed protocols derive their originality from the fact that they do not require the communicating parties to have prior relationships with each other, nor to have any previously shared authentication information with each other. Instead, all parties must establish prior relationships with the trusted entity, such as a centralized authentication server, and rely upon it to verify credentials and authorize sessions. Moreover, our proposed solutions incorporate context information to enforce automatic parameter expiry. This property is not supported by most of the earlier versions of the serverless protocol schemes, hence making them vulnerable to different attacks.

Three novel contributions are proposed in this thesis. First, we propose a serverless lightweight mutual authentication protocol for heterogeneous devices. The first contribution includes a formal validation using the AVISPA tool. Second, we propose two complementing protocols using RFID (Radio-Frequency Identification) as a core technology. The first protocol performs mass authentication between an RFID reader and a group of tags and the second protocol performs a secure search for a target tag among a group of tags. The second contribution includes two formal validations ; one is done using the AVISPA tool and the other is done using the CryptoVerif tool. After a thorough study of serverless protocols, we propose our third contribution, a concise guide on how to develop secure and efficient serverless protocols relevant to the pervasive systems.

KEYWORDS : *Security, Privacy, Serverless Protocols, Resource Constraints, Low Energy Footprint, Pervasive Computing, Ubiquitous Computing, Trust, Autonomy.*

Résumé

Les avancées technologiques permettent d'intégrer des capteurs et des modules de communication dans les objets du quotidien pour les rendre intelligents et faciliter leur intégration sur l'Internet. L'Internet du futur sera sans nul doute celui des objets connectés. Les objets connectés génèrent, collectent, stockent et partagent des informations entre eux et aussi avec les serveurs d'authentification centralisés. La plupart des informations collectées doivent être protégées pendant le stockage et le transfert. Par le passé, divers protocoles assurant une sécurité robuste basés sur la cryptographie asymétrique et d'autres sur la cryptographie symétrique ont été proposés dans la littérature. Du fait que les objets connectés possèdent de faibles capacités de calcul, de mémoire et d'énergie, et que l'accès au medium radio est très consommateur en ressources, les protocoles cryptographiques traditionnels ne sont pas adaptés aux objets connectés. Il y a lieu donc d'adapter ou de concevoir des protocoles propres et conformes à leurs exigences.

Dans cette thèse, nous abordons les défis de sécurité et de vie privée pertinents aux systèmes pervasifs avec des contraintes de ressources strictes. Nous regardons les protocoles d'authentification serverless, qui sont des mécanismes d'authentification qui ne nécessitent pas la présence du serveur central au cours de la phase d'authentification entre deux objets connectés.

Tout d'abord, nous fournissons les caractéristiques et les besoins pour les protocoles serverless. Grâce à ces besoins et caractéristiques, nous avons fait des recherches, des analyses complètes et des comparaisons des protocoles serverless existants en termes de sécurité, de vie privée et de performances. Nous examinons leurs capacités à résister à diverses attaques et leurs aptitudes à minimiser l'usage des ressources. Après quoi, notre objectif est de proposer des protocoles de sécurité serverless permettant aux objets de s'authentifier tout en garantissant efficacité, passage à l'échelle et efficacité énergétique, l'énergie étant une ressource très critique qui a une influence directe sur la durée de vie d'un objet connecté.

Trois nouvelles contributions sont proposées dans cette thèse. Notre première contribution est un protocole léger serverless d'authentification mutuelle pour les objets connectés hétérogènes. La première contribution fournit trois avantages par rapport aux protocoles existants. Cette contribution répond aux exigences des systèmes pervasifs. La validation de notre proposition a été faite en utilisant l'outil AVISPA et la validation informelle en utilisant sécurité et de vie privée des jeux.

Notre deuxième contribution comprend deux protocoles complémentaires dans le domaine des technologies RFID. Le premier protocole vise à l'authentification de masse entre un lecteur RFID et un groupe d'étiquettes tandis que le deuxième protocole effectue une recherche sécurisée pour une étiquette cible parmi un groupe d'étiquettes dans le voisinage du lecteur. Les deux protocoles proposés tiennent compte des contraintes de ressources des étiquettes RFID.

Après une étude approfondie des protocoles serverless, nous avons proposé une troisième contribution, un guide pour la conception des protocoles serverless sécurisé et efficaces pour les systèmes pervasifs. Le guide contient six principes et six meilleures pratiques en vue d'élaborer des protocoles serverless. Le guide est destiné à aider à la conception de protocoles serverless efficaces, sécurisés et simples en évitant des erreurs couramment faites dans les protocoles existants.

Mots clés : *Sécurité, Protocoles de recherche sécurisés, Vie privée, Contrôle d'accès, Protocoles légers serverless, Faible empreinte énergétique, Authentification mutuelle, Confiance*

Acknowledgement

Kidole kimoja hakivunji chawa.

- A Swahili Proverb

A THESIS is a long journey with a life of its own. For such a journey to be enjoyable and complete one needs a strong and dedicated army of help and support, in every possible way. I have been blessed to have had such a support throughout my journey. My experience for the past three years have been nothing short of amazing and fulfilling, thanks to the people who have, in one way or the other, participated in making it so. You might have considered our brief encounters during these past years trivial, but to me those moments were precious. Through the past years I had a pleasure to cooperate, discuss and share ideas with various wonderful people. Jotting down names in this page may not suffice to express my gratitude, but you should know that I am forever grateful and honored to have shared your experiences.

First and foremost, I would like to express my sincere gratitude to Prof. Maryline LAURENT, who is my thesis Director and was once my Masters program coordinator. Prof. LAURENT has been a great source of inspiration throughout the past 6 years. This journey was instigated by her full confidence and belief in me. I could not have asked for a better mentor, so qualified in what she does and yet so understanding of the diversity of ideas. I am utterly grateful to have been under her supervision.

I am deeply grateful to my thesis dissertation committee Professors ; their comments and feedback greatly improved the quality of my thesis dissertation.

TRAXENS has been very instrumental in making this thesis a success. My special thanks to the whole TRAXENS team, especially Michel FALLAH, Pascal DARAGON and Jacques DELORT for being so supportive and understanding through the years. I had such a conducive and challenging environment that helped as an incubator in nurturing and fostering ideas during the past three years. Moreover, I am grateful for the presence of such a charismatic colleague and a friend, Alexis AYACH, who has played a significant role in making this journey a success. Moreover, I had a rare opportunity on Thursday evenings to replenish my mind and body by weekly sporting activities. Thanks to the Trail Running Bees team, especially the “esteemed coach” Stephan LOTA, for those empowering and fun moments. The sessions have instilled a sense of endurance in me. I am grateful.

My sincere thanks to Muhidin Issa MICHUZI, his unrelenting belief and guidance in me made this journey worthwhile. I am indebted to your constant help for the past years. My heartfelt thanks to Goodluck BONGANI, whose inspiration taught me the meaning of perseverance through the years. He has been among a handful of people who believed in me from the start.

Friends and colleagues have been equally instrumental in making my life interesting during my thesis. I cannot mention the names of all friends that have been regularly sharing empowering moments with me. Nevertheless, I would like to convey my deepest gratitude to Sigfrid Blasius NGOWI, Julius MYOMBO, Isatou KEITA, Manyanda CHITIMBO, John MBULI, Boniphace MTAKWA and Andrei SAMBRA. You represent the people who have been so close and so important to enrich the day-to-day life during my PhD journey. Thanks for being there.

For as long as I remember, my family has played such a significant role throughout my studies. Even when I am far, they always maintain regular communication, the distance becomes imaginary and my struggles feel effortless and worthwhile. Thanks to my Mother, Yusta John KIDAWA, who has always taught me to follow the path less traveled. To my brother Rahim SAADAN, you are such a great person with a kind heart. I am blessed to have you as family. Dadia, thanks for being so wonderful and understanding through the years. Your unrelenting courage and unconditional love keeps me going even when the going gets rough. Your presence is priceless.

Last but not least, to everyone who, in one way or another, shared or crossed paths with me during the past wonderful three years, I would like to convey my heartfelt thanks. The long journey is finally coming to a close. Your presence left an indelible mark in my life.

I dedicate this to the four generations of women who have significantly marked my life :
My grandmother, My mother, Dadia and *Carys*.

Contents

Chapter 1

Introduction

Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives.

Father of ubiquitous computing
Mark D. Weiser, 1952 - 1999

THE Internet of Things (IoT) is the inclusion of small and everyday objects and machines to the Internet, a phenomenon predicted to connect around 50 billion small and resource constrained objects to the Internet within the next decade [67]. Figure 1.1 depicts the Internet-of-Things paradigm and some of the objects included. These small and constrained objects are expected to play a major role in the Internet arena, especially in the era of pervasive computing with the idea of connectivity anywhere with anything [89]. The heterogeneous mixture of devices, ranging from wireless sensors to smart home appliances and other devices, that were not previously connected, is set to transform the Internet technology and positively impact our daily lives by bringing new and customized services.

The essence of integrating pervasive devices to the Internet is not limited to the end user electronics but also extends to the industrial devices, objects or machines so as to enrich various business processes, especially for organizations highly dependent on connected technologies within their value chains [85]. This thesis was conducted in one of such industrial contexts where a logistic company, TRAXENS, aims at computerizing the information management and tracking for the container shipping process. More details on the project is provided in section 1.3.

The presence of pervasive devices is expected to push for more decentralized services and constrained devices will need to autonomously grant access to one another without

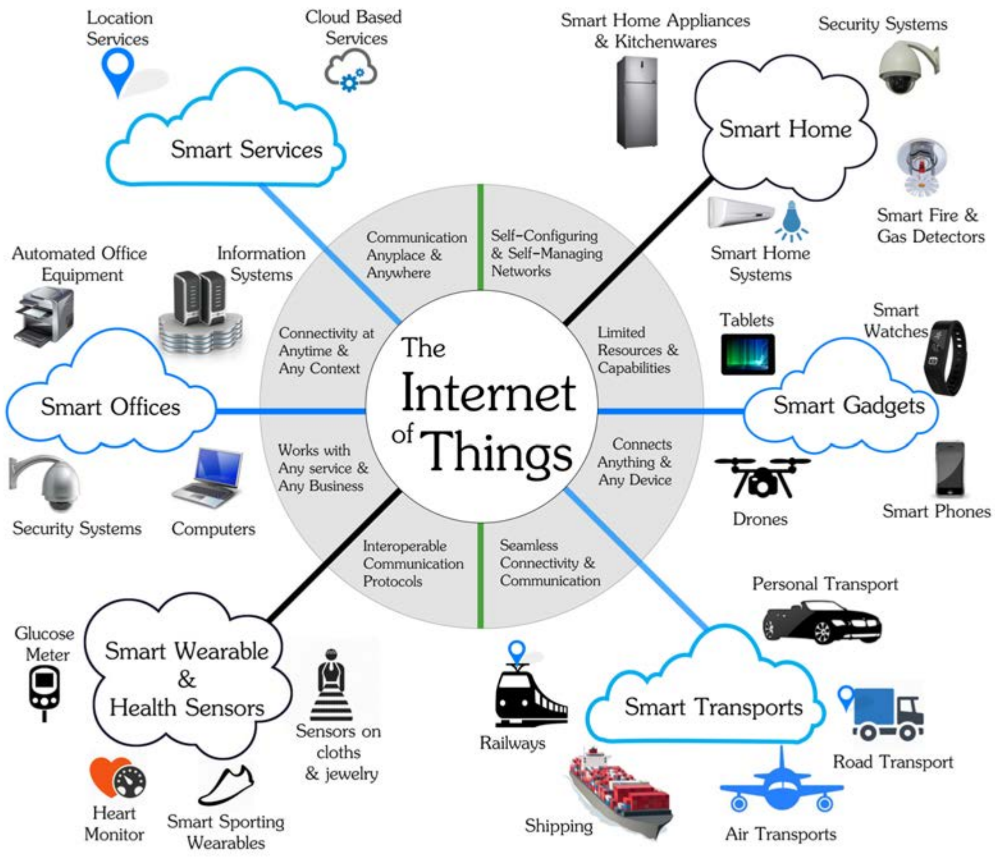

Location Services
Cloud Based Services
Smart Services
Smart Home Appliances & Kitchenwares
Security Systems
Smart Home
Smart Fire & Gas Detectors
Smart Home Systems
Automated Office Equipment
Information Systems
Communication Anyplace & Anywhere
Self-Configuring & Self-Managing Networks
Connectivity at Anytime & Any Context
Limited Resources & Capabilities
The Internet of Things
Tablets
Smart Watches
Smart Offices
Smart Gadgets
Works with Any service & Any Business
Connects Anything & Any Device
Security Systems
Computers
Drones
Smart Phones
Interoperable Communication Protocols
Seamless Connectivity & Communication
Personal Transport
Glucose Meter
Smart Wearable & Health Sensors
Sensors on cloths & jewelry
Railways
Smart Transports
Road Transport
Heart Monitor
Smart Sporting Wearables
Shipping
Air Transports

FIGURE 1.1 - The Internet of Things Paradigm

requiring constant interactions with centralized authorization or authentication servers for acquiring access control, or authentication information, for the services or data it offers. This mode of operation is referred to a "serverless authentication".

The term *serverless authentication* refers to "a mechanism of allowing centrally controlled devices to autonomously authenticate one another, or with other heterogeneous devices, without an active participation of the centralized authentication or authorization servers".

Pervasive computing is "the idea of computing services available anytime, anywhere and on demand, through seamless interactions between heterogeneous players in the Internet" [90]. The majority of pervasive devices have limited resources in terms of energy, computing power and storage capacity, which is key to keeping the technology cost as low as possible and its mass deployment more economical. However, resource limitations in pervasive devices greatly impact their capabilities to run robust security solutions as most of the classical security protocols demand enormous amounts of memory, energy and computation power. Hence, security and privacy are among the most recurring challenges in any resource constrained ecosystem [71]. The terms "Pervasive computing" and "ubiquitous computing" may sometimes be used interchangeably to refer to the same thing [23, 81].

To develop secure pervasive systems, security must be by design i.e. security and privacy aspects must be well researched and analyzed from the beginning of the development process to give the respective technologies a competitive advantage in the market, as security and privacy provided by products within the ecosystem are among the major concerns for end users. The research by Costin et al. [28] suggests that majority of the commercial available pervasive systems, such as VoIP phones, IP/CCTV cameras, are vulnerable to security and privacy attacks. Costin et al. added that these vulnerabilities result from constant pressure to release products to the market and beat competition, which limit the design time and production costs. Eventually, the final products are not thoroughly researched and tested to unveil major threats relevant to their working environments. Over the years, myriad of protocols have been proposed to solve security and privacy issues in the domain of resource constrained devices. Majority of the existing security solutions directly solve issues relevant to pervasive technologies as proposed in [33] and [35]; however, in other cases, the pervasive technologies demands are sufficiently different that new solutions must to be sought [20]. For instance, most of the classic security protocols work in a connected mode, which necessitates a persistent link between a pervasive device and the centralized server, to facilitate authentication or authorization. However, in pervasive computing era, maintaining a persistent connection between geographically distant parties may, at most times, prove futile and costly in terms of bandwidth. Hence, it is paramount to develop a new generation of protocols that render devices more autonomous in offering required services, regardless of the connectivity issues to remote servers.

This thesis focuses on the design of efficient and secure serverless protocols for pervasive devices with strict resource constraints. Serverless protocols optimize secure communication between proximity heterogeneous devices while efficiently using the available resources.

Serverless protocols are found in two major categories: first, serverless authentication protocols, used for establishing trust between communicating parties before securely sharing information, and second, serverless search protocols, used for identifying device(s) fulfilling particular criteria among a group of many. In the real world scenarios, these two kinds of protocols complement each other; whereas serverless authentication protocols are employed for mass authentication of devices at a time, serverless search protocols are used for identifying one device among a group. These protocols have been vastly studied and applied in the domain of Radio-Frequency Identification (RFID) systems [6,51,59,99]. We also use serverless authentication and search principles to solve the challenges in the TRAXENS scenario presented in section 1.3.2. Further analysis of these protocols is given in section 2.5.

1.1 Basic Definitions

This thesis discusses information security; we put forth basic security terminologies that are consistently and repeatedly referred throughout this document. Most ideas proposed in this document refer to *Information security*, which is the protection of information based on the *confidentiality, authorization, trust, integrity, authentication, privacy, non-traceability, non-repudiation*, *anonymity* and *availability.* The term "security" is used in the sense of minimizing the vulnerabilities of assets and resources [83]. Information security is done by uncovering *vulnerabilities* in order to thwart potential *threats* and eventually mitigating *attacks*. The definitions given below refer to the document "Recommendations of security architecture for open systems interconnection for CCITT Applications" [83], unless cited differently. CCITT stands for the International Telegraph and Telephone Consultative Committee.

In information security, *confidentiality* is "the property that information is not made available or disclosed to unauthorized individuals, entities, or processes". The authorized users can be differentiated using credentials, access levels or any other identifying information. *Authorization* is "the granting of rights, which includes the granting of access based on access rights". According to the ITU-T Recommendation X.509 specification [47], an entity can be said to *trust* a second entity when "it (the first entity) makes the assumption that the second entity will behave exactly as the first entity expects". *Integrity* is "the property that data has not been altered or destroyed in an unauthorized manner".

The *Authentication* is "a mechanism intended to ensure the identity of an entity by means of information exchange". The *privacy* is "the right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed". The *non-repudiation* property comes in two forms. *Non-repudiation with proof of origin*, which ensures that "the recipient of data is provided with proof of the origin of data, which protects against any attempt by the sender to falsely deny sending the data or its contents". *Non-repudiation with proof of delivery* where "the sender of data is provided with proof of delivery of data, which protects against any subsequent attempt by the recipient to falsely deny receiving the data or its contents". *Availability* is "the property of being accessible and usable upon demand by an authorized entity".

Anonymity is "the state of being not identifiable within a set of subjects, the anonymity set" [72]. *Non-traceability* is a subset of privacy which means that "no two sessions of the same instance should be able to be linked together".

In the context of information systems, a *threat* is "a potential violation of security". A *vulnerability* is "any weakness that could be exploited to violate a system or the information it contains" [83]. An *adversary* is "a party that attacks the system by uncovering system's vulnerabilities". An *attack* is "the instance or the realization of a potential threat by an adversary in order to gain unauthorized access to the system's services, resources, or information, or an attempt to compromise the integrity of a system" [15, 38].

1.2 Why Serverless Protocols

In this thesis, we research, analyze and propose solutions in the context of serverless protocols. Serverless protocols are ideal solutions for the contemporary and future pervasive technologies. Below are some of the reasons why serverless protocols are the promising solutions for the pervasive computing devices in the era of Internet-of-Things [99]. Refer to section 2.1 for the detailed description of advantages of serverless protocols.

- **Reliability**
  The lack of reliable connectivity between distant objects in the era of pervasive com-

puting is a recurring scenario. The existing classic connection-oriented protocols require constant communication to the centralized servers [45, 99]. Serverless protocols prioritize on proximity communication, hence more reliable, even during network outage.

- **Minimize Energy Consumption**
  Energy is a major concern for many autonomous pervasive devices. It is ideal to minimize energy consumption during communication of information between parties [22]. Serverless protocols prioritize on proximity communication and avoid unnecessary distant communication. The low range or proximity communication is less consuming in terms of energy [64, 93], which extends the lifetime of devices.
- **Improve Efficiency**
  Serverless protocols improve the efficiency of communication by optimizing proximity authentication between autonomous devices for a secure one-to-one exchange. As a result, the authentication process is more efficient with very short response times due to the lack of intermediaries.
- **Provide Resilience**
  Serverless protocols dissociate independent processes by rendering each part autonomous and resilient. In pervasive computing, resilience translates to availability.

1.3 TRAXENS' Container Shipping Scenario

This thesis was realized at TRAXENS S.A.S., a company founded by Michel Fallah in 2012 with the objective of developing logistic information systems services specialized in tracking and monitoring intermodal containers worldwide [4]. In this section, we give an overview of the existing challenges in the shipping industries that TRAXENS aims at tackling and the overview of how to solve these challenges.

1.3.1 Background

Roughly 90% of the world's goods are transported by sea with over 70% as containerized cargo, using approximately 33 million existing containers in circulations. The container tracking solutions are not new, but it is estimated that only 21% of the containers in circulation today are monitored or tracked. Furthermore, it is predicted that by 2020 around 1 billion TEUs will be traded globally [49]. A TEU (twenty-foot equivalent units) measures the cargo capacity of one standard 20-foot (6.1 m) long container.

Despite its fast growth and lucrative nature, most of the support services in freight industry are inadequate to cope with the increasing volume of information and level of sophistication that technology allows. Most of the available services, e.g. tracking and motoring, do not meet the required expectations of the larger audience i.e. end users. As a result, there is a growing need for secure, reliable and automated services that allow end customers, who are the main players in the logistic chains, to follow the state and progress of their merchandises and goods in transit, on timely basis.

Currently, the containerized shipping is fundamental to the global trade and key to the modern globalization [43]. Every year, more than 5,000 container ships transport millions of containers globally, mainly between Asia, Europe and the USA. Despite its significant contribution to the World's economy, the shipping industry still remains closed with little competition, which contributes to a lot of inefficient practices, especially for the services afforded to the end users.

Geersbro et al. [43] note that one of the major problems in shipping industry is the practice of overbooking. It is claimed that shipping companies overbook their vessels by as much as 180%, by putting a security margin to ensure that the vessels are sufficiently utilized, even if some of the cargoes will not show up on time. In case more than 100% of the booked cargo turn up, some containers must be left behind. This leads to delays and extra handling costs to the customers. To avoid such situations, many customers tend to book more capacity than their actual needs and cancel the extra capacity in the last minute. Eventually, this creates a vicious circle of overbooking for the shipping companies as they are forced to overbook even more than necessary.

Consequently, the services rendered to the customers are not satisfactory due to various inconveniences such as late arrivals, wrong invoices, long booking times, few e-booking options, no-transparent pricing, complexity of the handling processes, complicated transit documentation that could run into thousands of pages, to name just a few. Moreover, the shipping lines serving the Asia-Europe routes are mostly unreliable. For instance, 44% of all shipped containers arrive late; 11% with two days delays - and around 8% with delays exceeding eight days [43]. These delays translate to lost money as the values of some shipped products depreciate in quality and value beyond certain dates and times. Sometimes these delays lead to long term damages by tarnishing the reputation of company's image and reducing its brand value and eventually leading to the loss of credibility between trading partners.

Some parts of the logistic chain in shipping industry are automated, but there are still existing loopholes where little or no information is available on the whereabouts (location) and the condition of the goods in the containers (e.g. temperature, humidity, pressure, damage). In some cases, a single container can be managed by more than 15 entities (shippers/consignees, forwarders, haulers, ports, customs, etc.) in the course of a single door-to-door trip. Nevertheless, many of these actors have no information until the container is physically within their reach. In other cases, the available information on the whereabouts of the container can be incorrect due to omissions, errors, accidents, or change in the scheduled plans. Moreover, thefts and cargo damages are also among the major problems costing the industry billions of dollars with little or no information on when and where these incidents occur. Similarly, most manual operations in organizing the logistics of transport, distribution and storage of the millions of unused "dumb" containers contain errors, omissions and fraud.

1.3.2 TRAXENS' Concept

To fulfill the mission of equipping the existing 33 million multimodal containers with intelligent capabilities, TRAXENS innovates technology in three areas i.e., hardware, software and business model, leading to a complete and sophisticated network system consisting of a centralized information platform, "smart" containers and other external actors.

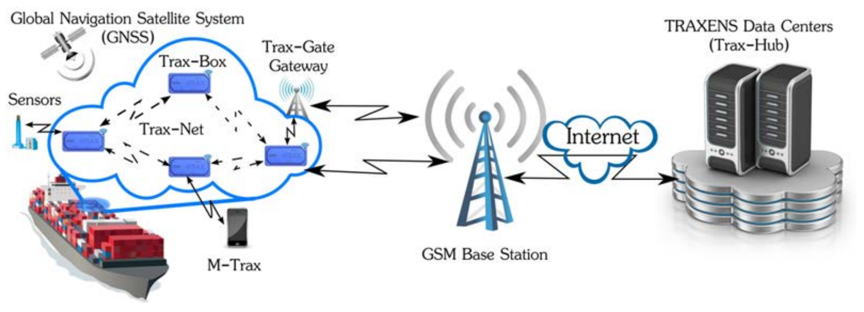

Global Navigation Satellite System (GNSS)
Trax-Box
Trax-Gate Gateway
Sensors
Trax-Net
M-Trax
GSM Base Station
Internet
TRAXENS Data Centers (Trax-Hub)

FIGURE 1.2 - The proposed TRAXENS monitoring system for intermodal containers

The TRAXENS system is based on the idea of "smart" containers, which refers to the containers attached with intelligent devices, named Trax-Box. The Trax-Box contains multiple embedded sensors e.g., temperature, humidity, pressure, RFID (Radio Frequency Identification) sensor tags, capable of monitoring various parameters such as location, safety and status of the carried goods. The data collected are periodically sent to the centralized data center, also known as Trax-Hub, via a mesh network called Trax-Net. The Trax-Gate acts as a relay and forwards information between the Trax-Boxes in Trax-Net and Trax-Hub. The Trax-Box can be solicited with an external party, dubbed M-Trax, which must have authorization from the Trax-Hub before securely communicating with the Trax-Box. The TRAXENS' system is depicted in Figure 1.2.

To improve the reliability and the quality of service, the information collected and communicated between players must be properly secured and only accessible to the right people with the right access rights. Security must be established in all necessary points as the following scenarios depict. However, this thesis focuses on solving the challenges of *scenarios B* and *C* because we are dealing with serverless protocols.

1.3.2.1 Scenario A: *Trax-Boxes and Trax-Hub*

The communication between Trax-Hub and Trax-Boxes is done via a mesh network called Trax-Net. The security between Trax-Hub and Trax-Boxes is treated as end-to-end i.e., only the recipient of the information should be able to decipher the information sent and the intermediary nodes are only used to forward the information between respective points but cannot read the encrypted data. Trax-Boxes are resource constrained and the connectivity to the Trax-Hub is not reliable.

1.3.2.2 Scenario B: *M-Trax and Trax-Boxes with Trax-Hub Support*

This scenario specifically targets one functionality within the Trax-Net:

- Secure serverless mutual authentication between Trax-Boxes and M-Traxes prior to exchanging data

The resource constrained devices, Trax-Boxes, are attached to the containers in transit from one geographic location to another. The Trax-Boxes are used to collect and log crucial information in the course of the trip but also serve as identifiers for the respective containers they are attached to, as shown in Figure 1.3. On transiting borders or checkpoints, controllers possessing mobile personal digital assistants (PDAs), called M-Traxes, must authenticate to the Trax-Boxes before accessing the stored information e.g., the type of goods transported, client (or owner of the goods), appropriate tax information, destination, etc. But, M-Traxes must obtain prior authorization from the remote authorization server i.e., Trax-Hub, before accessing the information in Trax-Boxes. Trax-Hub is only accessible via the network.

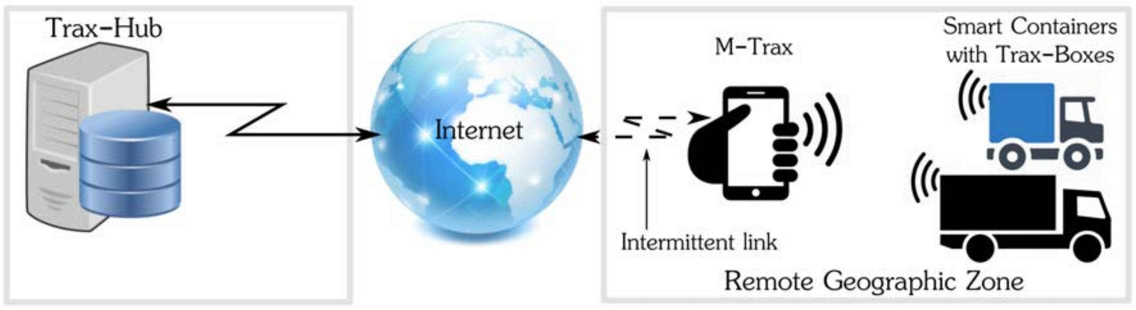


FIGURE 1.3 - Scenario B: Authenticating M-Traxes with Trax-Boxes

Due to the unreliable connectivity to the Trax-Hub, M-Trax should periodically connect to the Trax-Hub, authenticate itself and download the appropriate authentication parameters. The list of authorized Trax-Boxes and other authentication parameters will be stored locally. This process should be done whenever the connection between Trax-Hub and M-Trax is available. In doing so, M-Traxes can autonomously authenticate with Trax-Boxes without the need for a persistent connection to the Trax-Hub. This guarantees reliable authentication between legitimate M-Traxes and Trax-Boxes, even when the connectivity to the Trax-Hub is not reliable.

M-Traxes are treated as external entities i.e., not fully trusted within Trax-Net because they may belong to the external actors who want to access shipping information within the Trax-Net. Moreover, the security parameters granted to the M-Trax must have a defined period of validity, beyond which it should request for new parameters from the Trax-Hub.

1.3.2.3 Scenario C: *Trax-Boxes (or M-Traxes) and Sensors with Trax-Hub Support*

This scenario specifically targets two functionalities within the Trax-Net:

- Secure serverless mutual authentication between Trax-Boxes, or M-Traxes, and sensors
- Searching between Trax-Boxes (or M-Traxes) and sensors

A shipping company has several container yards for storing different types of containers, such as dry, insulated or reefer (a refrigerated container), and their accessories e.g., an RFID container seal or a genset, an electric generator used to power refrigerated containers. The containers in the yard are smart i.e., they are attached with Trax-Boxes and have RFID-embedded seals on the doors. RFID stands for Radio Frequency Identification.

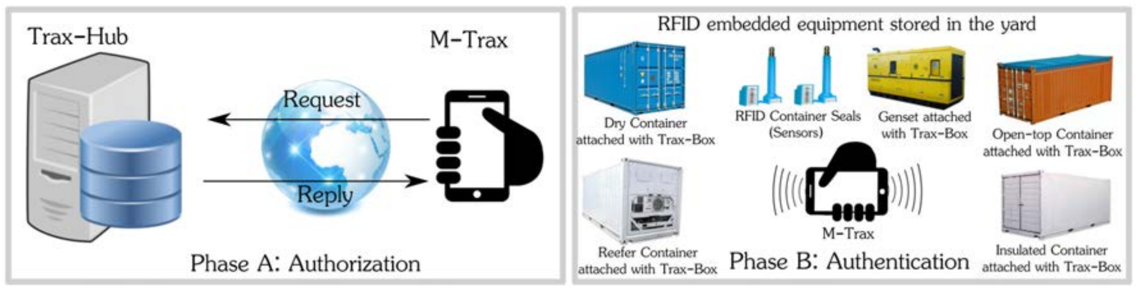

Trax-Hub
M-Trax
Request
Reply
Phase A: Authorization
RFID embedded equipment stored in the yard
Dry Container attached with Trax-Box
RFID Container Seals (Sensors)
Genset attached with Trax-Box
Open-top Container attached with Trax-Box
M-Trax
Reefer Container attached with Trax-Box
Phase B: Authentication
Insulated Container attached with Trax-Box

FIGURE 1.4 - Scenario C: RFID Authentication and Search Protocols

The company wants to setup a secure automated system to localize and acquire information on the containers such as usage, trip history and anti-theft using embedded sensors. Towards realizing the goal, the Trax-Boxes are equipped with all necessary sensors, including RFID reading capabilities. Likewise, the personnel at the yard are equipped with appropriate equipment to communicate with smart containers e.g., M-Traxes with RFID capabilities or RFID-enabled mobile phones in order to facilitate the secure communication with the respective containers and accessories, as depicted in Figure 1.4. This way, every interaction between the personnel and the equipment will be recorded and closely monitored.

The scenario calls for the functionality to easily search for a specific sensor-embedded equipment among a group of equipment, within the proximity. This functionality must be implemented respecting the proper security and privacy criteria granted by the Trax-Hub i.e., only authorized users and devices should be able to use the search functionality.

Due to the unreliable connectivity to the Trax-Hub, the Trax-Boxes, or M-Traxes, should periodically connect to the Trax-Hub, authenticate themselves, download and store the appropriate authentication parameters locally. This process should be done when there is a connectivity to the Trax-Hub. Thus, with the presence of the authentication parameters, Trax-Boxes, or M-Traxes, can autonomously authenticate with the authorized sensors without the need for a persistent connection to the Trax-Hub. This guarantees secure and reliable exchanges between legitimate parties within Trax-Net, even when the connectivity to the Trax-Hub is not reliable.

One of the main security requirement is that, the parameters granted to the Trax-Boxes, or M-Traxes, for authenticating with sensors must have a defined period of validity, beyond which they should request for new parameters from the Trax-Hub.

1.3.3 Advantages of TRAXENS' system

The TRAXENS' system brings the following benefits to the shipping industry:

- Empowering the respective companies to manage stock beyond what is present in their own warehouses by providing near real-time information on the location and condition of the cargo to the respective actors.

- Curb or deter theft of containerized goods by providing early warnings and alerts in case of unusual stops or unplanned container door openings.
- Create a platform to analyze and eliminate accident black spots, smoothing insurance claims for the shipped merchandises and making each actor in the transport chain more responsible for his part. This is done by providing the information on shocks during transit, which can pinpoint when and where accidents happen.
- Improving the efficiency in managing empty containers by ensuring the availability of the right number and type of containers at the right place according to the demand. Currently, thousands of containers are temporarily or permanently lost.
- Improving services while reducing the shipping costs and delivery delays. Eventually, this helps to mitigate the loss of the retail value of the transported goods.

1.4 Problem Statement & Objectives

The main objective of this thesis is to research, design and implement lightweight and efficient security schemes for protecting information generated and shared by resource constrained pervasive devices, such as Trax-Boxes, while taking into account key issues like energy consumption, memory demands, computation demands, efficiency and scalability.

Problem statement:

Pervasive technologies equip common and everyday objects with smart sensors and communication modules to facilitate their integration into the Internet arena. The connected objects generate, collect, store and share information among themselves and also with the centralized or cloud-based servers. Most of the information gathered and shared warrant proper security protection during storage and transfer as it may be of personal, sensitive, or secret nature. However, most of the pervasive devices are low cost with limited resources, which hinder the use of robust classical security solutions for protecting the information stored and exchanged. To properly protect the information in pervasive systems, it is ideal to find suitable security solutions that secure the communication between heterogeneous pervasive devices while making them autonomous. The suitable security solutions must respond to the challenges of the pervasive systems by providing efficient and lightweight security solutions relevant to the devices' resources profiles. One of the best alternative solutions is "Serverless Protocols", which refers to the mechanism of allowing centrally controlled devices to autonomously authenticate one another, or with other heterogeneous devices, without an active participation of the centralized authentication or authorization servers. Serverless protocols respond to the need of the pervasive systems by providing authentication and search protocols, the former is for mass authentication and the latter is for identifying a particular device among a group.

Towards attaining the thesis goals, the following core objectives were identified:

- **Objective 1**: Researching and identifying the characteristics and requirements of serverless protocols.
- **Objective 2**: Designing a serverless mutual authentication protocol between heterogeneous resource constrained devices by taking into account privacy and secure data exchange. This objective focuses on *scenario B* described in section 1.3.2.2.
- **Objective 3**: Designing a secure serverless search protocol for heterogeneous resource constrained pervasive devices. This objective focuses on *scenario C* described in

section 1.3.2.3.

- **Objective 4**: Proposing a guide for developing serverless protocols relevant to resource constrained pervasive devices.
- **Objective 5**: Provision of formal validation proof for the proposed security schemes.
- **Objective 6**: Provision of an informal security and privacy analysis based on some games.

1.5 Contributions

Towards attaining the planned objectives, the following contributions were made:

- A thorough and comprehensive survey on the existing serverless protocols was carried out to analyze their features, requirements and performances relative to the demands of the pervasive era. The synthesis of this survey led us to propose a guide on how to design efficient and secure serverless protocols. This contribution responds to objectives 1 and 4 and is submitted to:
    - Collins MTITA, Maryline LAURENT, and Jacques DELORT, *Designing Efficient & Secure Serverless Protocols*, under submission, 2016.
- The need for a secure lightweight protocol to mutually authenticate between a cluster of resource constrained devices with an external mobile terminal led to the proposal of a serverless lightweight mutual authentication protocol for heterogeneous devices, which answers the challenges in *scenario B* discussed in section 1.3.2.2. In this proposal, an untrusted mobile terminal can request for access from the centralized server to communicate with devices organized in form of clusters. The communication between the terminal and devices is preceded with authentication, both on the cluster level and then on the individual device level. The proposed protocol takes into account two major problems 1) the intermittent connectivity between a distant centralized server and heterogeneous resource constrained pervasive devices and 2) the uneven resource availability in the respective communicating devices. During this contribution we also perform a formal validation using AVISPA tool. This contribution answers objectives 2, 5 and 6 and is published in:
    - Collins MTITA, Maryline LAURENT, and Pascal DARAGON, *Serverless Lightweight Mutual Authentication Protocol for small mobile Computing Devices*, New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on, pp.1-5, 27-29 July 2015.
- One of the major challenges that was to be realized in TRAXENS' system is the secure communication between resource constrained sensors with other heterogeneous devices, as described in *scenarios B* and *C* in sections 1.3.2.2 and 1.3.2.3, respectively. The scenarios call for resource efficiency and scalability as important factors. We find that RFID technology is the best candidate to answer the challenges posed by these scenarios. This led to the proposal of two complementing protocols using RFID as a core technology. The first protocol performs mass authentication between an RFID reader and a group of tags, which answers the challenges in *scenario B*, and the other protocol performs a secure search for a target tag among a group of tags answering challenges in *scenario C* . The contributions include two formal validations;

one is done using the AVISPA tool and the other is done using the CryptoVerif tool. However, these proposals are not confined to RFID technology alone, but can be used to respond to the challenges in any technology with similar challenges. These contributions resolve objectives 2, 3, 5 and 6 and are published in:

- Collins MTITA, Maryline LAURENT, and Jacques DELORT, *Efficient Serverless RFID Mutual Authentication & Secure Tag Search Protocols with Untrusted Readers*, accepted for publication in the IET Information Security Journal, Special Issue on Lightweight and Energy-Efficient Security Solutions for Mobile Computing Devices, April 2016.

1.6 Organization

This dissertation is organized as follows:

Chapter 1 - *Introduction* which includes basic security definitions in Section 1.1, reasons for using serverless protocols in pervasive devices in section 1.2 and the TRAXENS' Container Shipping Scenario in section 1.3. Section 1.4 presents the problem statement & objectives and section 1.5 outlines our contributions.

Chapter 2 - *Security and Privacy in Serverless Protocols* is the state-of-the-art for the serverless protocols. The chapter presents important features for serverless protocols, their threat models, requirements and outlines their advantages before detailing and analyzing the performance, security and privacy properties of the existing serverless authentication protocols.

Chapter 3 - *Authenticating Cluster Devices with Untrusted Parties using a Serverless Paradigm* presents our contribution on a serverless lightweight mutual authentication protocol for heterogeneous devices. The proposed protocol is formally validated using the AVISPA tool. We also provide an informal security and privacy analysis based on games.

Chapter 4 - *Secure Serverless Search & Authentication Protocols for very Constrained Devices* describes another contribution for the RFID scenario where two schemes are obtained. First, the serverless authentication scheme solving the problem of mass authentication between legitimate RFID readers and tags and, second, the search scheme enabling an RFID reader to efficiently and securely search for a specific tag among a huge number of them. This chapter also includes two formal validations; one is done using the AVISPA tool and the other is done using the CryptoVerif tool. We also provide informal validation of the proposed solutions using games.

Chapter 5 - *A Guide on Designing Efficient & Secure Serverless Protocols* presents a guide on how to design secure and efficient serverless protocols for pervasive devices. This chapter highlights important design principles and best practices gathered from the existing serverless security protocols.

Chapter 6 - *Conclusion & Perspectives* provides a summary of the dissertation and gives possible future directions on ensuring adequate security and privacy in the pervasive computing era.

Chapter 2

Security & Privacy in Serverless Protocols

Civilization is the progress toward a society of privacy. The savage's whole existence is public, ruled by the laws of his tribe. Civilization is the process of setting man free from men.

- Ayn Rand

Contents

Security and privacy are paramount for the communication between intelligent devices that generate, store and share sensitive information. The era of pervasive computing pushes the idea of communication and information sharing further by connecting small and specialized resource constrained devices into the Internet [89]. The increasing presence of smart and connected devices running embedded operating systems poses serious privacy and security challenges to both individuals and businesses [28, 85].

In order to fully grasp the security challenges posed by smart pervasive devices to the existing technologies it is imperative to comprehend four important aspects. First, recognizing the strengths and weaknesses of the existing security solutions. Second, identifying the limitations of the existing security solutions towards pervasive devices. Third, unveiling the challenges of designing efficient security solutions for pervasive devices and fourth, building the link between current and future pervasive systems based on secure and lightweight cryptographic primitives.

This chapter presents the state-of-the-art for the security and privacy properties of serverless protocols. The rest of this chapter is organized as follows, section 2.1 elaborates the need for serverless protocols in the pervasive era and section 2.2 puts forth the basic security and privacy threats relevant to the serverless protocols. Section 2.3 describes the basic security and privacy requirements for serverless protocols while section 2.4 gives the basic features of serverless protocols. Section 2.5 reviews the existing serverless schemes by thoroughly analyzing their performance, security and privacy properties. Section 2.6 concludes the chapter.

2.1 Why Serverless Protocols

This section discusses the main reasons why serverless protocols are the ideal solutions for the contemporary and future pervasive technologies.

2.1.1 Unreliable Connectivity

The lack of reliable connectivity between distant objects in the era of pervasive computing is a recurring scenario. Some regions may be isolated due to unavoidable geographical reasons, hence afforded with only intermittent connectivity, and remain off-line for extended periods of time. Classical security protocols require constant communication between distant clients and centralized servers, also referred to as *connected mode*. Pervasive devices running in a connected mode are forced to maintain a persistent connection to the central server during the authentication session. The frequency of communication imposed by these protocols require devices to consume a lot of energy and bandwidth and the authentication process is slow because it is done via a distant server [45, 99].

Alternatively, there is a *serverless mode* of authentication that allows pervasive devices to intermittently connect to the server for acquiring credentials which serve for future authentication [62,99]. Serverless protocols prioritize local communication by allowing devices to securely authenticate and share information without requiring a persistent connection to the centralized server. Serverless protocols are ideal for minimizing resource consumption while optimizing security and privacy of communication between devices located in the same geographic proximity [62]. Table 2.1 summarizes the resource demands for the two modes of authentication.

Table 2.1 - Connectivity & Resource Consumption

Resource	**Connected Mode**	**Serverless Mode**
Bandwidth usage	High	Low
Energy consumption	High	Low
Communication frequency	High	Low
Computation demands	Low	High
Memory & Storage demands	Low	Moderate

2.1.2 Energy Issues

Most constrained devices have limited energy sources. When applying security to pervasive devices, it is interesting to understand the impact the security has on the lifespan of the energy source because it accounts for the proper functioning and lifetime of the device [22]. Communication is an intensely energy demanding operation in constrained devices and can consume up to 50% of the energy spent by the device [101]. For pervasive devices with limited energy sources, the protocol should have very few exchanges for security establishment between two communicating parties to minimize the energy consumption during transmission and reception of messages by respective parties [22]. Serverless protocols greatly reduce unnecessary communication between pervasive devices and centralized servers, which extends the lifetime of energy sources within devices.

2.1.3 Low Power Communication

Maintaining a reliable communication while consuming the least amount of power is the ultimate goal in wireless energy constrained pervasive systems [75]. Pervasive devices are equipped with different communication technologies to enable them conduct both proximity and distant communications with ease. Long distance communications or infrastructure based networks, such as cellular networks, primarily focus on the provision of quality of service (QoS) and bandwidth efficiency; energy conservation is secondary since base stations have unlimited power supply [10]. Such scheme is impractical for most energy constrained pervasive devices as power efficiency has direct influences on the lifetime of the devices.

On the other hand, the low range or proximity communication is less consuming in terms of energy and ideal for most pervasive devices [64, 93]. In some network setups, pervasive devices are organized in such a way that a few of them have direct power connections or their energy sources are regularly replenished. Then, these devices are designated as gateways to relay long distance communication between proximity pervasive devices and distant servers [21, 92]. Eventually, pervasive devices minimize energy consumptions as all communications are treated as local, from the energy perspective.

2.1.4 Autonomy

Autonomy is key to provision of reliable services. In the era of pervasive computing where devices have high mobility, autonomy allows them to offer services to more parties without the need of constantly forwarding the requests to the server. Access control rules must be enforced to allow granting correct access to the right requests within allotted time. Access control can be provided by a centralized authorization server and enforced by the device.

2.1.5 Efficiency

Efficiency can be measured at different levels of processes in communication. Serverless protocols eliminate the phase of regular authentication between pervasive devices and centralized servers while optimizing proximity authentication between autonomous devices for a secure one-to-one exchange. As a result, the authentication process is more efficient with very short response times due to the lack of intermediaries.

2.1.6 Availability

Security is not only limited to addressing authentication but also extends to availability and integrity. Classical security protocols are dependent on the state of the network as well as centralized servers. If any of the two is incapacitated, the whole service is interrupted. On the other hand, serverless protocols dissociate independent processes by rendering each part autonomous and resilient. For instance, a network outage or an incapacitated server (or a group of nodes) affects only a limited number of devices while the majority gracefully continue to offer services. In pervasive computing, resilience translates to availability.

2.2 Threat Models for Serverless Protocols

Security and privacy attacks are common to almost all information systems; however, security and privacy threats are more prevalent to cheap and easily accessible systems, such as pervasive devices. Moreover, attacks and technologies evolve at relatively the same pace [84]. Hence, before we measure how good a security solution is, one needs a clear understanding of the common security threats. In this section we analyze some of the common security threats relevant to serverless protocols. Serverless protocols operate in wireless medium, hence they share most of the threats with protocols operating in wireless environments.

2.2.1 Dolev-Yao Attacks

In Dolev-Yao attack model [34], the attacker controls the network i.e the attacker can overhear, intercept, and synthesize any message in a network. However, the attacker is not capable of breaking the secure cryptographic schemes used in the construction of the protocol. Dolev-Yao model covers the following attacks relevant to the pervasive systems.

2.2.1.1 Jamming

Pervasive devices use wireless medium as a means of communication. Wireless medium is vulnerable to jamming attacks, which occur when noise or powerful signals are introduced on the wireless channel between two communicating devices with the purpose to interference the communication in progress [31]. Jamming attacks may disturb normal communication flow, break the communication session in progress or completely disrupt the protocol.

2.2.1.2 Eavesdropping

Eavesdropping is common in wireless communication as transmitted messages can be intercepted, captured and analyzed by anyone within the range of the communicating devices [109]. If the communication is not well secured, the two devices may unwittingly divulge secret information to the listening adversary.

2.2.1.3 Traceability

Traceability is a breach of privacy of the device or owner [25]. Traceability happens when a device can be identified in a message exchange or when two or more sessions of communication exchanges can be linked to the same device, be it identical or not [58]. Through traceability, an adversary may associate, locate and identify messages and device or owner at any moment, either by analyzing communications or identifying key indicators within a message.

2.2.1.4 De-synchronization

De-synchronization attack is common to protocols that require updating secret parameters during or after authentication process in order to maintain synchronization between communicating devices [40]. In such cases, an adversary may intentionally hinder the delivery of some messages to induce inconsistency between the two communicating parties. De-synchronization leads to the denial-of-service (DoS) attack when communicating devices maintain inconsistent records of secret information and fail to recognize each other in the current or future sessions.

2.2.2 Physical Compromise

The physical attack on a device aims at gaining access to the information stored in it and, if possible, use it against the protocol e.g. attack similar devices using the information from the captured device.

Majority of the pervasive devices are expected to work in open, unprotected or harsh environments i.e., they are inherently prone to physical attacks. To maintain security, even after being physically compromised, pervasive devices must be equipped with anti-tampering mechanisms for storing secret information [103].

However, the majority of the pervasive devices are meant to be cheap, which limits their resource capabilities. The need for additional circuitry to accommodate anti-tampering mechanisms translates to additional costs. Increasing costs hinder pervasive devices from being equipped with secure memory to protect secret information such as encryption keys [29].

2.3 Security & Privacy requirements

An ideal serverless protocol should meet the following minimal security requirements.

2.3.1 Mutual Authentication

Authentication is proving that the device is really what it claims to be [74]. Authentication is important in pervasive computing due to the high mobility and interaction among devices, but it should be mutual. Mutual authentication verifies identities of both parties in the process of communicating, which is indispensable if the goal is to raise the level of trust between communicating parties.

2.3.2 Data Confidentiality

Data confidentiality ensures that the information generated, stored, shared or transmitted by a trusted party, is properly protected and only interpretable or accessible by authorized parties.

2.3.3 Data Integrity

Integrity of information raises the level of trust users give to the system offering services. Only authorized users should have permissions to modify or alter information depending on their respective levels of access. These permissions may include, but are not limited to *write, delete, status change* and *creation.* Integrity applies for the data stored and transmitted alike.

2.3.4 Data Freshness

Data freshness ensures the communicating entities that their interactions are performed in real time, that is, the received messages are answers to their requests [68, 70].

2.3.5 Non-repudiation

Non-repudiation refers to the ability to ensure that a party in a communication cannot deny originating a message [82]. This can also be a two-way property i.e., once the message is successfully delivered, the receiving party should not be able to deny the reception of the message.

2.3.6 Forward Secrecy

Forward secrecy means that the confidentiality of the past secret keys should be maintained even after the long-term keys are compromised in the future [66].

2.3.7 Backward Secrecy

Backward secrecy implies that the confidentiality of the future keys should be preserved even after the disclosure of the secret key [19, 73].

2.3.8 Non-traceability

Non-traceability should ensure that no two different sessions of the same user or device should be linked together. It requires that the messages exchanged between two pervasive devices be semantically indistinguishable by randomizing parameters from one session to the next, and from one device to another [25, 86].

2.3.9 Anti-cloning

Resistance to cloning attacks should ensure that the adversary cannot create copies of legitimate devices through active or passive observation of the communication exchange between legitimate devices [33,103]. Moreover, it should not be feasible for an adversary to use the information contained in a physically compromised devices to create other devices and attack the protocol or legitimate devices [103].

2.4 Serverless Protocol Features

An ideal protocol must fulfill the basic security requirements for each scenario as well as efficiently utilize the resources available in the target devices. In this section we outline fundamental features shared by most serverless protocols.

2.4.1 Resource Constraints

A serverless protocol mediates between heterogeneous devices with disparate resource capabilities. There are four important resources to consider in serverless protocols namely energy, computation, bandwidth and memory space.

Energy is essential for proper functioning of pervasive devices. Pervasive devices can have either *limited* or *unlimited* energy sources. The type of energy source available for a particular device determines the functionality and energy plans for the device. For protocols running on devices with limited energy sources, e.g. battery that cannot be changed or recharged, they should minimize energy consumption by limiting communication and, if necessary, prioritize local computation in order to extend the lifetime of the device. On the other hand, if the device's power source is unlimited, e.g. can be periodically replenished or connected to a reliable power source, then communication cost is not a limitation on the lifetime of the device.

An ideal protocol should minimize energy consumption by avoiding unnecessary computations, implementing optimized operations and most importantly limit communication frequency and message sizes to absolute minimum. There are various ways of reducing the computational load e.g., the protocol may asymmetrically distribute resource demands relative to the capabilities of the participating parties i.e., resource constrained devices should do less computation than their powerful counterparts.

Considering the memory scarcity, the serverless protocol should make efficient use of it e.g., storing minimum essential parameters useful in performing authentication and reducing the frequency of communication. Serverless security protocols must be designed with the idea that security is not a core activity in most pervasive devices and resources, such as storage space or energy, are very limited but highly demanded for other important activities. Thus, each protocol should suit the device's resource profile and share resources with other modules within the device.

2.4.2 Context-Awareness and Management

Classical computing environments are insensitive and therefore cannot make timely, context-sensitive decisions [88]. Context-awareness is an intrinsic characteristic of smart ecosystems, like pervasive computing environments, where various context-sensitive information such as time, location or other temporary factors can be accurately perceived [20].

Once a pervasive system accurately perceives the current context, it can effectively integrate it to aid or improve various security aspects. For instance, diversifying authentication parameters between different sessions by integrating date, time or location within access control parameters. The use of precise and temporary context information improve security by limiting permissions to appropriate moments, locations or situations defined for particular parties.

However, it should be impossible for anyone to predict the actual security aspects based on the context information. In addition, every context information used must be authentic and verifiable by genuine parties involved in the communication. Context information, if properly integrated for security purposes, may provide invaluable data to thwart malicious activities within a system [20].

2.4.3 Intermittent Connectivity

The era of pervasive computing demands a spontaneous connection between anything, everywhere, and in a seamless manner. The challenge arises when some geographic regions are hard to reach and have poor network coverage. The devices found in such areas can only intermittently connect to distant servers for acquiring verification and authentication credentials. An ideal serverless protocol should be generic enough to consider such worst case scenarios so as to allow devices to offer services regardless of the connectivity issues [89].

2.4.4 Localized Security Scalability

Scalability is a desirable feature, especially in pervasive ecosystem, where spontaneous proximity communication is a norm [20]. Contrary to the classic communication paradigm, pervasive devices prioritize local communication between devices over distant ones [89]. In distributed pervasive ecosystems with centralized authorization or authentication servers, the pervasive devices must be capable of autonomously authenticating and securely communicating with other devices, regardless of their number. This ensures efficient seamless access to the services and information. In order to enforce scalability in such setups, the security parameters essential for secure communication should not demand a lot of resources e.g., a lot of computation power to verify or huge amount of storage.

2.4.5 Dynamic Environment

Pervasive environment is highly dynamic with constantly changing actors, some new devices join while others leave [8]. A serverless protocol should embrace such dynamic while enforcing authentication between communicating devices and properly allocating resources relevant to the given access rights.

2.4.6 Multilevel

The security protocol for pervasive devices should be able to offer different levels of security depending on various factors such as context information, environmental situations, system policy, available resources. Likewise, it should be viable for users to change security parameters, whenever necessary, by properly authenticating to the device, to suit their requirements and improve the system's confidence [20].

2.4.7 Access Control

Pervasive computing is characterized by high interoperability and mobility among smart connected devices. Resources and services offered by resource constrained devices need to have appropriate access rights relevant to each request. Authorization to resources or services in such environments requires well defined access control rules to specifically define *who* can access *what*, *when* and *how* within the ecosystem.

It is important to be very careful when it comes to access control as badly defined access control models lead to unnecessary resource consumption or security vulnerabilities by creating loopholes that give adversaries a chance to misuse resources without the owner's consent [8].

If the setup requires fine grained access control rules, then they should be defined without leading to further resource demands from the pervasive devices. For instance, access control rules must be request specific to allow customized resource allocation i.e., the requesting party may ask authorization from the centralized server prior to accessing the respective service, and the granting party has only to enforce what has already been granted by authorization server.

2.4.8 Flexibility and Customizability

Pervasive environment is dynamic and contains heterogeneous players. A serverless protocol designed to work in such environment should be flexible to accommodate players with different resource capabilities and yet be simple enough to add or remove security functionalities depending on the required services [20].

2.5 Analysis of the Existing Serverless Protocols

In the literature, serverless protocols come in two categories, *authentication* and *search* protocols. The communication phases for serverless protocols are depicted in Figure 2.1.

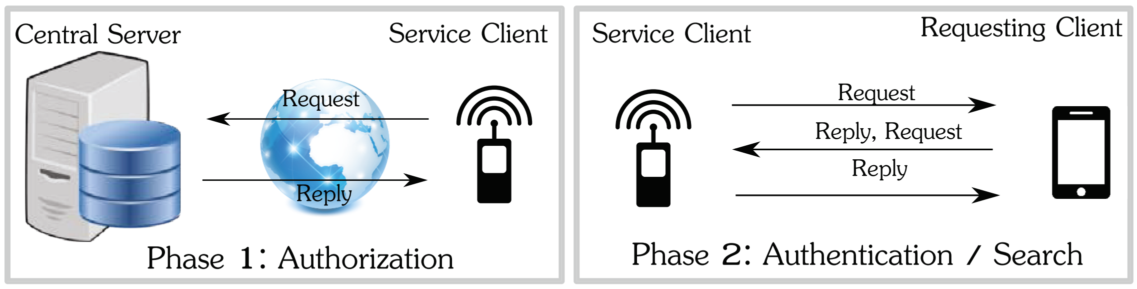


FIGURE 2.1 - Communication phases for serverless protocols

Serverless authentication protocols are used to establish trust between communicating parties before a session of secure information exchange between them. Mass authentication protocols authenticate multiple devices at once. Mutual authentication protocols allow both communicating devices to authenticate one another, while one-sided authentication protocols allow only one device to authenticate another.

Serverless search protocols are used to efficiently and securely identify or locate a device among a group of devices. Contrary to the authentication protocols, search protocols identify and authenticate only one device at a time.

In a practical setup, authentication and search protocols complement each other, hence they are useful in improving the efficiency of any domain requiring such functionalities e.g. inventories, warehouses, supermarkets, etc.

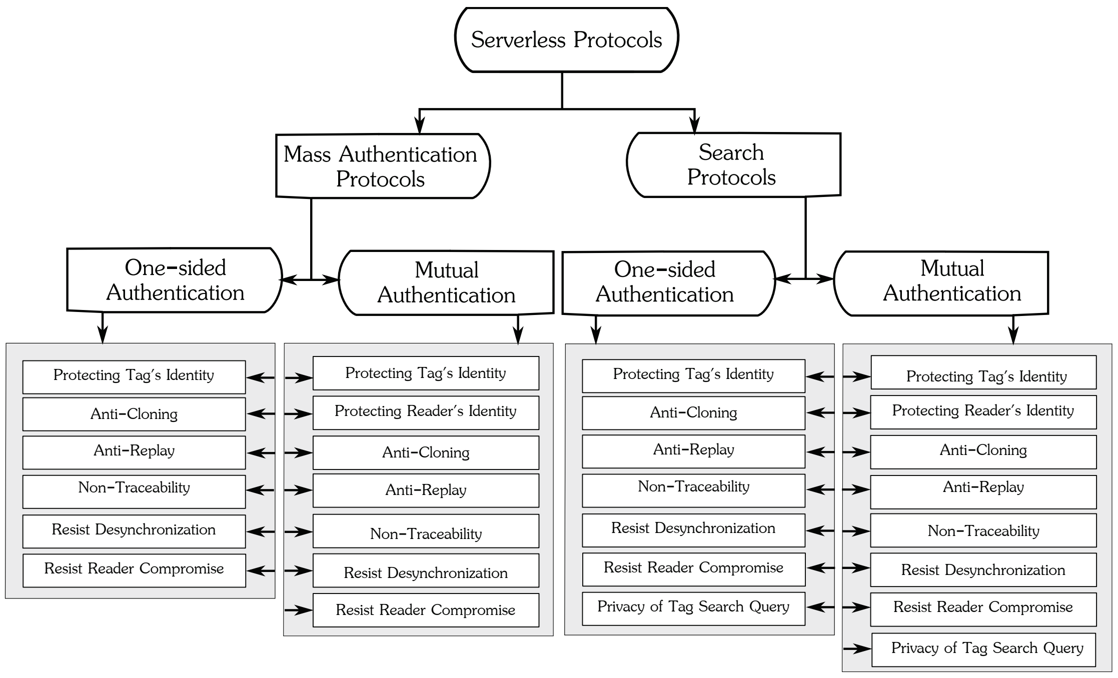


FIGURE 2.2 - Classification of serverless protocols with desirable security properties

The majority of the existing serverless protocols in the state-of-the-art are proposed in the domain of RFID. Hence, the terms backend server, RFID Reader and RFID Tags will be regularly used for the rest of the thesis. Normally, the RFID readers are fixed in one place, hence the leakage of its identifier does not lead to privacy problems. On the contrary, in serverless protocols the RFID readers and tags are mobile, hence leakage of their identifiers will lead to the corresponding breach of the privacy of the person possessing or associated with it.

We analyze the security, privacy and performance properties of the existing serverless protocols relative to the resources available in the respective pervasive devices. Figure 2.2 classifies the existing serverless protocols according to their common properties, including those introduced in section 2.3.

Table 2.2 classifies the existing serverless protocols according to the functionalities they offer. The requirements for search and authentication protocols vary due to the functionalities they provide i.e., authentication protocols broadcast general queries to solicit all tags within its vicinity while search protocols broadcast queries should be understood by only one target tag among a group of tags.

Table 2.2 - Classifying the existing serverless protocols (✓: Supported; ✗: Not Supported)

Group	*Protocol*	*Authentication Protocol*	*Search Protocol*
1	Tan et al. [99]	✓	✓
	Lin et al. [59]	✓	✓
	Lee et al. [57]	✓	✓
	Xie et al. [107]	✗	✓
	Jialiang et al. [51]	✓	✓
	Abdolmaleky et al. [6]	✓	✗
2	Hoque et al. [45]	✓	✓
	Jeon et al. [50]	✗	✓
	Deng et al. [32]	✓	✗
	Pourpouneh et al. [78]	✓	✗
3	Chun et al. [26]	✗	✓
	Ahamed et al. [9]	✓	✗
	Won et al. [106]	✗	✓

From sections 2.5.1 to 2.5.3, we give a thorough analysis of the existing serverless protocols by looking into their security, privacy and performance properties. Most of the recent proposals are based on the preceding ones, either by solving the vulnerabilities or by improving their efficiencies but keeping the basic principles. We group protocols together based on their similarities; we came up with three groups of serverless protocols.

The first group is made up of protocols based on hash functions. Hash functions are conceptually lighter than block ciphers, therefore commonly assumed to be a better choice for resource constrained devices. This group is comprised of Tan et al.'s [99] protocols together with the protocols proposed by Lin et al. [59] described in section 2.5.1.2, Lee et al. [57] described in section 2.5.1.3, Jialiang et al. [51] described in section 2.5.1.4, Xie et al. [107] discussed in section 2.5.1.5 and Abdolmaleky et al. [6] described in section 2.5.1.6.

The second group of protocols is comprised of protocols based on pseudo-random number generator (PRNG) functions. There are different types of PRNG functions. Some PRNGs are based on hash functions, which are robust but costly in terms of computation. The other kind of PRNGs are based on arithmetic calculations or cellular automata, which are lighter but produce randoms with lower quality than those based on hash functions [100]. The protocols in this group are proposed by Hoque et al. [45] described in section 2.5.2.1, Deng et al. [32] described in section 2.5.2.2, Pourpouneh et al. [78] described in section 2.5.2.3 and Jeon et al. [50] discussed in section 2.5.2.4.

The protocols in the third group are based on the classic encryption schemes such as Elliptic Curve Cryptography (ECC) and Advanced Encryption Standard (AES). This group contains protocols proposed by Ahamed et al. [9], Won et al. [106] and Chun et al. [26] described in sections 2.5.3.1, 2.5.3.2 and 2.5.3.3, respectively.

In terms of security, the protocols in group 3 are more robust than those in groups 1 and 2. Likewise, the protocols in group 1 are more robust compared to those in group 2. However, the protocols in group 3 are more consuming in terms of resources such as computation and memory compared to groups 1 and 2. Moreover, the protocols in group 1 demand more resources compared to those in group 2.

Table 2.3 - Notations for Serverless Protocols

Symbol	Description	Symbol	Description
$h(.)$	One-way hash function	K_j	Secret for RFID tag T_j
R_i	RFID reader i	CS	Central Server (Backend Server)
r_i	Identity for R_i	l	Number of bits of hash $h(.)$
$Seed_T$	Secret value between R_i and T_j stored in T_j.	m	A CS defined number of bits of tag's hash i.e. $m < l$. The tag's hash is truncated to m to facilitate the search in the R_i's database
$Seed_{old_j}$	Old secret shared between R_i and T_j stored in R_i.		
$DEC_k[y]$	Symmetric decryption of y with key k	n_j	A random number generated by tag
T_j	RFID tag j	T	A temporary value
id_j	Identity for T_j	K_j	A shared secret key between the tag T_j and CS
$\|\|$	Concatenation operation	X'	Computed value of X
$Seed_{new_j}$	New secret value between R_i and T_j.	L_i	Access list for RFID reader R_i
$P(.)$	Pseudo-random number generator	q	Temporary parameter
$ENC_k[y]$	Symmetric encryption of y with key k	n_i	Random number generated by reader
$ltime$	Last successful authentication time	$ctime$	Current time
$\oplus$	XOR operation	d, p	Variable used in protocol exchange
$A = B$	Overwrite the value of A with that of B	$A == B$	Compare if the values of A and B are equivalent

2.5.1 Protocols based on hash functions

The protocols in this section utilize hash function as the basic building block for their schemes. The protocol constructions may also require other primitives e.g. random number generators, XOR ($\oplus$) or concatenation ($||$).

2.5.1.1 Tan et al.'s Serverless Protocols

To the best of our knowledge, Tan et al. [99] were the first to propose serverless authentication and search protocols in 2007. Tan et al. propose two kinds of complementing protocols, one is the authentication protocol, described in section 2.5.1.1.1 and the other is the search protocol, described in section 2.5.1.1.2. Tan et al.'s [99] protocols aimed at solving two fundamental problems, first, the identification of tags and readers with no persistent connection to central database, and second, the secure tag search function with no key identifying information leakage.

2.5.1.1.1 Tan et al.'s Authentication Protocol

Tan et al.'s [99] authentication protocol performs mass authentication between an RFID reader and numerous tags within the reader's vicinity in two distinct phases namely authorization and authentication. During the authorization phase, an RFID reader R_i communicates with the Central Server CS to download the access list L_i of the authorized tags. During the authentication phase, the reader R_i communicates with the tags it is authorized access to i.e., the tags in the list L_i. These two phases are further explained below. The parameters used in Tan et al.'s protocols are depicted in Table 2.3.

Authorization phase

An RFID reader R_i with a unique identifier r_i authenticates to CS and downloads a list of tags L_i it is allowed access to. CS is a trusted party and controls access between all tags and RFID reader R_i. The communication between CS and R_i is done via a secure channel.
In turn, R_i receives the following access list:

$$L_i = \{(id_1, h(r_i||K_1)), (id_2, h(r_i||K_2)), ..., (id_n, h(r_i||K_n))\} \tag{2.1}$$

Authentication phase

The authentication phase is completed by the exchange of four messages between the reader and the tag as depicted in Figure 2.3.

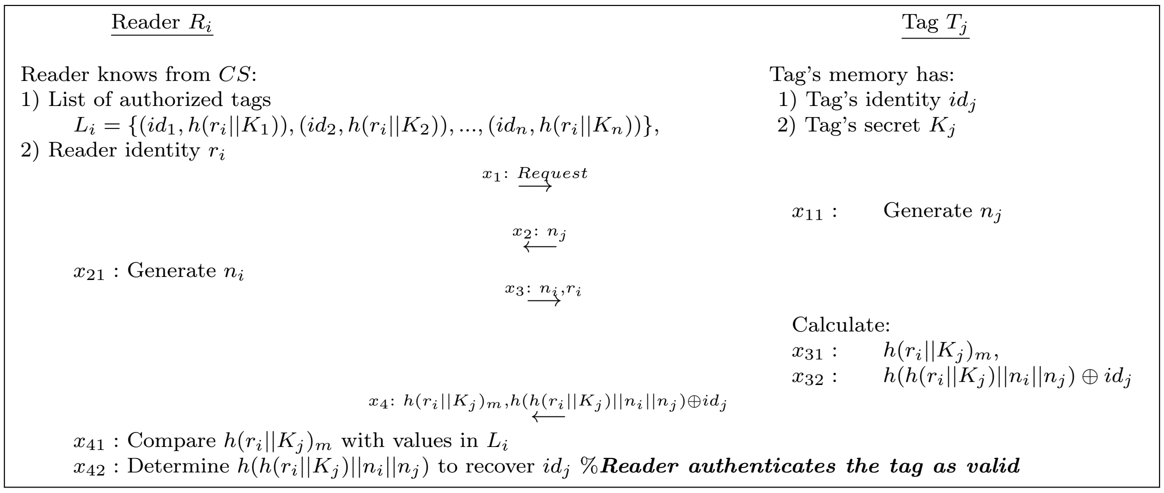

Reader R_i
Tag T_j
Reader knows from CS:
1) List of authorized tags
$L_i = \{(id_1, h(r_i||K_1)), (id_2, h(r_i||K_2)), ..., (id_n, h(r_i||K_n))\}$,
2) Reader identity r_i
Tag's memory has:
1) Tag's identity id_j
2) Tag's secret K_j
x_1: $\xrightarrow{Request}$
x_{11} : Generate n_j
x_2: $\xleftarrow{n_j}$
x_{21} : Generate n_i
x_3: $\xrightarrow{n_i, r_i}$
Calculate:
x_{31} : $h(r_i||K_j)_m$,
x_{32} : $h(h(r_i||K_j)||n_i||n_j) \oplus id_j$
x_4: $\xleftarrow{h(r_i||K_j)_m, h(h(r_i||K_j)||n_i||n_j) \oplus id_j}$
x_{41} : Compare $h(r_i||K_j)_m$ with values in L_i
x_{42} : Determine $h(h(r_i||K_j)||n_i||n_j)$ to recover id_j %***Reader authenticates the tag as valid***

FIGURE 2.3 - Tan et al.'s serverless RFID authentication protocol

2.5.1.1.2 Tan et al.'s Tag Search Protocol

Tan et al.'s [99] tag search protocol is an extension of the authentication protocol described in Section 2.5.1.1.1. In the secure tag search protocol, like in the authentication protocol, the reader must download the list of tags it is authorized to access from CS. When the reader wants to search for a particular tag, it issues a search request such that only a valid tag can understand. The tags receiving the reader's query will also reply in such a manner that only a valid reader can understand. An adversary can still observe all the transactions but cannot make anything useful out of it since he does not know the contents of the query. The search protocol is depicted in Figure 2.4.

2.5.1.1.3 Analysis of Tan et al.'s Serverless Protocols

The performance, security and privacy properties of Tan et al.'s [99] authentication and tag search protocols have been thoroughly analyzed over the years. Numerous views on

Reader R_i | Tag T_j

Reader knows from CS:
1) List of authorized tags
$L = (id_1, h(r_i||K_1)), (id_2, h(r_i||K_2)), ..., (id_n, h(r_i||K_n)),$
2) Reader identity r_i
y_{01} : Generate n_i
Calculate:
y_{02} : $h(h(r_i||K_j)||n_i) \oplus id_j$

Tag's memory has:
1) Tag's identity id_j
2) Tag's secret K_j

$\xrightarrow{y_1:\ h(h(r_i||K_j)||n_i) \oplus id_j, n_i, r_i}$

Calculate:
y_{11} : $h(h(r_i||K_j)||n_i)$
y_{12} : $id^* = h(h(r_i||K_j)||n_i) \oplus (h(h(r_i||K_j)||n_i) \oplus id_j)$
y_{13} : if $(id^* == id_j)\{$%***Authentic query from*** R_i
y_{14} : Generate n_j
y_{15} : Calculate $h(h(r_i||K_j)||n_j) \oplus id_j$

$\xleftarrow{y_2:\ h(h(r_i||K_j)||n_j) \oplus id_j, n_j}$

$\}$ else $\{$ %***Not the correct tag***
y_{16} : Generate n_j with probability λ

$\xleftarrow{y_3:\ n_j}$

$\}$

y_{21} : Verify $h(h(r_i||K_j)||n_j||n_i) \oplus id_j$
%***Reader knows the tag exists***

FIGURE 2.4 - Tan et al.'s serverless RFID search protocol

Tan et al.'s proposals show the need for serverless paradigm to solving the security issues in pervasive systems. The reviews have helped to raise awareness, point out major weaknesses and clarify requirements necessary for developing secure and efficient serverless protocols. In this section, we revisit major weaknesses found in Tan et al.'s protocols.

- *Unilateral authentication*:
  Tan et al.'s authentication and search protocols perform one sided authentication [26] i.e the reader authenticates a tag but the tag does not authenticate the reader. In turn, the tag cannot be certain of the authenticity of the reader as any other entity can pass for a reader and fool the tag. The ideal way is to perform mutual authentication where the tag authenticates the reader and the reader authenticates the tag before sharing secret information.
- *Reader's identity disclosure*:
  Ideally, the identities of the communicating parties must remain secret and can only be divulged to the parties that have a mutual trust i.e., after authentication. Tan et al.'s protocols do not protect reader's identity r_i as it is exchanged in clear, as shown in messages x_3 and y_1 for the authentication protocol and search protocol, respectively. The adversary can easily track and associate all communications corresponding to the given reader r_i with their respective locations. Since the identity of the reader r_i is static, it is possible to leak the location privacy of the user using the reader.
- *Tag's traceability*:
  The non-traceability security property requires that no two different communication sessions should be linked to the same device. Message x_4 of the authentication protocol depicted in Figure 2.3 contains tag's constant value $h(h(r_i||K_j))_m$ for the reader r_i. As the reader's identity r_i is always sent in clear, an adversary can easily derive an

association between tag's reply to the reader identity r_i. This facilitates future traceability whenever the two parties exchange information as the adversary has only to listen and compare the information exchanged to what he has already eavesdropped in the earlier sessions.

- *Tag's identity disclosure*:
  Safkhani et al. [87] provide a detailed analysis on Tan et al.'s [99] protocols on various important security properties such as the privacy of the tag's identity and location. Safkhani et al.'s analysis reveals that it is possible to acquire tag's identity id_j by combining Tan et al.'s authentication and search protocols together [87].
- *Tag impersonation attacks on search protocol*:
  Tan et al.'s secure tag search protocol is vulnerable to the impersonation attack. Following the protocol exchange sequences depicted in Figure 2.4, an adversary can eavesdrop the communication between the tag and the reader and store the message $h(h(r_i||K_j)||n_i) \oplus id_j, n_i, r_i$ from the reader R_i in message y_1. In the future search queries, when R_i sends $h(h(r_i||K_j)||n_i') \oplus id_j, n_i', r_i$ to the tag, an adversary can easily impersonate the tag T_j by replying R_i's message y_2 with $h(h(r_i||K_j)||n_i) \oplus id_j, n_i$. The values $h(h(r_i||K_j)||n_i) \oplus id_j, n_i$ come from the previous R_i's query. As R_i does not compare the values received to its previous queries, the reply is simply accepted as valid and the reader is fooled. Safkhani et al. [87] point out that this vulnerability is caused by the lack of incorporating R_i's random n_i in the tag's response i.e., the tag's response is independent of the reader's query. Another problem is that there is a perfect symmetry between reader's query and tag's response [87], which makes it easy to reply with previous reader's values and impersonate the tag.

2.5.1.2 Lin et al.'s Serverless Protocols

In 2009, Lin et al. [59] proposed their serverless RFID authentication and search protocols. Lin et al. aimed at improving the computational performance of Tan et al.'s protocols. First, it reduces the number of messages from four to three in the exchanges done during the authentication protocol. Second, it reduces the computation on the tag search protocol by eliminating three XOR ($\oplus$) computations i.e., removing two operations on the tag and one on the reader side.

2.5.1.2.1 Analysis of Lin et al.'s Serverless Protocols

In 2012, Lee et al. [57] observed that Lin et al.'s protocols still retain the weaknesses inherent to Tan et al.'s authentication protocols. First, Lin et al.'s protocols are vulnerable to the impersonation attack [57]. Second, like Tan et al.'s authentication protocols, Lin et al.'s protocols also perform one sided authentication i.e., the reader authenticates the tag, but the tag does not authenticate the reader.

Morever, Lin et al.'s protocol does not protect reader's privacy and is vulnerable to the replay and impersonation attacks [57,107]. In their article, Lin et al.'s [59] insist that their protocols retain the security and privacy levels of the original Tan et al.'s protocols. It implies that their proposals are also vulnerable to all attacks described in section 2.5.1.1.3.

2.5.1.3 Lee et al.'s Serverless Protocols

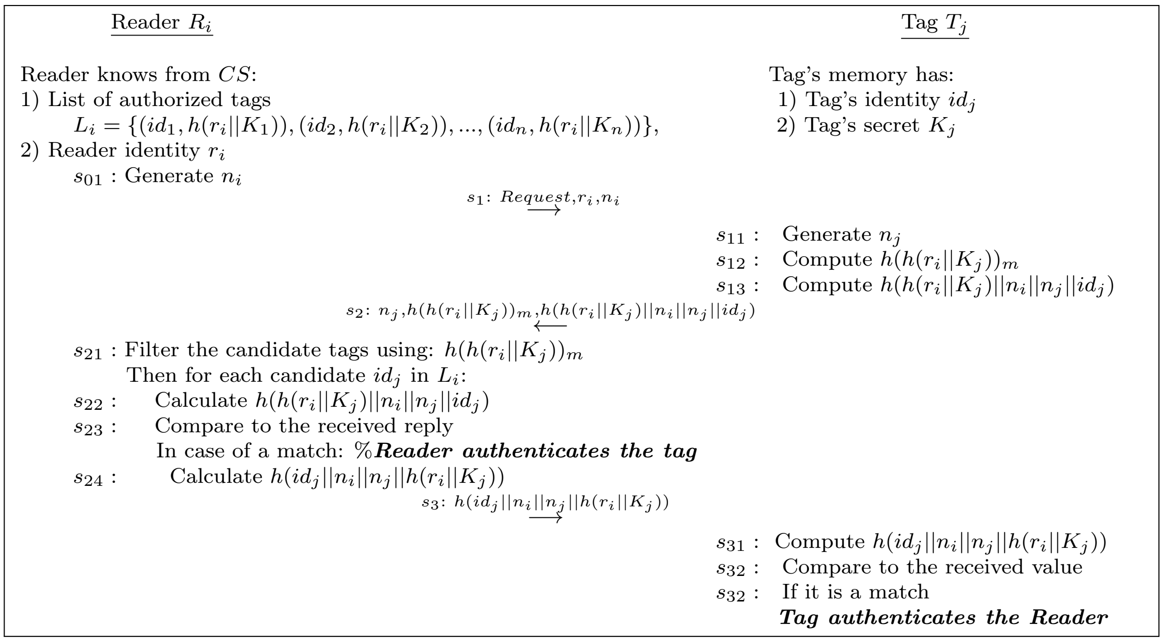
In 2012, Lee et al. [57] proposed two complementing serverless protocols. One is the mutual authentication protocol and the other is the tag search protocol. These protocols are expected to achieve mutual authentication, privacy or anonymity, untraceability and resistance to the denial-of-service (DoS) attacks.

2.5.1.3.1 Lee et al.'s Authentication Protocol

Before launching the serverless RFID authentication protocol, an RFID reader R_i authenticates itself to the Central Server CS, to download the access list L_i. The list L_i contains credentials for each tag T_j with identity id_j that the reader R_i is authorized access to. The reader is given a reader-specific key $h(r_i||K_j)$.

$$L_i = \{(id_1, h(r_i||K_1)), (id_2, h(r_i||K_2)), ..., (id_n, h(r_i||K_n))\} \tag{2.2}$$

Lee et al.'s serverless authentication protocol between R_i and T_j is depicted in Figure 2.5.



Reader R_i

Reader knows from CS:
1) List of authorized tags
$L_i = \{(id_1, h(r_i||K_1)), (id_2, h(r_i||K_2)), ..., (id_n, h(r_i||K_n))\}$,
2) Reader identity r_i
s_{01} : Generate n_i

$\xrightarrow{s_1:\ Request, r_i, n_i}$

Tag T_j

Tag's memory has:
1) Tag's identity id_j
2) Tag's secret K_j

s_{11} : Generate n_j
s_{12} : Compute $h(h(r_i||K_j))_m$
s_{13} : Compute $h(h(r_i||K_j)||n_i||n_j||id_j)$

$\xleftarrow{s_2:\ n_j, h(h(r_i||K_j))_m, h(h(r_i||K_j)||n_i||n_j||id_j)}$

s_{21} : Filter the candidate tags using: $h(h(r_i||K_j))_m$
Then for each candidate id_j in L_i:
s_{22} : Calculate $h(h(r_i||K_j)||n_i||n_j||id_j)$
s_{23} : Compare to the received reply
In case of a match: %***Reader authenticates the tag***
s_{24} : Calculate $h(id_j||n_i||n_j||h(r_i||K_j))$

$\xrightarrow{s_3:\ h(id_j||n_i||n_j||h(r_i||K_j))}$

s_{31} : Compute $h(id_j||n_i||n_j||h(r_i||K_j))$
s_{32} : Compare to the received value
s_{32} : If it is a match
Tag authenticates the Reader

FIGURE 2.5 - Lee et al.'s serverless RFID authentication protocol

2.5.1.3.2 Lee et al.'s Tag Search Protocol

Lee et al.'s search protocol requires the list of authorized tags to search for. The first phase of the search protocol is similar to that described in section 2.5.1.3.1, where the reader R_i connects to the Central Server CS to download the list L_i of tags it is allowed access to. The tag search phase is depicted in Figure 2.6.

Reader R_i		Tag T_j
Reader knows from CS: 1) List of authorized tags $L_i = \{(id_1, h(r_i\|\|K_1)), (id_2, h(r_i\|\|K_2)), ..., (id_n, h(r_i\|\|K_n))\}$, 2) Reader identity r_i c_{01} : Generate n_i c_{02} : Calculate $h(K_j\|\|n_i\|\|r_i) \oplus id_j$		Tag's memory has: 1) Tag's identity id_j 2) Tag's secret K_j
	c_1: *Broadcast* r_i, n_i, $h(K_j\|\|n_i\|\|r_i) \oplus id_j$ $\longrightarrow$	
		c_{11} : Compute $h(r_i\|\|K_j\|\|n_i)$ c_{12} : Recover $id^* = h(r_i\|\|K_j\|\|n_i) \oplus (h(K_j\|\|n_i\|\|r_i) \oplus id_j)$ if $(id_j == id^*)$ %***The right Tag being searched*** c_{13} : Generate n_j c_{14} : Compute $h(K_j\|\|id_j\|\|n_i\|\|n_j\|\|r_i)$ else c_{15} : Generate n_j with probability λ
	c_2: n_j,$h(K_j\|\|id_j\|\|n_i\|\|n_j\|\|r_i)$ *or* n_j $\longleftarrow$	
c_{21} : Calculate $h(K_j\|\|id_j\|\|n_i\|\|n_j\|\|r_i)$ and compare to the received value If the received value is valid c_{22} : Calculate $h(r_i\|\|id_j\|\|n_i\|\|n_j\|\|K_j)$ %***Authenticate the Tag*** else c_{23} : Generate n_i		
	c_3: $h(r_i\|\|id_j\|\|n_i\|\|n_j\|\|K_j)$ *or* n_i $\longrightarrow$	
		c_{31} : Compute $h(r_i\|\|id_j\|\|n_i\|\|n_j\|\|K_j)$ c_{32} : Verify with the received value %***Authenticate*** R_i

FIGURE 2.6 - Lee et al.'s serverless search protocol

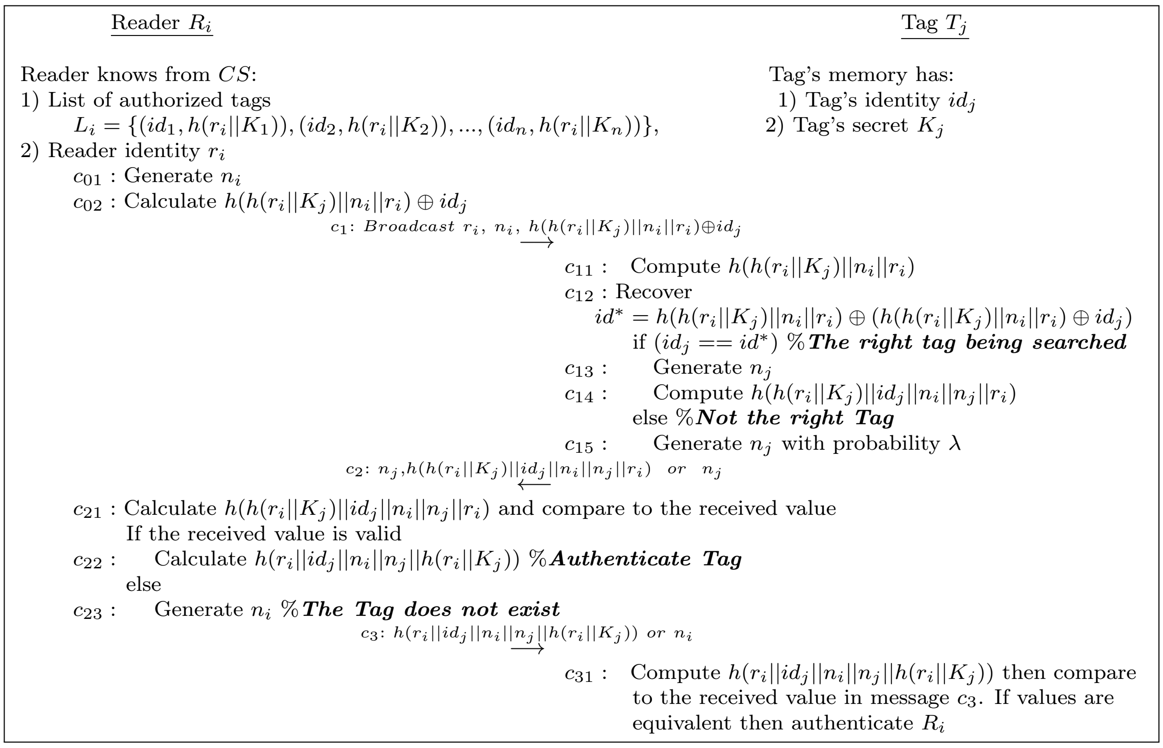
2.5.1.3.3 Analysis of Lee et al.'s Protocols

Lee et al.'s [57] and Tan et al.'s [99] protocol (refer to section 2.5.1.1) share similar features, hence suffer from the same vulnerabilities. Lee et al.'s protocol broadcasts the identity of the reader as cleartext during the authentication and search phases. This renders the reader traceable in all communications. The traceability of the reader allows the traceability of the person who carries the respective reader.

In 2014, Jialiang et al.'s [51] observed that Lee et al.'s protocol is vulnerable to the tag traceability attack. The tag always responds to the reader with a constant value $h(h(r_i||K_j))_m$ in message s_2 (refer to Figure 2.5). This leads to the traceability of the user attached to the reader.

In our analysis we also found that the list L_i in Lee et al.'s protocol contains constant authentication parameters which do not expire. This implies that the reader is only authorized once and the parameters remain valid forever. Moreover, once the reader is compromised, the parameters cannot be revoked, hence tags can be accessed by adversaries without any remedies to the problem.

Moreover, after analyzing Lee et al.'s search protocol, presented in Figure 2.6, we find out that it is incorrect because the protocol construction directly makes use of the secret key K_j, which is known only by CS and T_j but not R_i. The key K_j has been used in steps $c_{02}, c_{14}, c_2, c_{21}, c_{22}, c_3$ and c_{31}. This implies that, the protocol cannot run as proposed. That is why, we propose a correct revised version as depicted in Figure 2.7.



Reader R_i | Tag T_j

Reader knows from CS:
1) List of authorized tags
$L_i = \{(id_1, h(r_i||K_1)), (id_2, h(r_i||K_2)), ..., (id_n, h(r_i||K_n))\}$,
2) Reader identity r_i
c_{01} : Generate n_i
c_{02} : Calculate $h(h(r_i||K_j)||n_i||r_i) \oplus id_j$

Tag's memory has:
1) Tag's identity id_j
2) Tag's secret K_j

$\xrightarrow{c_1\text{: Broadcast } r_i,\ n_i,\ h(h(r_i||K_j)||n_i||r_i)\oplus id_j}$

c_{11} : Compute $h(h(r_i||K_j)||n_i||r_i)$
c_{12} : Recover
$id^* = h(h(r_i||K_j)||n_i||r_i) \oplus (h(h(r_i||K_j)||n_i||r_i) \oplus id_j)$
if $(id_j == id^*)$ %***The right tag being searched***
c_{13} : Generate n_j
c_{14} : Compute $h(h(r_i||K_j)||id_j||n_i||n_j||r_i)$
else %***Not the right Tag***
c_{15} : Generate n_j with probability λ

$\xleftarrow{c_2\text{: } n_j, h(h(r_i||K_j)||id_j||n_i||n_j||r_i) \text{ or } n_j}$

c_{21} : Calculate $h(h(r_i||K_j)||id_j||n_i||n_j||r_i)$ and compare to the received value
If the received value is valid
c_{22} : Calculate $h(r_i||id_j||n_i||n_j||h(r_i||K_j))$ %***Authenticate Tag***
else
c_{23} : Generate n_i %***The Tag does not exist***

$\xrightarrow{c_3\text{: } h(r_i||id_j||n_i||n_j||h(r_i||K_j)) \text{ or } n_i}$

c_{31} : Compute $h(r_i||id_j||n_i||n_j||h(r_i||K_j))$ then compare to the received value in message c_3. If values are equivalent then authenticate R_i

FIGURE 2.7 - Our revised version of Lee et al.'s serverless search protocol

2.5.1.4 Jialiang et al.'s Serverless Protocols

In 2014, Jialiang et al. [51] proposed secure and private protocols for serverless RFID systems. The proposal contains two protocols, one for mutually authenticating RFID readers and tags; and the other for securely searching for a particular tag among a group of tags. We studied these protocols and found various vulnerabilities that need to be addressed.

In this section, we analyze the security and privacy properties of Jialiang et al.'s protocols to unveil some of its fundamental problems. We start by presenting Jialiang et al.'s serverless authentication protocol in section 2.5.1.4.1, followed by a Secure Tag Search Protocol in section 2.5.1.4.2. Finally, we provide a general analysis of Jialiang et al.'s protocols in section 2.5.1.4.3.

2.5.1.4.1 Jialiang et al.'s Authentication Protocol

Jialiang et al.'s [51] proposed a serverless authentication protocol to solve the vulnerabilities in Tan et al.'s [99] and Lee et al.'s [57] protocols. Jialiang et al.'s [51] mutual authentication protocol was built for low-cost passive tags with strict resource constraints. Their protocol is depicted in Figure 2.8.

Authorization Phase

The initialization phase is a communication between the reader R_i and CS. During this phase, the reader R_i connects and identifies itself to the CS to acquire the access list $L_i = \{(id_1, h(r_i, K_1))...(id_n, h(r_i, K_n))\}$ containing the list of tags that the reader is allowed to authenticate. The list contains the real tag's identifier id_j and a secret value $h(r_i, K_j)$ to access the tag, which is a hash of reader's identity together with the tag's key. With the list in memory, the reader R_i can authenticate tags within its vicinity.

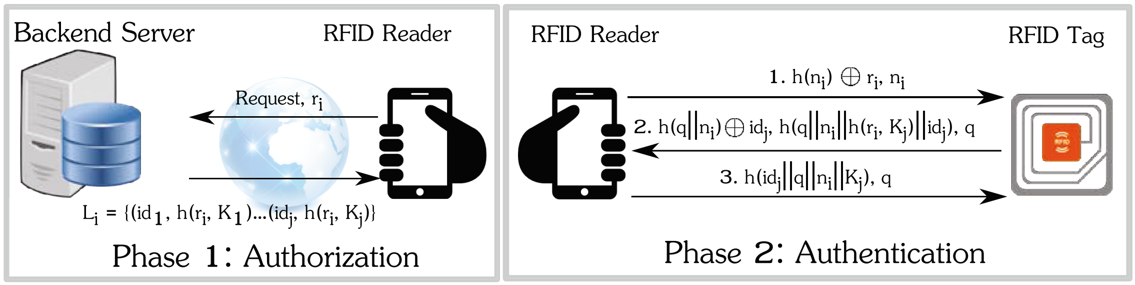

Backend Server
RFID Reader
Request, r_i
L_i = {(id$_1$, h(r_i, K_1)...(id$_j$, h(r_i, K_j)}
Phase 1: Authorization
RFID Reader
RFID Tag
1. h(n_i) ⊕ r_i, n_i
2. h(q||n_i) ⊕ id$_j$, h(q||n_i||h(r_i, K_j)||id$_j$), q
3. h(id$_j$||q||n_i||K_j), q
Phase 2: Authentication

FIGURE 2.8 - Jialiang et al.'s Serverless Mutual Authentication Protocol

Authentication Phase

The mutual authentication phase is an exchange between the Reader R_i and the tag or group of tags within its vicinity to establish secure communication between them. The mutual authentication phase between the tag and the reader goes through the following steps.

- The reader R_i initiates the authentication process by generating and broadcasting the random number n_i and $h(n_i) \oplus r_i$ to the tags in the vicinity.

$$R_i \longrightarrow Tag : h(n_i) \oplus r_i, n_i \tag{2.3}$$

- Upon receiving the message from the Reader, the tag recovers r_i, generates $q = h(T \oplus n_i \oplus id_j), h(q||n_i) \oplus id_j$ and $h(q||n_i||h(r_i||K_j)||id_j)$ then replies to reader with values as shown in line 2.8. The tag subsequently updates a temporary value $T = q \oplus h(q||n_i||K_j||id_j)$. The value T serves to randomize the values of q during each interrogation with R_i.

$$\begin{aligned} Tag \quad &: \quad Recover\ r_i = h(n_i) \oplus (h(n_i) \oplus r_i), && (2.4)\\ &: \quad Let\ q = h(T \oplus n_i \oplus id_j), && (2.5)\\ &: \quad h(q||n_i) \oplus id_j && (2.6)\\ &: \quad h(q||n_i||h(r_i||K_j)||id_j) && (2.7) \end{aligned}$$

$$R_i \longleftarrow Tag \quad : \quad h(q||n_i) \oplus id_j, h(q||n_i||h(r_i||K_j)||id_j), q \tag{2.8}$$

- Upon receiving tag's response, the reader calculates $id = h(q||n_i) \oplus (h(q||n_i) \oplus id_j)$, and searches for a matching value in the list L_i. In case of a match, the reader authenticates the tag and sends back the value $h(id_j||q||n_i||K_j)$ to the tag. Otherwise, the authentication process halts.

$$R_i \longrightarrow Tag \quad : \quad h(id_j||q||n_i||K_j) \tag{2.9}$$

- When the tag id_j receives reader's response, it calculates $H(id_j||q||n_i||K_j)$ and compares it to the received value. If the values match, the tag authenticates the reader to complete the mutual authentication process.

2.5.1.4.2 Jialiang et al.'s Tag Search Protocol

Jialiang et al.'s [51] tag search protocol was developed to solve the problem of RFID tag search within a group of tags. The protocol was also meant to solve the vulnerabilities found in the Tan et al.'s [99] and Lee et al.'s [57] search protocols. Jialiang et al.'s [51] tag search protocol complements their mutual authentication protocol, discussed in section 2.5.1.4.1 and share most of the features. Similar to their mutual authentication protocol, the tag search protocol is also done in two steps, namely initialization phase and search phase as depicted in Figure 2.9. As the initialization phase is similar to that presented in section 2.5.1.4.1, we are only presenting the tag search phase in section 2.5.1.4.2.

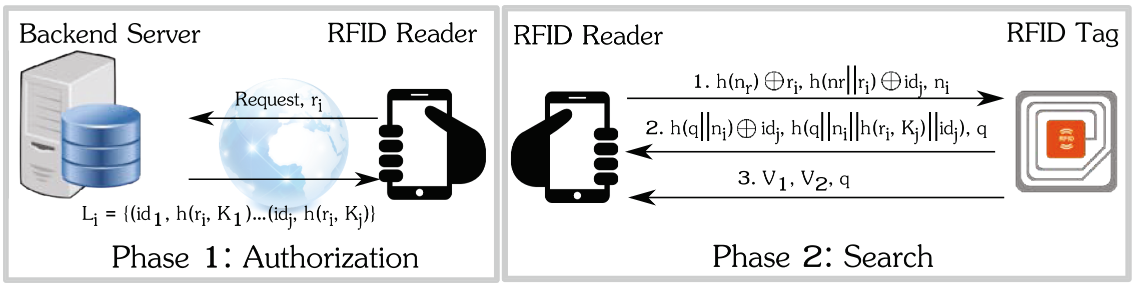

Backend Server
RFID Reader
Request, r_i
$L_i = \{(id_1, h(r_i, K_1)...(id_j, h(r_i, K_j)\}$
Phase 1: Authorization
RFID Reader
RFID Tag
1. $h(n_r) \oplus r_i$, $h(nr||r_i) \oplus id_j$, n_i
2. $h(q||n_i) \oplus id_j$, $h(q||n_i||h(r_i, K_j)||id_j)$, q
3. V_1, V_2, q
Phase 2: Search

FIGURE 2.9 - Jialiang et al.'s Secure Search Protocol

Tag Search Phase

The tag search is done when the reader wants to search for a specific tag within a group of tags. The identity of the tag to be searched must be known in advance. The tag search protocol is performed with the following steps.

- The reader R_i wants to search a specific tag with identity id_j. It starts by generating a random value n_i, and then calculates $h(n_i) \oplus r_j$ and $h(n_i||r_i) \oplus id_j$. Finally the reader broadcasts $h(n_i) \oplus r_i, h(n_i||r_i) \oplus id_j$, n_i to all tags within its neighborhood.

$$R_i \longrightarrow Tag^* \quad : \quad h(n_i) \oplus r_i, h(n_i||r_i) \oplus id_j, n_i \tag{2.10}$$

- The tags near the reader receive a query then recover r_i by calculating $h(n_i) \oplus (h(n_i) \oplus r_i)$. Using r_i the tag recovers the value of id* using the value $h(n_i||r_i) \oplus id_j$. If

$id^* == id_j$, then the tag goes to line 2.13. Otherwise, the tag does not correspond, hence it goes to line 2.17.

$$Tag^* \quad : \quad r_i = h(n_i) \oplus (h(n_i) \oplus r_i), \tag{2.11}$$
$$: \quad id^* = (h(n_i||r_i) \oplus id_j) \oplus h(n_i||r_i) \tag{2.12}$$

- The correct calculates $q = h(T \oplus n_i \oplus id_j), h(q||n_i) \oplus id_j, h(q||n_i||h(r_i||K_j)||id_j)$, and sends $h(q||n_i) \oplus id_j, h(q||n_i||h(r_i||K_j)||id_j), q$ back to the reader. The tag must simultaneously update the value $T = q \oplus h(q||n_i||K_j||id_j)$, to help future randomization of q.

$$Tag^* \quad : \quad q = h(T \oplus n_i \oplus id_j), \tag{2.13}$$
$$: \quad h(q||n_i) \oplus id_j, \tag{2.14}$$
$$: \quad h(q||n_i||h(r_i||K_j)||id_j) \tag{2.15}$$

$$R_i \longleftarrow Tag^* \quad : \quad id^* = h(q||n_i) \oplus id_j, h(q||n_i||h(r_i||K_j)||id_j), q \tag{2.16}$$

- After receiving the response from the tag, the reader recovers $id = h(q||n_i) \oplus (h(q||n_i) \oplus id_j)$ and searches for a matching value within a list L_i. If there is a match, the reader authenticates a tag and proves its presence.
- In order to confuse the attacker, the rest of the tags in the neighborhood that do not correspond to the searched tag id_j respond to the reader by sending two random numbers V_1, V_2 back to the reader R_i as shown in line 2.17.

$$R_i \longleftarrow Tag^* \quad : \quad V_1, V_2, q \tag{2.17}$$

2.5.1.4.3 Cryptanalysis of Jialiang et al.'s Protocols

This section analyzes the security and privacy properties of Jialiang et al.'s [51] protocols as presented in sections 2.5.1.4.1 and 2.5.1.4.2. We analyze their protocols based on the security and privacy standards as well as by examining if the protocols fulfill the requirements that they were built to achieve. As the two protocols are almost similar, here are some of the vulnerabilities that were found in the two protocols.

Reader's traceability problem

Jialiang and al. [51] aimed at solving the reader's traceability problem for which Tan et al.'s [99] protocol is vulnerable, but their protocol still suffers from the same problem. In their protocol construction, Jialiang et al. [51] assume that the value generated by the hash function $h()$ is secret. However, this is contrary to the Kerckhoffs' basic security principles which states that *a cryptosystem should be secure even if everything about the system, except the key, is public knowledge* [54, 95]. It follows that, all functions and algorithms used in any security protocol are public knowledge, hence accessible and usable by adversaries. The step 2.3 mutual of authentication protocol and step 2.10 of tag search protocol expose the reader's unique index, i.e., identifier, r_i. The adversary can easily obtain the value by simply performing the same operation as the tag because the value n_i is public i.e., the adversary obtains r_i by calculating $h(n_i)$ then recovering $r_i = h(n_i) \oplus (h(n_i) \oplus r_i)$.

Tag's traceability problem

Jialiang and al. [51] aimed at solving Tan et al.'s [99] tag's traceability problem but they augmented the privacy problem in their protocols by making tag's traceability easier. In step 2.8, the tag sends $h(q||n_i) \oplus id_j, h(q||n_i||h(r_i||K_j)||id_j)$ and q to the reader. Since the parameters q, n_i and hash function $h()$ are all public knowledge, it is easy for an adversary to recover id_j by calculating $h(q||n_i)$ then performing $id_j = h(q||n_i) \oplus (h(q||n_i) \oplus id_j)$. The identity of the tag authenticating to the reader can be easily traced.

Inefficient operations

The protocol performs unnecessary computations, costing energy and time. In step 2.3, the reader hides its identity r_i by calculating $h(n_i) \oplus r_i$ and, in step 2.8, the tag hides its identity id_j by calculating $h(q||n_i) \oplus id_j$. However, these computations are neither necessary nor valuable to enhance protocol's security. It would be more efficient to send these values in clear to save resources e.g. time and computation.

Incoherent operations

In step 2.9 of Jialiang's authentication protocol, the reader sends the value $h(id_j||q||n_i||K_j)$ to the tag. However, the reader does not know the value K_j, which makes this equation incoherent with the assumptions put forth in their proposal. This step may be corrected by replacing K_j with $h(r_i, K_j)$ and be written as $h(id_j||q||n_i||h(r_i, K_j))$.

Tag's vulnerability Issues

During the initialization phase, as presented in section 2.5.1.4.1, the reader R_i connects to the CS to download the list L_i for the tags it is allowed to access. The list L_i contains a couple of tag values i.e., tag's real identity id_j and the corresponding hash bound to the particular reader $h(r_i||K_j)$. The secret values $h(r_i||K_j)$ lack context, e.g. time, random, or a temporary value. As a result, these values remain constant and valid for a particular reader. This implies that, once the credentials are given to the reader R_i they can be reused for the entire lifetime of the reader and tags.

The use of real identities in the list L_i gives too much power to the reader as well as adversaries once they compromise the reader. The lack of context information makes it difficult to revoke the reader's credentials, even in the case of serious privacy breaches e.g. when the reader is compromised because an adversary can continue to use the parameters within the captured reader to authenticate tags at any time without raising any problems to the system. With the protocol setup, there are only two options to avoid further problems once the respective readers are stolen or their values acquired by an external entities. First, killing or voiding all tags to make it impossible for the stolen values to be used. Second, to change the values for all of them, that is, change the identity id_j for each tag and their corresponding keys. These two options are difficult and lead to destruction, which could be avoided by the use of revocable parameters.

2.5.1.5 Xie et al.'s Serverless Search Protocol

In 2014, Xie et al. [107] proposed a secure tag search protocol in their article, RFID seeking: Finding a lost tag rather than only detecting its missing, which is secure against common attacks such as replay, traceability and DoS [107].

Xie et al.'s protocol was later improved by Jeon et al. in 2014 with few modifications. As these two protocols are relatively similar, we discuss Jeon et al.'s protocol in section 2.5.2.4.

2.5.1.5.1 Analysis of Xie et al.'s Search Protocol

Like the majority of the existing serverless protocols, Xie et al.'s [107] search protocol also provides static authentication parameters in list L_i from CS, without binding any context information with it. In turn, the reader is not required to request for another authorization once it has the credentials for the given list of tags. Moreover, the adversary can still use the parameters in the list to communicate with the respective tags after compromising the reader. This attack cannot be avoided, even with the help of the Central Server CS.

2.5.1.6 Abdolmaleky et al.'s Serverless Authentication Protocol

Abdolmaleky et al.'s [6] proposal is based on the idea proposed by Hoque et al. [45]. Hoque et al. [45] proposed a serverless authentication protocol based on random number generators. Their protocol is discussed in section 2.5.2.1. However, Abdolmaleky et al.'s construction is based on the hash function, which is why it is in this first group of serverless protocols.

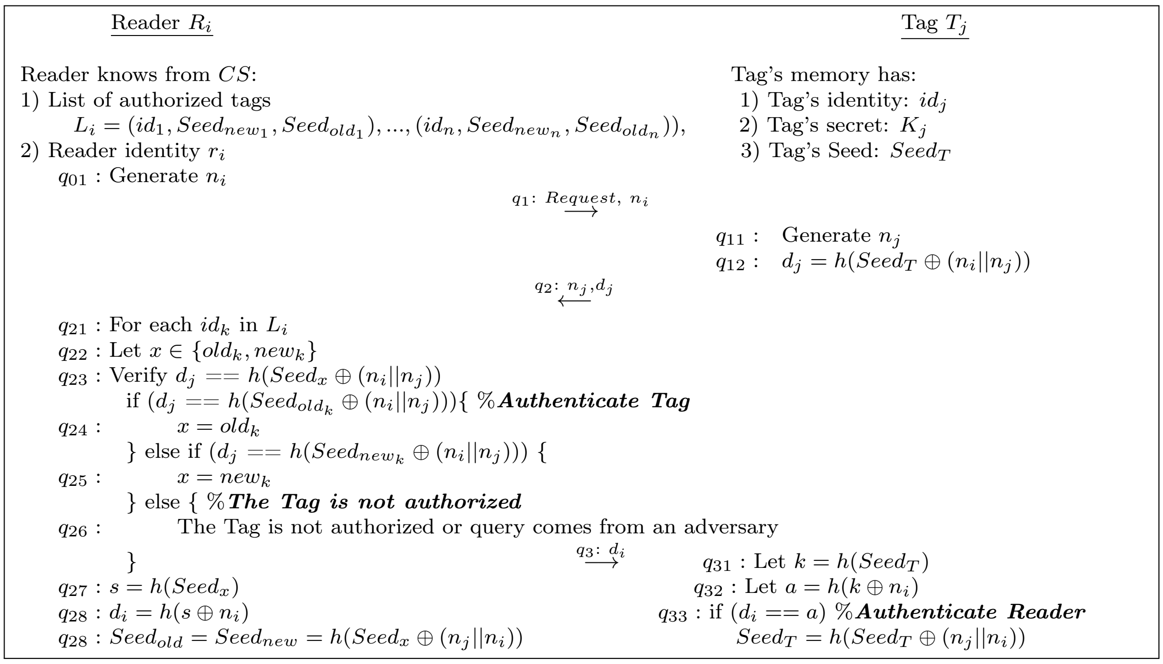
Abdolmaleky et al. studied the weaknesses in the proposals by Hoque et al. [45], Deng et al. [32] and Pourpouneh et al. [78] and proposed improvements by solving the weaknesses found in the former proposals i.e., secret parameters reveal, tag impersonation and reader impersonation attacks.

The major difference between Abdolmaleky et al.'s proposal and the preceding proposals of Hoque et al.'s protocol family is that, Abdolmaleky et al.'s proposal uses one-way hash function $h(.)$ for strengthening the security level, instead of PRNG $P(.)$ (refer to the last paragraph of section 2.5 for more details).

Like Hoque et al.'s proposal, the protocol requires that the reader communicate with the trusted Central Server CS to download the list of authorized tags L_i. In the end, each RFID reader R_i has a contact list L_i and a unique identifier r_i. L_i and r_i are obtained from the Central Server CS. In addition, each tag T_j includes a unique secret K_j and a unique identifier id_j. The subscripts i and j describe variables for particular R or T, respectively. The list L_i is represented as:

$$L_i = \begin{cases} Seed_{new_1}, & Seed_{old_1}, \quad id_1 \\ ..., & ... \\ Seed_{new_n}, & Seed_{old_n}, \quad id_n \end{cases}$$

The authentication phase between the reader R_i and tag T_j is depicted in Figure 2.10.



```
Reader $R_i$                                                          Tag $T_j$

Reader knows from $CS$:                                               Tag's memory has:
1) List of authorized tags                                             1) Tag's identity: $id_j$
   $L_i = (id_1, Seed_{new_1}, Seed_{old_1}), ..., (id_n, Seed_{new_n}, Seed_{old_n})$,   2) Tag's secret: $K_j$
2) Reader identity $r_i$                                               3) Tag's Seed: $Seed_T$
   $q_{01}$ : Generate $n_i$
                                  $\xrightarrow{q_1:\ Request,\ n_i}$
                                                                      $q_{11}$ : Generate $n_j$
                                                                      $q_{12}$ : $d_j = h(Seed_T \oplus (n_i||n_j))$
                                  $\xleftarrow{q_2:\ n_j, d_j}$
   $q_{21}$ : For each $id_k$ in $L_i$
   $q_{22}$ : Let $x \in \{old_k, new_k\}$
   $q_{23}$ : Verify $d_j == h(Seed_x \oplus (n_i||n_j))$
              if ($d_j == h(Seed_{old_k} \oplus (n_i||n_j))$){ %Authenticate Tag
   $q_{24}$ :         $x = old_k$
              } else if ($d_j == h(Seed_{new_k} \oplus (n_i||n_j))$) {
   $q_{25}$ :         $x = new_k$
              } else { %The Tag is not authorized
   $q_{26}$ :         The Tag is not authorized or query comes from an adversary
              }                   $\xrightarrow{q_3:\ d_i}$                 $q_{31}$ : Let $k = h(Seed_T)$
   $q_{27}$ : $s = h(Seed_x)$                                            $q_{32}$ : Let $a = h(k \oplus n_i)$
   $q_{28}$ : $d_i = h(s \oplus n_i)$                                    $q_{33}$ : if ($d_i == a$) %Authenticate Reader
   $q_{28}$ : $Seed_{old} = Seed_{new} = h(Seed_x \oplus (n_j||n_i))$            $Seed_T = h(Seed_T \oplus (n_j||n_i))$
```

FIGURE 2.10 - The improved version of the ServerLess RFID Authentication Protocol (SLRAP) proposed by Abdolmaleky et al. [6].

2.5.1.6.1 Analysis of Abdolmaleky et al.'s Authentication Protocol

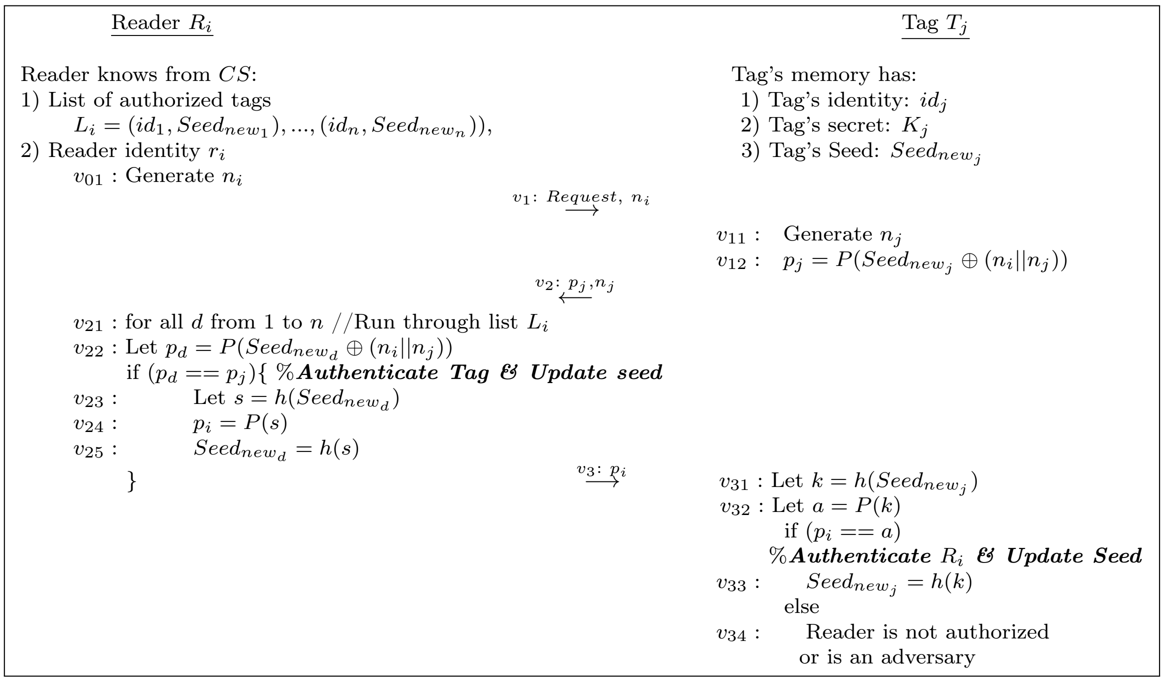
We find that Abdolmaleky et al.'s protocol suffers from two main weaknesses. These weaknesses are inherent to all protocols based on the Hoque et al.'s [45] proposal, described in section 2.5.2.1.

- *Limited in Functionality*
  These protocols are based on the idea that one tag can only be accessed by one reader (refer to section 2.5.2.1), hence not suitable for a distributed pervasive ecosystem. This feature limits the solution's usability in distributed pervasive environments where free interaction between pervasive devices is encouraged.
- *Vulnerable after the reader is compromised*
  Abdolmaleky et al.'s [6] protocol gives too much power to the reader. Once the access is granted, the Central Server CS no longer has access to the respective tags. The only party who can access the tags is the authorized reader. The problem with this kind of thinking is that, once the reader is compromised, the adversary can use the values stored in the reader to completely desynchronize the tags from the respective

legitimate readers and backend servers without the possibility of recovery, even with the help of the backend server. This happens because the authentication parameters stored in the tag are constantly changed during every interaction with the reader.

2.5.2 Protocols based on pseudo-random numbers

The protocols in this group use Pseudo-Random Number Generator (PRNG) functions as the basic building block for their schemes. Other cryptographic primitives may be required to facilitate functionalities e.g., XOR ($\oplus$), concatenation ($||$) and, rarely, hash functions.



```
Reader $R_i$                                                   Tag $T_j$

Reader knows from $CS$:                                        Tag's memory has:
1) List of authorized tags                                       1) Tag's identity: $id_j$
    $L_i = (id_1, Seed_{new_1}), ..., (id_n, Seed_{new_n})$,     2) Tag's secret: $K_j$
2) Reader identity $r_i$                                         3) Tag's Seed: $Seed_{new_j}$
    $v_{01}$ : Generate $n_i$
                                   $\xrightarrow{v_1:\ Request,\ n_i}$
                                                               $v_{11}$ :  Generate $n_j$
                                                               $v_{12}$ :  $p_j = P(Seed_{new_j} \oplus (n_i||n_j))$
                                   $\xleftarrow{v_2:\ p_j, n_j}$
    $v_{21}$ : for all $d$ from 1 to $n$ //Run through list $L_i$
    $v_{22}$ : Let $p_d = P(Seed_{new_d} \oplus (n_i||n_j))$
           if ($p_d == p_j$){ %**_Authenticate Tag & Update seed_**
    $v_{23}$ :      Let $s = h(Seed_{new_d})$
    $v_{24}$ :      $p_i = P(s)$
    $v_{25}$ :      $Seed_{new_d} = h(s)$
           }                        $\xrightarrow{v_3:\ p_i}$
                                                               $v_{31}$ : Let $k = h(Seed_{new_j})$
                                                               $v_{32}$ : Let $a = P(k)$
                                                                   if ($p_i == a$)
                                                                 %**_Authenticate_** $R_i$ **_& Update Seed_**
                                                               $v_{33}$ :   $Seed_{new_j} = h(k)$
                                                                   else
                                                               $v_{34}$ :   Reader is not authorized
                                                                     or is an adversary
```

FIGURE 2.11 - Hoque et al.'s serverless authentication protocol [45].

2.5.2.1 Hoque et al.'s Serverless Protocols

In 2010, Hoque et al. [45] proposed two serverless protocols, an authentication protocol and a secure tag search protocol. Hoque et al.'s contributions aimed at achieving four key goals. First, the protocol should perform mutual authentication between an RFID reader and RFID tag. Second, the protocol should enable tag ownership transfer. Third, the protocol should ensure the privacy and the security of the tags after ownership transfer i.e., only the new owner (with the aid of the RFID reader) should be able to access the tag. Fourth, the protocol should be scalable to allow the increasing number of tags.

2.5.2.1.1 Hoque et al.'s Authentication Protocol

In Hoque et al.'s proposal, the reader R_i has a unique identifier r_i and a contact list L_i. R_i obtains r_i and L_i from CS, after authenticating itself. The CS is a trusted party that deploys all the RFID tags and authorizes the communication between RFID readers and tags. It is assumed that R_i and CS communicate through a secure channel. On the other hand, each RFID tag T contains a unique identifier id_j and a unique secret K_j in its nonvolatile memory.

The authorization list L_i contains information about the tags which R_i has access to. And the information about each tag comprises a seed and the id_j of the tag. For instance, if R_i is authorized to access tags $T_1, ..., T_n$ then the list L_i will constitute of the following shape after authenticating itself to CS:

$$L_i = \begin{cases} Seed_{new_1} & : & id_1 \\ ... & : & ... \\ Seed_{new_n} & : & id_n \end{cases}$$

Where $Seed_{new_j}$ is initialized as $Seed_{new_j} = h(r_i||K_j)$. The initial seeds $Seed_{new_1}, ..., Seed_{new_n}$ are computed by CS for each T_j and stored in R_i. On the contrary, the tag T_j stores only one seed $Seed_{new_j}$ for the reader R_i i.e., the only reader authorized access to. The seed is stored in a nonvolatile memory. The authentication phase between the reader R_i and tag T_j is depicted in Figure 2.11.

```
Reader $R_i$                                                        Tag $T_j$

Reader knows from $CS$:                                             Tag's memory has:
1) List of authorized tags                                            1) Tag's identity: $id_j$
      $L_i = (id_1, Seed_{new_1}), ..., (id_n, Seed_{new_n})$,        2) Tag's secret: $K_j$
2) Reader identity $r_i$                                              3) Tag's Seed: $Seed_{new_j}$
      $u_{00}$ : Choose the Tag to search for: $T_j$ with $id_j$ and $Seed_{new_j}$
      $u_{01}$ : Compute $n_{desired} = P(Seed_{new_j})$
                                        $u_1$: $\xrightarrow{Request,\ n_{desired}}$
                                                                    $u_{11}$ :  $a = P(Seed_{new_j})$
                                                                    $u_{12}$ :  $if(a == n_{desired})\{$
                                                                        %**Authenticate $R_i$ & Update seed**
                                                                    $u_{13}$ :      Let $y = h(Seed_{new_j})$
                                                                    $u_{14}$ :      Let $x = P(y)$
                                                                    $u_{15}$ :      $Seed_{new_j} = h(y)$
                                                                                 } else{ %**Unauthorized Reader**
                                                                    $u_{16}$ :      Generate $n_j$ with probability $\lambda$
                                                                                 }
                                        $u_2$: $\xleftarrow{x\ or\ n_j}$
      $u_{21}$ : Let $s = h(Seed_{new_j})$
      $u_{21}$ : Let $m = P(s)$
      $u_{22}$ : for each response from the group of tags
             if $(m == response)\{$ %**Authorized Tag**
      $u_{24}$ :       $Seed_{new_j} = h(s)$
      $u_{25}$ :       $T_j$ found
             } else %**Unauthorized Tag**
      $u_{26}$ :       $T_j$ not found
```

FIGURE 2.12 - Hoque et al.'s serverless search protocol [45].

2.5.2.1.2 Hoque et al.'s Secure Search Protocol

Hoque et al.'s [45] search protocol is based on the idea that, the tag should respond only to its authorized reader and the reader should query only the tags it is authorized access to. Both tags and readers should update their seeds after a successful search. Moreover, the reader should issue a query that only a legitimate tag can understand and a tag replies in such a manner that only an authorized reader can understand. Hoque et al.'s search protocol is depicted in Figure 2.12.

2.5.2.1.3 Analysis of Hoque et al.'s Protocols

Hoque et al.'s protocols are vulnerable to the following attacks:

- *Desynchronization Attack*
  In 2014, Deng et al. [32] showed that Hoque et al.'s protocol is vulnerable to the desynchronization attack after one protocol run. This attack happens when the adversary prevents message v_3 (refer to Figure 2.11) from being delivered to the tag. At this stage, the reader's *Seed* is updated while the tag retains the old value. As a result, the tag and the reader are desynchronized.
  Moreover, our analysis concludes that Hoque et al.'s search protocol, described in Figure 2.12, also suffers from the desynchronization attack when message u_2 from the desired tag is not delivered to the reader. This way, the tag updates its $Seed_{new_j}$ while the reader still maintains the old seed. During the next authentication session, the reader still uses the same *Seed* value while the tag has a different *Seed* value; hence, the authentication fails.
- *Limited in Functionality*
  Hoque et al.'s protocols associate one tag to one reader (refer to section 2.5.2.1). The respective tag cannot be solicited by any other reader than the one it is paired with. This feature is not suitable for a distributed pervasive ecosystem as it limits the solution's usability where free interaction between pervasive devices is encouraged.

2.5.2.2 Deng et al.'s Serverless Authentication Protocol

Deng et al. [32] proposed improvements to Hoque et al.'s [45] protocol to avoid the desynchronization attack. Their proposal involved keeping the history of the seed update on the reader side i.e., the reader stores the current and the previous seeds in its memory. Once the authentication fails with the new seed, the reader reverts to the old seed to authenticate the tag.

The major modification proposed by Deng et al. is the inclusion of two seeds in the list L_i instead of one seed used in Hoque et al. [45], described in section 2.5.2.1.1. In turn, the list L_i becomes the following:

$$L_i = \begin{cases} Seed_{new_1}, & Seed_{old_1}, & id_1 \\ ..., & ..., & ... \\ Seed_{new_n}, & Seed_{old_n}, & id_n \end{cases}$$

2.5.2.2.1 Analysis of Deng et al.'s Authentication Protocols

In 2014, Pourpouneh et al. [78] observed that Deng et al.'s protocol is still vulnerable to the desynchronization attack after two protocol runs. This happens because Deng et al.'s protocol updates only the new seed, $Seed_{new_j}$, but keeps the old seed, $Seed_{old_j}$, constant and unchanged. This way, only two protocol runs are enough to successfully desynchronize the tags and readers.

In 2015, Abdolmaleky et al. [6] conducted a thorough analysis and proved that Deng et al.'s [32] protocol is also vulnerable to other attacks. They show that the adversary can easily obtain the secret parameters stored within the tag by eavesdropping one exchange session between the tag and the reader. They also show that the adversary can impersonate the tag or the reader simply by eavesdropping, storing and replaying the valid information exchanged between valid parties.

Moreover, as Deng et al.'s protocol inherits all the basic properties of Hoque et al.'s protocol (refer to section 2.5.2.1), then it is also inherent to the weaknesses described in section 2.5.2.1.3.

2.5.2.3 Pourpouneh et al.'s Authentication Protocol

Pourpouneh et al. [78] proposed an improvement to Deng et al.'s [32] protocol by solving the desynchronization attack. Their protocol construction is similar to that of Deng et al.'s protocol with one exception, they allowed the systematic update of both seeds $Seed_{new_j}$ and $Seed_{old_j}$ depending on the information received. The reader updates the value $Seed_{new_j}$ with the latest authorized value after every successful run of the protocol. This allows the reader to maintain similar $Seed$ values with the tags, even if the adversary attempts to hinder the delivery of some messages. Pourpouneh et al.'s proposal was improved by Abdolmaleky et al. [6], discussed in section 2.5.1.6, hence we do not provide the protocol sequence here.

2.5.2.3.1 Analysis of Pourpouneh et al.'s Protocol

Pourpouneh et al.'s [78] protocol is vulnerable to the following kinds of attacks.

- *Secret parameter disclosure and impersonation attacks*
  In 2015, Abdolmaleky et al. [6] conducted a thorough analysis and proved that Pourpouneh et al.'s [78] protocol is vulnerable to secret parameter disclosure, tag impersonation and reader impersonation attacks. Abdolmaleky et al. show that the adversary can easily obtain the secret parameters stored within the tag by eavesdropping one session of exchange between the tag and the reader. They also show that the adversary can impersonate the tag or the reader simply by eavesdropping, storing and replaying the valid information exchanged between valid parties.
  Moreover, with the exception of the desynchronization attack, Pourpouneh et al.'s protocol inherits all the basic properties of Hoque et al.'s protocol (refer to section 2.5.2.1), then it is also inherent to the weaknesses described in section 2.5.1.6.1.

- *Vulnerable after the reader is compromised*
  Pourpouneh et al. [78] protocol, like Abdolmaleky et al.'s [6] protocol, gives too much power to the reader. Once the access is granted to the reader R_i, the Central Server CS no longer has access to the respective tags. It is only R_i that can access the respective tags. This problem was further explained in section 2.5.1.6.1.

2.5.2.4 Jeon et al.'s Serverless Search Protocol

Jeon et al. [50] proposed an ultra-lightweight RFID search protocol for low-cost tags based on Xie et al.'s [107] (see section 2.5.1.5). They claim that their protocol is more realistic and suitable for low-cost RFID tags due to the use of Pseudo-Random Number Generator (PRNG) instead of the hash functions used in the Xie et al.'s protocol. Moreover, Jeon et al. claim that their protocol is resistant to all the attacks, even after the reader is compromised.

Jeon et al.'s protocol has three phases namely registration phase, initialization phase, and login and searching phase.

The registration phase takes place when the user using the RFID reader R_i registers to the Central Server CS by submitting username and password via a secure channel. CS stores the user's information in the database. The password is stored as a hashed value, i.e., h(Pwd), instead of a plaintext.

The initialization phase of Jeon et al.'s protocol is similar to that of Xie et al.'s protocol. The user holding an RFID reader R_i connects to the Central Server CS and downloads an Access List L_i. The main difference between the list in Jeon et al.'s proposal compared to that of Xie et al.'s proposal is that, Jeon et al.'s list contains username (UID) and password (Pwd) for the user authorized to use the list.

The list L_i has the following contents:

$$L_i = \{(UID, h(Pwd)), (id_1, P(r_i \oplus K_1)), ..., (id_n, P(r_i \oplus K_n))\} \tag{2.18}$$

The login and seeking phase are depicted in Figure 2.13. This phase takes place between the RFID reader R_i and tag T_j without the presence of the Central Server CS.

2.5.2.4.1 Analysis of Jeon et al.'s Search Protocol

Jeon et al. [50] protocol is vulnerable to the following attacks:

- *Traceability attacks on both Reader and Tag*
  Jeon et al.'s [50] protocol suffers from the reader traceability attack, which was not in the original protocol proposed by Xie et al.'s [107] and discussed in section 2.5.1.5. The traceability attack is caused by the exchange of constant parameter $A = P(id_j) \oplus r_i$ in message m_1 from the reader to a group of tags. This parameter contains a combination of identities of reader and tag, hence cannot change from one session to the next for a given pair of tag and reader. Moreover, the parameter acts as a signature for the pair of reader and tag combination. In turn, an adversary

```
Reader $R_i$                                                    Tag $id_j$

Reader knows from $CS$:                                         Tag's memory has:
1) List of authorized tags                                        1) Tag's identity $id_j$
    $L_i = \{(UID, h(Pwd)), (id_1, P(r_i \oplus K_1)), ..., (id_n, P(r_i \oplus K_n))\}$,   2) Tag's secret $K_j$
2) Reader identity $r_i$                                          3) Binary state value $S_j = 0$
    $m_{01}$ : Request username: $UID$ and password: $Pwd$         4) $Counter = 0$
    $m_{02}$ : Compute $h(Pwd)$
        if ($UID$ and $h(Pwd) \notin L_i$)
    $m_{03}$ :     EXIT
        else
            Compute:
    $m_{04}$ :     $A = P(id_j) \oplus r_i$
    $m_{05}$ :     Generate nonce $n_i$
    $m_{06}$ :     $B = P(r_i \oplus K_j) \oplus n_i$
    $m_{07}$ :     $C = P(n_i \oplus r_i)$
                            $\xrightarrow{m_1\text{: } Broadcast\ A,\ B,\ C}$
                                                           Compute:
                                                           $m_{11}$ :  $r'_i = A \oplus P(id_j)$
                                                           $m_{12}$ :  $n'_i = B \oplus P(r'_i \oplus K_j)$
                                                           $m_{13}$ :  $C' = P(n'_i \oplus r'_i)$
                                                               if ($C == C'$)
                                                           $m_{14}$ :      $Counter_j = Counter_j + 1$
                                                           $m_{15}$ :      $n_j = P(Counter_j \oplus K_j)$
                                                           $m_{16}$ :      $D = n'_i \oplus n_j$
                                                           $m_{17}$ :      $E = P(n_j \oplus id_j)$
                                                               else  %**Unauthorized** $R_i$
                                                           $m_{18}$ :      $D = n'_i \oplus C'$
                                                           $m_{19}$ :      $E = P(n'_i \oplus id_j)$
                            $\xleftarrow{m_2\text{: } D,\ E}$
    $m_{21}$ : $n'_j = D \oplus n_i$
    $m_{22}$ : $E' = P(n'_j \oplus id_j)$
        if ($E == E'$)  %**Authenticate Tag**
    $m_{23}$ :     $F = P(n_i \oplus n'_j \oplus r_i \oplus id_j)$
        else %**Unauthorized Tag**
    $m_{24}$ :     $F = n_i$ //Generate a random number $n_i$ with probability $\lambda$
                            $\xrightarrow{m_3\text{: } F}$
                                                           $m_{31}$ :  $F' = P(n'_i \oplus n_j \oplus r'_i \oplus id_j)$
                                                           $m_{32}$ :  if ($F == F'$)
                                                           $m_{33}$ :      $S_j = 1$  %**Authenticate** $R_i$
```

FIGURE 2.13 - Jeon et al.'s serverless search protocol

has only to store all broadcast queries from the reader and compare the values of parameter A to see if there is a match i.e., if the reader searches for the same tags or not.

- *Replay attacks*
  We observe that the lack of context in the messages m_1 and m_2 (refer to Figure 2.13) exchanged between R_i and T_j simplifies the replay attacks. The same valid message m_1 emitted by R_i can be replayed countless times to the same group of tags. During each replay, all tags reply and their responses reveal more than necessary. All the wrong tags respond with constant values in parameters $D = n' \oplus C'$ and $E' = P(n'_i \oplus id_j)$ in message m_2. However, only the correct tag being searched responds with a random message m_2, each time. This is because, for all the wrongs tags, the parameters D and E in message m_2 depend on the values sent in message m_1 without any additional randomization. Thus, the adversary has only to interro-

gate each tag individually by replaying the same message at least twice. The same message m_2 reveals that the tag is not the one being search, while a random message m_2 signifies it is the correct tag being searched. This weakness is not found in the original protocol; in Xie et al.'s [107] protocol the tag always responds with a random value. Additionally, combining messages m_1 and m_2 can simplify the identify of each individual tag T_j depending on R_i's queries.

- *Reader compromise attacks*
  Jeon et al. [50] claim that their additional layer of authentication using username and password restrict the malicious usage of parameters found in the list L_i. However, we find this idea unnecessary as it does not add any real security to the protocol. This is because, Jeon et al. confirm that the contents of the list L_i can be accessible to the adversary after compromising the reader R_i. Knowing that the contents of L_i are neither encrypted nor signed, the adversary can easily tamper with the list L_i i.e., an adversary can simply replace the credentials - username UID and password $h(Pwd)$ - in the list with his respective credentials without compromising the validity of the contents of L_i. Thereafter, the adversary can continue to use the parameters stored in the list L_i. This is possible because of two reasons. First, the authentication of the user is done locally in the reader R_i and is neither verified with CS nor T_j during the tag search session. Moreover, the tag T_j is ignorant of the user using the reader R_i, as it only authenticates the reader R_i not the user. Second, like the majority of the existing serverless protocols, Jeon et al.'s protocol provides static parameters in the list L_i to the reader R_i. After acquiring these parameters, the reader R_i does not need to authenticate again to the CS and request for new parameters. In turn, these parameters may be used by anyone without the possibility to revoke them. We discuss this problem in section 2.5.7.

2.5.3 Protocols based on classic encryption

The protocols in this group employ classic means of encryption to perform authentication between resource constrained devices e.g., Advanced Encryption Standard (AES) or Elliptic Curve Cryprography (ECC).

2.5.3.1 Ahamed et al.'s Serverless Authentication Protocol

In 2008, Ahamed et al. [9] proposed ECC based RFID Authentication Protocol (ERAP), a serverless scheme that performs mutual authentication between an RFID reader and authorized RFID tags. The protocol employs public key scheme, based on Elliptic Curve Cryptography (ECC), for performing authentication between the tag and the reader without the backend server's intervention. The protocol requires that the respective communicating devices i.e., RFID readers and tags must have the capabilities to perform calculations based on ECC. However, ECC demands are too much for the majority of the resource constrained pervasive devices, hence making Ahamed et al.'s protocol impractical in the context of resource constrained devices.

Ahamed et al.'s protocol is an interesting attempt to adapt public key cryptosystems to serverless authentication paradigm. Their proposal supports mutual authentication in a distributed environment, which is a desirable feature for pervasive systems.

2.5.3.1.1 Analysis of Ahamed et al.'s Authentication Protocol

In 2012, Mahalle et al. [61] observed that Ahamed et al.'s proposal is vulnerable to various security and privacy attacks. First, Ahamed et al.'s proposal is vulnerable to the Denial-of-Service (DoS). This attack is possible due to the use of asymmetric encryption schemes. The adversary can send various requests to the tag, which can be busy decrypting and encrypting and unable to respond to legitimate requests. Second, the protocol is also vulnerable to the man-in-the-middle and replay attacks. Third, the protocol does not support access control between RFID readers and tags. These are important characteristics for pervasive systems. These vulnerabilities make Ahamed et al.'s protocol unfit for pervasive technologies.

Moreover, we find that Ahamed et al.'s protocol is vulnerable once the readers, or the authentication data in them, are compromised. This is because, the parameters granted to the reader during the initialization phase remain valid for the whole lifetime of the reader and respective tags, as they do not have a limited period of validity. This leads to the security and privacy issues as elaborated in section 2.5.7.2.

2.5.3.2 Won et al.'s Serverless Search Protocol

In 2008, Won et al. [106] proposed a secure RFID tag search protocol which uses timestamps to ensure freshness of the search queries. It also uses a block cipher (AES-128) to hide a portable reader identifier. AES [2, 79, 91] stands for Advanced Encryption Standard, a widely used standard symmetric encryption algorithm. Won et al.'s protocol is depicted in Figure 2.14.

Won et al.'s protocol exchanges two messages w_1 and w_2 for search query and reply, respectively.

2.5.3.2.1 Analysis of Won et al.'s Search Protocol

Xie et al. [107] note that, the use of symmetric encryption is required on the tag, making the protocol far from lightweight. Moreover, the protocol is vulnerable to the Denial-of-Service (DoS) attack. The DoS attack can be launched when the adversary continuously sends requests to the tag. The tag will be busy with fake requests by doing encryption and decryption processes while other legitimate requests will go unanswered. This weakness is also found in Chun et al.'s [26] protocol.

Xie et al. [107] also observe that Won et al.'s protocol does not perform mutual authentication between the reader and the tag. Thus, the authenticity of the search query from the reader cannot be fully guaranteed as only the tag is authenticated during the process.

```
Reader $R_i$                                                      Tag $T_j$

Reader knows from $CS$:                                           Tag's memory has:
1) List of authorized tags                                         1) Tag's identity: $id_j$
    $L_i = \{(id_1, ENC_{K_1}[r_i \oplus id_1]), ..., (id_n, ENC_{K_n}[r_i \oplus id_n])\}$,    2) Tag's secret: $K_j$
2) Reader identity $r_i$                                           3) Last success time: $ltime$
    $w_{01}$ : Get $ctime$
    $w_{02}$ : Compute $S_1 = ENC_{id_j}[ctime \oplus r_i]$
    $w_{03}$ : Compute $S_2 = ENC_{ctime \oplus r_i}[ENC_{K_j}[r_i \oplus id_j]]$
                          $\xrightarrow{w_1:\ Broadcast,\ ctime, S_1, S_2}$
                                              $w_{11}$ :  if ($ctime > ltime$){
                                              $w_{12}$ :    $r_i = DEC_{id_j}(S_1) \oplus ctime$
                                              $w_{13}$ :    $id^* = DEC_{K_j}(DEC_{ctime \oplus r_i}(S_2)) \oplus r_i$
                                              $w_{14}$ :    if ($id_j == id^*$){ %**Authenticate** $R_i$
                                              $w_{15}$ :        Generate $n_j$
                                              $w_{16}$ :        $S_3 = ENC_{id_j \oplus n_j}[S_1]$
                                              $w_{17}$ :        $ltime = ctime$
                          $\xleftarrow{w_2:\ S_3, n_j}$
                                                               }
                                                           }
    $w_{21}$ :   If $S_1 == DEC_{id_j \oplus n_j}[S_3]$ %**Authenticate Tag**
```

FIGURE 2.14 - Won et al.'s search protocol [106].

We observe that Won et al. use timestamps in their protocol construction. However, the timestamps are only limited to ensuring the freshness of the messages but do not limit the usage of the authentication parameters in the reader. This implies that once the reader's information are stolen they can be used to authenticate the tags without any possibility to revoke them. This problem is further explained in section 2.5.7.

2.5.3.3 Chun et al.'s Serverless Search Protocol

In 2011, Chun et al. [26] proposed a tag search protocol that preserves the privacy of the communicating parties. The protocol is based on the symmetric encryption scheme, specifically AES-128 encryption. Chun et al.'s protocol has similar properties to that proposed by Won et al. [106] (refer to section 2.5.3.2) hence we do not provide the protocol sequence and the analysis here. The protocol analysis is included in section 2.5.3.2.1.

2.5.4 Comparing security and privacy of the existing serverless schemes

Tables 2.4 and 2.5 compare the security and privacy properties of the existing authentication and search protocols, respectively. We use the security and privacy requirements described in section 2.3 as base criteria.

Table 2.4 - Comparing the security and privacy properties of the existing serverless authentication protocols (✓: Supported; ✗: Not Supported)

Group	Protocol	Mutual Authentication	Tag Untraceability	Reader Untraceability	Tag Anti-Cloning	Reader Anti-Cloning	Protect Tag's Identity	Protect Reader's Identity	Anti-Replay	Reader Compromise Resistance	Desynchronization Resistance
	Tan et al.'s [99]	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓
	Lee et al. [57]	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗
1	Lin et al. [59]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
	Jialiang et al. [51]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
	Abdolmaleky et al. [6]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
	Hoque et al. [45]	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
2	Deng et al. [32]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	Pourpouneh et al. [78]	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓
3	Ahamed et al. [9]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓

Table 2.5 - Comparing the security and privacy properties of the existing serverless search protocols (✓: Supported; ✗: Not Supported)

Group	Protocol	Mutual Authentication	Tag Untraceability	Reader Untraceability	Tag Anti-Cloning	Reader Anti-Cloning	Protect Tag's Identity	Protect Reader's Identity	Anti-Replay	Reader Compromise Resistance	Desynchronization Resistance
	Tan et al.'s [99]	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓
	Lin et al. [59]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
1	Lee et al. [57]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
	Xie et al. [107]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
	Jialiang et al. [51]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
2	Jeon et al. [50]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
	Hoque et al. [45]	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
3	Chun et al. [26]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
	Won et al. [106]	✗	✓	✓	✓	✓	✓	✓	✓	✗	✗

2.5.5 Comparing performance of the existing serverless schemes

Table 2.6 compares the performances of various existing serverless protocols. The respective protocols are compared in terms of computation, communication overhead and storage costs and the tag side. To conduct a fair comparison, the parameters in respective protocols included in Table 2.6 are set to 128 bits each, with the exception of *ctime* and *ltime*, which are set to 32 bits together with m, which is set to 64.

TABLE 2.6 - Comparing performance of the existing serverless protocols

Group	**Protocol**	**PERFORMANCE CRITERIA ON TAG**			
		Computation	*Total Message*	*Message Sizes (bits) {Sent/Received}*	*Storage (bits)*
1	Tan et al.'s [99] Authentication	4 Hash / 1 PRNG	4	320 / 128	960
	Tan et al.'s [99] Search	3 Hash / 1 PRNG	2	256 / 384	1024
	Lin et al. [59] Authentication	1 Hash / 1 PRNG	2	256 / 32	512
	Lin et al. [59] Search	2 Hash / 1 PRNG	2	256 / 384	896
	Lee et al. [57] Authentication	4 Hash / 1 PRNG	3	320 / 384	960
	Lee et al. [57] Search	3 Hash / 1 PRNG	3	256 / 512	1024
	Jialiang et al. [51] Authentication	3 Hash / 1 PRNG	3	384 / 512	896
	Jialiang et al. [51] Search	3 Hash / 1 PRNG	2	384 / 384	1024
	Xie et al. [107]	4 Hash / 1 PRNG	3	256 / 512	1026
	Abdolmaleky et al. [6]	4 Hash / 1 PRNG	3	256 / 256	1152
2	Hoque et al. [45] Authentication	2 Hash / 3 PRNG	3	256 / 256	1024
	Hoque et al. [45] Search	2 Hash / 3 PRNG	3	128 / 256	1024
	Deng et al. [32]	2 Hash / 3 PRNG	3	256 / 256	1024
	Pourpouneh et al. [78]	2 Hash / 3 PRNG	3	256 / 256	1024
	Jeon et al. [50]	4 PRNG	4	512 / 256	896
3	Chun et al. [26]	2 AES Encryptions / 1 AES Decryption / 1 PRNG	2	512 / 256	896
	Won et al. [106]	1 AES Encryption / 2 AES Decryptions / 1 PRNG	2	256 / 384	832
	Ahamed et al. [9]	3 ECC Operations / 1 Certificate / 1 PRNG	3	384 / 384	>1024

In Table 2.6, PRNG stands for Pseudo-Random Number Generator. For each random value generation the PRNG is used; Hash function is based on SHA-1 which stands for Secure Hash Algorithm 1. We choose SHA-1 as a base Hash to facilitate the comparison of the protocols since the respective protocols do not specify which version of the Hash function they use; AES stands for Advanced Encryption Standard. ECC stands for Elliptic Curve Cryptography. ECC operations include point multiplication, modular division and certificate verification.

2.5.6 Drawbacks of Existing Serverless Schemes

The existing serverless schemes have several drawbacks. Two of the most common are the inability to enforce access parameter revocation and the lack of provision of fine grained access controls for each communication session [99]. Additionally, there are more weaknesses in each group of the serverless protocols as discussed in the sections below.

2.5.6.1 Drawbacks of serverless authentication schemes

Most of the existing serverless protocols are inherently vulnerable due to the lack of using context information for the authentication parameters. This problem is addressed in section 2.5.7.1. Moreover, most of the existing serverless authentication protocols do not provide mutual authentication, which result into diminished trust between the communicating parties. The lack of mutual authentication also leads to other attacks such as impersonation because only one entity proves the authenticity of the other.

2.5.6.2 Drawbacks of serverless search schemes

In most of the proposals, the search protocols are proposed to enhance the functionalities of the authentication protocols. This implies that these protocols have a lot in common, hence share some vulnerabilities. For instance, the use of parameters without a definite expiry period make these protocols inherently vulnerable to various attacks, especially when the respective readers or their data are compromised.

Moreover, as it can also be seen in search protocol security and privacy comparison (refer to Table 2.5), most search protocols do not provide mutual authentication. This may lead to the reader impersonation attack because it is only the tag that is authenticated.

The majority of the tag search protocols do not ensure the privacy of the reader and tag during the search process. Some protocols, such as Tan et al.'s [99] and Jialiang et al. [51] communicate the reader's identity as cleartext. This leads to the traceability issue when the two parties communicate in different sessions. This vulnerability has been experienced in Tan et al.'s [99], Jeon et al.'s [50], Lin et al.'s [59], Jialiang et al. [51] and Lee et al.'s [57] protocols.

2.5.7 Enhancing Security & Privacy in Serverless Schemes

The majority of the reviewed serverless protocols in section 2.5 are inherently vulnerable. Most of these vulnerabilities e.g., provision of the constant authentication parameters to the readers, are due to the lack of embedded clocks in the respective resource constrained pervasive devices. In this section, we describe the existing possible solutions that can help solve these challenges.

2.5.7.1 Incorporating Context Information

It is interesting to incorporate context information e.g. date and time or geographic position in the authentication for pervasive systems. The only obstacle is that some of the resource constrained devices, e.g. passive RFID tags, do not have embedded location sensors or clocks, which makes it a challenge to efficiently manipulate and make use of location or time.

In 2006, Tsudik [102] presented the use of time as an authentication parameter between RFID reader and RFID tags with a very simple idea; the reader has to periodically broadcast its current timestamp and the tags within the reader's vicinity stores the value statically as their current timestamp. With repeated broadcasts, the tag compares the reader's value to the stored timestamp. If the broadcast timestamp is larger than the stored timestamp, the tag updates its timestamp and replies with a keyed hash over its permanent key and the new timestamp. Otherwise, the tag computes a pseudo random number to confuse an adversary and avoid narrowing attacks. Narrowing attack occurs when the adversary queries a tag with a particular timestamp and then later tries to identify the same tag by querying a candidate tag with a timestamp slightly above the previous one [102].

Tsudik [102] himself noted that his proposed scheme is vulnerable to the denial-of-service (DoS) attack because an adversary can easily desynchronize a tag by sending a timestamp value that is in the distant future. As a consequence of which, the tag will deny any time value from the legitimate readers.

We further explore the idea of using date and time in serverless authentication in this thesis as the idea is incorporated in all our proposed protocols. We use date and time to thwart replay attacks and enforce automated parameter expiry, a feature that was not included in previous proposals for the serverless protocols such as [51], [57] and [59].

2.5.7.2 Enforcing Parameter Revocation or Validity

The cryptographic mechanisms alone are not sufficient to protect pervasive devices against attacks, especially those originating from compromised, but non-revoked, pervasive nodes. The majority of the existing serverless authentication schemes, such as [99], [51], [26], [107] and [50], are vulnerable to several security and privacy attacks due to the lack of proper mechanisms to revoke or nullify the parameters given to the RFID readers. The parameters granted to the readers are static, irrevocable and with an unlimited period of use. This means, the parameters can be reused to perform authentication between respective devices as long as they exist without requiring to contact the backend server. The respective protocols were designed without taking into account the reader's compromise attack. If an adversary obtains access to the valid information stored in a captured reader, it is impossible to revoke them as they always remain valid.

Access right revocation means to invalidate access of an entity to the given resources. One way of implementing access revocation is by using parameters with defined context e.g., time of use or geographic location. The parameter validity allows to ensure limited access to the given resources and also enforces automatic revocation once the respective criteria is no longer valid e.g., if the allocated date and time expire.

For instance, in Tan et al.'s [99] protocol (refer to section 2.5.1.1.1) or Jialiang et al.'s [51] protocols (refer to section 2.5.1.4.1) the parameters in the list L_i from the server do not have an expiry period or limit of validity. This implies that, the reader needs to request for access from the Central Server only once and use the parameters forever. With this setup, even the Central Server does not have the power to revoke the parameters in L_i, unless the secret values in the respective tags are changed.

Most authors of the existing serverless protocols, such as [51], [50] and [58], claim that even though the adversary may acquire the contents of the RFID reader after a physical compromise, it cannot create valid tags using the respective data to fool other legitimate readers. However, in some cases, once the adversary has the contents of the reader, which are not bound to expire, he might be less interested in creating fake tags. Rather, he will be more focused in acquiring the information from all the legitimate tags that the captured readers are authorized access to. In most setups, it is the tags that provide valuable information, not the readers. Hence, the capability of reading the contents of legitimate tags itself is sufficient enough to worry about giving away the contents of the reader.

In our proposed solutions (refer to chapters 3 and 4) we solve the problem of static parameters using static timestamps. The date and time in a format of unix-like 32-bit timestamp are included as context information to bind the parameters used during authentication. In turn, the parameters granted by the central server are only valid during the alloted time. This has been implemented in our protocols regardless of the lack of embedded clocks in the resource constrained devices considered in our scenarios.

2.6 Conclusion

In this chapter we introduce serverless protocols by outlining their advantages together with their core security, privacy and performance requirements. We show how and why security and privacy are important aspects for the information generated, stored or exchanged between different devices in pervasive systems. Then, we provide important features for serverless protocols in pervasive ecosystems.

We classify and provide thorough analysis of the existing serverless protocols by discussing their weaknesses and strengths in terms of security, privacy and performance relative to the resources offered by relevant pervasive systems. We also compare the discussed serverless schemes in terms of performance, privacy and security properties. It is apparent that most of the existing security solutions are still vulnerable or inadequate, hence cannot solve the security and privacy challenges of pervasive systems. There is a need for more researches to come up with efficient security solutions compatible with pervasive systems while respecting their specific resource constraints and requirements. We also suggest solutions to the common weaknesses found in most of the existing serverless protocols e.g., the use of authentication parameters without defined period of expiry.

In the next chapter, we present our first contribution, the authentication of cluster devices with untrusted parties using a serverless paradigm, a security solution suitable for autonomous resource constrained pervasive systems. Our proposed solution addresses some of the key challenges in the existing serverless schemes.

Chapter 3

Authenticating Cluster Devices with Untrusted Parties using a Serverless Paradigm

If your enemy is secure at all points, be prepared for him. If he is in superior strength, evade him. If your opponent is temperamental, seek to irritate him. Pretend to be weak, that he may grow arrogant. If he is taking his ease, give him no rest. If his forces are united, separate them. If sovereign and subject are in accord, put division between them. Attack him where he is unprepared, appear where you are not expected.

Sun Tzu
- The Art of War

Contents

3.1 Introduction

THIS chapter proposes a serverless mutual authentication protocol for mobile resource-constrained pervasive devices. The solution proposed in this chapter responds to the challenges posed by *scenario B* of the shipping scenario described in section 1.3.2.2. The scenario involves secure communication between a mobile terminal, referred to as M-Trax, and a resource constrained device, referred to as Trax-Box, with the authorization of the central server, referred to as Trax-hub.

Our proposed solution enforces parameter expiration, which is an important feature in improving the security of the pervasive systems, even after the respective devices are physically compromised. Thus, our proposed solution is a token-based authentication protocol, similar to the Kerberos authentication protocol [65]. Our proposed solution also takes into account some of the important privacy and security properties in serverless protocols. Moreover, it is symmetric and lightweight, as it is designed using simple primitives like keyed-Hash Message Authentication Code (HMAC), XOR ($\oplus$) and comparison operations.

The proposed protocol in this chapter has four advantages compared to the existing serverless protocols, as described in section 2.5. First, it preserves the secrecy of the communicating parties by avoiding to divulge the secret and unchanging private information during the authentication process. Second, it exchanges very few data during the mutual authentication, hence reducing the communication cost. Third, the protocol supports spontaneous communication as it does not require devices to share parameters before the authentication process. Lastly, the protocol enforces access rights revocation, which ensures that the authentication parameters have a defined period of validity.

The rest of this chapter is organized as follows, section 3.2 presents actors and features for *scenario B*, discussed in section 1.3.2.2, which introduces the authentication challenge between a Trax-Box and a mobile terminal, M-Trax, as discussed in section 1.3.2. Section 3.3 presents a background on the Kerberos protocol before discussing our proposed protocol in section 3.4. Section 3.5 presents the validation of our protocol using AVISPA tool, the protocol analysis is given in section 3.6 and section 3.7 concludes.

3.2 Actors & Features

This section focuses on *scenario B* presented in section 1.3.2.2. The scenario involves the communication between the Trax-Boxes and M-Trax devices. We further elaborate on the actors involved and their features before proposing an efficient security protocol for securing the communication between actors.

3.2.1 Actors

The proposed scenario contains three types of communicating parties namely, Trax-hub as the *Central Server (CS)*, Trax-Box referred to as *Lightweight Responder (LR)* and M-Trax dubbed as *Lightweight Initiator (LI)*.

Here is a brief description of each actor:

- *Central Server (CS)*: A powerful server with unlimited resources that administers LRs and controls access between LI and LRs.
- *Lightweight Initiator (LI)*: A terminal accessing information stored in LRs. LI has enough resources in terms of computing power and storage space.
- *Lightweight Responder (LR)*: A Trax-Box or a resource constrained terminal with limited resources in terms of storage capacity, computation power and energy. LR constantly collects and stores sensitive data.

Each LR has a secret key K_C provided by CS and a static timestamp T_C initialized by CS during device setup. LRs can be RFID Tags, NFC Tags or other constrained data capturing devices. LR is the most important player in the scenario that our proposed protocol aims to protect.

In our scenario, LRs are geographically distributed in form of *clusters*. A cluster is a collection of LRs within a small geographic region sharing the same secret key K_C. The common secret key K_C is used to verify the authenticity of Lightweight Initiators LIs interacting with LR cluster. Prior to accessing LR cluster, LI securely connects to CS, sends its identifier and geographic position and requests for the authorization to access cluster(s) of LRs within its proximity. CS knows all legitimate LR clusters, secret keys for each cluster and their respective geographic positions. CS replies to LI by providing the necessary security parameters to access the respective cluster(s) of LRs. Among the parameters sent to LI by CS are LI's *key* K_L, *access rights* AR, *Time Window* W_S and a list of temporary identities L of all LRs within a cluster.

3.2.2 Features

In addition to the serverless features described in section 2.4, our proposed scenario has also the following features:

- *No prior knowledge of each other*: LI and LR have no knowledge of each other's existence prior to the authentication phase.
- *Scalability*: In a cluster, LIs and LRs freely interact, subject to the LI's access right to the respective LRs.
- *Trust relationship*: LR and LI do not have mutual trust. The trust relationship is built during the authentication phase with the help of valid information from CS.

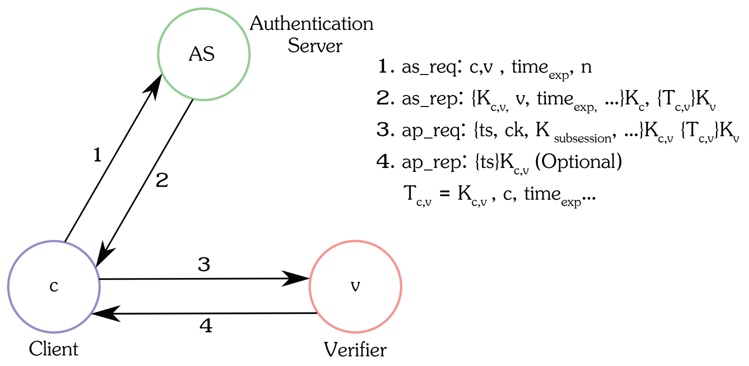
3.3 Background on Kerberos Authentication Protocol

Kerberos [65] is a token based distributed authentication protocol that allows a process (a client), running on behalf of a principal (a user), to prove its identity to a verifier (an application server) without necessarily sharing parameters beforehand [65]. Instead, the Kerberos authentication system employs a series of encrypted messages to prove to a verifier that a client is running on behalf of a particular user [96].

For clarity and simplicity, we use a simplified version of the Kerberos authentication protocol to describe its authentication mechanism as depicted in Figure 3.1. In the simplified version of Kerberos protocol, the authentication is done by involving three parties - a client c, an authentication server AS and a verifier v. In its basic implementation, Kerberos authentication server shares keys with each application server that the client has access to.

TABLE 3.1 - Notations for Kerberos protocol

Parameter name	Symbol	Parameter name	Symbol
Authentication server	AS	Checksum	ck
Client	c	Current timestamp	t_s
Service identifier (verifier)	v	Client's requested timestamp expiry	$time_{exp}$
Encryption key (between verifier & client)	$K_{c,v}$	Verifier's key (shared between v and AS)	K_v
Random number	n	Kerberos ticket	$T_{c,v}$
Subsession key	$K_{subsession}$	Session key established between x and y	$K_{x,y}$
abc encrypted using x's public key	$\{abc\}K_x$	*def* encrypted using the session $K_{x,y}$	$\{def\}K_{x,y}$



1. as_req: c,v , $time_{exp}$, n
2. as_rep: {$K_{c,v,}$ v, $time_{exp,}$ ...}K_c, {$T_{c,v}$}K_v
3. ap_req: {ts, ck, $K_{subsession}$, ...}$K_{c,v}$ {$T_{c,v}$}K_v
4. ap_rep: {ts}$K_{c,v}$ (Optional)

$T_{c,v}$ = $K_{c,v}$, c, $time_{exp}$...

FIGURE 3.1 - The basic Kerberos authentication protocol (simplified)

In Figure 3.1, a client c wanting to access a particular service v, requests for the permission from the authentication server AS by sending *message 1* containing its identifier c, the identifier of the requested service v, the requested expiration time $time_{exp}$ and the random number n as a nonce. In turn, the authentication server AS sends back *message 2* containing a session key $K_{c,v}$ to be used between client c and verifier v, the assigned expiration time $time_{exp}$, the random number from the request and name of the verifier v, all encrypted with the user's password registered with the authentication server [65,96]. In addition, the message also contains a ticket $\{T_{c,v}\}K_v$ for c to access the requested service.

The Kerberos ticket $\{T_{c,v}\}K_v$ is a ticket issued by an authentication server AS, encrypted using the verifier's key K_v. The Kerberos ticket contains similar information to that sent to the client i.e., random session key $K_{c,v}$, the name of the client c to whom the session key was issued, and an expiration time $time_{exp}$ after which the session key is no longer valid. The ticket is not sent directly to the verifier, but is instead sent to the client who forwards it to the verifier as part of the application request. The ticket is encrypted using the verifier's key, known only to the authentication server and verifier, hence a client cannot modify the ticket without detection [65].

When the client wants to access the service, it sends a Kerberos ticket to the respective application server (verifier) via *message 3* (refer to Figure 3.1). The verifier decrypts the ticket and verifies the contained information. The verifier can optionally send *message 4*. If the information is correct, the service is granted. The exchanges between c and v are secured using the session key $K_{c,v}$ generated by the authentication server. The client can use this ticket until the end of its expiry time, beyond which the client has to request for another ticket [65, 96].

3.4 Serverless Mutual Authentication Protocol

Our protocol leverages on the power of CS's knowledge on LR clusters with their respective credentials to facilitate authentication, even though CS does not actively participate during the mutual authentication phase.

Our proposed protocol adapts the principles of the basic Kerberos authentication protocol [65] in constrained environments. We are interested on how Kerberos authentication protocol incorporates context information i.e., date and time, to enforce the validity of the authentication parameters. In effect, we borrow some ideas from Kerberos which are useful to make our protocol work more reliably and efficiently in the pervasive environment.

3.4.1 Security and Privacy Requirements

The purpose of our protocol is to ensure a secure exchange of information between LI and LR, but it mostly protects the information sent by LRs. Together with the basic serverless protocol security and privacy requirements given in section 2.3, our proposed mutual authentication protocol is designed to fulfill two additional criteria.

- *Key Exchange*: Our protocol must securely establish a common key between LR and LI to be used during data exchange session. The key must be established by the respective communicating parties.
- *Parameter Expiration*: The access rights and parameters granted to LI must have a limited validity period.

3.4.2 Privacy & Security Threat Models and Games

Security and privacy games are used for supporting informal analysis of the protocol to the scheme's resistance against relevant threats. Games are modeled as exchanges between an adversary, referred to as β in the exchanges below, and a challenger. In security games, a challenger is an entity with some given knowledge of the system and is able to perform various scheme's functions.

We propose privacy and security games to model possible threats and demonstrate how resilient our protocol can be against attacks. The games described here refer to the exchanges shown in Figure 3.4.
Game 1: β masquerades as LI

- *Phase 1.1*: β eavesdrops several exchanges between one or more LR and various LIs.
- *Phase 1.2*: β sends message b_1 and then message b_3 to LR.
  β wins the game if he can reply to LR with a valid message b_3.

Game 2: β tracks LR_i

- *Phase 2.1*: β colludes with a legitimate device LI and listens to the exchanges between LI and responder LR_1 and then between LI and LR_2.
- *Phase 2.2*: Challenger selects LR_i, $i \in \{1, 2\}$, β listens to the exchanges between LI and LR_i, and β sends a guess value i to the challenger.
  β wins the game if the guessed value i is correct. The protocol is considered private if β cannot win the game with probability greater than 0.5.

Game 3: β depletes LR's resources

- *Phase 3.1*: β eavesdrops messages b_1 between LIs and LRs.
- *Phase 3.2*: β sends forged b_1 messages to a targeted LR within a cluster.
  β wins the game if he can successfully deplete LR's battery within 12 hours (This corresponds to the midnight attack [30, 80], where an attacker has only a limited time to launch an attack, beyond which he must demonstrate some improvements to attack the system).

3.4.3 Assumptions

The following assumptions are made when describing our proposed solution:

- All LRs running our protocol are capable of performing simple primitives such as *Keyed-Hash Message Authentication Code* (HMAC), comparison and XOR. In this thesis, HMAC is based on SHA1 (*Secure Hash Algorithm 1*) [36] and its output is truncated to the 128 bits (energy saving). However, our protocol works with any HMAC.
- *Pseudo Random Number Generator (PRNG)* and *HMAC* are assumed to be robust.
- CS and LI share secret parameters used to launch a secure channel, e.g. via secure protocol https, for exchanging secret information.
- CS shares a secret key K_C with each LR. K_C is common among all legitimate LRs within a specific cluster.

- *LI* uses key K_L to solicit *LR*s within a given cluster sharing the key K_C, provided W_S is still valid.

3.4.4 Protocol Notations

Table 3.2 shows notations used in our protocol. *AR* is a code for access right that *LI* has pertaining to the data stored in *LR*. In our protocol, *AR* is represented in form of a code, like Unix file permissions, with *Read, Write* and *Execute* options. H_K is a secret value from CS used to securely send initial parameters from LI to *LR*s. *LR*'s static timestamp T_C is initialized by a default timestamp value T_{init} during initial configuration by CS. Time Window $W_S = [T_0||T_Z]$ is a 64 bits parameter made of two timestamps, start timestamp T_0 and end timestamp T_Z, each with 32 bits.

TABLE 3.2 - Protocol notations with size estimations

Parameter name	Symbol	Bits	Parameter name	Symbol	Bits
LI's system Time	T_{LI}	32	LR's cluster Key	K_C	128
LR's stored timestamp	T_C	32	Derived session key	K_S	128
Start Time Window	T_0	32	Timestamp signature	H_T	128
End Time Window	T_Z	32	Random Value	R_1	128
Time Window	W_S	64	Access Rights	AR	128
LI's Identifier	ID_L	128	LI's secret code	H_K	128
LI's Key	K_L	128	List of *LR*s temporary identities	L	-
LR's identifier	Id_i	128	Concatenation Operator	$\|\|$	-
Computed value of X	X'	-	CS default timestamp initialized in the Tag	T_{init}	32

3.4.5 Protocol Description

The proposed protocol operates in two phases. *Phase A* involves an interaction between *LI* and CS, and *Phase B* involves an interaction between *LI* and *LR*.

3.4.5.1 Phase A - Authorization: Interaction Between CS and LI

LI requests authorization from CS to authenticate and access information stored in *LR*s. This process is done by establishing a secure channel. CS also uses this phase to synchronize time with LI. The LI securely sends message a_1 to CS to request for the authorization to access a cluster of *LR*s in its vicinity. Message a_1 contains *LR*'s identifier ID_L and its geographic location. The protocol exchanges are depicted in Figure 3.2.

Figure 3.3 shows how CS generates LI's authentication parameters using the secrets known to *LR*s. For instance, the list of temporary *LR*'s identities L is generated using the start time window T_0 and the real identity of each authorized *LR*s. The identities of *LR*s should remain secret; this process is meant to conceal them from LI. The parameters K_L and H_K are secrets granted to LI, both of which depend on K_C, the key for a given cluster of *LR*s. These secrets parameters, i.e., L, K_L and H_K are generated using two secret parameters known only to CS and LR, which are Id_i and K_C.

CS sends back message a_2 via the established secure channel. LI receives and decrypts message a_2 containing K_L, H_K, AR, W_S and L and store them in memory.

Central Server CS | Lightweight Initiator LI

Knows all legitimate LRs in clusters $\quad \overset{a_0:\ Secure\ Session\ Initialization}{\rightleftharpoons} \quad$ LI has an identity ID_L

$\overset{a_1:\ Request,\ ID_L}{\longleftarrow}$

a_{11}: Get a list of LRs that LI can access: $Id_1, Id_2, ..., Id_i$
a_{12}: Assign access rights relative to the LI: AR
a_{13}: Get the start and end dates that the LI is granted access to the cluster of LRs: T_0 and T_Z
a_{14}: Generate time window relative to LI's allowed access to the cluster of LRs: $W_S = \{T_0||T_Z\}$
a_{15}: Generate a temporary identity for each LR:
$\quad Id_{Temp_i} = HMAC_{Id_i}(T_0)$,
a_{16}: Compute $K_L = HMAC_{K_C}(ID_L||AR||W_S)$
a_{17}: Compute $H_K = HMAC_{K_C}(W_S)$
a_{18}: Generate a list of LRs credentials:
$\quad L = \{Id_{Temp_1}, ..., Id_{Temp_i}\}$ $\quad \overset{a_2:\ K_L, H_K, W_S, AR,\ L}{\longrightarrow}$

Receive and store:
a_{21}: List $L = \{Id_{Temp_1}, ..., Id_{Temp_i}\}$
a_{22}: Time window: W_S
a_{23}: Access Rights: AR
a_{24}: LI's Key K_L
a_{25}: LI's secret code H_K

FIGURE 3.2 - Phase A: Authorization phase between CS and LI

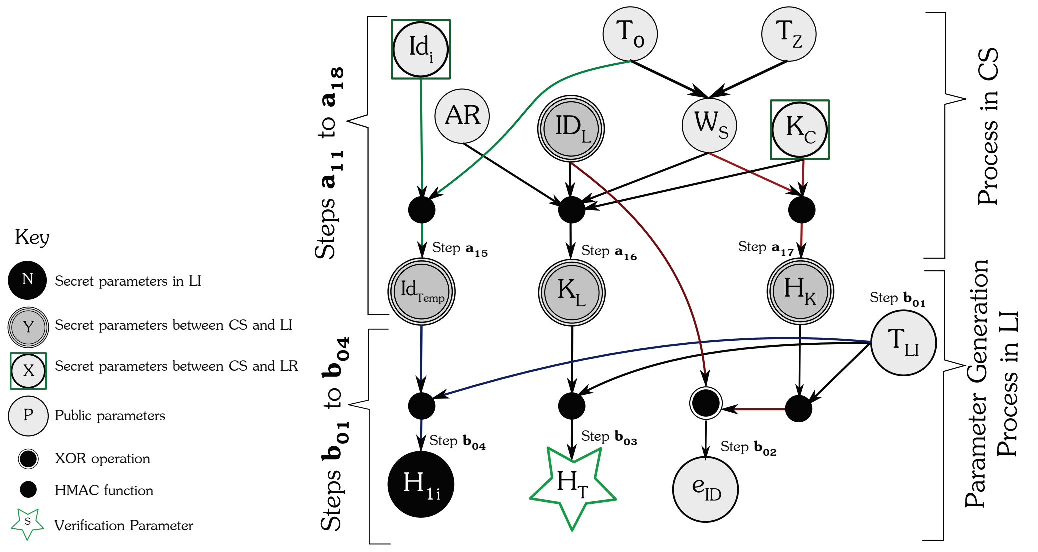


FIGURE 3.3 - Parameter Generation in CS and LI

3.4.5.2 Phase B - Mutual Authentication Between LR and LI

The mutual authentication phase between LI and LR is completed in three exchanges without the assistance from CS. The protocol exchanges are depicted in Figure 3.4.

Figure 3.5 shows how a legitimate LR in a cluster can easily verify and authenticate a valid LI by simply exchanging public parameters. LI broadcasts public parameters $W_S, H_T,\ e_{ID}$, AR, and T_{LI} via message b_1, where W_S and AR come from CS and H_T, e_{ID} and T_{LI} are generated by LI. Then, LI pre-generates the values $H_{1_i} = HMAC_{Id_{Temp_i}}(T_{LI})$ for the expected replies from the authorized LRs within the vicinity.

```
Lightweight Initiator $LI$                                                    Lightweight Responder $LR$

$LI$ knows from phase A:                                                      $LR$'s memory has:
1) List of authorized $LR$s $L = \{Id_{Temp_1}, ..., Id_{Temp_i}\}$,            1) Static timestamp: $T_C$
2) Time window: $W_S$, and                                                      2) $LR$'s identifier: $Id_i$
3) Access rights: $AR$ received from CS                                         3) Cluster Key: $K_C$
$b_{01}$ :  Get the current timestamp $T_{LI}$
$b_{02}$ :  $e_{ID} = HMAC_{H_K}(T_{LI}) \oplus ID_L$
$b_{03}$ :  $H_T = HMAC_{K_L}(T_{LI})$
$b_{04}$ :  Calculate $H_{1_i} = HMAC_{Id_{Temp_i}}(T_{LI})$ for all $Id_{Temp_i}$ in $L$

                    $b_1$: $H_T$, $e_{ID} \xrightarrow{,AR}$ , $W_S$, $T_{LI}$

                         $b_{101}$ :  $if(\ T_Z < T_C\ or\ T_Z < T_{LI}\ or\ T_{LI} < T_C\ or\ T_{LI} < T_0)\{$
                         $b_{102}$ :            END SESSION
                                      } else {
                         $b_{103}$ :        $H'_K = HMAC_{K_C}(W_S)$
                         $b_{104}$ :        $ID'_L = HMAC_{H'_K}(T_{LI}) \oplus e_{ID}$
                         $b_{105}$ :        $K'_L = HMAC_{K_C}(ID'_L||AR||W_S)$
                         $b_{106}$ :        $H'_T = HMAC_{K'_L}(T_{LI})$
                         $b_{107}$ :        $if(H'_T == H_T)\{//$**LI's Key $K_L$ is valid**
                         $b_{108}$ :            $T_C = T_{LI}$
                                            } else {
                         $b_{109}$ :            END SESSION
                                            }
                         $b_{110}$ :        $Calculate\ Id_{Temp_i} = HMAC_{Id_i}(T_0)$
                         $b_{111}$ :        $H_1 = HMAC_{Id_{Temp_i}}(T_{LI})$
                         $b_{112}$ :        Generate random $R_1$
                         $b_{113}$ :        $e_1 = R_1 \oplus Id_{Temp_i}$
                         $b_{114}$ :        $H_2 = HMAC_{Id_{Temp_i}}(R_1||K'_L)$
                                      }

                    $b_2$: $e_1 \xleftarrow{,\ H_1}$, $H_2$

$b_{201}$ :   Search for $H_1$ in the list of $H_{1_i}$
            if $(H_1 \notin H_{1_i})\{$
$b_{202}$ :     QUIT
            } else {
$b_{203}$ :     $R'_1 = e_1 \oplus Id_{Temp_i}$
$b_{204}$ :     $H'_2 = HMAC_{Id_{Temp_i}}(R'_1||K_L)$
$b_{205}$ :     $if(H'_2 \neq H_2)\{$
$b_{206}$ :         END SESSION
$b_{207}$ :     } else {  //**LR is authentic**
$b_{208}$ :         $H_R = HMAC_{Id_{Temp_i}}(R'_1)$
$b_{209}$ :         $K_S = HMAC_{Id_{Temp_i}}(R'_1||T_{LI})$
                                        $b_3$: $\xrightarrow{H_R}$
                }
            }           $b_{301}$ :        $H'_R = HMAC_{Id_{Temp_i}}(R_1)$
                        $b_{302}$ :        $if(H'_R == H_R)$  //**LI is authentic**
                        $b_{303}$ :            $K_S = HMAC_{Id_{Temp_i}}(R_1||T_{LI})$
```

FIGURE 3.4 - Mutual authentication between LI and LR

Upon the receipt of message b_1, each LR in the vicinity can verify and recover all necessary values using its two secret parameters i.e., K_C and Id_i. These values are recovered as follows:

Step b_{103}**:** LR computes $H'_K = HMAC_{K_C}(W_S)$ using the cluster key K_C and the time window W_S.

Step b_{104}**:** LR recovers the identity of LI as $ID'_L = HMAC_{H'_K}(T_{LI}) \oplus e_{ID}$.

Step b_{105}**:** LR computes LI's key $K'_L = HMAC_{K_C}(ID'_L||AR||W_S)$.

Step b_{106}**:** LR computes $H'_T = HMAC_{K'_L}(T_{LI})$ using the received time and computed

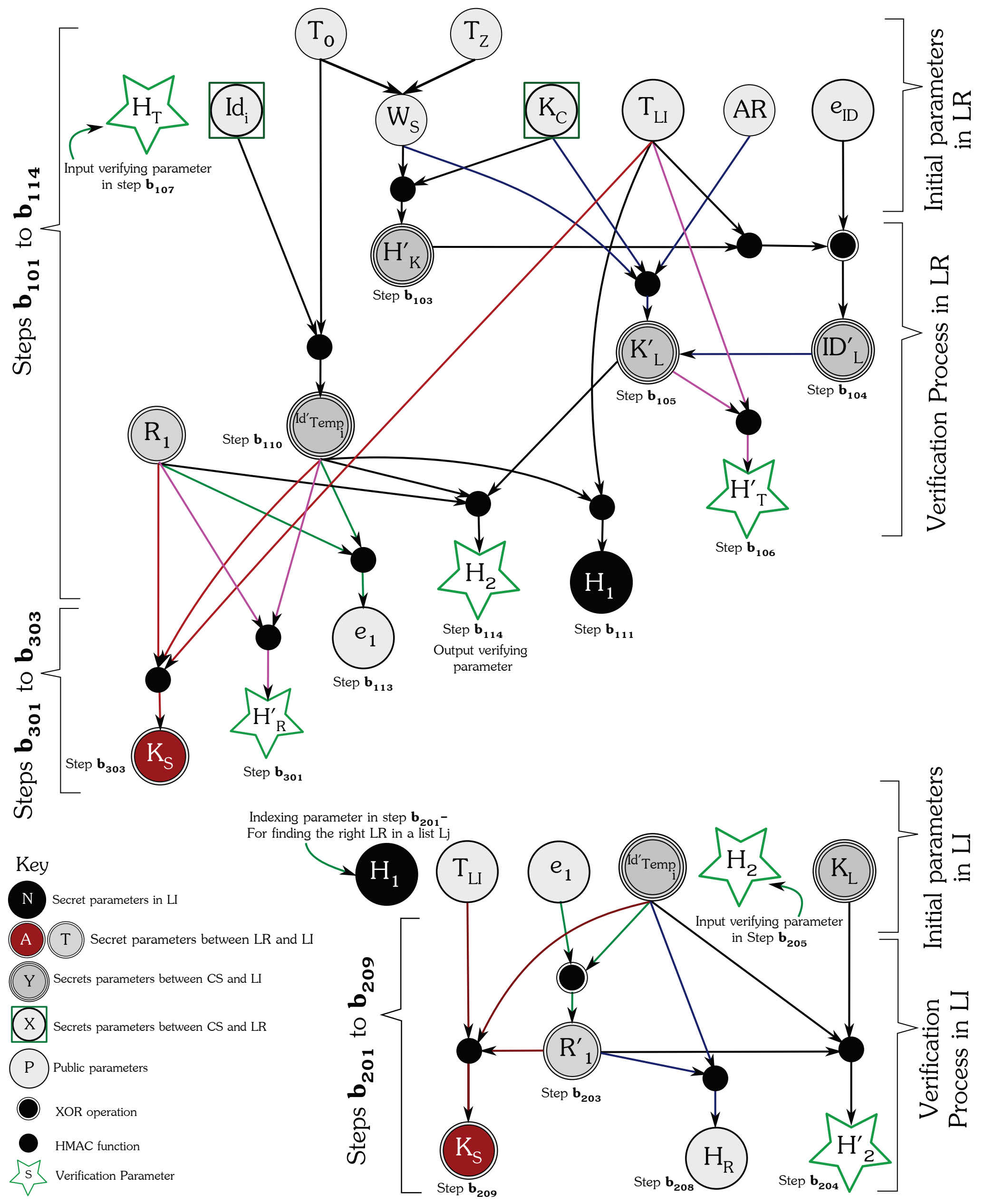

Steps b101 to b114
Steps b301 to b303
Steps b201 to b209
Initial parameters in LR
Verification Process in LR
Initial parameters in LI
Verification Process in LI
Input verifying parameter in step b107
Output verifying parameter
Indexing parameter in step b201 - For finding the right LR in a list Lj
Input verifying parameter in Step b205
Step b103
Step b104
Step b105
Step b106
Step b110
Step b111
Step b113
Step b114
Step b301
Step b303
Step b203
Step b204
Step b208
Step b209
Key
Secret parameters in LI
Secret parameters between LR and LI
Secrets parameters between CS and LI
Secrets parameters between CS and LR
Public parameters
XOR operation
HMAC function
Verification Parameter

FIGURE 3.5 - Parameter Verification & Authentication in Phase B between LI and LR

K_L.
Step b_{107}**:** *LR* verifies the validity of *LI* by comparing the computed H'_T against the received H_T. Their equivalence confirms that the key K_L is valid. Otherwise, *LR*s will terminate the session and no reply will be sent.

The processes in steps b_{103} to b_{109} are common to all *LR*s within the same cluster and sharing the same cluster key K_C. Then, each individual *LR* computes values using the secret identity Id_i, which is only known to itself. The process goes as follows:

Step b_{110}**:** LR computes $Id_{Temp_i} = HMAC_{Id_i}(T_0)$ using start time window T_0 and its identity Id_i.
Step b_{111}**:** LR calculates $H_1 = HMAC_{Id_{Temp_i}}(T_{LI})$, which will act as an identification parameter on LI.
Step b_{113}**:** LR computes $e_1 = R_1 \oplus Id_{Temp_i}$ using the generated random R_1 and its concealed identity Id_{Temp_i}.
Step b_{114}**:** LR computes $H_2 = HMAC_{Id_{Temp_i}}(R_1||K'_L)$.

The values e_1, H_1 and H_2 are sent to LI via message b_2. Upon receipt of message b_2, LI only compares the value H_1 against the values found in the table of H_{1_i}. If the value is not found, LI rejects the respective message. Otherwise, LI recovers the values of the corresponding LR as follows:
Step b_{203}**:** LI computes $R'_1 = e_1 \oplus Id_{Temp_i}$ using Id_{Temp_i} in the list L and the received e_1.
Step b_{204}**:** LI computes $H'_2 = HMAC_{Id_{Temp_i}}(R'_1||K_L)$ and uses it to verify the validity of H_2. If H_2 is not valid, LI ignores the message from the corresponding LR, else it proves its authenticity.
Step b_{208}**:** LI computes $H_R = HMAC_{Id_{Temp_i}}(R'_1)$ and sends it to LR via message b_3
Step b_{209}**:** LI computes the key $K_S = HMAC_{Id_{Temp_i}}(R'_1||T_{LI})$.

Upon the receipt of message b_3, LR verifies it and then calculates the session key $K_S = HMAC_{Id_{Temp_i}}(R_1||T_{LI})$ in **Step** b_{303}.

The processes depicted in Figures 3.3 and 3.5 show how LI and LR manage to authenticate one another using only public values and without needing to exchange or share parameters beforehand. The scheme works more like zero-knowledge authentication because LR proves that it knows K_C, and later Id_i, without divulging them to LI. These secret parameters are used by CS to generate the parameters granted to LI. Thus, the respective authenticating devices need only to establish an association with the trusted party, which is CS in this case, and use this association to complete the authentication process between them.

3.4.6 Advantages of the Proposed Protocol

In addition to the advantages of serverless protocols outlined in section 2.1, here are more advantages specific to our proposed solution:

- **Secrecy Preservation**: The protocol concentrates on securing the secrets (i.e., secret key and identifiers in each communicating party) as the communicating parties use public values from the server to exchange secrets and validate one another. Likewise, LI is forced to authenticate to the CS in the future before soliciting the same cluster of LRs.
- **Spontaneous Communication**: The protocol does not require that the communicating parties have prior relationships with each other, nor to have previously shared any authentication information with each other. Instead, all parties must establish

prior relationships with the centralized server and rely upon it to verify credentials and authorize sessions.

- **Reader Compromise Resistance**: Our authentication protocol is resistant to cloning and reader compromise attacks because CS grants LI with authentication parameters with a defined period of validity, beyond which they cannot be used to perform authentication with LRs. This ensures that LRs remain secure even after LIs are compromised or physically attacked. The adversary cannot use the information stored in LI to generate counterfeit LRs and fool other legitimate LIs.

3.4.7 Comparing our Proposed Protocol & Kerberos

Our proposed protocol is based on the token concept like Kerberos, where the Lightweight Initiator LI (equivalent to a client in Kerberos) requests for access to the cluster of Lightweight Responders LR (equivalent to an application or a service in Kerberos) from the central server CS (equivalent to the authentication server in Kerberos). We summarize the comparison between our proposed solution and Kerberos in Table 3.3.

As depicted in Table 3.3, our proposed protocol and Kerberos have a lot of features in common. Both our protocol and Kerberos protect against spoofing using cryptographic mechanisms, where Kerberos uses an encrypted ticket with the verifier's key while our protocol uses a Hashed Message Authentication Code (HMAC) over the values transmitted to the LI generated using the cluster key, to avoid LI from tampering with CS generated parameters. Likewise, both protocols support mutual authentication between the client (or LI) and verifier (or LR) and also employ date and time to thwart replay attacks. However, there is a major difference in the use of time and date because, contrary to Kerberos, the devices used in our proposal are constrained in resources and do not have embedded clocks, hence statically manage time and date parameters.

Table 3.3 - Comparing Kerberos Authentication Protocol vs our proposed protocol (✓: Included; ✗: Not included)

Property	Basic Kerberos Protocol	Our Proposed Protocol
Automatic parameter expiration	✓	✓
Timestamps to prevent replay attacks	✓(Uses real clocks)	✓(Memorizes latest timestamps)
Mutual Authentication	✓	✓
Protection against spoofing	Uses ticket encrypted with verifier's key	Uses HMAC generated using cluster's key
Session key generation	Authentication server generates the session key for both client and verifier	Each LR and LI participate in the generation of a unique session key known only to the respective LR and LI
Number of services per ticket (token)	One ticket for each verifier	Many (One token for a cluster)
Mode of encryption used	Asymmetric	Symmetric
Targeted devices	For powerful servers	For resource constrained devices
Assumption	Clocks are synchronized	Clients memorized timestamps are not compromised

Despite the fact that Kerberos was designed for powerful devices with ample resources and our protocol is designed for devices with strict resource constraints, there are other distinguishing features between our proposed protocol and Kerberos. First, unlike Kerberos, our protocol does not encrypt security parameters to be sent to the cluster of LRs via LI. This is done intentionally to reduce the computation demands on the resource constrained LR, especially the information is exchanged in the moment when none of the parties trust each other. In turn, only public parameters are exchanged and they can be independently verified by any legitimate LR within the respective cluster. In our proposed protocol, the token (similar to the ticket in Kerberos) is not generated for one particular device, but rather for all devices within the same cluster that share the same cluster key K_C. Second, unlike Kerberos, the authentication server CS in our protocol does not generate a session key between the cluster members LRs (similar to the verifier in Kerberos) and the Lightweight Initiator LI (similar to the client in Kerberos). The session key is uniquely generated between each cluster member LR and LI after the mutual authentication phase, hence only known to the two of them.

3.5 Protocol Validation

Our proposed protocol was validated using the *Automated Validation of Internet Security Protocols and Applications (AVISPA)* [14]. AVISPA is a widely used tool in evaluating the security aspects of Internet protocols, hence we used it for validating the resistance of our protocol against the common security attacks. The attacker model used in AVISPA validation is *Dolev-Yao*, described in section 2.2.1.

In Table 3.4, AVISPA outputs *SAFE* from three of its back-ends i.e., *OFMC, CL-AtSe*, and *SATMC*, which implies that AVISPA could not reproduce any attack on our protocol. The TA4SP back-end does not perform verification, hence gives $INCONCLUSIVE$ result. Nevertheless, our protocol is still safe. The *High Level Protocol Specification Language* (HLPSL) [104] code for LI and LR used in AVISPA are presented in Listings A.1 and A.2, respectively (Found in Appendix A).

Appendix A gives more details on AVISPA.

Table 3.4 - AVISPA validation results

AVISPA Engine	Result
OFMC	SAFE
CL-AtSe	SAFE
SATMC	SAFE
TA4SP	INCONCLUSIVE

3.6 Protocol Analysis

This section presents the analysis of our protocol compared to other serverless protocols, described in section 2.5. Then, we analyze the performance of our protocol in section 3.6.1 and finalize with security and privacy analysis in section 3.6.2.

3.6.1 Performance Analysis

Our proposed protocol is compliant to the serverless paradigm and is designed for resource constrained pervasive devices. To analyze its performance, we compare it to other serverless protocols namely Tan et al.'s protocol, discussed in section 2.5.1.1.1, and Jialiang et al.'s protocol, described in section 2.5.1.4.1 by analyzing the computational, storage and communication costs. For the sake of comparison of the protocols, we use the same sizes for the parameters in our protocol, Tan et al.'s protocol and Jialiang et al.'s protocol. Table 3.5 summarizes the comparison between these protocols.

Table 3.5 - Performance comparison between our protocols with existing serverless protocols

	PROTOCOLS		
Resource cost	Tan et al [99]	Jialiang et al [51]	Our proposed protocol
Computation	+	+	++ (with HMAC) + (with hash)
Storage	+++	++	+
Communication	++	+	+

1. Computational Cost: Our proposed protocol uses simple primitives such as comparison, XOR and HMAC. The HMAC demands more resources than a normal Hash function but guarantees optimal security by reducing the number of collisions compared to a normal Hash function [53]. In contrast, Tan et al.'s protocol, discussed in section 2.5.1.1.1, and Jialiang et al.'s protocol, described in section 2.5.1.4.1, use hash functions. In our proposed protocol, any secure hash can be used instead of HMAC to trade-off between efficiency and security.

2. Storage Cost: We compare the storage demands during the peak times in each protocol to understand the maximum storage space a protocol may require from the device.

Our protocol stores few parameters, K_C, Id and T_C amounting to 288 bits (36 bytes) in LR. During runtime, our protocol requires a maximum of 736 bits (92 bytes) of storage in LR (corresponds to step b_{105} of Figure 3.4). On the other hand, Tan et al.'s protocol requires a total of 928 bits (116 bytes) of storage at the peak time, corresponding to step y_{12} of their protocol sequence as presented in section 2.5.1.1.1. Jialiang et al.'s proposal requires 896 bits (112 bytes) of memory at the peak time, corresponding to step 2.8 of their protocol (see section 2.5.1.4.1). As a consequence, our protocol requires fewer memory space for storing parameters compared to Tan et al. [99] and Jialiang et al. [51].

3. Communication Cost: Constrained devices expend a lot of energy in transmitting and receiving information [60], [69], hence fewer and shorter messages reduce energy consumption. The three protocols under comparison give the same amount of exchanged data, that is 384 bits (48 bytes) but note that energy costs are different.

That is, the LR in our protocol and the tag in Jialiang et al.'s protocol transmit only one message with a total of 384 bits (48 bytes). However, Tan et al. [99] transmits a total of 384 bits (48 bytes) of data in two separate messages, which is more expensive in terms of energy.

3.6.2 Security and Privacy Analysis

In this section we analyze the security and privacy properties of our protocol. First, we compare the security and privacy properties of our proposed protocol with similar protocols based on hash functions (refer to section 2.5). The comparison is depicted in Table 3.6. Second, we analyze our protocol against the threat models put forth in Section 3.4.2.

Table 3.6 - Comparing the security and privacy properties of the our proposed serverless authentication protocol with the existing protocols (✓: Supported; ✗: Not Supported)

	SECURITY & PRIVACY CRITERIA									
Protocol	Mutual Authentication	Tag Untraceability	Reader Untraceability	Tag Anti-Cloning	Reader Anti-Cloning	Protect Tag's Identity	Protect Reader's Identity	Anti-Replay	Reader Compromise Resistance	Desynchronization Resistance
Tan et al.'s [99]	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓
Lee et al. [57]	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗
Lin et al. [59]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Jialiang et al. [51]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Abdolmaleky et al. [6]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
Our Protocol [62]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.6 gives a brief comparison between our protocol and other similar existing protocols. Protocols suggested by Tan et al. [99] and Lee et al. [57] broadcast the reader's identity as clear text when querying the tag, hence traceable and an adversary can easily track a reader at any time. Moreover, Tan et al.'s [99], Lee et al.'s [57], Lin et al.'s [59] and Jialiang et al.'s [51] protocols do not offer mutual authentication between tag and reader, which reduces the level of trust and security of the protocol. It can also be observed that our proposed authentication protocol satisfies all the necessary security and privacy requirements listed in Table 3.6. Moreover, it is only our protocol that is resistant to the reader's compromise attack as the parameters granted to the reader are ephemeral and expire after a given period. The rest of the protocols grant parameters that stay valid forever.

Analysis of Security Games
The analysis using security games complement AVISPA's validation as it analyzes the mechanisms of the exchanged information in details.

Game 1: *β masquerades as LI*; Referring to Game 1 in Section 3.4.2, β's objective is to send valid messages b_1 and b_3. That is, β can either try to crack the key K_L along with ID_L or generate a valid message b_3 based on previously sniffed messages b_2 and b_3 during Phase 1.1.

After cracking key K_L along with ID_L, β is able to spoof LI. One of the solutions involves extracting the values of H_T and T_{LI} from a known message b_1 and then cracking the key K_L. This assumes that the HMAC function is not robust to collision attacks, which is contrary to our assumption of Section 3.4.3. Alternatively, β can combine messages b_1, b_2 and b_3 and try to deduce valuable information. However, all messages behave like random or pseudo-random strings. Indeed, e_1 is randomized thanks to R_1. Values e_{ID}, H_T, H_2 and b_3 are HMAC outputs and, as stated in [39], they behave as pseudo-random strings and evolve independently from each other as their inputs are different. As such, whatever the number of sniffed messages b_1, b_2 and b_3 , β can neither extract useful information nor win the game.

Game 2: *β tracks LR*; Following *Game* 2 in Section 3.4.2, LR_i responds for session j with messages e_{11j} and e_{12j} which behave as random or pseudo-random strings. Such that, any response from LR_1 is semantically indistinguishable from responses of LR_2, and previous responses of LR_1. Hence, an adversary β is unable to guess with a probability greater than 0.5 which LR_i sent message *b2*.

Game 3: *β depletes LR's resources*; *Game* 3 in Section 3.4.2 is a form of *Denial of Service (DoS)* attack such that an adversary β constantly queries LR to utilize its resources and deplete its energy source. Our protocol tests message b_1 using *five comparison operations, four HMAC operations*, and *one XOR operation* to verify validity. But HMAC consumes more energy as explained in section 3.6.1.

Now, let us quantify the duration of time needed for β to deplete an alkaline long-life *AAA* battery with total energy of 5071 Joules [46]. If LR conforms to IEEE 802.15.4 [**?**] with an antenna frequency of 2.4 GHz band, data rate of 250 kbps, and power consumption of 1.475W in receive mode [24], then it will take approximately 2 ms (milliseconds) to receive 480 bits of data sent in message b_1 by dissipating 1.475*0.002 = 0.003 Joules. According to [76], HMAC function consumes 1.16μJ (microjoules) per byte of data. Parameters used in calculations have a total of 960 bits or 120 bytes for H'_K in step b_{103}, and ID'_L (step b_{104}), K'_L (step b_{105}), and H'_T (in step b_{106}). As such, each request from β costs 1.16*120 = 139.2μJ, which makes a total of 0.003 Joules for receiving and calculations. At this rate of consumption, it will take 1,706,718 rounds to deplete the battery. Suppose β sends message b_1 to LR after every 1 second, it will take around 20 days to deplete the battery, with most of the energy being spent in receiving message b_1. Hence this game cannot succeed.

3.7 Conclusion

This chapter presents our first contribution, a lightweight serverless authentication scheme for resource constrained pervasive devices. The proposed solution is token based and uses timestamps to limit the validity of the authentication parameters between communicating devices. This idea is inspired by Kerberos authentication protocol. The originality of our proposed solution is based on the fact that it does not require the communicating parties to have prior relationships with each other, nor to have previously shared any authentication information with each other. Instead, all parties must establish prior relationships with the centralized server and rely upon it to verify credentials and authorize sessions. We also outline the advantages that our solution offers to the pervasive systems.

In order to test our proposed solution, a thorough analysis is done to theoretically verify the security, privacy and performance properties of the protocol. These are done in two ways. First, we design informal security and privacy games to verify the defined security and privacy properties. Second, we formally validate the properties our proposed solution using the AVISPA tool.

The next chapter presents two contributions, one is a mutual authentication protocol and the other is a secure search protocol for resource constrained devices. The proposed protocols are serverless and lightweight. Moreover, the proposed protocols complement each other and are useful in improving the efficiency of any pervasive technology facing similar challenges.

Chapter 4

Secure Serverless Search & Authentication Protocols for very Constrained Devices

Securing a computer system has traditionally been a battle of wits: the penetrator tries to find the holes, and the designer tries to close them.

- Gosser

Contents

4.1 Introduction

This chapter focuses on the security and privacy challenges in the domain of Radio-Frequency Identification (RFID) technologies. This chapter responds to the challenges put forth in *scenario C*, described in section 1.3.2.3, by proposing serverless authentication and search schemes that simultaneously provide both resource efficiency and robust security.

We propose two robust and efficient complementing solutions to solve two core problems in the RFID domain. The serverless authentication scheme solves the problem of mass authentication between legitimate RFID readers and tags. The search scheme enables an RFID reader to efficiently and securely search for a specific tag among a huge number of them. The secure RFID tag search scheme is a relatively new but interesting research problem as it helps to improve the efficiency and reduce the resource demands during the search process without revealing the secret information to the adversaries. The issues addressed in this chapter serve to improve the efficiency in the inventory domain or areas requiring similar functionalities.

The relevant state-of-the art for this chapter is found in section 2.5. The majority of the existing protocols are vulnerable once the respective devices are compromised. The drawbacks of the existing authentication protocols as expressed in section 2.5.6.1 and those of serverless search protocols as explained in section 2.5.6.2 are addressed by our proposed protocols in this chapter.

The protocols proposed in this chapter address the key security and privacy challenges facing the existing protocols. Our protocols enforce authentication parameters revocation and also ensure privacy between the communicating parties by using temporary instead of the actual values during the mutual authentication phase. In turn, our protocols are resistant to the reader compromise and cloning attacks.

The rest of this chapter is organized as follows, section 4.2 presents actors and features relevant to the proposed solutions while section 4.3 lays the foundation by presenting the system model and assumptions for our proposed protocols. Section 4.4 presents a secure serverless RFID authentication protocol while section 4.5 presents a secure RFID tag search protocol. Section 4.6 presents the validation and verification of the protocol while section 4.7 concludes the chapter.

4.2 Actors & Features

This section focuses on *scenario C* described in section 1.3.2.3. The scenario involves the communication between the Trax-Boxes, or M-Traxes, with sensors. Here, we further elaborate on the actors and their features before proposing suitable security protocols to answer the corresponding challenges.

4.2.1 Actors

Our protocols involve three actors as presented below with their respective characteristics. The parameter used in our protocols are given in Table 4.1.

- **Backend Server**: denoted as S is a trusted, powerful entity with unlimited resources. S has a list of all legitimate tags and readers, hence it plays a role of assigning parameters to readers for accessing authorized tags. It is worth noting that the server is offline when the reader is launching an authentication session with the tags.

- **RFID Reader**: denoted as R_j, has finite resources for storage, computation and communication. R_j stores a list of tags L_j, which is a list of authorized tags that a reader r_j can authenticate and exchange information with. The list L_j comes from the backend server S. The reader R_j implements common primitives used in the protocol such as *Keyed Hash Message Authentication Code (HMAC)*, concatenation ($||$), and *Pseudo-Random Number Generator (PRNG)* functions. The reader R_j remains untrusted by the tags until the mutual authentication phase is successfully completed.

- **RFID Tag**: denoted by ρ_i, has scarce resources in terms of storage, computation, energy and communication. The tag ρ_i has a unique identifier id_i, which serves as a secret key shared with the backend server S. Also, the tag ρ_i has a static timestamp T_{SYS}, which is initialized at the time of tag's manufacture and does not need to be tag-unique. ρ_i implements *Keyed Hash Message Authentication Code (HMAC)*, Pseudo-Random Number Generator (PRNG), comparison and concatenation ($||$) operations used in our protocol. Tags considered in this chapter are cheap and do not possess any tamper-resistant mechanisms i.e., do not have secure memory to store secret information.

4.2.2 Features

Apart from the serverless protocol features described in section 2.4, our proposed protocols have the following additional features:

- **No prior knowledge of each other**: RFID tags and readers have no knowledge of each other's existence prior to the authentication phase.
- **Scalability**: The RFID readers and tags can freely interact provided that the RFID reader is authorized to communicate with the respective tag(s).

- **Trust relationship**: The RFID tags and readers do not have mutual trust. The trust relationship is built during the authentication phase with the help of valid information from the central backend server.

4.3 The Proposed System Model and Assumptions

In addition to the requirements of serverless protocols proposed in section 2.3, this section outlines the requirements, assumptions, threat models and specifications for each player involved in our proposed protocols.

4.3.1 Security and Privacy Requirements

In addition to the serverless protocol requirements given in section 2.3, our proposed mutual authentication protocol is designed to fulfill two more important requirements:

- Preserve tag's privacy: The central server must not divulge tag's private information to the reader i.e., the reader is only given minimum temporary information helpful to facilitate the authentication with the respective tags within a given session.
- Enforce parameter expiration i.e., the authentication information granted to the reader must have a limited period of validity. This forces the reader to periodically request for new credentials from the central server before communicating with similar tags.

4.3.2 Assumptions

The proposed protocols work under the following assumptions:

- RFID tags running our protocols are cheap but have the capability to execute basic primitives such as concatenation ($||$), addition, comparison, and Keyed - Hash Message Authentication Code *(HMAC)*.
- The backend server is a trusted entity and cannot be compromised.
- Pseudo Random Number Generator *(PRNG)* and *HMAC* are considered as robust.
- The backend server synchronizes time with the reader during the initialization phase. This ensures that the corresponding tags are always updated with relatively accurate timestamps parameters.
- The channel between the backend server and RFID reader is fully secure.

4.3.3 Protocol Notations

The protocol notations are given in Table 4.1, where AR_{ij} represents an encoded access right for the reader R_j to access data stored in tags $temp_{ij}$. AR_{ij} is represented in the form of a code, like the Unix file permissions, with *Read, Write* and *Execute* options. Time window W_{S_j} is a 64 bit parameter represented as $[T_{0_j}||T_{Z_j}]$, where T_{0_j} is the start date and T_{Z_j} is the end date defining the time limits for the reader R_j to access the tags within the list L_j.

Table 4.1 - Protocol notations with size estimations, where i represents tag's parameters and j represents reader's parameters

Parameter name	Symbol	Bits	Parameter name	Symbol	Bits
Tag's Static Timestamp	T_{SYS}	32	Tag's Identifier	id_i	128
Reader's Timestamp	t_j	32	Temporary Tag's Identifier	$temp_{ij}$	128
Start date	T_{0_j}	32	Tag's Key	K_{ij}	160
End date	T_{Z_j}	32	HMAC: Tag -> Reader	H_{ij}	160
Time window	W_{S_j}	64	HMAC: Reader -> Tag	V_{ji}	160
Acces Rights	AR_{ij}	128	List of authorized tags	L_j	-
Tag's Random Number	r_i	128	Concatenation operator	$\|\|$	-
Reader's Random Number	r_j	128	Computed value of X	X'	-
Session Key	K_S	128			-

4.3.4 Attack and Threat Models

This section details some of the security and privacy models that a polynomial time adversary α may use to gain access to secret information or disrupt normal protocol run. We have designed privacy and security games to show adversary's capabilities, limitations and options while trying to break the protocol. The games described hereafter apply to the proposed protocols depicted in Figures 4.3 and 4.5.

Game 1*: α masquerades as a Reader*

- **step 1.1**: α observes and eavesdrops several exchanges between R_j and one or more tags.
- **step 1.2**: α sends *messages* d_1 and d_3 (respectively, only *message* e_1 for the tag search protocol) to tag ρ_i.
  α wins the game if he can send valid *message* d_3 (or *message* e_1 for the search protocol).

Game 2*: α creates a new counterfeit Tag ρ_x*

- **step 2.1**: α physically attacks ρ_i's to access its data.
- **step 2.2**: α uses the data from valid ρ_i to create other counterfeit tags ρ_x where $x \neq i$.
  α wins the game if he can create counterfeit tag ρ_x and fool legitimate reader R_j.

Game 3*: α tracks tag ρ_i*

- **step 3.1**: α is able to observe exchanges between legitimate R_j and tags ρ_1 and ρ_2, one after the other, for a polynomial number of times each.
- **step 3.2**: The challenger selects a tag ρ_i, $i \in \{1, 2\}$, and let it authenticate to R_j. α listens to the exchanges and sends a guessed value i to the challenger.
  α wins the game if value i is correct. The protocol is considered private if α cannot win the game with a probability greater than 0.5.

4.4 Serverless RFID Authentication Protocol

Our proposed protocol takes place in two distinct phases namely *authorization phase* and *mutual authentication phase* as shown in Figure 4.1.

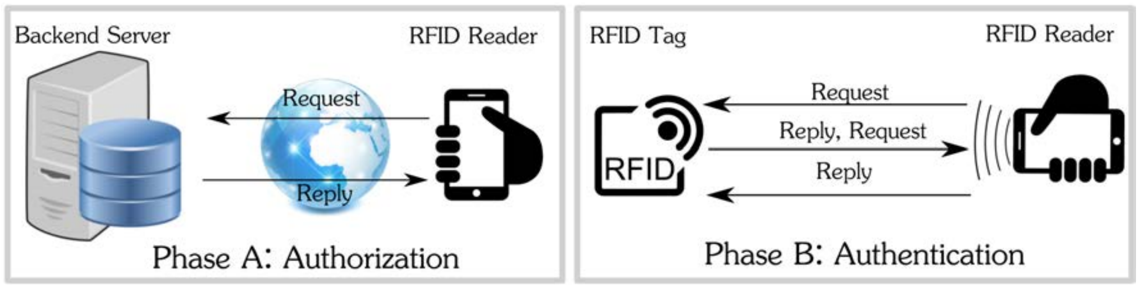


FIGURE 4.1 - The communication phases for the proposed serverless RFID authentication protocol

The *authorization phase*, depicted in Figure 4.2, involves the exchange between the *Reader* R_j and the *Backend Server S* through a secure channel, where the *Reader* R_j acquires appropriate access rights from the server to access a group of tags.

The *mutual authentication phase*, depicted in Figure 4.3, involves verification and authentication between the *Reader* R_j and the *Tag* ρ_i with the purpose of guaranteeing the authenticity of readers and tags during communication and exchange of secret data.

4.4.1 Phase A: Authorization between Backend Server and Reader

Server S		Reader R_j
Knows all legitimate tags and readers	$\overset{c_0:\ Secure\ Session\ Initialization}{\rightleftharpoons}$	The reader has an identity R_j
	$\overset{c_1:\ Request,\ R_j}{\longleftarrow}$	

c_{11}: Get the list of tags that R_j can access: $id_1, id_2, ..., id_i$
c_{12}: Assign access rights relative to the reader R_j: AR_{ij}
c_{13}: Get the start and end dates that the reader is granted access to tags: T_{0_j} and T_{Z_j}
c_{14}: Generate time window relative to R_j's allowed access to given tags: $W_{S_j} = \{T_{0_j} || T_{Z_j}\}$
For each tag ρ_i generate:
  c_{15}: Temporary identity: $temp_{ij} = HMAC_{id_i}(T_{0_j})$,
  c_{16}: Temporary Key: $K_{ij} = HMAC_{id_i}(W_{S_j} || AR_{ij})$
Then generate a list of tags credentials:
c_{17}) $L_j = \{(temp_{1j}, K_{1j}), ..., (temp_{ij}, K_{ij})\}$ $\quad \overset{c_2:\ Send\ L_j, W_{S_j}\ and\ AR_{ij}}{\longrightarrow}$

Receive and store:
c_{21}: List $L_j = \{(temp_{1j}, K_{1j}), ..., (temp_{ij}, K_{ij})\}$,
c_{22}: Time window: W_{S_j},
c_{23}: Access Rights: AR_{ij}

FIGURE 4.2 - Phase A: Authorization between the backend RFID server and reader supporting authentication and access rights assignment

During the authorization phase, depicted in Figure 4.2, the *Reader* R_j requests the permission to access several identified tags from the *Server S* via message C_1. The channel between S and R_j is fully secure. Then, S performs the following operations:

- **1**: S generates a key K_{ij} and a temporary identity $temp_{ij}$ corresponding to each tag ρ_i that the *Reader* R_j is authorized to access to with the given access rights AR_{ij}. The key K_{ij} and identity $temp_{ij}$ of each *Tag* ρ_i are ephemeral and derived from the time window W_{S_j} and start date T_{0_j} generated by S, respectively.

$$K_{ij} = HMAC_{id_i}(W_{S_j}||AR_{ij}) \tag{4.1}$$

$$temp_{ij} = HMAC_{id_i}(T_{0_j}) \tag{4.2}$$

- **2**: S builds a list of authenticated tags L_j granted to R_j for a given time window W_{S_j} with access rights AR_{ij}. S is assumed to assign different time windows W_{S_j} and AR_{ij} to different readers R_j.

$$L_j = \{(temp_{1j}, K_{1j}), (temp_{2j}, K_{2j}), ..., (temp_{ij}, K_{ij})\} \tag{4.3}$$

- **3**: S securely sends L_j, AR_{ij}, and W_{S_j} to the reader R_j via message C_2.

4.4.2 Phase B: Mutual Authentication between RFID Reader and Tag

The mutual authentication phase of our proposed protocol, depicted in Figure 4.3, involves verification, authentication and session key generation between reader R_j and tag ρ_i as presented in Figure 4.3. The description is made for each independent step in the following subsections.

4.4.2.1 Mutual Authentication

The reader R_j broadcasts a message d_1 containing W_{S_j}, AR_{ij}, and r_j. All tags within the vicinity of R_j responds with a challenge containing r_i and $H_{ij} = HMAC_{K'_{ij}}(r_i||r_j)$, where r_i is the random number generated by a respective tag. Upon receipt of message d_2 from multiple tags, R_j calculates $H'_{ij} = HMAC_{K_{ij}}(r_i||r_j)$ using the values of K_{ij} in the list L_j. If the corresponding value of H_{ij} is found, R_j authenticates the respective tag and replies with V_{ij} and t_j via message d_3.

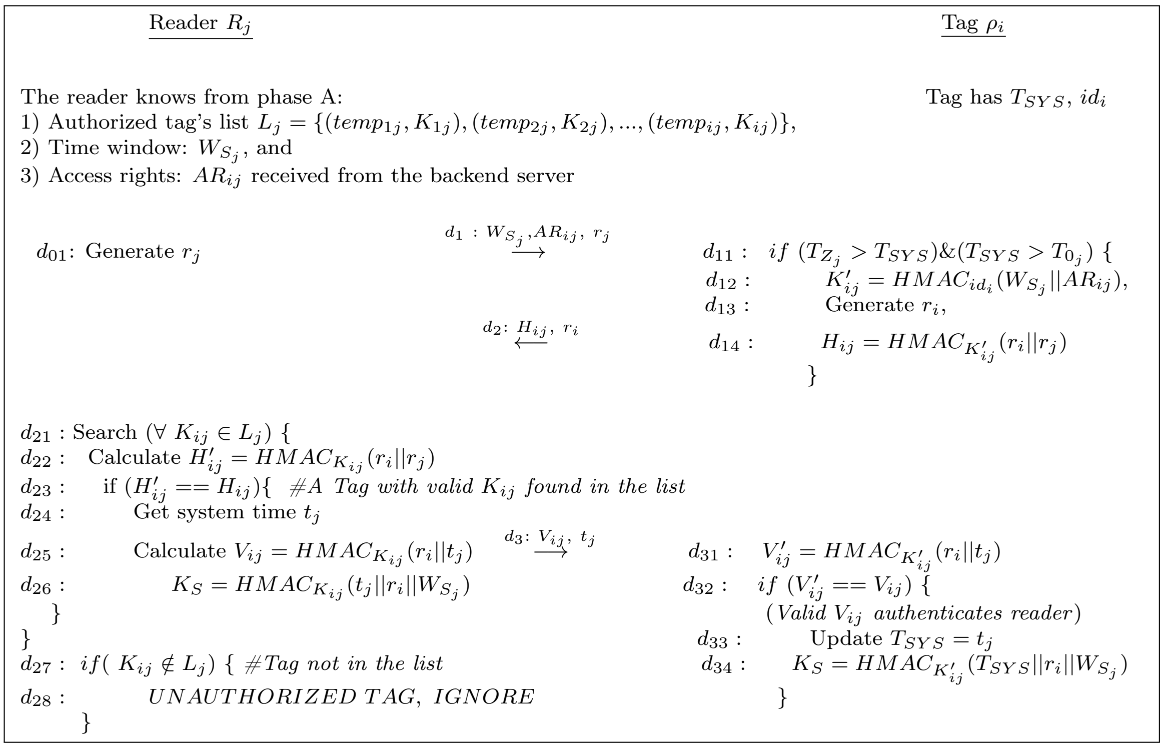
Upon receipt of message d_3, ρ_i checks the validity of V_{ij}. The correct value of V_{ij} authenticates R_j and leads ρ_i to update its timestamp T_{SYS} with a received timestamp t_j.

4.4.2.2 Session Key Generation

The shared session key $K_S = HMAC_{K'_{ij}}(t_j||r_i||W_{S_j})$ is locally generated in both R_j and ρ_i using parameters exchanged during the mutual authentication phase in steps d_{26} and d_{34}, respectively. K_S can be used during next secure data exchange sessions.

4.4.3 Advantages of Serverless RFID Authentication Protocol

In addition to the advantages offered by serverless protocols described in section 2.1, our authentication protocol has three main advantages. First, tags' secrets are never divulged



```
Reader $R_j$                                                       Tag $\rho_i$

The reader knows from phase A:                                     Tag has $T_{SYS}$, $id_i$
1) Authorized tag's list $L_j = \{(temp_{1j}, K_{1j}), (temp_{2j}, K_{2j}), ..., (temp_{ij}, K_{ij})\}$,
2) Time window: $W_{S_j}$, and
3) Access rights: $AR_{ij}$ received from the backend server

$d_{01}$: Generate $r_j$     $\xrightarrow{d_1 : W_{S_j}, AR_{ij}, r_j}$     $d_{11}$ : $if\ (T_{Z_j} > T_{SYS}) \& (T_{SYS} > T_{0_j})$ {
                                                                   $d_{12}$ :     $K'_{ij} = HMAC_{id_i}(W_{S_j} || AR_{ij})$,
                                                                   $d_{13}$ :     Generate $r_i$,
                             $\xleftarrow{d_2 : H_{ij}, r_i}$                   $d_{14}$ :     $H_{ij} = HMAC_{K'_{ij}}(r_i || r_j)$
                                                                              }

$d_{21}$ : Search ($\forall\ K_{ij} \in L_j$) {
$d_{22}$ :   Calculate $H'_{ij} = HMAC_{K_{ij}}(r_i || r_j)$
$d_{23}$ :     if ($H'_{ij} == H_{ij}$){ #A Tag with valid $K_{ij}$ found in the list
$d_{24}$ :         Get system time $t_j$
$d_{25}$ :         Calculate $V_{ij} = HMAC_{K_{ij}}(r_i || t_j)$   $\xrightarrow{d_3 : V_{ij}, t_j}$   $d_{31}$ :  $V'_{ij} = HMAC_{K'_{ij}}(r_i || t_j)$
$d_{26}$ :             $K_S = HMAC_{K_{ij}}(t_j || r_i || W_{S_j})$          $d_{32}$ :  $if\ (V'_{ij} == V_{ij})$ {
    }                                                                    ($Valid\ V_{ij}\ authenticates\ reader$)
}                                                               $d_{33}$ :      Update $T_{SYS} = t_j$
$d_{27}$ : $if(\ K_{ij} \notin L_j)$ { #Tag not in the list                 $d_{34}$ :    $K_S = HMAC_{K'_{ij}}(T_{SYS} || r_i || W_{S_j})$
$d_{28}$ :         $UNAUTHORIZED\ TAG,\ IGNORE$                           }
    }
```

FIGURE 4.3 - Our Serverless Authentication Protocol between Reader and Tag

to the reader. The reader only receives ephemeral tag secrets which expire after a given time. This ensures that tags remain secure even after readers are compromised or physically attacked. Even if the adversary can use the valid information to communicate with a few valid tags, he cannot use one reader's information to generate counterfeit tags and fool other legitimate readers.

Second, the proposed protocol does not require that the reader and the tag have prior relationships with each other, nor to have previously shared any authentication information with each other. Instead, the tags and readers must establish prior relationships with the backend server and rely upon it to verify credentials and authorize sessions.

Third, our protocol minimizes energy consumption by reducing the number of messages exchanged during mutual authentication. Our protocol exchanges only three messages and the size of each message is kept to a minimum. Communication frequency translate to the energy consumption in constrained devices [60, 69, 77].

4.4.4 Protocol Analysis

This section analyses our proposed mutual authentication protocol in terms of performances, security and privacy. We conducted security and privacy analysis in two different ways. First, in section 4.4.4.2, we perform an informal analysis based on the games defined in our threat model of section 4.3.4. Second, in section 4.6.2.1, we conduct an automated verification based on the computation model and the CryptoVerif tool [18].

4.4.4.1 Performance Analysis

Table 4.2 compares our mutual authentication protocol to other similar protocols, namely Tan et al.'s [99] protocol, discussed in section 2.5.1.1.1, Jialiang et al.'s [51] protocol, described in section 2.5.1.4.2, and Lee et al.'s protocol [57], discussed in section 2.5.1.3. In order to create fairness, we use similar parameter sizes to compare our protocol with Tan et al.'s, Lee et al. and Jialiang et al.'s protocols.

Table 4.2 - Performance comparison between our serverless authentication and similar protocols

Performance criteria	Tan et al. [99]	Lee et al. [57]	Jialiang et al.'s [51]	Our Protocol
Computation Costs	4 Hash	4 Hash	3 Hash	3 HMAC
Total messages exchanged	4	3	3	3
Storage costs	116 bytes	116 bytes	112 bytes	80 bytes

Communication Cost: The resource constrained tag ρ_i in our protocol sends 288 bits (36 bytes) and receives 512 bits (64 bytes) of data. In Lee et al. and Tan et al.'s protocols, the tag sends 384 bits (48 bytes) of data and receives 256 bits (32 bytes) of data. In Jialiang et al.'s protocol, the tag sends 384 bits (48 bytes) and receives the same amount. However, Tan et al.'s protocol completes the authentication process with four messages, thus requiring more energy due to the send and receive operations. Our protocol has the least communication cost relative to the rest of the compared protocols.

Computation Cost: The protocols proposed by Tan et al. [99], Lee et al. [57] and Jialiang et al.'s [51] use hash functions while our proposed protocol uses HMAC. In that regard, our proposed protocol has higher computational demands compared to the other two due to the use of HMAC function while the other protocols use hash functions. The HMAC function was selected to reduce the number of collision and increase the security of the protocol [53]. However, any secure hash function can suffice to replace a HMAC in our implementation.

Storage Cost: On the tag side, our protocol uses 160 bits for storing timestamp T_{SYS}, tag's identifier id_i and an additional 480 bits during operation for storing r_j, K_{ij} and V_{ij} which makes a total of 640 bits or 80 bytes at the peak moment, just after receiving message d_3. The reader R_j storage demands vary depending on the number of tags it is allowed to authenticate at a time, that is the number of tag parameters contained in list L_j. In Tan et al.'s protocol, the storage space of 928 bits (116 bytes) is required during the peak time while Jialiang et al. requires 896 bits (112 bytes) of space during the peak time. Our protocol has the least memory demands.

4.4.4.2 Security and Privacy Analysis

This section analyses the security and privacy properties of our proposed mutual authentication protocol. We first compare our protocol and similar protocols based on hash functions (refer to section 2.5). The comparison is depicted in Table 4.3. Then, we use the relevant threat models put forth in section 4.3.4 to analyze the strength of our protocol against attacks.

Table 4.3 - Security and privacy comparison between our serverless authentication and similar protocols

Protocol	Mutual Authentication	Tag Untraceability	Reader Untraceability	Tag Anti-Cloning	Reader Anti-Cloning	Protect Tag's Identity	Protect Reader's Identity	Anti-Replay	Reader Compromise Resistance	Desynchronization Resistance
Tan et al.'s [99]	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓
Lee et al. [57]	✓	✗	✓	✓	✓	✓	✗	✗	✗	✗
Lin et al. [59]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Jialiang et al. [51]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Abdolmaleky et al. [6]	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓
Our Authentication Protocol [63]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

From Table 4.3, Tan et al.'s [99], Lee et al.'s [57], Jialiang et al.'s [51] and Lin et al.'s [59] protocols do not protect the identities of either reader or tags, thus enabling an adversary α to easily identify parties during protocol exchange. Likewise, Tan et al.'s [99], Jialiang et al.'s [51] and Lin et al.'s [59] protocols do not provide mutual authentication between tag and reader, hence diminishing the level of trust and security of the protocol. Moreover, it is only our proposed protocol [63] that satisfies the necessary security and privacy properties including the protection against reader compromise attack because the authentication parameters have a limited period of validity.

Analysis of security games

We analyze our protocol against threat models put forth in Section 4.3.4. This is a complementary analysis to AVISPA's and CryptoVerif's validation presented in Appendices A and B, respectively. The results of the games are presented in Section 4.6.
Game 1 - α masquerades as a Reader: Referring to *Game* 1 of Section 4.3.4, α's objective is to send legitimate *messages* d_1 *and* d_3. That is, α can either crack the key K_{ij} or directly generate a valid *message* d_3 based on sniffed *messages* d_1, d_2 and d_3 of earlier legitimate sessions.

One way to crack K_{ij} is to extract from message d_2 the values of K_{ij} using public values r_i, r_j and H_{ij}. This assumes reversibility of HMAC function, which is contrary to the assumption made in section 4.3.2.

Alternatively, α may combine messages d_1, d_2 and d_3 in order to deduce valuable information and use it to crack the key K_{ij}. However, messages d_2 and d_3 behave as random or pseudo-random strings because they evolve independently from each other as their inputs are different. As such, regardless of the number of sniffed messages d_1, d_2 and d_3, it is infeasible to extract any valuable information, and the game cannot succeed.

Game 2 - α creates counterfeit tags: In our protocol, we do not consider any hardware-based defences against physical attacks. Hence, α may physically compromise a tag ρ_i and access everything in it, including secret information and the information exchanged with R_j. To create a fake tag ρ_x and fool R_j, α must know ρ_x's identity id_x. As the identity of each tag is secret, different and unique, α cannot guess the identity of tag ρ_x by knowing the identity of ρ_i. Thus, compromising a tag ρ_i does not give α the power to derive other tags in L_j, hence α cannot win the game.

Game 3 - α tracks ρ_i: Following Game 3 in Section 4.3.4, ρ_i and R_j use random values to generate *messages* d_2 and d_3, respectively. During session k, ρ_i with i $\in$ {1,2} respond with messages $d_{1_{1k}}$ and $d_{1_{2k}}$, which appear random to α. Any response from ρ_1 is semantically indistinguishable from the response of ρ_2, and even to the previously sent responses of ρ_1. As such, an adversary α is unable to guess with a probability greater than 0.5 which tag ρ_i sent message d_2.

4.5 Secure Serverless Search Protocol

RFID tag search procotol allows RFID reader to securely search for a particular tag among a group of tags within its vicinity, authenticate the tag and initiate a secure data exchange session. RFID tag search functionality is a basic and invaluable tool for efficiently searching among a large amounts of tags [99] without the need to authenticate all tags in the vicinity prior to finding the right one. The Tag search protocol minimizes the time to search for a known tag within a group of tags.

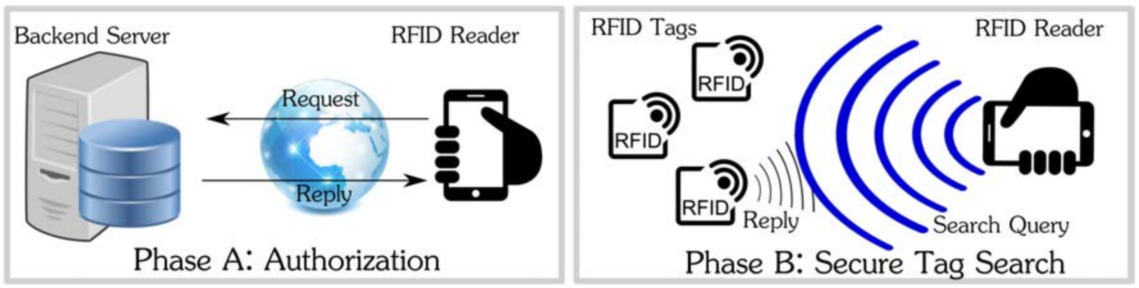


Figure 4.4 - The communication phases for the proposed secure tag search protocol

The practical use for RFID tag search protocols can be found in places like inventory or stores where vendors want to search for specific products among a large number of them. Using the search protocol in such situations ensures efficient and quick retrieval of the product together with its approximate location, due to the limited broadcast range of RFID readers.

Our secure tag search protocol must fulfill five important goals. First, it must allow for an efficient search of a known tag. Second, it must protect the privacy of the searched tag, together with the neighboring tags, during the search process. Third, it must secure the information exchanged between the reader and the tag during the search process. Fourth, the tag must authenticate the reader before replying back. Fifth, the tag should only reply to legitimate queries.

The tag search protocol is done in two distinct phases as depicted in Figure 4.4. First, during the authorization phase, the reader communicates with the backend server to download the credentials for the tags it is granted access to. Second, during the authentication phase, the reader sends a query to search for a particular tag within its vicinity.

The authorization phase, where the RFID reader R_j downloads the list of authorized tags L_j from the backend server, is common to the mutual authentication protocol and is described in section 4.4.1 and depicted in Figure 4.2,. The tag search phase, presented in Figure 4.5, where the protocol performs mutual authentication between the reader and the target tag before generating a common session key for securing the information exchanged, is described in section 4.5.

Secure Tag Search

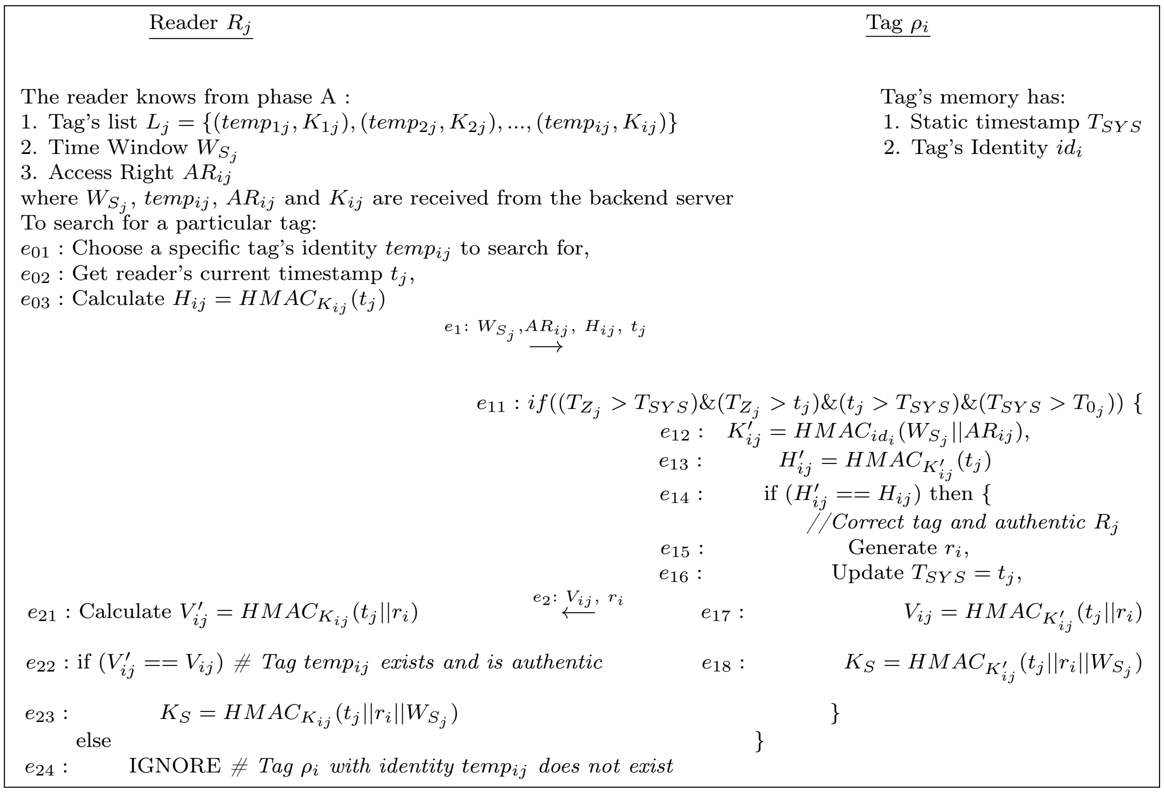
When a reader R_j wants to search for a specific tag with a temporary identity $temp_{ij}$ from the list of tags L_j, it calculates $H_{ij} = HMAC_{K_{ij}}(t_j)$ where t_j is the reader's current timestamp and K_{ij} is the key corresponding to a tag with identity $temp_{ij}$. R_j broadcasts *message* e_1 containing W_{S_j}, AR_{ij}, H_{ij} and t_j to all tags in the vicinity. *Message A* is unicast, which implies it is only intended for the tag possessing the key K_{ij}.

After receiving *message* e_1, a tag ρ_i validates the parameters received, calculates its temporary key K_{ij} and checks whether it is the intended recipient tag by calculating and comparing $H'_{ij} == H_{ij}$. If it is indeed the intended tag and the values are correct, the tag authenticates the reader R_j and ρ_i updates its timestamp T_{SYS} before replying with a challenge V_{ij} and r_i to R_j. The other tags do not respond to the query.

Upon receipt of *message* e_2, R_j verifies V_{ij}. If V_{ij} is valid, R_j authenticates ρ_i.

Session key generation

The reader R_j and a tag ρ_i compute a shared key K_S using parameters from both parties in steps e_{23} and e_{18}, respectively. K_S is used to securely exchange data between R_j and ρ_i using a preferred encryption scheme.



Reader R_j | Tag ρ_i

The reader knows from phase A :
1. Tag's list $L_j = \{(temp_{1j}, K_{1j}), (temp_{2j}, K_{2j}), ..., (temp_{ij}, K_{ij})\}$
2. Time Window W_{S_j}
3. Access Right AR_{ij}

where W_{S_j}, $temp_{ij}$, AR_{ij} and K_{ij} are received from the backend server
To search for a particular tag:
e_{01} : Choose a specific tag's identity $temp_{ij}$ to search for,
e_{02} : Get reader's current timestamp t_j,
e_{03} : Calculate $H_{ij} = HMAC_{K_{ij}}(t_j)$

Tag's memory has:
1. Static timestamp T_{SYS}
2. Tag's Identity id_i

$\xrightarrow{e_1:\ W_{S_j}, AR_{ij},\ H_{ij},\ t_j}$

```
e_11 : $if((T_{Z_j} > T_{SYS})\&(T_{Z_j} > t_j)\&(t_j > T_{SYS})\&(T_{SYS} > T_{0_j}))$ {
    e_12 :  $K'_{ij} = HMAC_{id_i}(W_{S_j}||AR_{ij})$,
    e_13 :  $H'_{ij} = HMAC_{K'_{ij}}(t_j)$
    e_14 :  if ($H'_{ij} == H_{ij}$) then {
                //Correct tag and authentic $R_j$
    e_15 :      Generate $r_i$,
    e_16 :      Update $T_{SYS} = t_j$,
    e_17 :      $V_{ij} = HMAC_{K'_{ij}}(t_j||r_i)$
    e_18 :      $K_S = HMAC_{K'_{ij}}(t_j||r_i||W_{S_j})$
            }
}
```

$\xleftarrow{e_2:\ V_{ij},\ r_i}$

```
e_21 : Calculate $V'_{ij} = HMAC_{K_{ij}}(t_j||r_i)$
e_22 : if ($V'_{ij} == V_{ij}$) # Tag $temp_{ij}$ exists and is authentic
e_23 :     $K_S = HMAC_{K_{ij}}(t_j||r_i||W_{S_j})$
       else
e_24 :     IGNORE # Tag $\rho_i$ with identity $temp_{ij}$ does not exist
```

FIGURE 4.5 - Phase B: Our Serverless Secure RFID Tag Search Protocol

4.5.1 Advantages of our Secure Search Protocol

Apart from classical advantages of the secure tag search protocols such as facilitating the search process of a particular tag instead of authenticating all tags and choosing the right one, our tag search protocol performs very few computations when searching for a tag within a group of tags because for each search query, only one response, from the appropriate tag, is expected, if the tag is present.

Another advantage is that our tag search protocol is not susceptible to the narrowing attacks [102] as *message* e_2 sent by ρ_i is semantically indistinguishable from previous messages sent by ρ_i or by other tags within the vicinity. Narrowing attack occurs when the adversary queries a tag with a particular timestamp and then later tries to identify the same tag by querying it with a timestamp slightly above the previous one [102]. Hence, an adversary cannot easily associate a message to a specific tag.

Finally, unlike the existing protocols described in section 2.5, our secure tag search protocol is resistant to cloning and reader compromise attacks. This is possible because CS grants authentication parameters with a defined period of validity, beyond which they cannot be used to perform authentication between the reader and tags. Hence, when the adversary compromises the reader it only gets temporary authentication values that can only be valid within the defined period of time.

4.5.2 Secure Tag Search Protocol Analysis

The similarities between RFID secure tag search protocol and Mutual authentication protocol lead to the similarities in the performance and security properties. Hence, this section discusses only a few properties that differ from those discussed in section 4.4.4.

4.5.2.1 Performance Analysis

The performance comparison between our secure tag search protocol with other similar protocols is given in Table 4.4. We analyze the communication cost of our protocol compared to other serverless protocols namely Lee et al.'s search protocol, discussed in section 2.5.1.3, and Jialiang et al.'s search protocol, described in section 2.5.1.4.2.

Communication Cost: Our tag search protocol exchanges two messages, one from each party where ρ_i sends 288 bits (36 bytes) and receives 384 bits (48 bytes) of data. Tan et al.'s and Lee et al.'s protocols exchange a total of 640 bits of data each while Jialiang et al.'s protocol exchanges a total of 768 bits of data.

TABLE 4.4 - Performance comparison between our tag search protocol and similar protocols

Performance criteria	Tan et al. [99]	Jialiang et al. [51]	Lee et al. [57]	Our Protocol
Computation cost	3 Hash	3 Hash	2 Hash	3 HMAC
Communication cost (bits)	640	768	640	672
Total replies out of N tags	N	N	N	1

4.5.2.2 Security and Privacy Analysis

This section analyses the security and privacy properties of our secure search protocol. First, we compare our protocol and similar search protocols based on hash functions (refer to section 2.5). The comparison is depicted in Table 4.5. Then, we use the relevant threat models put forth in section 4.3.4 to analyze the strength of our protocol against attacks.

Table 4.5 gives a brief comparison between our protocol and other similar protocols. Our protocol was constructed to protect tags' identities from adversaries, unlike Tan et al. [99] and Lee et al. [57] protocols, which communicate reader's identity as cleartext when querying the tag, hence an adversary α can easily track a reader at any time. Moreover, Tan et al.'s [99], Lee et al.'s [57], Lin et al.'s [59] and Jialiang et al.'s [51] protocols do not offer mutual authentication between tag and reader, which lead to replay attacks and reducing the level of trust and security of the protocol. It can also be observed that it is only our search protocol that satisfies all the privacy and security properties including the reader compromise attack because the parameters granted to the reader are ephemeral and expire after a given period. The rest of the protocols grant parameters that stay valid forever.

TABLE 4.5 - Security comparison between our tag search protocol and similar protocols

Protocol	Mutual Authentication	Tag Untraceability	Search Query Privacy	Tag Anti-Cloning	Reader Anti-Cloning	Protect Tag's Identity	Protect Reader's Identity	Anti-Replay	Reader Compromise Resistance	Desynchronization Resistance
Tan et al.'s [99]	✗	✗	✓	✗	✗	✗	✗	✓	✗	✓
Lin et al. [59]	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Lee et al. [57]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
Xie et al. [107]	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓
Jialiang et al. [51]	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
Our Search Protocol [63]	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Analysis of security games

We analyze our protocol against threat models put forth in Section 4.3.4. This is a complementary analysis to AVISPA's and CryptoVerif's validation presented in Appendices A and B, respectively.

Game 1 - α masquerades as a Reader: Referring to *Game* 1 in Section 4.3.4, α's objective is to send a valid *message* e_1 to ρ_i. The first idea would be that α replays a valid *message* e_1. However, the message is intended for one specific tag with the key K_{ij}, and processing of *message* e_1 by the tag leads to the tag updating its timestamp. As a consequence, assuming that the target tag is in the vicinity of the legitimate reader when transmitting a valid *message* e_1, replays remain useless as it will be considered by the target tag as out-of-date.

There are two other alternatives for α: cracking the key K_{ij} or generating a valid *message* e_1 based on sniffed messages of earlier valid sessions. For cracking K_{ij}, one way is to extract the value of K_{ij} from *message* e_1 or e_2 by reversing the HMAC function with known public values t_j or r_i. However, this contradicts our assumptions of section 4.3.2.

Alternatively, α can analyse several valid pairs of *messages* e_1 and e_2 to generate a new valid *message* e_1. However, *messages* e_1 and e_2 behave as random or pseudo-random strings due to their random inputs. Thus, it is not possible to guess a new valid *message* e_1, and the game cannot succeed.

Game 2 - α creates counterfeit tags: This game is similar to the one analyzed in previous protocol in section 4.4.4.

Game 3 - α tracks ρ_i: As our search tag protocol facilitates a legitimate R_j to search and communicate to a chosen tag within a group, it is also an ideal opportunity for α to track a tag and launch attacks.

However, launching a successful attack means α must link *message* e_2 to a particular tag. As *messages* e_2 coming from ρ_1 and ρ_2 are semantically indistinguishable due to the random inputs r_i and r_j, an adversary α cannot guess with a probability greater than 0.5 which tag ρ_i sent *messages* e_2, and he can not win the game.

4.6 Protocol Validation & Verification

The protocols proposed in this chapter were validated and verified using AVISPA and CryptoVerif tools, respectively. The AVISPA tool, explained in Appendix A, was used to verify the exchanges against common Dolev-Yao attacks while CryptoVerif, described in Appendix B, was used to verify the security of the protocol from the exchanges up to the function construction. Each of these two processes is explained below.

4.6.1 Validation Using AVISPA Tool

We used the *Automated Validation of Internet Security Protocols and Applications (AVISPA)* [14] tool to validate both our protocols. The AVISPA tool, described in Appendix A, uses the attacker model called *Dolev-Yao* [34] i.e., an attacker has full capabilities over the network and can listen or intercept communication, inject new messages or modify messages in transit. The results after running the protocol coded in the *High Level Protocol Specification Language* (HLPSL) [104] is given in Table 4.6.

Table 4.6 - AVISPA validation results

AVISPA Engine	Mutual Authentication	Tag Search
OFMC	SAFE	SAFE
CL-AtSe	SAFE	SAFE
SATMC	SAFE	SAFE
TA4SP	INCONCLUSIVE	INCONCLUSIVE

As shown in Table 4.6, AVISPA gives *SAFE* results for three back-ends *OFMC, CL-AtSe*, and *SATMC*, while the *TA4SP* back-end gives *INCONCLUSIVE* result due to unsupported operations. Thus, AVISPA could not detect any attacks in our proposed protocols.

4.6.2 Verification Using CryptoVerif Tool

In this section we describe the verification of the mutual authentication and secure search protocols using CryptoVerif tool [18].

4.6.2.1 Mutual Authentication Protocol Verification using CryptoVerif Tool

In this section, the computational model is used to prove some security and privacy properties of our mutual authentication protocol, as the computational model is widely accepted as being close to the real execution of protocols [17]. CryptoVerif [18] is the automated tool selected to verify our protocols according to the computational model. CryptoVerif transforms the initial game, which corresponds to the authentication protocol, to a series of games in order to prove whether the protocol meets the security criteria. That is, the successful execution of our protocol in CryptoVerif proves that the tested security properties hold. Note that CryptoVerif game references given in this section and section 4.6.2.2 are disconnected from the games described in section 4.3.4 and later analyzed in section 4.4.4.2.

Listing 4.1: Proof of results for our RFID authentication protocol using CryptoVerif

```
Proved one-session secrecy of Ks in game 7
Proved secrecy of $K_S$ in game 7
Adv[Game 1: one-session secrecy of $K_S$]
<= (2. * N1 * N0 + 2. * N0 * N0) / |nonce| + (2. * N0 + 2. * N1) / |hasht| +
Adv[Game 7: one-session secrecy of $K_S$]
Adv[Game 7: one-session secrecy of $K_S$] <= 0
RESULT Proved one-session secrecy of Ks up to
probability (2. * N1 * N0 + 2. * N0 * N0) / |nonce| + (2. * N0 + 2. * N1) / |hasht|
Adv[Game 1: secrecy of $K_S$]
<= (2. * N1 * N0 + 2. * N0 * N0) / |nonce| + (2. * N0 + 2. * N1) / |hasht| +
Adv[Game 7: secrecy of $K_S$]
Adv[Game 7: secrecy of $K_S$] <= 0
RESULT Proved secrecy of $K_S$ up to
probability (2. * N1 * N0 + 2. * N0 * N0) / |nonce| + (2. * N0 + 2. * N1) / |hasht|
Proved indistinguishability from game 7
Adv[Game 1: indistinguishability from the initial game]
<= (N1 * N0 + N0 * N0) / |nonce| + (N0 + N1) / |hasht| +
Adv[Game 7: indistinguishability from the initial game]
Adv[Game 7: indistinguishability from the initial game] <= 0
RESULT Proved indistinguishability from the initial game up to
 probability (N1 * N0 + N0 * N0) / |nonce| + (N0 + N1) / |hasht|
All queries proved.
```

The CryptoVerif tool is launched for proving the secrecy of the session key K_S (refer to Figure 4.3), as K_S is computed locally on tag and reader and is the fundamental element for guaranteeing the protection of later exchanges. The indistinguishability property is simultaneously proved on the exchanges between the reader and the tag. The code describing the first protocol (also referred to as a "game" by CryptoVerif) is presented in Appendix B.1, and the results obtained after running this game are presented in Listing 4.1.

The Listing 4.1 combines results of two different runs for the same protocol, but with different security properties, to verify the desired security parameters. First the secrecy of the key K_S between the reader and the tag is proved and then the indistinguishability of messages exchanged among different sessions is verified. The secrecy of the key K_S is obtained using statements *query secret* K_S and *query secret1* K_S (refer to the listing in Appendix B.1). $N1, N0$ and *hasht* are parameters set for running the CryptoVerif tool while *nonce* is a unique number used in each session set for the protocol. Nonces are used to simulate timestamps in our verification. The final result of the games, "All queries proved", means that the specified security properties were successfully verified. The result is given in probability which is cumulative probabilities for successful execution of each game in the sequence. For instance, *Adv[Game 7: secrecy of* K_S*] <= 0* means an adversary cannot acquire the secret key K_S at the end of Game 7. Likewise, the adversary cannot distinguish messages from different sessions with the result *Adv[Game 7: indistinguishability from the initial game] <= 0.*

4.6.2.2 Secure Search Protocol Verification using CryptoVerif Tool

The CryptoVerif [18] tool serves to prove two security and privacy properties of our secure tag search protocol, first the secrecy of the session key K_S generated locally by both the reader and the tag and, second, the indistinguishability of the messages exchanged between the reader and the tags.

Listing 4.2: Proof results for our secure tag search protocol using CryptoVerif

```
Proved one-session secrecy of Ks in game 4
Proved secrecy of $K_S$ in game 4
Adv[Game 1: one-session secrecy of $K_S$] <= 0 + Adv[Game 4: one-session secrecy of $K_S$]
Adv[Game 4: one-session secrecy of $K_S$] <= 0
RESULT Proved one-session secrecy of $K_S$
Adv[Game 1: secrecy of $K_S$] <= 0 + Adv[Game 4: secrecy of $K_S$]
Adv[Game 4: secrecy of $K_S$] <= 0
RESULT Proved secrecy of $K_S$
Proved indistinguishability from game 15
Adv[Game 1: indistinguishability from the initial game] <= Pmac(time(context for game 11) +
time, 2. * N, N0, max(maxlength(game 11: ea), maxlength(game 11: m1), maxlength(game 11: ea1)))
 + (N1 * N0 + N0 * N0) / |nonce| + (N0 + N1) / |hasht| +
 Adv[Game 15: indistinguishability from the initial game]
Adv[Game 15: indistinguishability from the initial game] <= 0
All queries proved.
```

The results in Listing 4.2 are obtained after running the code depicted in Appendix B.2 twice, with different security parameters each time, to verify the desired security and privacy properties. In the first run, the secrecy of the key K_S is proved using statement *query secret* K_S and *query secret1* K_S. The second run verifies the indistinguishability of the messages exchanged between reader and tag in different sessions. As shown in Listing 4.2, all queries are successfully proved and the requested security and privacy properties hold.

4.7 Conclusion

This chapter focuses on the security and privacy issues in the pervasive systems by looking at two core challenges in the RFID domain, which are mass tag authentication and secure tag search. Thus, we present two complementing protocols to solve these challenges. The first protocol enables mutual authentication and secure data exchange between an RFID reader and a tag. The second protocol allows an RFID reader to securely search for a particular tag among a group of tags. The proposed protocols have practical use in places such as inventory, shops or stores as they can be used for mass product authentication or facilitating the task of searching for a specific product among a group of products.

The ideas behind these proposed protocols are novel for several aspects. First, the RFID reader and tags do not have to share any material prior to running the mutual authentication session in between. Second, both protocols run in the absence of a backend server. Third, both our protocols rely on the same basic technical elements including primitives and materials shared with the backend server. Fourth, our protocols have low computation overhead and resource demands due to the use of lightweight primitives such as comparison, concatenation and HMAC operations.

We also conduct a thorough analysis to theoretically verify the security, privacy and performance properties of the protocol. The verification and validation were done in three different ways. First, we design informal security and privacy games to verify the defined security and privacy properties and exchanges. Second, we formally validate the properties our proposed solutions using AVISPA tool. Third, we formally verify the proposed solutions using CryptoVerif tool.

After a thorough analysis on serverless protocols, in chapter 2, and our proposals on the serverless paradigm for solving the challenges in the existing schemes, in chapters 3 and 4, the following chapter provides a concise guide on developing secure and efficient serverless protocols. The guide takes into account the common principles used to design the majority of the existing serverless protocols.

Chapter 5

A Guide on Designing Efficient & Secure Serverless Protocols

It used to be expensive to make things public and cheap to make them private. Now it's expensive to make things private and cheap to make them public.

- Clay Shirky

Contents

5.1 Introduction

The preceding chapters provide thorough analysis and contributions in the domain of serverless protocols. This chapter goes one step further by providing a concise guide on how to design efficient and secure serveless protocols.

Over the years, numerous serverless protocols have been proposed, some of which have been detailed in section 2.5. Even though most of the existing schemes are vulnerable or inefficient in various ways, they still provide important guidelines for developing serverless protocols. We thoroughly researched and analyzed the existing proposals for important and common practices based on the requirements of each scenario.

We identify six principles and six best practices for developing efficient security solutions relevant to the pervasive systems. With the help of the provided guide it is possible to develop resource-efficient and secure solutions for any pervasive system. Although useful, the principles and practices we outline in this guide do not impose design rules; they merely point out best ways towards designing serverless protocols. Adherence to these principles and practices may help to improve the efficiency of the designed protocol while avoiding a considerable number of common vulnerabilities found in most existing protocols.

The rest of this chapter is organized as follows, section 5.2 presents key prerequisites necessary to understand before embarking on the task of designing serverless protocols. Section 5.3 provides a concise guide on how to design and develop efficient and secure serverless protocols while section 5.4 gives a brief summary of the existing serverless protocols with brief analysis based on the guide presented in section 5.3. Section 5.5 concludes the chapter.

5.2 Prerequisites for Designing Serverless Protocols

This section outlines some of the basic principles useful for designing efficient and secure serverless protocols. These principles are not hard-coded rules, but merely helpful hints to take into account before embarking on the task of protocol design. The basic requirements for serverless protocols are put forth in section 2.3.

5.2.1 Understand the Requirements

A key to successful protocol design is a clear sense of requirements. This includes both the functional requirements that describe what a system must do, and the non-functional requirements that describe, among other things, the environment in which the system will operate. Getting the requirements right is the most important step. Most of the shortcomings of the existing systems often stem from a failure to get the basic requirements right [27]. This leads to our first principle of designing serverless protocols.

Principle 1: *Well-defined requirements lead to a secure and efficient protocol design*

Principle 1 embodies the idea that protocol design begins with the phase of requirement definition. Requirements must be well-defined to be clearly understood. Well-defined requirements facilitate the research and analysis phases to unveil the security threats relevant to the given scenario. Failure to research and identify the relevant security and privacy risks is a significant cause of protocol vulnerabilities.

Even though serverless protocols must be generic to a great extent, but scenario-specific requirements may require prioritizing some system's aspects to others. For instance, some pervasive objects have limited lifetime power sources while others have energy sources that can be replenished from time to time. These two kinds of devices belong into completely different computation and communication profiles, solely based on their energy sources. Failure to adhere to such details may lead to an inefficient or unusable security protocols. A protocol that is inefficient and unusable is as good as if it does not exist.

5.2.2 Think of Temporary Parameters

The interactions between pervasive objects are mostly ephemeral, so should be the credentials used during the authentication sessions i.e., authorization and access rights. It is not ideal to use permanent secret information for one-time or short-lived authentication sessions that are not bound to repeat for a very long time. For instance, generating secret information based on the identities of the communicating parties that will remain valid for a long time, even beyond the requested period of usability, does not constitute a good security. Thus, it is essential to take two principles into account.

Principle 2: *Ephemeral sessions should be authenticated using temporary, but legitimate and verifiable, parameters.*

Principle 2 insists that, the protocol should not unnecessarily divulge long-term secrets (or any private information) belonging to the communicating parties during the authentication session of ephemeral communications. Rather, the protocol should use temporary, but authentic and verifiable, parameters to complete the task at hand. The use of temporary parameters e.g., secret information and identities, avoids misusing the respective long-term parameters once the given devices are compromised or stolen. This principle must be adhered to whether the devices have mutual trust or not.

Principle 3: *The parameters used during authentication in a serverless protocol should have a limited period of validity.*

Principle 3 improves on Principle 2; it emphasizes on the incorporation of context information in the authentication parameters granted to the communicating parties in order to limit their period of validity. The problem of authentication parameters validity is recurrent in the existing serverless protocols, as described in section 2.5.7.2. To ensure security, it is important that the authentication parameters should only be valid during the course of the session at hand. In case of future communications, the respective parties should follow the procedures for establishing new security associations.

5.2.3 Be a Minimalist

Most pervasive devices limit the amount of resources allocated for the security purposes [62, 71]. Thus, it is important to efficiently utilize the available resources. The following principle can help to minimize the resource usage.

Principle 4: *Limit the protocol design to its most important elements.*

Principle 4 elaborates that the protocol design must have minimum resource requirements for pervasive devices. Resource requirements can be reduced in various ways, for instance, by storing and exchanging minimum, but crucial, information during authentication. The basic minimal information should be verifiable that they originate from a trusted source, which is mutually trusted by both communicating parties e.g., a centralized authorization server.

5.2.4 Leverage Capabilities

When designing a protocol for pervasive systems, it is helpful to exploit the capabilities that the respective devices already offer. Some of the features e.g., embedded clocks, random number generators, or common cryptographic algorithms, are important basic building blocks for security protocols.

Principle 5: *Leverage device capabilities to improve protocol design*

Principle 5 literally suggests that, if there are any features in the target pervasive devices that could be used to improve the design of the protocol in question, then they should be taken advantage of. For instance, several hybrid hardware-software approaches have been proposed to efficiently implement security functions [56]. The general-purpose embedded processor cores with hardware accelerators are useful for most performance critical steps. Hardware accelerators are useful in improving the efficiency of the respective functionalities compared to using software based implementation. However, hardware accelerators may not merely be used to improve performance, but also as a power-saving measure [44].

5.2.5 Be Prudent

Principle 6: *A secure protocol design should adhere to the prudent engineering practices for cryptographic protocols.*

Every stage of protocol design is prone to errors, especially if one is not careful to follow basic guidelines on designing cryptographic protocols [5, 98]. There are various proposals for principles for designing cryptographic protocols such as those put forth by Abadi et al. [5] and Anderson et al. [13]. Though their principles were not meant to be design rules, they are useful in avoiding common mistakes that are committed by many protocol designers. Thus, we recommend understanding the basics and following these principles of designing cryptographic protocols before the design phase.

5.3 A Guide on Designing Efficient and Secure Serverless Protocols

The advancement of technology leads to the advent of new services that require security to protect information. Each of these services has its own characteristics, requirements and constraints. It is hard to create a generic security framework applicable across all technologies. Pervasive computing is an example of the technology that challenges the norms and requires a partial or complete redesign of the existing security protocols to suit their needs.

This section provides a guide on how to design and develop security solutions based on the resource constraints and specific requirements imposed by the respective technology. The guide provides six best practices on how to design and develop efficient security solutions for pervasive devices based on specific constraints as described in the following subsections. Each protocol design choice has its advantages and drawbacks but it is important to consider how realistic and functional the protocol has to be, taking into account important factors such as the utilization of bandwidth, CPU, and memory within pervasive devices.

TABLE 5.1 - Resource Consumption (+: more demanding; - : less demanding)

	RESOURCE			
Operation	*Energy*	*Memory*	*Computation*	*Bandwidth*
Connectivity	++++	+	- - - -	++++
Communication	++++	++	- - - -	++++
Key establishment	+	++++	++++	++
Data processing	+	++++	++++	- - - -

5.3.1 Communication vs Computation: Energy Optimization

As depicted in Table 5.1, all activities require energy within the pervasive devices, some are more energy consuming than others. Communication and computation are two main energy intensive activities in pervasive devices. Furthermore, communication is comparatively more expensive in energy consumption than the heaviest computation within the same device [11, 76, 101]. Thus, from the energy perspective, it is more economical to perform computation within the device than frequent communication. As depicted in Table 5.1 both communication and computation consume memory, because data must be stored before it is sent.

According to the energy profiles in pervasive environments described in section 2.4.1, a protocol may optimize energy by doing computation and avoid frequent communication [76], if it has limited energy source, or it may favor communication in place of computation if its energy source can be regularly replenished.

5.3.2 End-to-End vs Node-to-Node Security: Security Optimization

Security in pervasive ecosystems can be either *end-to-end* or *node-to-node* depending on the requirements. Node-to-node encryption is ideal in a setup where the security scope is only limited to neighboring nodes, for instance when sensor nodes are aggregating and comparing information before forwarding to the centralized server [12, 94]. Each participating node in the transfer of information must receive, store, decrypt and encrypt information before forwarding to the next node. Node-to-node encryption has major setbacks such as high latency and resource demands due to regular storage, decryption, treatment and encryption operations for every message received [12]. Moreover node-to-node scheme requires trust between the sender and all the intermediary nodes. In sensor networks it is mandatory for every message to be authenticated before being processed or forwarded in order to thwart attacks such as data injection attacks in the network [108].

On the other hand, end-to-end encryption allows the source node to encrypt information that only the destination can decrypt and understand [97]. The intermediary nodes only forward the information and do not take active role in security of the information transferred. This mode of encryption is ideal for systems desiring to economize resources, avoid latency and offer higher levels of confidentiality, especially in environments where neighboring nodes cannot be trusted [12, 97].

Table 5.2 - Comparing Security Scopes

	Node-To-Node	End-To-End
Parties involved	Many	Two
Security level	Low	High
Computation demands	High	Low
Memory demands	High	Low
Energy consumption	High	Low
Implementation	Difficult	Easy

5.3.3 Bandwidth vs Storage Space: Cost Optimization

Bandwidth is a precious and often a scarce resource in wireless realm [41]. Most pervasive devices do not have very high bandwidth, which limit the total amount of information exchanged at any given time, either between local peers or with distant parties. Depending on the scenario, overusing the bandwidth, through frequent communication, may have adverse effects on the quality of service for the information exchanged due to the network congestion i.e. multiple devices share the limited bandwidth, which augments the rate of data loss. The best alternative is to reduce the frequency of communication by storing important information locally, which may help to compute necessary parameters once needed [62].

Bandwidth becomes costly when pervasive devices must communicate to distant parties via third party carriers who charge money per bandwidth consumption e.g. via Internet or 3G (Third Generation of Mobile Telecommunications Technology). In such cases, bandwidth is directly related to the frequency of communication and the volume of information exchanged. Moreover, frequent communication consumes more than just bandwidth, it also requires storage space within a device because before the device send the information it must store it locally during the phase of connection establishment.

5.3.4 Energy vs Security Level

Security operations and functionalities have different resource demands and pervasive devices come in different capabilities. If the pervasive devices have stringent energy limitations then security should be provided using simple and lightweight cryptographic primitives. Lightweight primitives have little resource demands from the device, which translates to little energy consumption. For instance, for energy constrained devices, integrity or authenticity of data can be efficiently provided using a hash function compared to the resource demanding Message Authentication Code (MAC) [53].

For pervasive systems without stringent energy constraints, higher security can be ensured using efficient but yet robust security mechanisms suitable for pervasive devices [105]. For instance, the use of lightweight symmetric encryption mechanisms to enforce the confidentiality of information. In case standard symmetric encryption algorithms e.g. AES, 3DES [37] are too resource expensive for the respective systems, simple methods like Exclusive-OR ($\oplus$) [42] may suffice to provide data confidentiality [62].

5.3.5 Storage vs Computation: Memory Optimization

Memory is a critical resource in pervasive devices. Computation and storage are equivalent, but it is important to find the balance between the two for maximum efficiency [7]. To some protocols, especially for systems where security parameters of communicating parties can be known in advance, pre-computation and storage are ideal to optimize the system's performance. Such systems prioritize time over storage as they reduce latency by computing and storing values ahead of time [62].

However, calculating results and storing them for later use may lead to large volumes of rarely used data, wastes space and energy, and makes it a very expensive strategy, especially for resource constrained devices. A space-time or time-memory tradeoff is when a protocol trades increased space for decreased time [7] i.e. tolerating the latency of obtaining results (through computation) for an increased space within the device. This is a good choice for devices with strict storage limitations, where parameters should be computed upon demand.

For maximum efficiency, the protocol must strike a balance between storage and computation by considering three factors. First, it should be explicitly known which parameters can be stored and which ones can be feasibly computed upon demand. Second, determine whether it is more efficient to recompute a result, as opposed to storing it; depending on whether the result will be reused later in the process and the potential resource penalties if the data is unavailable when needed [7]. Third, only a few parameters, with minimal sizes, and essential for the device to autonomously and efficiently perform core functionalities should be stored locally [22].

5.3.6 Tamper resistance vs Cost

Most pervasive devices are easily accessible hence prone to physical attacks. It is ideal to equip pervasive devices with appropriate mechanisms to avoid tampering, if they contain very sensitive information. The anti-tampering mechanisms ensure that the confidential or secret information is not divulged even when the device is physically compromised [29,103].

The use of anti-tampering mechanisms requires additional circuitry, which translates to increased costs per device. However, cost is an important factor for the success of pervasive systems, especially for systems which require mass deployment. For scenarios requiring cheap pervasive devices, the high cost may act as a hindrance factor [48, 52].

5.3.7 Enforcing Security and Privacy Properties

The six best practices described in sections 5.3.1 to 5.3.6 focus on efficient solutions from the resource perspective. However, there are additional desirable features to providing security and privacy in pervasive systems. For instance, mutual authentication authentication is indispensable for increasing the confidence of the system. Mutual authentication can be done through challenge-response by objects exchanging parameters that can be cryptographically verified between the communicating parties. In turn, authentication enforces data integrity.

Additionally, the use of relevant context information, for instance time or location, may prove useful in enforcing various security and privacy properties such as non-traceability, data freshness and thwarting attacks relevant to pervasive systems e.g. replay attacks [62]. Context information are inherent in pervasive systems hence easy to include for security implementations. Ideally, security and privacy properties can be further enforced by combining context information with some cryptographic primitives such as Pseudo-Random Number Generator (PRNG), Hash, Message Authentication Code (MAC) or Hashed based MAC (HMAC).

For instance, the use of PRNG to randomize exchanges in a pervasive systems may ensure non-traceability. The combination of hash function, secret parameters and context information (e.g. date and time) may suffice to ensure freshness of message exchanges. The confidentiality of the exchanges may be ensured with the use lightweight encryption schemes, such as AES [79] or XOR [42].

It should be noted that the use of cryptographic primitives raises the arguments discussed in sections 5.3.1 to 5.3.6. The final choice is solely dependent on the capability of the respective devices. For instance, the choice of using either Hash or MAC in ensuring data integrity is determined by the device's resource profile.

Recently, some of the major vendors of processors for devices with strict resource constraints, such as micro-controllers, integrate cryptographic libraries in their products. This plays part in influencing the choice of cryptographic schemes depending on the processor running on the pervasive device. For instance, AXSEM produces an Ultra-Low Power RF-Micro-controller [3] containing the hardware implementation of the Advanced Encryption Standard (AES) [79]. Other vendors include both hardware and software implementations

Table 5.3 - Comparison of existing security protocols with respect to our proposed guide

	CRITERIA CHOSEN TO PROPOSE SOLUTION				
Existing Protocols	*Robust Security vs Low Energy*	*Bandwidth vs Storage*	*Communication vs Computation*	*Storage vs Computation*	*End-to-end vs Node-to-node*
Tan et al. [99]	Robust Security	Storage	Communication	Computation	End-to-end
Kim et al. [55]	Low Energy	Storage	Computation	Storage	End-to-end
Zhu et al. [108]	Robust Security	Bandwidth	Communication	Computation	Node-to-node
Ahamed et al. [9]	Robust Security	Storage	Computation	Computation	End-to-end
Mtita et al. [62]	Low Energy	Storage	Computation	Storage	End-to-end
Perrig et al. [70]	Robust Security	Bandwidth	Communication	Storage	Node-to-node
Pourpouneh et al. [78]	Low Energy	Storage	Computation	Computation	End-to-end
Jialiang et al. [51]	Robust Security	Storage	Communication	Computation	End-to-end
Mtita et al. [63]	Low Energy	Storage	Computation	Computation	End-to-end

of various cryptographic functionalities e.g. the STM32 cryptographic library by ST Microelectronics includes implementation of all common cryptographic scheme such as RSA, ECC, AES and ARC4 [1].

5.4 Analysis of the Existing Solutions with Regard to the Guide

Various security protocols for pervasive systems exist, majority of which solve the problem in the domain of Radio Frequency Identification (RFID) technology [9, 51, 55, 78, 99] and sensor networks [70, 108]. This section analyses a few serverless protocols proposed in the state of the art, as summarized in Table 5.3, using the guide given in section 5.3.

Basically, the security protocols must meet the resource requirements for pervasive devices they are running on. Table 5.3 shows different security protocols for pervasive systems with varying implementation choices suiting different scenarios and device profiles.

Tan et al.'s. [99] serverless RFID authentication protocol is designed to offer security in constrained passive tags. Their protocols' designs focus on robust security over low energy, a feature shared by Ahamed et al.'s [9] and Jialiang et al.'s [51] protocols. Energy is not an issue for passive RFID tags because they use the energy transmitted electromagnetically from the Reader. Tan et al. prioritizes storage over bandwidth because the reader locally stores all the necessary information needed to authenticate tags, hence reducing the frequency of communication to the distant servers. This is an inherent feature for serverless protocols, hence shared by most protocols depicted in Table 5.3. The protocols prioritize the cheaper approach of local communication between devices over computation within the device. Moreover, the protocols privilege computation over storage of parameters because the reader and tags generate new randoms for each session and they do not keep states or remember session keys after the end of each exchange session.

Our proposal [62], authenticating cluster devices with untrusted parties using a serverless paradigm discussed in chapter 3, is designed to use low energy by reducing the frequency of communication between proximity devices. This feature is shared by protocols proposed by Pourpouneh et al. [78] and Kim et al. [55]. The protocol privileges computation over communication by locally computing and verifying parameters. Moreover, the protocol is built to increase efficiency by pre-calculating and storing values within the terminal so as to improve efficiency, this feature is shared by another protocol proposed by Kim et al. [55].

The features of our first contribution [62] in chapter 3 are also shared by two protocols in our second contribution [63] of chapter 4, which are serverless mutual authentication and the secure serverless search protocols, both in the domain of RFID. However, the serverless mutual authentication protocol has an exception that it prioritizes computation to storage. Our serverless mutual authentication protocol performs a computation and search for each reply it receives from the tags in the vicinity instead of pre-computing values beforehand. The search and computation correspond to steps d_{21} to d_{23} of Figure 4.3.

Zhu et al. [108] and Perrig et al. [70] protocols prioritize bandwidth consumption because they communicate more frequently in a dynamic environment. Sensor nodes periodically broadcast various information such as routing control messages [108]. Moreover, Zhu et al.'s [108] one-time authentication key scheme is different to Perrig et al.'s [70] protocol, based on $\mu TESLA$. $\mu TESLA$ prioritizes storage over computation because the information received has to be buffered before the key is revealed [70, 108].

It can be noted that most of the protocols depicted in Table 5.3 support end-to-end security mechanism. This is justifiable because it ensures higher security and lowers latency within the network. However, the protocols suggested by Zhu et al. [108] and Perrig et al. [70] work in a sensor network environment where data aggregation is done at each node, and therefore the information transferred must be interpreted by every node on its path.

5.5 Conclusion

This chapter provides a guide for designing efficient and secure serverless protocols. We start by providing six necessary principles, which are helpful hints to help avoid common mistakes during the protocols design phase.

The guide we provide contains six best practices on how to design and develop efficient and secure serverless protocols relevant to pervasive systems. These best practices were gathered by noticing some common features among serverless protocols that need improvements. The guide helps to develop efficient serverless protocols suiting the device's profile by optimizing resource usage while providing adequate security, privacy and guaranteeing reliability.

Lastly, we use the proposed guide to analyze some of the existing serverless protocols. The analysis shows how each protocol prioritizes some criteria in the system relative to the other, based on the resource availability and system requirements.

Chapter 6

Conclusion & Future Perspectives

A chick that grows into a cock can be spotted from the very day it hatches

- African Proverb

Contents

THIS dissertation focuses on the security and privacy issues for serverless protocols in the era of pervasive computing. This chapter summarizes the main contributions of this thesis in section 6.1 and section 6.2 identifies areas where further research is necessary to improve the security and privacy in the era of Internet-of-Things.

6.1 Conclusion

In this thesis we address the security and privacy challenges relevant to the pervasive systems with strict resource constraints. We mainly focus on serverless authentication protocols, which are authentication mechanisms that do not require persistent connections to the centralized authentication or authorization server during the authentication session between two heterogeneous devices. Serverless protocols are ideal mechanisms for authentication in the era of pervasive computing.

We lay a ground work by providing features and requirements for serverless protocols. Using these requirements and features of serverless protocols, we provide a comprehensive analysis and comparison of the existing serverless protocols in terms of security, privacy and performance.

Three novel contributions are proposed in this thesis. Our first contribution is a serverless lightweight mutual authentication protocol for heterogeneous pervasive devices. The first contribution provides three advantages relative to the existing protocols. First, it does not divulge the secret information of the communicating parties during the authentication phase. Second, it minimizes the communication cost by exchanging very few data during the mutual authentication phase. Third, the protocol supports spontaneous communication as it does not require devices to share parameters before the authentication process. This contribution satisfies the privacy, security and privacy requirements of pervasive systems. Towards validating our proposal, we provide a formal validation using the AVISPA tool and the informal validation using security and privacy games.

Our second contribution is made of two complementing protocols in the domain of RFID technologies. The first protocol aims towards mass authentication between an RFID reader and a group of tags while the second protocol performs a secure search for a target tag among a group of tags within the reader's vicinity. Both of the proposed protocols take into account the resource constraint nature of the RFID tags. The originality of these protocols is that they do not require RFID readers and tags to share authentication parameters beforehand, rather each of these parties need to have a prior relationship with the central server in order to establish the security association. Moreover, our protocol makes use of timestamps in passive tags that do not have embedded clocks. Timestamps are used to enforce access right revocation and ensure freshness of the messages exchanged. In our second contribution, we perform two formal validations; one is done using the AVISPA tool and the other is done using the CryptoVerif tool. Moreover, we also provide an informal validation using security and privacy games.

After a thorough study of serverless protocols, we propose our third contribution, a concise guide on how to develop secure and efficient serverless protocols relevant to the pervasive systems. The guide contains six principles and six best practices towards developing secure and efficient serverless protocols. The guide is meant to help towards designing efficient, secure and simple serverless protocols by avoiding common errors and confusions found in the existing protocols.

6.2 Future Perspectives

The Internet-of-Things technologies are still in their infancy. There are a lot of interesting open issues in the area of privacy and security that deserve to be fully researched to make authentication mechanisms in the pervasive era more efficient, usable and secure for the end users.

6.2.1 Access Rights Revocation

The issue of access right revocation is still a problem in most pervasive systems. Access revocation is a preventive means towards rogue parties in pervasive era. The future serverless protocol designs should take into account the use of context information, or any other relevant information, to restrict the parameters usage in pervasive systems e.g., period of validity or area of validity (specific to geographic zones).

6.2.2 Common Security Platforms

Pervasive computing calls for calm technologies, where technologies recede in the background and disappears from the human consciousness as they become one and indispensable part of the daily reality. However, this is only possible in the presence of self-configuring systems that share common security platforms and do not require constant human intervention for parameter configuration. So far, the idea of seamless interaction seems far fetched and hard to attain due to the security barriers i.e., two objects cannot seamlessly and securely interact without sharing common security platforms.

The challenge in this case comes with the fact that, the majority of pervasive devices do not have enough resources, especially storage, to hold a range of security protocols that can be selected when communicating with peers. Most pervasive devices will have at most one security protocol, which will be used to communicate with other peers. A research is needed to find out how can different security protocols communicate and share information without demanding a lot of resources from the host device.

6.2.3 Privacy Issues

Pervasive systems create smart spaces which improve how we work and interact with our environments. Our interactions with pervasive devices generate data, which are stored, shared and later mined in order to help improve various parameters in the provided services. But, pervasive systems, if not properly protected to hide user's personal information, become potential loopholes for privacy and security breaches. For instance, most companies aim at tracking and learning people's behaviors in order to turn them into potential markets i.e. sell them products and services. But these systems also arouse the interests of attackers and malicious users who would want to selectively steal from or harm the targeted people. One of the important areas that can be further researched to protect the privacy of users in the pervasive era is the identity exposure.

In pervasive era the exposure of an identity, whether intentional or accidental, may lead to adverse effects. An identity can be any set of information that uniquely identify the device, the most critical of which is the identifier of the device itself, as may be multiple per device (e.g. TraxID, MAC address), and usually remain static or unchanged over a period of time.

The issue of identifiers must be handled in a cross-layer way so any effort made in one of the layers is not undermined by another layer. Thus it is imperative to protect the identities of the communicating parties in the communication. For instance, if the exchange is ephemeral between two untrusted devices, then it is better to use temporary identifiers, instead of actual identifiers, to avoid any future misuses. Such prudence may help to improve the privacy and security of the respective pervasive systems.

6.2.4 Using Public Key Schemes

The desirable security properties can be fully attained in pervasive systems once the public key mechanisms can be efficiently exploited in resource constrained devices. Most of the existing problems in the pervasive systems arise from the use of security protocols solely based on symmetric algorithms as the majority of the devices are incapable of running the demanding and costly public key based protocols. The combination of both public key and secret key schemes strengthens the security while allowing autonomy for the respective devices to communicate and eliminate most of the vulnerabilities that exist today, such as key initialization or key distribution in resource constrained devices.

Asymmetric cryptography is useful in two important aspects. First, it facilitates the secure key exchange between two communicating parties. This is ideal as most of the communications in pervasive systems are ephemeral and done spontaneously. The use of public key schemes in exchanging keys will simplify the process of key management and increase the security of the respective systems. Second, the use of different keys for encryption and decryption in asymmetric systems guarantees non-repudiation, which is important to guarantee the origin of the messages in pervasive systems.

The fate of the majority of the existing cryptographic schemes will be in peril in the post-quantum computing era. As pervasive technologies are still emerging, it is ideal to research and design protocols that are secure even during the quantum computing era. Those mechanisms exist in the meantime such as lattice based cryptography or error-correcting codes. More researches in incorporating such asymmetric mechanisms into pervasive systems will ensure long term protection against threats to the security and privacy of the pervasive devices.

List of Figures

List of Tables

Bibliography

[1] User manual: Stm32 cryptographic library, http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/cd00208802.pdf, accessed: 16/09/2015.

[2] Specification for the advanced encryption standard (aes). Federal Information Processing Standards Publication 197, 2001.

[3] Datasheet: Soc ultra-low power rf-microcontroller for rf carrier frequencies, 2015. Accessed: 16/09/2015.

[4] Traxens: Real data for logistics excellence. `http://traxens.com/about-us/vision-and-mission.html`, accessed in 2015. Accessed: 11/09/2015.

[5] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE transactions on Software Engineering*, 22(1):6, 1996.

[6] Shahab Abdolmaleky, Shahla Atapoor, Mohammad Hajighasemlou, and Hamid Sharini. A strengthened version of a hash-based rfid server-less security scheme. *Advances in Computer Science: an International Journal*, 4:18–23, 2015.

[7] Ian F Adams, Darrell DE Long, Ethan L Miller, Shankar Pasupathy, and Mark W Storer. Maximizing efficiency by trading storage for computation. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, pages 17–17. USENIX Association, 2009.

[8] Mehdi Adda, Jabril Abdelaziz, Hamid Mcheick, and Rabeb Saad. Toward an access control model for iotcollab. *Procedia Computer Science*, 52:428–435, 2015.

[9] Sheikh Iqbal Ahamed, Farzana Rahman, and Endadul Hoque. Erap: Ecc based rfid authentication protocol. In *Future Trends of Distributed Computing Systems, 2008. FTDCS'08. 12th IEEE International Workshop on*, pages 219–225. IEEE, 2008.

[10] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *Communications magazine, IEEE*, 40(8):102–114, 2002.

[11] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[12] Madhukar Anand, Eric Cronin, Micah Sherr, Zachary G Ives, and Insup Lee. Sensor network security: More interesting than you think. In *HotSec*, 2006.

[13] Ross Anderson and Roger Needham. Robustness principles for public key protocols. In *Advances in Cryptology-CRYPT0'95*, pages 236–247. Springer, 1995.

[14] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *Computer Aided Verification*, pages 281–285. Springer, 2005.

[15] A Barbir, S Murphy, and Y Yang. Generic threats to routing protocols. 2006.

[16] Bruno Blanchet. A computationally sound mechanized prover for security protocols. *Dependable and Secure Computing, IEEE Transactions on*, 5(4):193–207, 2008.

[17] Bruno Blanchet. Mechanizing game-based proofs of security protocols. *Software Safety and Security-Tools for Analysis and Verification*, 33:1–25, 2011.

[18] Bruno Blanchet and David Cadé. Cryptoverif: Cryptographic protocol verifier in the computational model, 2011.

[19] Mike Burmester and Jorge Munilla. Lightweight rfid authentication with forward and backward security. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):11, 2011.

[20] Roy Campbell, Jalal Al-Muhtadi, Prasad Naldurg, Geetanjali Sampemane, and M Dennis Mickunas. Towards security and privacy for pervasive computing. In *Software Security - Theories and Systems*, pages 1–15. Springer, 2003.

[21] Bruno Da Silva Campos, Joel JPC Rodrigues, Luís ML Oliveira, Lucas DP Mendes, Eduardo F Nakamura, and Carlos Mauricio S Figueiredo. Design and construction of a wireless sensor and actuator network gateway based on 6lowpan. In *EUROCON-International Conference on Computer as a Tool (EUROCON), 2011 IEEE*, pages 1–4. IEEE, 2011.

[22] David W Carman, Peter S Kruus, and Brian J Matt. Constraints and approaches for distributed sensor network security (final). *DARPA Project report,(Cryptographic Technologies Group, Trusted Information System, NAI Labs)*, 1(1), 2000.

[23] Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi. Soupa: Standard ontology for ubiquitous and pervasive applications. In *Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on*, pages 258–267. IEEE, 2004.

[24] Jyh-Cheng Chen, Krishna M Sivalingam, Prathima Agrawal, and Shalinee Kishore. A comparison of mac protocols for wireless local networks based on battery power consumption. In *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 150–157. IEEE, 1998.

[25] Tom Chothia and Vitaliy Smirnov. A traceability attack against e-passports. In *Financial Cryptography and Data Security*, pages 20–34. Springer, 2010.

[26] Ji Young Chun, Jung Yeon Hwang, and Dong Hoon Lee. Rfid tag search protocol preserving privacy of mobile reader holders. *IEICE Electronics Express*, 8(2):50–56, 2011.

[27] MIT Kerberos Consortium et al. Why is kerberos a credible security solution?; 2008 by mit kerberos consortium, 2011.

[28] Andrei Costin, Apostolis Zarras, and Aurélien Francillon. Automated dynamic firmware analysis at scale: A case study on embedded web interfaces. *arXiv preprint arXiv:1511.03609*, 2015.

[29] Michael J Covington and Rush Carskadden. Threat implications of the internet of things. In *Cyber Conflict (CyCon), 2013 5th International Conference on*, pages 1–12. IEEE, 2013.

[30] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Advances in Cryptology - CRYPTO'98*, pages 13–25. Springer, 1998.

[31] Fernando Mendonça de Almeida, Admilson de Ribamar Lima Ribeiro, and Edward David Moreno. An architecture for self-healing in internet of things. *UBICOMM 2015*, page 89, 2015.

[32] Miaolei Deng, Weidong Yang, and Weijun Zhu. Weakness in a serverless authentication protocol for radio frequency identification. In *Mechatronics and Automatic Control Systems*, pages 1055–1061. Springer, 2014.

[33] Tassos Dimitriou. A lightweight rfid protocol to protect against traceability and cloning attacks. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 59–66. IEEE, 2005.

[34] Danny Dolev and Andrew C Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.

[35] D Nguyen Duc, Jaemin Park, Hyunrok Lee, and Kwangjo Kim. Enhancing security of epcglobal gen-2 rfid tag against traceability and cloning. 2006.

[36] Donald Eastlake and Paul Jones. Us secure hash algorithm 1 (sha1), 2001.

[37] Mansoor Ebrahim, Shujaat Khan, and Umer Bin Khalid. Symmetric algorithm survey: a comparative analysis. *arXiv preprint arXiv:1405.0398*, 2014.

[38] Jun Feng, Yu Chen, Wei-Shinn Ku, and Pu Liu. Analysis of integrity vulnerabilities and a non-repudiation protocol for cloud data storage platforms. In *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pages 251–258. IEEE, 2010.

[39] Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Hmac is a randomness extractor and applications to tls. In *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 21–32. ACM, 2008.

[40] Yong Gan, Lei He, and Yi-feng Yin. Rfid tag ownership transfer protocol with retrospective ability. *Cybernetics and Information Technologies*, 14(4):121–130, 2014.

[41] David Garlan, Daniel P Siewiorek, Asim Smailagic, and Peter Steenkiste. Project aura: Toward distraction-free pervasive computing. *Pervasive Computing, IEEE*, 1(2):22–31, 2002.

[42] Víctor Gayoso Martínez, Luis Hernández Encinas, Carmen Sánchez Ávila, et al. A survey of the elliptic curve integrated encryption scheme. 2010.

[43] Jens Geersbro and Thomas Ritter. Case: Reinventing the container shipping industry. *jbm-Journal of Business Market Management*, 7(1):301–305, 2014.

[44] Mark Hempstead, Nikhil Tripathi, Patrick Mauro, Gu-Yeon Wei, and David Brooks. An ultra low power system architecture for sensor network applications. In *ACM SIGARCH Computer Architecture News*, volume 33, pages 208–219. IEEE Computer Society, 2005.

[45] Md Endadul Hoque, Farzana Rahman, Sheikh I Ahamed, and Jong Hyuk Park. Enhancing privacy and security of rfid system with serverless authentication and search protocols in pervasive environments. *Wireless personal communications*, 55(1):65–79, 2010.

[46] http://www.allaboutbatteries.com/Energy tables.html. Battery types, energy and consumptions, June 2011.

[47] ITU ITU. Information technology–open systems interconnection–the directory: Publickey and attribute certificate frameworks. *Suíça, Genebra, ago*, 2005.

[48] Ravi Jain and John Wullert II. Challenges: environmental design for pervasive computing systems. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 263–270. ACM, 2002.

[49] Ph.D. James Castonguay. International shipping: Globalization in crisis. `http://www.visionproject.org/images/img_magazine/pdfs/international_shipping.pdf`. Accessed: 11/09/2015.

[50] Il-Soo Jeon and Eun-Jun Yoon. An ultra-lightweight rfid seeking protocol for low-cost tags. *Applied Mathematical Sciences*, 8(125):6245–6255, 2014.

[51] He Jialiang, Xu Youjun, and Xu Zhiqiang. Secure and private protocols for serverless rfid systems. *International Journal of Control and Automation*, 7(2):131–142, 2014.

[52] Ari Juels and Stephen A Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology–CRYPTO 2005*, pages 293–308. Springer, 2005.

[53] Marcio Juliato and Catherine Gebotys. An efficient fault-tolerance technique for the keyed-hash message authentication code. In *Aerospace Conference, 2010 IEEE*, pages 1–17. IEEE, 2010.

[54] A Kerckhoffs. La cryptographie militaire (military cryptography),|| j. *Sciences Militaires (J. Military Science, in French)*, 1883.

[55] Zeen Kim, Jangseong Kim, Kwangjo Kim, Imsung Choi, and Taeshik Shon. Untraceable and serverless rfid authentication and search protocols. In *Parallel and Distributed Processing with Applications Workshops (ISPAW), 2011 Ninth IEEE International Symposium on*, pages 278–283. IEEE, 2011.

[56] Paul Kocher, Ruby Lee, Gary McGraw, Anand Raghunathan, and Srivaths Moderator-Ravi. Security as a new dimension in embedded system design. In *Proceedings of the 41st annual Design Automation Conference*, pages 753–760. ACM, 2004.

[57] Chin-Feng Lee, Hung-Yu Chien, and Chi-Sung Laih. Server-less rfid authentication and searching protocol with enhanced security. *International Journal of Communication Systems*, 25(3):376–385, 2012.

[58] Yong Lee and Ingrid Verbauwhede. Secure and low-cost rfid authentication protocols. In *2nd IEEE International Workshop on Adaptive Wireless Networks (AWiN 2005)*, pages 1–5, 2005.

[59] Luon-Chang Lin, Shih-Chang Tsaur, and Kai-Ping Chang. Lightweight and serverless rfid authentication and search protocol. In *2009 Second International Conference on Computer and Electrical Engineering*, volume 2, pages 95–99, 2009.

[60] Gang Lu, Bhaskar Krishnamachari, and Cauligi S Raghavendra. Performance evaluation of the ieee 802.15. 4 mac for low-rate low-power wireless networks. In *Performance, Computing, and Communications, 2004 IEEE International Conference on*, pages 701–706. IEEE, 2004.

[61] Parikshit N Mahalle, Bayu Anggorojati, Neeli Rashmi Prasad, and Ramjee Prasad. Identity establishment and capability based access control (iecac) scheme for internet of things. In *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, pages 187–191. IEEE, 2012.

[62] Collins Mtita, Maryline Laurent, and Pascal Daragon. Serverless lightweight mutual authentication protocol for small mobile computing devices. In *New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on*, pages 1–5. IEEE, 2015.

[63] Collins Mtita, Maryline Laurent, and Jacques Delort. Efficient serverless radio-frequency identification mutual authentication and secure tag search protocols with untrusted readers. In *IET Information Security, Special Issue on Lightweight and Energy-Efficient Security Solutions for Mobile Computing Devices*, pages 1–11. The Institution of Engineering and Technology, 2016.

[64] Geoff Mulligan. The 6lowpan architecture. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 78–82. ACM, 2007.

[65] B Clifford Neuman and Theodore Ts' O. Kerberos: An authentication service for computer networks. *Communications Magazine, IEEE*, 32(9):33–38, 1994.

[66] Huansheng Ning, Hong Liu, and Laurence T Yang. Aggregated-proof based hierarchical authentication scheme for the internet of things. *Parallel and Distributed Systems, IEEE Transactions on*, 26(3):657–667, 2015.

[67] Afif Osseiran, Federico Boccardi, Volker Braun, Katsutoshi Kusume, Patrick Marsch, Michal Maternia, Olav Queseth, Malte Schellmann, Hans Schotten, Hidekazu Taoka, et al. Scenarios for 5g mobile and wireless communications: the vision of the metis project. *Communications Magazine, IEEE*, 52(5):26–35, 2014.

[68] Veronika del Carmen Peralta Costabel. Data freshness and data accuracy: a state of the art. *Reportes Técnicos 06-13*, 2006.

[69] Trevor Pering, Yuvraj Agarwal, Rajesh Gupta, and Roy Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232. ACM, 2006.

[70] Adrian Perrig, Robert Szewczyk, Justin Douglas Tygar, Victor Wen, and David E Culler. Spins: Security protocols for sensor networks. *Wireless networks*, 8(5):521–534, 2002.

[71] Hauke Petersen, Emmanuel Baccelli, and Matthias Wählisch. Interoperable services on constrained devices in the internet of things. In *W3C Workshop on the Web of Things*, 2014.

[72] Andreas Pfitzmann and Marit Hansen. Anonymity, unlinkability, unobservability, pseudonymity, and identity management-a consolidated proposal for terminology.

[73] Raphael C-W Phan, Jiang Wu, Khaled Ouafi, and Douglas R Stinson. Privacy analysis of forward and backward untraceable rfid authentication schemes. *Wireless Personal Communications*, 61(1):69–81, 2011.

[74] Selwyn Piramuthu. Rfid mutual authentication protocols. *Decision Support Systems*, 50(2):387–393, 2011.

[75] Joseph Polastre, Jason Hill, and David Culler. Versatile low power media access for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 95–107. ACM, 2004.

[76] Nachiketh R Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 30–35. ACM, 2003.

[77] Gregory J Pottie and William J Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58, 2000.

[78] Mohsen Pourpouneh, Rasoul Ramezanian, and Fatemeh Salahi. An improvement over a server-less rfid authentication protocol. *International Journal of Computer Network and Information Security (IJCNIS)*, 7(1):31, 2014.

[79] NIST FIPS Pub. 197. *Announcing the Advanced Encryption Standard (AES)*, 2001.

[80] Charles Rackoff and Daniel R Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology - CRYPTO'91*, pages 433–444. Springer, 1991.

[81] Anand Ranganathan, Jalal Al-Muhtadi, and Roy H Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, (2):62–70, 2004.

[82] Biplob Ray, Morshed Howdhury, Jemal Abawajy, and Monika Jesmin. Secure object tracking protocol for networked rfid systems. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on*, pages 1–7. IEEE, 2015.

[83] X Recommendation. 800, security architecture for open systems interconnection for ccitt applications. *International Telecommunication Union (ITU)*, 1991.

[84] Melanie R Rieback, Bruno Crispo, Andrew S Tanenbaum, et al. The evolution of rfid security. *IEEE Pervasive Computing*, 5(1):62–69, 2006.

[85] Frederick J Riggins and Samuel Fosso Wamba. Research directions on the adoption, usage, and impact of the internet of things through the use of big data analytics. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 1531–1540. IEEE, 2015.

[86] Masoumeh Safkhani, Nasour Bagheri, Pedro Peris-Lopez, and Aikaterini Mitrokotsa. On the traceability of tags in suap rfid authentication protocols. In *RFID-Technologies and Applications (RFID-TA), 2012 IEEE International Conference on*, pages 292–296. IEEE, 2012.

[87] Masoumeh Safkhani, Pedro Peris-Lopez, Nasour Bagheri, Majid Naderi, and Julio Cesar Hernandez-Castro. On the security of tan et al. serverless rfid authentication and search protocols. In *Radio Frequency Identification. Security and Privacy Issues*, pages 1–19. Springer, 2013.

[88] Debashis Saha and Amitava Mukherjee. Pervasive computing: a paradigm for the 21st century. *Computer*, 36(3):25–31, 2003.

[89] Mahadev Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.

[90] B Schechter. Seeing the light: Ibm's vision of life beyond the pc. *IBM research*, (2), 2000.

[91] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15, 1998.

[92] Z Shelby, S Chakrabarti, and E Nordmark. Neighbor discovery optimization for low power and lossy networks (6lowpan). *Work in progress, IETF draft-ietf-6lowpan-nd-18*, 2011.

[93] Zach Shelby and Carsten Bormann. *6LoWPAN: The wireless embedded Internet*, volume 43. John Wiley & Sons, 2011.

[94] Elaine Shi and Adrian Perrig. Designing secure sensor networks. *Wireless Communications, IEEE*, 11(6):38–43, 2004.

[95] Mark Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011.

[96] Jennifer G Steiner, B Clifford Neuman, and Jeffrey I Schiller. Kerberos: An authentication service for open network systems. In *USENIX Winter*, pages 191–202, 1988.

[97] Hui Suo, Jiafu Wan, Caifeng Zou, and Jianqi Liu. Security in the internet of things: a review. In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, volume 3, pages 648–651. IEEE, 2012.

[98] Paul Syverson. Limitations on design principles for public key protocols. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 62–72. IEEE, 1996.

[99] Chiu Chiang Tan, Bo Sheng, and Qun Li. Serverless search and authentication protocols for rfid. In *Pervasive Computing and Communications, 2007. PerCom'07. Fifth Annual IEEE International Conference on*, pages 3–12. IEEE, 2007.

[100] Arnaud Tisserand. Circuits électroniques pour la génération de nombres aléatoires. *Techniques de l'ingénieur Technologies des composants*, (H5215), 2014.

[101] J Toldinas, V Štuikys, R Damaševičius, and G Ziberkas. Application-level energy consumption in communication models for handhelds. *Elektronika ir Elektrotechnika*, 94(6):73–76, 2015.

[102] Gene Tsudik. Ya-trap: Yet another trivial rfid authentication protocol. In *Pervasive Computing and Communications Workshops, 2006. PerCom Workshops 2006. Fourth Annual IEEE International Conference on*, pages 4–pp. IEEE, 2006.

[103] Pim Tuyls and Lejla Batina. Rfid-tags for anti-counterfeiting. In *Topics in cryptology–CT-RSA 2006*, pages 115–131. Springer, 2006.

[104] David von Oheimb. The high-level protocol specification language hlpsl developed in the eu project avispa. In *Proceedings of APPSEM 2005 Workshop*, 2005.

[105] Arvinderpas S Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328. IEEE, 2005.

[106] Tae Youn Won, Ji Young Chun, and Dong Hoon Lee. Strong authentication protocol for secure rfid tag search without help of central database. In *Embedded and Ubiquitous Computing, 2008. EUC'08. IEEE/IFIP International Conference on*, volume 2, pages 153–158. IEEE, 2008.

[107] Wei Xie, Lei Xie, Chen Zhang, Qiang Wang, Jian Xu, Quan Zhang, and Chaojing Tang. Rfid seeking: Finding a lost tag rather than only detecting its missing. *Journal of Network and Computer Applications*, 42:135–142, 2014.

[108] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):500–528, 2006.

[109] Yao Zou and Guibin Wang. Intercept behavior analysis of industrial wireless sensor networks in the presence of eavesdropping attack. *2015 IEEE Transactions on Industrial Informatics*, 2015.

Appendix A

Formal Protocol Verification Using AVISPA Tool

THIS APPENDIX presents a validation of three protocols using an *Automated Validation of Internet Security Protocols and Applications (AVISPA)* [14] tool. AVISPA is a tool for building and analyzing security protocols. This tool provides a role-based, expressive formal language for protocol specification and integrates four different back-ends, which perform the actual analysis of the protocol.

AVISPA uses a formal specification language called High Level Protocol Specification Language (HLPSL). It is necessary to model the respective protocol and the intruder in HLPSL before executing and verifying the protocol in AVISPA tool [104].

As depicted in the Figure A.1, AVISPA's current version integrates four back-ends, *On-the-fly Model-Checker (OFMC)*, *Constraint-Logic-based Attack Searcher (CL-AtSe)*, *SAT-based Model-Checker (SATMC)*, and *Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP)*. The attacker model is *Dolev-Yao* [34] i.e an attacker has full capabilities over the newtork and can listen or intercept communication, inject new messages or modify messages in transit.

The protocols we validate here are *Authenticating Cluster Devices with Untrusted Parties using a Serverless Paradigm*, discussed in chapter 3, and *Secure Serverless Search & Authentication Protocols for very Constrained Devices* discussed in chapter 4.

A.1 Authenticating Cluster Devices with Untrusted Parties using a Serverless Paradigm

The protocol consists of three players, the centralized server *CS*, Lightweight Initiator (*LI*), and the Lightweight Responder (*LR*). We display independent and easily understood chunks of code for each party and then we present the complete code in Listing A.12.

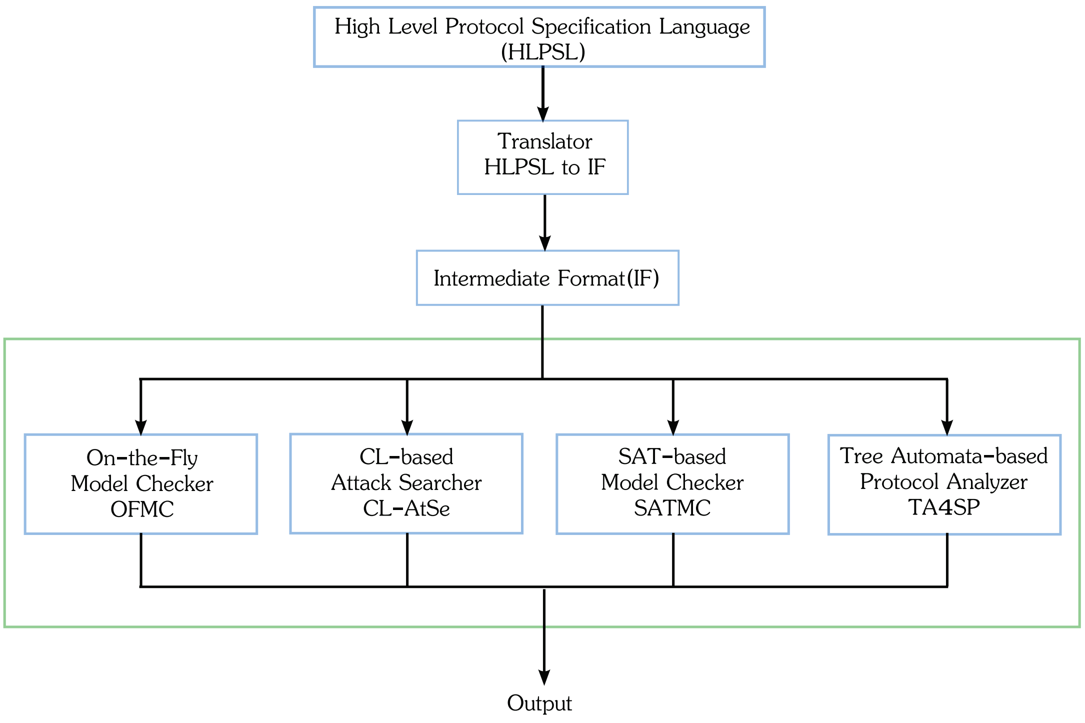

High Level Protocol Specification Language (HLPSL)
Translator HLPSL to IF
Intermediate Format(IF)
On-the-Fly Model Checker OFMC
CL-based Attack Searcher CL-AtSe
SAT-based Model Checker SATMC
Tree Automata-based Protocol Analyzer TA4SP
Output

FIGURE A.1 - AVISPA's Architecture

The code for LI presented in Listing A.1 then followed by code for LR is presented in Listing A.2.

A.1.1 HLPSL code for LI

The Lightweight Initiator LI communicates with the server and the Lightweight Responder LR, at different times. The code shows different sessions and channels of communications simulating the exchanges between S and LI and then between LI and LR. The code is shown in Listing A.1. This code is meant to prove the secrecy of two keys i.e., the key K_S between LI and LR and the key K'_S between LI and LR.

Listing A.1: HLPSL code for LI

```
%ROLE LI
role alice (             A, S, B: agent,
                               K: symmetric_key,
                        Succ, H1: hash_func,
                 M, AR, Na1, Ws : text,
                SND_SA, RCV_SA,
                SND_BA, RCV_BA  : channel(dy) )
played_by A
def=
local      Lj, Na, R1  : text,
                 State : nat,
       Idr, Ks, Kd, Hk : symmetric_key,
                     H : hash_func
const              id  : text,
     alice_bob_kd, ks,
      bob_alice_kd, kd : protocol_id
init  State := 0
transition
1.   State = 0   /\ RCV_SA(start) =|>
     State':= 2  /\ SND_SA( A.B.id.M)
2.   State = 2   /\ RCV_SA( A.B.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K )=|>
     State' := 4 /\ Na' := new()
                 /\ SND_BA(A.B.{id}_Hk'.AR'.Ws'.Na'.Na1')
                 /\ request(A, B, bob_alice_kd, Kd)
3.   State = 4   /\ RCV_BA(A.B.{R1'}_Idr.H1(Kd'.R1').H1(Na))=|>
     State':= 6  /\ SND_BA(A.B.H1(R1'))
                 /\ Ks' := H(Na.R1') /\ witness(A, B, bob_alice_kd, H1)
                 /\ secret(Ks, ks, {A, B})
end role
```

A.1.2 HLPSL code for LR

Lightweight Responder LR only communicates with LI. The code in Listing A.2 shows different sessions and channels of communications simulating exchanges between LR and LI. The code proves the secrecy of the session key K_S between LR and LI.

Listing A.2: HLPSL code for LR

```
%ROLE LR
role bob(            A, B : agent,
                 Succ, H1 : hash_func,
              AR, Na1, Ws : text,
                       Kc : symmetric_key,
           SND_AB, RCV_AB : channel(dy))
played_by B
def=
local               State : nat,
          Idr, Hk, Ks, Kd : symmetric_key,
                        H : hash_func,
                 Na, R1   : text
init State := 5
transition
1. State = 5 /\ RCV_AB(A.B.{id}_Hk.AR'.Ws'.Na'.Na1')=|>
   State':=7 /\ R1':= new() /\ Kd':=H(Kc.id.AR.Ws)
             /\ Idr':=H(id.Na1')/\ SND_AB(A.B.{R1'}_Idr.H1(Kd'.R1').H1(Na))
             /\ witness(B, A, bob_alice_kd, Kd')
             /\ request(B, A, alice_bob_kd , H1)
2. State = 7 /\ RCV_AB(A.B.H1(R1')) =|>
State' := 9  /\ Ks' := H(Na.R1)
             /\ secret(Ks', ks, {A,B})
end role
```

A.1.3 HLPSL code for Server

The code for the server shows the communication between the server and LI while including the LR as it is passively invoked for granting access rights and authorization to LI. The code is meant to prove the secrecy of the key between S and LI, denoted as Kd.

Listing A.3: HLPSL code for Server

```
role server (     A, B, S  : agent,
                    K, Kc  : symmetric_key,
                        M  : text,
            SND_AS, RCV_AS : channel(dy))
played_by S
def=
     local               State  : nat,
                Na1, Lj, AR, Ws : text,
                        Hk, Kd  : symmetric_key,
                         Hash   : hash_func
init State  := 1
transition
     1. State  = 1   /\ RCV_AS(A.{id.M}_K) =|>
         State':=3   /\ Ws' := new() /\ AR' := new()
                     /\ Kd' := Hash(Kc.id.AR'.Ws')
                     /\ Lj' := new()/\ Hk' := new()/\ Na1':= new()
                     /\ SND_AS(A.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K)
                     /\ secret(Kd', bob_alice_kd, {A,S,B})
end role
```

A.1.4 Full HLPSL protocol code

This section presents a full code showing the combination of different players and the general session and environment code for the candidate protocol.

Listing A.4: Full HLPSL protocol code

```
%ROLE LI
role alice (             A, S, B: agent,
                               K: symmetric_key,
                        Succ, H1: hash_func,
                 M, AR, Na1, Ws : text,
                SND_SA, RCV_SA,
                SND_BA, RCV_BA  : channel(dy) )
played_by A
def=
local      Lj, Na, R1  : text,
                 State : nat,
       Idr, Ks, Kd, Hk : symmetric_key,
                     H : hash_func
const              id  : text,
     alice_bob_kd, ks,
      bob_alice_kd, kd : protocol_id
init  State := 0
transition
1.   State = 0   /\ RCV_SA(start) =|>
     State':= 2  /\ SND_SA( A.B.id.M)
2.   State = 2   /\ RCV_SA( A.B.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K )=|>
     State' := 4 /\ Na' := new()
                 /\ SND_BA(A.B.{id}_Hk'.AR'.Ws'.Na'.Na1')
                 /\ request(A, B, bob_alice_kd, Kd)
3.   State = 4   /\ RCV_BA(A.B.{R1'}_Idr.H1(Kd'.R1').H1(Na))=|>
     State':= 6  /\ SND_BA(A.B.H1(R1'))
                 /\ Ks' := H(Na.R1') /\ witness(A, B, bob_alice_kd, H1)
                 /\ secret(Ks, ks, {A, B})
end role
```

```
%%ROLE SERVER
role server (     A, B, S  : agent,
                    K, Kc  : symmetric_key,
                        M  : text,
            SND_AS, RCV_AS : channel(dy))
played_by S
def=
     local              State  : nat,
                Na1, Lj, AR, Ws : text,
                        Hk, Kd  : symmetric_key,
                         Hash   : hash_func
init State  := 1
transition
     1. State  = 1   /\ RCV_AS(A.{id.M}_K) =|>
         State':=3   /\ Ws' := new() /\ AR' := new()
                     /\ Kd' := Hash(Kc.id.AR'.Ws')
                     /\ Lj' := new()/\ Hk' := new()/\ Na1':= new()
                     /\ SND_AS(A.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K)
                     /\ secret(Kd', bob_alice_kd, {A,S,B})
end role

%ROLE LR
role bob(            A, B : agent,
                 Succ, H1 : hash_func,
              AR, Na1, Ws : text,
                       Kc : symmetric_key,
           SND_AB, RCV_AB : channel(dy))
played_by B
def=
local              State : nat,
          Idr, Hk, Ks, Kd : symmetric_key,
                        H : hash_func,
                 Na, R1   : text
init State := 5
transition
1. State = 5 /\ RCV_AB(A.B.{id}_Hk.AR'.Ws'.Na'.Na1')=|>
   State':=7 /\ R1':= new() /\ Kd':=H(Kc.id.AR.Ws)
             /\ Idr':=H(id.Na1')/\ SND_AB(A.B.{R1'}_Idr.H1(Kd'.R1').H1(Na))
             /\ witness(B, A, bob_alice_kd, Kd')
             /\ request(B, A, alice_bob_kd , H1)
2. State = 7 /\ RCV_AB(A.B.H1(R1')) =|>
State' := 9  /\ Ks' := H(Na.R1) /\ secret(Ks', ks, {A,B})
end role

%%ROLE SESSION
role session (                  A, S, B : agent,
                          M, AR, Na1, Ws : text,
                                 Succ, H : hash_func,
                                   K, Kc : symmetric_key)
def=
  local
                              SSA, RSA,
                              SBA, RBA,
                              SAS, RAS,
                              SAB, RAB   : channel(dy)

  composition
        alice (A, S, B, K, Succ, H, M, AR, Na1, Ws, SSA, RSA, SBA, RBA)
     /\ server(A, B, S, K, Kc, M, SAS, RAS)
     /\ bob(A, B, Succ, H, AR, Na1, Ws, Kc,  SAB, RAB)

end role
```

```
role environment()
def=
const                 a, s, b : agent,
          ksi, kc, k, ka, ki  : symmetric_key,
                   succ,  h   : hash_func,
         id, m, ar, na1, ws   : text,
 bob_alice_r1, alice_bob_na   : protocol_id

intruder_knowledge = {a, b, s, ki, ws, ar, na1, m, succ, ksi}
composition
     session(a, s, b, m, ar, na1, ws, succ, h,  k, kc)
     /\ session(i, s, b, m, ar, na1, ws, succ, h, ki, kc)
end role

goal
 %secrecy of the shared keys
     secrecy_of ks, kd
     authentication_on bob_alice_kd
     authentication_on alice_bob_na
end goal
environment()
```

A.2 Secure Serverless Search & Authentication Protocols for very Constrained Devices

This section provides the HLPSL code for the protocols in our second contribution in chapter 4. The contribution provides two protocols, a serverless mutual authentication protocol and a serverless secure tag search protocol. We start by the authentication protocol and then we look into the search protocol.

A.2.1 AVISPA validation for the serverless mutual authentication protocol

We present the AVISPA validation for the serverless mutual authentication protocol described in section 4.4. For clarity, we show chunks of code for each player and then we list complete code for validation of the protocol. The code for the Reader is presented in Annex A.2.1.1 then followed by the Tag in Annex A.2.1.2 and finally the Server in Annex A.2.2.3. The rest of the code is presented in Annex A.2.1.4.

A.2.1.1 HLPSL code for the RFID Reader

The RFID Reader R_j communicates with the central server and the tag, at different times. The code shows different sessions and channels of communications simulating exchanges with the central server and the reader and then between the reader and the tag. The code is shown in Listing A.5. The code proves the secrecy of the key K_S denoted as $secret(K_S, ks, \{A, B\})$ between Reader and Tag.

Listing A.5: HLPSL code for the RFID Reader

```
%ROLE Reader
role alice (            A, S, B    : agent,
                              K    : symmetric_key,
                        Succ, H1   : hash,
                    M, AR, Na1, Ws : text,
    SND_SA, RCV_SA, SND_BA, RCV_BA  : channel(dy))
played_by A
def=
local
                Lj, Na, Ok, R1,T1  : text,
                            State  : nat,
                  Idr, Ks, Kd, Hk  : symmetric_key,
                                H  : hash
const                          id  : text,
 alice_bob_kd, ks, bob_alice_kd, kd : protocol_id
 init  State := 0
transition
 1. State   = 0 /\ RCV_SA(start) =|>
    State' := 2 /\ SND_SA(A.B.id.M)
 2. State   = 2 /\ RCV_SA(A.B.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K) =|>
    State' := 4 /\ Na' := new()
                /\ SND_BA(A.B.AR'.Ws'.Na')
                /\ request(A, B, bob_alice_kd, Kd)
 3. State  = 4  /\ RCV_BA(A.B.H1(id.Na'.Na1').Na') =|>
    State' := 6 /\ SND_BA(A.B.H1(T1'.R1'.Ws').T1)
                /\ Ks' := H(T1.Na1.Ws')
                /\ witness(A, B, bob_alice_kd, H1)
                /\ secret(Ks, ks, {A,B})
end role
```

A.2.1.2 HLPSL code for the Tag

The RFID Tag communicates with the Reader. The code in Listing A.5 shows different sessions and channels of communications simulating exchanges between the Reader and the Tag. Like the code in section A.2.1.1, the code in Listing A.6 also proves the secrecy of the key K_S denoted as $secret(K_S, ks, \{A, B\})$ between Reader and Tag

Listing A.6: HLPSL code for the RFID Tag

```
%ROLE Tag
role bob(             A, B    : agent,
                  Succ, H1    : hash,
               AR, Na1, Ws    : text,
                        Kc    : symmetric_key,
            SND_AB, RCV_AB    : channel(dy))
played_by B
def=
local                State    : nat,
             Idr, Hk, Ks, Kd  : symmetric_key,
                           H  : hash,
              Na, Ok, R1, T1  : text
init State  := 5
transition
1. State  =  5   /\ RCV_AB(A.B.AR'.Ws'.Na') =|>
   State' :=  7  /\ Na1' := new() /\ Kd' := H(id.AR.Ws)
                 /\ SND_AB(A.B.H1(Kd'.Na1'.Na').Na1)
                 /\ witness(B, A, bob_alice_kd,  Kd')
```

```
                  /\ request(B, A, alice_bob_kd , H1 )
2. State   = 7    /\ RCV_AB(A.B.H1(R1')) =|>
   State' := 9    /\ Ks' := H(Kd'.T1'.Na'.Ws') /\ secret(Ks', ks, {A,B})
end role
```

A.2.1.3 HLPSL code for the Central Server

The code in Listing A.7 shows the communication between the Server S and RFID Reader while including the Tag as it is passively invoked for granting access rights and authorization to Reader. The code provdes the secrecy of the key Kd used between S and the Reader R_j.

Listing A.7: HLPSL code for Central Server

```
%%ROLE SERVER
role server (        A, B, S        : agent,
                             K, Kc  : symmetric_key,
                                 M  : text,
                     SND_AS, RCV_AS : channel(dy))
played_by S
def=
local                        State  : nat,
                  Na1, Lj, AR, Ws   : text,
                        Hk, Kd      : symmetric_key,
                            Hash    : hash
init State  := 1
transition

1. State  = 1   /\ RCV_AS(A.{id.M}_K) =|>
   State':= 3   /\ Ws' := new() /\ AR' := new()
                /\ Kd' := Hash(Kc.id.AR'.Ws')
                /\ Lj' := new()
                /\ Hk' := new()
                /\ Na1':= new()
                /\ SND_AS(A.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K)
                /\ secret(Kd', bob_alice_kd, {A,S,B})
end role
```

A.2.1.4 Full HLPSL protocol code for the mutual authentication protocol

This section presents a full code showing the combination of different players and the general session and environment code for the candidate protocol. The code is depicted in Listing A.8.

Listing A.8: Full HLPSL protocol code for the serverless mutual authentication protocol

```
%Role Reader
role alice (         A, S, B    : agent,
                           K    : symmetric_key,
                     Succ, H1   : hash,
                 M, AR, Na1, Ws : text,
SND_SA, RCV_SA, SND_BA, RCV_BA  : channel(dy))
```

```
played_by A
def=
local        Lj, Na, Ok, R1,T1  : text,
                        State  : nat,
               Idr, Ks, Kd, Hk  : symmetric_key,
                            H  : hash
     const                 id  : text,

 alice_bob_kd, ks, bob_alice_kd, kd : protocol_id
     init  State := 0
 transition
 1. State   = 0 /\ RCV_SA(start) =|>
    State' := 2 /\ SND_SA(A.B.id.M)
 2. State   = 2 /\ RCV_SA(A.B.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K) =|>
    State' := 4 /\ Na' := new()
                /\ SND_BA(A.B.AR'.Ws'.Na')
                /\ request(A, B, bob_alice_kd, Kd)
 3. State   = 4 /\ RCV_BA(A.B.H1(id.Na'.Na1').Na') =|>
    State' := 6 /\ SND_BA(A.B.H1(T1'.R1'.Ws').T1)
                /\ Ks' := H(T1.Na1.Ws')
                /\ witness(A, B, bob_alice_kd, H1)
                /\ secret(Ks, ks, {A,B})
end role

%%ROLE SERVER
role server (       A, B, S        : agent,
                            K, Kc  : symmetric_key,
                                M  : text,
                    SND_AS, RCV_AS : channel(dy))
played_by S
def=
local                       State  : nat,
                  Na1, Lj, AR, Ws  : text,
                        Hk, Kd     : symmetric_key,
                            Hash   : hash
init State  := 1
transition

1. State  = 1   /\ RCV_AS(A.{id.M}_K) =|>
   State':= 3   /\ Ws' := new() /\ AR' := new()
                /\ Kd' := Hash(Kc.id.AR'.Ws')
                /\ Lj' := new()
                /\ Hk' := new()
                /\ Na1':= new()
                /\ SND_AS(A.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K)
                /\ secret(Kd', bob_alice_kd, {A,S,B})
end role

%%ROLE Tag
role bob(             A, B    : agent,
                  Succ, H1    : hash,
               AR, Na1, Ws    : text,
                        Kc    : symmetric_key,
            SND_AB, RCV_AB    : channel(dy))
played_by B
def=
local                 State   : nat,
             Idr, Hk, Ks, Kd  : symmetric_key,
                           H  : hash,
              Na, Ok, R1, T1  : text
init State  := 5
transition
1. State  =  5   /\ RCV_AB(A.B.AR'.Ws'.Na') =|>
```

```
   State' :=  7  /\ Na1' := new() /\ Kd' := H(id.AR.Ws)
                 /\ SND_AB(A.B.H1(Kd'.Na1'.Na').Na1)
                 /\ witness(B, A, bob_alice_kd,  Kd')
                 /\ request(B, A, alice_bob_kd , H1 )
2. State   = 7   /\ RCV_AB(A.B.H1(R1')) =|>
   State' := 9   /\ Ks' := H(Kd'.T1'.Na'.Ws') /\ secret(Ks', ks, {A,B})
end role

role environment()
def=
   const                 a, s, b      : agent,
                  ksi, kc, k, ka, ki  : symmetric_key,
                        succ,  h      : hash,
              id, m, ar, na1, ws      : text,
      bob_alice_r1, alice_bob_na      : protocol_id
intruder_knowledge = {a, b, s, ki, ws, ar, na1, m, succ, ksi}
composition
        session(a, s, b, m, ar, na1, ws, succ, h,  k, kc)
     /\ session(i, s, b, m, ar, na1, ws, succ, h, ki, kc)
end role
goal
 %secrecy of the shared keys
 secrecy_of ks, kd
 authentication_on bob_alice_kd
 authentication_on alice_bob_na
end goal
environment()
```

A.2.2 AVISPA validation for the serverless secure tag search protocol

We present the AVISPA validation for the serverless secure tag search protocol described in section 4.5. The chunks of code for each player are presented before the complete HLPSL code for the protocol is presented. The code for the Reader is presented in Annex A.2.2.1 then followed by the Tag in Annex A.2.2.2 and finally the Server in Annex A.2.2.3. The rest of the code is presented in Annex A.2.2.4.

A.2.2.1 HLPSL code for the RFID Reader

The RFID Reader R_j communicates with the central server and the tag, at different times. The code shows different sessions and channels of communications simulating exchanges with the central server and the reader and then between the reader and the tag. The code is shown in Listing A.9. The code proves the secrecy of the keys K_S between the tag and the reader and the key Kd between the reader and the Central Server.

Listing A.9: HLPSL code for the RFID Reader

```
role alice (           A, S, B    : agent,
                            K     : symmetric_key,
                       Succ, H1   : hash,
                   M, AR, Na1, Ws : text,
  SND_SA, RCV_SA, SND_BA, RCV_BA  : channel(dy))

played_by A
def=
local         Lj, Na, Ok, R1,T1  : text,
                          State  : nat,
                 Idr, Ks, Kd, Hk  : symmetric_key,
                              H  : hash
     const                   id  : text,

alice_bob_kd, ks, bob_alice_kd, kd : protocol_id

 init  State := 0
 transition
 1. State   = 0 /\ RCV_SA(start) =|>
    State' := 2 /\ SND_SA(A.B.id.M)

 2. State   = 2 /\ RCV_SA(A.B.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K) =|>
    State' := 4 /\ T1' := new()
                /\ SND_BA(A.B.AR'.Ws'.H1(Kd'.T1').T1')
                /\ request(A, B, bob_alice_kd, Kd)

 3. State   = 4 /\ RCV_BA(A.B.H1(Kd'.T1'.Na').Na') =|>
    State' := 6 /\ Ks' := H(T1.Na1.Ws')
                /\ witness(A, B, bob_alice_kd, H1)
                /\ secret(Ks, ks, {A,B})
end role
```

A.2.2.2 HLPSL code for the Tag

The RFID Tag communicates with the Reader. The code in Listing A.10 shows different sessions and channels of communications simulating exchanges between the Reader and the Tag. The code is meant to prove the secrecy of the key K_S between tag and reader as $secret(Ks', ks, \{A, B\}$.

Listing A.10: HLPSL code for the RFID Tag

```
%ROLE Tag
role bob(
                 A, B    : agent,
             Succ, H1    : hash,
          AR, Na1, Ws    : text,
                   Kc    : symmetric_key,
       SND_AB, RCV_AB    : channel(dy))

played_by B
def=
local            State   : nat,
        Idr, Hk, Ks, Kd  : symmetric_key,
                      H  : hash,
         Na, Ok, R1, T1  : text

init State  := 5

transition

1. State   =  5 /\ RCV_AB(A.B.AR'.Ws'.H1(Kd'.T1').T1') =|>

   State' :=  7 /\ Na' := new() /\ Kd' := H(id.AR.Ws)
                /\ SND_AB(A.B.H1(Kd'.T1'.Na').Na')
                /\ witness(B, A, bob_alice_kd,  Kd')
                /\ request(B, A, alice_bob_kd , H1 )
```

```
2. State   = 7  /\ RCV_AB(A.B.H1(R1')) =|>
   State' := 9  /\ Ks' := H(Kd'.T1'.Na'.Ws') /\ secret(Ks', ks, {A,B}
end role
```

A.2.2.3 HLPSL code for the Central Server

The code in Listing A.11 shows the communication between the Server S and RFID Reader while including the Tag as it is passively invoked for granting access rights and authorization to Reader. The code is meant to prove the secrecy of the key Kd between the Reader and the Central Server.

Listing A.11: HLPSL code for the Central Server

```
%%ROLE SERVER
role server (
                   A, B, S        : agent,
                           K, Kc  : symmetric_key,
                               M  : text,
                   SND_AS, RCV_AS : channel(dy))
played_by S
def=
local
                        State  : nat,
               Na1, Lj, AR, Ws : text,
                    Hk, Kd     : symmetric_key,
                        Hash   : hash
init State  := 1
transition
1. State  = 1  /\ RCV_AS(A.{id.M}_K) =|>
   State':= 3  /\ Ws' := new() /\ AR' := new()
               /\ Kd' := Hash(Kc.id.AR'.Ws')
               /\ Lj' := new()
               /\ Hk' := new()
               /\ Na1':= new()
               /\ SND_AS(A.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K)
               /\ secret(Kd', bob_alice_kd, {A,S,B})
end role
```

A.2.2.4 Full HLPSL protocol code for the secure tag search protocol

This section presents a full code showing the combination of different players and the general session and environment code for the candidate protocol.

Listing A.12: Full HLPSL protocol code for the serverless tag search authentication protocol

```
%Role Reader
role alice (           A, S, B    : agent,
                             K    : symmetric_key,
                       Succ, H1   : hash,
                   M, AR, Na1, Ws : text,
  SND_SA, RCV_SA, SND_BA, RCV_BA  : channel(dy))

played_by A
```

```
def=
local          Lj, Na, Ok, R1,T1  : text,
                           State  : nat,
                 Idr, Ks, Kd, Hk  : symmetric_key,
                               H  : hash
     const                    id  : text,

alice_bob_kd, ks, bob_alice_kd, kd : protocol_id

 init  State := 0
 transition
 1. State   = 0 /\ RCV_SA(start) =|>
    State' := 2 /\ SND_SA(A.B.id.M)

 2. State   = 2 /\ RCV_SA(A.B.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K) =|>
    State' := 4 /\ T1' := new()
                /\ SND_BA(A.B.AR'.Ws'.H1(Kd'.T1').T1')
                /\ request(A, B, bob_alice_kd, Kd)

 3. State   = 4 /\ RCV_BA(A.B.H1(Kd'.T1'.Na').Na') =|>
    State' := 6 /\ Ks' := H(T1.Na1.Ws')
                /\ witness(A, B, bob_alice_kd, H1)
                /\ secret(Ks, ks, {A,B})
end role

%%ROLE SERVER
role server (
                   A, B, S        : agent,
                           K, Kc  : symmetric_key,
                               M  : text,
                   SND_AS, RCV_AS : channel(dy))
played_by S
def=
local
                        State  : nat,
               Na1, Lj, AR, Ws : text,
                    Hk, Kd     : symmetric_key,
                        Hash   : hash
init State  := 1
transition
1. State  = 1  /\ RCV_AS(A.{id.M}_K) =|>
   State':= 3  /\ Ws' := new() /\ AR' := new()
               /\ Kd' := Hash(Kc.id.AR'.Ws')
               /\ Lj' := new()
               /\ Hk' := new()
               /\ Na1':= new()
               /\ SND_AS(A.AR'.Ws'.Na1'.{Hk'.Lj'.Kd'}_K)
               /\ secret(Kd', bob_alice_kd, {A,S,B})
end role

%%ROLE TAG
role bob(
                 A, B    : agent,
             Succ, H1    : hash,
          AR, Na1, Ws    : text,
                   Kc    : symmetric_key,
       SND_AB, RCV_AB    : channel(dy))

played_by B
def=
local            State   : nat,
        Idr, Hk, Ks, Kd  : symmetric_key,
                      H  : hash,
         Na, Ok, R1, T1  : text
```

```
init State  := 5

transition

1. State   =  5 /\ RCV_AB(A.B.AR'.Ws'.H1(Kd'.T1').T1') =|>

   State' :=  7 /\ Na' := new() /\ Kd' := H(id.AR.Ws)
                /\ SND_AB(A.B.H1(Kd'.T1'.Na').Na')
                /\ witness(B, A, bob_alice_kd,  Kd')
                /\ request(B, A, alice_bob_kd , H1 )

2. State   = 7  /\ RCV_AB(A.B.H1(R1')) =|>
   State' := 9  /\ Ks' := H(Kd'.T1'.Na'.Ws') /\ secret(Ks', ks, {A,B})

end role

%%ROLE SESSION
role session ( A, S, B           : agent,
               M, AR, Na1, Ws    : text,
                      Succ, H    : hash,
                        K, Kc    : symmetric_key)
def=
    local
                       SSA, RSA,
                       SBA, RBA,
                       SAS, RAS,
                       SAB, RAB   : channel(dy)
composition
         alice (A, S, B, K, Succ, H, M, AR, Na1, Ws, SSA, RSA, SBA, RBA)
      /\  server(A, B, S, K, Kc, M, SAS, RAS)
      /\  bob   (A, B, Succ, H, AR, Na1, Ws, Kc,  SAB, RAB)

end role

role environment()
def=
   const            a, s, b      : agent,
             ksi, kc, k, ka, ki  : symmetric_key,
                   succ,  h      : hash,
         id, m, ar, na1, ws      : text,
 bob_alice_r1, alice_bob_na      : protocol_id

intruder_knowledge = {a, b, s, ki, ws, ar, na1, m, succ, ksi}

composition
        session(a, s, b, m, ar, na1, ws, succ, h,  k, kc)
     /\ session(i, s, b, m, ar, na1, ws, succ, h, ki, kc)
end role

goal
 %secrecy of the shared keys
 secrecy_of ks, kd
 authentication_on bob_alice_kd
 authentication_on alice_bob_na
end goal

environment()
```

Appendix B

Protocol Verification Using CryptoVerif Tool

This chapter presents the CryptoVerif tool [16] for protocol verification. The CryptoVerif tool produces verification in the computational model, which is close to the actual execution of the protocol [17].

The CryptoVerif tool uses a series of games to prove the security of the protocol. The first game corresponds to the actual protocol that needs proof. The goal is to show that the probability of breaking a certain security property is negligible. Every intermediate game is obtained from the precedent one through transformations. The transformations are done in a way that the difference of probability between consecutive games is negligible. In the final game, the attacker has a negligible probability of breaking the security property set to prove, using only the game, without involving cryptographic assumptions [16].

We verify the two serverless protocols discussed in chapter 4 titled *Secure Serverless Search & Authentication Protocols for very Constrained Devices*. We begin with serverless authentication protocol in section B.1 and then we verify the serverless search protocol in section B.2.

B.1 Mutual Authentication Protocol code for CryptoVerif

This section presents the protocol code which corresponds to the CryptoVerif for the verification of security properties. The property proven using the code depicted in Listing B.1 is the secrecy of the session key K_S, which is computed after the mutual authentication session.

Listing B.1: CryptoVerif Code for the Mutual Authentication Protocol

```
param N.
param N0.
param N1.
type key [large,fixed].
type keyseed [large,fixed].
type seed [large,fixed].
type hasht [large,fixed].
type hasht2 [large,fixed].
type nonce [large,fixed].
type keymacs [bounded].
type mkey [bounded].
type mkeyseed [fixed].
proba PPRF.
type ra_num [fixed].
expand PRF(keyseed, key, seed, ra_num, kgen, f, PPRF).
fun concat2(ra_num, nonce):bitstring [compos].
fun concat(ra_num, ra_num):bitstring [compos].
fun concat3(ra_num, nonce, ra_num):bitstring [compos].
type hashkey [fixed].
expand ROM_hash(hashkey, bitstring, hasht, hash).
expand ROM_hash(key, bitstring, hasht, hash2).
proba Penc.
expand IND_CPA_sym_enc(keyseed,key,bitstring,bitstring,seed,kgen2,enc,dec,injbot,Z,Penc).
proba Pmac.
expand SUF_CMA_mac(mkeyseed, mkey, bitstring, bitstring, mkgen, mac, check, Pmac).
channel c0, ca, cstart, cb, cs, c1, c2, c3, c4, c5, c6, c7, c8, start, finish.
(* Queries *)
query secret Ks.
query secret1 Ks.
(* Process for the Reader *)
let processA =
      in(start, (ea: bitstring, ma: bitstring, W : ra_num, AR: ra_num));
      new Na : ra_num;
      new K: key;
      let Kas = kgen(Ks) in
      new ta: nonce;
      if check(ea, mKsb, ma) then  (
               let injbot(Kb) = dec(ea, Ksa) in
               out(c1, (Na, W ));
               in(c2, (Nb: ra_num, cT: hasht));
               if cT = hash(hk2, concat(Nb, Na)) then (
               let Kab:hasht = hash2(Kas, concat3(Nb, ta, W)) in
               let h2:hasht = hash(hk2, concat2(Nb,ta)) in
               out(c5, (ta, h2)))
        ).
(* Process for the Tag *)
let processB =
        in(c3, (Na: ra_num, W:ra_num));
        new Nb : ra_num;
        let Kbs = kgen(Ks) in
        let cT:hasht = hash(hk2, concat(Nb, Na)) in
        out(c4, (Nb, cT));
        in(c6, (ta: nonce, h2:hasht));
        if h2 = hash(hk2, concat2(Nb,ta)) then (
               let Kab:hasht = hash2(Kbs, concat3(Nb, ta, W)) in
               out(finish, ())
        ).
(*Process for the server*)
let processS =
        in(c0, ());
        new id: seed;
        new W: ra_num;
        new AR: ra_num;
        let m1:bitstring = concat(W,AR) in
        let Kb: bitstring = mac(m1, mKsb) in
        let ea1: bitstring = enc(Kb, Ksa, id) in
        let mea1: bitstring = mac(ea1, mKsb) in
        out(ca, (ea1, mea1, W, AR)).
process
        in(c7, ());
        new rmKsb: mkeyseed;
        let mKsb = mkgen(rmKsb) in
        new rKsa: keyseed;
        let Ksa = kgen(rKsa) in
        new Ks: keyseed;
        new hk2: hashkey;
        out(c8, ());
        ((! N processS) |
         (! N0 processA) |
         (! N1 processB))
```

B.2 Secure Tag Search Protocol code for CryptoVerif

The protocol code corresponding to the CryptoVerif specification for the verification of security properties in the secure tag search protocol is depicted in Listing B.2. The protocol code is meant to prove the secrecy of the session key K_S between the tag and the reader during the tag search query.

Listing B.2: CryptoVerif Code for the Secure Tag Search Protocol

```
param N.
param N0.
param N1.
type key [large,fixed].
type keyseed [large,fixed].
type seed [large,fixed].
type hasht [large,fixed].
type hasht2 [large,fixed].
type hasht3 [large,fixed].
type keymacs [bounded].
type mkey [bounded].
type mkeyseed [fixed].
type nonce [large,fixed].
proba PPRF.
type ra_num [fixed].
expand PRF(keyseed, key, seed, ra_num, kgen, f, PPRF).
 fun concat2(ra_num, nonce):bitstring [compos].
fun concat(ra_num, ra_num):bitstring [compos].
fun concat3(ra_num, nonce, ra_num):bitstring [compos].
type hashkey [fixed].
expand ROM_hash(hashkey, bitstring, hasht, hash).
expand ROM_hash(key, bitstring, hasht, hash2).
proba Penc.
expand IND_CPA_sym_enc(keyseed,key,bitstring,bitstring,seed,kgen2,enc,dec,injbot,Z,Penc).
proba Pmac.
expand SUF_CMA_mac(mkeyseed, mkey, bitstring, bitstring, mkgen, mac, check, Pmac).
channel c0, ca, c1, c2, c3, c4, c5, c6, start, finish.
(*query secret Ks.
query secret1 Ks.*)
(* Process for the Reader *)
```

```
let processA =
        in(start, (ea: bitstring, ma: bitstring, W : ra_num, AR: ra_num));
        new Na : ra_num;
        new ta: nonce;
        new W : ra_num;
        if check(ea, mKsb, ma) then
        (
                let injbot(Kb) = dec(ea, Ksa) in
                let Kas = kgen(Ks) in
                out(c1, (ta, W, hash(hk2, concat2(Na, ta))));
                in(c2, (Nb: ra_num, cT: hasht));
                if cT = hash(hk2, concat2(Nb, ta)) then
                        let Kab:hasht = hash2(Kas, concat3(Nb, ta, W)) in
                out(finish, ())
).
(* Process for the Tag *)
let processB =
        in(c3, (ta: nonce, W:ra_num, cT: hasht));
        new Nb : ra_num;
        let Kas = kgen(Ks) in
        if cT = hash(hk2, concat2(Nb, ta)) then
        (
                let Kab:hasht = hash2(Kas, concat3(Nb, ta, W)) in
                let h2:hasht = hash(hk2, concat2(Nb, ta)) in
                out(c4, (Nb, h2))
).
(* Process for the Server *)
let processS =
        in(c0, ());
        new id: seed;
        new W: ra_num;
        new AR: ra_num;
        let m1:bitstring = concat(W,AR) in
        let Kb: bitstring = mac(m1, mKsb) in
        let ea1: bitstring = enc(Kb, Ksa, id) in
        let mea1: bitstring = mac(ea1, mKsb) in
        out(ca, (ea1, mea1, W, AR)).
process
        in(c5, ());
        new rmKsb: mkeyseed;
        let mKsb = mkgen(rmKsb) in
        new rKsa: keyseed;
        let Ksa = kgen(rKsa) in
        new Ks: keyseed;
        new K: key;
        new hk2: hashkey;
        out(c6, ());
        ((! N processS) |
        (! N0 processA) |
        (! N1 processB))
```

Appendix C

Les Protocoles de Sécurité Serverless Légers Pour l'Internet des Objets: Résumé

C.1 Introduction

Les avancées technologiques permettent d'intégrer des capteurs et des modules de communication dans les objets du quotidien pour les rendre intelligents et faciliter leur intégration sur l'Internet. L'Internet du futur sera sans nul doute celui des objets connectés. Les objets connectés génèrent, collectent, stockent et partagent des informations entre eux et aussi avec les serveurs d'authentification centralisés. La plupart des informations collectées doivent être protégées pendant le stockage et le transfert. Par le passé, divers protocoles assurant une sécurité robuste basés sur la cryptographie asymétrique et d'autres sur la cryptographie symétrique ont été proposés dans la littérature. Du fait que les objets connectés possèdent de faibles capacités de calcul, de mémoire et d'énergie, et que l'accès au medium radio est très consommateur en ressources, les protocoles cryptographiques traditionnels ne sont pas adaptés aux objets connectés. Il y a lieu donc d'adapter ou de concevoir des protocoles propres et conformes à leurs exigences.

Dans ce mémoire, nous étudions les différents protocoles serverless légers proposés pour les objets connectés. Nous examinons leurs capacités à résister à diverses attaques et leurs aptitudes à minimiser l'usage des ressources. Après quoi, notre objectif est de proposer des protocoles de sécurité serverless permettant aux objets de s'authentifier tout en garantissant efficacité, passage à l'échelle et efficacité énergétique, l'énergie étant une ressource très critique qui a une influence directe sur la durée de vie d'un objet connecté.
Mots-clès: *Sécurité, Vie privée, Protocoles légers serverless, Authentification mutuelle, Protocoles de recherche sécurisés, Contrôle d'accès, Faible empreinte énergétique*

C.2 Les Objectifs

Pour réaliser cette thèse nous avons identifié les objectifs suivants:

- **Objectif 1**: Rechercher et identifier les caractéristiques et les exigences des protocoles serverless.
- **Objectif 2**: Concevoir un protocole serverless d'authentification mutuelle pour les objets connectés en tenant compte des aspects vie privée (privacy) et l'échange de données sécurisé.
- **Objectif 3**: Concevoir un protocole de recherche serverless sécurisé pour des objets connectés.
- **Objectif 4**: Proposer un guide pour la conception de protocoles serverless adapté aux objets connectés.
- **Objectif 5**: Fournir la validation formelle pour les protocoles de sécurité proposés.
- **Objectif 6**: Fournir une analyse de sécurité informelle basée sur les jeux.

C.3 Pourquoi les Protocoles Serverless ?

Dans cette thèse, nous recherchons, analysons et proposons des solutions dans le cadre des protocoles serverless. Ces protocoles sont s'avèrent idéaux pour les objets connectés. Voici les raisons pour lesquelles les protocoles serverless sont prometteurs.

C.3.1 Connectivité Intermittente

Dans l'Internet des objets, les objets sont connectés par intermittence. Pour des raisons géographiques, certaines régions peuvent être isolées et rester hors ligne pendant des périodes de temps prolongées. Les protocoles de sécurité classiques nécessitent une communication dite *mode connecté* du fait qu'elle est constante entre les clients distants et les serveurs centralisés. Les objets connectés fonctionnant en mode connecté sont obligés de maintenir une connexion persistante au serveur central durant la phase d'authentification. Ce mode de communication consomme beaucoup d'énergie et de bande passante. De plus, la phase d'authentification est particulièrement lente, car elle se fait via un serveur distant [45, 99].

Un autre mode de communication est le mode dit *serverless* du fait que les objets connectés se connectent par intermittence au serveur central pour acquérir des informations qui peuvent lui servir à de futures sessions d'authentification [62, 99]. Les protocoles serverless priorisent la communication locale en permettant d'authentifier les objets en toute sécurité et de partager des informations sans nécessiter une connexion persistante au serveur centralisé distant. Ainsi, les protocoles serverless minimisent la consommation des ressources, tout en optimisant la sécurité et la vie privée des communications entre les appareils situés dans la même proximité géographique [62]. La "vie privée" telle que discutée dans le résumé en français fait référence au terme "privacy" anglais. Elle couvre les problèmes liés à la divulgation de l'identité des appareils à des tiers et à la traçabilité des appareils qu'un tiers pourrait vouloir suivre dans leurs déplacements. En effet, ces appareils sont porteurs d'informations confidentielles (relevés de mesures, géolocalisation,...) pour une entreprise telle que TRAXENS. Ainsi, lorsqu'il est mentionné “a vie privée” des objets, c'est bien entendu un abus de langage signifiant l'intérêt porté aux objets dans la non divulgation d'informations qui pourrait nuire aux propriétaires de ces objets.

C.3.2 Energie

Deux opérations sont particulièrement consommatrices d'énergie dans les objets à ressources contraintes : la communication et le calcul. La communication peut consommer jusqu'à 50% de l'énergie dépensée par l'objet [101]. Pour les objets connectés avec des sources d'énergie limitées, le protocole doit veiller à minimimiser le nombre d'échanges lors de l'établissement de la connexion sécurisée entre les deux parties qui communiquent, et ce pour réduire la consommation d'énergie induite par la transmission et la réception des messages [22]. Les protocoles serverless réduisent considérablement les besoins de communication entre les objets et les serveurs centralisés, ce qui aide à prolonger la durée de vie des batteries dans les objets connectés.

C.3.3 Communication de faible puissance

Maintenir une communication fiable au travers de réseaux sans fil tout en consommant le moins d'énergie possible est le but ultime des objets connectés [75]. Les objets connectés sont équipés de différentes technologies de communication pour leur permettre de mener plusieurs type de communications à la fois. Les communications sont soit de proximité, soit à distance. Les communications longue distance nécessitent de passer par des infrastructures de réseaux, tels que les réseaux cellulaires, principalement axées sur la prestation de Qualité de Service (QoS) et de l'efficacité de la bande passante ; sur ce type de réseau, la consommation d'énergie est secondaire puisque les stations de base bénéficient d'une alimentation illimitée [10]. Les communications longue distance ne sont pas adaptées à la plupart des objets connectés car elles nécessitent beaucoup d'échanges et consomment beaucoup trop d'énergie.

Les communications de proximité sont moins consommatrices d'énergie et sont idéales pour les objets connectés [64, 93]. Dans certaines configurations de réseau, des objets peuvent être alimentés en énergie ou bénéficier de sources d'énergie régulièrement réapprovisionnées. Dans ce cas, ces objets servent de passerelles pour permettre à des objets connectés locaux de joindre un serveur distant par une communication longue distance [21, 92]. Du coup, en utilisant les passerelles, les objets connectés réduisent leur consommation d'énergie parce que toutes les communications sont traitées comme locales, du point de vue énergétique.

C.3.4 Autonomie

L'autonomie est un facteur primordial pour les services fiables dans l'infomatique ubiquitaire. À l'ère de l'informatique ubiquitaire, les objets connectés sont mobiles et autonomes. Il faut leur permettre d'offrir des services à plusieurs entités, sans la nécessité de maintenir un connexion au serveur centralisé. Un contrôle d'accès doit être effectué pour garantir que les accès demandés sont bien autorisés. Le contrôle d'accès peut être fourni par un serveur d'autorisation centralisé et exécuté par l'objet connecté.

C.3.5 Efficacité

Quand il s'agit de la communication, l'efficacité peut être mesurée différemment. Les protocoles serverless éliminent la phase de communication à distance avec un serveur centralisé, tout en optimisant l'authentification de proximité entre plusieurs objets connectés. Par conséquent, le processus d'authentification est plus efficace avec des temps de réponse très courts en raison de l'absence d'intermédiaires.

C.3.6 Disponibilité

La sécurité inclut également la disponibilité et l'intégrité des données. Pour le bon fonctionnement, les protocoles de sécurité classiques dépendent de l'état du réseau ainsi que des serveurs centralisés. Si l'un des deux disfonctionne, l'ensemble du service se trouve interrompu. D'autre part, les protocoles serverless dissocient des processus indépendants en rendant chaque partie autonome et résiliente. Par exemple, une panne de réseau ou un serveur non opérationnel (ou un groupe de noeuds) ne doit affecter qu'un nombre limité d'objets. Dans l'informatique omniprésente, la résilience se traduit par la disponibilité.

C.4 Modèles d'attaque

Les attaques sur la sécurité et la vie privée sont communes à tous les systèmes d'information ; cependant, les menaces de sécurité et de vie privée sont plus facilement réalisables pour les systèmes disposant de peu de ressources, comme les objets pervasifs. En outre, les attaques et les technologies évoluent à peu près au même rythme [84]. Dans cette section, nous analysons les menaces de sécurité classiques applicables aux protocoles serverless.

C.4.1 Attaques Dolev-Yao

Dans le modèle de l'attaquant Dolev-Yao [34], l'attaquant contrôle le réseau i.e. l'attaquant peut entendre, intercepter, et forger tout message dans le réseau. Cependant, l'attaquant n'est pas capable de rompre les schémas cryptographiques sécurisés utilisés dans la construction du protocole. Le modèle Dolev-Yao couvre les attaques suivantes ayant trait aux systèmes pervasifs.

C.4.1.1 Brouillage

Comme les objets connectés utilisent un réseau sans fil comme moyen de communication, ils sont vulnérables aux attaques de brouillage, qui se produisent lorsque du bruit ou de puissants signaux sont introduits sur le canal sans fil entre deux objets communiquant, le but étant de provoquer des interférences dans la communication en cours [31]. Les attaques de brouillage peuvent perturber le flux de communication normal, briser la communication en cours ou perturber le protocole.

C.4.1.2 Ecoute

L'écoute (par espionnage des échanges) est classique dans les communications sans fil du fait que les messages transmis entre deux appareils communicants peuvent être interceptés, capturés et analysés par une personne [109]. Si les échanges ne sont pas correctement protégés, les deux appareils peuvent involontairement révéler des informations secrètes à l'adversaire placé en écoute.

C.4.1.3 Traçabilité

La traçabilité est une violation de la vie privée [25]. La traçabilité se produit lorsqu'un équipement peut être identifié dans un échange de messages ou lorsque plusieurs sessions de communication peuvent être associées au même device [58]. Grâce à la traçabilité, un adversaire peut associer, localiser et identifier les messages et l'objet ou son propriétaire à tout moment, que ce soit en analysant les communications ou l'identification qui est faite des appareils dans les messages.

C.4.1.4 Dé-synchronisation

L'attaque par dé-synchronisation n' d'intérêt que pour les protocoles qui nécessitent la mise à jour des paramètres secrets pendant ou après une procédure d'authentification, afin de maintenir la synchronisation entre les appareils communicants [40]. Dans de tels cas, un adversaire peut intentionnellement empêcher la livraison de certains messages pour induire une incohérence entre les deux parties communicantes. La dé-synchronisation conduit à une attaque en déni de service (DoS) lorsque les dispositifs de communication conservent des informations incohérentes et ne parviennent plus à communiquer dans les sessions en cours ou dans les session suivantes.

C.4.2 Compromission physique

Cette attaque permet de reprogrammer, détruire ou voler des contenus dans un noeud légitime en accédant au logiciel ou au matériel qu'il utilise. Après la capture physique d'un noeud, l'extraction du code et des clés cryptographiques, l'adversaire peut tenter de substituer à ce noeud un « dispositif de remplacement » plus puissant.

C.5 Objectifs de sécurité

C.5.1 Authentification des entités

Elle permet de coopérer au sein des objets connectés sans risque, en contrôlant et en identifiant les participants. Si l'authentification est mal gérée, un attaquant peut se joindre au réseau et injecter des messages erronés. Ainsi le récepteur doit s'assurer que les données utilisées proviennent de la bonne source. L'authentification mutuelle vérifie l'identité des deux parties dans le processus de communication, ce qui est indispensable si l'objectif est d'élever le niveau de confiance entre les parties communicantes.

C.5.2 Confidentialité des données

La confidentialité des données est la garantie que l'information d'un noeud n'est rendue accessible ou révélée qu'à son destinataire. Elle consiste à préserver le secret des messages échangés et ne pas les révéler aux adversaires. Dans les communications, il est important qu'aucun acteur étranger au système ne puisse lire les informations échangées. La confidentialité ne peut être assurée que par l'usage de la cryptographie, soit à clé symétrique soit à clé asymétrique.

C.5.3 Intégrité des données

Elle assure que les données reçues n'ont pas été altérées pendant leur transit dans le réseau de manière volontaire ou accidentelle. Seuls les utilisateurs autorisés doivent pouvoir modifier ou altérer des informations en fonction de leurs niveaux d'accès. Ces autorisations peuvent inclure, les droit d'écriture, de suppression ou de modification des données stockées et transmises. L'intégrité s'applique aux données stockées et transmises.

C.5.4 Disponibilité

La disponibilité donne une assurance sur la réactivité et le temps de réponse d'un système pour transmettre une information d'une source à la bonne destination. Cela signifie aussi que le service réseau est disponible aux parties autorisées et assure bien son service d'acheminement en dépit des attaques en déni de service (DoS) pouvant affecter n'importe quelle couche du réseau.

C.5.5 Fraîcheur

Elle concerne la fraîcheur des données et la fraîcheur des clés. Puisque les réseaux de capteurs sont amenés à fournir des mesures qui n'ont de sens que sur une courte période de temps, nous devons assurer que chaque message est frais. La fraîcheur des données implique que les données sont récentes, et elle assure qu'aucun adversaire n'a rejoué les vieux messages.

C.5.6 Vie privée

Le droit à la vie privée est le droit des individus de contrôler ou d'agir sur des informations les concernant, qui peuvent être collectées et stockées, et sur les personnes par lesquelles et auxquelles ces informations peuvent être divulguées [83]. La "vie privée" telle que discutée dans le résumé en français fait référence au terme "privacy" anglais. Elle couvre les problèmes liés à la divulgation de l'identité des appareils à des tiers et à la traçabilité des appareils qu'un tiers pourrait vouloir suivre dans leurs déplacements. En effet, ces appareils sont porteurs d'informations confidentielles (relevés de mesures, géolocalisation...) pour une entreprise telle que TRAXENS. Ainsi, lorsqu'il est mentionné "la vie privée" des objets, c'est bien entendu un abus de langage signifiant l'intérêt porté aux objets dans la non divulgation d'informations qui pourrait nuire aux propriétaires de ces objets.

C.5.7 Non répudiation

C'est un mécanisme destiné à prévenir empêcher que la source ou la destination désavoue ses actions ou nie qu'un échange ait eu lieu.

C.6 Scénario

Cette thèse a été réalisée au sein de la société TRAXENS S.A.S., une société fondée par Michel Fallah en 2012 avec l'objectif de développer la logistique des services de systèmes d'information spécialisés dans le suivi et la surveillance des conteneurs intermodaux dans le monde entier [4]. Dans cette section, nous présentons une brève description des défis existants et de quelle façon TRAXENS les aborde.

On estime que près de 90% des marchandises du monde sont transportées par mer avec plus de 70% de marchandises expédiées par conteneurs, ce qui fait environ 33 millions de conteneurs en circulation. En 2020, environ 1 milliard de conteneurs seront négociés globalement [49].. Malgré sa croissance rapide et des profits élevés, la plupart des services de transport maritime sont insuffisants pour faire face au volume d'informations et de niveau de sophistication que la technologie permet. Certaines entités de la chaîne logistique dans l'industrie du transport maritime sont automatisées, mais il y a encore des lacunes comme le manque d'informations concernant la localisation et l'état des marchandises dans les conteneurs (par exemple la température, l'humidité, etc.) ou encore les cas de vols et d'accidents.

Le système TRAXENS est basé sur l'idée de " conteneurs intelligents ", qui se réfère aux conteneurs équipés de boites intelligentes, appelées Trax-Box. Le Trax-Box contient plusieurs capteurs intelligents, par exemple température, humidité, balise RFID (Radio Frequency Identification), capables de surveiller différents paramètres tels que la localisation, la sécurité et l'état des marchandises transportées. Les données recueillies sont périodiquement envoyées au serveur central, i.e., Trax-Hub, via un réseau maillé appelé Trax-Net. Le Trax-Box peut être sollicité par un appareil externe, baptisé M-Trax, qui doit avoir l'autorisation de la Trax-Hub avant de communiquer en toute sécurité avec le

Trax-Box. L'architecture du système TRAXENS est présentée à la figure C.1.

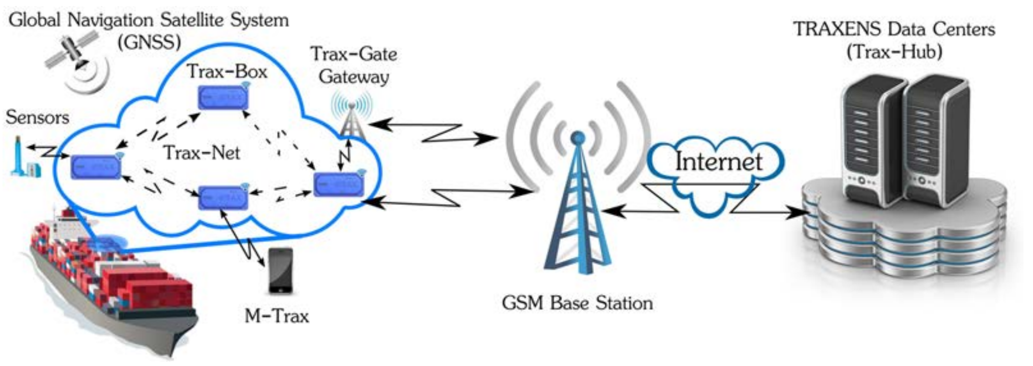

Global Navigation Satellite System (GNSS)
Trax-Box
Trax-Gate Gateway
TRAXENS Data Centers (Trax-Hub)
Sensors
Trax-Net
Internet
M-Trax
GSM Base Station

Figure C.1 - L'architecture du système de surveillance pour les conteneurs intermodaux proposée par TRAXENS

Pour améliorer la fiabilité et la qualité du service, les informations collectés et transmises entre les Trax-Box doivent être sécurisées et accessibles uniquement aux personnes ayant les droits d'accès. La sécurité doit être établie dans tous les points nécessaires.

Notez que cette thèse de doctorat se concentre uniquement sur la résolution des problèmes de sécurité et de vie privée dans la communication entre les objets de faibles ressources (dits "légers"), comme la communication entre Trax-Box et M-Trax ou Trax-Box et capteurs.

C.7 Contributions

Les contributions de cette thèse peuvent être classées en trois groupes comme indiqué dans les paragraphes ci-dessous.

C.7.1 Authentifier les objets connectés dans un cluster avec un terminal mobile en utilisant les protocoles Serverless

L'un des principaux objectifs de la thèse était de concevoir un protocole de sécurité pour les objets connectés légers contrôlées par un serveur d'autorisation centralisée. D'après le scénario TRAXENS présenté dans la section **??**, notre première contribution se concentre sur la communication sécurisée entre M-Trax et Trax-Box, cette communication étant autorisée par le Trax-Hub. M-Trax est un terminal mobile se connecte au Trax-Box pour accéder à des informations.

Dans le scénario, la communication entre le M-Trax et Trax-box peuvent avoir lieu dans des zones géographiques éloignées, où la connectivité au Trax-Hub n'est pas garantie. Par conséquent, nous considérons la connectivité intermittente comme l'un des facteurs majeurs pour proposer une solution qui est résiliente malgré l'intermittence de connectivité entre les objets connectés et le serveur central.

Dans le scénario, il est impératif de répondre aux propriétés de sécurité et de vie privée décrites dans la section **??**. En outre, notre protocole proposé tient compte de deux problèmes majeurs 1) la connectivité intermittente entre un serveur centralisé distant et des objets connectés à ressources contraintes et 2) la disponibilité asymétrique des ressources dans les objets communicants. Nous effectuons également une validation formelle en utilisant l'outil AVISPA. Notre contribution répond aux objectifs 2, 5 et 6 et l'article correspondant est publié dans:

- Collins MTITA, Maryline LAURENT, and Pascal DARAGON, *Serverless Lightweight Mutual Authentication Protocol for small mobile Computing Devices*, New Technologies, Mobility and Security (NTMS), 2015 7th International Conference on, pp.1-5, 27-29 July 2015.

C.7.1.1 Proposition d'authentification mutuelle en bref

Cette contribution propose un protocole serverless d'authentification mutuelle pour les objets connectés contraints en ressources. Notre proposition utilise le contexte pour limiter la validité des information d'authentification, ce qui est important pour améliorer la sécurité des systèmes pervasifs, et ce même après que les dispositifs respectifs soient physiquement compromis. Ainsi, notre solution proposée est un protocole d'authentification basée sur des jetons, semblable au protocole d'authentification Kerberos [65]. Notre solution proposée prend également en compte certaines des propriétés de vie privée et de sécurité importantes dans les protocoles sans serveur. De plus, notre protocole est symétrique et léger, car il est conçu en utilisant des primitives simples comme Message Authentication Code (HMAC), XOR ($\oplus$) et les opérations de comparaison.

Dans cette protocole, il y a trois acteurs, où deux des trois acteurs communiquent à la fois. Les acteurs de ce scénario sont M-Trax, Trax-Hub et Trax-Box. La communication se fait en deux phases - l'autorisation et l'authentification.

L'autorisation est la première phase, elle implique la communication entre le M-Trax et Trax-Hub. Au cours de cette phase, le terminal mobile (M-TRAX) demande une autorisation à partir du serveur central (Trax-Hub) pour communiquer avec les objets connectés (Trax-Boxes) dans les clusters. Cette communication a lieu lorsque le M-Trax peut se connecter au Trax-Hub. Pour demander une autorisation, M-Trax se connecte en toute sécurité à la Trax-Hub et envoie son identifiant ainsi que son location géographique. À son tour, le Trax-Hub donne accès M-Trax au cluster(s) de Trax-boxes dans son voisinage. Les informations d'authentification accordées à la M-Trax ont une période de validité, au-delà de laquelle le M-Trax doit se connecter à nouveau à Trax-Hub pour acquérir les autres informations d'authentification. M-Trax sauvegarde les informations d'identification et peut les utiliser dans la période de validité.

L'authentification est la deuxième phase du protocole. Pendant cette phase, M-Trax communique avec un cluster des objects connectés (Trax-Boxes). L'échange de données entre le terminal mobile et les appareils omniprésents est précédé d'une authentification mutuelle. L'authentification doit être effectuée à deux niveaux, un premier au niveau du cluster, puis un second au niveau de l'objet connecté. Au niveau du cluster, le M-Trax prouve qu'il a l'autorisation d'accéder au cluster et au niveau d'object connecté M-Trax prouve qu'on lui a accordé l'autorisation de communiquer avec le Trax-Box. Les Trax-Boxes vérifient la validité des paramètres avant d'accorder l'accès à la M-Trax.

A la fin de la phase d'authentification mutuelle, M-Trax et Trax-Box doivent générer une clé de session commune qui est utilisée pour échanger des données de façon sécurisée.

C.7.1.2 Avantages de notre protocole

Notre protocole présente les avantages suivants par rapport aux protocoles similaires existants.

- Notre protocole est léger. Le protocole utilise des primitives simples pour accomplir l'authentification mutuelle, tel que concaténation, XOR, HMAC.
- Le protocole consomme peu d'énergie car la session d'authentification mutuelle est achevée en trois messages.
- Le protocole impose une période de validité pour les paramètres d'authentifation. La notion de révocation est possible grâce à des informations de contexte. Ceci est possible même si l'objet connecté n'est pas équipé d'une horloge interne.
- Notre protocole utilise le contrôle d'accès pour chaque communication. Le contrôle d'accès est impératif pendant la communication entre les acteurs. Le contrôle d'accès permet à l'appareil d'accorder des droits d'accès, tel que de lire, écrire, ou modifier des information.

C.7.2 Les protocoles de recherche et d'authentification sécurisé pour des objets connectés contraints ressources.

Pour assurer une communication sécurisé des information sensibles, la collecte de données doit être effectuée en toute sécurité et avec précision. L'un des principaux défis qui doit être relevé dans le système TRAXENS est la communication sécurisée entre les objets connectés avec des capteurs intelligents. Ce scénario exige efficacité et passage à l'échelle.

D'après le scénario TRAXENS présenté dans la section **??**, notre deuxième contribution se concentre sur la communication sécurisée entre Trax-Box (ou M-Trax) et des capteurs (en particulier des capteurs RFID) avec l'autorisation du Trax-Hub. Trax-Boxes (ou M-Traxes) peuvent échanger des données d'information et d'accès à l'intérieur des capteurs avec l'autorisation préalable du Trax-Hub.

Nous constatons que la technologie RFID est le meilleur candidat pour répondre aux défis posés par ce scénario. Cela a conduit à la proposition de deux protocoles complémentaires utilisant la RFID comme une technologie de base. Le premier protocole effectue une authentification de masse entre un lecteur RFID et un groupe de balises RFID et l'autre protocole effectue une recherche sécurisée pour une balise cible parmi un groupe d'étiquettes RFID. Ces contributions comprennent deux validations formelles ; l'une fait appel à l'outil AVISPA et l'autre se fait en utilisant l'outil CryptoVerif. Cependant, nos propositions ne se limitent pas à la technologie RFID seule, mais peuvent être utilisés dans d'autres technologies ayant les mêmes exigences de ressources. Ces objectifs répondent aux objectifs 2, 3, 5 et 6 et l'article correspondant est publié dans:

- Collins MTITA, Maryline LAURENT, and Jacques DELORT, *Efficient Serverless RFID Mutual Authentication & Secure Tag Search Protocols with Untrusted Readers*, accepted for publication in the IET Information Security Journal, Special Issue on Lightweight and Energy-Efficient Security Solutions for Mobile Computing Devices, April 2016.

Ces deux contributions sont brièvement décrites ci-dessous.

C.7.2.1 RFID Serveless Mutual Authentication Protocol

Le protocole d'authentification mutuelle RFID est utilisé pour authentifier plusieurs étiquettes RFID dans son voisinage. Le protocole comporte trois acteurs, qui sont le lecteur RFID (similaire à M-Trax ou Trax-Box), le Serveur central (similaire à Trax-Hub) et des étiquettes (ou capteurs). La communication se fait en deux phases à savoir l'autorisation et d'authentification.

L'Autorisation
La phase d'autorisation implique la communication entre le lecteur RFID et la base de données centrale. Au cours de cette phase, le lecteur RFID (M-TRAX ou Trax-Box) demande une autorisation à partir du serveur centralisé (Trax-Hub) pour communiquer avec les balises RFID (capteurs). Cette communication a lieu lorsque le Trax-Box (ou M-Trax) peut se connecter au Trax-Hub. Pour demander une autorisation, Trax-Box (ou M-Trax) se connecte en toute sécurité à la Trax-Box et envoie son identifiant. À son tour, le serveur central donne accès aux lecteurs RFID en envoyant la liste de toutes les étiquettes RFID dont il autorise l'accès. Les autorisations accordées au lecteur RFID ont une durée de validité limitée, au-delà de laquelle le lecteur RFID doit se connecter au serveur central pour acquérir d'autres informations d'identification. Le lecteur RFID stocke les informations d'identification et peut les utiliser plusieurs fois avant l'expiration de la période de validité.

Authentication
La phase d'authentification implique la communication entre le lecteur RFID et les étiquettes. Le lecteur RFID diffuse une requête aux étiquettes de son voisinage et les étiquettes répondent au lecteur. Le lecteur doit vérifier et authentifier la réponse de chaque étiquette individuellement. Ensuite, le lecteur renvoie la réponse à chacune des étiquettes qui a répondu. Chaque étiquette s'authentifie auprès du lecteur. A la fin de la phase d'authentification mutuelle, l'étiquette et le lecteur doivent générer une clé de session commune qui peut être utilisée pour échanger des informations en toute sécurité.

C.7.2.2 Advantages of the Proposed RFID Authentication Protocol

Notre solution proposée présente les avantages suivants par rapport aux protocoles existants.

- Avec notre protocole, les secrets détenus par les étiquettes RFID ne sont jamais divulgués au lecteur lors de la phase d'authentification. Le lecteur ne reçoit que des secrets éphémères qui ont une période de validité. Cela préserve la sécurité des étiquettes, même dans le cas de la compromission du lecteur. L'adversaire ne peut pas utiliser les informations d'un lecteur pour générer des contrefaçons d'étiquettes et tromper les autres lecteurs légitimes.
- Les acteurs dans le protocole n'ont pas besoin de se connaître au préalable, ni de partager des informations d'authentification avant la phase d'authentification mutuelle. Au contraire, chaque partie doit établir des relations avec le serveur principal et compter sur lui pour vérifier les informations d'identification et d'autorisation utiles au montage des sessions.
- Notre protocole consomme peu d'énergie en réduisant le nombre de messages échangés pendant la phase d'authentification mutuelle. L'authentification mutuelle se fait par des échanges de trois messages de taille minimum. La faible fréquence des communication est adaptée aux objets connecté contraints en ressources [60, 69, 77].

C.7.2.3 Protocole de recherche RFID sécurisé et serverless

Le protocole de recherche sécurisé pour les étiquettes RFID permet un lecteur RFID de rechercher une étiquette cible parmi un groupe d'étiquettes dans son voisinage, d'authentifier l'étiquette et de lancer une session sécurisée pour l'échange de données. Ce protocole fournit un fonctionnalité indispensable pour améliorer l'efficacité de la recherche d'une étiquette parmi une grande nombre d'étiquettes [99] car il ne nécessite pas d'authentifier toutes les étiquettes du voisinage avant de trouver la bonne. Le protocole de recherche minimise le temps de recherche d'une étiquette connue dans un groupe d'étiquettes.

Les protocoles de recherche d'étiquettes RFID sont utiles dans plusieurs domaines comme l'inventaire ou les magasins où les vendeurs veulent rechercher des produits spécifiques parmi un grand nombre d'entre eux. Utiliser le protocole de recherche dans de telles situations permet une recherche efficace et rapide du produit ainsi que la connaissance de son emplacement approximatif.

Notre protocole doit répondre à cinq objectifs. 1) Il doit permettre une recherche efficace d'une étiquette connue. 2) Il faut protéger les éléments privés (privacy) des étiquettes recherchées, ainsi que des étiquettes voisines, au cours du processus de recherche. 3) Il doit sécuriser les informations échangées entre le lecteur et l'étiquette pendant le processus de recherche. 4) L'étiquette doit authentifier le lecteur avant de lui répondre. 5) L'étiquette ne doit répondre qu'aux requêtes légitimes.

De façon similaire au protocole d'authentification décrit dans la section **??**, ce protocole est réalisé en en deux phases. Tout d'abord, au cours de la phase d'autorisation, le lecteur communique avec le serveur central pour télécharger les informations d'identification pour les étiquettes et obtenir les droits d'accès. Ensuite, pendant la phase de recherche, le lecteur envoie une requête pour rechercher une étiquette particulière dans son voisinage.

C.7.2.4 Avantages de notre protocole de recherche

Notre solution proposée présente les avantages suivants par rapport aux protocoles existants.

- Le protocole de recherche est léger car il y a très peu de calculs pendant la phase de la recherche. Ceci est possible car pour chaque recherche, il y a seulement une réponse provenant de l'étiquette RFID ciblée, si elle est présente.
- Notre protocole de recherche n'est pas sensible aux attaques "Narrowing" du fait que les messages échangés sont sémantiquement indistinguables des messages des sessions précédentes. Cette forme d'attaque se produit lorsque l'adversaire interroge une étiquette avec un horodateur, puis un peu plus tard, tente d'identifier la même étiquette en l'interrogeant avec un timestamp légèrement au-dessus de la valeur précédente [102]. Par conséquent, un adversaire ne peut pas facilement associer un message à une étiquette.
- Contrairement aux protocoles existants dans l'état de l'art, notre protocole est résistant aux attaques de clonage et de compromission de lecteur RFID. Ceci est possible car le serveur central attribue des paramètres d'authentification avec une période de validité définie, au-delà de laquelle ils ne peuvent pas être utilisés pour effectuer une authentification entre le lecteur et les étiquettes. Par conséquent, lorsque l'adversaire compromet le lecteur, il ne reçoit que des valeurs d'authentification temporaires qui ne peuvent être valables dans la période de temps définie.

C.7.3 Un guide sur la conception des protocoles Serverless

Une recherche approfondie a été réalisée afin d'analyser les caractéristiques, les exigences et les performances des protocoles serverless existants relatifs aux besoins de l'ère omniprésente. La synthèse de cette enquête nous a conduits à proposer un guide sur la conception des protocoles serverless. Cette contribution répond aux objectifs 1 et 4 et correspond à l'article en cours de soumission :

Collins MTITA, Maryline LAURENT, and Jacques DELORT, *Designing Efficient & Secure Serverless Protocols*, under submission, 2016.

Dans ce guide, nous identifions six principes et six meilleures pratiques sur le développement des protocoles serverless sécurisés et efficaces qui répondent aux besoins des systèmes pervasifs. Avec l'aide du guide, il est possible de développer des protocoles efficaces et sécurisés pour des systèmes omniprésents. Les principes et les pratiques que nous décrivons dans ce guide soulignent la meilleure façon de concevoir des protocoles serverless. L'idée c'est d'améliorer l'efficacité du protocole conçu tout en évitant un nombre considérable de vulnérabilités classiques trouvées dans la plupart des protocoles existants.

Ce chapitre décrit des conditions importantes pour la conception de protocoles serverless. Ces conditions entraînent également les six principes suivants:

- **Principe 1**: *Le protocole sécurisé et efficace nécessite de définir clairement les besoins* Le principe 1 incarne l'idée que la conception du protocole commence par la phase de définition des besoins. Les besoins doivent être bien définis pour être clairement compris. Cela facilite les phases de recherche et d'analyse pour comprendre les menaces de sécurité pertinentes pour le scénario. La mauvaise identification des risques de sécurité et de vie privée pertinents à un scénario est une cause importante de vulnérabilités du protocole finale.

- **Principe 2**: *Les sessions éphémères doivent être authentifiées à l'aide de paramètres temporaires*
  Le principe 2 insiste sur la vie privée des parties communicantes. Le protocole ne doit pas divulguer de secrets appartenant aux parties communicantes au cours de la session d'authentification des communications éphémères. Au contraire, le protocole doit utiliser les paramètres temporaires, mais authentiques et vérifiables, pour achever la tâche. L'utilisation de paramètres temporaires évite un mauvais usage des paramètres secrets une fois que les objets en question ou leurs donnés sont compromis ou volés. Ce principe doit être respecté meme si les appareils sont déjà mutuellement authentifiés.
- **Principe 3**: *Les paramètres utilisés lors de l'authentification dans un protocole de serverless devraient avoir une durée de validité limitée.*
  Le principe 3 met l'accent sur l'intégration des informations de contexte dans les paramètres d'authentification accordés aux parties en communication afin de limiter leur durée de validité. Le problème de validité des paramètres d'authentification est récurrent dans les protocoles serverless existants. Pour assurer la sécurité, il est important que les paramètres d'authentification ne soient valides que pendant la session en cours.
- **Principe 4**: *Limiter la conception du protocole à ses éléments les plus importants.*
  Le principe 4 élabore que la conception du protocole doit accomplir ses besoins de base. Les besoins en ressources peuvent être réduits de différentes façons, par exemple, en stockant et en échangeant le minimum d'informations lors de l'authentification. Les informations minimales de base doivent être vérifiées comme provenant d'une source de confiance (ex : serveur d'autorisation centralisée).
- **Principe 5**: *Utiliser les fonctionnalités dans les objets connectés pour améliorer le protocole de sécurité*
  Le principe 5 suggère que, s'il y a des fonctionnalités dans les objets connectés ciblés qui peuvent être utilisés pour améliorer la conception du protocole en question, alors ils doivent être mis à profit. Par exemple, la plupart des systemes embarqués sont équipés d'accélérateurs matériels. Les accélérateurs matériels sont utiles pour améliorer l'efficacité. En plus, les accélérateurs matériels peuvent être utilisés pour économiser l'énergie [44].
- **Principe 6**: *Une conception de protocole sécurisé devrait suivre aux pratiques prudentes pour concevoir les protocoles cryptographiques.*
  Le principe 6 souligne l'importance de suivre les pratiques de base sur la conception de protocoles cryptographiques [5, 98]. Il existe plusieurs principes pour concevoir des protocoles cryptographiques tels que ceux mis en avant par Abadi et al. [5] et Anderson et al. [13]. Bien que leurs principes ne sont pas censés être des règles de conception, ils sont utiles pour éviter les erreurs qui sont généralement faites par de nombreux concepteurs de protocoles. Nous recommandons de suivre ces principes de conception de protocoles cryptographiques avant et pendant la phase de conception du protocole serverless.

Ce chapitre donne également les meilleures pratiques de choix des paramètres à base des besoins spécifiques de la solution de sécurité et les ressources disponibles dans les objets connectés. Ces pratiques sont résumées comme suit:

- **Communication vs. Calcul : Optimisation de l'énergie**
  Communication et calcul sont deux activités consommatrices d'énergie dans les objets connectés. De plus, la communication consomme plus d'énergie que le plus lourd des calculs dans l'objet [11, 76, 101]. Ainsi, du point de vue énergétique, il est plus économique d'effectuer le calcul que de faire la communication fréquente.
  Le meilleur choix est d'optimiser l'énergie en faisant le calcul et éviter de fréquentes communications [76].
- **Sécurité de bout-en-bout vs noeud-à-noeud : Optimisation de la sécurité**
  La sécurité dans les écosystèmes des objets connectés peut être soit *de bout-en-bout* ou *de noeud-à-noeud* en fonction des besoins. Le chiffrement de noeud à noeud est idéal dans une configuration où la sécurité est limitée aux seuls noeuds voisins. Par exemple, dans les réseaux de capteurs, les nœuds agrègent et comparer les informations avant de transmettre au serveur centralisé [12, 94]. Dans ce cas, chaque noeud participant au transfert d'informations doit recevoir, stocker, déchiffrer et chiffrer les informations avant de transmettre au noeud suivant. Le chiffrement de noeud-à-noeud a des inconvénients importants tels que les exigences de latence et de ressources élevées dues à des opérations régulières de stockage, de déchiffrement, de traitement et de chiffrement pour chaque message reçu [12]. Par ailleurs un chiffrement de noeud à noeud exige de la confiance entre l'expéditeur et tous les noeuds intermédiaires. Dans les réseaux de capteurs, il est obligatoire pour chaque message de voir sa source authentifiée avant d'être traité ou transmis afin de contrecarrer des attaques telles que les attaques par injection de données dans le réseau [108].
  D'autre part, le chiffrement de bout en bout permet au noeud de source de chiffrer les informations que seul le destinataire peut déchiffrer et comprendre [97]. Les noeuds intermédiaires transmettent uniquement les informations et ne prennent pas le rôle actif dans la sécurité de l'information transférée. Ce mode de chiffrement est idéal pour les systèmes désireux d'économiser des ressources, éviter les temps d'attente et d'offrir des niveaux plus élevés de vie privée, en particulier dans des environnements où des noeuds voisins ne peuvent pas faire confiance [12, 97].
- **Bande-passante vs espace de stockage : optimisation des coûts**
  La bande passante est précieuse et souvent une ressource rare dans les réseaux sans fil [41]. La plupart des appareils omniprésents n'ont pas accès à des débits élevés, ce qui limite la capacité totale de l'information échangée à un moment donné, soit entre pairs locaux, soit avec les parties distantes. Selon le scénario, l'utilisation excessive de la bande passante peut avoir des effets négatifs sur la qualité du service pour les informations échangées en raison de la congestion du réseau à savoir plusieurs périphériques partagent la bande passante limitée, ce qui augmente aussi le taux de perte de données. La meilleure solution consiste à réduire la fréquence de la communication en stockant des informations importantes au niveau local, ce qui peut aider à calculer les paramètres nécessaires une fois nécessaire [62].
- **Energie vs Niveau de sécurité**
  Les opérations de sécurité ont des demandes très variées en termes de ressources. Des

objets connectés ont aussi des capacités variées. Si les objets connectés ont des limites d'énergie strictes, l'utilisation de primitives cryptographiques simples et légères est idéale. Ces primitives exigent peu de ressources, ce qui se traduit par une faible consommation d'énergie. Pour les systèmes généralisés sans contraintes énergétiques strictes, une plus grande sécurité peut être assurée à l'aide des algorithmes de sécurité encore robustes [105].

- **Stockage vs Calcul : optimisation de la mémoire**
  Pour une efficacité maximale, le protocole doit trouver un équilibre entre le stockage et le calcul en tenant compte de trois facteurs. Tout d'abord, il faut savoir explicitement quels paramètres peuvent être stockés et ceux qui peuvent être calculés sur demande. Ensuite, déterminer s'il est plus efficace de recalculer un paramètre, plutôt que de le stocker ; en tenant compte des pénalités potentielles de ressources si les données doivent être réutilisés, mais ne sont pas disponibles en cas de besoin [7]. Enfin, seuls quelques paramètres, avec des tailles minimales et essentielles, nécessaires pour rendre l'appareil plus autonome, doivent être stockés localement [22].
- **Résistance à la compromission physique vs coût**
  Selon le scénario du système et les données collectées et stockées, il est idéal de mesurer le coût d'implémentation des mécanismes de résistance à la compromission physique. Si les données stockées sont sensibles, il est impératif de les protéger par un moyen résistant à la compromission physique.

C.8 Conclusion

Dans cette thèse, nous abordons les défis de sécurité et de vie privée pertinents aux systèmes pervasifs avec des contraintes de ressources strictes. Nous regardons les protocoles d'authentification serverless, qui sont des mécanismes d'authentification qui ne nécessitent pas la présence du serveur central au cours de la phase d'authentification entre deux objets connectés. Les protocoles serverless sont des mécanismes idéaux pour l'authentification à l'ère de l'informatique omniprésente.

Tout d'abord, nous fournissons les caractéristiques et les besoins pour les protocoles serverless. Grâce à ces besoins et caractéristiques, nous avons fait des recherche, des analyses complètes et des comparaisons des protocoles serverless existants en termes de sécurité, de vie privée et de performances.

Trois nouvelles contributions sont proposées dans cette thèse. Notre première contribution est un protocole léger serverless d'authentification mutuelle pour les objets connectés hétérogènes. La première contribution fournit trois avantages par rapport aux protocoles existants. Cette contribution répond aux exigences des systèmes pervasifs. La validation de notre proposition a éte fait en utilisant l'outil AVISPA et la validation informelle en utilisant sécurité et de vie privée des jeux.

Notre deuxième contribution comprend deux protocoles complémentaires dans le domaine des technologies RFID. Le premier protocole vise à l'authentification de masse entre un lecteur RFID et un groupe d'étiquettes tandis que le deuxième protocole effectue une recherche sécurisée pour une étiquette cible parmi un groupe d'étiquettes dans le voisinage du lecteur. Les deux protocoles proposés tiennent compte des contraintes de ressources des étiquettes RFID.

Après une étude approfondie des protocoles serverless, nous avons proposé une troisième contribution, un guide pour la conception des protocoles serverless sécurisé et efficaces pour les systèmes pervasifs. Le guide contient six principes et six meilleures pratiques en vue d'élaborer des protocoles serverless. Le guide est destiné à aider à la conception de protocoles serverless efficaces, sécurisés et simples en évitant des erreurs couramment faites dans les protocoles existants.