Protocoles de sécurité efficaces pour les réseaux de capteurs IP sans-fil et l'Internet des Objets

Kim Thuat Nguyen

These de Doctorat

Spécialité: Télécommunications

École doctorale: Informatique, Télécommunications et Électronique de Paris

Présenté par

Kim Thuat NGUYEN

Pour obtenir le grade de
DOCTEUR DE TÉLÉCOM SUDPARIS

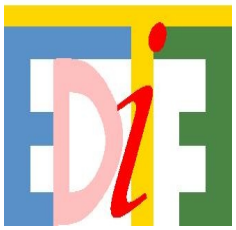
Protocoles de sécurité efficaces pour les réseaux de capteurs IP sans-fil et l'Internet des Objets

---

Soutenue le 08/12/2016 devant le jury composé de:

Isabelle CHRISTMENT	Professeur à Télécom Nancy	Rapporteur
Yves ROUDIER	Professeur à l'Université de Nice Sophia-Antipolis	Rapporteur
Sébastien TIXEUIL	Professeur à l'Université Pierre et Marie Curie	Examinateur
Saïd GHAROUT	Ingénieur de recherche à Orange lab	Examinateur
Nouha OUALHA	Ingénieur de recherche au CEA LIST	Encadrant
Maryline LAURENT	Professeur à Télécom SudParis	Directeur de thèse

Thèse No: 2016TELE0025



Thesis Dissertation

Speciality: Telecommunications

Doctoral School: Information, Telecommunications and Electronics of Paris

Presented by

Kim Thuat NGUYEN

For the degree of
DOCTOR OF TELECOM SUDPARIS

Lightweight Security Protocols for IP-based Wireless Sensor Networks and the Internet of Things

---

Defended on 08/12/2016 in front of a jury composed of:

Isabelle CHRISTMENT	Professor at Telecom Nancy	Reviewer
Yves ROUDIER	Professor at Université de Nice Sophia-Antipolis	Reviewer
Sébastien TIXEUIL	Professor at Université Pierre et Marie Curie	Examiner
Saïd GHAROUT	Research Engineer, Orange Labs	Examiner
Nouha OUALHA	Research Engineer at CEA LIST	Advisor
Maryline LAURENT	Professor at Telecom SudParis	Thesis director

Thesis No: 2016TELE0025

Contents

Abstract

The Internet of Things (IoT) enables billions of embedded computing devices to connect to each other. The smart things cover our everyday friendly devices, such as, thermostats, fridges, ovens, washing machines, and TV sets. It is easy to imagine how bad it would be, if these devices were spying on us and revealing our personal information. It would be even worse if critical IoT applications, for instance, the control system in nuclear reactors, the vehicle safety system or the connected medical devices in health-care, were compromised. To counteract these security threats in the IoT, robust security solutions must be considered. However, IoT devices are limited in terms of memory, computation and energy capacities, in addition to the lack of communication reliability. All these inconvenients make them vulnerable to various attacks, as they become the weakest links of our information system.

In this context, we seek for effective security mechanisms in order to establish secure communications between unknown IoT devices, while taking into account the security requirements and the resource constraints of these devices. To do so, we focus on two major challenges, namely, lightweight security protocols in terms of processing and infrastructure and lightweight key establishment mechanisms, as existing solutions are too much resource consuming.

First of all, traditional secure encryption methods (e.g. RSA) usually require operations that are consuming too much for the computing capacity of IoT devices. In addition, in conventional security solutions, public key cryptosystems usually rely on a Public Key Infrastructure (PKI) to provide identity management and authentication, but PKIs are not suitable for low-bandwidth and resource-constrained environments.

To address this first challenge, we first propose ECKSS - a new lightweight signcryption scheme which does not rely on a PKI. This proposal enables to encrypt and sign messages simultaneously while ensuring the confidentiality and unforgeability of the communication channels. In addition, the message exchanges are authenticated without relying on certificates. Moreover, we also propose OEABE which is a delegation-based mechanism for the encryption of the Ciphertext-Policy Attribute-Based Encryption (CP-ABE). CP-ABE is an attribute-based public key encryption scheme that gives users the flexibility to determine who can decrypt their data at runtime. Our solution enables a resource-constrained device to generate rapidly a CP-ABE ciphertext with authorization access rights to its data. This solution is particularly useful as the volume of data issued from IoT devices grows exponentially every year.

Second, the existing key establishment mechanisms do not take into account specific constraints of heterogeneous environments, e.g. IoT. Indeed, in order to establish a common key between two unknown entities, one usually employs a key transport scheme based on a public key encryption algorithm, which is generally expensive for limited-resource devices. The second approach consists of using a key agreement scheme, in which the two communicating parties are involved in the negotiation of the final key. We can employ, in this case, a Diffie-Hellman key exchange, or a server-assisted key agreement. However, these methods have several drawbacks

such as poor performances on limited-resource devices, or vulnerability to key-escrow attacks.

To solve this second challenge, we first propose two new key distribution modes for the standard key management protocol MIKEY, based on our signcryption scheme ECKSS. These modes inherit the lightness of ECKSS and avoid the use of a PKI. The experimental results, conducted in the Openmote sensor platform, have proven the efficiency of our solutions compared with other existing methods of MIKEY. Then, we propose a new key agreement scheme, named AKAPR. In case the two communicating parties are involved in the key negotiation procedure, AKAPR is very suitable in the context of IoT. As such, it can operate even if the two communicating parties are highly resource-constrained.

All our proposals were informally validated (for OEABE) or formally validated either through a sequence of games or the automatic cryptographic verifier ProVerif.

Résumé

L'Internet des Objets (IdO) permet à des milliards de dispositifs informatiques embarqués de se connecter les uns aux autres. Les objets concernés couvrent la plupart de nos appareils de la vie quotidienne, tels que les thermostats, les réfrigérateurs, les fours, les machines à laver et les téléviseurs. Il est facile d'imaginer l'ampleur du danger, si ces dispositifs venaient à nous espionner et révélaient nos données personnelles. La situation serait encore pire si les applications critiques IdO, par exemple, le système de contrôle des réacteurs nucléaires, le système de sécurité du véhicule ou les dispositifs médicaux, étaient compromis. Afin de garantir la sécurité et lutter contre des menaces de sécurité dans l'IdO, des solutions de sécurité robustes doivent être considérées. Cependant, les appareils pour l'IdO sont limités en mémoire, capacités de calcul et énergie, et disposent de moyens de communication peu fiables, ce qui les rend vulnérables à des attaques variées.

Dans ce contexte, nous recherchons des mécanismes de sécurité efficaces pour établir des communications sécurisées entre des entités IdO qui ne partagent pas préalablement de clés cryptographiques, tout en tenant en compte des exigences de sécurité et des contraintes en ressources de ces dispositifs. Pour ce faire, nous nous concentrons sur deux défis majeurs, à savoir des protocoles de sécurité légers en termes de calculs et d'infrastructure, et des mécanismes d'établissement de clés légers, les solutions existantes actuellement étant beaucoup trop coûteuses pour les dispositifs IdO.

Tout d'abord, les méthodes de chiffrement asymétriques traditionnelles (par exemple RSA) exigent généralement des opérations beaucoup trop coûteuses pour des dispositifs IdO. De plus, dans les solutions de sécurité conventionnelles, les primitives de chiffrement à clé publique reposent généralement sur une infrastructure à clé publique (PKI) pour gérer l'identité et l'authentification, mais de telles infrastructures sont beaucoup trop consommatrices de ressources.

En réponse à cette première problématique, nous avons, d'une part, proposé ECKSS - un nouveau schéma de signcryption léger qui évite l'utilisation de PKI. Cette proposition permet de chiffrer et signer simultanément des messages en garantissant la confidentialité et la non-falsification du canal de communication. De plus, les échanges de message sont authentifiés sans recourir à des certificats. Par ailleurs, nous avons aussi proposé OEABE qui est un mécanisme de délégation pour le chiffrement à base d'attributs CP-ABE (Ciphertext-Policy Attribute-Based Encryption). CP-ABE est un schéma de chiffrement par attributs qui permet aux utilisateurs de préciser au moment du chiffrement qui pourra déchiffrer leurs données. Notre solution, OEABE, permet à un dispositif contraint en ressources de générer rapidement un chiffré CP-ABE tout en précisant les droits d'accès à ses données. Cette solution est d'autant plus utile que le volume de données générées par les dispositifs IdO est en augmentation exponentielle chaque année.

Ensuite, les mécanismes d'établissement de clés existants ne prennent pas en compte de contraintesspécifiques pour des environnements hétérogènes comme l'IdO. En effet, pour convenir d'une clé commune entre deux entités inconnues, il est souvent fait appel à un schéma de transfert

de clés basé sur un algorithme de chiffrement à clé publique, ce qui est généralement trop coûteux pour les dispositifs limités en ressources. La deuxième approche consiste à utiliser un schéma d'échange de clés où les deux parties communicantes participent à la négociation de la clé finale. Dans ce cas là, se trouvent utiliser soit un échange de clés Diffie-Hellman, soit un échange de clés assisté par un serveur. Ces deux méthodes ont l'inconvénient d'être inadaptées à l'IdO avec des performances trop médiocres ou bien vulnérables aux attaques de tiers de séquestre (key-escrow).

En réponse à cette deuxième problématique, nous avons proposé tout d'abord deux modes de distribution de clés pour le protocole standard de gestion de clés MIKEY. Ils s'appuient sur notre schéma de signcryption ECKSS et héritent ainsi de la légèreté d'ECKSS à la fois en termes de calculs et de dispensent d'utilisation de PKI. Les résultats expérimentaux, obtenus à partir d'une plateforme de capteurs Openmote, ont prouvé l'efficacité de nos solutions comparativement aux autres méthodes de MIKEY. Nous avons aussi proposé un schéma d'échange de clés, appelé AKAPR qui est très adapté dans le cas où les deux parties qui participent à la négociation de clés sont très contraintes en ressources.

Nos propositions on été validées informellement (pour OEABE) ou formellement, soit à l'aide de séquences de jeux, soit avec le vérificateur automatique ProVerif.

Acknowledgment

This thesis becomes reality with the kind support and help of many individuals. I would like to express my sincerest gratitude to all of them.

I would like to express my special gratitude and thanks to my advisor **Dr. Nouha OUALHA**, for her keen interest on every stage of my research. Her prompt inspirations, timely supports with kindness, enthusiasm and dedication, have enabled me to complete this thesis.

I owe a deep sense of gratitude to my thesis director **Prof. Maryline LAURENT** for her scientific approach, meticulous scrutiny, vast knowledge and insightful suggestions during all phases of my thesis. It is a great honor for me to be her PhD student.

My sincere thanks also goes to **Prof. Isabelle CHRISTMENT** and **Prof. Yves ROUDIER**, for being reviewers and members of the jury. Their insightful comments and encouragement, but also their hard questions inspired me to widen my research from various perspectives. I would like to thank the rest of my thesis committee: **Prof. Sébastien TIXEUIL** and **Dr. Saïd GHAROUT**, for their interest, involvement and time.

I would like also to say thank you to all my colleagues at LSC laboratory for the stimulating discussions, and for all the fun we have had in the last three years.

Last but not the least, it is my privilege to thank my parents and my wife **Linh**, for their constant encouragement throughout my thesis period.

Thank you very much, everyone!

List of Tables

List of Figures

Chapter 1

Introduction

The Internet of Things (IoT) is designed as a network of highly connected devices (things). In today's perspective, the IoT includes various kinds of devices, e.g., sensors, actuators, RFID tags, or smartphones, which are very different in terms of size, weight, functionality and capabilities. IoT devices are increasingly deployed. Indeed, according to Gartner's forecast [113], the IoT will grow to more than 26 billion deployed units by 2020. The main challenge is how to make such devices operate in the conventional Internet. Inspired by this motivation, recent research efforts have focused on the design of new protocols and the adaptation and application of standard Internet protocols in the IoT. For instance, the initiative of the 6LoWPAN [5] working group allowed the smallest devices with limited processing capabilities to become part of the Internet by enabling the use of IP over these devices; thus, enabling the connection of literally billions of devices to the Internet, in which very different things such as temperature and humidity sensors can directly communicate with each other, with a human carrying a smartphone, with a heating and cooling system, or with a remote backend server.

While the concept of IoT is easy to grasp, major research efforts still need to be made, in particular, in resolving challenges associated with security, privacy, and trust. Allowing each single physical object to connect to the Internet and to share information may create more threats than ever for our personal data and business secret information. As depicted in Figure 1.1, concerned objects cover our everyday friendly devices, such as, thermostats, fridges, ovens, washing machines, and TV sets. It is easy to imagine how bad it would be, if these devices were spying on us and revealing our personal information. It would be even worse if critical IoT applications, for instance, the control system in nuclear reactors, the vehicle safety system or the connected medical devices in health-care, were compromised. In order to guarantee the security and privacy threats in the IoT, robust security solutions need to be considered. However, existing standard protocols for security protections can not be directly applied in the IoT due to IoT device's resource limitations e.g. in terms of energy, processing power and memory, in addition to the lack of communication reliability.

1.1 IoT security challenges

Being a very common model for monitoring physical data and environmental conditions, the Wireless Sensor Network (WSN) was initially designed as a closed network where all environmental data are collected by sensor nodes and then transferred to a remote location through a gateway. The direct connections between the end-users and the sensor nodes are not privileged in such model. All communications are forwarded between nodes and passed to the outside world by the gateway. The WSN was motivated by military applications for example, the use of sensors for detecting enemy intrusion on the field. As an extension of WSN, the IoT is an

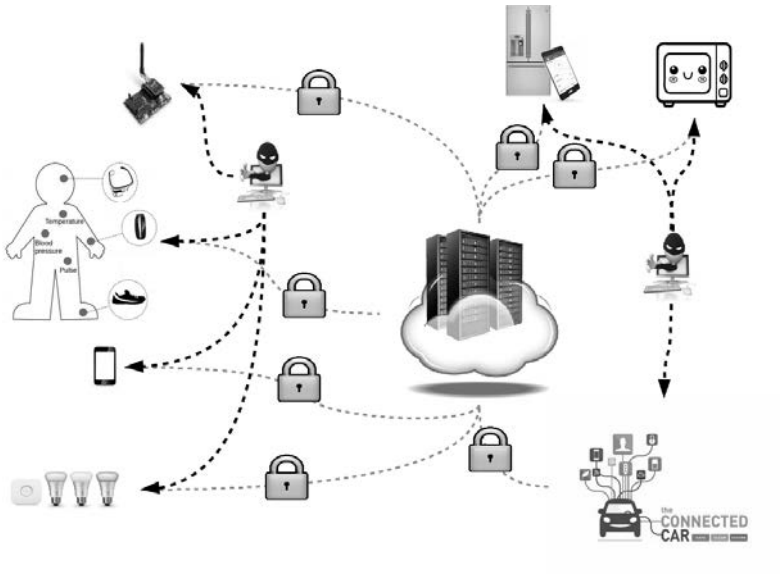


Figure 1.1: Secure communication is essential in the Internet of Things

evolution of traditional internet model where the digital world meets the physical world. The explosion of the number of connected devices obligates the extension of IPv4 to IPv6. Considered as the foundation of IoT, IPv6 is subject to the same attack threat as IPv4, such as, spoofing, fragmentation attacks, sniffing, neighbor discovery attacks, rogue devices, man-in-the-middle attacks, and others. However, IoT opens a completely new dimension of security threats. Indeed, the IoT offers connectivity for both types: human-to-machine and machine-to-machine communications. In the near future, everything is likely to be equipped with small embedded devices which are able to connect to the Internet. Such ability is useful for various domains in our daily life: i.e. from building automation, smart city, and surveillance system to all wearable smart devices. However, the more the IoT devices are deployed, the greater our information system is at risk. Indeed, a non-negligible number of devices in IoT are vulnerable to security attacks, for example, denial of service and replay attacks, due to their constrained resources and the lack of protection methods.

In order to obtain the needed security, IoT systems have to deal with multiple security challenges including the following:

- *Resource constraints* – the resource-constrained nature in memory, computation capacities and energy of IoT devices, may not support the expensive operations required in complex and evolving security algorithms. IoT devices often operate on lossy and low-bandwidth communication channels. It seems to be impossible to apply directly standard conventional security protocols of the Internet in the context of IoT. As an example, the use of small packets (i.e. IEEE 802.15.4 supports only 127-bytes packets [161]) may result in fragmentation of larger packets when using the standard protocols. This will exhaust the life time of sensor nodes and open new possibility of DoS attacks. Hence, the standard security protocols must be redesigned to adapt such difficult scenario, in order to offer equivalent security levels with more efficient performance for the IoT.

- *Resilience to attacks* – IoT devices are typically small, inexpensive with little or no physical protection. For example, a mobile/sensor device can be stolen or fixed devices can be moved. Such attacks may modify the data read by users without any one noticing. As a

result, the system has to avoid single points of failure so a compromised node will not affect the whole system. Besides, the secured network must also avoid the resource-depletion attacks launched against resource-constrained devices.

- *Privacy protection* – The popularity of RFID tags has raised privacy concerns because anyone can track tags and find the identity of the objects carrying them. In addition, as wearable technology increases its pace, we will be soon able to connect our bodies to the Internet by "putting on" tiny hardware devices (e.g. Implant chips inside our bodies). Consequently, our personal information (i.e. healthcare records) must remain secured and should not be traceable, linkable and identifiable.
- *Interoperability* – Deploying security solutions in the IoT should not hinder the functional operation of interconnected heterogeneous devices.
- *Availability* – The sensor nodes must be available when needed. High availability network of things should remain functional, especially against denial-of-service attacks, such as flooding of incoming messages to targeted nodes forcing them to shut down
- *Scalability* – The IoT network, for instance WSNs, is generally composed of a large number of devices. The proposed security protocol should be able to scale. This property is tightly related to the amount of information that each device has to keep in memory for a secure channel to be negotiated with as many entities as possible (other sensor nodes or Internet entities).

1.2 Problem Statement and Objectives

In the context of IoT, two devices in a large network composed of multiple devices are not necessarily known to each other. As a result, we cannot consider applying directly symmetric security solutions to create secure communications between them. In such case, the security link between these devices can be created by using an asymmetric security protocol or by relying on a trusted third party. In this thesis, we are interested in such procedure of establishing secure associations and exchanges between unknown entities in heterogeneous environments. Below, we clarify several existing problems that need to be addressed.

First, secure communications between unknown entities are generally provided via secure key establishment mechanisms, including key agreement and key distribution solutions [141]. Their objective is for both parties to possess a common secret key for initiating secure communications. On one hand, regarding key agreement methods, it is natural to use directly asymmetric techniques to generate a common secret key between two unknown entities [163]. However, this method requires generally expensive operations in both communicating parties. As an alternative, the server-based key agreement protocols are much more energy-saving than the asymmetric approaches as they are based on the symmetric techniques. However, this solution is vulnerable to the key escrow attacks in case the assisted server is compromised and discloses the negotiated session keys. On the other hand, the key distribution methods refer to the public key encryption mechanisms to encrypt the secret key, as the communicating parties do not share any credentials *a priori*. However, most of the public key encryption algorithms require intensive calculations (e.g. RSA) and a complicated identity management systems for the devices to authenticate each other. As a result, there is a continuing need for authenticated lightweight key establishment methods in IoT. The challenge is to limit not only the number of expensive operations (e.g. exponentiation) to be executed in both parties but also the communication overhead.

Second, there is a strong need for designing new lightweight encryption solutions adapted to the IoT constrained environments. Traditional secure encryption methods (e.g. RSA [165]) are indeed usually calculation-intensive with large key sizes which undermine the computation capacity of IoT devices. On the other hand, lightweight authenticated encryption mechanisms that are able to provide secure and fast data transmissions, are very useful in heterogeneous environments. There exist actually many public key cryptosystems in the literature [121]. The difficult task is how to propose a more lightweight public key cryptosystem for IoT devices based on existing techniques in an optimized manner.

Finally, in conventional security solutions, public key cryptosystems usually rely on a Public Key Infrastructure (PKI) to provide identity management and authentication. In such a model, each party is authenticated through X.509 certificates. However, the verification and management of certificates require important computation operations and bandwidth for communicating with remote entities and also sophisticated revocation mechanisms. The aforementioned requirements are not suitable for low-bandwidth and resource-constrained environments. The challenge is to propose solutions that avoid the burden of PKI-based solutions while satisfying the security requirements.

In order to cope with the aforementioned problems and challenges, we set the following objectives:

- **Objective A**: propose a lightweight public key encryption mechanism without PKI dependence and apply, if possible, the proposed solutions into a standard key management protocol.
- **Objective B**: propose an efficient key agreement protocol that is suitable for highly constrained devices.
- **Objective C**: provide a mathematical or formal security proof for the proposed schemes.
- **Objective D**: conduct an experimental performance assessment of the proposed solutions in emulated or real sensor platforms.

1.3 Contributions

The contributions of this thesis are summarized in the following:

- **Contribution 1** – A new lightweight and provably secure encryption scheme is proposed in the context of IoT and then applied to the standard key distribution protocol MIKEY. Our proposal provides a new lightweight key distribution method for MIKEY that can be used in resource-constrained devices (**Objective A**, **Objective C**, **Objective D**).
- **Contribution 2** – OEABE is an outsourcing mechanism for the Encryption of Ciphertext-Policy Attribute-based Encryption (CP-ABE). In other words, OEABE is a delegation-based variant of CP-ABE where even constrained IoT devices can generate a CP-ABE ciphertext. IoT devices generate an extremely high volume of data which cannot be all stored in user devices, but should be instead kept encrypted in powerful but not necessarily trusted servers (e.g. public cloud services). CP-ABE is one mechanism of interest but it requires significant computation costs which increase with the complexity of input access policies. Our OEABE proposal enables resource-constrained devices to compute a CP-ABE ciphertext with merely one exponentiation. This advantage can be extremely beneficial in many scenarios, for example eHealth applications (**Objective A**, **Objective D**).

- **Contribution 3** – AKAPR is a new lightweight key agreement protocol dedicated to resource-limited devices. AKAPR is composed of only four message exchanges. Thanks to the delegation-based approach, the constrained devices are free from any expensive cryptographic operation (e.g. exponentiation or pairing). Moreover, in our proposal, the assisted-server is unable to learn any agreed secret keys between the two communicating parties, thus mitigating the key-escrow attacks (**Objective B**, **Objective C**).

1.4 Thesis Outline

The remainder of this dissertation is organized as follows.

Chapter 2 - Preliminaries and Scientific Background this chapter presents the basic background on several cryptographic primitives. We give fundamental notions on symmetric/asymmetric cryptography and three asymmetric encryption algorithms: signcryption, attribute-based encryption and proxy re-encryption. Then, we introduce the Elliptic Curve Cryptography (ECC) since its use has been recommended in embedded devices [134]. Several cryptographic hard problems in the ECC setting are also presented as well. Finally, we survey the use of implicit certificates in recent proposed solutions for IoT environments.

Chapter 3 - Lightweight Cryptographic Primitives and Secure Communication Protocols for IoT In this chapter, we analyze existing lightweight cryptographic primitives and secure communication protocols for the Internet of Things. We focus mainly on lightweight cryptographic schemes and lightweight key establishment protocols and methods. Then, we identify promising directions that can be used to propose lightweight secure communication mechanisms for IoT.

Chapter 4 - ECKSS: Elliptic Curve Korean Signature-based Signcryption for IoT this chapter describe our first contribution. We introduce a novel lightweight asymmetric encryption scheme based on signcryption. Our scheme inherits the lightness of the signcryption. It is also provably secure in the random oracle and certificate-free. Then, we apply the proposed certificateless signcryption scheme to the standard key distribution MIKEY. We then asset the suitability of the new proposed MIKEY mode in real sensor platform.

Chapter 5 - OEABE: Outsourcing the Encryption of Ciphertext-Policy Attribute-Based Encryption this chapter describes our second contribution. In practice, an attribute-based encryption mechanism still requires important computation overhead for resource-constrained devices because of the exponentiation operations to be executed in the encryption and decryption phases. In this chapter, we present a novel mechanism for outsourcing the encryption of Ciphertext-Policy Attribute-based Encryption (CP-ABE). The proposed solution accelerates significantly the encryption phase of CP-ABE and hence suitable for constrained devices.

Chapter 6 - AKAPR: Authenticated Key Agreement Mediated by a Proxy Re-Encryption for IoT in this chapter, we specify our third contribution. Our objective is to facilitate the secure communication between any two entities in IoT environment, even if both entities are highly resource-constrained and unknown to each other. That is, we first focus on the idea behind the proxy re-encryption where there exists a proxy in the middle that can translate a ciphertext dedicated for one party into a ciphertext dedicated for another party. Indeed, we propose a lightweight proxy re-encryption mechanism that does not require any heavyweight cryptographic operation, such as modular exponentiation. Then, we employ the proposed mech-

anism to build a key agreement protocol devoted for use in highly constrained environments, such as IoT. The security of the proposal has been validated by the automatic cryptographic verifier ProVerif.

Chapter 7 - Conclusion and Perspectives this chapter concludes the dissertation by giving a synthesis of our contributions and several perspectives for future work.

Chapter 2

Preliminaries and Scientific Background

In this dissertation, we give a specific state of the art for each contribution. Hence, this chapter provides only scientific and technical backgrounds that are useful for the lecture of this document. Concretely, we first present the background on different cryptographic primitives. We concentrate mostly on the primitives that are related to our work. Then, we present the elliptic curve cryptography, in which we provide basic notions of ECC operations and several problems in ECC which are assumed to be computationally infeasible in cryptography. Finally, we discuss the notions of implicit certificate which is a good alternative method to avoid the burden of public key infrastructure (PKI) in the context of the Internet of Things.

2.1 Cryptographic Primitives

In this section, we present basic concepts and notions of different fundamental cryptographic primitives.

2.1.1 Symmetric Key Cryptography

This subsection provides an introduction to symmetric key cryptography. The latter can be mentioned as private key encryption. In fact, a secret key is used to protect the communication between any two parties, say Alice and Bob. They employ this key in both the encryption and the decryption processes. We define a symmetric key encryption scheme in the following Definition 1.

Definition 1 (**Symmetric key encryption scheme**).
A symmetric key encryption scheme with the input security parameter k, is defined by a pair of two deterministic algorithms $(\mathsf{Enc}, \mathsf{Dec})$ as follows:

- $\mathsf{Enc}(K, M) \rightarrow C$. This is the encryption algorithm that takes as input an element K of the set of keys $\mathcal{K}$, a message M from the set of plaintexts (or messages) $\mathcal{M}$ and outputs an encrypted message C from the set of ciphertexts $\mathcal{C}$. The set $\mathcal{K}$ is defined by the parameter k, i.e. $\mathcal{K} = \{0,1\}^l$, where l is a positive integer generated from k.
- $\mathsf{Dec}(K, C) \rightarrow M$. This is the decryption algorithm that takes as input the same secret key K, the ciphertext C and outputs the message M.

It is required that the equation $\mathsf{Dec}(K, \mathsf{Enc}(K, M)) = M$ holds for every $K \in \mathcal{K}$ and $M \in \mathcal{M}$.

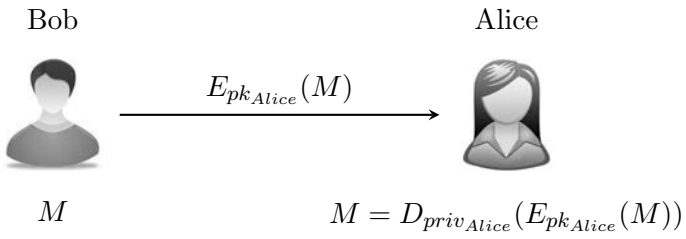


Figure 2.1: An example of public key encryption

In addition, we define the one-time indistinguishability (OT-IND) property of the symmetric key encryption (SKE) in the following:

Definition 2 (OT-IND for symmetric encryption scheme). Let $SKE = (\mathsf{Enc}, \mathsf{Dec})$ be a bijective one-time symmetric encryption scheme with security parameter k, $\mathcal{A}$ be a probabilistic polynomial time (PPT) adversary against the security of SKE in the sense of OT-IND. The advantage of $\mathcal{A}$ to win the following game must be negligible:

- The challenger uniformly chooses at random a secret $K \in \{0,1\}^l$, where l is an integer calculated from k.
- $\mathcal{A}$ is given the security parameter k. It then outputs a pair of messages (m_0, m_1) of equal length and passes them to the challenger.
- On receiving this pair, the challenger selects a bit $b \xleftarrow{\$} \{0,1\}$ and outputs the ciphertext $CT = \mathsf{Enc}(K, m_b)$ or $\perp$ if the messages do not have equal length.
- $\mathcal{A}$ receives the ciphertext CT and outputs b'. $\mathcal{A}$ wins the game if $b' = b$.

$\mathcal{A}$'s advantage is defined to be $Adv_{\mathcal{A}}^{OT-IND}(k) = 2Pr[b' = b] - 1$.

This property will be used in the formal security proof provided in Section 4.2.4.

2.1.2 Public Key Cryptography

Although the symmetric key cryptography outperforms in terms of efficiency, it has one major inconvenient, which is the need for secure key distribution channel. Indeed, when using a symmetric key encryption scheme, different entities are able to securely communicate together if and only if they have done the preparation of cryptographic keys. Such feature presents an important limitation when the number of communicating parties become large and when several parties wish to communicate privately from all the others. W. Diffie and M. E. Hellman [61] are the first to propose the theory of public key cryptography (PKC). In a PKC cryptosystem, each entity has a pair of keys: a public key and a private key. The two keys are usually mathematically associated. An attacker can find the private key from a public key if and only if he can resolve a hard mathematical problem. In the following, we give more details on the two variants of PKC: public key encryption and digital signature.

2.1.2.1 Public Key Encryption

Public key encryption allows secure transmission of a secret message. As an example, Figure 2.1 describes the procedure where Bob encrypts the message M under Alice's public key pk_{Alice}.

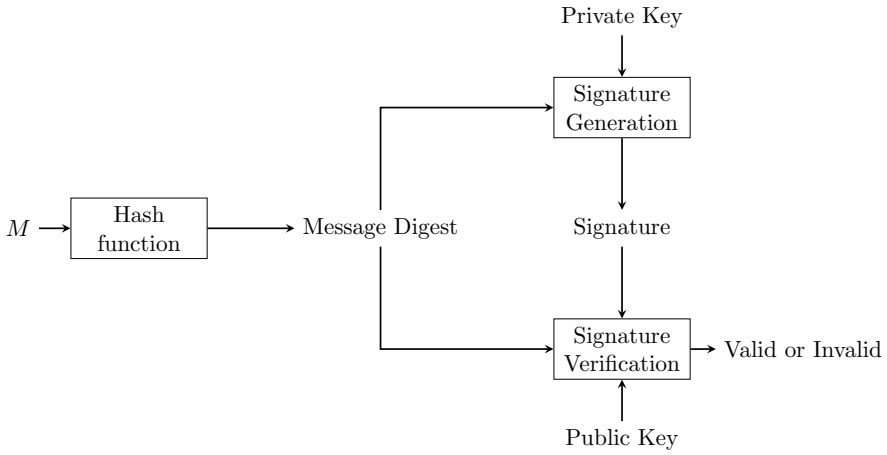


Figure 2.2: Digital signature processes [118]

The encrypted message is then sent to Alice. Only Alice can decrypt the ciphertext because she possesses the right private key $priv_{Alice}$.

As mentioned above, public key encryption schemes are usually based on a hard mathematical problem. For example, RSA [165] and Rabin [159] are built upon the Integer Factorization Problem, while ECC [100] is based on the Elliptic Curve Discrete Logarithm Problem. RSA [165] is the most popular algorithm for asymmetric cryptography. However, the algorithm is relatively slow, resource-demanding and hence incompatible with the constrained devices. On the other hand, ECC is very attractive for embedded systems as it is well-known to offer shorter keys, which leads to smaller computational requirements. More details on ECC are provided in Section 2.2.

2.1.2.2 Digital signature scheme

Digital signature is one of the most important public key primitives which is able to provide authentication, authorization and non-repudiation properties. It is an analog of written signature in the sense that the signer can claim the ownership of a document thanks to his signature. In the digital world, the signed documents are usually in the form of binary files or text messages. In order to generate a signature, the signer has to possess a key pair: a public key and a private key. A signature scheme consists of two algorithms: signature generation and signature verification. Figure 2.2 provides a brief description of the mentioned processes. As we shall see, the private key is employed in the signature generation process. This key must be kept secret, otherwise any body with the possession of such key can generate a valid signature. A digital signature scheme must satisfy the three following properties:

- *Unforgeability*: A digital signature is designed in a way that an adversary, who does not know the private key, cannot generate himself a valid signature on a different message. In other words, the signature generated by a signer can not be forged.
- *Verifiability*: A digital signature scheme must be *verifiable*. As such, the verifier must be able to mathematically validate a signature using the sender's public key. By doing so, he can be sure that the source message has not been modified. This procedure ensures the integrity of the transferred message.
- *Non-repudiation*: the verifier must be sure that the key pair owner actually generates the

signature. This property means that by proving the obtained signature and the signatory's public key, the verifier can prove to a "judge" that the signatory sent the message. A binding of a signer's identity and the signer's public key (e.g. given by the public key infrastructure [11]) shall be performed to provide such assurance.

2.1.3 Signcryption schemes

Suppose that Alice desires to send a message to Bob in a secure manner. Here, the latter means an authenticated communication. Their communication must be not only confidential, but also integrity protected. In case that they do not share any credentials *a priori*, a common approach is to use a combination of public key encryption and digital signature. Hence, one can *sign-then-encrypt* or *encrypt-then-sign* the data. Both mentioned methods require the addition of workloads of a public key encryption scheme and a digital signature scheme. These approaches are proved to be much more costly in terms of computation and communication complexity than the signcryption approach. This latter, initially proposed in [201], combines simultaneously signature and encryption in an optimized manner. As such, instead of using a public key encryption scheme to encrypt the data, a signcryption scheme uses actually a symmetric algorithm to encrypt/decrypt the secret data. Then, the unforgeability of the message is guaranteed by a digital signature. Formal definition of a signcryption scheme is given below.

2.1.3.1 Formal definition of a signcryption scheme

Definition 3 (A signcryption scheme). We define a signcryption scheme as a tuple of four probabilistic polynomial time (PPT) algorithms (Setup, KeyGen, $\mathsf{Signcrypt}$, $\mathsf{Unsigncrypt}$) with the following functionalities:
– $\mathsf{Setup}(k) \rightarrow cp$. Given a security level parameter k, output the public parameters cp. The other functions takes cp as an implicit input.
– $\mathsf{KeyGen}(cp) \rightarrow (priv_I, pk_I), (priv_R, pk_R)$. Generate public/private pair of keys for two parties (Initiator and Responder).
– $\mathsf{Signcrypt}(priv_I, pk_I, pk_R, M) \rightarrow C$ or $\perp$ (the error symbol). Given the public/secret keys of the Initiator, the public key of the Responder and a message M, return either a signcryptext C or $\perp$.
– $\mathsf{Unsigncrypt}(pk_I, priv_R, pk_R, C) \rightarrow M$. Given the signcryptext C, the public/secret keys of the Responder, the public key of the Initiator, return either a message M or $\perp$.

2.1.3.2 Example of Zheng's signcryption scheme

In this section, we describe the Zheng's original signcryption scheme (ZSCR) as mentioned in [201]. The algorithms in Definition 3 are briefly specified in the following:

- $\mathsf{Setup}(k) \rightarrow cp = (q, p, g, \mathsf{G}, \mathsf{H})$, where q is the finite field size, p is a large prime number such that $p|(q-1)$, g is chosen on $\mathbb{Z}_q$ such that p is its prime order, l is a positive number generated from the input security level k and $\mathsf{G} : \{0,1\}^* \rightarrow \{0,1\}^l, \mathsf{H} : \{0,1\}^* \rightarrow \mathbb{Z}_p$ are two hash functions.

- $\mathsf{KeyGen}(cp) \rightarrow (priv_I, pk_I), (priv_R, pk_R)$, where $priv_I$ and $priv_R$ are randomly generated from $\mathbb{Z}_p$. Additionally, $pk_I = g^{priv_I}$ and $pk_R = g^{priv_R}$.

- $\mathsf{Signcrypt}(priv_I, pk_I, pk_R, M) \rightarrow C$: To signcrypt a message M intended to R, I runs the following steps:

  1. Choose randomly x from $\mathbb{Z}_p$

2. Compute $K = pk_R^x$
3. Generate the ephemeral symmetric key: $\tau = \mathsf{G}(K)$
4. Compute $c = \mathsf{Enc}_\tau(M)$
5. Compute $r = \mathsf{H}(M, pk_I, pk_R, K)$
6. Compute $s = x/(r + priv_I)$
7. Obtain the ciphertext $C = (c, r, s)$

- $\mathsf{Unsigncrypt}(pk_I, priv_R, pk_R, C) \rightarrow M$: Upon receiving the ciphertext $C = (c, r, s)$ from I, R obtains the encrypted message by following the procedure below:

1. Compute $w = (pk_I g^r)^s$
2. Compute $K = w^{priv_R}$
3. Compute $\tau = G(K)$
4. Compute $M = \mathsf{Dec}_\tau(c)$
5. Verify if $\mathsf{H}(M, pk_I, pk_R, K) = r$

We can observe that ZSCR combines the symmetric encryption and a shortened ElGamal signature [71] in order to generate the ciphertext. The scheme has been formally proved to be confidentially and unforgeably secure in the random oracle [201]. In addition, ZSCR offers good computation performance since it requires only 4 exponentiations to complete a secure message transfer.

2.1.4 Attribute-Based Encryption

Attribute-Based Encryption (ABE) is a public key primitive where both encryption and decryption are based on *attributes* (e.g. job position, gender...). In addition, a user can restrict access to its data by defining an *access policy*. Sahai et al. [168] are the first to propose the notion of ABE. Their encryption scheme is initially applied for Identity-based encryption (IBE), where the entity identifier is considered as a combination of attributes. Subsequently, many related encryption schemes based on the ABE paradigm have been introduced. There exist two forms of ABE: Key-Policy Attribute-Based Encryption (KP-ABE) [91] or Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [28, 127, 195]. In a KP-ABE system, the user's private key is associated with access policies, while a set of attributes is embedded in the ciphertext. On the other hand, CP-ABE embeds the access policy in the ciphertext, while user keys are associated with attributes. Such property gives users the flexibility to determine who can decrypt data at runtime. In chapter 6, we propose a solution that makes changes to the original CP-ABE scheme proposed by Bethencourt et al. [28]. Similar applications of our proposal also work for the schemes proposed in [127, 195] but it is out of scope of this dissertation.

In this section, we first provide general definitions of an access structure and bilinear map. Then, we present a brief formal description of CP-ABE. Finally, we present some desirable properties of an ABE scheme.

2.1.4.1 Access structure

We define an access structure in the following:

Definition 4 (**Access structure** [25])**.** Let $\{P_1, P_2, ...P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, ...P_n\}}$ is monotone if $\forall B, C$: if $B \in A$ and $B \subseteq C$ then $C \in A$. An access structure (respectively, monotonic access structure) is a collection (respectively, monotone collection) $\mathbb{A}$

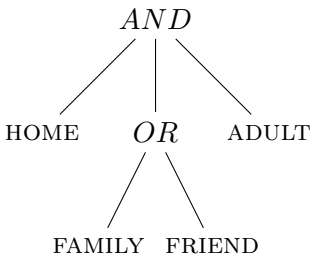


Figure 2.3: Example of an access policy

of non-empty subsets of $\{P_1, P_2, ...P_n\}$, i.e. $\mathbb{A} \subseteq 2^{\{P_1,P_2,...P_n\}} \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets , and the sets not in $\mathbb{A}$ are called the unauthorized sets.

An access structure $\mathbb{A}$ is often used in a secret sharing scheme [25] to define the sets of parties that can reconstruct the master secret key using their shared part of the master secret key. In the context of ABE scheme, the role of the parties is taken by the attributes [92]. Thus, $\mathbb{A}$ will contain the authorized sets of attributes. An access structure $\mathbb{A}$ is called *monotone access structure* if it does not contain the negation value of attributes. On the contrary, if $\mathbb{A}$ contains the "not" of each attribute, it is mentioned as a *non-monotone access structure*.

In an ABE scheme, an access structure is generally represented by an access tree. Each non-leaf node of an access tree represents a threshold-gate, such as AND and OR, and the leaves are associated with attributes. In general, we can define an access tree from a provided access policy. Figure 2.3 gives an example of an access policy. In a CP-ABE scheme, a user can decrypt an ABE ciphertext if and only if the attributes embedded in the user's private key satisfy the access tree. Please refer to [28] for more details on the construction of an access tree.

For illustration, we give a concrete example of an ABE application using the access policy defined in Figure 2.3. Bob desires to authorize the permission of driving his car to a specific person. A user can drive his car by having the key or presenting a token using his smartphone. Bob encrypts the token using an ABE mechanism. The user has to decrypt the ciphertext sent by Bob to get the token. In Bob's access policy, the user cannot start the car if the location is not at home. He must also be a family member or a close friend to Bob. In addition, if he is not an adult, he cannot start the car. Such condition prevents his child at home from driving the car. By using the access policy defined in Figure 2.3, Bob can easily express his requirement.

2.1.4.2 Bilinear Map

In an ABE mechanism, some facts about groups with efficiently computable bilinear maps will be used.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two cyclic elliptic curve groups of prime order p over finite field $\mathbb{F}_p$. Let P be a generator of $\mathbb{G}_0$ and e be a bilinear map, such that $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. e has the two following properties:

- Bilinearity: for all $R, S \in \mathbb{G}_0^2$ and $a, b \in \mathbb{Z}_p^2$, we have $e(aR, bS) = e(R, S)^{ab}$.
- Non-degeneracy: $e(P, P) \neq 1$.

Both operations on $\mathbb{G}_0$ and the bilinear map e must be efficiently computable.

2.1.4.3 Formal definition of Ciphertext-Policy Attribute-based Encryption

A CP-ABE scheme consists of four fundamental algorithms Setup, KeyGen, Encrypt and Decrypt with the following functions:

- $\mathsf{Setup}(sp) \rightarrow (PK, MK)$. This algorithm takes as input the security parameter sp and returns the system public key PK as well as the system master key MK. PK is used for message encryption, while MK is used to generate user private keys and is known only to the authority.
- $\mathsf{KeyGen}(MK, S) \rightarrow SK$. This algorithm takes as input the system master key MK, the set of attributes S and returns the user secret key SK.
- $\mathsf{Encrypt}(PK, M, \mathcal{T}) \rightarrow CT$. This algorithm takes as input the public key PK, the message M and the access structure T. It then outputs the ciphertext CT.
- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \rightarrow M$. This algorithm takes as input the ciphertext CT, the user secret key SK and returns the message M.

In ABE, including KP-ABE and CP-ABE, the algorithms Setup and KeyGen are run by a trusted authority in order to generate the system keys PK, MK and user private keys. Any user who has the public key PK can generate a ciphertext by using the algorithm Encrypt. However, only authorized users (i.e. users possessing a private key that satisfies the access structure embedded in the ciphertext) can decrypt the ciphertext by calling the algorithm Decrypt.

2.1.4.4 Properties of Attribute-based Encryption Schemes

As also mentioned in [126, 156], an ABE scheme is expected to provide the following features:

- *Data confidentiality*: The data to be sent using an ABE scheme is encrypted before any transmission. Unauthorized parties cannot learn any information from the encrypted data.
- *Fine-grained access control*: By providing different private keys for users, the system authority is able to restrict user access rights to specific resources.
- *Scalability*: Even if the number of authorized users increases, the system can work efficiently.
- *User revocation*: The system can revoke the access rights of a user at any moment. The revoked user can not access encrypted data anymore.
- *Collusion resistance*: If multiple users collude, they are only able to decrypt an ABE ciphertext if at least one of the colluded users can decrypt it. In other words, the colluded users can not combine their secret keys to generate new secret keys with superior access rights.

Some related work [101, 129] mentioned also another feature for an ABE scheme, named *user accountability*. This property ensures that legal user private keys can not be shared between unauthorized users. For example, the authors in [101] propose to bind user personal information to the decryption policy. As such, a token server is introduced to take part in every decryption process. Before any decryption operation, users have to get a token T from the token server in order to successfully decrypt the ciphertext. This interaction removes the key misuse problem and presents a simple way to revoke users.

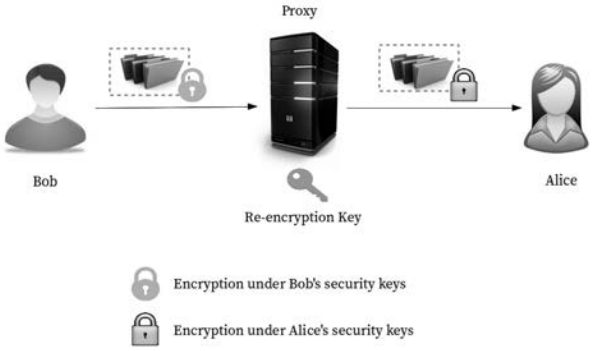


Figure 2.4: General architecture of a proxy re-encryption scheme

2.1.5 Proxy re-encryption schemes

In this section, we specify the most useful properties of a Proxy Re-encryption (PRE) scheme and general definition of a symmetric cipher proxy re-encryption scheme.

2.1.5.1 Properties of a proxy re-encryption scheme

In a proxy re-encryption scheme, as described in Figure 2.4, Bob can delegate the decryption right on an encryption to Alice with the help of a *semi-trusted* proxy (i.e. An entity that acts and returns correct results according to demanded tasks but can be untrusted when processing sensitive data). In general, the proxy uses a prior provided secret, namely, proxy key or re-encryption key, to translate a ciphertext dedicated to Bob to another ciphertext dedicated to Alice. However, it cannot gain any information on the secret keys of Bob or Alice and is unable to read the content of the encrypted messages.

Proxy re-encryption schemes are characterized according to different criteria. The works in [94] and [31] provide several properties by which to compare different proxy re-encryption schemes. We briefly redefine these desirable properties as follows.

- Uni-directionality: The proxy re-encryption scheme is said to be *unidirectional* if the re-encryption key of the proxy can be used in only one direction. In contrast, a *bidirectional* proxy re-encryption scheme permits the re-encryption key to be used to translate encrypted messages from Alice to Bob and vice versa.

- Non-Interactivity: In a *non-interactive* scheme, Alice can generate a re-encryption key, while offline, from its secret key and Bob's public values without the participation of the Key Distribution Center (KDC), the proxy, or Bob. On the other hand, *interactive* schemes require the participation of parties (including KDC) to generate the re-encryption keys.

- Multiple-use: Some proxy re-encryption schemes can re-encrypt a ciphertext multiple times. For example, Bob can demand a re-encryption of a ciphertext re-encrypted for him which is previously intended to Alice to obtain a ciphertext dedicated to Charlie without actually decrypting the message. Such scheme is called *mutiple-use*. In opposition, a *single-use* proxy re-encryption scheme permits the proxy to perform only one re-encryption on a ciphertext.

- Non-transitivity: In a *non-transitive* scheme, the proxy cannot combine provided re-encryption keys to re-delegate decryption rights. For example, given three entities A,

B and C, the proxy is unable to construct the re-encryption key $rk_{A\to C}$ from A to C from the two supplied re-encryption keys $rk_{A\to B}$ and $rk_{B\to C}$.

- Collusion resistance: In a proxy re-encryption scheme, it is desirable that Bob even colluding with the proxy, can not guess the secret key of Alice.

2.1.5.2 Symmetric cipher proxy re-encryption

We define below the main algorithms of a proxy re-encryption scheme that employs symmetric ciphers which is more adapted to be used in constrained sensor devices.

Definition 5 (Symmetric cipher proxy re-encryption)**.** A symmetric cipher proxy re-encryption consists of five algorithms (KeyGen, ReKeyGen, Encrypt, Decrypt, Reencrypt) with the following functions:

- $\mathsf{KeyGen}(k) \to (id_A, id_B, sk_A, sk_B)$. Given a security level parameter k, output the identifiers and the secret keys for two entities A and B. These keys are to be used in the encryption and decryption processes.
- $\mathsf{ReKeyGen}(id_A, id_B, sk_A, sk_B) \to rk_{A\to B}$. Given the identifiers and secret keys of A and B, output the re-encryption key $rk_{A\to B}$.
- $\mathsf{Encrypt}(id_A, sk_A, M, id_B) \to C_A$. Given the identifiers (id_A, id_B), the secret key sk_A and a message M, return the ciphertext C_A.
- $\mathsf{Reencrypt}(rk_{A\to B}, C_A) \to C_B$. Given a ciphertext C_A encrypted by the entity A and the re-encryption key $rk_{A\to B}$, return a ciphertext C_B to be decrypted by B.
- $\mathsf{Decrypt}(id_B, sk_B, C_B, id_A) \to M$. Given a secret sk_B and a ciphertext C_B and the identifiers (id_A, id_B), return the plaintext M.

2.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public key cryptosystem. It was independently discovered by Victor Miller [144] and Neil Koblitz [120] in 1985 as an alternative mechanism for implementing public key cryptography. Unlike RSA, ECC-based protocols are based on the problem of finding the discrete logarithm of random elliptic curve element (also known as ECDLP), which is much more difficult to resolve at equivalent key lengths [100].

In this section, we first describe the basic elliptic curve operations. Then, we define some problems which are assumed computationally unbreakable in ECC.

2.2.1 Basic ECC operations

To be used in cryptography, ECC operations consider finite field (Galois Field) algebra with focus on prime field (e.g. $\mathbb{F}_t$) and extended binary field (e.g. $\mathbb{F}_{2^m}$), since the latter provide fast and precise arithmetic. In this section, we introduce some details on the elliptic curve operations on the prime finite field.

The elliptic curve $E(\mathbb{F}_t)$ on the prime finite field $\mathbb{F}_t$ is actually defined by the following equation:

$$y^2 = x^3 + ax + b \bmod p, \text{ where } a, b \in \mathbb{F}_t \text{ and } 4a^3 + 27b^2 \neq 0 \tag{2.1}$$

Each point on the elliptic curve can be represented as a vector in affine coordinates such as $P = (p_x, p_y)$, where p_x and p_y must satisfy the equation 2.1. The only difference is that all

coordinates must be integers in modular p. A special point O is defined in the elliptic curve as point at infinity. In order to do any meaningful cryptographic operation, one has to understand the calculation of points on the elliptic curve. Three point operations are considered in ECC, as described in the following:

- Point addition: Considering the two points $P = (p_x, p_y)$ and $Q = (q_x, q_y)$, there exists a point $R = (r_x, r_y)$ such that $R = P + Q$ where P is distinct from Q and is also not $-Q = (q_x, -q_y)$. We can calculate each coordinate of R using the following formulas:

$$\begin{aligned} s &= \frac{p_y - q_y}{p_x - q_x} \\ r_x &= s^2 - p_x - q_x \\ r_y &= s(p_x - r_x) - p_y \end{aligned}$$

  For each point P on the elliptic curve, we also have $P + (-P) = O$ and $P + O = P$.

- Point doubling: This returns to the case where we find the point R such that $R = 2P$. We can obtain the coordinates of R by employing the following formulas:

$$\begin{aligned} s &= \frac{3p_x^2 + a}{2p_y} \\ r_x &= s^2 - 2p_x \\ r_y &= s(p_x - r_x) - p_y \end{aligned}$$

- Point multiplication: By using only the point addition, we can achieve the result of the multiplication between one point and a natural number k. As such, we can find the point R such that $R = kP$ by applying k point additions. Such multiplication can also be done using the simple *double and add algorithm* [100], which requires only $O(log\, k)$ steps. The inverse of that point multiplication, i.e. find the value of k from R and P, is known as *discrete logarithm*. This problem is considered computationally infeasible and is recognized as the foundation of ECC cryptosystems. More detailed examples on ECC operations can be found in [100, 45].

In order to use ECC, all parties must agree on a list of parameters that define an elliptic curve. This list is termed as *domain parameters*. For example, the elliptic curve $E(\mathbb{F}_q)$ domain parameters over prime field $\mathbb{F}_q$ can be defined by the sextuple (q, a, b, G, p, h), where t is the field size, two elements a, b specify the elliptic curve $E(\mathbb{F}_q)$, G is the base point, p is the prime order of G and $h = \#E(\mathbb{F}_q)/p$, is the cofactor. Further guidance on the selection of recommended domain parameters for elliptic curve cryptography can be found in [9].

2.2.2 Computational Hardness Assumptions for ECC

Let $\mathbb{G}$ be a cyclic group of prime order p. For our purposes, $\mathbb{G}$ is a subgroup of points of a suitable elliptic curve $E(\mathbb{F}_p)$ over finite field. P is a generator of $\mathbb{G}$. We define the following security assumptions:

Definition 6 (**Decisional Diffie-Hellman (DDH) Problem**)**.** Given the "Diffie-Hellman tuples" $< P, [a]P, [b]P, [c]P >$, decide whether $ab \equiv c\,(mod\, p)$ or not.

Definition 7 (**Computational Diffie-Hellman (CDH) Problem**)**.** Given $< P, [a]P, [b]P >$, for unknown $a, b \in \mathbb{Z}_p$, compute $[ab]P$.

Definition 8 (**Gap Diffie-Hellman (GDH) Problem**). Given that the DDH problem is easy in $\mathbb{G}$, solve an instance of the CDH problem $< P, [a]P, [b]P >$.

Definition 9 (**Discrete Logarithm Problem (DLP)**). Given the two points P and Q on the elliptic curve over finite field $\mathbb{F}_p$, find $d \in \mathbb{Z}_p$ so that $[d]P = Q$.

Definition 10 (**Gap Discrete Log (GDL) Problem**). Given that the DDH problem is easy in $\mathbb{G}$, solve an instance of the DLP problem $< P, Q >$.

More details on the assumptions can be found in [100, 151].

2.3 Implicit Certificate

In a public key infrastructure, the existence of a Certificate Authority (CA) is essential. As such, users of a public key need to be sure that the associated private key is possessed by the correct entity (person or system). This assurance can be provided by having a trusted CA that binds the public key of an entity and its identity in the form of a public key certificate. In fact, the latter contains a CA's signature on public information of a subject. If this signature is valid then one will have confidence that the data embedded in the certificate is correct. In other words, the user named in the certificate has the right private key associated to the public key included in the certificate.

The main problem of PKIs is that the size of each certificate is approximately 1 KB [104]. This requirement is sometimes impossible for certain range of resource-constrained devices (e.g. class 0 and class 1 devices [37]) or environments with limited bandwidth capacity. Consequently, one alternative for the PKI infrastructure in the resource-constrained environments is the *implicit certificate*. Implicit certificates are first introduced in the work of [89, 98]. As mentioned in [141], the implicit certificate public key schemes can be classified into two categories: (i) self-certified implicit certificate schemes and (ii) identity-based implicit certificate schemes. The two approaches keep the principal idea of using the CA's public key, user's identity and some reconstruction public data to regenerate the user's public key. This technique permits to embed directly a certificate to the public key instead of requiring a separate value. In the first category, the user computes himself his private key and the associated public key. In the second category, the CA is responsible for generating the user's private key. This approach is more lightweight than the first one as CA computes the private key for users. However, it has one major inconvenient which is the key-escrow attack that can be performed by the CA, where the CA is able to masquerade as the user.

Employing implicit certificates is a suitable approach for use in the constrained environments. Indeed, several works [174, 155] propose to use implicit certificates in key agreement protocols in the context of IoT. In chapter 4, we propose a key distribution mechanism that relies on *identity-based implicit certificate* public key approach. Even if our proposition requires a trusted key distribution center, it is very much lightweight and hence is suitable to the resource-constrained feature of WSNs and IoT.

2.4 Summary

In this chapter, we present a general introduction on cryptographic primitives and several related notions which are useful for further reading of this dissertation. Beside basic notations of symmetric/asymmetric cryptography, we introduce more details on signcryption, attribute-based encryption and proxy re-encryption. They are three asymmetric primitives that relate to our three proposals in chapters 4, 5 and 6, respectively. Then, we provide general information on the

elliptic curve cryptography (ECC). The reason is that all of our proposals are constructed based on ECC. In fact, ECC is a promising approach since it reduces significantly the key sizes and hence more relevant in the context of IoT than other asymmetric primitives such as RSA. We describe also several mathematically hard problems in the ECC setting. These assumptions are to be used throughout this thesis, especially in the security proof provided in chapter 4. Finally, we take a look at several related works on implicit certificate. The latter is a basic method to remove the burden of PKI in the context of constrained environments such as IoT.

Chapter 3

Lightweight Cryptographic Primitives and Secure Communication Protocols for IoT

All secure communication protocols and mechanisms must consider the resource-constrained nature of IoT devices. Recent researches [135, 53] survey multiple solutions to be used in constrained environments, such as IoT. As a secure communication protocol relies also on the used cryptographic algorithm, we classify these solutions into two main approaches: i) *lightweight cryptographic primitives* and ii) *lightweight cryptographic protocols and methods*. In the first approach, the main objective is to design and employ ultra-lightweight cryptographic algorithms that can be applied in IoT environments while proving desired security levels. On the other hand, the second approach concentrates on using relevant cryptographic primitives and techniques in order to build security protocols and methods for secure communications.

In this chapter, we survey several existing lightweight cryptographic primitives and secure communication solutions according to the two approaches mentioned above. Section 3.1 presents some related work on lightweight cryptographic primitives to be used in resource-constrained environments. Then, Section 3.2 classifies some lightweight security mechanisms and protocols with respect to their provided security services and performances. Finally, Section 3.3 presents some promising trends and ideas towards lightweight security solutions suitable for use in the IoT.

3.1 Lightweight cryptographic primitives for IoT

The rapid deployment of the Internet of Things implies an extensive coverage of billions of smart objects to the conventional Internet. The IoT devices are usually featured with constrained resources, such as low computation capability, low memory and limited battery. These constraints must be taken into account when it comes to selecting the appropriate security protocol to be used. As a result, many lightweight cryptographic primitives are proposed in the context of IoT while considering the trade-off between security guarantee and good performance.

Lightweight cryptographic primitives consist of cryptographic algorithms that meet the requirement of constrained environments. However, this does not mean they are less secure ciphers. In fact, these primitives should be referred to small footprint, low power computation and low energy consumption ciphers. In addition, they are supposed to support a sufficient security level even if adapted to resource-limited devices.

In this section, we provide an overview of lightweight cryptographic primitives in the lit-

Name	data size (e.g., RAM)	code size (e.g., Flash)
Class 0	<< 10 KB	<< 100 KB
Class 1	10 KB	100 KB
Class 2	50 KB	250 KB

Table 3.1: Classes of Constrained Devices (KB = 1024 bytes) [37]

erature. Lightweight symmetric ciphers are presented first. Then, several asymmetric ciphers, which can be suitable for resource-constrained devices, are introduced as well. Both approaches are actively studied by ongoing researches in order to provide suitable cryptosystems for IoT applications.

3.1.1 Symmetric Key Ciphers

For certain groups of ultra-constrained devices (e.g. class 0 and class 1 devices as depicted in Table 3.1), it is mostly impossible to communicate with other Internet nodes using a full Internet protocol stack, such as HTTP and Transport Layer Security (TLS). In addition, symmetric cryptography is widely believed to be suitable for resource-constrained devices. As a result, many recent researches try to optimize and design new symmetric algorithms for use in the Internet of Things.

The symmetric algorithms are designed to work in both software and hardware implementations. The software-based approach requires in general a microprocessor on embedded devices to operate, which makes it more portable than the hardware-based one. Indeed, hardware-based ciphers are designed to reduce the logic gates to be used to materialize the cipher [135]. Such optimization limits the deployment of these ciphers on specific platforms. However, these hardware-oriented algorithms can find their place in ultra-constrained devices, such as RFID tags. As such, the authors in [53] compare some lightweight symmetric ciphers in terms of key size, block size, consumed resources measured in gate equivalents (GE) and code size (in bytes) in both software and hardware implementations. The results show that the hardware-oriented ciphers, such as, PRESENT [34] and HIGHT [103] perform well when compared to software-based ciphers.

Among the proposed algorithms, the standardized cipher AES [55] is one of the most well-known and fastest symmetric algorithm. Even being really fast, AES requires significantly larger code size than several recently proposed symmetric algorithms, such as: Speck, Simon [24], PRESENT [34], HIGHT [103] and Piccolo [178]. In [62], the authors confirm this fact by evaluating 13 lightweight symmetric ciphers in terms of execution time, RAM footprint and code size. In particular, they conclude that Simon and Speck [24] are the smallest and fastest ciphers on the evaluated platforms.

Although the performance of symmetric key encryption algorithms are much more efficient than the asymmetric ones, the rapid deployment of smart devices requires through unpractical key provisioning and management. As such, the main difference between symmetric and asymmetric cryptosystems is the way the secret keys are employed. Symmetric cryptography uses a unique key for both encryption and decryption processes. As a result, the pair-wise secret key between any two entities must be agreed prior to communication. However, such key pre-distribution mechanism does not scale well in a large IoT networks.

3.1.2 Public Key Ciphers

Unlike symmetric cryptography, public key algorithms use a pair of public/private keys. Any one can encrypt a message under the public key of another party. However, only this latter can decrypt the encrypted data. The major advantage of using public key cryptography is that it facilitates the establishment of secure communications between any two parties previously unknown to each other. Public key algorithms are more scalable than their symmetric counterparts, but they are usually more complex in terms of computation.

We can mention some traditional asymmetric primitives such as ECC [144], RSA [165], Rabin [159] or ElGamal [159]. Among them, ECC [144] is usually mentioned as a suitable and attractive cryptosystem for embedded systems. Its main advantage is its key length which is smaller than RSA at the same security level.

Recent researches are also interested in the application of hyperelliptic curve cryptography (HECC) [140] in resource constrained environments. A hyperelliptic curve is a generalization of an elliptic curve. As such, a hyperelliptic curve of genus 1 is an elliptic curve. HECC is believed to provide better performance than ECC in terms of memory space and computational overhead. The authors in [193, 115, 78] proved the performance of HECC by presenting its applications in the Wireless Sensor Networks and several RFID tags.

As another lightweight cipher, NTRU [102] is one of the most promising cryptosystem for resource-constrained devices. It is a quantum-resistant cryptosystem which is known to be a lattice-based alternative to RSA and ECC. Indeed, its cryptographic elementary operations require only polynomial multiplications, which are highly efficient and suitable for resource-limited devices. Some other cryptosystems, such as the Code-based Cryptography McEliece [138] or the Multivariate-Quadratic [153], are efficient in the encryption and decryption processes and are good candidates for post-quantum cryptography. However, these cryptosystems require large key sizes, even when compared with RSA.

3.2 Lightweight protocols and methods for establishing secure communications in IoT

In this section, we describe different approaches that have been employed to propose lightweight security mechanisms in the context of IoT. We focus mainly on the key establishment protocols. Section 3.2.1 discusses some required security properties for the IoT. Section 3.2.2 gives a classification of recently proposed security protocols for IoT. In sections 3.2.3 and 3.2.4, we provide in-depth description of the protocols based on asymmetric key schemes and the protocols based on symmetric key pre-distribution schemes. Section 3.2.5 evaluates the solutions according to the different security criteria.

3.2.1 Security properties

Several security properties may need to be satisfied in order to secure the IoT. These general security properties have been also identified in [85], [173]. Generally, the security services that should be provided include confidentiality, integrity, authentication, authorization, and freshness. The security requirements are centered on data if sensitive data measured or shared by IoT devices may need to be protected. Security requirements may also involve controlled access to other resources, for instance the IoT network layer. The following security properties will be discussed in this section in relation with the security protocols and solutions proposed for the IoT:

- Confidentiality: Exchanged messages in the IoT may need to be protected. An attacker should not gain knowledge about the messages exchanged between a sensor node and any other Internet entity.
- Integrity: The alteration of messages should be detected by the receiver.
- Authentication: The receiver should be able also to verify the origin of the exchanged messages.
- Authorization: IoT devices should be able to verify whether certain entities are authorized to access their measured data. At the network layer, only authorized devices should be able to access the IoT network. Unauthorized devices should not be able to route their messages over the IoT devices, because it may deplete their energy.
- Freshness: This property ensures that no older messages are replayed. This is important to secure the communication channel against replay attacks.

3.2.2 Taxonomy of key establishment protocols for the IoT

The life cycle of a "thing" is composed of three phases (as denoted in [85]): bootstrapping, operational and maintenance phases. The bootstrapping phase refers to any processing tasks required before the network can operate. Sarikaya et al. [173] also define that this process involves a number of settings to be transferred between nodes that shared no prior knowledge of each other. The bootstrapping step of a device is complete when all security parameters (e.g., secret keys) are securely transferred to the device. This study focuses on recent security solutions proposed for a secure bootstrapping process.

In this section, we first describe the reference model in section 3.2.2.1 that illustrates the scenario in which the considered security protocols can be deployed. We then present, in section 3.2.2.2, our classification of the security protocols based on the key bootstrapping mechanism, and compare, in section 3.2.2.3, our classification with related work.

3.2.2.1 Scenario under consideration

The security protocols analyzed in this chapter, as illustrated in Figure 3.1, involve two entities. At least one of them is a device with resource constraints, whereas the second entity can be seen as another constrained device or an external Internet server (i.e., with rich resources). The considered network of "things" consists of a number of tiny nodes communicating with each other and with an unconstrained resource border router (6LBR). The 6LBR is the bridge between the sensor node and the outside world. The 6LBR may take part in the communication between two entities in a passive (transparent to the communicating parties) or active (as a mediator in the communication process) manners. Our study concentrates mainly on securing unicast communications between two entities. Note that group communications are out of scope of this chapter.

3.2.2.2 Classification

In this chapter, existing security solutions for IoT is categorized into two main types: solutions that rely on asymmetric key schemes and solutions that pre-distribute symmetric keys to bootstrap a secure communication. This section describes the two first levels of the proposed taxonomy.

– **Asymmetric key schemes (AKSs)**: The key schemes based on asymmetric cryptography, also known as Public-key cryptography (PKC) are considered as a very common approach to

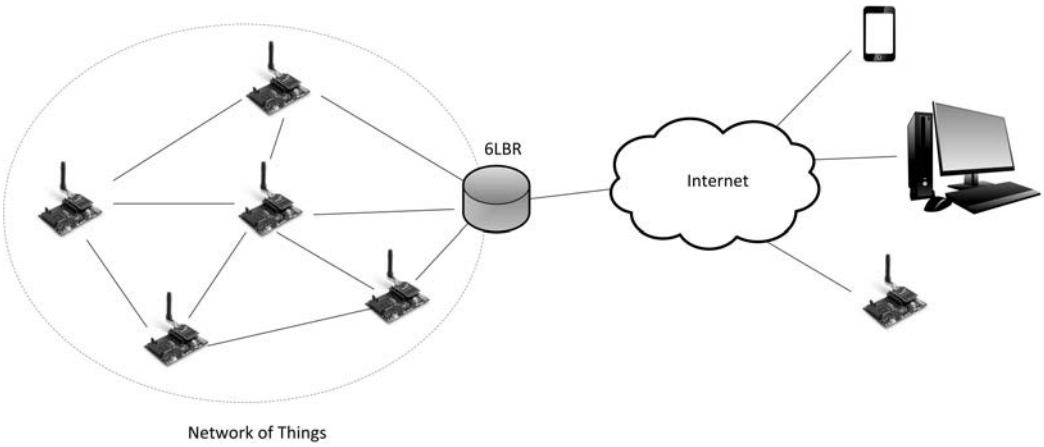


Figure 3.1: Network architecture of our scenario

establish a secure communication between two (or more) parties. They employ asymmetric algorithms and are widely deployed in the conventional Internet. The applicability of AKSs in the IoT has one major inconvenience, which is the computation cost and energy consumption. Despite of expensive operations, a lot of researches still seek to apply AKSs in the context of IoT. The proposed approaches can be classified into two categories: key transport based on public key encryption and key agreement based on asymmetric techniques.

- *Key transport based on public key encryption*: Similarly to the traditional key transport mechanism, the first category requires from the public key to securely transport information. Various key establishment techniques have been proposed for IoT, ranging from raw public key usage to complex implementations in X.509 standard.

- *Key agreement based on asymmetric techniques*: The second category is based on asymmetric primitives in which a shared secret is derived among two or more parties. In this category, we notice obviously the DH protocol [11] and its variants as we will mention later.

– **Symmetric key pre-distribution schemes**: In addition to asymmetric approaches, researchers propose also multiple techniques using symmetric key establishment mechanisms to bootstrap secure communication in the IoT. Symmetric approaches often assume that nodes involved in the key establishment share common credentials. The pre-shared credentials might be a symmetric key or some random bytes flashed into the sensor before its deployment. This category can be divided into two main sub-categories:

- *Probabilistic key distribution*: This sub-category concerns the mechanisms that distribute security credentials (keys, random bytes) chosen randomly from a key pool to constrained nodes. During their initial communication, each two nodes may discover a common key, with certain probability, to establish a secure communication.

- *Deterministic key distribution*: In this sub-category, a deterministic design is applied to create the key pool and to distribute uniformly the keys such that each two nodes share a common key.

Figure 3.2 summarizes our taxonomy. Each class of the security solutions provides its own advantages and disadvantages, as it will be discussed in Sections 3.2.3 and 3.2.4.

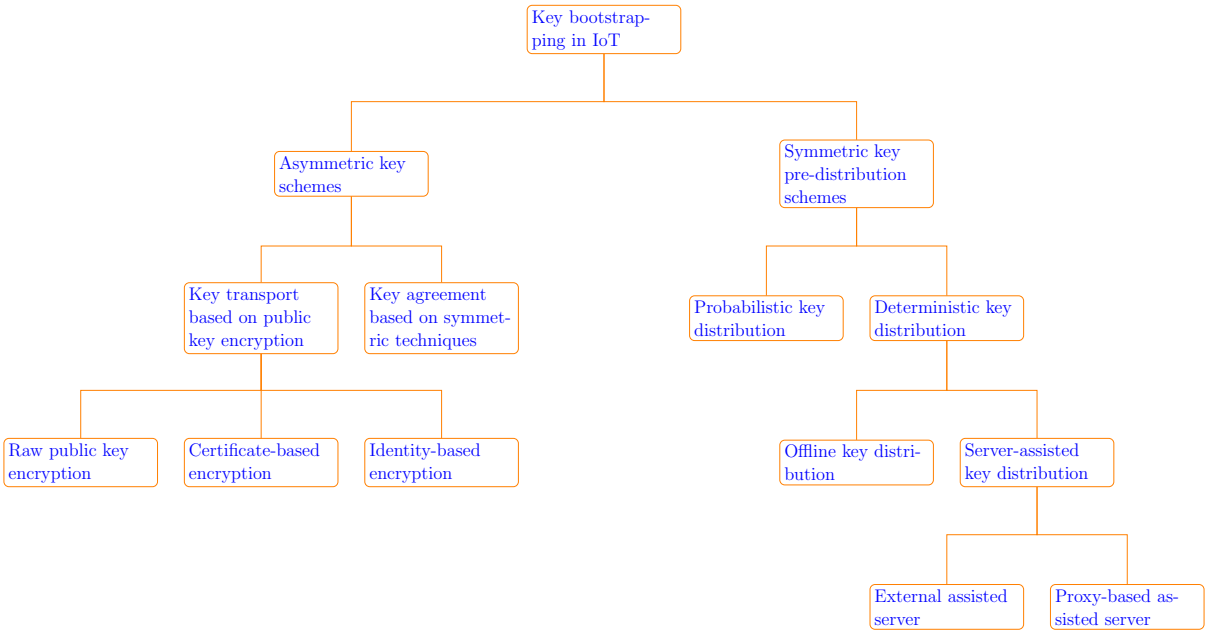


Figure 3.2: Classification of key bootstrapping mechanisms in IoT

3.2.2.3 Related work in IoT security protocol classification

Classification approaches have been proposed in several works [169, 43, 166, 194]. In [169], the authors propose several ways to classify key establishment approaches, for instance based on the employed authentication method or the underlying cryptographic primitive. Camtepe et al. [43] give a detailed classification of symmetric key distribution protocols for two different scenarios: distributed and hierarchical WSNs. In each scenario, the authors analyze diverse mechanisms to establish pair-wise and group-wise keys between sensor nodes. Similarly, Wang et al. [64] propose a classification of symmetric key management protocols in WSN, but based on the network structure and the probability of key sharing between a pair of sensor nodes. Their works at a very first level differentiate centralized and distributed key schemes. At a second level, they provide other differentiation based on the probabilistic and deterministic key establishment mechanisms. Roman et al. [166] give a high level classification based on the key management systems (KMS), namely: key pool framework, mathematical framework, negotiation framework and public key framework. They conclude that public key cryptography can be a viable solution for sensor nodes that run as client nodes (in the model client-server). For server nodes, mathematical-based KMS, such as polynomial scheme, provide better performances. The aforementioned approaches do not sufficiently cover possible key distribution mechanisms (asymmetric and symmetric methods), for example, only symmetric approaches are studied in [43, 194]. Besides, they provide heterogeneous classifications due to unrelated different criteria, as in [166, 194]. By taking into account the classifications described above, especially in [169], our taxonomy covers asymmetric key distribution mechanisms for IoT, in addition to symmetric approaches. The taxonomy is marking out different protocols by the key establishment scheme used to establish a secret session key: asymmetric or symmetric techniques. As mentioned in section 3.1, we do not consider protocols that establish group-wise keys between sensor nodes for which interested readers could refer to [43]. Only pair-wise key establishments are considered in this chapter. Our taxonomy has a high classification degree leading to a more in depth protocol evaluation. For instance, in the asymmetric approach, we do not only discuss on the applicability of public key cryptography in the context of IoT, as described in [166], but

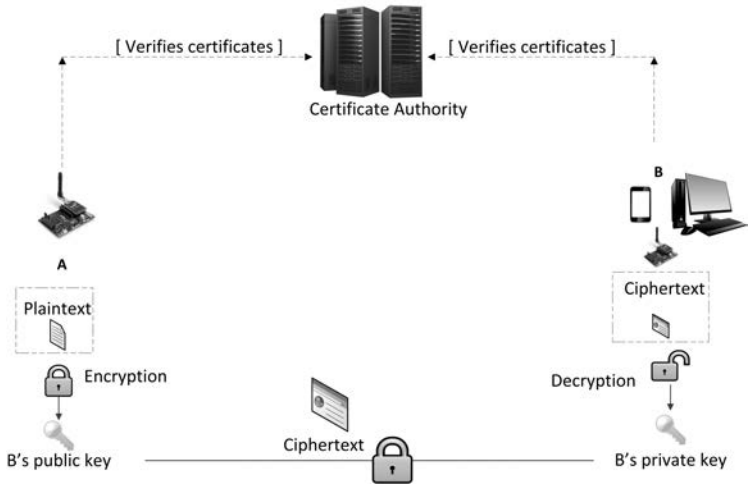


Figure 3.3: Public key transport mechanism

we also differentiate different asymmetric key schemes based on the key delivery scheme (key transport or key agreement).In symmetric key pre-distribution schemes, we organize the existing security protocols into two categories: probabilistic and deterministic key distribution. These categories have also been mentioned in [43, 194]. However, in the deterministic approach, we go further by distinguishing protocols that have server(s) participating in the key negotiation process from protocols that do not depend on any third party during key establishment phase.

3.2.3 Asymmetric key schemes

The position of asymmetric cryptography or PKC is clear in the conventional Internet. However, it is not the case in the context of IoT because of its expensive encryption and verification operations. However, the development and implementation of PKC in IoT has never been stopped. In fact, new improvements of several primitives (i.e. ECC, NTRU) continue to reduce the cost of cryptographic operations, so the PKC approach is of a growing interest for constrained environments. A brief study in the following sections demonstrates various possible forms of asymmetric key schemes in IoT.

3.2.3.1 Key transport based on public key encryption

This sub-category looks into the key establishment schemes where the public key is used to transport secret data or to negotiate a session key. Several methods are used to generate the pair of public and private keys. In this sub-category, we classify these mechanisms based on the public/private keys generation methods.

Figure 3.3 gives an example of a communication scenario between two entities A and B. In this scenario, A and B can use directly the public keys to create an encrypted channel. The Certificate Authority (CA) may participate to verify the identity of the message transmitter when certificates are supported. This method can be expensive for resource-constrained- sensor nodes, in particular when using a traditional algorithm like RSA. Without a verifiable relationship between the public key and the identity (i.e. ID-based cryptography, cryptographic-based ID or with CA mediation), this approach becomes vulnerable to the man-in-the-middle attack. Indeed, both A and B cannot authenticate each other's identity. An attacker may generate any public/private keys and pretend to be A when communicating with B.

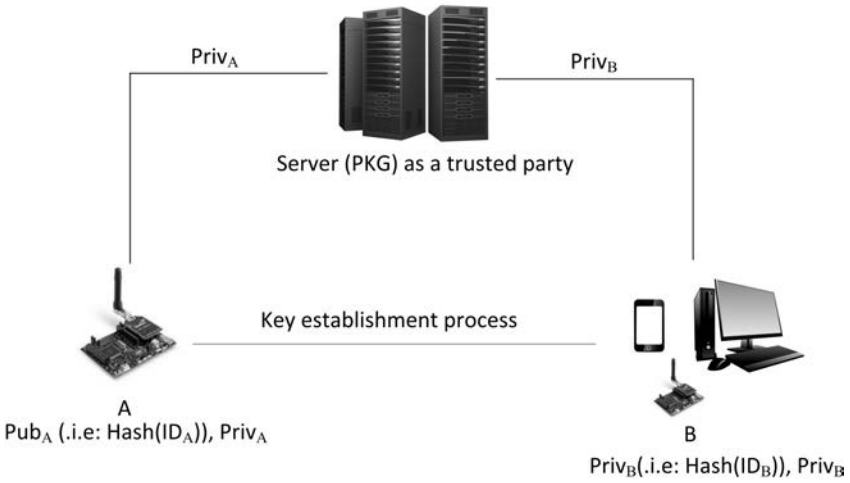


Figure 3.4: Identity-based Cryptography Infrastructure

- **Raw public key encryption**: Some mechanisms assume that the public key has been distributed beforehand or using out-of-band communications. These mechanisms offer small number of message exchanges but they are not scalable, because the public keys of all devices should be known by each device.

  Some "raw public key encryption" mechanisms, i.e. Rabin's scheme [159] or NtruEncrypt [88] have been recommended for WSNs. Rabin's scheme is very similar to the RSA algorithm (widely used cryptosystem), which is also based upon the hardness of the factorization problem. In fact, the scheme requires the same energy consumption for decryption operations than RSA with the same security level. However, it offers much faster mechanism for encryption operations because only one squaring is required to encrypt a message.

  NtruEncrypt is a cryptosystem which is known to be a lattice-based alternative to RSA and ECC (Elliptic Curve Cryptography) primitives. The mechanism is highly efficient and suitable for the most limited-resource devices such as smartcards and RFID tags. In [88], the authors give a comparison of the three PKC mechanisms proposed for constrained devices: the Rabin's scheme, NtruEncrypt and ECC. The results show that NtruEncrypt leads to the smallest average power consumption. Nevertheless, this cryptosystem often requires large-size messages, and might result in packet fragmentation at lower layers and many re-transmissions in the presence of communication errors. The protocols that are based on "raw public key encryption" require small number of exchanged messages. This is actually advantageous if the transmission power is the most important and limiting factor.

- **Certificate-based Encryption**: Certificate-based protocols are a popular choice to establish a secure communication between two entities over Internet. The trust relationship between the two entities is guaranteed by a well-known third party (CA) using the standard X.509 certificate that validates the identity of the entity as illustrated in Figure 3.3. Indeed, each sensor node possesses a certificate signed by the trusted CA. This latter can be loaded into the node before the deployment or can be directly acquired on request from a trusted party.

  TLS [60] has been recommended by many standards specified by IETF (Internet Engineering Task Force) for security services. However, it is mentioned in [123] and [162] that TLS is not a wise choice with respect to the security best practices in IoT. In fact, TLS runs normally in a reliable transport protocol like TCP which is unsuitable for constrained resource

devices, due to its congestion control algorithm. As a replacement for TLS in the tightly constrained environments, the DTLS [164] (Datagram Transport Layer Security) protocol has been proposed recently. It operates over the unreliable transport protocol i.e., UDP and provides the same high security levels as TLS.

The utilization of a certificate is basically expensive. To reduce the power consumption, both hardware and software related improvements have been considered by researchers:

- **Usage of cryptographic hardware accelerators**: The hardware accelerators are in charge of all cryptographic computations. Kothmayr et al. in [123] propose a method to implement DTLS using hardware assistance on sensor nodes. The solution assumes that each sensor is equipped with a TPM (Trusted Platform Module). A TPM is an embedded chip that offers secure generation of cryptographic keys and sealed storage as well as hardware support for cryptographic algorithms. The fully authenticated handshake can be performed between a sensor (equipped with TPM) and a subscriber (another sensor or external entity). Both sensor and subscriber transmit their X.509 certificate to initiate the authentication phase. These certificates are signed by a trusted CA and are included in a fully authenticated DTLS handshake. This solution not only has a high security level by establishing the trusted relationship with the assistance of an approved third party, but it also provides message integrity, confidentiality and authenticity with affordable energy, end-to-end latency and memory overhead as claimed by the authors.

  Nevertheless, the approach is expensive and complex with respect to deploying a hardware accelerator next to every sensor, especially for large number of sensors.

- **Optimization of existing protocols (software implementation)**: A security protocol employing the certificates is tailored to provide higher performance without affecting the robustness of the protocol. Raza et al. [162] propose a modification of DTLS using the 6LoWPAN compression mechanism [105]. The modified protocol reduces the size of some headers (i.e. the DTLS record header, the handshake header, the handshake message). These changes improve the performance of DTLS in terms of packet size, energy consumption, processing time and network response time. However, the proposed solution does not propose backward compatibility with the actual DTLS standard, in particular with respect to header compression.

  Hummen et al. [108] propose a design idea to effectively reduce the overhead of the DTLS handshake. Full handshake procedure requires 15 message exchanges, high dynamic storage capability (RAM) during the communication and long processing time for cryptographic tasks. In order to mitigate the full handshake inconvenience, the authors propose to delegate the handshake procedure to a rich-resource entity, e.g. the gateway or the device's owner. All certificate related tasks are performed in the rich-resource entity and only the session-state message is sent to the constrained device. The session can then be established using this message with no additional calculation. This modified DTLS can highly reduce communication overhead at the condition that the rich-resource server is trusted.

  Granjal et al. [93] present similar modifications to DTLS, but the DTLS handshake is mediated by the 6LoWPAN border router (6LBR). The 6LBR participates in the secure communication but is transparent to sensing devices and the Internet host. The border router intercepts and forwards packets at the transport-layer. From the point of view of the Internet host, it communicates with the 6LBR using traditional DTLS protocol where authentication is supported by ECC based certificate. On the other side, the 6LBR operates in the pre-shared key security mode for communicating to the constrained sensing devices. Moreover, the 6LBR authenticates the nodes with a mechanism inspired by Kerberos [122]. If the authentication is successful, a secret session key is generated to secure the

communication between the sensing devices and the 6LBR. Actually, it is used to encrypt the pre-master key in the ClientKeyExchange message that the Internet host sends to the 6LBR. When the pre-master secret key is computed successfully in the Internet host and the sensing device, end-to-end DTLS security is enabled. The proposed architecture delegates all the expensive operations (ECC computation, key agreement...) to the border router so that it offers better lifetime for sensing devices. Nevertheless, the 6LBR is considered to be a single point of failure. The IKE protocol [59] works usually jointly with IPsec to provide security associations (SAs) between two entities. This protocol has a variant where the mutual authentication is enabled using RSA-based certificates. Ray et al. [160] propose another variant for IKE that is based on ECC-based public key certificate for authentication and ECDH for key agreement instead of RSA and DH protocol. The proposal reduces the computation cost as it is mainly limited to the point multiplication operations and it requires smaller key size than RSA for the same level [100].

- **Identity-based encryption**: The first implementation of Identity-Based Cryptography was developed by Shamir in 1984 [177]. This type of cryptography defines a well-known string (identity) representing an individual or an organization, which is used as a public key. The private key of each entity is generated from its public key by a trusted party (Figure 3.4), named a Public Key Generator (PKG). This solution eliminates the need for certificates, which makes the solution advantageous especially for WSNs. Indeed, any sensor nodes can simply generate the public key of other nodes when needed to establish a secure communication using their identities. In addition, the revocation mechanism is supported by consulting the list of valid sensor identities. However, ID- based schemes are vulnerable to key-escrow attacks as the PKG knows the private keys of all nodes in the network. It can impersonate any node and consequently intercept all the traffic in the system. Therefore, the PKG is always considered as well protected and trusted by all network nodes.

  In a constrained environment, IBE paradigm is mostly implemented using the ECC primitive [197, 93]. Yang et al. [197] propose IBAKA – an IBE scheme inspired by Boneh et al. scheme [36]. However, they tailor the IBE method into an ECDH [100] key exchange in order to establish a session key. Their proposal still requires 2 bilinear pairings and 3 scalar point multiplications each time a secret key is bootstrapped.

  Szczechowiak et al. [185] propose TinyIBE – a very simple authenticated key distribution based on IBE for heterogeneous sensor networks. The scheme requires no pairing calculation. It is able to retrieve a session key for two nodes after only 2 message exchanges.

3.2.3.2 Key agreement based on asymmetric techniques

This sub-category is about key agreement protocols based on asymmetric primitives in the IoT. As mentioned in various research works, a key agreement protocol is the mechanism where two (or more) parties derive a shared secret and no other party can predetermine the secret value. Figure 3.5 illustrates the process of a typical asymmetric key agreement. Km is the secret generated after the agreement procedure. This symmetric key is then used to secure the communication.

The Diffie-Hellman (DH) protocol [163] and its variants are classical examples for asymmetric key agreement. However, DH protocols are considered expensive and unsuitable for the constrained nodes in particular, for class 0 and 1 according to the node classification in terms of resource capacity in lwig-terminology [37].

Some variants of the DH protocol are considered in constrained environments using ECC, i.e. ECDH. The ECDH cryptographic primitive offers smaller key size than RSA. Indeed, the

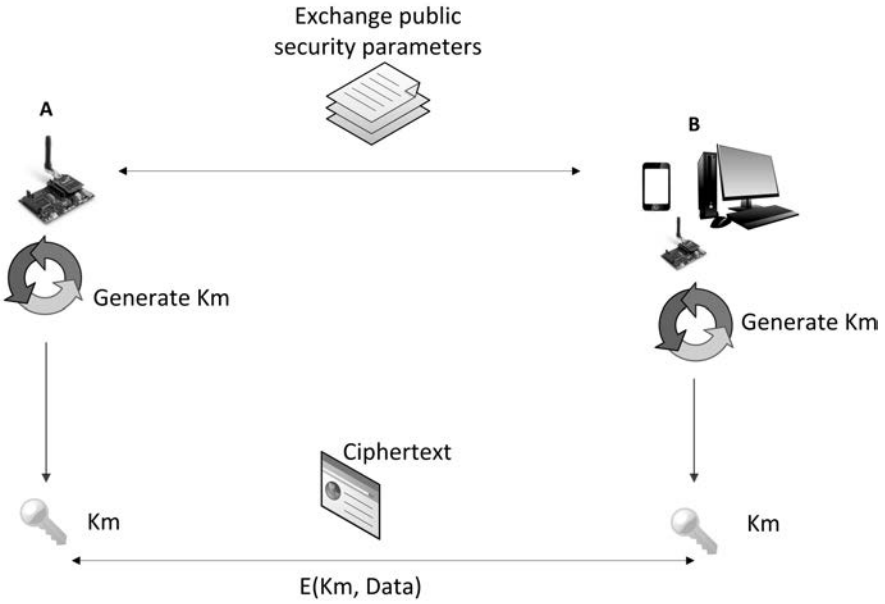


Figure 3.5: Key agreement based on asymmetric mechanisms

US National Institute for Standard and Technology (NIST) in [30] has showed that to achieve the security level of 128-bit AES key size, one can prefer 256 bit key size using elliptic curve instead of 3072 bit parameters in RSA and DH protocol. As an example, Giacomo et al. in [57] implemented a key agreement protocol based on ECDH providing authentication using the Elliptic Curve Digital Signature Algorithm (ECDH-ECDSA). Practical measurements on the MICAz and the TelosB sensors showed that ECDH-ECDSA is affordable in terms of computation complexity.

HIP-DEX (Host Identity Protocol Diet Exchange) [146] applies also the DH protocol to generate a session key between two entities after only a 4-messages exchange. This protocol is a variant of HIP Base Exchange [145] specially designed to reduce the complexity of cryptographic computations. It uses the smallest possible set of cryptographic primitives (e.g., AES-CBC instead of cryptographic hash functions), removes digital signatures and implements static ECDH to encrypt the session key, etc. This protocol has been largely taken into consideration in the context of IoT by many recent works [139, 145]. For instance, Mica et al. in [139] propose an efficient network access mechanism based on HIP-DEX for mobile nodes joining the local sensor network. Besides, Hummen et al. in [107] tailor HIP-DEX to the IoT, in particular, by adapting the session resumption mechanism as in TLS [75]. As such, the constrained node performs expensive operations once and maintains session-state for re-authentication and re-establishment of a secure channel.

The key agreement protocols based on DH require fewer messages to establish a session key but the computational tasks on sensor nodes are usually complex.

3.2.4 Symmetric key pre-distribution schemes

In this sub-category, the communicating parties often initially share some credentials before bootstrapping the communication. The key pre-distribution mechanisms may differ as described in the following sections.

3.2.4.1 Probabilistic key distribution

The mechanism of random key pre-distribution (RKP) was first proposed by Eschenauer et al [76]. A typical RKP consists of three phases: key pre-distribution, shared-key discovery and path-key establishment. In the scheme, a large key pool is generated. Keys are then randomly selected from the key pool and distributed to sensor nodes. Any two nodes may share a common key with a certain probability. The third phase is triggered when two nodes do not share any common key. In this process, one node first generates a random key K. It then sends the key to its neighbours using the pre-established secure channel. The process continues until the key K arrives at the other node. K is considered afterward as the pairwise key between both nodes.

Several solutions are inspired by this scheme [65, 46, 114, 110]. These proposals improve specially the pre-distribution phase to enhance the key connectivity between nodes and reduce the memory space needed for key storage. In fact, Du et .al [65] propose a key pre-distribution scheme that relies on the deployment knowledge and avoids unnecessary key assignments. Ito et al. [114] develop a scheme based on Du et al. [65] works but the keys are mapped on two-dimensional positions. They propose a probability density function which provides better key connectivity. Chan et al. [46] develop also a mechanism to reinforce the path-key establishment phase. The basic idea is that node A finds all possible links to a node B. It generates for each link a random value and routes these values to B. The common keys between A and B are protected by these random values. The generated key will be shared by both nodes, unless the adversary manages to eavesdrop on all paths between them.

The probabilistic key distribution generally does not guarantee session key establishment between all nodes even with the path-key establishment phase. Two nodes may not share any common keys with a certain probability.

3.2.4.2 Deterministic key distribution

In this sub-category, the described key schemes rely on a deterministic process to generate the key pool and to distribute keys to nodes in order to guarantee secure full connectivity in the network. In deterministic solutions, the key schemes are distinguished by the presence or not of a trusted third party during the key bootstrapping.

- **Offline key distribution**: The offline key distribution method is widely used in WSNs because of its simplicity. Depending on the used protocol, every node in the same network may share a network key or each two nodes may have a common pairwise key. The session key is then generated after very few data exchanges without the presence of any third party. The offline key distribution provides efficiency in terms of energy consumption because it does not require expensive cryptographic computations like asymmetric approaches. However, when a sensor node is physically attacked, the secret data stored inside the node can be exposed. Consequently, the attacker can gain access to several nodes which share the secret key with the attacked node, or in the worst case, it may access the whole network.

  In several existing works, mathematical properties have been applied to create the model for securing key exchanges between sensor nodes. These mechanisms are still applicable in the context of IoT. The most well-known schemes are based on bivariate polynomials ([79] and [133]). In these schemes, a node A shares with other nodes a bivariate n-degree polynomial $f(x, y)$. A can obtain the pairwise key with another node B by calculating the value of $f(id_A, id_B)$, where id_A and id_B are the respective identities of A and B. In the same way, B can obtain the same pairwise key, since $f(id_A, id_B)$ is equal to $f(id_B, id_A)$). In another scheme, called the Bloom's scheme [32], a secret symmetric matrix D is generated from the shared secret key between two nodes A and B. Each of them generates a public matrix I_A

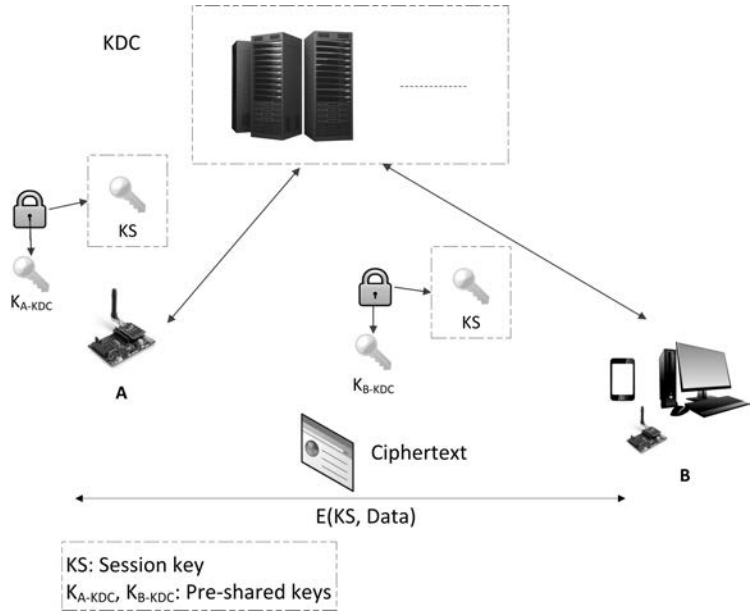


Figure 3.6: Server-assisted mechanism

and I_B respectively for A and B. The private keys are respectively $priv_A = D \times I_A$ and $priv_B = D \times I_B$ for A and B. Finally, the pairwise key is calculated by solving $(priv_A \times I_B)$ or $(priv_B \times I_A)$. The problem with these latter two schemes is that the session key will remain unchanged for every two nodes.

SNAKE [176] and BROSK [154] are two key establishment schemes where the session key is generated without the need for a key server to perform key management. These two protocols assume that all nodes in the same network share a master secret key. In SNAKE, the session key is obtained by hashing two random nonces generated from each communicating party using the pre-shared key. BROSK broadcasts the key negotiation message containing a nonce. Once a node receives the message from its neighbors, it can construct the session key by computing the message authentication code (MAC) of two nonces.

Raza et al. in [161] implement the standard Internet security protocol IPsec in an IP-based WSN (using 6LoWPAN). The authors propose mechanisms to compress the AH and ESP header in order to integrate IPsec with the 6LoWPAN layer but they keep a reasonable packet size. AH and ESP mechanisms provide origin authenticity, message integrity and confidentiality protection of IP packets but they do not handle the key exchange. The security associations are established manually using pre-shared key.

As another lightweight key pre-distribution scheme, Oscar et al. [84] propose HIMMO, a *quantum-resistant* solution that enables identity-based key sharing and verification of credentials of IoT devices in a single message. HIMMO allows a device to generate efficiently a pairwise key with another device based on its identity. They provide also a new operation mode, named as DTLS-HIMMO, for the standard protocol DTLS [164]. The authors claim that DTLS-HIMMO provides equivalent security properties when compared to DTLS-Certificate security suite, while being *quantum resistant* and lightweight as symmetric primitives.

The offline key distribution does not provide rekeying operations. When the system changes to other secret keys, all the entities in the network need to be updated to establish secure communications using the new keys.

- **Server-assisted key distribution**: Due to the resource limitations of constrained devices, the cryptographic computation and other expensive tasks (e.g., identity management, key generation) can be handled at rich–resource servers. Server-assisted approaches for key establishment protocols have been proposed in this respect in IoT. In such protocols, message exchanges engage two entities and one (or more) trusted server. The server shares long-term key a priori with each communicating entity. It often plays the role of a key distribution center (KDC) and then supplies the session key to each party by re-encrypting it using the shared keys as shown in Figure 3.6.

  1) **External assisted server**: In this sub-category, the assisted entities are external rich-resource servers which are located outside the WSN. As a result, they can handle the key distribution of one or several WSNs.

     The second approach proposes in [123] is inspired by the TLS Pre-Shared Key cipher suite [75]. Each sensor has to pre-install some random bytes called protokeys before the deployment. These random bytes are used to derive the PSK (Pre-Shared Key) key for each session. Instead of using TPM, a central rich-resource server is employed to create the security association between the sensor node and the subscriber. The protokeys are also known by the trusted server. The server then generates the same session key for the subscriber from the protokeys.

     MIKEY-Ticket [186] is an additional mode to the basic MIKEY [15] protocol, in which a KDC is involved in the process of establishing a security association between the two parties. MIKEY-Ticket originated from the ticket concept of Kerberos [122]. The KDC securely communicates with the node initiating the protocol (Initiator) and the responding node (Responder) by encrypting important data using the pre-shared master key shared with each node. Nevertheless, the protocol is vulnerable to Denial of Service (DoS) attacks, particularly replaying messages to the Responder. To prevent these attacks, Boudguiga et al. in [38] propose a new key establishment, called SAKE (Sever Assisted Key Establishment) based on the MIKEY-Ticket mode but removing the threat of DoS attacks. SAKE allows establishing security associations between the two parties after only five exchanged messages, compared to six messages in the original MIKEY-Ticket. Indeed, upon reception of the first message from the Initiator, the KDC generates the session key and contacts directly the Responder. This change reduces one message exchange comparing to MIKEY-ticket. Besides, as each message of SAKE contains a MAC computed with a key shared with the receiver, the DoS attack is mitigated.

     Other IoT solutions of key distribution based on an external server, include solutions that implement the PANA protocol (Protocol for Carrying for Network access) [150]. PANA runs over UDP and uses EAP (Extensible Authentication Protocol) for authentication that supports multiple authentication methods including pre-shared key distribution. Kanda et al. [116] propose an improvement of PANA to adapt the resource-constraints. The main modifications consist of reducing the number of message exchanges (e.g., choosing EAP-PSK as the only authentication method), removing unused PANA header fields, minimizing the collection of cryptographic primitives at the constrained device. These proposals may effectively reduce the PANA implementation code size at the device, but the authors do not give an estimation of the gains that might be obtained, for example, in terms of energy consumption or network-response time.

  2) **Proxy-based assisted server**: This sub-category does not require an external server but a proxy-based server (PBS) located within the WSN, as shown in Figure 3.7. This server is equipped with sufficient resources and storage capacity to execute all expensive tasks for constrained nodes. It often plays the role of a mediator to associate the sensor nodes

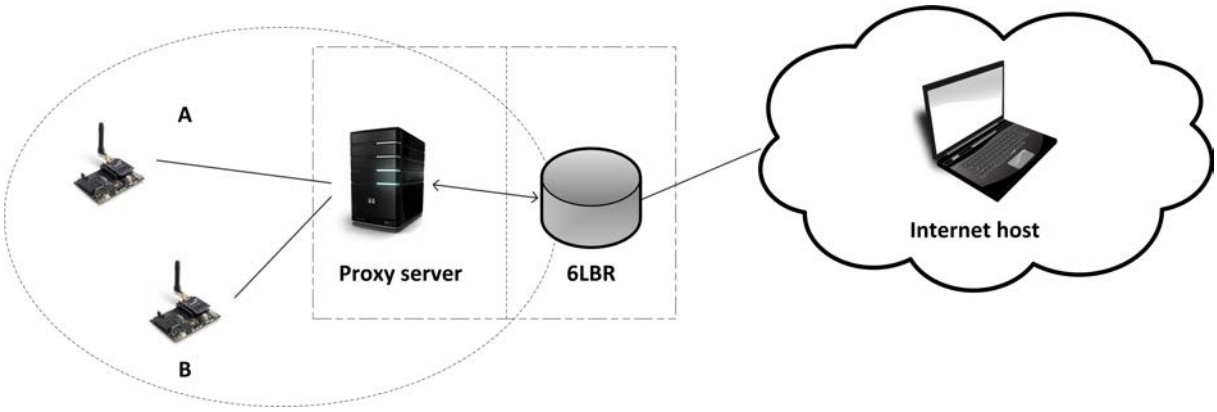


Figure 3.7: Proxy-based assisted server infrastructure

and other entities. Additionally, the PBSs usually share a symmetric secret key with the constrained nodes and the 6LBR router.

Using the same considerations, Hussen et al. [109] propose SAKES providing secure authentication and key establishment between a sensor node and an external Internet host. Upon the reception of a sensor node request, the PBS authenticates the sensor node with the help of 6LBR. It then applies a DH key agreement mechanism with the remote server and calculates the session key (SK) on behalf of the sensor node as the sensor node is resource constrained. Finally, the sensor node can communicate with the remote server in a secure manner using the SK received from the PBS.

In this same sub-category, Saied et al. [170] present the Distributed HIP Exchange (D-HIP) protocol inspired by HIP-BEX [145]. They use the same network model as described in Figure 3.7. During the key negotiation step, a constrained node establishes a session key with the server using the DH protocol by delegating the 2 modular exponentiation operations to the proxy nodes. It first splits its secret exponent a into n parts $a_1, a_2, ..., a_n$ where n is the number of the less constrained nodes. It then sends each part a_i to a neighbor node (proxies) PBSi. The node PBSi calculates its part of the final DH session key: SKi $= (g^b \, mod \, p)^{a_i}$ where the value $(g^b \, mod \, p)$ is achieved from the remote server (or Internet host). PBSi sends SKi back to the constrained node. From these values, the constrained node obtains the same final DH session key as the server (by multiplying the n values received). This approach has a major advantage that all expensive computation tasks are done by the PBS nodes. However, the number of message exchanges can be large depending on the number of PBS nodes. As we know, the transmission cost is non-negligible and packet lost during communication can happen at any time.

3.2.5 Discussion

Table 3.2 illustrates examples of security protocol solutions which are implemented in WSN and IoT. It compares these solutions using the identified criteria given in Section 3.2.1. At first glance, we can easily identify that most of the general security services are well provided by the proposed protocols. Nevertheless, few protocols support the Access control (AC) and Privacy Protection (PP) properties. The AC service is very important and needed in such perspective where an Internet host can only access the sensor node to execute actions or to retrieve data according to its access privileges. The server-based protocols usually offer this requirement, for example, with the help of an authorization server. On the other hand, the PP strengthens the

anonymity of communications. This property becomes very important in today perspective as personal data on sensor nodes must remain untraceable by any attackers.

					Confidentiality	Integrity	Authentication	Authorization	Freshness	Resilience	Computation Complexity	Communication Complexity	Memory	Scalability	Privacy Protection
Key bootstrapping in IoT	Asymmetric key schemes	Key transport based on public key encryption	RPKE	Protocols based on: NTRU [87], Rabin's scheme [159]	●	n/a	n/a	n/a	n/a	●	❍	❍	●	●	n/a
				Moustaine et al. [72]	n/a	●	●	❍	❍	●	●	●	●	●	●
				ZKP based on ECDLP[49]	n/a	●	●	❍	●	●	❍	●	●	●	●
			CBE	DTLS modified [162, 108]	●	●	●	❍	●	●	❍	❍	●	●	●
				IKEv2-ECC based [160]	●	●	●	❍	●	●	❍	❍	●	●	●
			IBS	TinyIBE[185]	●	❍	●	❍	❍	●	●	❍	●	❍	❍
				IBAKA[197]	●	●	●	❍	●	●	❍	❍	●	●	●
		Key agreement based on asymmetric techniques		ECDH-ECDSA[57]	●	●	●	❍	●	●	❍	●	●	●	❍
				HIP-DEX[146]	●	●	●	❍	●	●	●	●	●	●	❍
				E-G[76]	●	❍	❍	❍	❍	❍	●	❍	❍	❍	❍
		Probabilistic key distribution		Du et al. [65]	●	❍	❍	❍	●	❍	●	❍	❍	❍	❍
				Chan et al. [46]	●	❍	❍	❍	❍	❍	●	❍	❍	❍	❍
				Ito et al. [114]	●	❍	❍	❍	❍	❍	●	❍	❍	❍	❍
				Blom's scheme based [65, 33]	●	❍	❍	❍	❍	❍	●	●	●	❍	❍
	Symmetric key pre-distribution schemes	Deterministic key distribution	OKD	SNAKE[176]	●	●	●	❍	●	❍	●	●	●	❍	❍
				BROSK [125]	●	●	●	❍	●	❍	●	●	●	❍	❍
				Lightweight IPsec [161]	●	●	●	❍	●	❍	❍	❍	●	●	●
				Granjal et al. [93]	●	●	●	❍	●	●	❍	❍	●	●	●
				Diet-ESP [143]	●	●	●	❍	●	●	●	❍	n/a	❍	●
				Mikey-ticket[186]	●	●	●	●	●	❍	●	●	●	❍	❍
			EAS	SAKE[39]	●	●	●	❍	●	●	●	●	●	❍	❍
				PANA/EAP-PSK[150, 116]	●	●	●	❍	●	●	●	●	●	❍	❍
			PBAS	SAKES[109]	●	●	●	❍	●	●	❍	●	❍	❍	❍
				D-HIP[170]	●	●	●	❍	●	❍	❍	❍	❍	❍	❍

Table 3.2: Summary of proposed security solutions for IoT

Solutions are grouped based on the mentioned classification in Figure 3.2. Some abbreviations are used: (RPKE) – Raw public key encryption, (CBE) – Certificate based encryption, (IBS) – Identity based schemes, (OKD) – Offline key distribution, (EAS) – External server assisted, (PBAS) – Proxy-based assisted server. Eleven metrics are provided to evaluate the solutions: Confidentiality, Integrity, Authentication, Authorization, Freshness, Resilience, Computation complexity, Communication Complexity, Memory or storage space required for keying materials, Scalability and Privacy Protection. The Resilience, Computation complexity, Communication Complexity and Memory columns can take two different values: ●(good or medium performance level) and ❍(low performance level), which indicate the level of a specific protocol to support a property. Communication complexity refers to the number of message exchanges in general until a secret key is negotiated. The (n/a) notation means "not applicable". We define simple notations to evaluate the security services: ●- supported, ❍- not supported. The evaluation of RPKE assumes the protocols that used the mentioned primitives (no real protocol reference).

In the high level synthetic picture, the table shows that the asymmetric solutions usually require high computation complexity on sensor nodes. However, these approaches have high resilience against node capture attacks, low memory requirements for keying materials, few message exchanges and high scalability for large networks. On the other hand, the key pre-distribution schemes offer low complexity computation which is really beneficial for constrained nodes, but, they have their own inconveniences, such as high communication complexity, high memory space for keying materials, low level of scalability for large networks and vulnerability against node capture attacks.

3.3 Identified approaches towards lightweight security mechanisms

There are some new approaches being pushed by researchers. They always keep their interest in both asymmetric and symmetric approaches; even if the symmetric paradigm is considered to be more energy efficient. The asymmetric solutions are still preferable because of their deployment facility, flexibility and scalability in terms of key management. Besides, the public key paradigm allows two entities without any prior-trust relationship with each other, establishing a secure channel, which is generally an important feature in real time scenarios. The following points need to be highlighted before designing any efficient security protocols for constrained devices in IoT:

- **Optimizing asymmetric solutions**: The asymmetric approaches are generally energy-consuming. The first ambition is to reduce the required computation time in order to save energy for sensor nodes. One can think about adapting directly NTRU to the standard protocols because it is currently the most energy-efficient primitive. However, this primitive requires more memory space for keying materials than other asymmetric primitives. Some researchers are working on optimizing mathematical mechanisms used in cryptographic algorithms, i.e. Marin et al. in [136] discuss a solution to optimize the ECC primitives. They propose an optimization for the modular multiplication operation. The solution is evaluated in the widely-used microprocessor MSP430. The authors claimed that the optimization is presenting the lowest time and number of required operations for ECC multiplication. Another method to reduce the energy consumption on sensor nodes relies on pre-computation techniques. It helps diminishing the cost of modular exponentiations in several signature and key management schemes, such as ECDSA or Diffie-Hellman key exchange. The idea is to store a set of n Discrete Log pairs in the form $(a_i, g^{a_i} \, mod \, q)$. Then, a "random" pair $(r, g^r \, mod \, q)$ is generated from a subset of k pairs chosen randomly in the memory. The technique seems simple, but it requires the value of n to be sufficiently large in order to ensure the randomness of the generated pairs $(r, g^r \, mod \, q)$. Ateniese et al. [16] improve the pre-computation techniques above and apply it to ECDSA. They show that almost 50% of energy is saved with ECDSA with pre-computation compared to the original signature scheme and also to the NTRUsign signature scheme (which is considered to be a natural candidate in low-power devices).

  On the other hand, several researches adapt the properties of asymmetric primitives in an optimized manner to fit in the most constrained environment of IoT. Effectively, Moustaine et al. [72] propose an efficient authentication protocol for low-cost RFID systems based on an adaption of NTRU. This adaption first delegates the complex operations of NTRU (i.e. modular arithmetic, polynomial multiplication) to the server. Secondly, the tags require only additions and circular shifts to encrypt the challenges during the authentication phase. Besides, the protocol is resistant against classical attacks including replays,

tracking and man in the middle attacks with very low requirements for computation. As another asymmetric technique, Zero-knowledge proofs (ZKP) [49, 80] is also a candidate for future proposals in IoT. ZKP are interactive proof systems involving two entities: a prover and a verifier. The prover demonstrates the knowledge of a secret to the verifier without revealing a single bit about the secret. ZKP relies on some hard mathematical problems, such as the factorization of integers, i.e. [49] or the discrete logarithm problem (DLP) [80]. This mechanism is commonly used in WSN for node authentication. For example, the authors in [80] provide an efficient authentication scheme based on DLP over elliptic curve groups. The scheme requires only three messages between the prover and verifier. ZKP has advantages in terms of the amount of messages being sent and the memory usage on nodes as also mentioned in [49, 80]. One can benefit ZKP to propose an efficient key bootstrapping protocol in IoT with the node authentication provided by ZKP.

- **Tailoring the existing standard protocols to IoT**: Standard security protocols can be adapted to work in constrained and heterogeneous environments of IoT. As described in this chapter, many attempts have been done to adapt and apply standard protocols in the context of IoT, for example, DTLS [86, 162, 108], IPsec [161], IKEv2 [160], HIP-DEX [139, 146, 145]. As another example, Kivinen et al. [119] propose a minimum implementation of standard IKE [117] by removing the requirement for certificates. This minimum variant defines only two message exchanges for key negotiation and provides entity authentications using pre-shared key approach. On the other hand, Migault et al. [143] suppose that the security associations between entities are established using existing mechanism like IKEv2. They are interested in the security of packet transmissions by proposing Diet-ESP - an adaptation of ESP (Encapsulation Security Protocol) to IoT in order to compress and reduce the ESP overhead. The authors define mechanisms to remove or reduce some "unnecessary" or "larger than required" ESP fields for the specific needs or applications of IoT devices. However, the deployment of Diet-ESP has to keep the trade-off between the security requirements and the battery life time of constrained devices. Indeed, as depicted by the authors, small SPI (Security Parameters Index) size, small size of ICV (Integrity Check Value) and removing SN (Sequence Number) expose the devices to respectively Denial of Service, spoofing and replay attacks.

- **Using hybrid approaches**: Another trend consists of combining the advantages of both symmetric and asymmetric solutions. Meca et al. in [139] choose HIP-DEX (an asymmetric technique) [146] to provide access to a local sensor network. A mobile node is authenticated with help of a central server. If the authentication is successful, the server sends securely the necessary parameters for the mobile node by encrypting the data with the session key generated after the DH exchanges. These parameters are actually a bivariate polynomial used to bootstrap secure communications with a local node (a symmetric technique). The pairwise key generated by the shared polynomial is employed as a master key to generate multiple session keys for specific purposes. The presence of a third party in such hybrid approach becomes essential in the IoT. Firstly, the rich-resource server is expected to support almost all heavyweight computations. As such, the sensor nodes with limited energy and capabilities are no longer involved in this expensive process as described in [109, 170]. The constrained node can establish a communication with external hosts without implementing the full asymmetric process. Additionally, the assisted servers are capable to provide fine-grained access control such that only authorized actions are executed on sensor nodes.

3.4 Summary

In this chapter, we survey and analyze lightweight cryptographic primitives and secure communication protocols recently proposed in the literature for the Internet of Things. We present in the first place several lightweight cryptographic primitives that are suitable for resource-constrained devices. Then, we take a more in-depth look into the security protocols and methods that have been used to establish secure communications. Our analysis concentrates mainly on the key establishment procedure. We classify different protocols into different groups in order to analyze their forces and weaknesses. Finally, based on the obtained results, we identify some interesting and promising directions for lightweight security mechanisms in the context of IoT.

The rest of this thesis presents our contributions following the three identified directions. Concretely, by pursuing the first direction, we proposed in a first place ECKSS, an optimized signcryption scheme in chapter 4, which is very lightweight and exempted from the use of certificates. In addition, we introduced a modification to the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) in chapter 5 that can accelerate its encryption to be feasible even in resource-constrained devices. In the second direction, we adapted the standard key management protocol MIKEY to present two new MIKEY key distribution modes. As described in chapter 4, these modes use ECKSS as their public key encryption algorithm, and hence inherit the lightness of ECKSS in terms of computation overhead. In the third direction, we introduced a server-assisted key agreement protocol that can work with highly resource-limited devices. This proposal, as specified in chapter 6, allows the IoT devices with limited resources to agree on a session key without involving any expensive asymmetric operation.

Chapter 4

ECKSS: Elliptic Curve Korean Signature-Based Signcryption for IoT

In order to provide a lightweight security for constrained devices in IoT, one can think of using directly symmetric and hardware-based solutions. However, these methods have several inconveniences. Indeed, using only symmetric cryptography might put IoT devices on high risks even when only one among the symmetric secret keys is exposed. In addition, such approach should be considered uniquely in small scale networks. On the other hand, hardware-oriented solutions usually accelerate cryptographic operations and hence reduce computation charges on constrained devices. Nevertheless, it is difficult to attach a cryptographic accelerator or a trusted platform module (TPM) for each device in large scale IoT. In that respect, there is another approach which is to optimize existing lightweight cryptographic primitives in order to work with limited-resource devices. Such software-based approach is feasible to deploy in large-scale and heterogeneous networks such as IoT.

In this chapter, we present our first contribution. That is, in the first part, we propose ECKSS, a novel certificateless signcryption-based scheme in the elliptic curve setting [147]. This scheme has several advantages. First, the scheme is provably secure in the random oracle model. Second, it provides the following security properties: outsider/insider confidentiality and unforgeability; non-repudiation and public verifiability, while being efficient in terms of communication and computation costs. Third, the scheme offers the certificateless feature, so certificates are not needed to verify the user's public keys. For illustration, we conducted experimental evaluation based on an emulated sensor Wismote platform and compared the performance of the proposed scheme to concurrent signcryption schemes. In the second part, we apply our proposed signcryption scheme to propose novel lightweight ECC-based key distribution extensions for MIKEY [15]. The latter is a standard key management protocol, used to set up common secrets between any two parties for multiple scenarios of communications. To our knowledge, these extensions are the first ECC-based MIKEY extensions that employ signcryption schemes. Our proposed extensions benefit from the lightness of the signcryption scheme, while being discharged from the burden of the public key infrastructure (PKI) thanks to its certificateless feature. To demonstrate their performance, we implemented our proposed extensions in the Openmote sensor platform and conducted a thorough performance assessment by measuring the energy consumption and execution time of each operation in the key establishment procedure. The experimental results prove that our new MIKEY extensions are perfectly suited for resource-constrained devices.

This chapter is organized as follows. Section 4.1 recalls the two standard DSA-based signature schemes ECDSA and EC-KCDSA. Our proposed signcryption scheme is inspired from these signature schemes. As a result, the descriptions are provided in order to justify the lightness of our solution. Section 4.2 describes in detail our proposed signcryption mechanism. A formal security proof and an experimental evaluation of the scheme are also described. Then, in Section 4.3, we apply the proposed signcryption solution to propose two new modes for the standard key distribution MIKEY, while presenting also the security considerations and a thorough experimental evaluation of the two proposed extensions. Finally, we conclude our first contribution in Section 4.4.

4.1 Background on DSA variants

As the standardization is a crucial factor in practical cryptosystems, multiple signcryption schemes are derived from standard signatures, such as DSA [180, 190], ECDSA [189, 179], KCDSA [167, 128, 199]. The Digital Signature Algorithm (DSA) [118] is a well-known standard for digital signatures. Its elliptic curve analog ECDSA, is presented in [118]. Our proposed signcryption scheme is based on EC-KCDSA (ECC-based KCDSA). We present in the following the original ECDSA and EC-KCDSA signature schemes with the objectives of clarifying the difference between the two schemes and justifying our choice.

4.1.1 Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA was standardized by the National Institute of Standard and Technology (NIST) and then adopted by FIP 186-4 in 2013 [118]. In order to generate and verify an ECDSA signature, the signer (S) and verifier (V) must agree on the domain parameters $D = (q, a, b, G, p, h)$. The ECDSA key pair of S is generated as follows:

- Select a random number $priv_S$ on $\mathbb{Z}_p$ as the private key.
- Compute the public key $PK_S = priv_S.G$

The public key validation is highly recommended in practical scenario since it helps to prevent malicious insertion of an invalid public key, which can enable some attacks, such as the attacks on the Diffie-Hellman Key Agreement discovered by Lim et Lee [130]. The common method is to demand assurances from a trusted party (e.g. Certification Authority) for each used public key.

Then, the detailed description of ECDSA signature generation and verification is presented in the following:

- **Signature generation**: To sign a message M, the signer S does the following steps:
  1. Select randomly an integer k in $\mathbb{Z}_p$
  2. Calculate the elliptic curve point $(x_1, y_1) = k.G$
  3. Compute $r = x_1 \bmod p$. If $r = 0$ then go to step 1
  4. Compute $HASH(M)$ and convert this string to an integer e, where $HASH$ is a cryptographic hash function, such as SHA-2
  5. Compute $s = k^{-1}.(e + priv_S.r) \bmod p$. If $s = 0$ then go to step 1
  6. Send (r, s) to V as S's signature for the message M

- **Signature Verification**: To verify S's signature (r, s) on M, V first validates the domain parameters D and S's public key PK_S. Then, V does the following:

  1. Verify that (r, s) are integer in $\mathbb{Z}_p$
  2. Compute $HASH(M)$ and convert this string to an integer e
  3. Compute $w = s^{-1} \bmod p$
  4. Compute $u_1 = ew \bmod p$ and $u_2 = rw \bmod p$
  5. Compute $W = u_1.G + u_2.PK_S = (x_1, y_1)$
  6. The signature is valid if $r = x_1 \bmod p$

4.1.2 Elliptic Curve Korean Certificate-based Digital Signature Algorithm

We describe in this subsection the ECC-based version of KCDSA [131] since our proposal is based on Elliptic Curve Cryptography. This variant was called as EC-KCDSA, which is almost similar to KCDSA except the change of group operations. In fact, the underlying multiplicative group used in KCDSA is replaced by the additive group of elliptic curve points.

The detailed signature generation and verification of EC-KCDSA is presented in the following:

- **Security parameters generation**: Let $\mathbb{E}$ be an elliptic curve over prime finite field defined by the domain parameters (q, a, b, G, p, h). Suppose that a signer S wish to send a signature on the message M to a verifier V . The security keys of S are specified in the following:

  - $priv_S$: signer's private key such that $priv_S \in \mathbb{Z}_p$
  - PK_S: signer's public key such that $PK_S = {priv_S}^{-1}.G$
  - z: a hash value of $Cert\_Data$, i.e. $z = H_2(Cert\_Data)$. Here $Cert\_Data$ denotes the signer's certification data, which should contain at least signer's identity, the public key PK_S and the domain parameters (q, a, b, G, p, h).

  We define also two hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^l$, where l is an integer generated from the input security level.

- **Signature generation**: The signer generates a signature (r, s) for a message M as follows:

  - Choose randomly an integer number $k \in \mathbb{Z}_p$
  - Compute $W = k.G$
  - Compute the first part r of the signature as $r = H_1(W)$
  - Compute $e = r \oplus H_1(z||M)$
  - Compute the second part s of the signature as $s = priv_S.(k - e)$

- **Signature Verification**: On receiving the signature (M, r, s), the verifier can check the validity of the signature as follows:

  - Check the validity of signer's certificate
  - Exact the signer's certification data $Cert\_Data$ and compute $z = H_2(Cert\_Data)$
  - Compute $e = r \oplus H_1(z||M)$
  - Compute $W = s.PK_S + e.G$
  - Check that $r = H_1(W)$

As we shall see, EC-KCDSA is even more lightweight than ECDSA since it is exempted from two modular inversions in the signature generation and verification.

4.2 Our proposed signcryption scheme

The rest of this section is organized as follows. Section 4.2.1 presents the considered architecture and threat model for a signcryption scheme. Section 4.2.2 introduces related existing schemes on signcryption. Section 4.2.3 describes in detail our signcryption scheme. We present the basic security results of our scheme and its proofs in section 4.2.4. In section 4.2.5, we discuss the provided features of our proposal and propose a small modification to transform our scheme to be insider secure. The performance assessment of our proposal is given in section 4.2.6.

4.2.1 System and Threat model for a signcryption scheme

In this section, we introduce the considered system architecture and threat model for a signcryption scheme.

4.2.1.1 System model

The considered scenario throughout this document contains the following actors:

- An Initiator (I) and a Responder (R), which respectively initiates the communication and responds to incoming requests.
- A trusted Key Management Server (KMS), which is responsible for generating keying materials and that acts as the root of trust of the responder and initiator.

The KMS is in charge of providing key material for all communicating devices. In this document, it is considered that there is only one KMS. Applications may use multiple or distributed KMSs and hence may need different system parameters (general parameters, public/private keys). The mechanism for deciding which system parameters to use (when more than one KMS is available) is out of scope of this chapter.

The KMS first selects a secret value mk as the system secret master key. The KMS's public key PK_{KMS} is then generated from mk. This public key is the root of trust for both parties. The KMS then provides key material for each device in the system. The idea of key construction is inspired from works in [96]. For an entity I, it defines a public validation token (V_I) to validate the relation between the secret key of each device and PK_{KMS}. Our approach uses V_I to cryptographically bind I's public key to PK_{KMS}, instead of having a pair of public/private keys and a certificate. V_I does not require any further explicit certification. KMS also attributes a short unambiguous identifier for each device. A device identification must be unique and can be renewed along with its key material by the KMS. Note that the transfer of key parameters to each device must be secure.

4.2.1.2 Threat models for signcryption Schemes

This section presents the security models for two security notions of signcryption schemes: confidentiality against chosen-ciphertext attacks (CCA), which is also known as semantic security, and the unforgeability against chosen-message attacks (CMA). We consider a multi-user setting as already studied in [18, 14]. Concisely, there exist many other users in addition to the attacked Initiator (I) and Responder (R). The attacker can be either an insider or outsider that acts by replacing the initiator/responder public keys at will when accessing the signcryption/unsigncryption oracles. In the outsider setting, an attack is perpetrated by a third party which is different from I and R. On the other hand, an attack in the insider setting is issued from an internal party, meaning that the attacker is a compromised I or R. In such model, the owner of a private key is unable to retrieve any information on a ciphertext previously signcrypted

by himself without knowing the randomness used to produce that ciphertext. Thereafter, this chapter refers to confidentiality as the confidentiality against CCA in the outsider model, and it refers to unforgeability as the unforgeability against CMA in the insider model.

Definition 11 (SC-IND-CCA2 [18])**.** Let $\mathcal{A}$ be a PPT adversary against the confidentiality of a signcryption scheme between the (fixed) initiator I, and the (fixed) responder R, with security parameter k. $\mathcal{A}$ has negligible advantage to win the following game, denoted as $EXPT_{\mathcal{A}}^{SC-IND-CCA2}(k)$:

- The challenger runs the algorithms Setup and KeyGen to generate keying material for I and R. ($priv_I$, $priv_R$) are kept secret while (pk_I, pk_R) are given to $\mathcal{A}$.
- $\mathcal{A}$ can make calls to the signcryption and unsigncryption oracles. On each signcryption query, $\mathcal{A}$ produces a pair (m, pk_B) at will where pk_B is an arbitrary receiver's public key (that public key may differ from pk_R) and m is the message. On receiving this pair, the signcryption oracle $\mathcal{O}_{SC}$ returns the result of Signcrypt$(priv_I, pk_I, pk_B, m)$ to $\mathcal{A}$. On each unsigncryption query, $\mathcal{A}$ produces a pair (pk_A, C) at will where pk_A is an arbitrary initiator's public key and C is a signcryptext. On receiving this pair, the unsigncryption oracle $\mathcal{O}_{USC}$ returns the result of Unsigncrypt$(pk_A, priv_R, pk_R, C)$ to $\mathcal{A}$.
- $\mathcal{A}$ outputs a pair of messages of equal length (m_0, m_1). On receiving this pair, the challenger selects a bit $b \xleftarrow{\$} \{0,1\}$ and sends the challenge ciphertext C_{RS} = Signcrypt $(priv_I, pk_I, pk_R, m_b)$ to $\mathcal{A}$.
- $\mathcal{A}$ submits a number of queries to $\mathcal{O}_{SC}$ and $\mathcal{O}_{USC}$ as $\mathcal{A}$ did in previous steps. However, it is not allowed to query O_{USC} on (pk_I, C_{RS}). Note that $\mathcal{A}$ can query $\mathcal{O}_{USC}$ on (pk_A, C_{RS}) for any $pk_A \neq pk_I$ and query $\mathcal{O}_{USC}$ on (pk_I, C) for any $C \neq C_{RS}$.
- At the end of the game, $\mathcal{A}$ outputs b' and wins the game if $b' = b$.

$\mathcal{A}$'s advantage is defined to be $Adv_{\mathcal{A}}^{SC-IND-CCA2}(k) = 2Pr[b' = b] - 1$.

Definition 12 (SC-UF-CMA [18])**.** Let $\mathcal{A}$ be a PPT adversary against the unforgeability of a signcryption scheme with security parameter k. $\mathcal{A}$ has negligible advantage to win the following game, denoted as $EXPT_{\mathcal{A}}^{SC-UF-CMA}$:

- The challenger runs the algorithms Setup and KeyGen to generate a pair of public/private keys $(priv_I, pk_I)$ for the initiator I.
- $\mathcal{A}$ can make calls to $\mathcal{O}_{SC}$, but not to $\mathcal{O}_{USC}$, because it can generate by itself a pair of receiver's private/public keys. On each signcryption query, $\mathcal{A}$ produces a pair (m, pk_B) at will where pk_B is an arbitrary receiver's public key and m is the message. On receiving this pair, $\mathcal{O}_{SC}$ returns the result of Signcrypt$(priv_I, pk_I, pk_B, m)$ to $\mathcal{A}$.
- At the end of the game, $\mathcal{A}$ outputs a pair of receiver's private/public keys $(priv_R, pk_R)$ and a signcrypted text C_{RS}. We say that $\mathcal{A}$ wins the game if the following conditions are satisfied: (i) C_{RS} is a valid signcryptext from S to R (this means that the unsigncryption process is done under the initiator's public key pk_I and the receiver's private key $priv_R$); (ii) $\mathcal{A}$ did not query on (m_{RS}, pk_R) to $\mathcal{O}_{SC}$, where m_{RS} is the plaintext of the signcryptext C_{RS}.

4.2.2 Existing Signcryption Schemes

We are mainly interested in the signcryption schemes based on the Diffie-Hellman problem in this thesis. As surveyed in [58], there exist several schemes based on different security assumptions, such as: Bilinear Maps [22] and RSA problem [59]. Most of the signcryption schemes are derived from popular signature schemes (refer to the survey in [58]). For examples, Zheng's scheme [201] is based on ElGamal encryption and signature [71], which is computationally efficient, but requires complex interactive zero-knowledge proof to validate the non-repudiation and does not provide insider confidentiality. Bao et al. [19] modify Zheng's proposal to provide the public verifiability property without the need for the recipient's private key. However, the previous scheme is not semantically secure, as written by Shin et al. [180]. They claim their new signcryption proposal based on DSA (Digital Signature Algorithm) [81], namely SCDSA+, to be confidentially and unforgeably secure, without giving a formal proof. There exist also several schemes issued from the standardized signature algorithm ECDSA [99, 190]. Both schemes provide desirable security properties but still result in poorer performance than our schemes. Certificateless signcryption schemes remove the use of certificates. However, they usually require costly pairing operations for public key validation [20]. Some similar proposals are successful to remove pairing operations in their construction [196, 175]. However, they still require 10 and 12 modular exponentiations.

KCDSA (Korean Certificate-based Digital Signature Algorithm) [132] is a variant of DSA, whose design allows to relieve the signature generation and verification procedures of modular inversions required in DSA. Two signcryption variants of KCDSA are first proposed by Yum et al. [199]. However, their security has not been formally proved by the authors. Besides, the first variant is confidentially insecure in the insider model. The second one is not semantically secure due to the disclosure on the hash of the message, in addition to being more expensive in terms of performance comparing to our proposals (one extra exponentiation). Several works on identity-based signcryption scheme based on KCDSA exist, such as [128, 167]. Though, these schemes require 3 costly pairing operations, which is not practical for constrained nodes in the IoT.

4.2.3 The certificateless elliptic curve Korean signature-based signcryption ECKSS

In this subsection, we present a lightweight signcryption scheme based on the standardized signature KCDSA [131]. We name our proposed scheme as ECKSS. We start by describing the security parameter generation process. Then, we introduce our proposed signcryption scheme. Finally, we show that our scheme is exempted from certification requirements.

4.2.3.1 Security parameter generation process

Depending on the security parameter as input, KMS first runs the Setup algorithm to define an elliptic curve $\mathbb{E}$ over finite field $\mathbb{F}_q$ with a generator G, and p is the prime order. Further guidance on the selection of recommended domain parameters for elliptic curve cryptography can be found in [81].

Two hash functions are also defined: $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^l$, where l is an integer number defined from the input security level. (Enc, Dec) are the encryption and decryption algorithms of a symmetric cipher. Then, KMS executes the KeyGen algorithm to generate the keying material for I and R. From a chosen master key mk, KMS calculates its public key as $PK_{KMS} = mk.G$. For an entity A with the identifier id_A, KMS generates its public and private values as follows:

- Compute $V_A = x_A.G$, where x_A is a random number on $\mathbb{Z}_p$
- Compute the private key for A: $priv_A = (mk + x_A.H_1(id_A||V_A||G||PK_{KMS}))^{-1}$
- Compute the public key $PK_A = priv_A^{-1}.G$
- Set the public parameters of A as $Pub_A = (PK_A, V_A)$

4.2.3.2 ECKSS description

The detailed procedure of ECKSS is described in the following:

- $\mathsf{Signcrypt}(priv_I, PK_I, PK_R, M) \rightarrow CT$: To signcrypt a message M intended to R, I executes the following steps:
  1. Check the validity of R's public keys, as described in section 4.2.3.3
  2. Choose randomly $x \leftarrow \mathbb{Z}_p$
  3. Compute $K = x.PK_R$
  4. Generate a secret key: $\tau = H_2(PK_I||PK_R||K)$
  5. Compute $r = H_1(PK_I||PK_R||K||M)$
  6. Compute $s = priv_I.(x - r)$
  7. Compute $c = \mathsf{Enc}_\tau(M)$
  8. Send the ciphertext $CT = (r, s, c)$ to R
- $\mathsf{Unsigncrypt}(priv_R, PK_R, PK_I, CT) \rightarrow M$: Upon receiving the ciphertext $CT = (r, s, c)$ from I, R has to perform the following procedure:
  1. Check the validity of I's public keys, as described in section 4.2.3.3
  2. Compute $W = s.PK_I + r.G$
  3. Compute $K = priv_R^{-1}.W$
  4. Get the secret keys: $\tau = H_2(PK_I||PK_R||K)$
  5. Compute $\mathsf{Dec}_\tau(c) = M$
  6. Verify that $r = H_1(PK_I||PK_R||K||M)$

In a real implementation, we can combine the steps 2 and 3 of the Unsigncrypt algorithm into only one step. As such, we compute directly $K = (spriv_R^{-1}).PK_I + (rpriv_R^{-1}).G$. As a result, the Unsigncryption procedure will require only two exponetiations.

4.2.3.3 Public Key Validation

This section describes the algorithm to be executed in the first step of signcryption and unsigncryption phases. Concretely, it explains the process of validating the public pair (PK_I, V_I) for any entity I. To validate these public values, the used algorithm requires the identification of I, namely ID_I and the KMS public key PK_{KMS}. The following checks must be passed successfully:

- Validate that PK_I and V_I lie in the same defined elliptic curve E.
- Compute $H_1(ID_I||V_I||G||PK_{KMS})$, as an integer number on $\mathbb{Z}_p$.
- Validate the public key of I using the following equation:

$$PK_I = PK_{KMS} + H_1(ID_I||V_I||G||PK_{KMS}).V_I \tag{4.1}$$

The algorithm above can be only executed at the first run of the protocol. I and R may save the trusted public parameters of the other party for future uses. Besides, the revocation of I's public values can be checked easily if the identifier ID_I is correctly generated. For instance, the identifier format can include a timestamp in order to automatically enable the expiration of key material.

4.2.4 Game-based security proofs

In this section, we give a formal security analysis of our proposal. Our analysis is inspired from works conducted in [18] and [200]. First, we define several security notions needed for the proof. Then, we prove that the confidentiality and unforgeability of ECKSS are tightly related to the hardness of the GDH and GDL problems.

4.2.4.1 Notations for the security proof

The security proof requires complex interactions between the oracles. Hence, we use two lists L_0 and L_1 to keep track of queries to and responses from the hash, signcryption and unsigncryption oracles. Precisely, L_0 contains the values of type $(PK_A,\ PK_B,\ W,\ K,\ \tau) \in \mathbb{G}^2 \times \mathbb{G}_\star^2 \times \{0,1\}^l$. Likewise, L_1 contains the values of type $(m,\ PK_A,\ PK_B,\ W,\ K,\ r) \in \{0,1\}^* \times \mathbb{G}^2 \times \mathbb{G}_\star^2 \times \mathbb{Z}_p$. For any set $\mathcal{X}$, we define $\mathcal{X}_\star = \mathcal{X} \cup \{\star\}$, where the symbol $\star$ denotes the parameter that can not be calculated by the simulation. We define $\mathcal{O}$ to be a DDH oracle that is able to determine whether or not the tuple $(a.P, b.P, c.P)$ satisfies $ab \equiv c \,(\mathrm{mod}\, p)$. We index records in the list L_i by the set I_{L_i} $(i = 0, 1)$. The symbol ε defines an empty string. The symbol $\cdot$ specifies a parameter that "matches" any values. That is, if there exists $(x, y, \cdot) = (u, v, w)$ then we have $x = u$ and $y = v$. For any variable X calculated by a simulator, X^* is also a simulated value but its value is the same as the value calculated by the real oracles. We additionally consider that q_i (for $i = 0, 1$), q_{SC}, and q_{USC} are the maximum number of queries made to H_i, signcryption and unsigncryption oracles, respectively.

4.2.4.2 Confidentiality of our scheme

Theorem 13. *In the random oracle model, given a PPT adversary $\mathcal{A}$ against the SC-IND-CCA2 security of the ECKSS signcryption scheme, there exists a PPT adversary $\mathcal{B}_1$ against the GDH problem and a PPT adversary $\mathcal{B}_2$ against the OT-IND property of the symmetric encryption scheme such that:*
$Adv_{\mathcal{A}}^{SC-IND-CCA2}(k) \leq 2Adv_{\mathcal{B}_1}^{GDH}(k) + Adv_{\mathcal{B}_2}^{OT-IND}(k) + \frac{2q_{SC}(q_1+q_{SC}+q_{USC})}{2^{l_p(k)}} + \frac{2(q_{SC}+q_{USC})}{2^{l_p(k)}}$.

Proof. We will prove the theorem via a sequence of games [181]. We denote S_i to be the event that $\mathcal{A}$ outputs the bit b' in game G_i and $b' = b$.
Game $\mathbf{G_0}$: This is the original attack game $EXPT_{\mathcal{A}}^{SC-IND-CCA2}(k)$ defined in Definition 11. Hence,

$$Pr[S_0] = \tfrac{1}{2} + \tfrac{1}{2}Adv_{\mathcal{A}}^{SC-IND-CCA2}(k)$$

Game $\mathbf{G_1}$: This game replaces two random oracles H_0, H_1 by two random oracle simulators $\mathsf{H_0Sim}$ and $\mathsf{H_1Sim}$. We maintain the simulation of oracles by storing historical queries and responses into the two lists L_0 and L_1. We first define rules on how to determine membership in the list L_0 and L_1, as described in Figure 4.1. Based on these rules, we simulate $\mathsf{H_0Sim}$ and $\mathsf{H_1Sim}$ as denoted in Figure 4.2.

We observe that the simulation of H_0 and H_1 is modeled as random oracles and the consistency among hash queries is ensured by the lists L_0 and L_1. Besides, we assume in this game

$L_0Rule(PK_A, PK_B, K, W)$:
If $(PK_A, PK_B, \cdot, K, \cdot) = (PK_{Ai}, PK_{Bi}, W_i, K_i, \tau_i)$ or $(PK_A, PK_B, W, \cdot, \cdot) = (PK_{Ai}, PK_{Bi}, W_i, K_i, \tau_i)$, $i \in I_{L_0}$ then $\tau \xleftarrow{\$} \tau_i$
else if there exists $(PK_{Ai}, PK_{Bi}, W_i, K_i, \tau_i) \in L_0$ and $\mathcal{O}(W, PK_B, K_i) = 1$ or $\mathcal{O}(W_i, PK_B, K) = 1$ then $\tau \xleftarrow{\$} \tau_i$
else $\tau \leftarrow \perp$
return τ

$L_1Rule(m, PK_A, PK_B, K, W)$:
If $(m, PK_A, PK_B, \cdot, K, \cdot) = (m, PK_{Ai}, PK_{Bi}, W_i, K_i, r_i)$ or $(m, PK_A, PK_B, W, \cdot, \cdot) = (m, PK_{Ai}, PK_{Bi}, W_i, K_i, r_i)$, $i \in I_{L_1}$ then $r \xleftarrow{\$} r_i$
else if there exists $(m, PK_{Ai}, PK_{Bi}, W_i, K_i, \tau_i)$, $i \in I_{L_1}$ and $\mathcal{O}(W_i, PK_B, K) = 1$ or $\mathcal{O}(W, PK_B, K_i) = 1$ then $r \xleftarrow{\$} r_i$
else $r \leftarrow \perp$
return r

Figure 4.1: Functions which determine membership in the list L_0 and L_1 from partial information

$\mathsf{H_0Sim}(PK_A, PK_B, K)$:
$\tau \leftarrow L_0Rule(PK_A, PK_B, K, null)$
If $\tau = \perp$ then $\tau \xleftarrow{\$} \{0,1\}^l$; $Add(PK_A; PK_B, \star, K, \tau)$ to L_0.
return τ
$\mathsf{H_1Sim}(m, PK_A, PK_B, K)$:
$r \leftarrow L_1Rule(m, PK_A, PK_B, K, null)$
If $r = \perp$ then $r \xleftarrow{\$} \mathbb{Z}_p$; $Add(m, PK_A; PK_B, \star, K, r)$ to L_1.
return r

Figure 4.2: Random Oracle Simulators $\mathsf{H_0Sim}$ and $\mathsf{H_1Sim}$

that the signcryption and unsigncryption oracles are perfect. As a result, we have that Game 1 is equivalent to Game 0. Thus,

$$Pr[S_1] = Pr[S_0]$$

Game $\mathbf{G_2}$: In this game, we replace the signcryption oracle by the signcryption oracle simulator SCSim as described in Figure 4.3. This simulator does not require the initiator's private key $priv_I$ to generate a signcryptext. Since s, r are uniformly chosen at random in $\mathbb{Z}_p$ and W is computed as $W = [s]PK_S + [r]G$, W is therefore uniformly distributed in $\mathbb{G}$. As a result, as long as $\perp_{SC}$ does not occur, we have that Game 1 and Game 2 are equivalent. Note that the size of L_i is bounded by $(q_i + q_{SC} + q_{USC})$ for $i \in \{0, 1\}$. Thus, the probability that $\perp_{SC}$ happens is bounded by $(q_1 + q_{SC} + q_{USC})/2^{l_p(k)}$ and there are at most q_{SC} executions. Hence, we have:

$$|Pr[S_2] - Pr[S_1]| \leq q_{SC}\left(\frac{q_1 + q_{SC} + q_{USC}}{2^{l_p(k)}}\right)$$

Game $\mathbf{G_3}$: This game replaces the unsigncryption oracle by the simulator USCSim described in Figure 4.4, in order not to use the receiver's private key $priv_R$. We observe that Game 3 is identical to Game 2 except when the hash oracles are queried at $K^* = [priv_R]^{-1}([s]PK_S + [r]G)$. We consider this situation in three cases:

- H_0 is queried on (PK_S, PK_R, K^*) or H_1 is queried on (m, PK_S, PK_R, K^*) by the adversary $\mathcal{A}$. This means that $\mathcal{A}$ can recover K^*. As a result, this leads to an algorithm

$\mathsf{SCSim}(PK_A, (PK_B, m))$:
$s; r \xleftarrow{\$} \mathbb{Z}_p$, $W = [s]PK_S + [r]G$;
$\tau \leftarrow L_0Rule(PK_A, PK_B, null, W)$
If $\tau = \perp$ then $\tau \xleftarrow{\$} \{0,1\}^l$; Add $(PK_A, PK_B, W, \star, \tau)$ to L_0
$c \leftarrow Enc_\tau(m)$, $r' = L_1Rule(m, PK_A, PK_B, null, W)$
If $r' \neq \perp$ then return $\perp$ and halt all operations (event $\perp_{SC}$)
else Add $(m, PK_A, PK_B, W, \star, r)$ to L_1; $C \leftarrow (r, s, c)$
return C

Figure 4.3: Signcryption Oracle Simulator SCSim

$\mathsf{USCSim}(PK_B, (C, PK_A))$:
Parse C as (r, s, c)
$W = [s]PK_A + [r]G$; $\tau \leftarrow L_0Rule(PK_A, PK_B, null, W)$
If $\tau = \perp$ then $\tau \xleftarrow{\$} \{0,1\}^l$; Add $(PK_A, PK_B, W, \star, \tau)$ to L_0
$m = Dec_\tau(c)$; $r' = L_1Rule(m, PK_A, PK_B, null, W)$
If $r' = \perp$ then $r' \xleftarrow{\$} \mathbb{Z}_p$; Add $(m, PK_A, PK_B, W, \star, r')$ to L_1 if $r \neq r'$ then return $\perp$ else return m

Figure 4.4: Unsigncryption Oracle Simulator USCSim

$\mathcal{B}_1$ that can solve the GDH problem, because the adversary can verify the fact that $\mathcal{O}(PK_R, W, K^*) = 1$.

- The signcryption oracle could attempt to make such queries. However, this requires that the value of W must be equal to W^*. Since r, s are uniformly chosen at random in $\mathbb{Z}_p$, the probability that this event occurs, is bounded by the probability $q_{SC}/2^{l_p(k)}$.

- The unsigncryption oracle could attempt to make such queries. In such situation, the adversary must have made a query to $\mathcal{O}_{USC}$ on (c, r, s) such that: $[s]PK_S + [r]G = [s^*]PK_S + [r^*]G$ (1). If $(s, r) = (s^*, r^*)$ then we must have $c \neq c^*$, because $\mathcal{A}$ is not allowed to query exactly to $\mathcal{O}_{USC}$ on the signcryptext obtained from the signcryption oracle. We must also have $\tau = \tau^*$. Since the symmetric encryption scheme is one-to-one, we obtain that $(m = Dec_\tau(c)) \neq (m_b = Dec_{\tau^*}(c^*))$. As a result, this equation must hold $H_1(PK_S||PK_R||K^*||m) = H_1(PK_S||PK_R||K^*||m_b)$. However, as H_1 is modeled as a random oracle, the equation is true only with probability of $1/2^{l_p(k)}$. We then change the unsigncryption oracle so that it answers $\perp$ when queried on (c, r^*, s^*). The probability that it outputs incorrectly is bounded by $q_{SC}/2^{l_p(k)}$. On the other hand, if $(s, r) \neq (s^*, r^*)$, we show that the GDH problem can be solved. In fact, from (1), we obtain that $[(s - s^*)]PK_S = [(r^* - r)]G$. We can deduce that ${priv_S}^{-1} = (r^* - r)/(s - s^*)$. Hence, one can compute $[ab]G = [priv_S^{-1}]PK_R = [(r^* - r)/(s - s^*)]PK_R$. At this stage, $\mathcal{A}$ can verify the accuracy of $[ab]G$ by using the DDH oracle $\mathcal{O}$.

Consequently, we have:

$$|Pr[S_3] - Pr[S_2]| \leq \frac{q_S + q_{USC}}{2^{l_p(k)}} + Adv_{\mathcal{B}_1}^{GDH}$$

In G_3, τ^* is not used anywhere except when computing the challenge ciphertext c^*. Hence, if $\mathcal{A}$ outputs $b' = b$, then there exists an algorithm $\mathcal{B}_2$ that can break the OT-IND property of

the symmetric encryption scheme. Thus, $Pr[S_3] = \frac{1}{2} + \frac{1}{2}Adv_{\mathcal{B}_2}^{OT-IND}(k)$. Summarizing all the obtained bounds together, we have:

$$Adv_{\mathcal{A}}^{SC-IND-CCA2}(k) = 2|Pr[S_0] - \tfrac{1}{2}| \leq \frac{2q_{SC}(q_1+q_{SC}+q_{USC})}{2^{l_p(k)}} + \frac{2(q_{SC}+q_{USC})}{2^{l_p(k)}} + 2Adv_{\mathcal{B}_1}^{GDH}(k) + Adv_{\mathcal{B}_2}^{OT-IND}(k)$$

□

4.2.4.3 Unforgeability of our scheme

Theorem 14. *In the random oracle model, given a PPT adversary $\mathcal{A}$ against the SC-UF-CMA property of the proposed signcryption scheme, there exists a PPT algorithm $\mathcal{B}$ against the GDL problem such that:* $Adv_{\mathcal{A}}^{SC-UF-CMA}(k) \leq \sqrt{q_{\mathcal{R}} \cdot Adv_{\mathcal{B}}^{GDL}(k)} + \frac{q_{SC}(q_1+q_{SC})+q_{\mathcal{R}}+1}{2^{l_p(k)}}$.

We prove the theorem using two lemmas. First, we show that if there exists an attacker $\mathcal{A}$ against the SC-UF-CMA property, we can construct an efficient algorithm $\mathcal{B}'$ that solves the GDL' problem which is defined below. Then, we prove that any efficient algorithm $\mathcal{B}'$ can be transformed to an efficient algorithm $\mathcal{B}$ that solves the GDL problem, thus contradicting with the hardness assumption of GDL in section 2.2.2.

Definition 15 (GDL' problem)**.** Given $(G, n, p, [a]P)$, where $(G, n, p) \xleftarrow{\$} \mathsf{Setup}(k)$ and $a \xleftarrow{\$} \mathbb{Z}_p$, we define an oracle $\mathcal{R}$ as follows: for i=1..$q_{\mathcal{R}}$, on input $(PK_i, K_i) \in \mathbb{G}^2$, return $r_i \xleftarrow{\$} \mathbb{Z}_p$, where $q_{\mathcal{R}}$ is the maximum number of queries made to $\mathcal{R}$. The GDL' problem is to compute s^* and $i^* \in \{1..q_{\mathcal{R}}\}$ such that: $K_{i^*} = [as^* + r_{i^*}]PK_{i^*}$.

We first reduce the hardness of SC-UF-CMA property to the hardness of the GDL' problem as follows:

Lemma 16. If there exists a PPT adversary $\mathcal{A}$ against the SC-UF-CMA property, then there exists a PPT adversary $\mathcal{B}'$ against the GDL' problem, such that: $Adv_{\mathcal{A}}^{SC-UF-CMA}(k) \leq Adv_{\mathcal{B}'}^{GDL'}(k) + \frac{q_{SC}(q_1+q_{SC})+1}{2^{l_p(k)}}$.

Proof. We will prove the lemma via a sequence of games [181]. At the end of each game, $\mathcal{A}$ outputs a tuple consisting of $(priv_R^*, PK_R^*, C^*)$. Let Verify be the algorithm that verifies the two conditions listed in Definition 12. We denote S_i is the event in game G_i that Verify outputs 1.
Game $\mathbf{G_0}$: This is the original attack game $EXPT_{\mathcal{A}}^{SC-UF-CMA}$ in Definition 12. Hence,

$$Pr[S_0] = Adv_{\mathcal{A}}^{SC-UF-CMA}(k)$$

Game $\mathbf{G_1}$: This game replaces the random oracles H_0 and H_1 by the simulated oracles $\mathsf{H_0Sim}$ and $\mathsf{H_1Sim}$. $\mathsf{H_0Sim}$ remains unaltered as described in Figure 4.2, while $\mathsf{H_1Sim}$ is modified as described in Figure 4.5. The lists L_0 and L_1 are still employed to store historical queries on simulated oracles. The rules for determining membership of these lists remain unchanged. As we shall see, $\mathsf{H_1Sim}$ makes call to the oracle $\mathcal{R}$ defined in the GDL' problem. Note that $\mathcal{R}$ behaves differently from a random oracle, because it always returns random values even for repeated queries. Besides, we introduce the list $L_{\mathcal{R}}$ that contains the values of type $(PK_B, K, r, j) \in \mathbb{G}^2 \times \mathbb{Z}_p \times \mathbb{Z}$. The above simulation for the random oracle H_0 and H_1 is perfect. Hence, we have

$$Pr[S_1] = Pr[S_0]$$

Game $\mathbf{G_2}$: This game replaces the signcryption oracle by the simulated oracle simulator SCSim described in Figure 4.3. This simulator does not require the initiator's private key $priv_I$ in the signcryption stage.

$\mathsf{H_1Sim}(m, PK_A, PK_B, K)$:
$r \leftarrow L_1Rule(m, PK_A, PK_B, K, null)$
If $r = \perp$ then $j \leftarrow j + 1$; $r \xleftarrow{\$} \mathcal{R}(PK_B, K)$; Add (PK_B, K, r, j) to $L_{\mathcal{R}}$; $Add(m, PK_A; PK_B, \star, K, r)$ to L_1.
return r

Figure 4.5: Random Oracle Simulators $\mathsf{H_1Sim}$ in game G_1

Since (s, r, W) are independent and uniformly distributed over $\mathbb{Z}_p^2 \times \mathbb{G}$, the views of attacker in Game G_1 and Game G_2 are equivalent, as long as the event $\perp_{SC}$ does not happen. The size of L_i is bounded by $(q_i + q_{SC})$ for $i \in 0, 1$. Thus, the probability that $\perp_{SC}$ happens is bounded by $(q_1 + q_{SC})/2^{l_p(k)}$. There are maximum of q_{SC} queries on the signcryption oracle. Hence, we have

$$|Pr[S_2] - Pr[S_1]| \leq q_{SC}\left(\frac{q_1+q_{SC}}{2^{l_p(k)}}\right)$$

Now, we consider the event AskKey that $\mathsf{H_1Sim}$ has been queried on (m^*, PK_S, PK_R, K^*). Note that if AskKey does not occur, then the value r returned by $\mathsf{H_1Sim}$ is uniformly generated from $\mathbb{Z}_p$. If C^* is a valid signcryptext then $H_1(m^*, PK_S, PK_R, K^*)$ must not have been defined by the signcryption oracle. Thus, the probability that $r = H_1(m^*, PK_S, PK_R, K^*)$ is at most $1/2^{l_p(k)}$. As a result, we obtain that $Pr[S_2|\neg\mathsf{AskKey}] \leq 1/2^{l_p(k)}$ and consequently $Pr[S_2] \leq Pr[\mathsf{AskKey}] + 1/2^{l_p(k)}$.

On the other hand, we show that if AskKey occurs, then there exists an algorithm $\mathcal{B}'$ against the GDL' problem. Indeed, $\mathcal{B}'$ is given inputs (G, n, p, PK_S) and runs $\mathcal{A}$ on this input. If AskKey occurs, then $\mathcal{A}$ must return (PK_R, r^*, s^*, c^*) such that $\mathsf{H_1Sim}$ is queried on (m^*, PK_S, PK_R, K^*). Since (m^*, PK_R) has never been queried to SCSim, $\mathcal{R}$ must be queried on (PK_R, K^*) by $\mathsf{H_1Sim}$ and return r^*. Thus, there will exist an entry $(PK_R, K^*, r^*, j) \in L_{\mathcal{R}}$, where $1 \leq j \leq q_1$. As a result, (s^*, j) is a valid solution for the GDL' problem. Therefore, we have $Pr[\mathsf{AskKey}] \leq Adv_{\mathcal{B}'}^{GDL'}(k)$. In conclusion, we achieve the following reduction:

$$Adv_{\mathcal{A}}^{SC-UF-CMA}(k) \leq Adv_{\mathcal{B}'}^{GDL'}(k) + \frac{q_{SC}(q_1+q_{SC})+1}{2^{l_p(k)}}$$

□

In the following, we will apply the general forking lemma defined by Bellare et al. in [26] to reduce GDL' to GDL. This approach is also used by Zhang et al. [200] in their proof. We recall the general forking lemma as follows:

Lemma 17 (General forking lemma [26])**.** Fixing an integer $q_{\mathcal{R}} \geq 1$ and a set Z of size $h = 2^{l_p(k)} \geq 2$. Let $\mathcal{V}$ be a randomized algorithm that on input $(cp, r_1, r_2, ..., r_{q_R})$ returns a pair (J, σ) consisting of an integer $0 \leq J \leq q_{\mathcal{R}}$ and a side output σ. Let IG be a randomized algorithm that we call input generator. The accepting probability of $\mathcal{V}$, denoted as acc, is defined as the probability that $J \geq 1$ in the experiment: $cp \xleftarrow{\$} IG; r_1, r_2, ..., r_{q_R} \xleftarrow{\$} Z; (J, \sigma) \xleftarrow{\$} V(cp, r_1, r_2, ..., r_{q_R})$. The forking algorithm associated to $\mathcal{V}$ is defined as follows:
$F_{\mathcal{V}}(cp)$:
Pick coins ρ for $\mathcal{V}$ at random $r_1, ..., r_{q_R} \xleftarrow{\$} Z$; $(I, \sigma) \leftarrow \mathcal{V}(cp, r_1, ..., r_{q_{\mathcal{R}}}; \rho)$; $If\ I = 0$ return $(0, \varepsilon, \varepsilon)$; $r'_1, ..., r'_{q_R} \xleftarrow{\$} Z$; $(I', \sigma') \leftarrow \mathcal{V}(cp, r_1, ..., r_{I-1}, r_I, ..., r_{q_{\mathcal{R}}}; \rho)$
If $I = I'$ and $r_I \neq r'_I$ return $(1, \sigma, \sigma')$ else return $(0, \varepsilon, \varepsilon)$
Let

$$frk = Pr[b = 1 : cp \xleftarrow{\$} IG; (b, \sigma, \sigma') \xleftarrow{\$} F_{\mathcal{V}}(cp)]$$

Then $frk \geq acc \cdot (\frac{acc}{q_{\mathcal{R}}} - \frac{1}{h})$ and alternatively

$$acc \leq \frac{q_{\mathcal{R}}}{h} + \sqrt{q_{\mathcal{R}} \cdot frk}$$

Lemma 18. If there exists a PPT adversary $\mathcal{B}'$ against the GDL' problem, then there exists a PPT adversary $\mathcal{B}$ against the GDL problem such that: $Adv_{\mathcal{B}'}^{GDL'}(k) \leq \frac{q_{\mathcal{R}}}{h} + \sqrt{q_{\mathcal{R}} \cdot Adv_{\mathcal{B}}^{GDL}(k)}$

Proof. We will use the general forking lemma in this proof. As defined in the proof of Lemma 16, $\mathcal{B}'$ is the algorithm that can solve the GDL' problem. It takes as input (G, n, p, PK_S) where $a = priv_S^{-1}$, and returns (j^*, s^*, r^*) or $\perp$. We denote an algorithm $\mathcal{V}$ that runs $\mathcal{B}'$ as a subroutine. It takes as input $(G, n, p, PK_S, r_1, ..., r_{q_{\mathcal{R}}})$. It outputs values of type (j, σ) or $(0, \varepsilon, \varepsilon)$, where σ is a tuple of the form $(s, r) \in \mathbb{Z}_p^2$. The forking algorithm $F_{\mathcal{V}}$ is built as in Lemma 17. We define an algorithm $\mathcal{B}$ that runs $F_{\mathcal{V}}$ as a subroutine. If $F_{\mathcal{V}}$ returns $(1, \sigma, \sigma')$, such that: $\sigma = (s^*, r^*)$ and $\sigma' = (s^{*\prime}, r^{*\prime})$, we have $K^* = K^{*\prime}$ and $PK_{i^*} = PK_{i^{*\prime}}$ (because $j^* = j^{*\prime}$). As a result, the following equation holds: $[priv_S^{-1} \cdot s^* + r_{j^*}]PK_{i^*} = [priv_S^{-1} \cdot s^{*\prime} + r_{j^{*\prime}}]PK_{i^{*\prime}}$. Since $r_{j^*} \neq r_{j^{*\prime}}$ as defined in the forking algorithm $F_{\mathcal{V}}$, we can extract the initiator's private key as follows: $priv_I = (s^* - s^{*\prime})/(r_{j^{*\prime}} - r_{j^*})$. Then $\mathcal{B}$ outputs $priv_I$ as a solution for an instance of the GDL problem. As we can see, $\mathcal{V}$ outputs essentially what $\mathcal{B}'$ outputs. It is obvious that the accepting probability acc is equal to the success probability of $\mathcal{B}'$, $Adv_{\mathcal{B}'}^{GDL'}(k)$. Similarly, $\mathcal{B}$ outputs identically as $F_{\mathcal{V}}$, so that $Adv_{\mathcal{B}}^{GDL}(k) = frk$. Hence, by the general forking lemma, we have:

$$Adv_{\mathcal{B}'}^{GDL'}(k) \leq \frac{q_{\mathcal{R}}}{h} + \sqrt{q_{\mathcal{R}} \cdot Adv_{\mathcal{B}}^{GDL}(k)}$$

□

4.2.5 Provided security features and extension

We have formally proved in section 4.2.4 that our scheme ECKSS is confidentially secure in the outsider model and unforgeably secure in the insider model. In [201], the authors suggest that a signcryption scheme should also support the "public verifiability" and "non-repudiation" features. We claim that ECKSS provides these properties.

- **Public verifiability**: To prove to a trusted third party that the initiator I actually signed the plaintext m, R can forward the following tuple $(PK_S, PK_R, m, K, r, s, c)$. The third party can verify the signcryptext by executing the steps belows:
  - Compute $\tau = H_0(PK_S||PK_R||K)$
  - Verify if $m = Dec_\tau(c)$.
  - Verify if $r = H_1(PK_S||PK_R||K||m)$

  The knowledge on K does not leak any secrecy on the private key of either I or R, as long as the DLP assumption remains unbreakable for any resource-bounded attackers.

- **Non-repudiation**: The non-repudiation is a direct result of the unforgeability feature. The initiator usually can not deny the authority of the signcryptext when executing the above *public verifiability* process, if the ciphertext is really issued by him. However, if the aforementioned process passes successfully, then duplicating valid signcryptext is possible, which is inconsistent to the unforgeability feature.

It is possible to add the property of insider confidentiality to the previous proposed scheme with the cost of an extra point multiplication. This property was also considered in [18, 14, 200] and called "forward security" in several existing works [187, 190, 70]. Indeed, instead of returning (r, s, c), Signcrypt returns (Q, s, c), where $Q = r.G$. Similarly, Unsigncrypt verifies the

validity of Q instead of r, as follows: $Q \stackrel{?}{=} H_1(PK_S||PK_R||K||m).G$. As we can see, it is now computationally infeasible for a bounded resource adversary to recover messages of previous sessions even under exposure of the private key of the initiator due to the DLP assumption. We name the resulting scheme as ECKSS+.

4.2.6 ECKSS Performance Evaluation

This section first quantifies the performance of our proposed schemes and then estimates their energy consumption versus other related schemes on an emulated sensor platform.

4.2.6.1 Performance comparison

Table 4.1 illustrates the efficiency and supported security features of our schemes and multiple signcryption proposals in related work. The table shows if the scheme supports certificateless property. Then, the efficiency of each scheme is evaluated with respect to: communication and computational costs. The communication costs are evaluated as the packet length of signcryptext in bits. While, the computational costs are evaluated in terms of the number of expensive operations needed for the signcryption and unsigncryption processes. Finally, the table summarizes the supported security properties for each scheme.

As shown in Table 4.1, our proposed schemes not only support desirable security features, but also offer the best performance in terms of computational cost. Indeed, ECKSS requires only 1 point multiplications (PM) for signcryption, 2 PMs and one point addition for unsigncryption. ECKSS+ requires one more point multiplication in both the signcryption and unsigncryption process. When compared to the other elliptic curve based schemes ([70, 187, 190, 198, 99]), ECKSS+ needs the least number of costly operations and also generates the shortest signcrytext in bits.

Scheme	CL	Efficiency											Supported features					
		Computational cost																
		Communication cost	Signcryption					Unsigncryption										
			PM	PA	I	e	EXP	PM	PA	I	e	EXP	UF	OCF	NR	PV	ICF	StS
Zheng [201]	❍	$2\|p\| + \|m\|$	0	0	1	0	1	0	0	0	0	3	●	●	●	●	❍	n/a
SCDSA+ [180]	❍	$2\|p\| + \|m\|$	0	0	2	0	2	0	0	1	0	3	●	●	●	●	❍	DSA
Bao et al. [19]	❍	$2\|p\| + \|m\|$	0	0	1	0	2	0	0	0	0	3	❍	●	●	●	❍	n/a
Yum et al. [199]	❍	$2\|p\| + \|m\|$	0	0	0	0	1	0	0	0	0	3	❍	●	●	●	❍	KCDSA
Selvi et al. [175]	●	$2\|p\| + \|m\|$	0	0	0	0	5	0	0	0	0	7	●	●	●	❍	●	n/a
S-ECSC [202]	❍	$2\|p\| + \|m\|$	1	0	0	0	0	3	1	0	0	0	●	●	●	●	❍	n/a
ECGSC [99]	❍	$\|G\| + \|p\| + \|m\|$	2	0	1	0	0	3	1	1	0	0	●	●	●	●	●	ECDSA
NCLSC [198]	●	$3\|G\| + \|m\|$	3	1	0	0	0	2	2	0	2	0	●	●	●	●	●	n/a
Tso et al. [190]	❍	$\|G\| + \|p\| + \|m\|$	3	0	1	0	0	4	1	3	0	0	●	●	●	●	●	ECDSA
Toorani et al. [187]	❍	$\|G\| + \|p\| + \|m\|$	2	0	0	0	0	4	2	0	0	0	●	●	●	●	●	n/a
Dutta et al. [70]	❍	$\|G\| + \|p\| + \|m\|$	3	0	1	0	0	5	2	0	0	0	●	●	●	●	●	n/a
ECKSS	●	$2\|p\| + \|m\|$	1	0	0	0	0	2	1	0	0	0	●	●	●	●	❍	KCDSA
ECKSS+	●	$\|G\| + p\| + \|m\|$	2	0	0	0	0	2	1	0	0	0	●	●	●	●	●	KCDSA

Table 4.1: Performance comparison between our scheme and related work

Meaning of abbreviations: CL: Certificateless or Public key Verification without a trusted third party, PM: Point multiplication, PA: Point addition, EXP: Modular exponentiation, I: Modular inversion, e: Pairing operation, UF: Unforgeability, OCF: Outsider Confidentiality, NF: Non-repudiation, PV: Public verifiability, ICF: Insider Confidentiality or Forward secrecy, StS: Standard signature. We define simple symbols to evaluate the security services: ●- supported, ❍- not supported. The n/a notation means "not applicable". $|Y|$ denotes the length of Y in bits.

Parameters	Strength	Size	PM	PA	Inversion	Pairing
secg_p160	80	160	2460ms/16.25mJ	7ms/0.03mJ	298ms/1.90mJ	3533ms/23.32mJ
nist_p192	96	192	3463ms/22.53mJ	8ms/0.04mJ	403ms/2.67mJ	6586ms/43.47mJ
nist_p224	112	224	4782ms/32.05mJ	10ms/0.07mJ	577ms/3.81mJ	9573ms/63.19mJ
nist_p256	128	256	18.91s/124.07mJ	31ms/0.21mJ	1870ms/12.36mJ	36,16s/238.13mJ

Table 4.2: Energy consumption and time execution of atomic operations on Wismote [184]

4.2.6.2 Estimation of energy consumption on emulated sensor platform

In this subsection, we provide details on the implementation of our performance assessment. Thereafter, we report the performance and energy consumption results of our scheme compared with related work.

Experimental tools and platforms: We have implemented our assessment program in C for the operating system Contiki 2.7 [67]. Based on the Relic library [8] version 0.3.5, we evaluate some cryptographic operations on the four elliptic curves secg_p160, nist_p192, nist_p224 and nist_p256. Their domain parameters have been recommended by SECG [9] and NIST [81]. In addition, we opted for the emulated sensor node Wismote to evaluate the required operations on Cooja [74] - a Java-based simulator designed for the Contiki operating system. Wismote [184] is a low power wireless sensor module featured with 16 MHz MSP430x micro-controller, 16 kB of RAM, 128 kB of ROM and an IEEE 802.15.4 radio interface. This platform supports 20 bit addressing and sufficient RAM and ROM capacities. Such features are necessary for using a cryptographic library along with an application on top of it.

Performance: In order to assess the energy consumption, we employ a software-based online energy estimation mechanism described in [68]. In their model, the total energy consumption can be evaluated by the following formula: $E = U * (I_m t_m + I_l t_l + I_t t_t + I_r t_r + \sum I_{c_i} t_{c_i})$, where U is the supply voltage, I_i and t_i $(i = m, l, t, r)$ are the current draw and the time duration of the microprocessor in active mode, low power mode, transmit mode and receive mode respectively. I_{c_i} and t_{c_i} are the current draw and the time duration of the microprocessor for handling other components, such as sensors and LEDs.

In our scenario, we consider only the first four factors. The value of U is typically 3V, as with two new AA batteries. Furthermore, the current draw of the sensor node in each mode is extracted from its data sheet. As an example, the Wismote platform consumes I=2.2mA when in active mode. The time t_i that the component is in mode i, is measured by Powertrace. The latter is a pre-loaded tool in the Contiki OS, which provides an accuracy up to 94% of the energy consumption of a device [66].

Table 5.3 shows the execution time and energy cost of ECC operations over the Wismote platform. We consider only the most expensive operations: point multiplication(PM), point addition(PA), modular inversion and pairing operation. Each operation is evaluated in the four mentioned elliptic curves in increasing order of security level. Pairing-based calculation is, as expected, the most expensive operation. Point multiplication is also an expensive task. That is, even for the smallest security level of 80 bits, it requires up to 2.5s to compute and consumes 16.25mJ. In addition, we observe that for an elliptic curve with length of 256 bits of field order, the energy cost for point multiplications and pairing operations becomes huge, since for a single execution, they consume more than 124mJ and 239mJ, respectively. Besides, they are also time-consuming (18.91s for a PM and 36.16s for a pairing).

Gathering the measurement results in Table 4.1 and 5.3, we estimate the total energy consumption of our proposed signcryption schemes and five other ECC-based signcryption schemes. As depicted in Figure 4.6, our proposals ECKSS and ECKSS+ are the most efficient schemes. The ECGSC [99] scheme has a slightly higher computational cost in comparison with ours. However, it requires certificates to validate the public keys. This constraint could be very costly for a sensor node, since the verification of certificates is usually complicated and consuming in energy. Indeed, when using the nist_9256, ECKSS+ saves more than 17%, 43%, 58% and 60% of the overall energy consumption in comparison with the schemes of Dutta et al. [70], Tso et al. [190], Toorani et al. [187] and NCLSC [198], respectively. ECKSS is even more efficient than ECKSS+ and therefore can be applied on resource-constrained devices.

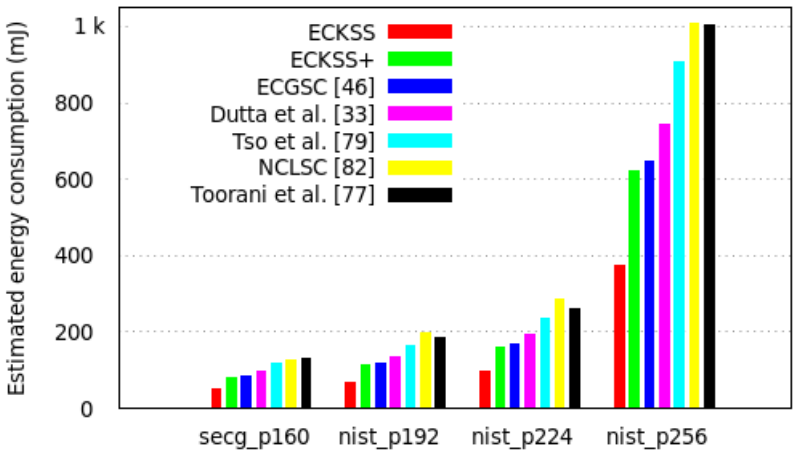


Figure 4.6: Total estimated energy consumption of our schemes and related work

4.3 ECKSS Application to MIKEY

Multimedia Internet KEYing (MIKEY) [15] is a key management protocol which is intended for use with real-time applications. MIKEY provides different methods to establish a session key with multiple parties, in addition to the authentication of parties if required. For example, MIKEY pre-shared key method permits any two parties with a pre-shared secret to set up a secure communication. However, this mechanism suffers from scalability issues since it is unpractical to pre-distribute a common key for any two parties in large networks, e.g. the Internet of Things (IoT). To be scalable, public key encryption-based methods, where any two parties can establish security communications without any *a priori* shared common keys, have been proposed to be employed by MIKEY.

These different key distribution mechanisms can be classified into two categories: (i) a key exchange mode and (ii) a key transport mode. The MIKEY key exchange modes, such as, MIKEY-DHSIGN [15], MIKEY-DHHMAC [77], are usually based on the Diffie-Hellman (DH) key exchange [142]. These modes provide the perfect forward secrecy property, i.e. the compromise of long-term keying materials does not reveal previously derived session keys. Additionally, both communicating parties participate in the session key generation process. As a result, DH-based modes require at least two message exchanges to set up a common secret key. As another disadvantage, these modes do not support the establishment of group keys. In key transport modes, on the other hand, the initiating party is responsible for the key generation. The generated keys are then encrypted using the public key of the responding party. Even if key transport modes do not provide perfect forward secrecy, they are more efficient in terms of computation and communication than DH-based modes. Indeed, only a half roundtrip is needed to set up a common key between two parties. Existing key transport modes of MIKEY generally employ a public key encryption algorithm to protect transferred keys, such as RSA [112] or ECIES [83] and an additionally public key signature algorithm to sign MIKEY messages.

In this section, we propose to use more lightweight key transport modes built upon our proposed signcryption scheme ECKSS defined in the previous section. ECKSS is based on Elliptic Curve Cryptography (ECC), thus inheriting multiple advantages of ECC in terms of performance. As mentioned in [83], ECC-based schemes require smaller key sizes and offer better security per bit, when compared with known cryptographic algorithms like RSA. Moreover, ECKSS offers the certificateless feature that allows to dispense the two parties with the provision

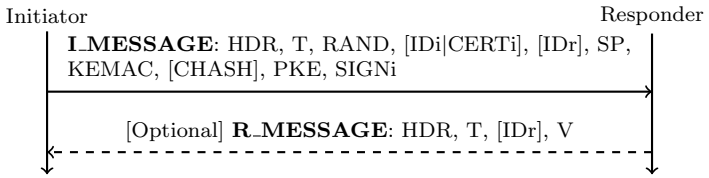


Figure 4.7: Basic message format for a MIKEY public key encryption method

of a digital certificate issued by a Public Key Infrastructure (PKI).

The remainder of this section is organized as follows. We first present the payload and data type formats of a MIKEY key transport mechanism in section 4.3.1. Then, in section 4.3.2, we clarify our design goals for proposing new key distribution methods for MIKEY. Section 4.3.3 and 4.3.4 give details on these extensions in respect to the original MIKEY payload formats. Finally, Section 4.3.6 presents the experimental results of the new proposed MIKEY modes.

4.3.1 Introduction to MIKEY modes and extensions

Figure 4.7 describes the basic message composition of a MIKEY key transport method that uses a public-key encryption algorithm, for example, in the MIKEY-RSA [15] and MIKEY-ECIES [83] modes. The mechanisms contain two message exchanges: the I_MESSAGE and the R_MESSAGE. The main objective of the Initiator's message is to distribute one or more TGKs and a set of security parameters in a secure manner. We recall the payload notions as defined in [15], in the following:

- HDR: The MIKEY header, which contains related data and information mapping to the specific security protocol used.
- T: The timestamp, used to prevent replay attacks.
- RAND: The random byte-string, which is used as a value for the freshness of the session key generation process.
- IDx: The identity of the entity X (IDi: Identity of the Initiator, IDr: Identity of the Responder).
- SP: The security policies.
- KEMAC: The Key Data Transport Payload, which contains the encrypted TGKs and a MAC.
- CHASH: The Cert Hash Payload, which is the hashes of the used certificates (e.g. CERTi).
- PKE: The Envelope Data Payload, which contains the encrypted envelope key, env_key.
- SIGNx: The signature covering the entire MIKEY message, which is generated using the signature key of the entity X.
- V: The verification message payload containing the MAC calculated over the entire MIKEY message.

As described in Figure 4.7, the MIKEY public key encryption method first chooses an envelope key env_key. This key is then to be encrypted using the public key PK_R of the Responder and conveyed in the PKE payload: PKE = $E(PK_R, \text{env\_key})$. Then, the encr_key and the auth_key are derived from the envelope key, env_key. These two keys are used to build the KEMAC payload where the encr_key is used to encrypt the TGKs. The encrypted part is then followed by a MAC calculated using auth_key over the entire KMAC payload. The whole MIKEY message is then integrity protected by the signature payload SIGNi.

4.3.2 Design motivations

As MIKEY [15] becomes more deployed, extensions to the base protocol have emerged [83], [10], [38]. Several of these extensions brought additional key distributions methods to MIKEY, for instance based on Elliptic Curve Cryptography (ECC) [158]. Since ECC support requires smaller keys while keeping the same security level as other asymmetric algorithms like RSA, ECC usage is considered interesting for devices with limited performance and storage capabilities. ECC extensions to MIKEY offer new mechanisms for authentication, encryption and digital signature to provide secure key distribution. ECC-based mechanisms such as ECDH to extend the Diff-Hellman exchange [142], ECDSA or ECGDSA for digital signatures, Elliptic Curve Integrated Encryption Scheme (ECIES) and Elliptic Curve Menezes-Qu-Vanstone Scheme (ECMQV) to provide, respectively, integrated encryption and authenticated key exchange, have been defined in [158]. To the best of our knowledge, ECC-based signcryption mechanisms have not been proposed for MIKEY, even though these mechanisms have been present in the literature for many years, and many ECC-based signcryption mechanisms offer a good performance thanks to their optimized authenticated public key encryption besides the advantages of ECC.

The novel key transport mechanisms for MIKEY are designed to put forwards the following motivations:

- **Performance and Efficiency**: Our proposed ECKSS signcryption scheme is able to transport secret data in a secure manner without intensive calculation. Thus, ECKSS-based methods for MIKEY is able to address the same scenario as the other key establishment methods in MIKEY [83]. In fact, existing MIKEY modes are intended for application-layer key management and multimedia applications. However, thanks to ECKSS lightweight computation requirements, the proposed methods can be considered in constrained environments such as IoT. We prove the feasibility of our proposed mechanisms in such environment in section 4.3.6. Furthermore, the mechanisms are based on elliptic curve cryptography (ECC). Additionally, when compared with existing ECC-based asymmetric methods of MIKEY, our proposed mechanisms are the most efficient while offering equivalent security guarantees. More details are provided in section 4.3.6.1.

- **PKI Independence**: ECKSS can be applied in the context where no access to a public-key infrastructure (PKI) is available. Indeed, the validation of entity's public keys is realized in equation (1) without certificates. Moreover, as pre-shared master secrets are not required, the proposed ECKSS-based schemes should be as scalable as other existing asymmetric mechanisms of MIKEY.

4.3.3 The MIKEY-ECKSS mode specification

Figure 4.8 defines the message exchanges for our first proposal MIKEY-ECKSS. Similarly to other MIKEY public key encryption methods such as MIKEY-RSA [15] or MIKEY-ECIES [83], the main objective of the Initiator's message is to distribute one or more TGKs in a secure manner. This method reuses the defined payload in section 4.3.1 except the payload ECKSSi

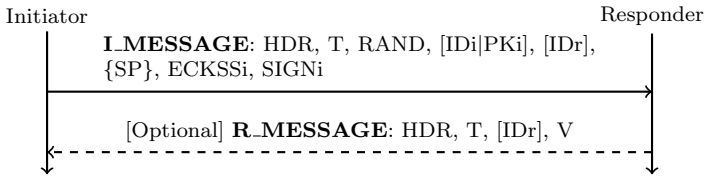


Figure 4.8: Elliptic curve Korean signature-based signcryption key distribution method for MIKEY

in the I_MESSAGE. This payload transports actually the encrypted TGKs through the triple (r, s, c). To guarantee the integrity protection, we employ the payload SIGNi, which is a signature covering the entire I_MESSAGE. As described in [15], the SIGNi payload will use either ECDSA or ECGDSA as the signature algorithm.

Upon receiving the I_MESSAGE, R first approves the integrity of the message by verifying the appended signature payload SIGNi. If the verification holds, it then uses the Unsigncrypt algorithm to decrypt the payload ECKSSi in order to obtain the values of TGKs. In case mutual authentication is required, the verification message, V, is calculated by building a MAC over the concatenation of the header HDR (the same as the one that was used in the I_MESSAGE), the timestamps, the identities of the two parties, using the authentication key. The latter is derived from the received TGKs. Then, we append the V payload to the concatenation (HDR, T, [IDi, PKi], [IDr]) to form the R_MESSAGE. However, as depicted in Figure 4.8, the R_MESSAGE is optional.

4.3.4 The MIKEY-ECKSS-HMAC mode specification

In this subsection, we describe in detail our second key distribution extension for MIKEY. We call this mode MIKEY-ECKSS-HMAC, since this mode uses ECKSS to envelop the TGKs and HMAC to ensure the authentication of the messages exchanged. As we shall see, the use of the signature payload SIGNx still requires multiple exponentiations in the signature generation and verification processes, e.g. 3 modular exponentiations are needed if using ECDSA. As a result, MIKEY-ECKSS-HMAC is even more lightweight than MIKEY-ECKSS, which is suitable for constrained devices.

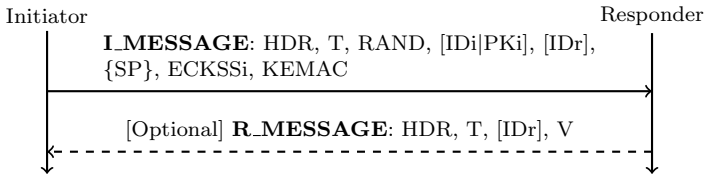


Figure 4.9: HMAC-authenticated Elliptic Curve Korean signature-based signcryption key distribution method for MIKEY

Figure 4.9 describes in detail the MIKEY-ECKSS-HMAC mode. We use the same notations for the payload names as defined in section 4.3.1. In the I_MESSAGE, the ECKSSi payload contains the triple (r, s, c) as depicted in section 4.2.3.2. The KEMAC payload conveys the Hash Message Authentication Code (HMAC) of the entire MIKEY message. This technique is also employed in the MIKEY-DHHMAC method [77]. The HMAC value is calculated using the secret key k_{auth}. This key is generated during the encryption of TGKs using the ECKSS

Signcrypt	Unsigncrypt
4) Generate a couple of secret keys: $\tau, k_{auth} = H_2(PK_I \| PK_R \| K)$	4) Get the secret keys: $\tau, k_{auth} = H_2(PK_I \| PK_R \| K)$

Table 4.3: Modifications made to the Signcrypt and Unsigncrypt algorithms

algorithm. Indeed, we make modifications in step 4 of the Signcrypt algorithm and in step 4 of the Unsigncrypt algorithm (see section 4.2.3.2 for more details). Another secret key k_{auth} in addition to τ is generated, as depicted in Table 4.3. This key is to be used in the creation of HMAC.

Upon receiving the I_MESSAGE, R first runs the Unsigncrypt algorithm to get the value of TGKs and k_{auth}. The authentication key k_{auth} is then used to verify that the I_MESSAGE has not been modified by an attacker. Indeed, a MAC is calculated over the entire I_MESSAGE using k_{auth}. This value is then compared with the hashed value extracted from the KEMAC payload. On the other hand, the R_MESSAGE's construction is optional as depicted in Figure 4.9. This message is only needed when mutual authentication between parties is required.

4.3.5 Security considerations

As this chapter proposes two new methods for the MIKEY protocol, existing security considerations discussed in [15] apply here as well.

As mentioned in [15], one should select a secure symmetric cipher supporting at least 128 bits as a practical choice for the actual media protection. In our proposals, the payload ECKSSi carries the actual encrypted TGKs, the used encryption algorithm should also offer a security level of at least 128 bits. For the selection of hash functions, we recommend to work at least with SHA-256 [56] when constructing the ECKSSi payload and other payloads as well. This should be seriously taken into account in order to achieve the 128-bit level.

Similar to other key transport mechanisms, our proposed methods do not provide the perfect forward secrecy property. A Diffie-Hellman key distribution resolves this issue but requires the transmission of the R_MESSAGE in order to set up a common secret key.

In order to provide the *certificateless* feature, our proposed methods rely on the binding of public values of communicating parties with the public keys issued by KMS. Thus, after validating a provided public value using equation (1), we can be sure that only KMS is able to generate such value. It also means that the KMS can read all traffic between any parties administrated by the KMS. However, we assumed that the KMS is a fully trusted party.

4.3.6 Experimental performance evaluation

In this section, we first quantify the performance of our schemes. Then, we describe our testing environment and the used methodology to achieve the experimental measurements. Finally, we provide in detail the performance results in terms of energy consumption and the time execution of our proposals including the ECKSS algorithm and the two proposed MIKEY modes.

4.3.6.1 Comparison with related work

Table 4.4 illustrates the performances of our two proposed methods and multiple ECC-based MIKEY modes in related work. The table first identifies if the scheme is a key exchange method or a key transport method. Then, it shows if the scheme is independent to the public key infrastructure or not. This property means that a PKI-independent scheme does not require

Mode	Type	PKI	I_MESSAGE			R_MESSAGE		
			PM	I	e	PM	I	e
MIKEY-DHSIGN [83]	KE	Yes	1 (DH) + 1 (SIGNi)	1(SIGNi)	0	1 (DH) + 2 (SIGNi)	1 (SIGNi)	0
MIKEY-ECQMV [83]		Yes	1(ECCPT)+1 (SIGNi)	1 (SIGNi)	0	1 (PKE)+2 (SIGNi)	1 (SIGNi)	0
MIKEY-SAKKE [97]	KT	No	3 (SAKKE)+1 (SIGNi)	1 (SIGNi)	0	2 (SAKKE)+4 (SIGNi)	0	1 (SAKKE)
MIKEY-ECIES [83]		Yes	2 (PKE)+1 (SIGNi)	1 (SIGNi)	0	1 (PKE)+2 (SIGNi)	1 (SIGNi)	0
MIKEY-ECKSS		No	1 (ECKSSi) + 1 (SIGNi)	0	0	2 (ECKSSi) + 2 (SIGNi)	0	0
MIKEY-ECKSS-HMAC		No	1 (ECKSSi)	0	0	2 (ECKSSi)	0	0

Table 4.4: Performance comparison of our propositions and ECC-based MIKEY modes in related work

Meaning of abbreviations: PM: Modular point multiplication; I: Modular Inversion; e: Pairing operation; PKI: Public Key Infrastructure; KE: Key Exchange; KT: Key Transport.

standard digital certificates to provide authentication between communicating parties and hence the scheme is discharged from complex operations during certificate verification, revocation and management processes. Then, the efficiency of each scheme is evaluated with respect to the computational cost demonstrated in terms of the number of expensive operations needed to generate the I_MESSAGE and the R_MESSAGE. Here, we consider the three most expensive operations for an ECC-based scheme: modular point multiplication (PM), modular inversion (I) and pairing operation (e). Furthermore, we provided also the name of the payload that requires these expensive operations. For example, in a PM column, the line "2 (PKE) + 1 (SIGNi)" means that two point multiplications are executed to build the PKE (public key encryption) payload and 1 other point multiplication is calculated to build the SIGNi (signature) payload. For simpler comparison, if not explicitly specified in each mode, we assume that SIGNi payload carries an ECDSA signature and its related data.

As we shall see, the first two modes in Table 4.4: MIKEY-DHSIGN and MIKEY-ECQMV, are two ECC-based key exchange methods proposed for MIKEY. These methods are based on the Diffie-Hellman key exchange [142]. Hence, the R_MESSAGE is compulsory in order to setup a common secret key. On the other hand, in a key transport mechanism, I envelops and sends directly a secret key/message that can be used as a session key without the response from R. As our proposed schemes are key transport mechanisms, we only make direct comparison with other key/message distribution mechanisms proposed for MIKEY. As depicted in Table 4.4, MIKEY-ECIES [83] seems to be our direct competitor in terms of performance since it is slightly more heavyweight than our first proposal MIKEY-ECKSS (with two more modular inversions to compute). In addition, MIKEY-ECKSS is more lightweight in the generation of the I_MESSAGE which can be beneficial for a very resource-constrained initiator. Our second proposal MIKEY-ECKSS-HMAC is even more efficient than our first one. As such, it requires only 1 point multiplication for generating the I_MESSAGE and 2 point multiplications for generating the R_MESSAGE. Furthermore, both proposals do not require certificates to validate the public values of communicating parties, which is not the case in MIKEY-ECIES mode. MIKEY-SAKKE [97] is also exempted from the use of PKI. However, this mode is much more expensive than our two methods since a pairing operation needs to be executed when receiving the R_MESSAGE.

4.3.6.2 Experimental tools and platforms

We implemented our assessment program in C for the operating system Contiki 3.0 [67]. Based on the Relic library version 0.4.0 [8], we evaluated our proposed MIKEY modes on the elliptic curves secg_k256. Its domain parameters have been recommended by SECG [157], which provides a security level of 128 bits. In addition, we opted for the sensor node Openmote to evaluate the required operations. Openmote [6] is a low power wireless sensor module featured with 32 MHz

Algorithm	Time (s)	Energy (mJ)
ECKSS Signcrypt	2.64	5.6
ECKSS Unsigncrypt	5.8	12.4
Public Key Validation	3.12	6.6

Table 4.5: Energy consumption and time execution of ECKSS algorithms on the Openmote platform

Cortex-M3 microcontroller, a CC2520-like radio transceiver, 32 kB of RAM, 512 kB of ROM and an IEEE 802.15.4 radio interface. This platform supports 32 bit addressing and sufficient RAM and ROM capacities. Such features are needed in order to use a cryptographic library along with an application on top of it.

In our testing scenario, we encrypted data using AES in CBC mode. For MAC and message verification function, we used SHA-256 as secure hash algorithm, which provides digests (hash values) that are 256 bits. Furthermore, we transported each time a TGK with the size of 32 bytes in our tests. In each case, the experimental measurements have been obtained by calculating the average of 100 executions for each operation.

4.3.6.3 Methodology

For measuring the processing time, we used two timers provided in the rtimer library of Contiki [67]. The first timer with 128 ticks per second was employed to measure the execution time of expensive operations. The second one is more powerful with 32768 ticks per second. It was used to measure the time duration of the mote running on a specific mode.

On the other hand, in order to assess the energy consumption, we employed a software-based online energy estimation mechanism described in [69]. In this model, we can calculate the energy consumption E in Joule of an operation on Contiki using the following equation:

$$E = U * \sum I_m * t_m \tag{4.2}$$

where U is the supply voltage, I_m is the hardware specific current draw, t_m is the time duration and m means a particular mode (e.g. active power mode, low power mode, transmit mode and receiver mode). In our scenario, the value of U is typically 3V, as with two new AA batteries. Besides, the current draw of the sensor node in each mode is extracted from its data sheet [1]. Concretely, we considered the following modes in our measurement: power mode 1 (cpu active mode), power mode 2 (low power mode), active-mode rx (receive mode) and active-mode tx (transmit mode). The consuming current draw for each mode are respectively: $I_{pm1} = 0.6mA$, $I_{pm2} = 1.3\mu A$, $I_{rx} = 20mA$ and $I_{tx} = 24mA$, as described in [1]. The time duration t_m that the mote is in mode m, is measured using Powertrace and Energest power profile [66]. These latters are pre-loaded tools in the Contiki OS, which provide an accuracy up to 94% of the energy consumption of a device.

4.3.6.4 Experimental results of ECKSS

In this subsection, we provide the experimental results of our signcryption scheme ECKSS. Table 4.5 shows the execution time and energy cost of ECKSS algorithms on the elliptic curve secg_k256 over the Openmote platform. The results reveal that even for a really lightweight signcrypt algorithm with only one point multiplication, it requires up to 2.6 s to compute and consumes 5.6 mJ. The resources required for an unsigncrypt are practically doubled since the

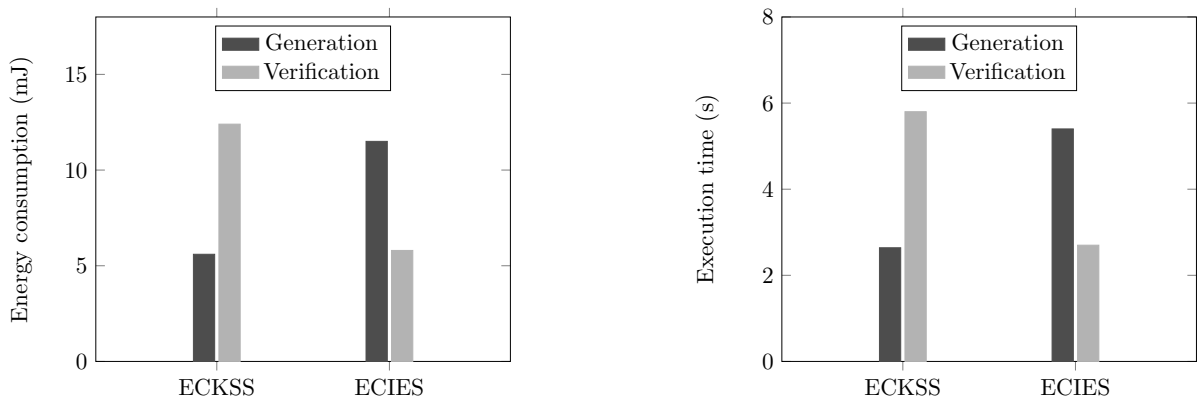


Figure 4.10: Performance comparison of our proposal ECKSS with the algorithm ECIES

Algorithm	Time(s)	Energy (mJ)
ECDSA signature generation	2.75	6.3
ECDSA signature verification	5.83	13.6

Table 4.6: Energy consumption and time execution of ECDSA algorithms on the Openmote platform

algorithm has to compute 2 point multiplications. We provide also in Table 4.5 the resources consumed by an entity to validate other party's public values. As we shall see, this process consumed the same order of magnitude of time and energy as the signcrypt algorithm. Such property is advantageous since the verification of certificates in a PKI-based scheme is usually complex and energy and time consuming.

In Figure 4.10, we compare the performance of our ECKSS implementation with the standard algorithm ECIES, as specified in [41]. ECIES's implementation is provided in the Relic library [8]. We remark that the total work load required by our scheme ECKSS is practically identical to the one of ECIES. As we can see in Table 4.4, ECKSS is more rapid in the data encryption process but slower in the data decryption process.

4.3.6.5 Experimental results of the proposed MIKEY modes

In this subsection, we describe the performance of our prototype implementations for the two proposed MIKEY modes.

In MIKEY-ECKSS's implementation, we use ECDSA as the signature algorithm. Table 4.6 provides the performance of ECDSA algorithms on the Openmote platform. These experimental results are measured based on the implementation of ECDSA provided in [8]. As we can see, ECKSS is even slightly more efficient than ECDSA both in the generation and verification processes. This fact is understandable since our proposal is exempted from two modular inversions compared to the ECDSA algorithm.

Additionally, to be more adapted to resource-constrained devices, we replace the timestamps payload by an incremental counter payload. This counter is used together with the random number (carried in RAND payload) to mitigate the replay attacks. If it is the first time that I communicates with R, the counter is set to 0. It is increased by 1 after every successful key/data transportation.

Mode	I_MESSAGE		R_MESSAGE	
	Time (s)	Energy (mJ)	Time (s)	Energy (mJ)
MIKEY_ECKSS	5.6	11.9	11.4	24.3
MIKEY_ECKSS_HMAC	2.6	5.6	5.7	12.2

Table 4.7: Energy consumption and time execution of our proposed MIKEY modes on openmote

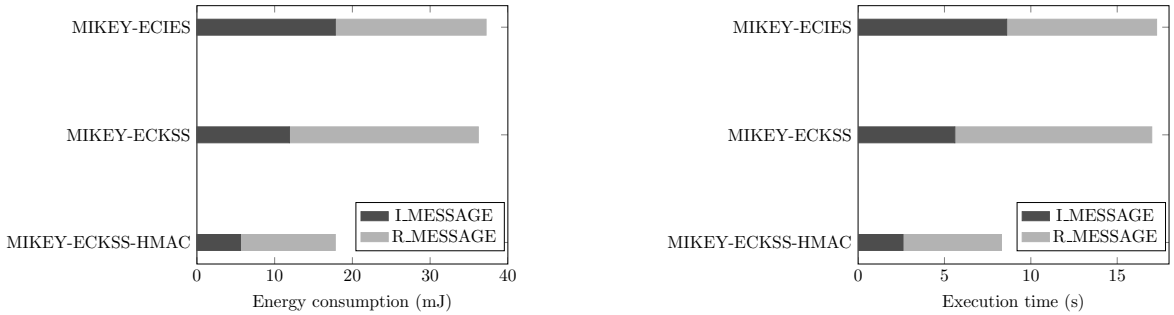


Figure 4.11: Performance comparison of our proposed MIKEY modes and MIKEY-ECIES mode

Table 4.7 demonstrates the average time and energy consumption for generating the I_MESSAGE and R_MESSAGE, respectively. The measures show that the MIKEY-ECKSS-HMAC mode is approximately two times more efficient than the MIKEY-ECKSS mode. The performance gap between them lies in the cost of creating the SIGNi payload. As such, instead of certificates and signatures, MIKEY-ECKSS-HMAC uses a keyed hash message authentication code (carried by the KEMAC payload) to guarantee the integrity of the messages exchanged.

Figure 4.11 provides a graphical view of the performance of our proposals in comparison with the MIKEY-ECIES mode. The performance of the latter are roughly estimated by summing the experimental results of the two algorithms ECIES and ECDSA given in Figure 4.10 and Table 4.6. As we shall see, the MIKEY-ECIES mode has a slightly higher computational cost in comparison with our proposed modes. However, it requires certificates to validate the public keys. This constraint could be very costly for a sensor node, since the verification of certificates is usually complex and consuming in time and energy.

4.4 Summary

In this chapter, we presented our first contribution [147, 148] in the respect of proposing lightweight software-based security mechanisms for the Internet of Things. The proposed solutions are twofold. Indeed, in the initial stage, we present a novel lightweight certificateless elliptic curve-based signcryption scheme, called ECKSS [147]. The proposed signcryption scheme has been formally proved to be outsider confidentially and insider unforgeablely secure against chosen ciphertext/message attacks in the random oracle model. In the second stage, we proposed two novel ECKSS-based key transport methods for MIKEY: MIKEY-ECKSS and MIKEY-ECKSS-HMAC [148]. Both methods are relieved from the dependance on a public key infrastructure thanks to the certificateless feature of ECKSS, and are more lightweight in terms of computation when compared with existing ECC-based key transport mechanisms proposed for MIKEY. The performance of the proposed methods have been demonstrated by experimental implemen-

tations on the Openmote sensor platform. The results confirmed that our proposed MIKEY extensions are feasible on resource-constrained devices. Hence, they can be used not only as key distribution mechanisms for real-time applications but also as lightweight key distribution solutions for the Internet of Things applications.

Following always the first direction defined in Section 3.3, we propose, in the next chapter, an optimization to an Attribute-Based Encryption (ABE) scheme. As such, in order to reduce the encryption cost of the considered ABE scheme, our solution delegates the most expensive operations to a semi-trusted server without revealing any sensitive information. Inherited from an ABE scheme, the proposed mechanism enables resource-limited IoT devices not only to rapidly distribute their data in a secure manner, but also to authorize access to the encrypted data.

Chapter 5

OEABE: Outsourcing the Encryption of Ciphertext-Policy Attribute-Based Encryption

In the context of Internet of things (IoT), there will be billions of embedded computing devices interconnected together [4]. These networks will massively increase the amount of data generated for analysis. Indeed, according to Cisco's forecast [2], the data created by the IoT devices will reach 507.5 zettabytes (i.e., 507.5 trillion gigabytes) per year by 2019. The access to this *big data* should be treated carefully, such that a user is able to only access data that he/she is authorized to. In general, we can manage user's access rights by performing role-based access control (RBAC) [172]. However, this model requires complex backend servers for the management of access rights, in addition to trusted storage sites enforcing access control.

In case the IoT devices authorize who can read a specific data, lightweight access control mechanisms are to be used in order to not rapidly consume resources (e.g. CPU, memory and energy) of constrained devices. As described in Section 2.1.4, Ciphertext-Policy Attribute-Based Encryption (CP-ABE) offers fine-grained access control while giving users the flexibility to determine who can decrypt data at runtime. However, CP-ABE is not only known to be complex in the decryption phase [95, 203] but also energy-consuming in the encryption process. In fact, CP-ABE uses complex cryptographic operations to encrypt a message. Additionally, the time and energy required for these operations grow with the number of attributes in the access policy.

In this chapter, we adapt the encryption algorithm of the original CP-ABE scheme proposed by Bethencourt et al. [28] in order to better fit the resource requirements of IoT devices. Concretely, we first propose a new method, namely, OEABE for Outsourcing mechanism for the Encryption of Ciphertext-Policy ABE. The main idea is to reduce the encryption cost by securely delegating the most expensive computations in the encryption phase of CP-ABE to a semi-trusted party. Second, we show that our proposed mechanism is resistant against both external and internal attackers, without revealing the data content and used security keys. Third, we present experimental performance results to compare our solution with the original CP-ABE [28]. In these experimental results, we estimate the execution time and the energy consumption of our proposal on an emulated Wismote sensor platform. Additionally, we implement our solution and compare its performance with the CP-ABE implementation proposed by Bethencourt [3] on a laptop. Similar applications of our solution to other schemes proposed in [127, 195] are possible, but they are out of scope of this work.

The rest of this chapter is organized as follows. Section 5.1 surveys related work on CP-ABE schemes and solutions proposed to reduce the cost of encryption for ABE schemes. Section 5.2

describes the considered system and threat model. We describe in detail our proposed method to outsource the encryption of CP-ABE in section 5.3. In section 5.4, we provide a security analysis of our proposed scheme. The performance assessment of our proposal is given in section 5.5. Section 5.6 gives concrete examples on how to apply OEABE in real world scenarios, while section 5.7 concludes our work.

5.1 Related work

In this section, we first survey existing proposals on the Attribute-Based Encryption schemes. Then, we present existing approaches that attempt to reduce the computational cost of the CP-ABE encryption.

5.1.1 Related work on Ciphertext-Policy ABE schemes

Attribute-Based Encryption (ABE) is a very promising technique for fine-grained access control applications on encrypted data. After Sahai et al. [168] introduced the first ABE scheme, many ABE-based schemes [92, 28, 152, 47, 48, 127] have been proposed in the literature. An ABE scheme can be roughly characterized according to the following criteria: the method that the policy is embedded (key-policy [92] or ciphertext-policy [28]), the type of access structure (monotonic [92] or non-monotonic access structure [152]) and the number of trusted attribute authority (single-authority [28] or multiple-authority [47, 48, 127]).

In this thesis, we interested in the ciphertext-policy ABE schemes. As such, although the KP-ABE scheme offers fine-grained access control feature, it has one main disadvantage. Indeed, the data owners cannot decide on who has access to their encrypted data, except by their choice of descriptive attributes for the data, since the access policy is embedded in the user private keys. As a result, the data owners have to trust the key issuer. Ciphertext-policy ABE schemes remove such inconvenience by directly embedding the access policy on the ciphertext. The data owners can now authorize who can have access on their encrypted data. Thanks to that interesting property, many CP-ABE schemes are then proposed (e.g. [51, 73, 91, 111, 149]).

In addition, our proposed idea is applied in a single-authority ABE scheme due to its simple architecture. We mention also some multiple-authorities ABE schemes as they can be used to extend our idea as well. In a single-authority ABE scheme, all user attributes and private keys are generated by a central attribute authority. This method facilitates the management of attributes and keys. However, if the number of authorized users increases, the attribute authority must be powerful and at the same time can be considered as a single point of failure. To address this problem, multiple-authorities ABE schemes [47, 48, 40, 127] have been proposed. These solutions use multiple parties to distribute attributes and private keys to users. Such approach offers the scalability for the system even if the number of users become large.

5.1.2 Reducing the computational cost of CP-ABE encryption

This subsection discusses existing solutions that reduce the cost of encryption in CP-ABE schemes. By doing so, CP-ABE schemes can become more adapted for resource-constrained data owners.

The first approach is to reduce the communication overhead due to the ciphertext length. As such, in a CP-ABE scheme, the ciphertext length and the number of pairing operations usually depend on the number of attributes embedded in the access structure. Several works [73, 64] propose CP-ABE schemes that have constant ciphertext length and fast decryption.

As another approach, many other works propose to outsource CP-ABE encryption operations. Among works conducted in the literature, Zhibin et al. [203] introduce a solution to

securely offload both encryption and decryption processes of CP-ABE to external cloud-based services. In their encryption process, the access policy is composed of two access structures $\mathcal{T}_1$ and $\mathcal{T}_2$, which are connected by an AND root node. $\mathcal{T}_1$ is sent to the outsider encryption service in order to generate the first part of ABE ciphertext. $\mathcal{T}_2$ is a sub-tree with only one attribute. The second part of the ciphertext is computed by the user using $\mathcal{T}_2$, where the message is actually encrypted. However, this solution requires 3 exponentiations on the user side which is less efficient than our proposal. Touati et al. [188] propose a proxy-based CP-ABE in the context of heterogeneous environment, such as, IoT. Their solution presumes that there exist at least two trusted, non-colluding and unconstrained proxies in the neighbourhood of each resource-constrained device. By dividing the secret random number s in n parts, each ciphertext part is computed by the assisting nodes, where n is the number of proxy nodes. This solution does not require any exponentiation to be executed in the constrained device. However, its architectural assumptions are excessively unpractical. In addition, the communication overhead is non-negligible, especially for highly resource-limited devices in IoT. Indeed, this approach necessitates $2n + 1$ message exchanges instead of one to form a valid ABE ciphertext. As another approach, Bianchi et al. [29] propose to employ directly CP-ABE in WSN, provided that the sensor nodes are equipped with energy harvesting capabilities. Such devices are able to harvest energy from solar light, or even artificial light, such as table lamps and ceiling. When the device is fully charged and the power consumption on device is lower than the harvested power, the surplus energy is used to pre-compute the CP-ABE policies. This approach is convenient when the ciphertext has been computed and stored in the device beforehand. However, the pre-calculated policies must be stored in the typically limited RAM memory.

Our idea is to delegate partially the CP-ABE encryption to a semi-trusted entity. We apply our solution in the Bethencourtet al.'s CP-ABE [28]. This scheme uses a single attribute authority and is able to support both monotonic and non-monotonic access structures. In addition, it also provides most of the desirable ABE features (i.e. Data confidentiality, Fine-grained access control, User revocation and Collusion resistance). Background knowledge on ABE is defined in Section 2.1.4.

5.2 System and Threat model

In this section, we provide descriptions of our considered system and threat model.

5.2.1 System model

In a typical scenario of an ABE system, we consider the three following actors:

- Two parties: a Data Producer (DP) and a Data Consumer (DC), which respectively encrypt and decrypt data.
- A trusted Key Distribution Center (KDC), which is responsible for the delegation credential management and distribution (i.e. user attributes and private keys). This function can be hosted by the CP-ABE KDC.

In our considered scenario, DP is a resource-constrained device, such as a sensor node, while DC can be a client or services that collect data. Figure 5.1 depicts a typical exchange where DP builds an ABE ciphertext (ABE-CT) and sends it to DC. To reduce the cost of ABE encryption, we remove direct exchanges between DP and DC as described in Figure 5.2. In the considered scenario, DP computes partially the ciphertext and leaves the remaining part of computation

Figure 5.1: DP builds an ABE cipertext and sends it to DC

for a new actor, namely Delegatee (DG). This party is a non-constrained resource devices, e.g. a smartphone, a proxy or a cloud service. The completed ABE-CT is generated by DG. It

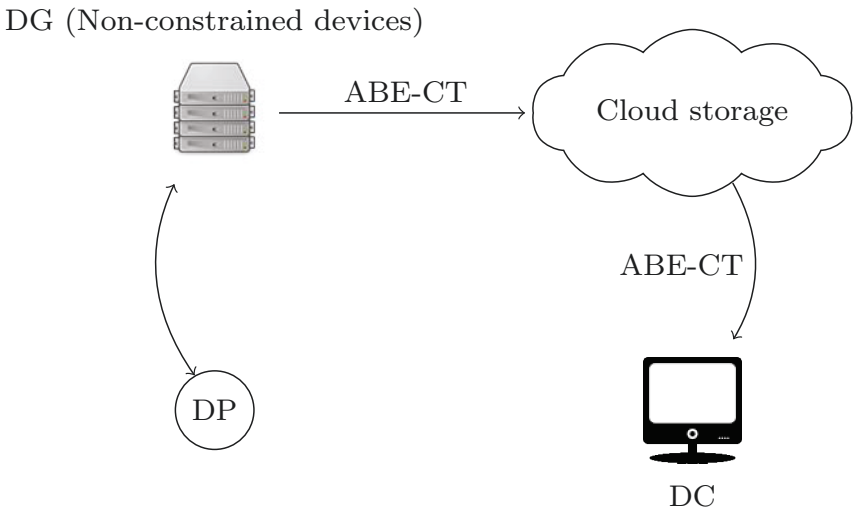


Figure 5.2: Our considered scenario: DP computes partially the ABE-CT, which is then completed by the DG

can be either stored by a cloud storage service or directly sent to DC. DC can retrieve the cleartext message only if its set of attributes satisfies the access structure. Even though not being considered in this work, mitigation of denial-of-service attacks is possible by providing data integrity protection to all the messages exchanged between DP, DG and DC, for example, using a pre-shared secret key. We do not present this in our protocol description for simplicity.

5.2.2 Threat model

This subsection describes the considered threat model in our proposal. Definition 19 provides an informal description of the considered adversary model. Formal definition of such model can be found in [90].

Definition 19 (Honest-but-curious adversaries)**.** The honest-but-curious (HBC) adversaries follow correctly the protocol specification. However, the adversary keeps a record of all intermediate computations in order to learn information that is supposed to remain private.

In our model, we consider that the KDC is a trusted party that provides credentials and attributes for each party in the system. On the other hand, DG is a *honest-but-curious* party, which may attempt to guess the content of DP's secret key or data from received ciphertexts produced by DP, while performing tasks according to the protocol and returning correct results, as described in Definition 19. Our assumption is far from being unpractical since DG acts as an

enforcement point that assures data authorization on behalf of the constrained-device DP. Its existence is considered transparent to DC which is an external party receiving encrypted data. Finally, we do not handle attacks from DC(s) (with collusion or not) targeting disclosure of secret keys of DP from the CP-ABE ciphertext as the security and the collusion-resistance property of original CP-ABE have been proved in [28]. In fact, the used CP-ABE [28] scheme is proven secure in the generic group model. This model is an *artificial* model which is weaker than the standard one. Several recent CP-ABE constructions (e.g. [195, 50]) are said to be secured in the standard model. However, in this work, we consider the application of our delegation mechanism to the Bethencourt et al.'s scheme [28] because it is the most simple and expressive CP-ABE construction. Similar applications of our delegation technique to other CP-ABE schemes are beyond the scope of this work.

In section 5.4, we first provide an informal security analysis of possible attacks from a computational bounded attacker, namely $\mathcal{A}$, that attempts to guess the cleartext and DP's private keys. $\mathcal{A}$ can be an external entity or even an internal entity which is a curious DG. Then, we confirm that our scheme does not weaken the security of CP-ABE and thus reveals no information on the cleartext message M and other secret parameters including the secret parameters of the original CP-ABE scheme and the delegation key used in our proposed mechanism.

5.3 Secure outsourcing encryption mechanism for CP-ABE

In this section, we first present in detail the CP-ABE scheme proposed by Bethencourt et al. [28]. Then, we describe in detail our outsourcing mechanism OEABE for the encryption of CP-ABE. Finally, we prove that our proposed delegation mechanism is consistent. For more details on the cryptographic definitions of a CP-ABE system, the reader can refer to Section 2.1.4 of this document.

5.3.1 Bethencourt et al.'s Ciphertext-policy Attribute-Based Encryption

As described in [28], a CP-ABE system offers users the capability to decide who can access to their encrypted data. Each user is associated with a set of attributes. These attributes are used by KDC to generate user's private keys. Upon receiving a message M, the encryptor first chooses a monotone access structure $\mathcal{T}$ which contains the authorized sets of attributes for M. The message is then encrypted using as input this access structure and the public key PK. Only users holding a set of attributes that satisfy $\mathcal{T}$ can decrypt the ciphertext. Note that PK is a global public key, meaning that it is not specific to any particular user.

The fundamental algorithms of a CP-ABE scheme are specified in the following:

- $\mathsf{Setup}(sp) \rightarrow (PK, MK)$. Given the implicit security parameter sp, output the public parameter $PK = (\mathbb{G}_0, P, Q = \beta.P, f = (1/\beta).P, e(P,P)^{\alpha})$ and the master key $MK = (\beta, \alpha.P)$.

- $\mathsf{KeyGen}(MK, S) \rightarrow SK$. Given the master key MK and a set of attributes S, return a private key $SK = (D = ((\alpha + r)/\beta).P, \forall j \in S : D_j = r.P + r_j.H(j), D'_j = r_j.P)$, where r and r_j are randomly chosen from $\mathbb{Z}_p$.

- $\mathsf{Encrypt}(PK, M, \mathcal{T}) \rightarrow CT$. Given the public key PK, the message M and the access structure $\mathcal{T}$, return a ciphertext CT. Concretely, we first choose randomly a number s from $\mathbb{Z}_p$. Then, let q_x be the polynomial for node x (including the leaves) in the tree $\mathcal{T}$. Starting from the root R, we set $q_R(0) = s$ and find polynomial q_x, such that $q_x(0) = q_{\mathsf{parent}(x)}(\mathsf{index}(x))$ for each node x in the tree. The function $\mathsf{index}(x)$ returns a number

associated with the order of node x. Let Y be the set of leaf nodes in $\mathcal{T}$ and q_y be a polynomial for each leaf node $y \in Y$. The ciphertext is constructed as follows:
$CT = (\mathcal{T}, \tilde{C} = M.e(P,P)^{\alpha s}, C = s.Q, \forall y \in Y : C_y = q_y(0).P, C'_y = q_y(0).H(att(y)))$,
where H is a hash function that transforms an attribute into an elliptic curve point on $\mathbb{G}_0$.

- $\mathsf{Decrypt}(\mathsf{CT}, \mathsf{SK}) \rightarrow M$. Given the ciphertext CT and the private key SK, return a message M.

5.3.2 OEABE description

The proposed solution OEABE consists of two algorithms PreDelegation and CompDelegation, as depicted in Figure 5.3. The first algorithm is executed by KDC in order to configure all the security parameters including the secret values to be used in the delegation process. The second one is initially run by DP and then DG each time data are encrypted. This algorithm devises the Encrypt process into two stages. The first lightweight stage is run on the constrained devices DP, while the second stage is executed by DG. The other algorithms of CP-ABE are kept unchanged. A detailed description of the outsourcing process is given as follows.

PreDelegation. In this phase, KDC first runs the Setup algorithm as in a basic CP-ABE system. Depending on the secure parameter k as input, the algorithm chooses a bilinear group $\mathbb{G}_0$ of prime order p with a generator P. Two hash functions $H : \{0,1\}^* \rightarrow \mathbb{G}_0$ and $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$, which behave as random oracles [27], are also chosen. Then, the public key and the master key are generated as follows: $PK = (\mathbb{G}_0, P, Q = \beta P, f = (1/\beta)P, e(P,P)^{\alpha})$ and $MK = (\beta, \alpha P)$. In the second step, KDC takes as input P, Q and H, and outputs a secret delegation key $d \xleftarrow{\$} \mathbb{Z}_p$ for each DP and a list of security parameters $(\gamma_1 = -dQ, \gamma_2 = -dP, \{\gamma_{3t}\}_{t \in \mathcal{U}} = \{-dH(t)\}_{t \in \mathcal{U}})$, where $\mathcal{U}$ is the list of all defined attributes in the system. d is securely transferred to DP. Similarly, the parameters $(\gamma_1, \gamma_2, \{\gamma_{3t}\}_{t \in \mathcal{U}})$ are securely sent to DG. As we shall see, DG can not retrieve any information on d due to the DLP problem (see section 2.2.2 for more details).

CompDelegation. This phase describes the delegation process. The main idea is that DG executes the most expensive operations on behalf of the constrained device to form a valid ABE-CT without any knowledge on the secret message M. In order to encrypt a message M under the access policy ap, DP first chooses randomly a number s from $\mathbb{Z}_p$. Then, to securely outsource expensive calculations to DG, DP binds the value of s using a random value generated from $\tilde{C}$. The detailed procedure on DP side is described as follows:

- Choose randomly $s \xleftarrow{\$} \mathbb{Z}_p$
- For a message M, compute $\tilde{C} = M(e(P,P)^{\alpha})^s$
- Compute $s' = s + h(\tilde{C})d$
- Generate the temporal ciphertext: $CT' = (ap, \tilde{C}, s')$

The expensive computations to generate C, C_y and C'_y are done by DG. DP is only required to compute one exponentiation to achieve the value of $\tilde{C}$. Then, the temporal ciphertext CT' is sent to DG.

Upon receiving the non-completed ciphertext from DP, DG uses the access policy ap to define an access tree $\mathcal{T}$. Let q_x be the polynomial for node x (including the leaves) in the tree $\mathcal{T}$. Starting from the root R, we set $q_R(0) = s'$ and find all polynomials q_x for each node x in the tree, as in the original Encrypt algorithm of CP-ABE. We also define Y as the set of leaf nodes

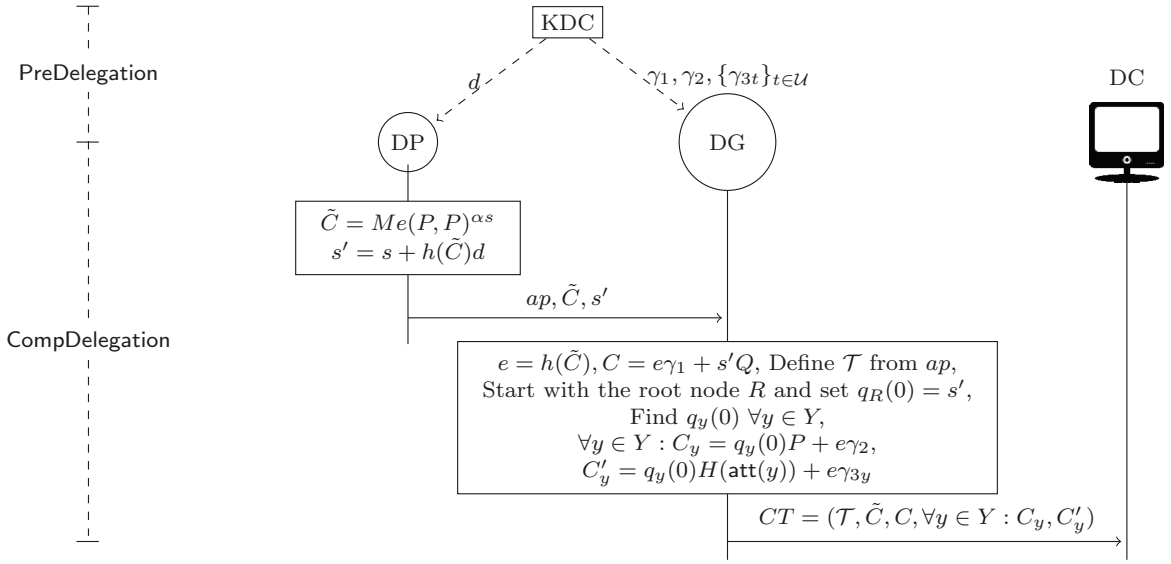


Figure 5.3: Secure delegation for the encryption of CP-ABE

Meaning of abbreviations: A - -> B: Secure channel between A and B; d: Delegation key of DP; U: List of all defined attributed in the system; $\gamma_1 = -dQ$; $\gamma_2 = -dP$; $\{\gamma_{3t}\}_{t\in\mathcal{U}} = \{-dH(t)\}_{t\in\mathcal{U}}$; ap: Access policy generated by DP; $\mathcal{T}$: Access tree; Y: Set of leaf nodes in $\mathcal{T}$.

in $\mathcal{T}$ and q_y as a polynomial for each leaf node $y \in Y$. Then, DG has to perform the following procedure:

- Generate $e = h(\tilde{C})$
- Compute $C = e\gamma_1 + s'Q$
- $\forall y \in Y$, compute $C_y = q_y(0)P + e\gamma_2$ and $C'_y = q_y(0)H(\mathsf{att}(y)) + e\gamma_{3y}$
- Send the final ciphertext $CT = (\mathcal{T}, \tilde{C}, C, \forall y \in Y : C_y, C'_y)$ to DC.

This proposal allows the constrained device to offload the expensive calculations of C, C_y and C'_y, $\forall y \in Y$ to DG. This intermediate information is then merged with CT' by DG to form the final ciphertext CT. As a result, the constrained device needs only one message exchange to generate an ABE-CT. Here, we note that sending s' does not reveal any secrets of DC. We will discuss this in section 5.4.

5.3.3 Correctness of our proposal

We need to verify that DC is able to decrypt the ciphertext CT provided by DG. Indeed, we have:

$$C_y = q_y(0)P + e\gamma_2 = (q_y(0) - ed)P$$

and

$$C'_y = q_y(0)H(\mathsf{att}(y)) + e\gamma_{3y} = (q_y(0) - ed)H(att(y)),\ \forall y \in Y$$

Upon receiving the ciphertext CT, DC runs the Decrypt algorithm as defined in [28]. For each leaf node y from $\mathcal{T}$, let $i = att(y)$ and S be the set of attributes possessed by DC. If $i \in S$, the recursion function $\mathsf{DecryptNode}(CT, SK, y)$ is proceeded as follows:

$$\mathsf{DecryptNode}(CT, SK, y) = \frac{e(D_i,\, C_y)}{e(D'_i,\, C'_y)} = \frac{e(rP + r_i H(i),\, (q_y(0) - ed)P)}{e(r_i P,\, (q_y(0) - ed)H(i))} = e(P,P)^{r(q_y(0)-ed)}$$

We define the Lagrange coefficient $\triangle_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, $\mathcal{S}$, of elements in $\mathbb{Z}_p$: $\triangle_{i,\mathcal{S}}(x) = \prod_{j \in \mathcal{S}, j \neq i} \frac{x-j}{i-j}$. For a non-leaf node x, we consider a recursive procedure. Indeed, for all nodes z that are children of x, it calls $\mathsf{DecryptNode}(CT, SK, z)$ and stores the output as F_z. Let S_x be an arbitrary $k_x - sized$ set of child nodes z, DC computes:

$$F_x = \prod_{z \in S_x} F_z^{\triangle_{i,S'_x}(0)} = \prod_{z \in S_x} (e(P,\, P)^{r(q_z(0)-ed)})^{\triangle_{i,S'_x}(0)} = \frac{\prod\limits_{z \in S_x} (e(P,P)^{rq_z(0)})^{\triangle_{i,S'_x}(0)}}{e(P,P)^{red \sum\limits_{z \in S_x} \triangle_{i,S'_x}(0)}} =$$

$$\frac{e(P,P)^{rq_x(0)}}{e(P,P)^{red}} = e(P,P)^{r(q_x(0)-ed)}, \text{where } i = \mathsf{index}(z) \text{ and } S'_x = \mathsf{index}(z) : z \in S_x$$

,

The above result is deduced from the polynomial interpolation property and the fact that $\sum\limits_{z \in S_x} \triangle_{i,S'_x}(0) = 1$.

Applying the above computation to the root node R of the tree $\mathcal{T}$, we set $A = \mathsf{DecryptNode}(CT, SK, R) = e(P,P)^{r(q_R(0)-ed)} = e(P,P)^{r(s'-ed)} = e(P,P)^{rs}$. Besides, we have that the value of C is identical to the one in original CP-ABE [28]. As such, $C = e\gamma_1 + s'Q = e(-d)Q + (s + ed)Q = sQ$. The algorithm decrypts by computing $\tilde{C}/(e(C,D)/A) = Me(P,P)^{\alpha s}/(e(s\beta P, (\alpha + r)/\beta)/e(P,P)^{rs}) = M$. Therefore, DC obtains the correct cleartext message from the ciphertext provided by DG.

5.4 Security analysis

In this section, we argue that the security of our delegation mechanism for CP-ABE is equivalent to the hardness of the DLP problem and the intractability of hash function in the random oracle model [27]. As such, if there is any vulnerabilities in our mechanism then these vulnerabilities must exploit mathematical problems related to the DLP problem or to the used cryptographic hash functions.

As described in section 5.2.2, an attacker $\mathcal{A}$ can be an external or internal entity. Thus, we consider the two following situations with regard to the attacker's position:
– **Case 1:** $\mathcal{A}$ is an external resource-bounded adversary. The main objective of $\mathcal{A}$ is to guess the cleartext message M and DP's secret values. To do so, $\mathcal{A}$ observes and eavesdrops several exchanges of different sessions between DP, DG and DC. It may rely on retrieved information from previous sessions to guess the current encrypted message and DP's private key. We consider a particular session j. Table 5.1 lists all the values that are available to DG, DC and $\mathcal{A}$. As we shall see, apart from public parameters, the values that $\mathcal{A}$ gets are ap_j, s'_j and CT_j (for session j). In order to guess the cleartext message M, $\mathcal{A}$ has to either break the ciphertext CT_j or guess the secret number s_j (for session j). The latter means that $\mathcal{A}$ needs to guess the delegation key d, as it is used to mask the value of s_j in session j.

We note that CT_j does not contain any information related to d and $\mathcal{A}$ is unable to figure out the message M from CT_j thanks to the security of the original CP-ABE scheme [28]. As a consequence, $\mathcal{A}$ now has only one possibility left guessing the value of d in order to obtain M. As a result, we need to be sure that s'_j does not disclose the value of d. Our arguments are based on a basic result on the integer ring $\mathbb{Z}_p$.

DG	DC	External $\mathcal{A}$
$\mathbb{G}_0, P, Q, f, e(P,P)^\alpha, CT,$ $\gamma_1, \gamma_2, \{\gamma_{3t}\}_{t\in\mathcal{U}}$	$\mathbb{G}_0, P, Q, f, e(P,P)^\alpha, CT,$ M, SK	$\mathbb{G}_0, P, Q, f, e(P,P)^\alpha, CT,$ ap, s'

Table 5.1: Information accessible to DG, DC and an external attacker $\mathcal{A}$

Lemma 20 ([12])**.** Let p be a prime number. Given an integer $k \in \mathbb{Z}_p$ and an r uniformly distributed over $\mathbb{Z}_p$, the values $\theta = k + r \,(mod\, p)$ and $\delta = kr \,(mod\, p)$ are also uniformly distributed over $\mathbb{Z}_p$.

In our assumption, the hash function h is modeled as random oracle [27]. As a result, $h(\tilde{C}_j)$ is randomly distributed in $\mathbb{Z}_p$. Besides, we have that s_j is uniformly chosen at random in $\mathbb{Z}_p$. Hence, applying the result in Lemma 20, we obtain that $s'_j = s_j + h(\tilde{C}_j)d$ is also uniformly distributed in $\mathbb{Z}_p$. Even if $\mathcal{A}$ eavesdrops the communication between DP and DG and hence will be able to re-generate the value of $e = h(\tilde{C}_j)$, it can not guess neither d nor s_j from s'_j. Thus, the only way $\mathcal{A}$ can guess the values of s_j and d from s'_j is to use a brute force attack on $\mathbb{Z}_p$. This is impossible because $\mathcal{A}$ is supposed to be a polynomial time adversary.

– **Case 2:** The attacker $\mathcal{A}$ is a curious DG. This means that such DG is an untrustworthy party, which is curious about the encrypted data sent between DC and DP. As in Case 1, DG observes and keeps record of intermediate computations of different sessions between DP and DC. DG's objective is to guess the cleartext message M and DP's private keys. As described in Table 5.1, although DG knows $\gamma_1 = -dQ$, $\gamma_2 = -dP$, $\{\gamma_{3t}\}_{t\in\mathcal{U}} = \{-dH(t)\}_{t\in\mathcal{U}})$, $C = sQ$, $C_y = (q_y(0) - ed)P$ and $C'_y = (q_y(0) - ed)H(att(y))\ \forall y \in Y$, it can not derive the values of d and s thanks to the hardness of the DLP problem. Besides, DG also has the value of s'. As another attempt, DG can try to guess the values of d and s from s'. However, the values of d and s are unconditionally secure, as proved above in Case 1.

Therefore, given the hardness of DLP on $\mathbb{G}_0$ and the intractability of hash functions in the random oracle model, an attacker can neither figure out the cleartext M nor derive the delegation key d nor guess any secret keys of DP.

5.5 Performance analysis

In this section, we first compare numerically the performance of our solution with respect to the related works described in section 5.1. Then, we present the experimental performance assessment of our proposal. We first estimate the energy consumption of our proposal in the emulated Wismote sensor platform. Then, we compare the performance results of our implementation OEABE and the original CP-ABE implementation [3] in terms of execution time on a laptop.

5.5.1 Quantified Comparison

Table 5.2 illustrates the cost of the encryption and the number of messages to be exchanged of our ABE scheme and several related works described in section 5.1. The encryption cost is depicted by the number of the exponentiations to be executed, which is considered as one of the most expensive operation in a cryptographic algorithm. On the other hand, the transmission is evaluated by the number of message exchanges. As shown in Table 5.2, our proposal does not only need one exponentiation, but also require merely one message exchange like the original CP-ABE scheme [28]. The C-CP-ABE [188] scheme does not require any exponentiations. However, their approach demands n trusted proxies ($n \geq 2$) and needs $2n + 1$ exchanges in order to form

a valid ciphertext. These requirements are not scalable and are costly for resource-constrained devices, where communication costs are non negligible.

	EXP	NME
original CP-ABE [28]	$2\|Y\|+2$	1
AGREE [29]	$3\|Y\|+1$	1
C-CP-ABE [188]	0	$2\|n\|+1$
PP-CP-ABE [203]	3	1
Our scheme: OEABE	1	1

Table 5.2: Comparison of our proposal and related work
EXP: number of exponentiations to be executed in the encryption phase; NME: Number of message exchanges; Y: number of leaf nodes in the access policy, n: number of trusted proxies.

5.5.2 Estimation of energy consumption on emulated sensor platform

We conducted experimental evaluation of elliptic curve based cryptographic (ECC) operations on an emulated sensor platform Wismote with a 16 MHz MSP430x micro-controller, 16 kB of RAM, 128 kB of ROM. Our testing environment is on Contiki 2.7 [67]. The evaluation program has been tested on four elliptic curves secg_p160, nist_p192, nist_p224 and nist_p256 in increase order of security level. Their domain parameters have been recommended by SECG [9] and NIST [81]. We measure the time execution and the energy consumption for ECC exponentiation and pairing. For more details on the estimation process, please refer to our paper [147]. Table 5.3 describes estimated time execution and energy consumption of an exponentiation and a pairing in 4 different elliptic curves. As we shall see, even for a medium-level security application (80 bits), if we use policy composed of 30 attributes, the time and energy needed to compute an ABE-CT in the original scheme [28] is approximately 152.5 s and 1004.4 mJ for only the exponentiation process, instead of 2.46 s and 16.25 mj using our proposal. Figure 5.4 presents the estimated energy consumption in log scale while the number of attribute in the access policy varies from 1 to 30. We adopt this range because it has been considered to be representative enough for many real world applications [13]. As shown in the figure, we can confirm that energy consumption in OEABE is much lower than the basic CPABE scheme. The gap between both solutions increases with the security level.

5.5.3 Execution time of OEABE encryption on a laptop

Our OEABE implementation is based on the cpabe library [3], which implements the scheme proposed by Bethencourt et al. [28]. We evaluate our program on a laptop device Ubuntu 14.4 LTS, 2.8 GHz Intel(R) Core(TM) i5-3360M CPU, 8GB RAM. For testing purposes, we changed

Parameters	Strength	Exponentiation	Pairing
secg_p160	80	2460ms/16.25mJ	3533ms/23.32mJ
nist_p192	96	3463ms/22.53mJ	6586ms/43.47mJ
nist_p224	112	4782ms/32.05mJ	9573ms/63.19mJ
nist_p256	128	18.91s/124.07mJ	36,16s/238.13mJ

Table 5.3: Time execution and Energy consumption of ECC operations on Wismote [147]

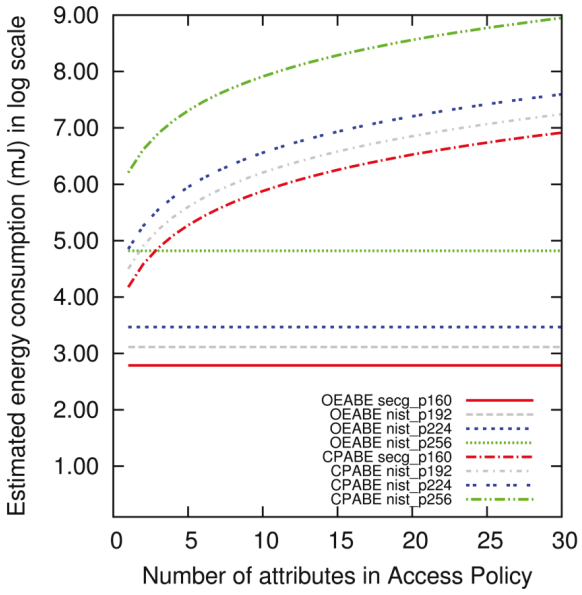


Figure 5.4: Estimation of energy consumption of OEABE and CPABE in the emulated Wismote platform

Security level	length of r in bits	length of q in bits
80	160	512
112	224	1024
128	256	1536

Table 5.4: Bit length of q and r to obtain desired security level

the input parameters of the cpabe library so that the security level of the ABE scheme varies from 80 up to 128 bits. As the cpabe library uses the pbc library [7] for pairing-based operations, we also use the Type A pairings supported by pbc in our tests. This type of pairing is constructed on the elliptic curve $y^2 = x^3 + x$ over the field $\mathbb{F}_q$ for some prime $q = 3\,(mod\,4)$. Such curve can offer different security levels by modifying the two parameters: the prime order r and the field q. We generate different input parameters with desired security level by varying the values of q and r as shown in Table 5.4. Note that the performance gap of our scheme and CP-ABE are independent from the size of the message, because the actual data encryption is performed with AES by means of a symmetric key which is then encrypted using the ABE mechanism. Figure 5.5 shows the average execution time of the encryption phase for both OEABE and CP-ABE implementations. The results are presented for three levels of security: 80, 112 and 128 bits. Each set of input parameters has been randomly chosen at runtime based on the security level. In each case, the results have been obtained with an average of 100 executions for each operation. The access policies are also chosen at random from one to 100 attributes. The policies with 100 attributes are not rare in practical usage when using complex attributes that contain integer comparison operators. For example, for a simple comparison $a < 11$, we need to convert it to a policy tree of 5 attributes composed of AND and OR gates as explained in [28]. As we can see in Figure 5.5, the encryption time of CP-ABE is almost linear with the number of attributes involved in the access structure. On the other hand, our solution requires much less time on the device to finish an encryption. Indeed, in a 128-bits scale, when using

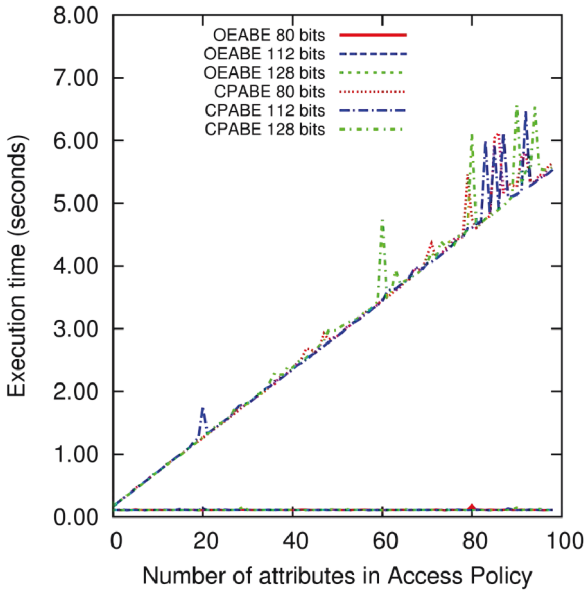


Figure 5.5: Average execution time in the encryption phase of OEABE and CPABE on a laptop

policies composed of 100 attributes, encryption requires 0.12 s with our solution and 5.64 s with the basic scheme, respectively. The reason is that our OEABE scheme does not require more than one exponentiation while CP-ABE demands up to $2|Y| + 2$ exponentiations, where $|Y|$ is the number of attributes in the access policy.

5.6 Examples of applications of OEABE

CP-ABE has been considered as a highly flexible cryptographic scheme with multiple applications in the context of IoT. Our proposal removes the intensive computation requirement of CP-ABE encryption, which makes the encryption of CP-ABE feasible even for highly constrained devices. Such feature is potentially of considerable interest to multiple applications in the context of IoT. In this section, we briefly describe two applications of OEABE: personal health data sharing and group key management.

5.6.1 Personal Health Data Sharing

Figure 5.6 describes a concrete example of application of our delegation-based CP-ABE scheme in the eHealth scenario. In such a context, the patient carries a number of embedded wearable sensing platforms (for measuring e.g. blood pressure, temperature, pulse) that monitor his health conditions. The patient desires to authorize the personal data produced by IoT devices to be only accessed by a set of appreciated users (e.g. doctor, nurses, patent relatives). To do so, the patient can employ an attribute-based encryption scheme, such as CP-ABE, to encrypt his personal data. In our solution, the IoT devices compute a partial CP-ABE ciphertext from patient's secret data and send it to the semi-trusted parties (e.g. smartphone, tablet, gateway, etc). These latter will execute the most expensive calculations and generate the complete CP-ABE ciphertext. This ciphertext is then stored in a cloud service. The doctor and patient's

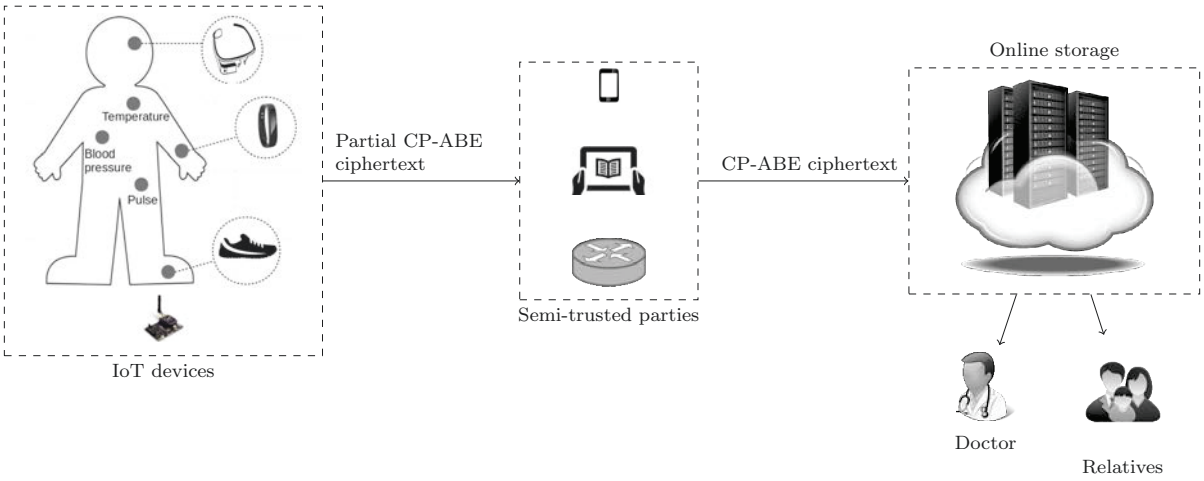


Figure 5.6: An eHealth scenario for our delegation-based CP-ABE scheme

relatives can then request for the encrypted data and decrypt it using their secret keys. We consider the patient's smartphone, tablet or gateway (at his home) as semi-trusted parties because these devices can be stolen or compromised at any moment. As a result, it is safer to conceal the patient's personal health records from these equipment as well.

5.6.2 Group Key Management

As another direct application, the proposal can provide an efficient mechanism to dynamically define groups and subgroups of smart objects according to different combinations of attributes. In fact, by using OEABE, a group controller has to compute only one exponentiation for updating membership in the group. As such, we consider the scenario where different IoT devices (group members - GMs) subscribe to receive data from another resource-constrained IoT device (group controller - GC). In such case, GC has to maintain a shared data encryption key K which will be used to encrypt multicast traffic between him and other GMs. New GMs are provided with this key through a secure OEABE encryption at the time of joining. However, the main challenge is when we need to revoke a subset of group members from future communications. This task consists of distributing the new group key K' to all remaining GMs, so that the revoked users cannot access future exchanges. To do so, GC computes a new access structure T' which is satisfied by the attribute sets of all remaining GMs but not satisfied by the attribute set of any revoked user. Then, GC computes an OEABE encryption on the new key K' using the access structure T'. Of course, the assisted parties are in charge of distributing the final CP-ABE ciphertext to GMs. Therefore, remaining GMs can use their secret keys to recover the new group key K'.

5.7 Summary

In this chapter, a novel delegation-based mechanism has been proposed for the encryption algorithm of CP-ABE. Our solution is secure based on the hardness of the DLP problem in the random oracle without weakening the security of the original CP-ABE scheme. Furthermore, as shown in the performance analysis on an emulated sensor platform and a laptop, our proposal offers significantly less computational and communication overhead at the encryptor side than

existing solutions in related work. Such approach can be easily applied in several applications, such as secure data storage and secure group communication, especially when IoT devices with extremely resource-constrained profile act as a data producer.

Until present, our proposed solutions, i.e. ECKSS, new MIKEY modes and OEABE, allow a device to distribute a secret key or data without consuming much energy. This secret key can be used to set up secure communications. However, in case the two communicating parties desire to negotiate a common key, the previous solutions cannot be considered. In the next chapter, we will address the need for lightweight key agreement in IoT. That is, we define a new efficient key agreement protocol inspired from the idea of proxy re-encryption. The proposed protocol can work even if the communicating parties are highly limited in resources.

Chapter 6

AKAPR: Authenticated Key Agreement Mediated by a Proxy Re-Encryptor for IoT

In the context of IoT, secure communications between devices, even if they are unknown to each other, are essential. As a result, due to limited resources and highly interconnected objects, there is a strong need to design lightweight and scalable key establishment protocols. As shown in Section 3.2, the existing solutions that require the pre-distribution of secret keys (offline key distribution) cannot be envisioned. Indeed, we cannot pre-share every time a common secret key in each device because the number of connected devices composing the network is very important. If the key pre-distribution is not considered, most of the existing schemes require expensive cryptographic operations to establish a session key between entities that do not share common credentials *a priori* such as ECDH-based approaches [174, 182]. As an example, Sciancalepore et al. [174] propose a key agreement protocol with implicit certificates in the context of IoT. Their approach requires four costly operations in order to negotiate a common key between two parties. In addition, the negotiation algorithm always produces the same key for a given couple of devices, which can be vulnerable to known-key attacks. Many other efforts (e.g. in [162], [160]) have been undertaken to reduce the overhead of standard security protocols so that they can fit in low power computing sensor platforms. However, these solutions still require the executions of costly cryptographic operations on such platforms. Apart from the mentioned methods, one can propose to use a server-assisted key distribution scheme to set up a secure communication, as described in Section 3.2.4.2. In such an approach, the aforementioned heavyweight computations can be handled by a resource-rich server. Server-assisted approaches for key establishment protocols have been proposed in this respect for IoT.

In this chapter, we first propose a lightweight proxy re-encryption that uses a symmetric cipher to encrypt data (see Section 2.1.5 for more details on the proxy re-encryption primitive). Our scheme is able to convert a ciphertext from one key to another with no heavyweight computational operations. Second, based on the proposed re-encryption scheme, we build an efficient authenticated key agreement mediated by a proxy re-encryptor, namely AKAPR, for IoT services. The scheme allows us to establish common secret keys between devices, even highly resource-constrained ones (e.g. class 1 devices [37]). Third, we present a formal security validation of AKAPR using ProVerif [30]. The results show that AKAPR provides mutual authentication for participants and ensures the secrecy of the generated session keys.

The rest of this chapter is organized as follows. Section 6.1 presents related work on the server-based key agreement protocols and motivates the use of proxy re-encryption in key establishment mechanisms. Section 6.2 surveys existing approaches on proxy re-encryption. We then

present a novel lightweight proxy re-encryption construction in Section 6.3. Section 6.4 describes in detail our proposed authenticated key agreement AKAPR for IoT. Section 6.5 provides an informal security analysis of AKAPR against common attacks with a formal security validation done by the cryptographic verifier ProVerif [30]. Finally, the conclusion remarks are given in Section 6.6.

6.1 From proxy re-encryption to server-assisted key agreement protocol

In this section, we first complete the related work presented in Section 3.2.4.2 on the server-assisted key establishment protocols. Then, we explain why we apply the idea of proxy re-encryption to propose a new server-assisted key agreement solution for IoT.

Fouladgar et al. [82] introduce an adaption and an extension of TLS (Transport Layer Security) handshake to the Wireless Sensor Network. Their solution describes an ECDH key establishment between a constrained sensor node and an external entity mediated by a partially trusted gateway. Such solution requires only two costly operations on the constrained node side. However, the gateway is able to launch a man-in-the-middle attack and to establish a common Diffie-Hellman key with each party without anyone noticing. Saied et al. [171] propose a lightweight collaborative key agreement based on Diffie-Hellman (DH) key establishment. Their idea is to delegate the heavyweight cryptographic calculation of DH values to the resource-unconstrained trusted proxies in neighborhood. Such mechanism requires a sufficient number of non-colluding neighbors in proximity. Besides, it may seem unpractical, since the two end nodes, which do not share any relation, may not be in possession of a secure established link with those common proxies. Several works attempt to build a common secret key for any two entities using the DTLS (Datagram TLS) protocol in the context of IoT. Their approach is to delegate partially [93, 192] or totally the DTLS handshake [106] to a third party. Such mechanism removes the overhead of intensive calculations for the constrained-devices. However, the third party can read all communications between sensor nodes and the Internet hosts. This feature is not desirable in certain scenarios especially when we do not entirely trust the server.

We remove the mentioned inconvenience by applying a lightweight proxy re-encryption mechanism in our proposed key establishment mechanism. In a proxy re-encryption scheme, the proxy can translate a ciphertext encrypted under one key to another but is not allowed to learn anything on either keys. There exists many PRE schemes in the literature (e.g. in [31], [17], [94, 137]). Their applications are diverse such as encrypted mail forwarding system, secure data storage on semi-trusted servers. As the proxy is generally considered as a rich-resource entity, apart from re-encrypting a ciphertext, it can also assist the communications between resource-constrained devices. As a result, in this chapter, we present another application of PRE to build a server-assisted key agreement protocol where the server is unable to recover not only the secret keys of communicating parties but also the negotiated session keys.

6.2 Existing approaches on proxy re-encryption

We first present several related PRE propositions in the literature. General definitions and the most useful properties of a PRE scheme are specified in Section 2.1.5.

Blaze et al. [31] first proposed the notion of proxy cryptography where Alice (A) can securely delegate her decryption rights or her digital signatures to another party Bob (B) with the help

Type	Typical operations of a proxy re-encryption scheme	Examples
PRA		[31], [17], [94, 137]
PRS		[54, 183]

Table 6.1: Two existing approaches of a proxy re-encryption scheme

Meaning of abbreviations: PRA: Proxy re-encryption schemes that employ asymmetric ciphers; PRS: Proxy re-encryption schemes that employ symmetric ciphers; E: An encryption function; M: Message; pub_X: public key of the entity X; sk_X: secret key of the entity X; PR: the proxy.

of a proxy. Many works on proxy re-encryption schemes have been proposed in the literature. We classify these schemes into two categories as depicted in Table 6.1: (a) Proxy re-encryption schemes that employ asymmetric ciphers (public key cryptography) to encrypt the message and (b) Proxy re-encryption schemes that employ symmetric ciphers to encrypt the message.

Most of the proposed schemes use a public key primitive to encrypt the message. In [31], the authors propose the very first proxy re-encryption scheme based on Elgamal cryptosystem [71]. Alice first generates the ciphertext $C_A = (m.g^r, g^{ar})$ on message m using its pair of public/private key ($sk_A = a$, $pk_A = g^a$). The proxy uses subsequently the re-encryption key $rk_{A\to B} = b/a$ to obtain $g^{br} = (g^{ar})^{rk_{A\to B}}$. Hence, B receives the new ciphertext $C_B = (m.g^r, g^{br})$ encrypted under his secret key. This scheme is bidirectional, transitive and exposed to collusion attacks. As such, the proxy can compute $(rk_{A\to B})^{-1}$ to obtain the re-encryption key in the opposite direction from B to A. In addition, the proxy can combine the two re-encryption keys $rk_{A\to B}$ and $rk_{B\to C}$ to get the valid re-encryption key from A to C ($rk_{A\to C} = a/c = (a/b).(b/c)$). Such property is sometimes unwanted. Furthermore, if the proxy colludes with one party, it is trivial for them both to learn the secret key of the other party. Ateniese et al. [17] proposed an unidirectional pairing-based proxy re-encryption scheme that fixes the above issues. They use a proxy key in the form of $rk_{A\to B} = g^{a/b}$. Such configuration provides non-transitivity and collusion-resistance properties. Indeed, the possession of ($rk_{A\to B} = g^{a/b}, rk_{B\to C} = g^{b/c}$) does not permit the proxy to find out $rk_{A\to C} = g^{a/c}$ due to the Decisional Diffie-Hellman Problem [35]. In addition, colluding with Bob does not help the proxy to discover the secret key of Alice and vice versa since having $g^{a/b}$ and b does not help him to recover a due to the Discrete Logarithm Problem. From then onwards, many schemes based on pairing operations have been proposed including Identity-based (IBE) proxy re-encryption schemes [94, 137]. They are proved to be secure under chosen ciphertext attack (CCA) assumption. Pairing-free proxy re-encryption schemes exist, for example [44, 52], but multiple modular exponentiations are still required.

There are several propositions on proxy re-encryption that employ symmetric ciphers to encrypt the message such as [54, 183]. The main advantage of symmetric cipher proxy re-encryption approach is the lightness of the employed symmetric cryptographic operations in terms of complexity and memory usage. In [54], Cook et al. propose two conversion functions for symmetric ciphers. In their first attempt, they assume that Alice shares with Bob a secret key k_{ab}. In addition, Alice and the proxy must share k_a. Then, Alice sends $E_{k_a}(E_{k_{ab}}(M))$ to the proxy. The proxy decrypts the obtained ciphertext with k_a and sends the result $E_{k_{ab}}(M)$ to Bob. Hence, Bob does not need to share a key with the proxy and yet he can still get the message M. However, this assumes Alice and Bob must always share a common secret. Such assumption is not trivial when there exists a significant number of devices in the network, such as in the context of IoT. In their second attempt (termed as CK to be used in Table 6.2), the authors provide the proxy the key $k_p = k_a \oplus k_b$, built from the secret keys (k_a, k_b) of A and B, respectively. A computes $C = M \oplus k_a$ and sends it to the proxy. The proxy performs the

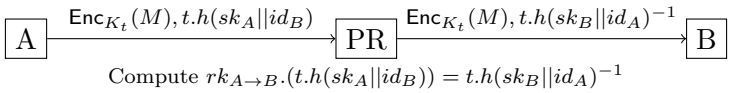


Figure 6.1: Our proposed symmetric cipher proxy re-encryption scheme

conversion by computing $C' = k_p \oplus C = k_b \oplus M$. B can then decrypt C' to get the message using its secret key k_b. This approach is efficient but not secure. Indeed, B can easily retrieve the secret key of A by computing $k_b \oplus C \oplus C' = k_a$. In [183], Syalim et al. propose a pure symmetric cipher proxy re-encryption algorithm. However, this approach requires that A and B share common secret keys *a priori*. Moreover, it is assumed that the proxy cannot collude with any previous users since a compromised user can recover the current encryption key if he/she has the re-encryption key.

6.3 Lightweight Bi-directionnal Proxy re-encryption Scheme with Symmetric Cipher

In this section, the concrete description of our proposed symmetric cipher PRE scheme is given. Then, we compare our proxy re-encryption scheme with related solutions in terms of supported properties and performance.

6.3.1 The proposed proxy re-encryption (PRE) scheme

In this section, we present in detail our proposed symmetric cipher proxy re-encryption. A symmetric cipher proxy re-encryption consists of five algorithms ($\mathsf{KeyGen}, \mathsf{ReKeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Reencrypt}$). In addition, we define ($\mathsf{Enc}, \mathsf{Dec}$) as the encryption and decryption algorithms of a symmetric encryption scheme. A key distribution center (KDC) is responsible for providing keying material. As such, KDC runs the two algorithms KeyGen and $\mathsf{ReKeyGen}$ to generate the needed security parameters. We suppose that Alice (A) desires to delegate the decryption right of a ciphertext C_A encrypted under her secret key to Bob (B) with the help of the proxy (PR). Figure 6.1 describes the message exchanges of our proposed PRE scheme. The procedure is detailed as follows.

- $\mathsf{KeyGen}(k) \to (id_A, id_B, sk_A, sk_B)$: Given the security parameter k, this algorithm outputs the identifiers (id_A, id_B) and the secret keys (sk_A, sk_B) for A and B, respectively.
- $\mathsf{ReKeyGen}(id_A, sk_A, id_B, sk_B) \to rk_{A\to B}$: Given the identifiers and the secret keys of A and B, this algorithm returns the re-encryption key $rk_{A\to B} = (h(sk_A||id_B).h(sk_B||id_A))^{-1}$, where $h : \{0,1\}^* \to \mathbb{Z}_p$ is a hash function that converts a string to a number on $\mathbb{Z}_p$. As we shall see, our construction results in the fact that $rk_{A\to B} = rk_{B\to A}$. This property makes our proxy-encryption scheme *bidirectional* meaning that the proxy only needs to store one re-encryption key to re-encrypt messages from A to B and vice versa.
- $\mathsf{Encrypt}(id_A, sk_A, M, id_B) \to C_A$: Given the identifier of B and a message M, A uses its identifier id_A and its secret key sk_A to generate a ciphertext C_A. A first chooses a random number $t \leftarrow \mathbb{Z}_p$. Then, it generates a symmetric key $K_t \leftarrow KDF(t)$, where KDF is a Key Derivation Function. Finally, it outputs the ciphertext $C_A = (\mathsf{Enc}_{K_t}(M), t.h(sk_A||id_B))$.

Property	BBS [31]	AFG [17]	GG [94]	CH [44]	CK [54]	SN [183]	Ours
Type	PRA	PRA	PRA	PRA	PRS	PRS	PRS
Directionality	bi-d	uni-d	uni-d	bi-d	bi-d	bi-d	bi-d
Non-Interactivity	No	No	Yes	No	No	No	No
Multiple-use	Yes	No	Yes	Yes	Yes	No	No
Non-Transitivity	No	Yes	Yes	No	No	Yes	Yes
Collusion resistance	No	Yes	Yes	No	No	No	Yes
Pairing-free	Yes	No	No	No	Yes	Yes	Yes
Exponentiation-free	No	No	No	No	Yes	Yes	Yes

Table 6.2: Comparison of our scheme and related work

Meaning of abbreviations: bi-d: Bidirectional; uni-d: Unidirectional; PRA: Proxy re-encryption scheme that uses asymmetric ciphers; PRS: Proxy re-encryption scheme that uses symmetric ciphers.

- $\mathsf{Reencrypt}(rk_{A\to B}, C_A) \to C_B$: Upon receiving the ciphertext $C_A = (C_1, C_2)$, PR keeps C_1 unchanged while multiplying C_2 with the re-encryption key $rk_{A\to B}$ to obtain the new ciphertext $C_B = (\mathsf{Enc}_{K_t}(M),\ t.h(sk_B||id_A)^{-1})$.
- $\mathsf{Decrypt}(id_B, sk_B, C_B, id_A) \to M$: Upon receiving $C_B = (C_1', C_2')$ $= (\mathsf{Enc}_{K_t}(M),\ t.h(sk_B||id_A)^{-1})$, B first calculates the value of $l = h(sk_B||id_A)$ from its secret key and the identifier of A. Then, it obtains the value of t by multiplying l to C_2'. From t, B generates the symmetric key $K_t \leftarrow KDF(t)$. Then, it gets the message M by decrypting C_1' using the generated key K_t: $M = \mathsf{Dec}_{K_t}(\mathsf{Enc}_{K_t}(M))$.

Correctness. The correctness of our proposed scheme is straightforward.

6.3.2 Comparison of our PRE scheme to related work

In Table 6.2, we compare several proxy re-encryption schemes in related work with our scheme based on the properties provided in Section 2.1.5.1. In comparing with asymmetric cipher PRE schemes, our scheme is much lighter in terms of computational cost. Indeed, the proposed construction does not necessitate any pairing or exponentiation operation. On the other hand, while providing equivalent performance compared with symmetric cipher proxy re-encryption schemes, our scheme is more robust against attacks from compromised receiver, semi-honest proxy and their corporation. We argue that our scheme provides most of the desirable properties as described in the following.

First, our scheme is *bidirectional* since $rk_{A\to B} = rk_{B\to A}$. This can be an advantage in the considered scenario (e.g. IoT) where the proxy has to store only one proxy key for any pair of devices. Second, in our construction, only KDC can provide the re-encryption key because it is generated from the secret keys of participants. This property makes our scheme *interactive*. However, the scheme can be made partially *non-interactive* such that A and B can negotiate a new proxy re-encryption key even when KDC is offline. In fact, A may generate a new secret key sk_A' and compute $k_1 = h(sk_A'||id_B).h(sk_A||id_B)$. B generates also a new secret key sk_B' and compute $k_2 = h(sk_B'||id_A).h(sk_B||id_A)$. k_1, k_2 are then sent to the proxy. The latter can now obtain the new proxy re-encryption key by computing $1/(k_1.k_2.rk_{A\to B})$ in $\mathbb{Z}_p$. Finally, as each proxy key is generated specifically for a pair of users, the proxy can only re-encrypt the ciphertext a *single* time. Such construction makes our scheme unconditionally *non-transitive* and *collusion-resistant*. Indeed, providing $rk_{A\to B} = (h(sk_A||id_B).h(sk_B||id_A))^{-1}$ and $rk_{B\to C} = (h(sk_B||id_C).h(sk_C||id_B))^{-1}$, the only way that the proxy can get $rk_{A\to C}$ is to have the secret keys of A and C due to one-way property of hash function. Even if B colludes with the proxy, they only have the value of $h(sk_A||id_B)$ which is only used in the communication

between A and B. Such knowledge will not help them to find out A's secret key sk_A. In addition, to obtain $rk_{A\to C}$, they still need both the secret keys of A and C.

6.4 Lightweight Authenticated and Mediated Key Agreement for IoT

In this section, we present the application of our PRE scheme presented in Section 6.3 to obtain a very lightweight key establishment mechanism. Our protocol is relevant even with highly resource-constrained devices in the context of IoT. The first subsection presents the network architecture and our considered scenarios. The second subsection provides the security assumptions needed for the description of the protocol. Then, we describe concretely the message exchanges of our proposal.

6.4.1 Network architecture and scenario description

Figure 6.2 describes the network architecture of our proposal. The considered network of *things* consists of a number of tiny nodes communicating with each other and with an unconstrained resource border router (or gateway). The gateway is the bridge between the sensor network and the outside world. It may take part in the communication between two entities in a passive (transparent to the communicating parties) or active (as a mediator in the communication process) manners.

Our key establishment protocol involves the four following actors:

- Two parties: an Initiator (I) and a Responder (R), which respectively initiates the communication and responds to incoming requests.
- A partial trusted party, named as Delegatee (DG), which is responsible for assisting the key establishment process between I and R. In fact, DG is provided with a re-encryption key that allows it to translate the ciphertext from I to R. In addition, it is considered as a *semi-trusted* party that acts and returns correct results according to the protocol but can be curious on transmitted messages.
- A trusted Key Distribution Center (KDC), which is responsible for generating keying material and acts as the root of trust of the whole system. Besides, KDC is also in charge of delegation credential management and distribution.

In our considered scenario, I and R can be both resource-constrained devices. At the beginning, KDC provisions the keying material for all users on the system. Hence it can stay offline until the security parameters need to be refreshed. On the other hand, DG must stay online and participate actively in the key establishment procedure. Our motivation is that DG acts as a partially-trusted third party helping the constrained devices to negotiate session keys without obtaining any knowledge about these keys.

As depicted in Figure 6.2, the initiator can be an external entity requesting for information of the Responder - a sensor platform device lying in a Wireless Sensor Networks (WSN). The key negotiation process is assisted by DG. In addition, when I and R are in the same WSN, DG can provide the delegation keys for the border router (or gateway) so that the key agreement process can be done locally. Note that the gateway is also considered *semi-trusted* as a consequence of which it only knows the delegation keys and is not able to recover the secret keys of I and R. We provide more details on the security analysis of our proposal in Section 6.5.

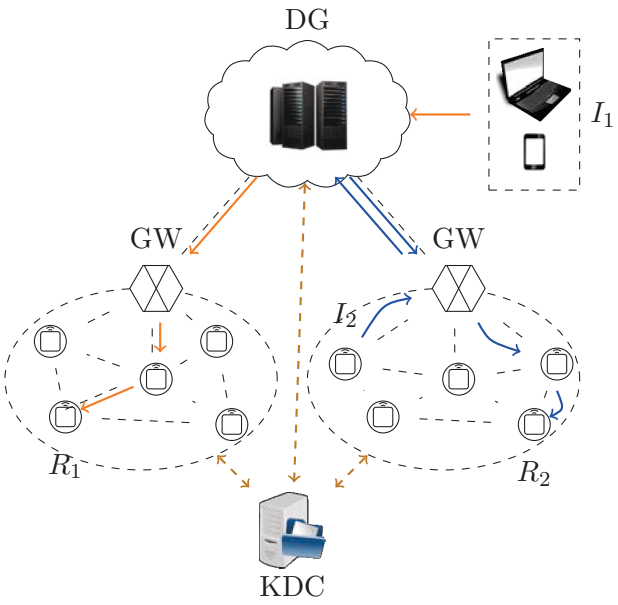


Figure 6.2: Network architecture and considered scenarios

→: KDC provides keying material for all actors in the system.

Examples of scenario: (1) →: The external user I_1 initiates a key agreement process (mediated by DG) with the resource-constrained sensor node R_1; (2) →: Two unknown resource-constrained nodes (I_2 and R_2) initiate a key agreement process with the help of DG and then GW.

6.4.2 Security assumptions and notations

We suppose that I and R possess their own secret keys (sk_I and sk_R, accordingly). However, they do not have any common secrets *a priori*. On the other hand, DG shares with each communicating entity X a secret symmetric key K_{xd} which is employed to protect the integrity of the traffic between X and DG. As a result, DG shares the secret keys K_{id} and the secret key K_{rd} with I and R, respectively. In addition, we use an incremental counter in both communicating parties to mitigate the replay attacks. For example, we maintain the counter CT_{IR} in I's side for all exchanges with R. If this is the first time that I communicates with R, CT_{IR} is set to 0. It is increased by 1 after every successful key agreement. Furthermore, for each entity X, we denote its identifier as id_X. Such identifier must be unique for each entity. We also define $(\mathsf{Enc}, \mathsf{Dec})$ as the encryption and decryption algorithms of a symmetric encryption scheme. While, $(\mathsf{AEnc}, \mathsf{ADec})$ is an authenticated encryption algorithm such that $\mathsf{AEnc}_{K_1,K_2}(M) = \mathsf{Enc}_{K_1}(M)||MAC_{K_2}(\mathsf{Enc}_{K_1}(M))$ and $\mathsf{ADec}_{K_1,K_2}(\mathsf{Enc}_{K_1}(M)||MAC_{K_2}(\mathsf{Enc}_{K_1}(M))) = M$, for each message M and two secret keys K_1, K_2. Each key agreement exchange of order i between I and R (Message i, for $i = 1, 2, 3$) has two components ED_i and $MAC_i(K)$. ED_i defines the appended security parameters and the encrypted data, while $MAC_i(K)$ denotes the MAC of ED_i computed with the symmetric key K.

In addition, two hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $H : \{0,1\}^* \rightarrow \{0,1\}^n$ are also defined, where n is an integer number generated from the input security level. These functions are modeled as random oracles [27]. Such oracle produces a random value for each new query. Of course, if an input is asked twice, identical answers are returned. In this work, we also use a Key Derivation Function (KDF) for generating a symmetric key. KDF is based on a solid pseudorandom number generator (PRNG) (e.g. in [21]). This function is initialized with several secret values, called seeds. An attacker with the knowledge of PRNG output should not be able

to guess the seeds other than by exhaustive guessing.

6.4.3 AKAPR protocol description

The proposed key establishment protocol AKAPR consists of four messages as depicted in Figure 6.3. The key negotiation process is mediated by DG. The detailed description of the key agreement process is given as follows.

– **Message 1 from I to DG**: To start a new session, I first increases CT_{IR} by one, where CT_{IR} denotes the current counter of I for all communications with R. CT_{IR} is set to zero if this is the first time I communicates with R. Next, it generates a session identifier SID at random (e.g. $SID = H(id_I||id_R||w)$, where w is randomly chosen in $\mathbb{Z}_p$). Then, I chooses at random two fresh numbers N_i and t from $\mathbb{Z}_p$. The ephemeral authentication keys $AK = (AK_e, AK_a)$ are then generated from id_I, id_R and t using a key derivation function (KDF). To construct the Message 1, I concatenates the session identifier SID, its identifier id_I and R's identifier id_R to (N_i, CT_{IR}). The concatenation is then encrypted using the algorithm AEnc. As we shall see, the resulting ciphertext is the encryption and MAC of the concatenation by the pair of keys (AK_e, AK_a). This guarantees that the attacker (including DG) cannot modify the encrypted text of the concatenation. Second, I masks the value of t by multiplying it with the hashed value $h(sk_I||id_R)$, where sk_I is the secret key of I. As we shall see, the result of such multiplication is randomly distributed in $\mathbb{Z}_p$ since the two used operands are also randomly generated in $\mathbb{Z}_p$. Then, the first five components of the message $(SID, id_I, id_R, \mathsf{AEnc}_{AK_e,AK_a}(id_I||id_R||N_i||CT_{IR}), t.h(sk_I||id_I))$ is completed by a MAC computed with K_{id}, to form the Message 1.

– **Message 2 from DG to R**: Upon receiving the Message 1 from I, DG first verifies that SID is fresh. We suppose that DG stores a list of SID values for each pair of I and R. Next, DG validates that the message has not been modified by an attacker by verifying its MAC using K_{id}. If the verification holds, DG is also certain that the Message 1 has not been replayed. Then, it modifies the fifth component of the encryption part (ED_1) in the Message 1 with the delegation key dk_{IR}. Indeed, it multiplies $t.h(sk_I||id_R)$ with $dk_{IR} = (h(sk_I||id_R).h(sk_R||id_I))^{-1}$ to obtain $t.h(sk_R||id_I)^{-1}$. DG now concatenates the obtained result to the first four components of the Message 1 to form ED_2. The encryption part of the Message 2, $ED_2 = (SID, id_I, id_R, \mathsf{AEnc}_{AK_e,AK_a}(id_I||id_R||N_i||CT_{IR}),\ t.h(sk_R||id_I)^{-1})$, is then appended with a MAC computed with K_{rd}.

– **Message 3 from R to I**: When receiving the Message 2 from DG, R first verifies the authenticity of the message by employing its shared key with DG, K_{rd}. Then, by multiplying the hashed value of its secret key sk_R and the identifier of I (id_I) to the fifth part of ED_2, $(t.h(sk_R||id_I)^{-1})$, it obtains t, which is a number on $\mathbb{Z}_p$. From t, I generates the secret ephemeral authentication keys $AK = KDF(id_I, id_R, t) = (AK_e, AK_a)$. Next, it decrypts the fourth part of the Message 2 using (AK_e, AK_a) to get the value of $(id_I, id_R, N_{i1}, CT')$. It verifies subsequently that CT' is superior or equal to its counter number CT_{RI} to be sure about the freshness of the Message 2 (see Section 6.5.1). The counter value of R, CT_{RI}, is now set to the value of CT'. To construct the Message 3, R first chooses randomly N_r from $\mathbb{Z}_p$. Next, it increases CT_{RI} by one. R now encrypts the concatenation of $(SID, id_R, id_I, N_{i1}, t, N_r)$ with the generated key AK_e. The encrypted data is then appended with the session identifier SID to obtain the encryption part. The latter is finally integrity protected with a MAC based on the generated secret key AK_a.

– **Message 4 from I to R**: After receiving the Message 3 from R, I first approves the authenticity of the message using AK_a. Next, it decrypts the encrypted part by employing the secret

key AK_e to get the values of $(SID_1, id_R, id_I, N_{i2}, t_1, N_{r1}, CT_{RI1})$. I verifies that (SID_1, N_{i2}, t_1) is equal to the generated values (SID, N_i, t). It also verifies that $CT_{RI1} = CT_{IR} + 1$. Finally, the session keys are generated from the values (CT_{RI1}, N_i, N_{r1}) and the identifiers of I and R: $K_s = KDF(CT_{RI1}, id_I, id_R, N_i, N_{r1})$. I macs the concatenation of $(SID, id_I, id_R, N_i, N_{r1})$ using the session key K_s and sends directly to R the hashed value appended with the session identifier SID as a key confirmation message.

Upon receiving the Message 4, R first generates the session key K_s from the identifiers (id_I, id_R), the obtained N_{i1} in the Message 2, the generated value N_r and its counter number CT_{RI}. Then, it calculates a MAC from the concatenation of $(SID, id_I, id_R, N_{i1}, N_r)$ using the generated session key K_s. If the latter is identical to the received Message 4, I and R can now start secure communications, e.g. using standard security protocols such as DTLS-PSK [75] where the pre-shared keys are provided beforehand by our proposal.

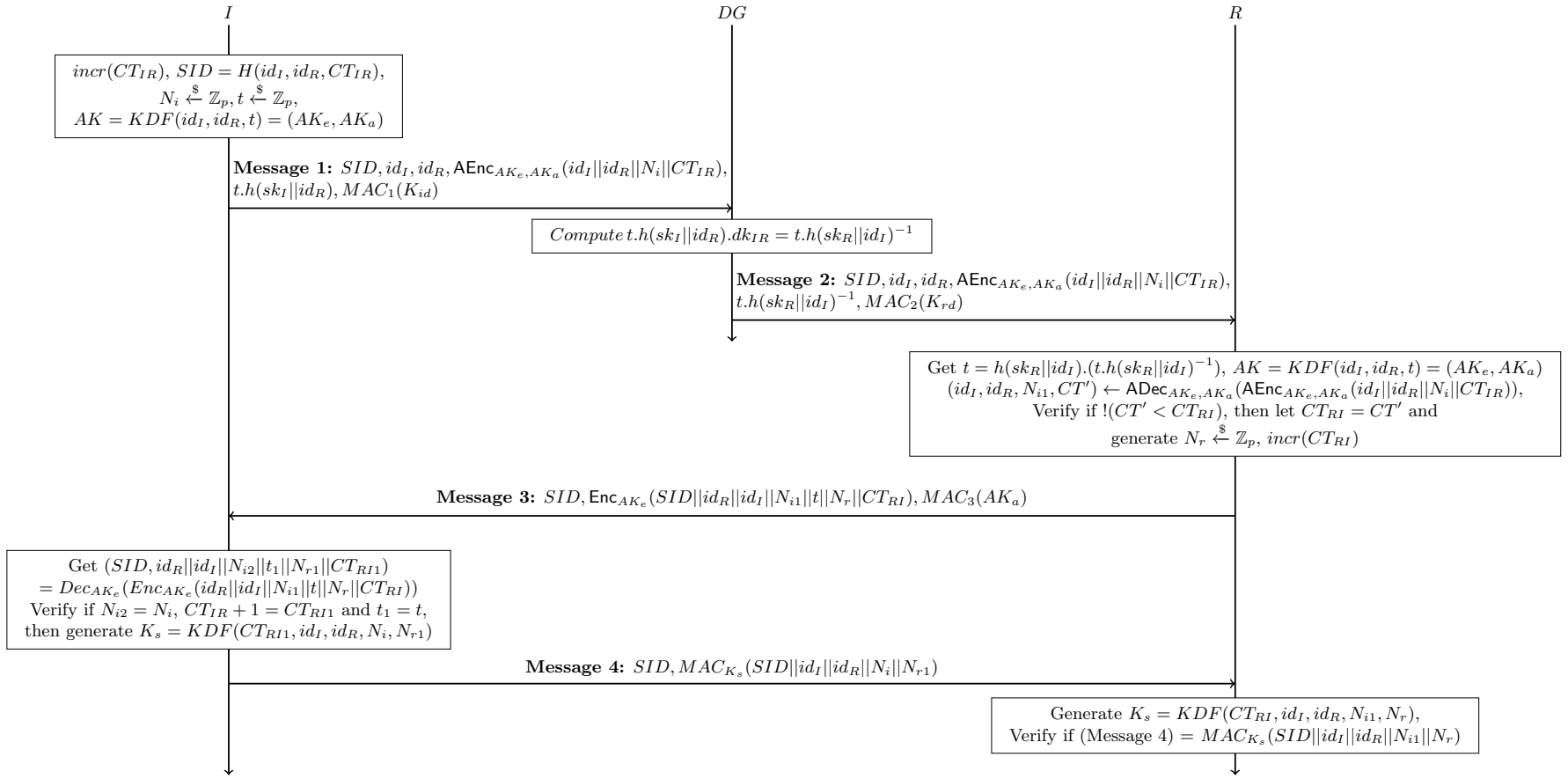


Figure 6.3: Lightweight Secure Key Agreement for IoT

Meaning of abbreviations: $dk_{IR} = (h(sk_I||id_R).h(sk_R||id_I))^{-1}$; $incr(CT)$: $CT = CT + 1$; Message $i = (ED_i, MAC_i(K))$ for $i = 1, 2, 3$, e.g. $ED_1 = (id_I, id_R, \mathsf{AEnc}_{AK_e,AK_a}(id_I||id_R||N_i||CT_{IR}), t.h(sk_I||id_R))$, $MAC_1(K_{id}) = MAC_{K_{id}}(ED_1)$.

Security keys needed for each participant: I (CT_{IR}, sk_I, K_{id}), DG $(K_{id}, K_{rd}, dk_{IR})$, R (CT_{RI}, sk_R, K_{rd}).

6.5 Security analysis of AKAPR

In this section, we first provide an informal security analysis of AKAPR by describing its resistance against common security attacks. Then, we validate the security of AKAPR using the cryptographic protocol analyzing tool ProVerif [30].

6.5.1 Resistance against attacks

Our proposal is resistant to the following attacks:

- Replay attack: This attack is mitigated by the used counter numbers (CT_{IR}, CT_{RI}) and the random numbers (N_i, N_r) at run-time. The replays of messages 1 and 2 are detected thanks to the counter numbers (CT_{IR}, CT_{RI}). Indeed, for any new session, I increases the value of CT_{IR} by one. This value is then encrypted inside the Message 1. Upon receiving the Message 2, R can be sure about the freshness of this message by comparing its counter number CT_{RI} with CT'. If the latter is inferior than CT_{RI} then the message is detected as replayed. On the other hand, the freshness of the Messages 3 and 4 are assured by the pair of random values N_r and N_i since they are newly generated for each session. DG can also prevent replay attacks by keeping the session identifier SID. Because CT_{IR} is increased by one for each communication, the latter will vary in each session.

- Denial-of-service attack (DoS): The Dos attacks aiming at each participant are reduced in our proposal because all exchanges between parties are authenticated. Indeed, each message is appended with an authentication code (MAC) that permits the receiving party to verify if the message is altered during the transmission. Further operations are canceled if the verification fails.

- Man in the middle attack (MITM): The attacker cannot impersonate any party in our protocol since each message is protected by the secret keys that are unknown to him. As such, the Message 1 and the Message 2 are encrypted-then-maced by (AK_e, K_{id}) and (AK_e, K_{rd}), respectively. The Message 3 is encrypted then maced by the ephemeral secret keys $AK = (AK_e, AK_a)$, while, the Message 4 is protected by the new generated session key K_s.

- Key escrow attack: DG is a blind participant in the key agreement procedure. It aids the key negotiation without having any knowledge on the agreed session key and the secret keys of I and R. Indeed, although DG participates in the key negotiation process, it possesses only the delegation key $dk_{IR} = (h(sk_I||id_R).h(sk_R||id_I))^{-1})$ for each pair of Initiator and Responder. In addition, without knowing the secret key of I and R, DG cannot distinguish dk_{IR}, $t.h(sk_R||id_I)$ and $t.h(sk_I||id_R)^{-1}$ from a random number on $\mathbb{Z}_p$. The only actor that can intercept message exchanges between I, R and DG is the KDC. However, we have assumed that KDC is a totally trusted party which is responsible for the keying material generations and stays offline.

- Collusion attack: This feature inherits the collusion-resistance property of the proposed PRE scheme in Section 6.3. As such, even if DG colludes with one party, it cannot retrieve the secret key of the other party thanks to the one-way property of the hash function h. Indeed, if R collaborates with DG, they will get the values of t, AK, N_i and N_r. However, only the messages dedicated for R of I are affected. In fact, DG can only have the value of $h(sk_I||id_R)$ which does not help him to find the secret key of I, sk_I. If DG colludes with I, I can then decrypt itself the Message 3, which contains no secret information of R. The

colluding parties can achieve the value of $h(sk_R||id_I)$. However, they are unable to guess the secret sk_R of R thanks again to the one-way property of hash functions.

The above security attacks except the MITM attacks, are usually impossible to be detected by an automatic software verifier (e.g. ProVerif [30]). In practice, the latter is used to verify if the essential security properties, such as mutual authentication and secret key protection, are provided in the testing cryptographic protocol. We provide more details on such software verification in the next section.

6.5.2 Formal security validation with ProVerif

In this section, we present a formal verification of AKAPR using ProVerif [30]. Our verification ensures that the proposed protocol provides the secrecy of the generated session keys and the authentication of participants.

ProVerif is an automatic verifier for cryptographic protocols defined in the Dolev-Yao model [63]. In such model, the attacker is an *active* eavesdropper, capable of obtaining any message passing in the network, initiating a conversation with any other users and impersonating as a legitimate receiver. It is only limited by the restrictions of the cryptographic methods used. In other words, the cryptographic primitives is considered idealized in the sense that they are unbreakable without knowing the employed secret keys.

In Listing 1.2, we provide the ProVerif verification code of our protocol AKAPR while respecting the description written in Section 6.4.3. A protocol description in ProVerif is divided into three parts: the *declarations*, the *process macros* and the *main process*. As described in Lines 1-44, the declaration part consists of the *user types*, the *security properties*, the cryptographic primitive *functions* and the list of defined *events* and *queries*. We define the types, the communication channel and the identifiers of the participating parties in Lines 1-6. The tables specified in Lines 8-11 are employed to model the storage of keys in a server. Only I, R and DG can use these tables to get the associations between host names and keys. Note that we use the table $\mathsf{ctr}(\mathsf{host}, \mathsf{Zp})$ to store the counter value of a specific host. To describe the synchronization of the counter values in both sides (I and R), we model only the ideal situation where there is no failed session between them. In such case, the counter values of I and R are equal. The detailed synchronization process is described in Lines 52-54, 68, 87 and 90 of Listing 1.2. Furthermore, the secrecy assumptions are specified in Lines 13-16. For example, sk_I and K_id define the secret key of I and its shared key with DG. These keys are kept secret to the attackers. Then, Lines 18-30 describe the cryptographic functions needed in our protocol. For example, the function $(\mathsf{kdf\_h}(\mathsf{Zp}, \mathsf{host}) : \mathsf{Zp})$ generates the hashed value $\mathsf{h}(\mathsf{aZpNumber}||\mathsf{aHostName})$. On the other hand, the function $(\mathsf{mask}(\mathsf{Zp}, \mathsf{Zp}) : \mathsf{Zp})$ denotes a simple multiplication on $\mathbb{Z}_p$. Other functions are self-explained according to the protocol specification as depicted in Section 6.4.3. As we shall see, the correctness of the re-encryption process is modeled in Lines 32-35 based on the commutativity of multiplication on $\mathbb{Z}_p$. Finally, we introduce a list of events and queries in Lines 37-44. For example, the event $\mathsf{beginRkey}(\mathsf{host}, \mathsf{host}, \mathsf{key})$ represents the request from I to create a trusted session with R. The defined events play as reference points for the protocol execution order.

In ProVerif, we can ensure the authentication by testing the correspondence assertions between the aforementioned events. Indeed, we verify the mutual authentication between I and R using queries defined in Lines 43-44. For example, the first query in Line 43 says that, if event $\mathsf{endRkey}(\mathsf{host}, \mathsf{host}, \mathsf{key})$ occurs then, event $\mathsf{beginRkey}(\mathsf{host}, \mathsf{host}, \mathsf{key})$ must have occurred before. Furthermore, our second interest of this protocol modeling is to verify the secrecy of the negotiated session key K_s. To do so, I and R choose a random number in each side and output the ciphertext encrypted with K_s. Then, they challenge the attacker to find the encrypted data

by the queries specified in Lines 41-42. The attacker can obtain the underlying data if and only if having the secret key K_s since the cryptographic primitives are considered as black-boxes in ProVerif.

The second part of AKAPR ProVerif program describes the process macros for participants I, R and DG. They are specified in Lines 46-74, Lines 76-99 and Lines 101-109, respectively. These macros present the operations of I, R and DG during AKAPR execution. Note that in lines 57, 71, 86 and 98, we insert the events that we specified earlier. The other four process macros processDK, processKD, processK and processCTR fill the four tables of secret keys defined in Lines 8-11.

In the last part of Listing 1.2, we specify the main process (Lines 127-141) of the AKAPR ProVerif program. It instantiates the keying materials needed, inserts these keys to the right tables and runs the defined macros unlimited times.

The output of the program when running with ProVerif is summarized in Listing 1.1.

```
1 RESULT event(endIkey(x_72,y_73,z)) ==> event(beginIkey(x_72,y_73,z)) is true.
2 RESULT event(endRkey(x_3724,y_3725,z_3726)) ==> event(beginRkey(x_3724,y_3725,
    z_3726)) is true.
3 RESULT not attacker(secretI[!1 = v_7305]) is true.
4 RESULT not attacker(secretR[]) is true.
```

Listing 6.1: AKAPR verification results

The result in Lines 1-2 informs us that AKAPR provides mutual authentication of the two participants I and R. As such, the proved correspondence property in Line 1 implies that R authenticates I by the fact that I can correctly retrieve the session key K_s. On the other hand, Line 2 shows that I authenticates R since the latter can obtain the correct ephemeral key AK after receiving the Message 2. In addition, Lines 3-4 show the results of the queries not attacker(secretI[]) and not attacker(secretR[]) returned by ProVerif. As we shall see, these results are true, which means that the secrecy of the random values secretI and secretR are preserved by the protocol. In other words, the secrecy of the session key generated by AKAPR is also preserved.

The above ProVerif verification has several limitations. Indeed, in ProVerif, the hypothesis of perfect cryptography is considered, meaning that the only way to decrypt an encrypted message is to use the right secret key. Besides, in Line 18-35, we have to model the modular multiplication and its commutative property required in the re-encryption process by defining several new functions. This is necessary because real modular multiplication cannot be handled by ProVerif. In fact, ProVerif verification might not terminate when dealing with protocols that use algebraic operations such as modular multiplication or Exclusive-or. In addition, several security protocols that are conceptually safe, but are found flawed when considering algebraic properties as described in [124]. As a result, one can complete the above formal verification using other tools such as CryptoVerif [42], CL-Atse [191] or OFMC [23], which support most of algebraic properties and provide more realistic assumptions, e.g. the hypothesis of perfect cryptography is not required.

6.6 Summary

In this chapter, we first introduced a novel proxy re-encryption scheme that requires only symmetric cipher to encrypt data. We showed that although our scheme is bidirectional and single-use, it provides the most important features: non-transitivity and collusion-resistance. Furthermore, the scheme is much more efficient when compared with related solutions that use asymmetric approaches. Second, we proposed a novel authenticated delegation-based and lightweight

key agreement protocol to be used in the Internet of Things. This protocol is built upon the proposed proxy re-encryption scheme. The security of our solution has been formally validated by ProVerif. In addition, thanks to the used symmetric primitives, the proposed key agreement mechanism is very lightweight since it does not require any expensive cryptographic operations such as pairing operation or modular exponentiation. The proposed protocol can be applied even to class 1 devices having extremely resource-constrained profile.

Chapter 7

Conclusion and Perspectives

This thesis aims to propose lightweight cryptographic primitives and security mechanisms for the Internet of Things. Our objective is to guarantee secure communications between IoT devices, while taking into account their resource-constrained nature. For this purpose, we conducted at first a comprehensive survey on existing IoT security protocols and techniques, and we demonstrated their weaknesses and limitations. Once we have identified areas of research at best opportunities, we proposed several contributions that can be regrouped into two main axes: efficient asymmetric encryption algorithms and lightweight key establishment protocols. What follows is a summary of our thesis contributions.

Thesis summary

In chapter 4, we introduced our two contributions. The first contribution consists of a new lightweight signcryption scheme, named ECKSS. Our proposed scheme inherits from the efficiency of the signcryption primitive by combining simultaneously public key encryption and digital signature. In addition, our scheme does not rely on the public key infrastructure to provide the authentication for each entity in the system. We have formally proved the security of ECKSS in the random oracle. The scheme ensures the confidentiality and the unforgeability of communication channels, while being more efficient in terms of computation and communication overhead than existing signcryption solutions. The second contribution extends the standard key management protocol MIKEY with an application of ECKSS. We proposed two new key distribution modes for MIKEY. The experimental results on the Openmote sensor platform validate the efficiency of our proposed MIKEY modes, in particular with respect to other existing MIKEY methods.

In chapter 5, we presented OEABE, a delegation-based mechanism to accelerate the encryption of the Ciphertext-Policy Attribute-Based Encryption algorithm. Our mechanism offloads the most expensive operations of the CP-ABE encryption phase to a semi-trusted server. OEABE allows the constrained devices to authorize the access to their data in a secure and lightweight manner. We conducted a performance evaluation of our proposal on the emulated Wismote sensor platform, in addition to a laptop. The results demonstrated that OEABE scales well even when the access tree becomes complex and large in terms of the number of attributes.

In chapter 6, we introduced AKAPR, a novel lightweight key agreement mediated by a proxy re-encryptor particularly suited to resource limitations in the Internet of Things. AKAPR allows two IoT devices even with highly constrained profile to establish a secure session key with

no asymmetric techniques. The scheme inherits from the efficiency of our proposed proxy re-encryption scheme by refraining from performing any expensive cryptographic operation at both communicating parties. Besides, AKAPR was formally validated using the cryptographic verifier ProVerif. The verification results showed that AKAPR provides mutual authentication for the two comminicating parties and secrecy of the negotiated session key.

Perspectives

As future perspectives, ECKSS+ could be formally proved to demonstrate that it is confidentially and unforgeably secure in the insider model. As such, even if ECKSS is more efficient than ECKSS+, ECKSS is only secure in the outsider model since the scheme does not provide the perfect forward secrecy property. Such feature might be useful in certain scenarios, for example, in an environment where the loss of data and physical attacks are frequent. In addition, a more thorough performance assessment of ECKSS and its MIKEY-based key distributions on other sensor platforms should be done. For instance, the testing scenario could be more realistic in terms of network topology, e.g., keys should be securely distributed to two communicating parties separated by a relevant number of intermediate hops the two communicating parties.

In the same way, the security of OEABE could be validated mathematically in the random oracle model by providing a formal security proof via a sequence of games. In addition, it would be more interesting to conduct the experimental evaluation of OEABE on a real sensor platform, such as OpenMote, and elaborate a real cloud service playing the role of the Delegatee. In such scenario, we can get more practical results on the performance of OEABE, while considering the communication overhead of our architecture. For future extension, one may apply the idea of OEABE on other CP-ABE variants, for example, using OEABE in a multiple-authority CP-ABE scheme to gain more scalability. However, in such scenario, the trade-off between the scalability and the communication overhead should be considered since a complex architecture also implies high latency requirements for the management of security parameters.

Regarding AKAPR, the security validation could be consolidated in both formal and mathematical aspects. Indeed, as ProVerif considers a less realistic assumption on perfect cryptography, a formal security validation in the computational model by using other automatic cryptographic verifiers (e.g. CryptoVerif, CL-Atse, OFMC) could be performed. In these verifiers, the hypothesis of black-boxes cryptographic primitives is removed. Additionally, they support algebraic operations such as modular multiplication or Exclusive-or. With respect to the mathematical proof, we can formally define each composition of AKAPR messages and provide a formal proof in the random oracle model. Last but not least, an experimental evaluation of the proposed key agreement on real constrained sensor platforms should be performed in order to confirm the efficiency of AKAPR compared to related work.

As final point, further research and consideration should be given to lightweight quantum-resistant primitives, such as NTRU and HIMMO. Indeed, with the advent of quantum-computing, most of the widely used cryptographic primitives and protocols, such as RSA, ECDH, ECDSA and other DLP-based cryptosystems, become breakable. As a result, post-quantum cryptographic primitives should be more taken into great consideration in the context of IoT while paying attention to providing acceptable performance on resource-constrained IoT devices.

Author Publications

Journals

- Kim Thuat Nguyen, Maryline Laurent, Nouha Oualha, *Survey on secure communication protocols for the Internet of Things*, Ad-Hoc Networks journal, Available online 9 February 2015, ISSN 1570-8705, http://dx.doi.org/10.1016/j.adhoc.2015.01.006.

- Kim Thuat Nguyen, Nouha Oualha, Maryline Laurent, *Securely Outsourcing the Ciphertext-policy Attribut-Based Encryption*, World Wide Web: Internet and Web Information Systems journal (WWWJ), submitted the 3rd October 2016.

Conferences

- Kim Thuat Nguyen, Nouha Oualha, Maryline Laurent, *Authenticated Key Agreement mediated by a Proxy Re-encryptor for the Internet of Things*, 21st European Symposium on Research in Computer Security (ESORICS 2016), Heraklion, September 2016.

- Kim Thuat Nguyen, Nouha Oualha, Maryline Laurent, *Novel Lightweight Signcryption-based Key Distribution Method for MIKEY*, the 10th International Conference on Information Security Theory and Practice (WISTP 2016), Heraklion, September 2016.

- Kim Thuat Nguyen, Nouha Oualha, *Lightweight Attribute-Based Encryption for the Internet of Things*, 25th International Conference on Computer Communication and Networks (ICCCN 2016), Hawaii, August 2016.

- Kim Thuat Nguyen, Nouha Oualha, Maryline Laurent, *Lightweight Certificateless and Provably-Secure Signcryptosystem for the Internet of Things*, 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM 2015), Helsinki, August 2015.

Annexes

```
type host.
type key.
type mkey.
type Zp.
free c: channel.
free I, R: host.

table msKey(host, Zp).
table transMsKey(host, host, Zp).
table keys(host, mkey).
table ctr(host, Zp).

not attacker(new K_id).
not attacker(new K_rd).
not attacker(new sk_I).
not attacker(new sk_R).

fun addone(Zp): Zp.
fun enc(bitstring,key): bitstring.
reduc forall x: bitstring, y: key; denc(enc(x,y),y) = x.
fun mac(bitstring, mkey): bitstring.
fun kdf_AK(host, host, Zp): key.
fun mkdf_AK(host, host, Zp): mkey.
fun kdf_h(Zp, host): Zp.
fun kdf_fn(Zp, host, host, Zp, Zp): key.
fun mkdf_fn(Zp, host, host, Zp, Zp): mkey.
fun mask(Zp, Zp): Zp.
fun kdf_rk(Zp, Zp): Zp.
fun inv(Zp): Zp.
fun sid_gen(host, host, Zp): bitstring.

reduc forall r:Zp, k1:Zp, k2:Zp;
  reenc(mask(r, k1), kdf_rk(k1, k2)) = mask(r, inv(k2)).
reduc forall r:Zp, k:Zp;
  unmask(mask(r, inv(k)), k) = r.

event beginIkey(host, host, key).
event endIkey(host, host, key).
event beginRkey(host, host, key).
```

```
event endRkey(host, host, key).

query attacker(new secretI);
  attacker(new secretR).
query x: host, y: host, z: key; event(endRkey(x, y, z)) ==> event(
    beginRkey(x, y, z)).
query x: host, y: host, z: key; event(endIkey(x, y, z)) ==> event(
    beginIkey(x, y, z)).

let processI =
  new secretI: bitstring;
  in(c, hostR: host);
  get keys(=I, kid) in
  new Ni: Zp;
  new t: Zp;
  get ctr(=I, ct_i0) in
  let ct_i: Zp = addone(ct_i0) in
  insert ctr(I, ct_i);
  let AK_e: key = kdf_AK(I, hostR, t) in
  let AK_a: mkey = mkdf_AK(I, hostR, t) in
  event beginRkey(I, hostR, AK_e);
  new w: Zp; let SID: bitstring = sid_gen(I, hostR, w) in
  let e1: bitstring = enc((I, hostR, Ni, ct_i), AK_e) in
  let me1: bitstring = mac(e1, AK_a) in
  get msKey(=I, ki) in
  let tb: Zp = mask(t, kdf_h(ki, hostR)) in
  let mac1: bitstring = mac((SID, I, hostR, e1, me1, tb), kid) in
  out (c, (SID, I, hostR, e1, me1, tb, mac1));
  in(c, (=SID, e2: bitstring, mac2: bitstring));
  if mac((SID, e2), AK_a) = mac2 then
  let (=SID, =hostR, =I, =Ni, =t, Nrp: Zp, ct_rp: Zp) = dec(e2, AK_e
    ) in
  if (ct_rp = addone(ct_i)) then
  let K_s: key = kdf_fn(ct_rp, I, hostR, Ni, Nrp) in
  let m_Ks: mkey = mkdf_fn(ct_rp, I, hostR, Ni, Nrp) in
  event beginIkey(I, hostR, K_s);
  let mac3: bitstring = mac((SID, I, hostR, Ni, Nrp), m_Ks) in
  out(c, (SID, mac3));
  out(c, enc(secretI, K_s)).

let processR =
  new secretR: bitstring;
  in(c, (SID: bitstring, hostI: host, =R, e4: bitstring, me4:
    bitstring, tbp: Zp, mac4: bitstring));
  get keys(=R, krd) in
  if mac((SID, hostI, R, e4, me4, tbp), krd) = mac4 then
  get msKey(=R, kr) in
  let tp: Zp = unmask(tbp, kdf_h(kr, hostI)) in
  let AK_ep: key = kdf_AK(hostI, R, tp) in
```

```
  let AK_ap: mkey = mkdf_AK(hostI, R, tp) in
  if mac(e4, AK_ap) = me4 then
  event endRkey(hostI, R, AK_ep);
  get ctr(=R, ct_r) in
  let (=hostI, =R, Nip: Zp, =ct_r) = denc(e4, AK_ep) in
  new Nr: Zp;
  insert ctr(R, addone(ct_r));
  let e5: bitstring = enc((SID, R, hostI, Nip, tp, Nr), AK_ep) in
  let mac5: bitstring = mac((SID, e5), AK_ap) in
  out(c, (SID, e5, mac5));
  in(c, (=SID, mac6: bitstring));
  let K_s: key = kdf_fn(addone(ct_r), hostI, R, Nip, Nr) in
  let m_Ks: mkey = mkdf_fn(addone(ct_r), hostI, R, Nip, Nr) in
  if mac((SID, hostI, R, Nip, Nr), m_Ks) = mac6 then
  event endIkey(hostI, R, K_s);
  out(c, enc(secretR, K_s)).

let processDG =
  in(c, (SID: bitstring, hostI: host, hostR: host, e7: bitstring, td:
    Zp, mac7: bitstring));
  get keys(=hostI, kd1) in
  if mac((SID, hostI, hostR, e7, td), kd1) = mac7 then
  get transMsKey(=hostI, =hostR, dk_ir) in
  let tdr: Zp = reenc(td, dk_ir) in
  get keys(=hostR, kd2) in
  let m7: bitstring = mac((SID, hostI, hostR, e7, tdr), kd2) in
  out(c, (SID, hostI, hostR, e7, td, m7)).

let processDK =
  in(c, (hi: host, hr: host, k: Zp));
  if (hi <> I) && (hr <> R) then insert transMsKey(hi, hr, k).

let processKD =
  in(c, (h: host, k: mkey));
  if (h <> I) && (h <> R) then insert keys(h, k).

let processK =
  in(c, (h: host, r: Zp));
  if (h <> I) && (h <> R) then insert msKey(h, r).

let processCTR =
  in(c, (h: host, r: Zp));
  if (h <> I) && (h <> R) then insert ctr(h, r).

process
  new sk_I: Zp;
  new sk_R: Zp;
  new K_id: mkey;
  new K_rd: mkey;
```

```
133  new cpt: Zp;
134  insert ctr(I, cpt);
135  insert ctr(R, cpt);
136  insert msKey(I, sk_I);
137  insert msKey(R, sk_R);
138  insert keys(I, K_id);
139  insert keys(R, K_rd);
140  let dgIR: Zp = kdf_rk(kdf_h(sk_I, R), kdf_h(sk_R, I)) in
141  insert transMsKey(I, R, dgIR);
142  ((!processI) | (processR) | (! processDG) | (! processK) | (!
      processKD) | (! processDK) | (! processCTR))
```

Listing 7.1: ProVerif code of AKAPR

Glossary of Acronyms

IoT	Internet of Things
PKI	Public Key Infrastructure
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
MIKEY	Multimedia Internet KEYing
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
WSN	Wireless Sensor Network
ECC	Elliptic Curve Cryptography
DoS	Denial-of-Service
PKC	Public Key Cryptography
ABE	Attribute-Based Encryption
IBE	Identity-based Encryption
KP-ABE	Key-Policy Attribute-Based Encryption
KMS	Key Management Server
DDH	Decisional Diffie-Hellman
CDH	Computational Diffie-Hellman
GDH	Gap Diffie-Hellman
DLP	Discrete Logarithm Problem
GDL	Gap Discrete Log
CA	Certificate Authority
6LBR	6LoWPAN Border Router
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
ECDH	Elliptic Curve Diffie-Hellman exchange
ECDSA	Elliptic Curve Digital Signature Algorithm
MAC	Message Authentication Code
TPM	Trusted Platform Module
CCA	Chosen-Ciphertext Attack
CMA	Chosen-Message Attack
PPT	Probabilistic Polynomial Time
HMAC	Hash-based Message Authentication Code
ABE-CT	Attribute-based Encryption Ciphertext
DP	Data Producer
DC	Data Consumer
I	Initiator
R	Responder
DG	Delegatee
PRE	Proxy Re-Encryption
KDF	Key Derivation Function
PRNG	Pseudorandom Number Generator

PKC	Public Key Cryptography
WSN	Wireless Sensor Network
KDC	Key Distribution Center
PKG	Private Key Generator
DH	Diffie-Hellman exchange

Bibliography

[1] CC2538 data sheet, `http://www.ti.com/lit/ds/symlink/cc2538.pdf`, last accessed: May 2016.

[2] Cisco global cloud index: Forecast and methodology, 2014-2019 white paper.

[3] CPABE toolkit: http://acsc.cs.utexas.edu/cpabe/.

[4] Gartner inc., forecast: The internet of things, worldwide, 2013.

[5] IPv6 over low power wpan (6LoWPAN) IETF working group.

[6] Openmote platform, `http://www.openmote.com/`, last accessed: May 2016.

[7] Pairing based cryptography library: `http://crypto.standford.edu/pbc`.

[8] Relic toolkit - an efficient library for cryptography, `https://github.com/relic-toolkit/relic`, last accessed: May 2016.

[9] SECG. Sec 2: Recommended elliptic curve domain parameters version 2.0.

[10] Mohammed Riyadh Abdmeziem and Djamel Tandjaoui. An end-to-end secure key management protocol for e-health applications. *Computers & Electrical Engineering*, 44:184–197, 2015.

[11] Carlisle Adams and Steve Lloyd. *Understanding public-key infrastructure: concepts, standards, and deployment considerations*. Sams Publishing, 1999.

[12] Basel Alomair and Radha Poovendran. Unconditionally secure authenticated encryption with shorter keys. In *WOSIS*, 2009.

[13] Moreno Ambrosin, Mauro Conti, and Tooska Dargahi. On the feasibility of attribute-based encryption on smartphone devices. IoT-Sys '15, 2015.

[14] JeeHea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In LarsR. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer Berlin Heidelberg, 2002.

[15] Jari Arkko, Elisabetta Carrara, Fredrik Lindholm, Mats Naslund, and Karl Norrman. RFC 3830: MIKEY: Multimedia internet keying. *Internet Engineering*, 2004.

[16] Giuseppe Ateniese, Giuseppe Bianchi, Angelo Capossele, and Chiara Petrioli. Low-cost standard signatures in wireless sensor networks: A case for reviving pre-computation techniques? In *Proceedings of NDSS 2013*, 2013.

[17] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[18] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. Formal proofs for the security of signcryption. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lecture Notes in Computer Science*, pages 80–98. Springer Berlin Heidelberg, 2002.

[19] Feng Bao and Robert H. Deng. A signcryption scheme with signature directly verifiable by public key. In *Public Key Cryptography, First International Workshop on Practice and Theory in Public Key Cryptography, PKC '98, Pacifico Yokohama, Japan, February 5-6, 1998, Proceedings*, volume 1431 of *Lecture Notes in Computer Science*, pages 55–59. Springer, 1998.

[20] Manuel Bernardo Barbosa and Pooya Farshim. Certificateless signcryption. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '08, pages 369–372, New York, NY, USA, 2008. ACM.

[21] Elaine B Barker and John Michael Kelsey. *Recommendation for random number generation using deterministic random bit generators (revised), NIST Special Publication SP 800-90.* 2007.

[22] Paulo S.L.M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Efficient and provably-secure identity-based signatures and signcryption from bilinear maps. In Bimal Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 515–532. Springer Berlin Heidelberg, 2005.

[23] David Basin, Sebastian Mödersheim, and Luca Vigano. An on-the-fly model-checker for security protocol analysis. In *European Symposium on Research in Computer Security*, pages 253–270. Springer, 2003.

[24] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. SIMON and SPECK: Block ciphers for the internet of things. In *NIST Lightweight Cryptography Workshop*, volume 2015, 2015.

[25] Amos Beimel. *PhD thesis: "Secure schemes for secret sharing and key distribution"*. Technion-Israel Institute of technology, Faculty of computer science, 1996.

[26] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 390–399. ACM, 2006.

[27] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM conference on Computer and communications security*, pages 62–73. ACM, 1993.

[28] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE S & P*, 2007.

[29] Giuseppe Bianchi, Angelo T. Capossele, Chiara Petrioli, and Dora Spenza. AGREE: exploiting energy harvesting to support data-centric access control in WSNs. *Ad hoc networks*, 11(8):2625–2636, 2013.

[30] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.

[31] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *Advances in Cryptology—EUROCRYPT'98*, pages 127–144. Springer, 1998.

[32] Rolf Blom. An optimal class of symmetric key generation systems. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 335–338. Springer, 1984.

[33] Rolf Blom. An optimal class of symmetric key generation systems. In *Advances in cryptology*, pages 335–338. Springer, 1985.

[34] Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 450–466. Springer, 2007.

[35] Dan Boneh. *Algorithmic Number Theory: Third International Symposiun, ANTS-III Portland, Oregon, USA, June 21–25, 1998 Proceedings*, chapter The Decision Diffie-Hellman problem, pages 48–63. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998.

[36] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In *Annual International Cryptology Conference*, pages 213–229. Springer, 2001.

[37] Carsten Bormann, Mehmet Ersue, and A Keranen. Terminology for constrained-node networks. *Internet Engineering Task Force (IETF), RFC*, 7228, 2014.

[38] Aymen Boudguiga, Alexis Olivereau, and Nouha Oualha. Server assisted key establishment for WSN: A MIKEY-Ticket approach. In *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 94–101, July 2013.

[39] Aymen Boudguiga, Alexis Olivereau, and Nouha Oualha. Server assisted key establishment for WSN: A MIKEY-Ticket approach. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 94–101. IEEE, 2013.

[40] Vladimir Božović, Daniel Socek, Rainer Steinwandt, and Viktória I Villányi. Multi-authority attribute-based encryption with honest-but-curious central authority. *International Journal of Computer Mathematics*, 89(3):268–283, 2012.

[41] Daniel R. L. Brown. Standards for efficient cryptography, sec 1: elliptic curve cryptography. *Released Standard Version*, 1, 2009.

[42] David Cadé and Bruno Blanchet. Proved generation of implementations from computationally secure protocol specifications1. *Journal of Computer Security*, 23(3):331–402, 2015.

[43] Seyit A Camtepe and Bülent Yener. Key distribution mechanisms for wireless sensor networks: a survey. *Rensselaer Polytechnic Institute, Troy, New York, Technical Report*, pages 05–07, 2005.

[44] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 185–194. ACM, 2007.

[45] Certicom. ECC tutorial, `https://www.certicom.com/ecc-tutorial`, last checked october 2016.

[46] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 197–213. IEEE, 2003.

[47] Melissa Chase. Multi-authority attribute based encryption. In *Theory of Cryptography Conference*, pages 515–534. Springer, 2007.

[48] Melissa Chase and Sherman SM Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 121–130. ACM, 2009.

[49] Ioannis Chatzigiannakis, Apostolos Pyrgelis, Paul G Spirakis, and Yannis C Stamatiou. Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 715–720. IEEE, 2011.

[50] Cheng Chen, Zhenfeng Zhang, and Dengguo Feng. Efficient ciphertext policy attribute-based encryption with constant-size ciphertext and constant computation-cost. In *Provable Security*, pages 84–101. Springer, 2011.

[51] Ling Cheung and Calvin Newport. Provably secure ciphertext policy ABE. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 456–465. ACM, 2007.

[52] Sherman SM Chow, Jian Weng, Yanjiang Yang, and Robert H Deng. Efficient unidirectional proxy re-encryption. In *Progress in Cryptology–AFRICACRYPT 2010*, pages 316–332. Springer, 2010.

[53] Simone Cirani, Gianluigi Ferrari, and Luca Veltri. Enforcing security mechanisms in the IP-based internet of things: An algorithmic overview. *Algorithms*, 6(2):197–226, 2013.

[54] Debra L Cook and Angelos D Keromytis. Conversion functions for symmetric key ciphers. *Journal of Information Assurance and Security*, 2:41–50, 2006.

[55] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[56] Quynh Dang. *Recommendation for applications using approved hash algorithms*. US Department of Commerce, National Institute of Standards and Technology, 2008.

[57] Giacomo De Meulenaer, François Gosset, O-X Standaert, and Olivier Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *Networking and Communications, 2008. WIMOB'08. IEEE International Conference on Wireless and Mobile Computing,*, pages 580–585. IEEE, 2008.

[58] Alexander W. Dent and Yuliang Zheng. *Practical Signcryption*. Springer-Verlag, 2010.

[59] AlexanderW. Dent and John Malone-Lee. Signcryption schemes based on the RSA problem. In Alexander W. Dent and Yuliang Zheng, editors, *Practical Signcryption*, Information Security and Cryptography, pages 99–117. Springer Berlin Heidelberg, 2010.

[60] Tim Dierks and Eric Rescorla. The transport layer security (TLS) protocol version 1.2, RFC 5246. 2008.

[61] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

[62] Daniel Dinu, Yann Le Corre, Dmitry Khovratovich, Léo Perrin, Johann Großschädl, and Alex Biryukov. Triathlon of lightweight block ciphers for the internet of things. *IACR Cryptology ePrint Archive*, 2015:209, 2015.

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

[64] Nishant Doshi and Devesh C. Jinwala. Fully secure ciphertext policy attribute-based encryption with constant length ciphertext and faster decryption. *Sec. and Commun. Netw.*, 7(11):1988–2002, November 2014.

[65] Wenliang Du, Jing Deng, Yunghsiang S Han, and Pramod K Varshney. A key predistribution scheme for sensor networks using deployment knowledge. *Dependable and Secure Computing, IEEE Transactions on*, 3(1):62–77, 2006.

[66] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks, swedish institute of computer science. 2011.

[67] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 455–462. IEEE, 2004.

[68] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*, EmNets '07, pages 28–32, New York, NY, USA, 2007. ACM.

[69] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32. ACM, 2007.

[70] Maitreyee Dutta, Anuj Kumar Singh, and Ajay Kumar. An efficient signcryption scheme based on ECC with forward secrecy and encrypted message authentication. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 399–403, Feb 2013.

[71] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in Cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[72] Ethmane El Moustaine and Maryline Laurent. A lattice based authentication for low-cost RFID. In *RFID-Technologies and Applications (RFID-TA), 2012 IEEE International Conference on*, pages 68–73. IEEE, 2012.

[73] Keita Emura, Atsuko Miyaji, Akito Nomura, Kazumasa Omote, and Masakazu Soshi. A ciphertext-policy attribute-based encryption scheme with constant ciphertext length. In *International Conference on Information Security Practice and Experience*, pages 13–23. Springer, 2009.

[74] Joakim Eriksson, Fredrik Österlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt, Robert Sauter, and Pedro José Marrón. Cooja/mspsim: interoperability testing for wireless sensor networks. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, page 27. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009.

[75] Pasi Eronen and Hannes Tschofenig. Pre-shared key ciphersuites for transport layer security (TLS). Technical report, RFC 4279, December, 2005.

[76] Laurent Eschenauer and Virgil D Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM, 2002.

[77] Martin Euchner. HMAC-authenticated diffie-hellman for multimedia internet keying (MIKEY), RFC 4650. 2006.

[78] Junfeng Fan, Lejla Batina, and Ingrid Verbauwhede. Light-weight implementation options for curve-based cryptography: HECC is also ready for RFID. In *Internet Technology and Secured Transactions, 2009. ICITST 2009. International Conference for*, pages 1–6. IEEE, 2009.

[79] Ali Fanian, Mehdi Berenjkoub, Hossein Saidi, and T. Aaron Gulliver. A scalable and efficient key establishment protocol for wireless sensor networks. In *2010 IEEE Globecom Workshops*, pages 1533–1538. IEEE, 2010.

[80] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988.

[81] PUB Fips. 186-2. digital signature standard (DSS). *National Institute of Standards and Technology (NIST)*, 2000.

[82] Sepideh Fouladgar, Bastien Mainaud, Khaled Masmoudi, and Hossam Afifi. Tiny 3-TLS: A trust delegation protocol for wireless sensor networks. In *Security and Privacy in Ad-Hoc and Sensor Networks*, pages 32–42. Springer, 2006.

[83] Steffen Fries and Dragan Ignjatic. On the applicability of various multimedia internet keying (MIKEY) modes and extensions. Technical report, 2008.

[84] Oscar Garcia-Morchon, D Gómez-Pérez, J Gutiérrez, R Rietman, B Schoenmakers, and L Tolhuizen. HIMMO-a lightweight, fully colluison resistant key-predistribution scheme. Technical report, Cryptology ePrint Archive, Report 2014/698, 2014.

[85] Oscar Garcia-Morchon, Sandeep S. Kumar, Sye Loong Keoh, Rene Hummen, and Rene Struik. Security considerations in the IP-based internet of things, draft-garcia-core-security-06. 2014.

[86] Oscar Garcia-Morchon, Ronald Rietman, Sahil Sharma, Ludo Tolhuizen, and Jose Luis Torre-Arce. DTLS-HIMMO: Achieving DTLS certificate security with symmetric key overhead. In *European Symposium on Research in Computer Security*, pages 224–242. Springer, 2015.

[87] Gunnar Gaubatz, J-P Kaps, Erdinc Ozturk, and Berk Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 146–150. IEEE, 2005.

[88] Gunnar Gaubatz, Jens-Peter Kaps, Ersin Ozturk, and Berk Sunar. State of the art in ultra-low power public key cryptography for wireless sensor networks. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 146–150, March 2005.

[89] Marc Girault. Self-certified public keys. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 490–497. Springer, 1991.

[90] Oded Goldreich. *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.

[91] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *Automata, languages and programming*, pages 579–591. Springer, 2008.

[92] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM CCS*, pages 89–98. Acm, 2006.

[93] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. End-to-end transport-layer security for internet-integrated sensing applications with mutual and delegated ECC public-key authentication. In *IFIP Networking Conference, 2013*, pages 1–9. IEEE, 2013.

[94] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In *Applied Cryptography and Network Security*, pages 288–306. Springer, 2007.

[95] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of ABE ciphertexts. In *USENIX Security Symposium*, volume 2011, 2011.

[96] Michael Groves. Elliptic curve-based certificateless signatures for identity-based encryption (ECCSI), RFC 6507. 2012.

[97] Michael Groves. MIKEY-SAKKE: Sakai-Kasahara key encryption in multimedia internet keying (MIKEY), RFC 6509. 2012.

[98] Christoph G Günther. An identity-based key-exchange protocol. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 29–37. Springer, 1989.

[99] Yiliang Han, Xiaoyuan Yang, Ping Wei, Yuming Wang, and Yupu Hu. ECGSC: Elliptic curve based generalized signcryption. In Jianhua Ma, Hai Jin, LaurenceT. Yang, and JeffreyJ.-P. Tsai, editors, *Ubiquitous Intelligence and Computing*, volume 4159 of *Lecture Notes in Computer Science*, pages 956–965. Springer Berlin Heidelberg, 2006.

[100] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[101] M Jason Hinek, Shaoquan Jiang, Reihaneh Safavi-Naini, and Siamak Fayyaz Shahandashti. Attribute-based encryption with key cloning protection. *IACR Cryptology ePrint Archive*, 2008:478, 2008.

[102] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.

[103] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, et al. HIGHT: A new block cipher suitable for low-resource device. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 46–59. Springer, 2006.

[104] Russell Housley, Warwick Ford, W Polk, and David Solo. RFC 5280: Internet x. 509 public key infrastructure certificate and crl profile, 2008.

[105] Jonathan Hui and Pascal Thubert. Compression format for IPv6 datagrams over ieee 802.15. 4-based networks, RFC 6282. 2011.

[106] René Hummen, Hossein Shafagh, Shahid Raza, Thiemo Voig, and Klaus Wehrle. Delegation-based authentication and authorization for the IP-based internet of things. In *Sensing, Communication, and Networking (SECON), 2014 Eleventh Annual IEEE International Conference on*, pages 284–292. Ieee, 2014.

[107] René Hummen, Hanno Wirtz, Jan Henrik Ziegeldorf, Jens Hiller, and Klaus Wehrle. Tailoring end-to-end ip security protocols to the internet of things. In *2013 21st IEEE International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2013.

[108] René Hummen, Jan H Ziegeldorf, Hossein Shafagh, Shahid Raza, and Klaus Wehrle. Towards viable certificate-based authentication for the internet of things. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*, pages 37–42. ACM, 2013.

[109] HR Hussen, GA Tizazu, Miao Ting, Taekkyeun Lee, Youngjun Choi, and Ki-Hyung Kim. Sakes: Secure authentication and key establishment scheme for M2M communication in the IP-based wireless sensor network (6l0wpan). In *Ubiquitous and Future Networks (ICUFN), 2013 Fifth International Conference on*, pages 246–251. IEEE, 2013.

[110] David D Hwang, Bo-Cheng Charles Lai, and Ingrid Verbauwhede. Energy-memory-security tradeoffs in distributed sensor networks. In *International Conference on Ad-Hoc Networks and Wireless*, pages 70–81. Springer, 2004.

[111] Luan Ibraimi, Milan Petkovic, Svetla Nikova, Pieter Hartel, and Willem Jonker. Mediated ciphertext-policy attribute-based encryption and its application. In *Information security applications*, pages 309–323. Springer, 2009.

[112] Dragan Ignjatic, Lakshminath Dondeti, Francois Audet, and Ping Lin. MIKEY-RSA-R: An additional mode of key distribution in multimedia internet keying (MIKEY). *RFC 4738, November*, 2006.

[113] Garner Inc. Forecast: The internet of things, worldwide. 2013.

[114] Takashi Ito, Hidenori Ohta, Nori Matsuda, and Takeshi Yoneda. A key pre-distribution scheme for secure sensor networks using probability density function of node deployment. In *Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks*, pages 69–75. ACM, 2005.

[115] Asha Liza John and Sabu M Thampi. Mutual authentication based on HECC for RFID implant systems. In *International Symposium on Security in Computing and Communication*, pages 18–29. Springer, 2016.

[116] Mitsuru Kanda, Yoshihiro Ohba, Subir Das, and Stephen Chasko. PANA applicability in constrained environments, 2012.

[117] Charlie Kaufman. Internet key exchange (IKEv2) protocol, RFC 4306. 2005.

[118] Cameron F Kerry. Digital signature standard (DSS). *National Institute of Standards and Technology*, 2013.

[119] Tero Kivinen. Minimal IKEv2, draft IETF. 2012.

[120] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[121] Neal Koblitz and Alfred J Menezes. A survey of public-key cryptosystems. *SIAM review*, 46(4):599–634, 2004.

[122] John Kohl and Clifford Neuman. The kerberos network authentication service (v5). Technical report, 1993.

[123] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. A DTLS based end-to-end security architecture for the internet of things with two-way authentication. In *Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on*, pages 956–963. IEEE, 2012.

[124] Pascal Lafourcade, Vanessa Terrade, and Sylvain Vigier. Comparison of cryptographic verification tools dealing with algebraic properties. In *International Workshop on Formal Aspects in Security and Trust*, pages 173–185. Springer, 2009.

[125] Bocheng Lai, Sungha Kim, and Ingrid Verbauwhede. Scalable session key construction protocol for wireless sensor networks. In *IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES)*, page 7. Citeseer, 2002.

[126] Cheng-Chi Lee, Pei-Shan Chung, and Min-Shiang Hwang. A survey on attribute-based encryption schemes of access control in cloud environments. *IJ Network Security*, 15(4):231–240, 2013.

[127] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Advances in Cryptology–EUROCRYPT 2011*, pages 568–588. Springer, 2011.

[128] Fagen Li and Chunxiang Xu. An improved identity-based KCDSA signcryption scheme. In *Data, Privacy, and E-Commerce, 2007. ISDPE 2007. The First International Symposium on*, pages 230–232, Nov 2007.

[129] Jin Li, Kui Ren, Bo Zhu, and Zhiguo Wan. Privacy-aware attribute-based encryption with user accountability. In *International Conference on Information Security*, pages 347–362. Springer, 2009.

[130] Chae Hoon Lim and Pil Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In *Annual International Cryptology Conference*, pages 249–263. Springer, 1997.

[131] Chae Hoon Lim and Pil Joong Lee. A study on the proposed korean digital signature algorithm. In *Advances in Cryptology-ASIACRYPT'98, LNCS 1514, Spinger-Verlag*, pages 175–186, 1998.

[132] Chae Hoon Lim and Pil Joong Lee. A study on the proposed korean digital signature algorithm. In *Advances in Cryptology—ASIACRYPT'98*, pages 175–186. Springer, 1998.

[133] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(1):41–77, 2005.

[134] David J Malan, Matt Welsh, and Michael D Smith. Implementing public-key infrastructure for sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 4(4):22, 2008.

[135] Charalampos Manifavas, George Hatzivasilis, Konstantinos Fysarakis, and Konstantinos Rantos. Lightweight cryptography for embedded systems–a comparative analysis. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 333–349. Springer, 2014.

[136] Leandro Marin, Antonio Jara, and Antonio F Skarmeta. Shifting primes: Optimizing elliptic curve cryptography for smart things. In *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on*, pages 793–798. IEEE, 2012.

[137] Toshihiko Matsuo. Proxy re-encryption systems for identity-based encryption. In *Pairing-Based Cryptography–Pairing 2007*, pages 247–267. Springer, 2007.

[138] Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. *Coding Thv*, 4244:114–116, 1978.

[139] Francisco Vidal Meca, Jan Henrik Ziegeldorf, Pedro Moreno Sanchez, Oscar Garcia Morchon, Sandeep S Kumar, and Sye Loong Keoh. HIP security architecture for the IP-based internet of things. In *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, pages 1331–1336. IEEE, 2013.

[140] Alfred Menezes, University of Waterloo. Dept. of Combinatorics, Optimization, University of Waterloo. Faculty of Mathematics, Robert Zuccherato, and Yi-Hong Wu. *An elementary introduction to hyperelliptic curves*. Faculty of Mathematics, University of Waterloo, 1996.

[141] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[142] Ralph C Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4):294–299, 1978.

[143] Daniel Migault, Guggemos Tobias, and Daniel Palomares. Minimal ESP, IP security maintenance and extensions, internet-draft, draft-mglt-lwig-minimal-esp-01.txt. 2014.

[144] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 417–426. Springer, 1985.

[145] Robert Moskowitz, Tobias Heer, Petri Jokela, and Thomas R. Henderson. Host identity protocol version 2 (HIPv2), RFC 7401. 2015.

[146] Robert Moskowitz and Rene Hummen. HIP Diet exchange (DEX): draft-moskowitz-hip-rg-dex-05. *Internet Engineering Task Force, Status: Work in progress, Tech. Rep*, 2011.

[147] Kim Thuat Nguyen, Maryline Laurent, and Nouha Oualha. Lightweight certificateless and provably-secure signcryptosystem for the internet of things. In *the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-15)*, 2015.

[148] Kim Thuat Nguyen, Maryline Laurent, and Nouha Oualha. Novel lightweight signcryption-based key distribution method for MIKEY. In *sthe 10th International Conference on Information Security Theory and Practice (WISTP)*, 2016.

[149] Takashi Nishide, Kazuki Yoneyama, and Kazuo Ohta. Attribute-based encryption with partially hidden encryptor-specified access structures. In *International Conference on Applied Cryptography and Network Security*, pages 111–129. Springer, 2008.

[150] Yoshihiro Ohba, Alper Yegin, Basavaraj Patil, Dan Forsberg, and Hannes Tschofenig. Protocol for carrying authentication for network access (PANA). 2008.

[151] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In *International Workshop on Public Key Cryptography*, pages 104–118. Springer, 2001.

[152] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 195–203. ACM, 2007.

[153] Jacques Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 33–48. Springer, 1996.

[154] 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.

[155] Pawani Porambage, Pardeep Kumar, Corinna Schmitt, Andrei Gurtov, and Mika Ylianttila. Certificate-based pairwise key establishment protocol for wireless sensor networks. In *Computational Science and Engineering (CSE), 2013 IEEE 16th International Conference on*, pages 667–674. IEEE, 2013.

[156] Zhi Qiao, Shuwen Liang, Spencer Davis, and Hai Jiang. Survey of attribute based encryption. In *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on*, pages 1–6. IEEE, 2014.

[157] Daniel R. L. Brown. Sec 2: Recommended elliptic curve domain parameters, 2010.

[158] Daniel R. L. Brown, Eugene Chin, and Chi Chiu Tse. ECC algorithms for MIKEY. *Work in Progress*, 2007.

[159] Michael O Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, DTIC Document, 1979.

[160] Sangram Ray and GP Biswas. Establishment of ECC-based initial secrecy usable for ike implementation. In *Proc. of World Congress on Expert Systems (WCE)*, 2012.

[161] Shahid Raza, Simon Duquennoy, Tony Chung, Thiemo Voigt, Utz Roedig, et al. Securing communication in 6LoWPAN with compressed ipsec. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8. IEEE, 2011.

[162] Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, and Thiemo Voigt. Lithe: Lightweight secure CoAP for the internet of things. *Sensors Journal, IEEE*, 13(10):3711–3720, 2013.

[163] Eric Rescorla. Diffie-hellman key agreement method, RFC 2631. 1999.

[164] Eric Rescorla and Nagendra Modadugu. Datagram transport layer security version 1.2, RFC 6347. 2012.

[165] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[166] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147–159, 2011.

[167] Jong-Ho Ryu, Youn-Seo Jeong, and Dong-Il Seo. Identity based KCDSA signcryption. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 2, pages 6 pp.–1374, Feb 2006.

[168] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005*, pages 457–473. Springer, 2005.

[169] Yosra Ben Saied. *Collaborative Security for the Internet of Thing*. PhD thesis, 2013.

[170] Yosra Ben Saied and Alexis Olivereau. D-HIP: A distributed key exchange scheme for HIP-based internet of things. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–7. IEEE, 2012.

[171] Yosra Ben Saied, Alexis Olivereau, Djamal Zeghlache, and Maryline Laurent. Lightweight collaborative key establishment scheme for the internet of things. *Computer Networks*, 64:273–295, 2014.

[172] Ravi Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, (2):38–47, 1996.

[173] Behcet Sarikaya, Yoshihiro Ohba, Robert Moskowitz, Zhen Cao, and Robert Cragie. Security bootstrapping solutions for resource-constrained devices, draft-sarikaya-core-sbootstrapping-05. 2012.

[174] Savio Sciancalepore, Angelo Capossele, Giuseppe Piro, Gennaro Boggia, and Giuseppe Bianchi. Key management protocol with implicit certificates for iot systems. In *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, pages 37–42. ACM, 2015.

[175] S.SharmilaDeva Selvi, S.Sree Vivek, and C.Pandu Rangan. Cryptanalysis of certificateless signcryption schemes and an efficient construction without pairing. In Feng Bao, Moti Yung, Dongdai Lin, and Jiwu Jing, editors, *Information Security and Cryptology*, volume 6151 of *Lecture Notes in Computer Science*, pages 75–92. Springer Berlin Heidelberg, 2010.

[176] S Seys and B Preneel. Key establishment and authentication suite to counter DoS attacks in distributed sensor networks. *Unpublished manuscript, COSIC*, 2002.

[177] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 47–53. Springer, 1984.

[178] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: an ultra-lightweight blockcipher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 342–357. Springer, 2011.

[179] Jun-Bum Shin, Kwangsu Lee, and Kyungah Shim. New DSA-verifiable signcryption schemes. In *Information Security and Cryptology—ICISC 2002*, pages 35–47. Springer, 2002.

[180] Jun-Bum Shin, Kwangsu Lee, and Kyungah Shim. New DSA-verifiable signcryption schemes. In PilJoong Lee and ChaeHoon Lim, editors, *Information Security and Cryptology — ICISC 2002*, volume 2587 of *Lecture Notes in Computer Science*, pages 35–47. Springer Berlin Heidelberg, 2003.

[181] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[182] Marcos A Simplicio Jr, Marcos VM Silva, Renan CA Alves, and Tiago KC Shibata. Lightweight and escrow-less authenticated key agreement for the internet of things. *Computer Communications*, 2016.

[183] Amril Syalim, Takashi Nishide, and Kouichi Sakurai. Realizing proxy re-encryption in the symmetric world. In *Informatics Engineering and Information Science*, pages 259–274. Springer, 2011.

[184] Arago Systems. Wismote, `http://www.aragosystems.com/images/stories/WiSMote/Doc/wismote_en.pdf`, year=last accessed: Feb. 2015.

[185] Piotr Szczechowiak and Martin Collier. Tinyibe: Identity-based encryption for heterogeneous sensor networks. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2009 5th International Conference on*, pages 319–354. IEEE, 2009.

[186] Tian Tian and John Mattsson. MIKEY-TICKET: Ticket-based modes of key distribution in multimedia internet keying (MIKEY). 2011.

[187] Mohsen Toorani and Ali Asghar Beheshti Shirazi. An elliptic curve-based signcryption scheme with forward secrecy. *CoRR*, abs/1005.1856, 2010.

[188] Lyes Touati, Yacine Challal, and Abdelmadjid Bouabdallah. C-CP-ABE: Cooperative ciphertext policy attribute-based encryption for the internet of things. In *IEEE INDS*, 2014.

[189] Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. ECDSA-verifiable signcryption scheme with signature verification on the signcrypted message. In *Information Security and Cryptology*, pages 11–24. Springer, 2007.

[190] Raylin Tso, Takeshi Okamoto, and Eiji Okamoto. ECDSA-verifiable signcryption scheme with signature verification on the signcrypted message. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Information Security and Cryptology*, volume 4990 of *Lecture Notes in Computer Science*, pages 11–24. Springer Berlin Heidelberg, 2008.

[191] Mathieu Turuani. The CL-Atse protocol analyser. In *International Conference on Rewriting Techniques and Applications*, pages 277–286. Springer, 2006.

[192] Floris Van den Abeele, Tom Vandewinckele, Jeroen Hoebeke, Ingrid Moerman, and Piet Demeester. Secure communication in IP-based wireless sensor networks via a trusted gateway. In *Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on*, pages 1–6. IEEE, 2015.

[193] V Vijayalakshmi, R Sharmila, and R Shalini. Hierarchical key management scheme using hyper elliptic curve cryptography in wireless sensor networks. In *Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on*, pages 1–5. IEEE, 2015.

[194] Yong Wang, Garhan Attebury, and Byrav Ramamurthy. A survey of security issues in wireless sensor networks. 2006.

[195] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC 2011*, pages 53–70. Springer, 2011.

[196] Wenjian Xie and Zhang Zhang. Certificateless signcryption without pairing, 2010.

[197] Lijun Yang, Chao Ding, and Meng Wu. Establishing authenticated pairwise key using pairing-based cryptography for sensor networks. In *Communications and Networking in China (CHINACOM), 2013 8th International ICST Conference on*, pages 517–522. IEEE, 2013.

[198] Gang Yu, Hongzhi Yang, Shuqin Fan, Yong Shen, and Wenbao Han. Efficient certificateless signcryption scheme from Weil pairing. *Journal of Networks*, 6(9), 2011.

[199] DaeHyun Yum and PilJoong Lee. New signcryption schemes based on KCDSA. In Kwangjo Kim, editor, *Information Security and Cryptology — ICISC 2001*, volume 2288 of *Lecture Notes in Computer Science*, pages 305–317. Springer Berlin Heidelberg, 2002.

[200] Wei Zhang. *Improvements and Generalisations of Signcryption Schemes*. PhD thesis, 2013.

[201] Yuliang Zheng. Signcryption or how to achieve cost (signature & encryption) <<cost (signature)+ cost (encryption). *http://www. pscit. monash. edu. au/yuliang/pubs/signcrypt. ps. Z*, 1999.

[202] Xuanwu Zhou, Zhigang Jin, Yan Fu, Huaiwei Zhou, and Lianmin Qin. Short signcryption scheme for the internet of things. *Informatica (Slovenia)*, 35(4):521–530, 2011.

[203] Zhibin Zhou and Dijiang Huang. Efficient and secure data storage operations for mobile cloud computing. In *Proceedings of the 8th International Conference on Network and Service Management*, 2012.