DESIGN AND ANALYSIS OF CERTIFICATELESS SIGNATURE SCHEME FOR VARIOUS APPLICATIONS

A
THESIS
SUBMITTED TO

GALGOTIAS UNIVERSITY
GREATER NOIDA

IN FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY
IN
COMPUTER SCIENCE AND ENGINEERING

By
PANKAJ KUMAR
Regd. No. – 14SCSE301006

SCHOOL OF COMPUTING SCIENCE & ENGINEERING
Galgotias University, Gr. Noida. INDIA

November 2019

Dedicated to my Parents, Wife and Kids

...and the eternal spirit of humanity

DECLARATION

It is certified that the work presented in this thesis entitled "Design and Analysis of Certificateless Signature Scheme for Various Applications" is original and has been carried out by the candidate in the Department of Computer Science, Galgotias University. To the best of candidate’s knowledge, this work has not been submitted in part or full for the award of any degree of this or any other university.

Pankaj Kumar

(Candidate)

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

Dr. Vishnu Sharma

Department of Computer Science and Engineering

Galgotias College of Engineering and Technology

Greater Noida

The Ph.D. Viva-Voice examination of Mr. Pankaj Kumar Research Scholar has been held on __________.

ACKNOWLEDGEMENT

The Research work presented in this thesis would not been possible without my close association with many peoples. I would like to take this opportunity to extend my sincere thanks and appreciation to all those who made this PhD thesis possible.

First and foremost, praises and thanks to the God, the Almighty, for his showers of blessings throughout my research work to complete the research successfully.

I express my deepest regards to my Ph.D. supervisor Dr. Vishnu Sharma. I will be always obliged to him for showing faith in my capabilities, when I myself was losing the faith in me. He has not only introduced me to the subject of network security but he has also inculcated my interest in this highly exciting and interdisciplinary research area. I am also thankful to him for his consistent support, guidance, and patience for the entire duration of adventurous Ph.D. years of my life.

Very special thanks to the Dr. Manoj Khanna, Principal of Ramjas College for giving me the valuable support during my doctoral research. I also express my heartly thanks to Dr. Ruchika Verma and all faculty of Department of Mathematics from Ramjas College specially Mr. Sandeep Chauhan, Suresh Kumar Yadav. I am very thanksful for support and help of Mr. Vinod Kumar.

I express my deep gratitude to my parents for this life. I thanks to God for this lovely family in which my brother and sister are my best friends. I owe my deepest gratitude towards my wife for his eternal support and understanding of my goals and aspirations. His love and support has always been my strength. At the end, my kids Yug and Drishti, my cute angels, I feel sorry to steal your time to complete the research work.

Thanks God for being with me always and providing all the strength, happiness and good health to carry out this study.

Thank you so much………

Pankaj Kumar

Contents

Page No.

CANDIDATE'S DECLARATION

DEDICATION PAGE

ACKNOWLEDGEMENTS

LIST OF PUBLICATIONS

LIST OF PERSENTATIONS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

ABSTRACT

Chapter-1: Introduction..1-21

1.1 Security thread..2

1.1.1 External and Internal Threats..2

1.1.2 Unstructured and Structured Threats..3

1.2 Cryptosystem..5

1.2.1 Fundamental targets of cryptography..5

1.2.2 Basic function of cryptography..6

1.3 Digital signature..6

1.3.1 Model of Digital Signature..7

1.3.2 Algorithm for the digital signature..8

1.4 Symmetric key cryptography..8

1.4.1 Basic function of symmetric key cryptography..9

1.5 Public key cryptography..10

1.5.1 Basic function of public key cryptography..11

1.6 Identity Based Public Key Cryptography 12

1.6.1 Algorithm for identity based signature scheme 14

1.7 Certificateless Public Key Cryptography (CL-PKC)…... 14

1.7.1 Formal definition of a CLS Scheme used in our work 16

1.8 Security Model of a certificateless signature scheme 17

1.9 Cryptanalysis.. 21

1.9.1 Some mathematical security attacks use in the CLAS... 21

1.9.1.1 Honest but curious attack.. 21

1.9.1.2 Insider attack... 21

1.9.1.3 Universal attack 21

1.9.1.4 Collision resistance attack... 21

Chapter-2: Mathematical Background..23-35

2.1 Division 23

2.1.1 Divisibility Theorems 23

2.1.2. Prime Numbers 24

2.1.3 Fundamental theorem of arithmetic.. 24

2.1.4 The Division Algorithm.. 24

2.1.5 Greatest Common Divisors.. 25

2.1.6 Relatively Prime Integers.. 25

2.1.7 Least Common Multiples.. 25

2.2. Modular Arithmetic 25

2.2.1 Congruence 25

2.2.2 Properties of Congruence... 25

2.2.3 Modular Arithmetic Operations... 26

2.2.4 Properties of Modular Arithmetic.. 26

2.2.5 Modular Inverse 26

2.2.6 Chinese Remainder Theorem (CRT) 27

2.2.7 Euler’s Theorem 27

2.2.8 Fermat’s Little Theorem 28

2.3 Concept related with Abstract Algebra 28

2.3.1 Group 28

2.3.2 Cyclic Group 29

2.3.3 Ring 29

2.3.4 Integral Domain 29

2.3.5 Field 29

2.3.6 Finite Fields of the Form GF (*p*) 30

2.3.7 Galois Fields 30

2.3.8 Finite Fields of Order p 30

2.4 Elliptic Curves 30

2.4.1 Elliptic Curves over Finite Fields 31

2.4.2 Elliptic curves over F_{2^m} 34

2.4.3 Elliptic Curve: Some useful Definitions 33

2.5 Bilinear Pairing 36

2.5.1 Some properties of Bilinear Pairing 34

Chapter-3: Literature Review 36-44

3.1 Some Basic Digital Signature Scheme 36

3.1.1 RSA Signature Scheme 36

3.1.2 Elgamal signature scheme 36

3.1.3 Schnorr signature scheme 37

3.2 Comparison between Various Existing CLS Scheme 41

Chapter-4: Objectives and Methodology…… 45-46

4.1 Objectives 45

4.2 Methodology 45

Chapter-5: Cryptanalysis of Existing Certificateless Signature Schemes 47-74

5.1 Preliminary 48

5.1.1 Bilinear Map 48

5.1.2 Security Models 48

5.2. Game Plan 49

5.3 Cryptanalysis of Deng et al CLS and CLAS scheme 50

5.3.1 Review of the Deng et al CLS Scheme 50

5.3.2 Review of Deng et al CLAS Scheme 51

5.3.3. Attack on Deng et al. CLS Scheme 52

5.3.4. Attack on Deng et al. CLAS scheme 54

5.3.5. Collision insider attack on Deng et al scheme 55

5.4 Cryptanalysis of Deng et al CLS and CLAS Scheme 56

5.4.1 Deng et al CLAS scheme 55

5.4.2 Review of the Deng et al CLS Scheme 57

5.4.3 Cryptanalysis of Deng et al CLS scheme 58

5.4.4 Cryptanalysis of Deng CLAS scheme 59

5.4.5 Improved Deng et al Certificateless Signature Scheme 60

5.5. Cryptanalysis of Liu et al. CLS Scheme 60

5.5.1. Review of Liu et al. CLS scheme 61

5.5.2. Analysis of Liu et al. CLS scheme 62

5.5.3. Improvement of Liu et al. CLS scheme 63

5.6.Cryptanalysis of Horng et al. CLS and CLAS Scheme 64

5.6.1. Review of the Horng et al. CLS and CLAS Scheme 64

5.6.2 Security analysis of Horng et al. CLAS Scheme 66

5.7 Cryptanalysis of Malhi and Batra CLS and CLAS Scheme 67

5.7.1 Review of Malhi and Batra CLAS scheme 67

5.7.2 Attack 1 Malhi and Batra CLAS scheme 69

5.7.3 Attack 2 Malhi and Batra CLAS scheme 70

5.7.4 Attack 3 Malhi and Batra CLAS scheme 71

5.7.5 Collision insider attack of Malhi-Batra Certificate Aggregate Signature Scheme.. 73

5.7.7 Improvement of Malhi and Batra CLAS Scheme 75

Chapter-6: Cryptanalysis and Improvement of a Certificateless Aggregate Signature Scheme..75-116

6.1 Formal security model of Certificateless signature (CLS) scheme 77

6.2 Review of He *et al.* CL-AS scheme 79

6.3 Security Analysis of He et al. CLS scheme... 81

6.3.1 Attack 1 81

6.3.2 Attack 2 82

6.4 Security Analysis of He *et al.* CL-AS scheme... 83

6.4.1 Attack 1 83

6.4.2 Attack 2 83

6.4. Our Certificateless signature scheme (CLS) scheme.. 84

6.4.1 Our Certificateless aggregate signature scheme (CLAS) scheme.. 85

6.4.1.1 Theorem1.. 86

6.4.1.2 Theorem 2... 91

Chapter 7: A Certificateless Aggregate Signature Scheme for Healthcare Wireless Sensor Network 95-119

7.1 Introduction 96

7.2 Related work.. 99

7.3 System Model... 99

7.4 Some special Symbols and their descriptions 101

7.5 Security Model of a CLS and CL-AS Scheme used in HWSN 102

7.6 Proposed Certificateless Signature Scheme used in HWSN 105

7.7 Security proofs 107

7.7.1 Theorem 1 107

7.7.2 Theorem 2 111

7.7.3 Theorem 3 114

7.8 Batch Verification 116

7.9 Performance Analysis 117

Chapter-8: Secure CLS and CL-AS Schemes Designed for VANETs 120-141

8.1. Some special symbol used in the chapter 120

8.2 Security Models of a CLS Scheme 121

8.3 Our Certificateless Signature Scheme 125

8.3.1 Framework of our system 125

8.3.2 Our Certtificateless Aggregate Signature Scheme 128

8.4. Security Analysis of the Proposed CLS Scheme 129

8.4.1 Theorem 1 129

8.4.2 Theorem 2 134

8.4.3 Security Analysis of CL-AS scheme 137

8.4.3.1 Theorem 3 138

8.5. Comparison of CL-AS scheme 140

Chapter-9: Conclusion and Future work 142-143

List of Tables

Table 1: Algorithms for CLS scheme 17

Table 2: Comparison between various existing CLAS Scheme 41

Table 7.1: Some symbols used in the chapter 7 99

Table 7.2: Performance Comparison of CL-AS schemes with running time (*ms*) 117

Table 7.3: Total Energy Consumption (mJ) 119

Table-8.1**:** Some special symbols used in the chapter 8 120

Table-8.2: Comparison of cost based on operations and running time (ms) Involved 140

List of figures

Fig 1.1 Basic model of communication 3

Fig 1.2: Interruption 4

Fig 1.3: Interception 4

Fig 1.4: Modification 5

Fig 1.5: Fabrication 5

Fig 1.6: Overview of digital signature 7

Fig 1.7: Pictorial view of symmetric key cryptography 10

Fig 1.8: Public key encryption scheme 11

Fig 1.9: Public key digital scheme 12

Fig 1.10: Security working against adversary A_1 19

Fig 1.11: Security working against adversary A_1… 20

Fig 1.12: Pictorial view of collision resistance attack 21

Fig 2.1: Elliptic Curve over R^2 : $y^2=x^3-3x+3$ 22

Fig 2.2: Sum of 2 points P and Q on the curve $t^3 = s^3 + as + b$ .... …32

Fig 2.3: Doubling of a point P, R = 2P over the elliptic curve $t^3 = s^3 - 3s + 3.32$ 33

Fig 7.1: Architecture of healthcare wireless sensor network (HWSN) 96

Fig 7.2: Code Blue project 97

Fig 7.3: General view of Aggregate Signature 98

Fig 7.4: Framework of healthcare wireless sensor network 100

LIST OF PUBLICATIONS

1. P. Kumar, V. Sharma,”Cryptanalysis and Improvement of two provably secure certificates signature schemes” Published *in* Jr. of Combinatorics, Information & System Sciences *(JCISS),* Vol 42, 2017.
2. P. Kumar, V. Sharma, V. Kumar, “Cryptanalysis of Efficient Certificateless Aggregate Signature Scheme” Published in Global and Stochastic Analysis Vol. 4 No. 1, 2017. (Scopus)
3. P. Kumar, V. Sharma, G. Sharma ”Certificateless Aggregate Signature Schemes: A Review” published in the proceedings of International conference on IEEE, held at Galgotia University , April 29-30, 2016, pp 531-536. (Scopus indexed)
4. P. Kumar, V. Sharma, G. Sharma” Cryptanalysis of a Certificateless Aggregate Signature Scheme” published in the proceedings of in International conference on IEEE, held at Galgotia University, April 29-30,2016 PP 1095-1098. (Scopus indexed)
5. P. kumar, V. Sharma” On the Security of Certificateless Aggregate Signature Scheme used in Vehicular Ad-hoc Network”, Published in the proceedings of international conference on Springer (SOCTA), held at Amity University, Dec 28-30, 2016. (Scopus indexed)
6. P. kumar, V. Sharma “A Comment on Efficient Certificateless Aggregate Signature Scheme” International conference on computing, Communications and Automation held in Galgotias University held at 5-6 may 2017. (Scopus indexed)
7. P. Kumar, V. Sharma, S. Kumari, A. K. Sangaiah, J. Wei, Li Xiong” A new certificateless aggregate signature scheme for Healthcare Wireless Sensor Network” published in Journal of Sustainable Computing, Informatics and Systems , Elsevier Journal Vol 18. (SCI)
8. P. Kumar, V. Sharma,V. Kumar “Collision Resistance Attack on Certificateless Signature Schemes in Vehicular Ad-hoc Networks”. International Journal of Pure and Applied Mathematics, Volume 118 No. 22 2018, 1441-1445. (Scopus, impact factor .29)

9. P. Kumar, V. Sharma, S. Kumari, A. K. Sangaiah, H.K. Islam, Li Xiong ”Secure CLS and CL-AS Schemes Designed for VANETs”. published in Journal of Super Computing, Springer Vol 74. (SCI)

10. P. Kumar, V. Sharma, “An efficient and secure certificateless aggregate signature from bilinear map”. published in International Journal of Information, Security and Privacy vol 13(4) (ESCI and Scopus).
11. P. Kumar, V. Sharma, “Insecurity of a Certificateless Signature Scheme” published in IEEE conference ICACCCN-2018 Galgotias University held at 12-13 Oct 2018. (Scopus indexed)

12. P. Kumar, V. Sharma, “Insecurity of a Secure Certificate-Based Signature Scheme”. published in IEEE conference ICACCCN-2018 Galgotias University held at 12-13 Oct 2018. (Scopus indexed)
13. P. Kumar, V. Sharma, V.Kumar, A. sharma, “Cryptanalysis and further improvement of a certificateless aggregate signature scheme” published in IOSR Journal of Computer Engineering, 16(3) 2017.

LIST OF PRESENTATIONS

1. P. Kumar, V. Sharma, G. Sharma."Certificateless Aggregate Signature Schemes: A Review" presented in International conference on IEEE, held at Galgotia University , April 29-30.
2. P. Kumar, V. Sharma, G. Sharma." Cryptanalysis of a Certificateless Aggregate Signature Scheme " presented in International conference on IEEE, held at Galgotia University , April 29-30, 2016.
3. P. Kumar, V. Sharma, " A Comment on Efficient Certificateless Aggregate Signature Scheme" presented in International conference on IEEE, held at Galgotia University, may 5-6, 2017.
4. P. kumar, V. Sharma"Cryptanalysis of Efficient Certificateless Aggregate Signature Scheme" presented in International conference on Forum for Interdisciplinary Mathematics held in Manipal University Jaipur Dec 22-25, 2016.
5. P. kumar, V. Sharma" On the Security of Certificateless Aggregate Signature Scheme used in Vehicular Ad-hoc Network", presented in international conference on Springer (SOCTA), held at Amity University, Dec 28-30, 2016.
6. Pankaj Kumar, V. Sharma" Security Analysis of Certificateless Aggregate Signature Scheme" presented in International Seminar on History of Mathematics, held at Ramjas College (University of Delhi) april, 26-28 2017
7. Pankaj Kumar, V. Sharma" Review of Security Attacks in Certificateless Aggregate Signature Scheme" presented in International Seminar on History of Mathematics, held at Ramjas College (University of Delhi) April, 26-28 2017
8. Pankaj Kumar, V. Sharma" A study on identity based authentication protocol for cloud computing" presented in the proceeding of National Conference (ETCCBDA-2017), held at MDU Rohtak, Haryana 6 March 2017, PP 39-42.
9. Pankaj Kumar, Vishnu Sharma "Cryptanalysis and further improvement of a certificateless aggregate signature scheme" presented in the proceeding of National Conference (ETCCBDA-2017), held at MDU Rohtak, Haryana 6 March 2017, PP 181-184.

10. Pankaj Kumar, V. Sharma ”Security analysis of certificateless signature scheme ” presented in the international conference Silver Jublee of Indian Society of Industrial and Applied Mathematics (ISIAM), 29-31 January 2016.
11. P. Kumar, V. Sharma, “Collision Resistance Attack on Certificateless Signature Schemes in Vehicular Ad-hoc Networks”. presented in National Conference on "Advances in Operations Research and Mathematical Sciences (AORMS-2018) held at 24-25 Feb 2018 Bijnor.
12. P. Kumar, V. Sharma ‘Energy Efficient Certificateless Signature Scheme for Healthcare Wireless Sensor Network’ presented in International Conference on Sustainable Agriculture, Energy, Environment and Technology (Icsaeet-2018) held on February 24-25, 2018.
13. P. Kumar, V. Sharma "Certificateless Aggregate Signature Scheme for Healthcare Wireless Sensor Network" presented in the conference ICAM 2018 held at Motilal Nehru College, Delhi from 19-20 February, 2018.
14. P.Kumar, “An Identity-Based Authentication Framework for Big Data Security” presented in the conference ICCCN 2018, 29-30 march 2018 held at NITTTKR Mohali Chandigarh.
15. P.Kumar "Secure CLS and CL-AS Schemes Designed for VANETs” presented in the conference "Applications of Graph & Network in Computational Studies, Bioinformatics & Engineering And their Technical Terminology" 16-18 march 2018, held at JNU Delhi.
16. P. Kumar, V. Sharma, “Insecurity of a Certificateless Signature Scheme” presented in IEEE conference ICACCCN-2018 Galgotias University held at 12-13 Oct 2018.
17. P. Kumar, V. Sharma, “Insecurity of a Secure Certificate-Based Signature Scheme”. presented in IEEE conference ICACCCN-2018 Galgotias University held at 12-13 Oct 2018.

PREFACE

We divide this thesis into nine chapters: first chapter gives an introduction of the network security, cryptography, digital signature, certificateless signature scheme and some about the provable security of the signature scheme.

Chapter two deals with the some mathematical concept used in the thesis mainly number theory, group theory, ring theory, integral domain, field, Galois theory and elliptic curve with its properties, in the last of chapter we discuss about the bilinear pairing and some useful theorems.

In Chapters three, we present the literature review of previous existing certificateless signature scheme. Chapter 4 gives the views of our thesis means describe about the our objectives of the thesis with methodology. In the chapter 5 we did the cryptanalysis of many schemes. First we gave the review of the schemes thereafter we find the security leaks of these schemes.

In the chapter 6, we have done security analysis of the He et al scheme and found the security leaks of the He et al scheme. We proposed an improved certificateless signature scheme to cover all the aspect about the leaks in He et al scheme. We proved the security of our improved scheme with random oracle model under Computational Diffie-Hellman assumption.

In the chapter 7 and 8, we proposed an efficient certificateless aggregate signature scheme in which first CLAS scheme is suitable for healthcare wireless sensor network and second CLAS scheme is targeted the vehicle ad-hoc network. Securities of both the schemes are proven with random oracle model under Computational Diffie-Hellman assumption and the results are compared with the previous schemes.

In the last chapter 9, we conclude the thesis and suggest some future directions of the research work.

ABSTRACT

Digital security is the necessity of our lives due increase the invisible communication. We ensured the security services like confidentiality, authentication, integrity and non-repudiation in our system. Cryptography is the important technique for providing the security services. Digital signature is the keynote part of cryptography that ensures the confidentiality, authentication, integrity and non-repudiation. We adopt the certificateless signature scheme (CLS) in our work that removes the leakage of ID based public key and public key cryptography. CLS scheme is very efficient techniques in recent scenario which is very helpful to provide security. We used the aggregate property and associate with our proposed CLS scheme. Aggregate scheme is a many to one map that allows the different signature on the different message map to a single signature. This feature is very beneficial in such environment where bandwidth and computational time saving is required for e.g., wireless sensor network, vehicular ad-hoc network, internet of things and an endless list. Certificateless aggregate signature (CLAS) is more efficient scheme that enjoying both the features of certificateless and aggregate concept. We have done the review of many existing CLS schemes. After studying a lot of CLS, we have done the cryptanalysis of these schemes and identify many schemes like Deng et al signature scheme, Horng et al signature scheme, Malhi and Batra signature scheme and Liu et al Signature scheme be insecure against many concrete attacks like adaptive chosen message attack, malicious but passive attack, honest but curious attack and collision insider attack. We fulfils the leakage present in He et al scheme and Malhi and Batra scheme, and proposed an improved CLS scheme. Also, we prove the security of our proposed CLS signature scheme in the random oracle model with Diffie-Hellman assumption. We proposed a novel CLAS using the concept of aggregate signature for secure communication in healthcare wireless sensor networks. The proposed helps to protect the online data, from the unauthorized entities in healthcare wireless sensor network. The security of our construction is proved by Diffie Hallman assumption under the random oracle model. Through experimental results, we have proved that our certificateless aggregate signature scheme is more computational and energy efficient as compared to the existing schemes.

Chapter 1

Introduction

In earlier decades, most of the operations are done by manually like bank office, business work and an endless list. Before the internet revolution, an organization collects the information, data files and put these physical files in the locker for security. Confidentiality of the file is the main question, an authorized person who has the locker's key can see these files and modified it. So privacy and integrity of the files is a very important concern for like these systems. For retaining secure from the unauthorized entity, we put these files in the professional organization.

But in the current era, it is not necessary to do manually. After the internet revolution, we can store data, transfer data via internet. The internet becomes a value able asset of our real life. The internet has become an essential part of our life and we share all our information by social media or keep data on the cloud.

Since the internet is a very useful and user friendly for the user. The drawback of the internet that the information is kept online and anyone unauthorized person can attack on the information. It is very necessary that our information should be kept secret and no one can attack on your information. Data is very crucial for every stage. For example, we take a bank scenario, every transaction in the bank is done by online and the data are stored on the server. If any unauthorized personal attacks on the bank server and access the all bank accounts, change the password of the bank employees than the all functions can be damaged. So no physical robbery needed just only one person sitting outside can break the security via the internet. So we need our system to be so much secure that no one can enter your security**.**

In the healthcare industry, Online data sharing is one of the requirements to increase the efficiency and reduced the time constraints in the healthcare industry. In the healthcare wireless sensor network, the Patient's report is available online to share with health professionals without any delay after the patient's checkup. Data privacy becomes an important issue in healthcare due to direct involvement of personal health related data of patients. Modified data may become a serious cause of casualty for the patient.

So we can say that internet becomes our lives very easier but risky [5, 6, 7]. To be secured the information from the unauthorized user, many challenges come like that the information should not be modified (integrity), to be secret from the unauthorized access (confidentiality), data should be available for the authorized user to excess (availability) and ensure that the data should be access by authorized user (authentication).

Network security is the branch of the computer engineering that deal with the issue to protect the data during their transmission. Before starting network security, we study about the attacks, virus, worm, phishing and many other things. For example, one of the famous attacks is computer virus where an intruder transfers the computer virus into the target computer via any method like internet, pen drive, website, email and any other possible method. A virus is a computer program that crushes the hard disk of the system.

1.1 Security thread:

Anything that comprises our system security is called the security thread. Security threats can be classified into two category external users and internal user.

1.1.2 External and Internal Threats

Security threats can come from two locations:

- External users
- Internal users

An external threat occurs when an attacker belongs to outside of the network. Someone outside of the network creates a security threat and tries to break the security of the network. For example, in a bank, a person outside of the bank tries to hack the security of the bank.

An internal threat occurs when an attacker belongs to inside of the network when someone from inside the network creates a security threat to the network.

1.1.3 Unstructured and Structured Threats

Security threats also falls under two categories:

- Unstructured threats
- Structured threats

An unstructured security threat is performed by an inexperienced person who wants to try the break of the network security or wants to enter into the system. A structured security threat is performed by an expert and experience person who wants to try the break of the network security or wants to enter into the system. Structure security threat is very dangerous for the system because it is very difficult to detect in the system.

Security attacks, we can classified security attacks into four major categories.

(i) Interruption
(ii) Interception
(iii) Modification
(iv) Fabrication

Basic model: we start to explain by a basic model of communication in which two parties sender S and receiver is involved in the communication. The sender wants to send the information to the receiver and the receiver receive the information from the sender.

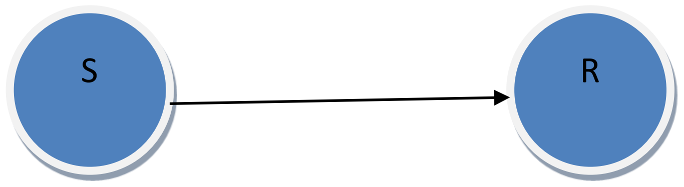

Fig 1.1 Basic model of communication

Interruption: Intruders destroy the system like hard disk, any part of hardware, or disable the file management system and make unavailable from the receiver. This attack is on the availability of the user.

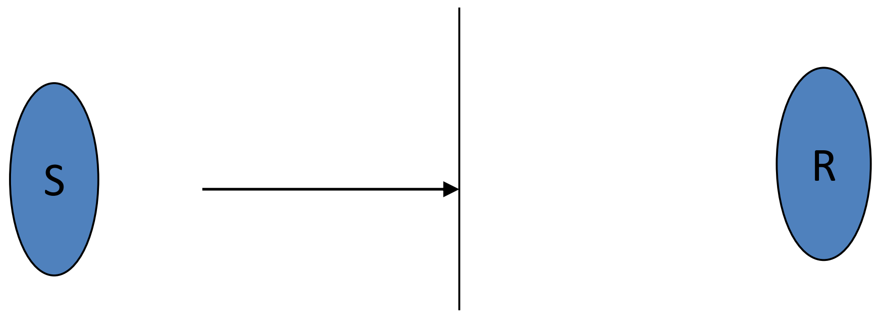

Fig 1.2: Interruption

Interception: An unwanted entity who gains access to an asset. This attack is done on confidentiality. The unwanted entity could be a program, a person, or a computer. This attack is a passive attack in which intruder analyzes the information transferring sender to receiver.

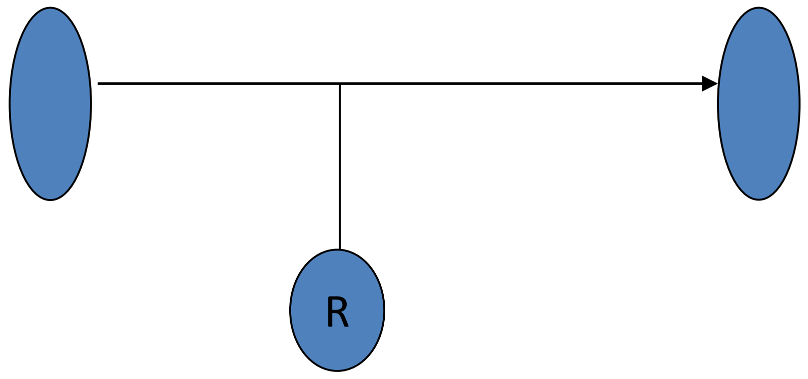

Fig 1.3: Interception

Modification: this attack is active attack. An unauthorized entity tries to change the data, alters the message or modifies the content during transfer the message.

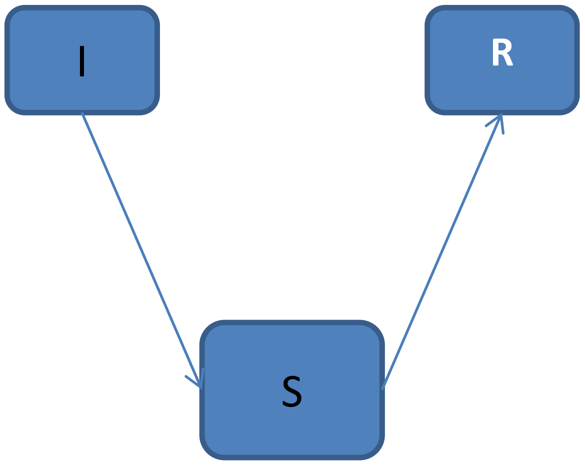

Fig 1.4: Modification

Fabrication: In this attack, unauthorized entities send a new message to the receiver and the receiver could not understand the origin of the message.

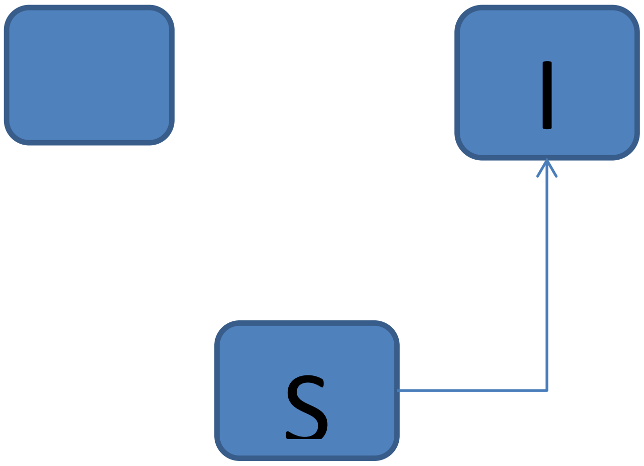

Fig 1.5: Fabrication

1.2 Cryptosystem

Due to spread of the information, we need to protect our system from the external and internal adversaries. Cryptography is a very important technique in the current scenario to protect information. In the data communication, at least two parties are involved in the communication called sender and receiver. Sender's intensity to send the information all the parties involved in the communication and the receiver is the authentic entities who accept the relevant information. The basic concept of the network security when the sender sends the information through the public channel them protect the system by the adversaries present in the channel who wants to capture the relevant information.

Cryptography is one of the approaches for the network security to secure our communication and pretended by the unauthorized entities. Cryptography assures many things such as data integrity, confidentiality, availability and secrecy of the information.

1.2.1 Fundamental targets of cryptography

(i) Confidentiality: To keep the data secure from the unauthorized entities who wants to access the information.

(ii) Data integrity: To make sure that data has not been modified by unauthorized entities.

(iii) Authentication: To ensure that the communication made between the authenticated entities. The intruder cannot enter in our network and excess the information.

(iv) Non-repudiation: The entities who want to send information cannot deny with their previous actions or commitments.

1.2.2 Basic function of cryptography

- Plain Text (P): plain text is the message for encryption that the sender wants to send the receiver.
- Secret Key (K): secret key is the key, by which the message is going to be encrypted.
- Cipher Text (C): cipher text is the message after encryption by secret key
- Encryption algorithm (EA): massage is to be encrypted by the sender with the encryption algorithm by using plain text and a secret key.
- Decryption algorithms (DA): massage is to be decrypted by the receiver with the decryption algorithm by using ciphertext and a secret key.

1.3 Digital signature

Digital signature is a very useful cryptographic technique for transferring the information securely. By digital signature we bind a person/entity with the digital data. That can be verified by the third independent party. In the physical world, the records are preserved wit manually sign. Digital signature is likely same concept like the physical world. A massage is bind with the digital signature and person who generate the binding message and signature cannot deny. In the digital signature, the signer binds the signature with the message with their private key. This property is known as non repudiation. By digital signature we achieve many cryptographic goals like privacy, integrity and non repudiation. Digital signature is very useful in the business applications and other application like e commerce, banking, networking, healthcare industry and an endless list.

1.3.1 Model of Digital Signature

As discussed earlier, the digital signature scheme is a cryptographic primitive. A general model of digital signature scheme is mentioned below;

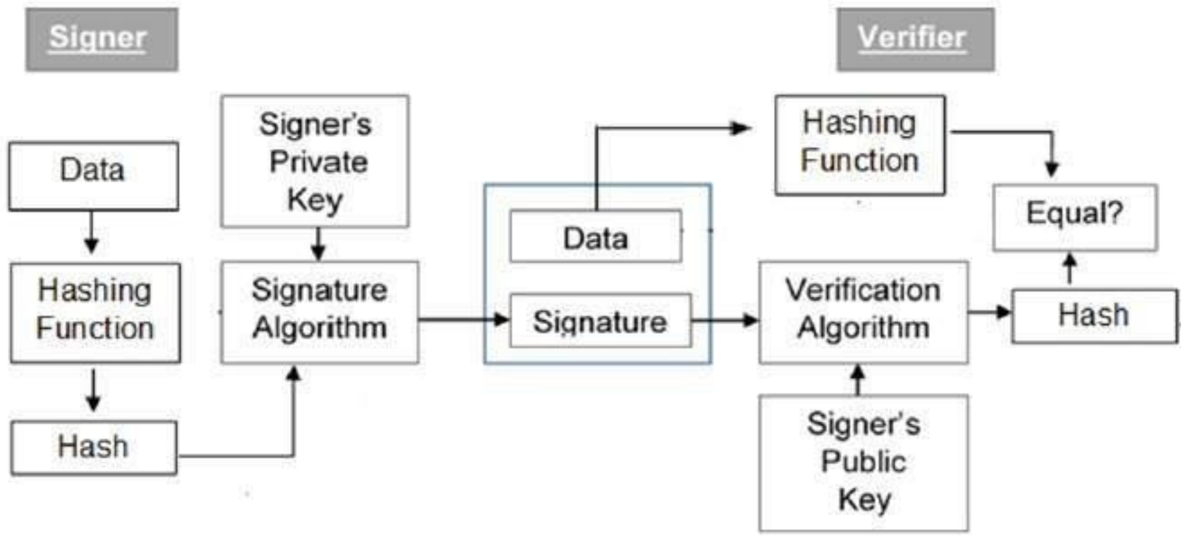

Fig 1.6: overview of digital signature

The following points explain the entire process in detail-

The basic signature scheme has started with the public key cryptography, in which every user has a private-public key pair. In general signing key and verifying key are different. Signer attaches data with the hash function and generate a hash value of the data. Thereafter signer generates a signature with their private key by a signature algorithm and send this signature to the verifier by insecure channel. In the verifier end, verifier knows the public key of the signer. With the help of the signer's public key and the verification algorithm, the verifier can verify the signature. If it is verified then this signature is accepted otherwise reject.

Since the signer uses their private key to generate the signature then he cannot deny letter for their authenticity. Since the signer uses the hash function of the message and take a hash value before generating the signature. This leads the integrity of the message such that no one can change or modified the data.

1.3.2 Algorithm for the digital signature

A traditional digital signature scheme consists the three algorithms.

KEY-GEN: Taking an input security parameter $1^{k,}$ this algorithm generates a private-public key pair.

SIGNATURE: This algorithm is run by the signer, signer sign the message m with their private key pk. Which can be notified as $\sigma = Sign_{pk}(m)$

VERIFICATION: This algorithm is run by the verifier, the verifier verifies the signature with the public key of the signer. If the verification is successful, then accept the signature other reject the signature.

$$Verify_{pk}(m, Sign_{pk}(m)) = 1$$

1.4 Symmetric key cryptography

Symmetric key cryptography is the very basic technique of the cryptography. In this technique sender and receiver agree with the same key. Both the party agrees with the key before staring the communication. The sender encrypts the plaintext with secret key as well as decrypt by the receiver with the same key [3]. The main view of the cryptography is the privacy of the information and set the communication with the different parties at different location. Since the agreement on the key is set up prior the starting the communication, this will create some major drawback with the symmetric key cryptography. Suppose n parties are involved in the communication, then we require n (n-1)/2 keys. When a large number of parties involve then we need a huge number of secret keys. It is very difficult to preserve the privacy of the private keys and huge number of keys overload on the system. Predistribution of the key also creates many problems. It is very difficult to design digital signature because of predistribution of keys and we cannot decide the accountability regarding non-repudiation.

1.4.1 Basic function of symmetric key cryptography

- Plain Text (P): plain text is the message for encryption that the sender wants to send the receiver.
- Secret Key (K): secret key is the key, by which the message is going to be encrypted.
- Cipher Text (C): cipher text is the message after encryption by the secret key (K).
- Encryption algorithm (EA): massage is to be encrypted by the sender with the encryption algorithm by using plain text and a secret key (K).
- Decryption algorithms (DA): massage is to be decrypted by the receiver with the decryption algorithm by using cipher text and a secret key (K).

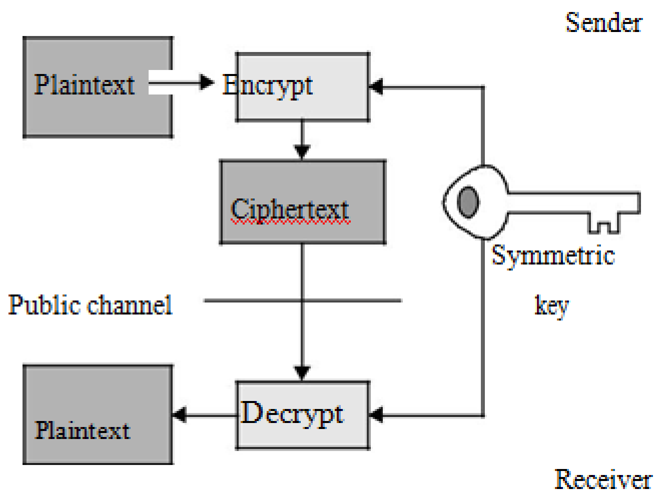

Fig 1.7: Pictorial view of symmetric key cryptography

Public key cryptography gives the solution of these drawbacks predistribution of key and the huge number of keys.

1.5 Public key cryptography

Public key cryptography is a cryptographic primitive which gives the solution of the drawback in the symmetric key cryptography. Diffe- Hellman introduces the public key cryptography in 1976 [3]. The basic concept of public key cryptography is that the key used in the encryption by the sender is different from the key used in the decryption used by the receiver. In the public key cryptography environment, each user has a key pair say public key and private key. The public key is publicly known to all, whereas private key is kept secret by the user. The only public key is used in the communication while the private key is not transmitted.

Many drawbacks are also attached with the public key cryptography. In the public key infrastructure, a trusted certificate authority (CA) is needed that issue a certificate to bind the key pair (public key with the corresponding private key). For getting a certificate, the user gives their identity and public key of the CA. CA verifies their identity and public key, then issue a certificate issue to bind the key pair. This will create a certificate management problem. If a lot of members are involved in the communication, then a large number of certificates are needed. In this procedure, the cost needed the like certificate creation, transmission, verification and revocation is very heavy in certificate management. This creates an overload on the system.

The solution of the major drawbacks of public key cryptography, provide by the Shamir in 1984 by introducing identity based public key cryptography.

1.5.1 Basic function of public key cryptography

- Plain Text (P): plain text is the message for encryption that the sender wants to send the receiver.
- Public and private Key (K, P) : K is the public key which is known to all and P is the private key that kept by the user.
- Cipher Text (C): cipher text is the message after encryption.
- Encryption algorithm (EA): massage is to be encrypted by the sender with the encryption algorithm by using plain text, public key and corresponding secret key.
- Decryption algorithms (DA): massage is to be decrypted by the receiver with the decryption algorithm by using ciphertext and sender public key (in case of digital signature).

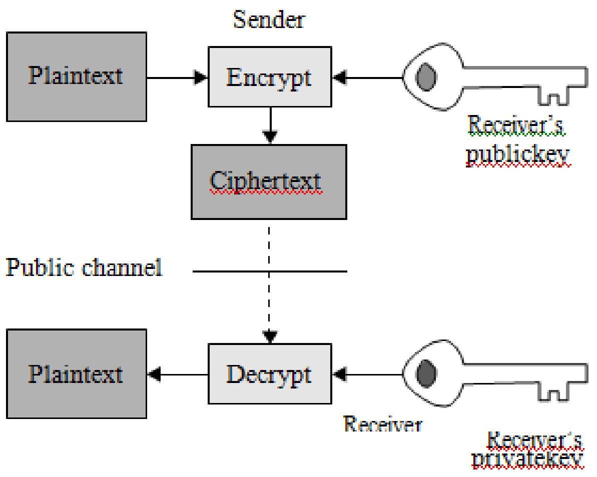

Fig 1.8: Public key encryption scheme

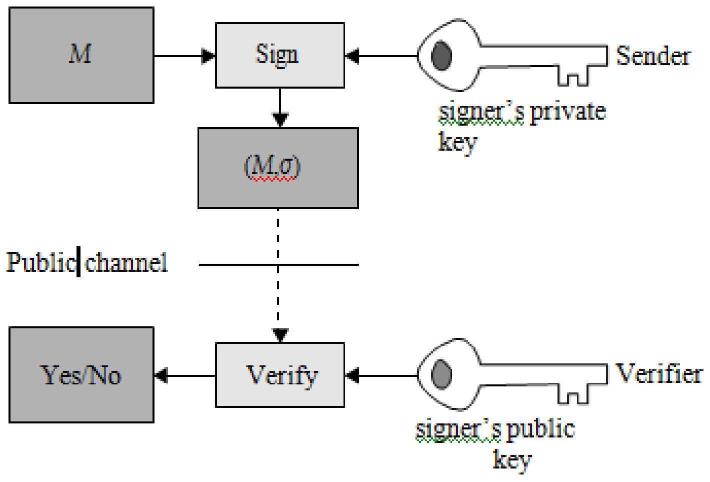

Fig 1.9: Public key digital scheme

1.6 Identity Based Public Key Cryptography

For solving the certificate management problem inherit in the public key cryptography Shamir introduced another cryptographic primitive say identity based public key cryptography.

Shamir [8] suggests onward Identity-based public key cryptography (ID-PKC) for shortening certificate management procedures of public key infrastructure (PKI). In which user's public key have just their email or telephone numbers. In the Identity-based public key cryptography (ID-PKC) third party say a private key generator (PKG) used to generate a private key. PKG initialize the system parameter and generate their public key and master key and kept their master key secretly by himself. Then PKG generate the private key of the user and transfer to the user by the secret channel. User can select their public key, according their suitable information like house number, street name, city name, phone number and email, etc. Now the user is able to use their operation like in encryption, digital signature or authentication. In case of encryption, user encrypts their private key generated by PKG and anyone can decrypt the ciphertext by the public key of the user. In case of digital signature, user generate a digital signature with their private key and the public key thereafter receiver can verify the signature with the public key of the sender.

In this complete event, a major drawback of this scheme came that the user's private key is generated by the third party say PKG. Although we take an assumption that PKG is a trusted party and it cannot be malicious. But in case PKG became malicious and he knows the complete private key of the user. Then no remains the meaning of the security of our algorithm and our system can be crush completely. This problem is known as a key escrow problem.

1.6.1 Algorithm for the identity based signature scheme

Algorithms	Who is Executing	Input	Output	Remarks
Master-Key-Gen	PKG	1^k, k is security parameter	$(mpk, msk), Params$	Keep msk secretly itself
Private--Key-Gen	PKG	mpk, user identity	User's Secret Key usk	Send usk to user via secure channel
Public-Key-Gen	User	Public parameter $Params$	upk	Depend on the selection on user.
Sign	Signer	Signing key (mpk, usk)	Signature σ	
Verify	Verifier	ID, upk σ on message m	Verification equation	If verification equation satisfied to then accept the signature otherwise reject

Solution of the key escrow problem inherits in identity based cryptography is suggested by the al-riyami and Peterson in 2004 say certificateless cryptography.

1.7 Certificateless Public Key Cryptography (CL-PKC)

Al-riyami and Paterson [9] proposed a new approach called certificateless public key cryptography (CL-PKC) to solve the key escrow problem of ID-PKC. The major drawback of the ID-PKC is that the private key is generated by the third party say PKG and he is aware of the user's private key. In CL-PKC, A third party is involved say key generation centre (KGC) which generates the user's partial-private key and private key is generated by the user with the help of the private key while corresponding public key of user generate by himself. Since KGC generate the partial private key not complete private key, whereas a private key is generated by the user. So KGC doesn't know about the private key of the user. By this procedure, the CL-PKC provides the solution of the key escrow problem inherit in ID-PKC.

In the original scheme suggested by the Al-riyami and Peterson have the seven algorithms say Setup, Partial-Private-Key-Gen, Private-Key-Gen, Set-Secret-Value, Public-Key-Gen, Signature, and Verification. The description the scheme is given below

Setup: KGC takes a security parameter 1^k as input and returns a master secret key α, master public key P_{pub} and publish the system parameter $params$.

Partial-Private-Key-Gen: KGC takes user's identity $\in \{0,1\}^*$, $params$ and master key α, as input and provides the user's partial private key psk_{ID}. KGC send the private key to the user via secure channel.

Set-Secret-Value: this algorithm is run by the user by taking input partial private key psk_{ID}. Then user sets a secret valuex_{ID}.

Private-Key-Gen: User takes the $params$, user's secret key x_{ID} as the input and precedes the user's private usk_{ID}.

Public-Key-Gen: User can choose their public key upk_{ID} after taking public parameter params.

Signature: Signer takes the public parameter $params,$ user's identity $\in \{0,1\}^*$, partial private key psk_{ID}. Then signer generates the signature σ on the corresponding message

Verification: This algorithm is run by the verifier. Verifier takes the input public parameter $params,$ user's identity $ID \in \{0,1\}^*$, user's public key upk_{ID} and the signature pair (m, σ) corresponding the message m thereafter verifies the signature if the signature is verified the accept the signature otherwise reject the signature.

Equivalent certificateless signature against Ai-riyami and Peterson signature scheme: al-riyami and Peterson [9] proposed seven algorithms as mentioned above in their digital signature scheme. Similar signature can be formed by eliminating the Set-Secret-Value algorithm. The result is same for both signature schemes. In the modified framework, six algorithms consist in the signature describe.

- Setup

- Partial-Private-Key-Gen
- Private-Key-Gen
- Public-Key-Gen
- Signature
- Verification.

Later on, it seems that the Private-Key-Gen and Public-Key-Gen algorithms are run by the user then no need to separate these algorithms. The user can generate the key pair public-private key pair in one step. In the new framework of certificateless signature, it consists five algorithms and merge the public-key-gen and private-key gen. Five algorithms framework is described as below in which four algorithm Setup, Partial-Private-Key-Gen, Signature and Verification are same as the original framework:

- Setup
- Partial-Private-Key-Gen
- User-Key-Gen: User runs this algorithm after taking the input partial private key generated by the KGC and the public parameter params and give output a key pair (usk_{ID}, upk_{ID}) and set usk_{ID}as private key and upk_{ID} as public key.
- Signature
- Verification.

In the working of certificateless signature scheme, the third party say KGC generates their public key and master key by taking input security parameter. Then KGC generates the partial private key of the user and send via a secure channel to the signer. Signer set their private key with the help of the partial private key thereafter signer sign a signature on the given message, with its private key and partial private key where verifier can verifies the signature with the help of the public key of the signer.

1.7.1 The formal definition of a CLS Scheme used in our work

A CLS scheme consists of five algorithms, called *Master-Key-Gen, Private-Key-Gen, User-Key-Gen, Sign, Verify*.

Table 1: Algorithms for CLS scheme

Algorithms	**Executes**	**Input**	**Output**	**Remarks**
Master-Key-Gen	KGC	1^k, k is security parameter	$(mpk, msk), Params$	Keep msk secretly itself
Partial-Private-Key-Gen	KGC	User identity, $Params$	Partial private key	Send securely to user via secure channel
User-Key-Gen	User	mpk, user identity	Public/Secret Key pair (upk, usk)	
Sign	Signer	Signing key (psk, usk)	Signature σ	
Verify	Verifier	mpk, ID, upk σ on message m	Verification equation	If verification equation satisfied to then accept the signature otherwise reject

1.8 Security Model of a certificateless signature scheme

Al-riyami and Peterson describe the two type of the security, say Type 1 security level and Type 2 security levels. There are two types of adversaries, A_1 andA_2. In general,A_1 and A_2 are involved in CLS scheme with different powers. $A_1$is an outsider attacker and $A_2$is a part of the system, say, malicious KGC, who is responsible for generating the partial-private key of the user. Description of power of adversaries is given below.

- **Adversary A_1:**$A_1$is capable to replace the public key of a user, but cannot obtain the master key of KGC.
- **Adversary A_2:** $A_2$is malicious KGC and can access the master key of KGC, but have no power to replace the user's public key.

Definition: A CLS/CL-AS scheme is said to be existentially unforgeable against adaptive chosen message and identity attacks, if both the adversaries A_1 and A_2 have negligible probabilities to forge the valid signature.

A_1 and A_2can access the following six oracles:

Create-User: While an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$. Oracle checks whether the proper entry corresponding to the target identity is available in the database. If found, then returns$PK_{ID_i}$to the adversary otherwise it executes *Reveal-Partial-private-key* and *Reveal-Secret-key* queries to find partial private key ppk_{ID_i}, private key $x_{i.}$ Thereafter, the Oracle computes PK_{ID_i} and inserts in the list $L = (ID_i, x_{i,} PK_{ID,} ppk_{ID_i})$. Finally,$PK_{ID_i}$returns to the adversary.

Reveal-Partial-Private-Key: While an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$, Oracle checks whether the proper entry corresponding to the target identity is available in the database. If found then returns $ppk_{ID_i}$to the adversary otherwise it returns $\perp$.

Reveal-Secret-Key: While an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$, Oracle checks whether the proper entry corresponding to the target identity is available in the database. If found then returns $x_i$to the adversary otherwise it returns $\perp$.

Replace-Public-Key: When an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$and private/public key pair$(x_i, PK^*_{ID_i})$. Then the oracle searches the list$L$, if proper entry corresponding to the target identity is not available in the database then no need to perform anything otherwise this oracle updates the list $L = \left(ID_i, x_{i,} PK_{ID,} ppk_{ID_i}\right)$to $L = (ID_i, x_{i,} PK^*_{ID}, ppk_{ID_i})$.

Sign: While an adversary submits a query on the message with target signer's identity$ID_i \in \{0,1\}^*$, Oracle executes one of the following activities.

i) Returns a valid signature σ_i without replacing private/public key pair if the target identityID_ihas been formed without swapping private/public key pair.

ii) Returns$\perp$ if target identity $ID_i$is not created.

iii) Returns the signature $(x_i, PK^*_{ID_i}, m_i)$ after replacing the public key.

We design two Games: Game I and Game II. Here, Game I and Game II are designed for A_1 and A_2 in CLS scheme, respectively [1].

Game I: τ_1 is the challenger/simulator that interacts with adversary A_1. This Game performs the following steps.

- Step1: τ_1 executes the *Setup* algorithm, which takes a security parameter k as input, produces a master key of KGC and a list of system parameters. Then τ_1 transfer the system parameters to A_1 while keeping the master key secret.
- Step 2: In this step, A_1 can submit *Reveal-Partial-Private-Key, Reveal-Secret-Key, Reveal-Public-Key, Replace-Public-Key* and *Sign* queries at any stage during the simulation in polynomial bound.
- Step 3: A_1 outputs a signature σ_i^* on a message m_i^* corresponding to a targeted identity ID_i^* with public key $PK_{ID_i^*}$.

A_1 successfully wins the game if any one of the following conditions is satisfied.

i) σ_i^* is a valid signature on m_i^* under ID_i^*.

ii) Oracle has never been performing the *Reveal-Partial-Private-Key query* for getting the partial private key corresponding to the targeted identity ID_i^* .

iii) *Sign* oracle has never been performed for m_i^* with the targeted identity ID_i^*.

Definition: A CLS scheme is called Type 1 secure if there does not exist any adversary A_1 who wins the Game I in probabilistic polynomial time bound with non-negligible advantage [1].

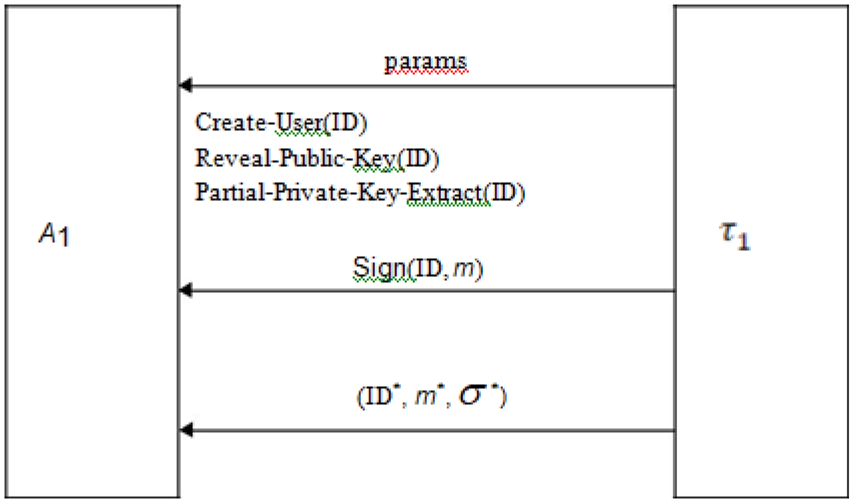

Fig 1.10: Security working against adversary A$_1$

Game II: τ_2 is the challenger/simulator that interacts with adversary A_2. This Game performs the following steps.

- Step1: τ_2 executes the *Setup* algorithm, which takes a security parameter k as input and produces a master key of KGC and a list of system parameters. Then τ_2 transfers the system parameters to A_2 while keeping the master key secret.
- Step 2: In this step, A_2 can submit *Reveal-Partial-Private-Key, Reveal-Secret-Key, Reveal-Public-Key, Replace-Public-Key* and *Sign* queries at any stage during the simulation in polynomial bound.
- Step 3: A_2 outputs a signature σ_i^* on a message m_i^* corresponding to a targeted identity ID_i^* with public key $PK_{ID_i^*}$.

A_2 successfully wins the game if any one of the following conditions is satisfied.

i) σ_i^* is a valid signature on m_i^* under ID_i^*.

ii) Oracle has never been executing the *Reveal-Secret-Key* for getting the secret key corresponding to the targeted identity ID_i^*.

iii) *Sign* oracle has never been executed for m_i^* under the targeted identity ID_i^*.

Definition: A CLS scheme is called Type 2 secure if there does not exist any adversary A_2 who wins the Game II in probabilistic polynomial time bound with non-negligible advantage.

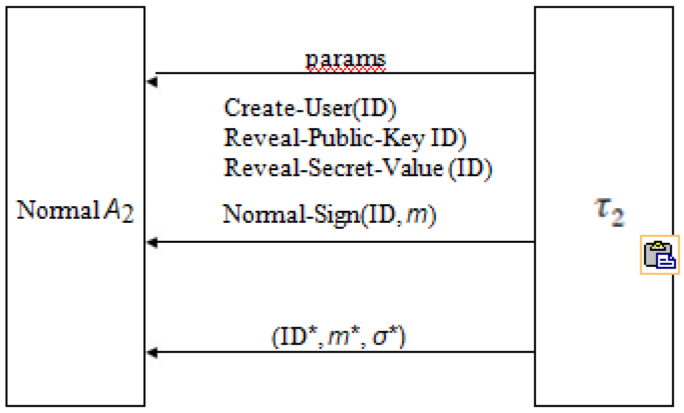

Fig 1.11: Security working against adversary A1

1.9 **Cryptanalysis**

In the network security, we study two most important concepts: first as we discuss above called the cryptography in which we make a cipher and cryptographic algorithms to secure our system from the all types of adversaries and other one is called the cryptanalysis in which we analysis the security of the previous cryptographic techniques, try breaking the existing ciphers.

In cryptanalysis, we try to find the mathematical technique for finding the weakness of the existing algorithms. Cryptanalysis is not the bad practices rather than it is in positive ways which checks the security of the existing scheme and find the loophole of the scheme thereafter it can be eliminated the weakness of the algorithm.

1.9.1 Some mathematical security attacks use in the CLAS

1.9.1.1 Honest but curious attack: An honest but curious attack is an attack where adversary A are restricted to following the protocol, but after the experiment is over, they may analyze the data they have received to try to recover other players' inputs. Once they have done so, they may return a result deviating from the regular computation.

1.9.1.2 Insider attack: This type of attack can apply to aggregate signature scheme where no one can identify the malicious signature. Some members participating inside in the communication become malicious and generate malicious signature. Verifier cannot identify the malicious signature on an individual level.

1.9.1.3 Universal attack: An adversary has no need the partial private key and private key to forge the security of the signature scheme.

1.9.1.4 Collision resistance attack: Collision resistant property defined that no signer groups holding the KGC together can generate a valid aggregate signature [56]. A dishonest user might be an internal user (one of the sharing user in the aggregate signature) or external user cooperates with malicious KGC to produce a valid forge aggregate signature.

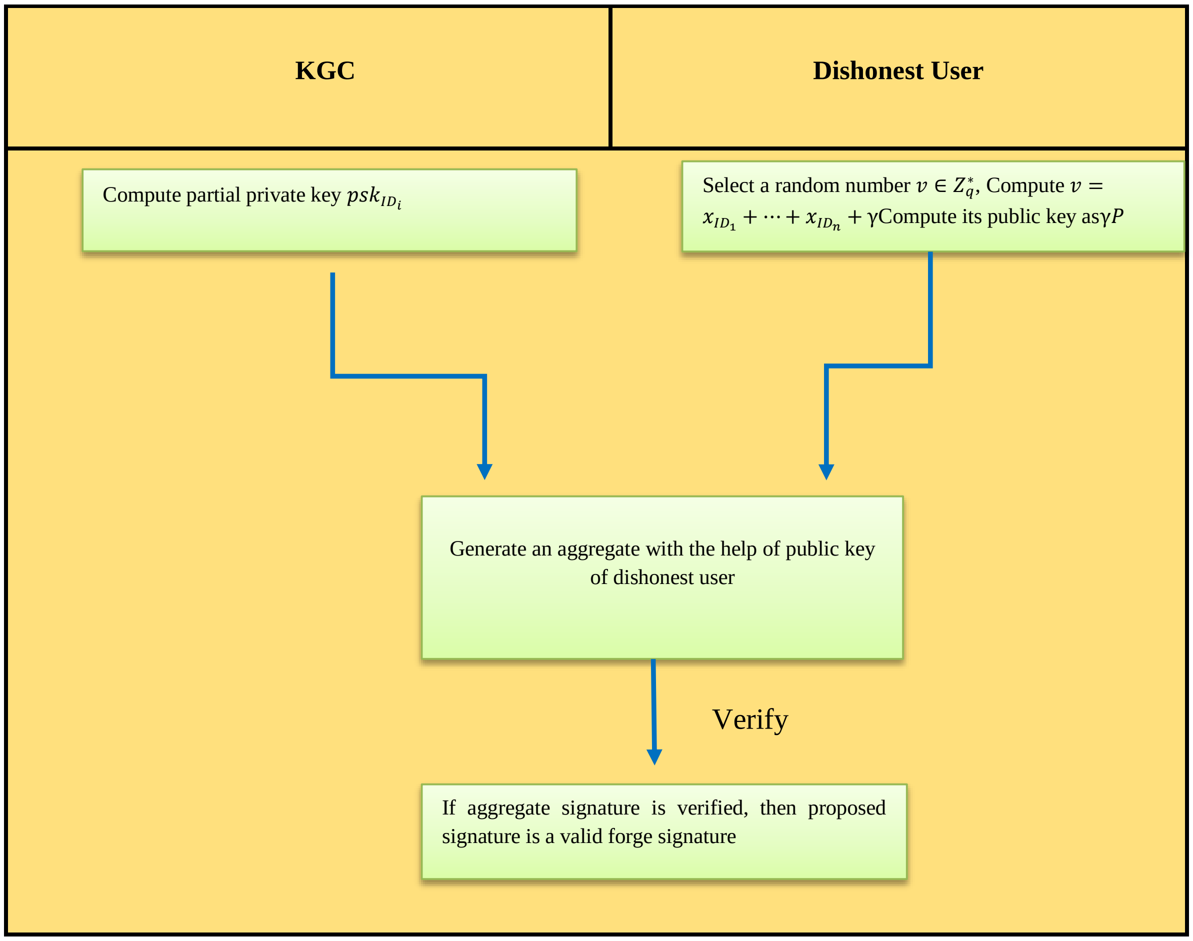
KGC
Dishonest User
Compute partial private key psk_{ID_i}
Select a random number $v \in Z_q^*$, Compute $v = x_{ID_1} + \cdots + x_{ID_n} + \gamma$Compute its public key asγP
Generate an aggregate with the help of public key of dishonest user
Verify
If aggregate signature is verified, then proposed signature is a valid forge signature

Fig 1.12: Pictorial view of collision resistance attack

Chapter 2

Mathematical Background

Number theory is about integers and their properties. It is dealing with the theory of numbers and is probably one of the oldest branches of mathematics. It is divided into several areas including elementary, analytic and algebraic number theory. Those are distinguished more by the methods used in each than the type of problems posed.

2.1 Division

If a and b are two integers not equal to 0 then we say that a divides b if there exist an integer c so that $b = ac$.

When $a$divides b we called that b is a multiple of a and a is a factor of b. We used the notation $a|b$ defines that a divides b.

The positive divisors of 20 are 1, 2, 4, 5, 10, and 20.
-5 | 30, 13 | 182, -3 |33, 17 |289, 17 | 0.

Subsequently, we describe some more properties of divisibility corresponding integers as follows.

2.1.1 Divisibility Theorems

For integers r, s, t it is true that

- If $r|1$, then r$= \pm 1$
- If $r|s$and s$|r$, then r$= \pm s$
- If $r|s$ and $s|t$ then r$|t$.
- if$r|s$and $r|t$, then $r|(s + t)$

Example: 2 | 4 and 4 | 8, so 2 | 8.

- if $r|t$, then $r|st$ for all integers t.

2.1.2. Prime Numbers

An integer p is called a prime integer greater than 1 if it has two positive divisors, 1 and itself. Therefore the maximum possible set of the factors or divisors should have only four integers $\pm p$ and ± 1. Prime numbers are of the greatest importance to certain cryptographic algorithms and most of the techniques used will not work without them.

2.1.3 Fundamental theorem of arithmetic:

Any positive integer $a \geq 2$, which can be either prime or can be expressed as the product of primes.

That is $a = p_1^{k_1} \times p_2^{k_2} \times p_3^{k_3} \times \dots \dots . . p_n^{k_n}$ where $p_1 < p_2 < p_3 < \cdots \dots \dots . . p_n$ and $k_n \geq 0$.

It can be seen from the definition of a prime number as mention above, that 1 is neither composite nor prime.

Examples: 20=4.5

56=2.2.2.7=2^3.7

Suppose n be a composite integer, then n contains a prime divisor less than or equal $\sqrt{n}$. .

2.1.4 The Division Algorithm

Let $\boldsymbol{a}$ be an integer and $\boldsymbol{b}$ positive integer. Then there exist two unique integers s and t, with $\mathbf{0} \leq \boldsymbol{t} < \boldsymbol{b}$ such that $\boldsymbol{a} = \boldsymbol{b}.\boldsymbol{s} + \boldsymbol{t}$.

In the above equation,

$\boldsymbol{a}$ is called the dividend,

$\boldsymbol{b}$ is called the divisor,

$\boldsymbol{t}$ is called the remainder,

$\boldsymbol{s}$ is called the quotient.

For example: if we divide 17 by 5, then we can write

16 = 5.3 + 1.

5 is the divisor,

16 is the dividend,

1 is called the remainder,

3 is called the quotient.

2.1.5 Greatest Common Divisors

Let a and b are the two integers with not equal to zero. The largest integer d with $d|a$ and $d|b$ is called the greatest common divisor (gcd) of $a$and b.

The gcd of a and b is denoted by $\gcd(a, b)$. gcd(32, 48)

The positive common divisors of 48 and 72 are 1, 2, 4, 8 and16 so gcd(32, 48) = 16.

2.1.6 Relatively Prime Integers

Two numbers a and b are called relatively prime if gcd(a, b) = 1.

2.1.7 Least Common Multiples

The least common multiple of the positive integers a and b is the smallest positive integer that is divisible by both a and b. We denote the least common multiple of a and b by lcm(a, b).

Examples: lcm(4, 7) =28.

2.2. Modular Arithmetic

Suppose a be an integer and m be a positive integer. We are defined by a mod m is the remainder when a divides m.

Examples:

10 mod 7 = 3

- 10 mod 7 = 4

9 mod 3=0

2.2.1 Congruence

Suppose a and b be two integers and m be a positive integer. We say that a is congruent to b modulo m if a-b is divides by m. $a \equiv b$ (mod m) is used to define that a is congruent to b modulo m.

Examples:

For verifying above definition 46 ≡ 68 (mod 11), we can check 11 | (46 – 68).

2.2.2 Properties of Congruence

Congruences have the following properties:

- $c \equiv d \ (mod\ m)$ if m|(c-d)
- $c \equiv d \ (mod\ m)$ iff $d \equiv c \ (mod\ m)$
- $c \equiv d \ (mod\ m)$ and $d \equiv e \ (mod\ m)$ imply $c \equiv e \ (mod\ m)$

Theorem: Let n be a positive integer. If c ≡ d (mod n) and e ≡ f (mod n), then c+e ≡ d+f (mod m) and ce ≡dc (mod m).

2.2.3 Modular Arithmetic Operations

According to the definition of the (mod) operator, that maps all integers to the set of integers {0, 1, …….. , (m - 1)}. We can perform some arithmetic operations within the elements of the set. These operations are called modular arithmetic operation**.**

Some modular arithmetic properties are shown as below:

1.[(c mod m) + (d mod m)] mod m = (c + d) mod m

2 .[(c mod m) - (d mod m)] mod m = (c - d) mod m

3.[(c mod m).(d mod m)] mod m = (c.d) mod m

2.2.4 Properties of Modular Arithmetic

Let Z_m be the set of nonnegative integers less than m:

Z_m = {0, 1,….. , (m-1), +}

Z_m is called the residue classes or set of residues mod n. To be further precise, each element in Z_m denotes a residue class. We can assign the residue classes, where (mod m) as [0], [1], ….,[m-2] , [m – 1]

[s] = {a: a is an integer, a ≡ s (mod n)}

The residue classes (mod 3) and (mod 2) are

[3] = { ….., -13, -9, -5, -1, 3, 7, 11, 15, 19, …… }

[2] = { … , -14, -10, -6, -2, 2, 6, 10, 14, 18, ….. }

In a residue class, the smallest nonnegative element among all the elements of the class is used to represent the residue class.

2.2.5 Modular Inverse

The concept of modular inverse is very important in ordinary arithmetic and cryptography. In number theory, any element is called the inverse of an element if we operate this element with corresponding element and we get identity in the end result. Identity can be different for different sets. It depends on the set and operation used in the set. In general two types of the inverse exist depends on the operation: (1) Multiplicative inverse (2) Additive inverse. 0 is the Identity element corresponding addition operation, whereas 1 is the Identity element corresponding multiplication operation. –A is the additive inverse of the element a where a belongs to the group and 1/a is the multiplicative inverse corresponding the element a. In case of arithmetic modulo m, a number x is a multiplicative inverse if it is a relatively prime to m i.e. gcd (m,x)=1.

Suppose $a \in Z_m$ and $x \in Z_m$

If $ax \equiv 1 \pmod m$ exist, then x is called the multiplicative inverse of a and denotes as a^{-1}.

2.2.6 Chinese Remainder Theorem (CRT)

Suppose $n_1, n_2, \ldots, n_r$ are relatively prime numbers, then consider a system of simultaneous congruence's as following

$$Y \equiv a_1 \pmod{n_1}$$
$$Y \equiv a_2 \pmod{n_2}$$
$$\ldots$$
$$Y \equiv a_r \pmod{n_r}$$

has a unique solution modulo $N = n_1 n_2 \ldots n_r$, which is given by

$$Y = \sum_{i=1}^{r} a_i N_i X_i \bmod N$$

Where $N_i = N/n_i$ and $X_{i=N}{}^{-1} \bmod n_i$, for $1 \le i \le r$.

Example

Consider a system of simultaneous Congruences equations

$$Y \equiv 5 \pmod 7$$
$$Y \equiv 3 \pmod{11}$$
$$Y \equiv 10 \pmod{13}$$

Then we got the unique solution $Y \equiv 894 \bmod 1001$.

2.2.7 Euler's Theorem

Mathematically, Euler's Theorem can be described as:

$a^{\varphi(n)} \equiv 1 \pmod m$, $\gcd(a, n) = 1$

Where, m is the modulus and a is any integer. The symbol $\varphi(n)$is called Euler's phi (or Totient) function which has positive integers $\le m$ and relatively prime to it. Some important points regarding $\varphi(n)$ describe as follow:

- The value of $\varphi(1)$is always equal to 1.

- If p is prime number, then $\varphi(p) = p - 1$ where $p - 1$ are positive integers $< p$ and relatively co prime to it.
- If p and q are two prime numbers with $n = pq$, then $\varphi(n) = \varphi(pq) = \varphi(p)\varphi(q) = (p-1)(q-1)$.
- If $n = p_1^{l_1} \times p_2^{l_2} \times p_3^{l_3} \times \ldots\ldots.. p_n^{l_n}$ then

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_1}\right)\ldots\ldots\ldots.\left(1 - \frac{1}{n}\right)$$

2.2.8 Fermat’s Little Theorem

Fermat’s Little Theorem is a special case of Euler’s theorem where p is a prime number. The theorem can be described as follows:

$a^{p-1} = 1(mod\ p)$, where n is a prime.

2.3 Concept related to Abstract Algebra

Let S is a non empty set. A binary operation is a mapping of the cross product of set S to set S as define as f : S × S → S

Such that for a, b, c ∈ S

f(a,b)=c

2.3.1 GROUP

Let G is a non empty set. An algebraic structure {G, ∗} is called group with binary operation (∗) if it satisfies 4 properties:

A1. Closure: For any two elements

c,d ∈ G, e = c∗d ∈ G

A2. Associativity: For any three elements

d,e,f ∈ G, (d ∗e) ∗ f = d∗(e∗f)

A3. Existence of Identity: There exists an Identity element

e∈ G such that ∀c ∈G, c∗e = e∗c= c.

A4.Existence of Inverse: Each element in G has an inverse i.e.

∀c∈ G ∃c^{-1}∈ G, such that c∗ c^{-1} = c^{-1}∗ c = e.

Abelian Group: A Group {G, ∗} is called an Abelian group, it satisfies one addition property say commutative with binary operation ∗, that is,

A5. Commutative: For any c, d ∈ G, c ∗ d = d ∗ c.

2.3.2 CYCLIC GROUP

A group G is called a cyclic group if every element of the group G can express a power a^k (where k is an integer) with a∈G. a is called the generator of a cyclic group G. Cyclic group is always abalian group. Order of a group can be finite or infinite. For example set of integer is a cyclic group with the generator 1 corresponding addition operation.

2.3.3 RING

Let R is a non empty set. An algebraic structure {R, +, ×} is called a ring if it satisfied following axioms with two binary operations say addition and multiplication:

1. R should be an abelian group corresponding addition operation such that it satisfies the all the axioms mention in A1 → A5. The identity element is 0 and the inverse of an element a of R is denoted as −a.

M1. Closure with multiplication: For any two elements c, d ∈ R, e = cd∈ R.

M2. Associativity with multiplication: For any elements c,d,e ∈ R, (c×d) ×e = c× (d×e).

M3. Distributive law: For any elements d, e, f ∈ R, c× (d + e) = c×d + c×e.

2.3.4 INTEGRAL DOMAIN

A Ring {R, +, ×} is a non empty set with two operations multiplication and addition is called integral domain, if it is satisfies the following properties:

M4. Commutative with multiplication:: For any c, d∈ R, cd = dc.

M5. Identity corresponding multiplication: There exists an element 1 in such that c1 = 1c = c for all c in R.

M6. Without zero divisors: If c,d ∈ R and cd = 0 then either d = 0 or c = 0.

2.3.5 FIELD

A Field {F,+,×} is a non empty set with two operations multiplication and addition if it satisfies the following properties:

1. Integral Domain (A1 −M6): It should be satisfies all the properties of the integral domain as mentioned above.

2. Inverse corresponding multiplication (M7): For each non zero element in F should posses its multiplicative inverse i.e.,

$$\forall d \neq 0 \in F, \exists d^{-1} \text{such that } dd^{-1} = d^{-1}d = 1.$$

2.3.6 FINITE FIELDS OF THE FORM GF (*p*)

We use the notation for finite field of order p^n is GF(p^n) and called Galois field named on the honor of the mathematician Galois. In cryptography, our special interest two type of the finite field. When n=1, we have the finite field GF (p) that have a different structure from the other finite fields and with n>1 we discuss in the next section.

2.3.7 Galois Fields

A field satisfies the axioms A1 → M7 as we discuss earlier. A field can have infinite order. For example set of real numbers R is a field under the addition and multiplication operations. In cryptography, however, our interest is not with the infinite fields because they have the memory limitations, etc. Cryptographers have interest with finite fields instead of infinite field.

It is noted that the order of a finite field should be in the power of a prime p^n where $n > 0$. The finite field of order p^n is denoted as GF (p^n).

We use mainly two Galois group such as GF(p) for some prime p and GF(2^n). (2 is selected as main prime of interest because of its computers operating in binary.

2.3.8 Finite Fields of Order p

We define the finite field of order GF (p) of order p, where p is the prime number, as the set Z_p of integers {0, 1, 2….. ,p - 1} with the arithmetic operations addition and multiplication modulo p.

2.4 Elliptic Curves

Elliptic curves are cubic curves of the form $t^3 = s^3 + as + b$ over the R^2 (R^2 is defined as R x R, where R is a set of real number) is defined by the point set (s,t) that satisfy the equation $t^3 = s^3 + as + b$, with the point O called point of infinity. O is the identity element corresponding additive operation. We represent the elliptic curve as E(R) for further use.

The following figure represents the pictorial view of an elliptic curve satisfying the equation $t^3 = s^3 + as + b$

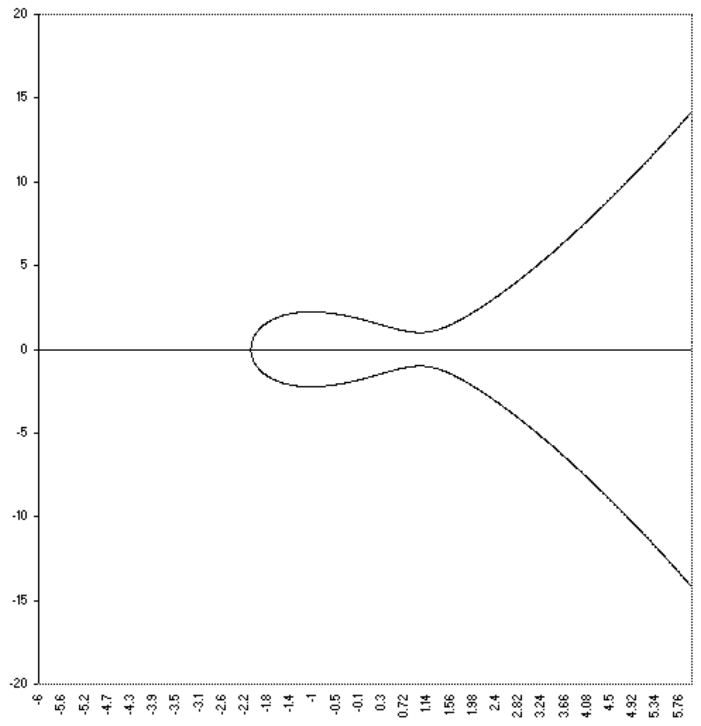
Fig 2.1: Elliptic curve over R^2: $y^2 = x^3 - 3x + 3$ [21]

2.4.1 Elliptic Curves over Finite Fields

Elliptic Curves over F_p

Let $E(F_p)$ be elliptic curve over a finite field F_p is denoted by the parameters a, b $\in F_p$ (where a, b satisfy the equation $4a^3 + 27b^2 \neq 0$), consists of the points set (s, t) $\in F_p$, satisfying the equation $t^3 = s^3 + as + b$. The possible set of points on $E(F_p)$ also include point O the point at infinity is the identity element under addition operation.

The operator is defined over $E(F_p)$ is addition. it can be easily verified that $E(F_p)$ forms an abelian group with respect to the addition.

The operation over addition in $E(F_p)$ is specified as follows.

- $Q + O = O + Q = Q$, $\forall$ $Q \in E(F_p)$
- If $Q = (s, t) \in E(F_p)$, then $(s, t) + (s, -t) = O$. (The point $(s, -t) \in E(F_p)$ is called the negative of Q and is denoted $-Q$)
- If $P = (s_1, t_1) \in E(F_p)$ and $Q = (s_2, t_2) \in E(F_p)$ and $P \neq Q$, then $T = P + Q = (s_3, t_3) \in E(F_p)$, where $s_3 = \lambda^2 - s_1 - s_2$, $t_3 = \lambda (s_1 - s_3) - t_1$, and $\lambda = (t_2 - t_1) / (s_2 - s_1)$,

i.e. the addition of 2 points can be shown as the point of intersection E(F_p) and the straight line which passing from both the points.

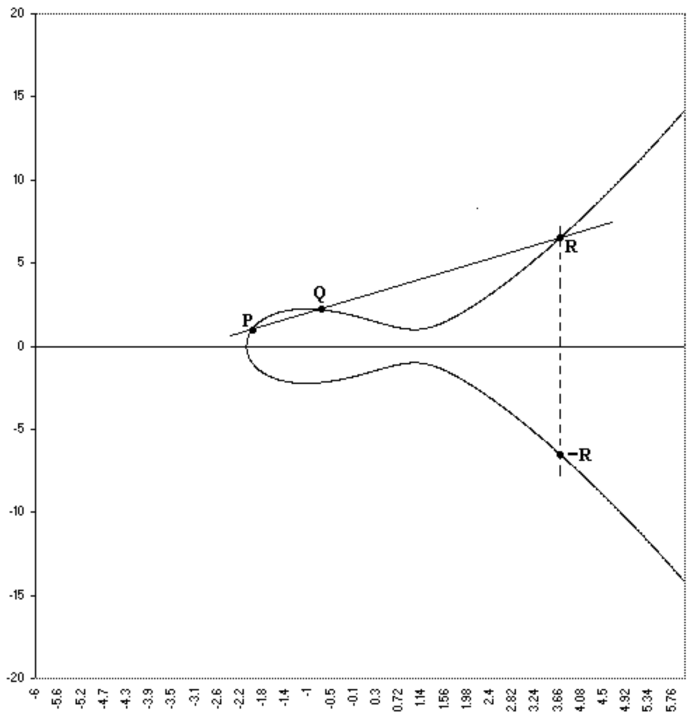

Fig2.2: Sum of 2 points P and Q on the curve $t^3 = s^3 + as + b$ *[21]*

- Let P = (s, t)∈E(F_p). Then the point Q = P + P = 2P = (s_1, t_1)∈ E(F_p), where $s_1 = \lambda^2 - 2s$, $t_1 = \lambda(s - s_1) - t$, where $\lambda = (3s^2 + a) / 2t$. We call this operation doubling of a point which can be shown as the point of intersection over the elliptic curve with the tangent at P.

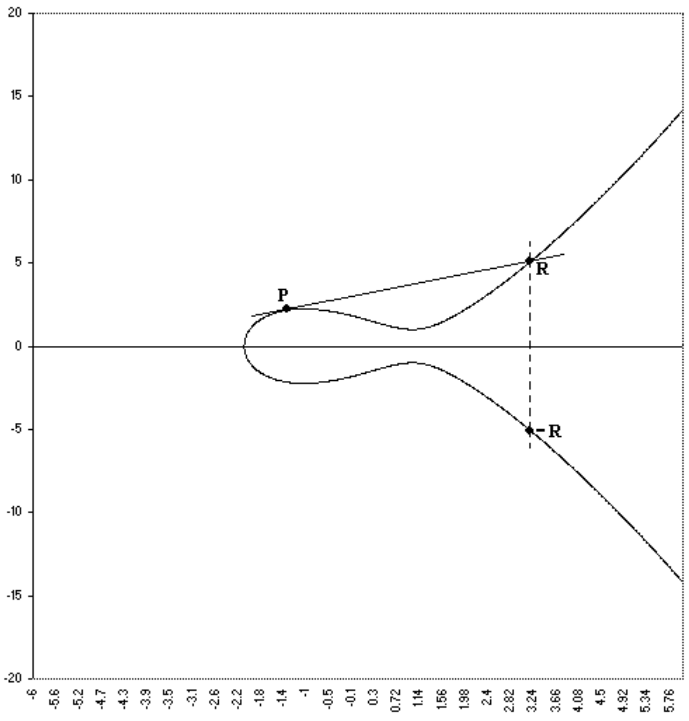

Figure 2.3: Doubling of a point P, R = 2P over the elliptic curve $t^3 = s^3 - 3s + 3$ [21]

Here, it is noticeable that the addition operation over elliptic curve $E(F_p)$ requires two multiplications, one inversion, six additions and one squaring. Similarly we can see, doubling a point over elliptic curve on $E(F_p)$ requires two multiplication, one inversion, two squaring, eight additions.

Consider the set $E(F_p)$ over addition. We can see that

- $\forall Q, P \in E(Fp)$, if $T = Q + P$, then $T \in E(F_p)$ (Closure axiom)
- $(P + Q) + R = P + (Q + R)$, $\forall R, Q, P \in E(F_p)$ (Associative axiom)
- $\exists O \in E(F_p)$, such that $\forall Q \in E(F_p)$, $Q + O = O + Q = P$ (Identity element axiom)
- $\forall Q \in E(F_p)$, $\exists -Q \in E(F_p)$ such that, $Q + (-Q) = (-Q) + Q = O$. (Inverse element axiom)
- $\forall Q, P \in E(F_p)$, $Q + P = P + Q$. (Commutative)

It can also be seen that the elliptic curve $E(F_p)$ forms an abelian group under addition operation.

2.4.2 Elliptic curves over F_{2^m}

Let E(F_{2^m}) be a elliptic curve over a finite field F_{2^m} is denoted by the parameters a, b $\in F_p$ (where a, b satisfy the equation $4a^3 + 27b^2 \neq 0$, $b \neq 0$), having the set of points (s, t) $\in$ F_{2^m} , satisfying the cubic equation $t^2 + st = s^3 + as + b$. The possible set of points on E(F_p) also include point O the point at infinity is the identity element under addition operation. Similar to elliptic curve E(F_p), addition operation is defined over the elliptic curve E(F_{2^m}) and it can be easily verify similarly as E(F_p) that even E(F_{2^m}) forms an abelian group with respect to addition operation.

The addition operation in E(F_{2^m}) is specified as follows.

- $Q + O = O + Q = Q$, $\forall$ Q $\in$ E(F_{2^m})
- If Q = (s , t) $\in$ E(F_{2^m}), then (s, t) + (s, – t) = O. (The point (s, – t) $\in$ E(F_{2^m}) and is called the negative of Q and is denoted –Q)
- If Q = (s_1, t_1) $\in$ E(F_{2^m}) and P = (s_2, t_2) $\in$ E(F_{2^m}) and Q$\neq$ P,

then T = Q + P = (s_3, t_3) $\in$ E(F_{2^m}), where $s_3 = \lambda^2 + \lambda + s_1 + s_2 + a$, $t_3 = \lambda (s_1 + s_3) + s_3 + t_1$, and $\lambda = (t_1 + t_2) / (s_1 + s_2)$, i.e. the sum of 2 points can be visualized as the point of intersection E(F_{2^m}) and the straight line passing through both the points.

- Let P = (s, t) $\in$E(F_{2^m}). Then the point Q = P + P = 2P = (s_1, t_1) $\in$E(F_{2^m}), where $s_1 = \lambda^2 + \lambda + a$, $t_1 = \lambda (s + s_1) + s_1 + t$, where $\lambda = s + (s / t)$. This operation is also called doubling of a point and can be visualized as the point of intersection of the elliptic curve and the tangent at P.

We can notice that addition over E(F_{2^m}) requires one inversion, two multiplications, one squaring and eight additions. Similarly, doubling a point on E(F_{2^m}) requires one inversion, two multiplication, one squaring and six additions.

Similar to E(F_p), consider addition under E(F_{2^m}),

- $\forall$Q, P $\in$ E(Fp), if T= Q + P, then T $\in$ E(F_p) (Closure axiom)
- (P + Q) + R = P + (Q + R) , $\forall$R, Q, P $\in$ E(F_p) (Associative axiom)

- $\exists O \in E(F_p)$, such that $\forall Q \in E(F_p)$, $Q + O = O + Q = P$ (Identity element axiom)
- $\forall Q \in E(F_p)$, $\exists -Q \in E(F_p)$ such that, $Q + (-Q) = (-Q) + Q = O$. (Inverse element axiom)
- $\forall Q, P \in E(F_p)$, $Q + P = P + Q$. (Commutative axiom)

Thus we see that $E(F_{2^m})$ forms an abelian group under addition.

2.4.3 Elliptic Curve: Some useful Definitions

- **Scalar Multiplication:** For an integer s and a point P on the elliptic curve, the scalar multiplication defined over the elliptic curve is sP as the result of adding Point P to itself s times.
- **Order:** Order of a point P defines over the elliptic curve is the smallest integer t such that

tP = O. Further if a and b are two integers, then aP = bP if and only if $a \equiv b \pmod t$).

- **Curve Order:** curve order is defines as the number of points on the elliptic curve and is denoted #E.

2.5 Bilinear pairing

Bilinear Map: Suppose G_1, G_2be two groups, where G_1 an additivecyclic group with a generator P and G_2be multiplicative cyclic group with same order q as well as G_1. Then a map $e: G_1 \times G_1 \rightarrow G_2$ is called an admissible bilinear mapping if it fulfilled the following properties:

- Bilinearity: For every $P, S, T \in G_1$, $e(P, S+T) = e(P,S)e(P,T)$ and for every $,t \in Z_q^*$, $e(sP, tP) = e(P,P)^{st} = e(stP, P) = e(P, stP)$.
- Non-degenerate: $(P,P) \neq 1$.
- Computability: $\exists$ $P, Q \in G_1$, then there exist an algorithm to compute $e(P,Q)$ for all $P, Q \in G_1$.

2.5.1 Some properties of Bilinear pairing

Some properties of Bilinear pairing are given below:

I. $e(P, 0) = 1$ and $e(0, P) = 1$ for all $P \in G_1$
II. $e(P, -Q) = e(-P, Q) = e(P,Q)^{-1}$ for all $P, Q \in G_1$
III. $e(P,Q) = e(Q,P)$ for all $P, Q \in G_1$
IV. Let $S \in G_1$, If $e(P,Q) = 1$ for all $Q \in G_1$ then $S = 0$.

CHAPTER 3

Literature Review

In public key cryptography, Digital signature is a key feature that assures authenticity, integrity and non repudiation in the network.

3.1 Some basic digital signature scheme

3.1.1 RSA Signature Scheme

Algorithm of RSA digital signature describes as follows:

KEY-GEN: Private key and public key of user creates in this step as follows.

i) Randomly select two prime numbers p and q such that$|p| \approx |q|$.
ii) Calculate $N = p \times q$
iii) Calculate $\emptyset(N) = (p-1)(q-1)$.
iv) Randomly select an integer $e < \emptyset(N)$ such that $gcd\left(e, \emptyset(n)\right) = 1$, then calculate the integer d such that $e \times d \equiv mod\ \emptyset(N)$.
v) Generate (N, e) as the user's public key and kept d as the user's private key.

SIGNATURE: for generating a signature$M \in Z_N^*$, the signer generates the signature $\sigma = M^d mod(N)$.

VERIFICATION: Verifier can verify the signature pair (M, σ) with the verification equation $M = \sigma^e mod(N)$, if the equation is satisfied, then accept the signature otherwise reject the signature.

3.1.2 Elgamal signature scheme [1]

The elagamal signature scheme is described as follows:

KEY-GEN: Private key and public key of user creates in this step as follows.

i) Select a random multiplicative generator g of Z_p^*. Where p is a random prime number.
ii) Select a random number $1 \leq x \leq p-2$ and set as the private key.

iii) Calculate the public key by $y = g^x (mod\ p)$.

iv) Generate (p, g, y) as the public key, and set x as the corresponding private key.

Signature: for generating a signature on the message $\in Z_p^*$. the signer choose a random number $1 \leq k \leq p - 2$ and generate a signature pair (r, σ) such that

$r = g^k mod(p), \sigma = k^{-1}(m - x \times r)(mod(p) - 1)$.

Verification: Verifier can verify the signature pair (r, σ) with the verification equation $y^r \times r^\sigma = g^m(mod(p))$, if the equation is satisatisfied, then accept the signature otherwise reject the signature.

3.1.3 Schnorr signature scheme [1]

The Schnorr signature scheme is described as follows:

KEY-GEN: Private key and public key of user creates in this step as follows.

i) Select two prime numbers p and q such that $q/(p - 1)$

ii) Select an element randomly $g \in Z_p^*$.

iii) Choose a hash function $H: \{0,1\}^* \rightarrow Z_q^*$.

iv) Choose a number $x \in Z_q^*$ randomly and set it as the private key of the user and calculate $y = g^{-x}(mod\ p)$ then set as user's public key. Publish (p, q, g, y, H) as the public key and key x as a private key of the user.

Signature: For generating the signature on the message$M$, the signer selects a random number $k \in Z_q$. And generate a signature such that $r = g^k mod(p)$, $e = H(m||r)$ and $s = k + x \times e(mod\ q)$.

Verification: Verifier can verify the signature pair $(m, (e, s))$ with the verification equation $r^{'} = g^s \times y^e mod(p)$ and $e^{'} = H(m||r^{'})$, if the equation is satisfied, then accept the signature, otherwise reject the signature.

Shamir [8] proposed an Identity Based public key cryptography (ID-PKC) signature scheme that solve the certification problem arise in public key cryptography and has no need of certification for binding private/public key pair. In ID-PKC, the user chose its public key such as their driving license, phone number, address or any other unique identity. Third party say, private key generator (PKG) involves in ID-PKC, generates the private key of the user. Although we assume PKG as a trusted party, but one possibility arises, if PKG became malicious who is responsible for generating the private key of the user creates key escrow problem. Key escrow problem defines, if PKG, who is responsible for generating the private key of the user becomes malicious that can break the security very easily with the help of the private key. In 2003, Al riyami and Peterson [9] recommend a solution to solve Key escrow problem inherit in ID-PKC. Al riyami and Peterson [9] proposed a certificateless signature scheme (CLS), in which key generation center (KGC) is the third party which produces the partial private key of user instead of the private key and private key is generated by the user with the help of partial private key. In this case KGC doesn't have knowledge of private key directly.

Boneh [10] introduce the concept of the aggregate signature scheme in Eurocrypt 2003.In the aggregate signature scheme, aggregator collects all individual n signatures and aggregates them to produce a compact signature. Aggregate signatures are very beneficial in practical real life application where bandwidth limitation creates a big issue such as ad-hoc networks, wireless sensor networks, the internet of things and an endless list. Certificateless aggregate signatures takes the advantage of certificateless and aggregation models, it generates the solution of certificate problem and decreases the computations and restrict the bandwidth.

In recent years, a lot of certificateless signature schemes [12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 23, 24, 25, 27] have been proposed by researchers. Zhang and Zhang [15] suggested a CL-AS scheme and claims that their scheme if existentially unforgeable against adaptive chosen message and identity attacks but Shim [56] found insecure Zhang and Zhang [15] CL-AS scheme against collision resistant attack .Xiong et al's [17, 18] presented an efficient CL-AS with constant pairing and proves that the proposed CL-AS scheme is secured against concrete attacks in random oracle model but unfortunately Zhang et al's [34] found that their scheme is fails to protect against collision inside attacks. Yum and Lee [31] describe a new CLS scheme and claims to be secure against identity and adaptive chosen message attack, thereafter Hu et al [32] demonstrate that

Yum and Lee [31] CLS is fails to secure against the type 1 attack. Gorthala and saxena [48] proposed a new pairing based CLS scheme in random oracle model. Gong et al. [48] proposed two certificateless aggregate signature (CLAS) schemes which are found insecure by [5]. Zhang et al. [15, 20] suggested two new CLAS schemes using the bilinear pairing, but proposed scheme is not so much efficient having many pairing operations. Xiong et al. [17] proposed an efficient CLAS scheme and demonstrate that their scheme is unforgeable against some concrete attacks. Xiong et al. [18] scheme is very efficient in real life, unfortunately it fails to secure against type 2 adversary [28, 34, 35, 36]. Chen et al. [14] presents an efficient CLAS scheme and show that their scheme is secure against type 1 and type 2 adversaries, unfortunately Zhang [20] discover a universal adversary different from type 1 and type 2 adversaries forge the proposed scheme by [14]. Many other security attacks offered by researchers on CLS schemes. Deng *et al.* [25] proposed an efficient CL-AS scheme unfortunately we found that their CL-AS scheme is not secure against collision resistance attack.

Zhang et al. [15] proposed CLS schemes based on elliptic curve and verification algorithms have four pairing computation. Yep et al [8] proposed an improvement having only two pairing operation. Z. Eslami and N. Pakniat [13] propose a concrete certificateless aggregate signcryption scheme which is based on Barbosa and Farshim's certificateless signcryption scheme and proves that their CLAS scheme is secure under the gap Bilinear Diffie–Hellman. Gong et al [48] present two certificateless aggregate signature scheme in which first CL-PKC scheme reduces the cost of communication and signer –side calculation and second kind of CL-PKC minimizes the storage but sacrifices the communication. Gong et al also proves that their schemes are secure in random oracle model.

Zhang and Wong [29] proposed an efficient CLS scheme from pairings. Xu et al. [22, 23] proposed two CLS schemes for mobile wireless cyber-physical systems, and claim for high efficiency and verifiable security. Xiong et al [18] proposed a CLAS scheme in which have no need of synchronization for the signers and is secure under standard computational Diffie-Hellman assumption. Xiong et al [18] proves that their CLAS scheme is secure against super type II adversary and describe their scheme is very useful for vehicular ad-hoc network because of no need of synchronization for the signer. The

Xiong et al [17] scheme is useful for mobile computation where limited computation and storage capacities of the devices are a big challenge. Xiong et al. [18] proposed a certificateless aggregate signature with constant pairing computations which is convenient for Ad-hoc Networks and claims to be secure in the computational D-H problem in random oracle. Cheng et al found in the cryptanalysis of Xiong et al proposed CLAS scheme that it is insecure against the malicious-but-passive attack and honest-but-curious attack. Cheng et al gives an improvement CLAS scheme and by comparing performance analysis they prove that their scheme is a much efficient scheme.

He et al. [40] proposed an improved anonymous authentication scheme for wireless body area network. Horng et al. [59] used the batch verification in their proposed CL-AS scheme which reduces the computational cost taken in the verification process of the signature. Recently, He and Zeadally [64] proposed an authentication protocol for Ambient Assisted Living (AAL) system, which provides the healthcare monitoring and tele-health services by leveraging information and communication technologies. He et al. [43] construct an efficient certificateless public auditing (CLPA) scheme for cloud-assisted wireless body area network. Many architectures constructed by the researchers for HWSN [42, 43, 44].

Huang et al [10] classified the adversaries on behalf of their potential and make three kinds of adversaries called strong adversary, super adversary and normal type adversary associate with the type I and type II adversary in the proposed certificateless aggregate scheme. Huang et al presents two schemes, for which he et al claims their scheme is secure against normal type I and super type II adversaries and second scheme is secure against super type I and super type II adversaries.

Liu et al. [23] proposed an efficient certificateless aggregate signature scheme in which he claims that it is unforgeable against adaptive chosen-message attacks. But it is found insecure against ordinary passive attack and malicious active attack by Yulei Zhang and Caifen Wang [29].

Malhi and Batra et al [22] presents a certificates aggregate scheme for Vehicular Ad-hoc Networks. They claim that in their scheme, RSU generates the pseudonyms after this RSU identifies the user’s identity and generates the corresponding pseudonym. KGC have no full control on pseudonyms and cannot forge the signatures.

Horng et al [59] proposed an efficient certificateless aggregate signature scheme with achieving conditional privacy preservation in vehicular ad-hoc networks, in which every message generated by any vehicle associate with a pseudo identity. In proposing schemes Horng et al used four entities two for trust authorities (TA) and one for trace authority and one for key generation center with the assumption that 1) TRA and KGC are always trusted. 2) Each vehicle has GPS for obtaining time information. 3) Every vehicle furnished with a tamper proof device. Horng's et al [59] proposed schemes is used for vehicular adhoc network that is secure under adaptive chosen-message attacks and attains authentication, identity privacy, message integrity, preservation and traceability. Recently Hou et al [24] proposed an CLAS scheme and demonstrate that their scheme is secure in random oracle model but this scheme is found insecure by applying a concrete attacks. Deng et al [26] found insure Hou et al [24] scheme and also construct a new CLAS scheme and provide a formal security proof.

3.2 COMPARISON BETWEEN VARIOUS EXISTING CLAS SCHEME

In this section we compare some popular existing algorithms and give a detailed discussion about them. We present a table of various CLAS algorithms used by researchers. The table is prepared on behalf of scheme performance, such as sign cost, verification cost and aggregate verification cost.

Table 3: Comparison between various existing CLAS Scheme

S. No.	CL-AS Scheme	Point Multiplication Cost	Verify Cost	Aggregate Verify	Security
1	Z.Zhang [19]	3S	4P	3P	Yes
3	Zang and Zang et al [15]	3S	4P	(n+3)P	No [56]
4	L. Zhang et al [57]	5S	5P+2S	5P+2nS	Yes
5	First scheme in Gong et al [48]	2S	3P	(2n+1)P	Yes
6	Second scheme	3S	3P	(n+2)P+	Yes

	in Gong et al [48]			nS	
7	Hang Tu et al [35]	3S	4P+2S	4P+2nS	Yes
8	Lin Cheng et al [36]	4S	3P+2S	3P+2nS	Yes
9	Z. Eslami, N.Pakniat [13]	3S+1P	4P	(n+3)P	Yes
10	Chen et al [14]	2S	3P+1S	(n+2)P+ 1nS	Yes
11	Hu Xiong et al [17]	3S	3P+3S	3P+3nS	Yes
11	Hu Xiong et al [18]	3S	3P+2S	3P+2nS	No [28, 34, 35, 36]
12	He et al [28]	3S	3P+2S	3P+2nS	Yes
13	Liu et al [27]	3S	3P+2S	3P+2nS	No
14	A. Malhi and S. Batra [22]	3S	3P+3S	3P+3nS	Yes
15	J. Deng et al [25]	4S	3P+3S	3P+3nS	Yes
16	Horng [59]	3S	3P+1S	3P+1nS	Yes
17	Hou et al [24]	3S	3P+1S	3P+1nS	No [26]

n: Positive Number,S: Unit of scalar point multiplication Cost, P: Unit of Pairing Cost

Discussion: In this subsection we discuss a healthy discussion on the existing algorithm which have mentioned above. Mostly three types of the operation called hash function, scalar point multiplication, pairing operation used by the researcher in their CLAS scheme. Some researcher used pairing scheme and pairing free CLAS scheme. Pairing operation is very costly in all operations. Hash operation is very efficient and cost of the hash function is very low. Cost of Scaler point multiplication lies between hash function and pairing operation that is more than hash function and less than pairing operation. So we conclude that if any scheme having more pairing operation is not an efficient scheme.

It may be possible that any scheme is efficiently on behalf of operation used, but not secure and attacked by other researcher, and then this scheme is not efficient. L. Zang and F.Zang [15] used 3 scalar point multiplications in signature algorithm and 4 pairing operation in verifying algorithm in CLS scheme while they used (n+3) pairing operation in aggregate verifying algorithm in CLAS scheme which makes it very costly due to the high cost of pairing operation in aggregate verifying. Kyung-Ah Shim [31] found insures the L. Zang and F. Zang[15] CLAS scheme by applying collision resistant attack. Zhang et al [57] proposed a CLAS scheme with 5 scalar point multiplications in signature algorithm and 5 pairing operation with 2 scalar point multiplications in verifying. Zhang et al [57] used 5 pairing operation in their CLAS scheme which is very costly and time consuming, so Zhang et al [57] scheme is not so much efficient. Gong et al [48] proposed two CLAS scheme which is also very costly because Gong et al [48] used a lot of pairing operation in both of CLAS scheme. Xiong et al [18] used 3 paring and 2 scalar point multiplications operations in his CLAS scheme, but much type of attacks has applied for their CLAS scheme. Lin Cheng et al [14], D. He et al [29], Hang Tu [22] et al proposed an improved CLAS scheme on applying attack on Xiong et al [18] CLAS scheme. Liu et al [27] used 3 scalar multiplications in signature and 3 pairing and 2 scalar point multiplications in the proposed scheme. Zheng and Wang [29] prove that Liu et al [27] scheme found insecure by applying security attack type 2 adversary A_2.In another Scheme of Xiong et al [18] used 3 scalar point multiplication in signature and 3 pairing operation with 3 point scalar point multiplication in verifying scheme, also have no found attack on their scheme. So Xiong et al [18] is a better scheme in computation. Z. Eslami and N.Pakniat [13]presented a CLAS scheme with 3 scalar point multiplications and 1 pair operation on signature stage and 4 pairing operation contains in verifying algorithms, but it has (n+3) pairing operation in their aggregate signature verifying which made this scheme not so much feasible scheme. Malhi et al [22] and Horng et al [59] proposed their scheme for Vehicular Ad-hoc Networks. Horng et al [59] used scaler 3 point multiplications in signature and 3 scaler point multiplications and 1 pairing in verifying and 3 pairing and 1 ns in aggregate verifying and also show results in simulating environment. Malhi [22] et al used scalar 3 point multiplications in signature and 3 scalar point multiplications and 3 pairing in verifying and 3 pairing and 3 nS in aggregate verifying in their CLAS scheme. These schemes are secure because no attacks having applied till now while comparing between Malhi et al [22] and Horng et al [59] Schemes, Horng et al [59] CLAS scheme is much more efficient rather than Malhi et al [22] CLAS scheme on behalf of operation applied by both of them. Hou et al [24] proposed a CLAS scheme which is found insure by Deng et al [26]. Deng et al [25] also proposed an improved CLAS scheme based on Hou et al [24] CLAS scheme. Hou et al [24] used 3 scalar point multiplications in signature algorithm and 3 pairing operation with 1 scalar point multiplication in verifying algorithm while Deng et al [25] used 3 scalar point multiplications in signature algorithm and 3 pairing operation with 3 scalar point multiplications in verifying algorithm. Deng et al [25] proposed CLAS is much more secure and more efficient rather than Hou et al [24] while Hou et al [24] CLAS used less operation.

Chapter 4

Objectives and Methodology

We study a lot of different types of digital signature schemes like blind signature, partially blind signature, aggregate signature, ring signature, compact signature, pairing and without pairing based signature, a group based signature, thrashedhold signature, proxy signature and a lot of signature techniques. Descriptions of some of the techniques are mentioned below:

- **Partially blind signature**: Partially blind signature is approximately same as blind signature, Signer and the user have agreed on some common information for ex. In the online cash system

- **Aggregate Signature**: Aggregate signatures allow aggregating an an individual signature, which takes the benefit of limited bandwidth and reduce the computational overhead.

- **Ring signature**: In the ring signature, group of n signer involves signer sign the message on behalf of any user but verifier can't understand who the singer is.

- **Compact signature**: length of the signature is short in the compact signature.

- **Pairing based signature**: signature has been created based on bilinear pairing.

- **Signature without pairing**: as above mention pairing cost of the signature is very costly, so we can create a signature without pairing, in which pairing is not involved.

- **Thrashedhold signature**: in this signature a group of signer generates the signature and every user has some responsibility for generating the signature.

- **Group signature**: in the group signature, on signer generates the signature on behalf of the group.

- **Proxy signature**: User sign the signature on behalf of the original signer. Anyone can verify proxy signature if it knows about the public key of original signer and proxy signer.

After studying these techniques, we study the existing schemes derived from the contemporary researchers. We decide our goals to find the security leaks of the previous scheme by applying some concrete attacks. Also, we introduce certificateless signature schemes for different applications like vehicle adhoc network and healthcare wireless sensor network which are more efficient in the appropriate environment. Our objectives of the thesis are discussed below:

4.1 Objectives

- Study the literature of many techniques such as Public key cryptography, ID based Signature scheme, Certificateless Signature Scheme.
- To study many cryptographic primitive under digital signature schemes such as Blind Signatures Partially blind signature, Aggregate, Ring signature, Compact signature, Pairing based signature, Signature without pairing, Thrashedhold signature, Group signature, Proxy signature, Policy based, Undeniable signature, leakage free, Strong designated verifier, Multisignature, Dual Signature
- Find the limitation of the existing model proposed by the researchers.
- Find the flaws in the previous schemes by cryptanalysis and remove the security leaks of previous schemes.
- Design our certificateless signature scheme with elliptic curve cryptography.
- Design a signature scheme for VANET and one for Healthcare Wireless Sensor Network.
- Prove the security of our scheme by using NP hard problem such as computational Diffie-Hellman problem, discrete logarithm problem.

- We use the Random Oracle Model for giving mathematical proof.
- Make mathematical models for certificateless aggregate signature scheme.

4.2 Methodology

In the methodology, we mention below some points by which we achieve our goals.

- To use mainly latest 5 years research papers regarding a digital signature scheme to complete literature review.
- Study the literature review to find the flaws of previous existing digital signature scheme.
- Do the cryptanalysis of the previous existing schemes and try to give the improvement of their schemes.
- Make the mathematical models to design the certificateless signature scheme for various applications.
- Use Random Oracle Model to prove the provable security of our schemes.

Chapter 5

CRYPTANALYSIS OF EXISTING CERTIFICATELESS SIGNATURE SCHEMES

Cryptanalysis is a science of network security in which we analyze the security previous existing schemes by applying some mathematical technique. In this chapter, we have done a cryptanalysis of a lot certificateless signature schemes.

5.1 Preliminary Although we have discussed preliminaries in the last section, but we have given some definition below as the necessity of the chapter.

5.1.1 Bilinear Map: Suppose G_1 be an additive cyclic group with a generator P and G_2 be multiplicative cyclic group with same order q as well as G_1. Then a map $e: G_1 \times G_1 \to G_2$ is called an admissible bilinear mapping if it satisfied the following properties:

Bilinearity: $P, A, B \in G_1$, $e(P, A+B) = e(P, A)e(P, B)$ and for every $x, y \in Z_q^*$ $e(xP, yP) = e(P, P)^{xy} = e(xyP, P) = e(P, xyP)$

- *Non-degenerate*: $e(P, P) \neq 1$
- *Computability*: $\exists A, B \in G_1$, then there exists an algorithm to compute $e(A, B)$ for all $A, B \in G_1$

5.1.2 Security Models

In [9] security model, two types of adversaries are involved in the CLAS scheme, adversary A1 and adversary A2 have different attacking powers which restrict them. A1 can replace the public key of the user on behalf of its false public key, but cannot access the master key of the user whereas A2 has the power to access the master key of KGC but cannot modify the public key of the user.

- *CreateUser*: When submitting a query of identity $ID_i \in \{0,1\}^*$ on message, then the oracle returns the corresponding public key. If it is not present in the list, then first it create a user's public key and update list thereafter returns public key upk_{ID_i} .

- *RevealPartialkey*: When submitting a query of identity $ID_i \in \{0,1\}^*$ on message m_i, then the oracle returns corresponding partial private key psk_{ID_i}.
- *RevealSecretKey*: When submitting a query of identity $ID_i \in \{0,1\}^*$ on message m_i, then the oracle returns corresponding secret key usk_{ID_i}.
- *ReplacePublicKey*: When submitting a query of identity $ID_i \in \{0,1\}^*$ on message m_i and user's private/public key pair $(usk^*_{ID_i}, upk^*_{ID_i})$. Then oracle searches in the list, if the corresponding identity found, then update the entry $(usk^*_{ID_i}, upk^*_{ID_i})$ with replacing the previous entry .
- *Sign*: When submitting a query of identity $ID_i \in \{0,1\}^*$ on message m_i, then the oracle performs one of the three cases.

1) Returns a valid signature σ_i without replacing, if ID_i has been created without replacing private/public key pair.
2) Returns $\perp$, if ID_i is not created.

The oracle returns the sign $(usk^*_{ID_i}, upk^*_{ID_i}, m_i)$ after replacing the private/public key pair.

5.2.1 Game Plan: ς is a simulator who communicates with the adversary A. This Game takes three phases for completion.

Phase 1: in the phase 1, ς takes an input parameter k and set the master key/ public key of KGC and generates the public parameters $params$. ς sends the public parameter to the adversary and keep master key secrets.

Phase 2: In this phase, A can submit *revealpartialkey, revealsecretkey, revealpublickey, replacepublickey* and *sign* queries at any time during the simulation.

Phase 3: A generate a signature σ^*_i on a message m^*_i corresponding to a targeted identity ID^*_i with public key $upk^*_{ID_i}$.

A wins the game if any one of the following conditions is satisfied.

iv) σ^*_i is a valid signature on message m^*_i under targeted identity ID^*_i corresponding public key $upk^*_{ID_i}$.

v) Sign has never been submitted on message m_i^* under targeted identity ID_i^*.

5.3 Cryptanalysis of Deng et al CLS and CLAS scheme:

In this subsection we have done cryptanalysis of Deng et al CLS and CLAS scheme.

5.3.1 Review of the Deng et al [25] CLS Scheme

In this section we provide a brief review of Deng et al CLAS scheme. Deng et al CLS scheme consist of five algorithms *Masterkeygen, Partialkeygen, Userkeygen, Sign, Verify.*

Masterkeygen: KGC runs the algorithm after taking an input security parameter k.

Generate two cyclic groups G_1 and G_2 , where G_1 additive group and G_2 is the multiplicative group with the same order q with two generators P,Q off G_1 and a bilinear pairing $e: G_1 \times G_2 \to G_T$

i) Select a random number $s \in Z_q^*$ and computes $P_{pub} = sP$, taking s as a master key of KGC and P_{pub} as a public key of KGC.

ii) Select five one way cryptography hash functions $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to G_1$, $H_3 : \{0,1\}^* \to Z_q^*, H_4 : \{0,1\}^* \to Z_q^*$, $H_5 : \{0,1\}^* \to Z_q^*$.

iii)Generates the system parameters say $Params$ are $\{q, G_{1,} G_2, e, P, Q, P_{pub}, H_1, H_2, H_3, H_4, H_5\}$ and keep secretly master key s by KGC.

Partialkeygen: After taking input user's identity ID_i , The KGC first computes the user's partial private key $psk_{ID_i} = sQ_{ID_i}$ where $Q_{ID_i} = H_1(ID_i)$ and forward it to the user via a secure way.

Userkeygen: The user chooses a random number $x_{ID_i} \in Z_q^*$ and set as secret key $usk_{ID\,i}$, then computes its public key $upk_{ID_i} = usk_{ID\,i}.P$

Sign: The user with identity ID_i takes the $Params$, the partial private key psk_{ID_i} , corresponding secret key $usk_{ID\,i}$ and then performs the following steps to generate the signature:

i) Select a random number $r_i \in Z_q^*$ and computes

$U_i = r_i.P$, $t_i = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_i = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $Q = H_2(q, P, P_{pub})$, $k_i = H_4(m_i, ID_i, upk_{iD_i}, U_i)$

ii) Compute: $V_i = psk_{ID_i} + t_i.r_i.P_{pub} + h_i.x_{ID_i}.Q + k_i.r_i.Q$

iii)Provides a signature (U_i, V_i) on message m_i.

Verify: Given a signature (U_i, V_i) with message m_i corresponding public key upk_{ID_i} regarding the identity ID_i verifier performs the following steps:

i) Computes $U_i = r_i.P$, $t_i = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_i = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $Q = H_2(q, P, P_{pub})$, $k_i = H_4(m_i, ID_i, upk_{iD_i}, U_i)$ Verify the following equation

$e(V_i, P) = e(Q_{ID_i} + t_i U_i, P_{pub}) e(h_i upk_{ID_i} + k_i U_i, Q)$

If it satisfied to then accept the signature.

5.3.2 REVIEW OF DENG ET AL [25] CLAS SCHEME

The CLAS scheme consists of seven steps in which five algorithms *Master keygen, Partialkeygen, Userkeygen, Sign, Verify* are same as CLS scheme and two extra algorithms say *Aggregate* and *Aggregateverify* are involved in the CLAS scheme whose description is given below:

1) *Aggregate*: for an aggregating set of n users $\{U_1, U_2 ...., U_n\}$ with their identities $\{ID_1, ID_2 ...., ID_n\}$ and the corresponding public keys $\{upk_1, upk_2, ..., upk_n\}$, and with signature pairs $\{(m_1, \sigma_1 = (U_1, V_1)), ... (m_n, \sigma_n = (U_n, V_n)\}$, then aggregator computes $V = \sum_{i=1}^{n} V_i$ and results an aggregate signature as $\sigma = (U_1, U_2 .... U_n, V)$.

2) *Aggregate Verify*: For verifying an aggregate signature $\sigma = (U_1, U_2 .... U_n, V)$ signing by n users $\{U_1, U_2 ...., U_n\}$ with their identities, verifier performs the following steps:

i) Computes $Q_{ID_i} = H_1(ID_i)$, $h_i = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $k_i = H_5(m_i, ID_i, upk_{iD_i}, U_i)$, $Q = H_2(q, P, P_{pub})$

$t_i = H_3(m_i, ID_i, upk_{iD_i}, U_i)$

ii) Verify

$$e(V,P) = e(\sum_{i=1}^{n}(Q_{ID_i} + t_i U_i), P_{pub}) \cdot e(\sum_{i=1}^{n}(h_i upk_{ID_i} + k_i U_i), Q)$$

Now, we proposed a cryptanalysis of Deng et al. [25] scheme.

5.3.3 ATTACK ON Deng et al. [25] CLS SCHEME

Deng et al. [25] proposed a certificateless signature scheme and claims it is unforgeable from adaptive chosen message and identity attacks and demonstrate that two types of adversary involve in the CLS scheme. In the proposed attack, we demonstrate that no need of adversary 1 and adversary 2 in the scheme, another universal adversary A exist who can forge the signature without knowing the master key of the KGC and private key of the user.

Adversary A takes following steps to forge the valid signature.

Step 1: Adversary A randomly selects an identity ID_i with four messages $m_1, m_{2,} m_3, m_4$ and found respective four legal signatures $\sigma_1 = (U_1, V_1), \sigma_2 = (U_2, V_2),\ \sigma_3 = (U_3, V_3), \sigma_4 = (U_4, V_4)$ on the same identity ID_i by submitting sign query.

Step 2: A computes $t^* = H_3(m^*, ID_i, upk_{iD_i}, U_i)$, $h^* = H_4(m^*, ID_i, upk_{iD_i}, U_i)$, $k^* = H_4(m^*, ID_i, upk_{iD_i}, U_i)$

Step 3: A can compute, $t_1 = H_3(m_1, ID_i, upk_{iD_i}, U_i)$, $h_1 = H_4(m_1, ID_i, upk_{iD_i}, U_i)$, $k_1 = H_5(m_1, ID_i, upk_{iD_i}, U_i)$,

$t_2 = H_3(m_2, ID_i, upk_{iD_i}, U_i)$, $h_2 = H_4(m_2, ID_i, upk_{iD_i}, U_i)$, $k_2 = H_5(m_2, ID_i, upk_{iD_i}, U_i)$

$t_3 = H_3(m_3, ID_i, upk_{iD_i}, U_i)$, $h_3 = H_4(m_3, ID_i, upk_{iD_i}, U_i)$, $k_3 = H_5(m_3, ID_i, upk_{iD_i}, U_i)$

$t_4 = H_3(m_4, ID_i, upk_{iD_i}, U_i)$, $h_4 = H_4(m_4, ID_i, upk_{iD_i}, U_i)$, $k_4 = H_5(m_4, ID_i, upk_{iD_i}, U_i)$

Step 5: Using $t_{1.}, t_2, t_3, h_{1.}, h_2, h_3, k_{1.}, k_2, k_3$ A can compute $\rho_1, \rho_2, \rho_3, \rho_4$ which satisfies the following equation:

$$\begin{pmatrix} t_1 & t_2 & t_3 & t_5 \\ h_2 & h_2 & h_3 & h_5 \\ k_1 & k_2 & k_3 & k_5 \\ 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ 1 \end{pmatrix} = \begin{pmatrix} t^* \\ h^* \\ k^* \\ 1 \end{pmatrix} modq$$

$$\rho_1 t_1 + \rho_2 t_2 + \rho_3 t_3 + \rho_4 t_4 = t^*$$

$$\rho_1 h_1 + \rho_2 h_2 + \rho_3 h_3 + \rho_4 h_4 = h^*$$

$$\rho_1 k_1 + \rho_2 k_2 + \rho_3 k_3 + \rho_4 k_4 = k^*$$

$$\rho_1 + \rho_2 + \rho_3 + \rho_4 = 1$$

Step 5: Finally A produces a forged signature $\sigma^* = (U^*, V^*)$ on a message m^* , where $U^* = \rho_1 r_1 + \rho_2 r_2 + \rho_3 r_3 + \rho_4 r_4$ and $V^* = \rho_1 V_1 + \rho_2 V_2 + \rho_3 V_3 + \rho_4 V_4$.

Theorem 1: The forged signature $\sigma^* = (U^*, V^*)$ obtained by an adversary A is a legal signature.

Proof: For proving $\sigma^* = (U^*, V^*)$ is a legal signature, we require to show that an adversary can be verifies the signature by the verification scheme,

$$e(V^*, P) = e(Q_{ID_i} + t^* U^*, P_{pub}) e(h^* upk_{ID_i} + k^* U^*, Q)$$

From the step 2, we can obtain the value of $\rho_1, \rho_2, \rho_3, \rho_4$.

$$e(V^*, P) = e(\rho_1 V_1 + \rho_2 V_2 + \rho_3 V_3 + \rho_4 V_4, P)$$

$$= e(\rho_1 (psk_{ID_i} + t_1.r_i.P_{pub} + h_1.x_{ID_i}.Q + K_1.r_i.Q)$$

$$+\rho_2 (psk_{ID_i} + t_2.r_i.P_{pub} + h_2.x_{ID_i}.Q + K_2.r_i.Q + \rho_3 (psk_{ID_i} + t_3.r_i.P_{pub} + h_3.x_{ID_i}.Q + K_3.r_i.Q)$$
$$+\rho_4 (psk_{ID_i} + t_4.r_i.P_{pub} + h_4.x_{ID_i}.Q + K_4.r_i.Q), P)$$
$$= e((\rho_1 + \rho_2 + \rho_3 + \rho_4) psk_{ID_i} + (\rho_1 t_1 + \rho_2 t_2 + \rho_3 t_3 + \rho_4 t_4) r_i P_{pub}).Q +$$

$$+(\rho_1 h_1+\rho_2 h_2+\rho_3 h_3+\rho_4 h_4)x_{ID_i}+(\rho_1 k_1+\rho_2 k_2+\rho_3 k_3+\rho_4 k_4)r_i Q, P)$$

$$=e(psk_{ID_i}+t^*.r_i.P_{pub}+h^*.x_{ID_i}.Q+k^* r_i.Q, P) = e(Q_{ID_i}+t^* U^*, P_{pub})e(h^* upk_{ID_i}+k^* U^*, P_{pub})$$

This completes the proof.

5.3.4 ATTACK ON Deng et al. [25] CLAS SCHEME

In this subsection, we demonstrate that adversary A forge the aggregate signature.

Step 1: Adversary A selects with four messages $m_1^{'}, m_2^{'} m_3^{'}, m_4^{'}$ and transferred to all users who are participant in CLAS and found four legal signatures $\sigma_1=(U_1^{'},V_1^{'}), \sigma_2=(U_2^{'},V_2^{'})$, $\sigma_3=(U_3^{'},V_3^{'}), \sigma_4=(U_4^{'},V_4^{'})$ from each user.

Step 3: With the help of the mentioned four signatures A can compute a legal forged signature $\sigma_i^*=(U_i^*,V_i^*)$ on the message m_i^* where $i \in [1,n]$.

Step 3: Finally A output an aggregate signature $\hat{\sigma}=(\hat{U},\hat{V})$ from n signatures $((m_1^*,\sigma_1^*=(U_1^*,V_1^*),(m_2^*,\sigma_2^*=(U_2^*,V_2^*)........(m_n^*,\sigma_n^*=(U_n^*,V_n^*))$ where $\hat{U}=\sum_{i=1}^{n} U_i^*$ and $\hat{U}=\sum_{i=1}^{n} V_i^*$

Theorem 2: The forged signature $\hat{\sigma}=(\hat{U},\hat{V})$ obtained by an adversary A is a legal signature.

Proof: For proving legality of the signature $\hat{\sigma}=(\hat{U},\hat{V})$, we require to show that Adversary A can be verifies the signature by the verification scheme,

$$e(\hat{V},P)=e(\sum_{i=1}^{n}(Q_{ID_i}+t_i^* U_i^*), P_{pub}) \,.\, e(\sum_{i=1}^{n}(h_i^* upk_{ID_i}+k_i^* U_i^*), P_{pub}) \text{ where } 1 \le i \le n$$

As proven theorem 1, A can obtain a forged legal signature $\sigma_i^* = (U_i^*, V_i^*)$ on the message m_i^*, then he can aggregate all the signature on message m_i^* and generate an aggregate signature $\hat{V} = \sum_{i=1}^{n} psk_{ID_i} + \sum_{i=1}^{n} t_i^*.r_i.P_{pub} + \sum_{i=1}^{n} h_i^*.x_{ID_i}.Q + \sum_{i=1}^{n} k_i^*.r_i.Q$ where $1 \le i \le n$.

As proven in theorem 1, we can show easily that the aggregate signature $\hat{\sigma} = (\hat{U}, \hat{V})$ verifies the verification equation.

5.3.5 Collision insider attack on Deng et al [25] scheme

A collision resistant property defined that no signer groups holding the KGC together can generate a valid aggregate signature [56]. A dishonest user might be an internal user (one of the sharing user in the aggregate signature) or external user cooperates with malicious KGC to produce a valid forge aggregate signatures.

KGC collaborate with the dishonest user u_{n+1} of identity ID_{n+1} to forge a certificateless aggregate signature scheme. This attack will be performing in 3 steps.

Step 1: A dishonest signer u_{n+1} chooses a random number $v \in z_q^*$, such that $v = usk_{ID_1} + usk_{ID_2} ......+ usk_{ID_n} + \gamma$ afterward u_{n+1} can calculate $\gamma P = vP - \sum_{i=1}^{n} upk_{IDi}$ and put its public key while u_n don't have a knowledge about γ.

Step 2: KGC and u_{n+1} chooses $r_1, r_2, ...r_n, r_{n+1} \in Z_q^*$ in cooperation and calculate $U_i = r_i P$ and $w_i = H_1(m_i, upk_{ID_i}, ID_i, U_i)$, $t_i = H_2(m_i, upk_{ID_i}, ID_i, U_i)$, where $i \in [1, n]$

Then compute $V^* = \sum_{i=1}^{n+1} psk_{ID_i} + \sum_{i=1}^{n+1} t_i.r_i.P_{pub} + \sum_{i=1}^{n+1} h_i.v.Q + \sum_{i=1}^{n+1} k_i.r_i.Q$

Since all psk_{ID_i} are well-known to malicious KGC afterwards they generates an aggregate signature $\sigma^* = (U_1, U_2 ....U_{n+1}, V^*)$ without using the private key of corresponding identities $\{ID_1, ID_2 ....ID_{n+1}\}$.

Step 3: $\sigma^* = (U_1, U_2 ....U_{n+1}, V^*)$ be a valid aggregate signature on the message set $\{m_1, m_2, ...m_{n+1}\}$ for corresponding identities $\{ID_1, ID_2 ....ID_{n+1}\}$ with public key

$\{upk_{ID_1}, upk_{ID_2}, ...upk_{ID_{n+1}}\}$ by verification. Validation of aggregate signature can be verified by the following signature.

Correctness:

Validation of aggregate signature can be checked as follow.

$$e(S^*, P) = e(\sum_{i=1}^{n+1} psk_{ID_i} + \sum_{i=1}^{n+1} t_i.r_i.P_{pub} + \sum_{i=1}^{n+1} h_i.v.Q + \sum_{i=1}^{n+1} k_i.r_i.Q, P)$$

$$= e(\sum_{i=1}^{n} (sQ_{ID_i}, P) e(\sum_{i=1}^{n} t_i r_i P, sP) e(\sum_{i=1}^{n+1} h_i.v.P, Q) e(\sum_{i=1}^{n+1} k_i.r_i.P, Q)$$

$$= e(\sum_{i=1}^{n+1} (Q_{ID_i}, P_{pub}) e(\sum_{i=1}^{n+1} t_i U_i, P_{pub}) e(\sum_{i=1}^{n+1} h_i.upk_{ID_i}, Q) e(\sum_{i=1}^{n+1} k_i.U_i, Q)$$

$$= e(\sum_{i=1}^{n+1} (Q_{ID_i} + t_i U_i, P_{pub}) e(\sum_{i=1}^{n+1} h_i.upk_{ID_i} + k_i.U_i, Q)$$

5.4 Cryptanalysis of Deng et al [26] CLS and CLAS Scheme

In this subsection we have done cryptanalysis of Deng et al CLS and CLAS scheme.

5.4.1 Review of the Deng et al [26] CLS Scheme

In this section we give a brief review of Deng et al [26] CLAS scheme. Deng et al CLS [26] scheme consists of five algorithms *Masterkeygen, Partialkeygen, Userkeygen, Sign, Verify.*

Masterkeygen: on taking a security input k, KGC starts the algorithm as follow:

iv) Generate two groups one is cyclic additive group G_1 and second is cyclic multiplicative group G_2 having the same order q with two generator P, Q of G_1 and a bilinear pairing $e: G_1 \times G_2 \rightarrow G_T$

v) Select a random number $s \in Z_q^*$ and computes $P_{pub} = sP$, taking s as a master key of KGC and P_{pub} as a public key of KGC.

vi) Select four hash functions $H_1:\{0,1\}^* \to G_1$, $H_2:\{0,1\}^* \to G_1$, $H_3:\{0,1\}^* \to Z_q^*$, $H_4:\{0,1\}^* \to Z_q^*$.

vii) Generates the system parameters say $Params$ are $\{q, G_{1,} G_2, e, P, Q, P_{pub}, H_1, H_2, H_3, H_4\}$ and keep secret master key s by KGC.

Partialkeygen: After taking input user's identity ID_i , The KGC first computes the user's partial private key $psk_{ID_i} = sQ_{ID_i}$ where $Q_{ID_i} = H_1(ID_i)$ and forward it to the user via a secure way.

Userkeygen: The user chooses a random number $x_{ID_i} \in Z_q^*$ and set as secret key usk_{IDi}, then computes its public key $upk_{ID_i} = usk_{IDi}.P$

Sign: The user with identity ID_i takes the $Params$, the partial private key psk_{ID_i} , corresponding secret key usk_{IDi} and then performs the following steps to generate the signature:

iv) Select a random number $r_i \in Z_q^*$ and computes

$U_i = r_i.P$, $h_{1i} = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_{2i} = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $K = H_2(q, P, P_{pub})$

v) Compute: $V_i = psk_{ID_i} + h_{1i} r_i P + h_{2i} x_i K$

vi) Provides a signature (U_i, V_i) on message m_i.

Verify: Given a signature (U_i, V_i) with message m_i corresponding public key upk_{ID_i} regarding the identity ID_i verifier performs the following steps:

ii) Computes $U_i = r_i.P$, $h_{1i} = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_{2i} = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $K = H_2(q, P, P_{pub})$

iii) Verify the following equation

$$e(V_i, P) = e(Q_{ID_i} + h_{1i} U_i, P_{pub})\ e(h_{2i} upk_{ID_i}, K)$$

If it satisfied then accept the signature.

5.4.2 Review of Deng et al [26] CLAS scheme

CLAS scheme consist of seven steps in which five algorithms *Masterkeygen, Partialkeygen, Userkeygen, Sign, Verify* are same as CLS scheme and two extra algorithms say Aggregate and Aggregate verify are involved in CLAS scheme whose description is given below:

3) *Aggregate*: for an aggregating set of n users $\{U_1, U_2 ...., U_n\}$ with their identities $\{ID_1, ID_2 ...., ID_n\}$ and the corresponding public keys $\{upk_1, upk_2, ..., upk_n\}$, and with signature pairs $\{(m_1, \sigma_1 = (U_1, V_1)), ...(m_n, \sigma_n = (U_n, V_n)\}$, then aggregator computes $V = \sum_{i=1}^{n} V_i$ and results an aggregate signature as $\sigma = (U_1, U_2 .... U_n, V)$.

4) *Aggregate Verify*: for verify an aggregate signature $\sigma = (U_1, U_2 .... U_n, V)$ signing by n users $\{U_1, U_2 ...., U_n\}$ with their identities $\{ID_1, ID_2 ...., ID_n\}$, verifier performs the following steps:

iii) Computes $Q_{ID_i} = H_1(ID_i)$, $h_{1i} = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_{2i} = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $K = H_2(q, P, P_{pub})$

iv) Verify

$$e(V, P) = e(\sum_{i=1}^{n} (h_{1i}.U_i + Q_{ID_i}), P_{pub}) \cdot e(\sum h_{2i}.upk_{ID_i}, K)$$

5.4.3 Cryptanalysis of Deng et al [26] CLS scheme

In this subsection we discuss the type II attack on behalf, we claim that the proposed CLS scheme is insecure. Since KGC knows the master key, then KGC computes the value . Since s and U are publicly known and with the help of known value sU_i we calculate $r_i P_{pub}$. With the help of master key KGC can compute the partial private key of user psk_{ID_i}, by $psk_{ID_i} = sQ_{ID_i}$ while $Q_{ID_i} = H_1(ID_i)$ is known quantity. h_{1i} and h_{2i} are the hash value then h_{2i}^{-1} also compute. Now he can compute the fix value $x_i K = h_{2i}^{-1}(V^* - psk_{ID_i} - h_{1i} s U_i^*)$ by capturing the signature (U^*, V^*) on message m_i , $h_{1i} = H_3(m_i^*, ID_i, upk_{iD_i}, U_i^*)$ $h_{2i} = H_4(m_i^*, ID_i, upk_{iD_i}, U_i^*)$. Since KGC don't know the user's secret key but he known about the fix value $x_i K$ then he can forge user's signature on any message in aggregate set. The description of this attack is given below:

Intercept partial signature: in the first step KGC intercept the signature of user U_i with the identity ID_i corresponding public key upk_{ID_i} and find the signature (U^*, V^*).

Compute fix value:

i) Compute $h_{1i} = H_3(m_i^*, ID_i, upk_{iD_i}, U_i^*)$ $h_{2i} = H_4(m_i^*, ID_i, upk_{iD_i}, U_i^*)$, $K = H_2(q, P, P_{pub})$

ii) Compute $r_i^* P_{pub} = r_i^* sP = sr_i P = sU_i^*$

iii)Computes $x_i K = h_{2i}^{-1}(V^* - psk_{ID_i} - h_{1i} sU_i^*)$

Forge partial signature: Now KGC perform the following step to forge CLS signature $(U_i^{'}, V_i^{'})$ on message $m_i^{'}$.

i) Select $U_i^{'} \in G_1$, and extract the value of sU_i from $r_i P_{pub}$.

ii) Computes $h_{1i} = H_3(m_i^{'}, ID_i, upk_{iD_i}, U^{'})$ $h_{1i} = H_4(m_i^{'}, ID_i, upk_{iD_i}, U_i^{'})$, $K = H_2(q, P, P_{pub})$

iii)Computes $V_i^{'} = psk_{ID_i} + h_{1i} sU_i^{'} + h_{2i} x_i K$

Then he provides an output $(U_i^{'}, V_i^{'})$ on the message $m_i^{'}$.

Verification:

$$\begin{aligned} e(V_i^{'}, P) &= e(psk_{ID_i} + h_{1i} sU_i^{'} + h_{2i} x_i K, P) \\ &= e(psk_{ID_i} + h_{1i} sU_i^{'}, P)\ e(h_{2i} x_i K, P) \\ &= e(sQ_{ID_i} + h_{1i} sU_i^{'}, P)\ e(h_{2i} x_i P, K) \\ &= e(Q_{ID_i} + h_{1i} U_i^{'}, sP)\ e(h_{2i} upk_{ID_i}, K) \\ &= e(Q_{ID_i} + h_{1i} U_i^{'}, P_{pub})\ e(h_{2i} upk_{ID_i}, K) \end{aligned}$$

5.4.4 Cryptanalysis of Deng [26] CLAS scheme

KGC can compute $r_i P_{pub}$ and $x_i K$ of any user's signature by the above method mention used to forge the CLS scheme. Then he can club the entire signatures to forge the aggregate signature.

Now KGC calculates $V^{**} = \sum_{i=1}^{n} V^{'}$ and provide the output $(U_1^{'}, U_2^{'}.........., U_n^{'}, V^{**})$ as the forge aggregate signature.

For $i \in [1, n]$, $Q_i = H_1(ID_i)$, $h_{1i} = H_3(m_i^{'}, ID_i, upk_{iD_i}, U_i^{'})$, $h_{1i} = H_4(m_i^{'}, ID_i, upk_{iD_i}, U_i^{'})$, $K = H_2(q, P, P_{pub})$

Forge aggregate signature is valid if it satisfied the following equation.

$$e(V^{**}, P) = e(\sum_{i=1}^{n}(h_{1i}.U_i^{'} + Q_{ID_i}), P_{pub})\ e(\sum h_{2i}.upk_{ID_i}, K)$$

5.4.5 Improved Deng et al [26] Certificateless Signature Scheme

We propose a modified CLAS scheme to remove the weakness of Deng et al CLAS scheme.

Masterkeygen: on taking a security input, KGC starts the algorithm as follows:

i) Generate two groups one is cyclic additive group G_1 and second is cyclic multiplicative group G_2 having the same order q with two generator P, Q of G_1 and a bilinear pairing $e: G_1 \times G_2 \rightarrow G_T$

ii) Select a random number $s \in Z_q^*$ and computes $P_{pub} = sP$, taking s as a master key of KGC and P_{pub} as a public key of KGC.

iii) Select four hash functions $H_1: \{0,1\}^* \rightarrow G_1$, $H_2: \{0,1\}^* \rightarrow G_1$, $H_3: \{0,1\}^* \rightarrow Z_q^*$, $H_4: \{0,1\}^* \rightarrow Z_q^*$.

iv) Generates the system parameters say *Params* are $\{q, G_{1,} G_2, e, P, Q, P_{pub}, H_1, H_2, H_3, H_4\}$ and keep secret master key s by KGC.

Partialkeygen: After taking input user's identity ID_i, The KGC first computes the user's partial private key $psk_{ID_i} = sQ_{ID_i}$ where $Q_{ID_i} = H_1(ID_i)$ and forward it to the user via a secure way.

Userkeygen: The user chooses a random number $x_{ID_i} \in Z_q^*$ and set as secret key usk_{IDi}, then computes its public key $upk_{ID_i} = usk_{IDi}.P$

Sign: The user with identity ID_i takes the *Params* , the partial private key psk_{ID_i} , corresponding secret key usk_{IDi} and then performs the following steps to generate the signature:

vii) Select a random number $r_i \in Z_q^*$ and computes

$U_i = r_i.P$, $h_{1i} = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_{2i} = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $K = H_2(q, P, P_{pub})$, $T = H_2(q, P, P_{pub})$

viii) Compute: $V_i = psk_{ID_i} + h_{1i}r_iT + h_{2i}x_iK$

ix) Provides a signature (U_i, V_i) on message m_i.

Verify: Given a signature (U_i, V_i) with message m_i corresponding public key upk_{ID_i} regarding the identity ID_i verifier performs the following steps:

iv) Computes $U_i = r_i.P$, $h_{1i} = H_3(m_i, ID_i, upk_{iD_i}, U_i)$, $h_{2i} = H_4(m_i, ID_i, upk_{iD_i}, U_i)$, $K = H_2(q, P, P_{pub})$

v) Verify the following equation

$e(V_i, P) = e(Q_{ID_i} + h_{1i}U_i, P_{pub})\ e(h_{2i}upk_{ID_i}, K)$

5.5 Cryptanalysis of Liu et al. [27] CLS Scheme: In this subsection we have done cryptanalysis of Deng et al CLS and CLAS scheme.

5.5.1 Review of Liu *et al.* [27] CLS scheme

Set up: On given input as a security parameter l where $l \in N$ then KGC generate an additive cyclic group and multiplicative group G_2 of same order q with generator P the bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$ .KGC generate a master key $\alpha \in Z_q^*$ and set public key of KGC as $P_{pub} = \alpha P$.

Select three hash functions $H_0: \{0,1\}^* \rightarrow G_1, H_1: \{0,1\}^* \rightarrow Z_q^*$ and $H_2: \{0,1\}^* \rightarrow Z_q^*$ .KGC generates the public parameter list $params = \{G_1, G_2, e, S, P, P_{pub}, H_0, H_1, H_2\}$.

PartialPrivateKey: Taking input master key α ,user's identity ID_i and $params$. Then KGC computes the partial private key such as $psk_{ID_i} = \alpha Q_i$, whereas $Q_i = H_0(ID_i)$.

UserKeyGen: On taking input $params$ and a user's identity ID_i, The user selects a random $x_{ID_i} \in Z_q^*$, and sets his secret value and public key as $upk_{ID_i} = x_{ID_i}P$.

Sign: The user with identity ID_i takes the $Params$, the partial private key psk_{ID_i} , corresponding secret key x_{ID_i} and public key upk_{ID_i} then performs the following steps to generate the signature.

i) Choose $r_i \in Z_q^*$ and set $R_i = r_i P$

ii) Compute $w_i = H_1(m_i, P_i, ID_i, R_i)$, $t_i = H_2(m_i, P_i, ID_i, R_i)$

iii) Compute $V_i = psk_{ID_i} + w_i x_{ID_i} S + t_i r_i P_{pub}$

iv) Output the signature $\sigma_i = (R_i, U_i)$.

5.5.2 Analysis of Liu *et al* [27] CLS scheme

In the type 2 attack adversary A2 be the malicious KGC and generate the system parameter $Params$ honestly but generate a particular generator S dishonestly.

AdversaryA2 select a random number t and compute $S = tP$ at the time of generating the system parameter. Adversary A2canperform the following steps to forge the signature.

i) Adversary A2 knows the master key α , then he can compute a user partial private key $psk_{ID_i} = \alpha Q_i$, where as $Q_i = H_0(ID_i)$

ii) To forge any signature pair $\sigma_i = (R_i, V_i)$ corresponding the message m_i Adversary A2 select a number $r_i \in Z_q^*$, $R_i = r_i P$

iii) Compute $w_i = H_1(m_i, P_i, ID_i, R_i)$, $t_i = H_2(m_i, P_i, ID_i, R_i)$

$V_i = psk_{ID_i} + t_i r_i P_{pub} + w_i t P_i$

Then output the signature $\sigma_i = (R_i, V_i)$

Correctness: Since $S = tP$, $R_i = r_i P$, $U_i = psk_{ID_i} + t_i r_i P_T + w_i t P_i$, then verify the signature

$$e(U_i, P) = e(psk_{ID_i} + t_i r_i P_{pub} + w_i t P_i, P)$$

$$= e(psk_{ID_i} + t_i r_i P_{pub}, P)\ e(w_i t P_i, P)$$

$$= e(\lambda Q_i + t_i r_i \alpha P, P)\ e(w_i P_i, tP)$$

$$= e(Q_i + t_i R_i, \alpha P)\ e(w_i P_i, S)$$

$$= e(Q_i + t_i R_i, P_{pub})\ e(w_i P_i, S)$$

5.5.3 Improvement of Liu *et al.* [11] CLS scheme

Set up: On given input as a security parameter l where $l \in N$ then KGC generate an additive cyclic group and multiplicative group G_2 of same order q with generator P the bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$ .KGC generate a master key $s \in Z_q^*$ and set public key of KGC as $P_{pub} = sP$.

Select three hash functions $H_0 : \{0,1\}^* \rightarrow G_1, H_1 : \{0,1\}^* \rightarrow Z_q^*$ and $H_2 : \{0,1\}^* \rightarrow Z_q^*$, $H_3 : \{0,1\}^* \rightarrow G_1$.

Finally KGC generates the system parameter list $params = \{G_1, G_2, e, P, P_{pub}, H_0, H_1, H_2, H_3\}$.

Rest of the algorithms PartialPrivateKeyGen, UserKeyGen, PseudonymGen retains same as above CLS scheme.

Sign: The user with identity ID_i takes the $Params$, the partial private key psk_{ID_i} , corresponding secret key x_i and public key P_i then performs the following steps to generate the signature.

i) Choose $r_i \in Z_q^*$ and set $R_i = r_i P$

ii) Compute $w_i = H_1(m_i, P_i, ID_i, R_i)$, $t_i = H_2(m_i, P_i, ID_i, R_i)$, $T = H_3(params)$

iii) Compute $V_i = psk_{ID_i} + w_i x_i T + t_i r_i P_{pub}$

Output the signature $\sigma_i = (R_i, V_i)$.

Verification: Given a signature $\sigma_i = (R_i, V_i)$ on the message m_i corresponding identity ID_i public key P_i verifier can verify as follow:

i) Choose $r_i \in Z_q^*$ and set $R_i = r_i P$

ii) Compute $w_i = H_1(m_i, P_i, ID_i, R_i)$, $t_i = H_2(m_i, P_i, ID_i, R_i)$, $T = H_3(Params)$

iii) $e(U_i, P) = e(Q_i + t_i R_i, P_T) e(w_i P_i, T)$

If satisfies then accept otherwise reject.

Correctness: $e(U_i, P) = e(psk_{ID_i} + w_i x_i T + t_i r_i P_{pub}, P)$

$$= e(psk_{ID_i}, P)e(w_i x_i T, P)e(t_i r_i P_{pub}, P)$$

$$= e(\alpha.Q_i, P)e(w_i x_i P, T)e(t_i r_i P, P_{pub})$$

$$= e(Q_i, \alpha P)e(w_i P_i, T)e(t_i R_i, P_{pub})$$

$$= e(Q_i, P_T)e(w_i P_i, T)e(t_i R_i, P_{pub})$$

$$= e(Q_i + t_i R_i, P_{pub})e(w_i P_i, T)$$

Recently Horng et al [59] and Malhi and Batra [22] proposed their CLAS schemes for vehicular ad-hoc networks to increase the security as well as the efficiency of vehicular ad-hoc networks. In this section, we demonstrate that Horng et al [59] and Malhi and Batra [22] CLAS schemes fail to protect. The security analysis of both the schemes found them insecure against collision resistance attack.

5.6 Cryptanalysis of Horng et al [59] CLS and CLAS Scheme

In this subsection we have done cryptanalysis of Horng et al CLS and CLAS scheme.

5.6.1 Review of the Horng et al [59] CLS and CLAS Scheme

The following section of this chapter covers a brief review of the Horng et al [59] CLAS Scheme and security analysis of their CLAS scheme.

Horng et al [59] proposed an efficient CLAS scheme fit for vehicular ad-hoc network keeping primary focus on privacy preserving. This CLAS scheme consists of seven algorithms such as *Setup, Pseudo-identity-Gen/Partial-Private-Key-Gen, Vehicle-Key-Generation, Sign, Individual verify, Aggregate, Aggregate Verify*.

Setup: Trusted Authorities (TA), Trace authority (TRA) and a key generation center (KGC) are involved in generating the system parameters. Taking an input security parameter*k*, TA generates an additive cyclic group G_1, a multiplicative group G_2 of same order q with two generators P and Q in G_1 and an admissible pairing map $e: G_1 \times G_1 \rightarrow G_2$. The KGC selects a random number $s \in Z_q^*$ and set public key of KGC as $P_{pub} = sP$, where s

is the master key of KGC. The TRA selects a random number $\beta \in Z_q^*$ and set public key of TRA as $T_{pub} = \beta P$, where β is the master key of TRA.

The KGC chooses two cryptographic hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : \{0,1\}^* \to Z_q^*$, thereafter publishes the system parameters as $Params = \{q, G_1, G_2, e, P, Q, P_{pub}, T_{pub}, H_1, H_2\}$.

Pseudo-identity-Gen/Partial-Private-Key Gen:

i. Vehicle V_i selects a random number $k_i \in Z_q^*$ and computes $ID_{i,1} = k_i P$. Transport a tuple $(RID_i, ID_{i,1})$ to TRA by a secure channel, where RID_i is a unique identity for recognizing vehicles.

ii. TRA computes $ID_{i,2} = RID_i \oplus H(\beta.ID_{i,1}, T_i)$ after checking of RID_i , where $H(.)$ is the cryptographic hash function as $H_2 : \{0,1\}^* \to Z_q^*$ and T_i be the period of pseudo identity.

   Finally, a pseudo identity $ID_i = (ID_{i,1}, ID_{i,2}, t_i)$ is transferred to KGC via a secure channel, where t_i is the time stamp.

iii. Taking input pseudo identity ID_i , KGC computes $Q_{ID_i} = H_1(ID_i)$ and sets a partial private key as $psk_{ID_i} = s.Q_{ID_i}$.

iv. Finally, KGC transfers the pair (ID_i, psk_{ID_i}) via a secure channel to the vehicle.

Vehicle-Key-Gen: The Vehicle with Identity ID_i select a random number $x_{ID_i} \in Z_q^*$, and set their secret key x_{ID_i} , and compute corresponding public key as $P_i = x_{ID_i}.P$.

Sign: The vehicle performs the following steps to generate a signature:

i. Choose a random number $r_i \in Z_q^*$ and computes $R_i = r_i P$.

ii. Compute $h_i = H_2(m_i, ID_i, P_i, R_i, t_i)$

iii. Compute $V_i = psk_{ID_i} + (x_{ID_i} + h_i.r_i).Q$

iv. Then $\sigma_i = (R_i, V_i)$ is the signature corresponding to the identity ID_i

Finally, the vehicle V_i transfers the final message $(ID_i, P_{ID_i}, m_i, t_i, \sigma_i)$ to nearby RSU.

Individual-Verify: Verifier performs the following actions for verification of the signature.

i. Compute $Q_{ID_i} = H_1(ID_i)$ and $h_i = H_2(m_i, ID_i, P_{ID_i}, R_i, t_i)$

ii. Verifies $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(P_{ID_i} + h_i.R_i, Q)$

The verifier accepts the signature if above verification equation satisfies.

*Aggregate:*A RSU can become an aggregator and generate an aggregate signature on messages $\{m_1, m_2........,m_n\}$ with pseudo identities $\{ID_1, ID_2.........ID_n\}$ corresponding public keys $\{P_{ID_1}, P_{ID_2}...........P_{ID_n}\}$ .Then RSU computes $V = \sum_{i=1}^{n} V_i$ and generates an output as aggregate signature $\sigma = (R_1, R_2........R_n, V)$.

Aggregate Verify: For verification, verifier performs the following steps:

i. Computes $Q_{ID_i} = H_1(ID_i)$ and $h_i = H_2(m_i, ID_i, P_{ID_i}, R_i, t_i)$ for $i \in [1, n]$

ii. Check whether $e(V, P) = e(\sum_{i=1}^{n} Q_{ID_i}, P_{pub}).e(\sum(P_{ID_i} + h_i.R_i), Q)$

The verifier accepts the signature if above verification equation satisfies.

5.6.2 Security analysis of Horng et al [59] CLAS Scheme

In the proposed CLAS scheme, Horng et al [59] proved that their scheme is secure against adaptive chosen message and identity attacks, but their scheme does not provide any security against the collision resistant attack. The collision resistant attack is being performed by a corrupt user with the collaboration of KGC. Collision resistant property describes that no signer groups containing the KGC together can generate a valid aggregate signature scheme. Unfortunately, we found that Horng et al [59] fails to protect their model against the collision resistant attack.

The user u_n with identity ID_n makes collaboration with KGC to attack this CLAS scheme. This attack performs following three steps:

Step 1: Suppose a corrupt signer u_n selects a number randomly, such that $v = x_{ID_1} + ... + x_{ID_{n-1}} + \gamma$, thereafter u_n computes $\gamma P = vP - \sum_{i=1}^{n-1} P_{ID_i}$ and declare it as public key. Although u_n has no information about γ .

Step 2: u_n select $r_1, r_2, ...r_n \in z_q^*$ in collaboration and compute $R_i = r_i P$ and

$h_i = H_2(m_i, ID_i, vpk_{ID_i}, R_i, t_i)$, then compute $S^* = \sum_{i=1}^{n} psk_{ID_i} + t.Q + \sum_{i=1}^{n} h_i r_i Q$

Since all psk_{ID_i} are known to malicious KGC thereafter generates an aggregate signature $\sigma^* = (R_1, R_2.........R_n, V^*)$ without using the private key of corresponding identities $\{ID_1, ID_2 ....ID_n\}$.

Step 3: Now $\sigma^* = (R_1, R_2.........R_n, V^*)$ is a valid aggregate signature on the message set $\{m_1, m_2, ...m_n\}$ for corresponding identities $\{ID_1, ID_2 ....ID_n\}$. Validation of aggregate signature can be verified by the following equation:

$$e(V^*, P) = e(\sum_{i-1}^{n} Q_{ID_i}, P_{pub}) e(\sum_{i=1}^{n} [P_{ID_i} + h_i R_i], Q)$$

Correctness

$$e(V_i^*, P) = e(\sum_{i-1}^{n} psk_{ID_i} + t.Q + \sum_{i=1}^{n} h_i r_i Q, P)$$

$$= e(\sum_{i-1}^{n} psk_{ID_i}, P) e(t.Q, P) e(\sum_{i=1}^{n} h_i r_i Q, P)$$

$$= e(\sum_{i-1}^{n} Q_{ID_i}, sP) e(\sum_{i=1}^{n} P_{ID_i}, Q) e(\sum_{i=1}^{n} h_i R_i, Q)$$

$$= e(\sum_{i-1}^{n} Q_{ID_i}, P_{pub}) e(\sum_{i=1}^{n} [P_{ID_i} + h_i R_i], Q)$$

Since $v.P = x_{ID_1}.P + x_{ID_2}.P........+ x_{ID_n}.P$, In other words, a corrupt signer without knowledge of the secret key γ and other secret key $x_{ID_i} (i = 1,2....n)$ makes sure to forge a CLAS scheme with the help of the v.

5.7 Cryptanalysis of Malhi and Batra [22] CLS and CLAS Scheme

In this subsection we have done cryptanalysis of Deng et al CLS and CLAS scheme.

5.7.1 Review of Malhi and Batra [22] CLAS scheme

Malhi and Batra [22] proposed a CLS for VANET and proves that their scheme is existential unforgeable under adaptive chosen message and identity attacks. Cryptanalysis of [22] CLS scheme and point out the security leaks of the CLS scheme by applying message and passive attacks.

Set up: Taking input as a security parameter 1^k where $k \in N$, KGC generates two groups G_1 and G_2 of the same order q with a generator P and a bilinear pairing.

KGC sets a master key $s \in Z_q^*$ and a public key of KGC as $P_{pub} = sP$. KGC selects two hash function $H_2 : \{0,1\}^* \to Z_q^*$ and $H_3 : \{0,1\}^* \to Z_q^*$, with message space $M = \{0,1\}^*$. Every RSU chooses a secret key $y_i \in Z_q^*$ and computes their corresponding public key as $P_{rsu_i} = y_i P$.

The KGC publishes the system parameter list $\{G_1, G_2, e, P, P_{pub}, H_2, H_3, P_{rsu_1}, P_{rsu_2}........, P_{rsu_i}\}$.

Vehicle registration:

I. RTA selects a cryptographic function $H_1 : \{0,1\}^* \to G_1$

II. Register the vehicle Identity ID_i with RTA as $Q_{ID_i} = H_1(ID_i) \in G_1$, where vehicle identity space is $\{0,1\}^*$.

PartialKeyGen: KGC takes as a parameter list, master key and, then KGC select user's partial private key $psk_{ID_i} = sQ_{ID_i}$, where $Q_{ID_i} = H_1(ID_i) \in G_1$.

UserKeyGen: The Vehicle with Identity Q_{ID_i} chooses a random number $x_i \in Z_q^*$ and sets it as a secret key of corresponding vehicle with public key of vehicle as $P_i = x_i P$

PseudonymGen: An autonomous network formed with 5 RSUs in the scheme and produces the pseudonym of the vehicle in two parts $PS1_j$ and $PS2_j$, such that $PS_j = PS1_j + PS2_j$. Select a random number $a_j \in Z_q^*$ for RSU_i and sets $PS1_j = a_j Q_{ID_i}$. Second part of pseudonym $PS2_j = a_j T_j$ where $T_j = H_3(PS1_j) \in Z_q^*$

Finally, the complete pseudonym will be $PS_j = PS1_j + PS2_j$

Sign: After taking a message $m_k \in M$, the partial private key psk_{ID_i}, the secret key x_i with vehicle identity Q_{ID_i} and the corresponding public key P_i, the user generates the signature as follows:

i. Select a random number $r_i \in Z_q^*$, and compute $R_i = r_i P \in G_1$

ii. Compute $h_{ijk} = H_2(m_k, PS1_j, P_i, R_i) \in Z_q^*$

iii. Compute $V_{ijk} = psk_{ID_i}.PS2_j + h_{ijk} r_i P_{pub} + h_{ijk} x_i P_{rsu_i}$

iv. Output (R_i, V_{ijk}) as a signature on m_k

Verify: for verification of signature (R_i, V_{ijk}) of message m_k

i) Compute $h_{ijk} = H_2(m_k, PS1_j, P_i, R_i)$, $T_j = H_3(PS1_j)$

ii) Verify $e(V_{ijk}, P) = e(PS1_j T_j + h_{ijk} R_i, P_{pub}) e(h_{ijk} P_i, P_{rsu_i})$

iii) If the above verify equation satisfies then accept otherwise reject the signature.

Aggregate Sign: The aggregator node collects all individual signatures σ_i with corresponding identity ID_i , with pseudonyms psk_{ID_i} corresponding to the vehicle's public key p_i on message m_i as input and creates an aggregate signature $\{\sigma_1 = (R_1, V_1), \sigma_2 = (R_2, V_2)........\sigma_n = (R_n, V_n)\}$ on messages $\{m_1, m_2........, m_n\}$. The aggregate signature can be computed as $V = \sum_{i=1}^{n} V_i$ and an aggregate signature pair is $\sigma = (R_1, R_2........R_n, V)$

Aggregate Verify: The verifier can verify the aggregate signature using the following steps:

Compute $h_i = H_2(m_i, PS1_i, P_i, R_i)$, $T_i = H_3(PS1_i)$ for $i \in [1, n]$

Verify the following equation

$$e(V, P) = e(\sum_{i=1}^{n} [PS1_i.T_i + h_i.R_i], P_{pub}) e(\sum_{i=n}^{n} h_i.P_i, P_{rsu})$$

If it satisfies then accept otherwise reject the Aggregate signature.

5.7.2 Attack 1 Malhi and Batra [22] CLAS scheme

Malhi and Batra [22] proposed an efficient certificateless aggregate signature and proved that it is secure against adaptive chosen message and identity attacks. In this section, we demonstrate that it is not secure against the type 2 adversary attack.

Since A2is the attacking adversary who knows the master key of KGC and with the help of master key, the adversary A2can find partial private key of user by $psk_{ID_i} = sQ_{ID_i}$ where

$Q_{ID_i} = H_1(ID_i) \in G_1$

A2can query the sign oracle and extract a valid signature (R_i, V_{ijk}) on the message m_k

i) Select a random number $r_i \in Z_q^*$, Compute $R_i = r_i P \in G_1$

ii) Compute $h_{ijk} = H_2(m_k, PS1_j, P_i, R_i) \in Z_q^*$, $T_j = H_3(PS1_j)$

iii) Compute $V_{ijk} = psk_{ID_i}.PS2_j + h_{ijk} r_i P_{pub} + h_{ijk} x_i P_{rsu_i}$

Now, A2 can intercept information $T_{ijk} = h_{ijk}^{-1}(V_{ijk} - psk_{ID_i}.PS2_j)$ with $h_{ijk}.h_{ijk}^{-1} \equiv 1$

Choose another message $m_k^{'}$, A2 computes $R^{'} = R$ and $h_{ijk}^{'} = H_2(m_k^{'}, PS1_j, P_i, R_i^{'}) \in Z_q^*$

Then, compute $V_{ijk}^{'} = psk_{ID_i}.PS2_j + h_{ijk}^{'} T_{ijk}$

Now, A2 outputs the signature $(R_i^{'}, V_{ijk}^{'})$ on the message $m_k^{'}$.

$$T_{ijk} = h_{ijk}^{-1}(V_{ijk} - psk_{ID_i}.PS2_j)$$

$$\begin{aligned} T_{ijk} &= h_{ijk}^{-1}(pp_i.PS2_j + h_{ijk} r_i P_{pub} + h_{ijk} x_i P_{rsu_i} - pp_i.PS2_j) \\ &= r_i P_{pub} + x_i P_{rsu_i} \end{aligned}$$

$$\begin{aligned} V_{ijk}^{'} &= pp_i.PS2_j + h_{ijk}^{'} T_{ijk} \\ &= pp_i.PS2_j + h_{ijk}^{'}(r_i P_{pub} + x_i P_{rsu_i}) \\ &= pp_i.PS2_j + h_{ijk}^{'} r_i P_{pub} + h_{ijk}^{'} x_i P_{rsu_i} \end{aligned}$$

Correctness:

$$\begin{aligned} e(V_{ijk}^{'}, P) &= e(pp_i.PS2_j + h_{ijk}^{'} r_i P_{pub} + h_{ijk}^{'} x_i P_{rsu_i}, P) \\ &= e(s.Q_{ID_i}.a_j.PS1_j, P) e(h_{ijk}^{'} r_i P, P_{pub}) e(h_{ijk}^{'} x_i P, P_{rsu_i}) \\ &= e(PS1_j.T_j, sP) e(h_{ijk}^{'} U_i, P_{pub}) e(h_{ijk}^{'} P_i, P_{rsu_i}) \\ &= e(PS1_j.T_j, P_{pub}) e(h_{ijk}^{'} U_i, P_{pub}) e(h_{ijk}^{'} P_i, P_{rsu_i}) \\ &= e(PS1_j.T_j + h_{ijk}^{'} U_i, P_{pub}) e(h_{ijk}^{'} P_i, P_{rsu_i}) \end{aligned}$$

5.7.3 Attack 2 Malhi and Batra [22] CLAS scheme

Adversary A2performs the following steps to forge the certificateless aggregate signature.

Since A2 knows the master key of KGC and he can find partial private key of user by $psk_{ID_i} = sQ_{ID_i}$ where $Q_{ID_i} = H_1(ID_i) \in G_1$

A2 queries the sign oracle and extract a valid signature (U_i, V_i) on the message m_i Where

i) Compute $R_i = r_i P \in G_1$, where $r_i \in Z_q^*$

ii) Compute $h_i = H_2(m_i, PS1_i, P_i, R_i) \in Z_q^*$, $T_i = H_3(PS1_i)$

iii) Compute $V_i = psk_{ID_i}.PS2_j + h_i r_i P_{pub} + h_i x_i P_{rsu}$

Now A2 can find $T_i = h_i^{-1}(V_i - psk_{ID_i}.PS2_i)$ where h_i satisfying $h_i.h_i^{-1} \equiv 1$

Choose any $m_i^{'}$, A2computes $R^{'} = R$ and $h_i^{'} = H_2(m_i^{'}, PS1_i, P_i, R_i^{'}) \in Z_q^*$

Then compute $V_i^{'} = pp_i.PS2_i + h_i^{'} T_i$

A2 outputs the valid signature $(R_i^{'}, V_i^{'})$ on the message $m_i^{'}$.

$T_i = h_i^{-1}(V_i - psk_{ID_i}.PS2_i)$

A2 collect the entire individual signature $V = \sum_{i=1}^{n} V_i^{'}$ and provides an aggregate signature $\sigma^{'} = (R_1^{'}, R_2^{'}.......R_n^{'}, V)$ with the corresponding public key p_i on messages.

$$\begin{aligned} V &= \sum_{i=1}^{n} pp_i.PS2_j + \sum_{i=1}^{n} h_i^{'}.T_i \\ &= \sum_{i=1}^{n} pp_i.PS_j + \sum_{i=1}^{n} h_i^{'}(r_i P_{pub} + x_i P_{rsu}) \\ &= \sum_{i=1}^{n} pp_i.PS2_j + \sum_{i=1}^{n} h_i^{'} r_i P_{pub} + \sum_{i=1}^{n} h_i^{'} x_i P_{rsu} \end{aligned}$$

A2 can verify the signature by the following equation.

$$e(V, P) = e(\sum_{i=1}^{n}[PS1_i.T_i + +h_i.U_i^{'}], P_{pub})e(\sum_{i=1}^{n} h_i^{'}.P_i, P_{rsu})$$

5.7.4 Attack 3 Malhi and Batra [22] CLAS scheme

Malhi and Batra [22] proposed a certificateless signature scheme in which they proved that their CLS Scheme is existentially unforgeable under adaptive chosen message and identity attacks. Unfortunately, we found it insecure against malicious, but passive attacks.

Adversary A2 can forge a signature after intercepting the information from the valid signature. Then Adversary A2 can submit a sign query and find a valid signature (U_i, V_{ijk}) corresponding identity ID_i .

Extract fix value: Adversary A2 knows the master key α in the type II adversary attack. So it can successfully find partial private key $pp_i = \alpha Q_i$ and easily compute $x_i P_{rsu_i}$ by the following steps.

i) Compute $h_{ijk} = H_2(m_k, PS1_j, P_i, R_i)$

ii) Compute $r_i P_{pub} = r_i sP = sr_i P = sR_i$

iii) Computes $x_i P_{rsu_i} = h_{ijk}^{-1}(V_{ijk} - pp_i.PS2_j - h_{ijk} sR_i)$

Forge partial signature: Adversary A2 can create a new to valid forge CLS signature $(R_i^{'}, V_{ijk}^{'})$ on message $m_i^{'}$ using fixed value $x_i P_{rsu_i}$ intercept in the previous step.

i) Select a random number $r_i^* \in Z_q^*$ and compute $R_i^* = r_i^* P$

ii) Computes $h_{ijk} = H_2(m_k, PS1_j, P_i, R_i) \in Z_q^*$

Computes $V^{'}{}_{ijk} = pp_i.PS2_j + h_{ijk} sR_i^{'} + h_{ijk} x_i P_{rsu_i}$

Generate a valid signature $(R_i^{'}, V_{ijk}^{'})$ on the new signature $m^{'}$.

Correctness: validity of signature $(R_i^{'}, V_{ijk}^{'})$ can be verified as follow.

$$e(V_i^{'}, P) = e(\ psk_{ID_i}.PS2_j + h_{ijk} sR^{'} + h_{ijk} x_i P_{rsu_i}, P)$$

$$= e(psk_{ID_i}.PS2_j, P) e(h_{ijk} sR^{'}, P) e(h_{ijk} x_i P_{rsu_i}, P)$$

$$= e(PS1_j T_j + h_{ijk} R^{'}{}_i, P_{pub}) e(h_{ijk} P_i, P_{rsu_i})$$

Forge signature is valid.

5.7.5 Collision insider attack of Malhi and Batra [22] Certificate Aggregate Signature Scheme

The KGC collaborate with the dishonest user u_n of identity ID_n to forge a certificateless aggregate signature. The steps of the proposed attack are:

Step 1: A corrupt signer u_n selects a random number $v \in z_q^*$, such that $v = x_1 + x_2 .... + x_{n-1} + \gamma$ thereafter u_n can compute $\gamma P = vP - \sum_{i=1}^{n-1} vpk_{IDi}$ and set its public key although u_n does not know about γ.

Step 2: KGC and u_n select $r_1, r_2, ... r_n \in z_q^*$ in collaboration and compute $R_i = r_i P$ and $h_i = H_2(m_i, PS1_i, vpk_{ID_i}, R_i)$, where $i \in [1, n]$

Then, compute $V^* = \sum_{i=1}^{n} psk_{ID_i}.PS2_i + \sum_{i=1}^{n} h_i r_i P_{pub} + \sum_{i=1}^{n} h_i v P_{rsu}$

Since all psk_{ID_i} are known to malicious KGC thereafter generates an aggregate signature $\sigma^* = (R_1, R_2 .... R_n, V^*)$ without using the private key of corresponding identities $\{ID_1, ID_2 .... ID_n\}$.

Step 3: $\sigma^* = (R_1, R_2 .... R_n, V^*)$ be a valid aggregate signature on the message set $\{m_1, m_2, ... m_n\}$ for corresponding identities $\{ID_1, ID_2 .... ID_n\}$ by verification. Validation of aggregate signature can be verified by the following equation:

$$e(V^*, P) = e(\sum_{i=1}^{n} PS1_i T_i + h_i R_i, P_{pub}) e(\sum_{i=1}^{n} h_i vpk_{ID_i}, P_{rsu})$$

Correctness

Validation of aggregate signature can be checked as follows.

$$
\begin{aligned}
e(V^*, P) &= e(\sum_{i=1}^{n} psk_{ID_i}.PS2_i + \sum_{i=1}^{n} h_i r_i P_{pub} + \sum_{i=1}^{n} h_i v P_{rsu}, P) \\
&= e(\sum_{i=1}^{n} \alpha Q_{ID_i}.a_i T_i, P) e(\sum_{i=1}^{n} h_i r_i \alpha P, P) e(\sum_{i=1}^{n} h_i v \delta P, P) \\
&= e(\sum_{i=1}^{n} a_i Q_{ID_i} T_i, P_{pub}) e(\sum_{i=1}^{n} h_i R_i, \alpha P) e(\sum_{i=1}^{n} h_i v pk_{ID_i}, \delta P) \\
&= e(\sum_{i=1}^{n} a_i Q_{ID_i} T_i, P_{pub}) e(\sum_{i=1}^{n} h_i R_i, P_{pub}) e(\sum_{i=1}^{n} h_i v pk_{ID_i}, P_{rsu}) \\
&= e(\sum_{i=1}^{n} (PS1_i T_i + h_i R_i), P_{pub}) e(\sum_{i=1}^{n} h_i v pk_{ID_i}, P_{rsu})
\end{aligned}
$$

5.7.7 Improvement of Malhi and Batra [22] CLAS Scheme

Setup: given input as a security parameter 1^k where $k \in N$ then KGC generates an additive cyclic group and multiplicative group G_2 of the same order q with generator P the bilinear pairing. The KGC generates a master key $s \in Z_q^*$ and set the public key of KGC as . The KGC generates two different hash functions $H_2 : \{0,1\}^* \rightarrow Z_q^*$ and $H_3 : \{0,1\}^* \rightarrow Z_q^*$, $H_4 : \{0,1\}^* \rightarrow Z_q^*$ where message space is $M = \{0,1\}^*$. Each RSU sets a secret key of $y_i \in Z_q^*$ and then a corresponding public key is $P_{rsu_i} = y_i P$. The KGC declares the system parameter list as $\{G_1, G_2, e, P, P_{pub}, H_2, H_3, H_4, P_{rsu_1}, P_{rsu_2} ...., P_{rsu_i}\}$. The next following algorithms Partialkeygen, Userkeygen, PseudonymGen are same as above CLS scheme.

Sign: Given a message $m_k \in M$, a partial private key pp_i, a secret key x_i with vehicle identity Q_{ID_i}, a corresponding public key P_i as input and generates the signature as follow:

i) Select a random $r_i \in Z_q^*$, Compute $U_i = r_i P \in G_1$

ii) Compute $h_{ijk} = H_2(m_k, PS1_j, P_i, U_i) \in Z_q^*$, $t_{ijk} = H_3(m_k, PS1_j, P_i, U_i) \in Z_q^*$

iii) Compute $V_{ijk} = psk_{ID_i}.PS2_j + h_{ijk} r_i P_{rsu_i} + t_{ijk} x_i P_{rsu_i}$

iv) Output (U_i, V_{ijk}) as a signature on m_k.

Verification: For signature verification (U_i, V_{ijk}) of message m_k, the verifier takes the following action.

i) Compute $h_{ijk} = H_2(m_k, PS1_j, P_i, U_i)$, $T_j = H_3(PS1_j) \in Z_q^*$,

$t_{ijk} = H_4(m_k, PS1_j, P_i, U_i) \in Z_q^*$

ii) Verify $e(V_{ijk}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}.U_i + t_{ijk}.P_i, P_{rsu_i})$

iii) If it satisfies then accept otherwise reject the signature

Correctness:

$$\begin{aligned} e(V_{ijk}, P) &= e(pp_i.PS2_j + h_{ijk}.r_i.P_{rsu_i} + t_{ijk}.x_i.P_{rsu_i}, P) \\ &= e(s.Q_{ID_i}.a_jT_j, P)e(h_{ijk}.r_i.P, P_{rsu_i})e(t_{ijk}.x_i.P, P_{rsu_i}) \\ &= e(PS_1.T_j, sP)e(h_{ijk}.U_i, P_{rsu_i})e(t_{ijk}.P_i, P_{rsu_i}) \\ &= e(PS_1.T_j, P_{pub})e(h_{ijk}.U_i, P_{rsu_i})e(t_{ijk}.P_i, P_{rsu_i}) \\ &= e(PS_1.T_j, P_{pub})e(h_{ijk}.U_i + t_{ijk}.P_i, P_{rsu_i}) \end{aligned}$$

Aggregate Verify: Verifier can verify the aggregate signature by the following steps:

1) Compute $h_i = H_2(m_i, PS1_i, P_i, U_i)$, $T_i = H_3(PS1_i)$ for $i \in [1, n]$
2) Compute $t_i = H_4(m_i, PS1_i, P_i, U_i) \in Z_q^*$

Verify the following equation

$$e(V, P) = e(\sum_{i=1}^{n} PS1_i.T_i, P_{pub})e(\sum_{i=1}^{n}(h_{i.}U_i + t_iP_i, P_{rsu})$$

Chapter 6

Cryptanalysis and Improvement of a Certificateless Aggregate Signature Scheme

In this chapter, we have done a cryptanalysis of He et al scheme. We demonstrate that their CLS and CLAS scheme is not secured against 'malicious but passive' attack and 'honest but curious' attacks. We introduce an improved CLAS scheme that covers that security leaks creates in the previous scheme. We proved our CLAS scheme with the Diffe-Hellman assumption under Random Oracle Model. At the end of the chapter, we show that our CLAS scheme is more efficient with some contemporary existing CLAS scheme.

6.1 Formal security model of Certificateless signature (CLS) scheme

Some prilimanary are discussed in chapter 5.

Adversary Model: We describe two types of the adversaries having different powers in the CLS scheme say, Type 1 adversary A_1 and Type 2 adversary A_2. Without loss of generality, adversary A_1 may be any external entity having the potential to access the public key of any user but cannot access the master key of KGC while adversary A_2 may be a malicious KGC having the power to access the master key but cannot change the public key of the user. Any adversary A_1 or A_2 can access five oracles while interacting with the Challenger τ throughout the Game.

- Reveal-Public-Key: When an adversary submits a query on the user identity ID_i , τ returns upk_{ID_i} to the adversary.
- Reveal-Secret-Key: When an adversary submits a query on the user identity ID_i , τ returns usk_{ID_i} to the adversary.
- Reveal-Partial-Key: When an adversary submits a query on the user identity ID_i , τ returns psk_{ID_i} to the adversary.

- Replace-Key: On taking an input ID_i corresponding user secret/public key pair $(usk'_{ID_i}, upk'_{ID_i})$, τ replaces $(usk'_{ID_i}, upk'_{ID_i})$ with the current user secret/public key pair (usk_{ID_i}, upk_{ID_i}).
- Sign: On Submitting a message m_i an adversary can ask for the sign at the identitysponding to identity ID_i, τ returns a valid signature-pair (m_i^*, σ_i^*) corresponding to identity ID_i.

We introduce two games Game 1 and Game 2, wherein adversary A_1 interacts in Game 1 and adversary A_2 interacts in Game 2.

Game 1: τ_1 is a simulator and k is a security parameter in Game 1. This game takes three steps for completion.

1) τ_1 takes an input parameter and generates master/public key pair (s, P_{pub}) of KGC.

2) A_1 can make queries to oracles *Reveal-partial-key-queries, Reveal-secret-key-Queries, Reveal-Public-key-Queries, Replace-Public-Key-Queries, Sign* throughout the game.

3) A_1 can generate the a signature σ_i^* on the message m_i^* with public key $upk_{ID_i}^*$ corresponding to identity ID_i^*.

A_1 can win the Game 1 iff

i) σ_i^* is a valid signature on the message m_i^* with public key $upk_{ID_i}^*$ corresponding to identity ID_i^*.

ii) *Reveal-partial-key-queries* (ID_i^*) have not issued a query corresponding to identity ID_i^* to get partial private key.

iii) *Sign* query has never been submitted corresponding to (ID_i^*, m_i^*).

Definition: A CLS scheme is said to be secure if adversary A_1 cannot win the Game 1 in probabilistic polynomial time with non-negligible probability.

Game 2: τ_2 is a simulator and k is a security parameter in Game 2. This game takes three steps for completion.

1) τ_2 takes an input parameter and generates master/public key pair (s, P_{pub}) of KGC. Assume A_2 has not submitted any query in this phase.

2) A_2 can make queries to oracles *Reveal-secret-key-Queries, Reveal-Public-key-Queries*, *Sign* throughout the game. A_2 does not have permission to query *Reveal-partial-key-queries* for getting a partial private key.

3) A_2 can generate the a signature σ_i^* on the message m_i^* with the public key $upk_{ID_i}^*$ corresponding to identify.

A_2 can win the Game 2 iff

i) σ_i^* is a valid signature on the message m_i^* with public key $upk_{ID_i}^*$ corresponding to identity ID_i^*.

ii) *Reveal-secret-key-Queries* (ID_i^*) have not issued a query corresponding to identity ID_i^* to get user's secret key.

iii) *Sign* query has never been submitted corresponding to (ID_i^*, m_i^*).

Definition: A CLS scheme is said to be secure if adversary A_2 cannot win Game 2 in probabilistic polynomial time with non-negligible probability.

6.2 Review of He *et al.* [28] CL-AS scheme

He *et al.* [28] proposed an efficient scheme with five algorithms *Master-Key-Gen, Partial-Key-Gen, User-Key-Gen, Sign, Verify, aggregate* and *aggregate-verify* algorithms.

Master-Key-Gen: 1) KGC selects two groups G_1 and G_2 of prime order q with two generators P and Q in G_1 and $e: G_1 \times G_1 \rightarrow G_2$, after taking security parameter k.

2) KGC selects a master key $msk \in Z_q^*$ and two hash functions $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow Z_q^*$ and sets its public key $P_{pub} = sP$ where s is the master key of KGC.

3) Then it publishes the system parameters $params = \{q, G_1, G_2, e, P, Q, P_{pub}, H_1, H_2\}$.

Partial-Key-Gen: KGC computes first $Q_{ID_i} = H_1(ID_i) \in G_1$ corresponding to user identity ID_i and then computes user's partial private key $psk_{ID_i} = sQ_{ID_i}$.

User-Key-Gen: The user with identity ID_i selects a secret value $x_{ID_i} \in Z_q^*$ and sets it as its private key and computes corresponding public key $upk_{ID_i} = x_{ID_i} P$.

Sign: Signer corresponding to ID_i performs the following steps for signature.

It computes $U_i = r_i P$ where $r_i \in Z_q^*$, $h_i = H_2(\text{"0"}, m_i, ID_i, upk_{ID_i}, U_i)$, $k_i = H_2(\text{"1"}, m_i, ID_i, upk_{ID_i}, U_i)$

$V_i = psk_{ID_i} + h_i.r_i.P_{pub} + k_i.x_{ID_i}.Q$

$\sigma_i = (U_i, V_i)$ as a signature on m_i.

Verify: Verifier can verify the signature $\sigma_i = (U_i, V_i)$ on the message m_i with respect to identity ID_i corresponding to a public key upk_{ID_i} by performing the following actions.

1. Compute $R_i = r_i P$ where $r_i \in Z_q^*$ $h_i = H_2(\text{"0"}, m_i, ID_i, upk_{ID_i}, R_i)$, $k_i = H_2(\text{"1"}, m_i, ID_i, upk_{ID_i}, R_i)$ and $Q_{ID_i} = H_1(ID_i)$
2. If the equation $e(V_i, P) = e(h_i.R_i + Q_{ID_i}, P_{pub}) e(k_i.upk_{ID_i}, Q)$ holds, then the signature is accepted as it is rejected.

Aggregate: For aggregating an aggregator takes input of the identities $(ID_1, ID_2, ...ID_n)$ of n user's, their public key $(upk_{ID_1}, upk_{ID_2}, ...upk_{ID_n})$ and message signature pair $((m_1, \sigma_1 = (R_1, V_1)), ...(m_n, \sigma_n = (R_n, V_n)))$, thereafter aggregates the signature $V = \sum_{i=1}^{n} V_i$. Finally, generates an aggregate signature $\sigma = (R_1, R_2 .....R_n, V)$.

Aggregate-Verify: For verifying an aggregate signature $\sigma = (R_1, R_2 .....R_n, V)$ verifier performs the following steps:

1. Compute $U_i = r_i P$ where $r_i \in Z_q^*$ $h_i = H_2("0", m_i, ID_i, upk_{ID_i}, R_i)$, $k_i = H_2("1", m_i, ID_i, upk_{ID_i}, R_i)$ and $Q_{ID_i} = H_1(ID_i)$

2. If the equation $e(V, P) = e(\sum_{i=1}^{n} h_i.R_i + Q_{ID_i}, P_{pub})e(\sum_{i=1}^{n} k_i.upk_{ID_i}, Q)$ holds, then the aggregate signature is accepted as it is rejected.

6.3 Security Analysis of He *et al.*[28] CLS scheme

He *et al's* [28] proposed a certificateless signature scheme, but we found it insecure. In the next two subsections, two attacks prove our claim.

6.3.1 Attack 1

This attack is a type 2 attack. In type 2 attack adversary, A$_2$ knows the master key of the KGC and it can find partial private key of the user by making master-key-gen query and gets $psk_{ID_i} = sQ_{ID_i}$ where $Q_{ID_i} = H_1(ID_i) \in G_1$

A$_2$ queries the sign oracle and find a valid signature (R_i, V_i) on the message m_i, Where $U_i = r_i P$, then A$_2$ obtained the value $h_i = H_2("0", m_i, ID_i, upk_{ID_i}, R_i)$, $k_i = H_2("1", m_i, ID_i, upk_{ID_i}, R_i)$ by hash queries.

$V_i = psk_{ID_i} + h_i.r_i.P_{pub} + k_i.x_{ID_i}.Q$ Where $r_i \in Z_q^*$

A$_2$ can intercept information $x_{ID_i}.Q = k_i^{-1}(V_i - psk_{ID_i} - h_i.r_i.P_{pub})$ where k_i^{-1} satisfying $k_i.k_i^{-1} \equiv 1 \bmod q$

$x_{ID_i}.Q = k_i^{-1}(V_i - psk_{ID_i} - h_i.r_i.P_{pub}) = k_i^{-1}(V_i - psk_{ID_i} - h_i.s.R_i)$

A$_2$ can forge any message m_i corresponding public key upk_{ID_i} by using the fixed value $x_{ID_i}.Q$

A2 selects a random $r_i^{'} \in Z_q^*$ and compute $R_i^{'} = r_i^{'}P$ where $r_i^{'} \in Z_q^*$

A2 computes $h_i^{'} = H_2("0", m_i, ID_i, upk_{ID_i}, R_i^{'})$, $k_i^{'} = H_2("1", m_i, ID_i, upk_{ID_i}, R_i^{'})$

$V_i = s.Q_{ID_i} + h_i^{'}.r_i^{'}.P_{pub} + k_i^{'}.(x_{ID_i}.Q)$

$\sigma_i^{'} = (R_i^{'}, V_i^{'})$ is a valid signature on the message on m_i .

As above result, A2 can forge a signature on any message.

Correctness:

$$e(V_i, P) = e(s.Q_{ID_i} + h_i^{'}.r_i^{'}.P_{pub} + k_i^{'}.(x_{ID_i}.Q), P)$$

$$= e(s.Q_{ID_i}, P)e(h_i^{'}.r_i^{'}.P_{pub})e(k_i^{'}.x_{ID_i}.Q, P)$$

$$= e(.Q_{ID_i}, sP)e(h_i^{'}.r_i^{'}.sP, P)e(k_i^{'}.x_{ID_i}.P, Q)$$

$$= e(.Q_{ID_i}, sP)e(h_i^{'}.R_i^{'}, sP)e(k_i^{'}.upk_{ID_i}, Q)$$

$$= e(.Q_{ID_i} + h_i^{'}.R_i^{'}, P_{pub})e(k_i^{'}.upk_{ID_i}, Q)$$

6.3.2 Attack 2

In this attack, type 2 adversary A2 selects a random number t and computes $Q = tP$ at the initial phase of generating system parameters. A2take the following steps to forge the signature.

1. A2 knows the master key of KGC and he can find partial private key of user by making master-key-gen query and gets $psk_{ID_i} = sQ_{ID_i}$ where $Q_{ID_i} = H_1(ID_i) \in G_1$
2. A2 select $U_i = r_i P$, then A2 can query to recover the hash value $h_i = H_2("0", m_i, ID_i, upk_{ID_i}, R_i)$, $k_i = H_2("1", m_i, ID_i, upk_{ID_i}, R_i)$ by hash queries.

$V_i = psk_{ID_i} + h_i.r_i.P_{pub} + k_i.t.upk_{ID_i}$ Where $r_i \in Z_q^*$

$\sigma_i = (R_i, V_i)$ be the output signature on the message on m_i .

Correctness

$$e(V_i, P) = e(psk_{ID_i} + h_i.r_i.P_{pub} + k_i.t.upk_{ID_i}, P) = e(psk_{ID_i}, P)e(h_i.r_i.sP, P)e(k_i.t.upk_{ID_i}, P)$$

$$= e(s.Q_{ID_i}, P)e(h_i.r_i.P, sP)e(k_i.upk_{ID_i}, t.P) \qquad = e(Q_{ID_i}, s.P)e(h_i.R_i, P_{pub})e(k_i.upk_{ID_i}, Q)$$

$$= e(Q_{ID_i}, P_{pub})e(h_i.R_i, P_{pub})e(k_i.upk_{ID_i}, Q) = e(Q_{ID_i} + h_i.R_i, P_{pub})e(k_i.upk_{ID_i}, Q)$$

6.4 Security Analysis of He *et al.*[28] CL-AS scheme

6.4.1 Attack 1

This attack is performed by adversary A_2, A_2 can forge a valid aggregate signature by aggregating the individual signature obtaining in section 6.4.1 attack 1.

1) As per discussion in section 6.4.1 attack 1, A_2 obtained individual signature

$$V_i = s.Q_{ID_i} + h_i^{'}.r_i^{'}.P_{pub} + k_i^{'}.(x_{ID_i}.Q)$$

2) A_2 aggregate the signatures $V^* = \sum_{i=1}^{n} V_i$

3) Finally, A_2 generates $\sigma^* = (R_1, R_2......, R_n, V_i)$ as a forge valid aggregate signature.

Correctness:

$$e(V^*, P) = e(\sum_{i=1}^{n} V_i, P)$$

$$e(V^*, P) = e(\sum_{i=1}^{n} (s.Q_{ID_i} + h_i^{'}.r_i^{'}.P_{pub} + k_i^{'}.(x_{ID_i}.Q)), P)$$

$$= e(\sum_{i=1}^{n} s.Q_{ID_i}, P)e(\sum_{i=1}^{n} h_i^{'}.r_i^{'}.P_{pub}, P)e(\sum_{i=1}^{n} k_i^{'}.(x_{ID_i}.Q)), P)$$

$$= e(\sum_{i=1}^{n} Q_{ID_i}, s.P)e(\sum_{i=1}^{n} h_i^{'}.r_i^{'}.sP, P)e(\sum_{i=1}^{n} k_i^{'}.x_{ID_i}.P, Q)$$

$$= e(\sum_{i=1}^{n} Q_{ID_i}, P_{pub})e(\sum_{i=1}^{n} h_i^{'}.R_i^{'}, P_{pub})e(\sum_{i=1}^{n} k_i^{'}.upk_{ID_i}, Q)$$

$$= e(\sum_{i=1}^{n} (Q_{ID_i} + h_i^{'}.R_i^{'}), P_{pub})e(\sum_{i=1}^{n} k_i^{'}.upk_{ID_i}, Q)$$

6.4.2 Attack 2

This attack is performed by adversary A_2, A_2 can forge a valid aggregate signature by aggregating the individual signature obtaining in section 6.4.2 attack 2.

1) As per discussion in section 6.4.1 attack 1, A_2 obtained individual signature

$V_i = psk_{ID_i} + h_i.r_i.P_{pub} + k_i.t.upk_{ID_i}$

2) A2 aggregate the signatures $V^* = \sum_{i=1}^{n} V_i$

3) Finally, A2 generates $\sigma^* = (R_1, R_2 ......, R_n, V_i)$ as a forge valid aggregate signature.

Correctness

$$e(V^*, P) = e(\sum_{i=1}^{n} V_i, P)$$

$$= e(\sum_{i=1}^{n} (psk_{ID_i} + h_i.r_i.P_{pub} + k_i.t.upk_{ID_i}), P)$$

$$= e(\sum_{i=1}^{n} psk_{ID_i}, P) e(\sum_{i=1}^{n} h_i.r_i.P_{pub}, P) e(\sum_{i=1}^{n} k_i.t.upk_{ID_i}, P)$$
$$= e(\sum_{i=1}^{n} sQ_{ID_i}, P) e(\sum_{i=1}^{n} h_i.r_i.sP, P) e(\sum_{i=1}^{n} k_i.upk_{ID_i}, tP)$$
$$= e(\sum_{i=1}^{n} Q_{ID_i}, sP) e(\sum_{i=1}^{n} h_i.r_i.P, sP) e(\sum_{i=1}^{n} k_i.upk_{ID_i}, Q)$$
$$= e(\sum_{i=1}^{n} Q_{ID_i}, P_{pub}) e(\sum_{i=1}^{n} h_i.R_i, P_{pub}) e(\sum_{i=1}^{n} k_i.upk_{ID_i}, Q)$$
$$= e(\sum_{i=1}^{n} Q_{ID_i} + h_i.R_i, P_{pub}) e(\sum_{i=1}^{n} k_i.upk_{ID_i}, Q)$$

6.4 Our Certificateless signature scheme (CLS) scheme

- *Master-Key-Gen*: 1) KGC selects two groups G_1 and G_2 of prime order q with two generators P in G_1 and $e : G_1 \times G_1 \to G_2$, after taking security parameter k.

2) KGC selects a random number $s \in Z_q^*$ and three hash functions $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to Z_q^*$ $H_3 : \{0,1\}^* \to G_1$ and computes its public key $P_{pub} = sP$ where s is the master key of KGC.

3) Then it publishes the system parameters $\{q, G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3\}$.

- *Partial-Key-Gen*: KGC computes first $Q_{ID_i} = H_1(ID_i) \in G_1$ corresponding to user identity ID_i, and then sets the user private key $psk_{ID_i} = sQ_{ID_i}$.

User-Key-Gen: The user with identity ID_i selects a secret value $x_{ID_i} \in Z_q^*$ and sets it as its private key and computes corresponding public key $upk_{ID_i} = x_{ID_i} P$.

- *Sign:* Signer corresponding ID_i with its private key x_{ID_i} , partial private key psk_{ID_i} performs the following steps to generate the signature.

Compute $R_i = r_i P$ where $r_i \in Z_q^*$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, R_i)$, $W = H_3(\Delta)$, where Δ be a state information.

$V_i = psk_{ID_i} + r_i.W + h_i.x_{ID_i}.W$

$\sigma_i = (R_i, V_i)$ as a signature on m_i .

- *Verify*: Given a signature σ_i on message m_i with identity ID_i verifier performs the following steps:

1. Compute $Q_{ID_i} = H_1(ID_i)$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, R_i)$, $W = H_3(\Delta)$, where Δ be a state information.

2. Check whether $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(R_i + h_i upk_{ID_i}, W)$

If it satisfies then accept the signature otherwise reject.

6.4.1 Our Certificateless aggregate signature scheme (CLAS) scheme

CL-AS scheme consist seven algorithms, five algorithms are same as discuss in section 4. Rests of the algorithms are describing follow.

Aggregate: For aggregating an aggregator takes input of identities $(ID_1, ID_2, ...ID_n)$ of n user's, their public key $(upk_{ID_1}, upk_{ID_2}, ...upk_{ID_n})$ and message signature pair $((m_1, \sigma_1 = (R_1, V_1)), ...(m_n, \sigma_n = (R_n, V_n)))$, thereafter aggregates the signature $V = \sum_{i=1}^{n} V_i$. Finally, generates an aggregate signature $\sigma = (R_1, R_2 .....R_n, V)$.

Aggregate-Verify: For verifying an aggregate signature $\sigma = (U_1, U_2 .....U_n, V)$ signed by n user's having identities $(ID_1, ID_2, ...ID_n)$ corresponding their public key $(upk_{ID_1}, upk_{ID_2}, ...upk_{ID_n})$ on messages $(m_1, m_2 ...m_n)$, verifier performs the following steps:

1. Compute $Q_{ID_i} = H_1(ID_i)$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, R_i)$, $W = H_3(\Delta)$, where Δ be a state information.

2. Check whether $e(V,P)=e(\sum_{i=1}^{n}Q_{ID_i},P_{pub})e(\sum_{i=1}^{n}R_i+h_i upk_{IDi},W)$

Holds then the aggregate signature is accepted otherwise it is rejected.

6.4.1.1 Theorem1: Proposed certificateless signature scheme is unforgeable under the adaptive chosen message and identity attacks under the computational Diffie-Hellman problem.

Proof: For a given random instance $(X=\alpha P, Y=\beta P)$ of the CDH problem in G_1 of prime order q, we will construct an algorithm Ω_1 to solve the CDH problem where $\alpha,\beta\in Z_q^*$ are chosen randomly, unknown to Ω_1. Adversary A_1 cooperates with the τ_1 in the game 1. τ_1 randomly selects as a challenge ID_i and publish the security parameter *params* $\{q,G_1,G_2,e,P,P_{pub},H_1,H_2,H_3\}$ to A_1 and sets $P_{pub}=X$. Now τ_1 is prepared to respond the oracle query. τ_1 preserves a list $(ID_i,x_{ID_i},upk_{ID_i},psk_{ID_i})$, while A_1 can submit queries throughout the game.

i) H_1 *query:* τ_1 preserves a list L_{H_1} in the form of (ID_i,γ_i,Q_{ID_i}). When an adversary A_1 submits a query on identity ID_i to H_1 oracle, τ_1 checks if list L_{H_1} holds the tuple (ID_i,γ_i,Q_i) then nothing needs to be done and Q_{ID_i} to A_1 is returned by τ_1. Otherwise if $ID_i=ID_t$, then τ_1 chooses a random number $\gamma_i\in Z_q^*$ and computes $Q_{ID_i}=\gamma_i Y\in G_1$ and inserts in the list L_{H_1}, thereafter returns to the adversary A_1. Otherwise $ID_i\neq ID_t$, τ_1 picks a random $\gamma_i\in Z_q^*$ and calculates $Q_{ID_i}=\gamma_i P\in G_1$ and inserts in the list L_{H_1}, thereafter returning to the adversary A_1.

ii) H_2 *query*: Ω_1 preserves a list L_{H_2} in the form of $(m_i,ID_i,upk_{IDi},R_i,h_i)$. When adversary A_1 submits a query on identity ID_i to H_2 oracle, τ_1 checks if list L_{H_2} contains the tuple $(m_i,ID_i,upk_{IDi},R_i,h_i)$ then nothing is done and τ_1 returns h_i to A_1. Otherwise Ω_1 chooses a random $h_i\in Z_q^*$ and inserts in the list L_{H_2} and returns h_i to A_1.

iii) H_3 *query*: τ_1 preserves a list L_{H_3} in the form of (m_i, b_i, W_i). When adversary A_1 submits a query on identity ID_i to H_3 oracle, τ_1 checks if list L_{H_3} contains the tuple (m_i, b_i, W_i) then nothing needs to be done and τ_1 responds b_i to A_1. Otherwise τ_1 chooses a random number $b_i \in Z_q^*$ and inserts in the list L_{H_3} and returns b_i to A_1.

iv) *Reveal-partial-key-queries:* In this, a request is submitted on identity ID_i by adversary A_1. If $ID_i = ID_t$ then τ_1 stops the simulation otherwise if L contains a tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then τ_1 checks whether $psk_{ID_i} = \perp$. Otherwise, if $psk_{ID_i} \neq \perp$ then τ_1 responds psk_{ID_i} to. If $psk_{ID_i} = \perp$, τ_1 looks up the list L_{H1} then returns psk_{ID_i} to A_1. If L does not contain a tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then, τ_1 sets $psk_{ID_i} = \perp$ and τ_1 checks the list. Ω_1 selects a number $\alpha_i \in Z_q^*$, sets $psk_{ID_i} = \alpha_i P_{pub} = \alpha_i X \in G_1$. τ_1 responds to the query and returns psk_{ID_i} to A_1.

v)*Reveal-Public-key-Queries*: τ_1 preserves a list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$. When adversary A_1 submits a query corresponding to identity ID_i, τ_1 checks whether $upk_{ID_i} = \perp$. If $upk_{ID_i} \neq \perp$, then τ_1 answer upk_{ID_i} to A_1. If $upk_{ID_i} = \perp$ then τ_1 chooses randomly $v_i \in Z_q^*$ and sets $upk_{ID_i} = v_i P$, τ_1 returns the upk_{ID_i} to A_1 and updates the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$. If L does not contain the list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, τ_1 puts $upk_{ID_i} = \perp$ then τ_1 selects randomly $v_i \in Z_q^*$ and sets $upk_{ID_i} = v_i P$. τ_1 returns the upk_{ID_i} to A_1 and updates the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

vi)*Reveal-secret-key-Queries*: τ_1 preserves a list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$. When an adversary A_1 submits a query corresponding to identity ID_i, τ_1 checks whether $x_{ID_i} = \perp$. If $x_{ID_i} \neq \perp$, then τ_1 answers x_{ID_i} to A_1. If $x_{ID_i} = \perp$ then τ_1 chooses randomly $v_i \in Z_q^*$ and sets $upk_{ID_i} = v_i P$, then τ_1 returns x_{ID_i} to A_1. If τ_1 does not hold list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, τ_1 sets $x_{ID_i} = \perp$ and if $upk_{ID_i} \neq \perp$, then τ_1 select randomly $v_i \in Z_q^*$

and sets $upk_{ID_i} = v_i P$, then τ_1 returns the upk_{ID_i} to A_1 and updates the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

vii)*Replace-Public-Key-queries*: A_1 makes this query on (ID_i, upk_{ID_i}). τ_1 looks up the list L , if list L contains $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then τ_1 replaces upk_{ID_i} with upk'_{ID_i} selected by A_1 and sets $x_{ID_i} = \perp$. Otherwise if list L does not contains upk_{ID_i} then τ_1 sets $upk_{ID_i} = upk'_{ID_i}$ and $x_{ID_i} = \perp$, and $psk_{ID_i} = \perp$ updates the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

viii)*Sign:* When A_1 make a request on (m_i, ID_i) then, τ_1 looks up first the list L, list L_{H_1}, list L_{H_2}, list L_{H_3} then τ_1 does the following steps:

If $ID_i = ID_t$, then τ_1 looks in the list L_{H_1} and list L for the tuple $(ID_i, \gamma_i, Q_{ID_i})$ and $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, where $Q_{ID_i} = \alpha_i Y \in G_1$. If L holds the list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then τ_1 checks whether if $x_{ID_i} = \perp$, If $x_{ID_i} \neq \perp$,then τ_1 goes for *replace-key-query* to generate $x_{ID_i} = v_i$, $upk_{ID_i} = v_i P$. If L does not hold the $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then $\tau_1$makes *Reveal-Public-key* query to generate a pair (x_{ID_i}, upk_{ID_i}) and updates the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

For generating the signature, τ_1 makes query to list L_{H_3} to recover the tuples (m_i, b_i, W_i), where $W_i = b_i X$, then τ_1 select three random numbers $r_i, b_i \in Z_q^*$ and computes $R_i = r_i.P - b_i^{-1}.Q_{ID_i}$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, R_i) \in Z_q^*, W = H_3(\Delta)$

$V_i = r_i.W + h_i.x_{ID_i}.W$

And Ω_1 returns a signature $\sigma_i = (R_i, V_i)$ to A_1 which can be verified easily by equation:

$$e(V_i, P) = e(Q_{ID_i}, P_{pub}) e(R_i + h_i upk_{ID_i}, W)$$

$$e(V_i, P) = e(r_i.W + h_i.x_{ID_i}.W, P)$$

$$= e(r_i.P + h_i.x_{ID_i}.P, W)$$

$$= e(R_i + b_i^{-1}.Q_{ID_i}, W)e(h_i.upk_{ID_i}, W)$$

$$= e(R_i, W)e(b_i^{-1}.Q_{ID_i}, W)e(h_i.upk_{ID_i}, W)$$

$$= e(R_i, W)e(Q_{ID_i}, b_i^{-1}.W)e(h_i.upk_{ID_i}, W)$$

$$= e(R_i, W)e(Q_{ID_i}, X)e(h_i.upk_{ID_i}, W)$$

$$= e(Q_{ID_i}, P_{pub})e(R_i + h_i.upk_{ID_i}, W)$$

The signature generated by $\tau_1, \sigma_i = (R_i, V_i)$ is a valid signature.

If $ID_i \neq ID_t$, τ_1 looks in the list L_{H_1} and list L for the tuple $(ID_i, \gamma_i, Q_{ID_i})$ and $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, where $Q_{ID_i} = \gamma_i P \in G_1$. If L holds the $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then τ_1 checks whether if $x_i = \perp$. If $x_i = \perp$, then τ_1 makes for *replace-key-query* to generate $usk_{IDi} = v_i$, $upk_{IDi} = v_i P$. If L does not contain the $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$ then τ_1 makes *Reveal-Public-key-query* to produce a pair (x_{ID_i}, upk_{ID_i}) and updates the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

For generating the signature, τ_1 makes query to list L_{H_3} to recover the tuples (m_i, b_i, W_i) , where $W_i = b_i X$, then τ_1 selects three random numbers $r_i, \gamma_i, b_i \in Z_q^*$ and set the values $W = b_i P$, computes $R_i = r_i.P$, $h_i = H_2(m_i, ID_i, upk_{ID_i}, R_i) \in Z_q^*$, $W = H_3(\Delta)$ where Δ be a state information.

$$V_i = psk_{ID_i} + r_i.W + h_i.x_{ID_i}.W$$

Ω_1 responds with a signature (U_i, V_i) to A_1 . It is very easy to verify the equation $e(V_i, P) = e(Q_{ID_i}, P_{pub})e(R_i + h_i upk_{IDi}, W)$.

By forgery lemma [26], τ_1 can get two valid signatures $\sigma^* = (R_i^*, V_i^*)$ and $\sigma^{'*} = (R_i^{'*}, V_i^{'*})$ with the same random tape of corresponding message m_i^* and $m_i^{'*}$ with identity ID_i^* and $ID_i^{'*}$, $upk_{ID_i}^*$ and $upk_{ID_i}^{'*}$ public key provide by A_1.

$$V_i^* = psk_{ID_i} + r_i^*.W + h_i^*.x_{ID_i}.W$$

$$V_i^{'*} = psk_{ID_i} + r_i^*.W + h_i^{'*}.x_{ID_i}.W$$

Now τ_1 query to the list L_{H_1} to recover $(ID_i, \gamma_i, Q_{ID_i})$ by setting $Q_{ID_i} = \gamma_i Y = \gamma_i bP$, $W = b_i P$, $psk_{ID_i} = \alpha Q_{ID_i} = \gamma_i \alpha\beta P$, then τ_1 output $\alpha\beta P = \gamma_i^{-1}[(h_i^{*-1} - h_i^{'*-1})^{-1}(h_i^{*-1} V_i^* - h_i^{'*-1} V_i^{'*}) - r_i b_i P]$ as a solution of CDH problem.

Analysis: Three events can be analyzed for finding the success of solving the CDH problem by τ_1 with the probability ς . We take an assumption that A_1 having an advantage ε to forge a signature with in bounded polynomial time span t can submit hash queries at most q_{H_i} times H_i $(i = 1,2,3)$, q_k queries to *Reveal-Partial-key,* q_s queries to *Reveal-secret-key* q_p queries for *reveal-public-key* and q_{sign} queries for sign. Also taking an assumption that A_1 could never repeat H_i query for the same input.

Event 1(E1): τ_1 does not terminate in all queries of *Reveal-partial-key* submitted by A_1.

Event 2(E2): A_1 could forge a valid signature.

Event 3(E3): The output generated by A_1 is valid even τ_1 does not abort all the queries of A_1.

A_1 can win after the happening of all events, that is,

$$P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_1 \mid E_2].P[E_3 \mid E_1 \wedge E_2]$$

The probability of τ_1 does not abort all queries of A_1 is at least $(1-\frac{1}{q_{H_1}})^{q_k}$, where it takes *Reveal-partial-key* at most q_k times.

The probability of A_1's to forge signature is ε .

Probability of valid and nontrivial forgery output of the A_1 valid even $\Im_1$ does not abort is $\frac{1}{q_{H_1}}$.

$$\varsigma = P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_1 \mid E_2].P[E_3 \mid E_1 \wedge E_2]$$

$$\geq (1-\frac{1}{q_{H_1}})^{q_k}.\frac{1}{q_{H_1}}.\varepsilon$$

τ_1 could solve the CDH problem with the probability having more than ε , that is the non-negligible. So, we get a contradiction against the resistance of CDH problem.

6.7.1.2 Theorem 2: In the Random oracle model, adversary A_2 having an advantage ε in forging a certificate less signature scheme wins the Game 2 if it is successful in finding an algorithm which can solve the CDH problem in G_1 with non-negligible probability.

Proof: For a given random instance $(X=\alpha P, Y=\beta P)$ of the CDH problem in G_1 of prime order q, we will construct an algorithm τ_2 to solve the CDH problem where $\alpha, \beta \in Z_q^*$ are random numbers unknown to τ_2. Adversary A_2 collaborates with the Ω_2 in the Game 2. τ_2 chooses a random number $\lambda \in Z_q^*$ and sets the master key of KGC and computes the public key of KGC as $P_{pub} = \lambda P$, then publish the system parameter $\{q, G_1, G_2, e, P, P_{pub}, H_1, H_2, H_3\}$ to A_2 . Since A_2 is type 2 adversary, it can access the master key of KGC so τ_2 and A_2 can compute the partial private key of the user and there is no need to query on hash function H_1 . τ_2 preserves list $L=(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

i) H_2 *query*: τ_2 preserves a list L_{H_2} in the form of $(m_i, ID_i, upk_{IDi}, R_i, h_i)$. When adversary A_2 submits a query on identity ID_i to H_2 oracle, τ_2 checks the list L_{H_2} , if list L_{H_2} holds the tuple $(m_i, ID_i, upk_{IDi}, R_i, h_i)$ then τ_2 respond h_i to A_2 . Otherwise Ω_2 picks a random number $h_i \in Z_q^*$ and inserts in the list L_{H_2} and responds h_i to A_2 .

ii) H_3 *query:* τ_2 preserves a list L_{H_3} in the form of (m_i, b_i, W_i). When adversary A_1 submits a query on identity ID_i to H_3 oracle, τ_2 checks if list L_{H_3} holds the tuple (m_i, b_i, W_i) then τ_2 responds b_i to A_2 . Otherwise τ_2 chooses a random number $b_i \in Z_q^*$ and inserts in the list L_{H_3} and respond b_i to A_2 .

iii)*Reveal-Public-key-Queries:* τ_2 preserves a list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$. When an adversary A_2 request is submitted on identity ID_i , τ_2 checks whether $upk_{ID_i} = \perp$. If $upk_{ID_i} \neq \perp$, then τ_2 answer upk_{ID_i} to A_2 . If $upk_{ID_i} = \perp$ then A_2 chooses a number randomly $v_i \in Z_q^*$ and sets $upk_{ID_i} = v_i P$, then Ω_2 responds upk_{ID_i} to A_2 and updates the tuple $L=(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$. If L does not hold list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, it checks further if $ID_i = ID_t$, then τ_2 chooses a random number $v_i \in Z_q^*$ and compute $upk_{ID_i} = v_i P \in G_1$ and adds in the list L , thereafter responding to adversary A_2 . Otherwise $ID_i \neq ID_t$, τ_2 picks a random number $v_i \in Z_q^*$ and set $upk_{ID_i} = v_i Y \in G_1$ and adds in the list L , thereafter responding to adversary A_2 .

iv)*Reveal-secret-key-Queries*: τ_2 preserves a list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$. When adversary A_2 request is submitted on identity ID_i , Ω_2 checks whether $x_{ID_i} = \perp$. If $x_{ID_i} \neq \perp$, then τ_2 answer x_{ID_i} to A_2 . If $x_{ID_i} = \perp$ then τ_2 select randomly $v_i \in Z_q^*$ and set $upk_{ID_i} = v_i P$, then τ_2 returns x_{ID_i} to A_2 . If τ_2 does not hold the list $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, then τ_2 put $x_{ID_i} = \perp$ and If $upk_{ID_i} \neq \perp$, τ_2 selects randomly $v_i \in Z_q^*$ and set $upk_{ID_i} = v_i P$. τ_2 returns the upk_{ID_i} to A_2 and update the tuple $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$.

v)*Sign:* Description is same as described in theorem 1.

When A_2 submitting a request on (m_i, ID_i) then τ_2 looks up first the list L , list L_{H_2} , list L_{H_3} to recover $(ID_i, x_{ID_i}, upk_{ID_i}, psk_{ID_i})$, $(m_i, ID_i, upk_{IDi}, R_i, h_i)$ and (m_i, b_i, W_i) then τ_2 performs following steps.

By the forgery lemma [26] A_2 can generate the two signatures on the same random tape with different value H_2 .The following two equation held for the two valid signature.

$$e(V_i^*, P) = e(Q_{ID_i}, P_{pub})e(R_i, W)e(h_i^*.P_i^*, W)$$

$$e(V_i^{'*}, P) = e(Q_{ID_i}, P_{pub})e(R_i, W)e(h_i^{'*}.P_i^*, W)$$

τ_2 performs a query on the list L_{H_3} for getting (m_i, b_i, W_i) by setting $W = b_i X$, $X = \alpha P,$ $Q_{ID_i} = \lambda P,$, $upk_{ID_i} = v_i^* Y = v_i^* \beta P.$ Finally, τ_2 respond the value $\alpha\beta P = (v_i^*.b_i)^{-1}((V_i^* - V_i^{'*})(h_i^* - h_i^{'*})^{-1})$ as a solution of CDH problem.

Analysis: Three events can be analyzed for finding the success of solving the CDH problem by τ_2 with the probability ς . We assume that A_2 having an advantage ε to forge a signature with in a time span t can submit hash queries at most q_{H_i} times H_i $(i = 1,2,3)$, q_s queries to *Reveal-secret-key* q_p queries for *reveal-public-key* and q_{sign} queries for sign. Also we assume that A_2 never repeats H_i *query* for the same input.

Event 1 (E1): τ_2 does not terminate in all queries of *Reveal-secret-key* submitted by adversary A_2

Event 2 (E2): A_2 can forge a valid signature.

Event 3 (E3): The output generated A_2 is valid even τ_2 does not abort all the queries of A_2

A_2 can win after occurrence of all the events, that is,

$$P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_1 \mid E_2].P[E_3 \mid E_1 \wedge E_2]$$

The probability of τ_2 not terminating all queries of A_2 is at least $(1-\frac{1}{q_{H_1}})^{q_s}$, where it takes *Reveal-partial-key* at most q_k times.

The probability of τ_2 not aborting key extraction queries and A_2 's signature queries is ε .

Probability of forgery output of the A_2 valid is valid even τ_2 does not abort is $\frac{1}{q_{H_1}}$.

$$\varsigma = P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_1 \mid E_2].P[E_3 \mid E_1 \wedge E_2]$$

$$\geq (1-\frac{1}{q_{H_1}})^{q_s} . \frac{1}{q_{H_1}} . \varepsilon$$

τ_2 could solve the CDH problem with the probability having more than ε , that is the non-negligible. So we get a contradiction against the resistance of CDH problem.

CHAPTER 7

A Certificateless Aggregate Signature Scheme for Healthcare Wireless Sensor Network

Healthcare industry is one of the areas where wireless sensor network provides a lot of opportunities. Online data sharing is one of the requirements to increase the efficiency and reduced the time constraints in the healthcare industry. In the healthcare wireless sensor network, the Patient’s report is available online to share with health professionals without any delay after the patient’s checkup. Data privacy becomes an important issue in healthcare due to direct involvement of personal health related data of patients. Modified data may become a serious cause of casualty for the patient. Digital signature scheme is a technique of public key cryptography that is used to retain the privacy and integrity in our system. Certificateless public key cryptography was proposed to remove the complication of certificate management in public key cryptography as well as the key escrow problem inherited in identity based cryptography. An aggregate signature scheme is a many to one map which maps different signatures on different messages to a single signature. This feature is very beneficial in an environment which is constrained by limited bandwidth and low computational time/effort, such as wireless sensor network, vehicular ad-hoc network and Internet of things. Our proposed certificateless aggregate signature enjoys the goodness of both the concepts, certificateless and aggregate. This chapter proposes a certificate less aggregate signature scheme and prove the security of the proposed scheme by using widely-accepted Random Oracle Model under the computational hard Diffie-Hallman assumption. Random Oracle Model based security analysis prove that our proposed scheme is provably secure against existential forgery on adaptive chosen message and identity attacks under the hardness of computational Diffie-Hellman problem and achieve the required goal such as confidentiality, non-repudiation, integrity. We use a batch verification technique for speedy verification of signatures. Results are evaluated by using NS 2 environment. The simulation results show that our scheme is most efficient in comparison of previous CLAS schemes.

7.1 Introduction

Wireless sensor network technologies have a broad area with many practical applications in retail, entertainment, medicine, travel, industry, emergency management and many other areas [6]. Healthcare is one from among these areas in which wireless sensor networking provides a lot of new opportunities. Sensor based technology has invented many medical tiny devices replacing thousands of wires connected with the devices situated in the hospitals and enhanced the mobility. Researchers make a broad vision of healthcare by integrating computer, networking and medical fields.

There are two major categories in healthcare application: medical applications, nonmedical applications [7]. The devices used in medical applications can also be divided into two types: Wearable devices and implanted devices. Wearable devices are used on the surface of a human body or kept just close to the human body (for e.g. Temperature measurement, Respiration monitor, Heart rate monitor, Pulse oximeter SpO2, Blood pressure monitor, pH monitor, Glucose sensor etc). The human body can move along with the wearable devices. On the other hand, in case of the implanted devices, the human body is injected in/with the medical devices (for e.g. Cardiac arrhythmia monitor/recorder, Brain liquid pressure sensor, Endoscope capsule etc). The devices used in non-medical applications are the handheld devices such as personal digital assistants (PDA), laptop, phones etc.,

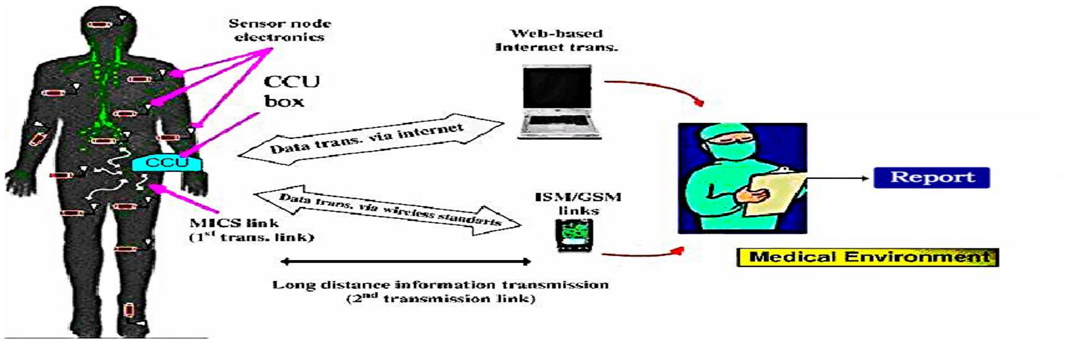
Sensor node electronics
CCU box
CCU
Web-based Internet trans.
Data trans. via internet
Data trans. via wireless standarts
ISM/GSM links
MICS link (1st trans. link)
Report
Medical Environment
Long distance information transmission (2nd transmission link)

Fig 7.1: Architecture of healthcare wireless sensor network (HWSN)[5]

A general architecture of healthcare wireless sensor network proposed by Yuce et al. [6] shown in figure 7.1, in which sensors, patient, internet and healthcare professionals are four components. Sensors are implanted in the patient's body and transferred the patient’s health data to healthcare professionals via internet. Healthcare professionals read the data comes from the sensors and prepare their report.

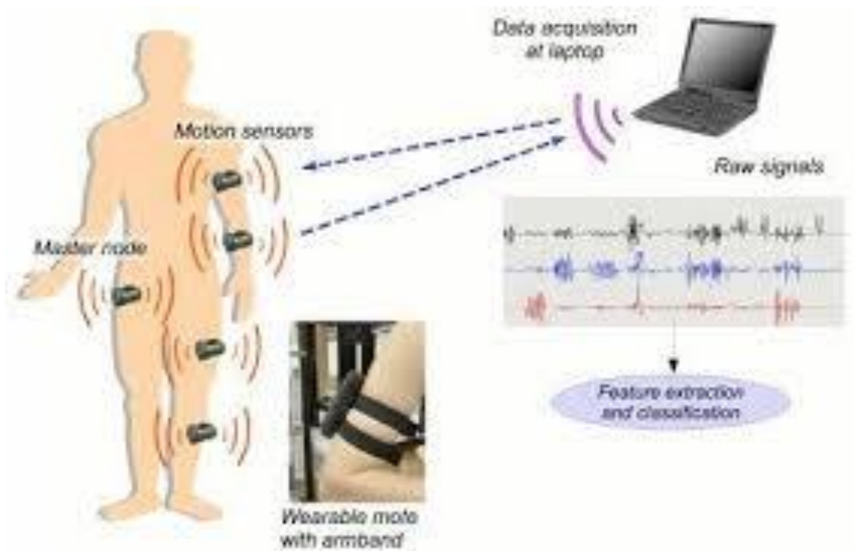
Data acquisition at laptop
Motion sensors
Raw signals
Master node
Feature extraction and classification
Wearable mote with armband

Fig 7.2: Code Blue project

A popular research project on a healthcare wireless sensor network (HWSN) named as Code Blue developed in the Harvard Sensor Network Lab (53). In this project, several medical sensors [7] (pulse oximeter, EMG, EKG,) are put on the patient's body to take the patient’s body data and transfer this to the end user devices for analysis. CodeBlue works on the idea that a doctor or other medical professional (e.g. Lab technician) submit a query for health data by using their personal digital assistant (PDA). The medical sensors respond data related to the query through a public channel, thereafter authentic user subscribes this channel by using their handheld devices [51, 52, 53].

Security and privacy issues are very sensitive in the HWSN due to direct involvement of patient’s personal health data. Information is transferred from sensor devices to the healthcare professionals who analyze this information and deliver appropriate solutions. If an attacker modifies the information in the midway then health care professionals receive this modified information, thereafter they diagnose according to the modified information. This may be extremely dangerous for the human life. Motivated with this scenario, this chapter proposes a certificatelessaggregate signature scheme for healthcare wireless sensor network (HWSN) to retain the integrity and privacy of health related information and thereby protecting the medical network from adversaries.

Al-riyami and Peterson [9] introduced the concept of certificateless signature scheme (CLS) which not only provides the solution of certificate management [3] in public key cryptography, but also resolves the key escrow problem inherited in identity based signature schemes (ID-PKC) [8]. In ID-PKC [8] private key is generated by a trusted third party, say, private key generator (PKG). Although, we have assumed that PKG is a trusted authority which cannot corrupt or make collaboration with the malicious adversaries. However, in a practical scenario, if PKG becomes malicious then private key can be compromised easily. In CLS scheme, trusted third party, say, key generation center (KGC) generates the partial private key instead of the original private key of the user and the private key is generated by the user with the help of partial private key. This resolves the issue of the key escrow problem.

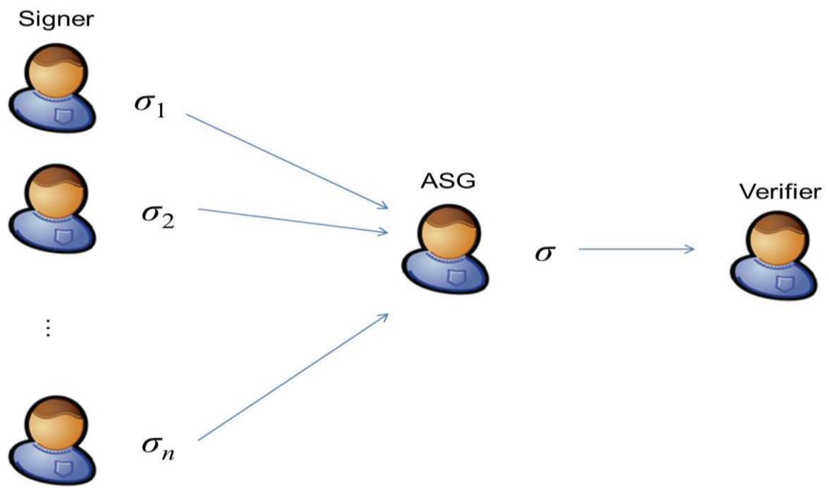

Fig 7.3: General view of Aggregate Signature

The concept of aggregate signature was introduced by Boneh et al. [11] in Eurocrypt 2003. An aggregate signature scheme maps n-signatures on n-different messages to a single signature. This process reduces the bandwidth and computational effort owing to the use of a single signature. Since devices used in HWSN are tiny devices which have limited storage power and less bandwidth, an aggregate signature becomes a suitable choice in the healthcare industry.

7.2 Related work:

Horng et al. [59] used the batch verification in their proposed CL-AS scheme which reduces the computational cost taken in the verification process of the signature. Recently, He and Zeadally [64] proposed an authentication protocol for Ambient Assisted Living (AAL) system, which provides the healthcare monitoring and tele-health services by leveraging information and communication technologies. He et al. [65] construct an efficient certificateless public auditing (CLPA) scheme for cloud-assisted wireless body area network. Many architectures constructed by the researchers for HWSN [66, 67, 68]. We propose a novel CL-AS scheme for secure communication in HWSN. The proposed CL-AS scheme achieves authentication, message integrity, non-repudiation and confidentiality. Our CL-AS scheme uses batch verification algorithms [46, 47] to enhance the verification process an aggregate signature by reducing the computational overheads.

7.3 System Model

We adopt the system model same as in [62]. The consideration of our system model is to provide authenticity, integrity and confidentiality to rule out the chances of false data transmission. It consists of four entities as shown in Fig 7.4: a larger number of sensors, Medical server (MS), Authorized healthcare professionals and aggregator. We assume the provision of Care-District, according to which one Cares-District is used to monitor one disease (for example, to monitor heart pulse rate all sensors used to find heart pulse rate are from one Care-District).

- Authorized healthcare professional's takes the data sensed by the sensors that have certain capabilities of calculation and communication. They analyze the data to provide an appropriate prescription for patients.
- Medical server has a strong computing power and storage space and is capable to process the large amount of data received by sensors. After processing, MS transfers the patient's information to the healthcare professionals. In the initialization phase, MS generates its private- public key pair (s, MS_{pub}) and

makes the public key MS_{pub}public. MS is also authorized to generate a partial private key for each sensor and transfers it to sensors via a secure channel.

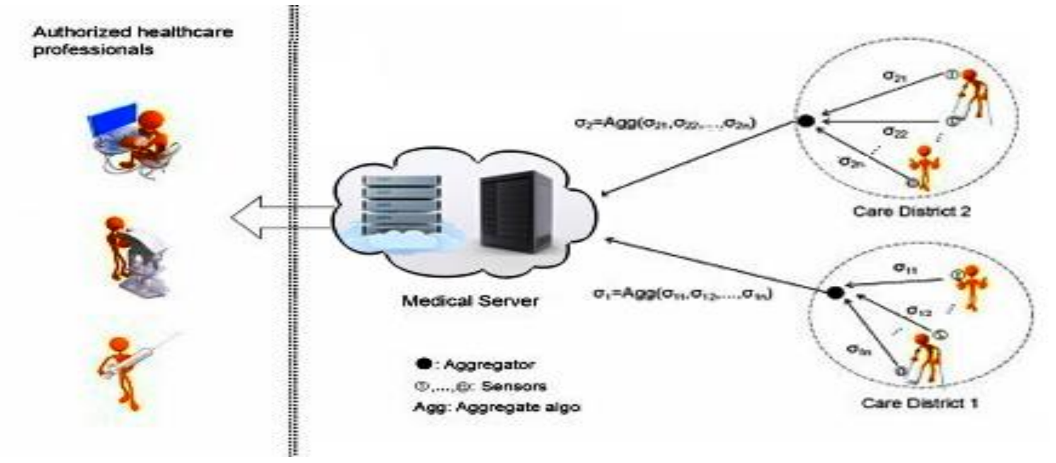

Fig 7.4: Framework of healthcare wireless sensor network

- The aggregator has limited power of calculation and computation. It collects the signatures from a Care-District, generates an aggregate signature, which it transfers to MS. We take an assumption that every Care-District contains one aggregator and many sensors.
- Sensors are resource limited small devices. We take assumptions that each sensor with identity ID_i and embedded with its private-public key pair $\left(SK_{ID_i}, PK_{ID_i}\right) = \left(x_i, PK_{ID_i}\right)$belongs to a Care-District. Each sensor with their respective identity has capability to generate a signed message from the physical world using its private key x_i, thereafter it transfers this signature to the aggregator.

Our CL-AS construction has four entities as discuss above: sensors on the patient's body, Medical server (MS), Healthcare professional and aggregator. MS generates the system parameters, generates partial private keys for sensors corresponding to their identities and transfers this partial private key to the sensors via a secure channel. Sensors chose their private key to generate a signed message in their respective Care-Districts. One of the sensors in the Care-District acts as the aggregator. It aggregates all signatures of its respective Care-District and transfers the single aggregate signature to the handheld devices of the healthcare professional via MS. Healthcare professionals verify the received aggregate signature in their handheld devices. Though the information is transferred via MS but MS cannot capture information due to lack of knowledge of the private key of the sensor. We use batch verification in our construction for speedy verification of signatures.

Definition 1: A function $\emptyset(i)$ is called negligible if for a given $\varepsilon \geq 0$ there exists a number t_0 such that$\emptyset(t) \leq 1/t^{\varepsilon}$ for every $t \geq t_0$.

Definition 2: *Computational Diffie-Hellman (CDH) problem*: For a given random instance$e(P, \alpha P, \beta P) \in G_2$, where $\alpha, \beta \in Z_q$ with P as generator of G_1 having order q, it is computationally hard to find α,$\beta \in G_1$.

Definition 3:*CDH assumption:* For every probabilistic polynomial time (PPT) bound there exists an algorithm T such that$Adv_{T,G_1}^{CDH} \leq \theta$ for some negligible function θ, where Adv_{T,G_1}^{CDH} is the advantage given to an adversary to solve CDH problem using algorithm T.

7.4 Some special Symbols and their descriptions

Here, we introduce some special symbols used in this chapter in Table 7.1 given below

Table 7.1: Some symbols used in the chapter 7

Symbols	**Description**
MS	Medical server
α	The master key of *MS*
P	Generator of the group
MS_{pub}	The public key of *MS*
ID_i	The sensor's identity
ppk_{ID_i}	The partial private key of the sensor with identity ID_i.
(x_i, PK_i)	Priva private / public key pair of the sensor with identity ID_i.
PK_{ID_i}	The public key of the sensor with

	identityID_i

7.5 Security Model of a CLS and CL-AS Scheme

Here, we design an adversary model for CLS and CL-AS schemes. We design two levels of securities in the proposed scheme: Type 1 security level and Type 2 security levels. There are two types of adversaries, A_1 andA_2. In general,A_1 and A_2 are involved in CLS scheme with different powers. A_1Is an outside attacker and $A_2$is a part of the system, say, malicious KGC, who is responsible for generating the partial-private key of the user. Description of power of adversaries is given below.

- **Adversary A_1:**$A_1$is capable to replace the public key of a user, but cannot obtain the master key of KGC.
- **Adversary A_2:** $A_2$is malicious KGC and can access the master key of KGC, but have no power to replace the user's public key.

Definition 4: A CLS/CL-AS scheme is said to be existentially unforgeable against adaptive chosen message and identity attacks, if both the adversaries A_1 and A_2 have negligible probabilities to forge the valid signature.

A_1 and A_2can access the following six oracles:

Create-User: While an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$. Oracle checks whether proper entry corresponding to the target identity is available in the database.If found then returns$PK_{ID_i}$to the adversary otherwise it executes*Reveal-Partial-private-key*and *Reveal-Secret-key* queries to find a partial private key ppk_{ID_i}, private key $x_{i.}$Thereafter, the Oracle computes $PK_{ID_i}$and inserts in the list $L = (ID_i, x_{i,} PK_{ID,} ppk_{ID_i})$. Finally,$PK_{ID_i}$returns to the adversary.

Reveal-Partial-Private-Key: While an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$, Oracle checks whether proper entry corresponding to the target identity is available in the database. If found then returns $ppk_{ID_i}$to the adversary otherwise it returns $\perp$.

Reveal-Secret-Key: While an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$, Oracle checks whether the proper entry corresponding to the target identity is available in the database. If found then returns $x_i$to the adversary otherwise it returns $\perp$.

Replace-Public-Key: When an adversary submits a query on a target identity$ID_i \in \{0,1\}^*$and private/public key pair$(x_i, PK^*_{ID_i})$. Then the oracle searches the list$L$, if proper entry corresponding to the target identity is not available in the database, then no need to perform anything otherwise this oracle updates the list $L = (ID_i, x_{i,} PK_{ID,} ppk_{ID_i})$to $L = (ID_i, x_{i,} PK^*_{ID}, ppk_{ID_i})$.

Sign: While an adversary submits a query on the message with target signer's identity$ID_i \in \{0,1\}^*$, Oracle executes one of the following activities.

iv) Returns a valid signature σ_i without replacing private/public key pair if the target identityID_ihas been formed without swapping private/public key pair.

v) Returns $\perp$ if target identity $ID_i$is not created.

vi) Returns the signature $(x_i, PK^*_{ID_i}, m_i)$ after replacing the public key.

We design two Games: Game I and Game II. Here, Game I and Game II are designed for A_1 and A_2 in CLS scheme, respectively.

Game I: τ_1 is the challenger/simulator that interacts with adversary A_1. This Game performs the following steps.

- Step1: $\tau_1$executes the *Setup* algorithm, which takes a security parameter k as input, produces a master key of KGC and a list of system parameters. Thenτ_1 transfer the system parameters to A_1 while keeping the master key secret.
- Step 2: In this step, A_1 can submit*Reveal-Partial-Private-Key, Reveal-Secret-Key, Reveal-Public-Key, Replace-Public-Key* and *Sign* queries at any stage during the simulation in polynomial bound.
- Step 3: A_1 outputs a signature σ^*_i on a message m^*_i corresponding to a targeted identity ID^*_i with public key $PK_{ID^*_i}$.

A_1successfully wins the game if any one of the following conditions is satisfied.

vi) $\sigma^*_i$is a valid signature on m^*_i under ID^*_i.

vii) Oracle has never been performing the *Reveal-Partial-Private-Key query* for getting the partial private key corresponding to the targeted identity ID^*_i .

viii) *Sign* oracle has never been performed for m^*_i with the targeted identity ID^*_i.

Definition 5: A CLS scheme is called Type 1 secure if there does not exist any adversary A_1 who wins the Game I in probabilistic polynomial time bound with non-negligible advantage.

Game II: τ_2 is the challenger/simulator that interacts with adversary A_2. This Game performs the following steps.

- Step1: τ_2 executes the *Setup* algorithm, which takes a security parameter k as input and produces a master key of KGC and a list of system parameters. Then τ_2 transfers the system parameters to A_2 while keeping the master key secret.
- Step 2: In this step, A_2 can submit *Reveal-Partial-Private-Key, Reveal-Secret-Key, Reveal-Public-Key, Replace-Public-Key* and *Sign* queries at any stage during the simulationin polynomial bound.
- Step 3: A_2 outputs a signature σ_i^* on a message m_i^* corresponding to a targeted identity ID_i^* with public key $PK_{ID_i^*}$.

A_2 successfully wins the game if any one of the following conditions is satisfied.

iv) σ_i^* is a valid signature on m_i^* under ID_i^*.

v) Oracle has never been executing the *Reveal-Secret-Key* for getting the secret key corresponding to the targeted identity ID_i^*.

vi) *Sign* oracle has never been executed for m_i^* under the targeted identity.

Definition 6: A CLS scheme is called Type 2 secure if there does not exist any adversary A_2 who wins the Game II in probabilistic polynomial time bound with non-negligible advantage.

We design one more game: Game III. Here, Game III is executed by A in CL-AS scheme.

Game III: τ is the challenger/simulator that interacts with adversary A. The following steps are performed in this Game.

- Step1: τ executes the *Setup* algorithm, which takes a security parameter k as input and produces a master key of KGC and a list of system parameters. Then τ transfers the system parameters to A while keeping the master key secret.

- Step 2: In this step, A can submit *Reveal-Partial-Private-Key*, *Reveal-Secret-Key*, *Reveal-Public-Key*, *Replace-Public-Key* and *Sign* queries at any stage during the ssimulation inpolynomial bound.
- Step 3: A outputs a signature σ_i^* on a set of messages $\{m_1^*, m_2^* \dots \dots . . m_n^*\}$ corresponding to the targeted identity set $\{ID_1^*, ID_2^* \dots \dots ID_i^*\}$with the corresponding public key set $\{PK_{ID_1}^*, PK_{ID_2}^* \dots \dots . PK_{ID_n}^*\}$.

A successfully wins the game if any one of the following conditions is satisfied.

i) $\sigma_i^*$is a valid signature on $\{m_1^*, m_2^* \dots \dots . . m_n^*\}$ under $\{ID_1^*, ID_2^* \dots \dots ID_i^*\}$ with public key set $\{PK_{ID_1}^*, PK_{ID_2}^* \dots \dots . PK_{ID_n}^*\}$.

ii) Oracle has never been performing the *reveal-partial-private-key query* for getting the partial private key corresponding to the targeted identity ID_i^* .

iii) *Sign* oracle has never been performed for m_i^* under the targeted identity ID_i^*.

Definition 7: A CL-AS scheme is called secure if there does not exist any adversary A who wins the Game III in probabilistic polynomial time bound with non-negligible advantage.

7.6 Proposed Certificateless Signature Scheme

In general, our CLS construction consists of five algorithms *Setup*, *Partial-Private-Key-Gen*, *Private-Key-Gen*, *Sign*,and *Verify*. Descriptions of these algorithms are given below:

Setup: MS runs this algorithm after taking a security parameter k as input. The following steps are performed:

Generates two cyclic groups G_1 and G_2 having the same order q with generator P of $G_1$and an admissible bilinear pairing $e: G_1 \times G_1 \rightarrow G_2$

viii) Selects a random number $\in Z_q^*$, computes $MS_{pub} = \alpha P$and setsα as a master key of MS and MS_{pub} as a public key of MS.

ix) Selects three one way cryptographic hash functions $H_1: \{0,1\} \rightarrow G_1$,$H_2: \{0,1\} \rightarrow G_1$, $H_3: \{0,1\} \rightarrow Z_q^*$.

x) Finally, publishes the system parameters $\{q, G_1, G_2, e, P, MS_{pub}, H_1, H_2, H_3\}$ called $Params$ and keeps the master key α secret.

Partial-Private-Key-Gen: Taking a sensor's identityID_i, MS computes$Q_{ID_i} = H_1(ID_i)$, $ppk_{ID_i} = \alpha. Q_{ID_i}$, sets$ppk_{ID_i}$ as partial private key and transfers it via a secure channel to the corresponding sensor.

Private-Key-Gen: A sensor of identity ID_i selects a random numberx_i, set its secret key and computes $PK_{ID_i} = x_i P$.Here PK_{ID_i}Is the public key of the sensor with identity ID_i.

Sign: A signer with identity ID_i takes the $params$ (system parameters), a partial private key ppk_{ID_i}, its secret keyx_i, state information Δ (we can select some element in public parameter such as Δ) and private-public key pair (x_i, PK_{ID_i}). Then, the signer generates a signature σ on the message m_i as follows:

i) Selects a random number $r_i \in Z_q^*$and computes $R_i = r_i P$

ii) Computes $W = H_2(\Delta)$, $h_i = H_3(m_i, ID_i, PK_{ID_i}, R_i)$

iii) Computes $V_i = ppk_{ID_i} + r_i W + h_i x_i MS_{pub}$

The output is a signature (R_i, V_i) on a message m_i.

Verify: A verifier takes a signature $\sigma_i = (R_i, V_i)$ of messagem_i on identity $ID_i$with public key PK_{ID_i} and state information Δ . Computes $Q_{ID_i} = H_1(ID_i)$, $= H_2(\Delta)$, $h_i = H_3(m_i, ID_i, PK_{ID_i}, R_i)$

i) Verifies$e(V_i, P) = e(Q_{ID_i} + h_i PK_{ID_i}, MS_{pub}) e(R_i, W)$

If the verification equation holds, then accepts the signature otherwise rejects.

Correctness:

$$\begin{aligned} e(V_i, P) &= e\big(ppk_{ID_i} + r_iW + h_ix_iMS_{pub}, P\big) \\ &= e\big(ppk_{ID_i}, P\big)e(r_iW, P)e\big(h_ix_iMS_{pub}, P\big) \\ &= e\big(\alpha Q_{ID_i}, P\big)e(r_iP, W)e\big(h_ix_iP, MS_{pub}\big) \\ &= e\big(Q_{ID_i}, \alpha P\big)e(r_iP, W)e\big(h_ix_iP, MS_{pub}\big) \\ &= e\big(Q_{ID_i}, MS_{pub}\big)e(r_iP, W)e\big(h_iPK_{ID_i}, MS_{pub}\big) \\ &= e\big(Q_{ID_i} + h_iPK_{ID_i}, MS_{pub}\big)e(r_iP, W) \end{aligned}$$

*Aggregate:*An aggregator collects all signatures on the message $\{m_1, m_2 \ldots\ldots..m_n\}$ of the sensors $\{S_1, S_2, .\ldots\ldots..S_n\}$ with corresponding identities $\{ID_1, ID_2, .\ldots\ldots..ID_n\}$ & public keys $\{PK_1, PK_2, .\ldots\ldots..PK_n\}$ and generates an aggregate signature. The aggregator computes $V = \sum_{i=1}^{n} V_i$ and generates an output as aggregate signature $\sigma = (R_1, R_2 \ldots\ldots R_n, V)$.

Aggregate-Verify: Taking a signature $\sigma_i = (R_i, V_i)$ of messagem_i on identity $ID_i$with public key PK_{ID_i} and state information Δ.

i) Computes$Q_{ID_i} = H_1(ID_i)$, $W = H_2(\Delta)$, $h_i = H_3(m_i, ID_i, PK_{ID_i}, R_i)$

ii) Verifies$e(V_i, P) = e(\sum_{i=1}^{n}(Q_{ID_i} + h_iPK_{ID_i}, MS_{pub})e(\sum_{i=1}^{n} R_i , W)$

If verification equation holds then accepts the signature otherwise rejects.

In this section, we prove that our CLS and CL-AS schemes are unforgeable against adaptive chosen message in the random oracle model with the help of following theorems.

7.7 Security proofs

7.7.1 Theorem 1

In a random oracle model, A_1 be a forger having advantage ε to forge a signature in a modelled attack game within running time t and making queries to various oracles by making q_{H_i} queries to oracle $H_i$for $i = 1,2,3 q_k$queries to *Reveal-partial-private –key* , q_s queries to *Reveal-Secret-Key*, q_p queries to *Reveal-Public-Key* and q_{sig} queries to sign, then the CDH problem can be solved with probability $\varepsilon' > \frac{\varepsilon}{e.(q_k+1)}$ with time $t' < t + \left(q_{H_1} + q_{H_2} + q_{H_3} + q_k + q_s + q_p + q_{sig}\right)t_m + \left(q_{sig} + 1\right)t_{mm}$ wheret_m is the time to compute a scalar multiplication in G_1 and t_{mm} is the time to perform a multi exponentiation in G_1.

Proof: Let $(P, X = aP, Y = bP) \in G_1 \times G_1$ be a random instance of the CDH problem, here$P$is the generator of group G_1 of prime order q and the numbers a and b are chosen randomly fromZ_q^*. Our target is to solve the CDH problem as to compute abP. We will construct an algorithm $\tau_1$to achieve our target.

Adversary A_1 cooperates with theτ_1 in the game I. $\tau_1$randomlypicks an identity as a target sensor's identity ID_i and sends the security parameter $params = \{G_1, G_2, e, P, MS_{pub}, H_1, H_2, H_3\}$ to A_1. $\tau_1$selects$c \in Z_q^*$ and sets $MS_{pub} = X$. Now τ_1 is ready to execute the oracle query. $\tau_1$maintains a list $L = (ID_i, x_{i,} PK_{ID,} ppk_{ID_i})$ whenever submits query throughout the game.
H_1query: While submitting an identity to H_1 oracle , according to Coron's proof [63], $\tau_1$flips a coin $X \in \{0,1\}$ that returns 0 with probability £ and 1 with probability $1 - £$ and selects randomly $\beta_i \in Z_q^*$. If $X_i = 0$ then the value of a hash function $H_1(ID_1)$can be set as $Q_{ID_i} = \beta_i P \in G_1$ otherwise $\tau_1$returns $Q_{ID_i} = \beta_i Y \in G_1$. Then τ_1 maintains a list $L_{H_1} = (ID_i, \beta_i, X_i, Q_{ID_i})$to answer the queries of A_1.
H_2query: τ_1 maintains a list $L_{H_2}$with the tuple (m_i, c, W). When A_1 submits the query of H_2 oracle, τ_1 checks if list L_{H_2}holds the tuple (m_i, c, W). If yes, nothing to be done and τ_1 returns W to A_1. Otherwise $\tau_1$picks a random $\in Z_q^*$, sets$W = cX$ and inserts it in the list L_{H_2}. Finally,it returns$W$ to A_1.
H_3query:τ_1 maintains a list $L_{H_3}$with the tuple$(m_i, ID_i, PK_{ID_i}, R_i, h_i)$. When A_1 submits the query of H_3 oracle,τ_1 checks if list L_{H_3}holds the tuple $(m_i, ID_i, PK_{ID_i}, R_i, h_i)$.If yes,nothing to be done and τ_1 returns h_i to A_1. Otherwise τ_1 picks a random $h_i \in Z_q^*$,updates it in the list L_{H_3} and returns$h_i$to A_1.
*Reveal-Partial-Private-Key-queries:*When A_1 submits a query on an identityID_i,τ_1 recalls the corresponding tuple $(ID_i, \beta_i, X_i, Q_{ID_i})$ from the list L_{H_1}. If $X_i = 1$then τ_1

returns failure and stops the simulation. Otherwise $\tau_1$checks the list L and executes the following:

When $X_i = 0$, if$L$contains a tuple $(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$ then τ_1 checks whether$ppk_{ID_i} = \perp$. If $ppk_{ID_i} \neq \perp$then τ_1 answers $ppk_{ID_i}$to A_1. If $ppk_{ID_i} = \perp$, then τ_1 checks the list L_{H1} and returns ppk_{ID_i} to A_1. If L does not contain a tuple $(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$ then τ_1 puts $ppk_{ID_i} = \perp$,look up the listL_{H_1}, puts $ppk_{ID_i} = \beta_i MS_{pub} = \beta_i X \in G_1$. $\tau_1$answers the query and returns$ppk_{ID_i}$to A_1.

Reveal-Secret-Key-queries: When adversary A_1 submits a request on identity ID_i , τ_1 maintains a list $(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$. τ_1 checks whether $x_i = \perp$. If $x_i \neq \perp$, then τ_1 returns x_i to A_1 . If $x_i = \perp$ then τ_1 randomlyselects $u_i \in Z_q^*$, puts $PK_{ID_i} = u_i P$ and returns $u_i$toA_1.

If τ_1 does not holdthe list $(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$, τ_1 set$x_i = \perp$. If $PK_{ID_i} \neq \perp$, then τ_1 randomly selects $u_i \in Z_q^*$, sets $PK_{ID_i} = u_i P$, returns x_i to A_1 and updates the tuple $(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$.

Replace-Public-Key-queries: When adversary A_1 is submits a request on(ID_i, PK_{ID_i}).$\tau_1$makes this query on (ID_i, PK_{ID_i}) and checks the list L , if list L holds$(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$ then τ_1 updates $PK_{ID_i}$with PK'_{ID_i}selected by A_1 and sets$x_i = \perp$. Otherwise if list L does not contains PK_{ID_i}then τ_1 sets$PK_{ID_i} = PK'_{ID_i}$and $x_i = \perp$, and $ppk_{ID_i} = \perp$, updates the list$(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i})$.

Sign: WhenA_1 submits a request on(ID_i, m_i),$\tau_1$checks the list L, $L_{H_1}, L_{H_2}, L_{H_3}$ and performsthe following.

If the list holds $\left(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i}\right)$, $\tau_1$checks whether $x_i = \perp$. If $x_i \neq \perp$ then returns $PK_{ID_i}$to A_1. If $x_i = \perp$, then τ_1 creates query on *Reveal-Public-key*to generate $PK_{ID_i} = u_i P$ where $u_i \in Z_q^*$.

If the list does not hold $\left(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i}\right)$ then τ_1 creates query on *Reveal-Public-key*itself on $ID_i$and updates the list$\left(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i}\right)$.

For generating the signature, $\tau_1$makes a query to list L_{H_2} for getting tuples (m_i, c, W), where $W = cX$.Then τ_1 choses two random numbers $r_i, \delta_i \in Z_q^*$ and sets $R_i = r_iP - c^{-1}Q_{ID_i}$,

Computes $W = H_2(\Delta)$, $h_i = H_1(m_i, ID_i, PK_{ID_i}, R_i)$

$$V_i = r_iW + h_iu_iMS_{pub}$$

And τ_1 generates a signature $\sigma_i = (R_i, V_i)$ which it sends to A_1 as a response of sign queries where A_1can verify the validity of the signature by verification equation.

$$e(V_i, P) = e(R_i, W)e(Q_{ID_i} + h_iMS_{pub}, W)$$

Correctness:

$$\begin{aligned} e(V_i, P) &= e(r_iW + h_iu_iMS_{pub}, P) \\ &= e(r_iW, P)e(h_iu_i\alpha P, P) \\ &= e(r_iP, W)e(h_iu_iP, MS_{pub}) \\ &= e(R_i + c^{-1}Q_{ID_i}, W)e(h_iPK_{ID_i}, MS_{pub}) \\ &= e(R_i, W)e(Q_{ID_i}, c^{-1}W)e(h_iPK_{ID_i}, MS_{pub}) \\ &= e(R_i, W)e(Q_{ID_i}, MS_{pub})e(h_iPK_{ID_i}, MS_{pub}) \\ &= e(R_i, W)e(Q_{ID_i} + h_iPK_{ID_i}, MS_{pub}) \end{aligned}$$

Based on the response of submitting the query by A_1, τ_1 can find two valid signatures on the same random tape as $\sigma_i^* = (m_i^*, ID_i^*, R_i^*, h_i^*, V_i^*)$ and $\sigma_i^{*'} = (m_i^*, ID_i^*, R_i^*, h_i^{*'}, V_i^{*'})$ within polynomial time.

$$V_i^* = ppk_{ID_i^*} + r_i^*W + h_i^*x_i^*MS_{pub}$$

$$V_i^{*'} = ppk_{ID_i^*} + r_i^*W + h_i^{*'}x_i^*MS_{pub}$$

Now τ_1 looks up the list $L = \left(ID_i, \beta_i, X_i, Q_{ID_i}\right)$ and recovers the tuple $(ID_i, \beta_i, X_i, Q_{ID_i})$ corresponding to the identityID_i.

$$h_i^*(V_i^* - r_i^*W) - h_i^{*'}\left(V_i^{*'} - r_i^*W\right) = (h_i^{*-1} - h_i^{*'-1})a\beta_ibP$$

Now, $\tau_1$can find the value of abP as a solution of CDH problem

$$abP = (h_i^*(V_i^* - r_i^*cP) - h_i^{*'}(V_i^{*'} - r_i^*cP))/\beta_i(h_i^{*-1} - h_i^{*'-1})$$

This will give a proof of CDH problem.

Now, we analyze the probability to solve a CDH problem by Type 1 adversary in the polynomial bounded time. For this, we analyze the three events.

E1: τ_1 does not abort all the queries of *Reveal-Partial-Private-Key*.

E2: A_1 can forge a signature when τ_1 does not abort all the queries to *Reveal-Partial-Private-Key*.

E3: A_1 generates a valid and nontrivial forgery when τ_1 does not abort all the queries generated by A_1.

From the simulation [21], we know that $\Pr[E_1] \geq (1-£)^{q_k}$,$\Pr(E_2|E_1) \geq \varepsilon$, $\Pr(E_3|E_1 \wedge E_2\ ) \geq £$

Thus, $P(E_1 \wedge E_2 \wedge E_3) = P[E_1]P[E_2|E_1]P[E_3|E_1 \wedge \mathrm{E}_2]$

$$\geq (1-£)^{q_k}.\varepsilon.£ = £.(1-£)^{q_k}.\varepsilon$$

Now, £ opt as $\frac{1}{q_{k\_}+1}$. Thus,

$$\varepsilon' \geq \frac{1}{q_k+1}.\left(1-\frac{1}{q_k+1}\right)^{q_k}.\varepsilon = \frac{\varepsilon}{e(q_k+1)}$$

Therefore, $\tau_1$could solve the CDH problem with non-negligible probability ε' where ε is the non-negligible probability that gives a contradiction to CDH hard problem. So, our CLS problem is existentially unforgeable against adaptive chosen message attack corresponding to Type 1 adversary in the random oracle model under the CDH assumption.

7.7.2 *Theorem 2*

In a random oracle model, A_2 be a forger having advantage ε to forge a signature in a modeled attack game within running time t and making queries to various oracles:q_{H_i} queries to oracleH_i for $i = 2,3$, q_s queries to *Reveal-Secret-Key*, q_{CU} queries to *Create-User* and q_{sig}queries to *sign*, then the CDH problem can be solved with probability $\varepsilon' > \frac{\varepsilon}{e.q_s+1}$ with time $t' < t + (q_{H_2} + q_{H_3} + q_s + q_p + q_{sig})t_m + (q_{sig}+1)t_{mm}$ wheret_m is the time to compute a scalar multiplication in G_1 and $t_{mm}$is the time to perform a multi exponentiation in G_1.

Suppose$(P, X = aP, Y = bP)$be arandom instance for the CDH problem in G_1 of the prime order q. We will construct an algorithm τ_2 to achieve the solution of CDH problem. Let $a, b\epsilon Z_q^*$be numbers chosen randomly which are unknown to τ_2. Adversary

A_2 interacts with the τ_2 in Game II. τ_2 selects a random number $\gamma \epsilon Z_q^*$, setsγ as the master key of KGC, thereafter selects the system parameters $\{q, G_1, G_2, e, P, MS_{pub}, H_1, H_2, H_3\}$ and returns the system parameters and public key of MS toA_2. $A_2$is type 2 adversary who has potential to access the master key of MS, therefore A_2 and τ_2 can compute the partial private key of the user. We have no need to model the hash function H_1 since master key is available to A_2 adversary. $\tau_2$preserves the list$L = (ID_i, x_{i,} PK_{ID,} X_i)$.

Create User:When adversary A_1 submits a request with identity ID_i, the *Oracle* checks if $\tau_2$contains the list $L = (ID_i, x_{i,} PK_{ID_i}, X_i)$, then τ_2 answers $PK_{ID_i}$to A_2. If the list L does not contain $(ID_i, x_{i,} PK_{ID_i}, X_i)$, according to Coron's proof [63], τ_2 flips a coin $X \in \{0,1\}$ which returns 0 with probability £ and 1 with probability $1 - £$. τ_2 selects a random number $\beta_i \in Z_q^*$ if $X_i = 0$ then the value of PK_{ID_i}can be set as $PK_{ID_i} = \beta_i P \in G_1$ otherwise τ_2 returns $PK_{ID_i} = \beta_i Y \in G_1$. τ_2 sets $x_i = \beta_i$ in both the cases and inserts in the tuple $(ID_i, x_{i,} PK_{ID,} X_i)$ in the list $L = (ID_i, x_{i,} PK_{ID,} X_i)$. τ_2 stores these values to answer the future queries submitted by A_2. τ_2 sends $PK_{ID_i}$to A_2.

H_2query: τ_2 maintains a list $L_{H_2}$with the tuple(m_i, c, W). When A_2 submits the query ofH_2 oracle,τ_2 checks if list L_{H_2}preserves the tuple (m_i, c, W). If yes then there is nothing to be done and τ_1 returns W to A_2. Otherwise τ_1 picks a random $\in Z_q^*$, computes $W = cP$, insertsit in the list L_{H_2} and returns$W$ to A_2.

H_3query:τ_2 maintains a list $L_{H_3}$with the tuple$(m_i, ID_i, PK_{ID_i}, U_i, h_i)$. When A_1 submits the query ofH_3 oracle,τ_2 checks if list L_{H_3} preserves the tuple $(m_i, ID_i, PK_{ID_i}, U_i, h_i)$.If yes then there is nothing to be done and τ_1 returns h_i to A_1. Otherwise τ_1 picks a random$h_i \in Z_q^*$, insertsit in the list L_{H_3} and returns$h_i$to A_1.

Reveal-secret-key-queries: When adversary A_2 submits a request on identity ID_i , τ_2 maintains a list $(ID_i, x_{i,} PK_{ID_{i,}} W_i)$. $\tau_2$checks whether$x_i = \perp$. If $x_i \neq \perp$, then $\tau_2$returns$x_i$to A_2. If $x_i = \perp$then τ_2 randomlyselects $\beta_i \in Z_q^*$, put $PK_{ID_i} = \beta_i P$and returns $x_i$toA_1.

If τ_2 does not holds the list $(ID_i, x_{i,} PK_{ID_{i,}} W_i)$, τ_2 sets $x_i = \perp$ .If $PK_{ID_i} \neq \perp$, τ_2 randomlyselects $u_i \in Z_q^*$, sets $PK_{ID_i} = u_i P$, then$\tau_2$returns x_i to A_2 and insertsPK_{ID_i} in the list $(ID_i, x_{i,} PK_{ID_{i,}} X_i)$.

Sign-queries: When A_2 submits a request on (ID_i^*, m_i), τ_2 lookup the list L, L_{H_2}, L_{H_3} to recover the tuples $\left(ID_i^*, x_i^*, PK_{ID_i}^*, X_i^*\right)$, (m_i^*, c^*, W^*) , $(m_i^*, ID_i^*, PK_{ID_i}^*, U_i^*, h_i^*)$ respectively.

When $X_i = 1$, Even A_2 does not submit a sign query on $\left(m_i^*, ID_i^*, PK_{ID_i^*}\right)$, the forged signature should satisfy

$$e(V^*, P) = e(Q_{ID_i^*} + h_i^* PK_{ID_{i^*}}, P_{pub})e(R_i^*, W^*), \text{ where} Q_{ID_{i^*}} = H_1(ID^*).$$

We set, $Q_{ID_{i^*}} = H_1(ID^*)$,$W^* = c^*P$, $PK_{ID_i^*} = \beta_i Y$, $X = MS_{pub}$

$$e\left(h_i^* PK_{ID_{i^*}}, P_{pub}\right) = e(V^*, P)\left(e\left(Q_{ID_i^*}, MS_{pub}\right)e(R_i^*, W^*)\right)^{-1}$$

Hence τ_2 finds the solution of CDH problems as,

$$abP = h_i^*(V_i - c^{-1}R_i - aQ_{ID_i})$$

Now, we analyze the probability to solve a CDH problem by Type 1 adversary in the polynomial bounded time. For this we analyze the three events.

E1: τ_2 does not abort all the queries of *Reveal-Secret-Key*.

E2: A_2 can forge a signature when τ_2 does not abort all the queries to *Reveal-Partial-Private-Key*.

E3: A_2 generates a valid and nontrivial forgery when τ_2 does not abort all the queries generated by A_2.

From the simulation [21], we know that $\Pr[E_1] \geq (1 - £)^{q_s}$,$\Pr(E_2|E_1) \geq \varepsilon$, $\Pr(E_3|E_1 \wedge E_2\ ) \geq £$

Thus, $P(E_1 \wedge E_2 \wedge E_3) = P[E_1]P[E_2|E_1]P[E_3|E_1 \wedge \mathrm{E}_2]$

$$\geq (1 - £)^{q_s}.\varepsilon.£ = £.(1 - £)^{q_s}.\varepsilon$$

Now, £ opt as $\frac{1}{q_s+1}$. Thus,

$$\varepsilon' \geq \frac{1}{(q_s + 1)}.\left(1 - \frac{1}{(q_s + 1)}\right)^{q_s}.\varepsilon = \frac{\varepsilon}{e(q_s + 1)}$$

Therefore, τ_2 can solve the CDH problem with non-negligible probability ε' where ε is the non-negligible probability which contradicts the CDH hard problem. So, our CLS scheme is existentially unforgeable against adaptive chosen message attack corresponding to Type 1 adversary in the random oracle model under the CDH assumption.

7.7.3 *Theorem 3*

If the base certificateless signature scheme is existential unforgeable against identity and adaptive chosen message attacks then certificateless aggregate signature scheme is also secure against existential forgery in the chosen aggregate model.

Proof: Let$(P, X = aP, Y = bP)$ be a random instance of the CDH problem in G_1 with prime order q , we will construct an algorithm τ to solve the CDH problem. Let $a, b\epsilon Z_q^*$ are numbers chosen randomly which are unknown to τ. τrandomlychooses a challenge identity ID_i and sends $\{q, G_1, G_2, e, P, MS_{pub}, H_1, H_2, H_3\}$ to adversary A . τ sets $X = MS_{pub}$ andgets ready to execute the oracle queries. τ preserve a list $L = (ID_i, x_{i,} PK_{ID,} X_i)$, while A can submit query throughout the game.

H_1*queries*: When A submits a query on identityID_i to H_1 oracle, τ preserve a list $L_{H_1} = (ID_i, \beta_i, X_i, Q_{ID\,i})$. If list L_{H_1} holds the tuple $(ID_i, \alpha_i, Q_{ID_i})$ then nothing has to be done and τ returns Q_{ID_i} to A.Next τ flips a coin $X \in \{0,1\}$ that return 0 with probability £ and 1 with probability $1 - £$. τ selects a random number $\beta_i \in Z_q^*$ if $X_i = 0$ and the value of $H_1(ID_i)$can be set as $Q_{ID\,i} = \beta_i P \in G_1$ otherwise τ returns $Q_{ID_i} = \beta_i MS_{pub} \in G_1$. In both the cases, τ inserts a tuple in the list $L_{H_1} = (ID_i, \beta_i, X_i, Q_{ID\,i})$ to answer the further queries of A.

A has an output of n user's $(u_1, u_2 \ldots\ldots u_n)$having identities $L_{ID} = (ID_1^*, ID_2^* \ldots\ldots\ldots ID_n^*)$, public keys $L_{PK} = (P_1^*, P_2^* \ldots\ldots P_n^*)$and an aggregate signature $\sigma^* = (U_1^*, U_2^* \ldots\ldots U_n^*, V^*)$ on the message set $\{m_1^*, m_2^* \ldots\ldots\ldots, m_n^*\}$. τ gets the corresponding tuple $(ID_i, \beta_i, X_i, Q_{ID\,i})$ where $i \in (1,2 \ldots\ldots n)$ by recalling L_{H_1} only when $X_k = 1$ and $X_j = 0$ for $j \in (1,2 \ldots\ldots n)$ and $j \neq k$. Here, $(ID_k, PK_{ID\,k^*}, m_k^*)$ has never been submitted to sign queries otherwise τ fails and stops the simulation where $Q_{ID_k} = \beta_k MS_{pub}, Q_j = \beta_j MS_{pub}$ for $j \in [1, n]$and $j \neq k.$ New generated signature is $\sigma^* = (U_1^*, U_2^* \ldots\ldots\ldots U_n^*, V^*)$ which is verifiable by the following aggregate verification equation.

$$e(V^*,P) = (e(\sum_{i=1}^{n}(Q_{ID_{i^*}} + h_i^* PK_{ID^*}, MS_{pub})e(\sum_{i=1}^{n} R_i^*, W)$$

τ looks up the list L and list L_{H_3} to recover the tuples $\left(ID_i, x_{i,}PK_{ID_{i,}}ppk_{ID_i}\right)$ and $(m_i, ID_i, PK_{ID_i}, R_i, h_i)$ respectively. Then τ sets $V_i^* = \beta_i MS_{pub}$ which can be verified as$e(V_i^*, P) = e(Q_{ID_i}, MS_{pub})$ for $i \in (1,2 \ldots\ldots n)$.

Finally, τ creates $V'^* = V^* - \sum_{i=1, j\neq k}^{n} V_i^*$.

$$V'^* = ppk_{ID_k^*} + \sum_{i=1}^{n} r_i W + \sum_{i=1}^{n} h_i x_i MS_{pub}$$

For a random number $r_i^* \in Z_q^*$ where $i \in (1,2 \ldots\ldots n)$ and computes $R_i^* = r_i^* P$. τ choose a random number $h_k^* \in Z_q^*$, and computes

$$R'^* = \sum_{i=1}^{n} R_i^*$$

Thereafter, τ calculates $PK'_{ID_k^*} = (h_k^*)^{-1} \sum_{i=1}^{n} h_i^*. PK_{ID_i^*}$ and updates $PK_{ID_k^*}$ to $PK'_{ID_k^*}$ by making *Replace-Public-Key-queries*. Then τ defines the hash value $H_3(m_k^*, ID_k^*, PK_{ID_k^*}, R_k^*)$ as h_k^* *i.e* $H_3\left(m_k^*, ID_k^*, PK_{ID_k^*}, R_k^*\right) = h_k^*$. If the tuple $(m_k^*, ID_k^*, PK_{ID_k^*}, R_k^*)$ is already present in the listL_{H_3}, then tries another h_k^* to avoid collision. Then, $(R'^*, V'^*)$is a valid signature on the message$m_k^*$for the identity$ID_k$with corresponding public key$PK'_{ID_k^*}$with its verification given by

$$\begin{aligned}
&e\left(Q_{ID_k^*} + h_k PK'_{ID_k^*}, P_{pub}\right) e(R_k'^*, W) \\
&\qquad = e\left(Q_{ID_k^*} + h_k (h_k^*)^{-1} \sum_{i=1}^{n} h_i^*. PK_{ID_i^*}, MS_{pub}\right) e\left(\sum_{i=1}^{n} R_i^*, W\right) \\
&\qquad = e\left(Q_{ID_k^*} + \sum_{i=1}^{n} h_i^*. x_{ID_i^*} P, MS_{pub}\right) e\left(\sum_{i=1}^{n} R_i^*, W\right) \\
&\qquad = e\left(Q_{ID_k^*} + \sum_{i=1}^{n} h_i^*. x_{ID_i^*} P, sP\right) e\left(\sum_{i=1}^{n} r_i^* P, W\right) \\
&\qquad = e\left(ppk_{ID_i^*} + \sum_{i=1}^{n} h_i^*. x_{ID_i^*} MS_{pub}, P\right) e\left(\sum_{i=1}^{n} r_i^* W, P\right) \\
&\qquad = e\left(ppk_{ID_i^*} + \sum_{i=1}^{n} r_i^* W + \sum_{i=1}^{n} h_i^*. x_{ID_i^*} MS_{pub}, P\right)
\end{aligned}$$

Finally, τ generates a forged signature of the certificateless aggregate signature scheme.

7.8 Batch Verification

In this section, we describe how batch verification works to verify a set of certificateless signature received by health professionals. Comenisch et al. [60] used batch verification for increasing the speed of verification.

Definition: Suppose k be a security parameter, (*setup, Partial-private-key-gen, Private-key-gen, Sign, Verify*) be a CLS scheme and $n \in poly(k)$. We call that a probabilistic batch is a batch verification algorithm when the following conditions are satisfied.

1) If individual verification $(ID_1, PK_1, m_1, \sigma_1) = 1, \forall\, i \in (1,2 \ldots .n)$ then batch $(ID_1, PK_1, m_1, \sigma_1) \ldots \ldots \ldots \ldots (ID_n, PK_n, m_n, \sigma_n) = 1$
2) If individual verification $(ID_1, PK_1, m_1, \sigma_1) = 1$, for any $i \in (1,2 \ldots .n)$ then batch $(ID_1, PK_1, m_1, \sigma_1) \ldots . \ldots \ldots (ID_n, PK_n, m_n, \sigma_n) = 0$

We demonstrate how the sensor's certificateless signature can be verified. Without loss of generality, we assume that an aggregator receives a set of message $(ID_1, PK_1, m_1, \sigma_1). \ldots \ldots (ID_n, PK_n, m_n, \sigma_n)$ from the set of sensors with identities $(ID_1, ID_2 \ldots . ID_n)$. Then, aggregator compute $Q_{ID_i} = H_1(ID_i)$, $h_i = H_3(m_k^*, ID_k^*, PK_{ID_k^*}, R_k^*)$ and generates a vector $\delta = (\delta_1, \delta_2 \ldots \ldots . \delta_n)$ where each δ_i is the random number of k bits from Z_q^*. Then the aggregator verifies the signature in a batch by investigating whether the following equation is satisfied.

$$e\left(\sum_{i=1}^{n} \delta_i V_i, P\right) = e\left(\sum_{i=1}^{n} \left(\delta_i (Q_{ID_i} + h_i PK_{ID_i}\right), P_{pub}\right) e\left(\sum_{i=1}^{n} \delta_i R_i, P\right)$$

Correctness:

$$e\left(\sum_{i=1}^{n} \delta_i(ppk_{ID_i} + r_i W + h_i x_{ID_i} MS_{pub}), P\right)$$

$$= e\left(\sum_{i=1}^{n} \delta_i(ppk_{ID_i}\right.$$

$$\left.+ h_i x_{ID_i} MS_{pub}, P\right) e\left(\sum_{i=1}^{n} \delta_i r_i W, P)\right)$$

$$= e\left(\sum_{i=1}^{n} \delta_i(sQ_{ID_i} + h_i x_{ID_i} sP, P\right) e\left(\sum_{i=1}^{n} (\delta_i r_i W, P)\right)$$

$$= e\left(\sum_{i=1}^{n} \delta_i(Q_{ID_i} + h_i PK_{ID_i}, MS_{pub}\right) e\left(\sum_{i=1}^{n} (\delta_i r_i P, W)\right)$$

$$= e\left(\sum_{i=1}^{n} \left(\delta_i(Q_{ID_i} + h_i PK_{ID_i}\right), MS_{pub}\right) e\left(\sum_{i=1}^{n} \delta_i R_i, P\right)$$

With batch verification, we can see an aggregator takes 3 pairing operations to execute the signatures of all sensors. Therefore, it consumes $3n$ pairing operations to verify n individual signatures. Let k be a security parameter and there be an error probability which is set to at most 2^{-k}. If one the signature out of *n* signatures is “wrong” then aggregator should detect it, except with probability 2^{-k}.

7.9 Performance Analysis

Table 7.2: Performance Comparison of CL-AS schemes with running time (*ms*)

CL-AS	Type	Sign	Individual verify	Aggregate verify
[15]	*Sync*	3S =1.17*ms*	4P=12.84*ms*	(n+3)P=((n+3)3.21*ms*
[57]	*Sync*	5P =1.95*ms*	5P+2S=16.83*ms*	5P+2nS=(16.05+0.78 n)*ms*
[48]	*Ad-hoc*	2S =0.76*ms*	3P=9.63*ms*	(2n+1)P=(2n+1)3.21 *ms*
[48]	*Sync*	3S= 1.17*ms*	3P=9.63*ms*	(n+2)P+nS=((n+2)3.21+n0.39)*ms*
[18]	*Ad-hoc*	3S =1.17*ms*	3P+2S=10.39*ms*	3P+2nS=(9.63+0.78n)*ms*
Our scheme	*Ad-hoc*	3S =1.17*ms*	3P+3S=10.78*ms*	3P+nS=(9.63+1.17n) *ms*

Sync means normal mode of transfer,Ad hoc means temporary mode of transfer, S-Scalar multiplication, P-Pairing

We take the setup of experiment that analyzes the processing time for the Tate pairing on a 159-bit subgroup of an MNT curve with an implanting degree 6 at an 80-bit security level, running on an Intel i7 3.07 GHz machine. According to experiment the time consumed by various operations is as follows: Pairing cost is 3.21 *ms*, Signing cost is 0.39 *ms* and Hashing cost is 0.09 *ms*. We claim that the proposed CL-AS scheme is much efficient than the other existing schemes on the basis of Table 7.2. Proposed CL-AS scheme takes time 1.17 *ms* in signing phase, 10.78 *ms* in verification phase and (9.63+1.17n) *ms* in aggregate verification.

Energy consumption can be compute as $E_c = T_c P$, where E_c is the energy consumption, T_c is the total computational time for a signature delivered, and P is the CPU maximum power (10.88W). Table 7.3 describes the comparison of energy consumption with other existing CL-AS schemes. Total individual energy consumption of CL-AS proposed in [48] is efficient more than our scheme but total aggregate verifying cost of our scheme is efficient than the corresponding scheme [48]. On the basis of the table 7.3, we can claim that our CL-AS scheme is much energy efficient than the other existing schemes.

Table 7.3: Total Energy Consumption (mJ)

CL-AS	Individual total computational cost (ms)	Total individual energy consumption (mJ)	Aggregate verify	Total aggregate verifying energy consumption (mJ)
[15]	14.01 ms	152.4288 mJ	(n+3)P=((n+3)3.21 *ms*	(n+3)34.9248 mJ
[57]	18.75 ms	204.3264 mJ	5P+2nS=(16.05+0.78n)*ms*	174.624+n8.4864 mJ
[48]	10.39 ms	113.0432 mJ	(2n+1)P=(2n+1)3.21*ms*	(2n+1)34.9248 mJ
[48]	10.8 ms	104.004 mJ	(n+2)P+nS=((n+2) 3.21+n0.39)*ms*	(n+2)34.9248+n4.2432 mJ
[18]	11.56 ms	125.7728 mJ	3P+2nS=(9.63+0.78n)*ms*	104.7744+8.4864 n mJ
Our scheme	11.19 ms	121.7472 mJ	3P+nS=(9.63+1.17 n)*ms*	104.7744+12.7296n mJ

In this chapter, we have proposed a novel certificateless signature (CLS) scheme using the concept of aggregate signature for secure communication in healthcare wireless sensor networks. Our proposed (CL-AS) scheme has advantages of aggregate signature and certificateless signature. The proposed (CL-AS) helps to protect the online data from the unauthorized entities in healthcare wireless sensor network. Security of our construction is proved by Diffie Hallman assumption under the random oracle model that claims our system is not forgeable against the present adversaries in the system. Through experimental results, we have proved that our CL-AS scheme is more computational and energy efficient as compared to the existing schemes.

CHPATER 8

Secure CLS and CL-AS Schemes Designed for VANETs

VANET, a part of intelligent transport system, consists of three components, Vehicles, Road side unit (RSU), and Infrastructure (I). There are three types communications in VANET, vehicle-to-vehicle (V-V), vehicle-to-infrastructure (V-I) and infrastructure-to-infrastructure (I-I). Vehicles communicate with each other in a high speed with dedicated short range radio signals (DSRS) to share traffic related information. Information is shared among vehicles in VANET which leads to some basic security problems in the network such as, authentication, anonymity, non-repudiation, privacy, integrity and availability. We used the digital signature for preserving authentication, anonymity, non-repudiation privacy, integrity, availability in VANET. Bandwidth limitation is also an issue when communication takes place in VANET. To improving upon bandwidth problem, we use aggregate digital signature scheme.

8.1. Some special symbol used in the chepter

First of all, we summarize some special symbols used in table 8.1.

Table-8.1**:** Some special symbols used in the chapter 8

Symbols	**Description**
RTA	Regional Transport authority
KGC	Key Generation Center
P_{rsu_i}	The public key ofith RSU_i
y_i	The private key of RSU_i
pp_i	The partial private key of ith user's identity ID_i
x_i	Vehicle's private key corresponding identity ID_i
$Params$	The system parameters generated by KGC
PS_i	Vehicle's pseudonym generated by RSU_j in an autonomous network
P_i	The public key of vehicle having ith user's identity ID_i

8.2 Security Models of a CLS Scheme

In this subsection, we define the adversaries' model for a CLS scheme and CL-AS scheme. We consider two level of securities in the proposed models: Type 1 security and Type 2 security with two type of adversaries, A_1 and A_2. Basically, A_1, A_2 are involved in CLS scheme with different capabilities. Here A_1 behaves like an outsider and A_2 behaves like a malicious KGC, who can generate the partial-private key of user.

- **Adversary** A_1**:** A_1 has power to replace the public key of a user, but cannot access the master key of KGC.
- **Adversary** A_2**:** A_2 is allowed to access the master key of KGC, but cannot replace the public key of a user.

Definition 4: A CLS scheme/CL-AS scheme is said to be existentially unforgeable against adaptive chosen message and identity attacks, if the adversaries A_1 and A_2 have negligible probabilities to forge the signature.

A_1 and A_2 can access the following six oracles:

I. *CreateUser*: On submitting a target identity $ID_i \in \{0,1\}^*$, if this query is already executed for this identity then nothing to do, otherwise *RevealPartialkey* and *RevealSecretkey* queries for this identity are executed to compute the partial private key pp_i and private/public key pair (x_i, P_i) . Afterwards that these keys are stored in list $L = (ID_i, x_i, P_i, pp_i, PS_j)$ and then P_i is returned in both cases, where $PS_j$is pseudonym generated by RSU_j in an autonomus network..

II. *RevealPartialkey*: On submitting a request of target identity $ID_i \in \{0,1\}^*$, oracle looks up in to the list L . If a proper entry found in L, it returns the corresponding partial private key pp_i otherwise it returns $\perp$.

III. *Revealpseudonym*: On submitting a request of target identity $ID_i \in \{0,1\}^*$, oracle looks up in to the list L . If a proper entry found in L, it returns the corresponding pseudonym PS_i otherwise it returns $\perp$.

IV. *RevealSecretKey*: On submitting a request of target identity $ID_i \in \{0,1\}^*$, oracle looks up in to the list L . If a proper entry found in the list it returns the corresponding secret key x_i otherwise it returns $\perp$.

V. *ReplacePublicKey*: On submitting a request of target identity $ID_i \in \{0,1\}^*$ and private/public key pair (x_i, P_i^*), oracle looks up in to the list L, if the corresponding identity is not found then there is nothing to perform, otherwise, this oracle updates $L = (ID_i, x_i, P_i, pp_i, PS_j)$ to $L = (ID_i, x_i, P_i^*, pp_i, PS_j)$.

VI. *Sign*: On submitting a request on the message with target signer's identity $ID_i \in \{0,1\}^*$, oracle performs one of the three activities.

3) Returns a valid signature σ_i without replacing private/public key pair, if ID_i has been created without replacing private/public key pair.
4) Returns $\perp$, if ID_i is not created.
5) Returns the signature (x_i^*, P_i^*, m_i) after replacing the private/public key pair, if ID_i has been created after replacing private/public key pair.

We construct two Games: Game I and Game II. Here, Game I and Game II are designed for A_1 and A_2 in CLS scheme, respectively.

Game I: τ_1 is the challenger/simulator and deals with A_1. This Game executes the following steps.

- Step1: τ_1 start the *Master-Key-Gen* algorithm, which takes a security parameter 1^k as input and generates a master key and list of system parameters. Then τ sends the system parameters to A_1 while keeping the master key secret.
- Step 2: In this step, A_1 can execute *revealpartialkey, revealsecretkey, revealpublickey, revealpseudonym*, *replacepublickey* and *sign* queries at any stage during the simulation in polynomial bound.
- Step 3: A_1 outputs a signature σ_i^* on a message m_i^* corresponding to a targeted identity ID_i^* with public key P_i^*.

A_1 wins the game if any one of the following conditions is satisfied.

ix) σ_i^* is a valid signature on m_i^* under ID_i^* and P_i^*

x) If ID_i^* has never been executed by the oracle *revealpartialprivatekey* for getting the partial private key.

xi) *Sign* oracle has never been executed for m_i^* under ID_i^*.

Definition 5: A CLS scheme is called Type 1 secure if there does not exist any adversary A_1 who wins the Game I in probabilistic polynomial time bound with non-negligible advantage.

Game II: τ_2 is the challenger/simulator and deals with A_2. This Game executes the following steps.

- Step1: τ_2 starts the *Master-Key-Gen* algorithm, which takes a security parameter 1^k as input, and generates a master key and system parameters. Then τ_2 sends the system parameter to A_2 while keeps the master key secret.
- Step 2: In this step, A_2 can execute *revealpartialkey, revealsecretkey, revealpublickey, revealpseudonym, replacepublickey* and *sign* queries at any stage during the simulation in polynomial time bound.
- Step 3: A_2 output a signature σ_i^* on a message m_i^* corresponding to a targeted identity ID_i^* with public key P_i^*.

A_2 wins the game if any one of the following conditions is satisfied.

i) σ_i^* is a valid signature on m_i^* under ID_i^* and P_i^*

ii) If ID_i^* has not been queried to the oracle *revealpartialprivatekey* for getting the partial private key.

iii) *Sign* oracle has never been executed for m_i^* under ID_i^*.

Definition 6: A CLS scheme is called Type 2 secure if there does not exist any adversary A_2 who wins the Game I in probabilistic polynomial time bound with non-negligible advantage.

We construct two more games: Game III and Game IV. Here, Game III and Game IV are executed by A_1 and A_2 in CL-AS scheme, respectively.

Game III: τ_1 is the challenger/simulator and deals with the adversary A_1. This Game executes the following steps.

Step1: τ_1 starts the *Master-Key-Gen* algorithm, which takes a security parameter 1^k as input and generates master key and system parameters. Then $\Im$ sends system parameters to A_1 while keeps the master key secret.

Step 2: In this step, A1 can execute *revealpartialkey, revealsecretkey, revealpublickey, revealpseudonym, replacepublickey* and *sign* queries at any stage during simulation in polynomial bound.

Step 3: A1 output an aggregate signature σ_i^* on the set of n user whose identity set is $\{ID_1^*, ID_2^*.........ID_n^*\}$, corresponding public key set is $\{P_1^*, P_2^*.........P_n^*\}$, and pseudo identities set is $\{PS_1^*, PS_2^*.........PS_n^*\}$ on a message set $\{m_1^*, m_2^*.........m_n^*\}$.

A1 wins the game III if any one of the following conditions is satisfied.

i) σ_i^* is a valid signature on $\{m_1^*, m_2^*.........m_n^*\}$ under $\{ID_1^*, ID_2^*.........ID_n^*\}$ and $\{P_1^*, P_2^*.........P_n^*\}$.

ii) At least one of the identities is not submitted during query to the oracle *revealpartialprivatekey* for getting partial private key.

iii) The *Sign* query has never been submitted on $\{m_1^*, m_2^*.........m_n^*\}$ under $\{ID_1^*, ID_2^*.........ID_n^*\}$.

Definition 7: A CL-AS scheme is called Type 2 secure if there does not exist any any adversary A1 who wins the Game III in probabilistic polynomial time bound with non-negligible advantage.

Game IV: τ_2 is the challenger/simulator and deals with the adversary A2. This game executes the following steps.

Step1: τ_2 starts the *Master-Key-Gen* algorithm, which takes a security parameter 1^k as input and generates a master key and system parameters. Then τ_2 sends system parameters to A2 while keeps the master key secret.

Step 2: In this step, A2 can execute *revealpartialkey*, *revealsecretkey*, *revealpublickey*, *revealpseudonym*, *replacepublickey* and *Sign* queries at any stage during simulation within polynomial time bound.

Step 3: A2 output an aggregate signature σ_i^* on the set of n user whose identity set is $\{ID_1^*, ID_2^*.........ID_n^*\}$, public key set is $\{P_1^*, P_2^*.........P_n^*\}$, and pseudo identity set is $\{PS_1^*, PS_2^*.........PS_n^*\}$ on a message set $\{m_1^*, m_2^*.........m_n^*\}$.

A2 wins the Game IV if any one of the following conditions is satisfied.

i) σ_i^* is a valid signature on $\{m_1^*, m_2^*.........m_n^*\}$ under $\{ID_1^*, ID_2^*.........ID_n^*\}$ and $\{P_1^*, P_2^*.........P_n^*\}$.

ii) At least one of the identities is not submitted during query to the oracle *revealpartialprivatekey* for getting partial private key.

iii) The *Sign* query has never been submitted on $\{m_1^*, m_2^*.........m_n^*\}$ under $\{ID_1^*, ID_2^*.........ID_n^*\}$.

Definition 8: A CL-AS scheme is called Type 2 secure if there does not exist any adversary A_2 who wins the Game IV in probabilistic polynomial time bound with non-negligible advantage.

8.3 Our Certificateless Signature Scheme

8.3.1 Framework of our system

The proposed model is divided into two levels, namely upper level and lower level. Upper level deals with two trust authorities Regional Transportation Authority (RTA) and Key Generation Centre (KGC) with the assumption that it is impossible for any adversary to compromise them. The lower level deals with Road Side Unit (RSU) and vehicles. Upper level authority controls the lower level authority. Since VANET supports three types of communication, such as Vehicle-to-Vehicle (V-V), Vehicle-to-Infrastructure (V-I) and Infrastructure-to-Infrastructure (I-I). We assume that V-V and V-I communications using dedicate short range radio signals [30] in our framework.

Our CLS scheme consists of four entities: RTA, KGC, RSU and Onboard Unit (OBU), which is installed in every vehicle. A trusted authority RTA is responsible for the registration of the vehicle after verifying their real identity and creates another identity for further communication, when a vehicle enters in the range. Vehicle sends their identity generated by RTA to KGC then KGC is responsible to generate partial private key of the vehicle and send it to the user via a secure channel. After getting partial private key, the corresponding vehicle prepares a private/public key pair and generates a signature using the private key.

We take some assumptions as follows: 1) RTA and KGC are always trusted, nobody can conspire them [46, 61]. 2) Each vehicle is equipped with a tamper proof device, which is a password protected hardware that prevents intruder to extract data stored in device [49, 50]. Each vehicle has facility of global positioning system (GPS).

Our CLS scheme consist six algorithms: *Setup, Registration, Partialprivatekeygeneration, userkeygeneration, Pseudonymgeneration, Sign and Verify* , which are described as follows*:*

- *Setup*: This algorithm is executed by the KGC, which works under RTA. Taking 1^k as a security parameter, KGC chooses two cyclic groups: one additive cyclic group G_1, other is multiplicative cyclic group G_2 of prime order q with a generator point P and defines an admissible bilinear map $e: G_1 \times G_1 \rightarrow G_2$. KGC generates a master key $s \in Z_q^*$ and computes public key as $P_{pub} = sP$. KGC chooses four distinct one-way cryptographic hash functions $H_2:\{0,1\}^* \rightarrow Z_q^*$, $H_3:\{0,1\}^* \rightarrow Z_q^*$, $H_4:\{0,1\}^* \rightarrow Z_q^*$, $H_5:\{0,1\}^* \rightarrow G_1$, defined on message space $\psi \in \{0,1\}^*$. Each RSU in the region under RTA sets a secret key $y_i \in Z_q^*$ and its public key as $P_{rsu_i} = y_i P$ and sends its public key to RTA. Then RTA sends the public keys $P_{rsu_1}, P_{rsu_2}, P_{rsu_3} ....P_{rsu_n}$ of all RSUs under its region to KGC. KGC then publishes a system parameter list $\{G_1, G_2, e, P, P_{pub}, H_2, H_3, H_4, H_{5,} P_{rsu_1}, P_{rsu_2} ...., P_{rsu_i}\}$.
- *Registration:* RTA runs this algorithm for registering the vehicle with identity ID_i when the vehicle enters in their region. RTA maps the vehicle identity ID_i to the Q_{ID_i} so that Q_{ID_i} is to be used in all further communications. Vehicles need to register again, after changing the region of RTA. RTA selects a one way cryptographic hash function $H_1:\{0,1\}^* \rightarrow G_1$ for registration of vehicles. Following steps show the process of registration:

  i) Vehicle sends its identity ID_i to RTA

  ii) RTA registers the vehicle ID_i as $Q_{ID_i} = H_1(ID_i) \in G_1$ after verifying the vehicle's identity.

  iii)RTA sends Q_{ID_i} to the vehicle for further communication.

- *Partialprivatekeygeneration*: KGC takes input as parameter list, master key and vehicle's identity Q_{ID_i}, then KGC selects user's partial private key $pp_i = sQ_{ID_i}$, where $Q_{ID_i} = H_1(ID_i) \in G_1$. This partial private key is also a signature, which can be verified as $e(pp_i, P) = e(Q_{ID_i}, P_{pub})$
- *Userkeygeneration*: This algorithm is run by vehicles to generate their secret key as well as public key. The vehicle with identity Q_{ID_i} selects a random number as its private key $x_i \in Z_q^*$ then sets the corresponding public key as $P_i = x_i P$.
- *Pseudonymgeneration*: In the proposed scheme, whole network is divided into autonomous sub-networks on behalf of number of nodes and population in the networks. In dense population, it comprises with three RSUs and five RSUs in scarcely. RSU generates pseudonym for each vehicle in an autonomous network for maintaining liability and privacy. This algorithm is run by each RSU. RSU$_i$ takes the identity Q_{ID_i} of a vehicle. RSU$_i$ selects a random number $a_j \in Z_q^*$ and sets $PS1_j = a_j.Q_{ID_i}$, thereafter calculates a hash value $T_j = H_3(PS1_j)$. In the second part, it calculates $PS2_j = a_j.T_j$. Finally, the pseudonym is calculated as $PS_j = PS1_j + PS2_j$ for the vehicle *ID$_i$*.
- *Sign*: Any vehicle with identity Q_{ID_i} can sign having its partial private key pp_i, secret key x_i and public key p_i on a message m_k as follows:

  i) Signer vehicle selects a random $r_i \in Z_q^*$ and computes $U_i = r_i P$

  ii) Computes $h_{ijk} = H_2(m_k, PS1_j, P_i, U_i)$, $t_{ijk} = H_4(m_k, PS1_j, P_i, U_i)$ and $W = H_5(\Delta)$

  iii)Computes $V_{ijk} = pp_i.PS2_j + h_{ijk} r_i W + t_{ijk} x_i P_{rsu_i}$

  iv)Outputs (U_i, V_{ijk}) as a signature on m_k
- *Verify*: Verifier vehicle can verify the signature $\sigma_i = (U_i, V_{ijk})$ signed by the vehicle with identity Q_{ID_i} on the message m_k, with pseudonym PS_j and with the public key P_i as follows:

  i) Verifier vehicle computes $h_{ijk} = H_2(m_k, PS1_j, P_i, U_i)$, $t_{ijk} = H_4(m_k, PS1_j, P_i, U_i)$, $W = H_5(\Delta)$, $T_j = H_3(PS1_j)$

ii) $e(V_{ijk}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}.U_i, W) e(t_{ijk}.P_i, P_{rsu_i})$. If this equation is satisfied then accepts the signature otherwise rejects.

Correctness: $e(V_{ijk}, P) = e(pp_i.PS2_j + h_{ijk}.r_i.W + t_{ijk}.x_i.P_{rsu_i}, P)$

$$= e(s.Q_{ID_i}.a_j T_j, P) e(h_{ijk}.r_i.P, W) e(t_{ijk}.x_i.y_i.P, P)$$

$$= e(s.Q_{ID_i}.a_j T_j, P) e(h_{ijk}.r_i.P, W) e(t_{ijk}.x_i.P, y_i P)$$

$$= e(PS_1.T_j, sP) e(h_{ijk}.U_i, W) e(t_{ijk}.P_i, P_{rsu_i}) = e(PS_1.T_j, P_{pub}) e(h_{ijk}.U_i, W) e(t_{ijk}.P_i, P_{rsu_i})$$

8.3.2 **Our Certtificateless Aggregate Signature Scheme**

Our CL-AS scheme consists of the following algorithms: *Setup, Registration, Partialprivatekeygeneration, userkeygeneration, Pseudonymgeneration, Sign, Verify, Aggregate* and *AggregateVerify*. Note that the algorithms *Setup, Registration, Partialprivatekeygeneration, userkeygeneration, Pseudonymgeneration, Sign* and *Verify* are same as described in CLA scheme, whereas *Aggregate* and *AggregateVerify* are described as follows:

- *Aggregate:* Any vehicle can generate an aggregate signature after collecting all individual signatures. Respective vehicle takes as input *n* signatures from *n* users $(u_1, u_2, u_3 ....u_n)$, with their pseudonym identities $(pp_1, pp_2, ...pp_n)$, corresponding public keys $(P_1, P_2, ...P_n)$ and signatures $\{\sigma_1 = (U_1, V_1), \sigma_2 = (U_2, V_2)........\sigma_n = (U_n, V_n)\}$ on messages $\{m_1, m_2 ...., m_n\}$. Then aggregate signature is computed as $V = \sum_{i=1}^{n} V_i$ and an aggregate signature pair as $\sigma = (U_1, U_2 ....U_n, V)$.
- *Aggregateverify*: Verifier can verify the aggregate signature $\sigma = (U_1, U_2 ....U_n, V)$ by the following procedure.
- Computes $h_i = H_2(m_i, PS1_i, P_i, U_i) \in Z_q^*$, $t_i = H_4(m_i, PS1_i, P_i, U_i) \in Z_q^*$, $W = H_5(\Delta)$, $T_i = H_3(PS1_i)$
- Checks whether

$$e(V, P) = e(\sum_{i=1}^{n} PS1_i T_i, P_{pub}) e(\sum_{i=1}^{n} h_i U_i, W) e(\sum_{i=1}^{n} t_i.P_i, P_{rsu})$$

- Takes an assumption that all the messages are signed under single RSU. If the above equation is satisfies, then the signature is correct, otherwise the signature is incorrect.

8.4. Security Analysis of the Proposed CLS Scheme

In this section, we describe the security analysis of our CLS scheme with the hardness assumption of the CDH problem.

8.4.1 Theorem 1: In the random oracle model, A$_1$ having advantage ε in forging a CLS scheme, wins the Game 1 if A$_1$ successfully constructs an algorithms to solve the CDH problem with nonnegligible probability.

Proof: For a random instance $(P, X = aP, Y = bP)$ of the CDH problem, we will establish an algorithm τ_1 to solve the CDH problem, where $a,b \in Z_q^*$ are chosen randomly and these are unknown to τ_1. In the Game 1, A$_1$ interacts with τ_1. τ_1 selects a random identity ID_t as a challenged identity and then sends $\{G_1, G_2, e, P, P_{pub}, H_2, H_3, H_4, H_{5,}P_{rsu}\}$ to A$_1$. τ_1 also selects $c \in Z_q^*$ and sets $P_{pub} = X$ and $P_{rsu} = cP$. Now τ_1 is ready to execute the oracle query. $\Im_1$ maintains a list $(ID_i, x_i, P_i, pp_i, PS_j)$, while A$_1$ can query throughout the game.

- H_1 *query***:** After submitting a challenged identity ID_t to H_1 oracle, τ_1 maintains a list L_{H_1} in the form of $(ID_i, \alpha_i, Q_{ID_i})$. If L_{H_1} contains the tuple (ID_i, α_i, Q_i), then there is nothing to do and τ_1 returns Q_{ID_i} to A$_1$. Otherwise, if $ID_i = ID_t$, then $\tau_1$picks a random number $\alpha_i \in Z_q^*$ and calculates $Q_{ID_i} = \alpha_i Y \in G_1$ and inserts it in L_{H_1}, thereafter returns it to A$_1$. If $ID_i \neq ID_t$, $\tau_1$picks a random $\alpha_i \in Z_q^*$ and calculates $Q_{ID_i} = \alpha_i P \in G_1$ and inserts in the list L_{H_1}, thereafter returns α_i to A$_1$.
- H_2 *query*: τ_1 maintains a list L_{H_2} in the form of $(m_k, PS1_j, P_i, U_i, h_{ijk})$. When A$_1$ submits the query to H_2 oracle, τ_1 checks if L_{H_2} contains the tuple $(m_k, PS1_j, P_i, U_i, h_{ijk})$ then there is nothing to do and τ_1 returns h_{ijk} to A1. Otherwise, τ_1 picks a random $h_{ijk} \in Z_q^*$ and inserts in L_{H_2} and returns h_{ijk} to A1.
- H_3 *query*: τ_1 maintains a list L_{H_3} in the form of $(PS1_j, t_{1_j})$. When A1 submits the query of H_3 oracle, τ_1 checks if list L_{H_3} contains the tuple $(PS1_j, t_{1_j})$, then τ_1 returns t_{1_j} to A1. Otherwise $\Im$picks a random number $t_{1_j} \in Z_q^*$ and inserts in L_{H_2} and returns t_{1_j} to A1.

- H_4 *query*: τ_1 maintains a list L_{H_4} in the form of $(m_k, PS1_j, P_i, U_i, t_{ijk})$. When A1 submits the query of H_4 oracle, $\tau_1$checks if L_{H_4} contains the tuple $(m_k, PS1_j, P_i, U_i, t_{ijk})$ then there is nothing to do and τ_1 returns t_{ijk} to A1. Otherwise, $\tau_1$picks a random number $t_{ijk} \in Z_q^*$ and inserts in L_{H_4} and returns t_{ijk} to A1.
- H_5 *query*: $\tau_1$maintains a list L_{H_5} in the form of (m_k, b_i, W_i). When A1 submits the query of H_5 oracle, $\Im_1$ checks if L_{H_5} contains the tuple (m_k, b_i, W_i) then there is nothing to do and τ_1 returns b_i to A1. Otherwise, $\tau_1$picks a random number $b_i \in Z_q^*$ and inserts in L_{H_5} and returns b_i to A1.
- *Revealpseudonymqueries*: Suppose this request is submitted with an identity ID_i by A1. Then $\Im_1$ searches whether the list L contains for a tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ and checks the value of PS_j. If $PS_j \neq \perp$, then τ_1 returns PS_j to A1. Otherwise, τ_1 selects a random number $k_j \in Z_q^*$ and calculates $PS1_j = k_j Q_{ID_i}$ with ID_i corresponding tuple $(ID_i, \alpha_i, Q_{ID_i})$ and $PS2_j = k_j t_{1_j}$, where t_{1_j} is computed from the list L_{H_3}, then τ_1 answers with $PS_j = (PS1_j + PS2_j)$ to A1 and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in L_{H_3}. If L does not contains the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$, then τ_1 sets $PS_j = \perp$. If $PS_j \neq \perp$, then τ_1 answers with PS_j to A1. Otherwise, $\tau_1$selects randomly $k_j \in Z_q^*$ and calculates $PS1_j = k_j Q_{ID_i}$ with ID_i corresponding the tuple $(ID_i, \alpha_i, Q_{ID_i})$. If $ID_i \neq ID_t$ and $PS2_j = k_j t_{1_j}$, where t_{1_j} is computed from the list L_{H_3}, then τ_1 answers with $PS_j = (PS1_j + PS2_j)$ to A1 and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in L_{H_3}.
- *Revealpartialkeyqueries:* Suppose this request is submitted with an identity ID_i by A1. If $ID_i = ID_t$ then τ_1 stops the simulation, otherwise, if $ID_i \neq ID_t$ and L contains a tuple $(ID_i, x_i, P_i, pp_i, PS_j)$, then τ_1 checks if $pp_i = \perp$. If $pp_i \neq \perp$, then τ_1 answers with pp_i to A1. If $pp_i = \perp$, τ_1 checks the list L_{H1} and returns with pp_i to A1. If L does not contains a tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ then τ_1 sets $pp_i = \perp$ and τ_1 checks L_{H1}, then τ_1 sets $pp_i = \alpha_i P_{pub} = \alpha_i X \in G_1$ and returns it to A1.

- *Revealpublickeyqueries*: Suppose this request is submitted with an identity ID_i by A1. $\Im_1$ maintains a list that includes the tuple of the form $(ID_i, x_i, P_i, pp_i, PS_j)$. τ_1 checks whether $P_i = \perp$. If $P_i \neq \perp$, then τ_1 answers with P_i to A1. If $P_i = \perp$, then τ_1 randomly selects $v_i \in Z_q^*$ and set $P_i = v_i P$, then τ_1 returns P_i to A1 and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in the list. If L does not contain the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$, then $\Im_1$ sets $P_i = \perp$, randomly selects $v_i \in Z_q^*$ and sets $P_i = v_i P$. Then $\tau_1$returns P_i to A1 and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in the list.
- *Revealsecretkeyqueries*: Suppose this request is submitted with an identity ID_i by A1. $\Im$ maintains a list that includes the tuple of the form $(ID_i, x_i, P_i, pp_i, PS_j)$. τ_1 checks whether $x_i = \perp$. If $x_i \neq \perp$, then τ_1 answers with x_i to A1. If $x_i = \perp$, then τ_1 selects a random number $v_i \in Z_q^*$, sets $P_i = v_i P$ and returns x_i to A1. If the list does not contains the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$, then τ_1 sets $x_i = \perp$ and if $P_i \neq \perp$, then τ_1 selects a random number $v_i \in Z_q^*$, sets $P_i = v_i P$, and returns P_i to A1 and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in the list.
- *Replacepublickeyqueries*: τ_1 makes this query on (ID_i, P_i). τ_1 looks up the list L, if list L contains $(ID_i, x_i, P_i, pp_i, PS_j)$ then τ_1 replaces P_i with $P_i^{'}$ chosen by A1 and sets $x_i = \perp$. Otherwise if list L does not contains P_i then τ_1 sets $P_i = P_i^{'}$, $x_i = \perp$, $pp_i = \perp$ and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in the list.
- *Sign:* When A1 submits a request on (m_i, ID_i) then τ_1 looks up first the list L, list L_{H_1}, list L_{H_2}, list L_{H_3}, list L_{H_4}, list L_{H_5} then τ_1 does as follows.

If $ID_i = ID_t$, then τ_1 checks the list L_{H_1} and list L for the tuple $(ID_i, \alpha_i, Q_{ID_i})$ and $(ID_i, x_i, P_i, pp_i, PS_j)$, where $Q_{ID_i} = \alpha_i Y \in G_1$. If L contains $(ID_i, x_i, P_i, pp_i, PS_j)$ then τ_1 checks whether $x_i = \perp$. If $x_i = \perp$, then τ_1 goes for *replacekeyquery* to generate $x_i = v_i$, $P_i = v_i P$. If L does not contain $(ID_i, x_i, P_i, pp_i, PS_j)$ then τ_1 goes for *RevealPublickey* query to produce a pair (x_i, P_i) and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in the list.

To generate the signature, τ_1 makes query to list L_{H_5} for getting tuples (m_k, b_i, W_i), where $W_i = b_i X$, then τ_1 selects three random numbers $r_i, \gamma_i, b_i \in Z_q^*$ and computes $U_i = r_i.P - h_{ijk}^{-1}.b_i^{-1}.PS1_j.t_{1_j}$, $h_{ijk} = H_2(m_k, PS1_j, P_i, U_i) \in Z_q^*$, $t_{ijk} = H_4(m_k, PS1_j, P_i, U_i) \in Z_q^*$, $W = H_5(\Delta)$

$$V_{ijk} = h_{ijk}.r_i.W + t_{ijk}.x_i.P_{rsu_i}$$

τ_1 returns a signature (U_i, V_{ijk}) to A1 which can be verified easily by equation $e(V_{ijk}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}.U_i, W) e(t_{ijk}.P_i, P_{rsu_i})$.

If $ID_i \neq ID_t$, τ_1 checks the list L_{H_1} and list L for the tuple $(ID_i, \alpha_i, Q_{ID_i})$ and $(ID_i, x_i, P_i, pp_i, PS_j)$, where $Q_{ID_i} = \alpha_i P \in G_1$. If L contains $(ID_i, x_i, P_i, pp_i, PS_j)$ then $\Im_1$ checks whether $x_i = \perp$. If $x_i = \perp$, then τ_1 goes for *replacekeyquery* to generate $x_i = v_i$, $P_i = v_i P$. If L does not contain $(ID_i, x_i, P_i, pp_i, PS_j)$ then τ_1 goes for *RevealPublickeyquery* to produce a pair (x_i, P_i) and inserts the tuple $(ID_i, x_i, P_i, pp_i, PS_j)$ in the list.

To generate the signature, τ_1 makes query to list L_{H_5} for getting tuples (m_k, b_i, W_i), where $W_i = b_i X$, then τ_1 selects three random numbers $r_i, \gamma_i, b_i \in Z_q^*$ and set the values $pp_i = \alpha_i X, PS2_j = k_j t_{1_j}, P_{rsu} = cP$ computes $U_i = r_i.P$, $h_{ijk} = H_2(m_k, PS1_j, P_i, U_i) \in Z_q^*$, $t_{ijk} = H_4(m_k, PS1_j, P_i, U_i) \in Z_q^*$, $W = H_5(\Delta)$

$$V_{ijk} = \alpha_i.X.k_j.t_{1_j} + h_{ijk}.r_i.W + t_{ijk}.x_i.c.P$$

τ_1 returns a signature (U_i, V_{ijk}) to A1. It is very easy to verify the equation $e(V_{ijk}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}.U_i, W) e(t_{ijk}.P_i, P_{rsu_i})$.

In both the cases, signature generated on message m_k is valid. We can show it by verification of signatures.

$$ID_i = ID_t,\ V_{ijk} = h_{ijk}.r_i.W + t_{ijk}.x_i.P_{rsu_i},$$

$$e(V_{ijk}, P) = e(h_{ijk}.r_i.W + t_{ijk}.x_i.P_{rsu_i}, P)$$

$$= e(h_{ijk}.r_i.P, W) e(t_{ijk}.x_i.cP, P)$$

$$= e(PS1_j.t_{1_j}, X) e(h_{ijk}.U_i, W) e(t_{ijk}.x_i.P, cP)$$

$$= e(PS1_j.t_{1_j}, P_{pub}) e(h_{ijk}.U_i, W) e(t_{ijk}.x_i.P, P_{rsu_i})$$

By forking lemma [15], τ_1 can obtain two valid signature $\sigma^{*} = (U_i^{*}, V_i^{*})$ and $\sigma^{'*} = (U_i^{'*}, V_i^{'*})$ with the same random type of corresponding message m_k^{*} with identity ID_i^{*} and public key P_i^{*} provided by A1.

$$e(V_{ijk}^{*}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}^{*}.U_i, W) e(t_{ijk}^{*}.P_i^{*}, P_{rsu_i})$$

$$e(V_{ijk}^{'*}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}^{'*}.U_i, W) e(t_{ijk}^{'*}.P_i^{*}, P_{rsu_i})$$

τ_1 checks the list L_{H_1} and list L for the tuple $(ID_i, \alpha_i, Q_{ID_i})$ and $(ID_i, x_i, P_i, pp_i, PS_j)$, where $Q_{ID_i} = \alpha_i P \in G_1$. Then τ_1 makes query to list L_{H_5} for getting tuple (m_k, b_i, W_i), where $W_i = b_i X$. Finally, τ_1 returns the solution of CDH problem such that, $abP = (h_{ijk}^{'*}(V_{ijk}^{*} - c.t_{ijk}^{*}.P_i^{*}) - h_{ijk}^{*}(V_{ijk}^{'*} - c.t_{ijk}^{'*}.P_i^{'*})) / \alpha_i.k_j.t_{i_1}(h_{ijk}^{'*} - h_{ijk}^{*})$.

Analysis: We analyze the success of three events for solving the CDH problem by τ_1 with the probability Ω. With the assumption that A1 having an advantage ε to forge a signature with in a time span *t* can submit hash queries at most q_{H_i} times with H_i $(i = 1,2,3,4,5)$, q_k times *RevealPartialkey* queries, q_s times queries to *Revealsecretkey* queries, q_p times *revealpublickey* queries, q_{ps} times *revealpseudonym* queries and q_{sign} times *sign* queries. Also take assumption that A1 never repeats H_i query for the same input.

E1: τ_1 does not abort all queries of *Revealpartialkey* submitted by A1.

E2:A1 can forge a valid signature.

E3: the output of A1 is valid even τ_1 does not abort all queries submitted by A1.

Probability of success that A1 can win after all events happen is,

$P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_2 \mid E_1].P[E_3 \mid E_1 \wedge E_2]$

The probability that τ_1 does not abort all queries of A1 is at least $(1 - \frac{1}{q_{H_1}})^{q_k}$, when it takes *Revealpartialkey*at most q_k times.

The probability that τ_1 does not abort key extraction queries and A1's signature queries is at least ε , $P[E_2 \mid E_1] \geq \varepsilon$.

Probability of a nontrivial forgery output of A1 to be valid even if τ_1 does not abort all the queries of A1 is $\frac{1}{q_{H_1}}$.

$$\Omega = [E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_2 \mid E_1].P[E_3 \mid E_1 \wedge E_2] \geq (1 - \frac{1}{q_{H_1}})^{q_k}.\frac{1}{q_{H_1}}.\varepsilon$$

τ_1 could solve the CDH problem with non-negligible probability since the ε is non-negligible. This gets a contradiction against the hardness of CDH problem.

8.4.2 Theorem 2: In the Random oracle model, adversary A2 having advantage ε in forging a certificateless signature scheme wins the Game 1 if it is able to successfully construct an algorithms to solve the CDH problem in G_1 with non-negligible probability.

Proof: For a random instance $(P, X = aP, Y = bP)$ of the CDH problem in G_1 of the prime order q, we will establish an algorithm τ_2 to solve the CDH problem. Where $a, b \in Z_q^*$ are chosen randomly and are unknown to τ_2. In the Game 2, adversary A2 interacts with the $\Im_2$. $\Im_2$ selects a random number $\lambda \in Z_q^*$, sets the master key of KGC, computes the public key of KGC as $P_{pub} = \lambda P$ and also sets $P_{rsu} = aP = X$. Afterwards τ_2 sends the system parameters $\{G_1, G_2, e, P, P_{pub}, H_2, H_3, H_4, H_{5,} P_{rsu_1}, P_{rsu_2} ...., P_{rsu_i}\}$ to A2. Since A2 is type 2 adversary and has power to access the master key of KGC so $\Im_2$ and A2 can compute the partial private key of the user. No need of hash function H_1 to be modeled. τ_2 maintains list $L = (ID_i, x_i, P_i, PS_j)$.

- H_2 *query*: τ_2 maintains a list L_{H_2} in the form of $(m_k, PS1_j, P_i, U_i, h_{ijk})$. When A2 submits the query of H_2 oracle, τ_2 checks if list L_{H_2} contains the tuple $(m_k, PS1_j, P_i, U_i, h_{ijk})$. If so, then nothing has to be done and τ_2 returns h_{ijk} to A2. Otherwise $\tau_2$picks a random value $h_{ijk} \in Z_q^*$ inserts it in the list L_{H_2} and returns h_{ijk} to A2.
- H_3 *query*: τ_2 maintains a list L_{H_3} in the form of $(PS1_j, t_{1_j})$. When A2 submits the query of H_3 oracle, τ_2 checks if list L_{H_3} contains the tuple $(PS1_j, t_{1_j})$. If so, then τ_2 returns t_{1_j} to A2. Otherwise $\tau_2$picks a random $t_{1_j} \in Z_q^*$ and inserts in the list L_{H_3} and return t_{1_j} to A2.

- H_4 *query:* $\tau_2$maintains a list L_{H_4} in the form of $(m_k, PS1_j, P_i, U_i, t_{ijk})$. When A2 submits the query of H_4 oracle. $\tau_2$checks, If list L_{H_4} contains the tuple $(m_k, PS1_j, P_i, U_i, t_{ijk})$ then nothing has to be done and τ_2 returns t_{ijk} to A2. Otherwise τ_2 picks a random $t_{ijk} \in Z_q^*$ and insert in the list L_{H_4} and return t_{ijk} to A2.
- H_5 *query:* τ_2 maintains a list L_{H_5} in the form of (m_k, b_i, W_i). When A2submitting the query of H_5 oracle. τ_2 checks, If list L_{H_5} contains the tuple (m_k, b_i, W_i) then nothing has to be done and τ_2 returns b_i to A2. Otherwise $\tau_2$picks a random $b_i \in Z_q^*$ and insert in the list L_{H_5} and return b_i to A2.
- *Revealpseudonymqueries*: When the request is submitted on identity ID_i by adversary A2.

  If L contains the corresponding required tuple(ID_i, x_i, P_i, PS_j), then τ_2 checks if $PS_j = \perp$. if $PS_j \neq \perp$, then τ_2 answer the PS_j to A2. Otherwise $\Im_2$select randomly $k_j \in Z_q^*$and calculates $PS1_j = k_j Q_{ID_i} \in G_1$ with ID_i corresponding tuple $(ID_i, \alpha_i, Q_{ID_i})$and $PS2_j = k_j t_{1_j}$, where t_{1_j} is computed from the list L_{H_3}, then τ_2 answer the $PS_j = \left(PS1_j + PS2_j\right)$to A2 and insert in the tuple (ID_i, x_i, P_i, PS_j).

  If L does not holds the corresponding required tuple(ID_i, x_i, P_i, PS_j), then τ_2 set$PS_j = \perp$. if $PS_j \neq \perp$, then τ_2 answer the PS_j to A2. Otherwise $\Im_2$chooses randomly $k_j \in Z_q^*$and calculates $PS1_j = k_j Q_{ID_i} \in G_1$ with ID_i corresponding tuple $(ID_i, \alpha_i, Q_{ID_i})$, $PS2_j = k_j t_{1_j}$, where t_{1_j} is computed from the list L_{H_3}, then τ_2 answer the $PS_j = \left(PS1_j + PS2_j\right)$to A2 and insert in the tuple (ID_i, x_i, P_i, PS_j).
- *RevealPublickeyQueries:* When the request is submitted on identity ID_i by adversary A2. $\Im_2$ maintains a list (ID_i, x_i, P_i, PS_j). τ_2 checks whether $P_i = \perp$. If $P_i \neq \perp$, then τ_2 answer P_i to A2. If $P_i = \perp$ then τ_2 select randomly $v_i \in Z_q^*$ and set $P_i = v_i P$, then $\tau_2$returns P_i to A2 inserts the tuple(ID_i, x_i, P_i, PS_j) in the list.

If L does not contain list (ID_i, x_i, P_i, PS_j) and $ID_i = ID_t$, then τ_2 picks a random number $\gamma_i \in Z_q^*$. τ_2 calculates $P_i = \gamma_i P \in G_1$ and inserts in the list L, thereafter returns P_i to adversary A2.

If $ID_i \neq ID_t$, τ_2 randomly picks $\gamma_i \in Z_q^*$ and calculates $P_i = \gamma_i Y \in G_1$ and inserts in the list L, thereafter returns P_i to adversary A2.

- *RevealsecretkeyQueries*: When the request is submitted on identity ID_i by adversary A2, τ_2 maintains a list (ID_i, x_i, P_i, PS_j). τ_2 checks whether $x_i = \perp$. If $x_i \neq \perp$, then τ_2 answers x_i to A2. If $x_i = \perp$ then τ_2 randomly select $v_i \in Z_q^*$ and sets $P_i = v_i P$, then τ_2 returns x_i to A2.

  If $\Im_2$ does not contain list (ID_i, x_i, P_i, PS_j) then τ_2 puts $x_i = \perp$. If $P_i \neq \perp$ then τ_2 randomly selects $v_i \in Z_q^*$, sets $P_i = v_i P$, returns P_i to A2 and inserts the tuple (ID_i, x_i, P_i, PS_j) in list.

- *Sign:* When A2 submits a request on (m_i, ID_i) then τ_2 looks up first the list L, list L_{H_1}, list L_{H_2}, list L_{H_3}, list L_{H_4}, list L_{H_5} then τ_2 does as follows.

  If $ID_i \neq ID_t$, A2 will not submit the sign query on (ID_i, x_i, P_i, PS_j) to forge the signature.

$$e(V_{ijk}, P) = e(PS1_j T_j, P_{pub}) e(h_{ijk}.U_i, W) e(t_{ijk}.P_i, P_{rsu_i})$$

  If the above equation holds then the signature is valid otherwise τ_2 fails. If τ_2 does not fail then the signature on $(m_k^*, ID_i^*, P_i^*, PS_j^*)$ is

$$e(V_{ijk}^*, P) = e(PS1_j^* t_{1_j}^*, P_{pub}) e(h_{ijk}^*.U_i, W) e(t_{ijk}^*.P_i^*, P_{rsu_i})$$

  By setting, $PS1_j^* = k_j.Q_{ID_i}^*$, $Q_{ID_i}^* = H_1(ID_i^*)$, $P_{pub} = \lambda.P$, $P_{rsu} = a.P$, $P_i = \gamma_i^* Y = \gamma_i^* bP$ and $t_{1_j}^* = H_3(PS1_j^*)$

$$e(V_{ijk}^*, P) = e(k_j.Q_{ID_i}^*.t_{1_j}^*, \lambda P) e(h_{ijk}^*.U_i^*, b_i P) e(t_{ijk}^*.\gamma_i^*.bP, aP)$$

$$e(t_{ijk}^*.\gamma_i^*.bP, aP) = e(V_{ijk}^*, P)(e(k_j.Q_{ID_i}^*.t_{1_j}^*, \lambda P) e(h_{ijk}^*.U_i^*, b_i P))^{-1}$$

$$e(t_{ijk}^*.\gamma_i^*.abP, P) = e(V_{ijk}^* - \lambda(k_j.Q_i^*.t_1^* + h_{ijk}^*.U_i^*.b_i), P)$$

$$t_{ijk}^*.\gamma_i^*.abP = V_{ijk}^* - \lambda(k_j.Q_i^*.t_1^* + h_{ijk}^*.U_i^*.b_i)$$

$$abP = (t_{ijk}^*.\gamma_i^*)^{-1}.(V_{ijk}^* - \lambda(k_j.Q_i^*.t_1^* + h_{ijk}^*.U_i^*.b_i))$$

τ_2 gives the solution of CDH problem.

Analysis: We analyze the three event is to be success of solving the CDH problem by τ_2 with the probability Ω. With the assumption that A2 having an advantage ε to forge a signature with in a time span t can submit hash queries at most q_{H_i} times where H_i $(i = 1,2,3,4,5)$, q_s times *Revealsecretkey* queries, q_p times *revealpublickey* queries, q_{ps} times *revealpseudonym* queries and q_{sign} times *sign* queries. Also take assumption that A2 never repeats H_i *query* for the same input.

E1: τ_2 does not abort all the queries of *Revealsecretkey* submitted by A2.

E2: A2 can forge a valid signature.

E3: the output of the A2 is valid even τ_2 does not abort all the queries of A2.

Probability of success that A2 can win after all events happen is,

$$P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_2 \mid E_1].P[E_3 \mid E_1 \wedge E_2]$$

The probability that τ_2 does not abort all the queries of A2 is at least $(1 - \frac{1}{q_{H_1}})^{q_s}$, when it takes *Revealsecretkey* at most q_s times.

The probability that $\Im_2$ does not abort key extraction queries and A2's signature queries is at least ε, $P[E_2 \mid E_1] \geq \varepsilon$.

Probability of nontrivial forgery output of the A2 to be valid even if τ_2 does not abort all the queries of A2, is $\frac{1}{q_{H_1}}$.

$$\Omega = [E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_2 \mid E_1].P[E_3 \mid E_1 \wedge E_2] \geq (1 - \frac{1}{q_{H_1}})^{q_s} . \frac{1}{q_{H_1}} . \varepsilon$$

τ_2 could solve the CDH problem with the non-negligible probability since the ε is non-negligible. This gives a contradiction against the hardness of CDH problem.

8.4.3 Security Analysis of CL-AS scheme

8.4.3.1 Theorem 3: If the base certificateless signature scheme is secure against identity and adaptive chosen message attacks then certificateless aggregate signature scheme is also secure against existential forgery in the chosen aggregate model.

Proof: For a random instance $(P, X = aP, Y = bP)$ of the CDH problem in G_1 of the prime order q, we will establish an algorithm $\Im$ to solve the CDH problem. Where $a,b \in Z_q^*$ are chosen randomly and are unknown to τ. $\Im$ randomly selects an identity ID_i as a challenge and sends $\{G_1, G_2, e, P, P_{pub}, H_2, H_3, H_4, H_{5,} P_{rsu}\}$ to A. τ selects $c \in Z_q^*$ and sets $P_{pub} = X$ and $P_{rsu} = cP$. Now τ is ready to execute the oracle queries. τ maintains a list $(ID_i, x_i, P_i, pp_i, PS_j)$, while A1 can query throughout the game.

- H_1 *queries*: After submitting an identity to H_1 oracle, τ maintains a list L_{H_1} in the form of $(ID_i, \alpha_i, Q_{ID_i})$. If list L_{H_1} contains the tuple $(ID_i, \alpha_i, Q_{ID_i})$ then nothing has to be done and τ returns Q_{ID_i} to A. If $ID_i = ID_t$, then $\Im$ randomly picks $\alpha_i \in Z_q^*$, calculates $Q_{ID_i} = \alpha_i.P_{pub} \in G_1$ and insert in the list L_{H_1} , thereafter returns Q_{IDi} to adversary A. If $ID_i \neq ID_t$, τ randomly picks $\alpha_i \in Z_q^*$, calculates $Q_{ID_i} = \alpha_i P \in G_1$ and insert in the list L_{H_1} , thereafter returns Q_{ID_i} to adversary A. L_{H_1} maintains tuple $(ID_i, \alpha_i, Q_{ID_i})$ in both the cases. Now, A has an output of n user's $(u_1, u_2, u_3 ....u_n)$ with their identities $L_{ID}^* = (ID_1^*, ID_2^* .....ID_n^*)$, pseudonym identities $L_{PS}^* = (PS_1^*, PS_2^*, ...PS_n^*)$, public keys $L_P^* = (P_1^*, P_2^*, ...P_n^*)$ and an aggregate signature $\sigma^* = (U_1^*, U_2^* ....U_n^*, V^*)$ on the message set $\{m_1^*, m_2^* ...., m_n^*\}$. $\Im$ gets the corresponding tuples $(ID_i, \alpha_i, Q_{ID_i})$ where $i \in 1$ to n by processing L_{H_1} only when $ID_k = ID_t$ and $ID_i \neq ID_t$ for $t \in 1$ to n and $t \neq k$. The sign query has never been submitted otherwise $\Im$ fails and stop the simulation where $Q_k = \alpha_k P_{pub}, Q_j = \alpha_j P_{pub}$ for $j \in [1, n]$ and $j \neq k$. New generated signature is $\sigma^* = (U_1^*, U_2^* ....U_n^*, V^*)$ which can be verified by the following aggregate verification equation.

$$e(V, P) = e(\sum_{i=1}^{n} PS1_i^*.T_i^*, P_{pub}) e(\sum_{i=1}^{n} h_i^*.U_i^*, W) e(\sum_{i=1}^{n} t_i^*.P_i^*, P_{rsu})$$

τ looks up the list L , list L_{H_1} ,list L_{H_2} ,list L_{H_3} ,list L_{H_4} ,list L_{H_5} to find tuples $(ID_i^*, x_i^*, P_i^*, pp_i^*, PS_i^*)$, $(m_i^*, PS1_i^*, P_i^*, U_i^*, h_i^*)$, $(PS1_j^*, t_{1_j}^*)$, $(m_i^*, PS1_i^*, P_i^*, U_i^*, t_i^*)$, (m_k^*, b_i^*, W_i^*) . Then sets $V_i^* = \alpha_i.P_{pub}$ that can be verified by $e(V_i^*, P) = e(Q_{ID_i}, P_{pub})$ for $i \in 1$ to n .

Finally, $\Im$ generates $V_i^{'*}$ as $V^* - \sum_{i=1, i\neq k}^{n} V_i^*$. Now

$$V^{'*} = \sum_{i=1}^{n} pp_i.PS2_i + \sum_{i=1}^{n} h_i^* r_i^* W + \sum_{i=1}^{n} t_i^* x_i^* P_{rsu}$$

τ Takes, $U_i^* = r_i^* P$ and $r_i^* \in Z_q^*$ where $i \in 1$ to n. Now τ selects a random numbers $y_k^* \in Z_q^*$ and $z_k^* \in Z_q^*$ calculates $U^{i^*} = (y_k^*)^{-1} \sum_{i=1}^{n} h_i^*.U_i^*$ and $P_k^{'*} = (z_k^*)^{-1} \sum_{i=1}^{n} t_i^*.P_i^*$.

τ submits *replacepublickey* query and updates public key. Then sets $H_2(m_k^*, PS1_k^*, P_k^*, U_k^*, h_k^*)$ as y_k^* and $H_4(m_k^*, PS1_k^*, P_k^*, U_k^*, t_k^*)$ as z_k^* . So $(U^{'*}, V^{'*})$ is a valid signature for the identity ID_k^* with the pseudonym PS_k^* and corresponding public key P_k^* on the message m_k^* . We can obtain a valid aggregate signature by τ without submitting *sign* queries in the following manner:

$$e(\sum_{i=1}^{n} PS1_k^* t_{1_k}^*, P_{pub}) e(y_k^*.U^{'*}, W) e(z_k^*.P_i^{'*}, P_{rsu})$$
$$= e(\sum_{i=1}^{n} PS1_k^* t_{1_k}^*, P_{pub}) e(y_k^*.(y_k^*)^{-1} \sum_{i=1}^{n} h_i^*.U_i^*, W) e(z_k^*.(z_k^*)^{-1} \sum_{i=1}^{n} t_i^*.P_i^*, P_{rsu})$$
$$= e(\sum_{i=1}^{n} pp_i^*.PS2_k^*, P), e(\sum_{i=1}^{n} h_i^*.r_i^*.P, W) e(\sum_{i=1}^{n} t_i^*.x_i^*.P, P_{rsu})$$
$$= e(\sum_{i=1}^{n} pp_i^*.PS2_k^*, P), e(\sum_{i=1}^{n} h_i^*.r_i^*.W, P) e(\sum_{i=1}^{n} t_i^*.x_i^*.P_{rsu}, P)$$
$$= e(\sum_{i=1}^{n} pp_i^*.PS2_k^* + \sum_{i=1}^{n} h_i^*.r_i^*.W + \sum_{i=1}^{n} t_i^*.x_i^*.P_{rsu}, P)$$
$$= e(V^{i^*}, P)$$

Output $(U^{'*}, V^{'*})$ generated by τ is a forged signature of the CL-AS scheme.

Analysis: We analyze the three event is to be success of solving the CDH problem by τ with the probability Ω. With the assumption that A having an advantage ε to forge a signature with in a time span t can submit hash queries at most q_{H_i} times H_i $(i = 1,2,3,4,5)$, q_k times *RevealPartialkey* queries, q_s times *Revealsecretkey* queries, q_p times *revealpublickey* queries, q_{ps} times *revealpseudonym* queries and q_{sign} times *sign* queries. Also take assumption that A never repeats H_i query for the same input.

E1: τ does not abort all the queries of *Revealpartialkey* submitted by A.

E2: A can forge a valid signature.

E3: the output of *A* is valid even if τ does not abort all queries submitted by *A*.

Probability of success that *A* can win after all events happen is,

$$P[E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_2 \mid E_1].P[E_3 \mid E_1 \wedge E_2]$$

The probability that τ does not abort all the queries of *A* is at least $(1-\frac{1}{q_{H_1}})^{q_k}$, when it takes *Revealpartialkey* at most q_k times.

The probability that τ does not abort key extraction queries and *A*'*s* signature queries is at least ε which is given by $P[E_2 \mid E_1] \geq \varepsilon$.

Probability of a nontrivial forgery output of *A* to be valid even if τ does not abort all the queries of *A* is $\frac{1}{q_{H_1}}(1-\frac{1}{q_{H_1}})^{n-1}$.

$$\Omega = [E_1 \wedge E_2 \wedge E_3] = P[E_1].P[E_2 \mid E_1].P[E_3 \mid E_1 \wedge E_2]$$

$$\geq (1-\frac{1}{q_{H_1}})^{q_k}.\frac{1}{q_{H_1}}(1-\frac{1}{q_{H_1}})^{n-1}.\varepsilon$$

$$\geq (1-\frac{1}{q_{H_1}})^{q_k+n-1}.\frac{1}{q_{H_1}}.\varepsilon$$

τ could solve the CDH problem with the nonnegligible probability since the ε is non negligible. This gives a contradiction against the hardness of CDH problem.

8.5 Comparison of CL-AS scheme:

Table-8.2: Comparison of cost based on operations and running time (ms) involved

CL-AS	Type	Sign	Sign	Individual verify	Individual verify	Aggregate verify	Aggregate verify
[15]	*Sync*	3S	1.17 ms	4P	4P=12.84ms	(n+3)P	(n+3)3.21ms
[57]	*Sync*	5P	1.95 ms	5P+2S	5P+2S=16.83 ms	5P+2nS	(16.05+0.78n) ms
[48]	*Ad-hoc*	2S	0.76 ms	3P	3P=9.63ms	(2n+1)P	(2n+1)3.21ms
[48]	*Sync*	3S	1.17 ms	3P	3P=9.63ms	(n+2)P+ nS	((n+2)3.21+n 0.39)ms
[18]	*Ad-hoc*	3S	1.17 ms	3P+2S	3P+2S=10.39 ms	3P+2nS	(9.63+0.78n) ms
Our scheme	*Ad-hoc*	3S	1.17 ms	3P+3S	3P+3S=10.39 ms	3P+3nS	(9.63+1.17n) ms

Sync means normal mode of transfer; *Ad hoc* means temporary mode of transfer; *S* denotes scalar multiplication; *P* denotes pairing

In this section, we compare our scheme with some existing schemes in Table-8.2. Three types of cost hash cost, scalar multiplication cost, and pairing cost are involved in CL-AS scheme. We take the setup of experiment as in [59] which give processing time for the Tate pairing on a 159-bit subgroup of an MNT curve with an implanting degree 6 at an 80-bit security level, running on an Intel i7 3.07 GHz machine. The results of the experiment obtained are as follows: Pairing cost is 3.21 ms, Signing cost is 0.39 ms and Hashing cost is 0.09 ms. Hash cost is very nominal so we omit this cost in our comparison. Pairing cost is very high which increases the computational effort of a pairing based CL-AS scheme. Zhang and Zhang [15] used 3 scalar point multiplications in signature algorithm and 4 pairing operations in verification while they used (n+3) pairing operations in aggregate verifying algorithm in CL-AS scheme which makes it very costly due to high cost of pairing operation. Zhang et al. [57] proposed a CL-AS scheme with 5 scalar point multiplications in signature algorithm and 5 pairing operations with 2 scalar point multiplications in verification. Zhang et al. used 5 pairing operations in their CL-AS scheme which is very costly and time consuming. Gong et al. [48] proposed two CL-AS schemes with very high cost owing to usage of many pairing operations. Xiong et al. [18] used 3 paring and 2 scalar point multiplication operations in his CL-AS scheme for good efficiency but it is found insecure in [28, 34, 35, 36].

CHAPTER 9

Conclusion and Future work

Due to the information technology revolution in the twenty one century, information security becomes a major issue for the researchers. Cryptography is a very important tool for providing the security in communication. Digital signature is a cryptographic technique that ensures the confidentiality, authentication, integrity and non repudiation. A lot of digital signatures are existing like group signature, ring signature, proxy signature, blind signature, partially blind signature, identity based signature, certificateless signature, aggregate signature scheme and a lot of schemes. Among all the existing schemes, certificateless schemes are very efficient techniques in recent scenario which is very helpful to provide security. The aggregate signature scheme is a many to one map that allows the different signature on the different message map to a single signature. This feature is very beneficial in such environment where bandwidth and computational time saving is required for e.g., wireless sensor network, vehicular ad-hoc network, internet of things and an endless list. Certificateless aggregate signature is much efficient scheme that enjoying both the features of certificateless and aggregate concept.

We adopt a certificateless signature scheme in our dissertation because of its efficient properties. We have done a review of many certificateless signature schemes. After studying a lot signature schemes we have done a cryptanalysis of these schemes. We found many schemes like Deng et al signature scheme, Horng et al signature scheme, Malhi and Batra signature scheme and Liu et al Signature scheme be insecure many concrete attacks like adaptive chosen message attack, malicious but passive attack, honest but curious attack and collision insider attack. We have done cryptanalysis of He et al scheme and demonstrate that their scheme is failing to protect against malicious, but passive attack and honest, but curious attack thereafter we proposed a modified certificateless signature scheme. Also, we prove the security of our signature scheme in the random oracle model with Diffie-Hellman assumption.

We proposed an efficient certificateless aggregate scheme for vehicular adhoc networks. Our scheme removes the critical key escrow problem which generates in the Identity

Based Signature schemes. Security of the proposed Certificateless signature scheme proves under the Diffie-Hallman assumption with the Random Oracle Model technique. Results are evaluated under the NS-2 environment. By comparison table, we proved that the proposed CL-AS is cost effective and take lesser cost comparing with the other proposed previous schemes. This scheme takes less bandwidth which makes it suitable for adhoc networks.

We have proposed a novel certificateless signature scheme using the concept of aggregate signature for secure communication in healthcare wireless sensor networks. Our proposed scheme has advantages of aggregate signature and certificateless signature. The proposed helps to protect the online data, from the unauthorized entities in healthcare wireless sensor network. The security of our construction is proved by Diffie Hallman assumption under the random oracle model that claims our system is not forgeable against the present adversaries in the system. Through experimental results, we have proved that our certificateless aggregate signature scheme is more computational and energy efficient as compared to the existing schemes.

In the future work, we can construct certificateless signature scheme with current technology like Big data, internet of things and cloud computing. We can set a framework for authentication by using the signature scheme.

[1] Hu Xiong, Zhen Qin, Athanasios V. Vasilakos, :Introduction to Certificateless Cryptography, CRC Press

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

[3] W. Diffie, M.E Hellman,New Directions in Cryptography, IEEE Trans. on Inform. Theory, 22(6), 644-654 (1976).

[4] D.J.Cook, J.C.Augusto, V.R Jakkula, Ambient intelligence: technologies, applications, and opportunities, *Pervasive and Mobile Computing*, 5, 277–298(2009)DOI 10.1016/j.pmcj.2009.04.001.

[5] M. A. Ameen, J. Liu, K., Kwak, Security and Privacy Issues in Wireless Sensor Networks for Healthcare Applications, *Journal of Med Syst* 36, 93–101(2012), DOI 10.1007/s10916-010-9449-4.

[6] M. R.Yuce, S.W,Ng , N. L.Myo., J. Y.Khan,W. Liu, Wireless Body Sensor Network Using Medical Implant Band, *Journal of Med Syst*, 31(6), 467-474 (2007), DOI 10.1007/s10916-007-9086-8.

[7] P.Kumar, H.J.Lee, Security Issues in Healthcare Applications Using Wireless Medical Sensor Networks: A Survey, *Sensors*12, 55-91 (2012), DOI:10.3390/s120100055.

[8] A. Shamir, "Identity based cryptosystems and signature schemes",G.R. Blakley, D. Chaum (Eds.), Crypto'84, LNCS 196, Springer-Verlag, Santa Barbara, California, USA, 1984, pp. 47–53.

[9] Al-Riyami, S., Paterson, K. "Certificateless Public Key Cryptography",Asiacrypt'03,

[10] X. Huang, W. Susilo, Y. Mu, F. Zhang, "On The Security Of A Certificateless Signature Scheme", in: Proceedings of the CANS, Lecture Notes in Computer Science, vol. 3810, 2005, pp. 13–15. LNCS 2894, Springer-Verlag. (2003) 452-473.

[11] D. Boneh, C. Gentry, B. Lynn, H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", E. Biham (Ed.), EUROCRYPT 2003, LNCS 2656,

[12] Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong, Wei Wu " Certificateless signatures: New Schemes and Security Models" . The computer journal, vol 55 No.4, 2012.

[13] Z. Eslami, N.Pakniat“Certificateless aggregate signcryption: Securitymodel and a concrete construction securein the random oracle model”, Journal of King Saud University – Computer and Information Sciences (2014) 26, 276–286.

[14] Yu-Chi Chen, GwoboaHorng, Chao-Liang Liu, Yuan-Yu Tsai, and Chi-Shiang Chan “Efficient Certificateless Aggregate Signature Scheme,” JOURNAL of electronic science and technology, vol. 10, no. 3, september 2012pp 209-214.

[15] L. Zhang, F. Zhang, “A New Certificateless Aggregate Signature Scheme”, Comput. Commun. 32 (6) (2009) 1079–1085.

[16] Z. Xu, X. Liu, G. Zhang, W. He, G. Dai, and W. Shu, “A certificateless signature scheme for mobile wireless cyber-physical systems,” in 28th International Conference on Distributed Computing Systems Workshops, ICDCS Workshops 2008, pp. 489–494, chn, June 2008.

[17] Xiong, H., Guan, Z., Chen, Z., Li, F.: An efficient certificateless aggregate signature with constant pairing computation. Inform. Sci. 219, 225–235 (2013).

[18] Xiong, H., Wu, Q., Chen, Z.: Strong Security Enabled Certificateless Aggregate Signatures Applicable to Mobile Computation. Third International Conference on Intelligent Networking and Collaborative Systems, Fukuoka, Japan, 92-99 (2011), DOI 10.1109/INCoS.2011.151.

[19] Z. Xu, X. Liu, G. Zhang, and W. He, “McCLS: certificateless signature scheme for emergencymobile wireless cyber-physical systems,” International Journal of Computers, Communications and Control, vol. 3, no. 4, pp. 395–411, 2008.

[20] Zhang, Z. and Wong, D. “Certificateless Public-Key Signature: Security Model and Efficient Construction”. Lecture Notes in Computer Science, 3989, 293-308 (2006).

[21] G. Sharma, S. Bala, and A. K. Verma “On the Security of Certificateless Signature Schemes” International Journal of Distributed Sensor Networks Volume 2013.

Springer-Verlag, Warsaw, Poland, 2003, pp. 416–432.

[22] A. K.Malhi, S.Batra “An efficient certificateless aggregate signature scheme for vehicular ad-hoc networks” Discrete Mathematics & Theoretical Computer Science. /4DMTCS, 2015, 17 (1).

[23] J.K. Liu, M.H. Au, W. Susilo, “Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model”, ASIACCS’07, 2007, pp. 273–283.

[24] H.Hou, X.Zhang,X.Dong, “Improved certificateless aggregate signature scheme ”, Journary of Shandong University (Natural Science), 48(9),pp. 29-34,2013.

[25] Jiang Deng ,Chunxiang Xu, Huai Wu and Liju Dong, “A new certificateless signature with enhanced security and aggregation version” , Special issue paper, currency and computation: practice and experience (2015).

[26] Jiang Deng, Chunxiang Xu, Huai Wu, Guangyuan Yang, “An Improved Certificateless Aggregate Signature”, 2014 IEEE Internatational Conference on Computer and Information Technology pp 919-922.

[27] Liu H, Wang S, Liang M, Chen Y.: New Construction of Efficient Certificateless Aggregate Signatures. International Journal of Security and Its Applications Vol.8, No.1 (2014), pp. 411-422.7

[28] Debiao He, Miaomiao Tian, Jianhua Chen, “Insecurity of an efficient certificateless aggregate signature with constant computations ” Information sciences 268 (2014) pp 458-462.

[29] Yulei Zhang and Caifen Wang “Comment on new construction of efficient certificateless aggregate signatures”, International journal of security and its applications, vol. 9,No. 1 (2015), pp 147-154.

[30] Jayo, U.H., Mmmu A.S.K., Iglesia I.D.: Reliable Communication in Cooperative Ad hoc Networks. Chapter 6, 213-244, DOI dx.doi.org/10.5772/59041.

[31] Yum, D. H., Lee, P. J.: Generic construction of certificateless signature. Information Security and Privacy, LNCS_3108, 200–211, (2004), DOI 10.1007/978-3-540-27800-9_18.

[32] Hu, B., Wong, D., Zhang, Z., Deng, X.: Key Replacement Attack Against a Generic Construction of Certificateless Signature. Proceedings of the ACISP’06, LNCS **4058**, 235–346 (2006).

[33] Cao X., Paterson K.G., Kou W.: An attack on a certificateless signature scheme. Report 2006/367, Cryptology, ePrint Archive,(2006).

[34] Futai Zhang , Limin Shen, Ge Wu “Notes on the security of certificateless aggregate signature schemes. Inform. Sci. **287**, 32–37 (2014).

[35] Tu, H., He, D., Huang B.:Reattack of a Certificateless Aggregate Signature Scheme with Constant Pairing Computations. Scientific World Journal **2014**, 10 pages, (2014) DOI 10.1155/2014/343715.

[36] Cheng, L., Wen, Q., Jin , Z., Zhang, H., Zhou, L.: Cryptanalysis and improvement of a certificateless aggregate signature scheme. Inform. Sci. **295**, 337-46 (2015).

[37] Hu, B.C., Wong, D.S., Zhang, Z., and Deng, X.: Certificateless signature: a new security model and an improved generic construction. Designs, Codes and Cryptography, **42**(2), 109–126 (2007).

[38] Du, H. and Wen, Q.: Efficient and provably-secure certificateless short signature scheme from bilinear pairings. Computer Standards & Interfaces, **31**(2):390–394, (2009).

[39] Choi, K.Y., Park, J. H., Hwang, J. Y., Lee, D. H.: Efficient certificateless signature schemes. Applied Cryptography and Network Security, **4521**, 443–458 (2007).

[40] He D., Wang D.: Robust biometrics-based authentication scheme for multiserver environment, IEEE System Journal, 9(3), 816-823 (2015).

[41] He D., Kumar N., Shen H., Lee H. J.: One to Many authentication for access control in mobile pay TV system, Science China Information Science, 2016, 59 (5).

[42] Martinelli F., Mercaldo F., Orlando A., Nardone V., Santone A., Sangaiah A.K.,: Human behavior characterization for driving style recognition in vehicle system, Computers & Electrical Engineering 1-16 (2018) DOI 10.1016/j.compeleceng.2017.12.050.

[43] Chahal M., Harit S., Mishra K.K., Sangaiah A.K., Zheng Z.: A Survey on software-defined networking in vehicular ad hoc networks: Challenges, applications and use cases,Sustainable Cities and Society, VOL.35, 830-840 (2017), 10.1016/j.scs.2017.07.007

[44] Chen C., Min X., Qiu T. Q., Liu L., Sangaiah A.K.: Latency estimation based on traffic density for video streaming in the internet of vehicles, Computer Communications, Vol. 111, 176-186 (2017), DOI 10.1016/j.comcom.2017.08.010

[45] Chen C., Liu X., Tie Q. , Sangaiah A.K.: A short-term traffic prediction model in the vehicular cyber–physical systems. Future Generation Computer Systems, 2017 DOI 10.1016/j.future.2017.06.006,

[46] Lee, U., Magistretti, E., Zhou, B., Gerla, M., Bellavista, P., Corradi A.: Mobeyes smart mobs for urban monitoring with a vehicular sensor network. IEEE Wireless Commun. **13** (5), 52–57 (2006)..

[47] Shim, K.A.: CPAS: an efficient conditional privacy-preserving authentication scheme for vehicular sensor networks, IEEE Trans. Vehic. Technol. **61** (4), 1874–1883 (2012).

[48] Gong, Z., Long, Y., Hong, X., K. Chen.: Two certificateless aggregate signatures from bilinear maps. Proceedings of the IEEE SNPD, 3, 188–193 (2007), DOI 10.1109/snpd.2007.132.

[49] Hubaux, J.P., Capkun, S., Luo, J.: The security and privacy of smart vehicles. IEEE Security Privacy **2** (3) 49–55 (2004).

[50] Raya, M., Hubaux, J.P.: Securing vehicular ad hoc networks. J. Comput. Security **15** (1) 39–68 (2007).

[51] T. Dimitriou,K. Loannis,Security Issues in Biomedical Wireless Sensor Networks,Proceedings of Applied Sciences on Biomedical and Communication Technologies (ISABEL'08), Aalborg, Denmark, (2008) DOI 10.1109/isabel.2008.4712577.

[52] K. Lorincz, D.J. Malan, J.Fulford, T.R.F., A. Nawoj, A. Clavel, V. Shayder, G. Mainland, M. Welsh, Sensor Networks for Emergency Response: Challenges and Opportunities,*IEEE Pervas. Comput.*3, 16-23 (2004) DOI 10.1109/mprv.2004.18.

[53] Malan, D. Jones, T.F. Welsh, M. Moulton, S. CodeBlue,An Ad-Hoc Sensor Network Infrastructure for Emergency Medical Care, Proceedings of the *Applications of Mobile*

[54] M. Gorantla, A. Saxena, An Efficient Certificateless Signature Scheme, In *Computational Intelligence and Security*, LNCS 3802,110–116 (2005) DOI 10.1007/11596981_16.

[55] Y.C. Chen, R.Tso, W.Susilo, X. Huang, GHorng., Certificateless Signatures: Structural Extensions of Security Models and New Provably Secure Schemes, IACR Cryptology ePrint Archive (2013).

[56] K..A. Shim, Security Models for Certificateless Signature Schemes Revisited, *Inform. Sci.* 296, 315-321 (2015)DOI 10.1016/j.ins.2014.10.055.

[57] L. Zhang, B. Qin, Q. Wu, F. Zhang, Efficient Many-to-One Authentication with CertificatelessAggregate Signatures,*Comput. Netw.*54 (14) 2482–2491 (2010)

[58] D. He, M. Tian, J. Chen, Insecurity of an Efficient Certificateless Aggregate Signature with Constant Computations, *Inform. Sci.* 268, 458-462, (2014).

[59] S.J.Horng, S.F.Tzeng, P. H.Huang, X.Wang, T.Li, M. K. Khan, An Efficient CertificatelessAggregate Signature with Conditional Privacy-Preserving for Vehicular Sensor Networks, *Inform. Sci.* 317, 48–66(2015).

[60] J.Camenisch, S.Hohenberger, M.O.Pedersen, Batch Verification of Short Signatures, Proceedings of the *Eurocrypt*, LNCS 4515, 246–263 (2007).

[61] C. Zhang, R. Lu, X. Lin, P.H. Ho, X. Shen, An Efficient Identity-Based Batch Verification Scheme for Vehicular Sensor networks. Proceedings of the *IEEE Infocom*, 816–824 (2008), DOI 10.1109/infocom.2008.58.

[62] L.Shen, J.Ma1,X.Liu, M.Miao, A Provably Secure Aggregate Signature Scheme for Healthcare Wireless Sensor Networks,*Journal of Med Syst* (2016) 40:244, DOI 10.1007/s10916-016-0613-3.

[63] J.S.Coron, On the Exact Security of Full Domain Hash, in the proceeding *Advances in Cryptology-CRYPTO 2000*, LNCS 1880, 229–235 (2000).

[64] D. He, S. Zeadally, L. Wu, " Certificateless Public Auditing Scheme for Cloud-Assisted Wireless Body Area Networks" IEEE Systems Journal, 99, 1-10 DOI 10.1109_jsyst.2015.24286.

[65] D. He, S. Zeadally, N. Kumar, J. H. Lee,"Anonymous Authentication for Wireless Body Area Networks With Provable Security" IEEE Systems Journal, 1-12, 2016, 10.1109/JSYST.2016.2544805.

[66] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci,"A survey on sensor networks", IEEE Communications Magazine, 40 (8), 102-114, 2002, DOI 10.1109_mcom.2002.1024422.

[67] V. sai and M. H. mickle,"Exploring energy efficient architectures in passive wireless nodes for IoT applications," IEEE Circuits Syst. Mag., 14(2) 48-54, 2014, DOI 10.1109_MCAS.2014.2314265.

[68] M. Chan, D. Estève, C. Escriba, E. Campo," A review of smart homes—Present state and future challenges" Computer Methods and Programs in Biomedicine, 91(1), 55-81, 2008, 10.1016/j.cmpb.2008.02.001.

[69] Lee, U., Magistretti, E., Zhou, B., Gerla, M., Bellavista, P., Corradi A.: Mobeyes smart mobs for urban monitoring with a vehicular sensor network. IEEE Wireless Commun. **13** (5), 52–57 (2006).