A PUF-BASED LIGHTWEIGHT GROUP AUTHENTICATION AND KEY DISTRIBUTION PROTOCOL

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HÜSNÜ YILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
COMPUTER ENGINEERING

AUGUST 2020

Approval of the thesis:

A PUF-BASED LIGHTWEIGHT GROUP AUTHENTICATION AND KEY DISTRIBUTION PROTOCOL

submitted by **HÜSNÜ YILDIZ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Ertan Onur
Supervisor, **Computer Engineering, METU** _____

Assoc. Prof. Dr. Murat Cenk
Co-supervisor, **Institute of Applied Mathematics, METU** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering, METU _____

Prof. Dr. Ertan Onur
Computer Engineering, METU _____

Prof. Dr. Ali Aydın Selçuk
Computer Engineering, TOBB ETÜ _____

Assoc. Prof. Dr. Sevil Şen
Computer Engineering, Hacettepe University _____

Assist. Prof. Dr. Pelin Angın
Computer Engineering, METU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Hüsnü Yıldız

Signature :

ABSTRACT

A PUF-BASED LIGHTWEIGHT GROUP AUTHENTICATION AND KEY DISTRIBUTION PROTOCOL

Hüsnü Yıldız,
Ph.D., Department of Computer Engineering
Supervisor: Prof. Dr. Ertan Onur
Co-Supervisor: Assoc. Prof. Dr. Murat Cenk

August 2020, 78 pages

Securing Internet of Things (IoT) applications that collect and transport sensitive data by guaranteeing authenticity, integrity, and confidentiality is a critical challenge. Reducing computation and communication overhead of security functions is also a key concern since a large number of constrained devices may take place in such applications. Our main focus in this thesis is group authentication and key management in IoT. The existing group authentication and key management protocols in the literature perform computations using asymmetric ciphers, which are costly for IoT. Therefore, applications generally employ simple security primitives that are prone to or lead to cyber-attacks by using IoT devices. In this thesis, we propose a physically unclonable function (PUF) based lightweight group authentication and key distribution (PLGAKD) protocol that employs PUF, factorial tree, and the Chinese Remainder Theorem (CRT). In PLGAKD, PUF facilitates lightweight authentication and key distribution for group members. Factorial tree and CRT help us reduce the number of keys stored in nodes and the number of communication messages contrary to the binary tree. The communication overhead in PLGAKD is reduced to almost half in

comparison with binary trees for almost the same number of group members. Moreover, the computation overhead is significantly less since only symmetric encryption, hash functions, and XOR are performed for group operations.

Keywords: Group authentication, group key distribution, security, Internet of Things (IoT), lightweight cryptography, physically unclonable function (PUF), the Chinese Remainder Theorem (CRT).

ÖZ

PUF TABANLI HAFİF AĞIRLIKLI KİMLİK DOĞRULAMA VE ANAHTAR DAĞITIMI PROTOKOLÜ

Hüsnü Yıldız,
Doktora, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Ertan Onur
Ortak Tez Yöneticisi: Doç. Dr. Murat Cenk

Ağustos 2020 , 78 sayfa

Kimlik denetimini, bütünlüğü ve gizliliği garanti ederek hassas verileri toplayan ve aktaran Nesnelerin İnterneti (IoT) uygulamalarının güvenliği önemli bir sorundur. Bu tür uygulamalarda çok sayıda kısıtlı özelliklere sahip cihaz olabileceğinden, güvenlik işlevlerinin hesaplama ve iletişim yükünü azaltmak da önemli bir konudur. Bu tezde ana odağımız IoT'de grup kimlik doğrulaması ve anahtar yönetimidir. Literatürdeki mevcut grup kimlik doğrulaması ve anahtar yönetimi protokolleri, IoT için maliyetli olan asimetrik şifreleri kullanarak hesaplamalar yapar. Bu nedenle, IoT uygulamaları genellikle basit güvenlik yöntemlerini kullanır. Bu ise siber saldırılarda, cihazların kolayca ele geçirilip kullanılmasına olanak tanımaktadır. Bu tezde fiziksel olarak klonlanamayan fonksiyon (PUF) tabanlı hafifsiklet grup kimlik doğrulaması ve anahtar dağıtım (PLGAKD) protokolü PUF, faktöriyel ağaç ve Çinli Kalan Teoremi (CRT) kullanılarak önerilmektedir. Bu protokolde, PUF grup üyeleri için hafifsiklet kimlik doğrulama ve anahtar dağıtımını gerçekleştirir. Faktöriyel ağaç ve CRT, ikili arama ağacına kıyasla, düğümlerde saklanan anahtar sayısını ve iletişim mesajlarının sayı-

sını azaltmamıza yardımcı olur. PLGAKD'deki iletişim yükü ikili arama ağacı algoritmalarına göre neredeyse yarı yarıya düşürülmüştür. PLGAKD'de sadece simetrik şifreleme, özet fonksiyonu ve XOR operasyonları kullanıldığından, hesaplama yükü oldukça düşüktür.

Anahtar Kelimeler: grup kimlik doğrulaması, grup anahtar dağıtımı, güvenlik, nesnelerin interneti (IoT), hafif işlemli kriptografi, fiziksel olarak klonlanamayan fonksiyonlar, Çinli Kalan Teoremi

To my family...

ACKNOWLEDGMENTS

I would like to express my great appreciations to my advisor Prof. Dr. Ertan Onur and my co-advisor Assoc. Dr. Murat Cenk for the continuous support of my Ph.D study, for their patience, motivation, and immense knowledge. I would also like to thank the rest of my thesis jury committee: Prof. Dr. İsmail Hakkı Toroslu, Prof. Dr. Ali Aydın Selçuk, Assoc. Dr. Sevil Şen, and Assist. Dr. Pelin Angın, for their insightful comments and guidance which incented me to widen my research from various perspectives.

I am deeply grateful to my friends, Ulaş Nacar, Sibel Bekiroğlu, Merve Fidan, Özgür Kaya, Alperen Eroğlu, and Adnan Kılıç for their support and encouragement.

Last but not the least, I would like to thank my family for their endless patience, encouragement and support throughout writing this thesis and my life in general.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

6LoWPAN	IPv6 over Low -Power Wireless Personal Area Networks
BAN	Body Area Networks
COAP	Constrained Application Protocol
CPU	Central Process Unit
CRT	Chinese Remainder Theorem
CU	Control Unit
DDOS	Distributed Denial of Service
DDS	Data Distribution Service
DOS	Denial of Service
DSN	Diffusion Switching Mechanism
ECC	Error Correction Code
ECDH	Elliptic Curve Diffie Hellman
FN	Feistel Network
GE	Gate Equivalence
HMAC	Hash-base Message Authentication Code
HSS	Home Subscriber Server
IoT	Internet of Things
IPv4	Internet Procotocol version 4
IPv6	Internet Procotocol version 6
ISO	International Organization for Standardization
LFSR	Linear Feedback Shift Register
LKH	Logical Key Hierarchy

MAC	Message Authentication Code
MC	Main Controller
MITM	Man in the Middle
MME	Mobile Management Entity
MQTT	Message Queuing Telemetry Transport
NFSR	Non-linear Feedback Shift Register
PKC	Public Key Cryptography
PKSG	PANAMA Like Keystream Generator
PRNG	Pseudorandom Number Generator
PUF	Physically Unclonable Function
RAM	Random Access Memory
ROM	Read Only Memory
SPN	Substitution Permutation Network
XMPP	Extensible Messaging and Presence Protocol

CHAPTER 1

INTRODUCTION

The number of IoT devices is expected to be 75 billion in 2025 [32]. Since IoT devices are designed to perform only one task or a few tasks on a limited energy budget for a very long period of time, most of them have limited CPU, memory, and disk capacity. Due to these constraints, they cannot perform conventional cryptographic operations. Therefore, these devices either have no security or straightforward security policies. According to the Kaspersky Lab report, over 105 million IoT attacks coming from $279,000$ unique IP addresses were detected in the first half of 2019 [28]. These attacks are performed mainly using Mirai Malware, NyaDrop Trojan, and Gafgyt Malware. The number of attacks has increased almost ten times compared to the previous year. In 2016, a distributed denial of service (DDoS) attack was performed to a few of the largest technology companies by using IoT devices, and the services they provided during DDoS could not be accessed. IoT devices are not only used to perform cyber-attacks to other companies but also pose a threat to lives through the data they collect and share with providers. Therefore, these devices must be secured. For IoT security, symmetric ciphers or hash functions are suitable contrary to asymmetric ciphers due to the fact that their computation overhead is considerably lower than asymmetric ciphers except for elliptic curve cryptography algorithms [36, 37, 42, 64, 77, 79, 82, 89, 94, 101]. New lightweight symmetric ciphers and hash functions are designed by reducing the number of rounds, block size and key size in different ranges from 32 bits to 128 bits. These lightweight algorithms are examined under two groups: hardware- or software-oriented that can be chosen tailored to requirements. Moreover, as an integrated circuit, physically-unclonable function (PUF) reduces the computational overhead of cryptographic operations, specifically in authentication and key distribution [25, 31, 40, 74, 92].

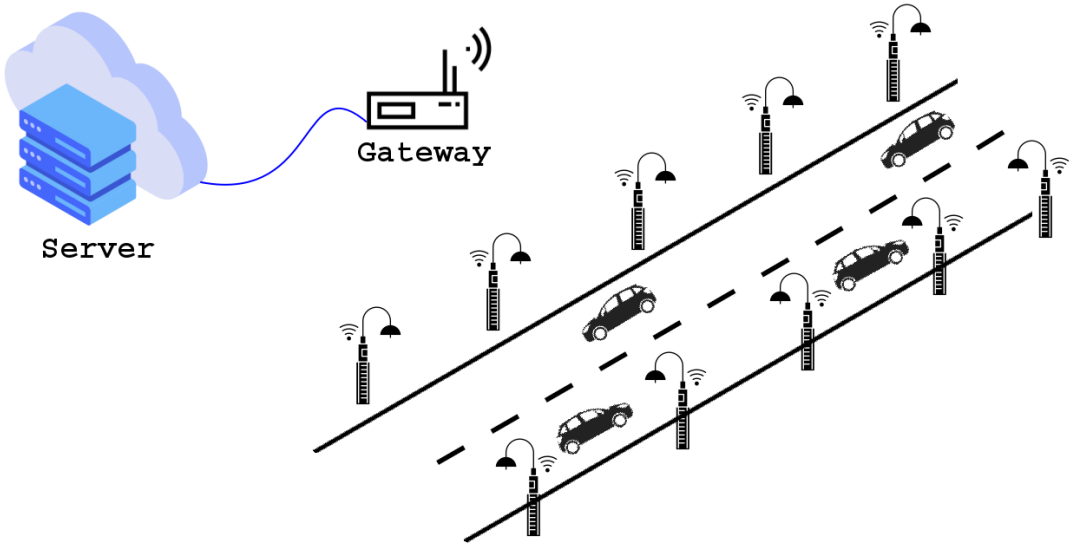

Figure 1.1: Group communication in smart lighting.

In addition to IoT security problems, increasing the number of IoT devices can cause network congestion, message losses, and communication delays while sharing data, updating firmware/configuration, or video streaming. Instead of using unicast messaging, broadcast and multicast mechanisms can be used to eliminate these problems in IoT applications such as software/configuration update, traffic management, border security, smart ligthing, emergency broadcast, and management of smart home appliances. Therefore, group communication algorithms may be used in large-scale networks like smart lighting. A good illustration for this point could be the one as shown Figure 1.1. In this example, a powerful server communicates with smart lights over a gateway. Group communication in smart lighting enables efficient firmware/configuration updates of IoT devices. Moreover, switching on/off lights with minimum delays is achieved by using group communication. These algorithms use a group key for secure and lightweight communication among group members. In this way, network overhead can be significantly reduced while meeting security requirements. Group communication can be examined under three main architectures, which are centralized, decentralized, and distributed [41].

In a distributed architecture, each node in the group must join the group key agreement process in the initialization phase. Since this method requires a large amount of

communication and computation overhead, distributed methods may not be feasible in IoT applications that require a central unit. Therefore, centralized or decentralized key distribution methods are used. In these architectures, one or more group leaders manage the key generation process and distribute the keys to the nodes. The hardest part of group communication algorithms is the key renewal process that is performed when any node is evicted from the group or added to the group to guarantee backward and forward secrecy. Many studies in the literature prefer key hierarchy using binary tree structures in which the rekeying process is completed efficiently for backward and forward secrecy with logarithmic complexity [85, 98, 105].

1.1 Problem Definition

Group communication algorithms take place in IoT-based solutions to perform security primitives with minimum cost. The main bottleneck of these approaches is the cost of a key renewal process rather than the authentication of nodes and the distribution of group key in the initialization phase. Owing to the fact that information is encrypted by using the group key, adding a new node to the group or evicting a node from the group requires the group key to be updated for preserving forward and backward secrecy. While updating the group key, this process should be performed with a minimum cost on the network and it should be completed as quickly as possible. The existing methods are inappropriate for IoT devices on account of the fact that these methods do not consider lightweight cryptographic primitives, storage, and communication overhead in the same design.

1.2 Proposed Protocol and Contributions

PUF-based Lightweight Group Authentication and Key Distribution Protocol is designed fully compatible with IoT devices. In order to eliminate the costly public key cryptography operations, the cheap and energy-efficient structure of PUF is preferred without compromising security. For efficient key renewal process, many studies in the literature use tree-based key hierarchy that provides logarithmic communication complexity with extra storage. Therefore, PLGAKD uses a factorial tree that contains

more nodes on fewer levels to reduce storage overhead. However, this leads to an increase in the communication overhead in comparison with binary-tree structures. At this point, the Chinese Remainder Theorem is used to reduce network overhead.

In this thesis, our contribution is summarized as follows:

- We propose a PUF-based lightweight group authentication and key distribution (PLGAKD) protocol. This protocol can perform in both centralized and decentralized architectures. The proposed protocol provides efficient group lightweight authentication and key distribution by further reducing the number of messages and stored keys for the key renewal process in comparison to other approaches in the literature that employ binary tree.
- To guarantee lightweight secure authentication and key distribution, a PUF, which provides unique keys without storing any information on a device, is used with symmetric ciphers.
- To reduce storage overhead of nodes, factorial tree is preferred that holds more nodes at lower levels. Although factorial tree structure reduces storage overhead, it does not offer an advantage to reduce the number of messages. Therefore, CRT is utilized for combining multiple messages into a single message for further decreasing the number of messages.

1.3 Outline

The remainder of this thesis is structured as follows:

- Chapter 2 presents a detailed description of the lightweight cryptography and describes the main components of the proposed protocol.
- Chapter 3 gives a brief review of the existing group communication algorithms and lightweight approaches.
- Chapter 4 introduces complete model of the PLGAKD protocol. How the PLGAKD protocol achieves lightweight group authentication, key distribution, and efficient key renewal are presented in this chapter.

- Chapter 5 discusses the threat, communication, computation, and storage analysis of the proposed protocol.
- Chapter 6 presents the conclusion of this thesis.

CHAPTER 2

PRELIMINARY AND BACKGROUND

This chapter introduces key terms of the proposed protocol and gives a detailed explanation about lightweight cryptography.

2.1 Lightweight Cryptography

The number of connected devices will increase with the deployment of Internet of Things (IoT) and is expected to reach 75 billion in 2025 [32]. Since the number of devices increases in the next decade, the amount of data will increase while these devices are communicating with each other. Therefore, a large bandwidth will be required. Because of the massive amounts of data generated by IoT devices using machine-to-machine communication and human-centric applications, the existing communication protocols will encounter communication delays, network congestions, and message losses. Since 5G provides machine-to-machine communication, wider coverage, ultra-large capacity, low latency, and green communication, it is expected that 5G becomes the de-facto communication and solves all problems arising from IoT devices. Additionally, IPv6 may substitute IPv4 in the near future due to the lack of sufficient address space of IPv4. Thanks to IPv6 that has 2^{128} addresses, devices will directly communicate with each other in the network. 5G and IPv6 technologies will pave the way for new security vulnerabilities while having many advantages.

Conventional cryptographic algorithms are employed to provide confidentiality, integrity, availability and non-repudiation. They are usually designed to satisfy security requirements without considering limitation or constraints on energy consumption,

computation power or memory. They are mostly executed on powerful devices such as desktops, smartphones, tablets and servers. However, constrained devices cannot perform conventional cryptographic operations since they have limited CPU, memory and power. In near future, billions of constrained devices are expected to be used in smart homes, smart cities, agriculture, healthcare and transportation to gather information via sensors and send to clouds. While transporting the data, new challenges arise due to constraints of the employed equipment that cannot perform security primitives. Therefore, new protocols and security mechanisms have to be developed since conventional cryptographic algorithms are not performed on constrained devices. Because of the limited CPU, memory and storage, there are new solutions like Constrained Application Protocol (COAP) [84] as the web transfer protocol, 6LoWPAN [62] as the low cost internet connectivity, Message Queuing Telemetry Transport (MQTT) [8] as the machine to machine lightweight messaging protocol for monitoring, Extensible Messaging and Presence Protocol (XMPP) [83] as the communication protocol for messaging, Data Distribution Service (DDS) [3] as the data exchange service. However, these protocols and/or services cannot be performed by constrained devices because of the their limitations. To establish secure channels for constrained devices, novel lightweight or ultra-lightweight cryptographic approaches have to be designed to provide a sufficient security level. The term "lightweight" is used for energy efficiency. It does not indicate lack of security or lower level security.

The main communication design of IoT, integrated with transportation, smart cities, smart homes, healthcare, agriculture and other digital systems, and 5G is shown in Figure 2.1. The vision of 5G and IoT integration with transportation, smart cities, smart homes, healthcare, agriculture and other digital systems is illustrated in Figure 2.1. Many applications are envisioned, such as adjusting home temperature, checking unusual situations, emergency broadcast, measuring the amount of moisture of fields, border security, optimizing traffic density, adjusting power distribution, and smart lighting. Data centers store data gathered by sensors. Moreover, cloud-like servers are used to reduce computation overhead on clouds and communication overhead on IoT. They are located near to a source of the data.

New devices are produced for new IoT design, and the amount of energy consumption has been increasing. However, it affects our daily life. One of the well-known

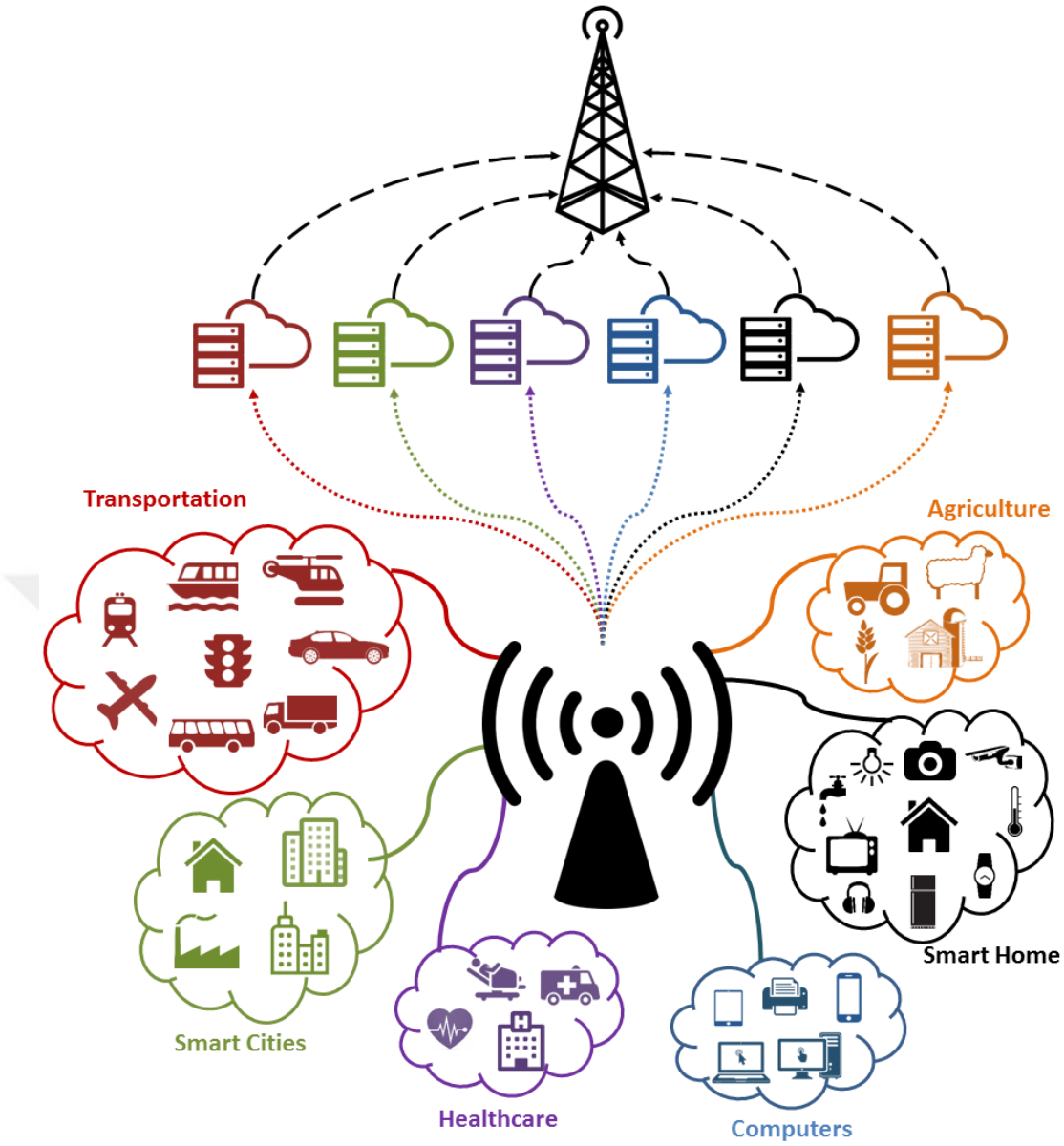

Figure 2.1: 5G and IoT integration in intelligent transportation systems, smart cities, smart homes, healthcare, agriculture and other vertical sectors.

effect is global warming since CO_2 emissions are increasing through the production and the consumption of energy. Nowadays, factories consume a huge portion of energy as shown in Figure 2.2 [12]. In addition, constrained devices will be one of the main energy consuming domains since billions of them will take part in our daily life, soon. Moreover, changing the batteries of constrained devices frequently is hard. Therefore, the energy consumption of IoT devices should be reduced by designing

energy efficient solutions. Green and secure technology is related with lightweight cryptography since their design considers minimum energy consumption with sufficient security principles.

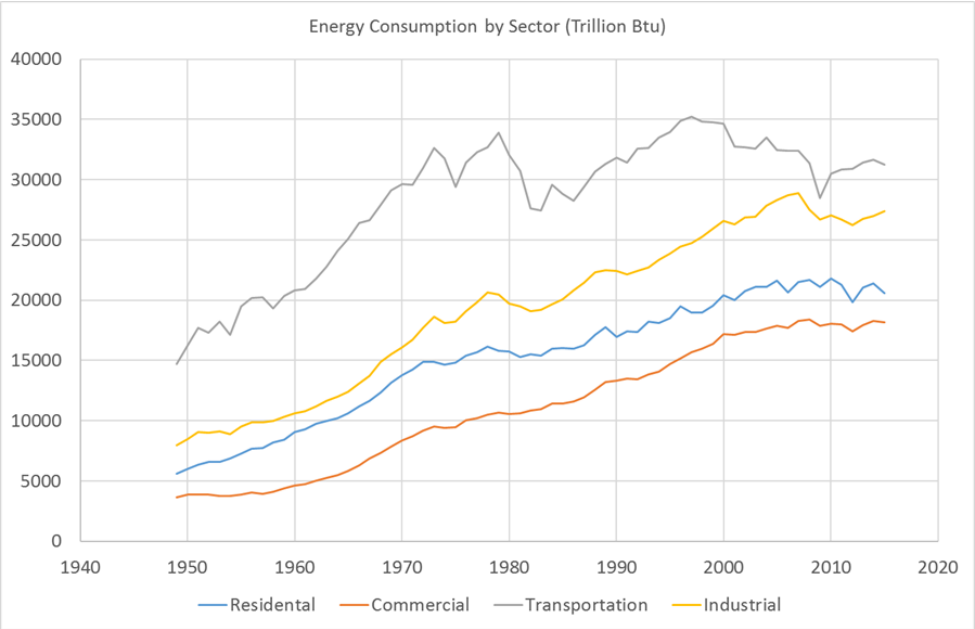

Figure 2.2: Energy Consumption between 1945 and 2015 (redrawn from [12])

Although extra energy saving via lightweight cryptography per unit might be negligible, the total amount will be considerably large when implemented in a network that contains billions of devices. Therefore we give a detailed explanation on lightweight cryptography in this section. This section covers lightweight cryptography standards, block ciphers, stream ciphers, and hash functions, respectively.

2.2 Constrained Devices and IoT Security

One of the main components of IoT is constrained devices. The constraints of these devices can be chip area, energy consumption, program code size, RAM size, communication bandwidth and execution time [6]. They can be used in many areas such as smart cities, smart homes, agriculture, transportation and healthcare to gather and send information since they are small and portable devices. However, their limitations can be determinative factors including communication range, application size

and its functionalities, data transfer speeds and units, storage etc. Therefore, most of the devices are designed to perform only one task. RFC 7228 [22] classifies constrained devices according to their data and code size, energy limitation and power usage. Data and code size show how many bytes can be stored in ROM and RAM. The classification of energy limitations refers to the type of power source, which can be event energy-limited, period energy-limited, lifetime energy-limited or no limitations. Use of power by constrained devices are categorized as follows: always-on, normally-off and low power.

The main challenge of IoT originates from its heterogeneity: various devices are assembled into a network and they interact with each other. Since these devices have different amounts of resources, their protocols should be adapted according to the devices' characteristics. However, conventional protocols do not satisfy the requirements of heterogeneity. From the viewpoint of security, compatible protocols with heterogeneous devices should be standardized for authentication, end-to-end security and data transfers. On the other hand, communication of heterogeneous devices, vary from sensors to servers, introducing new threats for IoT. According to the "Security Considerations in the IP-based Internet of Things" draft [48], these threats can be listed as cloning things, malicious substitution of things, eavesdropping attack, man in the middle attack, firmware replacement attack, extraction of security parameters, routing attack, privacy threat and denial of service attack. These threats can be examined in two groups: bootstrapping and operational phase. Bootstrapping phase is installation of the requirements of devices (firmware, key materials etc.) and codifying. Operational phase implies run time.

The lightweight cryptographic algorithms can be examined under two groups: hardware and software. The comparison of hardware implementations are performed with gate equivalents (GEs) and cell library that determines the energy consumption. In software implementation, the key factors are allocated space in RAM and ROM [9]. International Organization for Standardization (ISO) determines lightweight cryptography standards for general purposes [6], block ciphers [2], stream ciphers [10], asymmetric techniques [1], and hash functions [4]. Lightweight cryptography algorithms are classified as: symmetric ciphers, hash functions and public key cryptography. Additionally, symmetric ciphers are divided into two groups: stream cipher and block

cipher.

As a part of the IoT security, lightweight cryptographic algorithms are divided into three perspectives: performance, cost and security. To design new lightweight algorithms, security versus performance, security versus cost or performance versus cost trade-offs (Figure 2.3) are usually chosen. Because of the restrictions imposed by the constrained devices [80], balancing these factors in the same design is extremely hard. In fact, cryptographic algorithms form the backbone of security but security does not only consist of symmetric ciphers, asymmetric ciphers and hashing algorithms but also design of the system and its interaction, which determine the overall security. Public Key Cryptographic algorithms have more mathematical operations than other

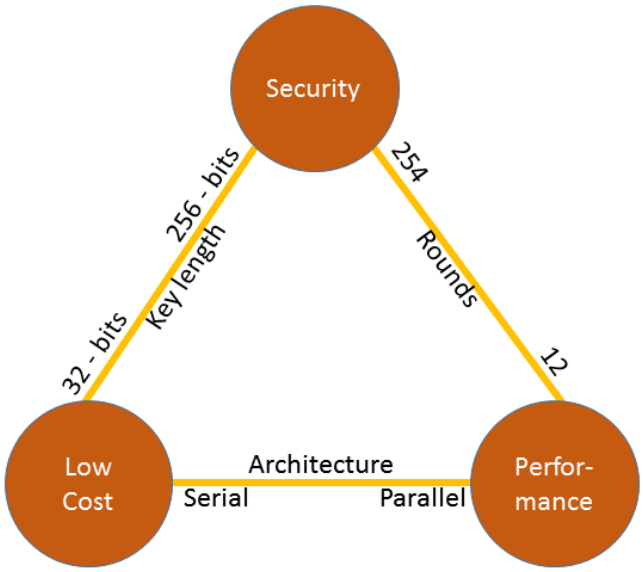

Figure 2.3: Design principles of lightweight cryptography (redrawn from [80])

algorithms and this affects power consumption. Indeed, a large portion of energy consumption occurs when wireless communication is used. In [78], it is mentioned that public key cryptographic algorithms consume as much energy as physical layer of the wireless communication stack. If public key algorithms are used, the lifetime of devices will be very short. Therefore, symmetric ciphers and hashing algorithms are usually preferred to provide long battery life.

2.3 Lightweight Block Ciphers

Lightweight block ciphers are designed by taking into consideration cell library, gate equivalence (GE), a number of rounds, block size, key size, and throughput to achieve optimal cipher. In the literature, there are proposed many block ciphers with different approaches such as fewer number of rounds, usually supporting a 64-bit block size with different key size options or smaller substitution boxes. We examine the main design issues of the proposed algorithms broadly in this section. As stated in the ISO standard [2], PRESENT [21] and CLEFIA [88] satisfy the lightweight cryptography requirements. Moreover, SIMON and SPECK [17] ciphers are analyzed to present different design concerns. To compare different lightweight block ciphers in the literature, we briefly epitomize the main characteristics of block ciphers in Table 2.1 [7].

Lightweight cryptographic algorithms are specially designed considering hardware or software implementations. Their requirements of power consumption, memory or CPU are optimized considering the selected cell library. To determine these requirements appropriately, algorithms should be tested in their practical environments. Due to the lack of these environments, we classify lightweight block ciphers using their software implementations' results on the same sensor [5, 23]. The number of encryption and decryption cycle of the algorithms on the same sensor gives an opinion about power consumption illustrated in Figure 2.4. This figure shows that we can divide lightweight block ciphers into three groups. From the minimum energy consumption to the maximum energy consumption, they can be listed as blue, red and green groups. Generally, those ciphers with larger block sizes or key sizes consume a larger amount of energy. Most of these algorithms perform encryption and decryption process with almost the same range of cycle counts except MCRYPTON and IDEA. Speck 64-bit block size with 96-bit and 128-bit key sizes has the minimum power consumption since it is a software-oriented cipher. This situation emphasizes the importance of implementation environments.

Table 2.1: Lightweight Block Ciphers [7].(NA = Not Available)

Algorithm	Block Size	Key Size	Rounds	Area (GE)	CellLibrary (µm)	Structure	Throughput (kb/s,100khz)	Power (µW)	Reference
CLEFIA	128	128 192 256	18 22 26	4950/5979 8536 8482	0.09	FN	355.6/711.1 NA NA	NA NA NA	[88]
DESL	64	64	16	1848	0.18	FN	5.55	0.89	[66]
DESXL	64	184	16	2168	0.18	FN	44.4	1.6	[66]
EPCBC	48 96	96	32	1008 1333	0.18	SPN	12.12	2.21 3.63	[102]
HIGHT	64	128	32	3048	0.25	FN	188.2	NA	[54]
KATAN	32 48 64	80	254	802 927 1054	0.13	LFSR	12.5 18.8 25.1	0.381 0.439 0.555	[30]
KTANTAN	32 48 64	80	254	462 588 688	0.13	LFSR	12.5 18.8 25.1	0.146 0.234 0.292	[30]
KLEIN	64	64 80 96	12 16 20	1220 1478 1528	0.18	SPN	30.9 23.62 19.1	NA NA NA	[43]
LBLOCK	64	80	32	1320	0.18	FN	200	NA	[99]
LED	64	64 128	32 80	966 1265	0.18	SPN	5.1 3.4	NA NA	[45]
mCRYPTON	64	64 96 128	12	2420 2681 2949	0.13	SPN	492.3 NA NA	NA NA NA	[67]
MIBS	64	64 80	32	1396 1530	0.18	FN	200 NA	NA NA	[56]
Piccolo	64	80 128	25 31	1136 1197	0.13	FN	237.04 193.94	NA NA	[86]
PRESENT	64	80 128	31	1570 1886	0.18	SPN	200	5	[21]
PRINTCIPHER	48 96	80 160	48 96	503 967	0.18	SPN	100 100	NA NA	[60]
Puffin	64	128	32	2577	0.18	SPN	NA	NA	[26]
Rectangle	64	80 128	25	1600 2064	0.13	SPN	246 246	74.31 72.15	[103]
SEA	96	96		449	0.13	FN	NA	3.213	[91]
SIMON	32 48 64 96 128	64 72/96 96/128 96/144 128/192/256	32 36 42/44 52/54 68/69/72	NA NA/763 838/1000 984/NA 1317/NA/NA	0.13	FN	NA NA NA NA NA	NA NA NA NA NA	[17]
SPECK	32 48 64 96 128	64 72/96 96/128 96/144 128/192/256	22 22/23 26/27 28/29 32/33/34	NA NA/884 984/1127 1134/NA 1396/NA/NA	0.13	FN	NA NA NA NA NA	NA NA NA NA NA	[17]
TWINE	64	80 128	36	1503 1866	0.09	FN	178 178	NA NA	[93]
XTEA	64	128	64	3490	0.13	FN	57.1	19.5	[70]

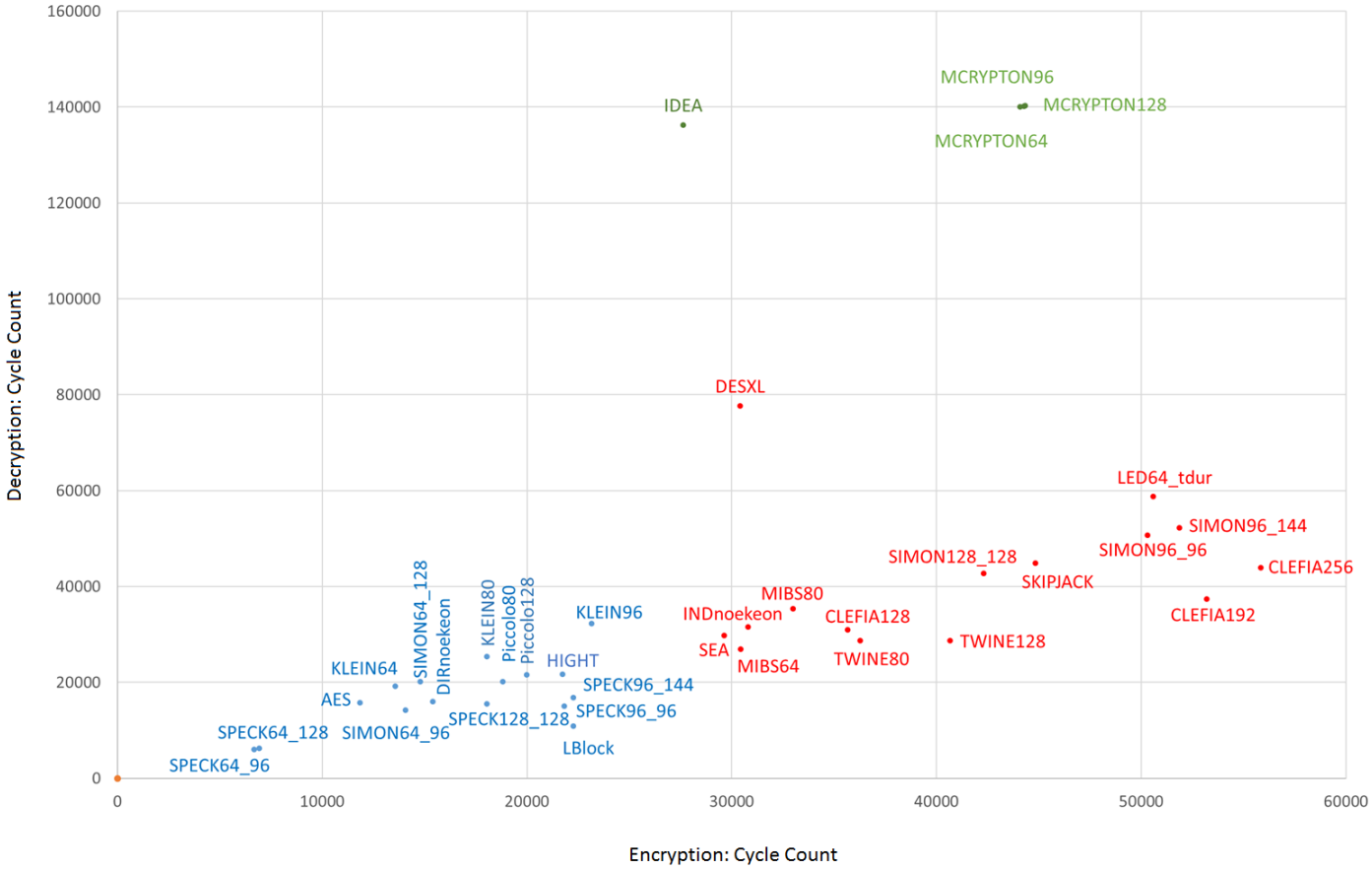

Figure 2.4: The number of encryption and decryption cycle of the algorithms on the same sensor.

2.3.1 PRESENT

PRESENT is one of the lightweight block ciphers standardized by the International Standardization Organization (ISO). The block size of the PRESENT is 64-bit and the key size can be 80-bit and 128-bit using 1570 GE. Present uses Substitution Permutation Network (SPN) [51], that has single 4-to-4 bit S-Box, regular bit permutation for P-Box and 31 rounds. The encryption and decryption processes need same physical resources. Moreover, encryption keys are processed on the fly. Although the present cipher is resistant to most known attacks, it is vulnerable to key rectangle attacks on 17 rounds and side-channel attacks [99].

2.3.2 CLEFIA

One of the other ISO compatible lightweight block ciphers is CLEFIA that its block size is a 128-bit with 128, 192, and 256-bit key lengths. Moreover, CLEFIA presents a different number of rounds: 18, 22, and 26. CLEFIA uses type-2 Feistel Network (FN) [53] that 32-bit F-functions and consists of 4 data lines in each round. To prevent cryptographic attacks, F-functions consist of Diffusion Switching Mechanism (DSM)[87] with two different diffusion matrices and S-Box. DSM not only reduces the number of rounds but also accelerates the encryption and decryption processes without compromising the security.

2.3.3 SIMON and SPECK FAMILY

SIMON and SPECK algorithms[17] are proposed by NSA to provide flexibility for different environments such as ASIC, FPGA, microcontroller and microprocessor. These algorithms consist of different block sizes and key sizes. In addition, they provide high throughput on these platforms. The flexibility provided by SIMON and SPECK is used for applying security primitives to a wide range of constrained devices. This flexibility can be divided into two groups: hardware and software oriented. Therefore, SIMON is designed for hardware and SPECK is designed for software. SPECK has fewer rounds than SIMON and SPECK can reuse key scheduling.

2.4 Lightweight Stream Ciphers

Stream ciphers encrypts data, a bit or byte stream, using keystream which they produce. Stream ciphers have to be fast and use low memory compared to block ciphers contrary to block ciphers. There are recommended two stream ciphers by ISO: ENOCORO [97] and TRIVIUM [29]. Table 2.2 [7] presents several stream ciphers and their properties.

2.4.1 ENOCORO

Enocoro stream cipher uses PANAMA like keystream generator (PKSG) [27], which uses 80 bit and 128-bit key length with 64-bit Initialization Vector (IV) for encryption and decryption. The internal state of Enocoro consists of two functions, called ρ and λ. The λ function contains XOR operations and byte-wise rotation. The ρ function consists od S-Box and linear transformation. Substitution-Permutation-Substitution structure provides stronger encryption contrary to SPN. Enocoro performs 80 bit and 128 bit key lengths operations with 2.7 K and 4.1 K gate area, respectively.

2.4.2 TRIVIUM

Trivium is a synchronous stream cipher that uses 80-bit IV and 80-bit key length. As a part of ESTREAM Project [11], Trivium has a hardware efficient design. Moreover, its compact design provides a suitable environment for constrained devices. Applications that require high throughput is performed fast by using Trivium. The algorithm uses three feedback registers in the internal state with different key lengths, 84, 93, and 111 bits. In addition, 2^{64} keystream bits are produced for each IV and key.

Table 2.2: Stream Ciphers and their properties [7].

Algorithms	Key Size	IV Size	Cell Library (µm)	Area (GE)	Throughput (kb/s,100khz)	Power (µW)	Reference
TRIVIUM	80	80	0.13	2599	100	5.5	[29]
ENOCORO	80 128	64	0.18	2.7K 4.1K	28	NA NA	[97]
GRAIN	80 128	64 80	0.13	1294 1857	0.1 0.1	3.3 4.34	[49]
MICKEY2.0	80 128	80 128	0.18	3188 5039	NA NA	NA NA	[15]

2.4.3 GRAIN

Grain is another hardware-efficient synchronous stream cipher. Grain uses 64-bit and 80-bit IV as inputs with 80-bit and 128-bit key length, respectively. The main components of the grain are Non-linear Feedback Shift Register (NFSR) [39], Linear Feedback Shift Register (LFSR) [76], and output function. NFSR and nonlinear output function guarantees the the nonlinearity of the algorithm. Moreover, LFSR guarantees that the period of keystream is large and provides a good balance for the output. In case of the expandable hardware, the performance of the Grain increases. Due to some weaknesses, the recommended versions of Grain can be found in [13, 49].

2.5 Lightweight Hash Functions

A Hash function maps arbitrary data to a fixed size output. It is widely used in cryptography for data integrity, digital signatures and so on. A perfect hash function should resist to the collision, preimage, and second preimage attacks. Although There are many design structures for hash functions, lightweight has functions are built on Merkle-Damgard [75] and Sponge [18]. ISO 29192-5 [4] recommends three hash functions: PHOTON [44], SPONGENT [20] and Lesamnta-LW [52]. Table 2.3 presents the main features of hash functions [7].

2.5.1 PHOTON

PHOTON is a hash function which is designed for extremely constrained devices. Domain extension algorithm of PHOTON is designed with sponge construction with different internal state and output digest sizes: 100, 144, 196, 256 and 288 bits and 80, 128, 160, 224, 256, respectively. Although the internal memory used by sponge construction is low, it reduces the speed of the small message hashing. The flexibility of the algorithm is achieved by reducing time spending in squeeze process. It is also reduces preimage-resistance as far as possible.

Table 2.3: Lightweight Hash-Functions and their properties [7].

Algorithms	Size	Round	Int. Size	Cell Library (μm)	Area (GE)	T.put (kb/s, 100khz)	Power (µW)	Collision resistance	Preimage resistance	Second Preimage	Ref.
PHOTON	80	12	100	0.18	865/1168	2.82/15.15	1.59	64	40	40	[44]
	128	12	144	0.18	1122/1708	1.61/10.26	2.29	112	64	64	
	160	12	196	0.18	1396/2177	2.70/20	2.74	124	80	80	
	224	12	256	0.18	1736/2786	1.86/15.69	4.01	192	112	112	
	256	12	288	0.18	2177/4362	3.21/20.51	4.55	224	112	112	
SPONGENT	88	45	88	0.13	738/1127	0.81/17.78	1.57/2.31	80	40	40	[20]
	128	70	136	0.13	1060/1687	0.34//11.43	2.20/3.58	120	64	64	
	160	90	176	0.13	1329/2190	0.4/17.78	2.85/4.47	144	80	80	
	224	120	240	0.13	1728/2903	0.22/13.33	3.73/5.97	208	112	112	
	256	140	272	0.13	1950/3281	0.17/11.43	4.21/6.62	240	128	128	
Lesamnta-LW	256	NA	384	0.09	8240	125.55	NA	128	256	256	[52]
QUARK	136	504	136	0.18	1379/2392	1.47 / 11.76	2.44 /4.07	128	64	64	[14]
	176	704	176	0.18	1702/2819	2.27 / 18.18	3.10 /4.76	160	80	80	
	256	1024	256	0.18	2296/4640	3.13 / 50.0	4.35 /8.39	224	112	112	
ARMADILLO2	80	NA	256	0.18	4030 /2923	109 / 27	NA	40	80	80	[16]
	128	NA	384	0.18	6025 /4353	1000 / 250	NA	64	128	128	
	160	NA	480	0.18	7492 /5406	100 / 250	NA	80	160	160	
	224	NA	576	0.18	8999 /6554	100 / 25	NA	96	192	192	
	256	NA	768	0.18	11914/8653	100 / 25	NA	128	256	256	

2.5.2 SPONGENT

SPONGENT achieves robustness by performing PRESENT type permutation with sponge construction. It provides different hash sizes, 88, 128, 160, 224 and 256, with different number of rounds, 45, 70, 90, 120 and 140. By using serialization, SPONGENT provides compact design. Moreover, it produces small footprint. To minimize footprint, SPONGENT algorithm uses simple round function with 4-to-4 bit S-Box.

2.5.3 Lesamnta-LW

Lesamnta-LW is a 256-bit hash function with Merkle-Damgard domain extension algorithm [52]. It provides at least 120-bit security. Unlike other hashing algorithms standardized by ISO, this algorithm uses 128-bit key for hashing. Moreover, algorithm uses different compression function called LW1, that occupies less memory, since the key size is smaller than the block size.

2.6 Physically Unclonable Function

A Physically Unclonable Function (PUF) [50, 81, 95] is an integrated circuit (IC) that takes a certain input (challenge) and produces a certain output (response) without storing in digital memory:

$$f : X \rightarrow Y, X \in \{0,1\}^{\lambda}, Y \in \{0,1\}^{\eta},$$

where f is a one-way function that maps X to Y. Each PUF generates a different response to the same challenge due to the fabrication process of ICs that provides uniqueness for PUFs since it is extremely hard to produce the same IC. Therefore, PUFs are unclonable. Moreover, data, while producing a response, cannot be read since it is tamper-evident. If there is any physical attempt to read, it will generate totally different responses from expected responses since this attempt directly affects the generation procedure of PUF. It also generates responses only in online mode. Because of the aforementioned reasons, PUFs can be used for security purposes such

as authentication, data integrity, key generation, access control, and privacy. Besides, a PUF provides a cheap and energy-efficient environment since it is only responsible for the generation of outputs by using integrated circuits, which only consist of different series of logic gates. In order to perform a security task with PUF, the following procedure is usually performed:

- The server chooses challenges and stores the responses corresponding to these challenges from PUF before deployment. These challenges and responses are used for performing security primitives,
- After deployment of PUF, each communication attempt is verified with a challenge-response pair (CRP),
- When the server starts the communication, the server sends the challenge to the node,
- The PUF on the node produces a response using this challenge and sends it to the server,
- The server verifies the response using the stored values.

There are two types of PUFs, weak and strong. These types are related to the size of the chip. The chip size determines the number of CRPs that can be generated by a PUF. Since the chip size of a weak PUF is small, the number of CRPs is small. The CRPs of weak PUF are usually used as a pre-shared key. However, strong PUFs can generate about half a million different outputs in which each transaction can be secured with different CRPs. Rather than using only a few shared keys, thousands of different keys provide excellent security for communication since each authentication attempt or transaction is performed with a different key. Therefore, it is assumed that the proposed protocol in this thesis uses strong PUF. The different types of PUFs [81] can be summarized as follows:

Weak PUFs:

- **Static Random-Access Memory (SRAM) PUF**: contains a pair of cross-coupled of inverters. The produced bit is determined according to the value of fastest inverter.

- **Ring Oscillator (RO) PUF**: comparison of self-oscillating loops that consist of odd number of inverters are used to generate response bits.

Strong PUFs:

- **RO Sum PUF**: m pairwise frequency differences are used to generate response bits.
- **Loop PUF**: comparison of single reconfigurable paths in every stage produce response bits.
- **Arbiter PUF**: comparison of two reconfigurable paths that have nominally identical delays are used to generate response bits.

PUF is very suitable for constrained devices since it consumes a low amount of energy when it generates responses. On the one hand, PUF is vulnerable to environmental factors such as temperature and voltage variations that cause noise on responses. These factors also affect cryptographic operations since different responses can be generated for the same challenge. Therefore, error correction mechanisms as secure sketches are used to generate the same response corresponding to the same challenge, even so, environmental factors cause noise. On the other hand, PUF responses are not uniformly random and are not used as cryptographic keys. Therefore, a fuzzy extractor, which is a one-way function, is used to produce uniform random responses. In the literature, some proposed algorithms perform operations without using secure sketches and fuzzy extractors.

2.6.1 Secure Sketch

Secure sketch is an error correcting algorithm that consists of two parts: sketch (SS) and recover (Rec).

- Sketching procedure takes an input $w \in \mathcal{M}$, where $\mathcal{M}$ is a metric space with distance function $dis()$, then returns an output of bit string $s \in \{0,1\}^*$.

- Recover procedure takes an input $w^{'} \in \mathcal{M}$ and bit string $s \in \{0,1\}^{*}$. This process assure that $Rec(w^{'}, SS(w)) = w$ if $dis(w, w^{'}) \leq t$, t is a threshold distance value in the metric space.

- The security property guarantees that for any W over $\mathcal{M}$ with min-entropy m, an adversary can recover $w \in W$ if an adversary eavesdrops s with a probability smaller or equal to $2^{-\widetilde{m}}$

2.6.2 Fuzzy Extractors

Fuzzy extractor [34] maps two very close strings to same value using helper data in order to get rid of the noises on data. There are two phase for fuzzy extractor: Generation (Gen) and Reproduction (Rep).

- For any string $w \in \mathcal{M}$, the generation function, using w, produces two outputs: an extracted string $R \in \{0,1\}^{l}$ and a helper string $P \in \{0,1\}^{l}$.

- The reproduction phase takes a $w^{'} \in \mathcal{M}$ which is very close to w and a helper string $P \in \{0,1\}^{l}$. If distance between w and $w^{'}$ is under some threshold t, the reproduction function assures:

$$Gen(w) = (R, P) = Rep(w^{'}, P). \tag{2.1}$$

- The security property guarantees that for any W over $\mathcal{M}$ with min-entropy m since R is nearly uniform distributed

Moreover, fuzzy extractors can be used with secure sketchs to minimize the impact of noise due to environmental factors as shown in Figure 2.5.

2.7 Group Communication

Group communication algorithms provide efficiency when too many devices communicate among each other. Instead of one-to-one communication that requires agreement on the secret key for each communication attempt to be secure, the group members share information using a group key. Since the group key is shared with members

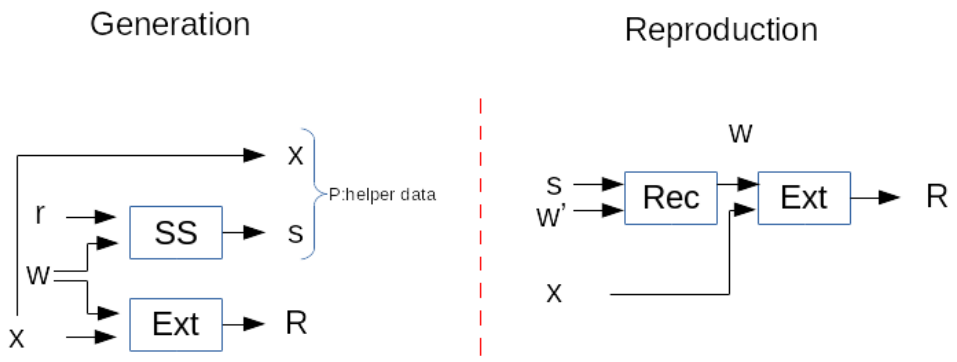

Figure 2.5: Generation (Gen) and Reproduction (Rep) procedure of fuzzy extractor (redrawn from [57]).

in the group initialization phase, the cost of the one-to-one communication for key agreement is eliminated. However, this method causes a vulnerability when adding or evicting operations are performed for group membership. Therefore, group communication algorithms must guarantee backward and forward secrecy:

- When any node is added to a group, the group key must be updated for backward secrecy to prevent the decryption of older messages in the system. New members must not be able to read old messages.
- In the same way, forward secrecy must be guaranteed by updating the group key that prevents accessing the new messages when the group member is evicted.

There are three types of group communication architectures: centralized, decentralized, and distributed [24, 41] as illustrated in Figure 2.6. In distributed architectures, all group members join the decision making procedure. Each member of the group shares their owns message with other members and agrees on the group key. Therefore, this process requires too many network messages and computations. In centralized architectures, there exists a group leader that manages whole communication. The group leader generates the required materials and distributes them to the nodes. Moreover, each member only communicates with the group leader in contrast to distributed systems. The decentralized architectures are almost the same with cen-

tralized systems except that decentralized systems have more than one group leader. This system resolves the single-point-of-failure challenge and distributes the load to the group leaders.

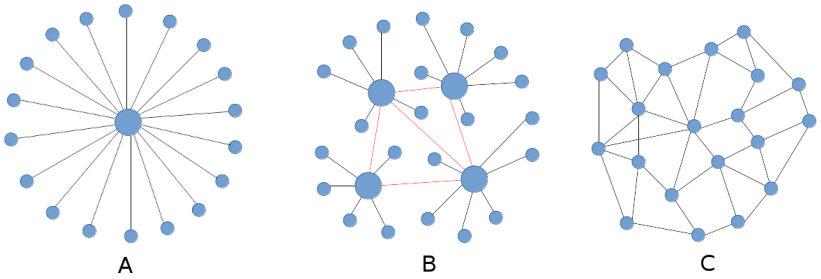

Figure 2.6: Group communication systems: A) Centralized B) Decentralized C) Distributed

IoT devices cannot handle a large amount of communication overhead. They spend the most energy during communication. Therefore, minimizing the communication overhead is one of the key issues for the IoT environment. However, it cannot be achieved in distributed architectures. Centralized and decentralized group key distribution methods provide a suitable environment for group communication due to the limitations of these devices. Instead of communicating with each participant, group members only communicate with the group leaders.

2.8 Tree-based Group Key Distribution

There are proposed algorithms for group key distribution in centralized structures [33, 42, 61, 68, 79, 85, 90, 101, 104, 105]can be examined under two phases: initialization and updating the group key. For the initialization phase, group authentication and key distribution require $O(n)$ operations for n members since group members must be verified, and the group key must be distributed one by one to members. The most complicated part for group operation is adding a node to a group or evicting a node from a group since the group key is updated. These operations require that all nodes of the group are informed to update the group key except the added/evicted node. The communication cost of the update procedure is calculated as the sum of the unicast, multicast, and broadcast messages. The communication cost is inversely proportional

to the storage cost that the amount of space required to store key materials in each node. These operations have $O(n)$ complexity in the worst-case to distribute a new group key to all group members. The server sends a new group key to all nodes one by one and each node only stores the group key and the shared key. The best-case scenario, which is $O(1)$, is performed by storing all possible keys in all nodes [38]. Although only a broadcast message is sufficient for a key renewal process, storage cost is extremely high in this scenario for all nodes in a group. In order to reduce communication costs between the server and nodes, algorithms based on a binary tree are preferred by using key hierarchy. In key hierarchy approach, nodes are divided into different subgroups and each subgroup has different sets of key materials. By using these sets, the cost of the key renewal process is reduced. In binary tree structures, the group key is located at the root node within $Level\ 0$ as illustrated in Figure 2.7. Group members are located in leaf nodes within $Level\ 3$ at the same level, and the rest of the nodes store internal keys within $Level\ 1, 2$ that are used for the key renewal process. Since nodes only store the keys from the leaf node to the root node on the shortest path, the storage overhead of group members is only $log(n)$ keys for this structure. Although binary tree structures increase storage requirements, the key renewal process is reduced to logarithmic complexity. The key renewal process must include the renewal of the internal keys since the adding/evicting operations with the same internal keys cause information leakage. Instead of updating all internal keys, only internal keys, which are on the key path of the added/evicted node, are updated. For a detailed comparison, the required number of messages for key renewal and stored keys for k-ary trees are represented in Table 2.4.

2.8.1 Factorial Tree

In group communication for n nodes, a binary tree reduces the communication cost of both adding and evicting operations on the network where each node in a binary tree has at most two children. However, each node on the group must store $log(n)$ keys. In order to reduce the storage cost for n nodes, factorial trees can be used. Unlike the binary tree, each node in factorial tree has at most $(t + 2)$ children at level t as shown in Figure 2.7. The total number of nodes of each level is calculated by using factorial of the $(t + 1)$, as shown in Table 2.4.

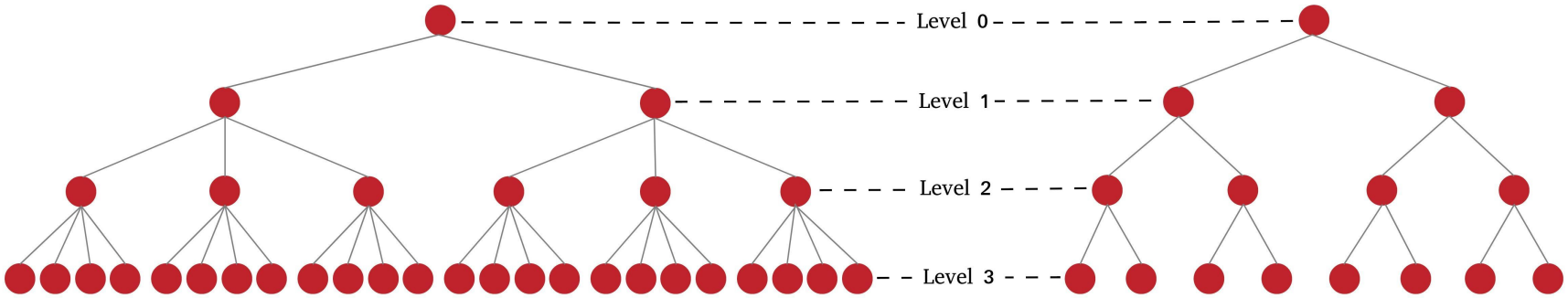

Figure 2.7: Factorial and binary tree key hierarchy. The number of leaf nodes per level is 2^t and $(t+1)!$ in binary tree and factorial tree, respectively where t is the level of the tree.

Table 2.4: The number of nodes and key update messages for complete Factorial Tree and k-ary tree (t: level of the tree, n: the number of total nodes in tree).

	Factorial Tree	k-ary Tree
# nodes per level	$(t+1)!$	k^t
# key update messages	$t(t+1)/2$	$(k-1)\log_k n$

The number of keys stored in each leaf node for a binary tree is $log(n)$ where n is the number of nodes and t is the level of the tree for a leaf node. Also, the key renewal process (adding or evicting nodes) takes $log(n)$ messages for the whole tree. However, a factorial tree contains more nodes than the binary tree starting from Level 2 as shown in Table 2.4. Therefore, the number of keys stored in members is lower than the binary tree, which is shown in Table 2.5. Although Factorial Tree accommodates more group members at lower levels, the key renewal process takes $t(t+1)/2$ messages on the network, which is higher in comparison with the binary tree. Our aim is to reduce total messages in the network while increasing the number of nodes. Therefore, the Chinese Remainder Theorem is used to minimize the number of messages.

Table 2.5: The number of stored keys on nodes with different tree structures.

# of Group Members	**Factorial Tree**	**Binary Tree**	**3-ary Tree**	**4-ary Tree**	**5-ary Tree**	**6-ary Tree**
16	4	5	4	3	3	3
32	5	6	5	4	4	3
64	5	7	5	4	4	4
128	6	8	6	5	5	4
256	6	9	7	5	5	5
512	6	10	7	6	5	5
1024	7	11	8	6	6	5
2048	7	12	8	7	6	6
4096	7	13	9	7	7	6
8192	8	14	10	8	7	7

2.9 The Chinese Remainder Theorem (CRT)

The proposed protocol in this thesis achieves the key-renewal process by significantly reducing communication overhead with CRT. In the key-renewal process for tree structures, internal nodes are used to share the new group key with group members. However, the number of horizontal internal nodes in a factorial tree is higher than the binary tree beginning from level 2, which causes the number of messages to increase. To reduce the number of messages originating from horizontal nodes for updating the group key, PLGAKD that we present in Chapter 4 uses CRT that combines multiple messages into one.

Let $k_1, k_2, \ldots, k_n$ be pairwise relatively prime integers ($gcd(k_i, k_j) = 1$ for $i \neq j$) greater than 1 and let $r_1, r_2, \ldots, r_n$ be integers. The system of simultaneous congruences:

$$\begin{aligned} x &\equiv r_1 \, mod \, k_1, \\ x &\equiv r_2 \, mod \, k_2, \\ &\vdots \\ x &\equiv r_n \, mod \, k_n, \end{aligned}$$

has a unique solution in $mod\ K$ where $K = k_1 k_2 \ldots k_n$. Specifically, x is between 0 and $k_1 k_2 \ldots k_n$. Then, x is calculated as follows:

$$x = \sum_{i=1}^{n} r_i K_i K_i^{'} \quad (mod\, K),$$

where

$$K_i = K/k_i$$

and $K_i^{'}$ is the multiplicative inverse of $K_i \ (mod \ k_i)$ that is $K_i K_i^{'} \equiv 1 \ (mod\, k_i)$. Suppose that, we wish to solve

$$\begin{aligned} x &\equiv 4 \quad (mod\ 7) \\ x &\equiv 3 \quad (mod\ 11) \end{aligned}$$

By using above formula:

$$
\begin{aligned}
K &= 7 \times 11 = 77 \\
K_1 &= 77 \,/\, 7 = 11 \\
K_2 &= 77 \,/\, 11 = 7 \\
K_1^{'} &= 2 \pmod{11} \\
K_2^{'} &= 8 \pmod{7} \\
x &= 4 \times 11 \times 2 + 3 \times 7 \times 8 \pmod{77} \\
&= 25
\end{aligned}
$$

CHAPTER 3

RELATED WORK

In the literature, there are many proposed group authentication and key distribution algorithms, which mainly focus on tree-based structures, Shamir secret sharing, CRT, ECC and aggregate message authentication codes we summarize in Table 3.1.

Fiat and Naor [38] proposed an information-theoretic approach for broadcast transmissions that master node broadcasts messages to dynamic groups by minimizing key issues. This method consists of different schemes based on broadcasting messages securely, while a coalition of malevolent users tries to find secret key. In this method, the main computational cost arises from the resiliency number. The total key store for each user is $O(dlogdlogn)$, where d is a resiliency number, and n is the number of users in the system. Also, the master node broadcasts $O(d^2log^2dlogn)$ messages.

Wong *et al.* present key graphs for group communications to resolve the scalability problem of large networks [98]. In this method, star, tree, and complete topologies are analyzed with user-oriented, key-oriented, and group-oriented rekeying options. The key tree graph is more suitable for rekeying operations since it reduces rekeying operations to logarithmic time. It is especially essential for large groups.

Luk *et al.* emphasize seven properties for sensor networks broadcast authentication, which are resistant to node compromise, low computation and communication overhead, message reliability, authentication delay, asynchronous message, high message entropy [71]. They proposed two authentication methods using hash chains that satisfy the six of seven properties. One of them performs authentication in regular and predictable times. The other performs authentication with low entropy messages.

In [47], Lein proposed threshold-based group authentication using Shamir Secret

Sharing. A group manager calculates tokens by using Lagrange interpolation according to the threshold and the number of group members. After the generation of tokens, the group manager sends tokens to the nodes. Each node calculates responses via tokens and shares responses with all other nodes. After collecting all responses, nodes use Lagrange interpolation to authenticate. If verification is successful, then all nodes are authenticated, and they get secret from the message for group communication. If verification fails, the authentication procedure is canceled, and it is guaranteed that at least one node sent the wrong message, but their identity could not be determined. In [73], Mahalle *et al.* use Paillier Threshold Cryptography based on Shamir Secret for authentication and distribution of the group key like [47]. In this method, the server generates a public key for its, and multiple private keys for nodes. Private keys are distributed to the nodes securely. For group operations, every node in the system calculates its own messages and sends other nodes. Then, all messages are combined for verification. After successful verification, all nodes are authenticated. If any node sends the wrong message, it cannot be authenticated. Both methods require $O(n^2)$ operations in order to authenticate and obtain group secret. This method performs less communication than Lein's algorithm.

In [69], Liu *et al.* use CRT for group key distribution. Authentication and key distribution perform in two different phases. Group members can recover group key using CRT. Verification of group members is performed by using secrets distributed in the registration phase. Zheng [105] proposed only group key management methods by using CRT, which are CRGK and FCRGK. In CRGK, the master node computes CRT by using the user's information. Then, broadcast this information to the users. After obtaining the broadcast message, each user computes the group key by using its own key (one modulo and one XOR operation). Adding or evicting operation in this method has $O(1)$ complexity for users. However, the master node must perform all computations to generate a new group key from the beginning. The difference between CRGK and FCRGK method is that FCRGK protocol performs operations on the large group size to accelerate join and leave operations. Another CRT-based security approach is proposed by Kavin and Ganapathy [58]. This method proposes two encryption schemes for secured storage and privacy-preserving model.

Yao *et al.* proposed a lightweight multicast authentication mechanism using revised

Nyberg's fast one way accumulator with some changes [101]. This algorithm uses shared keys and hash-based message authenticaon code (HMAC) for data integrity and source verification. In [59], Khalid et al. proposed a blockchain-based authentication using fog computing for constrained devices. In this design, firstly, nodes are registered by the closest blockchain-enabled fog node. Then, the identities of the nodes are stored on the other blockchain servers. Whenever nodes want to authenticate, they send only their identification credentials to the system.

In [64], Lai *et al.* propose group authentication using lightweight public-key infrastructure for LTE networks. In this study, a group leader is chosen according to hardware capacities. The group leader initializes the authentication procedure between Mobile Management Entity (MME) and Home Subscriber Server (HSS). After the verification procedure, a temporary group key is generated, and the group leader is authenticated. Other entities in the group perform mutual authentication with MME. Backward/forward Secrecy is guaranteed by using Elliptic Curve Diffie Helman (ECDH), and public-key cryptography is used to protect user's privacy. There is a similar method proposed by Lai *et al.* that designed different group authentication protocol for constrained devices using aggregate Message Authentication Codes (MAC) [63]. In this method, authentication is performed by using XOR of MACs. Each device has a private id and a pre-shared key in the initialization phase. Other operations are similar to [64], except that authentication is performed by using symmetric cryptography. They also extended their methods in [65].

In [35], Dong *et al.* propose a group communication method for Body Area Networks (BAN). A BAN network consists of sensors and a control unit (CU) with a strong PUF. In the initialization part, each sensor has a unique id and establishes a secret key with the CU. In the generation and distribution part, one of the sensors sends a group request to CU. It generates a unique group name by using sorted IDs of sensors. Group secret is generated by using the group name. Then CU distributes key using sensors secret. Since CU's PUF is unique, other entities or eavesdroppers cannot imitate this procedure.

Table 3.1: Group Communication Algorithms. (ECDH: Elliptic Curve Diffie Hellman, MAC: Message Authentication Code, PKC: Public Key Cryptography, LKH: Logical Key Hierarchy, HMAC: Hash-Based Message Authentication Code).

Reference	Architecture	Method	Security Primitives	IoT-based
[73]	Distributed	Paillier Threshold Cryptography, Shamir Secret Sharing	PKC	No
[47]	Distributed	Shamir Secret Sharing	PKC	No
[64]	Centralized	Not a specific method	ECDH, PKC	No
[65]	Centralized	Aggregate MAC	Symmetric Encryption, Hash	Yes
[69]	Centralized	CRT, Shamir Secret Sharing	PKC	No
[63]	Centralized	Aggregate MAC	Symmetric Encryption, Hash	Yes
[79]	Centralized, Distributed	Not a specific method	ECC, Hash	Yes
[42]	Centralized	Not a specific method	ECC, One-way accumulator	Yes
[101]	Centralized	Nyberg's fast one-way accumulator	Hash, HMAC	Yes
[55]	Centralized	PUF-based	Hash, Symmetric Encryption	Yes
[35]	Centralized	PUF-based	Hash, Symmetric Encryption	Yes
[61]	Centralized	LKH, CRT	ECC, Hash, Symmetric Encryption	Yes
[36]	Centralized	Lamport's Scheme, CRT	Hash, Symmetric Encryption	Yes
[77]	Centralized	Tree-based	ECC, MAC	Yes
[105]	Centralized	Key Tree, CRT	Not mentioned	No
[104]	Centralized	CRT	XOR, addition, multiplication	No
PLGAKD	Centralized	PUF, Factorial Tree, CRT	XOR, Hash, Symmetric Encryption	Yes

In [55], Huang *et al.* proposes a PUF-based group key distribution scheme that consists of three phases. In the preliminary phase, all CRPs are stored on the Main Controller (MC). In the coordination phase, group key and coordination messages are generated using stored CRPs by MC. Then, they are sent to the Auxiliary Controller (AC). In the distribution phase, AC verifies the message and sends a message to the node. Node decrypts the message by using CRP. If all operations on the system are accepted, the node sends the message to the MC, forwarded by AC, and authentication and key distribution are completed. In this phase, AC decrypts only related parts of the message. The remaining part of the message is decrypted only by the node. This method uses AES, hash, XOR, and pseudorandom number generator (PRNG) operations while constructing and verifying messages and majority vote as an error correction mechanism.

CHAPTER 4

A PUF-BASED LIGHTWEIGHT GROUP AUTHENTICATION AND KEY DISTRIBUTION PROTOCOL

In this chapter, a PUF-based Lightweight Group Authentication and Key Distribution Protocol (PLGAKD) is presented in detail. Firstly, overall system and security model is defined. Then, initialization and group authentication and key distribution phase of the protocol is explained. Finally, thre required steps for key renewal process is explained when a node is added to the group or evicted from the group.

4.1 Overall System and Security Model

While a large number of IoT devices are communicating, some bottlenecks may occur on both networks and devices. Therefore, the PLGAKD protocol is designed to resolve these bottlenecks by using PUF, factorial tree, and CRT. In this protocol, we assume that a high-performance server manages IoT devices over a gateway as illustrated in Figure 4.1. Due to the constraints of IoT devices, the PLGAKD protocol performs costly operations on the server-side without causing security weaknesses on the node side. The server's task is to authenticate the IoT devices and distribute the required key materials to the IoT devices in a more efficient manner. In order to perform a lightweight authentication and key distribution, each node is equipped with a PUF, which produces different CRPs for verification. Before deploying the nodes, the server stores CRPs for authentication. Since storing all possible CRPs may cause storage problem, the server may only store a set of CRPs.

In group communication such as installing a new firmware, updating configurations, or sharing data between group members, the costly operations are performed when

adding a node to the group or evicting a node from the group since the group key must be changed. These costs can be reduced by using tree structures that help us perform these operations with logarithmic storage and communication cost. However, these costs can be further reduced by using factorial tree and CRT. In this protocol, the factorial tree is preferred since it accommodates more nodes than the binary tree at the same levels, which reduces storage costs. However, reducing the storage overhead increases communication messages. In order to reduce communication costs, the key-renewal messages sent with the internal keys of the same level are combined into one by using CRT.

In this protocol, outside attackers cannot access the secret information since each communication is encrypted with a different responses. In addition, there is no chance to impersonate the PUF since it is unclonable. Moreover, inside attackers cannot decrypt messages even if one of the CRPs is the same for nodes. The PLGAKD protocol performs cryptographic operations with CRP and secret key of nodes. This protocol also assumes that group members are trusted in the initialization phase.

4.2 PUF-based Lightweight Group Authentication and Key Distribution Protocol (PLGAKD) Protocol

In the literature, proposed group-based algorithms are based on Public Key Cryptography (PKC) that provides confidentiality, authentication, and non-repudiation by performing costly operations. Moreover, conventional protocols require too much communication and storage overhead. However, IoT devices cannot perform such operations because of their constraints. To provide secure communication for IoT environment, lightweight ciphers and algorithms must be in the same design. Therefore, a lightweight protocol must be designed by taking into account computation, communication, and storage overhead. To reduce computation overhead in PLGAKD protocol, symmetric ciphers, hash functions, and XORing operations are used with PUF that provides cheap and energy-efficient key generation. As aforementioned in Section 2.8, there is a trade-off between communication and storage overhead. Therefore, factorial trees and CRT are chosen for efficiency. Since a factorial tree has more horizontal nodes than the binary tree, it accommodates more nodes than the binary

tree for the same level. However, the number of messages for the key renewal process increases because of the horizontal nodes. This disadvantage is eliminated via CRT that helps reduce the number of messages.

The PLGAKD protocol assumes that each node in the group has at least one PUF on its hardware, a centralized architecture is used for group key distribution and the set of CRPs is collected in manufacturing process. The collected CRPs are stored on the server in the initialization phase. Moreover, this protocol can be adapted to decentralized architectures. PLGAKD protocol consists of initialization phase and group authentication and key distribution phases as shown in Figure (4.1).

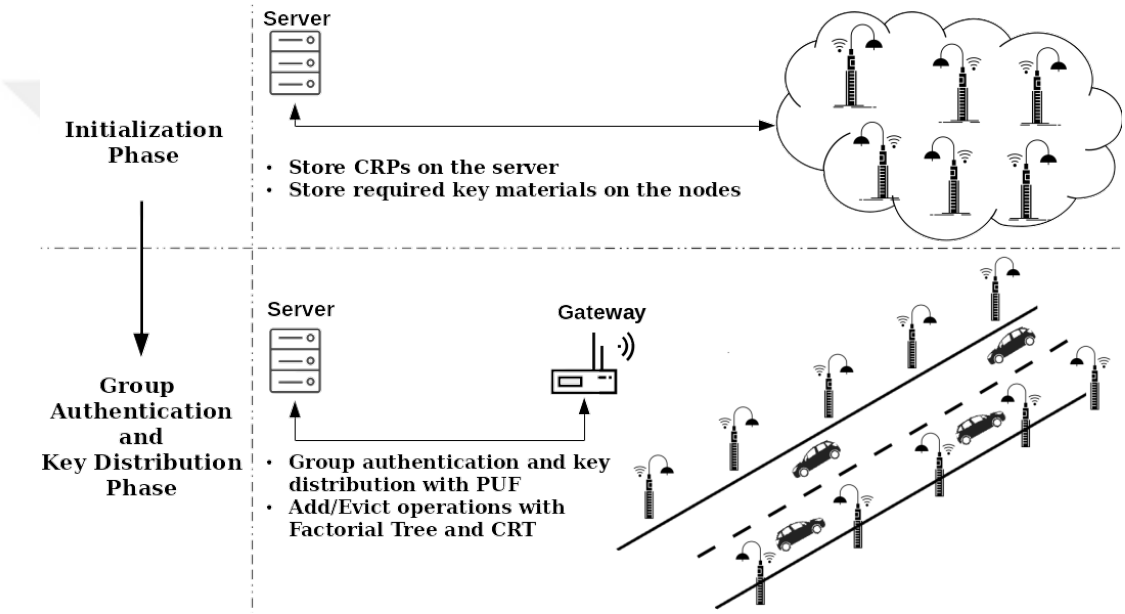

Figure 4.1: System architecture diagram.

4.2.1 The Initialization Phase

Before each node participates in the group, there are two required operations for group communication. Firstly, the server determines the set of challenges. These challenges are used for authentication and key distribution. Then, it stores the set of challenges with the corresponding responses of each node before PUF nodes are deployed. Although physical differences of each PUF generate different CRPs, the same responses can be found for the same challenges in the group members. Therefore, the server as-

Table 4.1: Nomenclature.

Notation	Description
CRP	Challenge-Response Pair
$C_{i,j}$	j^{th} challenge of $Node_i$
$R_{i,j}$	j^{th} response of $Node_i$
G_K	Group key
T_{GK_i}	Temporary group key of the $Node_i$
I_s	Public identity of the server
ID_i	Public identity of the $Node_i$
K_i	Secret key of $Node_i$ shared with the server
K_s	Secret key of the server
G_{ID}	Group ID
T_{GID_i}	Temporary group id of $Node_i$
P_i	Prime numbers of $Node_i$
$Hash()$	Hash function
$HMAC_K(M)$	Hash-based message authentication code
	M is the message and K is the secret key
ECC	Error correcting code
N_i	Nonce of $Node_i$
N_s	Nonce of the server
E_A	Symmetric encryption with key A
D_A	Symmetric decryption with key A
M_i	Message for $Node_i$
c_i	Ciphertext for $Node_i$
T	Timestamp

signs a unique public ID and a secret key (K_i) to each node and stores them on the server and the node. Public IDs are used to identify nodes and secret keys are used with CRPs to prevent CRP collisions.

4.2.2 The Group Authentication and Key Distribution Phase

In this section, the server performs lightweight operations to authenticate nodes and distributes the group key. The server generates the group key by combining the secrets of the group members. This protocol sends the temporary group key to the nodes. Then, the node performs some operations to obtain the actual group key. Also, the node stores the temporary group key instead of the actual group key. When the node sends messages, encrypted with the group key, it computes the actual group key using a temporary group key. In PLGAKD protocol, the following steps are required for group authentication and key distribution:

Step 1: The server starts the authentication phase by sending a message to each member that contains a challenge (C_i), message authentication code (MAC_i), identity of the server (I_s), and the timestamp (T) as:

$$\begin{aligned} MAC_i &= HMAC_{R_i \oplus K_i}(C_i||I_s||T), \\ auth_i &= C_i||MAC_i||I_s||T. \end{aligned} \tag{4.1}$$

In this authentication message ($auth_i$), MAC_i is obtained by processing with C_i, I_s, and T as a message and both response (R_i) and K_i as a cryptographic key so that group members can authenticate the server and validate the integrity of the message.

Step 2: Since each node in the system has its K_i and generates its own response R_i corresponding to the challenge, each node produces its own MAC_i value with C_i, I_s, and T. Then, nodes compare own MAC_i with MAC_i, sent by the server.

$$MAC_i \stackrel{?}{=} HMAC_{R_i \oplus K_i}(C_i||I_s||T). \tag{4.2}$$

If they are not equal, the authentication of the server fails. Each node also checks the timestamp for freshness.

Step 3: After validation of the message by nodes, a nonce value (N_i) is chosen by each node. To provide message integrity and authenticity, nodes calculate MAC_i using N_i, ID_i, T and its shared secret key K_i. N_i, MAC_i, and T are encrypted with $R_i \oplus K_i$ and the encrypted message (c_i) is sent to the server by adding ID_i as:

$$\begin{aligned} MAC_i &= HMAC_{K_i}(N_i|ID_i||T), \\ c_i &= E_{(R_i \oplus K_i)}(N_i||MAC_i||T)||ID_i. \end{aligned} \tag{4.3}$$

Since only the server has R_i and K_i apart from the node, there is no chance to decrypt the message for an eavesdropper.

Step 4: The server collects group membership requests and verifies the nodes. If decryption of the message, verification of the MAC_i or checking the timestamp fails, nodes that are not verified are not included in the group communication. Then, the server determines the size of the group, where in the rest of the chapter, the group is assumed to have n members. By using this information, the server specifies the level of leaf nodes where group members are located, and it generates prime numbers for adding/evicting operations to reduce communication overhead on the network that we explain in Sections 4.2.3, 4.2.4, and 4.2.5. Each internal node in factorial tree contains one prime number. Next, it generates a random value as a group ID (G_{ID}). It is used as a secret key for a hash-based message authentication code that provides the authenticity and integrity of the group message when group members communicate with each other.

Step 5: The server chooses two CRPs for each node, where one of them ($C_{i,1}$ and $R_{i,1}$) is used for generating the group key, and the other ($C_{i,2}$ and $R_{i,2}$) is used for authentication. While constructing the group key, the server uses N_s, nonce of the server, and N_i values, sent by each node of the group. It computes a secret value for each node using the member's secret keys (K_i) and N_i values and encrypts them with the response ($R_{i,1}$) that corresponds to the challenge ($C_{i,1}$) as follows:

$$\begin{aligned} A_0 &= E_{K_s}(N_s||K_s), \\ A_1 &= E_{R_{1,1}}(N_1||K_1), \\ A_2 &= E_{R_{2,1}}(N_2||K_2), \\ &\vdots \\ A_n &= E_{R_{n,1}}(N_n||K_n). \end{aligned} \tag{4.4}$$

Step 6: By XORing the secret values of each group member, the group key G_K is generated as:

$$G_K = A_0 \oplus A_1 \oplus A_2 \oplus \cdots \oplus A_{n-1} \oplus A_n. \tag{4.5}$$

Step 7: The server prepares a specific message (M_i) for each node. Since the message is encrypted by the node's response and secret key, only the eligible nodes can decrypt

the message. The message contains a temporary group ID (T_{GID_i}), a temporary group key (T_{GK_i}), a set of prime numbers (P_i), a timestamp (T), a message authentication code (MAC_i) and two challenges ($C_{i,1}$ and $C_{i,2}$). While sending information to the nodes by the server, the group key and group ID are not sent to the nodes. Instead of sending them, T_{GK_i} and T_{GID_i} are sent by subtracting $HMAC_{K_i}(R_i)$ and A_i from G_{ID} and G_K, respectively as:

$$\begin{aligned} T_{GK_i} &= G_K \oplus A_i, \\ T_{GID_i} &= G_{ID} \oplus HMAC_{K_i}(R_i). \end{aligned} \tag{4.6}$$

Moreover, each member of the group only obtains P_i values on the key path, which denotes a path for group members located from the leaf node to the root node.

Step 8: The server computes hash-based message authenticaton code (MAC_i) for each node by using $T_{GK_i}, N_i + 1, T_{GID_i}, P_i, C_{i,1}, T$, and $R_{i,1}$ is used as an encryption key. The server encrypts node specific messages with XOR of the node's response ($R_{i,2}$) and K_i as:

$$\begin{aligned} MAC_i &= HMAC_{R_{i,1}}\left(T_{GK_i}||N_i+1||T_{GID_i}||T||P_i||C_{i,1}\right), \\ M_i &= \left(T_{GK_i}||MAC_i||T_{GID_i}||T||P_i||C_{i,1}\right), \\ c_i &= E_{(R_{i,2}\oplus K_i)}\left(M_i\right)||C_{i,2}. \end{aligned} \tag{4.7}$$

The server sends this information (c_i) to the nodes. In this message, N_i is not included since the node has already had this value.

Step 9: When the node receives c_i, it computes response $R_{i,2}$ corresponding to $C_{i,2}$ using PUF. Then, it decrypts message using $R_{i,2}$ and K_i:

$$\begin{aligned} M_i &= D_{(R_{i,2}\oplus K_i)}\left(c_i\right), \\ M_i &= \left(T_{GK_i}||MAC_i||T_{GID_i}||T||P_i||C_{i,1}\right). \end{aligned} \tag{4.8}$$

Step 10: First, the node computes $R_{i,1}$ using challenge $C_{i,1}$. Then, the node validates the integrity of MAC_i and checks the timestamp. If one of them is not validated, authentication and key distribution fails. If they are validated, it computes A_i value using with $R_{i,1}$, N_i and K_i as

$$A_i = E_{R_{i,1}}(N_i||K_i). \tag{4.9}$$

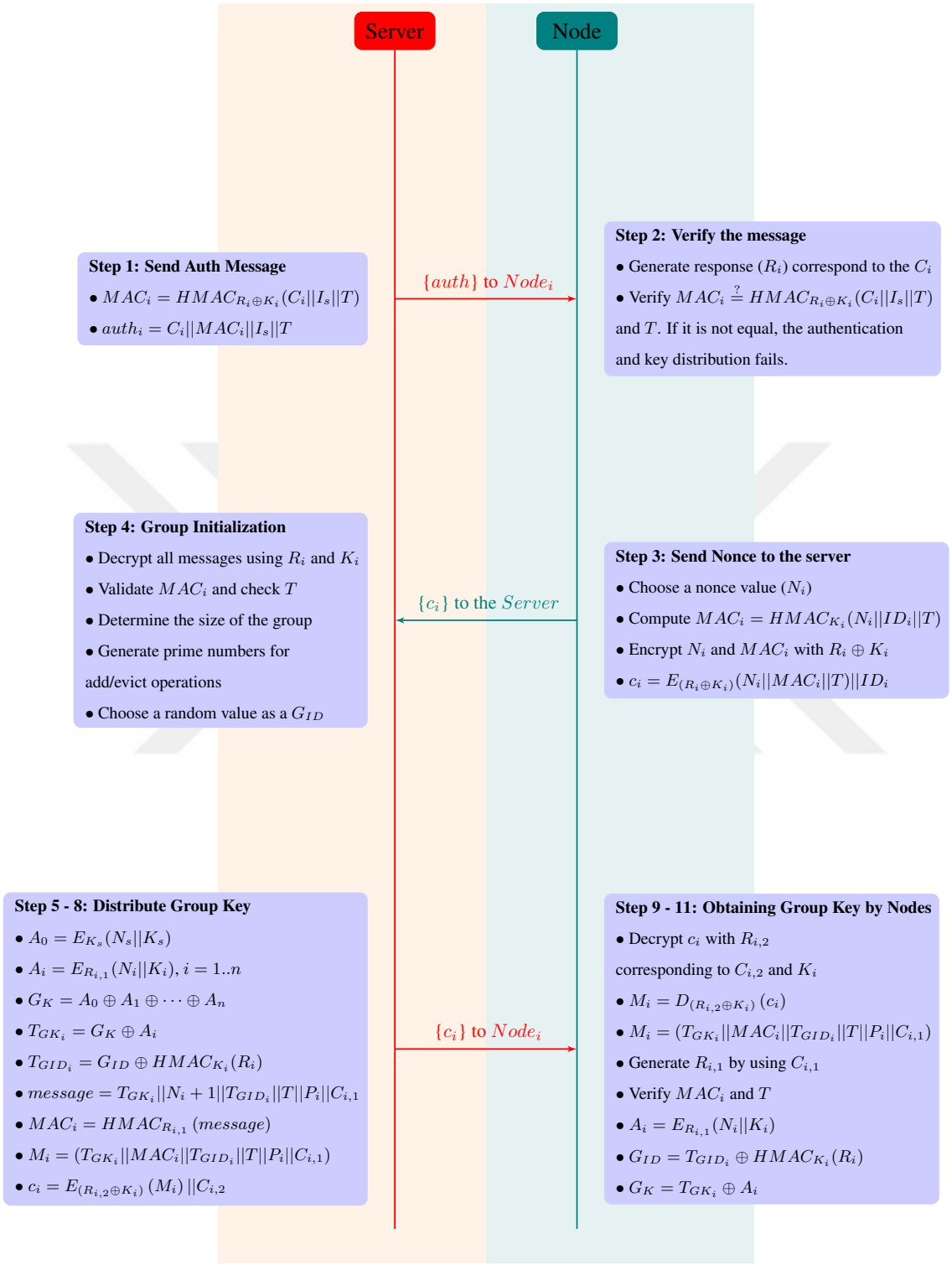

Figure 4.2: The authentication and key distribution phase of the PLGAKD protocol.

Step 11: The node obtains group key and group ID as follows:

$$\begin{aligned} G_K &= T_{GK_i} \oplus A_i, \\ G_{ID} &= T_{GID_i} \oplus HMAC_{K_i}(R_i), \end{aligned} \tag{4.10}$$

respectively. In the PLGAKD protocol summarized in Figure 4.2, the authentication is provided by using CRPs. Nodes are distinguishable from each other since PUF is unclonable, and each node produces different CRPs. However, there is a possibility that two different PUF can have the same CRP. Therefore, K_i is integrated into the protocol to eliminate this possibility. Moreover, our algorithm does not send the exact group key while sending messages. Only eligible nodes can generate group key and group ID. Without knowing the nodes' secret key and response, there is no chance to generate group key and group ID. After we present the group authentication and key distribution, we discuss how to add a new node, and remove an existing node from the group in the sequel.

4.2.3 Adding a Node to the Group

If a node is to be added to the group, the group key must be updated in order to provide backward secrecy, which ensures that prior messages cannot be decrypted by the new node. Since the new member of the group does not know the current group key, the new group key is sent to the existing group members by using the current group key. Therefore, adding a node to the group usually has a minimum cost that a server only sends a new group key to the nodes with a broadcast message. Also, a new member of the group is initialized with a new group key. Therefore, prior messages in the system cannot be decrypted by the new node, which satisfies backward secrecy.

Some of tree-based structures updates internal nodes on the key path when a new member of the group is added while renewing the group key. The rest of the internal nodes of the tree remains the same. In a binary tree, this operation has $O(log(n))$ communication complexity. However, it has $O(t^2)$ complexity in factorial tree without using CRT if internal nodes are updated where t is the level of the tree. In PLGAKD protocol, there is no need to update internal nodes for adding operation if the newly added node is at the same level as the group members. The new group key is composed of the existing group key and a random number, which is sent by the

server. Moreover, the new member of the group is initialized with the new group key, which provides backward secrecy for former messages. In addition, the group ID also is updated. Therefore, the operation of adding a new node to the group has $O(1)$ complexity. The new group key ($G_{K_{New}}$) and group ID ($G_{ID_{new}}$) are computed by using:

$$\begin{aligned} G_{K_{New}} &= Hash(G_K \,||\, N), \\ G_{ID_{new}} &= Hash(G_{ID} \,||\, N), \end{aligned} \tag{4.11}$$

where N is the nonce value sent to nodes by the server, which is encrypted with the current group key. All group members use a one-way function to produce a new group key and a new group ID except for a new member.

In the case of adding a node to the complete factorial tree, the newly added node and the first node of the tree are located at the next level as illustrated in Figure 4.3. Therefore, the required key materials are shared with the first node and the newly added node. Suppose that V_7, a new member of the group, is added to the complete factorial tree, as shown in Figure 4.3. The server chooses a prime number (P_3) that satisfies $gcd(P_3, V_i) = 1$, where $i = 2, 3$ and send P_3 to the V_1 and V_7 in addition to the G_K and P_1.

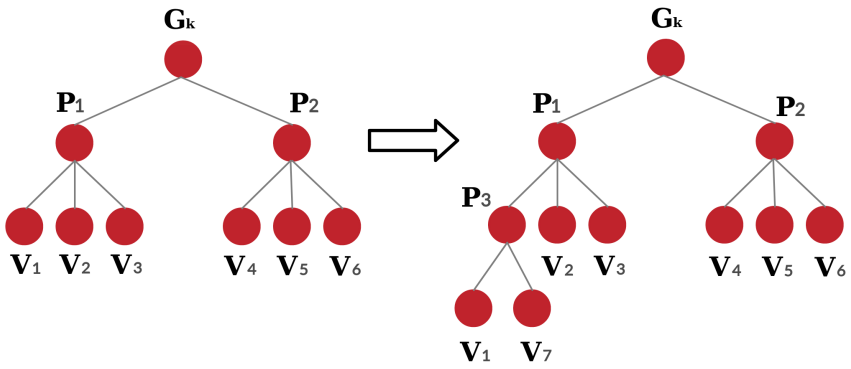

Figure 4.3: Adding a node to the complete Factorial Tree.

4.2.4 Evicting a Node from the Group

When a node is evicted from the group, the group key must be updated as soon as possible for forward secrecy since the evicted node can decrypt all messages in a

group with the current group key. However, the group key update is different from the adding operation. Since distributing the new group key cannot be performed with the current group key, it is sent to the group members by using internal keys, which are P_i values sent to the nodes in the initialization phase. Moreover, some of the internal keys must be updated, which are held by the evicted node.

The cost of the removal process of any node in the group is $O(log(n))$ in a binary tree. Suppose that V_8 is removed from the group shown in Figure 4.4a. The group key and internal keys, P_2 and P_6, must be changed. Nodes, located on the leftside of the tree, receive a new group key with P_1. V_5 and V_6 receive the new group key with P_5. Finally, V_7 is notified with unicast communication.

The cost of the removal process in the factorial tree depends on the level of the leaf node. It requires $t(t+1)/2$ messages to guarantee forward secrecy where t is the level of the tree. Since the number of leaf nodes at the same level is greater than the binary tree, the renewal of keys requires more network messages. Figure 4.4b shows that P_8, P_2 and G_K must be changed when node V_{24} is removed from the group. In this case, P_1, P_6 and P_7 is used to send new group key to the nodes between V_1 and V_{20}. Moreover, V_{21}, V_{22} and V_{23} are notified one by one for the new group key. Table 4.2 shows how many messages are required to distribute a new secret to construct a new group key in the factorial and binary tree.

By using CRT, the number of messages to renew a group key and internal keys can be reduced as we explain in the next section. On the side of the removed node, internal keys at the same level are used to send a new group key by using CRT except internal keys that are held by the removed node. Suppose that V_{24} is evicted from the group represented in Figure 4.4b. The leftside of the tree, from V_1 to V_{12}, receives group key update information with P_1. The rightside of the tree is informed with the internal keys - P_6 and P_7 - and nodes - V_{21}, V_{22} and V_{23}. In order to reduce network messages, P_6 and P_7 is combined as one message using CRT. Also, the same operation is applied for V_{21}, V_{22} and V_{23}. By using CRT, key renewal messages are completed in three messages instead of six for 24 nodes. Table 4.2 shows how many messages are required to distribute the new secret to construct a new group key in the factorial and binary tree.

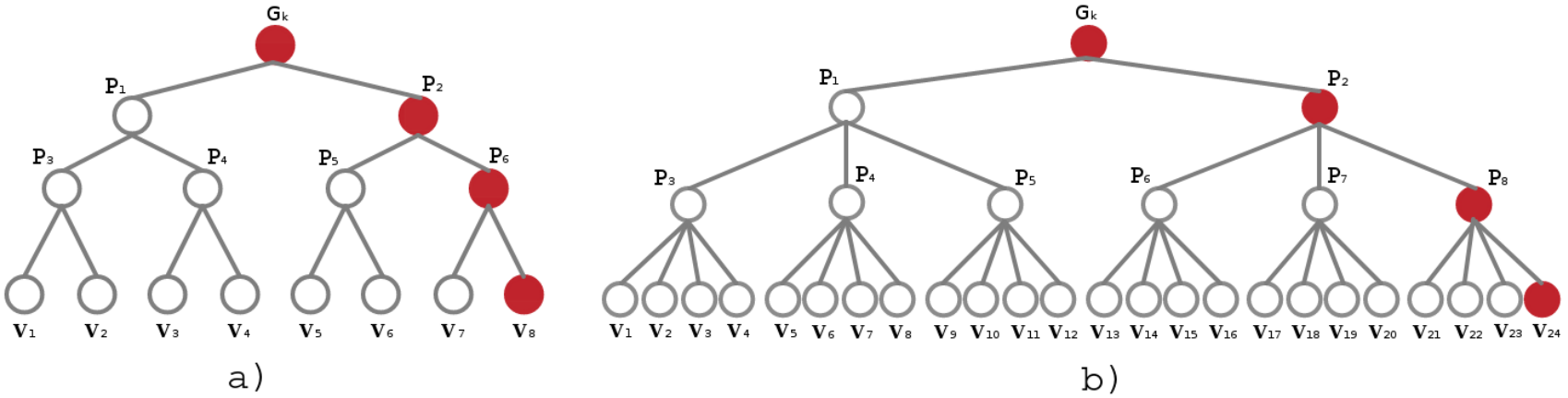

Figure 4.4: The representation of evicting operations: a) Evicting node from Binary Tree b) Evicting node from Factorial Tree (leaf nodes are group members).

Table 4.2: Required number of messages to renew the group key when the node is evicted in the factorial and binary tree with CRT and without CRT.

Factorial Tree				Binary Tree		
#Nodes	**Stored Keys**	**Key Renewal Messages Without CRT**	**Key Renewal Messages With CRT**	**#Nodes**	**Stored Keys**	**Key Renewal Messages**
120	5	10	4	128	7	7
740	6	15	5	1024	10	10
5040	7	21	6	4096	12	12
40320	8	28	7	32768	15	15
362880	9	36	8	262144	18	18

4.2.5 Key Distribution with CRT

When the server uses CRT to update the group key and internal keys, it must guarantee that each member of the group obtains the same secret at the end of the communication. Since all of the remainders should not be the same for CRT, a different strategy must be used. To solve this issue, this protocol uses a mathematical model.

Suppose that the group key renewal process for evicting node is performed with k prime numbers (internal keys) as explained in Section 2.8, where $k > 1$. Let P_1, P_2, ..., P_k be pairwise coprime integers (i.e., $gcd(P_i, P_j) = 1$ for $i \neq j$) greater than 1. The prime numbers that belong to the evicted node are not included in this process. The server chooses a random positive integer Y used for a new group key generation that satisfies $Y < min(P_i)$, where $i = 1, 2, \ldots, k$. Moreover, Y is used for updating the internal keys. Y also satisfy the following condition:

Suppose that the evicted node has t internal keys where $t > 0$. Let $H_1, \ldots, H_t$ be the internal keys of the evicted node. Note that, t shows the level of the tree. Let M_t be the set of horizantal internal keys except H_t. Under these assumptions

$$gcd(Y + H_i, M_i) = 1,$$

must be satisfied for $i = 1, 2, \ldots, t$. Since the evicted node does not access the Y value, the evicted node cannot obtain new internal keys. For the CRT computation, remainders are used to construct a unique solution using coprime integers. Therefore,

the server computes remainders substracting P_i values from Y as:

$$\begin{aligned} r_1 &= P_1 - Y, \\ r_2 &= P_2 - Y, \\ &\vdots \\ r_k &= P_k - Y. \end{aligned} \tag{4.12}$$

Since each member of the group has P_i values on its key path, Y can be recalculated by using remainders and P_i values. Then, the unique solution X is computed by using CRT as:

$$\begin{aligned} X &\equiv r_1 \, mod \, (P_1), \\ X &\equiv r_2 \, mod \, (P_2), \\ &\vdots \\ X &\equiv r_k \, mod \, (P_k). \end{aligned} \tag{4.13}$$

The server broadcasts the message X encrypting with the current group key to the members that hold P_i values. Instead of a broadcast to the entire network, only group members can process this message. All of the group members, including the evicted node, decrypt the message using the current group key. However, all of the group members except the evicted node can solve X and obtain the key material Y by performing mathematical computations as:

$$\begin{aligned} r_i &= X \ mod \ P_i, \\ Y &= P_i - r_i. \end{aligned} \tag{4.14}$$

The security of this process relies on the secrecy of the primes. The new group key can only be generated by nodes that have P_i keys. Suppose that, key length of the each P_i is 128 bits. If evicted members try to solve X without knowing the internal keys that are used in CRP, this process takes 2^{128} in the worst case. Moreover, the evicted member may try to solve X using numbers, which are coprime integers with evicted members' internal keys. In this case, the evicted member must perform computations: finding coprime numbers using the greatest common divisor algorithm, solving X with that number, obtaining Y, computing G_K with a hash function, and decryption

of the message. However, this cannot be performed for all coprime integers in a reasonable time. Therefore, the key renewal process provides secure communication in the PLGAKD protocol.

After obtaining Y value, the new group key ($G_{K_{New}}$) is computed by:

$$G_{K_{New}} = Hash(G_K \,||\, Y). \tag{4.15}$$

Moreover, the internal keys of the evicted node, and G_{ID} are updated by:

$$\begin{aligned} H_{i_{new}} &= Y + H_i, \\ G_{ID_{new}} &= Hash(G_{ID} \,||\, Y), \end{aligned} \tag{4.16}$$

where $H_i \in \{H_1, \ldots, H_t\}$ and $H_{i_{new}}$ is the updated internal key.

In PLGAKD protocol, there is no difference between bulk adding/evicting operations with single ones. All operations are performed one by one for each node.

CHAPTER 5

ANALYSIS OF THE PLGAKD PROTOCOL

In this chapter, the PLGAKD protocol is analyzed in terms of security threats and computation, communication and storage overhead. Moreover, the PLGAKD protocol is compared with the existing methods in terms of adding/evicting costs.

5.1 Threat Analysis

Since PUF is unclonable, tamper-evident, and generates different CRPs for each operation, it is resistant to cloning attack, eavesdropping attack, impersonation attack, replay attack, and man in the middle attack (MITM) as illustrated in Table 5.1. Since PUF is unclonable, the PLGAKD protocol is resistant to cloning attacks. Eavesdropping attack, unauthorized party steals, modifies or removes required information while two parties are communicating, cannot be performed since all group members are authenticated, and each communication between the server and nodes are encrypted with different encryption keys. CRP guarantees that only the server and node can securely communicate with each other since each PUF has different characteristics. Since each PUF is unique and all required information is stored in the initialization phase by the server, eavesdroppers cannot impersonate without knowing CRP. Threat analysis of the PLGAKD protocol can be examined under two parts: group authentication and group communication. For the authentication part, the use of different CRPs with the secret key requires different encryption/decryption keys for each message, and these keys are generated on the fly by each node using PUF, which provides confidentiality. Moreover, the integrity of the messages is guaranteed by using MAC. The PLGAKD protocol is resistant to MITM attack where eavesdroppers

relay or alter data between the server and the node. Since the proposed protocol uses a CRP and a secret key, eavesdroppers must send valid CRP. However, CRP is shared with only the server. Without knowing a CRP and a secret key, this attack cannot be performed. In addition, eavesdroppers can try to perform a replay attack in which data is intercepted and fraudulently repeated or delayed. By using a replay attack, eavesdroppers try to gain access. Since the PLGAKD protocol uses timestamp and MAC, they cannot perform a replay attack in this protocol. Moreover, the server does not send the actual group key and ID, in which only eligible nodes can generate them. It is not possible to obtain the group key and group ID via eavesdropping because of the uniqueness of PUF. For the group communication part, to share information between group members, group key and group ID are used to encrypt and verify the messages, respectively. This process provides confidentiality and integrity among group members' communication. Since this protocol renews the group key and group ID periodically, it prevents machine learning attacks. For group communication, one of the most crucial parts is to provide backward and forward secrecy. The PLGAKD protocol provides efficient backward and forward secrecy without compromising any details with minimum network messages since all operations are performed by using internal keys and CRT. Without knowing the group key and internal keys, eavesdroppers cannot obtain the new group key.

Table 5.1: The resistance of the PLGAKD protocol against security attacks.

Attack Type	**Protection**
Impersonation	Uniquness of PUF
MITM	A secret key and different CRPs
Replay	MAC and Timestamp
Cloning	Uniquness of PUF
Eavesdropping	Authentication, different encryption keys for each communication

5.2 Communication Analysis

The PLGAKD protocol performs one-to-one communication with each node in the authentication and key distribution phase to authenticate the nodes, determine the group members, and distribute the key materials. Therefore, there is no enhancement in terms of the number of messages in this phase. The PLGAKD protocol offers lightweight communication overhead thanks to a factorial tree and CRT in terms of communication messages and storage overhead rather than the message size and bandwidth for the key renewal phase. The combination of the factorial tree and CRT visibly decreases the number of total communication messages contrary to the required number of messages for a binary tree shown in Figure 5.1. While the total communication messages are equal for eight nodes, this protocol reduces total messages to half compared to the binary tree for thousands of nodes. Even if there are hundreds of thousands of nodes in the group, the PLGAKD protocol performs key renewal operations efficiently since the communication overhead of group operations is slower than logarithmic growth. Therefore, the proposed protocol also provides scalability. However, a higher number of nodes only increase the message size since the PLGAKD protocol uses CRT to reduce message count during the key renewal process. The size of the messages is directly related to the levels of the factorial tree. Since the number of combined messages increases from top to bottom of the factorial tree, the size of the message is calculated as the multiplication of the size of the internal key size and the number of nodes, at most. Although increasing message size affects bandwidth, the bandwidth problem can be solved with new technologies and network approaches such as 5G, decentralized systems and so on. In addition to reducing the communication messages, the PLGAKD protocol reduces storage overhead in the nodes for the same number of devices presented in Figure 5.2 since the factorial tree accommodates more nodes than a binary tree beginning from level 2.

Table 5.2 shows the cost of the group key distribution for different methods. The significance of this table is the evicting operations that the other methods perform $log(n)$ or n operations for evicting the node from the group except [36]. The new group key is distributed with one broadcast message by using CRT in [36]. However, it is not efficient when the group consists of thousands of devices. The PLGAKD

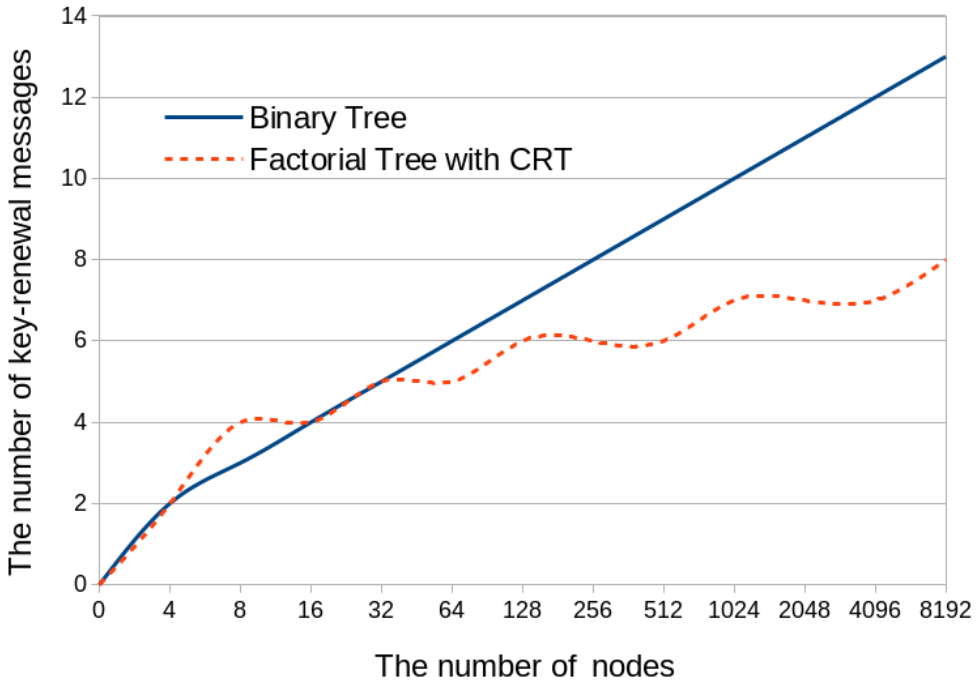

Figure 5.1: A comparison of key renewal process for a binary and factorial tree combined with CRT.

protocol achieves group key distribution with t messages. Moreover, the number of nodes while performing CRT is not more than t. Although [36] achieves the key distribution with one CRT, the PLGAKD protocol reduces the computation overhead by dividing nodes into small subgroups. For example, for the leaf node located in Level 5, the total number of different CRT computations is four. Moreover, these computations are performed with four prime numbers at most.

Table 5.2: The comparison of communication costs for different methods (*: worst case).

Methods	**Type of System**	**Cost of Adding**	**Cost of Evicting**
[77]	Asymmetric	$log(n)$	$log(n)$
[61]	Symmetric	1	$log(n)$
[35]	Hash	1	$n*$
[36]	Asymmetric	1	1
PLGAKD	Symmetric	1	t

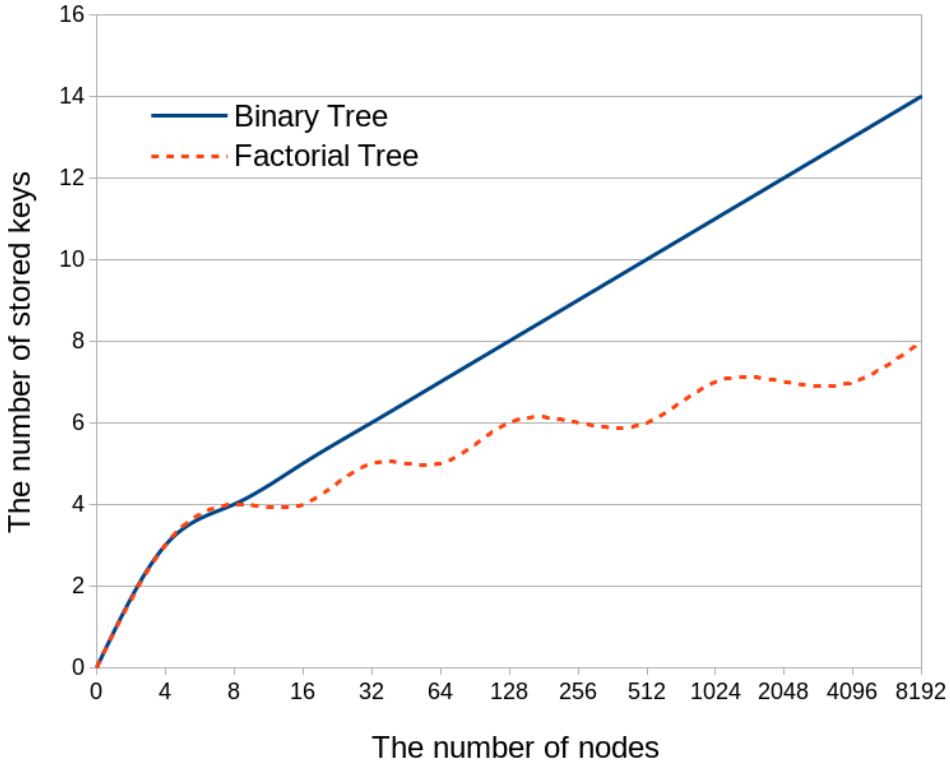

Figure 5.2: A comparison of node stored keys for a binary and factorial tree.

5.3 The Cost of the PLGAKD Protocol

Based on the assumption that there is one server and many nodes, the required operations on the server and nodes can be examined under two parts:

5.3.1 Analysis of the Initialization Phase

The time to generate or store a response can vary since PUF type, area, process technology, error correction code (ECC) affect the generation of the response, as shown in Table 5.3. The PLGAKD protocol uses strong PUF, which provides a huge number of CRPs. In order to obtain more accurate responses, ECC algorithms are preferred that these algorithms can fix incorrect responses arising from environmental factors or eavesdroppers. However, ECC increases the storage overhead and generation time of CRPs. According to [46], the storage overhead of a CRP is about 2000 bits with ECC. By using this information, the storage overhead of nodes for different number

of CRPs is shown in Figure 5.3 in which 50000 CRPs and 25000 nodes are stored in $312.5\ GB$. Moreover, Figure 5.4 represents the storage overhead of the different number of nodes without applying ECC. In this figure, the total size of the CRP is 256 bit.

Table 5.3: The time to generate a response with different PUF types. (GE: Gate Equivalent, $Tech.$: Technology)

PUF Type	**Tech.**	**AREA (GE)**	**CRP Space bits**	**Response Bits**	**ECC**	**Time**
Lattice [96]	$45\ nm$	-	2^{136}	100	Yes	$4.4\ ms$
SA [19]	$65\ nm$	3963	2^{128}	128	No	$40\ ns$
RO [72]	$45\ nm$	-	2^{128}	128	Yes	$5.62\ ms$
Latch-based [100]	$90\ nm$	$5.4K$	$2^{192.7}$	256	Yes	$0.4\ ms$
RO [46]	$28\ nm$	2210	-	128	Yes	$6.72\ ms$

The total time required to store CRPs with a different number of nodes and different number of CRPs is shown in Table 5.4. This table is created by selecting two different response times: $0.4\ ms$ and $6.72\ ms$ which are the lowest and highest values to generate a response using ECC. Without using the ECC algorithm, the generation of the response takes $40ns$ in [19]. According to this, 50000 nodes and 50000 CRPs are completed in less than a minute. As a result, PUF types and their features play an important role in the initialization phase in terms of storage and time.

5.3.2 Analysis of the Group Authentication and Key Distribution Phase

During the group authentication and key distribution, the following operations are performed on the server and nodes. Suppose that the group consists of n nodes. The server performs $(4n)\ HMAC$ operations, $(2n+1)$ encryptions, (n) decryptions, and $(5n)\ XOR$ operations. A node performs 3 responses generations, 2 encryptions, 1 decryption, $5\ XOR$s operations and $4\ HMAC$ operations. When the node is evicted from the group, servers perform the key renewal process with CRT where the total

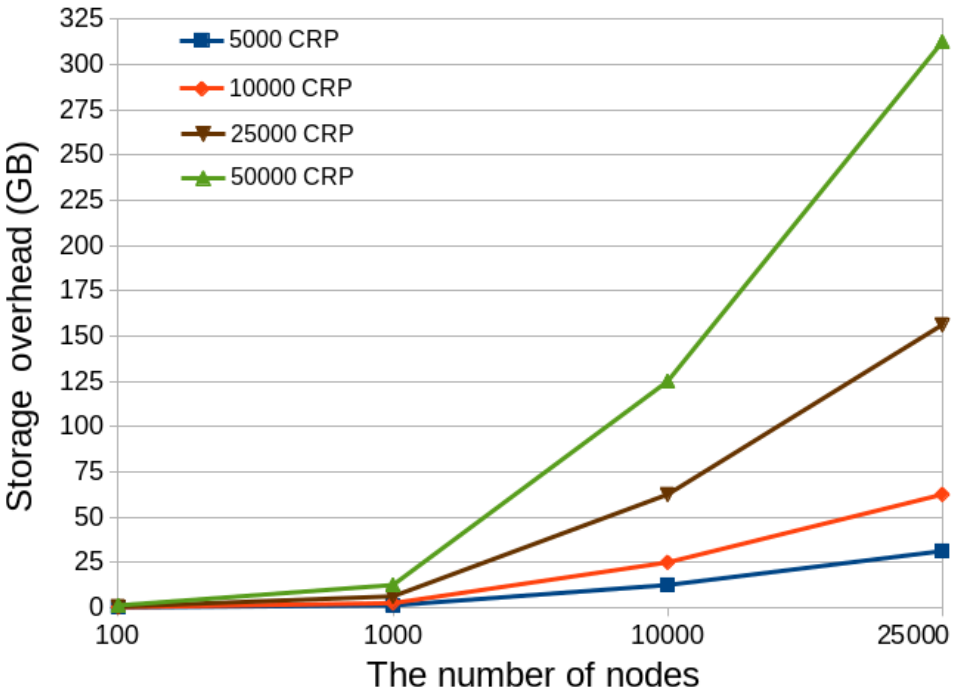

Figure 5.3: The storage overhead of nodes with different number of CRPs (ECC included).

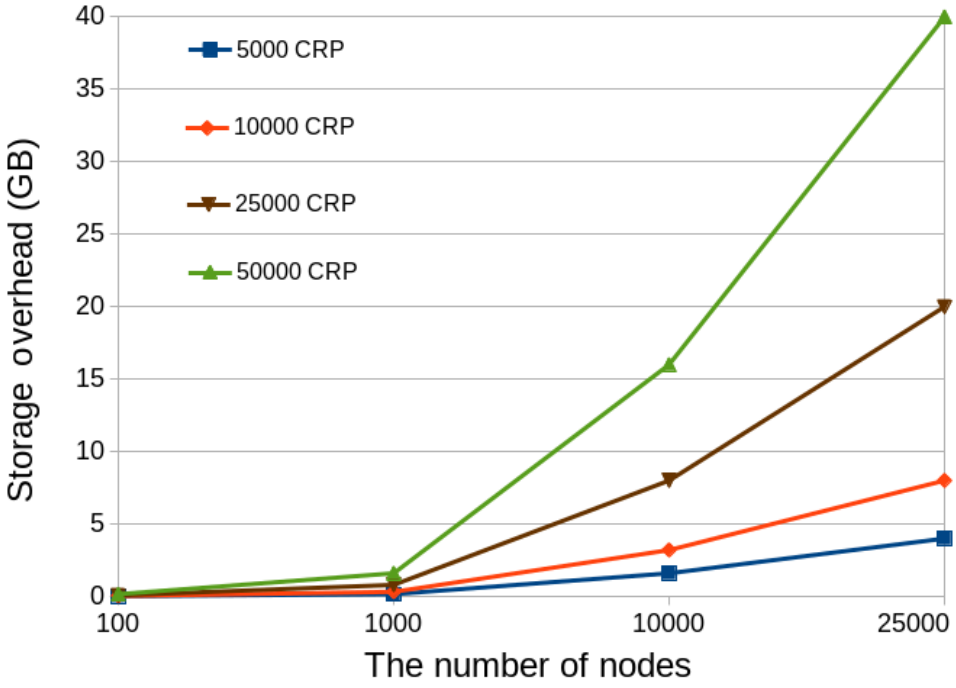

Figure 5.4: The storage overhead of nodes with different number of CRPs (without ECC).

Table 5.4: The time (in hours) required to store CRPs to the server with a different number of nodes.

The response time: 0.4ms				
# of stored CRPs / # of nodes	**1000**	**10000**	**25000**	**50000**
100	0.01	0.11	0.27	0.55
1000	0.11	1.11	2.77	5.55
5000	0.55	5.55	13.88	27.77
10000	1.11	11.11	27.77	55.55
50000	5.55	55.55	138.88	277.77
The response time: 6.27ms				
# of stored CRPs / # of nodes	**1000**	**10000**	**25000**	**50000**
100	0.19	1.87	4.67	9.33
1000	1.87	18.67	46.67	93.33
5000	9.33	93.33	233.33	466.67
10000	18.67	186.67	466.67	933.33
50000	93.33	933.33	2333.33	4666.67

number of CRT computations is determined according to the level of the leaf nodes.

Since these operations are simple computations, the PLGAKD protocol meets these requirements for IoT devices by using PUF, symmetric encryption, hash function, and XOR. Moreover, the key renewal process provides lightweight computations. The IoT devices perform only one modular arithmetic operation and one hash function to generate the new group key.

CHAPTER 6

CONCLUSION

In this chapter, we summarize the conclusions of the thesis and highlight future work.

6.1 Conclusion

IoT and machine type communication will be an integral part of the Digital Age. IoT devices collect and transmit data over global network in healtcare, agriculture, tranportation, smart city and smart home. However, these devices lack of security primitives. With the inclusion of insecure IoT devices to the global network, these devices have become a target to launch cyber attacks. The data collected by IoT devices is another critical target since the data can be used to perform attacks that threaten social or personal life. Besides, the number of connected devices in the network is increasing day by day, which can cause communication delay, network congestion, and message loss. Therefore, IoT devices must be equipped with lightweight cryptographic algorithms and ligtweight protocols. In this thesis, a PLGAKD protocol is proposed that eliminates these threats and network problems using lightweight secure group authentication and key distribution. The proposed protocol achieves group authentication and key distribution using PUF. Moreover, the key renewal process for backward and forward secrecy is achieved by using a Factorial Tree and the CRT. By using a Factorial Tree and the CRT, the proposed protocol not only reduces communication overhead but also storage overhead. For thousands of nodes, the PLGAKD protocol almost halves the overhead for backward and forward secrecy compared to the binary tree solutions.

6.2 Future Work

The following future extensions can be applied to this work:

- The PLGAKD protocol can be performed with sensors, actuators, desktops, tablets, and servers to evaluate computation and communication overhead in real-time by using different lightweight symmetric ciphers and lightweight hash algorithms.
- Since PUF has different types, PUF performance analysis can be performed on the PLGAKD protocol.

REFERENCES

[1] Asymmetric Techniques, ISO/IEC. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56427. Accessed: 2020-08-16.

[2] Block Ciphers, ISO/IEC. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56552. Accessed: 2020-08-16.

[3] Data Distribution Service (DDS), howpublished = http://www.omg.org/spec/dds/1.4/pdf/, note = Accessed: 2020-08-16.

[4] Hash-Functions, ISO/IEC. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=67173. Accessed: 2020-08-16.

[5] Implementations of lightweight block ciphers on a WSN430 sensor. http://bloc.project.citi-lab.fr/library.html. Accessed: 2020-08-16.

[6] Information technology – Security techniques – Lightweight cryptography – Part 1: General, howpublished = http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=56425, note = Accessed: 2020-08-16.

[7] Lightweight Cryptography Lounge. http://cryptolux.org/index.php/Lightweight_Cryptography. Accessed: 2020-08-16.

[8] Message Queuing Telemetry Transport (MQTT), howpublished = https://www.iso.org/standard/69466.html, note = Accessed: 2020-08-16.

[9] Report on Lightweight Cryptography, NIST. http://csrc.nist.gov/publications/drafts/nistir-8114/nistir_8114_draft.pdf. Accessed: 2020-08-16.

[10] Stream Ciphers, ISO/IEC. http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56426. Accessed: 2020-08-16.

[11] The ECRYPT Stream Cipher Project. http://www.ecrypt.eu.org/stream/. Accessed: 2020-08-16.

[12] Total Energy Consumption by U.S Energy Information Administration Energy Consumption. https://www.eia.gov/totalenergy/data/monthly/pdf/sec2_2.pdf. Accessed: 2020-08-16.

[13] M. Ågren, M. Hell, T. Johansson, and W. Meier. A new version of grain-128 with authentication. In *Symmetric Key Encryption Workshop*, volume 2011, 2011.

[14] J.-P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia. Quark: A lightweight hash. pages 1–15, 2010.

[15] Babbage, Steve, and Matthew Dodd. The stream cipher MICKEY 2.0. ECRYPT. http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf. Accessed: 2020-08-16.

[16] S. Badel, N. Dağtekin, J. Nakahara, K. Ouafi, N. Reffé, P. Sepehrdad, P. Sušil, and S. Vaudenay. Armadillo: A multi-purpose cryptographic primitive dedicated to hardware. In S. Mangard and F.-X. Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010*, pages 398–412, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[17] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers. The simon and speck families of lightweight block ciphers. Cryptology ePrint Archive, Report 2013/404, 2013. https://eprint.iacr.org/2013/404.

[18] G. Bertoni, J. Daemen, M. Peeters, and G. Assche. Sponge functions. *ECRYPT Hash Workshop 2007*, 01 2007.

[19] M. Bhargava and K. Mai. An efficient reliable puf-based cryptographic key generator in 65nm cmos. In *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1–6. IEEE, 2014.

[20] A. Bogdanov, M. Knezevic, G. Leander, D. Toz, K. Varici, and I. Verbauwhede. Spongent: The design space of lightweight cryptographic hashing. *IEEE Transactions on Computers*, 62(10):2041–2053, 2013.

[21] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe. Present: An ultra-lightweight block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 450–466. Springer, 2007.

[22] C. Bormann, M. Ersue, and A. Keranen. Terminology for constrained-node networks. *Internet Engineering Task Force (IETF): Fremont, CA, USA*, pages 2070–1721, 2014.

[23] M. Cazorla, K. Marquet, and M. Minier. Survey and benchmark of lightweight block ciphers for wireless sensor networks. In *Security and Cryptography (SECRYPT), 2013 International Conference on*, pages 1–6. IEEE, 2013.

[24] O. Cheikhrouhou. Secure Group Communication in Wireless Sensor Networks: A survey. *Journal of Network and Computer Applications*, 61:115–132, 2016.

[25] B. Chen, T. Ignatenko, F. M. Willems, R. Maes, E. van der Sluis, and G. Selimis. A Robust SRAM-PUF Key Generation Scheme Based on Polar Codes . In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[26] H. Cheng, H. M. Heys, and C. Wang. PUFFIN: A Novel Compact Block Cipher Targeted to Embedded Digital Systems. In *2008 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools*, pages 383–390, 2008.

[27] J. Daemen and C. Clapp. Fast hashing and stream encryption with panama. In S. Vaudenay, editor, *Fast Software Encryption*, pages 60–74, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[28] Y. S. Dan Demeter, Marco Preuss. IoT: a malware story. `https://securelist.com/iot-a-malware-story/94451/`, 2019. Accessed: 2020-08-16.

[29] C. De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *International Conference on Information Security*, pages 171–186. Springer, 2006.

[30] C. De Canniere, O. Dunkelman, and M. Knežević. Katan and ktantan—a family of small and efficient hardware-oriented block ciphers. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 272–288. Springer, 2009.

[31] J. Delvaux, R. Peeters, D. Gu, and I. Verbauwhede. A Survey on Lightweight Entity Authentication with Strong PUFs. *ACM Computing Surveys (CSUR)*, 48(2):1–42, 2015.

[32] S. R. Department. Internet of Things - number of connected devices worldwide 2015-2025. `https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/`, 2019. Accessed: 2020-08-16.

[33] Deuk-Whee Kwak, Seung Joo Lee, Jong Won Kim, and E. Jung. An Efficient LKH Tree Balancing Algorithm for Group Key Management. *IEEE Communications Letters*, 10(3):222–224, March 2006.

[34] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In C. Cachin and J. L. Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, pages 523–540, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[35] P. Dong, W. Wang, X. Shi, and T. Qin. Lightweight key management for group communication in body area networks through physical unclonable functions. In *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*, pages 102–107, 2017.

[36] M. H. Eldefrawy, N. Pereira, and M. Gidlund. Key Distribution Protocol for Industrial Internet of Things Without Implicit Certificates. *IEEE Internet of Things Journal*, 6(1):906–917, Feb 2019.

[37] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu. Authentica-

tion Protocols for Internet of Things: A Comprehensive Survey. *Security and Communication Networks*, 2017, 2017.

[38] A. Fiat and M. Naor. Broadcast Encryption. In *Annual International Cryptology Conference*, pages 480–491. Springer, 1993.

[39] C. Fontaine. Nonlinear feedback shift register. In *Encyclopedia of Cryptography and Security*, pages 415–416. Springer, 2005.

[40] K. B. Frikken, M. Blanton, and M. J. Atallah. Robust Authentication Using Physically Unclonable Functions. In *International Conference on Information Security*, pages 262–277. Springer, 2009.

[41] M. Ge, K.-K. R. Choo, H. Wu, and Y. Yu. Survey on key revocation mechanisms in wireless sensor networks. *Journal of Network and Computer Applications*, 63:24–38, 2016.

[42] T. Gebremichael, U. Jennehag, and M. Gidlund. Lightweight iot group key establishment scheme using one-way accumulator. In *2018 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–7, 2018.

[43] Z. Gong, S. Nikova, and Y. W. Law. Klein: A new family of lightweight block ciphers. In A. Juels and C. Paar, editors, *RFID. Security and Privacy*, pages 1–18, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[44] J. Guo, T. Peyrin, and A. Poschmann. The photon family of lightweight hash functions. In *Annual Cryptology Conference*, pages 222–239. Springer, 2011.

[45] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw. The led block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 326–341. Springer, 2011.

[46] I. Haider and B. Rinner. Securing cloud-based iot applications with trustworthy sensing. In A. Longo, M. Zappatore, M. Villari, O. Rana, D. Bruneo, R. Ranjan, M. Fazio, and P. Massonet, editors, *Cloud Infrastructures, Services, and IoT Systems for Smart Cities*, pages 218–227, Cham, 2018. Springer International Publishing.

[47] L. Harn. Group authentication. *IEEE Transactions on computers*, 62(9):1893–1898, 2013.

[48] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. medi, and K. Wehrle. Security challenges in the ip-based internet of things, 12 2011.

[49] M. Hell, T. Johansson, and W. Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, 2007.

[50] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas. Physical Unclonable Functions and Applications: A Tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014.

[51] H. M. Heys. A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3):189–221, 2002.

[52] S. Hirose, K. Ideguchi, H. Kuwakado, T. Owada, B. Preneel, and H. Yoshida. A lightweight 256-bit hash function for hardware and low-end devices: Lesamnta-lw. In K.-H. Rhee and D. Nyang, editors, *Information Security and Cryptology - ICISC 2010*, pages 151–168, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[53] V. T. Hoang and P. Rogaway. On generalized feistel networks. In *Annual Cryptology Conference*, pages 613–630. Springer, 2010.

[54] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. Hight: A new block cipher suitable for low-resource device. In L. Goubin and M. Matsui, editors, *Cryptographic Hardware and Embedded Systems - CHES 2006*, pages 46–59, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[55] M. Huang, B. Yu, and S. Li. PUF-Assisted Group Key Distribution Scheme for Software-Defined Wireless Sensor Networks. *IEEE Communications Letters*, 22(2):404–407, 2018.

[56] M. Izadi, B. Sadeghiyan, S. S. Sadeghian, and H. A. Khanooki. Mibs: A new lightweight block cipher. In J. A. Garay, A. Miyaji, and A. Otsuka, editors,

Cryptology and Network Security, pages 334–348, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[57] H. Kang, Y. Hori, T. Katashita, M. Hagiwara, and K. Iwamura. Performance Analysis for PUF Data Using Fuzzy Extractor. In Y.-S. Jeong, Y.-H. Park, C.-H. R. Hsu, and J. J. J. H. Park, editors, *Ubiquitous Information Technologies and Applications*, pages 277–284, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[58] B. P. kavin and S. Ganapathy. A secured storage and privacy-preserving model using CRT for providing security on cloud and IoT-based applications. *Computer Networks*, 151:181 – 190, 2019.

[59] U. Khalid, M. Asim, T. Baker, P. C. Hung, M. A. Tariq, and L. Rafferty. A decentralized lightweight blockchain-based authentication mechanism for iot systems. *Cluster Computing*, pages 1–21, 2020.

[60] L. Knudsen, G. Leander, A. Poschmann, and M. J. Robshaw. Printcipher: a block cipher for ic-printing. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–32. Springer, 2010.

[61] Y.-H. Kung and H.-C. Hsiao. GroupIt: Lightweight Group Key Management for Dynamic IoT Environments. *IEEE Internet of Things Journal*, 5(6):5155–5165, 2018.

[62] N. Kushalnagar, G. Montenegro, and C. Schumacher. Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals, 01 2007.

[63] C. Lai, H. Li, R. Lu, Rong Jiang, and X. Shen. Lgth: A lightweight group authentication protocol for machine-type communication in lte networks. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 832–837, 2013.

[64] C. Lai, H. Li, R. Lu, and X. S. Shen. Se-aka: A secure and efficient group authentication and key agreement protocol for lte networks. *Computer Networks*, 57(17):3492–3510, 2013.

[65] C. Lai, R. Lu, D. Zheng, H. Li, and X. S. Shen. GLARM: Group-based lightweight authentication scheme for resource-constrained machine to machine communications. *Computer Networks*, 99:66–81, 2016.

[66] G. Leander, C. Paar, A. Poschmann, and K. Schramm. New Lightweight DES Variants. In A. Biryukov, editor, *Fast Software Encryption*, pages 196–210, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[67] C. H. Lim and T. Korkishko. mCrypton – A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In J.-S. Song, T. Kwon, and M. Yung, editors, *Information Security Applications*, pages 243–258, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[68] I.-C. Lin, S.-S. Tang, and C.-M. Wang. Multicast Key Management without Rekeying Processes. *The Computer Journal*, 53(7):939–950, 2010.

[69] Y. Liu, L. Harn, and C.-C. Chang. An authenticated group key distribution mechanism using theory of numbers. *International Journal of Communication Systems*, 27(11):3502–3512, 2014.

[70] J. Lu. Related-key rectangle attack on 36 rounds of the xtea block cipher. *International Journal of Information Security*, 8(1):1–11, 2009.

[71] M. Luk, A. Perrig, and B. Whillock. Seven Cardinal Properties of Sensor Network Broadcast Authentication. In *Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 147–156. ACM, 2006.

[72] R. Maes, A. Van Herrewege, and I. Verbauwhede. Pufky: A fully functional puf-based cryptographic key generator. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 302–319. Springer, 2012.

[73] P. N. Mahalle, N. R. Prasad, and R. Prasad. Threshold cryptography-based group authentication (tcga) scheme for the internet of things (iot). In *2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE)*, pages 1–5. IEEE, 2014.

[74] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas. Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by

Substring Matching. In *2012 IEEE Symposium on Security and Privacy Workshops*, pages 33–44. IEEE, 2012.

[75] R. C. Merkle. *Secrecy, authentication, and public key systems.* Stanford University, 1979.

[76] M. Murase. Linear feedback shift register. Google Patents, 1992. US Patent 5,090,035.

[77] B. L. Parne, S. Gupta, and N. S. Chaudhari. SEGB: Security Enhanced Group Based AKA Protocol for M2M Communication in an IoT Enabled LTE/LTE-A Network. *IEEE Access*, 6:3668–3684, 2018.

[78] C. Patrick and P. Schaumont. The role of energy in the lightweight cryptographic profile. In *NIST lightweight cryptography workshop*, 2016.

[79] P. Porambage, A. Braeken, C. Schmitt, A. Gurtov, M. Ylianttila, and B. Stiller. Group Key Establishment for Enabling Secure Multicast Communication in Wireless Sensor Networks Deployed for IoT Applications. *IEEE Access*, 3:1503–1511, 2015.

[80] A. Poschmann. Lightweight Cryptography-Cryptographic Engineering for a Pervasive World. Ph. D. Thesis, Ruhr University Bochum. 2009.

[81] M. Roel. Physically unclonable functions: Constructions, properties and applications. *Katholieke Universiteit Leuven, Belgium*, 2012.

[82] M. Saadeh, A. Sleit, M. Qatawneh, and W. Almobaideen. Authentication techniques for the internet of things: A survey. In *2016 Cybersecurity and Cyberforensics Conference (CCC)*, pages 28–34, 2016.

[83] P. Saint-Andre et al. Extensible messaging and presence protocol (xmpp): Core, 2004.

[84] Z. Shelby, K. Hartke, and C. Bormann. The constrained application protocol (CoAP), 2014.

[85] A. T. Sherman and D. A. McGrew. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. *IEEE transactions on Software Engineering*, 29(5):444–458, 2003.

[86] K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai. Piccolo: An ultra-lightweight blockcipher. In B. Preneel and T. Takagi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2011*, pages 342–357, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[87] T. Shirai and B. Preneel. On feistel ciphers using optimal diffusion mappings across multiple rounds. In P. J. Lee, editor, *Advances in Cryptology - ASIACRYPT 2004*, pages 1–15, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[88] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-bit blockcipher clefia (extended abstract). In A. Biryukov, editor, *Fast Software Encryption*, pages 181–195, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[89] V. L. Shivraj, M. A. Rajan, M. Singh, and P. Balamuralidhar. One Time Password Authentication Scheme basedon Elliptic Curves for Internet of Things (IoT). In *2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW)*, pages 1–6, Feb 2015.

[90] R. Song, L. Korba, and G. O. Yee. A Scalable Group Key Management Protocol. *IEEE Communications Letters*, 12(7):541–543, 2008.

[91] F.-X. Standaert, G. Piret, N. Gershenfeld, and J.-J. Quisquater. Sea: A scalable encryption algorithm for small embedded applications. In J. Domingo-Ferrer, J. Posegga, and D. Schreckling, editors, *Smart Card Research and Advanced Applications*, pages 222–236, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[92] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14, 2007.

[93] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi. TWINE: A Lightweight Block Cipher for Multiple Platforms. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography*, pages 339–354, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[94] M. Turkanović, B. Brumen, and M. Hölbl. A novel user authentication and key agreement scheme for heterogeneous ad hoc wireless sensor networks, based on the Internet of Things notion. *Ad Hoc Networks*, 20:96–112, 2014.

[95] C. Wachsmann and A.-R. Sadeghi. Physically unclonable functions (PUFs): Applications, models, and future directions. *Synthesis Lectures on Information Security, Privacy, & Trust*, 5(3):1–91, 2014.

[96] Y. Wang, X. Xi, and M. Orshansky. Lattice puf: A strong physical unclonable function provably secure against machine learning attacks, 2019.

[97] D. Watanabe, K. Ideguchi, J. Kitahara, K. Muto, H. Furuichi, and T. Kaneko. Enocoro-80: A hardware oriented stream cipher. In *2008 Third International Conference on Availability, Reliability and Security*, pages 1294–1300, 2008.

[98] C. K. Wong, M. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. *IEEE/ACM transactions on networking*, 8(1):16–30, 2000.

[99] W. Wu and L. Zhang. Lblock: a lightweight block cipher. In *International Conference on Applied Cryptography and Network Security*, pages 327–344. Springer, 2011.

[100] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, M. Takenaka, and K. Itoh. Variety enhancement of puf responses using the locations of random outputting rs latches. *Journal of Cryptographic Engineering*, 3(4):197–211, 2013.

[101] X. Yao, X. Han, X. Du, and X. Zhou. A Lightweight Multicast Authentication Mechanism for Small Scale IoT Applications. *IEEE Sensors Journal*, 13(10):3693–3701, 2013.

[102] H. Yap, K. Khoo, A. Poschmann, and M. Henricksen. Epcbc - a block cipher suitable for electronic product code encryption. In D. Lin, G. Tsudik, and X. Wang, editors, *Cryptology and Network Security*, pages 76–97, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[103] W. Zhang, Z. Bao, D. Lin, V. Rijmen, B. Yang, and I. Verbauwhede. Rectangle: a bit-slice lightweight block cipher suitable for multiple platforms. *Science China Information Sciences*, 58(12):1–15, 2015.

[104] X. Zheng, C.-T. Huang, and M. Matthews. Chinese remainder theorem based group key management. In *Proceedings of the 45th Annual Southeast Regional Conference*, ACM-SE 45, page 266–271, New York, NY, USA, 2007. Association for Computing Machinery.

[105] J. Zhou and Y.-h. Ou. Key tree and chinese remainder theorem based group key distribution scheme. In A. Hua and S.-L. Chang, editors, *Algorithms and Architectures for Parallel Processing*, pages 254–265, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Yıldız, Hüsnü
Nationality: Turkish (TC)
E-Mail: hyildiz@ceng.metu.edu.tr
Phone: 0 312 2105548

EDUCATION

Degree	Institution	Year of Graduation
Ph.D.	Middle East Technical University	2020
M.S.	Middle East Technical University	2014
B.S.	Eskişehir Osmangazi University	2009

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2010-2011	Middle East Technical University	Research and Teaching Assistant
2011-2020	Middle East Technical University	System Administrator

PUBLICATIONS

Book Chapters

[1] Hüsnü Yıldız, Adnan Kılıç, and Ertan Onur, Lightweight Cryptography in 5G Machine-Type Communication in Networks of the Future edited by Mahmoud

Elkhodr, Qusay F. Hassan, Seyed Shahrestani, ISBN:9781498783972

Master's Thesis

[2] Yıldız, H., 2014. A New collaborative filtering algorithm using near-clique bipartite graph clusters, MSc Thesis, Middle East Techical University.