AN EFFICIENT KEY MANAGEMENT SCHEME FOR WIRELESS SENSOR NETWORK

A Thesis submitted in fulfillment of the requirement
for the award of degree
of
DOCTOR OF PHILOSOPHY

by

Suman Bala
(900903006)

Under the guidance of
Dr. Anil K. Verma
Associate Professor
Computer Science and Engineering Department

Computer Science and Engineering Department
Thapar University, Patiala - 147004, INDIA

July 2014

Certificate

I hereby certify that the work being presented in this thesis entitled "**An Efficient Key Management Scheme For Wireless Sensor Network**", for the award of degree of "**Doctor of Philosophy**" submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Dr. Anil K. Verma and refers other researchers works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Suman Bala.

(Suman Bala)
900903006

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Anil K. Verma

(Dr. Anil K. Verma)
Associate Professor
Computer Science and Engineering Department
Thapar University
Patiala

Abstract

Wireless sensor networks (WSN) have been widely used in various applications. WSN consists of sensor motes, which are tiny, low power, computationally limited and battery constrained electromechanical devices. Deployment of such devices in the physical environment gives an opportunity to the attackers that could tamper devices, eavesdrop communications, alter transmitted data, or attach unauthorized devices to the network. To provide security for WSN it is necessary to encrypt the message sent among communicating nodes; for encryption/decryption network needs to manage a key. Hence, Key Management is a major challenge to achieve security in WSN. Key management with symmetric cryptography is inflexible as compared to Public Key Cryptography (PKC). For PKC, sensor motes need to agree on shared session key to establish a secure communication. However, achieving such key agreement in a resource-constrained environment is not a trivial task, as security protocols always require additional overhead on the computational, storage and energy resources. In such a scenario, the variants of PKC are better options, as traditional PKC have some limitations, which are unsuitable for WSN. Identity Based Cryptography (IBC) and CertificateLess Public Key Cryptography (CL-PKC) are advantageous in terms of computation cost and storage cost.

The contributions of this thesis to the area of key management in WSN are manifold. In this thesis, we present a historical perspective and brief introduction of key management in WSN. We present an overview of WSN, security requirements for WSN. We also present, the literature review of symmetric key management schemes especially designed for WSN. This thesis also reviews the literature of asymmetric key agreement protocols. Based on the literature review, we propose four protocols. The proposed protocols are as follows:

(i) A Pairing-Free Identity-based Two-Party Authenticated Key Agreement (PF-ID-2PAKA) protocol for WSN.

(ii) Breaking of CertificateLess Key Agreement Protocol against Key-Compromise Impersonations attack and providing a viable solution.

(iii) A Non-Interactive Certificateless Two-Party Authenticated Key Agreement (NI-CTAKA) Protocol for WSN.

(iv) An Improved Forward Secure Elliptic Curve Signcryption Key Management Scheme for WSN.

The first and fourth protocols are based on IBC, the second and third protocols are based on CL-PKC. PF-ID-2PAKA, the improved protocol and NI-CTAKA are based on

the hardness assumption of Gap Diffie-Hellman (GDH) and proven secure in eCK model for IBC and CL-PKC presented by Liang et al. and Lipold et al. respectively. For the verification and validation of these protocols, we implemented them on the experimental setup, which includes the MICAz mote, TinyOS, RELIC-Toolkit and AVRORA. We also perform the comparative analysis of these protocols with the existing schemes based on the computation cost, which includes the important operations like scalar point multiplication, point addition, etc., communication cost, running time, energy consumption and storage cost.

We cryptanalyze the existing Certificateless Two-Party Authenticated Key Agreement (CTAKA) protocol against Key-Compromise Impersonation (K-CI) attack. We present a forward secure elliptic curve signcryption key management scheme for WSN. The scheme is an improvement in the existing scheme.

Keywords: Cryptography, WSN, Authenticated Key Agreement, Signcryption, Identity Based Two-Party Authenticated Key Agreement (ID-2PAKA) protocol, Certificate-Less Two-party Authenticated Key Agreement (CTAKA) protocol.

Acknowledgements

First of all, I express my gratitude to the Almighty, Who blessed me with the zeal and enthusiasm to complete this work successfully. I am extremely thankful to my supervisor, Dr. Anil K Verma, Associate Professor, Thapar University, Patiala for his suggestions and constant support during this research. I am grateful to him for motivating and inspiring me to explore deeply the field of Wireless Sensor Network and supporting me throughout the research cycle of my Ph.D. work. His wide knowledge and logical way of thinking have been a great value for me. Especially, the extensive comments, discussions and the interactions with Dr. Anil K. Verma had a direct impact on the final form and quality of my Ph.D. work.

I am also thankful to Dr. Deepak Garg, Head of Computer Science and Engineering Department, for his guidance through the early years of chaos and confusion. I am thankful to my Doctoral committee members Dr. Deepak Garg, Dr. V. P. Singh, Dr. Kulbir Singh and Dr. O.P. Pandey, Dean of Research and Sponsored Projects for their constructive comments and ensuring the progress of my research work in right direction. My deep regards to Dr. Prakash Gopalan, Director, Thapar University for all the facilities, which have been immensely helpful for the completion of my work. I extend my thanks to Dr. Inderveer Chana and Dr. Neeraj Kumar for their generous time and invaluable suggestions throughout the research work. I wish to thank the faculty and staff members of Computer Science and Engineering Department of Thapar University, Patiala for their co-operation and support. I would particularly thank all my lab mates and friends especially Sahil Singla, Kuldeep Sharma and Neha Kapoor for their constant support.

Special thanks to my best friend and colleague Gaurav Sharma (now, Dr.), for providing me constant support and encouraging me during low times. I would like to thank Diego F. Aranha and Conrado, the lead developers of the RELIC library, for their generous and timely help in using and understanding the code for implementation purpose.

This thesis would not have been possible without the support of my family. My deep regards to my Parents Mrs. Shashi Bala and Mr. Harnek Singh for their patience, affection and blessings. Their lessons on honesty, ethics, humbleness and patience have always amazed me. I would also thank my siblings, Mr. Jasdeep and Miss Ramandeep for their untiring love, companionship, and support.

Contents

List of Figures

List of Tables

Abbreviations

AKA	**A**uthenticated **K**ey **A**greement
BP	**B**ilinear **P**airing
CA	**C**ertificate **A**uthority
CDHP	**C**omputational **D**iffie-**H**ellman **P**roblem
CL-PKC	**C**ertificate**L**ess - **P**ublic **K**ey **C**ryptography
CTAKA	**C**CertificateLess Key Agreement **T**wo-party **A**uthenticated **K**ey **A**greement
DDHP	**D**ecisional **D**iffie-**H**ellman **P**roblem
DHP	**D**iffie-**H**ellman **P**roblem
DLP	**D**iscrete **L**ogarithm **P**roblem
ECC	**E**lliptic **C**urve **C**ryptography
ECDLP	**E**liptic **C**urve **D**iscrete **L**ogarithm **P**roblem
GDHP	**G**ap **D**iffie-**H**ellman **P**roblem
ID	**ID**entity
IBC	**I**dentity **B**ased **C**ryptography
IBC	**I**dentity **B**ased **C**ryptography
ID-2PAKA	**I**dentity-based **T**wo **P**arty **A**uthenticated **K**ey **A**greement
KGC	**K**ey **G**eneration **C**enter
PKC	**P**ublic **K**ey **C**ryptography
PKG	**P**rivate **K**ey **G**enerator
PKI	**P**ublic **K**ey **I**nfrastructure
WSN	**W**ireless **S**ensor **N**etwork

Symbols

k	security parameter
$param$	system parameters
s	master secret key of PKG / KGC
P_{pub}	public key of PKG / KGC
ID_i	the identity of the user
$D_i = (s_i, R_i)$	partial-private-key of user identity ID_i generated by KGC in CL-PKC
s_i	secret key in IBC or secret-value chosen by KGC of user identity ID_i
R_i	public key in IBC or public-value computed by KGC of user identity ID_i
x_i	secret-value chosen by user with identity ID_i
P_i	public-key computed by user with identity ID_i
$sk_i = (s_i, x_i)$	private key of user identity ID_i
$pk_i = (R_i, P_i)$	public key of user identity ID_i
m	message
$U = ID_1, \cdots, ID_n$	set of parties with each party ID_i
$\mathcal{A}(\mathcal{A}_I/\mathcal{A}_I I)$	Adversary (Type I / Type II)
$\mathcal{C}$	Challenger, who responds Adversary's queries
$\mathbb{L}_i$	List maintained by Challenger
$\mathbb{G}$, $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$	the cyclic group composed of the points on $\mathbb{E}/\mathbb{F}_q$ of prime order $q \geq 2^k$
q, n	prime order of a group
P	a generator of group $\mathbb{G}_1$

$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$	bilinear pairing
H_0, H_1	Hash Functions
$\{0,1\}^*$	string with arbitrary length
$\{0,1\}^l$	string with length l
A	Alice
B	Bob
esk_A, t_A, epk_B, t_B	ephemeral-secret-key of Alice and Bob respectively
t_i, t_j	ephemeral-secret-key of party i with intended partner party j
epk_A, T_A, epk_B, T_B	ephemeral-public-key of Alice and Bob respectively
T_i, T_j	ephemeral-public-key of party i with intended partner party j
K_{AB}, K_{BA}	secret shared key by Alice and Bob respectively
sk	secret session key
$\prod_{i,j}^{s}$	the s^{th} session runs for party i with intended partner party j

List of Publications

I International Journals

1. **Suman Bala**, Anil K. Verma, "*A Non-Interactive Certificateless Two-Party Authenticated Key Agreement Protocol for Wireless Sensor Networks*", **International Journal of Ad Hoc and Ubiquitous Computing** (SCIE-Indexed, Impact Factor-0.511). (Accepted)
2. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*On the security of certificateless Signature Scheme Without Pairing*", **International Information Institute Journal**, Japan, Proceedings of the 6th International Conference on Information (Info'13), Vol. 16, No. 11, pp. 7827-7830, 2013 (Impact Factor - 0.36).
3. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*Classification of Symmetric Key Management Schemes for Wireless Sensor Networks*", **International Journal of Security and Its Applications**, Vol. 7, Issue 2, pp. 117-138, March 2013.
4. **Suman Bala**, Anil K. Verma, "*Impersonation Attack on CertificateLess Key Agreement Protocol*", **International Journal of Ad Hoc and Ubiquitous Computing** (SCIE-Indexed, Impact Factor-0.511). (Communicated)
5. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*PF-ID-2PAKA: Pairing Free Identity-based Two-Party Authenticated Key Agreement Protocol for Wireless Sensor Network*", **Algorithmica** (SCIE-Indexed, Impact Factor-0.567). (Communicated)
6. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*Feasibility of Public key cryptography in small-embedded devices*", **IEEE Security & Privacy**(SCI-Indexed, Impact Factor: 0.962). (Communicated)
7. Murari Mandal, Gaurav Sharma, **Suman Bala**, Anil K. Verma, *Feasibility of Public Key Cryptography in Wireless Sensor Network*, **Journal of Theoretical Physics and Cryptography**. (Accepted)

II Book Chapter

1. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*An Improved Forward Secure Elliptic Curve Signcryption Key Management Scheme for Wireless Sensor Networks*", Book Chapter in International Conference on IT Convergence and Security 2012 (ICITCS'12), **Lecture Notes in Electrical Engineering (LNEE, Springer)**, Vol. 215, pp. 141-149, (Dec 5-7, 2012, Republic of Korea), 2013.

II International Conferences

1. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*Optimized Elliptic Curve Cryptography for Wireless Sensor Networks*", in the Proceedings of 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC'12), pp. 89-94, Solan, Dec 6-8, 2012.
2. **Suman Bala**, Gaurav Sharma, Anil K. Verma, "*A Survey and Taxonomy of Symmetric Key Management Schemes for Wireless Sensor Networks*", in the Proceedings of the CUBE International Information Technology Conference (CUBE'12) ACM, pp. 585-592, 2012.

To my parents

Chapter 1

Introduction

Recent engineering advances especially in the field of communications lead to an emerging world of inexpensive and mobile devices, which not only solves the purpose of checking email and browsing on the go but also provides various kinds of useful applications like vehicles tracking, industrial production processing, environment conditions monitoring, patient health monitoring, battlefield surveillance etc.

Wireless Sensor Networks (WSN) [1–3] gained attention during last decade and enabled the development of low-powered sensor networks. Generally, WSN consists of a base station and large number of sensor motes. A typical sensor mote is equipped with integrated 8-bit microcontroller, radio transceiver, sensors such as photodiodes, thermistors, etc., 4KB of RAM, 128 KB of program space and a battery. The task of sensor mote is to gather, process and forward the physical information from the environment to the base station. WSN possess lots of security challenges. These are usually deployed in hostile areas. As sensor motes transmit information over the air, it attracts vulnerabilities, which can harm from malfunctioning of transmitted messages to physical capturing of motes.

Energy consumption is another important challenge in WSN, which has to be focused. Due to limited bandwidth available for communication and low battery life span of a sensor mote, wireless transmission is very expensive in terms of energy usage. There are various guidelines for providing security in WSN [4, 5], such as, fewer keys to store, communication and computation overhead should be less, and size of the key should be less. In order to maintain the integrity and authentication of the message, encryption and signcryption can be used. Key is used for the encryption/decryption of a message. So, efficient key management is one of the primary concerns. Key management involves three processes namely, key establishment, rekeying and revocation.

A key agreement is an approach to establish a common key between two or more parties in a cooperative manner. There are various key agreement protocols in literature but the security of these protocols is always open for discussion. Most of the protocols lack in providing satisfactory level of security. There are basically three types of key agreement protocols studied in typical network environments: trusted server protocols, public key protocols and pre-distribution protocols. Trusted servers have central servers for issuing certificates. Due to the complexity of the structure of trusted server mechanism for resource-constrained device, they are discarded as a security measure. Public key Cryptography (PKC) protocols use asymmetric cryptography, which is very costly in terms of computations than symmetric cryptographic algorithms and requires authentication of public-keys [6, 7]. To tag the user identity and the public-key of an identity, traditional Public Key Infrastructure (PKI) requires a trusted Certificate Authority (CA). The management of certificates consumes lots of storage as well as increases computation and communication overheads.

Symmetric-key based key agreement protocols were earlier popular in the field of WSN as they were associated with lesser computation overhead and simplicity in resource utilization. Symmetric-key based key agreement protocols are based on random key-pre-distribution, where a master key is stored prior to deployment of motes in the network environment. The master key is used to establish pairwise-keys between any pair of sensor motes. However, it does not maintain the resilience of the network. As, the sensor motes are tamper-resistant; if a single key is compromised then the entire network will be compromised. On the other hand, if $n-1$ pairwise-keys are stored on n motes, it assurances perfect resilience but is impractical for memory-constrained devices, if n is very large. There are various schemes based on symmetric-key cryptography that maintains the resilience and key connectivity by negotiating between the two. Hence, symmetric-key based key agreement protocols have major limitation of perfect connectivity between communicating parties. It does not guarantee scalability, as it requires new keying material to be added to each existing mote.

There are two variants of PKC, Identity Based Cryptography (IBC) and Certificate-Less Public Key Cryptography (CL-PKC), which are responsible to reduce number of keys in the network. Identity Based Cryptography (IBC) is a public key based concept, where public-keys are derived from the identities of motes, which are known in the network. Shamir [8] introduced the revolutionary Id-based infrastructure in 1984 to resolve the issues of PKC. This seminal concept of IBC allows the user to choose a public-key of its own choice such as email-id, phone number, name etc. In IBC, users do not generate their own private-key as in traditional PKC [9]. Private Key Generator (PKG) generates private-keys and maintains the private-keys of all the users but there is always a possibility of the misuse of these private-keys as they can be used to decrypt

any ciphertext and forge the signature of the user on any message for signature generation. Eventually, this new paradigm solved the problem of certificate management but engendered congenital key escrow problem.

In 2003, Al-Riyami and Paterson [10] proposed a novel approach to eliminate the congenital key escrow problem of IBC as well as the use of certificates in traditional PKC. This approach is known as CertificateLess Public Key Cryptography (CL-PKC), where Key Generation Centre (KGC) generates the partial-private-key for the user, while the user chooses its own secret-value. Then, the user collaborates its secret-value with partial-private-key generated by KGC to form a private-key. In other words, CL-PKC differs from IBC in terms of arbitrary public-key and when a signature is transmitted, user's public-key is attached with it but not certified by any of the trusted authority. Moreover, KGC is not aware about the secret-key of the user. In asymmetric approaches, security is always proportional to computation cost. For WSN, it is always vital to use less computation to increase the lifetime of the network.

1.1 Introduction to Wireless Sensor Networks

Computer systems are not capable for the perception of the physical world. Sensor network hardware tries to build the gap between the digital world to the physical world. A typical Wireless Sensor Networks (WSN) is a distributed and dynamic network, which consists of large number of small low-cost sensor motes having the capability of sensing, computation and wireless communication to base-station for the supervision of many promising applications limited from military applications to civilian applications.

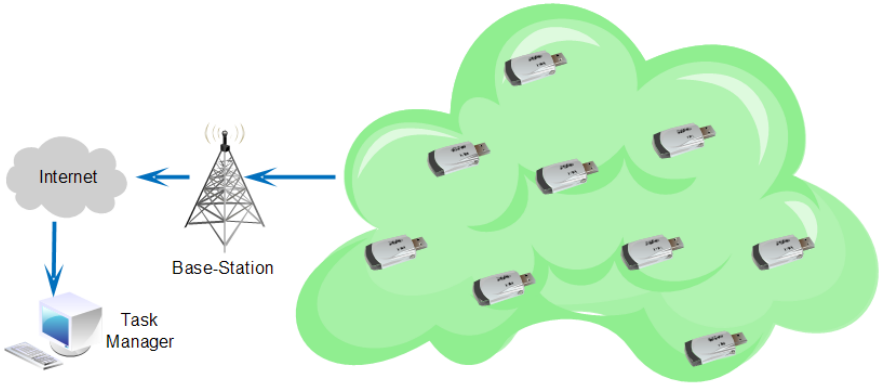

FIGURE 1.1: Components of Wireless Sensor Network

The base-station acts as a gateway for wired devices, which is used by the network operator to retrieve information from all sensor motes in the network. Figure 1.1 shows the diagrammatic view of a typical WSN. A node or mote of WSN, generally, consists of various kinds of sensors, microprocessor, memory unit, power unit and radio transceiver.

WSN was initiated as the research project by DARPA (Defense Advanced Research Projects Agency) in 1980 with a program known as Distributed Sensor Networks (DSN) [11]. Later, Carnegie Mellon University and MIT Lincoln Laboratory partnering the DSN project. In mid-1990s DARPA launched Low-power Wireless Integrated Microsensors (LWIM) project and SensIT project in 1998 for large distributed military sensor systems. Now, WSN is used for various industrial and commercial applications. WSN is suitable for applications (see figure 1.2) like military operations, precision agriculture, habitat monitoring, vehicular traffic management, environmental monitoring etc.

FIGURE 1.2: Applications of Wireless Sensor Network

1.1.1 Challenges in Wireless Sensor Networks

WSNs have unique characteristics, which makes them useful for different applications. As, there are lots of constraints involved with WSN, so the implementation of real time applications is itself a challenging task. The challenges in WSN are as follows:

1.1.1.1 Challenges in Sensor Motes

The sensor motes have limited amount of memory, processing and battery. On the contrary, all security approaches require a certain amount of resources for implementation.

However, currently these resources are very limited in sensor mote. The challenges in the sensors hardware are as follows:

- **Limited Memory:** A sensor mote is having a limited amount of memory and storage space. For effective security mechanism in WSN, it is necessary to limit the code size of the security algorithm.
- **Limited Processing:** The microprocessors used in sensor motes are of 8 bit or 16 bit. They are not powerful for the implementation of complex security algorithms.
- **Limited Energy:** This is the biggest challenge in WSN. The batteries of WSN cannot be replaced once deployed in remote areas. So, for the implementation of security algorithm, it is necessary to limit the communication and computation cost.
- **Tamper-Resistant Hardware:** A WSN is usually deployed in hostile areas without any fixed infrastructure. It is difficult to perform continuous surveillance after network deployment, as sensor mote is not temper resistant hardware.

1.1.1.2 Challenges in the Network

Sometimes, large numbers of sensor motes are deployed in the network (e.g. environmental application). It is difficult to place each mote at a specific position, rather, they are scattered randomly in the field. So, the network is infrastructure-less and self-organized. The challenges in the network are as follows:

- **Unreliable Communication:** In WSN, sensor motes communicate with each other wirelessly through radio interface configured at same frequency band. The adversary can monitor and participate in the network. Moreover, the packets received at the other end may get damaged due to channel errors or lack of radio coverage because the communication range of sensor mote is limited.
- **Random Topology:** As WSNs are self-organized and dynamic in nature and deployed in random distribution, so, they do not follow a topology beforehand. Due to limited battery, the sensor motes may be disappeared and affects the topology of the network.
- **Hostile and Remote Environment:** Depending upon the application of WSN, the sensor motes may be left unattended. It becomes difficult to attend sensor motes at remote locations. Adversaries can tamper sensor motes physically or introduces its malicious motes inside the network.

- **Latency:** In WSN, multi-hop routing and mote processing may lead to great latency in the network and it makes synchronization difficult in the network among sensor motes.
- **Fault Tolerance:** In WSN, resource failure e.g., quick depletion of battery or buffer overflow due to very low memory, may happen regularly which results the shutdown of mote or fail to perform the intended operations. Such problems need to be avoided by the strategies of fault tolerance to keep on networking.
- **Scalability:** The scalability factor comes into consideration when network is dynamic in nature. In WSN, addition and deletion of a sensor mote is a usual event. Hence, networking must keep on working whatever the number of sensor motes are placed.

1.1.2 Security Challenges in Wireless Sensor Networks

- **Resource Constraints:** WSN have limited battery, processing power and storage capacity. All the security algorithms are heavy in terms of computation and communication.
- **Standard Activity:** Most routing protocols for WSN are known publicly and do not include potential security considerations at the design stage.
- **Complex Algorithms:** The security algorithms in the literature are heavy in terms of computation and communication. They cannot be implemented on WSN. Lot of research has been carried out for resource constrained WSN to reduce the computation and communication cost of security algorithms.

1.1.3 Security Requirements for Wireless Sensor Networks

WSN are vulnerable to various attacks, due to broadcast nature of transmission medium in an uncontrolled environment. WSN has the following security requirements:

- **Node Authentication:** Node authentication ensures the reliability of the message by identifying its origin. A mote has to prove its validity to other motes in the network and the base-station. This avoids the adversary to send malicious information in the network. The base-station confirms the authentication of the sensor mote.
- **Availability:** Availability ensures the services of resources offered by the network, or a sensor mote must be available whenever required. This can be maintained by regulating the sleep patterns for a sensor mote.

- **Integrity:** Integrity ensures the reliability of the data. It refers to the ability to confirm that a message has not been tampered with, altered or changed in the network.
- **Confidentiality:** Confidentiality ensures the concealment of messages from a passive attacker so that the message communicated in WSN remains confidential.
- **Perfect Forward Secrecy:** Perfect forward secrecy ensures that a session key derived from a set of long-term public and private keys will not be compromised if the long-term private key of one of the party is compromised.

1.2 Key Management

Key management [12] is the set of techniques and procedures, which support the establishment and maintenance of keying relationships between authorized parties. This includes dealing with the generation, exchange, storage, use, and replacement of keys. Keys are of two types: symmetric keys and asymmetric keys. Symmetric key is identical for both encryption and decryption of a message. Symmetric keys must be chosen, distributed and stored securely. Asymmetric keys, in contrast, are two distinct keys that are mathematically linked. They are typically used in conjunction to communicate. A WSN key management protocol consists of three main components (see figure 1.3): (i) Key establishment: creating a session key between the parties that need to communicate securely with each other; (ii) Key refreshment: prolongs the effective lifetime of a cryptographic key; (iii) Key revocation: ensures that an evicted mote is no longer be able to decipher the sensitive messages that are transmitted in the network.

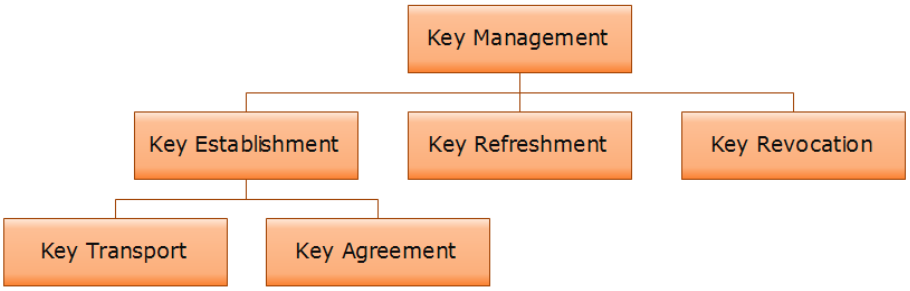

FIGURE 1.3: Components of Key Management Protocols

1.2.1 Components of Key Management

1. **Key Establishment:** Key establishment is a process or protocol whereby a shared secret key becomes available to two or more parties, for subsequent cryptographic

use. So, Key establishment is about creating a session key between the parties that need to communicate securely with each other. There are two types of key establishment protocols:

(a) **Key Transport:** In key transport, one party creates or otherwise obtains a secret value, and securely transfers it to the other party.

(b) **Key Agreement:** In key agreement, two or more parties derive a shared secret key as a function of information associated with each of the parties such that no party can predetermine the resulting value.

There are three types of key establishment techniques [12]:

(a) **The trusted-server scheme:** The trusted server scheme depends on a trusted server.

(b) **The self-enforcing scheme:** The self-enforcing scheme depends on asymmetric cryptography using public keys. Public key algorithms such as Diffe-Hellman and RSA as pointed out in [13] require high computation resources.

(c) **The key pre-distribution scheme:** The key pre-distribution scheme, where key information is embedded in the sensor motes before they are deployed. A key pre-distribution scheme is a key agreement protocol where the resulting established keys are completely determined a priori by initial keying material.

2. **Key Refreshment:** The main goal of key management is to maintain confidentiality of the information. Keys can also assist in authenticating legitimate motes and checking the integrity of the transferred messages. Adversaries try to guess secret keys and get access to the confidential information. In order to avoid adversaries from getting access to secret information, it is important to refresh the secret keys at regular intervals, which depend upon the frequency of communication and frequency of key usage. Key refreshment prolongs the effective lifetime of a cryptographic key.

3. **Key Revocation:** Key revocation is the process of removing keys from operational use prior to their originally scheduled expiry, for reasons such as mote capture. So, key revocation ensures that an evicted mote is no longer too able to decipher the sensitive messages that are transmitted in the network.

1.3 Motivation

WSN is one of the most constrained pervasive systems with minimal resources. Wireless communication between sensor motes is insecure by its nature and requires lightweight

cryptographic methods to ensure security. To establish a session in WSN applications, key agreement is the most important concern, which can be achieved with the help of a lightweight key agreement protocol. The researchers have offered several key agreement approaches but still the field is open to find lightweight approach without compromising security.

Symmetric cryptography is simple and efficient in resource utilization. The main benefit to use symmetric key system is low computing cost but it needs a key pre-distribution process and does not guarantee a perfect connectivity established between the two communicating parties. This approach is also impractical for real time implementation of large scale WSN, as it fails to provide scalability.

Traditional public key (asymmetric) cryptography is more expensive than symmetric cryptography, as it needs certificates for authentication, which requires high computations. There is a scope for identity-based key agreement, which can authenticate the sender and, adversary cannot able to take the advantage by impersonating as a user. By this way, access to the network resources will be restricted only to genuine motes.

Some WSN applications require the elimination of key escrow problem (PKG performs impersonation attack). The research investigation can also be focused on designing a Certificateless key agreement protocol, which can provide the security against key impersonation attack, forward secrecy attack, unknown key share etc. The research can be directed to find the vulnerabilities in the existing protocols and countermeasures can be provided, if any.

In terms of computation overhead, pairing operation is the most expensive operation among other public key operations, such as elliptic curve point multiplication, elliptic curve point addition, etc. The major objective is to reduce the number of computation operations, so that the approach shows its applicability to WSN.

1.4 Thesis Contribution

1.4.1 Research Objectives

Several cryptographic key agreement protocols for WSN have been proposed so far. All the protocols have certain pros and cons and each algorithm behaves differently under different conditions. Keeping in mind the network environment and the application, a particular protocol should be designed. The objectives of the research are as follows:

- Survey and review of the literature on key management in wireless sensor networks

- Improvement in the existing protocol / a novel design / a novel scheme for a feasible algorithm
- Verification and validation of the proposed scheme.

1.4.2 Summary of Contribution(s)

To achieve the first objective, a rigorous review of literature has been done and found that computation cost (in terms of number of operations), running time (in sec), energy consumption (in Joules), memory consumption (RAM and ROM both in bytes) are possible parameters to compare the performance of key agreement protocols. We have reviewed symmetric as well as asymmetric key agreement protocols. To fulfill the requirement of second objective, four protocols have been proposed. Keeping in mind the limitations of symmetric cryptography, the four proposed protocols are based on asymmetric cryptography. The proposed protocols are as follows:

1. A Pairing-Free Identity-based Two-Party Authenticated Key Agreement protocol for WSN.
2. Breaking of CertificateLess Key Agreement Protocol against Key Compromise Impersonations attack and providing a viable solution.
3. A Non-Interactive Certificateless Two-Party Authenticated Key Agreement Protocol for WSN.
4. An Improved Forward Secure Elliptic Curve Signcryption Key Management Scheme for WSN.

The first and fourth protocols belong to the class of identity-based cryptography, second and third protocols are based on CertificateLess cryptography. The first proposed protocol is an efficient Pairing-Free Identity-based two-Party Authenticated Key Agreement (PF-ID-2PAKA) protocol, which takes only three point multiplications operation. PF-ID-2PAKA protocol does not use pairing operation. Also, PF-ID-2PAKA protocol has been compared with the existing identity based key agreement protocols and found to be efficient in terms of computation cost. PF-ID-2PAKA is proven to be secure in extended Canetti-Crawzyk (eCK) model [14] for identity-based setting and simulation result shows the viability on WSN.

The second protocol is based on Certificateless cryptography, which focuses on the elimination of key escrow problem (a well-known problem based on the impersonation of sensor mote by PKG) and it is an improvement of Kim et al.'s [15] protocol. We proved

that Kim et al.'s protocol is insecure against Key-Compromise Impersonation (K-CI) attack. The proposed protocol is secure in the eCK model [16] for certificateless setting.

The third protocol is a Non-Interactive Certificateless Two-party Authenticated Key Agreement (NI-CTAKA) protocol. NI-CTAKA is proven to be secure in eCK model [16]. This protocol takes only three point multiplication operations. The protocol has been compared with existing certificateless key agreement protocols and proved to be more efficient in terms of computation cost. The simulation result shows the viability on WSN.

The last protocol is an improved forward secure key management scheme based on elliptic curve signcryption scheme for WSN. Forward secrecy is the most important concern in WSN as joining of new motes in the network is a common affair. The proposed protocol has been compared with the existing protocols and found to be efficient in terms of computation cost.

For third objective, verification of the proposed method(s) is achieved by making use of mathematical theorems and lemmas. The same is validated using experimental simulation setup. The proposed protocols have been implemented on MICAz platform [17]. The simulation setup includes the following:

1. Ubuntu 12.04
2. TinyOS-2.1.1 (Operating System for mote) [18, 19]
3. RELIC-0.3.3 cryptographic library [20]
4. AVRORA-1.7.106 [21] simulator for computing running time and energy consumption.

1.5 Thesis Outline

The thesis has been organized into seven chapters. The remainder of this thesis is organized as follows:

Chapter 2 provides an overview of cryptographic approaches (symmetric and asymmetric), cryptographic primitive like hash functions, elliptic curve cryptosystem, bilinear pairing, complexity assumptions. This chapter also covers an overview of authenticated key agreement, provable security, formal and security models of identity-based and certificateless two-party authenticated key agreement protocols and the concept of signcryption.

Chapter 3 presents the simulation and experimental setup for the implementation of the proposed work. It includes an overview of MICAz mote, TinyOS-2.1.1 operating system for the MICAz mote, RELIC-0.3.3 cryptographic library, and the simulator used AVRORA-1.7.106. This experimental setup is used to verify and validate the proposed protocols for the computation of running time and energy consumption.

Chapter 4 presents a Pairing-Free Identity-based Two-Party Authenticated Key Agreement (PF-ID-2PAKA) protocol for WSN based on Gap Diffie-Hellman (GDH) complexity assumption. It requires three point multiplications in key agreement phase. In order to evaluate the performance of the proposed protocol for WSN, the parameters used are running time, energy consumption and memory consumption. This chapter also presents the correctness and security analysis of PF-ID-2PAKA for eCK model presented by Liang et al. [14].

Chapter 5 presents two Certificateless Two-party Authenticated Key Agreement (CTAKA) protocols, one is an improvement to existing protocol and another is a proposed protocol. This chapter includes the cryptanalysis of an exiting protocol presented by Kim et al. [15] against Key-Comromise Impersonation (K-CI) attack. Further, an improvement in Kim et al.'s algorithm has been presented, which is secure in the eCK model presented by Lippold et al. [16]. In addition, a Non-Interactive Certificateless Two-party Authenticated Key Agreement (NI-CTAKA) protocol has been proposed, which is based on GDH complexity assumption. This chapter also presents the correctness and security analysis of NI-CTAKA for eCK model presented by Lippold et al. [16]. For the evaluation of performance of NI-CTAKA for WSN, the parameters used are running time, energy consumption and memory consumption.

Chapter 6 presents a forward secure elliptic curve signcryption key management scheme for WSN. It includes the comparison illustrating the number of basic operations like point addition on a group, point scalar multiplication on a group, multiplication on a field group, pairing operation, hash operation, which can be helpful in deciding the efficiency of the protocol in the signcryption and unsigncryption phases.

Chapter 7 concludes the thesis by highlighting contributions and suggestions for future work.

1.6 Summary

Wireless Sensor Network (WSN) is an example, which is being widely used for collecting and communicating information from military to civilian tasks. To realize the full potential of these networks, security challenges must be addressed properly. In this chapter,

we investigate that due to resource-constrained nature of these battery powered sensor devices, traditional security mechanisms are not feasible to be applied. Key exchange protocols are the key factors of establishing communication between two parties. The next chapter in line discusses the background theory on cryptographic approaches by highlighting the contributions of various researchers in this field.

Chapter 2

Background Theory and Literature Survey

After analyzing the limitations of Wireless Sensor Networks (WSN), it is understood that WSN needs lightweight algorithms to provide security in their applications. For key management in WSN, the cost of communication and computation should be less. In symmetric key cryptography the communication cost is high and computation cost is low, but it has some limitations, which restrict their use in WSN. This chapter reviews symmetric key management schemes for WSN. Further, the chapter presents the cryptographic approaches, key agreement protocols, formal and security model for identity based and certificateless two-party authenticated key agreement protocols. This chapter also presents the introduction to signcryption. [1]

2.1 Symmetric Cryptography

In Symmetric cryptography, a single private key is shared between communicating parties. The same key is used for the encryption and decryption. The primitives used in symmetric cryptographic are block ciphers, stream ciphers, cryptographic hash functions, and Message Authentication Codes (MAC).

[1] The major findings of this chapter have been published

- "*Classification of Symmetric Key Management Schemes for Wireless Sensor Networks*," **International Journal of Security and Its Applications**, Vol. 7, Issue 2, pp. 117-138, March 2013.
- "*A Survey and Taxonomy of Symmetric Key Management Schemes for Wireless Sensor Networks*," in the Proceedings of the CUBE International Information Technology Conference (CUBE'12), pp. 585-592, 2012.

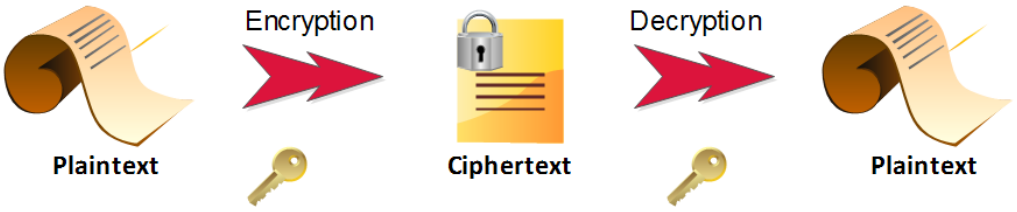

FIGURE 2.1: Symmetric Cryptography

2.1.1 Review of Symmetric Key Management Schemes for WSN

Key management protocols are the backbone for providing security in WSN. The main goal of key management scheme is to provide secure communication between sensor to sensor, a group of sensors and sensor to base station by making use of unicast, multicast and broadcast respectively. There are various ways in which we can classify the key management schemes in WSN by considering different benchmarks. Various researchers presented different taxonomies. Key management schemes in WSN can be classified broadly into dynamic and static solutions based on whether rekeying of administrative keys is enabled post network deployment. These schemes are also classified into homogeneous or heterogeneous networks with regard to the role of network nodes (motes) in the key management process. Homogeneous network based schemes generally assume a flat network model, while heterogeneous network based schemes are intended for both flat and clustered networks. Another criterion is hierarchical and distributed sensor networks based on network models. Other classification criteria may be whether nodes are anonymous or have pre-deployment identifiers, and if so, when (pre-, post-deployment, or both), and what deployment knowledge (location, degree of hostility, etc.) is imparted to the nodes.

We classify symmetric key management schemes broadly into three categories as: base-station participation scheme, trusted third node based scheme and pre-distribution schemes. Pre-distribution schemes are further classified into nine categories as master key based pre-distribution scheme, pairwise key pre-distribution schemes, pure probabilistic key pre-distribution schemes, polynomial based key pre-distribution schemes, matrix based key pre-distribution schemes, tree based key pre-distribution schemes, hierarchical key management scheme, combinatorial design based key pre-distribution schemes, EBS based key pre-distribution schemes. These are enumerated in figure 2.2. The following subsection defines these schemes.

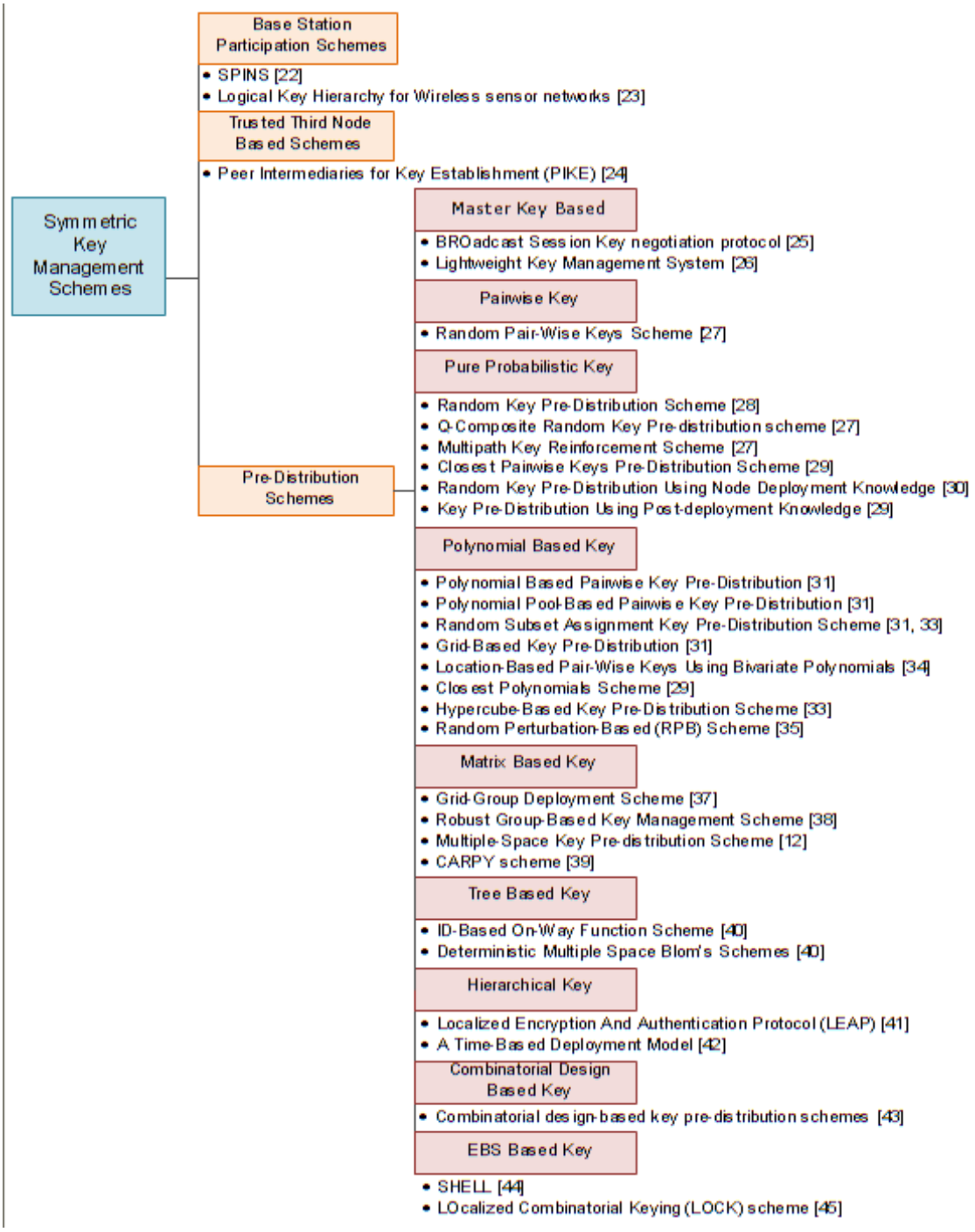

FIGURE 2.2: Symmetric Key Management Schemes for WSN

2.1.1.1 Base Station Participation Scheme

In base station participation schemes, a trusted, secure base station is used as an arbiter to provide link keys to nodes. Each node shares a unique key with a base station. The base station acts as a Key Distribution Centre (KDC). Thus, the scheme is also called centralized Key Distribution Centre (KDC) approach. The authentication is provided by the nodes itself to the base station. The base station generates a link key and sends

it to both parties securely. The scheme requires less memory and perfectly controlled node replication. Also, it is resilient to node capture and can revoke key pairs to be compromised. But it is not scalable and hence the base station becomes the target of attacks. The following are the base station participation schemes:

- *SPINS* [22]: SPINS is a security building block, optimized for resource-constrained environments and wireless communication, an authenticated routing scheme and a secure node-to-node key agreement protocol. The protocol is based on trusted base server. It has two secure building blocks: SNEP (Secure Network Encryption Protocol), which provides the data confidentiality, two-party data authentication, and data freshness with low overhead and μTESLA (the micro version of the Timed, Efficient, Streaming, Loss-tolerant Authentication Protocol), which provides authenticated broadcast for severely resource-constrained environments.
- *Logical Key Hierarchy for WSN (LKHW)* [23]: LKHW is a secure group communication scheme, which is based on directed diffusion and Logical Key Hierarchy (LKH) scheme. It is used to protect the directed diffusion protocol. It is a key tree structure with source nodes as leafs and a sink node as the root, where each leaf node holds keys along the path from it to the root node. LKHW integrates security and routing in a single framework.

2.1.1.2 Trusted Third Node Based Scheme

In trusted third node based scheme, a peer node is used as a trusted intermediary for the establishment of a shared key between nodes.

- *Peer Intermediaries for Key Establishment PIKE* [24]: PIKE is a key distribution scheme, which is based on using peer nodes as trusted intermediaries. The objective of the scheme was to address the lack of scalability of existing symmetric key distribution schemes. The scheme establishes keys between any two nodes by using one or more nodes trusted intermediaries regardless of network topology or node density. The scheme is having less communication overhead as compared to Key Distribution Center (KDC) approach.

2.1.1.3 Pre-Distribution Schemes

In the initialization of the scheme, each node is distributed with the secret keys or secret information before deployed into the sensing area. The key pre-distribution scheme

comprises of three phases: key pre-distribution phase, shared-key discovery phase and path-key establishment phase. The following are the nine categories of pre-distribution schemes:

1. **Master Key Based Pre-Distribution Scheme:** A single or master key is loaded to all the nodes in the network before deployment in the network. After deployment, every node in the network can use the master key for the encryption and decryption of the messages. The benefits of such schemes include minimal storage requirements and avoidance of complex protocols. This also prevents the process of key discovery or key exchange. But, this will lay the network to the compromising state because of the single node shared throughout the network.

    - *BROadcast Session Key negotiation protocol (BROSK)* [25]: BROSK broadcasts key negotiation messages to construct link-dependent keys, where each node can negotiate a session key with its neighbors. The scheme solves the problem of authentication by constructing trust levels among the nodes.
    - *Lightweight Key Management System* [26]: Lightweight Key Management System is based on the basic bootstrapping protocol, where secure links are established between nodes, which are deployed in different phases. It authenticates and establishes secure local links by using initial trust, which is put up from a small set of shared keys. A group authentication key and a key-generation key have been stored to each node. The authentication key is used for the authentication of the two nodes, which must be from same generation. For the establishment of the session key random nonce values are exchanged.

2. **Pair-Wise Key Pre-Distribution Scheme:** Pair-wise keys are loaded to all the nodes before they are being deployed in the network. For example, if n nodes in the network are deployed, then $n-1$ unique pair-wise keys loaded to each node in the network, which allows each node to communicate with all other nodes in its communication range. The merit of this approach is increased resilience against the node capture and offers node-to-node authentication. This minimizes the chance of node replication but has an additional overhead needed for each node to establish $n-1$ unique keys with all the other nodes in the network and to maintain the keys in the memory. The key-distribution process consists of three phases: (i) key pre-distribution phase; (ii) shared-key discovery; (iii) path-key establishment phase. Following is the pre-distribution schemes presented in the literature.

    - *Random Pair-Wise Keys Scheme* [27]: The random pair-wise keys scheme is a pair-wise key pre-distribution scheme in which distinct pair-wise keys are loaded to the nodes before deployment.

3. **Pure Probabilistic Key Pre-Distribution Schemes:** The master key based scheme and pair-wise key pre-distribution scheme are trivial solutions. In both the cases if memory requirement and key connectivity is considered then resilience would be compromised or vice versa. Thus, to overcome such disadvantages, there is one more solution, which ensures some probability that any two nodes can communicate using a pair-wise key. The scheme does not, however, ensure that two nodes are always able to compute a pair-wise key for secure communication. Following are the schemes based on pure probabilistic key pre-distribution:

    - *Random Key Pre-Distribution Scheme* [28]: A random key pre-distribution scheme was proposed for distributed sensor network, which is dynamic in nature. This facilitates the addition and deletion of nodes in the network after deployment. The scheme basically focuses on the bootstrapping problem in sensor networks.
    - *Q-Composite Random Key Pre-distribution Scheme* [27]: Q-common keys chosen from a large key pool instead of one common key are loaded in each of the node. This will increase the resilience of the network against node capture.
    - *Multipath Key Reinforcement Scheme* [27]: The scheme is used with basic random key scheme to strengthen the security of an established link key by establishing the link key through multiple paths.
    - *Closest Pair-wise Keys Pre-Distribution Scheme* [29]: The scheme is based on Pseudo Random Function (PRF) and a master key is shared between each node and the setup server, where each node share pair-wise keys with a number of other nodes whose expected locations are closest to the expected location of the node.
    - *Random Key Pre-Distribution Scheme Using Node Deployment Knowledge* [30]: The most important knowledge for pre-distribution is the knowledge of the nodes that are likely to be neighbors of each nodes. Deployment knowledge can be exhibited using non-uniform probability density functions (pdfs) which means that the positions of nodes to be at certain areas.
    - *Key Pre-Distribution Using Post-deployment Knowledge* [29]: The objective of the scheme is to improve the pair-wise key pre-distribution in static sensor networks. Pre-distributed keys got priority based on post-deployment knowledge, after assigning an excessive amount of pre-distributed keys to each node, and discard low priority keys to prevent node compromise attacks and return memory to the applications.

4. **Polynomial-Based Key Pre-Distribution Schemes:** Polynomial-based key pre-distribution scheme is based on pair-wise keys pre-distribution schemes. Thus, these schemes overcome some of the probabilistic pre-distribution schemes disadvantages. These are: (1) Any two nodes can definitely establish a pair-wise key, when there are no compromised nodes; (2) Even with some nodes compromised, the others in the network can still establish pair-wise keys; (3) A node can find the common keys to determine whether or not it can establish a pair-wise key and thereby help reduce communication overhead. The following are the polynomial-based key pre-distribution schemes:

   - *Polynomial Based Pair-wise Key Pre-Distribution* [31]: The scheme uses the concept of the protocol in [32], which was developed for group key pre-distribution.
   - *Polynomial Pool-Based Pair-wise Key Pre-Distribution* [31]: It is a general framework, which is based on polynomial based key pre-distribution and key pool [27, 28]. A pool of randomly generated bivariate polynomials is used to establish pair-wise keys between nodes. The polynomial pool has two special cases. (1) When the polynomial pool has only one polynomial, the general framework degenerates into the polynomial-based key pre-distribution. (2) When all the polynomials are 0-degree ones, the polynomial pool degenerates into a key pool as in the scheme presented by Eschenauer and Gligor [28] or the Q-Composite scheme [27].
   - *Random Subset Assignment Key Pre-Distribution Scheme* [31, 33]: This scheme is based on polynomial pool-based pair-wise key pre-distribution. A random strategy is used for subset assignment during the setup phase. The setup server selects a random subset of polynomials in F and a polynomial share is assigned to the node.
   - *Grid-Based Key Pre-Distribution* [31]: This scheme is based on the components of the general framework designed in [31]. The scheme guarantees that any two nodes can establish a pair-wise key, when there are no compromised nodes, provided that the nodes can communicate with each other.
   - *Location-Based Pair-Wise Keys Scheme Using Bivariate Polynomials* [34]: It is based on polynomial-based key pre-distribution technique and closest pair-wise keys scheme. The target field partitioned into small areas called cells, each of which is associated with a unique random bivariate polynomial.
   - *Closest Polynomials Scheme* [29]: It is a combination of the expected locations of nodes with the random subset assignment scheme in [31] to overcome the limitations of nodes. The polynomials for each node are chosen based on

its expected location instead of random selection as in the original random subset assignment scheme.

- *Hypercube-Based Key Pre-Distribution Scheme* [33]: It is a generalization of grid-based key pre-distribution scheme [31]. It guarantees that any two nodes can establish a pair-wise key when there are no compromised nodes, assuming that the nodes can communicate with each other.
- *Random Perturbation-Based (RPB) Scheme* [35]: The scheme is based on polynomials to generate pair-wise keys. The polynomials are defined over a finite field denoted as $\mathbb{F}(q)$, where q is a prime number.

5. **Matrix-based key pre-distribution schemes:** In matrix-based key pre-distribution schemes, all possible link keys in a network of size n can be represented as an $n \times n$ key matrix, which is based on Blom's concept [36]. Small amount of information is stored to each node, so that every pair of nodes can calculate corresponding field of the matrix, and uses it as the link key.

   - *Grid-Group Deployment Scheme* [37]: Sensor nodes are uniformly deployed in a large area instead of randomly distributing keys from a large key pool to each node. Secret keys are systematically distributed to each node from a structured key pool.
   - *Robust Group-Based Key Management Scheme* [38]: It uses the concept of group-based key management scheme using node deployment knowledge. The sensor field is partitioned into hexagonal grids.
   - *Multiple-Space Key Pre-distribution Scheme* [12]: It is based on Blom's key pre-distribution scheme and combines the random key pre-distribution method [28] with it, which offers improved network resilience. In this scheme, complete graph is converting to a connected graph, so that each node needs to carry less key information.
   - *ConstrAined Random Perturbation based pair-wise keY establishment (CARPY) scheme* [39]: Two nodes communicate with each other only for exchanging the respective column, which can be known by the adversary. If each node itself can generate each column, then communications will no longer be necessary. The second variation is (CARPY+) Communication-Free CARPY Scheme.

6. **Tree-based Key Pre-Distribution Schemes:** In tree-based key pre-distribution schemes, nodes are arranged in a tree in which each node communicates with its parent node. So, the key establishment has done between neighboring nodes along the aggregation tree. The new node receives two tickets that can be verified by two existing nodes randomly selected by the network administrator, before joining

the network. After the deployment of a new node into the network, it generates a pair-wise key for its parent node. For securely transmitting the key to the parent, the new node splits the key into two parts and sends them with its tickets to the nodes selected by the administrator, which authenticates the new node and forwards key materials to the parent of the new node. The merit of a tree-based key distribution is the significantly reduction of the memory cost.

- *ID-Based On-Way Function Scheme* [40]: In this scheme a public one way hash function are used to reduce the number of keys stored in the node. A unique ID is assigned to each node and this ID is used to compute secret keys.
- *Deterministic Multiple Space Blom's Scheme* [40]: The basic idea of the scheme is to weaken the connectivity of the network graph to improve the resiliency of the multiple IOSs. The scheme considers the complete bipartite graph instead of a complete graph.

7. **Hierarchical Key Management Scheme:** A tree of keys is built for the hierarchical network, where the keys at a certain level are distributed to the corresponding class of nodes. The keys at higher levels can be used to derive the keys at lower levels, but not vice versa. The intention of the hierarchical network is to facilitate data collection, fusion and query propagation in hostile environments.

   - *Localized Encryption and Authentication Protocol (LEAP)* [41]: It supports the in-network processing. It restricts the security impact of a node compromise to the immediate network neighborhood of the compromised node. It supports the establishment of four types of keys for each node an individual key shared with the base station, a pair-wise key shared with another node, a cluster key shared with multiple neighboring nodes, and a group key that is shared by all the nodes in the network.
   - A Time-Based Deployment Model [42]: It would localize the impact of key compromise within the time intervals.

8. **Combinatorial design-based key pre-distribution scheme:** Combinatorial design-based key pre-distribution scheme [43] decides number of selected keys to assign to each key-chain before deployment of the nodes in the network. This scheme arranges the elements of a finite set into subsets to satisfy certain properties.

9. **EBS-Based Key Pre-Distribution Scheme:** EBS-Based Key Pre-Distribution Scheme [27] uses a combinatorial optimization methodology for key management of group communication setups. The EBS scheme exploits the trade-off between the number of administrative keys k and the number of rekeying messages m.

- *SHELL Scheme* [44]. SHELL stands for Scalable, Hierarchical, Efficient, Location-aware, and Light-weight. The scheme supports rekeying, thus enhances network security and survivability against node capture. Key assignment has been tracked by cluster gateways not by the actual keys. SHELL is collusion-resistant. It uses post-deployment location information in key assignment.
- *LOcalized Combinatorial Keying (LOCK) Scheme* [45]. LOCK performs localized rekeying to minimize overhead. The physical network model is a three-tier WSN with the base station (BS) at the top, followed by cluster leader nodes (CLs), then regular nodes. In LOCK, no pre-deployment information is assumed about the expected locations of the nodes.

Symmetric algorithms are generally preferable to asymmetric algorithms in the field of WSN because they are provides the most lightweight solution, but it does not provide all the security requirements. However, when symmetric algorithms are used, two problems arise: (1) key distribution and (2) number of keys stored. When individual keys are used in WSN with n nodes, each node has to store $(n-1)$ keys. This solution provides the perfect connectivity in the network and good resilience but does not provide scalability, hence, unsuitable for large WSN. Every time a new node enters into the network, the pair-wise keys should be distributed to each node in the network. Moreover, forward secrecy is not given after a nodes key has been compromised. When single symmetric key is used, memory requirement is greatly reduced, but at the same time this is not resilient anymore. To cope up with this problem, various researchers have presented many probabilistic key distribution schemes for symmetric algorithms. In general these approaches either need pre-distributed keys, which means a higher configuration effort before deployment, as a result, it require complicated key management that may cause large memory and communication overhead, which results in higher energy consumption.

2.2 Asymmetric Cryptography

In asymmetric cryptography, each party has a key pair consists of a public key and a private key. Public key is used for encryption and private key is used for decryption (see figure 2.3). Diffie and Hellman [46] introduced the concept of asymmetric cryptography (also known as Public Key Cryptography) in 1976. The private key is kept secret, while public key is widely distributed over the network. Both the keys have mathematical relation, where the private key is derived from the public key. Any party who want to communicate with another party can encrypt the message using the recipient's public key. The recipient then decrypts the ciphertext by using its private key. So, this might

be advantageous in a way that the participant does not require a previously generated session key in order to interact. This maintains the freshness of the shared secret.

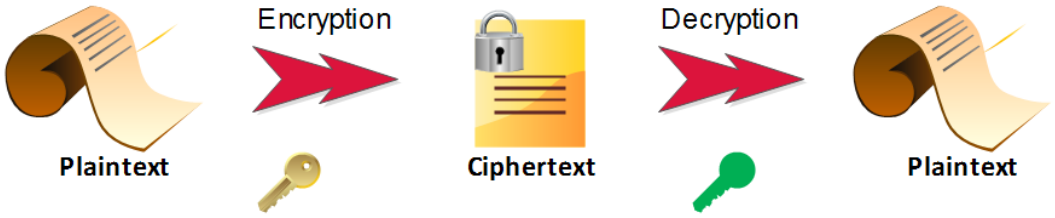

FIGURE 2.3: Asymmetric Cryptography

Unlike symmetric keys, public/private key pairs need not to be changed frequently in order to protect from vulnerabilities. It does not provide the verification of the authenticity of public keys. In WSN, the network deployer is a trusted entity, which can deploy nodes with an embedded private/public key pair. In this way the trusted key generation center is no longer necessary. A single node can broadcast its public key to all its neighbors, who can use it for encryption. For the authentication of the public keys, a trusted Certificate Authority (CA) issues appropriate certificates. The security of any public-key cryptosystem lies on users being assured that they cannot be fooled into using a false public key. There are various methods to achieve PKC, viz. traditional public-key infrastructure, identity-based cryptography and certificateless public key cryptography.

2.2.1 Public-Key Infrastructure

Public-Key Infrastructure (PKI) is the traditional method for authenticating public keys. In PKI system, Certificate Authority (CA), a trusted third party, who is responsible for the establishment and verification of the authenticity of public keys. The task of a CA is to create, manage, store, distribute and revoke digital certificates. Apart from this, the CA might be responsible for creating a public-key pair and sending a copy of this information to the specified user over a secure channel. The aim of CA is to bind public keys with respective user's identities through a digital signature with its private key and store it in a repository. Alternatively, users can create their own public-key pair and transfer their public keys to the CA in a secure manner, which then creates the necessary certificates. In both cases, CA must verify the identity of the user before granting the corresponding public-key certificates. If the CA does not know the private key corresponding to a given public key, it must verify that the associated user does know this information, as otherwise a dishonest user might claim someone else's public key as their own. Figure 2.4 depicts the working of Public Key Cryptography (PKC).

A typical public key infrastructure consists of the following components: Certificate Authority (CA), Registration Authority (RA), Certificate repository and Certificate

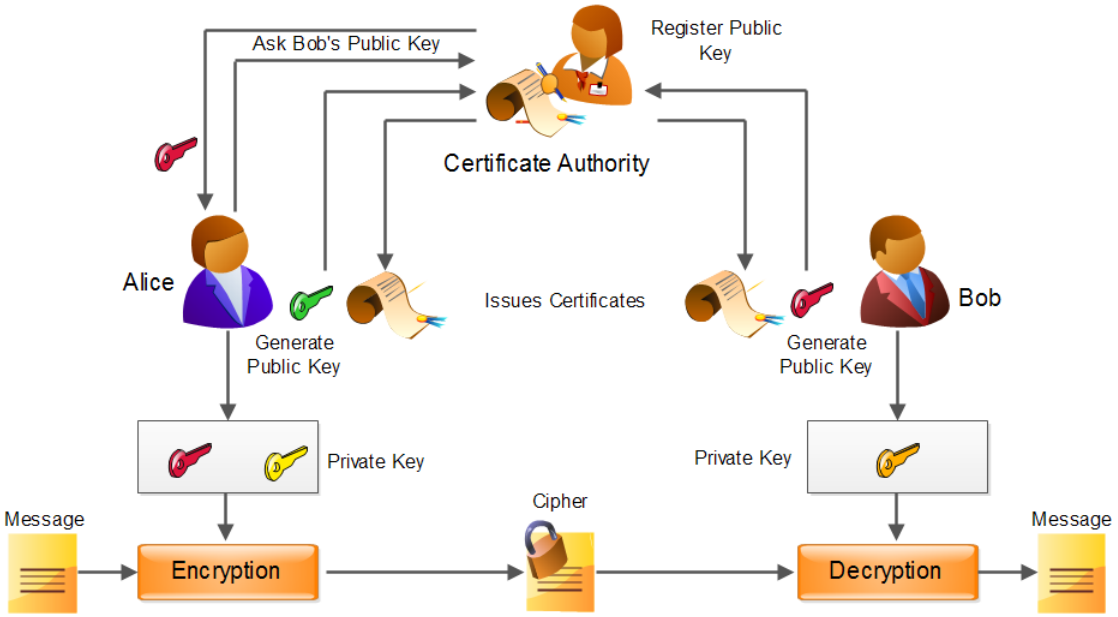

FIGURE 2.4: Traditional Public Key Cryptography

management system. The task of RA is to perform initial authentication and acts as the verifier for the Certificate Authority before a digital certificate is issued to the user. Certificate repository is a directory, where certificates with their public keys and Certificate Revocation Lists (CLRs) are stored. For two communicating parties *Alice* and *Bob*, *Bob* wants to communicate with *Alice* and asks for his digital certificate. *Bob* contacts the CA in order to validate the received certificate. The CA checks in the repository to see if the certificate is still valid (has not expired or been revoked). *Bob* verifies the digital certificate by using CA's public key. After verification *Bob* sends ciphertext to *Alice* by using her public key.

On application of PKI to WSN, base-station is a trusted third party and performs the role of Certificate Authority (CA) and Registration Authority (RA). The base-station also creates the public/private key pairs for each sensor node. WSN has multi-hop communication; so base-station cannot act as a certificate repository. PKI is impractical to apply on WSN, as PKI is complex, inefficient in terms of resource utilization, having storage and communication overhead. Moreover, certificate management needs the continuous connectivity of base-station to the network, which makes it vulnerable to attacks. PKI needs key revocation process as it maintains certificates.

2.2.2 Identity-Based Cryptography

Adi Shamir [8] formulated the concept of Identity-Based Cryptography (IBC) in 1985. The aim of IBC is to generate public keys from the unique identity of the user, e.g. name,

address etc. by using hash functions. The trusted third party, known as Private Key Generator (PKG), generates the private key of the user from his public key and sends it to the user via a secure channel. This eliminates the registration of users to PKG in advance, hence, eliminates the need for certificates. The main limitation of IBC is the key escrow problem as KGC is having access of all users' private keys. IBC does not support non-repudiation; a valid signature does not guarantee the user actually signed the message. Unlike PKI, it removes the need of key revocation. Figure 2.5 depicts the working of Identity Based Cryptography (IBC).

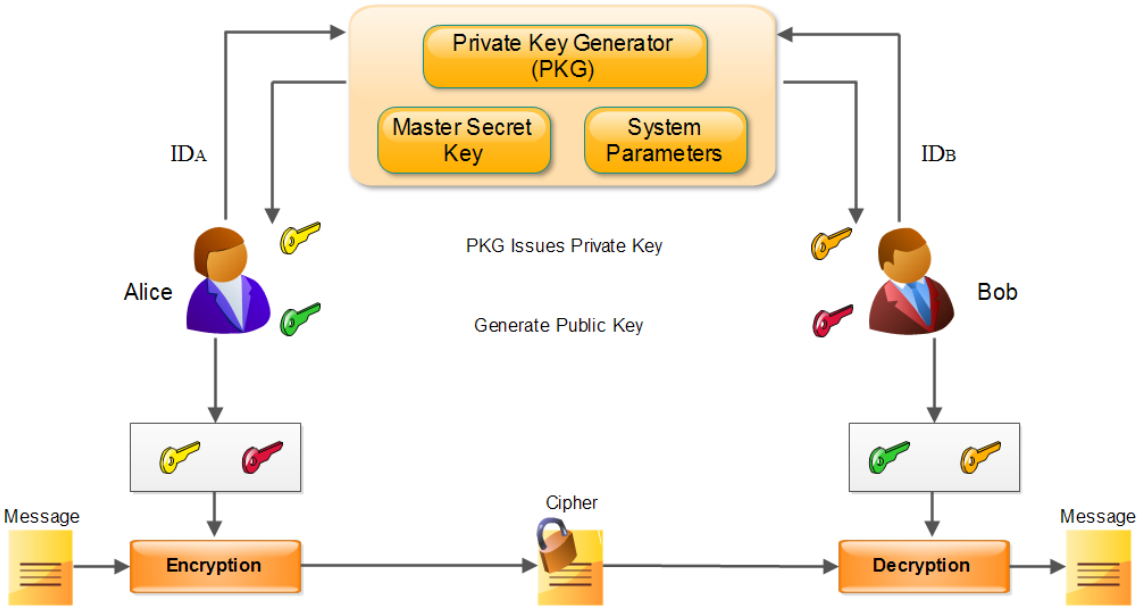

FIGURE 2.5: Identity-Based Cryptography

For two communicating parties *Alice* and *Bob*, *Bob* wants to communicate with *Alice*. *Bob* uses his private key to encrypt the message and transmits the ciphertext through a secure channel to *Alice*. Then *Alice* authenticates herself to the PKG and receives her private key and decrypts the message. The main problem with IBC is the need for a trusted PKG. However, in some applications of WSN, base-station is a trusted entity. IBC can be applicable to WSN for the following reasons: (i) no need to maintain public key directory, (ii) nodes generate a public key for a given node only when they want to communicate with it for the first time, (iii) after agreeing upon a shared session key, nodes can use cheap symmetric key mechanisms to encrypt the messages and to communicate in a secure manner, (iv) provides scalable security mechanism in which the number of keys are kept minimum.

2.2.3 CertificateLess Public Key Cryptography

Al-Riyami and Paterson [10] introduced the concept of CertificateLess Public-Key Cryptography (CL-PKC) in 2003. CL-PKC eliminates the congenital key escrow problem of IBC as well as the use of certificates in traditional PKI and preserves the advantages of both. Figure 2.6 depicts the working of Public Key Cryptography (PKC).

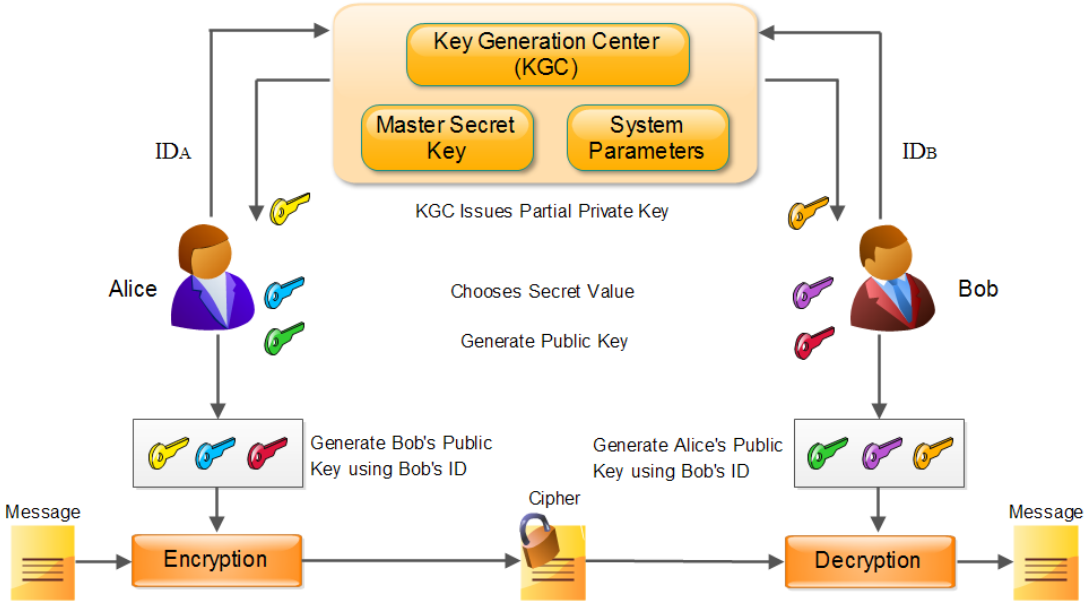

FIGURE 2.6: Certificateless Public Key Cryptography

Key Generation Center (KGC) is a trusted third party, generates a partial-private-key for the user, while user's secret-value and partial-private-key are used to generate public-key of the user. In other words, CL-PKC differs from IBC in terms of arbitrary public-key and when a signature is transmitted, user's public-key is attached with it but not certified by any of the trusted authority. Moreover, KGC does not aware of the secret-key of the user.

2.3 Cryptographic Primitives

In this section, we will discuss various cryptographic primitives used in the proposed work.

2.3.1 Hash functions

Hash functions or one-way hash functions are used to map an arbitrary-length message string to fixed-size message string. The final value is called hash value. Hash function

produces a short fingerprint for a message string. It increases the security of the key in key agreement algorithms by mapping it with an element of an appropriate group in such a way that preserves the uniqueness property. Hash functions must be one-way and collision resistant. A hash function H is said to be one-way if it is computationally impossible to recover the message x from a hash value $H(x)$. A collision resistant hash function implies that no two messages should generate the same output. The formal definition of a hash function is as follows:

Definition 2.1. A hash function is a function $H : \mathcal{D} \rightarrow \mathcal{R}$, where the domain $\mathcal{D} = \{0,1\}^*$ and the range $\mathcal{R} = \{0,1\}^n$ for some $n \geq 1$.

If the hash value is used as some kind of key then we will refer the hash value as key derivation function (kdf) or keyed hash function. Keyed hash function is essentially an element of a family of hash functions parameterized by some key k. Hash functions that are not parameterized by a given key are referred to as unkeyed hash functions.

Definition 2.2. A keyed hash function is a function $H_k : \mathcal{D} \rightarrow \mathcal{R}$, where $\mathcal{K} = \{0,1\}^k$ is the key space and $\mathcal{R} = \{0,1\}^n$ for some $n \geq 1$.

Hash functions should satisfies the following properties:

- *Preimage resistance*: We say, H is preimage resistant or one-way, if given $r \in_R \mathcal{R}$, it is difficult to find $d \in \mathcal{D}$ such that $H(d) = r$.
- *Second preimage resistance*: We say, H is second preimage resistant, if given $d \in_R \mathcal{D}$, it is difficult to find $d' \in \mathcal{D}$ such that $d' \neq d$ and $H(d') \neq H(d)$.
- *Collision resistance*: We say, H is collision resistant, if it is difficult to find $d, d' \in \mathcal{D}$ such that $d' \neq d$ and $H(d') \neq H(d)$. Note: collision resistance implies second preimage resistance, but not necessarily preimage resistance.

2.3.2 Elliptic Curve Cryptosystem

Elliptic Curve Cryptography (ECC) has emerged as a security solution for WSN as it offers same security level to RSA with quite less key size and very low computational overhead. For example, the security level of 160-bit ECC is equivalent to 1024-bit RSA. The first implementation of ECC over WSN was presented by David Malan [47] and concluded that PKI may also be tractable in 4KB of primary memory on the 8-bit, 7.3828 MHz device. Later, Gura et al. [48] compared RSA and ECC and proved that ECC is far better than RSA in terms of key size and processor word size. Liu et al. [49]

provided an open source cryptographic library for ECC operations in WSN known as TinyECC.

Elliptic curve system is based on hardness assumption of Discrete Logarithm Problem (DLP) over elliptic curve known as Elliptic Curve Discrete Logarithm Problem (ECDLP). Neal Koblitz [50] and Victor S. Miller [51] introduced the concept of elliptic curves in cryptography. ECC is a kind of public key cryptography in which each user or the device is taking part in the communication generally has a pair of keys, a public key and a private key, and a set of operations associated with the keys to perform the cryptographic operations. Point multiplication is a fundamental operation underlying ECC, which is defined over finite field operations. The finite fields may be either prime integer fields $\mathbb{GF}(p)$ or binary polynomial fields $\mathbb{GF}(2^m)$.

An elliptic curve is a set of points over a finite field $\mathbb{GF}(p)$, a Galois field of order p, which satisfies the Weierstra$\mathcal{B}$ equation defined as follows:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \tag{2.1}$$

and the coefficients $a_i \in \mathbb{GF}(p)$. But for simplification of computations, cryptographic applications prefers the simple form of Weierstra$\mathcal{B}$ equation as

$$y^2 = x^3 + ax + b, \tag{2.2}$$

where $a, b \in \mathbb{GF}(p)$. Figure 2.7 shows the elliptic curve with parameter, $a = -3$ and $b = 1$.

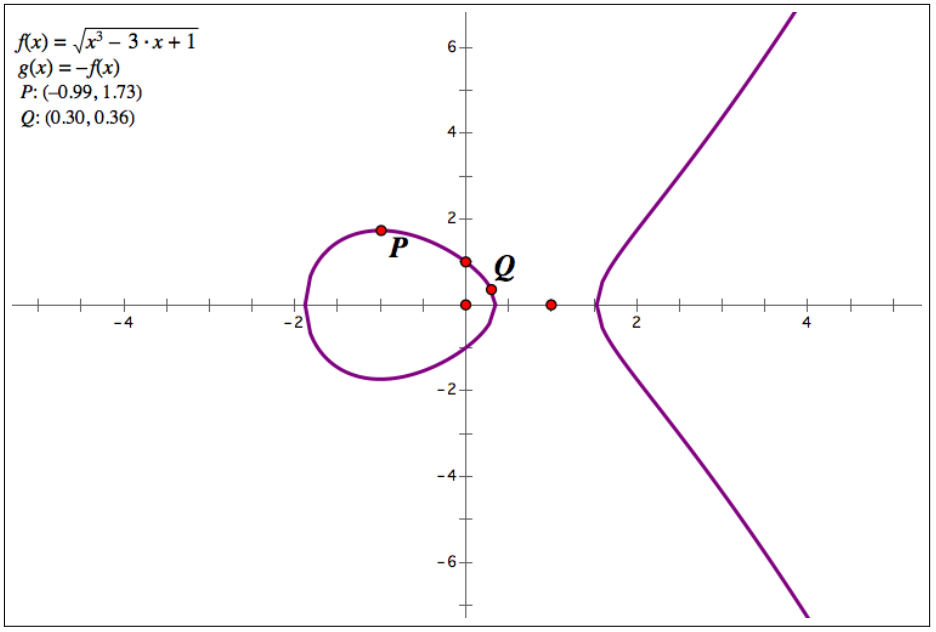

FIGURE 2.7: Elliptic Curve, Parameters: $a = -3$ and $b = 1$

The elliptic curve group operation is closed so that the addition of any two points is a point in the group. Let the coordinates of two points P and Q are (x_1, y_1), (x_2, y_2), respectively, point R having coordinate (x_3, y_3) is the addition result of points P and Q, where x_3 and y_3 satisfy the equation

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3) \tag{2.3}$$

such that $x_3 = m^2 + m + x_1 + x_2 + a$ and $y_3 = m(x_1 + x_3) + x_3 + y_1$, where $m = (y_1 + y_2)/(x_1 + x_2)$.

The security of ECC depends on the hardness assumption of ECDLP. Let P and Q be two points on an elliptic curve such that $kP = Q$, where k is a scalar. It is computationally impossible to obtain k for given P and Q, if k is very large. The scalar k is known as the discrete logarithm of Q to the base P. Thus, point multiplication is the main operation involved in ECC.

2.3.3 Pairings (Bilinear Maps) on Elliptic Curve Groups

Menezes et al. [52] introduced the concept of pairings in 1993 in MOV attack. Whereas Sakai et al. [53] and Joux [54] used it for cryptography in 2004. Later, Boneh and Franklin [55] used the concept of pairing with identity based encryption scheme. Pairings are used to map the pairs of element of the same group or different group to an element of another group. Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order q. $\mathbb{G}_T$ is referred as target group, then bilinear pairing is defined as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ or $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Generally, $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of points on an elliptic curve over a finite field, and $\mathbb{G}_T$ is a subgroup of a multiplicative group of a related finite field. Pairings are based on the Weil and Tate pairings on an elliptic curve over finite field. The objective of pairing is to transport the hardness problem on a certain class of elliptic curves over a finite field to the hardness problem on a smaller finite field. The formal definition of a pairing is as follows:

Definition 2.3. Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_T$ be a cyclic multiplicative group of the same order q. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$ be a bilinear pairing with the following properties:

- *Bilinearity*: Given any $P, Q, R \in_R \mathbb{G}_1$, then $\hat{e}(P + Q, R) = \hat{e}(P, R) \cdot \hat{e}(Q, R)$ and $\hat{e}(P, Q + R) = \hat{e}(P, Q) \cdot \hat{e}(P, R)$. In particular, for any $a, b \in_R \mathbb{Z}_q^*$, then $\hat{e}(aP, bP) = \hat{e}(P, P)^{ab} = \hat{e}(P, abP) = \hat{e}(abP, P)$.
- *Non-degeneracy*: There exists $P, Q \in \mathbb{G}_1 \ni \hat{e}(P, Q) \neq I_{\mathbb{G}_T}$, where $I_{\mathbb{G}_T}$ is the identity of $\mathbb{G}_T$.

- *Computability*: For all $P, Q \in \mathbb{G}_1$, there is an efficient algorithm to compute $\hat{e}(P, Q)$.

2.3.4 Computational Hardness Problems

When using standard cryptographic groups, the security relies on hardness assumption. The security of the proposed protocols lies on the following hardness problems:

Definition 2.4. *Elliptic Curve Discrete Logarithm Problem (ECDLP).* Given $Q \in_R \mathbb{G}$, where P is a generator of $\mathbb{G}$ and for unknown $a \in_R \mathbb{Z}_n^*$, the task of *ECDLP* is to find a such that $Q = aP$.

Definition 2.5. *Computational Diffie-Hellman (CDH) problem.* Given a generator P of $\mathbb{G}$ and (aP, bP) for unknown $a, b \in_R \mathbb{Z}_n^*$, the task of *CDH* problem is to compute abP.

Definition 2.6. *Decisional Diffie-Hellman (DDH) problem.* Given a generator P of $\mathbb{G}$ and (aP, bP, cP) for unknown $a, b, c \in_R \mathbb{Z}_n^*$, the task of *DDH* problem is to decide whether the equation $abP = cP$ holds.

Definition 2.7. *Gap Diffie-Hellman (GDH) problem.* Given a generator P of $\mathbb{G}$, (aP, bP) for unknown $a, b \in_R \mathbb{Z}_n^*$ and an oracle $DDH(aP, bP, cP)$, which returns 1 if and only if $abP = cP$, the task of *GDH* problem is to compute abP. The *GDH* assumption states that the probability of any polynomial-time algorithm solving the *GDH* problem is negligible.

If there exists a polynomial time algorithm $\mathcal{A}$ that can solve the *ECDLP*, then it can be used to solve the *CDH* problem in polynomial time. The algorithm first computes a from aP and then computes $a(bP) = abP$. Moreover, if the algorithm can solve *CDH* problem, it can also solve the *DDH* problem. Given a group element cP, $\mathcal{A}$ can determine whether $cP = abP$. Thus, the *DDH* problem reduces to the *CDH* problem, which again reduces to the *ECDLP*. Hence, the security of a protocol is strongest if it reduces to the *ECDLP*.

2.4 Key Agreement Protocols

Authenticated Key Agreement (AKA) is a mechanism in which two or more parties can mutually agree upon a shared secret over an adversarial-controlled network. Depending upon the number of participants in the process of establishing a shared secret key, the protocol is referred as two-party, tripartite or group key agreement protocol. Diffie and Hellman [46] introduced the concept of key agreement, which is based on public key

cryptography. It has a security flaw of man-in-the-middle attack, as it does not attempt to authenticate the communicating entities.

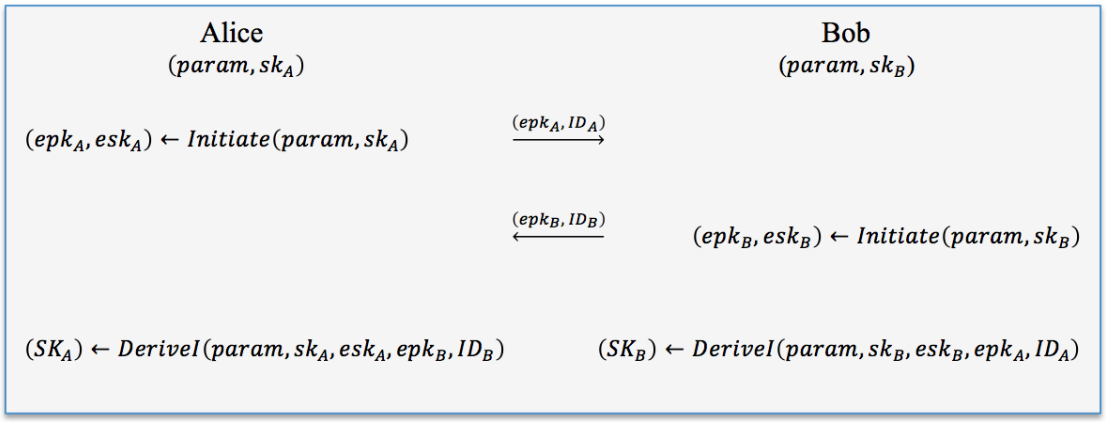

FIGURE 2.8: Two-party Authenticated Key Agreement

The mathematical description of key agreement protocol is as follows: AKA is a probabilistic polynomial-time interactive algorithm, which involves two entities *Alice* and *Bob*. The inputs are system parameters *param* for both *Alice* and *Bob*, plus (ID_A, sk_A, pk_A) for *Alice*, and (ID_B, sk_B, pk_B) for *Bob*. Here, sk_A and sk_B are the respective private-keys of *Alice* and *Bob*; ID_A is the identity of *Alice* and ID_B is the identity of *Bob*; pk_A and pk_B are the respective public-keys of *Alice* and *Bob*. Assuming, *Alice* is the initiator and *Bob* is the responder, then the key agreement algorithm further be consists of four polynomial time algorithms. These are defined as follows:

1. *Initiate (param, sk_A)*: This algorithm is run by the initiator *Alice*. This algorithm produces an ephemeral-public-key epk_A by taking system parameter *param* and private-key sk_A as input. Then, *Alice* send the epk_A to the responder *Bob* and store its corresponding ephemeral-private-key esk_A.

2. *Respond (param, pk_B)*: This algorithm is run by the responder *Bob*. This algorithm produces an ephemeral-public-key epk_B by taking system parameter *param* and public-key pk_B as input. Then, *Bob* send the epk_B to the initiator *Alice* and store its corresponding ephemeral-private-key esk_B.

3. *DeriveI (param, sk_A, esk_A, epk_B, ID_B)*: This algorithm is run by the initiator *Alice* takes system parameter *param*, private-key sk_A, ephemeral-private-key esk_A, ephemeral-public-key epk_B and identity of the responder ID_B as input to derive the session key SK_A with responder *Bob*.

4. *DeriveR (param, sk_B, esk_B, epk_A, ID_A)*: This algorithm is run by the responder *Bob* takes system parameter *param*, private-key sk_B, ephemeral-private-key esk_B,

ephemeral-public-key epk_A and identity of the responder ID_A as input to derive the session key sk_B with initiator *Alice*.

Figure 2.8 shows the diagrammatical view of the AKA protocol. Eventually, if the protocol does not fail, *Alice* and *Bob* will obtain a secret session key $sk_A = sk_B = sk$.

2.4.1 The Diffie-Hellman Key Exchange Protocol

In 1976, Diffie and Hellman [46] introduced a novel concept of key exchange in public key cryptography, which changes the vision of cryptography. The security of Diffie-Hellman key exchange protocol is based on Discrete Logarithm Problem (DLP), which assumed to be intractable. This is also known as the Diffie-Hellman (DH) problem. *Alice* and *Bob* are two parties, who want to establish a session key, denoted as A and B, respectively.

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g. *Alice* chooses a number $a \in_R Z_n^*$ and sends $T_A = g^a \bmod q$ to *Bob*. Similarly, *Bob* chooses a number $b \in_R Z_n^*$ and sends $T_B = g^b \bmod q$ to *Alice*. *Alice* computes $K_{AB} = {T_B}^a \bmod q$ and *Bob* computes $K_{BA} = {T_A}^b \bmod q$. The session key is $K_{AB} = K_{BA} = g^{ab}$.

Alice		Bob
Chooses $a \in \mathbb{Z}_n^*$		Chooses $b \in \mathbb{Z}_n^*$
Computes $T_A = g^a \bmod q$		Computes $T_B = g^b \bmod q$
	$\xrightarrow{T_A}$	
	$\xleftarrow{T_B}$	
Computes $K_A = {T_B}^a \bmod q$		Computes $K_B = {T_A}^b \bmod q$

FIGURE 2.9: Diffie-Hellman Key Exchange Protocol

Figure 2.9 shows the key agreement phase of Diffie-Hellman key exchange protocol. The generation of key refers the computational hardness problem, which states that it is difficult to compute g^{ab}, even if the values of q, g, g^a and g^b is known. This is known as the Computational Diffie-Hellman (CDH) problem. However, if the session key has been listened passively, and substituted by an adversary with his own block of bits from g^{ab} and known values of g^a and g^b. Then, this is known as Decisional Diffie-Hellman (DDH) problem.

2.4.2 Security Attributes

The Adversary can break a key agreement protocol in a number of ways, so, the construction of key agreement protocol is done in such a way that it can resists various attacks by

possessing few security attributes. It is desired for authenticated key agreement protocol to possess the following security attributes:

1. *Known-Key Secrecy (K-KS)*: A key agreement algorithm is resistant to Known-Key Secrecy (K-KS) attack if it is secure under the assumption that the generation of session keys at each round of key agreement should be independent and unique. That is, the compromise of one session key does not allow an adversary to expose other session keys.

2. *Unknown Key-Share (UK-S)*: A key agreement algorithm is resistant to Unknown Key-Share (UK-S) attack if it is secure under the assumption that an entity cannot be obliged to share a session key with a different party to the one intended without their knowledge. That is, *Alice* should not be able to be obliged to share a key with adversary when in fact *Alice* thinks that she is sharing the key with *Bob*.

3. *Forward Secrecy (FS)*: A key agreement algorithm is resistant to Forward Secrecy (FS) attack if it is secure under the assumption that if the long-term private keys of one or more of the entities are compromised, the session keys used in the past should not be recovered.

    - *Partial Forward Secrecy (PaFS)*: if the long-term private key of one participating entity in one round of key agreement is compromised without compromising previously established session keys
    - *Perfect Forward Secrecy (PeFS)*: if the long-term private keys of both participating entities in one round of key agreement are compromised without compromising previously established session keys
    - *Weak Perfect Forward Secrecy (wPeFS)*: if all long-term private keys are known, but the attacker was not actively involved in choosing ephemeral keys during the sessions of interest.
    - *KGC Forward Secrecy (KGCFS)*: if the master private key of the KGC gets compromised then it cannot affect the secrecy of previously established session keys (while KGCFS implies PeFS in the ID-based setting, it is a particular property defined for ID-based AKA protocols in the escrowless mode).

4. *No-Key Control (N-KC)*: A key agreement algorithm is resistant to No-Key Control (N-KC) attack if it is secure under the assumption that neither the participants nor adversary can force the session key to be a preselected value, or predict the value of the session key.

5. *Key-Compromise Impersonation (K-CI)*: A key agreement algorithm is resistant to Key-Compromise Impersonation (K-CI) attack if it is secure under the assumption that the compromise of a long-term private key of an entity does not allow adversary to impersonate entity's identity to other entities in a key agreement round to obtain session key. That is, adversary impersonates as *Alice* to obtain the session key with *Bob*.

6. *Unleakage of Ephemeral Keys (U-EK)*: If the attacker has access to the ephemeral keys of a given protocol run, he should be unable to determine the corresponding session key. Adversaries may gain access to this information through a side-channel attack or use of a weak random number generator; alternatively this information might be stored insecurely.

2.4.3 Authenticated Key Agreement

There are two types of key authentication, Implicit Key Authentication, which ensures that none other than the intended entities can obtain the value of the secret key, and Explicit Key Authentication, which ensures that each participating entity that the intended other entities have actually computed the key [56]. Former is known as authenticated key agreement protocol while, later is called authenticated key agreement with key confirmation protocol. In Authenticated Key Agreement (AKA) protocols, *Alice* will be assured that *Bob* is the only party, whom is able to compute the shared key. Similarly, *Bob* would also be assured that *Alice* has computed the shared key. Then, this can be achieved by applying key confirmation methods.

2.4.4 Provable Security

In 1984, Goldwasser and Micali [57] introduced the concept of provable security. A protocol is said to be provable secure if there is a polynomial reduction proof from a known hard computational problem to an attack against the security of the protocol. Thus, if there is a polynomially bounded adversary that breaks the protocol, then the problem assumed to be hard and can be solved in polynomial time. The protocol can be provable secure through the following steps: (i) define the goals of the protocol and then prove that it meets those goals, (ii) define the capabilities of the adversary by specifying a security model, (iii) define attacks that it should resist and (iv) provide a security proof of the protocol. The security proofs of many authenticated key agreement protocols require the hash functions, which are modeled as random oracles. This approach, commonly referred to as the random oracle model, was first formulated by Bellare and Rogaway [58] and further streamlined by Blake et al. [59].

The security of AKA protocols should be formally proven before they are deployed. Bellare and Rogaway [58] presented the first formal security model for authentication of key agreement for symmetric key cryptography (shared key), known as Bellare-Rogaway (BR) model. The adversary is responsible for the whole communication. Adversary simulates the protocol run and tries to win the game by guessing the output of the session key with a randomly generated value. Later, Blake et al. [59] extended the BR model for public key cryptography, known as Blake-Johnson-Menezes (BJM) model. In 2005, Kudla and Paterson [60] presented a modified Bellare-Rogaway (mBR) model for authenticated key agreement protocols. This model follows closely the model of Bellare et al. [61], which extends the original Bellare-Rogaway model [58] for providing the security of protocols based on some form of Gap assumption.

Canetti and Krawczyk [62] designed a model, called Canetti-Krawczyk (CK) Model, to analyze the use of key establishment protocols with symmetric key setting and authentication functions. This is an extension of BR model with the replacement of adversarial model derived by Bellare and Rogaway [58]. The CK model is not resistant to leakage of ephemeral keys and key compromise to impersonation attack.

In 2007, LaMacchia et al. [63] presented a new security model known as the extended Canetti-Krawczyk (eCK) model for AKA protocols based on Public Key Infrastructure (PKI). The eCK model is a considerably strong security model, which captures many desirable security properties including Key-Compromise Impersonation (K-CI) resilience, weak Perfect Forward Secrecy (wPeFS) and ephemeral secrets reveal resistance etc. In other words, the eCK model captures all the security properties of the CK model, as well as resilience to KCI attacks, malicious insiders, and known session-specific temporary information attacks. The eCK model is based on CK model but replaces the notion of matching sessions with matching conversations derived in BR model [58]. In eCK model, adversaries can register public keys on behalf of an entity.

Swanson and Jao [64] introduced the first formal security model for certificateless authenticated key agreement protocol. The Swanson and Jao's model is an extended version of the model presented by LaMacchia et al. [63] (eCK). There are two types of adversaries involved in the certificateless key agreement protocols, type I adversary (without the master secret key), who can replace public keys, and type II adversaries (with the master secret key), who are not allowed to replace public keys. The model introduced a new security attribute: resistance to leakage of ephemeral information.

Based on Swanson and Jao [64] model, Lippold et al. [16] presented a strong security model for certificateless authenticated key agreement protocols. They provided the first formal proof for a strongly secure certificateless key agreement protocol in the random oracle model. The model fulfilled all notions of security and resists recent

attacks on CTAKA protocols. Lippold et al.'s model is different in the sense that after the key replacement of public-key of a user by an adversary, the user will use the new public/private key-pair for the rest of the game, while in Swanson and Jao's model, user go along with its original public/private key-pair. Lipold et al. also proved that the certificateless key agreement protocol could not be constructed by combining ID-based key agreement protocol with a public-key based key agreement protocol.

Liang et al. [14] presented a security model for identity based authenticated key agreement protocols. The model is an extended version of eCK model [63]. Table 2.1 shows the comparison of security models with respect to the security attributes.

TABLE 2.1: Comparison of Security Models

Security Model	**K-KS**	**U-KS**	**Pe-FS**	**N-KC**	**K-CI**	**U-EK**
Bellare and Rogaway (BR) [58]	✓	✓	✓	✓	×	×
Blake et al. (BJM) [59]	✓	✓	✓	✓	×	×
Kudla and Paterson (mBR) [60]	✓	✓	✓	✓	×	×
Canetti and Krawczyk (CK) [62]	✓	✓	✓	✓	×	×
LaMacchia et al. (eCK) [63]	✓	✓	✓	✓	✓	×
Swanson and Jao [64]	✓	✓	✓	✓	✓	✓
Lippold et al. [16]	✓	✓	✓	✓	✓	✓
Liang et al. [14]	✓	✓	✓	✓	✓	✓

K-KS: Known-Key Secrecy,U-KS: Unknown Key-Share, Pe-FS: Perfect Forward Secrecy, N-KC: No-Key Control, K-CI: Key-Compromise Impersonation, U-EK: Unleakage of Ephemeral Keys.

2.5 Identity-based Two-party Authenticated Key Agreement (ID-2PAKA) Protocol

2.5.1 Formal Model

An ID-2PAKA protocol consists of three polynomial-time algorithms: *Setup*, *Extract* and *Key-Agreement*. These algorithms are defined as follows:

Setup: For a given security parameter k as input, this algorithm returns the system parameters $param$ and keeps master key s secret.

Extract: For given system parameters $param$, master key s and a user's identity ID_i as inputs, this algorithm returns a private-key s_i for user's identity ID_i.

Key-Agreement: This is a probabilistic polynomial-time interactive algorithm. This performs the session between two entities A and B and agrees up on a session key (see figure 2.10). For given system parameters $param$ for both A and B ((ID_A, s_A, pk_A) for A, and (ID_B, s_B, pk_B) for B), this algorithm returns session key sk. s_A and s_B are

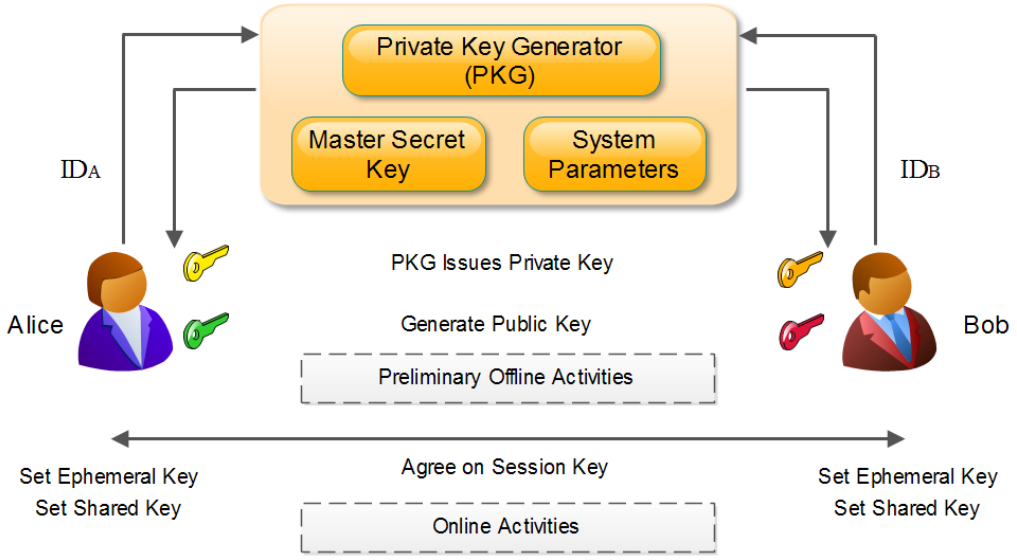

FIGURE 2.10: Identity-based Key Agreement

the respective private-keys of A and B; ID_A is the identity of A and ID_B is the identity of B; pk_A and pk_B are the respective public-key of A and B.

2.5.2 Security Model

There is a single kind of adversary associated with ID-2PAKA protocol. The adversary is represented as $\mathcal{A}$, who is a dishonest user. Private Key Generator (PKG) is assumed to be secure in identity-based infrastructure. The adversary $\mathcal{A}$ does not know the master key, but can replace the public-keys of any entity with a value of own choice.

Liang et al. [14] presented a security model for ID-based two-party authenticated key agreement protocols, which is an adaption of the original eCK model presented by [63] for traditional PKI-based authenticated key agreement protocols.

Let $U = \{ID_1, \cdots, ID_n\}$ be a set of parties with each party is having identity ID_i. The key agreement protocol may run between any two parties from set U. Each party having identity $ID_i \in U$ may execute a polynomial number of key agreement protocol sessions in parallel. The adversary $\mathcal{A}$ has full control of the communication between two parties in the network. The adversary may eavesdrop, delay, replay, alter and insert messages.

In the security model, $\prod_{i,j}^{s}$ represents the s^{th} session runs for party i with intended partner party j. A session $\prod_{i,j}^{s}$ may enters an *accepted* state, when it computes a session-key $sk_{i,j}^{s}$ for party i and j. A session may terminate without entering into an *accepted* state. Two sessions $\prod_{i,j}^{s}$ and $\prod_{j,i}^{t}$ are called matching sessions, if they have the same session identity. The session $\prod_{i,j}^{s}$ is assigned a partner $ID_{prid} = (ID_i, ID_j)$. Let *comms*

denote the transcript of the messages exchanged between the owner and the peer during the session. Two sessions $\prod_{i,j}^{s}$ and $\prod_{i,j}^{t}$ are considered matching if they have the same *comms* (and *prid*), though in different order.

The security model (game) has two phases. During the first phase, Challenger $\mathcal{C}$ responds the the adversary $\mathcal{A}$'s following queries in any order:

1. *Create(i)*: On receiving such a query $\mathcal{C}$ generates the private-key for participant i with identity ID_i.

2. *Reveal-Static-Private-Key(ID_i)*: This query returns the static private-key of ID_i to adversary.

3. *Reveal-Ephemeral-Key($\prod_{i,j}^{s}$)*: This query returns the ephemeral secret in session $\prod_{i,j}^{s}$ of party ID_i to adversary.

4. *Reveal-Session-Key($\prod_{i,j}^{s}$)*: If the session has not accepted, it returns $\perp$, otherwise it reveals the accepted session key to the adverasry.

5. *Send($\prod_{i,j}^{s}, m$)*: The adversary sends the message m to party ID_i on behalf of party ID_j for the execution of session $\prod_{i,j}^{s}$ and gets a response according to the protocol specification (i.e. If $m =$, party ID_i initiates the session $\prod_{i,j}^{s}$). This query allows $\mathcal{A}$ to order ID_i to start a session $\prod_{i,j}^{s}$ with ID_j and to provide communications from ID_j to ID_i. In general, we require $i \neq j$, i.e., a party will not run a session with itself.

   Once the adversary $\mathcal{A}$ decides that the first phase is over, it starts the second phase by choosing a fresh session $\prod_{i,j}^{s}$ and issuing a *Test*($\prod_{i,j}^{s}$) query, where the fresh session and *Test* query are defined as follows:

   Definition 2.8. *Fresh Session.* A session $\prod_{i,j}^{s}$ owned by a party ID_i with party ID_j is fresh if:

   - Both parties ID_i and ID_j are honest.
   - $\prod_{i,j}^{s}$ has accepted and unopened.
   - There is no opened session $\prod_{i,j}^{t}$, which has a matching conversation to $\prod_{i,j}^{s}$.
   - None of the following conditions hold:
     - ID_j is engaged in session $\prod_{i,j}^{t}$ matching to $\prod_{i,j}^{s}$ and the adversary $\mathcal{A}$ reveals either both the static private-key of ID_i and the ephemeral secret of $\prod_{i,j}^{s}$, or both the static private-key of ID_j and the ephemeral secret of $\prod_{i,j}^{t}$.

– No session matching to $\prod_{i,j}^{s}$ exists and the adversary $\mathcal{A}$ reveals either the static private-key of ID_j, or both the static private-key of ID_i and the ephemeral secret of $\prod_{i,j}^{s}$.

6. *Test*($\prod_{i,j}^{s}$): The input session $\prod_{i,j}^{s}$ must be fresh. Challenger $\mathcal{C}$ flips a fair coin $b \in_R \{0,1\}$ and returns the session-key held by $\prod_{i,j}^{s}$, if $b = 0$, the adversary $\mathcal{A}$ is given the session key, otherwise a uniformly chosen random value from the distribution of valid session keys is returned to $\mathcal{A}$.

   After *Test*($\prod_{i,j}^{s}$) has been issued, the adversary may continue querying except the test session $\prod_{i,j}^{s}$.

   At the end of the game, the adversary outputs a guess b' for b. If $b' = b$, we say that the adversary wins. The adversary's advantage in winning the game is defined as $Adv_{\mathcal{A}}(k) = |Pr[\mathcal{A} \text{ wins }]\frac{1}{2}|$, where k is a security parameter.

   Definition 2.9. *Secure AKA Protocol.* An AKA protocol is secure (in the eCK model) if matching sessions compute the same session keys and for any PPT $\mathcal{A}$ the advantage in winning the above game is negligible.

In an authenticated key exchange protocol, two parties exchange information and compute a secret key as a function of at least four pieces of secret information: their own long-term and ephemeral secret keys and the other partys static and ephemeral secret keys. An adversary can reveal any subset of the four secret information, which does not contain both the long-term and ephemeral secrets of one of the parties. AKE test sessions (sessions which are subject to attack by an adversary) are of two types. Let Alice and Bob be the participants of the test session. In sessions of the first type (passive sessions), the adversary does not cancel or modify communications between the two parties. In sessions of the second type (active sessions), the adversary may forge the communication of the second party.

2.6 Certificateless Two-party Authenticated Key Agreement (CTAKA) Protocol

2.6.1 Formal Model

A CTAKA protocol consists of six polynomial-time algorithms: *Setup*, *Partial-Private-Key-Extract*, *Set-Secret-Value*, *Set-Private-Key*, *Set-Public-Key* and *Key-Agreement*. These algorithms are defined as follows:

Setup: This algorithm takes security parameter k as input and returns the system parameters $param$ and master key s.

Partial-Private-Key-Extract: This algorithm takes the system parameters $param$, master key s and a user's identity ID_i as inputs and returns a partial-private-key $D_i = (s_i, R_i)$.

Set-Secret-Value: This algorithm takes the system parameters $param$ and a user's identity ID_i as inputs, and generates a secret-value x_i.

Set-Private-Key: This algorithm takes the system parameters $param$, a user's partial-private-key D_i and his secret-value x_i as inputs, and outputs the full private-key $sk_i = (s_i, x_i)$.

Set-Public-Key: This algorithm takes the system parameters $param$ and a user's secret-value x_i as inputs, generates a public-key P_i for the user and outputs the full private-key $pk_i = (R_i, P_i)$.

Key-Agreement: This is a probabilistic polynomial-time interactive algorithm, which involves two entities A and B. The inputs are system parameters $param$ for both A and B, plus (ID_A, sk_A, pk_A) for A, and (ID_B, sk_B, pk_B) for B. Here, sk_A and sk_B are the respective private-keys of A and B; ID_A is the identity of A and ID_B is the identity of B; pk_A and pk_B are the respective public-key of A and B. Eventually, if the protocol does not fail, A and B will obtain a secret session-key sk. All algorithms except *Key-Agreement* are preliminary offline activities and performed before deployment as shown in figure 2.11.

2.6.2 Security Model

There are two kinds of adversaries with different capabilities associated with CTAKA protocol. The Type I adversary $\mathcal{A}_I$ acts as a dishonest user while the Type II adversary $\mathcal{A}_{II}$ acts as a malicious KGC. $\mathcal{A}_I$ does not know the master key, but $\mathcal{A}_I$ can replace the public-keys of any entity with a value of their choice. $\mathcal{A}_{II}$ knows the master key, but cannot replace any user's public-key.

Let $\prod_{i,j}^{s}$ represent the s^{th} session, which runs for party i with intended partner party j. A session $\prod_{i,j}^{s}$ enters an *accepted* state when it computes a session-key $sk_{i,j}^{s}$. A session may terminate without ever entering into an *accepted* state. Two sessions $\prod_{i,j}^{s}$ and $\prod_{j,i}^{t}$ are called matching sessions, if they have the same session identity. Lippold et al. [16] presented the strengthened version of Swanson and Jao's model [64], which is in turn a transformed version of original eCK model [63] from the traditional PKC setting to the

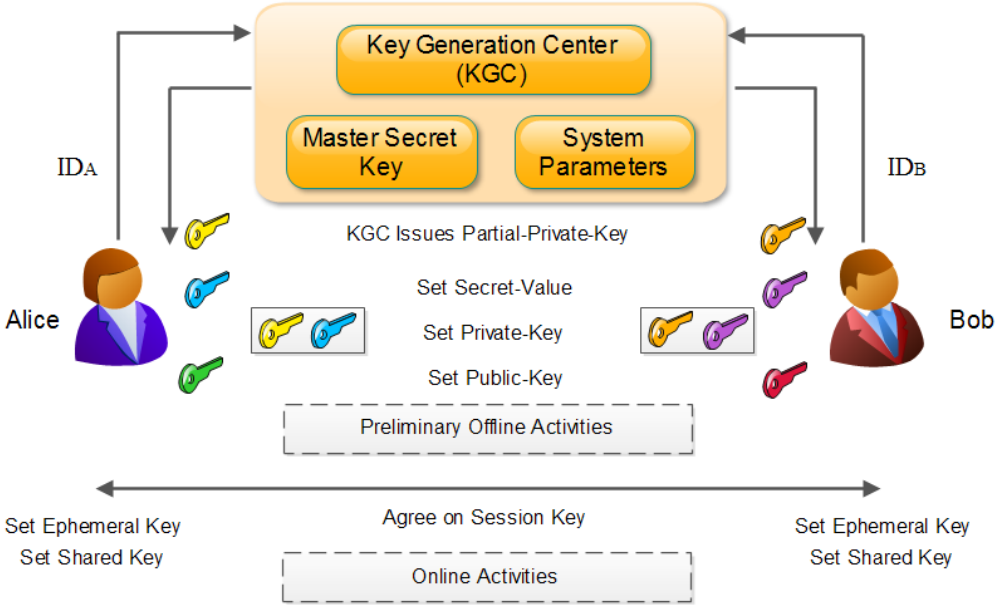

FIGURE 2.11: Certificateless Key Agreement

CL-PKC setting. The eCK model in the CL-PKC setting is defined by the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$. The game runs in two phases. During the first phase, the adversary $\mathcal{A}$ is allowed to issue the following queries in any order:

1. *Create(i)*. On receiving such a query, $\mathcal{C}$ generates the public-private-key pair for participant i with identity ID_i.

2. *Reveal-Master-Key*. $\mathcal{C}$ gives the master key to $\mathcal{A}$.

3. *Reveal-Session-Key*$(\prod_{i,j}^{s})$. If the session has not been accepted, $\mathcal{C}$ returns $\perp$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ return the accepted session-key to $\mathcal{A}$.

4. *Reveal-Partial-Private-Key(i)*. $\mathcal{C}$ returns participant i's partial-private-key to $\mathcal{A}$.

5. *Reveal-Secret-Value(i)*. $\mathcal{C}$ returns participant i's secret-value to $\mathcal{A}$.

6. *Replace-Public-Key*(i, pk). $\mathcal{C}$ replaces participant i's public-key with the value chosen by $\mathcal{A}$.

7. *Reveal-Ephemeral-Key*$(\prod_{i,j}^{s})$. $\mathcal{C}$ returns participant i's ephemeral-secret-key to $\mathcal{A}$.

8. *Send*$(\prod_{i,j}^{s}, m)$. The adversary sends the message m to the session $(\prod_{i,j}^{s})$ and gets a response according to the protocol specification.

   Once the adversary $\mathcal{A}$ decides that the first phase is over, it starts the second phase by choosing a fresh session $(\prod_{i,j}^{s})$ and issuing a *Test* $(\prod_{i,j}^{s})$ query, where the fresh

session and *Test* query are defined later. The Type I adversary $\mathcal{A}_I$ could get any user's secret-value, since it can replace the public-key of any entity with a value of its choice. The Type II adversary $\mathcal{A}_{II}$ could get any user's partial-private-key, since it has access to the master key.

Definition 2.10. *Freshness for the CTAKA Protocol Against a Type I Adversary.* Let for instance $\prod_{i,j}^{s}$ be a completed session, which is executed by an honest party i with another honest party j. We define $\prod_{i,j}^{s}$ to be fresh if none of the following three conditions holds:

- The adversary $\mathcal{A}_I$ reveals the session-key of $\prod_{i,j}^{s}$ or of its matching session (if the latter exists).
- j is engaged in $\prod_{j,i}^{t}$ the session matching to $\prod_{i,j}^{s}$ and $\mathcal{A}_I$ either reveals both i's partial-private-key and $\prod_{i,j}^{s}$'s ephemeral-secret or both j's partial-private-key and $\prod_{j,i}^{t}$'s ephemeral-secret.
- No session matching to $\prod_{i,j}^{s}$ exists and $\mathcal{A}_I$ either reveals both i's partial-private-key and $\prod_{i,j}^{s}$'s ephemeral-secret or j's partial-private-key.

Definition 2.11. *Freshness for the CTAKA Protocol Against a Type II Adversary.* Let for instance $\prod_{i,j}^{s}$ be a completed session, which is executed by an honest party i with another honest party j. We define $\prod_{i,j}^{s}$ to be fresh if none of the following three conditions holds:

- The adversary $\mathcal{A}_{II}$ reveals the session-key of $\prod_{i,j}^{s}$ or of its matching session (if the latter exists).
- j is engaged in $\prod_{j,i}^{t}$, the session matching to $\prod_{i,j}^{s}$, and $\mathcal{A}_{II}$ either reveals both i's secret-value and $\prod_{i,j}^{s}$'s ephemeral-secret or both j's secret-value and $\prod_{j,i}^{t}$'s ephemeral-secret.
- No session matching to $\prod_{i,j}^{s}$ exists and $\mathcal{A}_{II}$ either reveals both i's secret-value and $\prod_{i,j}^{s}$'s ephemeral-secret or j's secret-value.

9. *Test*($\prod_{i,j}^{s}$). At some point, $\mathcal{A}$ may choose one of the oracles, say $\prod_{i,j}^{s}$, to ask a single *Test* query. This oracle must be fresh. To answer the query, the oracle flips a fair coin $b \in \{0, 1\}$, and returns the session-key held by $\prod_{i,j}^{s}$ if $b = 0$, or a random sample from the distribution of the session-key if $b = 1$. At the end of the game, $\mathcal{A}$ must output a guess bit b'. $\mathcal{A}$ wins if and only if $b' = b$. $\mathcal{A}$'s advantage for winning the above game, denoted by $Adv_{\mathcal{A}}(k)$, is defined as $Adv_{\mathcal{A}}(k) = |Pr[b' = b] - \frac{1}{2}|$, where k is a security parameter.

Definition 2.12. A CTAKA protocol is said to be secure if:

TABLE 2.2: Nine Strategies to Break CTAKA Protocol

Case	**Strategies**	**Type of Adversary Attacked**
1	$\mathcal{A}$ may learn neither x_i nor x_j	$\mathcal{A}_{II}$
2	$\mathcal{A}$ may learn neither t_i nor t_j	$\mathcal{A}_I, \mathcal{A}_{II}$
3	$\mathcal{A}$ may learn neither x_i nor t_j	$\mathcal{A}_{II}$
4	$\mathcal{A}$ may learn neither t_i nor x_j	$\mathcal{A}_{II}$
5	$\mathcal{A}$ may learn neither s_i nor x_j	N.A.
6	$\mathcal{A}$ may learn neither x_i nor s_j	N. A.
7	$\mathcal{A}$ may learn neither s_i nor t_j	$\mathcal{A}_I$
8	$\mathcal{A}$ may learn neither t_i nor s_j	$\mathcal{A}_I$
9	$\mathcal{A}$ may learn neither s_i nor s_j	$\mathcal{A}_I$

- In the presence of a benign adversary in sessions $\prod_{i,j}^{s}$ and $\prod_{j,i}^{t}$ both oracles always agree on the same session-key, and this key is distributed uniformly at random.
- For any adversary $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{II}\}$, $Adv_{\mathcal{A}}(k)$ is negligible.

The adversary $\mathcal{A}(\mathcal{A}_I/\mathcal{A}_{II})$ is allowed to corrupt at most two of three secrets for each party. These are: s_i, partial-private-key for user identity ID_i computed by KGC; x_i, user's chosen secret-value for identity ID_i and t_i, ephemeral-secret-key for user identity ID_i. So, there are nine possibilities to break the protocol according to Lipold et al. [16]. $\mathcal{A}$ may learn (i) neither x_i nor x_j; (ii) neither t_i nor t_j; (iii) neither x_i nor t_j; (iv) neither t_i nor x_j; (v) neither s_i nor x_j; (vi) neither x_i nor s_j; (vii) neither s_i nor t_j; (viii) neither t_i nor s_j and (ix) neither s_i nor s_j, key agreement between two users having identity ID_i and ID_j as shown in Table 2.2. Strategies (v) and (vi) are not feasible, as, it includes the partial-private-key and secret-value of the users. But, not all the nine strategies can be applied to break the protocol. For Type I adversary $\mathcal{A}_I$, which is also known as malicious user, does not know the partial-private-key s_i for user identity ID_i. So, $\mathcal{A}_I$ can break the protocol with four strategies (shown in Table 2.2). For Type II adversary $\mathcal{A}_{II}$, which is also known as malicious KGC, does not know the secret-value x_i for user identity ID_i. So, $\mathcal{A}_{II}$ can break the protocol with four strategies (shown in Table 2.2).

The proofs of lemmas are divided into two cases, first when the test-session has a matching session owned by another honest party and other when no honest party owns a session matching with the test-session. The former case is further divided into four cases, according to Type I and Type II adversary.

2.7 Signcryption

Signcryption was introduced by Yuliang Zheng [65] in 1997, which provides the encryption and signature in a single logical step with a cost lot lower than that required by the traditional signature followed by encryption to obtain confidentiality, integrity, authentication and non-repudiation and proposed methodology based on concept has been discussed in Chapter 6.

2.8 Summary

In this chapter, we presented a review of symmetric key management schemes for WSN. We discussed cryptographic primitives, which includes the introduction of hash functions, elliptic curve cryptography, complexity assumptions used for the proposed algorithms. We also presented key agreement protocols; the formal and security model for identity-based and certificateless two-party authenticated key agreement protocols and introduction to signcryption. In the next chapter, we will discuss the experimental environment, which is common for various algorithms developed and tested.

Chapter 3

Experimental Setup

The implementation of public key algorithms on WSN is itself a challenging task. In this chapter, we address various issues for the implementation of algorithms for WSN such as selection field, selection of curve, cryptographic standards, the key-size etc. This chapter provides an overview of sensor motes (micro-controller ATmega128L) used in WSN, the softwares used to operate and program these devices such as TinyOS (the operating system for sensor motes); NesC language; RELIC-toolkit (cryptographic library). Finally, the simulator AVRORA, a cycle-accurate simulator used to analyze energy consumption and running time will be discussed. [1] [2]

3.1 Specifications of Target Platform

3.1.1 Wireless Sensor Networks - Hardware/Software

Wireless Sensor Network (WSN) bridges the gap of digital world to the physical world. Densely deployed sensor motes in an application area build a WSN. In most deployments, the sensor motes have self-organizing capabilities, to form an appropriate structure in order to collaboratively perform a particular task. The size and low-cost of sensor mote bound them in memory, CPU power, and energy supply. WSN generally, creates a wireless ad-hoc network with multihop communication patterns. The life span of the battery should be for several months up to years. Hence, it requires low power consumption for running any application. Communication causes a lot of power consumption, so the traffic should be as low as possible.

[1] The major findings of this chapter has been published as "*Optimized Elliptic Curve Cryptography for Wireless Sensor Networks*," in the Proceedings of 2nd IEEE International Conference on Parallel Distributed and Grid Computing (PDGC'12), pp. 89-94, Dec 6-8, 2012.

[2] The major findings of this chapter has been accepted for publication as "*Feasibility of Public Key Cryptography in Wireless Sensor Network*," **Journal of Theoretical Physics and Cryptography**.

A sensor mote is consists of mainly five components embedded on a same circuit board: (i) a low-powered microprocessor and a limited amount of RAM (for runtime memory), and ROM (for code space); (ii) sensors (light, humidity, vibration, motion, etc.); (iii) a low-powered radio, for the transmission of data from the mote; (iv) battery to provide power for the entire board and (v) user I/O block to provide a limited debug interface for programmers. Most commonly used motes are MicaZ, Telosb, Tmote Sky, iMote2 etc. Various companies manufacture sensor motes, includes, Crossbow, Texas Instrument, DUST, Moteiv etc.

Sensor motes communicate by using a multihop routing algorithm, which transmits data from the source mote to the destination mote through one or more intermediate motes. The network stack of two sensor motes consists of five layers: (i) the radio layer provides the radio hardware to communicate over the wireless medium, (ii) The Hardware Abstraction Layer (HAL) provides the mechanism for the PHY layer to talk to the various radio chips in the radio layer, (iii) PHY layer is based on the IEEE 802.15.4 protocol, (iv) MAC layers also based on the IEEE 802.15.4 protocol and (v) Application and Network layer to provide functions and capabilities that the PHY and MAC layers cannot provide such as multihop routing, and security.

3.1.2 Introduction to MICAz Mote

The MICAz mote [17] has been developed at the University of California, Berkeley (see figure 3.1). The motes are manufactured and distributed by Crossbow Technology Inc. USA. MICAz mote is used for enabling low-power WSN, which is having a range of 2.4 - 2.48 GHz frequency transceiver, consists of three main components: ATmega128L microcontroller, AT45DB041 4-Mbit serial FLASH chip, and Chipcon CC2420 radio frequency transceiver shown in figure 3.2. In addition, the board is equipped with an interface "51-Pin Expansion Connector" used to enable various sensors and data acquisition boards, available at Crossbow Technology, Inc. MICAz mote is also known as MPR2400.

The ATmega128L is a low-power 8-bit AVR microcontroller provided by Atmel Corporation. The hardware specifications of MICAz mote are shown in table 3.1. All motes feature Atmel AT45DB041 4-Mbit serial flash, which is connected to one of the USART on the ATMega128L. It can be used for storing data, measurements, and other user-defined information. The AVR microcontroller product line is designed for low power consumption and high performance.

FIGURE 3.1: MICAz mote [17]

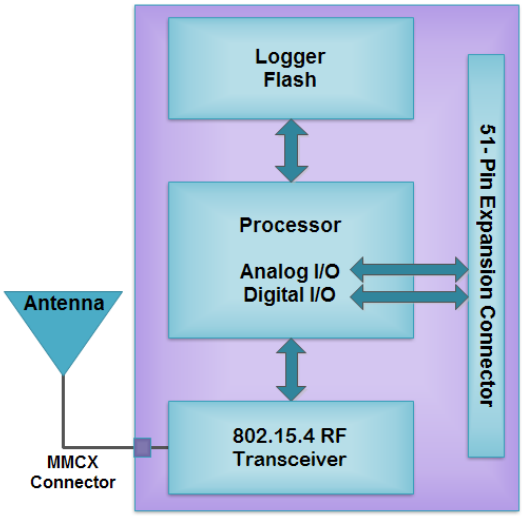

FIGURE 3.2: Block Diagram of MICAz Mote [17]

3.1.2.1 Programming a Mote

For programming a mote, an operating system TinyOS is required. The command "make micaz" is used for the compilation of a program for MICAz mote in TinyOS. This command generates an executable file in the same directory and helps in debugging the program. The command "make micaz reinstall mib510,/dev/USBx" is used to install a program on MICAz mote, which compiles the program and then uploads it to the mote. mib510 is the PC interface board mounted on /dev/USBx.

The following libraries are required for programming a MICAz mote:

- avr-libc is a C library for AVR microcontroller that was designed to conform the standard C library as described by the ANSI X3.159-1989 and ISO/IEC 9899:1990

Table 3.1: MICAz Hardware Specification [17]

CPU Performance	Atmel Atmega 128L 7.37 MHz, 8 bit 128 kB Program flash memory 512 kB Serial Flash 4K Configuration EEPROM (SRAM)
Sensor Board Interface	51 pin connector for connection to a sensor board UART Serial Communications 10 bit ADC, Digital I/O, I2C, SPI and 3 Diagnostic LEDs for User Interface (Red, green and yellow) 8 mA and < 15 μA Current Draw at Active mode and Sleep mode resp.
RF Transceiver	CC2420 Chip 2400 MHz to 2483.5 MHz Frequency Band MMCX Antenna Connector 250 kbps Transmit Data Rate (max) -24 dBm to 0 dBm RF Power -90 dBm (min), -94 dBm (typ) Receive Sensitivity 75 m to 100 m Outdoor Range 20 m to 30 m Indoor Range 19.7 mA Current Draw at Receive mode 11 mA, 14 mA, 17.4 mA Current Draw at TX (-10, -5, 0) dBm 20 μA Current Draw at Idle Mode, voltage regular on 1μA Current Draw at Sleep Mode, voltage regular off
Flash Data Logger Memory	AT45DB014B Chip SPI Connection Type
Electromechanical	2×AA batteries 2.7 V - 3.3 V Molex connector provided
Sensing	Temperature Sensor Humidity Sensor Barometric Sensor Visible Light Sensor Visible to IR Sensor Pressure Sensor Acceleration/Seismic Sensor Acoustic Sensor Magnetic Sensor

(ANSI- C) standard, as well as parts of their successor ISO/IEC 9899:1999 (C99) [66].

- The GNU Binutils [67] and the GNU Compiler Collection [68] (GCC) contains the avr-assembly and the avr-gcc for compiling assembly and C. The assembly and linker are usually not invoked manually, but rather using the C compiler front end (avr-gcc) that in turn will call the assembly and linker as required. This procedure is done with the help of a makefile containing all parameters.

3.1.3 Introduction to TinyOS

TinyOS [18] is a free and open source component-based operating system specifically designed for WSN. It was started as a project of the DARPA NEST program, developed at the University of California, Berkeley in 2001. Now, it is an alliance among the University of California, Berkeley, Intel Research and Crossbow Technology. TinyOS is an object-oriented, event driven operating system. The aim of TinyOS is to support the concurrency intensive operations required by WSN with minimal hardware requirements. It allows the customization and adaption to several hardware platforms. It is implemented in nesC (network embedded system C) programming language, which is a dialect of C language. NesC is a part of the TinyOS project developed to simplify and reduce race conditions, which can be exhibited in C parallel programming. The additive tools have a support of Java and shell script front-ends like TOSSIM, which are having a support of python. Other associated libraries and tools like nesC compiler and AVR binutils toolchains, are written in C language. As it is a open source, it has created a broad user community with thousands of developers. It supports microprocessors with 8-bit architectures with 2KB of RAM to 32-bit processors with 32MB of RAM or more.

Since, in sensor networks the direct user interaction with the nodes or the embedded devices is minimum as compared to the general PC usages, so the static approach in nesC is justified. The wiring occurs at compile time, which minimizes the RAM usage at runtime. With TinyOS, it has become possible to write code with minimum memory storage requirements as well as low power consumption, which are the essential requirements for WSN.

3.1.4 RELIC-Toolkit - Cryptographic Library

Writing algorithms for the cryptographic operations is not only time consuming but also there is fair chance that the efficiency may not be up to the mark. Various libraries are available (both open source and proprietary) are required for algorithms for the efficient implementations of cryptographic operations. MIRACL and RELIC are two libraries, which have full support for RSA, ECC and pairing based cryptography. RELIC is an open source library, which has been developed and maintained by UNICAMP whereas MIRACL is from Shamus Software. The literature of implementation in PKC has mainly focused on these two libraries.

- *MIRACL*: The algorithms in MIRACL (Multiprecision Integer and Rational Arithmetic C/C++ Library) [69] works on 18 characters sized blocks. Other than RSA and ECC (for both prime fields and binary fields) it also supports Diffie-Hellman

Key exchange and DSA digital signature. Though very popular among cryptographers, MIRACL has not been used much in the embedded systems where space constraint is very high. MIRACL is the most efficient library for 80x86/Pentium platform.

- *RELIC*: The primary focus of RELIC [20] is efficiency and flexibility. Especially for ECC implementation, RELIC is more often the first choice for the researchers and developers. In RELIC, algorithms work on blocks of 40 characters. Another important feature is to provide help in building architecture dependent code. RELIC provides a complete set of algorithms for multi-precision integer arithmetic, ECC, Bilinear maps (Tate pairing and optimal pairing), extension fields, RSA, BLS short signatures, ID-based authenticated key agreement, Rabin, ECMQV and ECSS.

Pigatto *et al.* [70] compared two algorithms ECC and El-Gamal for MIRACL and RELIC libraries and concluded that RELIC outperforms MIRACL for parameters like response time and key-size. The comparison between the average response times achieved by ECC based algorithms is performed using message size of 50 kB and considering two different key sizes 160-bit and 256-bits. The algorithm is based on MIRACL library has a considerably higher time than the one based on RELIC, in both cases. The running time obtained is approximately 9 seconds in case of MIRACL and 3.3 seconds in case of RELIC, in which the key size is 160-bit. When using 256-bit key size, the running time is 20.9 seconds in case of MIRACL and 10.6 seconds in case of RELIC.

3.1.5 Introduction to Simulator - AVRORA

AVRORA [21] is a cycle-accurate simulator designed for sensor networks and is written in Java. A Compilers group in UCLA has been working on this project. It is intended to simulate and analyze the programs written for AVR microcontrollers, MICAz and MICA2 motes. With the help of AVRORA, simulation can be performed on sensor networks of thousands nodes. There is no GUI support for AVRORA but it does a good job in consuming less execution time. It provides a framework for program analysis. The memory consumption of the program code can be computed by compiling the program using command "make micaz". After the compilation, AVRORA is used to calculate the number of cycles. The command "convert-avrora main.exe main.od" is used to convert the .exe to .od file. The command "avrora -platform = micaz -monitors = sleep main.od" will count the number of cycles in active and sleep state. By using this count, running time and energy consumption can be calculated easily. The formula for energy consumption is as follows [71]:

Total Energy Consumption = Voltage Level × Running Time

3.2 Parameter Selection

Choosing or designing an efficient protocol for implementing public key cryptographic protocol in WSN is an important task. Many factors influence the selection or design of such protocol. Few of them may be choosing the elliptic curve, choosing an efficient pairing method and choice of field. The modifications required, choosing the proper field, the order of the field, library for the algorithms, key sizes etc. The parameter selection may vary depending on the hardware characteristics, power constraints, computation time requirements and the memory space constraints. Depending on the characteristic of the WSN these decisions have to be made.

1. *Choice of Field*:

   In ECC, we are primarily concerned with the field arithmetic and the elliptic curve arithmetic defined over a particular field. An elliptic curve can be defined over a binary field ($\mathbb{F}_{2^m}$) or a prime field ($\mathbb{F}_p$), where the values of m and p are chosen according to the FIPS (Federal Information Processing Standards) [72] recommended fields to ensure the security of the protocols and the curves are chosen so that computation of reduction algorithms is fast.

   - *Prime field*: The simplified Weierstrassβ equation for prime field is given by: $y^2 = x^3 + ax + b$, where $a = -3$. The point doubling in Jacobian coordinates can be computed with less multiplication. In the prime field, the use of Jacobian coordinates takes $4M$ (multiplication), $4S$ (squaring) for doubling and the uses of mixed Jacobian affine coordinates take $8M$, $3S$ for the addition of two points [73].
   - *Binary field*: The simplified Weierstrassβ equation for binary field is given by: $y^2+xy = x^3+ax^2+b$. The usage of Lopez-Daheb projective coordinates takes $4M$ for doubling and mixed coordinates (Lopez-Daheb and affine coordinates) take $8M$ for addition of two points [73].

   The execution time in ECC schemes is dominated by the point or scalar multiplication operation. The point multiplication is defined as to compute kP, where k is a randomly generated integer and P is a point on an elliptic curve defined over a field. It can be concluded that binary field operations are less expensive than prime fields for comparable bit size. However, it is not always straightforward. For both prime and binary field, there exist various methods and algorithms to reduce the

scalar multiplication time and required memory storage. The common examples are: double and add, sliding window method, wNAF method and Montgomery Ladder [48, 49].

In WSN, 8/16-bit processors are in common use. Wenger and Hutter [74] compared the performance of two distinct ECC-hardware implementations that are based on prime field (NIST P-192) and binary field (ANSI X9.62 c2tnb191v1) arithmetic and concluded that the $\mathbb{F}_{2^m}$ based processors outperform the $\mathbb{F}_p$ based processors in run time, area required to make the chip and power consumption.

2. *Selection of Curve*:

   NIST (National Institute of Standards and Technology), USA, has recommended a collection of elliptic curves for Federal government use and it contains choices of private key length and underlying fields.

   The parameter sets of the binary koblitz curve standardized by NIST are listed below. The curves are of the form $y^2 + xy = x^3 + ax^2 + 1$ over a binary field. For the curve $K - 163$ the following parameters are listed as:

   $p(t)$: the reduction polynomial (in explicit and hexadecimal form)

   a: the curve's a coefficient

   G_x, G_y: the x and y coordinates of the base point G

   n: the base point's order

   h: the curve's cofactor

   For $K - 163$:
   $p(t) = t^{163} + t^7 + t^6 + t^3 + 1 = 800000000000000000000000000000000000000C9$
   $a = 1$
   $G_x = 2fe13c0537bbc11acaa07d793de4e6d5e5c94eee8$
   $G_y = 289070fb05d38ff58321f2e800536d538ccdaa3d9$
   $n = 5846006549323611672814741753598448348329118574063$
   $h = 2$

   NIST has presented five "security levels" of 80, 112, 128, 192 and 256 bits, and a curve would match that level only if its key size is at least twice the security level. So the standard curve for each level ought to be the smallest curve, which is large enough for that level. This should yield Koblitz curves in fields of size 163, 233, 277, 409 and 571 bits, respectively.

3. *Selection of key-size*:

   In cryptography, an algorithm is said to be having k-bit security if key used in the algorithm is k-bits long and there is no computational attack possible on the

algorithm. The security of an algorithm can never exceed the key length. In most symmetric key algorithms, security level is equal to key length but there are few exceptions also. For example, Triple DES provides at most 112 bits of security but key size used here is 168 bits. The reason behind this is because there is a computational attack of complexity 2^{112} is known. There is no public key algorithm having the same level of security as key size; ECC provides the effective security of roughly half its key length. In WSN, 80-bit security is considered enough, which can be achieved using around 160-bit key in public key algorithms. For ECC implementation, curve $K-163$ over binary field has been chosen.

4. *Cryptographic Standards*:

   Cryptographic standards provide the facility for the use of proper cryptographic techniques and also it ensures the scope for interoperability among the various implementations. The standards are specified for the cryptographic schemes independent of the area of implementations. It provides the assurance to both the developers and the customers about the security. These standards have been specified by organizations that have thoroughly analyzed the parameters (to the date), the drawbacks and the loopholes (if present) in corresponding algorithms and protocols. They have also been verified by the researchers community. These standards are updated regularly. The standards specify several things like the parameters for the scheme and data formats, key size, steps for the schemes and the message exchanges. For providing security in WSN the PKC can follow either of the recommended standards. Some of the standards for ECC are: ANSI X9.62, ANSI X9.63, FIPS 186.2, SEC 1 and SEC 2 are some of the most popular standards followed in the literature. Other standards are IEEE 1363-2000, IEEE P1363a, ISO/IEC 15946-1, ISO/IEC 15946- 2 etc. [72].

3.3 Parameter Selection for the Implementation Setup

On the basis of above discussion, we have chosen parameters for the implementation of the proposed schemes for WSN has been shown in Table 3.2.

3.4 Summary

For efficient implementation of any cryptographic algorithm, parameter selection is an important concern. The performance depends on the various factors like choice of field,

TABLE 3.2: Parameter Selection for the Implementation Setup

Parameter	**Used**	**Remark**
Operating System	TinyOS	Specifically designed for WSN
Sensor Mote	MICAz	Support for AVR microcontroller and AVRORA simulator
Cryptographic Library	RELIC-toolkit	Most efficient and flexible cryptographic library
Simulator	AVRORA	Cycle Accurate simulator
Field	Binary	$\mathbb{F}_{163}$ in case of ECC & $\mathbb{F}_{271}$ in case of PBC, providing 80 bit security
Pairing	η_T	Less computations in comparison to Weil Pairing

selection of curve and key size. The next two chapters define the three major contributions of the presented work, simulated on the discussed setup.

Chapter 4 presents efficient pairing-free identity-based two-party authenticated key agreement protocol and main motivation behind this protocol is of pairing free nature and computation efficient. A detailed discussion on CertificateLess Two-Party Authenticated Key Agreement (CTAKA) protocol, cryptanalysis of CTAKA protocol and an improved CTAKA protocol has been proposed in chapter 5. This chapter also proposes a new CTAKA protocol for WSN.

Chapter 4

Identity-based Authenticated Key Agreement

Two-party authenticated key agreement protocol is used to authenticate entities, who wish to establish session key in an adversary-controlled network in order to provide secure communications between two parties. To ensure secure communication between two entities, authenticated key agreement (AKA) protocol is the primary step to focus. The security of an identity-based system relies on a trusted private key generator (PKG) that generates private keys for users. The notable contribution is the proposal of a pairing-free identity-based two-party authenticated key agreement protocol for WSN. The proposed protocol is efficient as it does not use any pairing operation as well as reduces number of scalar point multiplications and proven to be secure in the security model presented by Liang et al. [14] based on extended Canetti-Krawczyk (eCK) model. [1]

4.1 Related Work

Recently, various Identity-based 2-Party Authenticated Key Agreement (ID-2PAKA) protocols have been presented based on bilinear pairing and without bilinear pairing. Smart [75] presented the first ID-2PAKA protocol based on Weil pairing. The protocol is using Boneh and Franklins [55] Identity-based Encryption (IBE) protocol. Though, Chen and Kudla [76] and Shim [77] found that the protocol is not resistant to perfect forward secrecy attack. Chen and Kudla [76] presented the first formal security analysis

[1] The major findings of this chapter has been communicated for publication as "*PF-ID-2PAKA: Pairing Free Identity-based Two-Party Authenticated Key Agreement Protocol for Wireless Sensor Network*," **Algorithmica**, Springer. (SCI-Indexed, Impact Factor-0.567)

(known as CK model) of ID-2PAKA protocol in the random oracle model. Chen and Kudla [76] presented five identity-based key agreement protocols.

Chen and Kudla's first identity-based key agreement protocol is more efficient in terms of computation cost (reducing number of pairing operations to one and uses only two elliptic curve scalar multiplication) than Smart's [75] ID-2PAKA protocol. Further, Chen and Kudla [76] modified Smart's and their protocol to include the single and different Trusted Authorities (TA) to provide forward secrecy, which avoids TA does not access user communication.

Shim [77] also presented an efficient ID-2PAKA protocol by reducing the number of evaluation of Weil pairings. Xun [78] also improved the protocol presented by Smart [75] by reducing pairing operation. Shim's [77] protocol uses single evaluation of Weil pairing and elliptic curve scalar multiplication to calculate the shared secret and heuristically claimed that the protocol is secure against conventional attacks. Conversely, Sun and Hsieh [79] proved that Shims [77] protocol is insecure against man-in-the-middle attack. Likewise, Boyd and Choo [80] proved that the protocol presented by Ryu et al. [81] is vulnerable to key-compromise impersonation (K-CI) attack, whereas Wang et al. [82] proved it insecure against reflection attack (RA) and presented an improved protocol resistant to several attacks. While McCullagh and Barretos [83] ID-2PAKA protocol proved insecure by Xie [84] against K-CI and presented an improved protocol, which was cryptanalyzed later by Li et al. [85] with K-CI attack. All the above protocols discussed are based on bilinear pairing. Recently, Zhu et al. [86] and Cao et al. [87] independently presented paring free ID-2PAKA protocol having three messages exchange.

Later, Cao et al. [88] presented a paring free ID-2PAKA protocol having two message exchange; consequently it reduces the number of messages and minimizes computation costs. The security of the protocol is based on modified Bellare-Rogaway (mBR) model [60]. But, Hafizul and Biswas [89] cryptanalyzed the protocol [88] against Known Session-specific Temporary Information Attack (KSTIA) and Key Offset Attack (KOA), and presented some modification in the protocol. The security of the protocol is based on the Blake-Johnson-Menezes (BJM) model [59].

4.2 Proposed Pairing Free Identity-based Two-Party Authenticated Key Agreement (PF-ID-2PAKA) Protocol

In this section we describe our Pairing-Free Identity-based Two-Party Authenticated Key Agreement (PF-ID-2PAKA) protocol for WSN. The proposed protocol is composed of three phases, i.e. *Setup*, *Extract*, and *Key Agreement*. For WSN scenario, the role of

Private Key Generator (PKG) is played by the Base-Station (BS), which is a trusted entity, responsible for the generation of keys for all sensor motes. The base-station runs the *Setup* and *Extract* phase of PF-ID-2PAKA protocol before deployment of the sensor motes in the network. The *Key Agreement* phase is performed by sensor motes post deployment, to establish a secure session for the communication between two sensor motes.

Setup. Suppose $\mathbb{G}$ be a cyclic additive group of prime order q and P is the generator of group $\mathbb{G}$. For a given security parameter k, PKG chooses two hash functions H_0 and H_1 such that $H_0 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_n^*$ and $H_1 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \{0,1\}^k$. The PKG chooses a random number $s \in \mathbb{Z}_n^*$ as a master key and computes $P_{pub} = s \cdot P$. Then, PKG publishes the system parameters as $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1\}$ and keep master key s secret.

Extract. For a given PKG's master key s, user's identity ID_i and system parameters $param$, PKG computes a private-key for user identity ID_i. PKG chooses a random number $r_i \in \mathbb{Z}_n^*$ and computes $R_i = r_i \cdot P$, $h_i = H_0(ID_i, R_i)$ and $s_i = r_i + h_i \cdot s \bmod n$. The user can validate its correction by checking whether $s_i \cdot P = R_i + h_i \cdot P_{pub}$. PKG issues a private public key is the pair $\{s_i, R_i\}$ to the user having identity ID_i through a secure channel.

Key Agreement. For user Alice with identity ID_A has private-key s_A and public-key R_A and the user Bob with identity ID_B has private-key s_B and public-key R_B. Alice and Bob will establish an authenticated session shown in figure 4.1:

1. Alice chooses a random number $t_A \in \mathbb{Z}_n^*$, set t_A as its ephemeral-secret-key and computes $T_A = t_A \cdot P$
2. Alice computes a message $M_1 = \{ID_A, R_A, T_A\}$ and send it to Bob
3. Bob chooses a random number $t_B \in \mathbb{Z}_n^*$, set t_B as its ephemeral-secret-key and computes $T_B = t_B \cdot P$
4. Bob computes a message $M_2 = \{ID_B, R_B, T_B\}$ and sends to Alice
5. Upon receiving M_1 and M_2 respectively both Alice and Bob compute their shared secrets as follows:
   (a) Alice computes a shared secret as $K_{AB}^1 = (s_A + t_A)(T_B + R_B + h_B \cdot P_{pub})$ and $K_{AB}^2 = t_A \cdot T_B$ and the session-key as $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{AB}^1 \| K_{AB}^2)$.
   (b) Bob computes a shared secret as $K_{BA}^1 = (s_B + t_B)(T_A + R_A + h_A \cdot P_{pub})$ and $K_{BA}^2 = t_B \cdot T_A$ and the session-key as $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{BA}^1 \| K_{BA}^2)$.
   (c) Alice and Bob both holds the same session-key.

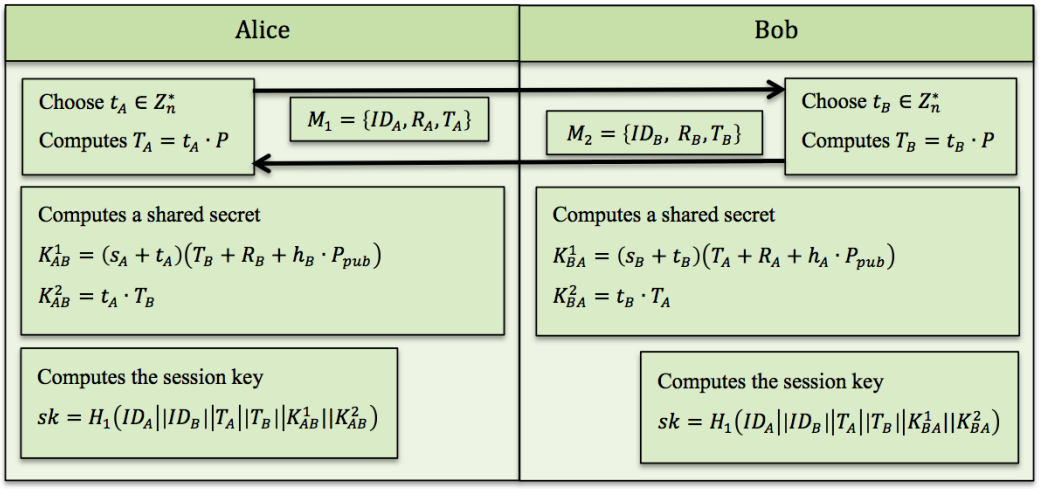

FIGURE 4.1: Key Agreement of PF-ID-2PAKA Protocol

4.3 Analysis of PF-ID-2PAKA Protocol

4.3.1 Correctness Analysis

This section demonstrates the correctness of the session-key generated during key agreement phase.

Lemma 4.1. *If two oracles are matching, both of them will be accepted and will get the same session-key, which is distributed uniformly at random in the session-key sample space.*

Proof. If two oracles are matching, then both of them are accepted and have the same session-key. The session-keys are distributed uniformly since t_A and t_B are selected uniformly during the protocol execution.

1. Correctness proof of $K_{AB}^1 = K_{BA}^1$

$$\begin{aligned} K_{AB}^1 &= (s_A + t_A)(R_B + H_0(ID_B \| R_B)P_{pub} + T_B) \\ &= (s_A + t_A)(s_B P + t_B P) \\ &= (s_B + t_B)(s_A P + t_A P) \\ &= (s_B + t_B)(R_A + H_0(ID_A \| R_A)P_{pub} + T_A) \\ &= K_{BA}^1 \end{aligned}$$

2. Correctness proof of $K_{AB}^2 = K_{BA}^2$

$$\begin{aligned} K_{AB}^2 &= t_A \cdot T_B \\ &= t_A \cdot t_B \cdot P \\ &= t_B \cdot T_A \\ &= K_{BA}^2 \end{aligned}$$

□

4.3.2 Security Analysis

This section demonstrates that the proposed PF-ID-2PAKA protocol is provably secure in the random oracle model by using eCK model. Hash functions H_0 and H_1 are considered as two random oracles. The following theorem proves that the *GDH* problem is intractable against forging attack by an adversary $\mathcal{A}$. $\mathcal{C}$ be a Challenger, who can respond the queries asked by an adversary to solve a particular problem defined in the protocol. The objective of the security game is to solve the hardness problem with a non-negligible advantage $Adv_{\mathcal{A}}(k)$ in polynomial time t, which is impractical.

Lemma 4.2. *Based on the assumption that* GDH *problem is intractable, the advantage of an adversary $\mathcal{A}$ against the proposed PF-ID-2PAKA protocol is negligible.*

Proof. The correctness of the PF-ID-2PAKA protocol (shown in Section 4.3.1) ensures that matching sessions compute the same session keys. According to Definition 2.8, we show that no efficient PPT adversary $\mathcal{A}$ against PF-ID-1PAKA protocol has a non-negligible advantage in winning the game outlined in Section 2.5.2.

Suppose adversary $\mathcal{A}$ can win the game defined in Section 2.5.2 with a non-negligible advantage $Adv_{\mathcal{A}}(k)$ in polynomial time t. Let, $\mathcal{C}$ be a Challenger. $\mathcal{C}$ randomly chooses $P_0 \in \mathbb{G}$ as the system public-key P_{pub}. $\mathcal{C}$ also chooses the system parameter $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1\}$, and sends it to $\mathcal{A}$. Let k be the security parameter. Let $n_0(k)$ be the maximum number of sessions that any one party may have. Suppose adversary $\mathcal{A}$ activates at most $n_1(k)$ distinct honest parties and $n_2(k)$ distinct hash queries. Assuming $Adv_{\mathcal{A}}(k)$ is non-negligible. Here, $n_0(k)$, $n_1(k)$ and $n_2(k)$ are polynomially bounded in k. Since, H_0 and H_1 are random oracle. After the adversary issues the *Test* query, there are three possible ways to distinguish the tested session-key from a random string. These are:

1. *Guessing attack.* The adversary $\mathcal{A}$ correctly guesses the session-key.

2. *Key-replication attack.* The adversary $\mathcal{A}$ obligates a non-matching session to have the same session-key as the test-session. The adversary $\mathcal{A}$ could learn the session-key by querying the non-matching session.

3. *Forging attack.* The adversary $\mathcal{A}$ computes the values K_{IJ}^1, K_{IJ}^2 itself on a test-session $\prod_{I,J}^T$. The adversary $\mathcal{A}$ queries H_1 on the value $(ID_I, ID_J, T_I, T_J, K_{IJ}^1, K_{IJ}^2, sk)$ in the test-session initiated by I and communicating with J.

The success probabilities of guessing attack and key-replication attack are negligible [63]. The probability of guessing the output of a random oracle H_1 is $(1/2^k)$. As, two non-matching sessions cannot have the same identities and the same ephemeral public-keys, therefore, guessing attack and key-replication attack can be suspended, and the focus is mainly on the analysis of forging attack.

For winning an advantage in the game against the protocol, the adversary $\mathcal{A}$ has to query the random oracle H_1 on the session-key. We use a classical reduction approach to relate the advantage for the adversary $\mathcal{A}$ against our proposed PF-ID-2PAKA protocol to the *GDH* assumption. The challenger $\mathcal{C}$ helps the adversary $\mathcal{A}$ to bring $\mathcal{A}$'s advantage indistinguishing the tested session key from a random string into an advantage in solving the *GDH* problem.

Let $Adv_{\mathcal{C}}^{GDH}(k)$ be the advantage that the challenger $\mathcal{C}$ acquires in solving the *GDH* problem with known security parameter k. To solve the *GDH* problem using $\mathcal{A}$, $\mathcal{C}$ is given a *GDH* challenge $U = u \cdot P$, $V = v \cdot P$ and an oracle $DDH(*, *, *)$, where $u, v \in \mathbb{Z}_n^*$, and $\mathcal{C}$'s task is to compute $GDH(U, V) = u \cdot v \cdot P$.

Challenger $\mathcal{C}$ is responsible for the simulation of the game described in Section 2.5.2. $\mathcal{C}$ responds all queries of the adversary $\mathcal{A}$ throughout the game.

Before commencement of the game, $\mathcal{C}$ randomly chooses two indexes $I, J \in \{1, \cdots, n_1\}$: $I \neq J$, which represents the I^{th} and the J^{th} distinct honest parties. Also, $\mathcal{C}$ chooses $S \in 1, \cdots, n_0$. Let $\prod_{J,I}^S$ be the matching session of $\prod_{I,J}^S$. The challenger $\mathcal{C}$ has to guess that $\prod_{J,I}^S$ is the test-session, which is correct with probability larger than $\frac{1}{n_0(k)n_1(k)^2}$. The challenger simulates the game according to its own guess. $\mathcal{C}$ aborts the game whenever it finds (especially by observing $\mathcal{A}$'s queries and other actions) that it has missed its guess. Otherwise, the game proceeds in a usual manner.

$\mathcal{C}$ maintains two lists to handle random oracle queries. These lists records entries (including input/output pairs) and simulates the H_0 and H_1 oracles by using the separate lists. If $\mathcal{A}$ issues a random oracle query that matches one of the previous inputs recorded in the corresponding list, $\mathcal{C}$ responds with the output of the matching entry. Else, $\mathcal{C}$

responds back to $\mathcal{A}$ with the new generated value and includes the new value in the corresponding list.

According to the fresh session definition, $\mathcal{C}$ has four options for $\mathcal{A}$'s strategy:

1. $\mathcal{A}$ may neither learn the ephemeral-private-key of ID_I i.e t_I nor the static private-key of ID_J i.e s_J.
2. $\mathcal{A}$ may neither learn the ephemeral-private-key of ID_J i.e t_J nor the static private-key of ID_I i.e. s_I.
3. $\mathcal{A}$ may neither learn the static private-key of ID_I i.e s_I nor of ID_J i.e s_J.
4. $\mathcal{A}$ may neither learn the ephemeral-private-key of ID_I i.e t_I nor of ID_J i.e t_J.

Based on selected strategy, $\mathcal{C}$ performs the simulation and responds the *Test* query as follows:

- If the test session chosen by $\mathcal{A}$ does not match with $\mathcal{C}$'s aforehand guess, $\mathcal{C}$ aborts.
- Else, $\mathcal{C}$ responds to the adversary $\mathcal{A}$ with a randomly selected value $sk \in \{0,1\}^k$.

As there are four strategies, the probability that $\mathcal{C}$ guessed both the strategy and the test session beforehand is larger than $\frac{1}{4n_0(k)n_1(k)^2}$. The session key sk is generated by querying H_1 on $(ID_I, ID_J, T_I, T_J, K_{IJ}^1, K_{IJ}^2)$. The four strategies are as follow:

Case 1. *$\mathcal{A}$ may learn neither the ephemeral-private-key of ID_I i.e. t_I nor the static private-key of ID_J i.e. s_J.*

$\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

(a) *Create(i).* $\mathcal{C}$ maintain an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i \rangle$.

- If $ID_i = ID_J$, $\mathcal{C}$ chooses random numbers $h_i \in \mathbb{Z}_n^*$, computes $R_i = U - h_i \cdot P_0$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID i, \perp, R_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, $\mathcal{C}$ chooses three random numbers $r_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = x_i \cdot P$, $s_i = r_i + h_i \cdot x \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
  Else, $\mathcal{C}$ chooses two random numbers $s_i, h_i \in \mathbb{Z}_n^*$, computes $R_i = s_i \cdot P - h_i \cdot P_{pub}$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

(b) $H_0(ID_i, R_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_0} comprising of tuple $\langle ID_i, R_i, h_i \rangle$.

- If (ID_i, R_i) is in the list L_{H_0}, $\mathcal{C}$ returns h_i.
- Else, $\mathcal{C}$ chooses a random number h_i, stores $\langle ID_i, R_i, h_i \rangle$ in L_{H_0} and returns h_i.

(c) $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ responds the queries as follows:
  - If $ID_i = ID_J$, $\mathcal{C}$ checks the list L_S for tuple $\langle ID_i, ID_j, T_i, T_j, * \rangle$.
    - If $\mathcal{C}$ finds the tuple, it computes $\bar{Z}_1 = Z_1 - t_i \cdot (T_j + R_j + H_0(ID_j, R_j)) - s_j \cdot (R_i + H_0(ID_i, R_i))$. $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*, *, *)$ outputs 1, when the tuple $\langle T_j, (R_i + H_0(ID_i, R_i) \cdot P_{pub}), \bar{Z}_1 \rangle$ is input. $\mathcal{C}$ checks the correctness of Z_2 by checking whether the equation $Z_2 = t_i \cdot T_j$ hold. If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S .
    - Else, $\mathcal{C}$ picks a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.
  - Else, $\mathcal{C}$ checks the list L_S for tuple $\langle ID_i, ID_j, T_i, T_j, * \rangle$.
    - If $\mathcal{C}$ finds the tuple, it stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S.
    - Else, $\mathcal{C}$ picks a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

(d) *Reveal-Static-Private-Key(ID_i).* $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

- If $ID_i = ID_J$ then $\mathcal{C}$ stops the simulation of the game.
- Else, $\mathcal{C}$ checks the list L_E and responds with the corresponding static private-key s_i to adversary $\mathcal{A}$.

(e) *Reveal-Ephemeral-Key*($\prod_{i,j}^t$). $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

- If $\prod_{i,j}^t = \prod_{I,J}^T$ then $\mathcal{C}$ stops the simulation of the game.
- Else, $\mathcal{C}$ responds with the corresponding stored ephemeral-private-key to $\mathcal{A}$.

(f) *Reveal-Session-Key*($\prod_{i,j}^t$). $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

- If $\prod_{i,j}^t = \prod_{I,J}^T$ or $\prod_{i,j}^t = \prod_{J,I}^L$, then $\mathcal{C}$ stops the simulation of the game.
- Else, $\mathcal{C}$ responds with the corresponding session-key sk to $\mathcal{A}$.

(g) *Send*($\prod_{i,j}^{t}, m$). $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk\rangle$ and answers $\mathcal{A}$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{S}$, then $\mathcal{C}$ returns $T_i = V$ to $\mathcal{A}$.
- Else, if $ID_i = ID_J$, it generates a random $t_i \in \mathbb{Z}_n^*$, and computes $\bar{Z}_1 = Z_1 - t_i \cdot (T_j + R_j + H_0(ID_j, R_j)) - s_j \cdot (R_i + H_0(ID_i, R_i))$. $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*,*,*)$ outputs 1, when the tuple $\langle T_j, (R_i + H_0(ID_i, R_i) \cdot P_{pub}), \bar{Z}_1\rangle$ is input. $\mathcal{C}$ also checks the correctness of Z_2 by checking whether the equation $Z_2 = t_i \cdot T_j$ hold.
  - If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, sk\rangle$ in L_S, where the value sk comes from L_{H_1}.
  - $\mathcal{C}$ picks a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, sk\rangle$ in L_S.
- Else, $\mathcal{C}$ responds as per the specifications of the key agreement protocol.

(h) *Test*($\prod_{i,j}^{t}$). $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

- If $\prod_{i,j}^{t} \neq \prod_{I,J}^{S}$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ responds to $\mathcal{A}$ with the randomly generated number $sk \in \{0,1\}^k$.

As the adversary $\mathcal{A}$ mounts the forging attack, if $\mathcal{A}$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I) \cdot (T_J + R_J + H_0(ID_J, R_J) \cdot P_{pub}) = (t_I + s_I) \cdot (T_J + U)$ and $Z_2 = t_I \cdot T_J$, where $T_I = V$ is the outgoing message of the test-session given by the simulator and T_J is the incoming message from the adversary $\mathcal{A}$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly picks one entry with the probability $\frac{1}{n_2}$ and proceeds as follows:

$\mathcal{C}$ computes $\bar{Z}_1 = Z_1 - s_I \cdot (T_J + U)$. It is easy to verify the equation $\bar{Z}_1 = GDH(T_I, T_J) + GDH(U, V)$ holds. Then, $\mathcal{C}$ computes $GDH(U, V) = \bar{Z}_1 - GDH(T_I, T_J) = \bar{Z}_1 - Z_2$.

The advantage of $\mathcal{C}$ for solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}}(k)$.

Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 2. *$\mathcal{A}$ may neither learn the ephemeral-private-key of ID_J i.e t_J nor the static private-key of ID_I i.e. s_I.*

By exchanging the roles of I and J in the above case, we can prove that $Adv_{\mathcal{C}}^{GDH}(k)$ is negligible by the same method as in the above case.

Case 3. *$\mathcal{A}$ may neither learn the static private-key of ID_I i.e s_I nor of ID_J i.e s_J.*

$\mathcal{C}$ answers $H_0(ID_i, R_i)$, *Reveal-Ephemeral-Key*($\prod_{i,j}^{t}$), *Reveal-Session-Key*($\prod_{i,j}^{t}$) and *Test*($\prod_{i,j}^{s}$) as it does in *Case1*. It responds other queries in the following way:

(a) *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i \rangle$.

- If $ID_i = ID_I$, $\mathcal{C}$ picks a random number $h_i \in \mathbb{Z}_n^*$, computes $R_i = U - h_i \cdot P_{pub}$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle IDi, \perp, R_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, if $ID_i = ID_J$, $\mathcal{C}$ picks a random number $h_i \in \mathbb{Z}_n^*$, computes $R_i = V - h_i \cdot P_{pub}$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, \perp, R_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, $\mathcal{C}$ picks two random numbers $s_i, h_i \in \mathbb{Z}_n^*$, computes $R_i = s_i \cdot P - h_i \cdot P_0$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

(b) $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ responds the queries as follows:
  - If $ID_i = ID_I$ or $ID_i = ID_J$, $\mathcal{C}$ simulates the oracle in the same way as does in *Case1.*
  - Else, $\mathcal{C}$ checks the list L_S for tuple $\langle ID_i, ID_j, T_i, T_j, * \rangle$.
    - If $\mathcal{C}$ finds the tuple, it stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S .
    - Else, $\mathcal{C}$ picks a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

(c) *Reveal-Static-Private-Key(ID_i).* $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:

- If $ID_i = ID_I$ or $ID_i = ID_J$, then $\mathcal{C}$ stops the simulation of the game.
- Else, $\mathcal{C}$ checks the list L_C and responds with the corresponding static private-key s_i to the adversary $\mathcal{A}$.

(d) *Send*$(\prod_{i,j}^t, m)$. $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and responds to $\mathcal{A}$'s queries as follows:

- If $ID_i = ID_I$ or $ID_i = ID_J$, $\mathcal{C}$ simulates the oracle in the same way as does in *Case1.*
- Else, $\mathcal{C}$ responds as per the specification of the key agreement protocol.

As, the adversary $\mathcal{A}$ mounts the forging attack, if $\mathcal{A}$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I) \cdot (T_J + R_J + H_0(ID_J, R_J) \cdot P_{pub}) = (t_I + s_I) \cdot (T_J + V)$ and $Z_2 = t_I \cdot T_J$, where $R_I + H_0(ID_I, R_I) \cdot P_{pub} = U$ and T_J are the incoming messages from the adversary $\mathcal{A}_I$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly picks one entry with the probability $\frac{1}{n_2}$ and proceeds as follows:

$\mathcal{C}$ computes $GDH(U,V) = Z_1 - t_I \cdot (T_J + V) - t_J \cdot U$.

The advantage of $\mathcal{C}$ for solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}}(k)$.

Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible, since we assume that $Adv_{\mathcal{A}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 4. *$\mathcal{A}$ may neither learn the ephemeral-private-key of ID_I i.e t_I nor of ID_J i.e t_J.*

$\mathcal{C}$ answers $H_0(ID_i, R_i)$, *Reveal-Session-Key*$(\prod_{i,j}^{t})$ and *Test*$(\prod_{i,j}^{s})$ as it does in *Case1*. It responds to other queries as follows:

(a) *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i \rangle$. $\mathcal{C}$ chooses two random numbers $s_i, h_i \in \mathbb{Z}_n^*$, computes $R_i = s_i \cdot P - h_i \cdot P_{pub}$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

(b) $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ responds to these queries as follows:
  - $\mathcal{C}$ checks the list L_S for tuple $\langle ID_i, ID_j, T_i, T_j, * \rangle$. If $\mathcal{C}$ finds the tuple, it computes $\bar{Z}_1 = Z_1 - s_i \cdot (T_j + R_j + H_0(ID_j, R_j)) - s_j \cdot T_i$. Then, $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*,*,*)$ outputs 1, when the tuple $\langle T_i, T_j, \bar{Z}_1 \rangle$ is input.
    - If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S.
    - Else, $\mathcal{C}$ picks a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.
  - Else, $\mathcal{C}$ picks a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

(c) *Reveal-Static-Private-Key(ID_i).* $\mathcal{C}$ checks the list L_C and responds with the corresponding static private-key s_i to the adversary $\mathcal{A}$.

(d) *Reveal-Ephemeral-Key*$(\prod_{i,j}^{t})$. $\mathcal{C}$ responds to $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{T}$ or $\prod_{i,j}^{t} = \prod_{J,I}^{L}$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ responds with the corresponding stored ephemeral-private-key to $\mathcal{A}$.

(e) *Send*$(\prod_{i,j}^{t}, m)$. $\mathcal{C}$ maintains an initially empty list L_S with entries of the form $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and responds to $\mathcal{A}$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{T}$, $\mathcal{C}$ returns $T_i = U$ to $\mathcal{A}$.

- Else, if $\prod_{i,j}^{t} = \prod_{I,J}^{L}$, $\mathcal{C}$ returns $T_i = V$ to $\mathcal{A}$.
- Else, $\mathcal{C}$ replies as per the specifications of the key agreement protocol.

As, the adversary $\mathcal{A}$ mounts the forging attack, if $\mathcal{A}$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I) \cdot (T_J + R_J + H_0(ID_J, R_J) \cdot P_{pub})$ and $Z_2 = t_I \cdot T_J$, where $T_I = U$ is the outgoing message of the test-session given by the simulator and $T_J = V$ is the incoming message from the adversary $\mathcal{A}$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly picks one entry with the probability $\frac{1}{n_2}$ and returns Z_2 as the solution to $GDH(U, V)$.

The advantage of $\mathcal{C}$ for solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}}(k)$.

Then, $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

□

4.3.3 Comparative Analysis

The comparison has been carried out with two recent pairing free ID-2PAKA protocols (see Table 4.1): Cao et al. [88] (denoted as CK) and Hafizul and Biswas [89] (denoted as HB).

Table 4.1: Comparative Analysis of ID-2PAKA Protocols

protocol	**Communication Overhead**	**Computation Overhead**	**Security Model**	**Hardness**	**Cryptanalyze**
CK [88]	2	$5T_{mul} + 2T_{add} + 2T_h$	mBR	CDH	HB [89]
HB [89]	2	$4T_{mul} + 2T_{add} + 3T_h$	BJM	CDH	–
PF-ID-2PAKA	2	$3T_{mul} + 3T_{add} + 2T_h$	eCK	GDH	–

The factors used to evaluate the performance of the proposed PF-ID-2PAKA protocol are computation overhead, communication overhead, security model and hardness problem. The computation overhead includes the number of complex cryptographic operations such as scalar point multiplication, point addition, modular inversion and one way hash function while the communication overhead includes total number of messages exchanged in each protocol run. T_{mul} is the execution time for a scalar point multiplication, T_{add} is the execution time for a point addition operation, T_{inv} is the execution time for a modular inversion operation and T_h is the execution time for a one-way hash function. Table 4.1 shows that the proposed PF-ID-2PAKA Protocol is having three point multiplications, whereas the other two protocols are having more point multiplication

operations. It is extremely enviable for a security protocol to have low computational overhead on resource constrained WSN. The computation overhead on a sensor node for PF-ID-2PAKA protocol is three point multiplications and three point additions to compute session-key, which is fairly very low as compared to other protocols.

4.3.4 Implementation and Performance Analysis

The evaluation of the proposed PF-ID-2PAKA protocol for WSN has been carried out for a couple of parameters such as running time and energy consumption. The implementation results of PF-ID-2PAKA protocol for WSN has been shown in Table 4.2.

TABLE 4.2: Performance Analysis of PF-ID-2PAKA Protocol with Other Existing Protocols

protocol	**Running Time (s)**	**Energy Consumption (mJ)**
CK [88]	1.96	47.04
HB [89]	1.568	37.362
PF-ID-2PAKA	1.176	28.224

We assume that the system parameter *param*, generated by the base station are stored at each mote before deployment in the sensor field. The base station acts as a Private Key Generator (PKG) and is responsible for the generation of private keys of the sensor mote. So, the computations involved in the generation of public-private-key-pair are not included for the calculations of the computation overhead. We implemented the PF-ID-2PAKA protocol for MICAz mote [17] by Crossbow Technology. The MICAz mote is embedded with 8-bit ATmega128L processor, 4KB of SRAM, 128KB of flash memory (ROM) with a clock speed of 7.3828MHz, RF transceiver complies with IEEE 802.15.4/ZigBee. We use TinyOS-2.1.2 [18] operating system for MICAz. nesC programming language is used for the implementation of the protocol. We, use RELIC-toolkit [20], a cryptographic library especially designed for avr and msp microcontrollers. It is an efficient and flexible cryptographic library (version 0.3.3), which has been used to perform operations on elliptic curves. We use curve K-163 for ECC, to meet the 80-bit security level. The running time and energy consumption is computed by using AVRORA simulator. The storage cost have been taken in terms of usage of RAM and ROM. PF-ID-2PAKA protocol utilizes only 2.1 kB of RAM and 885 bytes of ROM excluding cryptographic library, which is easily justifiable to any sensor based application.

4.4 Summary

This chapter discussed the proposed pairing-free identity-based two-party authenticated key agreement (PF-ID-2PAKA) protocol for WSN. The proposed protocol had been proven secure in the eCK model presented by Liang et al. [14] The protocol has been compared with existing protocols and found to be more efficient with lesser computation cost. PF-ID-2PAKA had also been compared with existing protocols on the basis of running time and energy consumption. The PF-ID-2PAKA protocol was implemented for WSN on MICAz platform using TinyOS-2.1.2 and RELIC-0.3.3 cryptographic library. Results prove that PF-ID-2PAKA protocol is having less running time as well as consumes less energy as compared to existing protocols.

Chapter 5

Certificateless Authenticated Key Agreement

Key exchange protocol(s) is one of the important factors for establishing communication between two parties. Recent works support the use of pairing free certificateless authenticated key exchange protocol(s) and becomes a promising base in a energy-famished WSN. Certificateless concept provides an authentication by eliminating the need of certificates of traditional Public Key Cryptography (PKC) and key escrow problem of Identity Based Cryptography (IBC). The notable contribution is the proposal of a Non-Interactive Certificateless Two-party Authenticated Key Agreement (NI-CTAKA) protocol for WSN. The proposed protocol is pairing-free as well as reduces number of scalar point multiplications and proven secure in the security model presented by Lipold et al. [16], which is based on LaMacchia et al.'s [63] extended Canetti-Krawczyk (eCK) model. Further, a recent protocol presented by Kim et al. [15] has been cryptanalyzed against public-key replacement attack, presented a defensive measure, which is proven secure in the eCK security model by Lippold et al. [1] [2]

The chapter is organized as follows: Section 5.1 discusses about the related work of certificateless two-party authenticated key agreement protocols. Section 5.2 discusses about the review of Kim *et al.*'s [15] scheme. Section 5.3 presents the cryptanalysis of Kim *et al.*'s scheme followed by its improvement in Section 5.4 and security analysis in Section 5.5. Section 5.6 discusses about our second CTAKA protocol called NI-CTAKA

[1]The major findings of this chapter has been accepted for publication as "*A Non-Interactive Certificateless Two-Party Authenticated Key Agreement Protocol for Wireless Sensor Networks*," **International Journal of Ad Hoc and Ubiquitous Computing**. (SCI-Indexed, Impact Factor-0.511)

[2]The major findings of this chapter has been communicated as "*Impersonation Attack on Certificate-Less Key Agreement Protocol*," **International Journal of Ad Hoc and Ubiquitous Computing**. (SCI-Indexed, Impact Factor-0.511)

followed by correctness, security, comparative, implementation and performance analysis in Section 5.7.

5.1 Related Work

The evolution of CertificateLess Public Key Cryptography (CL-PKC) is to unravel the inadequacies of identity based cryptography by captivating the assets of traditional PKI and IBC. Al-Riyami and Paterson [10] diverts the attention towards a new direction by proposing the brilliant concept of CL-PKC in 2003, which outperforms dual problems of traditional approaches in the field of asymmetric cryptography, the public-key certificates in PKI and familial key escrow problem in IBC. The generation of public-private-keys is done in two phases: partial-private-key generation by Key Generation Center (KGC) and private-public-key-setup by user. In CL-PKC, a trusted third party known as KGC is involved to issue partial-private-key to the user based on the identity provided by the user. The partial-private-key is computed by using user's identity and KGC's private-key. The user independently generates its own secret-value; and computes its private-key and public-key by using its secret-value and partial-private-key. It prevents the impersonation attack unlike IBC, by not knowing one of the keys are generated by the user or KGC. Al-Riyami and Paterson [10] proposed signature scheme by using bilinear maps with security proofs. Though the scheme is costly, as the verification process requires four pairing computations. However, Al-Riyami and Paterson [10] scheme has been proved insecure against Type I adversary by Huang et al. [90].

CTAKA protocol based on bilinear pairing was first introduced by Al-Riyami and Paterson [10], without any security proofs. Thereafter, lots of CTAKA schemes based on bilinear pairings have been proposed [91–94]. Since, bilinear pairing operation is computationally very expensive, researchers focused to design pairing free protocols. In 2009, Geng and Zhang [95] proved that Shi and Li's scheme [94] based on bilinear pairing is insecure and proposed a pairing free scheme based on Gap-Diffie-Hellman (GDH) problem, having five exponentiation operations. The scheme is not resistant to man-in-the-middle attack. In the same year Hou and Xu [96] proposed another pairing free scheme based on Computational-Diffie-Hellman (CDH) assumption. The authors claimed that the scheme provides perfect-forward secrecy, PKG-forward secrecy, known-key secrecy, key-compromise impersonation resilience, unknown key-share resilience, known session-specific temporary information security, message independence and no-key control.

According to Yang and Tan [97], both Geng and Zhang's scheme [95] and Hou et al's scheme [96] are not secure. Further, they proposed an improved provable secure pairing-free CTAKA protocol. Also, He et al. [98] proposed a CTAKA protocol without pairing,

which was proved insecure in [99] against both Type I and Type II adversary. Later, Xiong et al. [100] stated a formal security model for CTAKA, encompassing standard definitions of authenticated key agreement protocols and certificateless cryptography and also proved the security of the scheme using modified-Bellare-Rogaway (mBR) model [58]. He et al. [101] proposed a scheme secure under random oracle model, having five scalar multiplication operations. He et al. [102] proposed a CTAKA protocol based on eCK model [62]. Cheng [103] proved that the scheme [101] was insecure against Type I adversary. Later on, lots of pairing-free CTAKA protocols were proposed [15, 104], but the emphasis was the reduction of scalar point multiplication operation.

5.2 Review of Kim et al.'s CTAKA Protocol

This section reviews the Kim et al.'s [15] CTAKA protocol. The protocol works as follows:

Setup: Let $\mathbb{G}$ be a cyclic additive group of prime order q and P is the generator of group $\mathbb{G}$. For a given security parameter k, KGC chooses two hash functions H_0 and H_1 such that $H_0 : \{0,1\}^k \rightarrow \mathbb{Z}_n^*$ and $H_1 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G}^4 \rightarrow \mathbb{Z}_n^*$. The KGC picks a random number $s \in \mathbb{Z}_n^*$ as a master secret key and computes $P_{pub} = s \cdot P$. Then, KGC publishes the system parameters as $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1\}$ and keep master key s secret.

Partial-Private-Key-Extract: KGC chooses a random number $r_i \in \mathbb{Z}_n^*$ and computes $R_i = r_i \cdot P$, $h_i = H_0(ID_i, R_i)$ and $s_i = r_i + h_i \cdot s \bmod n$. KGC issues a partial-private-key $D_i = (s_i, R_i)$ to the user having identity ID_i through a secure channel. The user can validate its correction by checking whether $s_i \cdot P = R_i + h_i \cdot P_{pub}$.

Set-Secret-Value: The user with identity ID_i chooses a random number $x_i \in \mathbb{Z}_n^*$ and set x_i as its secret-value.

Set-Private-Key: The user with identity ID_i issues its private-key pair as $sk_i = (x_i, s_i)$.

Set-Public-Key: The user computes its public value as $P_i = x_i \cdot P$ and issues its public-key.

Key-Agreement: Assuming two users *Alice* and *Bob*. *Alice* is having identity ID_A has private keys $sk_A = \{s_A, x_A\}$ and public key P_A. While, *Bob* is having identity ID_B has private keys $sk_B = \{s_B, x_B\}$ and public key P_B. *Alice* and *Bob* will establish an authenticated session as shown in figure 5.1, the following steps will be executed as follows:

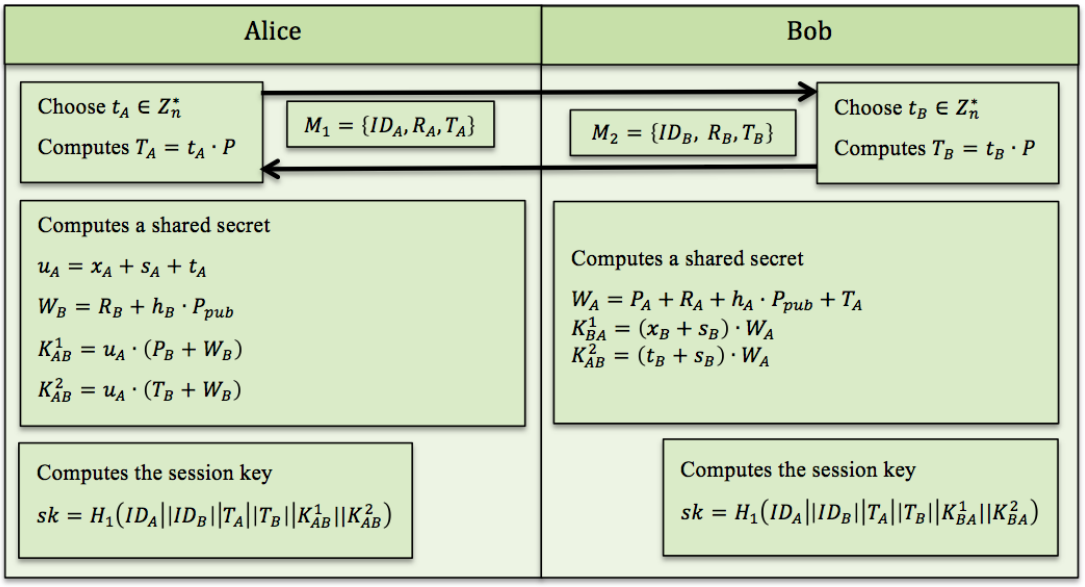

FIGURE 5.1: Key Agreement of Kim et al.'s CTAKA Protocol

1. *Alice* chooses a random number $t_A \in \mathbb{Z}_n^*$ and computes $T_A = t_A \cdot P$
2. *Alice* computes a message $M_1 = \{ID_A, R_A, T_A\}$ and send it to *Bob*
3. *Bob* chooses a random number $t_B \in \mathbb{Z}_n^*$ and computes $T_B = t_B \cdot P$
4. *Bob* computes a message $M_2 = \{ID_B, R_B, T_B\}$ and sends to *Alice*
5. Upon receiving M_1 and M_2 respectively both *Alice* and *Bob* compute their shared secrets as follows:

   (a) *Alice* computes the following:

   - $u_A = x_A + s_A + t_A$
   - $W_B = R_B + h_B \cdot P_{pub}$
   - $K_{AB}^1 = u_A \cdot (P_B + W_B)$
   - $K_{AB}^2 = u_A \cdot (T_B + W_B)$
   - $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{AB}^1 \| K_{AB}^2)$

   (b) *Bob* computes the following:

   - $W_A = P_A + R_A + h_A \cdot P_{pub} + T_A$
   - $K_{BA}^1 = (x_B + s_B) \cdot W_A$
   - $K_{BA}^2 = (t_B + s_B) \cdot W_A$
   - $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{BA}^1 \| K_{BA}^2)$

   (c) *Alice* and *Bob* both holds the same session key.

5.3 Key-Compromise Impersonation (K-CI) Attack on Kim et al.'s CTAKA Protocol

This section describes that the Kim et al.'s protocol is insecure in the eCK model against Key Compromise Impersonation (KCI) attack, the vulnerability in Kim et al.'s protocol is exposed, as it fails to provide implicit authentication.

5.3.1 K-CI Attack

The type I adversary $\mathcal{A}_I$ chooses a random number $t_{\mathcal{A}_I} \in \mathbb{Z}_n^*$ and computes $R_B = t_{\mathcal{A}_I} \cdot P$. The adversary $\mathcal{A}_I$ replaces party *Bob*'s public-key as $P_B = -h_B \cdot P_{pub}$. Type I adversary $\mathcal{A}_I$ can mount the attack on the protocol as follows:

1. *Alice* chooses $t_A \in \mathbb{Z}_n^*$ randomly and computes $T_A = t_A \cdot P$. Then, *Alice* sends the message $M_1 = \{ID_A, R_A, T_A\}$ to party *Bob*.

2. Upon intercepting the message M_1, the adversary $\mathcal{A}_I$ computes $T_B = -h_B \cdot P_{pub}$, where $R_B = t_{\mathcal{A}_I} \cdot P$ and $h_B = H_1(ID_B, R_B)$. Then, the adversary $\mathcal{A}_I$ impersonates *Bob* and sends $M_2 = \{ID_B, R_B, T_B\}$ to *Alice*. Finally, the adversary $\mathcal{A}_I$ computes the shared session key as follows:

   $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K^1_{\mathcal{A}_I A} \| K^2_{\mathcal{A}_I A})$

   where,

   $K^1_{\mathcal{A}_I A} = t_{\mathcal{A}_I}(P_A + R_A + h_A \cdot P_{pub} + T_A)$

   $K^2_{\mathcal{A}_I A} = t_{\mathcal{A}_I}(P_A + R_A + h_A \cdot P_{pub} + T_A)$

3. *Alice* computes the same session key by following usual algorithm steps as follows:

   - $u_A = x_A + s_A + t_A$
   - $W_B = R_B + h_B \cdot P_{pub}$
   - $K^1_{AB} = u_A \cdot (P_B + W_B)$
   - $K^2_{AB} = u_A \cdot (T_B + W_B)$
   - $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K^1_{AB} \| K^2_{AB})$

5.3.2 Correctness Analysis of K-CI Attack

The correctness of the session keys generated by *Alice* and the adversary $\mathcal{A}_I$ (impersonates *Bob*) is as follows:

1. Correctness proof of $K^1_{AB} = K_(\mathcal{A}_I A)^1$

$$\begin{aligned} K^1_{AB} &= u_A \cdot (P_B + W_B) \\ &= (x_A + s_A + t_A)(-h_B \cdot P_{pub} + t_{\mathcal{A}_I} \cdot P + h_B \cdot P_{pub}) \\ &= (x_A + s_A + t_A) \cdot t_{\mathcal{A}_I} \cdot P \\ &= (x_A \cdot P + s_A \cdot P + t_A \cdot P) \cdot t_{\mathcal{A}_I} \\ &= (P_A + R_A + h_A \cdot P_{pub} + T_A) \cdot t_{\mathcal{A}_I} \\ &= K^1_{\mathcal{A}_I A} \end{aligned}$$

2. Correctness proof of $K^2_{AB} = K_(\mathcal{A}_I A)^2$

$$\begin{aligned} K^2_{AB} &= u_A \cdot (T_B + W_B) \\ &= (x_A + s_A + t_A)(-h_B \cdot P_{pub} + t_{\mathcal{A}_I} \cdot P + h_B \cdot P_{pub}) \\ &= (x_A + s_A + t_A) \cdot t_{\mathcal{A}_I} \cdot P \\ &= (x_A \cdot P + s_A \cdot P + t_A \cdot P) \cdot t_{\mathcal{A}_I} \\ &= (P_A + R_A + h_A \cdot P_{pub} + T_A) \cdot t_{\mathcal{A}_I} \\ &= K^2_{\mathcal{A}_I A} \end{aligned}$$

5.4 Improvement of Kim et al.'s CTAKA Protocol

According to the proposed countermeasure, a new hash function needs to be introduced. Let $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_n^*$ be a hash function based on random oracle. In the Key-Agreement algorithm, on multiplying P_B with l, where $l = H_2(ID_B, R_B, P_B, T_B)$, while *Alice* computes a session-key and x_B with l, while *Bob* computes a session-key, thwarts the impersonation attack. The *Partial-Private-Key-Extract*, *Set-Secret-Value*, *Set-Private-Key* and *Set-Public-Key* algorithms are same as discussed in Section 5.2. The rest of the modified algorithms are as follows:

Setup: Let $\mathbb{G}$ be a cyclic additive group of prime order q and P is the generator of group $\mathbb{G}$. For a given security parameter k, KGC chooses two hash functions H_0 and H_1 such that $H_0 : \{0,1\}^k \rightarrow \mathbb{Z}_n^*$, $H_1 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_n^*$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_n^*$. The KGC picks a random number $s \in \mathbb{Z}_n^*$ as a master secret key and computes $P_{pub} = s \cdot P$. Then, KGC publishes the system parameters as $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1, H_2\}$ and keep master key s secret.

Key-Agreement: Assuming two users *Alice* and *Bob*. *Alice* is having identity ID_A has private keys $S_A = \{s_A, x_A\}$ and public key P_A. While, *Bob* is having identity ID_B

has private keys $S_B = \{s_B, x_B\}$ and public key P_B. *Alice* and *Bob* will establish an authenticated session as shown in figure 5.2 the following steps will be executed:

1. *Alice* chooses a random number $t_A \in \mathbb{Z}_n^*$ and computes $T_A = t_A \cdot P$
2. *Alice* computes a message $M_1 = \{ID_A, R_A, T_A\}$ and send it to *Bob*
3. *Bob* chooses a random number $t_B \in \mathbb{Z}_n^*$ and computes $T_B = t_B \cdot P$
4. *Bob* computes a message $M_2 = \{ID_B, R_B, T_B\}$ and sends to *Alice*
5. Upon receiving M_1 and M_2 respectively both *Alice* and *Bob* compute their shared secrets as follows:
   (a) *Alice* computes the following:
   - $u_A = x_A + s_A + t_A$
   - $W_B = R_B + h_B \cdot P_{pub}$
   - $K_{AB}^1 = u_A \cdot (l \cdot P_B + W_B)$, where $l = H_2(ID_B, R_B, P_B, T_B)$
   - $K_{AB}^2 = u_A \cdot (T_B + W_B)$
   - $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{AB}^1 \| K_{AB}^2)$

   (b) *Bob* computes the following:
   - $W_A = P_A + R_A + h_A \cdot P_{pub} + T_A$
   - $K_{BA}^1 = (l \cdot x_B + s_B) \cdot W_A$
   - $K_{BA}^2 = (t_B + s_B) \cdot W_A$
   - $sk = H_2(ID_A \| ID_B \| T_A \| T_B \| K_{BA}^1 \| K_{BA}^2)$

   (c) *Alice* and *Bob* both holds the same session key.

5.5 Analysis of Improved Kim et al.'s CTAKA Protocol

This section demonstrates the correctness analysis and security analysis of the Improved Kim et al.'s CTAKA Protocol.

5.5.1 Correctness Analysis

The correctness of the session keys generated by *Alice* and *Bob* during key agreement phase is as follows:

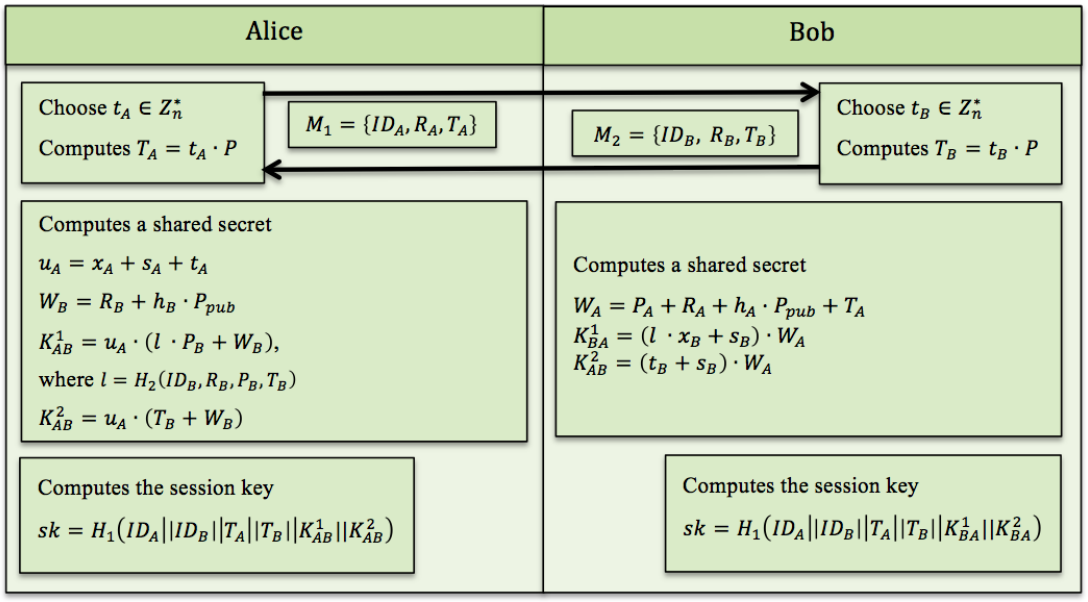

FIGURE 5.2: Key Agreement of Improved Kim et al.'s CTAKA Protocol

Lemma 5.1. *If two oracles are matching, both of them will be accepted and will get the same session-key, which is distributed uniformly at random in the session-key sample space.*

Proof. If two oracles are matching, then both of them are accepted and have the same session-key. The session-keys are distributed uniformly since t_A and t_B are selected uniformly during the protocol execution.

1. Correctness proof of $K_{AB}^1 = K_{BA}^1$

$$\begin{aligned} K_{AB}^1 &= u_A \cdot (l \cdot P_B + W_B) \\ &= (x_A + s_A + t_A)(l \cdot P_B + R_B + h_B \cdot P_{pub}) \\ &= (x_A + s_A + t_A)(l \cdot P_B + s_B \cdot P) \\ &= (x_A + s_A + t_A)(l \cdot x_B \cdot P + s_B \cdot P) \\ &= (x_A + s_A + t_A)(l \cdot x_B + s_B) \cdot P \\ &= (P_A + R_A + h_A \cdot P_{pub} + T_A)(l \cdot x_B + s_B) \\ &= W_A(l \cdot x_B + s_B) \\ &= K_{BA}^1 \end{aligned}$$

2. Correctness proof of $K^2_{AB} = K^2_{BA}$

$$\begin{aligned} K^2_{AB} &= u_A \cdot (T_B + W_B) \\ &= (x_A + s_A + t_A)(T_B + R_B + h_B \cdot P_{pub}) \\ &= (x_A + s_A + t_A)(t_B \cdot P + s_B \cdot P) \\ &= (x_A + s_A + t_A)(t_B + s_B) \cdot P \\ &= (P_A + R_A + h_A \cdot P_{pub} + T_A)(t_B + s_B) \\ &= W_A(t_B + s_B) \\ &= K^2_{BA} \end{aligned}$$

□

5.5.2 Security Analysis

This section demonstrates that the improved protocol is provably secure in the random oracle model by using extended Canetti-Krawczyk (eCK) model [16]. H_0, H_1 and H_2 hash functions are considered as three random oracles. Lemma 5.2 proves that the *GDH* problem is intractable against forging attack by Type I (malicious user) and Type II (malicious KGC) adversary. Type I adversary is denoted as $\mathcal{A}_I$ and Type II adversary is denoted as $\mathcal{A}_{II}$ (defined in Section 2.6.2). $\mathcal{C}$ be a Challenger, who can respond the queries asked by an adversary to solve a particular problem defined in the protocol. The objective of the security game is to solve the hardness problem with a non-negligible advantage $ADV_{\mathcal{A}}(k)$ in polynomial time t, which is impractical.

Lemma 5.2. *Based on the assumption that* GDH *problem is intractable, the advantage of Type I adversary against the proposed protocol is negligible.*

Proof. Suppose Type I adversary $\mathcal{A}_I$ can win the game defined above with a non-negligible advantage $Adv_{\mathcal{A}_I}(k)$ in polynomial time t. Let, $\mathcal{C}$ be a Challenger. $\mathcal{C}$ randomly chooses $P_0 \in \mathbb{G}$ as the system public-key P_{pub}. $\mathcal{C}$ also selects the system parameter $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1, H_2\}$, and sends $param$ to $\mathcal{A}_I$.

Let n_0 be the maximum number of sessions that any one party may have. Suppose adversary $\mathcal{A}_I$ activates at most n_1 distinct honest parties and n_2 distinct hash queries. Assuming $Adv_{\mathcal{A}_I}(k)$ is non-negligible. Since, H_0, H_1 and H_2 are modeled as a random oracle, after the adversary issues the *Test* query, it has only three possible ways to distinguish the tested session key from a random string:

1. *Guessing attack*. The adversary $\mathcal{A}_I$ correctly guesses the session-key.

2. *Key-replication attack.* The adversary $\mathcal{A}_I$ forces a non-matching session to have the same session key as the test session. In this case, the adversary $\mathcal{A}_I$ can simply learn the session-key by querying the non-matching session.

3. *Forging attack.* The adversary $\mathcal{A}_I$ computes the values K_{IJ}^1, K_{IJ}^2 itself on a test session $\prod_{I,J}^T$. The adversary $\mathcal{A}_I$ queries H_1 on the value $(ID_I, ID_J, T_I, T_J, K_{IJ}^1, K_{IJ}^2, sk)$ in the test session initiated by I and communicating with J.

The success probabilities of guessing attack and key-replication attack are negligible [63]. The probability of guessing the output of a random oracle H_1 is $(1/2^k)$. The input of H_1 includes all information that can uniquely identify the matching sessions. As, two non-matching sessions cannot have the same identities and the same ephemeral-public-keys. Therefore, guessing attack and key-replication attack can be ruled out, and the focus is mainly on the analysis of forging attack. For winning an advantage in the game against the protocol, the adversary $\mathcal{A}_I$ has to query the random oracle H_1 oracle on the session-key. To relate the advantage of the adversary $\mathcal{A}_I$ against our protocol to the *GDH* assumption, we use a classical reduction approach.

The challenger $\mathcal{C}$ is interested to use the adversary $\mathcal{A}_I$ to turn $\mathcal{A}_I$'s advantage indistinguishing the tested session-key from a random string into an advantage in solving the *GDH* problem. Let $Adv_{\mathcal{C}}^{GDH}(k)$ be the advantage that the challenger $\mathcal{C}$ acquires in solving the *GDH* problem given the security parameter k. To solve the *GDH* problem using $\mathcal{A}_I$, $\mathcal{C}$ is given a *GDH* challenge $U = u \cdot P$, $V = v \cdot P$ and an oracle $DDH(*, *, *)$, where $u, v \in \mathbb{Z}_n$, and $\mathcal{C}$'s task is to compute $GDH(U, V) = u \cdot v \cdot P$.

$\mathcal{C}$ simulates the game outlined in Section 2.6.2. $\mathcal{C}$ has to answer all queries of the adversary $\mathcal{A}_I$ throughout the game.

Before the game starts, $\mathcal{C}$ randomly chooses two indexes $I, J \in \{1, \cdots, n_1\} : I \neq J$, which represent the I^{th} and the J^{th} distinct honest parties. Also, $\mathcal{C}$ chooses $S \in 1, \cdots, n_0$. Let $\prod_{J,I}^S$ be the matching session of $\prod_{I,J}^S$. The challenger $\mathcal{C}$ has to guess that $\prod_{J,I}^S$ is the test-session, which is correct with probability larger than $\frac{1}{n_0 n_1^2}$. Then, there are two cases, which $\mathcal{C}$ has to be considered, (i) the test-session has a matching session owned by another honest party and (ii) no honest party owns a session matching with the test-session.

Case 1. *The test session has a matching session owned by another honest party.*

$\mathcal{A}_I$ is Strong Type I adversary and is able to acquire all user's secret-value x_i through *Replace-Public-Key* queries. $\mathcal{C}$ has four choices for $\mathcal{A}_I$'s strategy as described in Definition 2.10, $\mathcal{A}_I$ may learn (i) neither the ephemeral-secret of ID_I

nor that of ID_J, (ii) neither the ephemeral-secret of ID_J nor the partial-private-key of ID_I, (iii) neither the partial-private-key of ID_I nor that of ID_J and (iv) neither the ephemeral-secret of ID_I nor the partial-private-key of ID_J.

Case 1.1. $\mathcal{A}_I$ may learn neither the ephemeral-secret of ID_I i.e. t_I nor that of ID_J i.e. t_J.

$\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i\rangle$. $\mathcal{C}$ chooses three random numbers $s_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = s_iP - h_iP_{pub}$, $P_i = x_iP$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i\rangle$ and $\langle ID_i, R_i, h_i\rangle$ in L_C and L_{H_0} separately.

ii. $H_0(ID_i, R_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_0} comprising of tuple $\langle ID_i, R_i, h_i\rangle$.

- If $\langle ID_i, R_i\rangle$ is in the list L_{H_0}, $\mathcal{C}$ returns h_i.
- Else, $\mathcal{C}$ chooses a random number h_i, stores $\langle ID_i, R_i, h_i\rangle$ in L_{H_0} and returns h_i.

iii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answers to these queries in the following way:
  - $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, *\rangle$. If $\mathcal{C}$ finds the entry, it does the following:
    - If $ID_i = ID_I$, it computes $\bar{Z}_1 = Z_1 - (x_i + s_i)\cdot(l\cdot P_j + W_j) - l\cdot s_j \cdot T_i$ and $\bar{Z}_2 = Z_2 - (x_i + s_i)\cdot(T_j + W_j) - s_j \cdot T_i$, where $W_j = R_j + h_j \cdot P_0$.
    - If $ID_i = ID_J$, it computes $\bar{Z}_1 = Z_1$ and $\bar{Z}_2 = Z_2 - s_i \cdot W_j - s_i \cdot T_j$, where $W_j = P_j + R_j + h_j \cdot P_0 + T_j$.
    - $\mathcal{C}$ checks the correctness of $\bar{Z}_d, d = 1, 2$ is correct by checking whether the oracle $DDH(*, *, *)$ outputs 1 when the tuple $\langle *, *, \bar{Z}_d\rangle$ i.e. $\langle T_i, T_j, \bar{Z}_1\rangle$ is inputed. If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$ in L_{H_1}, where the value sk comes from L_S. Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$ in L_{H_1}.
  - Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$ in L_{H_1}.

iv. $H_2(ID_i, R_i, P_i, T_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_2} comprising of tuple $\langle ID_i, R_i, P_i, T_i, l\rangle$.

- If $\langle ID_i, R_i, P_i, T_i \rangle$ is in the list L_{H_2}, $\mathcal{C}$ returns l.
- Else, $\mathcal{C}$ chooses a random number l, stores $\langle ID_i, R_i, P_i, T_i, l \rangle$ in list L_{H_2} and returns l.

v. *Reveal-Partial-Private-Key(i).* $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:
- If $ID_i = ID_J$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ looks up the list L_C and returns the corresponding partial-private-key s_i to the adversary $\mathcal{A}_I$.

vi. *Reveal-Secret-Value(i).* $\mathcal{C}$ looks up the table L_C for entry $\langle ID_i, *, *, *, * \rangle$.
- If $\mathcal{C}$ finds the entry, it returns x_i.
- $\mathcal{C}$ carries out the query *Create(i)* and returns the corresponding x_i.

vii. *Replace-Public-Key*(i, P_i). $\mathcal{C}$ looks up the table L_C for entry $\langle ID_i, *, *, *, * \rangle$.
- If $\mathcal{C}$ finds the entry, it replaces x_i and P_i with x_i' and P_i' separately, where $P_i' = x_i' \cdot P$.
- $\mathcal{C}$ carries out *Create(i)* and replaces x_i and P_i with x_i' and P_i' separately.

viii. *Reveal-Ephemeral-Key*$(\prod_{i,j}^t)$. $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:
- If $\prod_{i,j}^t = \prod_{I,J}^T$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the stored ephemeral-private-key to $\mathcal{A}_I$.

ix. *Reveal-Master-Key.* $\mathcal{C}$ stops the simulation.

x. *Reveal-Session-Key*$(\prod_{i,j}^t)$. $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:
- If $\prod_{i,j}^t = \prod_{I,J}^T$ or $\prod_{i,j}^t = \prod_{J,I}^L$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the session-key sk to $\mathcal{A}_I$.

xi. *Send*$(\prod_{i,j}^t, m)$. $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and answers $\mathcal{A}_I$'s queries as follows:
- If $\prod_{i,j}^t = \prod_{I,J}^S$, then $\mathcal{C}$ returns $T_i = U$ to $\mathcal{A}_I$.
- Else, if $\prod_{i,j}^t = \prod_{J,I}^L$, then $\mathcal{C}$ returns $T_j = V$ to $\mathcal{A}_I$.
- Else, $\mathcal{C}$ replies according to the specification of protocol.

xii. *Test*$(\prod_{i,j}^t)$. $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:
- If $\prod_{i,j}^t \neq \prod_{I,J}^S$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ generates a random number $sk \in \{0,1\}^k$ and returns it to $\mathcal{A}_I$.

As the adversary $\mathcal{A}_I$ mounts the forging attack, if $\mathcal{A}_I$ succeeds, it must have queried oracle H_1 on the form:

- If $ID_i = ID_I$, then $u_i = x_i + s_i + t_i$, $W_j = R_j + h_j \cdot P_0$, $Z_1 = u_i \cdot (l \cdot P_j + W_j)$, where $l = H_2(ID_j, R_j, P_j, T_j)$, $Z_2 = u_i \cdot (V + W_j)$.
- If $ID_i = ID_J$, then $W_j = P_j + R_j + h_j \cdot P_0 + U$, $Z_1 = (l \cdot x_i + s_i) \cdot W_j$, where $l = H_2(ID_i, R_i, P_i, T_i)$, $Z_2 = (t_i + s_i) \cdot W_j$.

As, $T_i = U$ is the outgoing message of the test session given by the simulator and $T_j = V$ is the incoming message from the adversary $\mathcal{A}_I$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $1/n_2$ and proceeds with the following steps:

- If $ID_i = ID_I$, then $\bar{Z}_2 = Z_2 - (x_i + s_i) \cdot (V + W_j) - s_j \cdot U = GDH(U, V)$
- If $ID_i = ID_I$, then $\bar{Z}_2 = Z_2 - s_i \cdot W_j - s_i \cdot U = GDH(U, V)$, where $W_j = P_j + R_j + h_j \cdot P_0 + U$ and returns $\bar{Z}_2$ as the solution to $GDH(U, V)$.

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_I}(k)$.
Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_I}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 1.2. *$\mathcal{A}_I$ may learn neither the ephemeral-secret of ID_J i.e. t_J nor the partial-private-key of ID_I i.e. s_I.*

Case 1.3. *$\mathcal{A}_I$ may learn neither the partial-private-key of ID_I i.e. s_I nor that of ID_J i.e. s_J.*

Case 1.4. $\mathcal{A}_I$ may learn neither the ephemeral-secret of ID_I i.e. t_I nor the partial-private-key of ID_J i.e. s_J. We can prove that $Adv_{\mathcal{C}}^{GDH}(k)$ is negligible in *Case 1.2*, *Case 1.3* and *Case 1.4* by the same method as in *Case 1.1*.

Case 2. *No honest party owns a session matching with the test session.*

According to Definition 2.10, the following two cases should be considered.

Case 2.1. At some point, the static private key owned by the party I has been revealed by the adversary $\mathcal{A}_I$. Hence, according to the freshness definition, $\mathcal{A}_I$ is not permitted to reveal the ephemeral-private-key of the *Test* session. This case is similar to *Case 1.1* and can be solved using the same method as discussed above.

Case 2.2. The static private key owned by the party I has never been revealed by the adversary $\mathcal{A}_I$. Hence, according to the freshness definition, $\mathcal{A}_I$ may reveal party I's ephemeral-private-key in the *Test* session. This case is similar to *Case 1.1* and can be solved using the same method as discussed above.

If the adversary $\mathcal{A}_I$ succeeds with non-negligible probability in any of the cases above, we can also solve the *GDH* problem with non-negligible probability, which contradicts the assumed security of the *GDH* problem. So, we can conclude that the security of the proposed protocol is based on the *GDH* problem. □

Lemma 5.3. *Based on the assumption that* GDH *problem is intractable, the advantage of a Type II adversary against the proposed protocol is negligible.*

Proof. Suppose Type II adversary $\mathcal{A}_{II}$ can win the game defined above with a non-negligible advantage $Adv_{\mathcal{A}_{II}}(k)$ in polynomial time t. The proof of Lemma 5.3 is similar to Lemma 5.2, so we will skip the description and we will start the game.

Before the game starts, $\mathcal{C}$ randomly chooses two indexes $I, J \in \{1, \cdots, n_1\} : I \neq J$, which represent the I^{th} and the J^{th} distinct honest parties. Also, $\mathcal{C}$ chooses $S \in 1, \cdots, n_0$. Let $\prod_{J,I}^{S}$ be the matching session of $\prod_{I,J}^{S}$. The challenger $\mathcal{C}$ has to guess that $\prod_{J,I}^{S}$ is the test session, which is correct with probability larger than $\frac{1}{n_0 n_1^2}$. Then, there are two cases, which $\mathcal{C}$ has to be considered, (i) the test session has a matching session owned by another honest party and (ii) no honest party owns a session matching with the test session.

Case 1. *The test session has a matching session owned by another honest party.* $\mathcal{A}_{II}$ is Strong Type II adversary and is able to acquire all user's secret-key s_i values, as it acts as a malicious KGC and has access to the master key.

$\mathcal{C}$ has four choices for $\mathcal{A}_I I$'s strategy as described in Definition 2.11, $\mathcal{A}_I I$ may learn (i) neither the ephemeral-secret of ID_I nor the secret value of ID_J, (ii) neither the ephemeral-secret of ID_J nor the secret-value of ID_I, (iii) neither the secret-value of ID_I nor that of ID_J and (iv) neither the ephemeral-secret of ID_I nor that of ID_J.

Case 1.1. *$\mathcal{A}_{II}$ may learn neither the secret-value of ID_I i.e. x_I nor that of ID_J i.e. x_J.*

$\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows.

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i \rangle$.

- If $ID_i = ID_I$, $\mathcal{C}$ chooses two random numbers $h_i, r_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = U$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle IDi, s_i, R_i, \perp, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, if $ID_i = ID_J$, $\mathcal{C}$ chooses two random numbers $r_i, h_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = V$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, $\mathcal{C}$ chooses three random numbers $r_i, h_i, s_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = x_i \cdot P$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

ii. $H_0(ID_i, R_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_0} comprising of tuple $\langle ID_i, R_i, h_i \rangle$.
- If (ID_i, R_i) is on the list L_{H_0}, $\mathcal{C}$ returns h_i.
- Else, $\mathcal{C}$ chooses a random number h_i, stores $\langle ID_i, R_i, h_i \rangle$ in L_{H_0} and returns h_i.

iii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.
- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answers to these queries in the following way:
  - $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$. If $\mathcal{C}$ finds the entry, it does the following:
    * If $ID_i = ID_I$, it computes $\bar{Z}_1 = Z_1 - (s_i + t_i) \cdot (l \cdot P_j + W_j) - s_j \cdot P_i$ and $\bar{Z}_2 = Z_2 - (s_i + t_i) \cdot (T_j + W_j) - t_j \cdot P_i$, where $W_j = R_j + h_j \cdot P_0$.
    * If $ID_i = ID_J$, it computes $\bar{Z}_1 = Z_1 - (s_j + t_j) \cdot l \cdot P_i - s_i \cdot W_j$ and $\bar{Z}_2 = Z_1$, where $W_j = P_j + R_j + h_j \cdot P_0 + T_j$.
    * $\mathcal{C}$ checks the correctness of $\bar{Z}_d, d = 1, 2$ is correct by checking whether the oracle $DDH(*, *, *)$ outputs 1 when the tuple $\langle *, *, \bar{Z}_d \rangle$ i.e. $\langle P_i, P_j, \bar{Z}_1 \rangle$ is inputted. If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S. Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.
  - Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

iv. $H_2(ID_i, R_i, P_i, T_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_2} comprising of tuple $\langle ID_i, R_i, P_i, T_i, l \rangle$.
- If $\langle ID_i, R_i, P_i, T_i \rangle$ is in the list L_{H_2}, $\mathcal{C}$ returns l.
- Else, $\mathcal{C}$ chooses a random number l, stores $\langle ID_i, R_i, P_i, T_i, l \rangle$ in list L_{H_2} and returns l.

v. *Reveal-Partial-Private-Key(i).* $\mathcal{C}$ looks up the list L_C and returns the corresponding partial-private-key s_i to the adversary $\mathcal{A}_{II}$.

vi. *Reveal-Secret-Value(i).* $\mathcal{C}$ looks up the table L_C for entry $\langle ID_i, *, *, *, * \rangle$:
- If $ID_i = ID_I$ or $ID_i = ID_J$ then $\mathcal{C}$ stops the simulation.
- Else, if $\mathcal{C}$ find the entry then it returns x_i.
- Else, $\mathcal{C}$ carries out the query *Create(i)* and returns the corresponding x_i.

vii. *Reveal-Ephemeral-Key*($\prod_{i,j}^{t}$). $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:

- If $\prod_{i,j}^{t} \neq \prod_{I,J}^{T}$ and $\prod_{i,j}^{t} \neq \prod_{J,I}^{L}$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the stored ephemeral-private-key to $\mathcal{A}_{II}$.

viii. *Reveal-Master-Key.* $\mathcal{C}$ returns the master key s to $\mathcal{A}_{II}$.

ix. *Reveal-Session-Key*($\prod_{i,j}^{t}$). $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{T}$ or $\prod_{i,j}^{t} = \prod_{J,I}^{L}$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the session-key sk to $\mathcal{A}_{II}$.

x. *Send*($\prod_{i,j}^{t}, m$). $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk\rangle$ and answers $\mathcal{A}_{II}$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{T}$, then $\mathcal{C}$ returns $P_i = U$ to $\mathcal{A}_{II}$.
- Else, $\mathcal{C}$ replies according to the specification of protocol.

xi. *Test*($\prod_{i,j}^{t}$). $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:

- If $\prod_{i,j}^{t} \neq \prod_{I,J}^{T}$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ generates a random number $sk \in \{0,1\}^k$ and returns it to $\mathcal{A}_{II}$.

As the adversary $\mathcal{A}_{II}$ mounts the forging attack, if $\mathcal{A}_{II}$ succeeds, it must have queried oracle H_1 on the form:

- If $ID_i = ID_I$, then $u_i = x_i + s_i + t_i$, $W_j = R_j + h_j \cdot P_0$, $Z_1 = u_i \cdot (l \cdot V + W_j)$, where $l = H_2(ID_j, R_j, P_j, T_j)$, $Z_2 = u_i \cdot (T_j + W_j)$.
- If $ID_i = ID_J$, then $W_j = U + R_j + h_j \cdot P_0 + T_j$, $Z_1 = (l \cdot x_i + s_i) \cdot W_j$, where $l = H_2(ID_i, R_i, P_i, T_i)$, $Z_2 = (t_i + s_i) \cdot W_j$.

As, $P_i = U$ is the outgoing message of the test session given by the simulator and $P_j = V$ is the incoming message from the adversary $\mathcal{A}_{II}$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $1/n_2$ and proceeds with the following steps:

- If $ID_i = ID_I$, then $\bar{Z}_1 = Z_1 - (s_i + t_i) \cdot (l \cdot V + W_j) - s_j \cdot U$
- If $ID_i = ID_J$, then $\bar{Z}_1 = Z_1 - s_i \cdot W_j - (s_j + T_j) \cdot V = GDH(U, V)$, where $W_j = U + R_j + h_j \cdot P_0 + T_j$ and returns $\bar{Z}_1$ as the solution to $GDH(U, V)$.

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{9n_0 n_1^2 n_2} Adv_{\mathcal{A}_{II}}(k)$.
Then, $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_{II}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 1.2. *$\mathcal{A}_{II}$ may learn neither the ephemeral secret of ID_I i.e. t_I nor the secret value of ID_J i.e. x_J.*

Case 1.3. *$\mathcal{A}_{II}$ may learn neither the ephemeral secret of ID_J i.e. t_J nor the secret value of ID_I i.e. x_I.*

Case 1.4. *$\mathcal{A}_{II}$ may learn neither the ephemeral secret of ID_I i.e. t_I nor that of ID_J i.e. t_J.*

We can prove that $Adv_{\mathcal{A}_{II}}(k)$ is negligible in *Case 1.2*, *Case 1.3* and *Case 1.4* by the same method as in *Case 1.1*.

Case 2. *No honest party owns a session matching with the test session.*

According to Definition 2.11, the following two cases should be considered.

Case 2.1. At some point, the secret-value owned by the party I has been revealed by the adversary $\mathcal{A}_{II}$. Hence, according to the freshness definition, $\mathcal{A}_{II}$ is not permitted to reveal the ephemeral-Private-Key of the *Test* session. This case is similar to *Case 1.1* of this lemma and can be solved using the same method as discussed above.

Case 2.2. The secret-value owned by the party I has never been revealed by the adversary $\mathcal{A}_{II}$. Hence, according to the freshness definition, $\mathcal{A}_{II}$ may reveal party I's ephemeral-private-key in the *Test* session. This case is similar to *Case 1.1* of this lemma and can be solved using the same method as discussed above.

If the adversary $\mathcal{A}_{II}$ succeeds with non-negligible probability in any of the cases above, we can also solve the *GDH* problem with non-negligible probability, which contradicts the assumed security of the *GDH* problem. So, we can conclude that the security of the proposed protocol is based on the *GDH* problem.

□

5.6 Proposed Non-Interactive Certificateless Two-party Authenticated Key Agreement (NI-CTAKA) Protocol

This section describes the proposed authenticated non-interactive pairing-free key agreement protocol based on certificateless public key cryptosystem. The protocol consists of six phases: *Setup*, *Partial-Private-Key-Extract*, *Set-Secret-Value*, *Set-Private-Key*, *Set-Public-Key* and *Key-Agreement*. *Setup* and *Partial-Private-Key-Extract* algorithms run by Key Generation Center (KGC). The Base station performs the role of KGC. *Set-Secret-Value*, *Set-Private-Key*, *Set-Public-Key* and *Key-Agreement* algorithms run by the user. A sensor mote performs the role of the user.

Setup. Suppose $\mathbb{G}$ be a cyclic additive group of prime order q and P is the generator of group $\mathbb{G}$. For a given security parameter k, KGC chooses two hash functions H_0 and H_1 such that $H_0 : \{0,1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_n^*$ and $H_1 : \{0,1\}^* \times \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_n^*$.

The KGC chooses a random number $s \in \mathbb{Z}_n^*$ as a master key and computes $P_{pub} = s \cdot P$. Then, KGC publishes the system parameters as $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1\}$ and keep master key s secret.

Partial-Private-Key-Extract. For a given KGC's master key s, user's identity ID_i and system parameters $param$, KGC computes a partial-private-key for user identity ID_i. KGC chooses a random number $r_i \in \mathbb{Z}_n^*$ and computes $R_i = r_i \cdot P$, $h_i = H_0(ID_i, R_i)$ and $s_i = r_i + h_i \cdot s \bmod n$. KGC issues a partial-private-key $D_i = (s_i, R_i)$ to the user having identity ID_i through a secure channel. The user can validate its correction by checking whether $s_i \cdot P = R_i + h_i \cdot P_{pub}$.

Set-Secret-Value. The user with identity ID_i chooses a random number $x_i \in \mathbb{Z}_n^*$ and set x_i as its secret-value.

Set-Private-Key. For given system parameters $param$, user's partial-private-key D_i and its secret-value x_i, user with identity ID_i issues its private-key pair as $sk_i = (s_i, x_i)$.

Set-Public-Key. The user computes its public value as $P_i = x_i \cdot P$. For given system parameters $param$, user's partial-private-key D_i and its public value P_i, user with identity ID_i issues its public-key pair as $pk_i = (R_i, P_i)$.

Key-Agreement. For user Alice with identity ID_A has private-keys $sk_A = \{s_A, x_A\}$ and public-key $pk_A = \{R_A, P_A\}$ and the user Bob with identity ID_B has private-keys $sk_B = \{s_B, x_B\}$ and public-key $pk_B = \{R_B, P_B\}$. Alice and Bob will establish an authenticated session shown in figure 5.3:

1. Alice chooses a random number $t_A \in \mathbb{Z}_n^*$, set t_A as its ephemeral-secret-key and computes $T_A = t_A \cdot P$
2. Alice computes a message $M_1 = \{ID_A, T_A\}$ and send it to Bob
3. Bob chooses a random number $t_B \in \mathbb{Z}_n^*$, set t_B as its ephemeral-secret-key and computes $T_B = t_B \cdot P$
4. Bob computes a message $M_2 = \{ID_B, T_B\}$ and sends to Alice
5. Upon receiving M_1 and M_2 respectively both Alice and Bob compute their shared secrets as follows:
   (a) Alice computes a shared secret as $K_{AB}^1 = (s_A + x_A + t_A)(T_B + R_B + h_B \cdot P_{pub} + P_B)$ and $K_{AB}^2 = t_A \cdot T_B$ and the session-key as $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{AB}^1 \| K_{AB}^2)$.

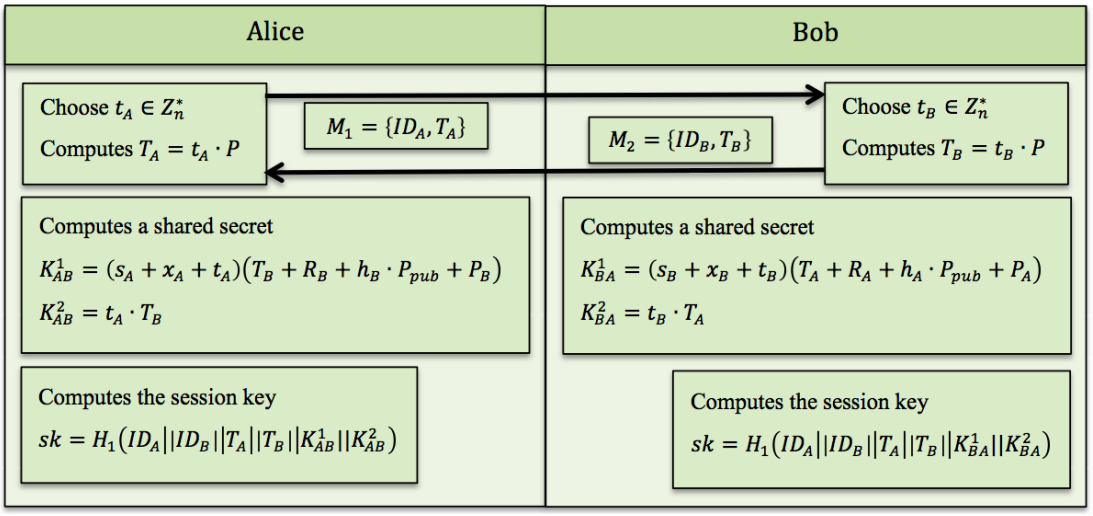

FIGURE 5.3: Key Agreement of NI-CTAKA Protocol

(b) Bob computes a shared secret as $K_{BA}^1 = (s_B + x_B + t_B)(T_A + R_A + h_A \cdot P_{pub} + P_A)$ and $K_{BA}^2 = t_B \cdot T_A$ and the session-key as $sk = H_1(ID_A \| ID_B \| T_A \| T_B \| K_{BA}^1 \| K_{BA}^2)$.

(c) Alice and Bob both holds the same session-key.

5.7 Analysis of NI-CTAKA Protocol

5.7.1 Correctness Analysis

This section demonstrates the correctness of the session-key generated during key agreement phase.

Lemma 5.4. *If two oracles are matching, both of them will be accepted and will get the same session-key, which is distributed uniformly at random in the session-key sample space.*

Proof. If two oracles are matching, then both of them are accepted and have the same session-key. The session-keys are distributed uniformly since t_A and t_B are selected uniformly during the protocol execution.

1. Correctness proof of $K_{AB}^1 = K_{BA}^1$

$$\begin{aligned} K_{AB}^1 &= (s_A + x_A + t_A)(T_B + R_B + h_B \cdot P_{pub} + P_B) \\ &= (s_A + x_A + t_A)(t_B \cdot P + r_B \cdot P + h_B \cdot s \cdot P + x_B \cdot P) \\ &= (s_A + x_A + t_A)(t_B + r_B + h_B \cdot s + x_B) \cdot P \\ &= (s_A + x_A + t_A)(t_B + s_B + x_B) \cdot P \\ &= (s_B + x_B + t_B)(t_A + r_A + h_A \cdot s + x_A) \cdot P \\ &= (s_B + x_B + t_B)(t_A \cdot P + r_A \cdot P + h_A \cdot s \cdot P + x_A \cdot P) \\ &= (s_B + x_B + t_B)(T_A + R_A + h_A \cdot P_{pub} + P_A) \\ &= K_{BA}^1 \end{aligned}$$

2. Correctness proof of $K_{AB}^2 = K_{BA}^2$

$$\begin{aligned} K_{AB}^2 &= t_A \cdot T_B \\ &= t_A \cdot t_B \cdot P \\ &= t_B \cdot T_A \\ &= K_{BA}^2 \end{aligned}$$

□

5.7.2 Security Analysis

This section demonstrates that the proposed protocol is provably secure in the random oracle model by using eCK model. H_0 and H_1 hash functions are considered as two random oracles. The Lemma 5.5 and Lemma 5.6 are provided for the security of the proposed protocol and proves that the *GDH* problem is intractable against forging attack by Type I (malicious user) and Type II (malicious KGC) adversary. Type I adversary is denoted as $\mathcal{A}_I$ and Type II adversary is denoted as $\mathcal{A}_{II}$ (defined in Section 2.6.2). $\mathcal{C}$ be a Challenger, who can respond to the queries asked by an adversary to solve a particular problem defined in the protocol. The objective of the security game is to solve the hardness problem with a non-negligible advantage $Adv_{\mathcal{A}}(k)$ in polynomial time t, which is impractical.

Lemma 5.5. *Based on the assumption that* GDH *problem is intractable, the advantage of a Type I adversary against the proposed protocol is negligible.*

Proof. Suppose Type I adversary $\mathcal{A}_I$ can win the game defined in Section 2.6.2 with a non-negligible advantage $Adv_{\mathcal{A}_I}(k)$ in polynomial time t. Let, $\mathcal{C}$ be a Challenger. $\mathcal{C}$

randomly chooses $P_0 \in \mathbb{G}$ as the system public-key P_{pub}. $\mathcal{C}$ also selects the system parameter $param = \{\mathbb{F}_p, \mathbb{E}/\mathbb{F}_p, \mathbb{G}, P, P_{pub}, H_0, H_1\}$, and sends $param$ to $\mathcal{A}_I$.

Let n_0 be the maximum number of sessions that any one party may have. Suppose adversary $\mathcal{A}_I$ activates at most n_1 distinct honest parties and n_2 distinct hash queries. Assuming $Adv_{\mathcal{A}_I}(k)$ is non-negligible. Since, H_0 and H_1 are modeled as a random oracle, after the adversary issues the *Test* query, it has only three possible ways to distinguish the tested session-key from a random string:

1. *Guessing attack*. The adversary $\mathcal{A}_I$ correctly guesses the session-key.
2. *Key-replication attack*. The adversary $\mathcal{A}_I$ forces a non-matching session to have the same session-key as the test-session. In this case, the adversary $\mathcal{A}_I$ can simply learn the session-key by querying the non-matching session.
3. *Forging attack*. The adversary $\mathcal{A}_I$ computes the values K_{IJ}^1, K_{IJ}^2 itself on a test-session $\prod_{I,J}^T$. The adversary $\mathcal{A}_I$ queries H_1 on the value $(ID_I, ID_J, T_I, T_J, K_{IJ}^1, K_{IJ}^2, sk)$ in the test-session initiated by I and communicating with J.

The success probabilities of guessing attack and key-replication attack are negligible [63]. The probability of guessing the output of a random oracle H_1 is $(1/2^k)$. The input of H_1 includes all information that can uniquely identify the matching sessions. As, two non-matching sessions cannot have the same identities and the same ephemeral-public-keys. Therefore, guessing attack and key-replication attack can be ruled out, and the focus is mainly on the analysis of forging attack. For winning an advantage in the game against the protocol, the adversary $\mathcal{A}_I$ has to query the random oracle H_1 oracle on the session-key. To relate the advantage of the adversary $\mathcal{A}_I$ against our protocol to the *GDH* assumption, we use a classical reduction approach. The challenger $\mathcal{C}$ is interested to use the adversary $\mathcal{A}_I$ to turn $\mathcal{A}_I$'s advantage indistinguishing the tested session-key from a random string into an advantage in solving the *GDH* problem. Let $Adv_{\mathcal{C}}^{GDH}(k)$ be the advantage that the challenger $\mathcal{C}$ acquires in solving the *GDH* problem given the security parameter k. To solve the *GDH* problem using $\mathcal{A}_I$, $\mathcal{C}$ is given a *GDH* challenge $U = u \cdot P$, $V = v \cdot P$ and an oracle $DDH(*, *, *)$, where $u, v \in \mathbb{Z}_n^*$, and $\mathcal{C}$'s task is to compute $GDH(U, V) = u \cdot v \cdot P$.

$\mathcal{C}$ simulates the game outlined in Section 2.6.2. $\mathcal{C}$ has to answer all queries of the adversary $\mathcal{A}_I$ throughout the game.

Before the game starts, $\mathcal{C}$ randomly chooses two indexes $I, J \in \{1, \cdots, n_1\} : I \neq J$, which represent the I^{th} and the J^{th} distinct honest parties. Also, $\mathcal{C}$ chooses $S \in 1, \cdots, n_0$. Let $\prod_{J,I}^S$ be the matching session of $\prod_{I,J}^S$. The challenger $\mathcal{C}$ has to guess that

$\prod_{J,I}^{S}$ is the test-session, which is correct with probability larger than $\frac{1}{n_0 n_1^2}$. Then, there are two cases, which $\mathcal{C}$ has to be considered, (i) the test-session has a matching session owned by another honest party and (ii) no honest party owns a session matching with the test-session.

Case 1. *The test-session has a matching session owned by another honest party.*

$\mathcal{A}_I$ is Strong Type I adversary and is able to acquire all user's secret-key x_i values through *Replace-Public-Key* queries. $\mathcal{C}$ has four choices for $\mathcal{A}_I$'s strategy as described in Definition 2.10, $\mathcal{A}_I$ may learn (i) neither the ephemeral-secret of ID_I nor the partial-private-key of ID_J, (ii) neither the ephemeral-secret of ID_J nor the partial-private-key of ID_I, (iii) neither the partial-private-key of ID_I nor that of ID_J and (iv) neither the ephemeral-secret of ID_I nor that of ID_J.

Case 1.1. *$\mathcal{A}_I$ may learn neither the ephemeral-secret of ID_I i.e. t_I nor the partial-private-key of ID_J i.e. s_J.*

$\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i \rangle$.

- If $ID_i = ID_J$, $\mathcal{C}$ chooses two random numbers $h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = U - h_i \cdot P_0$, $P_i = x_i \cdot P$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle IDi, \perp, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, $\mathcal{C}$ chooses three random numbers $s_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = s_i \cdot P - h_i \cdot P_{pub}$, $P_i = x_i \cdot P$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

ii. $H_0(ID_i, R_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_0} comprising of tuple $\langle ID_i, R_i, h_i \rangle$.

- If (ID_i, R_i) is in the list L_{H_0}, $\mathcal{C}$ returns h_i.
- Else, $\mathcal{C}$ chooses a random number h_i, stores $\langle ID_i, R_i, h_i \rangle$ in L_{H_0} and returns h_i.

iii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answers to these queries in the following way:
  - If $ID_i = ID_J$, $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$.
    - If $\mathcal{C}$ finds the entry, it computes $\bar{Z_1} = Z_1 - t_i \cdot (T_j + R_j + H_0(ID_j, R_j) + P_j) - s_j \cdot (R_i + H_0(ID_i, R_i) + P_i) - x_j \cdot (R_i + H_0(ID_i, R_i) + P_i) - t_j \cdot (R_i + H_0(ID_i, R_i))$. $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*, *, *)$ outputs

1 when the tuple $\langle T_j, (R_i + H_0(ID_i, R_i) \cdot P_{pub}), \bar{Z}_1 \rangle$ is input. $\mathcal{C}$ also checks the correctness of Z_2 by checking whether the equation $Z_2 = t_i \cdot T_j$ hold. If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S .

* Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

– Else, $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$.

* If $\mathcal{C}$ finds the entry, it stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S.

* Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

iv. *Reveal-Partial-Private-Key(i).* $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

- If $ID_i = ID_J$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ looks up the list L_E and returns the corresponding partial-private-key s_i to the adversary $\mathcal{A}_I$.

v. *Reveal-Secret-Value(i).* $\mathcal{C}$ looks up the list L_C for entry $\langle ID_i, *, *, *, * \rangle$.

- If $\mathcal{C}$ finds the entry, it returns x_i.
- $\mathcal{C}$ carries out the query *Create(i)* and returns the corresponding x_i.

vi. *Replace-Public-Key*(i, P_i). $\mathcal{C}$ looks up the list L_C for entry $\langle ID_i, *, *, *, * \rangle$.

- If $\mathcal{C}$ finds the entry, it replaces x_i and P_i with x_i' and P_i' separately, where $P_i' = x_i' \cdot P$.
- $\mathcal{C}$ carries out *Create(i)* and replaces x_i and P_i with x_i' and P_i' separately.

vii. *Reveal-Ephemeral-Key*$(\prod_{i,j}^t)$. $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^t = \prod_{I,J}^T$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the stored ephemeral-private-key to $\mathcal{A}_I$.

viii. *Reveal-Master-Key.* $\mathcal{C}$ stops the simulation.

ix. *Reveal-Session-Key*$(\prod_{i,j}^t)$. $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^t = \prod_{I,J}^T$ or $\prod_{i,j}^t = \prod_{J,I}^L$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the session-key sk to $\mathcal{A}_I$.

x. *Send*$(\prod_{i,j}^t, m)$. $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and answers $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^t = \prod_{I,J}^S$, then $\mathcal{C}$ returns $T_i = V$ to $\mathcal{A}_I$.
- Else, if $ID_i = ID_J$, it generates a random $t_i \in \mathbb{Z}_n$, and computes $\bar{Z}_1 = Z_1 - t_i \cdot (T_j + R_j + H_0(ID_j, R_j) + P_j) - s_j \cdot (R_i + H_0(ID_i, R_i) +$

$P_i) - x_j \cdot (R_i + H_0(ID_i, R_i) + P_i) - t_j \cdot (R_i + H_0(ID_i, R_i))$. $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*,*,*)$ outputs 1 when the tuple $\langle T_j, (R_i + H_0(ID_i, R_i) \cdot P_{pub}), \bar{Z}_1 \rangle$ is input. $\mathcal{C}$ also checks the correctness of Z_2 by checking whether the equation $Z_2 = t_i \cdot T_j$ hold.

- If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ in L_S, where the value sk comes from L_{H_1}.
- $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ in L_S.

- Else, $\mathcal{C}$ replies according to the specification of protocol.

xi. *Test*($\prod_{i,j}^t$). $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^t \neq \prod_{I,J}^S$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ generates a random number $sk \in \{0,1\}^k$ and returns it to $\mathcal{A}_I$.

As the adversary $\mathcal{A}_I$ mounts the forging attack, if $\mathcal{A}_I$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I + x_I) \cdot (T_J + R_J + H_0(ID_J, R_J) \cdot P_{pub} + P_J) = (t_I + s_I + x_I) \cdot (T_J + U + P_J)$ and $Z_2 = t_I \cdot T_J$, where $T_I = V$ is the outgoing message of the test-session given by the simulator and T_J is the incoming message from the adversary $\mathcal{A}_I$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $\frac{1}{n_2}$ and proceeds with the following steps:

$\mathcal{C}$ computes $\bar{Z}_1 = Z_1 - (s_I + x_I) \cdot (T_J + U + P_J) - t_I \cdot P_J$. It is easy to verify that the equation $\bar{Z}_1 = GDH(T_I, T_J) + GDH(U, V)$ holds. Then, $\mathcal{C}$ computes $GDH(U, V) = \bar{Z}_1 - GDH(T_I, T_J) = \bar{Z}_1 - Z_2$.

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_I}(k)$.

Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_I}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 1.2. *$\mathcal{A}_I$ may learn neither the ephemeral-secret of ID_J i.e. t_J nor the partial-private-key of ID_I i.e. s_I.*

By exchanging the roles of I and J in the above case, we can prove that $Adv_{\mathcal{C}}^{GDH}(k)$ is negligible by the same method as in the above case.

Case 1.3. *$\mathcal{A}_I$ may learn neither the partial-private-key of ID_I i.e. s_I nor that of ID_J i.e. s_J.*

$\mathcal{C}$ responds to $H_0(ID_i, R_i)$, *Replace-Public-Key* (i, P_i), *Reveal-Secret-Value(i)*, *Reveal-Ephemeral-Key* ($\prod_{i,j}^s$), *Reveal-Master-Key*, *Reveal-Session-Key* ($\prod_{i,j}^s$) and *Test* ($\prod_{i,j}^s$) as it does in *Case 1.1*. It also answers the other queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i \rangle$.

- If $ID_i = ID_I$, $\mathcal{C}$ chooses two random numbers $h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = U - h_i \cdot P_{pub}$, $P_i = x_i \cdot P$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle IDi, \perp, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, if $ID_i = ID_J$, $\mathcal{C}$ chooses two random numbers $h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = V - h_i \cdot P_{pub}$, $P_i = x_i \cdot P$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, \perp, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, $\mathcal{C}$ chooses three random numbers $s_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = s_i \cdot P - h_i \cdot P_0$, $P_i = x_i \cdot P$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

ii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answers to these queries in the following way:
  - If $ID_i = ID_I$ or $ID_i = ID_J$, $\mathcal{C}$ simulates the oracle in the same way as in *Case 1.1.*
  - Else, $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$.
    - If $\mathcal{C}$ finds the entry, it stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S .
    - Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

iii. *Reveal-Partial-Private-Key(i).* $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

- If $ID_i = ID_I$ or $ID_i = ID_J$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ looks up the list L_C and returns the corresponding partial-private-key s_i to the adversary $\mathcal{A}_I$.

iv. *Send*$(\prod_{i,j}^t, m)$. $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and answers $\mathcal{A}_I$'s queries as follows:

- If $ID_i = ID_I$ or $ID_i = ID_J$, $\mathcal{C}$ simulates the oracle in the same way as in *Case 1.1.*
- Else, $\mathcal{C}$ replies according to the specification of the protocol.

As, the adversary $\mathcal{A}_I$ mounts the forging attack, if $\mathcal{A}_I$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I + x_I) \cdot (T_J + R_J + H_0(ID_J, R_J) \cdot P_{pub} + P_J) = (t_I + s_I + x_I) \cdot (T_J + V + P_J)$ and $Z_2 = t_I \cdot T_J$, where $R_I + H_0(ID_I, R_I) \cdot P_{pub} = U$ and T_J are the incoming messages from the adversary $\mathcal{A}_I$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $\frac{1}{n_2}$ and proceeds with the following steps:

$\mathcal{C}$ computes $GDH(U,V) = Z_1 - t_I \cdot (T_J + V) - t_J \cdot U$.

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_I}(k)$.

Then, $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_I}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 1.4. *$\mathcal{A}_I$ may learn neither the ephemeral-secret of ID_I i.e. t_I nor that of ID_J i.e. t_J.*

$\mathcal{C}$ responds to $H_0(ID_i, R_i)$, *Replace-Public-Key* (i, P_i), *Reveal-Secret-Value(i)*, *Reveal-Master-Key*, *Reveal-Session-Key* $(\prod_{i,j}^{s})$ and *Test* $(\prod_{i,j}^{s})$ as it does in the above case. It also answers the other queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i \rangle$. $\mathcal{C}$ chooses three random numbers $s_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = s_i \cdot P - h_i \cdot P_{pub}$, $P_i = x_i \cdot P$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

ii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answers to these queries in the following way:
  - $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$. If $\mathcal{C}$ finds the entry, it computes $\bar{Z}_1 = Z_1 - (s_i + x_i) \cdot (T_j + R_j + H_0(ID_j, R_j) + P_j) - (s_j + x_j) \cdot T_i$. Then, $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*,*,*)$ outputs 1 when the tuple $\langle T_i, T_j, \bar{Z}_1 \rangle$ is input.
    - If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S.
    - Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.
  - Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

iii. *Reveal-Partial-Private-Key(i).* $\mathcal{C}$ looks up the list L_C and returns the corresponding partial-private-key s_i to the adversary $\mathcal{A}_I$.

iv. *Reveal-Ephemeral-Key*$(\prod_{i,j}^{t})$. $\mathcal{C}$ answers $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{T}$ or $\prod_{i,j}^{t} = \prod_{J,I}^{L}$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the stored ephemeral-private-key to $\mathcal{A}_I$.

v. *Send*$(\prod_{i,j}^{t}, m)$. $\mathcal{C}$ maintains an initially empty list L_S with entries of the form $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and answers $\mathcal{A}_I$'s queries as follows:

- If $\prod_{i,j}^{t} = \prod_{I,J}^{T}$, $\mathcal{C}$ returns $T_i = U$ to $\mathcal{A}_I$.

- Else, if $\prod_{i,j}^{t} = \prod_{I,J}^{L}$, $\mathcal{C}$ returns $T_i = V$ to $\mathcal{A}_I$.
- Else, $\mathcal{C}$ replies according to the specification of protocol.

As, the adversary $\mathcal{A}_I$ mounts the forging attack, if $\mathcal{A}_I$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I + x_I) \cdot (T_J + R_J + H_0(ID_J, R_J) \cdot P_{pub} + P_J)$ and $Z_2 = t_I \cdot T_J$, where $T_I = U$ is the outgoing message of the test-session given by the simulator and $T_J = V$ is the incoming message from the adversary $\mathcal{A}_I$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $\frac{1}{n_2}$ and returns Z_2 as the solution to $GDH(U, V)$.

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_I}(k)$.

Then, $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_I}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 2. *No honest party owns a session matching with the test-session.*

According to Definition 2.10, the following two cases should be considered.

Case 2.1. At some point, the static private-key owned by the party I has been revealed by the adversary $\mathcal{A}_I$. Hence, according to the freshness definition, $\mathcal{A}_I$ is not permitted to reveal the ephemeral-private-key of the test-session. This case is similar to *Case 1.1* and can be solved using the same method as in the above section.

Case 2.2. The static private-key owned by the party I has never been revealed by the adversary $\mathcal{A}_I$. Hence, according to the freshness definition, $\mathcal{A}_I$ may reveal party I's ephemeral-private-key in the test-session. This case is similar to *Case 1.2* and can be solved using the same method as in the above section.

If the adversary $\mathcal{A}_I$ succeeds with non-negligible probability in any of the cases above, we can also solve the *GDH* problem with non-negligible probability, which contradicts the assumed security of the *GDH* problem. So, we can conclude that the security of the proposed protocol is based on the *GDH* problem.

□

Lemma 5.6. *On the assumption that the* GDH *problem is intractable, the advantage of a Type II adversary against the proposed protocol is negligible.*

Proof. Suppose Type II adversary $\mathcal{A}_{II}$ can win the game defined in Section 2.6.2 with a non-negligible advantage $Adv_{\mathcal{A}_{II}}(k)$ in polynomial time t. The proof of Lemma 5.6 is similar to Lemma 5.5, so we will skip the description and we will start the game.

Before the game starts, $\mathcal{C}$ randomly chooses two indexes $I, J \in \{1, \cdots, n_1\} : I \neq J$, which represent the I^{th} and the J^{th} distinct honest parties. Also, $\mathcal{C}$ chooses $S \in 1, \cdots, n_0$. Let $\prod_{J,I}^{S}$ be the matching session of $\prod_{I,J}^{S}$. The challenger $\mathcal{C}$ has to guess that $\prod_{J,I}^{S}$ is the test-session, which is correct with probability larger than $\frac{1}{n_0 n_1^2}$. Then, there are two cases, which $\mathcal{C}$ has to be considered, (i) the test-session has a matching session owned by another honest party and (ii) no honest party owns a session matching with the test-session.

Case 1. *The test-session has a matching session owned by another honest party.* $\mathcal{A}_{II}$ is Strong Type II adversary and is able to acquire all user's secret-key s_i values, as it acts as a malicious KGC and has access to the master key. $\mathcal{C}$ has four choices for $\mathcal{A}_{II}$'s strategy as described in Definition 2.11, $\mathcal{A}_{II}$ may learn (i) neither the ephemeral-secret of ID_I nor the secret-value of ID_J, (ii) neither the ephemeral-secret of ID_J nor the secret-value of ID_I, (iii) neither the secret-value of ID_I nor that of ID_J and (iv) neither the ephemeral-secret of ID_I nor that of ID_J.

Case 1.1. *$\mathcal{A}_{II}$ may learn neither the ephemeral-secret of ID_I i.e. t_I nor the secret-value of ID_J i.e. x_J.*

$\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i \rangle$.

- If $ID_i = ID_J$, $\mathcal{C}$ chooses two random numbers $h_i, r_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = U$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle IDi, s_i, R_i, \perp, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.
- Else, $\mathcal{C}$ chooses three random numbers $r_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = x_i \cdot P$, $s_i = r_i + h_i \cdot x \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i \rangle$ and $\langle ID_i, R_i, h_i \rangle$ in L_C and L_{H_0} separately.

ii. $H_0(ID_i, R_i)$. $\mathcal{C}$ maintains an initially empty list L_{H_0} comprising of tuple $\langle ID_i, R_i, h_i \rangle$.

- If (ID_i, R_i) is on the list L_{H_0}, $\mathcal{C}$ returns h_i.
- Else, $\mathcal{C}$ chooses a random number h_i, stores $\langle ID_i, R_i, h_i \rangle$ in L_{H_0} and returns h_i.

iii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answerss to these queries in the following way:
  - If $ID_i = ID_J$, $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$.

* If $\mathcal{C}$ finds the entry, it computes $\bar{Z_1} = Z_1 - (t_i + s_i) \cdot (T_j + R_j + H_0(ID_j, R_j) + P_j) - x_i \cdot (R_j + H_0(ID_j, R_j) + P_j)$. $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*, *, *)$ outputs 1 when the tuple $\langle P_i, T_j, \bar{Z_1} \rangle$ is input. $\mathcal{C}$ also checks the correctness of Z_2 by checking whether the equation $Z_2 = t_i \cdot T_j$ hold. If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S .
* Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

– Else, $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, * \rangle$.
* If $\mathcal{C}$ finds the entry, it stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}, where the value sk comes from L_S.
* Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk \rangle$ in L_{H_1}.

iv. *Reveal-Partial-Private-Key(i).* $\mathcal{C}$ looks up the list L_C and returns the corresponding partial-private-key s_i to the adversary $\mathcal{A}_{II}$.

v. *Reveal-Secret-Value(i).* $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:
- If $ID_i = ID_J$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ looks up the list L_C for entry $\langle ID_i, *, *, *, * \rangle$ and returns x_i.

vi. *Reveal-Ephemeral-Key*($\prod_{i,j}^t$). $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:
- If $\prod_{i,j}^t = \prod_{I,J}^S$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the stored ephemeral-private-key to $\mathcal{A}_{II}$.

vii. *Reveal-Master-Key.* $\mathcal{C}$ returns the master key s to $\mathcal{A}_{II}$.

viii. *Reveal-Session-Key*($\prod_{i,j}^t$). $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:
- If $\prod_{i,j}^t = \prod_{I,J}^S$ or $\prod_{i,j}^t = \prod_{J,I}^T$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ returns the session-key sk to $\mathcal{A}_{II}$.

ix. *Send*($\prod_{i,j}^t, m$). $\mathcal{C}$ maintains an initially empty list L_S comprising of tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ and answers $\mathcal{A}_{II}$'s queries as follows:
- If $\prod_{i,j}^t = \prod_{I,J}^T$, then $\mathcal{C}$ returns $T_i = V$ to $\mathcal{A}_{II}$.
- Else, if $ID_i = ID_J$, it generates a random $t_i \in \mathbb{Z}_n^*$, and computes $\bar{Z_1} = Z_1 - (t_i + s_i) \cdot (T_j + R_j + H_0(ID_j, R_j) + P_j) - x_i \cdot (R_j + H_0(ID_j, R_j) + P_j)$. $\mathcal{C}$ checks the correctness of Z_1 by checking whether the oracle $DDH(*, *, *)$ outputs 1 when the tuple $\langle P_i, T_j, \bar{Z_1} \rangle$ is input. $\mathcal{C}$ also checks the correctness of Z_2 by checking whether the equation $Z_2 = t_i \cdot T_j$ hold.
  – If Z_1 and Z_2 are correct, $\mathcal{C}$ stores the tuple $\langle ID_i, ID_j, T_i, T_j, sk \rangle$ in L_S, where the value sk comes from L_{H_1}.

– $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, sk\rangle$ in L_S.

- Else, $\mathcal{C}$ replies according to the specification of protocol.

x. *Test*($\prod_{i,j}^{t}$). $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:

- If $\prod_{i,j}^{t} \neq \prod_{I,J}^{T}$, then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ generates a random number $sk \in \{0,1\}^k$ and returns it to $\mathcal{A}_{II}$.

As the adversary $\mathcal{A}_{II}$ mounts the forging attack, if $\mathcal{A}_{II}$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I + x_I)\cdot(T_J + R_J + H_0(ID_J, R_J) + U)$ and $Z_2 = t_I \cdot T_J$, where $T_I = V$ is the outgoing message of the test-session given by the simulator and T_J is the incoming message from the adversary $\mathcal{A}_{II}$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $\frac{1}{n_2}$ and proceeds with the following steps:

$\mathcal{C}$ computes $\bar{Z}_1 = Z_1 - (x_I + s_I)\cdot(T_J + R_J + H_0(ID_J, R_J) + U) - t_I \cdot (R_J + H_0(ID_J, R_J))$. It is easy to verify that the equation $\bar{Z}_1 = GDH(T_I, T_J) + GDH(U, V)$ holds. Then $\mathcal{C}$ computes $GDH(U, V) = \bar{Z}_1 - GDH(T_I, T_J) = \bar{Z}_1 - Z_2$.

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_{II}}(k)$.

Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_{II}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 1.2. *$\mathcal{A}_{II}$ may learn neither the ephemeral-secret of ID_J i.e. t_J nor the secret-value of ID_I i.e. x_I.*

By exchanging the roles of I and J in the above case, we can prove that $Adv_{\mathcal{C}}^{GDH}(k)$ is negligible by the same method as in the above case.

Case 1.3. *$\mathcal{A}_{II}$ may learn neither the secret-value of ID_I i.e. x_I nor that of ID_J i.e. x_J.*

$\mathcal{C}$ answers $H_0(ID_i, R_i)$, *Reveal-Partial-Private-Key* (i), *Reveal-Ephemeral-Key*($\prod_{i,j}^{s}$), *Reveal-Master-Key*, *Reveal-Session-Key*($\prod_{i,j}^{s}$) and *Test*($\prod_{i,j}^{s}$) as it does in *Case 1.1* of this lemma. It also answers the other queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C consisting of tuple of the form $\langle ID_i, s_i, R_i, x_i, P_i\rangle$.

- If $ID_i = ID_I$, $\mathcal{C}$ chooses two random numbers $h_i, r_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = U$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle IDi, s_i, R_i, \perp, P_i\rangle$ and $\langle ID_i, R_i, h_i\rangle$ in L_C and L_{H_0} separately.
- Else, if $ID_i = ID_J$, $\mathcal{C}$ chooses two random numbers $h_i, r_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = V$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$

and stores $\langle IDi, s_i, R_i, \perp, P_i\rangle$ and $\langle ID_i, R_i, h_i\rangle$ in L_C and L_{H_0} separately.

- Else, $\mathcal{C}$ chooses three random numbers $r_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = x_i \cdot P$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i\rangle$ and $\langle ID_i, R_i, h_i\rangle$ in L_C and L_{H_0} separately.

ii. $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$. $\mathcal{C}$ maintains an initially empty list L_{H_1} comprising of tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$.

- If the tuple is in the list L_{H_1}, $\mathcal{C}$ returns sk.
- Else, $\mathcal{C}$ answers to these queries in the following way:
  - If $ID_i = ID_I$ or $ID_i = ID_J$, $\mathcal{C}$ simulates the oracle in the same way as in *Case 1.1* of this lemma.
  - Else, $\mathcal{C}$ looks up the list L_S for entry $\langle ID_i, ID_j, T_i, T_j, *\rangle$.
    - If $\mathcal{C}$ finds the entry, it stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$ in L_{H_1}, where the value sk comes from L_S.
    - Else, $\mathcal{C}$ chooses a random number $sk \in \{0,1\}^k$ and stores the tuple $\langle ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk\rangle$ in L_{H_1}.

iii. *Reveal-Secret-Value(i).* $\mathcal{C}$ answers $\mathcal{A}_{II}$'s queries as follows:

- If $ID_i = ID_I$ or $ID_i = ID_J$ then $\mathcal{C}$ stops the simulation.
- Else, $\mathcal{C}$ looks up the list L_C for entry $\langle ID_i, *, *, *, *\rangle$ and returns x_i.

iv. *Send*$(\prod_{i,j}^{t}, m)$. $\mathcal{C}$ maintains an initially empty list L_S with entries of the form $\langle ID_i, ID_j, T_i, T_j, sk\rangle$ and answers $\mathcal{A}_{II}$'s queries as follows:

- If $ID_i = ID_I$ or $ID_i = ID_J$, $\mathcal{C}$ simulates the oracle in the same way as in *Case 1.1* of this lemma.
- Else, $\mathcal{C}$ replies according to the specification of protocol.

As the adversary $\mathcal{A}_{II}$ mounts the forging attack, if $\mathcal{A}_{II}$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I + x_I) \cdot (T_J + U)$ and $Z_2 = t_I \cdot T_J$, where $P_I = U$, $P_J = V$ and T_J is the incoming message from the adversary $\mathcal{A}_{II}$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $\frac{1}{n_2}$ and proceeds with the following steps:

$\mathcal{C}$ computes $GDH(U, V) = Z_1 - t_I \cdot (T_J + U) - t_J \cdot V$

The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_{II}}(k)$.

Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_{II}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 1.4. *$\mathcal{A}_{II}$ may learn neither the ephemeral-secret of ID_I i.e. t_I nor that of ID_J i.e. t_J.*

$\mathcal{C}$ responds to $H_0(ID_i, R_i)$, *Reveal-Partial-Private-Key*(i), *Reveal-Master-Key*, *Reveal-Session-Key* $(\prod_{i,j}^{s})$ and *Test* $(\prod_{i,j}^{t})$ as it does in *Case 1.3* of this lemma. It also responds to $H_1(ID_i, ID_j, T_i, T_j, Z_1, Z_2, sk)$, *Send* $(\prod_{i,j}^{t}, m)$, *Reveal-Ephemeral-Key* $(\prod_{i,j}^{s})$ as it does in *Case 1.4* of lemma 5.5. It also answers the other queries as follows:

i. *Create(i).* $\mathcal{C}$ maintains an initially empty list L_C comprising of tuple $\langle ID_i, s_i, R_i, x_i, P_i\rangle$. $\mathcal{C}$ chooses three random numbers $r_i, h_i, x_i \in \mathbb{Z}_n^*$, computes $R_i = r_i \cdot P$, $P_i = x_i \cdot P$, $s_i = r_i + h_i \cdot s \bmod n$, sets $H_0(ID_i, R_i) \leftarrow h_i$ and stores $\langle ID_i, s_i, R_i, x_i, P_i\rangle$ and $\langle ID_i, R_i, h_i\rangle$ in L_C and L_{H_0} separately.

As the adversary $\mathcal{A}_{II}$ mounts the forging attack, if $\mathcal{A}_{II}$ succeeds, it must have queried oracle H_1 on the form $Z_1 = (t_I + s_I + x_I) \cdot (T_J + U)$ and $Z_2 = t_I \cdot T_J$, where $T_I = U$ is the outgoing message of the test-session given by the simulator and and $T_J = V$ is the incoming message from the adversary $\mathcal{A}_{II}$. To solve $GDH(U, V)$, for all entries in L_{H_1}, $\mathcal{C}$ randomly chooses one entry with the probability $\frac{1}{n_2}$ and returns Z_2 as the solution to $GDH(U, V)$. The advantage of $\mathcal{C}$ solving the *GDH* problem is such that $Adv_{\mathcal{C}}^{GDH}(k) \geq \frac{1}{n_0 n_1^2 n_2} Adv_{\mathcal{A}_{II}}(k)$.
Then $Adv_{\mathcal{C}}^{GDH}(k)$ is non-negligible since we assume that $Adv_{\mathcal{A}_{II}}(k)$ is non-negligible. This contradicts the *GDH* assumption.

Case 2. *No honest party owns a session matching with the test-session.*

According to definition of freshness, the following two cases should be considered.

Case 2.1. At some point, the secret-value owned by the party I has been revealed by the adversary $\mathcal{A}_{II}$. Hence, according to the freshness definition, $\mathcal{A}_{II}$ is not permitted to reveal the ephemeral-private-key of the test-session. This case is similar to *Case 1.1* of this lemma and can be solved using the same method as in the above section.

Case 2.2. The secret-value owned by the party I has never been revealed by the adversary $\mathcal{A}_{II}$. Hence, according to the freshness definition, $\mathcal{A}_{II}$ may reveal party I's ephemeral-private-key in the test-session. This case is similar to *Case 1.2* of this lemma and can be solved using the same method as in the above section.

If the adversary $\mathcal{A}_{II}$ succeeds with non-negligible probability in any of the cases above, we can also solve the *GDH* problem with non-negligible probability, which contradicts the assumed security of the *GDH* problem. So, we can conclude that the security of the proposed protocol is based on the *GDH* problem.

□

5.7.3 Comparative Analysis

Recently two security models have been followed by the researchers: modified Bellare-Rogaway (mBR) [60] and extended Canetti-Krawczyk (eCK) [63]. eCK is being considered as the strongest security model in the literature. Lipold et al. [16] presented the strengthened version of Swanson and Jao's security model [64]. Both the models were based on eCK model. The comparison has been carried out with eight recent pairing free CTAKA protocols (see Table 5.1): Geng and Zhang [95] (denoted as GZ), Hou and Xu [96] (denoted as HX), Yang and Tan [97] (denoted as YT), He et al. [98] (denoted as HH), He et al. [101] (denoted as HZ), He et al. [102] (denoted as HP), Mohamed et al. [104] (denoted as MH) and Kim et al. [15] (denoted as KK).

Table 5.1: Comparative Analysis of CTAKA Protocols

Scheme	**Year**	**Comm. Cost**	**Computation Cost**	**Security Model**	**Hardness**	**Cryptanalyze**
GZ [95]	2009	2	$7T_{mul} + 2T_h$	eCK	GDH	[97]
HX [96]	2009	2	$6T_{mul} + 2T_h$	mBR	CDH	[97]
YT [97]	2011	2	$9T_{mul} + 2T_h$	eCK	GDH	–
HH [98]	2011	3	$5T_{mul} + 3T_{add} + T_{inv} + 2T_h$	mBR	DCDH	[99]
HZ [101]	2011	2	$5T_{mul} + 4T_{add} + 2T_h$	mBR	CDH	–
HP [102]	2012	2	$5T_{mul} + 3T_{add} + 2T_h$	eCK	GDH	[103]
MH [104]	2012	1	$4T_{mul} + 2T_{add} + 1T_h$	eCK	GDH	–
KK [15]	2013	2	$4T_{mul} + 3T_{add} + 2T_h$	eCK	GDH	–
NI-CTAKA	–	2	$3T_{mul} + 3T_{add} + 2T_h$	eCK	GDH	–

The factors used to evaluate the performance of the proposed protocol are computation overhead, communication overhead, security model and hardness problem. The computation overhead includes the number of complex cryptographic operations such as scalar point multiplication, point addition, modular inversion and one way hash function while the communication overhead includes total number of messages exchanged in each protocol run. Table 5.1 also provides the detail of protocols which have already been proved insecure against type I and type II attacks. GZ, HX, HH and HP schemes are proved insecure by [97, 99, 103]. T_{mul} is the execution time for a scalar point multiplication, T_{add} is the execution time for a point addition operation, T_{inv} is the execution time for a modular inversion operation and T_h is the execution time for a one-way hash function. It is extremely enviable for a security protocol to have low computational overhead on resource constrained sensor nodes. The computation overhead on a sensor node for NI-CTAKA protocol is three point multiplications and three point additions to compute session-key, which is fairly very low as compared to other schemes.

5.7.4 Implementation and Performance Analysis

In order to evaluate the performance of the proposed NI-CTAKA protocol for WSN, a couple of parameters should be carefully focused. The basic parameters used to evaluate are running time and energy consumption. Table 5.2 shows the evaluation results of the NI-CTAKA protocol for WSN.

For the implementation of NI-CTAKA for WSN, we assume that the system parameter *param* is generated by the base station and is stored at each sensor mote before deployment. The base station also performs the generation of partial-private-key of each sensor mote. Although, the public-private-key-pair is generated by sensor mote and base-station (KGC), which is stored before deployment of the sensor motes. So, we do not include the computations involved in the generation of public-private-key-pair in our calculations of computation overhead. The proposed protocol has been implemented on MICAz platform [17], which is developed by Crossbow Technology. The MICAz device is built upon the 8-bit ATmega128L processor, 4KB of SRAM, 128KB of flash memory (ROM) with a clock speed of 7.3828MHz, RF transceiver complies with IEEE 802.15.4/ZigBee. We use TinyOS-2.1.2 [18] operating system for MICAz platform and the programming language used for the implementation is nesC. RELIC-toolkit [20] an efficient and flexible cryptographic library (version 0.3.3) has been used to perform operations on elliptic curves. It implements multiple-precision integer arithmetic, prime and binary field arithmetic (and preliminary ternary field arithmetic), elliptic curves over prime and binary fields (NIST curves and pairing-friendly curves), bilinear maps (tate pairing over supersingular binary curves and optimal pairings over BN curves) and related extension fields, etc. SHA-1 algorithm is used as a one-way hash function. We use curve K-163 for ECC, to meet the 80-bit security level. The running time and energy consumption is computed by using AVRORA simulator. The storage cost have been taken in terms of usage of RAM and ROM. NI-CTAKA protocol utilizes only 2.1 KB of RAM and 938 bytes of ROM excluding cryptographic library, which is easily justifiable to any sensor based application.

Table 5.2: Performance Analysis of NI-CTAKA Protocol with Other Existing Protocols

Scheme	**Running Time (s)**	**Energy Consumption (mJ)**
YT [97]	3.528	84.672
HZ [101]	1.96	47.04
MH [104]	1.568	37.632
KK [15]	1.568	37.632
NI-CTAKA	1.176	28.224

5.8 Summary

This chapter discussed the cryptanalysis of an existing protocol presented by Kim et al. [15] and its improved protocol, which is proven secure in eCK model. This chapter also discussed the proposed pairing free NI-CTAKA protocol, secure in eCK model. The protocol had been compared with existing protocols and found to be more efficient with lesser computation cost. NI-CTAKA had also been compared with existing protocols on the basis of running time and energy consumption. The NI-CTAKA protocol was implemented for WSN on MICAz platform using TinyOS-2.1.2 and RELIC-0.3.3 cryptographic library. Results showed that NI-CTAKA protocol is having less running time as well as consumes less energy as compared to the existing protocols.

Chapter 6

Signcryption Based Key Management

Signcryption [65] can lessen the cost of communication and computation to a great extent, which can process the signature and encryption together in a single logical step. This requires a cost lot lower than that required by the traditional signature followed by encryption to obtain confidentiality, integrity, authentication and non-repudiation. In 2011 Hagras et al. [105] proposed a key management scheme for heterogeneous WSN, which satisfies confidentiality, authentication, integrity and unforgetability but lacks forward secrecy. This chapter dicusses about the shortcomings of the victim scheme, which has been extricated and repaired with the help of Elliptic Curve Discrete logarithm problem (ECDLP). [1]

The chapter is organized as follows: Section 6.1 discusses the related work in this field. Section 6.2 discusses about the signcryption key management scheme presented by Hagras et al. [105]. In Section 6.2, we discuss a proposed signcryption key management scheme, which supports forward secrecy. Section 6.4 discusses about security analysis of the proposed sincryption key management scheme.

6.1 Related Work

Signcryption is a cryptographic process proposed by Zheng [65] to join the functionalities of encryption along with digital signature in a single logical step. The author further

[1] The major findings of this chapter has been published as "*An Improved Forward Secure Elliptic Curve Signcryption Key Management Scheme for Wireless Sensor Networks*," Book Chapter in International Conference on IT Convergence and Security 2012 (ICITCS12), Lecture Notes in Electrical Engineering (LNEE, Springer), Vol. 215, pp. 141-149, (Dec 5 - 7, 2012, Republic of Korea), 2013.

finds out that the signcryption costs 58% less in average computation time and 70% less in message expansion than does signature-then-encryption based on the Discrete Logarithm Problem (DLP) [65]. Numerous schemes [106–110] are proposed over the years to provide different level of security measures and communication/computational costs. Later, Zheng [106] proposed two key exchange protocols using signcryption, which are based on DLP called Direct Key Exchange Using a Nonce (DKEUN) and Direct Key Exchange Using Time-stamp (DKEUTS). But, the scheme fails to provide forward secrecy of message confidentiality, when the sender's private key is disclosed [110].

Moreover, Zheng and Imai [111] proposed a signcryption scheme based on Elliptic Curve Discrete Logarithm Problem (ECDLP), which saves 58% in computational cost and 40% in communication overhead as compared with signature-then-encryption on elliptic curves but it lacks forward secrecy, public verifiability and encrypted message authentication. The schemes [106, 111] cannot be used in applications, where third party validation is necessary using a public key as performed in signature schemes. The solution is provided by Zheng [107], which introduces an independent judge. But when dispute occurs the judge cannot verify the signature, as he is not having the private key of the recipient. To overcome the above problem Bao and Deng [108] enhanced Zheng's scheme [107] in such a way that verification of a signature does not need the recipient's private key but the scheme was not as efficient as Zheng's scheme. Gamage et al. [109] also modifies Zheng's [65] signcryption scheme in such a way that anyone can verify the signature of ciphertext to protect confidentiality of message.

Jung et al. [110] proposed a signcryption scheme based on DLP with forward secrecy. Whereas, Hwang et al. [112] proposed a signcryption scheme based on ECDLP, which provides forward secrecy for message confidentiality and public verification along with other basic security notions. When dispute occurs, the judge can verify sender's signature without the sender's private key. Later, Toorani and Beheshti [113] proved that the scheme presented by Hwang et al. [112] is insecure. It does not provide all the desired and essential security attributes of Signcryption. Kim and Youm [114] proposed two protocols named Secure Authenticated Key Exchange (SAKE) protocol and Elliptic Curve-Secure Authenticated Key Exchange (EC-SAKE) protocol. The protocols are efficient in terms of computation complexity and communication performance as compared to DKEUN, DKEUTS and EC-DKEUN, EC-DKEUTS respectively. Zhou [115] proposed a scheme based on ECDLP with public verifiability through a trusted third party without disclosing private key. Toorani and Beheshti [116] and Elsayed and Hassan [117] proposed the schemes based on ECDLP, which provides forward secrecy for message confidentiality and public verification. Hamed and El-Khamy [118] proposed a scheme based on ECDLP for cluster based WSN. Later, Hagras et al. [105] proposed a scheme, which is efficient [118] in terms of total number of operations, key

storage, energy consumption and communication overhead as 75%, 96%, 23.79 mJ and 40% respectively and satisfies all the security requirements, but lacks to provide forward secrecy.

6.2 Hagras et al.'s Signcryption Key Management Scheme

The Hagras et al. [105] scheme works in three phases in the following manner.

Phase-I **Generation of Public/Private Key:** This phase is responsible for creating public/private key pair for Base-Node (B), Cluster-Heads (H) and Cluster-Nodes (N). It creates the BH symmetric keys, which is used for secure communication between the cluster-heads among each other and with the base-node. Also, it creates the HN symmetric keys, which is used for secure communication between the cluster-nodes among each other in the cluster and with the corresponding cluster-head.

(a) *Base-Node generates public/private key pair.*

- s_B: Base-node (B) choose its private-key uniformly at a random from $[1 \cdots q-1]$
- P_B: Base-node (B) computes the public-key, $P_B = s_B \cdot P$

(b) *Cluster-Head generates public/private key pair.*

- s_{H_i}: Each cluster-head (H_i) choose its private-key uniformly at a random from $[1 \cdots q-1]$, where $i \in n_1$; n_1: the number of cluster-heads
- P_{H_i}: Each cluster-head (H_i) computes its public-key, $P_{H_i} = s_{H_i} \cdot P$

All cluster-heads (H_i) send their public-key P_{H_i} to the Base-node.

(c) *Cluster-Node generates public/private key pair.*

- s_{N_j}: Each cluster-node (N_j) choose its private-key uniformly at a random from $[1 \cdots q-1]$, where $j \in n_2$; n_2: the number of cluster-nodes
- P_{N_j}: Each cluster-node (N_j) computes its public-key, $P_{N_j} = s_{N_j} \cdot P$

All cluster-nodes (N_j) send their public-key P_{N_j} to corresponding cluster-head (H_i).

(d) *Symmetric key Generation for CHs and CNs.*

- Base-node (B) creates the symmetric key K_{BH_i}, which is used for secure communication between the base-node and the cluster-heads, and among the cluster-heads.

- Cluster-heads (H_i) create the symmetric key ($K_{HN_{i,j}}$), which is used for secure communication between the cluster-head and their corresponding cluster-nodes, and among the cluster-nodes within the cluster-head.

Phase-II **Base-Node Cluster-Head Key Establishment:** This phase is responsible for the base-node cluster-head key establishment. The base-node generates the shared symmetric key for each cluster-head by using their public-keys; signcrypts the symmetric key K_{BH_i} generated in the first phase and send to the cluster-heads, which later unsigncrypts by the cluster-head as follows:

- **Signcryption:** The base-node signcrypts the symmetric key K_{BH_i} using its private key and sends the ciphertext (C_i, R_i, S_i) to each cluster-head as follows:
  - The base-node chooses $r_i \in \{1, \cdots, q-1\}$
  - $(k_{i,1}, k_{i,2}) = hash(r_i \cdot P_{H_i})$
  - $C_i = E_{k_{i,1}}(K_{BH_i})$
  - $R_i = KH_{K_{i,2}}(K_{BH_i}, P_{H_i}, bind - inf.)$
  - $S_i = r_i/(R_i + s_B)$
- **Unsigncryption:** The base-node sends the cipher text (C_i, R_i, S_i) to each cluster-head and each cluster-head unsigncrypts the symmetric key as follows:
  - $U_i = S_i \cdot s_{H_i}$
  - $(k_{i,1}, k_{i,2}) = hash(U_i \cdot (R_i \cdot P + P_B))$
  - $K_{BH_i} = D_{k_{i,1}}(C_i)$
  - Accept K_{BH_i}, if and only if $R_i = KH_{K_{i,2}}(K_{BH_i}, P_{H_i}, bind - inf.)$

Phase-III **Cluster-Head Cluster-Node Key Establishment:** This phase is responsible for the cluster-head cluster-node key establishment. Each cluster-head generates the shared symmetric key for each cluster-node in the corresponding cluster by using their public-keys; signcrypts the symmetric key ($K_{HN_{i,j}}$) generated in the first phase and send to the cluster-nodes, which later unsigncrypts by the cluster-node as follows:

- **Signcryption:** The cluster-head signcrypts the symmetric key $K_{HN_{i,j}}$ using its private key and sends the ciphertext (C_j, R_j, S_j) to all the cluster-nodes in the corresponding cluster as follows:
  - The cluster-head (H_i) chooses $r_j \in \{1, \cdots, q-1\}$
  - $(k_{j,1}, k_{j,2}) = hash(r_j \cdot P_{N_j})$
  - $C_j = E_{k_{j,1}}(K_{HN_{i,j}})$

- $R_j = KH_{K_{j,2}}(K_{HN_{i,j}}, P_{N_j}, bind - inf.)$
- $S_j = r_j/(R_j + s_{H_i})$

- **Unsigncryption:** The cluster-head sends the cipher text (C_j, R_j, S_j) to each cluster-node and each cluster-node unsigncrypts the symmetric key as follows:
  - $U_j = S_j \cdot s_{N_j}$
  - $(k_{j,1}, k_{j,2}) = hash(U_j \cdot (R_j \cdot P + P_{H_i}))$
  - $K_{HN_{i,j}} = D_{k_{j,1}}(C_j)$
  - Accept $K_{HN_{i,j}}$, if and only if $R_j = KH_{K_{j,2}}(K_{HN_{i,j}}, P_{N_j}, bind - inf.)$

6.3 An Improved Forward Secure Elliptic Curve Signcryption Key Management Scheme

This section discusses the proposed forward secure elliptic curve signcryption scheme. The proposed scheme satisfies forward secrecy along with the basic security requirements. The forward secrecy of the proposed scheme will be compromised only if the attacker can solve the ECDLP that is computationally infeasible with the selected domain parameters. The proposed scheme has secure key exchange, less storage requirement, scalability and low complexity. The proposed scheme works in three phases as follows:

Phase-I **Generation of Public/Private Key:** This phase is responsible for creating public/private key pair for Base-Node (B), Cluster-Heads (H) and Cluster-Nodes (N). It creates the BH symmetric keys, which is used for secure communication between the cluster-heads among each other and with the base-node. Also, it creates the HN symmetric keys, which is used for secure communication between the cluster-nodes among each other in the cluster and with the corresponding cluster-head as shown in figure 6.1.

A.1 *Base-Node generates public/private key pair.*

- s_B: Base-node (B) choose its private-key uniformly at a random from $[1 \cdots q-1]$
- P_B: Base-node (B) computes the public-key, $P_B = s_B \cdot P$

A.2 *Cluster-Head generates public/private key pair.*

- s_{H_i}: Each cluster-head (H_i) choose its private-key uniformly at a random from $[1 \cdots q-1]$, where $i \in n_1$; n_1: the number of cluster-heads
- P_{H_i}: Each cluster-head (H_i) computes its public-key, $P_{H_i} = s_{H_i}\dot{P}$

A.3 *Cluster-Node generates public/private key pair.*

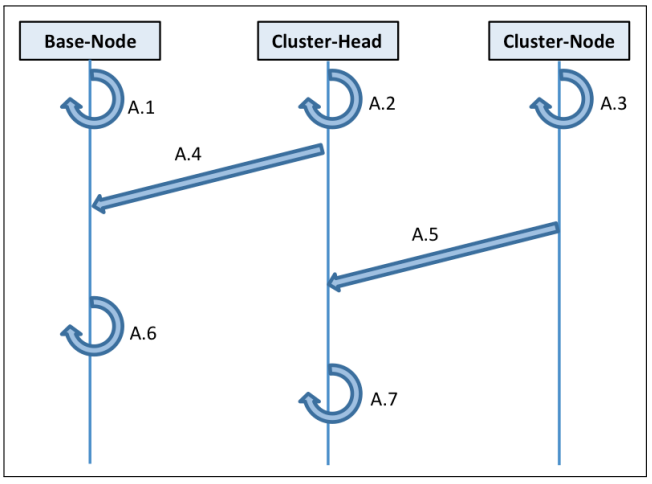

FIGURE 6.1: Generation of public/private keys (Phase-I)

- s_{N_j}: Each cluster-node (N_j) choose its private-key uniformly at a random from $[1 \cdots q-1]$, where $j \in n_2$; n_2: the number of cluster-nodes
- P_{N_j}: Each cluster-node (N_j) computes its public-key, $P_{N_j} = s_{N_j} \cdot P$

A.4 *Cluster-Head sends its public key to Base-Node.*

All cluster-heads (H_i) send their public-key P_{H_i} to the Base-node.

A.5 *Cluster-Node sends its public key to Cluster-Head.*

All cluster-nodes (N_j) send their public-key P_{N_j} to corresponding cluster-head (H_i).

A.6 *Base-Node creates BH symmetric key.*

Base-node (B) creates the symmetric key K_{BH_i}, which is used for secure communication between the base-node and the cluster-heads, and among the cluster-heads.

A.7 *Cluster-Heads creates HN symmetric key.*

Cluster-heads (H_i) create the symmetric key (K_{HN_i}), which is used for secure communication between the cluster-head and their corresponding cluster-nodes, and among the cluster-nodes within the cluster-head.

Phase-II **Base-Node Cluster-Head Key Establishment:** This phase is responsible for the base-node cluster-head key establishment as shown in figure 6.2. The base-node generates the shared symmetric key for each cluster-head by using their public-keys; signcrypts the symmetric key K_{BH_i} generated in the first phase and send to the cluster-heads, which later unsigncrypts by the cluster-head as follows:

B.1 *Base-Node generate a shared symmetric key for each cluster-head by using their public-key.*

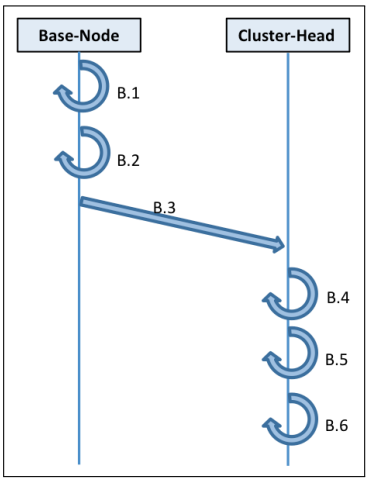

FIGURE 6.2: Key Establishment of base-node cluster- head (Phase-II)

B.2 *Base-Node encrypt and signature the BH symmetric key using shared symmetric key.*

B.3 *Base-Node sends the encrypted BH symmetric key and its encrypted signature.*

B.4 *Cluster-Head generates a shared symmetric key using private key of Cluster-Head and received signature.*

B.5 *Cluster-Head decrypts the BH symmetric key and its signature using shared symmetric key.*

B.6 *Cluster-Head verifies BH symmetric key signature.*

The algorithm works as follows:

- **Signcryption:** The base-node signcrypts the symmetric key K_{BH_i} using its private key and sends the ciphertext (C_i, T_i, S_i) to each cluster-head as follows:
  - The base-node chooses $r_i \in \{1, \cdots, q-1\}$
  - $(k_{i,1}, k_{i,2}) = hash(r_i \cdot P_{H_i})$
  - $C_i = E_{k_{i,1}}(K_{BH_i})$
  - $R_i = KH_{K_{i,2}}(K_{BH_i} \parallel P_{H_i} \parallel ID_A \parallel ID_B)$
  - $S_i = r_i/(R_i + s_B)$
  - $T_i = R_i \cdot P$
- **Unsigncryption:** The base-node sends the cipher text (C_i, T_i, S_i) to each cluster-head and each cluster-head unsigncrypts the symmetric key as follows:
  - $U_i = S_i \cdot s_{H_i}$
  - $(k_{i,1}, k_{i,2}) = hash(U_i \cdot (T_i + P_B))$

- $K_{BH_i} = D_{k_{i,1}}(C_i)$
- Accept K_{BH_i}, if and only if $T_i = U_i \cdot P$

Phase-III **Cluster-Head Cluster-Node Key Establishment:** This phase is responsible for the cluster-head cluster-node key establishment as shown in figure 6.3. Each cluster-head generates the shared symmetric key for each cluster-node in the corresponding cluster by using their public-keys; signcrypts the symmetric key ($K_{HN_{ij}}$) generated in the first phase and send to the cluster-nodes, which later unsigncrypts by the cluster-node as follows:

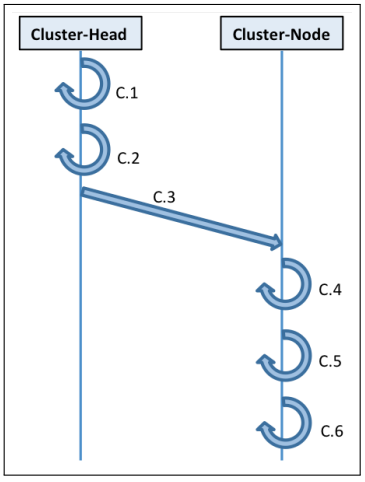

FIGURE 6.3: Key establishment of cluster-head cluster-node (Phase-III)

C.1 *Cluster-Head generates a shared symmetric key using public key of cluster-node*

C.2 *Cluster-Head encrypts and signature the HN symmetric key using shared symmetric key*

C.3 *Cluster-Head sends the encrypted HN symmetric key and its encrypted signature*

C.4 *Cluster-Node generates a shared symmetric key using private key of cluster-node and received signature*

C.5 *Cluster-Node decrypts the HN symmetric key and its signature using shared symmetric key*

C.6 *Cluster-Node verify HN symmetric key signature*

The algorithm works as follows:

- **Signcryption:** The cluster-head signcrypts the symmetric key $K_{HN_{i,j}}$ using its private key and sends the ciphertext (C_j, T_j, S_j) to all the cluster-nodes in the corresponding cluster as follows:

- The cluster-head (H_i) chooses $r_j \in \{1, \cdots, q-1\}$
- $(k_{j,1}, k_{j,2}) = hash(r_j \cdot P_{N_j})$
- $C_j = E_{k_{j,1}}(K_{HN_{i,j}})$
- $R_j = KH_{K_{j,2}}(K_{HN_{i,j}} \parallel P_{N_j} \parallel ID_A \parallel ID_B)$
- $S_j = r_j/(R_j + s_{H_i})$
- $T_j = R_j \cdot P$

- **Unsigncryption:** The cluster-head sends the cipher text (C_j, T_j, S_j) to each cluster-node and each cluster-node unsigncrypts the symmetric key as follows:
  - $U_j = S_j \cdot s_{N_j}$
  - $(k_{j,1}, k_{j,2}) = hash(U_j \cdot (T_j + P_{H_i}))$
  - $K_{HN_{i,j}} = D_{k_{j,1}}(C_j)$
  - Accept $K_{HN_{i,j}}$, if and only if $T_j = U_j \cdot P$

6.4 Security Analysis

The concept of forward secrecy proves its importance in WSN. If a sensor node (mote) runs out of energy and gets replaced with a new node, the new node should not be able to unsigncrypt the previous signcrypted messages. The flaw of the existing scheme has been found and repaired. The proposed key management using public key elliptic curves signcryption for WSN provides all security functions like, key confidentiality, authentication, integrity and unforgeability but lacks in forward secrecy. The security proof of all the required parameters can be directly taken from the parent scheme. The security of improved scheme is based upon the ECDLP, which is computationally infeasible to solve for the specified parameters. The new scheme does not require the extra storage for sensor nodes.

Table 6.1 depicts the comparison of different schemes based on cost of computation. The result shows that the proposed scheme is having minimum compution cost for both signcryption and unsigncryption and provides forward secrecy along with all security measures. Although, the elliptic curve point multiplication of Zheng [107], Zheng and Imai [111], Bao and Deng [108], Gamage et al. [109], Jung et al. [110], Hwang et al. [112], Hagras et al. [105] are less than the proposed scheme but they are not satisfies all the security measures.

TABLE 6.1: Comparative Analysis of Signcryption Key Management Schemes

Scheme	**ECPM**	**ECPA**	**EXP**	**DIV**	**MUL**	**ADD**	**HASH**
Zheng [107]	0,0	0,0	1,2	1,0	0,2	1,0	2,2
Zheng and Imai [111]	1,2	0,1	0,0	1,0	1,2	1,0	2,2
Bao and Deng [108]	0,0	0,0	2,3	1,0	0,1	1,0	3,3
Gamage et al. [109]	0,0	0,0	2,3	1,0	0,1	1,0	2,2
Jung et al. [110]	0,0	0,0	2,3	1,0	0,1	1,0	2,2
Hwang et al. [112]	2,3	0,1	0,0	0,0	1,0	1,0	1,1
Toorani and Beheshti [116]	2,4	0,2	0,0	0,0	2,0	2,0	2,2
Elsayed and Hassan [117]	3,4	1,2	0,0	1,0	0,0	0,0	3,3
Hagras et al. [105]	1,3	1,1	0,0	1,0	0,0	0,0	2,2
Proposed Scheme	2,3	1,1	0,0	1,0	0,0	0,0	2,1

ECPM the number of elliptic curve multiplication operation
ECPA - the number of elliptic curve addition operation
EXP - the number of modular exponentiation operation
DIV - the number of modular division operation
MUL - the number of modular multiplication operation
ADD - the number of modular addition operation
HASH - the number of one-way or keyed one-way hash function operation

6.5 Summary

For WSN, forward secrecy is a vital security requirement. In this chapter, an elliptic curve based key management scheme has been improved in terms of forward secrecy. The proposed scheme satisfies all the basic security requirements of key management schemes. The security of the proposed scheme can be proved with the help of the scheme presented by Hagras et al. [105] The forward secrecy of the proposed scheme will be compromised only if the attacker can solve the ECDLP, which is computationally infeasible with the selected domain parameters.

Chapter 7

Conclusion and Future Scope

This chapter consists of summary of our work and also includes some recommendations for future work. This thesis contributes to the active research area of authenticated key agreement in WSN. The major focus of this thesis is to apply key management schemes in WSN. The demonstration of the proposed work has been captured on a typical sensor mote, MICAz. Research objectives have been satisfied with the proposal of four key agreement schemes. To recapitulate the contributions made in the thesis are as follows:

The first contribution is towards the development of a Pairing-Free Identity-based two-Party Authenticated Key Agreement (PF-ID-2PAKA) protocol for WSN (discussed in Chapter 4). The motivation behind this protocol is to make WSN more energy efficient. PF-ID-2PAKA protocol takes only three point multiplications and three point additions in the key agreement phase. Also, PF-ID-2PAKA protocol is based on Gap Diffie-Hellman problem and found to be secure in extended Canetti-Crawzyk (eCK) model presented by Liang et al. [14] for identity-based setting. Also, PF-ID-2PAKA protocol has been compared with the existing identity based key agreement protocols and found to be efficient in terms of computation cost. PF-ID-2PAKA protocol has been implemented for WSN environment on MICAz mote for the viability of its existence. The result shows that PF-ID-2PAKA protocol has less running time and energy consumption compared to two latest existing protocols.

Certificateless cryptography is key escrow secure version of ID-based cryptography. The second contribution is towards the cryptanalysis of a certificateless authenticated key agreement protocol presented by Kim et al. [15] against Key-Compromise Impersonation (K-CI) attack (discussed in Chapter 5). Further, we propose an improvement in Kim et al.'s protocol. The improved protocol is based on Gap Diffie-Hellman problem and secure in extended Canetti-Crawzyk (eCK) model presented by Lippold et al. [16] for certificateless setting.

The third contribution is the development of a Non-Interactive Certificateless Two-party Authenticated Key Agreement (NI-CTAKA) protocol for WSN (discussed in Chapter 5). NI-CTAKA protocol does not use pairing operation. NI-CTAKA protocol takes only three point multiplications and three point additions in the key agreement phase. Also, NI-CTAKA protocol is based on Gap Diffie-Hellman problem and found to be secure in extended Canetti-Crawzyk (eCK) model presented by Lippold et al. [16] for certificateless setting. Also, NI-CTAKA protocol has been compared with the existing identity based key agreement protocols and found to be efficient in terms of computation cost. NI-CTAKA protocol has been implemented for WSN environment on MICAz mote for the viability of its existence. The result shows that NI-CTAKA protocol has less running time and energy consumption compared to two latest existing protocols.

The last contribution is the development of an improved forward secure key management scheme based on elliptic curve signcryption scheme for WSN (discussed in Chapter 6). Forward secrecy is the most important concern in WSN as joining of new motes in the network is a common affair. The proposed protocol has been compared with the existing protocols and found to be efficient in terms of computation cost.

The computation cost has been measured by counting the number of basic operations like pairing operation, point scalar multiplication on a group, scalar point addition on a group, multiplication on a field group, addition on a field group, XOR operation, division operation, inversion, hash operation, which can be helpful in deciding the efficiency of the algorithm. The implementation of the proposed protocols has been performed on MICAz mote (8-bit AVR microcontroller), TinyOS-2.1.1 (Operating System for mote), high precision cryptographic library-RELIC-0.3.3 and AVRORA-1.7.106 simulator for computing running time and energy consumption.

Future Work: The work discussed in this thesis opens new avenues for future research. Some interesting research problems in the area of authenticated key agreement in WSN still requires further investigation.

- The proposed authenticated key agreement protocols can be implemented to some existing real world test bed and this would check the performance and robustness of the solutions developed.
- In this thesis work, energy consumption has been calculated using direct formula with voltage level and current for that particular mote and running time of the algorithm. But, the actual energy consumption is quite challenging to compute because voltage and current may vary in active and sleep state.

- In large scale sensor networks, where dynamicity is very high, the real time implementation of all aspects of key agreement protocol especially the scalability, key refresh and key revocation is difficult to deliver.
- The main focus of this thesis is to reduce the computational overhead. Communication overhead is also another important aspect to focus.

Bibliography

[1] Akyildiz I. F., Su W., Sankarasubramaniam Y., and Cayirci E. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, March 2002. ISSN 1389-1286. doi: 10.1016/S1389-1286(01)00302-4. URL `http://dx.doi.org/10.1016/S1389-1286(01)00302-4`.

[2] Chen Y. and Zhao Q. On the lifetime of wireless sensor networks. *Communications Letters, IEEE*, 9(11):976–978, 2005. ISSN 1089-7798. doi: 10.1109/LCOMM.2005.11010.

[3] Olariu S. and Xu Q. Information assurance in wireless sensor networks. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing*, pages 5 pp.–. IEEE Computer Society, April 2005. ISBN 0-7695-2312-9. doi: 10.1109/IPDPS.2005.257.

[4] Chen X., Makki K., Yen K., and Pissinou N. Sensor network security: A survey. *IEEE Communications Surveys Tutorials*, 11(2):52–73, 2009. ISSN 1553-877X. doi: 10.1109/SURV.2009.090205.

[5] Walters J., Liang Z., Shi W., and Chaudhary V. *Security in Distributed, Grid, and Pervasive Computing*, chapter 17 Wireless Sensor Network security: A survey, pages 1–51. CRC Press, 2007.

[6] Zhang X., Heys H., and Cheng L. Energy efficiency of symmetric key cryptographic algorithms in wireless sensor networks. In *25th Biennial Symposium on Communications*, QBSC '10, pages 168–172, 2010. doi: 10.1109/BSC.2010.5472979.

[7] Wander A., Gura N., Eberle H., Gupta V., and Shantz S. Energy analysis of public-key cryptography for wireless sensor networks. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications*, PERCOM '05, pages 324–328, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2299-8. doi: 10.1109/PERCOM.2005.18. URL `http://dx.doi.org/10.1109/PERCOM.2005.18`.

[8] Shamir A. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin Heidelberg, 1985. ISBN 978-3-540-15658-1. doi: 10.1007/3-540-39568-7_5. URL http://dx.doi.org/10.1007/3-540-39568-7_5.

[9] Amin F., Jahangir H., and Rasifard H. Analysis of public-key cryptography for wireless sensor networks security. 2(5):403 – 408, 2008. ISSN 1307-6892. URL http://waset.org/Publications?p=17.

[10] Al-Riyami S. and Paterson K. Certificateless public key cryptography. In *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 452–473. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-20592-0. doi: 10.1007/978-3-540-40061-5_29. URL http://dx.doi.org/10.1007/978-3-540-40061-5_29.

[11] The evolution of wireless sensor networks. http://www.silabs.com/Support%20Documents/TechnicalDocs/evolution-of-wireless-sensor-networks.pdf.

[12] Du W., Deng J., Han Y., Varshney P., Katz J., and Khalili A. A pairwise key predistribution scheme for wireless sensor networks. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):228–258, 2005. ISSN 1094-9224. doi: 10.1145/1065545.1065548. URL http://doi.acm.org/10.1145/1065545.1065548.

[13] Perrig A., Stankovic J., and Wagner D. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, June 2004. ISSN 0001-0782. doi: 10.1145/990680.990707. URL http://doi.acm.org/10.1145/990680.990707.

[14] Liang N., Gongliang C., and Jianhua L. Escrowable identity-based authenticated key agreement protocol with strong security. *Computers & Mathematics with Applications*, 65(9):1339 – 1349, 2013. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/j.camwa.2012.01.041. URL http://www.sciencedirect.com/science/article/pii/S089812211200051X. Advanced Information Security.

[15] Kim Y., Kim Y., Choe Y., and Hyong O. An efficient bilinear pairing-free certificateless two-party authenticated key agreement protocol in the eck model. In *Journal of Theoretical Physics and Cryptography*, volume 3, pages 1–10. 2013. URL http://www.ijtpc.org/volume3/JTPC1451.pdf.

[16] Lippold G., Boyd C., and Gonzalez J. Strongly secure certificateless key agreement. In *Pairing-Based Cryptography Pairing 2009*, volume 5671 of *Lecture Notes in*

Computer Science, pages 206–230. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03297-4. doi: 10.1007/978-3-642-03298-1_14. URL http://dx.doi.org/10.1007/978-3-642-03298-1_14.

[17] Rev A. Mpr-mib series user manual. http://www-db.ics.uci.edu/pages/research/quasar/MPR-MIB%20Series%20User%20Manual%207430-0021-06_A.pdf, 2004.

[18] Levis P., Madden S., Polastre J., Szewczyk R., Whitehouse K., Woo A., Gay D., Hill J., Welsh M., Brewer E., and Culler D. Tinyos: An operating system for sensor networks. In *Ambient Intelligence*, pages 115–148. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-23867-6. doi: 10.1007/3-540-27139-2_7. URL http://dx.doi.org/10.1007/3-540-27139-2_7.

[19] Gay D., Levis P., Behren R., Welsh M., Brewer E., and Culler D. The nesc language: A holistic approach to networked embedded systems. *ACM SIGPLAN Notices*, 38(5):1–11, May 2003. ISSN 0362-1340. doi: 10.1145/780822.781133. URL http://doi.acm.org/10.1145/780822.781133.

[20] Aranha D. and Gouvêa C. RELIC is an Efficient LIbrary for Cryptography. http://code.google.com/p/relic-toolkit/.

[21] Titzer B, Lee D., and Palsberg J. Avrora: scalable sensor network simulation with precise timing. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, pages 477–482, 2005. doi: 10.1109/IPSN.2005.1440978.

[22] Perrig A., Szewczyk R., Tygar J., Wen V., and Culler D. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002. ISSN 1022-0038. doi: 10.1023/A:1016598314198. URL http://dx.doi.org/10.1023/A:1016598314198.

[23] Pietro R., Mancini L., Yee L., Etalle S., and Havinga P. Lkhw: a directed diffusion-based secure multicast scheme for wireless sensor networks. In *Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on*, pages 397–406, Oct 2003. doi: 10.1109/ICPPW.2003.1240395.

[24] Chan H. and Perrig A. Pike: peer intermediaries for key establishment in sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 1, pages 524–535 vol. 1, March 2005. doi: 10.1109/INFCOM.2005.1497920.

[25] Lai B., Kim S., and Verbauwhede I. Scalable session key construction protocol for wireless sensor networks. In *In IEEE Workshop on Large Scale RealTime and Embedded Systems (LARTES*, page 7, 2002.

[26] Dutertre B., Cheung S., and Levy J. Lightweight key management in wireless sensor networks by leveraging initial trust, sdl. Technical report, Tech. Rep. SRI-SDL-04-02, System Design Laboratory, 2004.

[27] Chan H., Perrig A., and Song D. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, SP '03, pages 197–. IEEE Computer Society, 2003. ISBN 0-7695-1940-7. URL `http://dl.acm.org/citation.cfm?id=829515.830566`.

[28] Eschenauer L. and Gligor V. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, pages 41–47. ACM, 2002. ISBN 1-58113-612-9. doi: 10.1145/586110.586117. URL `http://doi.acm.org/10.1145/586110.586117`.

[29] Liu D. and Ning P. Improving key predistribution with deployment knowledge in static sensor networks. *ACM Trans. Sen. Netw.*, 1(2):204–239, 2005. ISSN 1550-4859. doi: 10.1145/1105688.1105691. URL `http://doi.acm.org/10.1145/1105688.1105691`.

[30] Du W., Deng J., Han Y., Chen S., and Varshney P. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 1, pages –597, March 2004. doi: 10.1109/INFCOM.2004.1354530.

[31] Liu D. and Ning P. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 52–61. ACM, 2003. ISBN 1-58113-738-9. doi: 10.1145/948109.948119. URL `http://doi.acm.org/10.1145/948109.948119`.

[32] Blundo C., Alfredo S., Herzberg A., Kutten S., Vaccaro U., and Yung M. Perfectly-secure key distribution for dynamic conferences. In *Advances in Cryptology CRYPTO 92*, volume 740 of *Lecture Notes in Computer Science*, pages 471–486. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-57340-1. doi: 10.1007/3-540-48071-4_33. URL `http://dx.doi.org/10.1007/3-540-48071-4_33`.

[33] Liu D., Ning P., and Li R. Establishing pairwise keys in distributed sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(1):41–77, 2005. ISSN 1094-9224. doi: 10.1145/1053283.1053287. URL `http://doi.acm.org/10.1145/1053283.1053287`.

[34] Liu D. and Ning P. Location-based pairwise key establishments for static sensor networks. In *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '03, pages 72–82. ACM, 2003. ISBN 1-58113-783-4. doi: 10.1145/986858.986869. URL http://doi.acm.org/10.1145/986858.986869.

[35] Zhang W., Tran M., Zhu S., and Cao G. A random perturbation-based scheme for pairwise key establishment in sensor networks. In *Proceedings of the 8th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '07, pages 90–99. ACM, 2007. ISBN 978-1-59593-684-4. doi: 10.1145/1288107.1288120. URL http://doi.acm.org/10.1145/1288107.1288120.

[36] Blom R. An optimal class of symmetric key generation systems. In *Advances in Cryptology*, volume 209 of *Lecture Notes in Computer Science*, pages 335–338. Springer Berlin Heidelberg, 1985. ISBN 978-3-540-16076-2. doi: 10.1007/3-540-39757-4_22. URL http://dx.doi.org/10.1007/3-540-39757-4_22.

[37] Huang D., Mehta M., Medhi D., and Harn L. Location-aware key management scheme for wireless sensor networks. In *Proceedings of the 2Nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '04, pages 29–42. ACM, 2004. ISBN 1-58113-972-1. doi: 10.1145/1029102.1029110. URL http://doi.acm.org/10.1145/1029102.1029110.

[38] Yu Z. and Yong G. A robust group-based key management scheme for wireless sensor networks. In *Wireless Communications and Networking Conference, 2005 IEEE*, volume 4, pages 1915–1920 Vol. 4, March 2005. doi: 10.1109/WCNC.2005.1424812.

[39] Yu C., Lu C., and Kuo S. A simple non-interactive pairwise key establishment scheme in sensor networks. In *Sensor, Mesh and Ad Hoc Communications and Networks, 2009. SECON '09. 6th Annual IEEE Communications Society Conference on*, pages 1–9, June 2009. doi: 10.1109/SAHCN.2009.5168906.

[40] Lee J. and Stinson D. Deterministic key predistribution schemes for distributed sensor networks. In *Selected Areas in Cryptography*, volume 3357 of *Lecture Notes in Computer Science*, pages 294–307. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24327-4. doi: 10.1007/978-3-540-30564-4_21. URL http://dx.doi.org/10.1007/978-3-540-30564-4_21.

[41] Zhu S., Setia S., and Jajodia S. Leap: Efficient security mechanisms for large-scale distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security*, CCS '03, pages 62–72, 2003. ISBN 1-58113-738-9. doi: 10.1145/948109.948120. URL http://doi.acm.org/10.1145/948109.948120.

[42] Jang J., Kwon T., and Song J. A time-based key management protocol for wireless sensor networks. In *Information Security Practice and Experience*, volume 4464 of *Lecture Notes in Computer Science*, pages 314–328. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72159-8. doi: 10.1007/978-3-540-72163-5_24. URL `http://dx.doi.org/10.1007/978-3-540-72163-5_24`.

[43] Camtepe S. and Yener B. Combinatorial design of key distribution mechanisms for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 15(2):346–358, April 2007. ISSN 1063-6692. doi: 10.1109/TNET.2007.892879.

[44] Younis M., Ghumman K., and Eltoweissy M. Location-aware combinatorial key management scheme for clustered sensor networks. *Parallel and Distributed Systems, IEEE Transactions on*, 17(8):865–882, Aug 2006. ISSN 1045-9219. doi: 10.1109/TPDS.2006.106.

[45] Eltoweissy M., Moharrum M., and Mukkamala R. Dynamic key management in sensor networks. *Communications Magazine, IEEE*, 44(4):122–130, April 2006. ISSN 0163-6804. doi: 10.1109/MCOM.2006.1632659.

[46] Diffie W. and Hellman M. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, Nov 1976. ISSN 0018-9448. doi: 10.1109/TIT.1976.1055638.

[47] Malan D., Welsh M., and Smith M. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 71–80, Oct 2004. doi: 10.1109/SAHCN.2004.1381904.

[48] Gura N., Patel A., Wander A., Eberle H., and Shantz S. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-22666-6. doi: 10.1007/978-3-540-28632-5_9. URL `http://dx.doi.org/10.1007/978-3-540-28632-5_9`.

[49] Liu A. and Ning P. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN '08. International Conference on*, pages 245–256, April 2008. doi: 10.1109/IPSN.2008.47.

[50] Koblitz N. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177): 203–209, 1987. ISSN 1088-6842. doi: 10.2307/2007884. URL http://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/.

[51] Miller V. Use of elliptic curves in cryptography. In *Advances in Cryptology CRYPTO 85 Proceedings*, volume 218 of *Lecture Notes in Computer Science*, pages 417–426. Springer Berlin Heidelberg, 1986. ISBN 978-3-540-16463-0. doi: 10.1007/3-540-39799-X_31. URL http://dx.doi.org/10.1007/3-540-39799-X_31.

[52] Menezes A., Okamoto T., and Vanstone S. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5): 1639–1646, 1993. ISSN 0018-9448. doi: 10.1109/18.259647.

[53] Sakai R., Ohgishi K., and Kasahara M. Cryptosystems based on pairing. In *Proceedings of the 1st Symposium on Cryptography and Information Security*, SECON '04, pages 71–80, 2000. doi: 10.1109/SAHCN.2004.1381904.

[54] Joux A. A one round protocol for tripartite diffiehellman. *Journal of Cryptology*, 17(4):263–276, 2004. ISSN 0933-2790. doi: 10.1007/s00145-004-0312-y. URL http://dx.doi.org/10.1007/s00145-004-0312-y.

[55] Boneh D. and Franklin M. Identity-based encryption from the weil pairing. In *Advances in Cryptology CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42456-7. doi: 10.1007/3-540-44647-8_13. URL http://dx.doi.org/10.1007/3-540-44647-8_13.

[56] Boyd C. Towards extensional goals in authentication protocols. In *In Proceedings of the 1997 DIMACS Workshop on Design and Formal Verification of Security Protocols*, pages 7–9, 1997.

[57] Goldwasser S. and Micali S. Probabilistic encryption. *Journal of Computer and System Sciences*, 28(2):270 – 299, 1984. ISSN 0022-0000. doi: http://dx.doi.org/10.1016/0022-0000(84)90070-9. URL http://www.sciencedirect.com/science/article/pii/0022000084900709.

[58] Bellare M. and Rogaway P. Entity authentication and key distribution. In *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-57766-9. doi: 10.1007/3-540-48329-2_21. URL http://dx.doi.org/10.1007/3-540-48329-2_21.

[59] Blake S., Johnson D., and Menezes A. Key agreement protocols and their security analysis. In *Cryptography and Coding*, volume 1355 of *Lecture Notes in Computer*

Science, pages 30–45. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-63927-5. doi: 10.1007/BFb0024447. URL http://dx.doi.org/10.1007/BFb0024447.

[60] Kudla C. and Paterson K. Modular security proofs for key agreement protocols. In *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 549–565. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30684-9. doi: 10.1007/11593447_30. URL http://dx.doi.org/10.1007/11593447_30.

[61] Bellare M., Pointcheval D., and Rogaway P. Authenticated key exchange secure against dictionary attacks. In *Advances in Cryptology EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67517-4. doi: 10.1007/3-540-45539-6_11. URL http://dx.doi.org/10.1007/3-540-45539-6_11.

[62] Canetti R. and Krawczyk H. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42070-5. doi: 10.1007/3-540-44987-6_28. URL http://dx.doi.org/10.1007/3-540-44987-6_28.

[63] LaMacchia B., Lauter K., and Mityagin A. Stronger security of authenticated key exchange. In *Provable Security*, volume 4784 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-75669-9. doi: 10.1007/978-3-540-75670-5_1. URL http://dx.doi.org/10.1007/978-3-540-75670-5_1.

[64] Swanson C. and Jao D. A study of two-party certificateless authenticated key-agreement protocols. In *Progress in Cryptology - INDOCRYPT 2009*, volume 5922 of *Lecture Notes in Computer Science*, pages 57–71. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-10627-9. doi: 10.1007/978-3-642-10628-6_4. URL http://dx.doi.org/10.1007/978-3-642-10628-6_4.

[65] Zheng Y. Digital signcryption or how to achieve cost (signature & encryption) $\ll$ cost (signature) + cost (encryption). In *Advances in Cryptology CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 165–179. Springer Berlin Heidelberg, 1997. ISBN 978-3-540-63384-6. doi: 10.1007/BFb0052234. URL http://dx.doi.org/10.1007/BFb0052234.

[66] Free Software project. Avr libc - high quality c library for use with gcc on atmel avr microcontrollers. http://www.nongnu.org/avr-libc/.

[67] GNU-Binutils. Collection of binary tools: Gnu linker (ld) and assembly (as). `http://www.gnu.org/software/binutils/`.

[68] GNU Project. Gnu compiler collection. `http://gcc.gnu.org/`.

[69] Shamus software ltd., miracl library. `http://www.compapp.dcu.ie/~mike/shamus.html`.

[70] Pigatto D., Silva N., and Branco K. Performance evaluation and comparison of algorithms for elliptic curve cryptography with el-gamal based on miracl and relic libraries. *Journal of Applied Computing Research*, 1(2):95–103, 2011. ISSN 2236-8434. URL `http://revistas.unisinos.br/index.php/jacr/article/view/1789/0`.

[71] Szczechowiak P., Oliveira L., Scott M., Collier M., and Dahab R. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *Wireless Sensor Networks*, volume 4913 of *Lecture Notes in Computer Science*, pages 305–320. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77689-5. doi: 10.1007/978-3-540-77690-1_19. URL `http://dx.doi.org/10.1007/978-3-540-77690-1_19`.

[72] Johnson D., Menezes A., and Vanstone S. The elliptic curve digital signature algorithm (ecdsa). `http://cs.ucsb.edu/~koc/ccs130h/notes/ecdsa-cert.pdf`, 2010.

[73] Hankerson D., Menezes A., and Vanstone S. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003. ISBN 038795273X.

[74] Wenger E. and Hutter M. Exploring the design space of prime field vs. binary field ecc-hardware implementations. In *Information Security Technology for Applications*, volume 7161 of *Lecture Notes in Computer Science*, pages 256–271. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29614-7. doi: 10.1007/978-3-642-29615-4_18. URL `http://dx.doi.org/10.1007/978-3-642-29615-4_18`.

[75] Smart N. Identity-based authenticated key agreement protocol based on weil pairing. *Electronics Letters*, 38(13):630–632, Jun 2002. ISSN 0013-5194. doi: 10.1049/el:20020387.

[76] Chen L. and Kudla C. Identity based authenticated key agreement protocols from pairings. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop, 2003*, pages 219–233, June 2003. doi: 10.1109/CSFW.2003.1212715.

[77] Shim K. Efficient id-based authenticated key agreement protocol based on weil pairing. *Electronics Letters*, 39(8):653–654, April 2003. ISSN 0013-5194. doi: 10.1049/el:20030448.

[78] Xun Y. Efficient id-based key agreement from weil pairing. *Electronics Letters*, 39 (2):206–208, Jan 2003. ISSN 0013-5194. doi: 10.1049/el:20030168.

[79] Sun H. and Hsieh B. Security analysis of shim's authenticated key agreement protocols from pairings. Cryptology ePrint Archive, Report 2003/113, 2003. `http://eprint.iacr.org/`.

[80] Boyd C. and Choo K. Security of two-party identity-based key agreement. In *Progress in Cryptology Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 229–243. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28938-8. doi: 10.1007/11554868_17. URL `http://dx.doi.org/10.1007/11554868_17`.

[81] Ryu E., Yoon E., and Yoo K. An efficient id-based authenticated key agreement protocol from pairings. In *Networking 2004*, volume 3042 of *Lecture Notes in Computer Science*, pages 1458–1463. Springer Berlin Heidelberg, 2004. ISBN 978-3-540-21959-0. doi: 10.1007/978-3-540-24693-0_135. URL `http://dx.doi.org/10.1007/978-3-540-24693-0_135`.

[82] Wang S., Cao Z., Choo K., and Wang L. An improved identity-based key agreement protocol and its security proof. *Information Sciences*, 179(3):307 – 318, 2009. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2008.09.020. URL `http://www.sciencedirect.com/science/article/pii/S002002550800399X`.

[83] McCullagh N. and Barreto P. A new two-party identity-based authenticated key agreement. In *Topics in Cryptology CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 262–274. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-24399-1. doi: 10.1007/978-3-540-30574-3_18. URL `http://dx.doi.org/10.1007/978-3-540-30574-3_18`.

[84] Xie G. Cryptanalysis of noel mccullagh and paulo s. l. m. barretos two-party identity-based key agreement. Cryptology ePrint Archive, Report 2004/308, 2004. `http://eprint.iacr.org/`.

[85] Li S., Yuan Q., and Li J. Towards security two-part authenticated key agreement protocols, 2005. URL `http://eprint.iacr.org/2005/300`.

[86] Zhu R., Yang G., and Wong D. An efficient identity-based key exchange protocol with {KGS} forward secrecy for low-power devices. *Theoretical Computer Science*, 378(2):198 – 207, 2007. ISSN 0304-3975. doi: http://dx.doi.org/10.1016/j.

tcs.2007.02.021. URL http://www.sciencedirect.com/science/article/pii/S0304397507001120.

[87] Cao X., Kou W., Yu Y., and Sun R. Identity-based authentication key agreement protocols without bilinear pairings. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E91A(12):38333836, 2008. ISSN 0304-3975. doi: 10.1093/ietfec/e91-a.12.3833. URL https://www.jstage.jst.go.jp/article/transfun/E91.A/12/E91.A_12_3833/_article.

[88] Cao X., Kou W., and Du X. A pairing-free identity-based authenticated key agreement protocol with minimal message exchanges. *Information Sciences*, 180 (15):2895 – 2903, 2010. ISSN 0020-0255. doi: http://dx.doi.org/10.1016/j.ins.2010.04.002. URL http://www.sciencedirect.com/science/article/pii/S0020025510001519.

[89] Hafizul S. and Biswas G. An improved pairing-free identity-based authenticated key agreement protocol based on {ECC}. *Procedia Engineering*, 30 (0):499 – 507, 2012. ISSN 1877-7058. doi: http://dx.doi.org/10.1016/j.proeng.2012.01.890. URL http://www.sciencedirect.com/science/article/pii/S1877705812009009. International Conference on Communication Technology and System Design 2011.

[90] Huang X., Mu Y., Susilo W., Wong D., and Wu W. Certificateless signature revisited. In *Information Security and Privacy*, volume 4586 of *Lecture Notes in Computer Science*, pages 308–322. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73457-4. doi: 10.1007/978-3-540-73458-1_23. URL http://dx.doi.org/10.1007/978-3-540-73458-1_23.

[91] Shao Z. Efficient authenticated key agreement protocol using self-certified public keys from pairings. *Wuhan University Journal of Natural Sciences*, 10(1):267–270, 2005. ISSN 1007-1202. doi: 10.1007/BF02828666. URL http://dx.doi.org/10.1007/BF02828666.

[92] Wang S., Cao Z., and Wang L. Efficient certificateless authenticated key agreement protocol from pairings. *Wuhan University Journal of Natural Sciences*, 11(5):1278–1282, 2006. ISSN 1007-1202. doi: 10.1007/BF02829251. URL http://dx.doi.org/10.1007/BF02829251.

[93] Mandt T. and Tan C. Certificateless authenticated two-party key agreement protocols. In *Advances in Computer Science - ASIAN 2006. Secure Software and Related Issues*, volume 4435 of *Lecture Notes in Computer Science*, pages 37–44. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77504-1. doi: 10.1007/

978-3-540-77505-8_4. URL http://dx.doi.org/10.1007/978-3-540-77505-8_4.

[94] Shi Y. and Li J. Two-party authenticated key agreement in certificateless public key cryptography. *Wuhan University Journal of Natural Sciences*, 12(1):71–74, 2007. ISSN 1007-1202. doi: 10.1007/s11859-006-0194-y. URL http://dx.doi.org/10.1007/s11859-006-0194-y.

[95] Manman G. and Zhang F. Provably secure certificateless two-party authenticated key agreement protocol without pairing. In *Computational Intelligence and Security, 2009. CIS '09. International Conference on*, volume 2, pages 208–212, 2009. doi: 10.1109/CIS.2009.152.

[96] Hou M. and Xu Q. A two-party certificateless authenticated key agreement protocol without pairing. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, pages 412–416, 2009. doi: 10.1109/ICCSIT.2009.5234917.

[97] Yang G. and Tan C. Strongly secure certificateless key exchange without pairing. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '11, pages 71–79. ACM, 2011. ISBN 978-1-4503-0564-8. doi: 10.1145/1966913.1966924. URL http://doi.acm.org/10.1145/1966913.1966924.

[98] He D., Chen J., and Hu J. A pairing-free certificateless authenticated key agreement protocol. *International Journal of Communication Systems*, 25(2):221–230, 2012. ISSN 1099-1131. doi: 10.1002/dac.1265. URL http://dx.doi.org/10.1002/dac.1265.

[99] Zhu Z. Cryptanalysis of pairing-free certificateless authenticated key agreement protocol. Cryptology ePrint Archive, Report 2012/253, 2012. http://eprint.iacr.org/.

[100] Xiong H., Wu Q., and Chen Z. Toward pairing-free certificateless authenticated key exchanges. In *Information Security*, volume 7001 of *Lecture Notes in Computer Science*, pages 79–94. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-24860-3. doi: 10.1007/978-3-642-24861-0_6. URL http://dx.doi.org/10.1007/978-3-642-24861-0_6.

[101] He D., Chen Y., Chen J., Zhang R., and Han W. A new two-round certificateless authenticated key agreement protocol without bilinear pairings. *Mathematical and Computer Modelling*, 54(11 12):3143 – 3152, 2011. ISSN 0895-7177. doi: http://

dx.doi.org/10.1016/j.mcm.2011.08.004. URL http://www.sciencedirect.com/science/article/pii/S0895717711004845.

[102] He D., Padhye S., and Chen J. An efficient certificateless two-party authenticated key agreement protocol. *Computers & Mathematics with Applications*, 64(6):1914 – 1926, 2012. ISSN 0898-1221. doi: http://dx.doi.org/10.1016/j.camwa.2012.03.044. URL http://www.sciencedirect.com/science/article/pii/S0898122112002490.

[103] Cheng Q. Cryptanalysis of an efficient certificateless two-party authenticated key agreement protocol. Cryptology ePrint Archive, Report 2012/725, 2012. http://eprint.iacr.org/.

[104] Mohamed N., Hashim M., Bashier E., and Hassouna M. A secure and efficient key agreement protocol based on certificateless cryptography. In *International Journal of Intelligent Computing Research*, volume 3, pages 298–304. infonomics-society, 2012. URL http://www.infonomics-society.org/IJICR/Contents%20Page%20Volume%203%20Issue%201%20and%202.pdf.

[105] Hagras E., Aly H., and Saied D. Energy efficient key management scheme based on elliptic curve signcryption for wireless sensor networks. In *Proceedings of the 28th national radio science conference*, National Telecommunication Institute, Egypt, 2011.

[106] Zheng Y. Shortened digital signature, signcryption and compact and unforgeable key agreement schemes. In *IEEE P1363a: standard specifications for public-key cryptography: additional technique*, pages 1–51, 1998. URL http://grouper.ieee.org/groups/1363/StudyGroup/contributions/signcr.pdf.

[107] Zheng Y. Signcryption and its applications in efficient public key solutions. In *Information Security*, volume 1396 of *Lecture Notes in Computer Science*, pages 291–312. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64382-1. doi: 10.1007/BFb0030430. URL http://dx.doi.org/10.1007/BFb0030430.

[108] Bao F. and Deng R. A signcryption scheme with signature directly verifiable by public key. In *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 55–59. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64693-8. doi: 10.1007/BFb0054014. URL http://dx.doi.org/10.1007/BFb0054014.

[109] Gamage C., Leiwo J., and Zheng Y. Encrypted message authentication by firewalls. In *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 69–81. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-65644-9. doi: 10.1007/3-540-49162-7_6. URL http://dx.doi.org/10.1007/3-540-49162-7_6.

[110] Jung H., Chang K., Lee D., and Lim J. Signcryption schemes with forward secrecy. In *Proceedings of Information Security Application-WISA'01*, pages 403–475, Seoul, Korea, 13-14 September 2001, 2001.

[111] Zheng Y. and Imai H. How to construct efficient signcryption schemes on elliptic curves. *Information Processing Letters*, 68(5):227 – 233, 1998. ISSN 0020-0190. doi: http://dx.doi.org/10.1016/S0020-0190(98)00167-7. URL `http://www.sciencedirect.com/science/article/pii/S0020019098001677`.

[112] Hwang R., Lai R., and Su F. An efficient signcryption scheme with forward secrecy based on elliptic curve. *Applied Mathematics and Computation*, 167(2):870 – 881, 2005. ISSN 0096-3003. doi: http://dx.doi.org/10.1016/j.amc.2004.06.124. URL `http://www.sciencedirect.com/science/article/pii/S0096300304005351`.

[113] Toorani M. and Beheshti A. Cryptanalysis of an efficient signcryption scheme with forward secrecy based on elliptic curve. In *Computer and Electrical Engineering, 2008. ICCEE 2008. International Conference on*, pages 428–432, Dec 2008. doi: 10.1109/ICCEE.2008.147.

[114] Kim R. and Youm H. Secure authenticated key exchange protocol based on ec using signcryption scheme. In *Hybrid Information Technology, 2006. ICHIT '06. International Conference on*, volume 2, pages 74–79, Nov 2006. doi: 10.1109/ICHIT.2006.253592.

[115] Zhou X. Improved signcryption scheme with public verifiability. In *Knowledge Engineering and Software Engineering, 2009. KESE '09. Pacific-Asia Conference on*, pages 178–181, Dec 2009. doi: 10.1109/KESE.2009.54.

[116] Toorani M. and Beheshti A. An elliptic curve-based signcryption scheme with forward secrecy. *Journal of Applied Sciences*, 9(6):10251035, 2009. ISSN 1812-5662. URL `http://scialert.net/qredirect.php?doi=jas.2009.1025.1035&linkid=pdf`.

[117] Elsayed M. and Hasan E. Elliptic curve signcryption with encrypted message authentication and forward secrecy. *International Journal of Computer Science and Network Security*, 9(1):395398, 2009. ISSN 1738-7906. URL `http://paper.ijcsns.org/07_book/200901/20090155.pdf`.

[118] Hamed A. and El-Khamy S. New low complexity key exchange and encryption protocols for wireless sensor networks clusters based on elliptic curve cryptography. In *Radio Science Conference, 2009. NRSC 2009. National*, pages 1–13, March 2009.