Energy Efficient Secure Key Management Schemes for WSNs and IoT

by

Wen Wen

Thesis submitted to the Faculty of Graduate and Postgraduate Studies

In partial fulfillment of the requirements

For Master of Applied Science degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science

Faculty of Engineering

University of Ottawa

May 2015

Abstract

Secret sharing is critical to most applications making use of security and remains one of the most challenging research areas in modern cryptography. In this thesis, we propose a novel efficient multi-secret sharing scheme based on the Chinese remainder theorem (CRT) with two verification methods, while the previous works are mostly based on the Lagrange polynomial.

Key management schemes play an important role in communication security in Wireless Sensor Networks (WSNs). While the previous works mainly targeting on two different types of WSNs: distributed and hieratical, in this thesis, we propose our flexible WSN key management scheme, which is based on (*n*,*t*,*n*) multi-secret sharing technique, to provide a key management solution for heterogeneous architecture. The powerful key managers are responsible for most of the communicational and computational workload. They can provide Peer-to-Peer pair-wise keys for a pair of sensors to establish a secure communication session, and in the same time, they can also form communication clusters as cluster heads according to different application requirements.

Internet of Things (IoT) becomes more and more popular and practical in recent years. Considering the diversity of the devices and the application scenarios, it is extremely hard to couple two devices or sub-networks with different communication and computation resources. In this thesis, we propose novel key agreement schemes based on (*n*,*t*,*n*) multi-secret sharing techniques for IoT in order to achieve light weighted key exchange while using Host Identity Protocol (HIP). We refer the new schemes as HIP-MEXs with different underlying multi-secret sharing techniques. We analyzed the computational and communication costs of the extremely resource constrained device which is referred to as Initiator, and CRT based HIP-MEX successfully outsource the heavy workload to the proxy, which are considered more powerful, when establishing new secret key.

Table of Content

List of Figures

List of Tables

Acronyms

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
CRT	Chinese Remainder Theorem
DES	Data Encryption Standard
D-H	Diffie-Hellman
D-HIP	Distributed key exchange for HIP-based Internet of things
DNS	Domain Name System
DoS	Deny of Service
DWSN	Distributed WSN
ECDH	Elliptic Curve Diffie-Hellman
HIP	Host Identity Protocol
HIP-BEX	HIP Base Exchange
HIP-DEX	HIP-based
HIP-MEX-CRT	CRT based HIP Multi-secret sharing Key Exchange
HIP-MEX-LPI	Lagrange polynomial interpolation based HIP Multi-secret sharing Key Exchange
HWSN	Hierarchical WSN
IoT	Internet of Things
IPSec	Internet Protocol Security
LHIP	Lightweight HIP

M2M	Machine-to-Machine
MEMS	MicroElectroMechanical Systems
MSSS	Multiple Secret Sharing Scheme
NIST	The National Institute of Standards and Technology
P2P	Peer-to-Peer
PKI	Public Key Infrastructure
RFID	Radio-Frequency IDentification
SS	Secret Sharing
SSH	Secure SHell
TD-HIP	(*t*,*n*) Threshold Distributed key exchange for HIP
TLS	Transport Layer Security
WSNs	Wireless Sensor Networks

Acknowledgement

I would like to express my sincere gratitude to my supervisor Prof. Dimitrios Makrakis and my co-supervisor Prof. Carlisle Adams. Their guidance, instructions, and valuable advices helped me a lot with my thesis work and will continue helping me with my future career.

I would like to express my heartfelt gratitude to Dr. Binod. He has walked me through all the stages of this thesis. Without his consistent and illuminating instruction, this thesis would not have reached its present form.

Last, my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years.

Above all, I also own my sincere gratitude to my friends, my roommates and my fellow classmates who offered me their helps and time in listening to me and helping me with problems I encountered during the difficult course of the thesis.

And in the end, I am much appreciated these professors who take their valuable time to evaluate my thesis.

Thank you! And thank you very much for what you have done for me! Thank you to you all!

Chapter 1

Introduction

In recent years, security and privacy become more and more important in our daily life. Sensitive data are kept in various devices such as cell phones and personal computers, however, most of them are not well protected or under poor supervision while malicious attacks who are targeting sensitive information are growing extremely fast. The awareness of the importance of the security and privacy makes the topic of security very popular.

There are a lot of ways to achieve security and privacy, and among all those methods, cryptography is the most important and fundamental one. Cryptography has a very long history. More than two thousand years ago, Julius Caesar was already using simple Caesar cipher to deliver classified information for military and political purposes. Cryptographers and mathematicians worked for centuries to find better ways to encrypt the plaintexts as well as try to decrypt the ciphertexts. It was until 1976 [1], the society of cryptography ushered the biggest or the only "revolution" brought by Diffie and Hellman's public key cryptography. Since then, cryptography has two main branches, the first one is the conventional symmetric encryption, and the second one is the novel asymmetric encryption. We will use both types of encryption technologies later in this research work.

However, the advanced encryption technologies are still "not safe enough". On the contrary, because of the progress people have made in computer hardware and software, the electronic devices become more and more powerful which makes it much easier for a malicious attacker to

crack the poorly encrypted information. A simple way to counter this situation is extend the length of the encryption key, and the more complicated way is design different schemes using various technologies to ensure the security and privacy according to the different application scenarios. This increasingly growing requirement for security and privacy soon migrates from the legendary Internet to the Wireless Sensor Networks and the Mobile Networks both of which are consisted with relatively small and resource constrained devices.

Wireless Sensor Networks (WSNs) are widely accommodated in different areas performing various functions. Some areas, such as military and tele-health, have extremely high expectations on security and privacy performance of the WSNs since the data collected and transmitted by them are highly sensitive. Many schemes and approaches have been proposed to strengthen the security in WSNs, however, the efficiency and scalability are still far from perfect.

And the emerging of Internet of Things (IoT), which is currently a very hot topic in telecommunication area, brings even more challenges and opportunities in the same time. It raises the problem of coupling two devices which have different resources and capabilities. For example, the affordable symmetric encryption cannot be used when two devices do not share a symmetric key while the asymmetric encryption is too expensive or even impossible for a resource constrained device to perform.

In this thesis, we propose an efficient CRT based decentralized (n,t,n) multi-secret sharing scheme which is further exploited to build a key agreement scheme and a key management scheme, respectively. The proposed schemes can be used to achieve key establishment and key management for further private and secure communication.

1.1 Overview of Wireless Sensor Networks

Wireless sensor networks (WSNs) become more and more feasible and widely used since Micro-Electro-Mechanical system (MEMS) was introduced to manufacture small sensor devices [2]. A sensor node is a resource constrained device with one or more sensors integrated. Those sensors may have different functions, such as detecting and collecting different environment variables. And because of the sensor nodes are typically deployed in locations which are difficult to access, the collected data needs to be transmitted and reported to a base station or a sink through wireless communication techniques. Thousands of sensor nodes are usually deployed within a large area in order to monitoring and reporting the collected physical parameters [3]. Because of the limited energy resource that a sensor node has, its communication range tends to be short, thus making almost impossible for two nodes located far apart to communicate directly with each other. In most cases, communication between nodes is achieved through multi-hopping (similarly to the case of ad-hoc networks).

Several low power consumption wireless communication standards are designed for WSNs over the years [2], such as IEEE 802.15.4 [4], ZigBee [5] and ANT protocol [6]. ZigBee is widely used in many WSNs applications. This standard enables the formation of several types of topologies, as shown in Figure 1. In this figure, the end devices and routers are all sensor nodes despite they are shown differently. The ANT protocol is relatively new and is able to operate on resource constrained sensor devices. According to the analysis presented in [7], the ANT+ protocol, which is built based on the base ANT protocol, is more power efficient and faster than ZigBee.

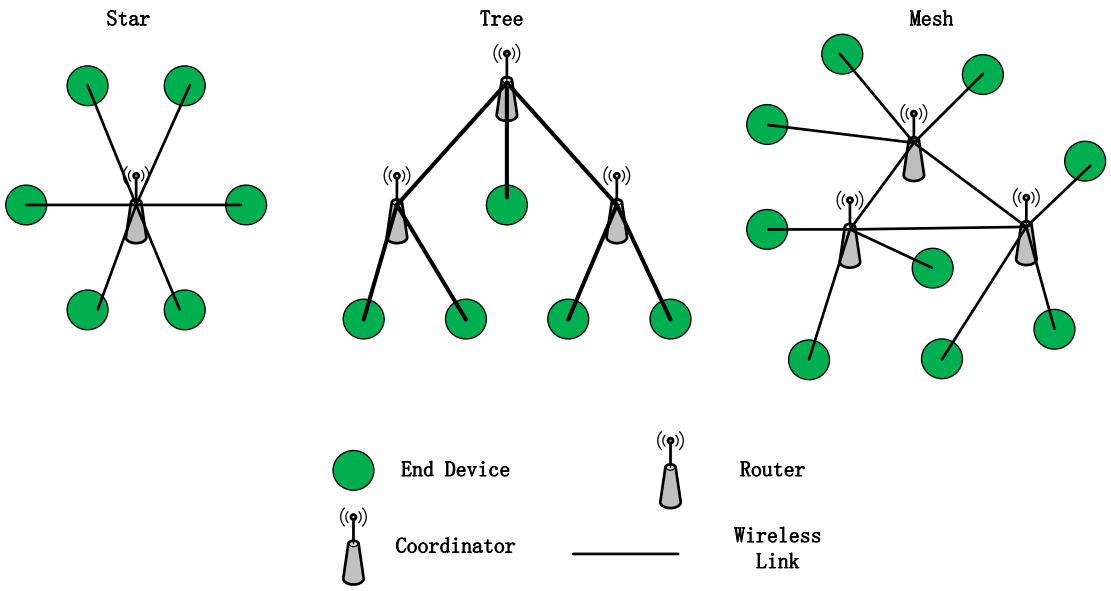

Figure 1 Common Topology of WSNs in ZigBee

WSNs can be classified in two categories based on their underlying communication architectures: hierarchical WSN (HWSN) and distributed WSN (DWSN) [8].

In HWSN, all the involved entities can be divided into three tiers. The first tier contains a centralized base station which is relatively powerful and controls the whole network. The second tier consists of several cluster heads. Each cluster head manages a set of sensor nodes and reports to the base station while able to behaving as sensor nodes in the same time. The third tier contains the resource constrained sensor nodes and they are used to collect data and submit the useful information to the cluster head. This approach is commonly used for data aggregation but suffers several disadvantages such as the single point failure.

In DWSN, all the sensor nodes are able to talk to each other directly, so it is more like P2P communication architecture rather than client-server architecture. That means, a sensor node can be sender and responder at the same time, and in order to achieve security requirements, a secure

communication channel needs to be established between every pair of sensor nodes. This decentralized property makes key management even harder for DWSN.

WSNs can be used in various applications and have the potential to be used in more areas. Military surveillance, health parameters monitoring and environment monitoring are three examples of the typical applications of WSNs. A main challenge encountered when deploying WSNs is the constraint of energy, computational and storage memory resources. As we discussed, the sensor nodes are most likely deployed in difficult-to-access areas with a large number, which makes the replacing of batteries extremely hard to impossible. In addition, the small size of sensor nodes limits the size of batteries, computational strength of processors and RAM. Most applications, such as e-health applications, long-term operation and reliable service is expected. So, the lack of resources limits the life time and the functionality of the sensor nodes, and further limits the application of WSNs. It is important to make all operations performed by small sensor nodes be as energy and computation efficient as possible.

1.2 Overview of Internet of Things (IoT)

Unlike the legacy Internet, IoT describes a paradigm where objects are part of the Internet and the whole society is "always connected" [9].

The transition from legacy Internet to IoT starts from Wireless Sensor Networks (WSNs), which allows similar wireless sensors to communicate in order to achieve certain functions [10]. Machine to Machine (M2M) communication [11] extended the model of WSN by introducing embedded intelligence and self-organization based on the wireless networks' capability of delivering broadband data service at a significantly low cost. The networks get more complex both logically

and topologically. Even devices which have considerably different capabilities and are located far apart can communicate under the concept of M2M. IoT is a further extension of M2M. It tends to interconnect wider sets of objects as well as achieve universality and interoperability [12].

The devices connected by IoT can be extremely powerful as servers, or be extremely resource constrained as RFID tags. In IoT, those devices are expected to exist everywhere, and they work and communicate automatically in order to minimum the human interactions. Those special features enable a lot of potential applications under the concept of IoT despite the fact that only a small part of them is currently available [13]. Among all potential applications, RFID related globalized cargo transportation is in the front line of the application of IoT [14].

1.3 Security in WSNs and IoT

WSNs and IoT share many similarities. For example; most of the communications are achieved through insecure wireless communication channels, which enables the attackers to capture the transmitted packets very easily. However, because of the differences between them, the secure requirements are not exactly the same. Later in this section, we will discuss about the security requirements and issues in both areas.

1.3.1 Security in WSNs

Security is one of the main challenges in WSNs due to the broadcast nature of the communication channel, the limited energy, computational and memory resource they tend to have, and their potential deployment in remote, and physically insecure areas. Certain WSNs become valuable targets to attackers because of their important functions and the sensitive information they are collecting and transmitting. WSNs are vulnerable to many attacks [15], such as DoS attack,

Man-in-the-Middle attack and black hole attack. The security concern is more serious in applications related to military and e-health. In WSNs, the goal of security is to efficiently achieve availability, survivability, scalability, confidentiality and integrity. Authentication, encryption and other techniques and protocols are proposed and widely used in WSN applications in order to cope with various attacks. But ultimately, the strength of the security schemes depends on the length and secrecy of the keys, and this is the reason that key management schemes are so important and fundamental in WSNs.

Using proper encryption techniques to protect the information is fundamental in achieving security in WSNs, and this requires the involved communication entities to have either symmetric keys or asymmetric keys for the encryption/decryption operations. The generating, distributing, exchanging, storage, use and replacement of keys needs to be properly managed using key management schemes [16]. Due to the diversity of WSN technologies/implementations and the plurality of WSN deployment areas abd applications, it is difficult to provide a "one solution fits all" solution in terms of securing transfer of information; the corresponding key management schemes may have to be selected/designed in accordance to the specific nature of deployed sensor nodes, the environment of deployment and the security requirements of the serviced application.

Key management schemes can be roughly classified into two groups: asymmetric encryption based and symmetric encryption based. The asymmetric encryption operations are considerably more resource consuming compared to the symmetric ones, and resource constrained sensor nodes cannot perform such complex operations. The symmetric encryption based schemes also have their own limitations, one of the major concerns been that the small size of memory placed on-board of sensor nodes limits the number of symmetric keys a node can store, which further limits the scale of

the network. However, because of the lower energy cost it has compared to asymmetric encryption techniques, the symmetric encryption based ones are popular and widely researched and applied. They are also the major focus in this thesis.

Dustin et al [8] gave a review on the existing key management schemes in WSNs. They classified the key management schemes into three categories: pair-wise key management schemes, random key-chain based key pre-distribution solutions, and network-wise key management schemes. In each category, there are several examples which exploit different underlying techniques. They also critiqued and rated those schemes using the following metrics: scalability, resilience, storage, power and communication.

Based on the architecture of the key management scheme, in general key distribution, the schemes can also be widely classified into three different categories: network-wise, pair-wise and group keying [15]. Network-wise key management means that all the members in the network share the same secret key. This approach is simple and easy to use, but if one of the members is compromised, the entire network is compromised. Pair-wise key management indicates each pair of the members within the network share a unique secret key. Assuming there are $\boldsymbol{n}$ sensors in the network, every sensor needs to remember ($\boldsymbol{n}$-1) keys while the whole network has $n(n-1)/2$ keys. This approach is not efficient enough and has limitations in terms of scalability and flexibility. Group key management is a clustered approach, and it combines the previous two approaches by allowing all the members in the same group share the same key while different groups use different keys. In Chapter 2, we will introduce and review some of the key management schemes proposed in the past according to this classification that is explained earlier in this

paragraph, and in Chapter 4 we propose our key management scheme that can be applied to WSNs with a flexible and heterogeneous architecture.

1.3.2 Security in Internet of Things

Potential application areas of WSNs and IoT, such as military, tele-health and commercial transactions usually involve sensitive, even highly classified information. How to achieve secure communication under the concept of IoT is becoming a primary problem and challenge. This problem is hard to solve because of two reasons. The first is that the vast number of involved objects makes it impossible for every device to become "known" to all the others, thus there is lack of trust among devices and lack of secure channels between them. The second one is that there is significant diversity and dissimilarity between devices in terms of energy availability, computing power and storage space. Thus, highly resource constrained devices cannot run heavy cryptographic algorithms like powerful devices do.

To address the first problem, the Host Identity Protocol (HIP) [17] was proposed. It introduced a new name space, which is similar to the name spaces we have in the legacy Internet: Domain Name Service (DNS) and Internet Protocol (IP) addresses. HIP also provides a secured Base Exchange (BEX) mechanism for two devices to agree on a shared key. In this thesis work, we propose to use HIP together with our novel key agreement scheme which is designed to address the second problem. Then, the resource constrained devices can utilize the HIP protocol to obtain a universal identity for identification purpose, and the associated key agreement or key exchange schemes can establish a secure communication channel between two devices.

1.4 Overview of Secret Sharing

Secret sharing means dividing one secret into pieces and sharing them with a set of shareholders. Each piece of the secret is called a share of the secret. An interesting example of secret sharing application is the storing of the nuclear missile launch code.

If the code is possessed by only one individual, the missile can be easily launched by mistake or maliciously. And if this individual is captured or killed, the missile cannot be launched because of the missing of the launch code. So the code should be shared by multiple people, each of them possessing only part of the code. When the code is needed, all these people will gather together and combine all the code parts. This method prevents accidental or malicious launch of the nuclear weapons, however, the problem still remains when someone who possesses partial code is captured or killed. How can we still be able to launch the missile with some parts of the code missing?

The concept of threshold secret sharing solves this problem. It allows the majority of the code owners to recover the rest of the code parts while preventing the minority from launching the missiles accidentally or maliciously. This concept applies the voting mechanism and relies on the judgment of the majority. This secret sharing concept has already been widely used in multiple areas, such as electronic voting, image encryption, et al.

The first threshold Secret Sharing (SS) scheme was independently proposed by Shamir [18] and Blakley [19] in 1978 for the safeguarding cryptographic keys. Since then, secret sharing has been intensively studied and used in various applications, such as key distribution and key exchange.

In Shamir's scheme, a centralized dealer shares a secret with multiple shareholders; however, any entity, including the shareholders, cannot retrieve the secret until it obtains the help of a certain number of shareholders. Secret sharing schemes with this property are called (***t***,***n***) threshold secret

sharing schemes, where ***t*** represents the minimum number of shareholders that are required to retrieve the secret and ***n*** stands for the total number of existing shareholders. This threshold property is an important feature when applying secret sharing in practical cases. This feature could make the scheme more robust against attackers and allows the system to work when some of the shareholders are compromised or disabled. Figure 2 shows the general concept of the threshold secret sharing.

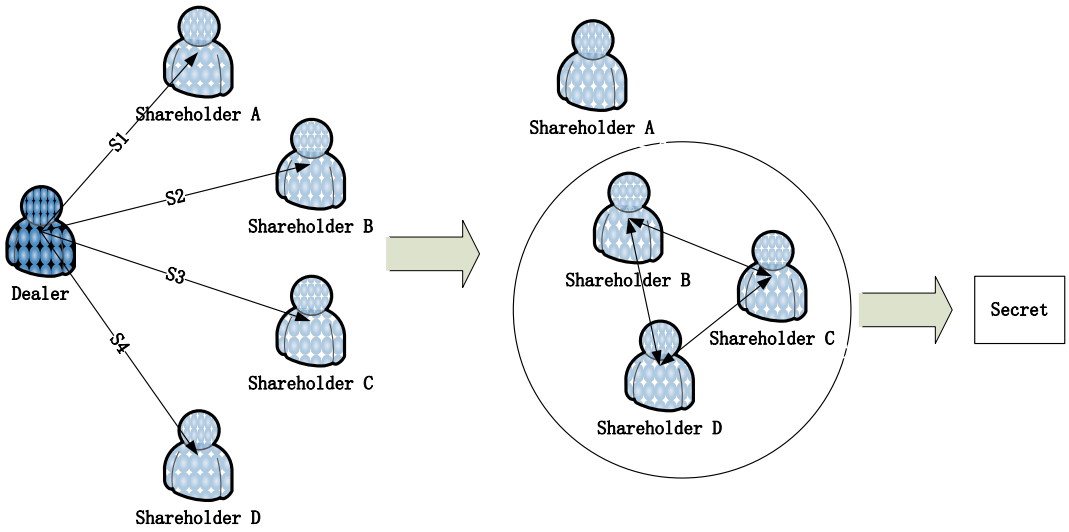

Figure 2 Basic Threshold Secret Sharing (shareholders B, C, D reconstruct final secret collaboratively without the help from shareholder A)

In 1982, Mignotte [20] proposed a new secret sharing scheme that is based on the Chinese Remainder Theorem (CRT) [21]. However, with this scheme, the exposure of any individual share can reveal some information of the original secret and narrow down the searching space of the secret, thus it is not theoretically secure. In 1983, Asmuth and Bloom [22] proposed another secret sharing scheme that is also based on CRT but incorporated some improvements. According to [23], Asmuth-Bloom secret sharing scheme is "asymptotically ideal and perfect zero-knowledge if the parameters of the system satisfy a natural condition", which makes the reveal of secret shares not

affect the size of the searching space too much and a minimum searching space is guaranteed. [23] also indicates that the computational complexity of the secret reconstruction from t shares for the Asmuth-Bloom's scheme is $O(t)$ while it is $O(t(\log^2 t))$ for the Shamir's secret sharing scheme. This makes the Asmuth-Bloom's scheme theoretically more computationally efficient than Shamir's scheme, since the architecture they are using is the same. The improvement comes from the use of different underlying mathematical mechanisms; CRT for Asmuth-Bloom scheme and Lagrange polynomial interpolation for Shamir's scheme.

Both, Shamir's scheme and Asmuth-Bloom's scheme can share only one secret by distributing a set of shares once. This is not efficient when there are many secrets to be shared, because distributing shares involves communication between the nodes and heavy encryption/decryption operations. Researchers then started to find different ways to share multiple secrets. They did so by distributing the shares to the shareholders only once. In 1994, He and Dawson [24] proposed a multi-stage secret sharing scheme. Since then, many multi-secret sharing schemes (MSSS) have been proposed and the design of all of them is based on the Lagrange polynomial interpolation. He and Dawson's scheme was soon proven to be actually a one-time secret sharing scheme, however most of the existing multi-secret sharing schemes, such as those described in [25] [26] [27], are relatively effective and efficient when the dealer wants to share more than one secret. Harn and Lin [28] proposed their strong (n,t,n) verifiable secret sharing scheme in 2010. This scheme led to the (n,t,n) multi-secret sharing scheme, proposed by Liu, Harn, Yang and Zhang in 2012 [29].

1.5 Problem Statement and Motivation

As we can see from the previous sections, many works have been done in the areas of multi-secret sharing, key management and Internet of Things. However, the existing solutions have their own limitations.

For the key management schemes in WSNs, most of them may lack flexibility, or can not provide adequate security, or have some other disadvantages. Nowadays, WSNs have been integrated to the Internet of Things. Also existing and new applications of the WSNs are becoming more complex and complicated [30]. Complex and highly intelligent applications and engineered structures can be manufacturing by combining large number of sensing, processing and actuating nodes of limited capabilities [31]. Such technology finds its way to supporting more critical tasks, thus making the impact of them compromised by cyber-attackers considerably more destructive, thus the urgency of equipping those nodes with stronger cyber-security technology becomes more paramount. At the same time, new factors such as node mobility are added, further complicating the development of such advanced cyber-security technology.

As for the previously mentioned HIP, which is an identity-based crypotosystem, is implemented using public key encryption technologies [32]. HIP provides an alternative to IP address and enables devices to achieve independent end-to-end operations even if the devices have changed their locations or accessing points. This allows devices to be accessed or authenticated everywhere using their unique identifications. HIP-BEX, provided by HIP, is designed to achieve authenticated key agreement between two HIP peers that have legitimate host identifiers using the Diffie-Hellman key exchange protocol [21]. The HIP peers will then be able to use symmetric encryption technologies instead of asymmetric encryption ones.

However, since Host identities in HIP are generated and verified using RSA and HIP-BEX is based on the Diffie-Hellman key exchange protocol [1], their involved heavy cryptographic operations make HIP computationally intensive and energy consuming. Thus HIP is not suitable for use with resource constrained devices. A number of research contributions were made, addressing this problem. HIP Diet key exchange (HIP-DEX) [33] and Lightweight HIP [34] are two examples that proposed to reduce the resource consumption of the key establishment operations.

In 2011, Moskowitz et al [33] proposed HIP Diet EXchange (HIP-DEX) to further improve the HIP-BEX by using long term Elliptic Curve Diffie-Hellman (ECDH) public value as HIP hosts identifiers. HIP-DEX requires one pair of DH keys instead of two pairs that are required by HIP-BEX. Lightweight HIP (LHIP) [34], proposed by T. Heer in 2007, only uses hash chains to bind successive messages to provide security of communication. However, LHIP only guarantees the ongoing session is not hijacked, which is minimal security requirement. In 2012, Y. B. Saied and A. Olivereau [12] [35] [36] proposed their schemes to replace the heavy Diffie-Hellman key agreement of HIP-BEX, including this (*t*,*n*) threshold distributed key exchange for HIP [36], which we refer to as TD-HIP in later discussion. They successfully replaced DH key exchange with lighter public key infrastructure (PKI) and introduced a new collaborative approach to share the heavy workload of the constrained host. In TD-HIP, Shamir's (*t*,*n*) secret sharing scheme [18] is used to allow the newly introduced third entity, proxy, to collaboratively help the resource constrained device during the key agreement process.

However, the previous works are still not efficient enough for a highly resource constrained device to frequently establish a shared key for every communication session. In order to design a

more efficient and more flexible key management and key agreement scheme, we propose our own multi-secret sharing schemes based on the concept of (***n***,***t***,***n***) multi-secret sharing.

The existing (***n***,***t***,***n***) multi-secret sharing schemes are not resource efficient enough for the targeting key management scheme and the HIP-based key exchange scheme, to address this problem, we proposed our CRT-based (***n***,***t***,***n***) multi-secret sharing scheme. To the best of our knowledge, except of one scheme reported in [37], which shares multiple secrets with different groups, no multi-secret sharing scheme based on CRT has been proposed until now.

1.6 Our Contribution

In this thesis, we propose an efficient and CRT based decentralized (***n***,***t***,***n***) multi-secret sharing scheme. We also propose a new method of generating verifiable shares, thus making the proposed scheme a verifiable multi-secret sharing scheme, without revealing information about the final secret. The previous works have proposed similar (*n*,*t*,*n*) multi-secret sharing approach, however, we provide a more energy efficient way of generating and distributing secret shares. The possible communicational and computational costs of the secret holder are significantly reduced. And based on this research work, we published a paper "Decentralized CRT-Based Efficient Verifiable (***n, t, n***) Multi-secret Sharing Scheme" [38] in 2015.

We propose novel key agreement schemes based on (*n*,*t*,*n*) multi-secret sharing techniques for the global HIP-based IoT infrastructure, in order to achieve lightweight key exchange between HIP peers with different resources and capabilities. The proposed schemes have used different underlying multi-secret sharing techniques but share a similar collaborative approach that enables reduction of consumed energy. We analyzed the computational and communication costs

occurring for a highly resource constrained device. The proposed Chinese Remainder Theorem (CRT) based key agreement scheme successfully reduced the energy cost for both perspectives. Most of the required processes have been outsourced to a more powerful and resourceful proxy. With the proxy collaboratively helping the resource constrained device during the key exchange process, the energy consumption occurring due to the complex public key encryption processes does not occur at the resource constrained device. This allows two devices having significantly different resource constraints to be able to communicate secretly, while satisfying the constraints. However, because of the time limitation, this key management scheme is not thoroughly evaluated in terms of energy and time efficiency. This will be the future work.

We also proposed a multi-secret sharing based key management method, having a heterogeneous architecture. Our key management scheme is using relatively resource sufficient and powerful assistant nodes that are acting as the decentralized key infrastructure managers, the intermediate packet hoping nodes and in some cases the cluster heads of a hierarchical WSN. By using this approach, we outsource expensive computational operations from the highly resource constrained nodes to the key managers which are powerful and replaceable, and allow all the sensor nodes to be able to obtain session keys or group session keys, having only few pre-distributed keys stored. The proposed multi-secret sharing based distributed key management scheme achieves flexible and efficient management of the secret keys, allows the joining and leaving of the sensor nodes and key managers, and it also guarantees the security and privacy of the entire system.

The rest of the thesis is organized as follows. In chapter 2, we give some background and review previous related works. In chapter 3, our decentralized CRT-based (***n***,***t***,***n***) multi-secret sharing

scheme is presented in detail. In chapters 4 and 5, we discuss about the key agreement and key management schemes, created using the (***n***,***t***,***n***) multi-secret sharing techniques. Chapter 6 provides conclusions and present potential future work.

Chapter 2

Literature Review

Nowadays, WSNs and IoT are becoming more and more popular and practical for use in different areas. While new technology has been proposed solving satisfactory a number of challenges such as routing and interference, cyber-security remains a major challenge, due to the relatively low resource cost and hardware requirements such devices have. With plenty of the secret sharing schemes proposed in the open literature, many of them have been applied to design key establishment schemes and key management schemes.

The underlying mathematic foundation for most of the existing secret sharing schemes proposed for the mentioned area are the Lagrange polynomial interpolation and the Chinese Remainder Theorem (CRT).

Since the secret sharing schemes also apply some cryptographic techniques, such as Data Encryption Standard (DES) and RSA, in our own multi-secret sharing scheme and the extended key management and key agreement schemes, those techniques are also utilized. Hence, in section 2.1, we give a brief review of cryptography and some related concepts and techniques.

In section 2.2, we introduce the basic mathematic formulations and several representational secret sharing schemes, in section 2.3 we give a brief review on the proposed multi-secret sharing schemes. As we discussed in Chapter 1, the design of most of the multi-secret sharing schemes is based on Lagrange polynomial interpolation, and is used with different approaches.

We present some previous works in the areas of key management and key exchange in section 2.4 and 2.5, respectively, and the focus is placed on works which accommodate the secret sharing schemes. Upon the reviewing of the previous schemes, we compare their advantages and disadvantages and provide a better understanding of the related areas.

However, key distribution or key exchange is just part of the key management or key agreement schemes. In our key management and key agreement schemes, we also applied some techniques different from just the (***n***,***t***,***n***) multi-secret sharing. For example, we use Merkle tree structure [39] to authenticate multiple devices and the Host Identity Protocol to achieve universal identification verification. Those related techniques are also introduced and presented in section 2.6.

2.1 Brief Review of Cryptography

Cryptography studies and designs technology for data encryption and decryption [21]. Encryption is the process of converting the original message, or referred to as plaintext, to an encrypted message, ciphertext. Decryption is the inverse process of encryption.

Cryptography has a long history going back thousands of years. However, along with the great advance in computer technology and mathematics, most of ciphers or encryption technologies are not secure nowadays. Today's encryption technologies can be divided into two main systems: symmetric encryption and asymmetric encryption.

Most commonly used symmetric encryption technology was DES using feistel network.. Feistel networks [40] were designed by Horst Feistel, an IBM cryptography researcher, and were first seen in IBM's Lucifer cipher which was co-designed by Horst Feistel and Don Coppersmith [41]. The Lucifer cipher was then modified by the NSA and gained DES. But DES-like

cryptosystems are still utilizing the conventional cryptographic techniques such as substitution and diffusion [41]. However, DES is now considered lack of enough capacity to withstand an attack, and in 1997, NIST recommended a new national standard would be called as AES (Advanced Encryption Standard), which was presented later in 2001 and uses permutation-substitution instead of Feistel network, to replace DES [42].

The real significant leapfrog evolution in cryptography is the introduction of the public key cryptography in 1976 by Diffie and Hellman [1], also known as asymmetric encryption. However public key cryptography is also vulnerable to brute-force attack, which forced the public key encryption algorithms having to use large keys. Another way of breaking public key encryption is by deriving the private key from the public key, however this way is not mathematically feasible to date

In the following two sub-sections, we will introduce the modern symmetric encryption and asymmetric encryption techniques.

2.1.1 Symmetric Encryption

Symmetric encryption, which is also referred as private key encryption, is used to provide secure communications over insecure channels. The encryption and decryption need to be done using the same secret key pre-shared in advance or obtained following the same operations [43]. Figure 3 Symmetric Encryption with a Trusted Third Party Acting as the Key Distributer shows the symmetric encryption in a general manner. As shown in the figure, the shared key can be installed or agreed in advance, or obtained from a trusted third party through secure channels. This channel can be established by pre-shared keys and symmetric encryption, or by using public key encryption.

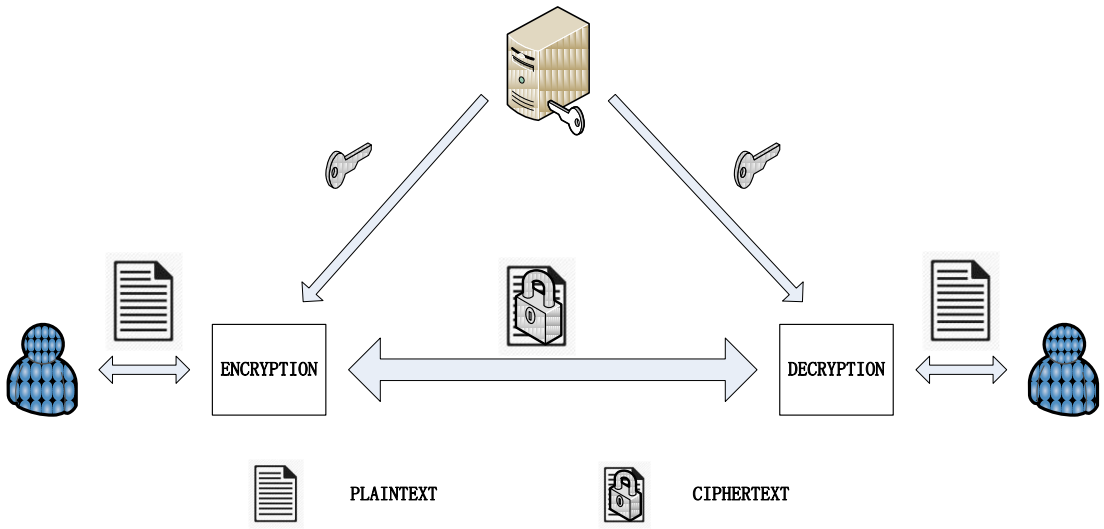

Figure 3 Symmetric Encryption with a Trusted Third Party Acting as the Key Distributer

The most commonly used symmetric encryption algorithms are Advanced Encryption Standard (AES), RC2, Blowfish and RC6 [44]. According to the analysis presented in [44], AES has a better performance and was designed to replace the widely used DES and 3DES. Thus, in the rest of the thesis, we select AES as the used symmetric encryption technique.

AES was proposed by the U.S. National Institute of Standards and Technology (NIST) in 2001 [45]. AES uses 128-bit block size and has three different key length options which are 128 bits, 192 bits and 256 bits, respectively. AES uses permutation-substitution and the multiple rounds structure to achieve better security, for different key size, the number of rounds is different with 10 rounds for a 16-byte (128-bit) key, 12 rounds for a 24-byte (192-bit) key, and 14 rounds for a 32-byte (256-bit) key [21]. Most of the low power wireless communication standards and devices supports the use and implementation of AES-128, such as AES-CBC-MAC-128, which is a strong security mechanism used for ZigBee [7]. We choose AES-256 to provide secure channels between entities. The security strength for AES-256 is currently considered enough for our applications considering the fact that some attacks have shown some promise in terms of cracking AES-128 [46] [47].

2.1.2 Asymmetric Encryption

Unlike symmetric encryption, asymmetric encryption requires two different keys: private key which is kept secret by the entity and public key which is exposed to all the entities. The encryption and decryption processes use different keys.

In order to better explain the mechanism of public key encryption technique, we use the scenario which Alice wants to send a message to Bob privately. Alice possesses Bob's public key and Bob keeps his private key a secret. Figure 5 shows the asymmetric encryption process for the considered scenario. In the basic application of asymmetric encryption, which is encrypting a message into ciphertext, the encryption process uses the public key

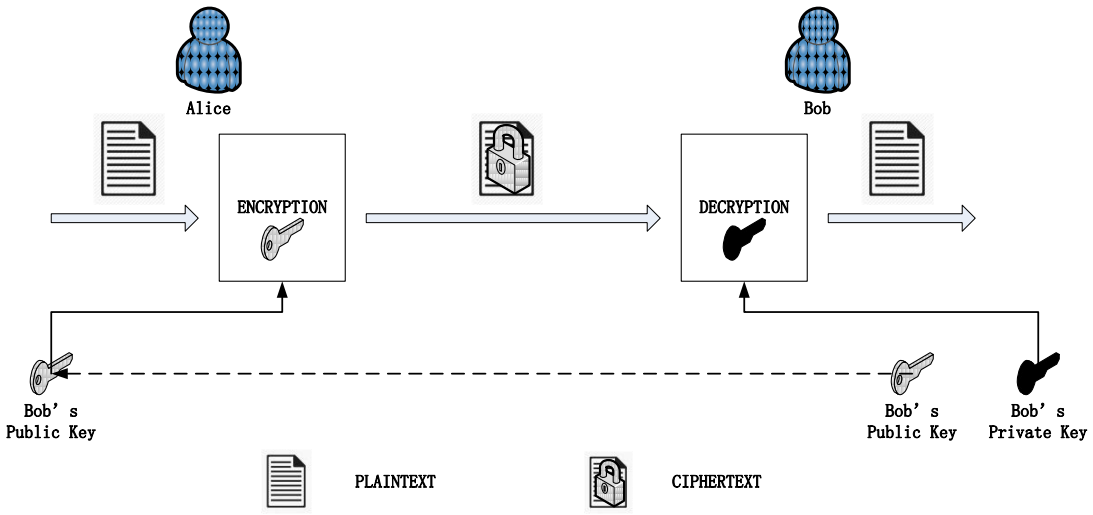

Figure 4 Alice Sends Message to Bob Using Asymmetric Encryption [21]

As we can see from Figure 4 Alice Sends Message to Bob Using Asymmetric Encryption, Alice and Bob do not need to pre-share a secret key between each other as they did in symmetric encryption. When Alice initiates the communication session, she will obtain the public key of Bob and start encrypting and transmitting. Despite the fact that Bob's public key is publicly known to everyone,

an adversary cannot decrypt the ciphertext since it can only be decrypted using Bob's private key, which is kept privately, known only by Bob.

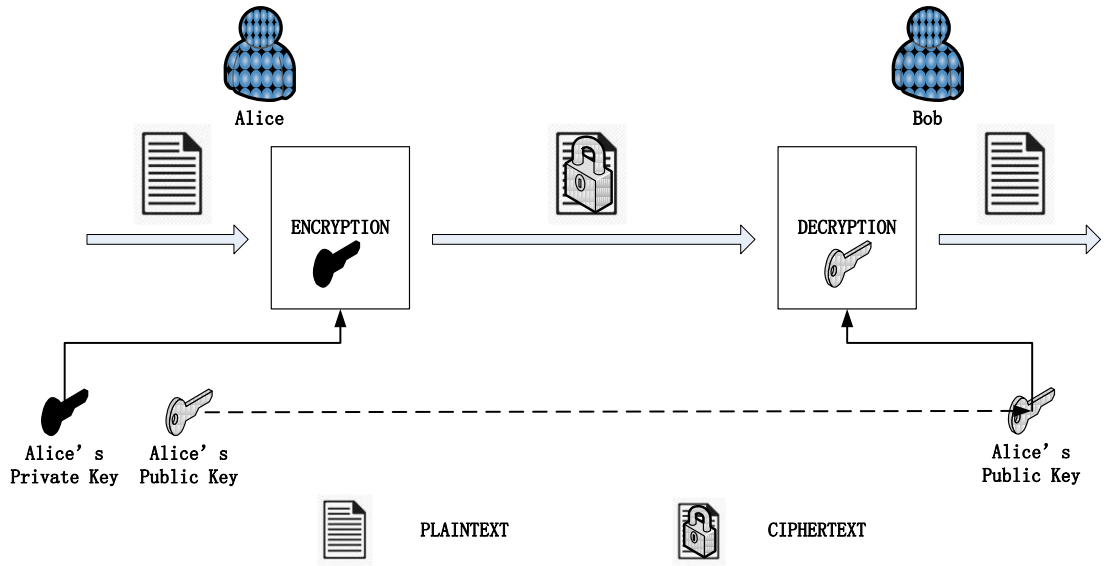

Figure 5 Alice Signs a Signature and Sends to Bob, Bob Verifies the Signature Using Alice's Public Key [21]

Figure 5 shows another way of using a pair of asymmetric keys. When Alice wants to send some information to Bob, she uses Bob's public key to encrypt the plaintext which is only able to be decrypted by Bob's private key. However, when signing a signature, which is used to indicate the source of the message, Alice will need to use the information only possessed by her, which is her private key. As shown in Figure 5, Alice encrypts the plaintext using her private key, and the resulting ciphertext could only be decrypted using Alice's public key. It is clear that all the entities which possess the public key of Alice can decrypt the ciphertext, but the content of the decrypted plaintext does not need to be kept secret. And since they do not have the corresponding private key, they cannot forget the signature which can only be signed by Alice. This digital signature technique is extremely important and is widely used over many applications. In our proposed schemes, we also utilize this technique for authentication and identification purposes.

As we indicated above, there are two common ways the attackers can break the cipher. The first is brute-force attack which requires the adversary to try all the possible private keys to decrypt the ciphertext. The other way is to derive the private key from the public key since the two keys are linked mathematically.

The existing public key encryption algorithms and standards are based on Mathematical problems, such as integer factorization, discrete logarithm and elliptic curve, which have no current efficient solution [48]. Some of the most commonly used public key encryption algorithms are the RSA Cryptosystem [49] (stands for its three designers Ron Rviest, Adi Shamir and Lenard Adleman), the Diffie-Hellman key exchange [1], the ElGamal Cryptosystem [50] and the Elliptic Curve Cryptosystems [51] [52].

The Diffie-Hellman [1] key exchange scheme is the most commonly used public key infrastructure for key exchange and key distribution. It is also the first published public-key algorithm in open literature and its purpose is to enable two users to securely exchange a key for further communication. Many cryptographic protocols, such as Secure Shell (SSH) [53] [54] [55], Transport Layer Security (TLS) [56] and Internet Protocol Security (IPSec) [57], implement this fundamental key agreement technique [48]. Following is the brief description of the algorithm as shown in Figure 6.

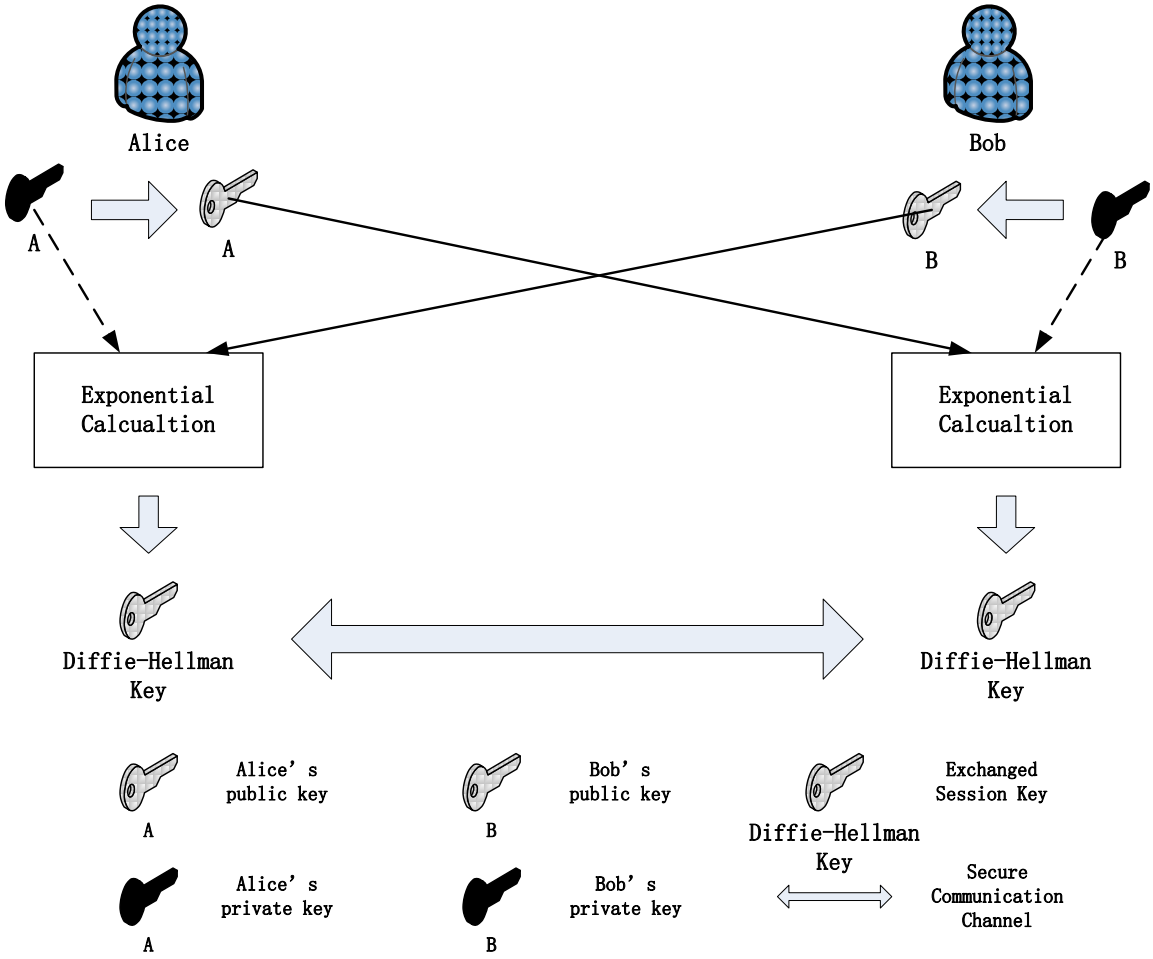

Figure 6 Diffie-Hellman Key Exchange Between Alice and Bob [21]

The Diffie-Hellman key exchange algorithm exploits the discrete logarithm problem. It helps two communication entities derive a shared key to achieve secure communication over insecure channel. But for networking comprised by devices of low computing power and storage memory, such as WSNs and IoT, the Diffie-Hellman key exchange is too expensive because of the intensive exponential computation operations. We will discuss more about this in later sections.

2.2 Basic Mathematics

For further discussion, we need to introduce the underlying mathematic mechanisms related to secret sharing. Most of the commonly seen secret sharing schemes are using two mathematic

mechanisms to achieve secret sharing. The first is Lagrange polynomial interpolation; the second is the Chinese Remainder Theorem. In this sub-section, we briefly introduce these two theorems.

2.2.1 Lagrange Polynomial Interpolation

Given a set of distinct points x_i and corresponding numbers y_i, a Lagrange polynomial is the polynomial of the least degree which could draw a graph passing through all the points with coordinates (x_i, y_i).

Consider a given set of points are $\{(x_0, y_0),(x_1, y_1),...,(x_{t-1}, y_{t-1})\}$, where the members in $\{x_0, x_1,..., x_{t-1}\}$ are distinct.

Since there are "*t*" different solutions, the degree of the corresponding Lagrange polynomial *f*(*x*) is (*t*-1). Its expression is:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \tag{2.2.1}$$

We can easily get:

$$\begin{aligned} &f(x_0) = y_0 \\ &f(x_1) = y_1 \\ &\vdots \\ &f(x_{t-1}) = y_{t-1} \end{aligned}$$

Then the Lagrange polynomial interpolation can be expressed as follows:

$$f(x) = \sum_{j=0}^{t-1} f(x_j)\left(\prod_{r=0, r\neq j}^{t-1} \frac{x - x_r}{x_j - x_r}\right) \tag{2.2.2}$$

2.2.2 Chinese Remainder Theorem

The Chinese Remainder Theorem was first proposed in the 3rd to 5th centuries by Chinese mathematician and strategist Sun Tzu [58].

Description of CRT [21]**.**

Suppose $m_1, m_2, ..., m_t$ are positive co-prime integers, then, for a given sequence of integers $a_1, a_2, ..., a_t$, there exists an integer A that could solve the following system of simultaneous congruencies:

$$\begin{aligned} A &\equiv a_1 (\mathrm{mod}\, m_1) \\ A &\equiv a_2 (\mathrm{mod}\, m_2) \\ &\vdots \\ A &\equiv a_t (\mathrm{mod}\, m_t) \end{aligned}$$

The problem here is how to get A when given $\{(a_1, m_1), (a_2, m_2), ..., (a_t, m_t)\}$.

Let $M = \prod_{i=1}^{t} m_i$, $M_i = M / m_i$, $c_i = M_i \times (M_i^{-1} \bmod m_i)$, where $1 \le i \le t$. Note: $M_i^{-1} \bmod m_i$ is the multiplicative inverse of M_i in GF(m_i). The calculation of multiplicative inverse will be discussed in detail next.

A can be derived as follows:

$$A \equiv (\sum_{i=1}^{t} a_i c_i) (\mathrm{mod}\, M) \qquad (2.2.3)$$

There is another assertion of the CRT, concerning arithmetic operations, follows from the rules for modular arithmetic:

For

$A \leftrightarrow (a_1, a_2, ..., a_t)$

$B \leftrightarrow (b_1, b_2, ..., b_t)$

which means:

$$\begin{aligned} &A \equiv a_1 (\mathrm{mod}\, m_1); B \equiv b_1 (\mathrm{mod}\, m_1) \\ &A \equiv a_2 (\mathrm{mod}\, m_2); B \equiv b_2 (\mathrm{mod}\, m_2) \\ &\vdots \\ &A \equiv a_t (\mathrm{mod}\, m_t); B \equiv b_t (\mathrm{mod}\, m_t) \end{aligned}$$

Then:

$$(A \pm B) \bmod M \leftrightarrow ((a_1 \pm b_1) \bmod m_1, (a_2 \pm b_2) \bmod m_2, ..., (a_t \pm b_t) \bmod m_t) \qquad (2.2.4)$$

$$(A \times B) \bmod M \leftrightarrow ((a_1 \times b_1) \bmod m_1, (a_2 \times b_2) \bmod m_2, ..., (a_t \times b_t) \bmod m_t) \qquad (2.2.5)$$

This assertion is proved to be very useful in multi-secret sharing schemes which will be discussed later.

Calculation of Multiplicative Inverse.

As we discussed above, the calculation of multiplicative inverse is required in the reconstruction of the integer during the CRT operations.

In modular arithmetic, modular multiplicative inverse of $a \bmod m$ is the integer x which makes $a \bullet x \bmod m = 1$ [59]. Which means $x \equiv a^{-1} \bmod m$. And if ***a*** and ***m*** are co-prime to each other, which means greatest common divider ***gcd***(***a***, ***m***)=1, there is only one possible ***x*** within GF(*m*).

The calculation of the modular multiplicative inverse could be done by two different ways. The first commonly used way is by using the extended Euclidean algorithm [21] shown in the following equations. Assume *x* is the multiplicative inverse of $a \bmod m$ with ***a*** and ***m*** being co-prime to each other, *q* is an integer. The following equations hold:

$$a \bullet x \bmod m = 1 \qquad (2.2.6)$$

$$ax - 1 = qm \qquad (2.2.7)$$

$$ax - qm = 1 \qquad (2.2.8)$$

Since *m* and *a* are given, *x* and *q* could be found accordingly.

Another way of computing multiplicative inverse is using Euler's theorem [21].

According to Euler's theorem, if ***m*** is a prime number ***a*** is a positive integer, they co-prime to each other, which means ***gcd***(***a***, ***m***)=1, then:

$$a^{\varphi(m)} \equiv 1 (\bmod m) \qquad (2.2.9)$$

$\varphi(m)$ is Euler's totient function. it stands for the number of positive integers smaller than *m* and relatively prime to *m*. If *m* is a prime number, then all the positive integers smaller than *m*, including number 1, are relatively prime to *m*, which results to $\varphi(m) = m - 1$. Thus Eq.(2.2.9) can be modified to Eq.(2.2.10)

$$a^{m-1} \equiv 1 (\mathrm{mod}\, m) \tag{2.2.10}$$

From Eq.(2.2.10) we can deduce Eq.(2.2.11) below:

$$a^{-1} \equiv a^{m-2} (\mathrm{mod}\, m) \tag{2.2.11}$$

and according to the definition of multiplicative inverse, we have:

$$x \equiv a^{-1} \equiv a^{m-2} (\mathrm{mod}\, m) \tag{2.2.12}$$

Thus when ***a*** and ***m*** are given, the multiplicative inverse can be found using Eq.(2.2.12).

The module calculation is relatively simple, but the reconstruction operation is quite the opposite. It involves multiple multiplications and expensive exponential computations required by the calculation of multiplicative inverse.

2.2.3 Primality Test

The selection of prime numbers is important in later discussions, and the CRT already involved prime numbers. In this sub-section, we are going to talk about how to obtain large prime numbers.

Usually, the random prime numbers are generated following two steps, the first is generate a random number within a certain range, the second is testing the primality of the random number, if this number is not a prime number, repeat the two steps. Miller-Rabin primality test or Rabin-Miller primality test, designed by Miller [60] and Rabin [61], is an algorithm that typically used to test a large number for primality [21].

Assuming ***n*** is a random positive odd integer with $n > 3$, the following pseudo-code [21] is the detailed procedure for describing the Miller-Rabin primality test.

TEST(***n***)

Find positive integer *k* and odd integer *q* so that $(n-1=2^k q)$.

Select x random integer *a* from the range $(1, n-1)$, for each a repeat the following process, the probability of declaring a composite as a prime number is at most 4^{-x}.

If $a^q \equiv 1 (\mathrm{mod}\, n)$, *n* could be a prime number, so return "inconclusive".

For integer *j* = 0 to *k*-1, do

If $a^{2^j q} \equiv (n-1) (\mathrm{mod}\, n)$, then return "inconclusive";

If not, return "composite" which indicates *n* is not a prime number. Then re-select *n* and perform the test again.

It becomes evident by reading the pseudo-code that the Miller-Rabin primality test algorithm cannot guarantee all the integers that passed this test are prime numbers (the test classifies the outcome "inconclusive"). There is a small probability that some of the integers which are not prime numbers could still pass the test. However, that probability is too small and we could normally trust the testing result.

2.3 Basic Secret Sharing Schemes

In Chapter 1, we briefly introduced the basics of the secret sharing, and we will discuss more about several basic schemes in this chapter.

Before the introduction, we need to make some assumptions and preparations.

In a (***t***,***n***) secret sharing scheme, usually there are two different roles, dealer and shareholder. Dealer is the one which has a secret to share and is responsible for dividing a secret into multiple

shares, and the shareholder is responsible for holding one piece of the share of the secret. The process that a dealer sends different shares to corresponding shareholders is called share distribution. Dealer may or may not delete the secret from its memory depending on the application requirement, and here it is usually assumed that one of the shareholders is trying to recover the secret with the help from the other shareholders. ***n*** stands for the total number of the shareholders, and ***t*** stands for the threshold of the scheme, which means at least ***t*** shareholders are required to retrieve the secret.

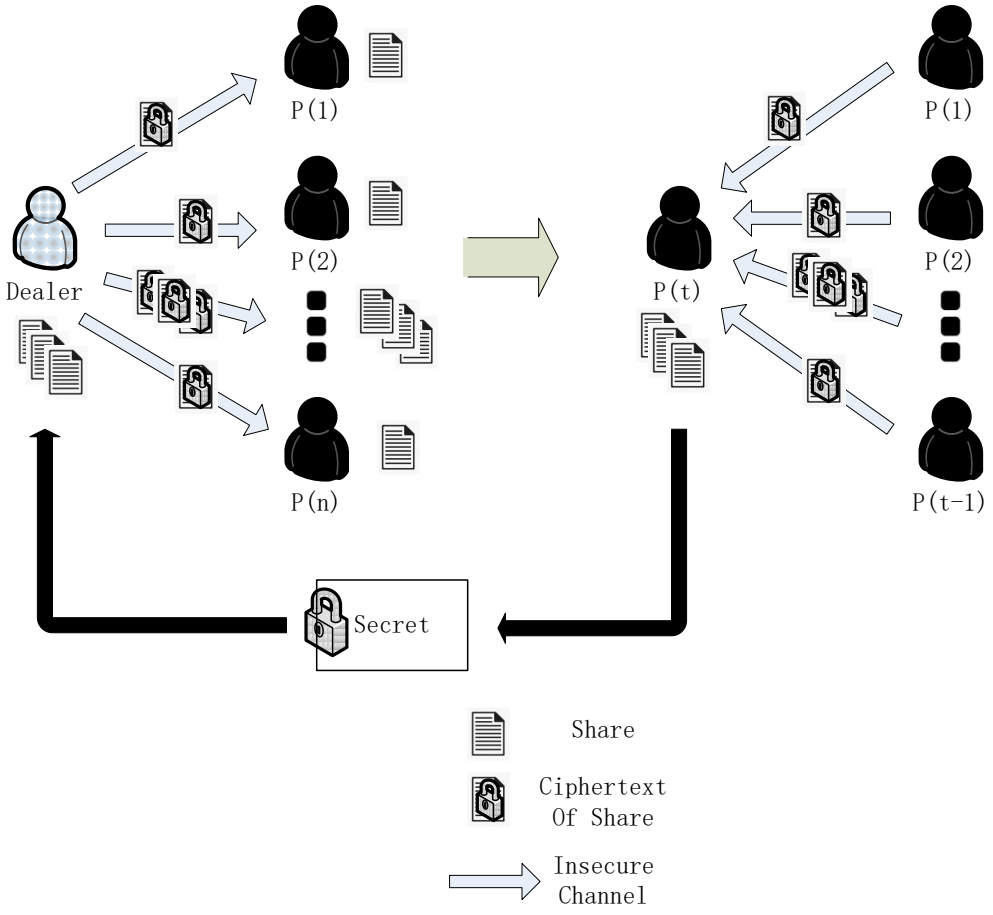

Figure 7 Generalized and Detailed Architecture for (*t*,*n*) Secret Sharing

In later discussion, we denote the dealer as D and the shareholders (or proxies) as P_i with $i = 1, 2, ..., n$.

In Figure 2, we gave an example of the architecture of secret sharing schemes. Here, we give a more precise and generalized one in Figure 7. In this figure, P(*i*) stands for shareholder, with *i=1,2,…,n*, and all the shares are encrypted first and then transmitted to the shareholders through insecure channels. The secure channel is achieved using symmetric encryption techniques, such as AES-256 as we previously discussed. In later discussion, we assume there is a secure channel between each pair of participants, both dealer and shareholders. And in this stage, we do not consider the existence of the compromised shareholders, all the involved parties are trustworthy, the secure channel is used to prevent the attacker from eavesdropping.

2.3.1 Shamir's (***t***,***n***) Secret Sharing Scheme [18]

Shamir's secret sharing scheme uses the Lagrange Polynomial Interpolation. Denote the subject who wants to share a secret as the dealer ***D***, or sometimes called secret-holder, name the subjects who keep one share of the secret as participants $\boldsymbol{P_i}$, sometimes referred as share-holders.

In the share generation/distribution stage.

Dealer ***D*** randomly selects $\{\boldsymbol{S}, \boldsymbol{a_1},...,\boldsymbol{a_{t-1}}\}$, where ***S*** is the secret that is needed to be shared later. ***S*** and $\boldsymbol{a_i}$ are used to construct a random polynomial $f(x)$ of degree (***t***-1):

$$f(x) = S + a_1 x + ... + a_{t-1} x^{t-1} (\text{mod } p) \qquad (2.3.1)$$

S and all $\boldsymbol{a_i}$ belong to the finite filed GF(p) with $p>\boldsymbol{S}$, where p is a large prime number. ***D*** computes ***n*** shares: $\{s_1 = f(x_1), s_2 = f(x_2), ..., s_n = f(x_n)\}$.

Note: the values of unknown integer ***x*** can be set as simple as 1, 2, ..., ***n***, or selected randomly from a finite group. The exposure of these values does not affect the secrecy of the secret.

D then distributes each share $\boldsymbol{s_i}$ to the corresponding shareholder $\boldsymbol{P_i}$ through a secret channel.

In the reconstruction stage.

Assume there are ***t*** participants, $\boldsymbol{P_1,...,P_t}$, which wish to reconstruct the secret ***S***. The Lagrange polynomial interpolation has the following expression:

$$f(x) = \sum_{j=0}^{t-1} f(j)\left(\prod_{r=0, r\neq j}^{t-1} \frac{x-r}{j-r}\right) (\text{mod } p) \tag{2.3.2}$$

So if set ***x***=0, the final secret ***S*** could be derived by simply applying Lagrange Interpolation formula as shown in Eq.(2.3.3)

$$S = f(0) = \sum_{j=1}^{t} f(j)\left(\prod_{r=1, r\neq j}^{t} \frac{-r}{j-r}\right) (\text{mod } p) \tag{2.3.3}$$

2.3.2 Mignotte's (*t*,*n*) Secret Sharing

In this scheme, CRT was used to share a secret for the first time. Dealer will select a sequence of prime numbers $m_1 < m_2 \cdots < m_n$ such that $\prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$, which means the product of any ***t*** integers is always greater than the product of any (***t***-1) integers. Here, ***n*** is the total number of the shareholders and ***t*** is the threshold. We note that the regulations above give the secret sharing scheme (***t***,***n***)-consistency. A set of ***n*** shares is (***t***,***n***) consistent only if all the combinations of at least ***t*** shares (each combination could be called an access structure) within the set can reconstruct the same secret. This (***t***,***n***)-consistency is the key to achieve the threshold property and will be used to test the correctness of the collected shares of the final secret.

Then dealer generates a secret ***S*** and using the following equation to generate the corresponding shares s_i of this secret.

$$s_i \equiv S(\text{mod } m_i) \tag{2.3.4}$$

Those shares are secretly distributed to the corresponding shareholders, and when one of the shareholders wants to reconstruct the secret, it will gather the information from at least ***t*** shareholders including itself. Assuming the set of shareholders used to reconstruct the secret is $\{P_1, P_2, ..., P_t\}$, using Eq.(2.2.3) to calculate the secret ***S***.

$$S \equiv (\sum_{i=1}^{t} s_i c_i) (\mathrm{mod}\, M) \tag{2.3.5}$$

Here, $M = \prod_{i=1}^{t} m_i$, $M_i = M / m_i$, $c_i = M_i \times (M_i^{-1} \bmod m_i)$, $1 \le i \le t$.

2.3.3 Asmuth-Bloom (***t,n***) Secret Sharing

In this scheme, two integers ***t*** and ***n*** are given, with $1 \le t \le n$ and $n \ge 2$. Then select a sequence of pair-wised co-prime positive integers $m_0 < m_1 < \cdots < m_n$ such that: $m_0 \prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$. This regulation of the selected co-prime numbers gives this scheme the (***t,n***)-threshold property.

Then the shares used to share a secret ***S*** $(S \in (0, m_0))$ are generated as follows:

$$y = S + \alpha m_0 \tag{2.3.6}$$

$$Sh_i \equiv (S + \alpha m_0) (\mathrm{mod}\, m_i) \tag{2.3.7}$$

Here, α is a positive integer randomly selected subject to the condition $1 \le y \le M$ where $M = \prod_{i=1}^{t} m_i$.

As we discussed above, according to CRT, we could reconstruct the secret by applying the following processing:

$$S + \alpha m_0 \equiv (\sum_{i=1}^{t} Sh_i \cdot c_i) (\mathrm{mod}\, M) \tag{2.3.8}$$

$$S \equiv (S + \alpha m_0) (\mathrm{mod}\, m_0) \tag{2.3.9}$$

Where $M = \prod_{i=1}^{t} m_i$, $M_i = M / m_i$, $c_i = M_i \times (M_i^{-1} \bmod m_i)$, $1 \le i \le t$.

Quisquater et al. [23] analyzed the schemes based on CRT and indicated that Asmuth-Bloom's scheme is asymptotically ideal and perfect zero-knowledge if the parameters of the system satisfy a natural condition. That means, any (***t***-1) shares of such a (***t***,***n***) threshold secret sharing scheme will give no information about the secret, or, in other words, the revealing of (***t***-1) or fewer shares will not narrow down the search space for ***S***.

2.4 Previous Multi-Secret Sharing Schemes

The most simple way to share multiple secrets is applying the basic Shamir's scheme or other basic (***t***,***n***) secret sharing schemes multiple times. But this will require encryption and decryption for the messages every time, which has certainly high communication and computation overheads. And the other computational workload is also relatively high for the resource constrained devices in WSNs or IoT. In this section, we briefly introduce some representative multi-secret sharing schemes.

2.4.1 Adjustment Multi-Secret Sharing Scheme

Herranz et al. [62] proposed their own scheme based on the previous works [24] [25]. The basic idea is using the original shares as the input of a one-way hash function. The outputs will be shifted to a new set of shares which could be used to reconstruct a new secret.

Setup Phase:

Select a set of participants $\{P_1, P_2, ..., P_n\}$, each participant P_j is assigned a value j, which means value j is assigned to P_j exclusively. And then select positive integers $1 \le t_1 < t_2 < \cdots < t_l \le n$, and $t_1, t_2, \ldots, t_l$ are thresholds for each secret. There is a one-way hash function $H : \mathbb{N} \times \mathbb{Z}_p^* \to \mathbb{Z}_p$, here p is a chosen prime number and $p > n$.

Distribution Phase:

Assume a dealer D wants to share the secrets, then D chooses random values $Sh_j \in \mathbb{Z}_p^*$ as corresponding secret shares for each participant, and they are pair-wise different for $j = 1, 2, ..., n$.

Then D chooses random polynomials $f_i(x) \in \mathbb{Z}_p[x]$ of degree at most $t_i - 1$, for $i = 1, 2, ..., l$, such that $f_i(0) = s_i$.

Both D and participants compute $h_{ij} = H(i, sh_j)$, whereas only dealer D computes $r_{ij} = f_i(j) - h_{ij} \bmod p$. Each participant p_j will compute h_{ij} by itself and secretly receive r_{ij} from D when generating the secret share for *i-th* secret, and then the share will be $f_i(j)$.

Reconstruction Phase:

We assume it was also the dealer D who wants to reconstruct the secret. Each participant already got $f_i(j)$, and they secretly send $f_i(j)$ to D, and D uses Lagrange polynomial interpolation formula to compute the secret s_i.

Here, the basic idea is using a one-way hash function and secret values to make the shares reusable. However, we found that the shift value r_{ij} could be public, so it can eliminate the trouble of encrypting/decrypting part of the message. But, still, the communication cost is very high, since we need to transmit the shift values to every participant every time when we want to share a new secret.

2.4.2 Matrix Based Multi-Secret Sharing Scheme

The introduction of matrix gives another way to achieve multi-secret sharing based on Lagrange polynomial interpolation. In 2012, Waseda, A. and Soshi, M. [63] proposed their scheme which is considered as a (t_i, n) multi-secret sharing scheme since the threshold will reduce along with the reveal of the secrets.

Setup and Distribution Phase:

- for secret S_i, generate shares $\{s_{i_1},...,s_{i_n}\}$, here $i=1,2,...,h$.
- Randomly choose one-way functions $f_i(x)$ and corresponding vectors $R_i=(r_{i,1},r_{i,2},...,r_{1,t_i})^T$.

Randomly choose $n\times t_1$ matrix M_1 on Z_q, where q is a large prime number. This M_1 should satisfies the following relationship:

$$M_1R_1=\begin{pmatrix} m_{1,1} & \cdots & m_{1,t_1} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,t_1} \end{pmatrix}\begin{pmatrix} r_{1,1} \\ r_{1,2} \\ \vdots \\ r_{1,t_1} \end{pmatrix}=\begin{pmatrix} f_1(s_{1_1}) \\ \vdots \\ f_1(s_{1_n}) \end{pmatrix} \tag{2.4.1}$$

- Do the same operation for $i=2,3,...,h$, there will be a $n\times t_i$ matrix M_i:

$$M_i=\begin{pmatrix} & m_{1,(t_{i-1}+1)} & \cdots & m_{1,t_i} \\ M_{i-1} & \vdots & \ddots & \vdots \\ & m_{n,(t_{i-1}+1)} & \cdots & m_{n,t_i} \end{pmatrix} \tag{2.4.2}$$

Where $m_{j,k}\in Z_q$, such that the rank of any $t_i\times t_i$ sub-matrix of matrix M_i is t_i, and then M_i should also satisfies the following relationship:

$$M_1R_1=\begin{pmatrix} & m_{1,(t_{i-1}+1)} & \cdots & m_{1,t_i} \\ M_{i-1} & \vdots & \ddots & \vdots \\ & m_{n,(t_{i-1}+1)} & \cdots & m_{n,t_i} \end{pmatrix}\begin{pmatrix} r_{i,1} \\ r_{i,2} \\ \vdots \\ r_{i,t_i} \end{pmatrix}=\begin{pmatrix} f_i(s_{i_1}) \\ \vdots \\ f_i(s_{i_n}) \end{pmatrix} \tag{2.4.3}$$

- Then compute public parameter: $C=\{c_{1,1},...,c_{1,m_{i_1}},c_{2,1},...,c_{h,m_h}\}$

$$c_{j,l}=\sum_{i=1}^{m_j}k_{j,i}g^{(\sum_{p=1}^{i-1}m_p+l-1)(i-1)}+\sum_{i=1}^{t_j}r_{j,i}g^{(\sum_{p=1}^{j-1}m_p+l-1)(m_j+i-1)} \tag{2.4.4}$$

Where m_i indicates the number of the secrets, g is a primitive element in Z_q, $j=1,2,...,h$ and $l=1,2,...,m_h$.

- Distribute s_{i_j} to corresponding participant P_j as the share through a secret channel, and public M_h, $c_{j,l}$, f_j and g.

Reconstruction Phase:

- When t_i or more shares in $\{s_{i_1},...,s_{i_n}\}$ have been collected, get the inverse matrix $(M_i')^{-1}$ of $t_i \times t_i$ sub-matrix M_i' of matrix M_h, and then recover $R_i = (r_{i,1}, r_{i,2},..., r_{1,t_i})^T$, so that $c_{i,1},...,c_{i,m_i}$ have m_i unknown symbols, the secrets $k_{i,1},...,k_{i,t_i}$ are reconstructed by solving $c_{i,1},...,c_{i,m_i}$.

This scheme actually reconstructs multiple secrets through one reconstruction operation.

2.4.3 (***n***,***t***,***n***) Secret Sharing Schemes

In this section, we first introduce the (***n***,***t***,***n***) Secret Sharing Scheme [28] and the Verifiable (***n***,***t***,***n***) Multi-Secret Sharing Scheme [29]. Then, we will discuss the security of the Verifiable (***n***,***t***,***n***) Multi-Secret Sharing Scheme and explain the limitation of this multi-secret sharing approach.

2.4.3.1 (***n***,***t***,***n***) Secret Sharing Scheme

Master Secret Generation Phase.

Given a set of participants $\{P_1, P_2,..., P_n\}$, each selects randomly a $(t-1)$ degree sub-polynomial $f_i(x)$ with $i = 1,2,...,n$, thus the corresponding sub-secret is $S_i = f_i(0)$, then the master secret is generated as: $S = \sum_{i=1}^{n} S_i$.

Master Shares Generation Phase.

- Each ***P**i*** computes sub-shares: $s_{i,j} = f_i(x_j) \Rightarrow \{s_{i,1}, s_{i,2},..., s_{i,n}\}$, where x_j could be used as identifications, and $j = 1,2,...,n, i \neq j, x_i \neq x_j$.

- ***P**i*** sends each $s_{i,j}$ to corresponding ***P**j*** secretly by encrypting the message using the shared key between these two participants.

- Each ***P**i*** computes the master shares as $ms_i = \sum_{j=1}^{n} s_{j,i}$.

Master Secret Reconstruction Phase.

Any ***t*** master shares can reconstruct the interpolating polynomial because:

$$ms_i = f_1(x_i) + f_2(x_i) + \cdots + f_n(x_i) = F(x_i) \tag{2.4.5}$$

$$F(x) = f_1(x) + f_2(x) + \cdots + f_n(x) \tag{2.4.6}$$

And according to Eq.(2.3.2) and Eq.(2.3.3) the master secret can be obtained as: $S = F(0) = \sum_{i=1}^{n} S_i$.

2.4.3.2 (***n***,***t***,***n***) Multi-Secret Sharing Scheme

Above is the original (***n***,***t***,***n***) secret sharing scheme, the modifications did by the (***n***,***t***,***n***) multi-secret sharing scheme are as follows.

- All the participants work together to select a set of ***n***-tuple weight vectors, denoted as $e_i = (e_{i,1}, e_{i,2}, ..., e_{i,n})$, for $i = 1, 2, ..., n-t+1$. All e_i are linearly independent so that there would be $(n-t+1)$ different master secrets shared among the participants, the l-th master secret will be:

$$M_l = \sum_{i=1}^{n} e_{l,i} S_i \tag{2.4.7}$$

- In order to generate such master secrets, while remain the other steps the same as (***n***,***t***,***n***) secret sharing scheme, the master shares will be generated as:

$$ms_{i,l} = \sum_{j=1}^{n} e_{l,j} s_{j,i} \tag{2.4.8}$$

- So the final polynomial will be:

$$F_l(x) = \sum_{i=1}^{n} e_{l,i} f_i(x) \tag{2.4.9}$$

- Then compute the master secret by $M_l = F_l(0)$.

2.4.3.3 Discussion on Proposed (***n***,***t***,***n***) Multi-Secret Sharing Scheme

As was indicated above, the multi-secret sharing is simply achieved by multiplying the sub-shares by a ***n***-tuple weight vector.

Here it is necessary to understand that the number of the secrets shared by one set of shares is limited. The ***n***-tuple weight vectors are publicly known to every participant. Each participant only knows the sub-shares generated for itself, and participant P_j could only receive: $ms_{i,l} = \sum_{j=1}^{n} e_{l,j} s_{j,i}$

This final reconstructed secret itself will leak no information to the attackers. However, the master shares could be calculated if the attacker collected enough reconstructed secrets. The final polynomial is represented as $F_l(x) = \sum_{i=1}^{n} e_{l,i} f_i(x)$.

After reconstructing ***n*** secrets, a polynomial system could be revealed:

$$\begin{aligned} &F_1(x) = e_{1,1} f_1(x) + e_{1,2} f_2(x) + \cdots e_{1,n} f_n(x) \\ &F_2(x) = e_{2,1} f_1(x) + e_{2,2} f_2(x) + \cdots e_{2,n} f_n(x) \\ &\vdots \\ &F_n(x) = e_{n,1} f_1(x) + e_{n,2} f_2(x) + \cdots e_{n,n} f_n(x) \end{aligned} \tag{2.4.10}$$

If the reconstruction is done by an outside attacker, since all the $F_i(x)$ and $e_{i,j}$, where $1 \le i \le l, 1 \le j \le n$, are known, this attacker can solve the polynomial system and get all the $f_j(x)$. Next time, it could calculate the master secret by itself using all the $f_j(x)$ and the public ***n***-tuple weight vector.

But above is the best case such an attacker could work, since this is an (***n***,***t***,***n***) threshold multi-secret sharing scheme, we needs to ensure that the scheme could still work when there are no more than (***t***-1) colluders. Here, assuming that $p_1, p_2, ..., p_{t-1}$ are colluders, so $f_1(x), f_2(x), ..., f_{t-1}(x)$ are known, deleting the known elements, it is easy to get the following polynomial system:

$$\begin{aligned} &F_1'(x) = e_{1,t} f_t(x) + e_{1,t+1} f_{t+1}(x) + \cdots e_{1,n} f_n(x) \\ &F_2'(x) = e_{2,t} f_t(x) + e_{2,t+1} f_{t+1}(x) + \cdots e_{2,n} f_n(x) \\ &\vdots \\ &F_l'(x) = e_{n,t} f_t(x) + e_{n,t+1} f_{t+1}(x) + \cdots e_{l,n} f_n(x) \end{aligned} \tag{2.4.11}$$

There are only $(n-t+1)$ unknowns, so the system can be solved if there are at least $(n-t+1)$ equations. In other words, using this scheme, at most $(n-t+1)$ secrets can be shared using one set of the shares.

As a result, this (***n***,***t***,***n***) multi-secret sharing scheme needs to update the master shares frequently, and every time a new master secret is about to be shared, the dealer needs to select and broadcast a ***n***-tuple weight vector.

2.4.4 CRT-Based Multi-Secret Sharing Scheme [37]

Assuming the total number of users is *n*, and the threshold is defined as *t*, the dealer selects *n* *r*-tuple weight binary vectors to form a vectors set SET, and $v_1, v_2, ..., v_k$ are any sub set of SET, here *k* indicates the number of vectors in set v_k. SET should satisfy the following requirement: if and only if $k \geq t$, performing OR operation (In the original paper [37], the authors claim to perform XOR operation, we make this correction here) on all the vectors in v_k and get a new binary vector v, the Humming weight of v is *r*. A vector set satisfying the above requirement is called a sharing matrix. Dealer selects co-prime positive integers $m_1, m_2, ..., m_r$ and sends them to the users based on the previously selected sharing matrix. The *j*th row of the sharing matrix is corresponding to the *j*th user, and if we note the entries of the selected sharing matrix as s_{ij}, then if s_{ij} is 1, dealer sends m_i to user U_j. In this way, each user could only get part of the secret and any t of them could work together to collect all the required the m_i s.

Above is the way proposed to achieve (*t*,*n*) threshold property, and a method of reusing the keys is also proposed. Dealer selects co-prime positive integers m_{ij} with $1 \leq i \leq t$ and $1 \leq j \leq n$. Users are divided into different groups, the *i*th member in the *j*th group is denoted as U_{ij}. m_{ij} is sent to

the corresponding U_{ij} secretly. Then the key for *j*th group is calculated as $m_j = \prod_{i=1}^{t} m_{ij}$. Dealer then selects temporary keys q_{ij} with $m_{ij} \geq q_{ij}$, then calculates temporary group key $q_j = \prod_{j=1}^{t} q_{ij}$. After establishing all the required keys, dealer selects secret data a_j $(a_1, a_2, ..., a_n)$ from $\mathbb{Z}_{q_j}$ for each corresponding group. Dealer computes $a_{ij} \equiv a_j (\mathrm{mod}\, q_{ij})$, and uses the calculated a_{ij} to compute x using CRT, where $a_{ij} \equiv x (\mathrm{mod}\, m_{ij})$. Dealer broadcasts x to all the users, and the users could compute a_{ij} accordingly. Since m_{ij} is secretly kept by the users, a_{ij} is actually sent to the corresponding user secretly. And all the members in the same group could reconstruct a_j using CRT, where $a_{ij} \equiv a_j (\mathrm{mod}\, q_{ij})$.

So, after reconstructed once, the dealer only needs to reselect a new set of data and then recalculate a corresponding *x*. The expose of q_{ij} will not affect the security of the scheme since m_{ij} is secretly kept.

The proposed CRT based multi-secret sharing scheme is a combination of the previous two methods. Detailed scheme is briefly introduced as follows.

Set-up Phase:

Dealer selects multiple sharing matrices A_i with size (n_i, t_i) and order $n_i \times r_i$, here $1 \leq i \leq z$, r_i is some positive integer.

Dealer then selects a set of pair-wise co-prime keys m_{ij}, with $1 \leq i \leq z$ and $1 \leq j \leq r_i$. So for each group G_i, the keys are m_{ij} with $1 \leq j \leq r_i$. For each group, those keys are distributed to the group members using the corresponding sharing matrix A_i as explained above.

Secret Sharing:

Dealer selects a secret S_i for each group G_i with $1 \leq S_i \leq M_i$, where $M_i = \prod_{j=1}^{r_i} m_{ij}$.

Dealer uses CRT to compute x which satisfies congruence $x \equiv S_i (\mathrm{mod}\, M_i)$. Then dealer broadcasts x to all the users. In this way, each group could reconstruct the secret using the following functions.

$$s_{ij} \equiv x (\mathrm{mod}\, m_{ij}) \tag{2.4.12}$$

Using s_{ij}, any t_i members of group G_i can reconstruct S_i following CRT. But, S_i could be calculated using an extremely simple way. Let the member which is reconstructing the secret to collect all the m_{ij} from any t_i members of group G_i, then compute $M_i = \prod_{j=1}^{r_i} m_{ij}$, using broadcasted x we could simply do $S_i \equiv x (\mathrm{mod}\, M_i)$. There is no need for CRT in the final reconstruction.

However, as we could see here, the proposed CRT based multi-secret sharing scheme is actually a multi-group secret sharing scheme. When the dealer wants to share multiple secrets with the same group of members, all the secret keys need to be updated since those keys were exposed to the members reconstructing the last secret.

With a little modification, we improve the proposed multi-group secret sharing scheme into a multi-group multi-secret sharing scheme by simply utilizing the two methods proposed in [37]. The detailed scheme is as follows.

Dealer.

- selects multiple sharing matrices A_i with size (n_i, t_i) and order $n_i \times r_i$, here $1 \le i \le z$, r_i is some positive integer.

- Selects co-prime numbers m_{ij} as private keys and co-prime numbers q_{ij} as public keys, with $m_{ij} \geq q_{ij}$, $1 \leq i \leq z$ and $1 \leq j \leq r_i$. m_{ij} are distributed to the group members using the corresponding sharing matrix A_i while q_{ij} are set public.
- Computes group private keys: $M_i = \prod_{j=1}^{r_i} m_{ij}$.
- Selects secrets for each group, denoted as S_i with $1 \leq S_i \leq M_i$.
- Calculates $s_{ij} \equiv S_i (\mathrm{mod}\, q_{ij})$, using CRT to calculate S_i' from $S_i' \equiv s_{ij} (\mathrm{mod}\, m_{ij})$. Then uses CRT to calculate *x* from $x \equiv S_i' (\mathrm{mod}\, M_i)$.
- Broadcasts *x* to all the members.

Members in the i-th group.

- Receive *x* and calculate:

$$s_{ij} \equiv x (\mathrm{mod}\, m_{ij}) \tag{2.4.13}$$

  Since only at least t_i members of group G_i possess all the private keys in the *i*th group, all the members calculate Equation (2.4.13) and any t_i members of group G_i can provide all the s_{ij}.
- Calculate S_i from $s_{ij} \equiv S_i (\mathrm{mod}\, q_{ij})$ using CRT.

  Since m_{ij} are always kept secret, they can be reused as private keys. When sharing multiple secrets, the dealer just needs to re-calculate the broadcasted integer *x*.

This modified new scheme is a (*t*,*n*) threshold multi-group multi-secret sharing scheme since the secrets shared for each group are independent from each other and the secrets shared for the same group in different session are also independent from each other. But, as we can see from the scheme description, the dealer needs to perform heavy computational operations, such as the utilization of

CRT. And all the members need to remember two sets of the keys, one is private key set and the other public key set. These two big drawbacks diminish the efficiency and the feasibility of this scheme, despite that it is already improved compared to the schemes given in [37].

2.5 Previous Key Management Schemes in WSNs

Based on [64], the key management schemes can be classified into three categories, self-enforcing schemes, arbitrated keying schemes and pre-distribution schemes, based on their different approaches.

The self-enforcing schemes are using asymmetric cryptography so they can enable the devices to generate or establish keys after deployment. The performances of these schemes mostly depend on the performances of the existing asymmetric algorithms. As we explained in chapter 2, public key encryption techniques always involve expensive operations, which make the asymmetric encryption algorithms not feasible for highly resource constrained devices in the WSNs.

Arbitrated keying schemes rely on a trustworthy centralized key management center. All the keys are generated and managed by this central point which makes it the most valuable target for the attackers. The risk of single point failure cannot be ignored. Additionally, if the scale of the WSN is too large, the central point may not be able to efficiently manage all the devices.

The last category is pre-distributed scheme. Schemes in this category use keys which are distributed to all the devices before deployment. Prior to the deployment, a key distribution center generates and loads keys to the sensor nodes for further secure communications. The key generation and distribution could be done in various ways. This kind of key management scheme

lacks flexibility but avoids the overhead of key generation processes for sensor nodes and has little or no risk of single point failure.

Later in Chapter 5, we propose our multi-secret sharing based key management scheme which is a hybrid approach with some features of both arbitrated keying schemes and pre-distributed schemes. Next, we will focus on the review of schemes in these two categories considering the assumption of the existence of highly resource constrained devices in WSNs. And we also focus on the schemes using symmetric cryptography because of the relatively low cost than asymmetric cryptography. Interested reader could find more on key management schemes in [8] [64] [65] [66].

2.5.1 Network-wide Key

The basic idea of using network-wide key is loading all the devices in the same WSN with the same universal key, any device outside the network cannot decrypt the messages encrypted using this key. Many later schemes are modifications of this basic idea, such as the one adopted by the Zigbee standard [67]. This scheme is called Symmetric-key key establishment (SKKE), and in this scheme, all the devices are pre-loaded a master key ***K***. When two devices nodes A and B want to communicate with each other, they separately generate and exchange random numbers N_A and N_B . The secret shared by these two sensor nodes is computed as $S_{A,B} = PRF(K \| ID_A \| ID_B \| N_A \| N_B)$, here $PRF()$ stands for pseudo-random function. In order to avoid the possible errors during the computation and communication, both devices will use the shared secret to further compute two more keys, $K_{A,B} = Hash(S_{A,B} \| 1)$ and $K'_{A,B} = Hash(S_{A,B} \| 2)$, here $Hash()$ stands for one-way hash functions. They both compute the following tags.

$Tag_A = PRF(K'_{A,B} \| 1 \| S_{A,B})$ sent from A to B.

$Tag_B = PRF(K'_{A,B} \| 2 \| S_{A,B})$ sent from B to A.

Using those two tags, the two devices can make sure that the other side successfully gets the correct secret key $K_{A,B}$, so they can then use $K_{A,B}$ as the symmetric key to enable the secure communication link.

2.5.2 Pair-wise Key

Another approach for pre-distributed key management is pre-load pair-wised keys to all the sensor nodes in the network. The basic idea behind this approach is loading a lot of keys into sensors to make sure that every two nodes share one unique secret key. In this case, assuming there are ***n*** sensor nodes in the network, each node needs to possess (***n***-1) different keys, which means the total number of keys in the network is $n(n-1)/2$.

As we can see, the above approach requires the sensor nodes have relatively large memory for storing keys and lack of flexibility. This approach is called full pair-wise key management scheme. There is another pair-wise key approach which is called probabilistic pair-wise key management.

In probabilistic approach, the network has a key pool and each sensor nodes selects some of the keys from this key pool based on probability or controlled by a center point to form their own key chains. Since they obtain their keys from the same key pool, the different key chains might have same key or keys. All the keys have their own unique ID for distinguishing purpose. When two sensor nodes want to communicate, they firstly inform each other the keys in their key chains by posting the IDs of the keys. If they possess same key or keys, they can then use the shared key or keys to encrypt messages or generate a new key. If all the keys they possess are different, they need to discover a path by using other sensor nodes as the intermediate hop or hops. The intermediate

node or nodes share keys with both end sensor nodes, but the communication security could only achieved hop-by-hop instead of end-to-end.

However, since all the sensor nodes obtain their keys from the same key pool, some keys might be possessed by more than just two sensors. So the shared keys cannot be used for private communications between two nodes.

In order to achieve better performances, this approach is improved and modified into multiple schemes, such as hashed random key pre-distribution scheme, key redistribution scheme and pair-wise keys establishing scheme. In the next sub-section, we will review a key establishing scheme exploits the concept of threshold secret sharing.

2.5.3 Pair-wise Key Establishing scheme with Threshold [68]

This scheme or protocol is designed for two nodes to establish a pair-wised shared key without requiring the help from an on-line key distribution center. It is a combined design of two different techniques, probabilistic key management and (***t,n***) threshold secret sharing. Those two techniques are all mentioned in the previous discussion, and in this scheme, the authors chose to use Shamir's secret sharing technique over the CRT based Asmuth-Bloom secret sharing.

Also assuming there are n sensor nodes in the system and nodes A and B among them are starting a secure communication session. Before the deployment of the sensors, each sensor nodes are loaded with l keys randomly selected from the key pool which contains keys with a total number of m, which means any key has a possibility of l / m to be assigned to a sensor node. Since each key has a unique ID, A and B can inform each other the keys they possess without revealing the keys themselves. Assuming the keys shared by both A and B form a key set $R_{A,B}$, the number of the keys in this set is denoted as $\left|R_{A,B}\right|$ with $\left|R_{A,B}\right| \geq 0$.

If $|R_{A,B}| = 0$, A and B do not share any secret key, they cannot directly communicate with each other privately, in this case, they need to find multiple paths between them by using the other neighbor sensor nodes as intermediate hops. Denotes node X is one of the intermediate nodes, and if X shares some keys with both A and B, this X called one-hop proxy since on the path of A-X-B, B is only one hop away from A. The path through X is called indirect path. If there are two or more nodes are needed to form the path between A and B, this kind of path is usually not considered to be used. If $|R_{A,B}| > 0$, A and B do share some key or keys, so A and B can form direct path or paths for further communication purpose. And direct path is always preferred to indirect paths. All the direct paths between A and B belong to class C1, indirect paths through one hop belong to class C2, and all the other indirect paths using multiple hops belong to class C3.

Using some algorithm to determine the number of shares based on the information of C1, C2 and C3, one path is only used to transmit one share. Assuming the result of the algorithm is *n*, those shares are generated using simple XOR operations as following equation:

$$s_1 \oplus s_2 \oplus \cdots \oplus s_n = S \qquad (2.5.1)$$

The final secret can be recovered only when all the *n* shares are received correctly by the receiver (the node initiates the communication session is the sender and the node responses is the receiver, here B is the receiver).

The transmission of one share is different on direct path and indirect path. On direct path, all the keys in $R_{A,B}$ are XORed into a new key $K_{A,B}$. The share is then encrypted using $K_{A,B}$ and sent directly from A to B. On indirect path, assuming one hop node is X, key set shared by A and X is $R_{A,X}$ and key set shared by X and B is $R_{X,B}$. All the keys in $R_{A,X}$ XORed into key $K_{A,X}$ and keys in $R_{X,B}$ XORed into key $K_{X,B}$. The share is first encrypted by $K_{A,X}$ and sent to X, then X

decrypts the cipher-text and re-encrypts the share using $K_{X,B}$, B receives the cipher-text and then gets the share by decryption. X here is acting like a translator between A and B.

This scheme provides a method of exchanging new secret between two nodes using pre-distributed keys. However, the disadvantages of this scheme are also significant. The first disadvantage is that one compromised path will lead to the compromise of multiple paths since the final secret could be recovered only when all the shares are received correctly. Additionally, the secret transmitted is not a secret to all the intermediate nodes, which means this scheme only could achieve hop-by-hop security instead of end-to-end security. And the communication cost is also expensive since this scheme involves multiple communication paths with multiple intermediate hops.

2.5.4 Polynomial Based Key Pre-distribution [69]

The pre-distributed keys could not only be randomly generated binary strings with fix length, but also could be polynomials. In [69], the authors provided a new method of establishing a shared key between two devices using the pre-distributed polynomials. This scheme has two different phases: setup and key establishment.

In the setup phase, a trustworthy server is used to generate and assign keys to the devices. The server randomly selects a bivariate *t*-degree polynomial $f(x,y)=\sum_{i,j=0}^{t} a_{i,j}x^i y^j$ so that $f(x,y)=f(y,x)$. Each device in the network has a unique ID, assuming node *i* and *j* are two nodes which want to initiate a private conversation. The server computes $f(i,y)$ for node *i* and computes $f(j,y)$ for node *j*.

In the key establishment phase, two nodes inform each other their IDs, node *i* computes $f(i, j)$ and node *j* computes $f(j,i)$. Since $f(x, y) = f(y, x)$, $f(i, j) = f(j,i)$. This $f(j,i)$ is used as the shared key between the two nodes.

Since $f(x, y)$ is kept only to the server, $f(j,i)$ could only be computed by node *i* and *j*, no information will be leaked when there is no more than *t* compromised nodes in the network. And we also need to notice that *t* is the total number of the nodes in the network. However, the storage cost of this scheme is exponential in terms of the value of ***t***.

2.5.5 Key Management Based on Secret Sharing [70]

In the above discussion, we mostly focus on the review of key management schemes for WSNs in a distributed manner. In a distributed WSN, two nodes communicate directly with each other, however, in some applications, such as smart metering, multiple nodes need to report to a more powerful device, and they together form a cluster with this relatively powerful device acting as the cluster head. The direct communication between two sensor nodes is not required. This hieratical architecture has three types of participants, base station, cluster heads and sensor nodes. For hieratical WSNs, a new category of key management schemes is required for communications within a cluster and between cluster heads. Here, we only review the one based on secret sharing technique.

This secret sharing based key management scheme [70] provides two types of keys: cluster-in keys and cluster-between keys. And this scheme is also considered as a key pre-distribution scheme since there must be a secure channel between each sensor node and the cluster head, and this secure channel also exists between the cluster heads and the base station. That means, each sensor node

needs to remember a shared key with its cluster head, and all the keys are unique. It is the same for cluster heads and the base station.

The scheme is briefly described below.

Initialization Phase.

Assuming there are m cluster heads in the WSN, base station randomly selects *m* (*t*-1) polynomials and each for one cluster head. One of the polynomial is as Eq. (2.5.2)

$$f_{C_i}^{l}(x) = s_{C_i}^{l} + a_{i,1}^{l}x + \cdots + a_{i,t-1}^{l}x^{t-1} \tag{2.5.2}$$

Here, $i = 1,2,...,m$ and l stands for the number of session period. Then, the base station adds all polynomials up to form a new polynomial with (*t*-1) degree for key distribution for cluster-between keys. This new polynomial is shown in Eq.(2.5.3)

$$f_{C_{in}}^{l}(x) = \sum_{i=1}^{m} f_{C_i}^{l}(x) = S_{C_{in}}^{l} + a_{in,1}^{l}x + \cdots a_{in,t-1}^{l}x^{t-1} \tag{2.5.3}$$

Hide the selected keys $K_{C_{in}}^{l}$ and $k_{C_i}^{l}$ with the following equations.

$$z_{C_i}^{l} = k_{C_i}^{l} + s_{C_i}^{l} \tag{2.5.4}$$

$$Z_{C_{in}}^{l} = K_{C_{in}}^{l} + S_{C_{in}}^{l} \tag{2.5.5}$$

Cluster-Between Key Establishment.

Base station sends $(ID_{CH_i}^{l}, f_{C_{in}}^{l}(ID_{CH_i}^{l}))$ to corresponding cluster head through the pre-established secure channel. Here $ID_{CH_i}^{l}$ is the ID for the *i*-th cluster head.

Base station then broadcast $a_{in,j}^{l} \bullet g \bmod p$ and $Z_{C_{in}}^{l}$ to all the cluster heads, $a_{in,j}^{l}$ is the coefficient of polynomial $f_{C_{in}}^{l}(x)$ with $j = 0,1,2,\ldots,t-1$, and because of the difficulty of computing $a_{in,j}^{l}$ from $a_{in,j}^{l} \bullet g \bmod p$, $f_{C_{in}}^{l}(x)$ is still a secret. g is a base point in a publicly selected elliptic curve, and p is a large prime number.

Assuming one of the cluster heads wants to reconstruct the secret, it needs to collect $(ID^l_{CH_i}, f^l_{C_{in}}(ID^l_{CH_i}))$ from t cluster heads including itself. Before the reconstruction, using the broadcasted coefficients $a^l_{in,j} \bullet g \bmod p$ to form a new polynomial $f^l_{C_{in}}(x) \bullet g \bmod p$. Let $x = ID^l_{CH_i}$, the results will be $f^l_{C_{in}}(ID^l_{CH_i}) \bullet g \bmod p$. And using collected information $(ID^l_{CH_i}, f^l_{C_{in}}(ID^l_{CH_i}))$, computes $f^l_{C_{in}}(ID^l_{CH_i}) \bullet g \bmod p$ to see if all the results matches. If they do, all the collected information is legitimate and could be used to reconstruct the final secret. If not, restart the whole process from the beginning.

The reconstruction is done follow Eq.(2.3.3), and the result is $S^l_{C_{in}}$, further computes $K^l_{C_{in}}$ through $K^l_{C_{in}} = Z^l_{C_{in}} - S^l_{C_{in}}$.

Cluster-in Key Establishment.

The cluster-in key establishment is similar to cluster-between keys, and the difference here is the cluster head of the *i*-th cluster acting similar to base station and the sensor nodes are similar to cluster heads. We do not need to give a redundant review on this part.

The best feature of this scheme is that it has threshold property and a relatively simple way of verifying the collected shares. But in each communication session, a new set of polynomials needs to be selected and distributed to the participants. Additionally, the sensor nodes are restricted to the cluster, which means they can only report to their superior or communicate within the small group.

2.6 Previous Key Agreement Schemes in IoT

Key agreement protocols, or referred to as key establishment protocols, are designed for two parties to generate a shared secret key ***S***. This shared secret key is used to solve a fundamental

problem in cryptography, which is ensuring the secure communication between two parties over an insecure transmission channel [71].

One approach for implementing key agreement protocols is deploying a public key infrastructure (PKI) system [72]. However, such a system requires private or public key exchange, large memory space for key directories, services provided by a third party, and expensive exponential computations.

In 1985, the first identity based cryptosystem were proposed by Adi Shamir [73]. He suggested that, instead of randomly generating two keys (public key and private key) as it does in PKI, the users could use their own names and network addresses as their public keys. The corresponding private key is generated by a trusted centralized key generation center. This idea has been exploited since then and many identity-based key agreement protocols have been proposed. However, Shamir did not actually give identity-based cryptosystems but a way of implementing identity-based signature schemes.

HIP protocol is using the same idea. The host identifier is globally unique and could be chosen as the public key of an asymmetric encryption key pair. Compare to the legacy Internet, which has two namespaces, Domain Name Service (DNS) and Internet Protocol (IP) addresses, HIP separates the Host Identity and the location identity, and uses IP addresses as the locator while using host identifiers to represent the new namespace. In this way, user who has a legitimate host ID could be considered as a legitimate user. Key exchange algorithms used in HIP are using host identifier to authenticate the users and then using different ways to exchange keys with authenticated users.

Following, we will briefly introduce few ways of achieving key agreement as well as HIP related key exchange mechanisms.

2.6.1 Host Identity Protocol [32]

IP addresses are dual used as “locaters” (internetworking routing vector) and “identifiers” (end-points or host identifiers), and in order to allow a host to be simultaneously reachable through different interfaces, there is the need of separating the two functions of the IP addresses. HIP provides the alternative to IP address, the basic architecture is shown in Figure 8 [74]. In the Internet, a mobile device might change its location and move into another access point, a new IP address will be assigned dynamically, but the Internet could only recognize the new IP address not the device itself. HIP enables the hosts to achieve independent end-to-end operations. This feature is extremely useful for devices to communicate directly under the context of Internet of Things.

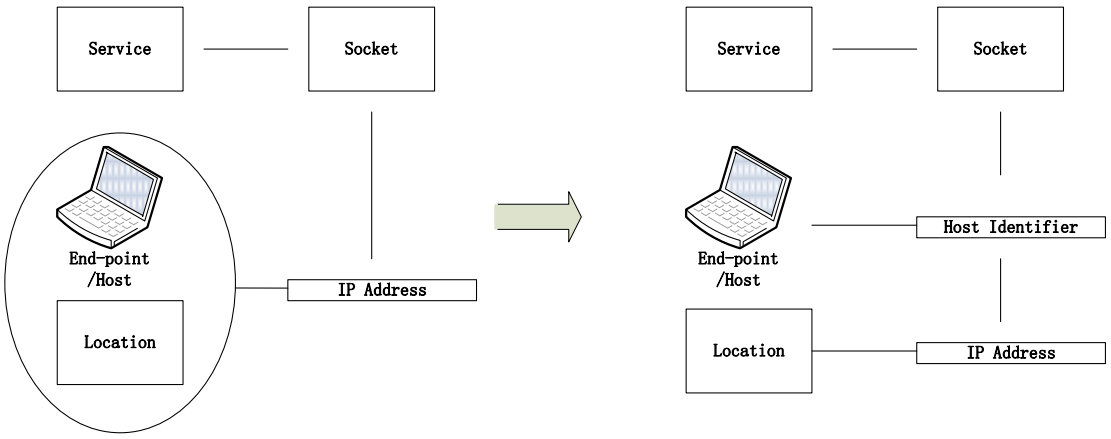

Figure 8 Architectures Comparison between IP Address and HIP

Host Identifier (HI) is a globally unique name in this namespace [17]. And one device may have multiple Host Identities but no HI could be shared by two different devices. A HI is the public key of a pair of asymmetric keys, and according to our previous discussion, the public key could be used in two different ways, the first is encryption and the other is signing digital signatures. Here, HI is used for authentication purpose to ensure the successful recognition of a device/host.

HIP is now typically using RSA and Digital Signature Algorithm (DSA) together with SHA-1 hash. Performing SHA-1 hash on HI could result a 160-bit static globally unique Host Identity Tag (HIT) which is similar to a domain name in the Internet. RSA and DSA are used to generate Host Identifiers and authenticate each other.

After two devices authenticated each other, the private communication is required. However, the public key encryption is resource consuming, so HIP provides HIP based key exchange to allow both sides to agree on a secret key for later symmetric encryption operations.

2.6.2 HIP Base Exchange (HIP-BEX)

This sub-protocol is used to achieve authenticated key agreement between two HIP peers who have legitimate host identifiers. Figure 9 HIP Base Exchange (HIP-BEX) shows the entire key exchange process.

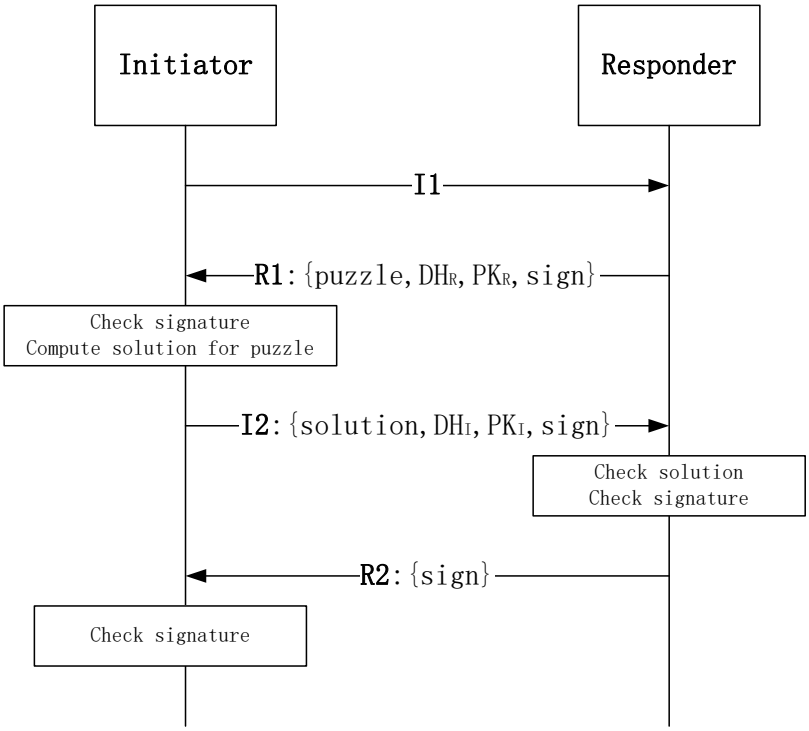

Figure 9 HIP Base Exchange (HIP-BEX)

As we can see from Figure 9, HIP-BEX is a four-way handshake protocol between two hosts. Initiator who wants to initiate a communication session sends I1 packet to the responder, and this packet is unencrypted and unsigned, it just contains the information to inform the initiation of a new session. This packet is routed to the responder's IP address which is derived from the DNS.

Upon the reception of I1, the responder will send R1 back to initiator if the responder wants to start a communication session with the initiator. The R1 packet contains one puzzle challenge which could help avoid DoS attacks, and it also contains the Diffie-Hellman public value of responder, the public key (Host identifier) and the signature signed by its private key.

After received the R1 packet, initiator will first check the signature, and then compute the solution for the puzzle. The solution will be send back to the responder together with the initiator's Diffie-Hellman public value, public key (initiator's host identity) and the signature signed by the initiator's private key in I2 packet.

The responder will then receive the I2 packet and check the solution to see if it matches the puzzle, after that, the signature will also be checked to make sure the initiator is the legitimate host. Then the responder will send another packet, R2, to the initiator to finalize the exchange.

After finished all the steps above, the Diffie-Hellman public values of both hosts exchanged during the handshake procedure will be used in the Diffie-Hellman key exchange. As we indicated above, the two hosts could share the same secret key after the Diffie-Hellman key exchange, and this key could be used as the private key in the following communication between two hosts.

There are heavy cryptographic computations on both initiator and responder. The computation of the Diffie-Hellman key, the signature generation and the computation of solution for the puzzle

consumed most of the resources. There are two modifications of HIP have been proposed ease this situation to make HIP more suitable for resource unbalanced environment. In the following two sub-sections, we give a brief introduction to these two modified schemes.

2.6.2 HIP Diet Exchange (HIP-DEX) [33]

HIP-BEX, there are two pairs of asymmetric keys, one pair is the Diffie-Hellman keys, and the other pair is the assigned public/private key pair where the public key is also the host identifier. In HIP-DEX, there is only one pair of asymmetric keys, the Diffie-Hellman keys. Hosts use long term Elliptic Curve Diffie-Hellman (ECDH) public value as their host identifiers.

The knowledge of the Diffie-Hellman key will be enough for a host to be considered as legitimate. And the initiator will use responder's public key to encrypt a selected secret value $\boldsymbol{x}$ and later the responder will use the initiator's public key to encrypted another selected secret value $\boldsymbol{y}$. In this way, both responder and initiator could confirm that $\boldsymbol{x}$ and $\boldsymbol{y}$ comes from each other and both values are kept secret. Those two random values are eventually used as seeds to derive the final secret.

2.6.3 Lightweight HIP (LHIP) [34]

In LHIP, there is no Diffie-Hellman key is computed, no RSA operation is performed, and there is even no secure IPsec tunnel is set up after the key exchange. It uses hash chains to achieve the similar goal by binding successive messages. It sacrificed certain degree of security since the hash chain only meets the minimal security requirement. But, in order to allow better security and offer flexible services, the LHIP could be upgraded to HIP-BEX in certain circumstances. So, as a result of the sacrifice, LHIP only guarantees the ongoing session is not hijacked without strong host

authentication services provided, further, the HIP messages are also unprotected because of the lack of shared secret key.

So we can get the conclusion that the two modifications we listed above either not efficient enough or sacrificed too much secrecy for efficiency. HIP Tiny exchange (HIP-TEX) [12] and threshold distributed key exchange for HIP (we referred to as TD-HIP) [36] is motivated by this situation and designed based on two considerations: replace Diffie-Hellman key exchange with lighter public key infrastructure (PKI) and introduce a new collaborative approach to share the heavy workload of the constrained host.

2.6.4 Threshold Distributed Key Exchange for HIP (TD-HIP) [36]

TD-HIP introduces new participants, proxy, into the system. Proxy is a relatively powerful device/host which have enough resources to perform heavy computation and cryptography operations. Some messages send between initiator and responder will be routed hop-by-hop through the proxy. This can be achieved by applying secret sharing schemes.

In TD-HIP, there are three entities involved, the first one is the initiator which is assumed to be resource constrained, the second is the responder which can be any less resource constrained device in the network, the last one is a group of proxy which help the initiator to avoid heavy cryptographic computation.

TD-HIP is designed for the global Internet of Things infrastructure which interconnects different devices with different capacities.

Detailed TD-HIP scheme is shown in Figure 10 and described following.

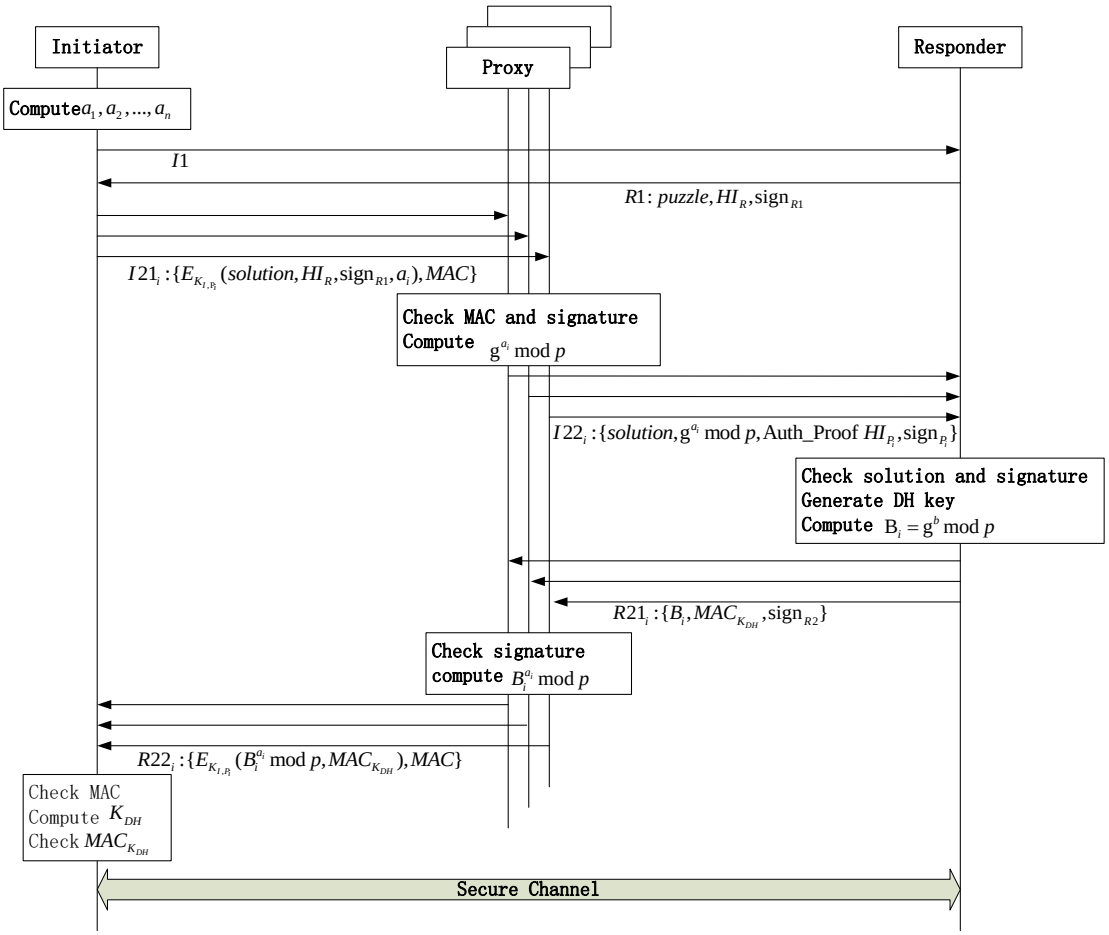

Figure 10 TD-HIP key Exchange

- ***Message I1***: initiator sends a message to the responder to initiate a communication session. Therefore, initiator uses I1 to inform the responder about its supported cryptographic algorithms and the cooperative key establishment techniques, and there is no need to encrypt I1.
- ***Message R1***: the responder receives Message ***I1***, if it agrees to establish the secure communication session. The responder will reply the initiator with Message ***R1***. ***R1*** contains the public key K_R of the responder and the ***puzzle*** generated for the initiator to solve.
- ***Message I21$_i$***: After received ***R1***, the initiator and the responder agrees to establish a secure communication session. The initiator will send Message ***I21$_i$*** to proxy P_i for help. Message ***I21$_i$***

contains the ***solution*** for received ***puzzle***, the responder's public key K_R, one secret share x_i of the final secret X, and a Message Authentication code (MAC) for secret share x_i. All the information should be private, so they are encrypted by the pre-shared secret key between the initiator and the proxy P_i.

- ***Message I22$_i$***: The proxy P_i will decrypt the received message, get ***solution***, K_R, and secret share x_i. Then it will communicate with the responder on behalf of the initiator. Message $I22_i$ contains solution for the ***puzzle***, proxy P_i's public key LK_i, signature signed by P_i's private key and secret share x_i encrypted by the responder's public key K_R.
- ***Message R21$_i$***: The responder will first check the ***solution*** to see if it matches the ***puzzle***, if matches, then check the signature. After all the checking procedures have been done, the responder derives the secret X from all the received secret shares, and then generates secret value Y. The Message $R21_i$ then contains Y which is encrypted by X, a MAC for X to acknowledge that the responder received the correct X, and a signature signed by the responder's private key.
- ***Message R22$_i$***: After proxy P_i finishes checking the signatures, it encrypts the received message with the pre-shared secret key, and then sends the ciphertext to the initiator as Message $R22_i$.
- ***Key Derivation***: After decryption, the initiator and the responder will obtain the same key materials: X, Y, ***puzzle***. They use the same one-way hash function to compute the final secret key K_{RI}:
- $K_{RI} = H(puzzle \mid X \mid Y)$

The final secret key can be used to establish the secure communication channel and will be expired when current communication session has ended. So the next time, when two hosts want to communicate again, they will have to repeat the whole process to get a new secret key.

In this scheme, the initiator outsources all the heavy exponential computation to the proxy. The initiator only performs the secret distribution to provide the responder the required key material for further session key generation.

$$a_i \equiv f(x_i) (\mathrm{mod}\, p) \tag{2.6.1}$$

$$c_i = \prod_{j=1, j\neq i}^{n} \frac{-x_j}{x_i - x_j} \tag{2.6.2}$$

$$K_{DH} = \prod_{i=1}^{t} g^{b \cdot f(x_i) \cdot c_i} \bmod p = g^{ab} \bmod p \tag{2.6.3}$$

Since the most energy consuming operations involved in HIP-BEX are public cryptographic operations, mainly the exponential calculation, TD-HIP improves the energy efficiency for the resource-constrained initiator.

TD-HIP is the most efficient and lightweight existing key exchange scheme for HIP comparing with previous HIP-BEX, LHIP, and D-HIP [35]. However, despite TD-HIP has already avoid a lot of heavy asymmetric encryption operation, it also introduces more communication and symmetric encryption workload. The newly introduced workload is smaller than the workload when performing asymmetric encryption/decryption, but it is still not negligible compare to the extremely constrained resources. And if the session key expires frequently, the resource consumption for the initiator will be more significant. So we need to find a way to reduce the interactions between the initiator and the proxy as well as reduce the symmetric encryption/decryption operation.

Chapter 3

Decentralized CRT-Based (*n*,*t*,*n*) Multi-Secret Sharing Scheme

3.1 Assumptions

We assume that the extremely resource constrained device is dealer ***D***, so the workload for the dealer should be as small as possible, or, in the best case, we even can outsource most of the work to the proxy to establish a decentralized secret sharing scheme. While outsourcing the workload to the proxy, we also need to keep in mind that the proxy is relatively more resourceful but still resource constrained devices, only the reader ***R*** which wants to initialize a communication session with ***D*** is powerful.

3.2 Preliminary and Architecture

As shown in Figure 11 Communication Architecture of (*n*,*t*,*n*) Threshold Secret Sharing Scheme, the traditional architecture only involves two entities with a secure channel established between them. However, in our proposed scheme, there are three kinds of entities involved: dealer, proxy, and reader. Instead of the direct communication between reader and dealer, the proxy working as the intermediate layer to help both side to exchange information.

We denote the dealer as ***D***, and each proxy as $\boldsymbol{P_i}$, where $i = 1, 2, ..., n$. And all the proxies form a set of $\boldsymbol{P_i}$: $\{P_1, P_2, ..., P_n\}$. There is also a reader R_r outside the local network who come and want to

initiate the ***r***-th communication session and get the secret S_r, here r is the session number, $r = 1, 2, ..., l$ and $S_1 \neq S_2 \neq ... \neq S_l$.

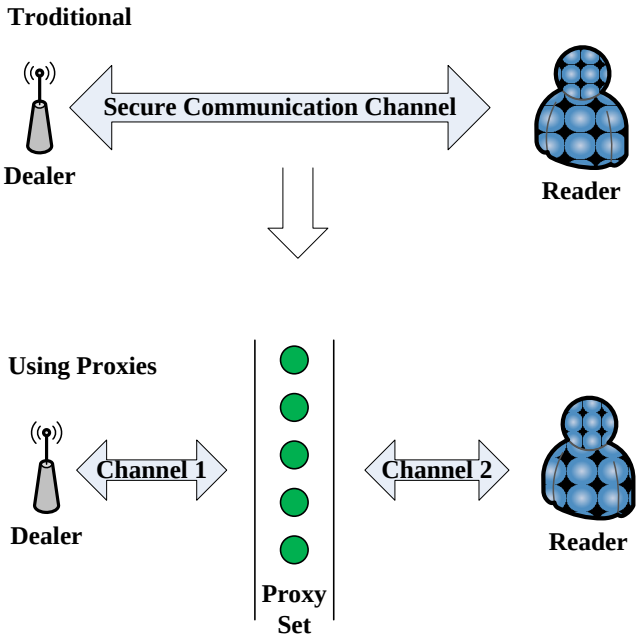

Figure 11 Communication Architecture of (*n,t,n*) Threshold Secret Sharing Scheme

As depicted in Figure 11, there are two kinds of secure channels. The first one is secure channel between the proxy and the dealer. Dealer has a secure channel with every proxy, and every pair of proxy shares a unique secure channel. This secure channel is achieved by symmetric encryption, here we choose AES-256. The shared secret key is given before deployment by a trusted third party or the manufacturer itself, every two participants share a unique secret private key, so they can use that key to encrypt their communication. In this way, participants can communicate with each other without revealing any information to others.

The second one is secure channel between the proxy and the reader. This kind of secure channel is achieved by asymmetric cryptography, such as RSA and Elliptic Curve algorithm, since the proxy and readers do not share private keys. In our assumption, the readers and proxy are relatively

more powerful than the dealer, they can afford the costs of the heavy asymmetric encryption/decryption.

We list all the involved notations and their corresponding descriptions in, this notation will be used in the rest of this chapter.

Table 1 Notation in CRT-Based (*n*,*t*,*n*) Multi-Secret Sharing Scheme

Notation	Description
D	Dealer, highly resource constrained device
P_i / P_j	The *i*-th/*j*-th proxy, less resource constrained device, $i, j = 1,2,3,...,n, i \neq j$
R_r	The *r*-th reader, powerful device outside the local network, *r*=1,2,3,.... Here *r* could also indicate the current session is the *r*-th session.
t	The threshold of the scheme, the minimum number of shares required to retrieve the shared secret
n	The total number of involved proxy
m_0	Large positive integer selected to set up the upper bound of the key space
m_i	Large randomly selected positive integers, *i*=1,2,3,...,*n*. $m_0, m_1, m_2, ..., m_n$ are all pair-wise co-prime, and $m_0 < m_1 < m_2 < \cdots < m_n$, they also fulfill the requirement: $m_0\left(\prod_{i=n-t+2}^{n} m_i\right) < \prod_{i=1}^{t} m_i$
S	Master secret, S is an integer and $0 < S < m_0$
S'	Pseudo master secret, $S' = S + Am_0$, where A is a randomly selected positive integer, and $0 < S' < M$, $M = \prod_{i=1}^{t} m_i$
SH_i	Secret shares of the master secret S , $SH_i \equiv S' (\mathrm{mod}\, m_i)$
S_r	Secret generated and reconstructed in the *r*-th communication session
$H(x_1, x_2)$	One-way hash function. x_1 and x_2 stand for the two different inputs of the hash function

$h_{i,r}$	Pseudo-random number generated by P_i in the *r*-th session using one-way hash function, $h_{i,r} = H(r, SH_i)$.
$Sh_{i,r}^j$	Sub-share generated by P_i in the *r*-th session, and will be sent to P_j
Sh_r^j	Reconstruction share generated by P_j for secret S_r
N_y	Number of possible secrets
ShV_r^j	P_j 's verification share
P_x	Cheater, or corrupted proxy, $x = 1, 2, 3, ..., n.$
V	Verifier, trusted entity used to verify shares and identify the cheaters

3.3 Description

Dealer Centralized Distribution Phase.

As we could see in Figure 12, which shows the centralized distribution phase, only dealer and all the proxy are involved in this phase.

For Dealer D:

- ***D*** randomly selects (***n***+1) positive pair-wise co-prime integers: $m_0, m_1, m_2, ..., m_n$, they follow Asmuth-Bloom's scheme to define the boundaries: $m_0 \left(\prod_{i=n-t+2}^{n} m_i \right) < \prod_{i=1}^{t} m_i$.
- ***D*** then selects the master secret S. Here $0 < S < m_0$, so if the m_0 is large enough, the key space will be sufficient to achieve required computational security. After ***S*** is selected, ***D*** computes: $S' = S + Am_0$, where A is a randomly selected positive integer, and $0 < S' < M$, $M = \prod_{i=1}^{t} m_i$.
- ***D*** computes shares for all the proxy: $SH_i \equiv S' \bmod m_i$.

So (m_i, SH_i) is the actual share which will be sent to the corresponding proxy $\boldsymbol{P_i}$, and all the $m_0, m_1, m_2, ..., m_n$ will be public to all members.

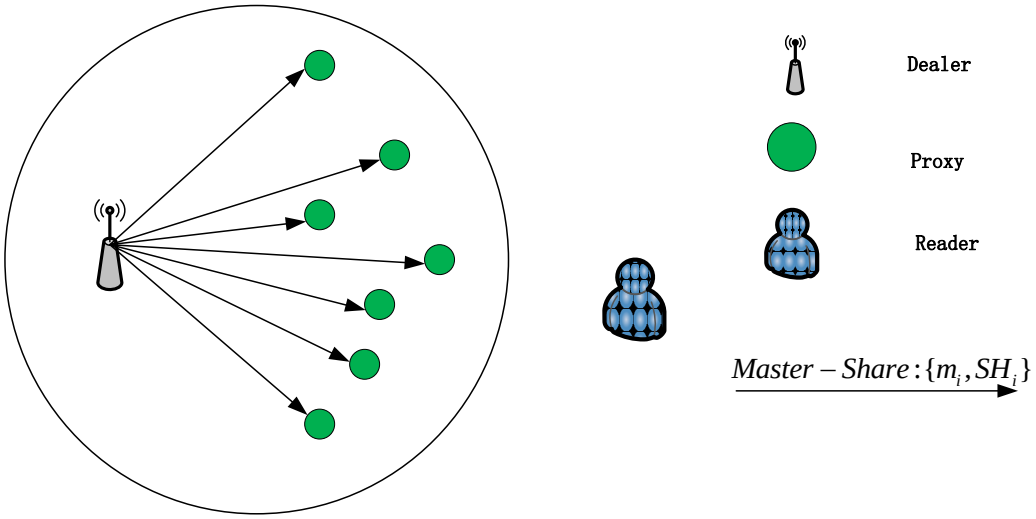

Figure 12 Communication Flow During Dealer Centralized Distribution Phase

Proxy Sub-Shares Distribution Phase.

In this phase, proxy become senders and receivers at the same time, and when a proxy is under the Sender mode, its behavior will be similar to the dealer. And the real dealer and the reader will not participate in the process. As it is shown in Figure 13, the dealer and the reader are both waiting, and all the proxy start to distribute the generated sub-shares. We use one proxy as an example of sender state proxy and show this proxy in square in order to distinguish it from others in Figure 13. In later discussion, in order distinguish two proxy states, we use $\boldsymbol{P_i}$ to represent the proxy in sender mode, and use $\boldsymbol{P_j}$ to represent the proxy in receiver mode.

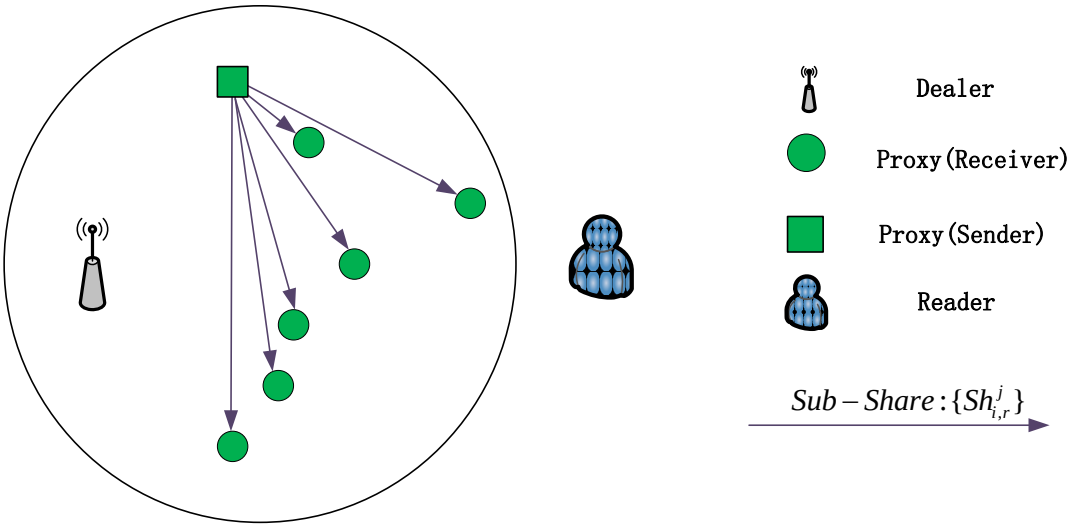

Figure 13 Proxy Pi Distribute Sub-shares to Other Proxy

For Proxy P_i:

- When R_r wants to reconstruct S_r, $\boldsymbol{P_i}$ uses a two variable one-way hash function $H(x_1, x_2)$ to generate a random number: $H(r, SH_i) = h_{i,r}$.
- $\boldsymbol{P_i}$ computes sub-shares $Sh_{i,r}^{j} \equiv SH_i \cdot h_{i,r} (\mathrm{mod}\, m_j)$, then distributes each sub-share to corresponding proxy P_j ($j = 1, 2, ..., n, j \neq i$).

For Proxy P_j:

- P_j will obtain ***n*** sub-shares from both other proxy and itself: $\{Sh_{1,r}^{j}, Sh_{2,r}^{j}, ..., Sh_{1,r}^{j}\}$.
- P_j then adds all sub-shares together: $Sh_r^j \equiv \left(\sum_{i=1}^{n} Sh_{i,r}^{j} \right) (\mathrm{mod}\, m_j)$, we call Sh_r^j a reconstruction share of secret S_r.

Final Reconstruction Phase.

After receiving all the sub-shares and generating the final reconstruction share, the reader will first request reconstruction shares from ***t*** or more proxies. The dealer will locally perform the same one-way hash function operations as the proxies did, and compute the final secret without the help

from the proxies. This method requires more memory space while reducing the communication and computation costs.

Figure 14 shows the final reconstruction phase. Proxy could provide reconstruction shares to both the dealer and the reader, however, the dealer may choose not to get the final secret by using reconstruction shares.

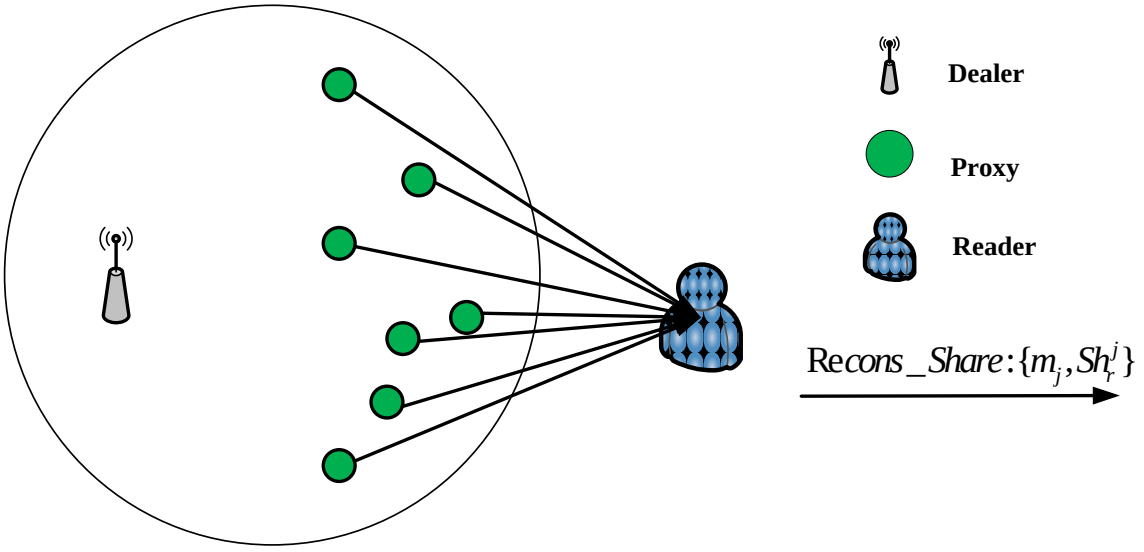

Figure 14 Dealer and Reader Collecting Reconstruction Shares from Proxy

For Receiver or Reader R_r :

- R_r selects a group of proxy: $\{P_1, P_2, ..., P_t\}$. Each proxy will send its own reconstruction share to R_r through the secret channel, so that R_r could get: $\{Sh_r^1, Sh_r^2, ..., Sh_r^t\}$ and $\{m_1, m_2, ..., m_t\}$.
- Let $M_r = \sum_{i=1}^{t} m_i$, $M_{i,r} = M_r / m_i$, $y_{i,r} = M_{i,r}^{-1} (\mathrm{mod}\, m_i)$. Then, based on CRT, R_r could compute the unique solution:

$$S_r' \equiv (\sum_{i=1}^{t} Sh_i^r \cdot M_{i,r} \cdot y_{i,r}) (\mathrm{mod}\, M_r) \tag{3.3.1}$$

$$S_r \equiv S_r' (\mathrm{mod}\, m_0) \tag{3.3.2}$$

Note: $S_r' \equiv (\sum_{i=1}^{n} SH_i \cdot h_{i,r}) (\mathrm{mod}\, M)$, since $h_{i,r}$ will change every time, so S_r will be different accordingly.

3.4 Two Methods of Generating Verifiable Shares

Although the algorithm itself appears to be secure, as demonstrated above, it is still important to consider the possibility that not all the received shares are correct. Some errors may occur during the transmission, or the proxy may intentionally send false shares to hamper the operation of the system.

We assume that the majority of the proxy are trustworthy, and the shares could be verified through the test of (***t***,***n***)-consistency. In order to do the test, we need to collect shares and perform the reconstruction operation to see if the reconstruction result of every set of ***t*** shares is the same. But this also means the reconstruction shares and the final secret are both exposed.

Here in this sub-section, we present two methods to generate verifiable shares without revealing any information about the final secret.

3.4.1 (*t*,*n*) Consistency Test

The (***t***,***n***)-consistency test could be done by reconstructing the secret using all the possible combinations. That means the process for the reconstruction of the secret needs to be done C_t^n times, which is not efficient for large ***t*** and ***n***.

However, in [75], Harn and Lin proposed testing the (***t***,***n***)-consistency by reconstructing the secret only once. This approach is based on Shamir's secret sharing scheme, and if the interpolation of the collected ***n*** points (x_1, s_1), (x_2, s_2),..., (x_n, s_n) yields a polynomial with

degree (***t***-1), then all the shares passed the (***t***,***n***)-consistency test and there was no cheater in the system.

This is achieved using all the ***n*** points to do the following calculation:

$$S_n = f(0) = \sum_{j=1}^{n} f(j)\left(\prod_{r=1, r\neq j}^{n} \frac{-r}{j-r}\right) (\mathrm{mod}\ p) \qquad (3.4.1)$$

If $S_n = S$, then all the collected shares are legitimate, and they all passed the (***t***,***n***)-consistency test.

In our scheme, the same approach can be applied as well. The (***t***,***n***)-consistency test is done by collecting all the shares received from the proxies and use them to reconstruct the original master secret. And in order to avoid exposing the real secret and shares to possible attackers, we proposed two ways of generating verifiable shares for the (***t***,***n***)-consistency test only.

3.4.2 Uniform Vectors for Generating Verifiable Shares

The first method of generating verifiable shares is using uniform vectors. This method was proposed by Liu, Harn, Yang and Zhang in 2012 [29].

They proposed to ask the dealer to broadcast two ***n***-tuple weight vectors at the same time. One of them is used to generate the reconstruction shares, the other is used to form the verifiable shares. However, since the publicly exposed ***n***-tuple weight vectors will reveal some information about the secret as we illustrated in Section 3, the dealer needs to more frequently updates the privately kept polynomials.

Here, we simply ask the proxy to locally generate two sets of random numbers by performing the hash function, then following the similar steps in the reconstruction share generation. Reader

receives two sets of shares, one is used for secret reconstruction purpose, the other one is for verification.

3.4.3 Using CRT for Generating Verifiable Shares

The second method exploits a special property of CRT, and helps the verifier to get reconstruction shares and verifiable shares at the same time without asking the dealer to broadcast an ***n***-tuple weight vector as done in the first method.

As we mentioned before, in the proxy sub-distribution phase, proxy P_j receives the following information: $\{Sh_{1,r}^j, Sh_{2,r}^j, ..., Sh_{n,r}^j\}$. The verifiable shares are simply generated by using the following operation: $ShV_r^j \equiv (\prod Sh_{i,r}^j) (\mathrm{mod}\, m_j)$.

If all the verification shares have the (***t***,***n***)-consistency property, then the reconstruction shares reveal the correct secret.

In our scheme, we choose the second method because it is more efficient and simple to be exploited.

Here, we could also use the verification shares as the reconstruction shares while using the reconstruction ones to do the verification. How to decide which one to be used as reconstruction shares? That is according to the actual application requirements. Since the multiplication operation requires more computational resources and more time, and the verification does not need to happen frequently, we shall use $ShV_r^j \equiv (\prod_{i=1}^{n} Sh_{i,r}^j) (\mathrm{mod}\, m_j)$ as the verification share.

3.4.4 Cheater Detection in (*n*,*t*,*n*) Multi-Secret Sharing

Cheater could be a compromised legitimate node or an attacker disguised itself as a legitimate node. The main purpose of a cheater is launching attacks from the inside and those attacks have different kinds.

The passive attacks are intended to eavesdrop and steel the shared secret. In our distributed approach, every proxy only possesses part of the secret. A successful attack requires at least t cheaters to collude with each other, which is difficult to achieve. As long as the number of cheaters is up to $t-1$, the risk of leaking the secret can be ignored since $t-1$ shares cannot construct the secret as we discussed in this section.

The active attacks are trying to degrade the quality of service by modifying the messages or simply denying the request of information. If the attackers are so powerful that they can make the honest proxy to obliviously collude with the fellow cheaters, in the worst case, there is no way to detect nor identify the cheaters when $n = 2t-1$ and up to $t-1$ proxy are compromised [76].

We highlight our assumption again, which is that the majority of the proxy are honest and trustworthy. Because of the similarities shared by the Lagrange Polynomial Interpolation and the Chinese Remainder Theorem, in our scheme, we test the (***t***,***n***)-consistency of all the collected reconstruction shares as follows. We use all the received reconstruction shares to reconstruct the final secret:

$$Sn_r' \equiv (\sum_{i=1}^{n} Sh_i^r \cdot M_{i,r} \cdot y_{i,r}) \pmod{M_r} \tag{3.4.2}$$

Here $M_r = \sum_{i=1}^{n} m_i$, $M_{i,r} = M_r / m_i$, $y_{i,r} = M_{i,r}^{-1} (\mathrm{mod}\, m_i)$. If $S_r' = Sn_r'$, all the reconstruction shares are legitimate.

However, since the CRT operation involves exponential computations, we propose to reconstruct the secret by randomly selecting ***t*** shares from all the collected shares pool first. Then,

instead of using all the shares to do the reconstruction again, we use the reconstructed secret to re-generate a copy for the unselected reconstruction shares:

$Sh_r^{t+1\prime} \equiv S_r' (\mathrm{mod}\, m_{t+1}), \cdots, \quad Sh_r^{n\prime} \equiv S_r' (\mathrm{mod}\, m_n)$.

If all the calculated reconstruction shares equal to the collected corresponding ones, then all the shares are legitimate; otherwise, there are one or more cheaters among the majority-honest proxy.

3.5 Performance Analysis

3.5.1 Computational Cost Analysis

There are few types of resource consuming process: primality test, secret reconstruction, communication and encryption/decryption. As we could learn from the details of the proposed scheme, the heavy computation does not frequently happen on the dealer side.

The primality test is one of the operation that takes a long time, especially when the device is not powerful. However, we only need one set of well tested prime numbers, so in a long time, this type of computation could only happen once on the dealer or even do not happen at all.

The exponential operation is another resource consuming process. It will be introduced when calculating the multiply inverse during the secret reconstruction phase. Since the dealer possesses all the required information, so the new secret could be reconstructed locally, which means the exponential operation will only be executed by the resourceful reader.

And since the master shares are reusable, so once the master shares are distributed, the dealer has no need to communicate with proxy which omits the frequent communication as well as the heavy encryption operations.

From the evaluation above, we can see that more operations have been introduced to the proxies and the reader, and the overall workload of the entire system has also been increased. However, since we are focusing on offloading workload from the resource constrained dealer, the tradeoff is considered acceptable as the workload of the dealer has been significantly reduced.

3.5.2 Security Performance Analysis

The security analysis on Asmuth-Bloom's scheme has been done by [10], which indicates that as long as there are no more than (***t***-1) shares (s_i, m_i) revealed, the secret ***S*** is secure. Here, we reused part of the shares, so it is necessary to discuss the possible influence on the revealing of all the m_i, for $i = 1, 2, 3, ...n$.

We give the example in the worst case using actual numbers as follows.

Assume: $x \equiv 1 (\mathrm{mod}\, 3)$, $x \equiv 2 (\mathrm{mod}\, 5)$. In the range of (0, 15], we have the smallest ***x***, 7, and in the range of (15, 30], $x = 22$, in the range of (30, 45], $x = 37$, so within every 15 numbers, we could find one ***x*** that fulfill the requirements. The same thing will happen when there are (***t***-1) shares are revealed. In the worst case, the biggest (***t***-1) shares are revealed, and within every $\prod_{i=n-t+2}^{n} m_i$ numbers, there will be one number that could be the possible secret. Then, since we have $m_0 \prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$, and the secret $y = S + am_0$, where $S \in_R \mathbb{Z}_{m_0}$, is chosen in the range, we can compute the number of the possible secrets using the following function: $N_y = \left\lfloor y / \prod_{i=n-t+2}^{n} m_i \right\rfloor$

In Mignetto's scheme, since the secret selection range is defined by $\prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$, N_y could be any integer larger than number "1", but in Asmuth-Bloom's scheme, if we assume

$N = \left\lfloor \prod_{i=1}^{t} m_i / \prod_{i=n-t+2}^{n} m_i \right\rfloor$, then N_y could be $N \cdot m_0$. In the worst case, when $N = 1$, the number of possible secrets is still m_0. And, considering all the conclusions we got from above is based on the fact that all the m_i, for $i = 1,2,3,...n$, are known, so as long as m_0 is big enough, the key space will not be narrowed down to an unacceptable level.

And also because both SH_i and $h_{i,r}$ are unknown, it is hard for an adversary to compute SH_i from the received $Sh_{i,r}^{j} \equiv SH_i \cdot h_{i,r} (\mathrm{mod}\, m_j)$.

The above discussion could be generalized to prove that the exposure of all the selected prime numbers $m_0, m_1, m_2, ..., m_n$ does not decrease the security strength of the secret sharing scheme.

In the worst case, assuming $(t-1)$ shares, which correspond to prime numbers $m_{n-t+2}, m_{n-t+3}, ..., m_n$, are known to the attacker, and all the co-prime numbers $m_0, m_1, m_2, ..., m_n$ are publicly exposed. S is the selected secret within the range $(0, m_0)$, and S' is in the range of $\left(0, \prod_{i=1}^{t} m_i\right)$ with $S' \equiv S + am_0 (\mathrm{mod}\, M)$, $M = \prod_{i=1}^{t} m_i$. And the generated shares are $s_i \equiv S' (\mathrm{mod}\, m_i)$.

Since the adversary has only $(t-1)$ shares, together with the corresponding m_i, the adversary could perform CRT reconstruction operation on $(t-1)$ shares to get all the information that $(t-1)$ shares could reveal.

Multiply inverse of m_i is $M_i^{-1} = [(\prod_{i=n-t+2}^{n} m_i) / m_i]^{-1}$, denotes the reconstructed result as X,

$X \equiv (\sum_{i=n-t+2}^{n} M_i \times (M_i^{-1} \bmod m_i) \times s_i) (\mathrm{mod} \prod_{i=n-t+2}^{n} m_i)$.

We could know that $X < \prod_{i=n-t+2}^{n} m_i$, which means X only one possible value of S', and S' cannot be smaller than X. We assume there is an integer Y let $S' = X + Y$, and in order to meet

the requirement that $s_i \equiv S'(\mathrm{mod}\, m_i)$ for $m_{n-t+2}, m_{n-t+3}, ..., m_n$, Y is dividable by $m_{n-t+2}, m_{n-t+3}, ..., m_n$, since they are co-prime to each other, so $Y = b \prod_{i=n-t+2}^{n} m_i$, b is a non-negative integer, thus $S' = X + b \prod_{i=n-t+2}^{n} m_i$.

Next, if we could get the value of b, we can obtain S'. Within the searching space for S', the number of possible values for b is $N = (\prod_{i=1}^{t} m_i - X) / \prod_{i=n-t+2}^{n} m_i$. Because $m_0 \left(\prod_{i=n-t+2}^{n} m_i \right) < \prod_{i=1}^{t} m_i$, $N > (m_0 \prod_{i=n-t+2}^{n} m_i - X) / \prod_{i=n-t+2}^{n} m_i$. And since $X < \prod_{i=n-t+2}^{n} m_i$, we could get the conclusion that at least $N > (m_0 \prod_{i=n-t+2}^{n} m_i - \prod_{i=n-t+2}^{n} m_i) / \prod_{i=n-t+2}^{n} m_i = (m_0 - 1)$.

That means, despites the adversary knows all the prime numbers and $(t-1)$ shares, it still needs to try at least $(m_0 - 1)$ times to find out the real secret while m_0 is a large number.

In 2007, Kamer Kaya and Ali Aydın Selçuk [77] slightly modified the basic Asmuth-Bloom secret sharing scheme in order to achieve better security. They select co-prime numbers $m_0, m_1, m_2, ..., m_n$ so that $m_0^2 \left(\prod_{i=n-t+2}^{n} m_i \right) < \prod_{i=1}^{t} m_i$. This modification is done for better using Asmuth-Bloom secret sharing in function sharing schemes. However, in our scheme, the security strength of the basic Asmuth-Bloom secret sharing scheme is sufficient as we proved above, thus, we created our (***n***,***t***,***n***) multi-secret sharing scheme based on Asmuth-Bloom's scheme.

3.6 Theoretically Calculation and Testing Result

In order to evaluate the performances of our CRT-based multi-secret sharing scheme, we have implemented the involved operations in C (involving the GNU MP Bignum Library) [78] and the cryptographic operations are introduced from OpenSSL.

We tested and performed the whole process for generating and sharing multiple secrets between two entities, the resource constrained Dealer and the powerful Reader, and the sharing and reconstruction are successful for every time. We further evaluate the energy costs for the whole system and the individual devices, and our focus is the cost of the Dealer since the Proxy and the Reader are relatively powerful than the Dealer.

We also implemented a Lagrange Polynomial Interpolation based (***n,t,n***) multi-secret sharing scheme which derives the Shamir's (***t,n***) secret sharing scheme. This scheme is entirely built on the same structure as the CRT based scheme proposed in this thesis. In order to evaluate the performance of the two schemes, we retrieved the CPU time of each scheme's implementation, and according to the Equation (3.6.1) [79], which is the Processor Performance Equation, on the same testing computer, based on the CPU time it takes to execute the program, we can get the total clock cycles the program required to be executed.

$$CPU\_time = \frac{instructions}{program} \times \frac{clock\_cycles}{instruction} \times \frac{\text{sec}onds}{clock\_cycle} \qquad (3.6.1)$$

The bigger the *CPU_time* is, the more instructions have been executed, and also means more clock cycles have been taken for the execution of the testing program. Since the testing is done on powerful processor, the test can only reflect or mimic the performance test on resource

constrained small chips based on the following equation [36] (assuming the target resource constrained device is Telos_B sensor):

$$E_{TelosB} = \frac{U_{TesloB} \bullet I_{TelosB}}{N_{TelosB}} \bullet \frac{\mathrm{Re}\, gister\_size_{processor}}{\mathrm{Re}\, gister\_size_{TelosB}} \bullet \alpha \bullet C_{processor} \qquad (3.6.2)$$

Here, $U_{TelosB} \bullet I_{TelosB}$ shows the working power of TelosB, N_{TelosB} represents the instructions per second of TelosB CPU, and $C_{processor}$ stands for the number of cycles we get from the powerful processor running our implementations of the schemes. α represents the richer instructions of our powerful processor, which is approximately 2.

Since the tests are doing under the same environment, all the other parameters are considered same and stable, except for the number of clock cycles required will be different. Thus, we can use the CPU time to represents the energy cost on the resource constrained device.

For CRT-based secret sharing scheme, the main operation is hash function operation. As we indicated above, Dealer locally computes the hash result and based on the other information it possesses, it can generate the shared secret locally without any further communication or interactions with other involved entities. For Shamir-based secret sharing scheme, the Dealer is also responsible for the secret functions distribution and vectors generation and distribution. Those responsibilities require the Dealer to perform two main operations, local SHA-256 hash and AES-256 encryption.

Assuming the energy costs for SHA-256 and AES-256 encryption are E_{Hash} and E_{Enc}, respectively, cost of other operations such as vector generation and final secret calculation is E_{Other}, since the other operations cost little compare to hash operation and symmetric encryption operation, we assume the other costs for Dealers in both schemes are approximately equal. Based

on the behaviors of two schemes, we can get the following functions for the total energy cost for Dealers in CRT-based scheme and Shamir-based scheme.

Dealer in CRT-based scheme:

$$E_{CRT-Dealer} = n \times E_{Hash} + E_{Other} \tag{3.6.3}$$

Dealer in Shamir-based scheme:

$$E_{Shamir-Dealer} = \frac{1}{n-t+1}(n \times t \times E_{Hash} + n \times t \times E_{Enc}) + E_{Other} \tag{3.6.4}$$

As we can see from the functions, the relation between *n* and *t* is critical to the final performances. For a given system, the number of proxy is relatively fixed, we take the threshold as a variable, if the threshold is too small, the attacker will be able to compromise the system using less efforts which means the security level of the system is lowered, on the other hand, if the threshold is too big, the robust level of the system will be lowered. In the voting scenario, the majority of the involved entities are trusted, so here, we set the threshold *t* at least bigger than *n*/2, which means $t \geq \frac{n}{2} + 1$, or $n \leq 2t - 2$.

When $n = 2t - 2$, for the second function, we get:

$$E_{CRT-Dealer} = (2t-2) \times E_{Hash} + E_{Other} \tag{3.6.5}$$

$$E_{Shamir-Dealer} = \frac{1}{2t-2-t+1}((2t-2) \times t \times E_{Hash} + (2t-2) \times t \times E_{Enc}) + E_{Other} = 2t(E_{Hash} + E_{Enc}) + E_{Other}$$

(3.6.6)

Basically, the other operations can be ignored for now, theoretically, the energy cost difference between two Dealers can be expressed as follows:

$$E_{Diff} = 2 \times E_{Hash} + 2t \times E_{Enc} \tag{3.6.7}$$

When the value of t becomes bigger, the energy cost difference between two Dealers will become greater linearly. We set the parameters in our implementation to verify the theoretical calculations. The testing result is shown in Figure 15 and Figure 16.

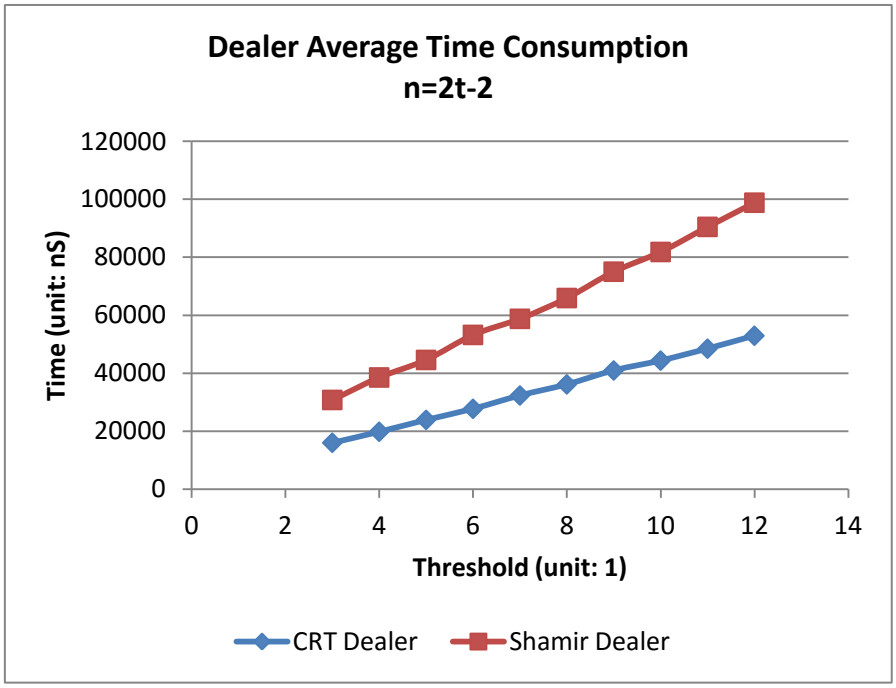

Figure 15 Dealer Average Time Consumption When Set $n = 2t - 2$

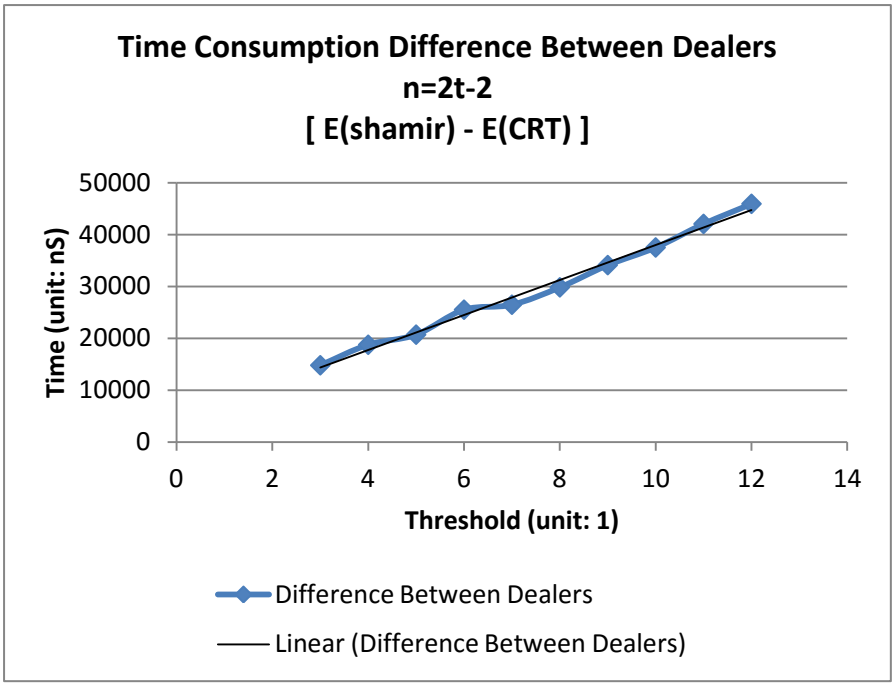

Figure 16 Time Consumption Difference between Dealers When Set $n = 2t - 2$

The performance improving of Dealer is the tradeoff of the increase of energy cost in Proxy, next, we evaluated the payoff on the Proxy' sides. Main operations for Proxy in CRT-based scheme are local SHA-256 hash, sub-share AES-256 encryption (encrypted and send to others) and decryption (received from others and decrypt for further usage), and reconstruction share encryption. Proxy in Shamir-based scheme performs different operations in different stages as the Dealer did, if there are no shared secret functions or those functions have expired, the Proxy performs AES-256 decryption (decrypt encrypted function coefficients received from the Dealer), sub-share AES-256 encryption (encrypted and send to others) and decryption (received from others and decrypt for further usage), and reconstruction share encryption; if the shared functions can be used in current cycle, then the Proxy only perform the reconstruction share encryption. Other operations, such as sub-share generation and reconstruction share generation have been ignored here.

According to the discussion above, we get two functions for any individual Proxy in two schemes respectively.

For Proxy in CRT-based scheme:

$$E_{CRT-\mathrm{Proxy}} = E_{Hash} + (n-1) \times E_{Enc} + (n-1) \times E_{Dec} + E_{Enc} + E_{Other} \tag{3.6.8}$$

For Proxy in Shamir-based scheme:

$$E_{Shamir-\mathrm{Proxy}} = \frac{1}{n-t+1} \times [t \times E_{Dec} + (n-1) \times E_{Enc} + (n-1) \times E_{Dec}] + E_{Enc} + E_{Other} \tag{3.6.9}$$

When $n = 2t - 2$, we can calculate the following:

$$E_{CRT-\mathrm{Proxy}} = E_{Hash} + (2t-2) \times E_{Enc} + (2t-3) \times E_{Dec} + E_{Other} \tag{3.6.10}$$

$$E_{Shamir-Proxy} = \frac{1}{t-1} \times [t \times E_{Dec} + (2t-3) \times (E_{Enc} + E_{Dec})] + E_{Enc} + E_{Other}$$
$$E_{Shamir-Proxy} = 3 \times E_{Dec} + 3 \times E_{Enc} - \frac{1}{t-1} \times E_{Enc} + E_{Other} \quad (3.6.11)$$

Under the assumption of $n = 2t - 2$, we have the difference of energy consumption between two proxies in different schemes:

$$E_{diff} = E_{Shamir-proxy} - E_{CRT-proxy} \quad (3.6.12)$$

$$E_{diff} = (5-2t) \times E_{Enc} + (6-2t) \times E_{Dec} - \frac{1}{t-1} \times E_{Enc} - E_{Hash} \quad (3.6.13)$$

$$E_{diff} = -2 \times (E_{Enc} + E_{Dec}) \times t - \frac{1}{t-1} \times E_{Enc} + (5 \times E_{Enc} + 6 \times E_{Dec} - E_{Hash}) \quad (3.6.14)$$

As we can see, the energy cost of proxy in CRT-scheme is larger than in Shamir-scheme, which is expected to happen. And the increasing trend of the difference in energy cost is similar to the graph shown in Figure 17, which is the graph of function $f(x) = -x - \frac{1}{x}$. Which means, in most of the cases, the difference between two kinds of proxy will increase along with the increase of the value of t, in this case, the value of n in specific.

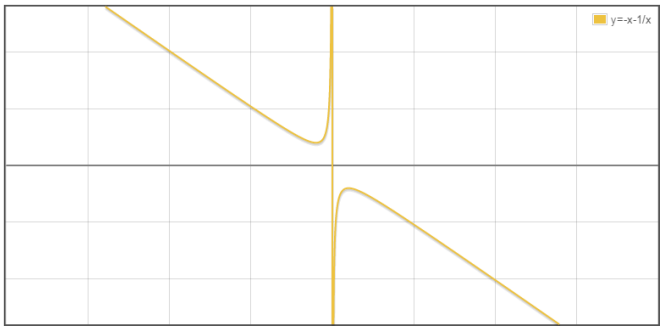

Figure 17 Functional Figure of $f(x) = -x - \frac{1}{x}$

The test result of energy cost of Proxy is shown in Figure 18 and Figure 19.

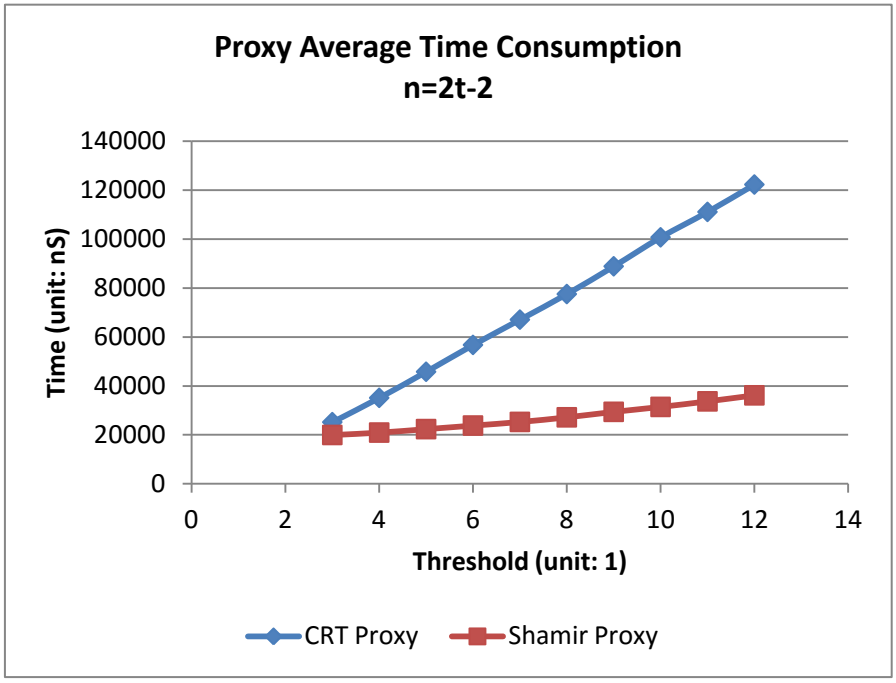

Figure 18 Proxy Average Time Consumption When Set $n = 2t - 2$

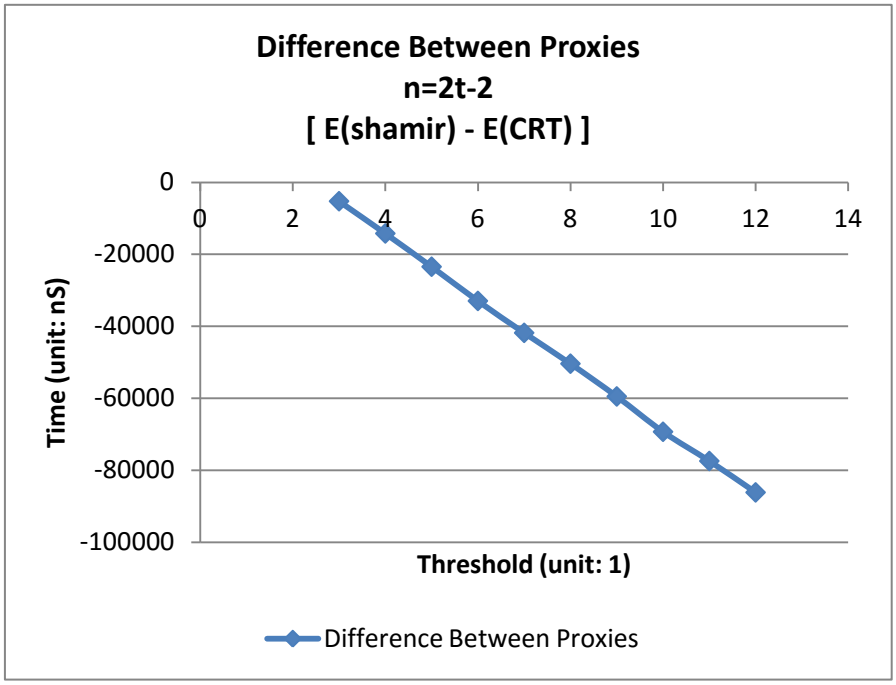

Figure 19 Time Consumption Difference between Proxy When Set $n = 2t - 2$

The third entity involved in both schemes is Reader, which is considered most powerful and less concerned about the energy cost. The differences between CRT and Lagrange polynomial

interpolation are the ways of creating secret shares and reconstructing secrets. The reconstruction of the secret is done by simple arithmetic operations in Lagrange polynomial interpolation. However, since division is involved in the reconstruction process, the rounding might cause the mismatching between reconstructed secret and the actual secret. Thus, we introduce multiplicative inverse calculations to both schemes for the final secret reconstruction in the Reader side, which means the energy cost of Readers in both schemes is similar.

The testing results are shown in Figure 20.

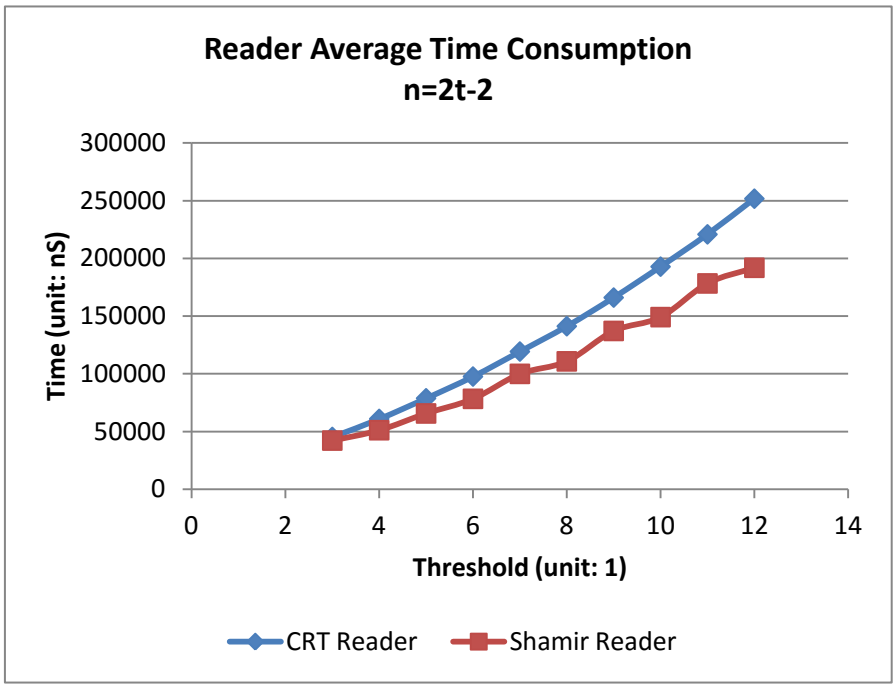

Figure 20 Reader Average Time Consumption When Set $n = 2t - 2$

According to the benchmark of OpenSSL on the testing computer CPU, the clocks required to finish the SHA-256 and AES-256 are approximately equal, we then assume the energy consumption for both algorithms has the relationship of $E_{Hash} \approx E_{Enc}$.

Let two types of Dealers have the same energy consumption, and set $E_{Hash} = E_{Enc}$, through calculation, we have:

$$E_{CRT-Dealer} = E_{Shamir-Dealer}$$

$$n \times E_{Hash} + E_{Other} = \frac{1}{n-t+1}(n \times t \times E_{Hash} + n \times t \times E_{Enc}) + E_{Other} \quad (3.6.15)$$

$$n = \frac{2 \times n \times t}{n-t+1} \Rightarrow n = 3t-1$$

The calculation result indicates that if we set the parameters to $n = 3t-1$, the energy cost of both type of Dealers should be approximately equal. To verify the calculation result, the corresponding testing results are shown in following figures (Figure 21, Figure 22 and Figure 23):

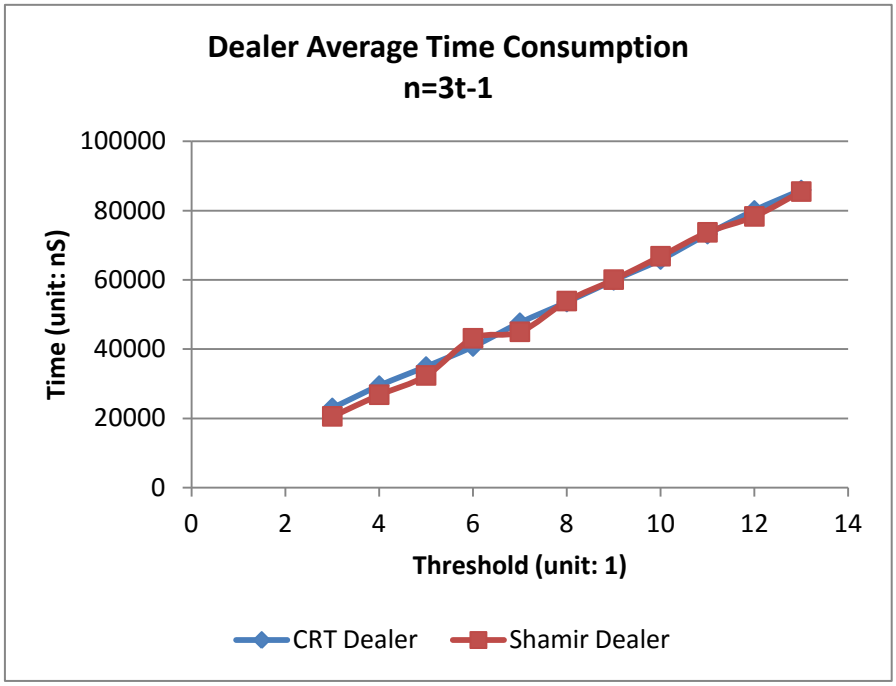

Figure 21 Dealer Average Time Consumption When Set $n = 3t-1$

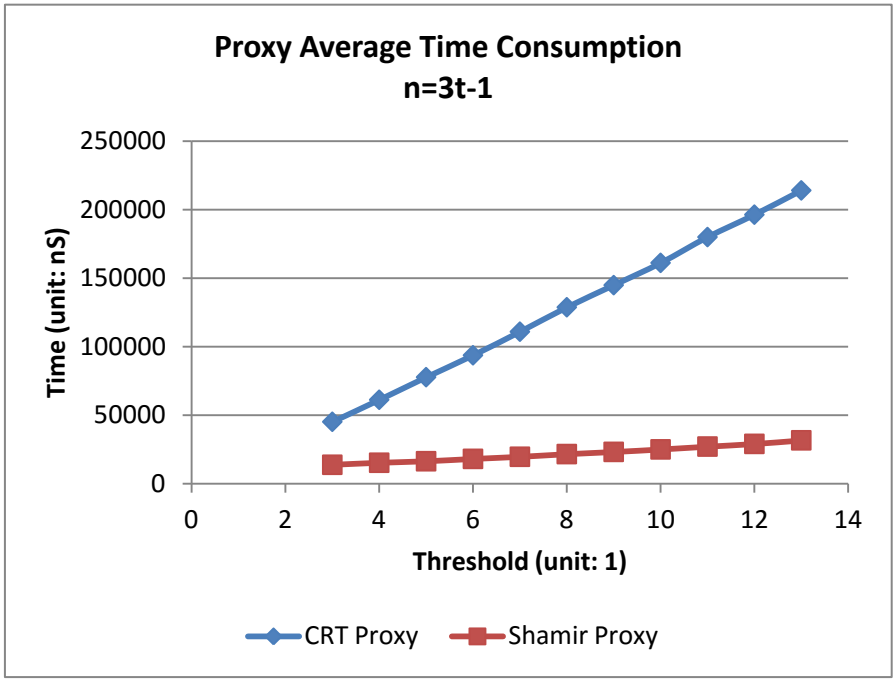

Figure 22 Proxy Average Time Consumption When Set $n = 3t - 1$

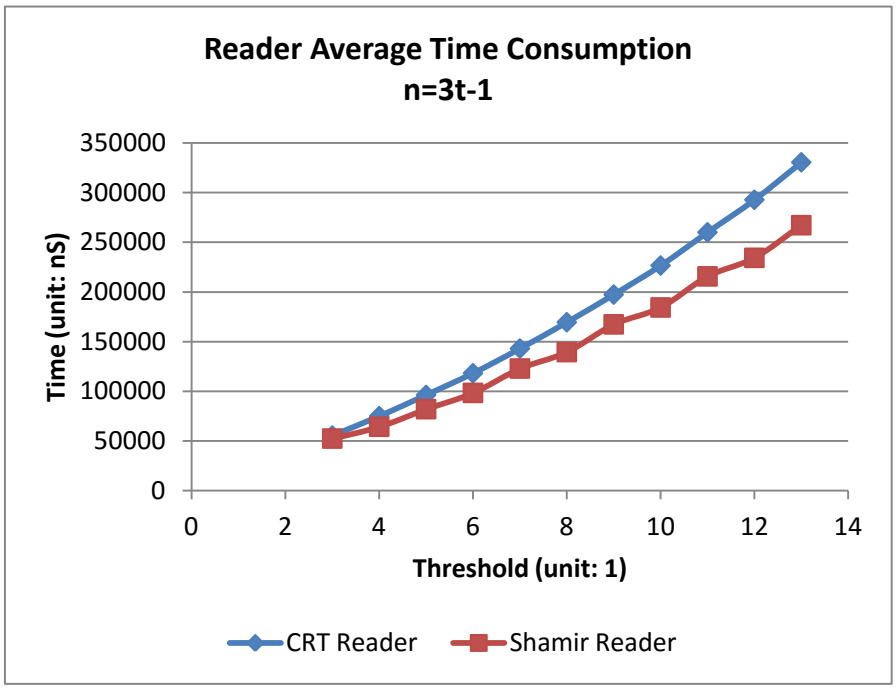

Figure 23 Reader Average Time Consumption When Set $n = 3t - 1$

Then we consider CRT-scheme and Shamir-scheme seperately.

We first analysis the Shamir-scheme by setting $n=17$ with $t=3,4,...,13$ and $t=3$ with $n=5,6,...,14$, respectively.

According to (3.6.4), set $n=17$, we have:

$$E_{Shamir-Dealer}=\frac{18\times t}{18-t}(E_{Hash}+E_{Enc})+E_{Other} \tag{3.6.16}$$

Despite E_{Other} which is relatively small, the energy cost increases as the threshold get bigger. The testing result is shown in Figure 24.

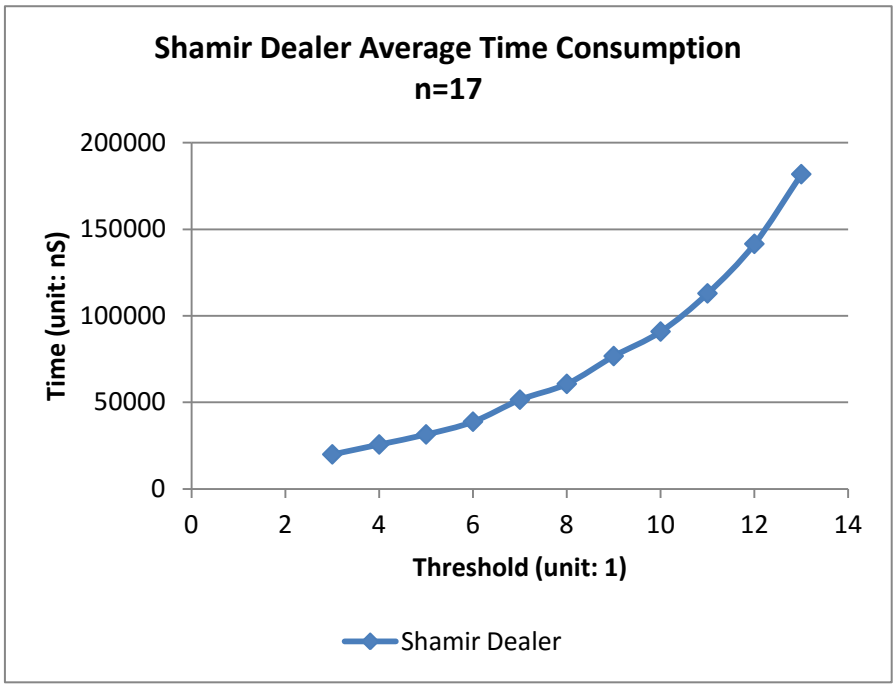

Figure 24 Dealer Average Time Consumption in Shamir's Scheme When Set $n=17$

Set $t=3$, we have:

$$\begin{aligned} E_{Shamir-Dealer}&=\frac{n\times 3}{n-2}(E_{Hash}+E_{Enc})+E_{Other}\\ E_{Shamir-Dealer}&=(3+\frac{6}{n-2})\times(E_{Hash}+E_{Enc})+E_{Other}\end{aligned} \tag{3.6.17}$$

Thus, the long term average energy cost will be decreasing if the total number of the proxy in the system becomes bigger. The testing result is shown in Figure 26.

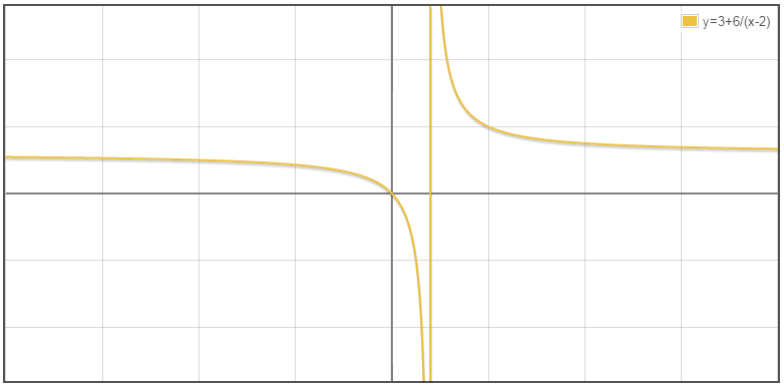

Figure 25 Functional Figure of Function $f(x)=3+\frac{6}{x-2}$

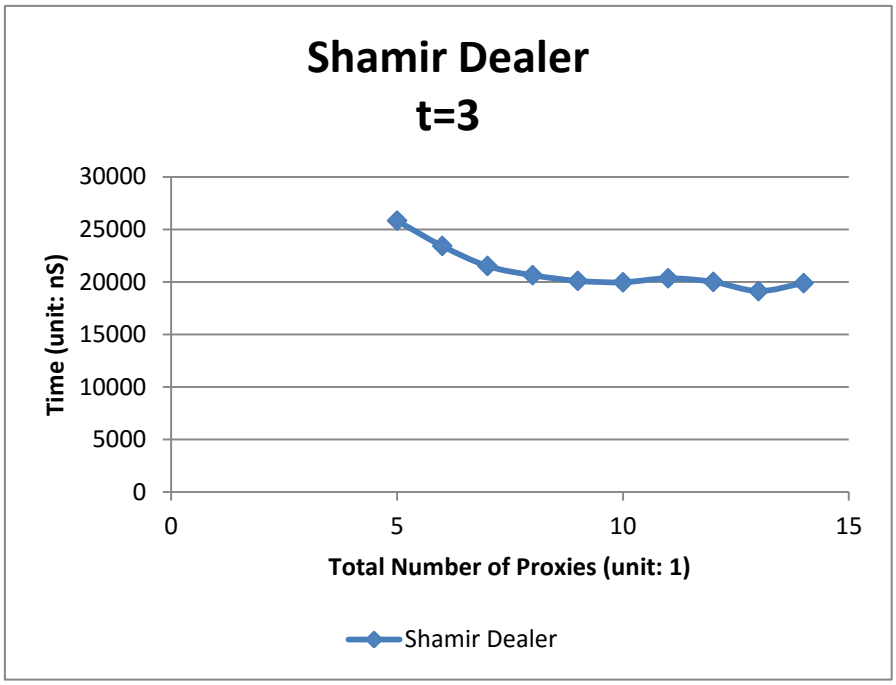

Figure 26 Dealer Average Time Consumption in Shamir's Scheme When Set $t=3$

The same analysis on CRT-scheme is done below:

Set $n=17$, according to Equation (3.6.3), we have:

$$E_{CRT-Dealer}=17\times E_{Hash}+E_{Other} \tag{3.6.18}$$

Which indicates that the energy cost does not affected by the increasing of threshold if we despite the other costs. However, since the threshold will affect the secret computation and reconstruction operations, the other costs will increase if the threshold is set to a larger value. The testing result is shown in Figure 27.

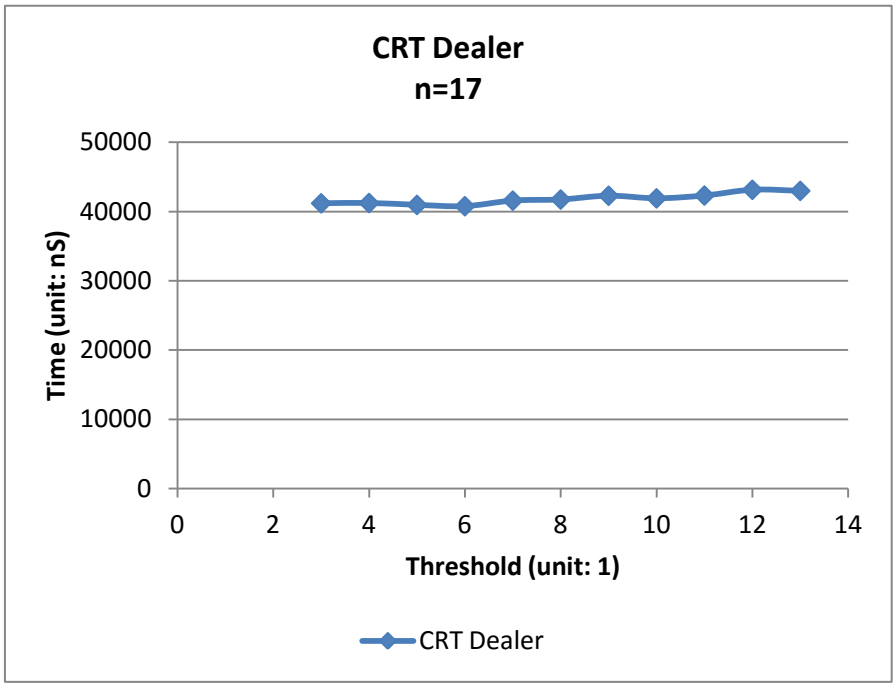

Figure 27 Dealer Average Time Consumption in CRT-Based Scheme When Set $n = 17$

Set $t = 3$, we have:

$$E_{CRT-Dealer} = n \times E_{Hash} + E_{Other} \tag{3.6.19}$$

Which have no difference with equation (1). The energy cost will linearly grow along with the increase of total number of proxy in the system. The testing result also shows the same trend in Figure 28.

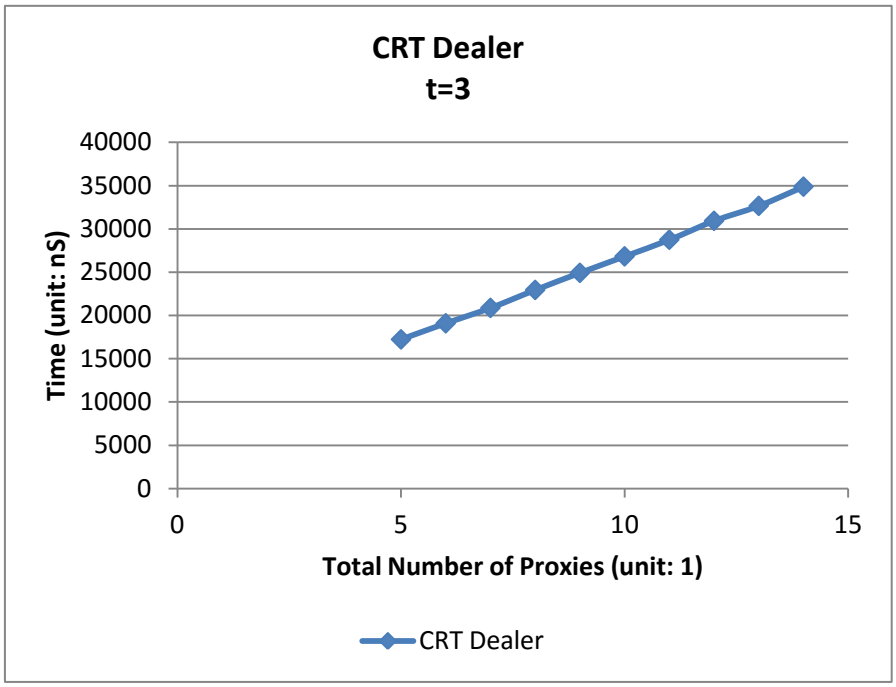

Figure 28 Dealer Average Time Consumption in CRT-Based Scheme When Set $t = 3$

Put the results we get from both schemes, we can have the following comparison between two Dealers in different schemes shown in Figure 29 and Figure 30.

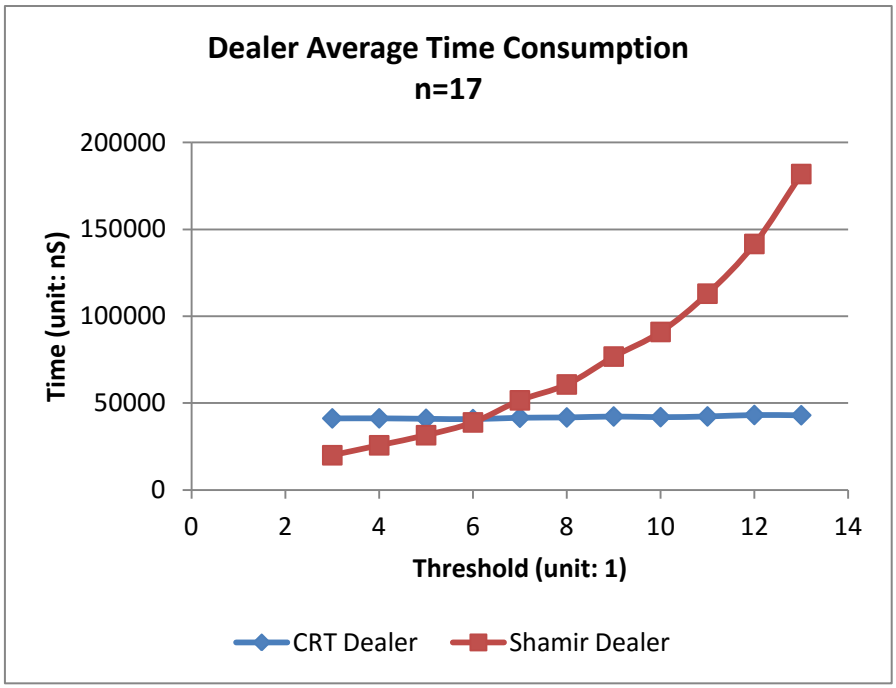

Figure 29 Dealer Average Time Consumption When Set $n = 17$

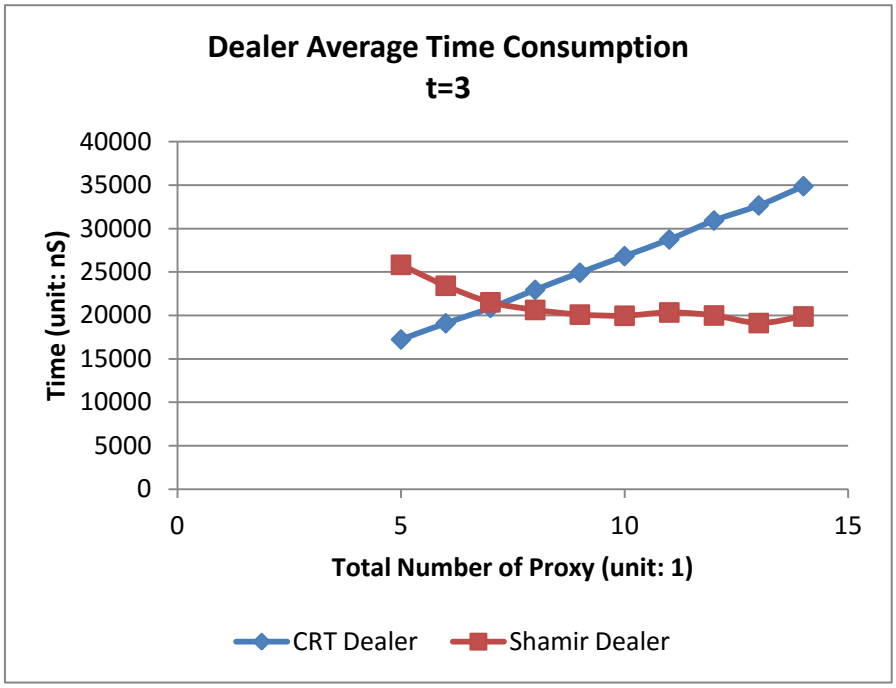

Figure 30 Dealer Average Time Consumption When Set $t = 3$

We further estimated and tested the results on Proxy and Readers when set $n = 17$ and $t = 3$ separately, the results for Proxy are shown in Figure 31 and Figure 32 while the results for Readers are shown in Figure 33 and Figure 34.

For Proxy:

Set $n = 17$:

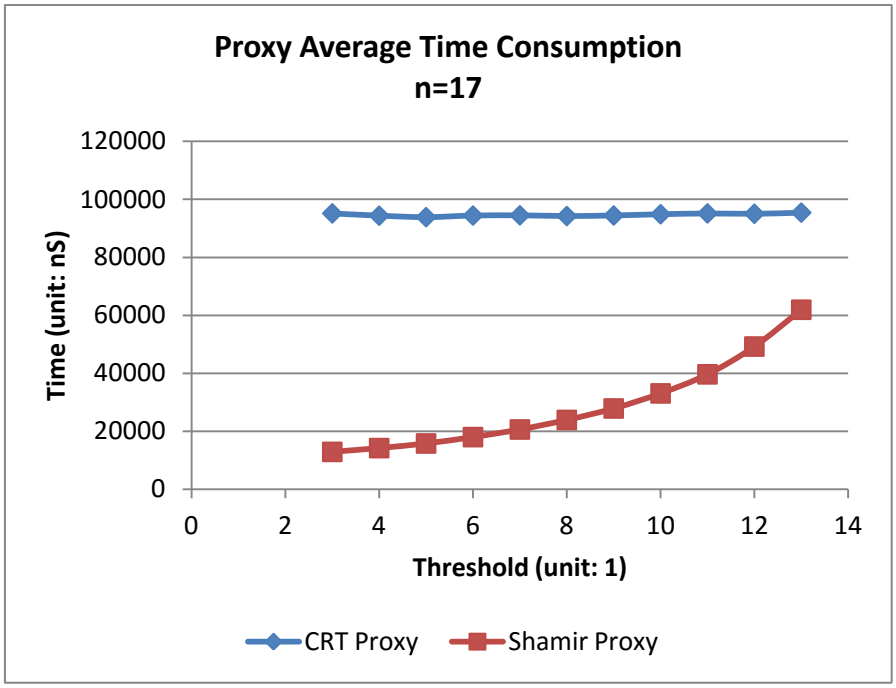

Figure 31 Proxy Average Time Consumption When Set $n = 17$

Set $t = 3$:

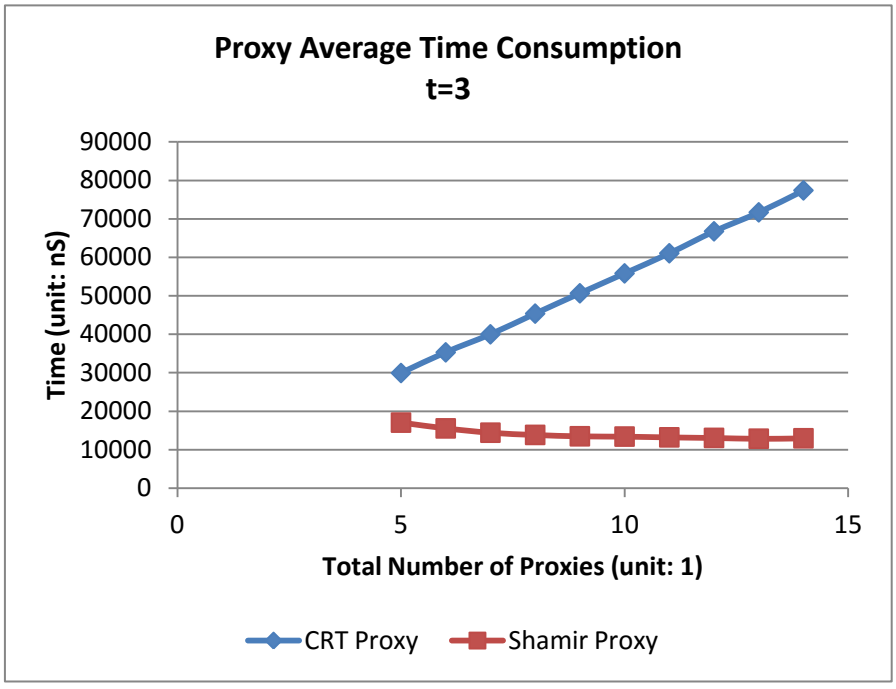

Figure 32 Proxy Average Time Consumption When Set $t = 3$

For Readers:

Set $n = 17$:

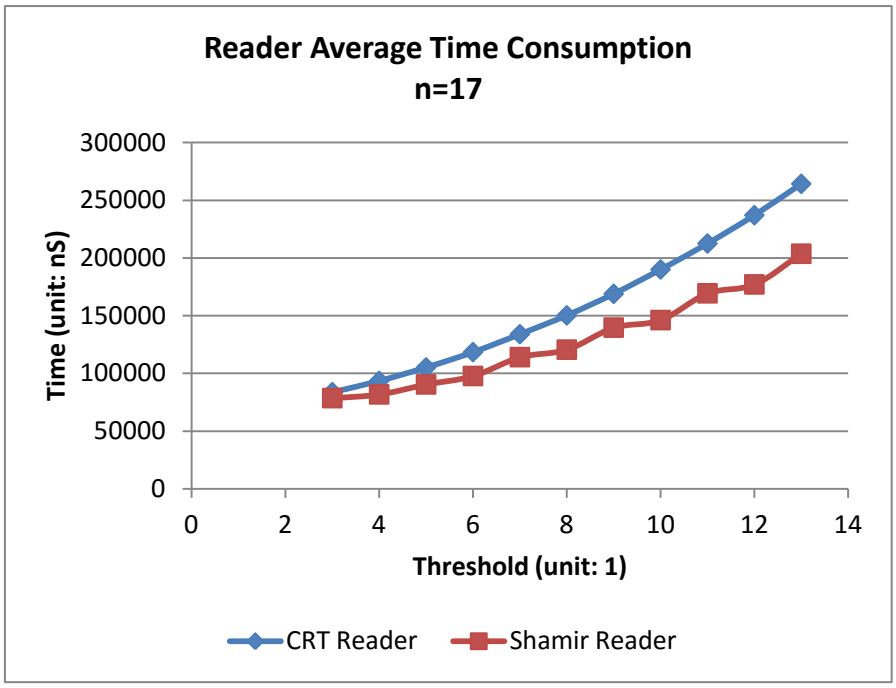

Figure 33 Reader Average Time Consumption When Set $n = 17$

Set $t = 3$:

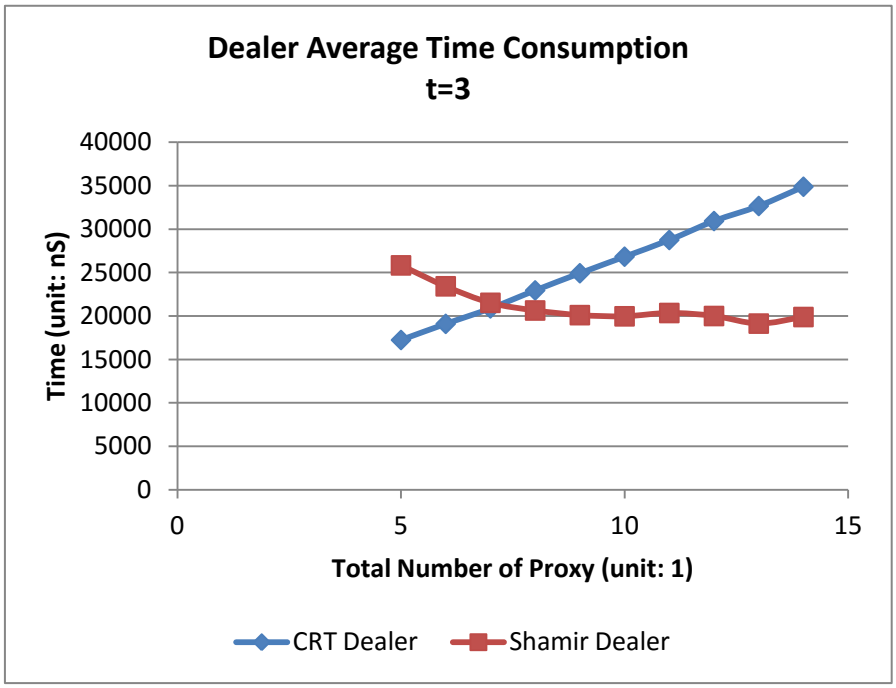

Figure 34 Reader Average Time Consumption When Set $t = 3$

Conclusion:

According to our theoretical calculation and testing results, we come to the following conclusion.

First, the expected system parameter setting is $t \geq \frac{n}{2}+1$ to make sure that the attackers needs to compromise the majority of the Proxy to finally revealling the secrets and information. In the meanwhile, the value of t should not be too close to n to ensure certain level of robustness.

Second, if the total number of the Proxy are set to be stable, the energy cost of the Dealer in Shamir-scheme is affected by the increasing of the threshold while the Dealer in CRT-scheme is not. In other words, the Dealer in Shamir-scheme costs more energy when the system becomes more secure. On the contrary, the CRT-scheme can be set to any security level by adjusting the threshold without concerning to much about the energy cost on the Dealer.

Third, under the desired system parameter settings, which is $t \geq \frac{n}{2}+1$, CRT-scheme has better performance than Shamir-scheme concerning about the energy consumption on the Dealer side. Because of the second conclusion we get, in the range of $[\frac{n+2}{2}, n)$, the CRT-scheme can provide better performance when the threshold increases. In the worst case, let $t = \frac{n+2}{2}$, the improvements of the energy cost on the Dealer side is $E_{Diff} = 2 \times E_{Hash} + 2t \times E_{Enc}$.

Fourth, the energy cost on the Proxy side is increased as we expected, however, since the Proxy are considered more powerful and resourceful than the resource constrained Dealer, the increase of the energy cost will be considered acceptable.

Our CRT based multi-secret sharing scheme outsourced the computational workload of the Dealer to the relatively powerful Proxy and successfully decreased the energy cost, meanwhile, our scheme also decreases the frequent communications happened between the Dealer and the Proxy.

Chapter 4

Flexible WSN Key Management Scheme Based on (*n*,*t*,*n*) Multi-Secret Sharing

4.1 Architecture

The architecture of our proposed key management scheme is hybrid architecture, which means the sensor nodes could communicate using both Peer-to-Peer (P2P) and clustered communication model. And in order to make this hybrid architecture to work efficiently and securely, we introduce dedicated relatively more powerful devices in to the system. Here, we reference these devices as key managers (KMs). However, the key managers can also become the intermediate hops to help a sensor node or another key manager to transmit its encrypted messages to another remote node. And in clustered communication, some key managers might become cluster heads as well. Therefore, after the deployment, all key managers will collaboratively generate an efficient topology for communication and routing purpose, and this topology will keep updating based on the trustworthiness of the key managers and their relative locations. And it is also possible to add or delete a key manager according to the feedbacks from the sensor nodes who are receiving the services from them. The whole architecture is outlined in Figure 35, and the two communication models are shown in Figure 36 and Figure 37, respectively.

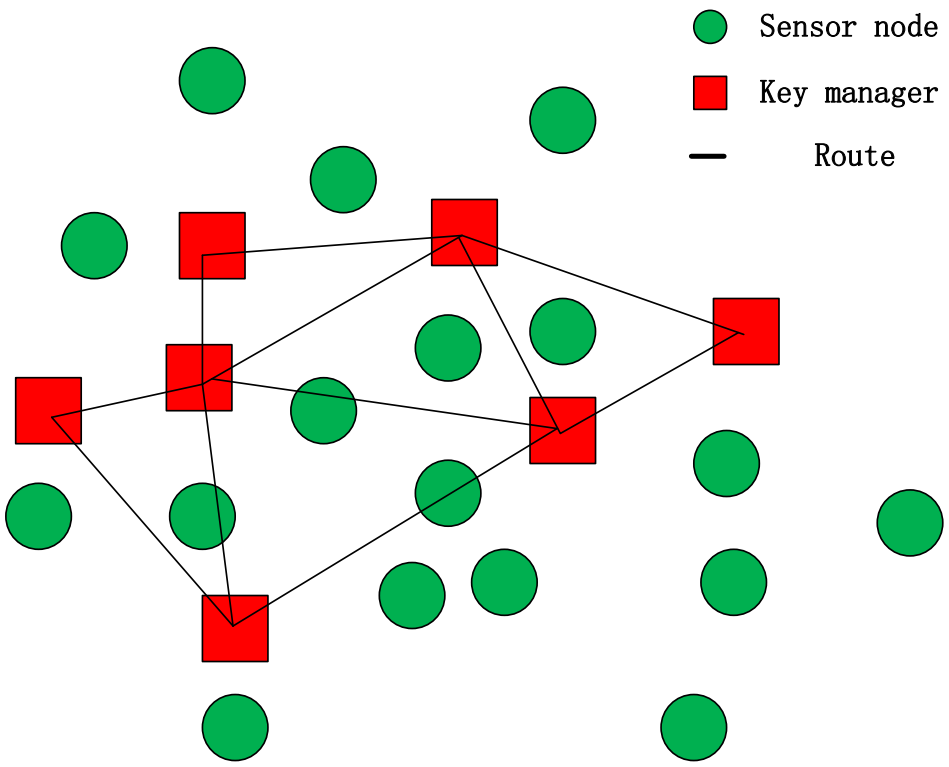

Figure 35 Basic Key Manager and Sensor Node Distribution and Architecture

In Figure 35, key managers and sensors are randomly deployed in the same area which makes them geologically close to each other. The key managers are powerful devices with longer communication range and more resources. Unless it is necessary, we only use key managers to route the packets through the established topology. This will save energy for the resource constrained sensor nodes, and from the system's perspective, using key managers as intermediate hops can save the overall number of hops and the system will be more efficient. After the deployment, every key manager should be able to find more than one key managers within its own communication range, and every sensor should also be able to communicate directly with more than one key managers in case of some manager might be compromised by the attackers or get offline after the deployment.

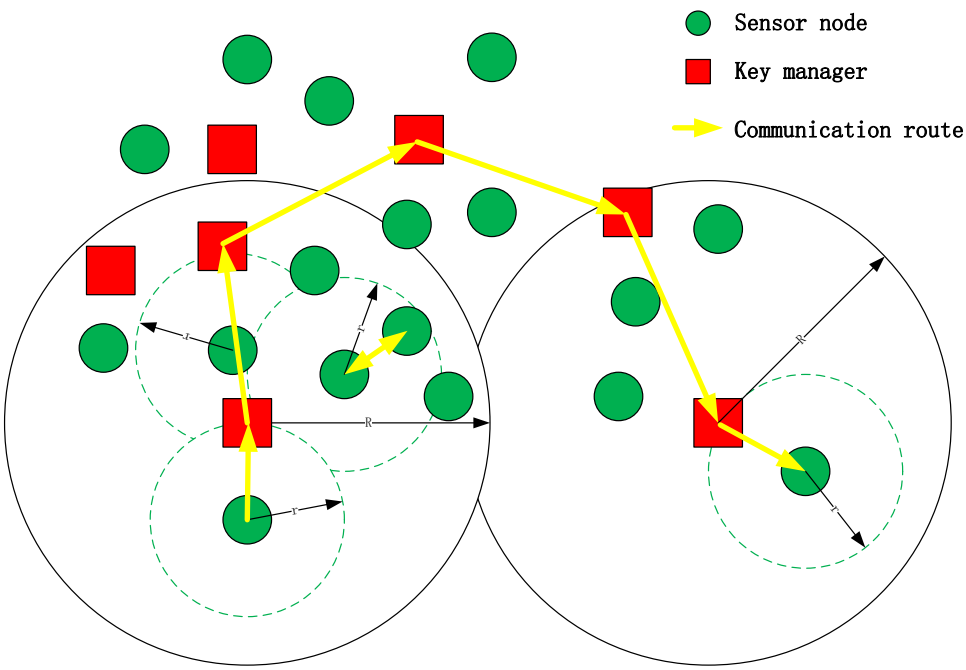

Figure 36 P2P Communication Model

In **Figure 36**, we use yellow arrows indicate the communication connection between two sensors. There are two types of complete connections: one is the direct connection connecting two geologically close sensor nodes; the other one is multi-hop connection using multiple key managers as intermediate hops.

Figure 37 shows the clustered communication model in our proposed scheme. Sensors could communicate with the key manager which is in their communication range, and the key managers, which have larger communication ranges, could act as routers between two sensors. So, there are two ways of constructing a cluster, the first one is constructing cluster based on the geological location information, the other one is constructing cluster based on the function of the sensor nodes, which means, several remote sensors which have the same function, such as temperature testing, as long as they have the same cluster key, they can be considered as cluster members of the same cluster.

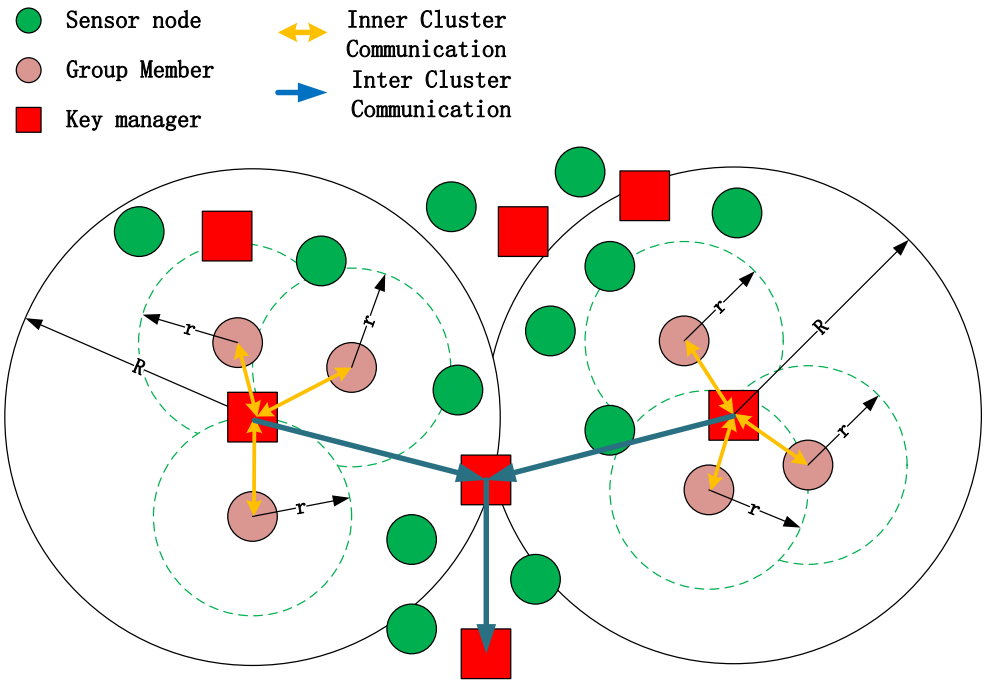

Figure 37 Clustered Communication Model

In Figure 37, there are two clusters constructed based on the geological location information, each has a key manager as the cluster head, and since the two cluster heads cannot communicate directly because of the communication range, they collect data from their own cluster members and submit to an intermediate key manager, which continuously transmits the collected data to the next hop until the packets arrive the destination. This is a hieratical multi-tier structure, we take the intermediate key manager as a superior cluster head which manages two cluster heads and two cluster managed by these two cluster heads. This hierarchical structure can be established based on the actual application requirements. And multiple hierarchical structures can be established at the same time.

4.2 Background

4.2.1 Basic Secret Sharing Schemes

We presented two different (***n,t,n***) multi-secret sharing approaches in Chapter 3, one is Lagrange polynomial interpolation based (***n,t,n***) secret sharing scheme and the other one is CRT based (***n,t,n***) secret sharing scheme. They both use proxy to outsource the workload from the highly resource constrained devices. However, they also have some differences because of the underlying mathematical mechanisms.

Regarding to our application scenario, the key managers are responsible for the secret shares generation and distribution, the sensors are responsible for the secret reconstruction operations which is energy consuming in both schemes. Since the proxy are the most powerful devices in this application, we propose to use CRT base multi-secret sharing scheme instead of Lagrange polynomial interpolation based one since the theoretical computational complexity for Shamir's scheme is $O(t\log^2 t)$ while it is $O(t)$ for Asmuth-Bloom scheme [23] and the testing results of computational costs of the two schemes also indicate the CRT based scheme is more efficient regarding to the dealer while the two schemes have the similar cost regarding to the reader. We made some modifications to further offload the computational workload from the sensor which needs to reconstruct the secret.

As we can see from the description in Chapter 3, an (*n,t,n*) multi-secret sharing scheme involves three entities: dealer/initiator, reader, and multiple proxy. Here, we only have two types of devices: sensors and key managers. The key managers can be viewed as proxy, and the sensors can be seen as initiators or readers depending on their actions during the communication.

4.2.2 Notation Used in Key Management Scheme

Before the detailed discussion of our proposed key management scheme, we list the important notation and the corresponding descriptions in Table 2. The notation will be used during the discussion and presentation of the proposed scheme.

Table 2 Key Management Scheme Related Notation and Corresponding Descriptions

Symbol	Description
N_j / N_p	Sensor nodes with a sequence number of j or q , where $j,q=1,2,...,l$, $j \neq q$
KM_i / KM_p	Distributed key manager with a sequence number of i / p , where $i, p=1,2,...,t,t+1,...,n$, and $i \neq p$
n	The total number of the key manager
t	Pre-defined threshold
l	The total number of the sensor nodes
ID_i	The initial identification number for i -th key manager
IN_j	The identification number for j -th node
k	Random number that generated by the sensor and initiates the communication for the current session
S_k	Session key generated in the session tagged with random number k
$KP_{i,j}$	Secret key pre-shared by KM_i and N_j
$KK_{i,p}$	Secret key pre-shared by KM_i and KM_p
$sh_{i,p}^k$	Sub-share generated for KM_p by KM_i , here this k is a superscript
$SH_{i,j}$ / $SH_{p,j}$	Master share, shared between KM_i and N_j
Sh_i^k / Sh_p^k	Reconstruction shares generated by KM_i / KM_p , here this k is a superscript
IC	Identification for cluster

$H(x, y)$	One-way function with two inputs
RN_i^k	Random number generated by the *i*-th key manager in k -th session, here this k is a superscript
$Seed_{i,j}$	Secret seed used in hash function shared between KM_i and N_j
m_i	Large prime number which satisfy $m_0 \prod_{i=n-t+2}^{n} m_i < \prod_{i=1}^{t} m_i$, and $0 < m_0 < m_1 < \cdots < m_n$
c_i	$M = \prod_{i=1}^{t} m_i$, $M_i = M / m_i$, $c_i = M_i \times (M_i^{-1} \bmod m_i)$, $1 \le i \le t$.

4.3 Flexible Key Management Scheme

We modified CRT based (***n***,***t***,***n***) multi-secret sharing scheme to allow the decentralized key managers collaboratively generate and assign new session secret keys for the sensors. As long as there are ***t*** or more legislate key managers and they are able to communicate with other, the sensors within their communication range can always obtain secret keys for P2P communication or clustered communication.

In different communication modes, the involved entities have different or similar behaviors and the following subsections will present a detailed description.

4.3.1 P2P Communication Mode

As shown in Figure 38, the P2P communication mode has two phases: installation and key generation.

Installation Phase.

- Before deployment, $\{ID_i,(Seed_{i,1},...,Seed_{i,l}),(SH_{i,1},...,SH_{i,l}),(KP_{i,1},...,KP_{i,l}),(KK_{i,1},...,KK_{i,n})\}$ are assigned to key manager KM_i

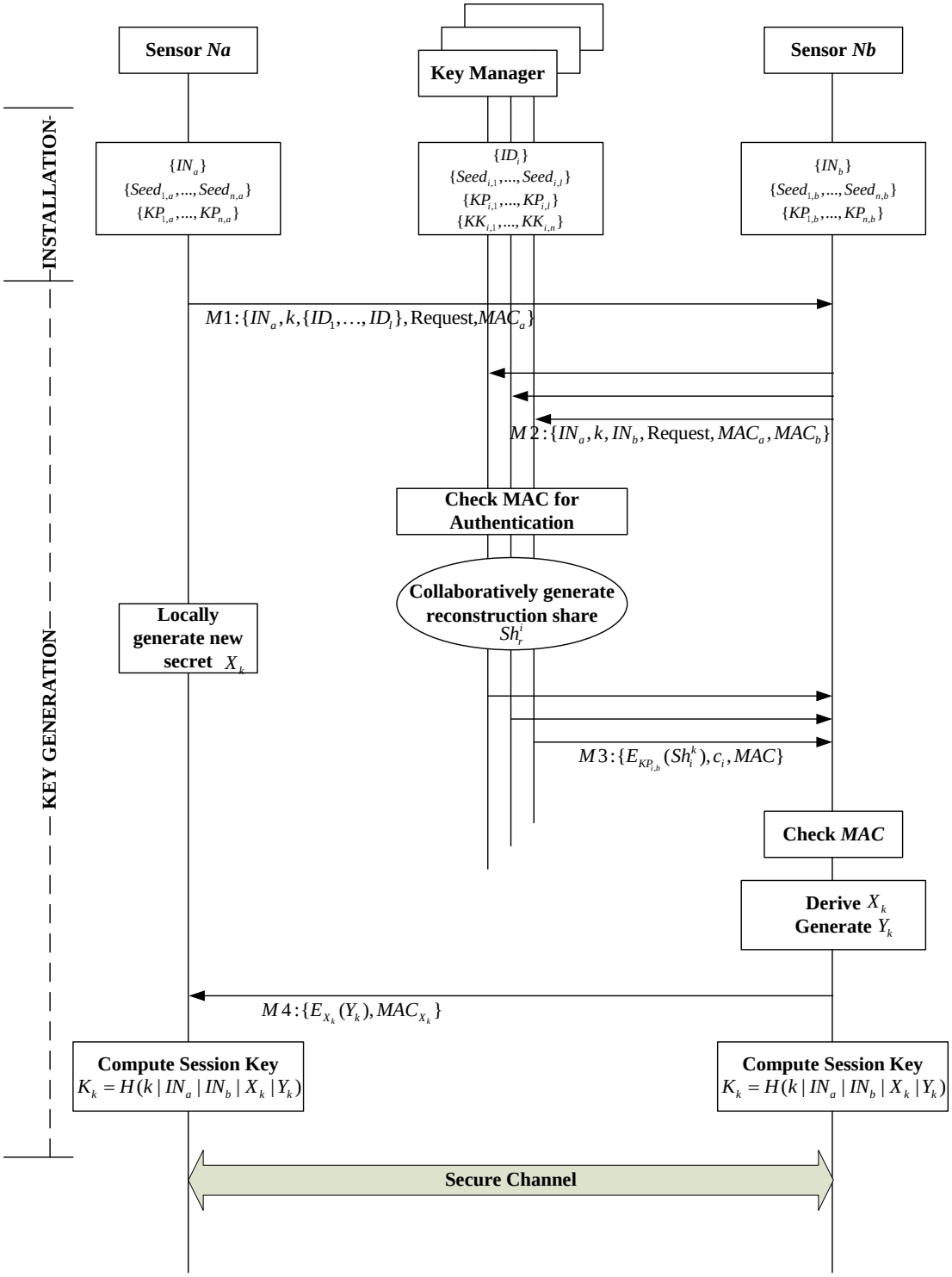

Figure 38 P2P Communication Mode of Flexible Key Management Scheme

- $\{IN_j,(Seed_{1,j},...,Seed_{n,j}),(SH_{1,j},...,SH_{n,j}),(KP_{1,j},...,KP_{n,j})\}$ are assigned to sensor node N_j.

- After the deployment, all the key managers and sensor nodes are randomly distributed in a large area. They will instantly generate the topology for clustering and routing purposes. This topology will be updated periodically.

Key Generation.

1) Sensor $\boldsymbol{S_a}$ generates a random number k for the purpose of distinguish different communication sessions in case of sessions between the same two nodes always get the same session key. Then $\boldsymbol{S_a}$ selects a sub-set of key managers within the network and send the identification numbers to sensor $\boldsymbol{S_b}$ together with the random number k and the key setup request through message M1. Sensor $\boldsymbol{S_a}$ also generates HMACs for the message for integrity check and authentication purpose.

Key material X_k is then calculated locally by the following operations:

$$RN_i^k = H(k, Seed_{i,a}) \tag{4.3.1}$$

$$X_k \equiv \sum_{i=1}^{n} (RN_i^k \cdot SH_{i,a}) (\mathrm{mod}\, M) \tag{4.3.2}$$

2) Sensor $\boldsymbol{S_b}$ receives the message M1 and if the request is accepted, sensor $\boldsymbol{S_b}$ contacts the key managers appointed by sensor $\boldsymbol{S_a}$ by sending message M2. M2 passes along the content in M1 and adds its own identification number and the HMACs.

3) After receiving message M2, key managers will check the HMACs generated by both sensors to check the integrity of the message, and since the seed used in the HMAC operation is known only by the corresponding sensors and the key managers, only legitimate sensors are able to generate the HMACs correctly.

4) All the key managers collaboratively generate reconstruction shares Sh_i^k.

Using master shares and shared seeds, the key manager KM_i can generate sub-shares for KM_p on sensor $\boldsymbol{S_a}$'s behalf:

$$sh_{i,p}^{k} \equiv H(k, Seed_{i,a}) \cdot SH_{i,a} (\mathrm{mod}\, m_p) \tag{4.3.3}$$

Upon receiving all the sub-shares from selected key managers, KM_i can generate the reconstruction share:

$$Sh_i^k \equiv \sum_{p=1}^{n} sh_{p,i}^{k} (\mathrm{mod}\, m_i) \tag{4.3.4}$$

KM_i also computes c_i for sensor $\boldsymbol{S_b}$ in order to further reduce the energy consumption. The reconstruction shares are then encrypted using the unique shared keys shared between key managers and sensor $\boldsymbol{S_b}$, HMACed using the shared seeds, and sent to sensor $\boldsymbol{S_b}$ for the key material retrieving.

5) Sensor $\boldsymbol{S_b}$ decrypts M3 and uses the reconstruction shares and c_i to reconstruct the shared secret, X_k.

$$X_k \equiv \sum_{i=1}^{t} Sh_i^k \cdot c_i (\mathrm{mod}\, M) \tag{4.3.5}$$

6) Then sensor $\boldsymbol{S_b}$ generates its own key material Y_k, encrypts Y_k by X_k. Y_k is then sent back to sensor $\boldsymbol{S_a}$ and both sensors calculate the final session key using the pre-agreed hash function:

$$K_k = H(k \mid IN_a \mid IN_b \mid X_k \mid Y_k) \tag{4.3.6}$$

4.3.2 Clustered Communication Mode

Since we are using key managers to collaboratively generate and share secret keys, we could generate one key for just two nodes or a set of nodes. In the first case, two nodes share an exclusive key, noted as $EK_{j,q}^{k}$, means this key reconstruction is initialized by node N_j in the k-th session

with node N_q. In the second case, we say that the key shared by more than two nodes is a group key, denoted as $GK_{c,j}^{k}$, means this group is initialized by node N_j in the k-th session with a cluster identification number c, $c = 1, 2, 3, ...,$ and actually $GK_{c,j}^{k} = S_k$.

Assuming key manager KM_i claims to be a cluster head and initiates the construction of a new cluster based on the function of the sensor nodes, which means the sensor nodes are not geologically close to each other and KM_i. Assuming the selected cluster members are N_m, $m = 1, 2, ..., l$.

There are two methods of constructing a new cluster.

Since the key manager KM_i shares a unique secret key with each sensor node, the cluster can be constructed by asking KM_i to generate a cluster communication session key and distribute to the selected sensor nodes secretly by encrypting all the messages using the pre-shared keys. Then all the members within the newly created cluster start communicate privately using the distributed secret key.

The other way is much more complex but necessary in some scenarios. For example, if the cluster head is a new member of the WSN or a sensor node, it does not share a pre-shared key with every sensor node. Thus, it cannot use the previous method to construct a new cluster. We propose to use the other key managers to authenticate newly added cluster head first and then exchange pre-shared keys between key managers using asymmetric encryption techniques. Once the cluster head is accepted by all the other key managers, it inquiries the selected sensor nodes to establish a new clustered communication session, in the meantime, it also inquiries the other key managers to collaboratively generate a new secret. The sensor nodes which accepted the communication request will contact the key managers for the secret key, this new key will be assigned to both KM_i and

the selected sensor nodes; otherwise, the key managers won't generate this new key, and the communication session won't be established. The key generation process is the same with P2P communication model, the only difference is the same reconstruction shares are delivered secretly to multiple sensor nodes.

4.4 Key Managers Leave or Join the System

Delete Key Manager.

In several occasions, we need to delete a key manager from the system. This key manager might be removed from the targeting location, have used up all the energy, be labeled as untrustworthy based on the feedbacks of the sensors, and so on. To delete a key manager, most of the legitimate key managers need to send an encrypted command to the sensors to inform them the left of a key manager. All the sensors received this command will delete all the pre-shared keys associated with this key manager. This method works because of the previous assumption that most of the key managers are trust worthy.

New Key Managers Join the System.

A new key manager does not share a pre-shared key with all the sensor nodes. If the key manager wants to obtain those pre-shared keys with all the sensor nodes, we propose to use the following procedures.

Assuming the newly joined key manager is KM_p, all the other existing key managers are KM_i with $i = 1, 2, ..., n$. KM_i first uses public key infrastructure to authenticate the new member and then exploits asymmetric encryption techniques, such as Diffie-Hellman key exchange scheme, to share a symmetric key with KM_p.

Key manager KM_i computes random numbers as the secret:

$$H(KP_{i,j}, Seed_{i,j}, ID_p) = R_{i,j}^p \quad (4.4.1)$$

ID_p are known by all the members, but $KP_{i,j}$ and $Seed_{i,j}$ is only known by KM_i and the j-th sensor node, so $R_{i,j}^p$ is also known only by KM_i and the j-th sensor node. All KM_i then encrypt $R_{i,j}^p$ and send to KM_p. KM_p can obtain all the $R_{i,j}^p$ with $i = 1,2,...,n$ and $j = 1,2,\ldots,l$.

KM_p will further compute the new pair-wised key using one-way hash function:

$$KP_{p,j} = H(\sum_{i=1}^{n} R_{i,j}^p, ID_p) \quad (4.4.2)$$

When KM_p needs to communicate with the j-th sensor node, the sensor will computes $KP_{p,j}$ locally using Equation (4.4.1) and Equation (4.4.2) based on the necessary information.

The first time KM_p joins the system, it obtains the keys from the other key managers using the method we presented above. All the sensor nodes will be informed about the new key manager, and locally computes the secret key when necessary. During the first communication session between KM_p and the sensor node, the sensor node will encrypt a puzzle and send to KM_p, if KM_p successfully decrypts the message and provides the correct answer, that means all the other legitimate key managers already authenticated this new key manager and provided KM_p the required keys with their trust.

Chapter 5

HIP-MEX: New Key Agreement Schemes Based on (*n*,*t*,*n*) Multi-Secret Sharing for HIP Based Internet of Things

5.1 Introduction

In Chapter 1, we briefly mentioned HIP based key agreement schemes and introduced few of them in Chapter 2. As we indicated above, since we are trying to reduce the workload as much as possible for the resource constrained devices, the current schemes can still be improved or modified.

In this chapter, we first introduce TD-HIP, which is the most efficient existing key agreement scheme for HIP, in detail. Then, we apply the (*n*,*t*,*n*) multi-secret sharing scheme based on Lagrange polynomial interpolation to HIP context. We also propose to modify our CRT-based (***n,t,n***) multi-secret sharing scheme into key agreement scheme in HIP to further reduce the communication and computation workload for the resource constrained devices. The last two schemes we also referred to as Lagrange polynomial interpolation based HIP Multi-secret sharing Key Exchange (HIP-MEX-LPI) and CRT based HIP-MEX-CRT.

Through the introducing of different schemes, we will analyze and compare those schemes in terms of the energy cost on communication and computation operations.

5.2 Background and Architecture

Before we start to describe the proposed scheme, we need to make some basic assumptions and those assumptions apply to all the three schemes we are about to review or propose.

The initiator and the proxy can communicate with each other through secure channels. We assume that before the deployment of the devices, the unique shared secret keys will be installed between every pair of the devices, so they can communicate with each other privately using the symmetric encryption techniques.

We applied two types of encryption techniques in the proposed scheme, the first type is symmetric encryption, used to encrypt the communication between proxy and the initiator, as well as among the proxy; and the later communications happened directly between the dealer and the responder are also encrypted using symmetric encryption techniques; the second type is asymmetric encryption, such as RSA and Elliptic curve cryptography, used to encrypt the communication between proxy and the responder.

The majority of the proxy are trustworthy. Otherwise, the cheaters among honest proxy will compromise the whole system without taking the risk of being exposed [75].

All the involved devices are wireless devices with limited communication range and mobility. The initiator and its proxy should be geologically close to each. The responder, in the other hand, can be any device moved into this local wireless network.

5.3 HIP-MEXs

In this section, we present two key exchange schemes for HIP using the same architecture as TD-HIP. They are both based on (n,t,n) multi-secret sharing schemes and according to the

different underlying mathematical principle, we refer to them as HIP-MEX-LPI (Lagrange polynomial interpolation based multi-secret sharing key exchange for HIP) and HIP-MEX-CRT (CRT based multi-secret sharing key exchange for HIP).

5.3.1 HIP-MEX-LPI

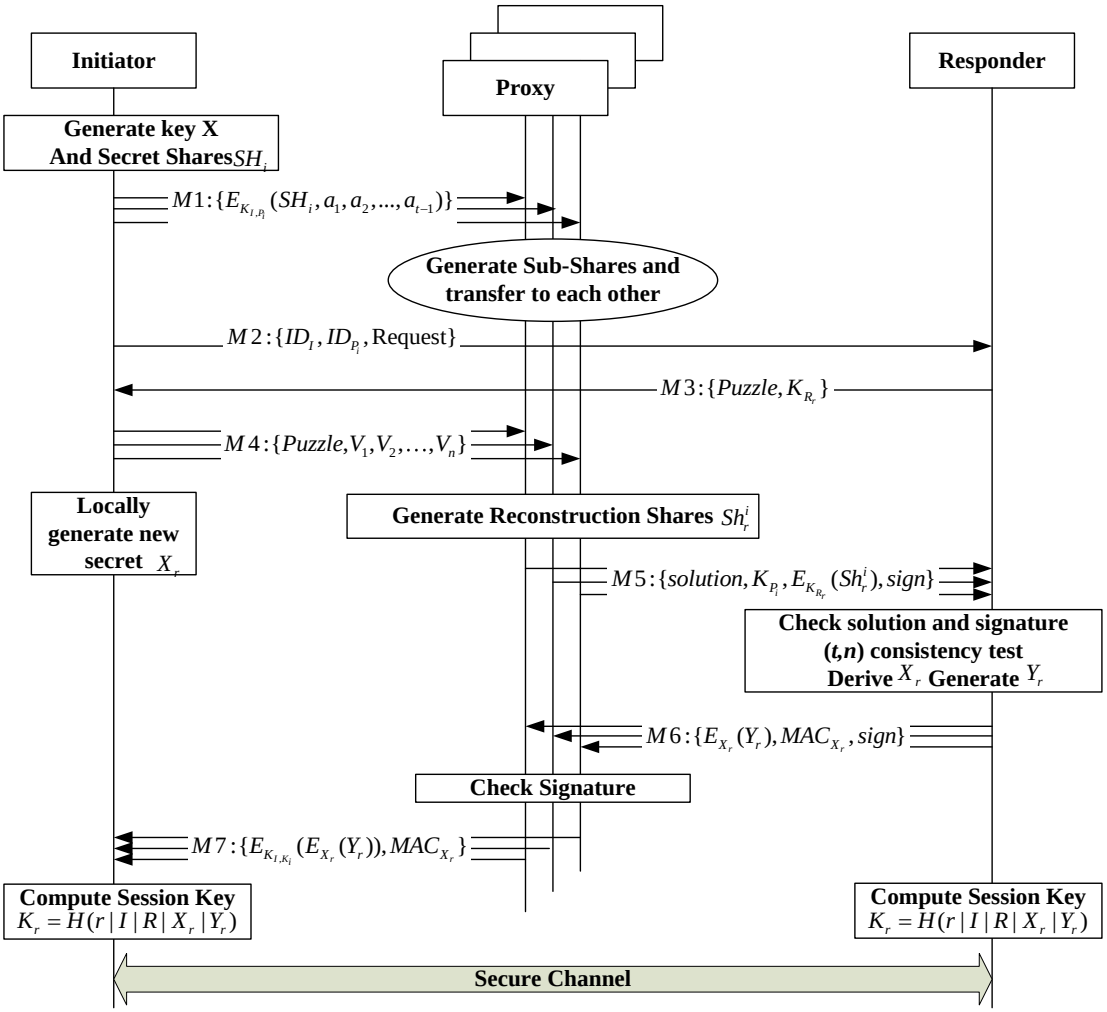

Figure 39 Lagrange Polynomial Interpolation Based HIP-MEX

HIP-MEX-LPI, as shown in Figure 39 is a direct application of the (*n*,*t*,*n*) multi-secret sharing scheme we reviewed in Chapter 2 proposed by Liu, Yanxiao et al [29].

This type of HIP-MEX consists of two phases: preparation phase and key generation phase.

Preparation Phase:

The preparation phase is active when every (*n-t+1*) secret is generated and exchanged to make sure that the attackers cannot compromise the system.

Message M1: The initiator generates (*t*-1) degree sub-polynomials $f_i(x) = SH_i + a_1x^1 + a_2x^2 + \cdots + a_{t-1}x^{t-1} \bmod p$ for each proxy, and then encrypts these master shares using the pre-installed shared secret key K_{I,P_i}. Here SH_i is the master share for the *i*-th proxy. The cipher texts will be distributed to proxy through message *M1*. Upon receiving the ciphers, the proxy will decrypt them and get their own master share, and using this master share, the proxy P_i can generate sub-shares $sh_i^j = f_i(x_j)$ and distribute to proxy P_j secretly.

Key Generation Phase:

Message M2: The initiator sends request to the responder together with its own id and the most trusted proxy's ID.

Message M3: The responder will verify the identity of the initiator, and if the responder accepts the request, it will send a response message containing its public key K_{R_r} and a puzzle back to the initiator.

Message M4: The initiator receives message M3 and then broadcast the generated vectors $\{V_1, V_2, \cdots, V_n\}$ and the received puzzle to all its proxy.

Message M5: The proxies compute the solution for the responder and they then can use the received vectors and previously obtained sub-shares to locally generate the reconstruction shares:

$$Sh_r^i = (V_1 \bullet sh_i^1 + V_2 \bullet sh_i^2 + \cdots + V_n \bullet sh_i^n) \bmod p$$

The reconstruction shares will be secretly sent to the responder through message M5.

Message M6: The responder will verify the solution and the signed signature first, and then check the (*t*,*n*) consistency of the received reconstruction shares. If everything checks out, the responder will be able to derive the secret using $X_r = \sum_{j=1}^{t} Sh_r^j (\prod_{r=1, r\neq j}^{t} \frac{-x_r}{x_j - x_r}) (\mathrm{mod}\ p)$. Then the responder generates key material Y_r and encrypts Y_r using X_r.

Message M7: The proxy will verify the authentication and integrity of the received information $E_{X_r}(Y_r)$, and then encrypt the received cipher using the secret key shared with the initiator. The initiator can decrypt all the information delivered through message M7, which means the initiator and the responder exchanged their key material and will be able to generate a session key for further communication.

5.3.2 HIP-MEX-CRT

Figure 40 shows the proposed HIP-MEX-CRT in detail. As we can see, the proposed scheme has a similar architecture to HIP-MEX-LPI. However, by using CRT-based multi-secret sharing [38], HIP-MEX-CRT can finish key exchange with minimum interaction between the initiator and the other entities; most of the heavy workload and expensive cryptographic operations are outsourced to the relatively powerful devices.

Table 3 lists all the useful notation and their corresponding descriptions. The notation will be used to describe HIP-MEX-CRT (slightly different from symbols previously used in MIP-MEX-LPI) later in this section.

Table 3 Notation and Corresponding Descriptions for HIP-MEX-CRT

Notation	Description
I	Initiator, highly resource constrained device, also used as the Host ID of the

	initiator
$\boldsymbol{R_r}$	Responder, powerful device, ***r*** indicates different responder or communication session, ***r***=1,2,3,…, also used as the host ID of the responder
$\boldsymbol{P_i}$ /$\boldsymbol{P_j}$	Proxy, relatively powerful devices, could also be used as the Host ID of the proxy. ***i,j***=1,2,3,…,***n***, and $i \neq j$
t	The threshold of the scheme, the minimum number of shares required to retrieve the shared secret
n	The total number of involved proxy
m_0	Large positive integer selected to set up the upper bound of the key space
m_i	Randomly selected pair-wise co-prime integers, m_i with *i*=1,2,3,...,*n* and $m_0 < m_1 < m_2 < \cdots < m_n$. They also fulfill this requirement: $m_0\left(\prod_{i=n-t+2}^{n} m_i\right) < \prod_{i=1}^{t} m_i$
X	Master secret, X is an integer and $0 < X < m_0$
X'	Pseudo master secret, $X' = X + Am_0$, where A is a random positive integer, and $0 < X' < M$, $M = \prod_{i=1}^{n} m_i$
K_{I,P_i}	Pre-shared key between initiator ***I*** and the proxy $\boldsymbol{P_i}$.
K_{R_r}	Public key used by the responder. And it is also the host identity, $K_{R_r} = R_r$.
K_{P_i}	Public key used by the proxy $\boldsymbol{P_i}$. And it is also the host identity, $K_{P_i} = P_i$.
SH_i	Secret shares of the master secret X , $SH_i \equiv X' (\mathrm{mod}\, m_i)$
X_r	Partial secret key, secretly generated and reconstructed in the *r*th communication session
X_r'	Pseudo partial secret key which is used to further derive the partial secret key X_r
$\boldsymbol{Y_r}$	Partial secret key generated by the responder, and it is also used to generate the final secret key.
$H(x_1, x_2)$	One-way hash function, x_1 and x_2 stand for the two different inputs of the hash function
$h_{i,r}$	Pseudo-random number generated by P_i in the *r*-th session using one-way hash function, $h_{i,r} = H(r, SH_i)$.

$Sh_{i,r}^{j}$	Sub-share generated by P_i in the *r*th session, and will be sent to P_j
Sh_{r}^{j}	Reconstruction share generated by P_j for secret S_r

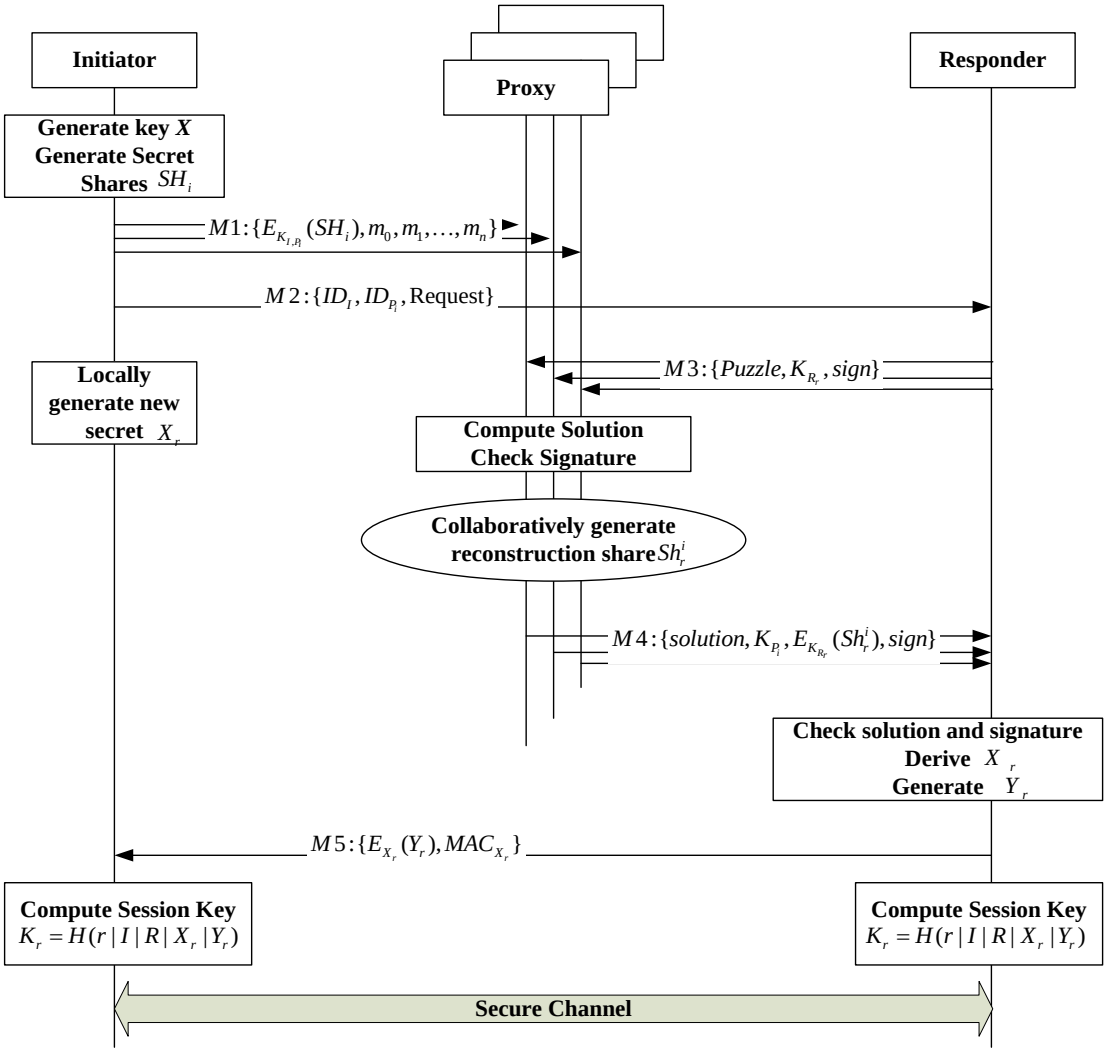

Figure 40 Proposed HIP Multi-Secret Sharing Based Key Exchange (HIP-MEX) Scheme

HIP-MEX-CRT is proposed to further reduce the workload on extremely resource constrained devices. Instead of distributing secret shares or vectors every time, HIP-MEX-CRT proposes to distribute key materials once for all the key inquiries in a long period. Within the same period, the session key will be derived from the same key materials, and each session key will be unique and

independent from another. Every session key will have its own expiration time, which will further enhance the security of HIP-MEX-CRT.

HIP-MEX-CRT can be divided into several phases, and we will give a detailed description of each phase in the following discussion.

Installation Phase

There are two ways to assign proxy to the initiator.

The first one is to privately assign a secret symmetric key to each pair of the devices before the deployment of the system, so initiator can "recognize" its proxy through the verification of the knowledge of the pair-wise secret key.

The second one is more flexible and costly. A trusted third party will assign a HIP host identity to all the devices. After the deployment, the initiator can use the HIP protocol to authenticate each other and then use the Diffie-Hellman key exchange scheme to share a secret key. The Diffie-Hellman key exchange is only performed a limited number of time.

Here, in order to make the system simple, we propose to use the first way for an initiator to select and communicate with the proxy.

The initiator randomly selects a large positive integer $\boldsymbol{X}$ as the master key. In the meantime, it secretly selects random large prime numbers $m_0, m_1, m_2, ..., m_n$.

Then, the initiator generates the shares of the master key $\boldsymbol{X}$; those shares will be used as key materials later.

$$SH_i \equiv X' (\mathrm{mod}\, m_i) \qquad (5.3.1)$$

Master shares (m_i, SH_i) with $i = 1, 2, \ldots, n$ will be encrypted using symmetric encryption techniques. Here we propose to use AES-256 to provide sufficient secrecy and efficiency.

Then the initiator constructs the ***M1*** message.

Initialization Phase

When the whole system starts to operate, the initiator initiates a new communication session with responder $\boldsymbol{R_r}$ by sending the inquiry message ***M2***.

Message ***M2*** contains the HIP host identities of the initiator and all the proxy selected. And it also contains the inquiry command to initiate a new communication session as well as the underlying communication protocol and security techniques.

Upon receiving ***M2***, the responder needs to decide whether it wants to communicate with the initiator or not. If it does not, the responder will just stop responding and the initiator will wait until the request times out. If it does, the responder will generate message ***M3*** for each proxy selected and informed by the initiator.

The *Puzzle* is generated distinctly for every proxy, and the signature *sign* is generated by the responder's private key which is paired with the public key K_{R_r}. So the proxy will be able to authenticate the responder on behalf of the initiator.

Key Material Generation Phase

The key is generated in parallel by both the initiator and the responder using different methods.

The initiator avoids most communicational and cryptographic operations; it computes the partial secret key X_r locally based on the key shares distributed during installation.

$$X_r' \equiv (\sum_{i=1}^{n} SH_i \cdot h_{i,r}) (\mathrm{mod}\, M) \qquad (5.3.2)$$

$$X_r \equiv X'_r \left(\mathrm{mod}\, m_0\right) \tag{5.3.3}$$

X_r is the partial secret and will be kept, so in this phase, the initiator will not generate message.

The proxies only have part of the key generation materials, and they will help the responder to finally derive the secret. The proxy $\boldsymbol{P_i}$ will first check the signature created by the responder to authenticate the received message, and then compute the solution for the received *Puzzle*; finally, it will perform the following operations to generate the sub-shares:

$$Sh_{i,r}^{j} \equiv SH_i \cdot h_{i,r} (\mathrm{mod}\, m_j) \tag{5.3.4}$$

$Sh_{i,r}^{j}$ are the sub-shares generated by P_i in the *r*-th session and will be sent to P_j. Messages sent between proxy are not shown in Figure 40 because of the limitation of the space of the figure. However, they are necessary for completing the whole process. Every message is encrypted by the symmetric key pre-shared between every pair of them, and since only the legitimate node possesses the secret key, there is no need for authentication using public key infrastructure.

After the proxy have received all the sub-shares, they will then add all their received sub-shares together to generate the reconstruction share, and for proxy P_i, the reconstruction share would be as follows:

$$Sh_r^i \equiv \left(\sum_{j=1}^{n} Sh_{j,r}^{i} \right) \left(\mathrm{mod}\, m_i\right) \tag{5.3.5}$$

Each proxy will construct their own message ***M4***, and deliver it to the responder with their own signature.

Note: ***M4*** contains the solution for the received Puzzle, the proxy's own public key which is also the HIP host identity, sub-share encrypted using the responder's public key, and a signature generated by the proxy's private key which is paired with the public key K_{P_i}.

Key Reconstruction Phase

The responder performs the key reconstruction after it received all the ***M4*** messages. It will decrypt all the messages first, and then check the solutions and the signatures. If all the solutions and signatures match, the responder will then select ***t*** reconstruction shares randomly and derive the partial key X_r.

Assuming the selected shares are $\{Sh_r^1, Sh_r^2, ..., Sh_r^t\}$, let $M_r = \sum_{i=1}^{t} m_i$, $M_{i,r} = M_r / m_i$, $y_{i,r} = M_{i,r}^{-1} (\mathrm{mod}\, m_i)$. Then, based on CRT, R_r can compute the unique solution:

$$X_r' \equiv (\sum_{i=1}^{t} Sh_r^i \cdot M_{i,r} \cdot y_{i,r}) (\mathrm{mod}\, M_r) \tag{5.3.6}$$

$$X_r \equiv X_r' (\mathrm{mod}\, m_0) \tag{5.3.7}$$

Then the responder will randomly select a new large positive integer $\boldsymbol{Y_r}$, and use this new integer as the other partial key. Then the responder will generate message ***M5*** and send it to the initiator directly.

Since the legitimate proxy have already authenticated the responder on behalf of the initiator, if the responder possesses the knowledge of $\boldsymbol{X_r}$, the initiator will consider the responder legitimate.

Then the initiator receives the message ***M5*** and decrypts it, and both the initiator and the responder will perform the final reconstruction at the same time using a one-way hash function:

$$K_r = H(r \mid I \mid R \mid X_r \mid Y_r) \tag{5.3.8}$$

So, both entities are able to encrypt all their further communications using symmetric encryption techniques and the session key K_r. Here, we also propose to use AES-256.

5.4 Comparison and Performance Analysis

In this section, we will compare the three schemes in terms of the computation and communication energy costs of the initiator under the following assumptions:

- Communication protocol is IEEE 802.15.4 (Zigbee) with four operating modes: transmit, receive, listen and sleep. The power consumption for different modes is also different [80].
- The initiator will switch to deep sleep mode after finishing the transmission/reception
- We use TelosB sensor which has 16-bit MSP430 microcontroller running under 4MHz. The claimed data rate during transmission is 250kbps.
- All three presented schemes use the same underlying cryptographic techniques to generate random numbers and encrypt/decrypt messages.

The power consumption of TelosB at 4MHz with a transmission power of -5DBM [80] is shown in Table 4.

Table 4 Power Consumption of TelosB at 4MHz with A Transmission Power of -5dBM

Power Mode	Power Consumption
Transmit	54 mW
Receive	61 mW
Listen	60 mW
Sleep	35 μW
Compute	4.8 mW

5.4.1 Comparison of Computational Workload

Denote the computational cost for a hash operation as E_H /byte, for symmetric encryption E_E /byte, for symmetric decryption E_D /byte, for all other computational operations E_O. We can also assume the X_i in TD-HIP and the SH_i in HIP-MEX-LPI and HIP-MEX-CRT approximately have the same length L. Then we can calculate the energy cost theoretically using those parameters. We did not take MAC generation and verification into account.

For TD-HIP initiator:

$$E_{T-D-HIP} = L\bullet(n\bullet E_H + n\bullet E_E + 2n\bullet E_D + E_H) + E_O \tag{5.4.1}$$

For HIP-MEX-LPI initiator:

$$E_{HIP-MEX-LPI} = \frac{L}{n-t+1}[n\bullet t\bullet(E_H + E_E)] + L\bullet(E_H + 2n\bullet E_D) + E_O \tag{5.4.2}$$

For HIP-MEX-CRT initiator:

$$E_{HIP-MEX-CRT} = L\bullet(n\bullet E_H + E_D + E_H) + E_O \tag{5.4.3}$$

Assume we set $n = 2t - 2$, which means $t = \frac{n}{2} + 1$, and t is given the least accept value to ensure that the majority of the proxy need to be compromised to breach the system. Then we can get the following functions:

$$E_{T-D-HIP} - E_{HIP-MEX-LPI} = -2L \times (E_H + E_E) \tag{5.4.4}$$

$$E_{T-D-HIP} - E_{HIP-MEX-CRT} = L\bullet[n\bullet E_E + (2n-1)\bullet E_D] \tag{5.4.5}$$

Apparently, HIP-MEX-LPI do not reduce the energy cost on the initiator side in terms of computational workload, however, HIP-MEX-CRT successfully avoid most of the encryption and decryption operations required in TD-HIP, and when the scale of the system becomes larger, the more energy will be saved.

5.4.2 Comparison of Communicational Workload

We denote that L_H is the byte length of packet header, L_D is the length of useful content, R is the data transfer rate. And from Table 4, we can get the power consumptions of transmission and reception of TelosB at 4MHz are 54 mW and 61 mW, respectively.

For TD-HIP:

$$E_{T-D-HIP} = \frac{(L_H + L_D) + (L_H + n \cdot L_D)}{R} \cdot 54 + \frac{(L_H + L_D) + n \cdot (L_H + L_D)}{R} \cdot 61 \qquad (5.4.6)$$

For HIP-MEX-LPI:

$$E_{HIP-MEX-LPI} = \frac{54ntL_D}{R(n-t+1)} + \frac{54(L_H + L_D + L_H + nL_D)}{R} + \frac{61[(L_H + L_D) + n(L_H + L_D)]}{R} \qquad (5.4.7)$$

For HIP-MEX-CRT:

$$E_{HIP-MEX-CRT} = \frac{L_H + L_D}{R} \cdot 54 + \frac{L_H + L_D}{R} \cdot 61 \qquad (5.4.8)$$

The heavy computation operations involved in HIP-MEX, such as authentication and secret reconstruction, are mainly happening on the proxy and the responder's side. Here the number 54 and 61 are from Table 4 Power Consumption of TelosB at 4MHz with A Transmission Power of -5dBM

As presented in Figure 40, after the completion of the installation phase, the initiator only needs to locally execute simply hash computation and arithmetical operations in order to obtain the final secret key. Further communication between the initiator and the responder can be then encrypted using symmetric encryption techniques. Comparing to the previous schemes, HIP-MEX-CRT successfully offload most of the workload for the resource constrained initiator during the key exchange process.

5.4.3 Simulation of HIP-MEX-CRT

We implemented the CRT-based key exchange scheme for HIP to evaluate the performance. We use C language (compiled with GNU MP Bignum Library) and OpenSSL to implement the computational operations, including all the cryptographic operations such as hash function and symmetric encryption. The communication simulation is done by OMNET++. We set the total number of proxy to be 8 and the threshold to be 5 (n = 8 and t = 5) so that $n = 2(t-1)$.

CPU time for executing all the operations on the testing Linux machine (3.6GHz AMD processor) is recorded, then the total clock cycles the program required to execute on a TelosB sensor board with TI CC2420 (a true single-chip 2.4 GHz IEEE 802.15.4 compliant RF transceiver) can be derived from the following equations:

$$CPU\_time = \frac{instructions}{program} \times \frac{clock\_cycles}{instruction} \times \frac{\text{sec}onds}{clock\_cycle} \tag{5.4.9}$$

$$E_{TelosB} = \frac{U_{TesloB} \bullet I_{TelosB}}{N_{TelosB}} \bullet \frac{\text{Re}gister\_size_{processor}}{\text{Re}gister\_size_{TelosB}} \bullet \alpha \bullet C_{processor} \tag{5.4.10}$$

Here, $C_{processor}$ stands for the number of cycles we get from the powerful processor running our implementations of the schemes. α represents the richer instructions of our powerful processor, which is approximately 2. And the $U_{TesloB} \bullet I_{TelosB}$ is the power consumption of the TelosB sensor when it is computing. N_{TelosB} stands for the frequency of the microcontroller of TelosB sensor.

The computation testing result for HIP-MEX-CRT and Elliptic Curve Diffie Hellman (ECDH) key exchange is shown in Table 5, and the overall energy cost for HIP-MEX-CRT is presented in Table 6. The testing results of HIP-MEX-CRT are for the resource constrained initiators. In order to make the result more intuitive, we use energy instead of just CPU time for the evaluation.

Table 5 Results of Computational Energy Cost Simulation

Scheme	HIP-MEX-CRT	ECDH
CPU Time (nS)	19350.75	1632602.18
Energy Cost (mJ)	0.668762	56.422731

Table 6 Total Energy Cost of HIP-MEX-CRT

Energy Cost (mJ)	Computation	Communication	Total
HIP-MEX-CRT	0.668762	0.416616	1.085378

Chapter 6

Conclusion and Future Work

In Chapter 1, we introduced the basics of WSNs and IoT, the security and privacy preserving issues in these two areas, as well as an overview of secret sharing schemes used to share and distribute secrets securely. Throughout the introduction, we explained the application areas and the motivation of this thesis work, the main focus of this thesis work is using multi-secret sharing techniques to reduce and outsource the workload of the resource constrained devices deployed in different applications and scenarios.

Chapter 2 provided a brief introduction to the basic mathematics and various cryptographic techniques related to the previous works and our work. We also reviewed some previous related works in the area of Key management scheme in WSNs and Key agreement scheme in IoT.

In Chapter 3, we proposed our own CRT-based efficient (n,t,n) multi-secret sharing scheme with two verifiable methods introduced. The use of CRT reduces the computational expense compared to Lagrange Polynomial Interpolation, and by allowing the proxy and the dealer to compute the one-way hash function locally, our scheme further omits the requirement of obtaining the n-tuple weight vector from the dealer and the periodically updating of the secretly distributed polynomials. The computational costs of our scheme have been carefully simulated and tested using C and OpenSSL. The results show that under expected system settings, the proposed scheme successfully offloaded more workloads comparing to the previous works. The efficiency of this scheme makes it possible to be used in multiple applications such as Internet of

Things and WSNs. However, in order to adapt this scheme to various applications, further security enhancements and performances improvements should be done as well.

In Chapter 4, we presented our flexible WSN key management scheme based on (*n*,*t*,*n*) multi-secret sharing scheme. All the sensor nodes within this system only need to remember information associated with several key managers, and the key managers will provide the sensors communication session keys after deployment according to the requirements of applications. This scheme is designed for heterogeneous WSNs with two types of communication architecture: distributed and hieratical. While more powerful and replaceable key managers are responsible for the most communication and computation costs, resource constrained sensors offload most of the workload and can work more efficiently. The testing of computational and communicational costs of the scheme under different system settings and deployment environments will be done on real sensor devices in future work.

As Internet of Things becomes more and more popular and available around world, security and energy efficiency are two important issues. In Chapter 5, we present two new key agreement schemes for HIP based IoT using a distributed approach. Those two schemes are built on LPI-based (*n*,*t*,*n*) multi-secret sharing and CRT-based (*n*,*t*,*n*) multi-secret sharing, respectively. While the LPI-based key agreement scheme failed to improve the energy efficiency, the CRT-based key agreement scheme successfully reduced the energy cost for both computation and communication operations. Most of the workload has been outsourced to more powerful and resourceful proxy.

With the proxy collaboratively helping the initiator during the key exchange process, the energy consuming and complex public key encryptions can be avoided on the initiator. This

allows two devices with significantly different resources to be able to communicate secretly. And because of the low power consumption, the lifetime of the initiator can also be extended. In future work, we will further strengthen the security of the proposed HIP-MEX-CRT.

Reference

[1] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions,* pp. 644-654, 22 6 1976.

[2] J. Yick, B. Mukherjee and D. Ghosal, "Wireless Sensor Network Survey," *Computer networks,* vol. 52, no. 12, pp. 2292-2330, 2008.

[3] K. Maraiya, K. Kant and N. Gupta, "Wireless Sensor Network: A Review on Data Aggregation," *International Journal of Scientific & Engineering Research,* vol. 2, no. 4, pp. 1-6, 2011.

[4] I. Howitt and J. Gutierrez, "IEEE 802.15. 4 low rate-wireless personal area network coexistence issues," *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE,* vol. 3, pp. 1481-1486, 16-20 March 2003.

[5] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards," *Computer communications,* vol. 30, no. 7, pp. 1655-1695, 2007.

[6] ANT Wireless, Dynastream Innovations Inc, [Online]. Available: http://www.thisisant.com.

[7] S. Adibi, "Link Technologies and BlackBerry Mobile Health (mHealth) Solutions: A Review," *Information Technology in Biomedicine, IEEE Transactions on,* pp. 586-597, 2012.

[8] M. Dustin, J. Shankarappa, M. Petrowski, H. Weerasinghe and H. Fu, "Analysis of key management in wireless sensor networks," *Electro/Information Technology, 2007 IEEE International Conference on,* pp. 263-271, 2007.

[9] L. Coetzee and J. Eksteen, "The Internet of Things-promise for the future? An introduction," *IST-Africa Conference Proceedings, 2011,* pp. 1-9, 2011.

[10] V. Potdar, A. Sharif and E. Chang, "Wireless sensor networks: A survey," *Advanced Information Networking and Applications Workshops, 2009. WAINA'09. International Conference on.,* pp. 636-641, 2009.

[11] G. Wu, S. Talwar, K. Johnsson, N. Himayat and K. D. Johnson, "M2M: From mobile to embedded internet," *Communications Magazine,IEEE,* vol. 49, no. 4, pp. 36-43, 2011.

[12] Y. Ben Saied and A. Olivereau, "HIP Tiny Exchange (TEX): A distributed key exchange scheme for HIP-based Internet of Things," *Communications and Networking (ComNet), 2012 Third International Conference on,* pp. 1-8, 2012.

[13] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," *Computer networks,* vol. 54, no. 15, pp. 2787-2805, 2010.

[14] F. Villanueva, D. Villa, F. Moya, M. Santofimia and J. Lopez, "Internet of Things Architecture for an RFID-Based Product Tracking Business Model," *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on,* pp. 811-816, 4-6 July 2012.

[15] D. Manivannan and P. Neelamegam, "WSN: key issues in key management schemes—a review," *Research Journal of Applied Science, Engineering and Technology,* vol. 4, no. 18, pp. 3188-3200, 2012.

[16] N. Liu, J. Chen, L. Zhu and J. Zhang, "A key management scheme for secure communications of advanced metering infrastructure in smart grid," *Industrial Electronics, IEEE Transactions on,* vol. 60, no. 10, pp. 4746-4756, 2013.

[17] F. Al-Shraideh, "Host identity protocol," *Networking, International Conference on Systems and*

International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on, p. 203, 23-29 April 2006.

[18] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[19] G. R. Blakley, "Safeguarding cryptographic keys," *in National Computer Conference,1979*, vol. 48, pp. 313-317, 1979.

[20] M. Mignotte, "How to Share a Secret," *Proceedings of the 1982 conference on Cryptography*, pp. 371-375, 1982.

[21] W. Stallings, Cryptography and Network Security, 4 ed., Pearson Education India, 2006.

[22] C. A. Asmuth and J. Bloom, "A Modular Approach to Key Safeguarding," *IEEE Transactions on Information Theory*, Vols. IT-29, no. 2, pp. 208-210, 1983.

[23] M. Quisquater, B. Preneel and J. Vandewalle, "On the Security of the Threshold Scheme Based on the Chinese Remainder Theorem," *Public Key Cryptography*, pp. 199-210, 2002.

[24] J. He and E. Dawson, "Multistage Secret Sharing Based on One-way Function," *Electronics Letters*, vol. 30, no. 19, pp. 1591-1592, 1994.

[25] J. He and E. Dawson, "Multisecret-sharing Scheme Based on One-way Function," *Electronics Letters*, vol. 31, no. 2, pp. 93-95, 1995.

[26] H. Y. Lin and Y. S. Yeh, "Dynamic Multi-secret Sharing Scheme," *International Journal of Contemporary Mathematical Sciences*, vol. 3, no. 1, pp. 37-42, 2008.

[27] T. Chang, M. Hwang and W. Yang, "A New Multi-stage Secret Sharing Scheme Using One-way Function," *ACM SIGOPS Operating Systems Review*, vol. 39, no. 1, pp. 48-55, 2005.

[28] L. Harn and C. Lin, "Strong (n,t,n) Verifiable Secret Sharing Scheme," *Information Sciences*, vol. 180, no. 16, pp. 3059-3064, 2010.

[29] Y. Liu, L. Harn, C. Yang and Y. Zhang, "Efficient (n,t,n) Secret Sharing Schemes," *Journal of Systems and Software*, vol. 85, no. 6, pp. 1325-1332, June 2012.

[30] R. Roman, C. Alcaraz, J. Lopez and N. Sklavos, "Key management systems for sensor networks in the context of the Internet of Things," *Computers & Electrical Engineering*, vol. 37, no. 2, pp. 147-159, 2011.

[31] S. Mekid, “Further structural intelligence for sensors cluster technology in manufacturing,” *Sensors 6*, No. 6, pp. 557-577, 2006.

[32] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson, "Host identity protocol," *RFC5201*, April 2008.

[33] R. Moskowitz, "Hip diet exchange (dex)," *draft-moskowitz-hip-dex-00 (WiP), IETF*, 2012.

[34] T. Heer, "LHIP Lightweight Authentication Extension for HIP," *draft-heer-hip-lhip-00 (IETF work in progress)*, 2007.

[35] Y. B. Saied and A. Olivereau, "D-HIP: A distributed key exchange scheme for HIP-based Internet of Things," *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pp. 1-7, 25-28 June 2012.

[36] Y. Ben Saied and A. Olivereau, "(k, n) threshold distributed key exchange for hip based internet of things," *Proceedings of the 10th ACM international symposium on Mobility management and wireless access*, pp. 79-86, October 2012.

[37] Y. Rao and C. Bhagvati, "CRT Based Threshold Multi-secret Sharing Scheme," *International Journal of Network Security,* vol. 16, no. 4, pp. 249-255, July 2014.

[38] W. Wen, B. Vaidya, D. Makrakis and C. Adams, "Decentralized CRT-Based Efficient Verifiable (n, t, n) Multi-secret Sharing Scheme," *Foundations and Practice of Security,* pp. 279-293, 2014.

[39] R. C. Merkle, "Method of providing digital signatures". US patent 4,309,569, 5 January 1982.

[40] H. Feistel, "Cryptography and computer privacy," *Scientific american,* vol. 228, pp. 15-23, 1973.

[41] K. S. Rao, M. R. Krishna and D. Babu, "Cryptanalysis of a Feistel Type Block Cipher by Feed Forword Neural Network Using Right Sigmoidal Signals," *International Journal of Soft Computing,* vol. 4, no. 3, pp. 131-135, 2009.

[42] P. Van De Zande, "The Day DES Died," *SANS Institute*, July, 2001.

[43] A. Al-Vahed and H. Sahhavi, "An overview of modern cryptography," *World Applied Programming,* vol. 1, no. 1, pp. 3-8, 2011.

[44] D. S. A. Elminaam, H. M. Abdual-Kader and M. M. Hadhoud, "Evaluating The Performance of Symmetric Encryption Algorithms," *IJ Network Security,* vol. 10, no. 3, pp. 216-222, 2010.

[45] N.-F. Standard, "Announcing the Advanced Encryption Standard (AES)," *Federal Information Processing Standards Publication 197,* 2001.

[46] M. Shakiba, M. Dakhilalian and H. Mala, "On computational complexity of impossible differential cryptanalysis," *Information Processing Letters,* vol. 114, no. 5, pp. 252-255, 2014.

[47] W. Zhang, W. Wu and D. Feng, "New results on impossible differential cryptanalysis of reduced AES," *Information Security and Cryptology-ICISC 2007, Lecture Notes in Computer Science,* vol. 4817, pp. 239-250, 2007.

[48] I. C. Paar and I. J. Pelzl, "Introduction to Public-Key Cryptography," in *Understanding Cryptography*, Springer Berlin Heidelberg, 2010, pp. 149-171.

[49] R. L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM,* vol. 21, no. 2, pp. 120-126, 1978.

[50] T. Elgamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *Information Theory, IEEE Transactions on,* vol. 31, no. 4, pp. 469-472, July 1985.

[51] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation,* vol. 48, no. 177, pp. 203-209, 1987.

[52] V. S. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology—CRYPTO'85 Proceedings,* pp. 417-426, January 1986.

[53] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture," *RFC 4251, IETF,* Jan. 2006.

[54] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Transport Layer Protocol," *RFC 4253, IETF,* Jan. 2006.

[55] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) Connection Protocol," *RFC 4254, IETF,* Jan. 2006.

[56] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) protocol version 1.2," *RFC 5246, IETF,* August 2008.

[57] K. Seo and S. Kent, "Security Architecture for the Internet Protocol," *RFC 4301, IETF,* December 2005.

[58] WIKIPEDIA, [Online]. Available: http://en.wikipedia.org/wiki/Chinese_remainder_theorem.

[Accessed 19 August 2014].

[59] WIKIPEDIA. [Online]. Available: http://en.wikipedia.org/wiki/Modular_multiplicative_inverse. [Accessed 21 August 2014].

[60] G. L. Miller, "Riemann's hypothesis and tests for primality," *Journal of computer and system sciences,* vol. 13, no. 3, pp. 300-317, 1976.

[61] M. O. Rabin, "Probabilistic algorithm for testing primality," *Journal of number theory,* vol. 12, no. 1, pp. 128-138, 1980.

[62] J. Herranz, A. Ruiz and G. Sáez, "New results and applications for multi-secret sharing schemes," *Designs, Codes and Cryptography,* pp. 1-24, 2013.

[63] A. Waseda and M. Soshi, "Consideration for multi-threshold multi-secret sharing schemes," *Information Theory and its Applications (ISITA), 2012 International Symposium on,* pp. 265-269, October 2012.

[64] M. A. Simplício Jr, P. S. Barreto, C. B. Margi and T. C. Carvalho, "A survey on key management mechanisms for distributed wireless sensor networks," *Computer Networks,* vol. 54, no. 15, pp. 2591-2612, 2010.

[65] J. Kuriakose, S. Joshi, R. V. Raju, A. Kilaru, "A review on localization in wireless sensor networks," *Advances in signal processing and intelligent recognition systems,* pp. 599-610, 2014.

[66] A. Barati, M. B. H. Dehghan and A. A. Mazreah, "Key management mechanisms in wireless sensor networks," *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on,* pp. 81-86, August 2008.

[67] Z. Specification, "ZigBee Document 053474r06 Version 1.0.," *ZigBee Alliance,* December 2004.

[68] S. Zhu, S. Xu, S. Setia and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach," *Network Protocols, 2003. Proceedings. 11th IEEE International Conference on,* pp. 326-335, November 2003.

[69] C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung, "Perfectly-secure key distribution for dynamic conferences," *Advances in cryptology—CRYPTO'92,* pp. 471-486, January 1993.

[70] W. Chunying, L. Shundonga and Z. Yiying, "Key Management scheme based on secret sharing for Wireless Sensor Network," *Emerging Intelligent Data and Web Technologies (EIDWT), 2013 Fourth International Conference on,* pp. 574-578, September 2013.

[71] U. M. Maurer, "Secret key agreement by public discussion from common information," *Information Theory, IEEE Transactions on,* vol. 39, no. 3, pp. 733-742, 1993.

[72] Y. Wang, "Efficient identity-based and authenticated key agreement protocol," *arXiv preprint arXiv:1207.5438,* 2012.

[73] A. Shamir, "Identity-based cryptosystems and signature schemes," *Advances in cryptology,* pp. 47-53, January 1985.

[74] R. Moskowitz and P. Nikander, "Host Identity Protocol (HIP) Architecture," *RFC 4423,* May 2006.

[75] L. Harn and C. Lin, "Detection and Identification of cheaters in (t, n) secret sharing scheme," *Designs, Codes and Cryptography,* vol. 52, no. 1, pp. 15-24, 2009.

[76] H. Ghodosi, "Comments on Harn–Lin's cheating detection scheme," *Designs, Codes and Cryptography,* vol. 60, no. 1, pp. 63-66, 2011.

[77] K. Kaya and A. A. Selçık, "Threshold Cryptography Based on Asmuth–Bloom Secret Sharing," *Information Sciences*, vol. 177, no. 19, pp. 4148-4160, 2007.

[78] T. Granlund, “GMP, the GNU multiple precision arithmetic library,” 2014.

[79] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, Elsevier, 2011.

[80] G. d. Meulenaer, F. Gosset, F.-X. Standaert and O. Pereira, "On the energy cost of communication and cryptography in wireless sensor networks," *IEEE International Conference on Wireless and Mobile Computing,Networking and Communications (WIMOB 2008),* pp. 580-585, October 2008.