STUDIES ON PAIRING-BASED AND CONSTANT ROUND DYNAMIC GROUP KEY AGREEMENT PROTOCOLS

Thesis submitted to Indian Statistical Institute

by

Ratna Dutta
Stat-Math Unit
Indian Statistical Institute
2005

Studies on Pairing-Based and Constant Round Dynamic Group Key Agreement Protocols

Thesis submitted to Indian Statistical Institute in partial fulfillment of the requirements for the award of the degree of
Doctor of Philosophy

by

Ratna Dutta
Stat-Math Unit
Indian Statistical Institute
203, B. T. Road, Calcutta 700 108, INDIA

under the supervision of

Prof. Rana Barua
Stat-Math Unit
Indian Statistical Institute
203, B. T. Road, Calcutta 700 108, INDIA
e-mail : rana@isical.ac.in

To my family

Preface

This thesis describes research that I conducted during the course of my Ph.D. study at Indian Statistical Institute (ISI), Kolkata. I hope that this work is of use and will be carried on.

I would like to begin by thanking my supervisor, Prof. Rana Barua for his support and endless patience. He has provided me background in Cryptography, Combinatorics and Theory of Automata during the period 2000-2002 of my course work of M.Tech in Computer Science at Indian Statistical Institute. I really appreciate his extraordinary patience in reading my numerous inferior drafts, for listening and analyzing all my ideas, forcing me to be thorough with details, not to take shortcuts and for all his interesting discussions and contributions. He is a co-author of all my works included in this thesis. His constant supervision, valuable advice and continuous encouragement have added greatly to the quality of this thesis.

I most sincerely thank Prof. Bimal Roy for introducing me into the area of Cryptography and Coding Theory with such an excellence during my course work of M.Tech. I was fascinated by his teaching since the first day. Apart from being grateful for introducing me to cryptography, I also have to thank him for providing all supports through advise, encouragement and financial aid.

Another person of importance to my work is Dr. Palash Sarkar. I would like to express my utmost gratitude to him for introducing me to the field of Elliptic Curve Cryptography and Pairings. I have enjoyed and benefited from his fruitful discussions and unlimited valuable suggestions. He gave me insight to find problems with several revisions of papers and helped me to learn to think. He inspired me to design key agreement protocols in multi-party scenario and contributed significantly on some works described in this thesis and my research. In fact, Chapter 3 and Chapter 5 are based on our published papers which are done jointly with him. The most important for me was probably that he taught me how to write. If I am a better writer or speaker because of my time as a Ph.D. student, it is largely because of him and Prof. Rana Barua.

I must thank all anonymous referees of my published papers, whose comments have always added a new dimension to my works. I am thankful to the faculty and staff of the Stat Math Unit and Applied Statistics Unit for all the help during my studies in ISI. I would also like to express my special thanks to all members of Cryptology Research Group of ISI.

Moreover I thank many other people who have provided much emotional supports in my life. In particular, I wish to express my whole hearted gratitude to my family and friends for their continuous support and encouragement. I dedicate this thesis to them.

Contents

1 Introduction **1**

1.1 Synopsis 6

2 Preliminary Background and Definitions **8**

2.1 Introduction 8

2.2 Elementary Concepts on Elliptic Curves 8

2.2.1 Supersingular Elliptic Curves 10

2.2.2 Divisors and Weil Pairing 10

2.2.3 Weil Pairing and Bilinear Map 11

2.2.4 Notations and Terminologies 14

2.3 Diffie-Hellman Problems 14

2.4 Bilinear Diffie-Hellman Problems 17

2.5 Miscellaneous Problems 17

2.6 Security Models 18

2.6.1 Security Model for Encryption Schemes 19

2.6.2 Security Model for Digital Signature Schemes 22

2.6.3 Security Attributes and Model for Key Agreement Schemes 25

2.7 Conclusion 30

3 Overview of Key Agreement Protocols **31**

3.1 Introduction 31

3.1.1 Two-party key agreement 33

3.1.2 Three-party key agreement 36

3.1.3 Group key agreement 36

3.1.4 Organization 40
3.2 Two Party Key Agreement 40
3.2.1 Diffie-Hellman Key Agreement 40
3.2.2 MTI Key Agreement 41
3.2.3 MQV Key Agreement 43
3.2.4 Jeong, Katz and Lee's Key Agreement 44
3.2.5 Smart's ID-Based Key Agreement 45
3.2.6 Scott's ID-Based Key Agreement 46
3.2.7 Chen and Kudla's ID-Based Key Agreement 47
3.2.8 McCullagh and Barreto's ID-Based Key Agreement 48
3.3 Three Party Key Agreement 49
3.3.1 Joux Key Agreement 49
3.3.2 Zhang, Liu and Kim's ID-Based Key Agreement 50
3.4 Multi Party Key Agreement 52
3.4.1 Ingemarsson, Tang and Wong's Group Key Agreement 52
3.4.2 Steiner, Tsudik and Waidner's Group Key Agreement 53
3.4.3 Burmester and Desmedt's Constant Round Group Key Agreement 55
3.4.4 Octopus Protocol and Cube Protocol 56
3.4.5 Boyd and Nieto's Constant Round Group Key Agreement 58
3.4.6 Bresson and Catalano's Constant Round Group Key Agreement 59
3.4.7 Bresson, Chevassut and Pointcheval's Dynamic Group Key Agreement 60
3.4.8 Bresson, Chevassut, Essiari and Pointcheval's Dynamic Group Key Agreement 63
3.4.9 Nam, Kim, Kim and Won's Dynamic Group Key Agreement 65
3.4.10 Kim, Lee and Lee's Dynamic Group Key Agreement 68
3.5 Conclusion 71

4 Tree-Based Group Key Agreement using Pairing 72
4.1 Introduction 72
4.2 Dynamic Group Key Agreement Protocol 75
4.2.1 Unauthenticated Key Agreement Protocol 76
4.2.2 Authenticated Key Agreement Protocol 80

4.2.3 Dynamic Key Agreement Protocol 86
4.3 Security Analysis 94
4.3.1 Security of the Unauthenticated Protocol 94
4.3.2 Security of the Authenticated (Static) Protocol 98
4.3.3 Security of the Dynamic Authenticated Protocol 102
4.4 Efficiency 104
4.5 Conclusion 107

5 Constant Round Dynamic Group Key Agreement 108
5.1 Introduction 108
5.2 Protocol 109
5.2.1 Unauthenticated Key Agreement Protocol 110
5.2.2 Authenticated Key Agreement Protocol 111
5.2.3 Dynamic Key Agreement Protocol 113
5.3 Security Analysis 117
5.3.1 Decision Diffie-Hellman (DDH) problem 117
5.3.2 Security of the Unauthenticated Protocol 118
5.3.3 Security of the Authenticated (Static) Protocol 122
5.3.4 Security of the Dynamic Authenticated Protocol 124
5.4 Efficiency 127
5.5 Conclusion 128

6 Concluding Remarks 130

A Overview of Pairing-Based Cryptographic Protocols 142
A.1 Introduction 142
A.2 Encryption Schemes 146
A.2.1 ID-Based Encryption 146
A.2.2 Searchable Public Key Encryption 147
A.2.3 Hierarchical ID-Based Encryption (HIDE) 148
A.2.4 Dual-HIDE : Dual-Hierarchical-Identity-Based Encryption 150
A.2.5 ID-Based Encryption Without Random Oracle 151

A.2.6 Hierarchical ID-Based Encryption (HIBE) Without Random Oracle 152
A.2.7 Another ID-Based Encryption Without Random Oracle 153
A.3 Signature Schemes 154
A.3.1 BLS Short Signature 154
A.3.2 Blind Signature 155
A.3.3 Multi signature 156
A.3.4 Aggregate Signature 157
A.3.5 Bilinear Verifiably Encrypted Signature 157
A.3.6 Bilinear Ring Signature 159
A.3.7 ZSS Short Signature 159
A.3.8 ID-Based Blind Signature (Schnorr type) 160
A.3.9 ID-Based Ring Signature 161
A.3.10 ID-Based Signature from Pairing 162
A.3.11 Unique Signature Without Random Oracle 163
A.3.12 Authentication-Tree Based Signature Without Random Oracle 164
A.3.13 Short Signature Without Random Oracle 166
A.4 Key Agreement Schemes 166
A.5 Threshold Schemes 167
A.5.1 Threshold Signature 167
A.5.2 Pairing Based (t, n)-Threshold Decryption 168
A.5.3 ID-based (t, n)-Threshold Decryption 169
A.6 Miscellaneous Applications 171
A.6.1 Key Sharing Scheme 171
A.6.2 ID-Based Chameleon Hashes from Bilinear Pairings 172
A.6.3 Signcryption Schemes 174
A.6.4 Identification Scheme based on GDH 176
A.6.5 Other Signature Schemes 177
A.7 Conclusion 177

Chapter 1

Introduction

The fundamental objective of cryptography is to enable two people to communicate over an insecure channel (for example, public telephone line or computer network) in such a way that a third party (called adversary) cannot understand what is being said, only the intended recipient can read the message.

The pioneering work of Diffie-Hellman [49] in 1976 broadly classified the modern cryptography into two complementary and completely distinct type:

1. Secret Key Cryptography,
2. Public Key Cryptography.

The secret key cryptography, also called symmetric key cryptography is essentially based on sharing a secret key between two people who want to communicate. This secret key is used both in the encryption process and also in the decryption process. The decryption process is simply the reverse of the encryption process. Since the encryption key and the decryption key are the same, the method is termed as symmetric. Secret key cryptography is extremely fast to implement, both in hardware and software. The most widely used method for secret key cryptography is the currently standardized AES [96]. AES can be programmed on a smart card without using up very much valuable memory. However, there are the following drawbacks of secret key cryptography that make it unsuitable for use in certain applications.

(a) *Key management problem* : A pair of users have to select a key in secret before commencing communication over an insecure channel. A (secure) secret channel for selecting a key may not be available. This is Key distribution problem. The number of keys increases quadratically with the number of users who want to communicate. In a network of n users, every pair of users must share a secret key, for a total of $n(n-1)/2$ keys. If n is large, then the number of keys becomes unmanageable.

(b) *No signature (without a trusted party) possible* : A digital signature scheme allows the receiver of a message to convince any third party that the message in fact originated from the sender.

This analogue of hand-written signature is not possible in a secret key cryptosystem because two party A and B have the same capabilities for encryption and decryption, and thus B can not convince a third party that a message he receives from A in fact originated from A.

To address above deficiencies in secret key cryptography, Diffie-Hellman invented public key cryptography in their seminal paper [49]. This celebrating work revolutionized the field of modern cryptography and had a profound effect on the direction of research in computational number theory. All known methods for public key cryptography are rather slow compared to private key cryptography. Public key cryptosystems are used as a complement to private key cryptosystems, either for signature or authentication or for key exchange.

Public key cryptosystems, also called asymmetric cryptosystems, derive their security from some computationally hard mathematical problems. The most popular ones are based mainly over two such problems:

1. Integer factorization problem,
2. Discrete logarithm problem (DLP).

Integer factorization problem, in rough terms, states that it is very easy to create at will quite large prime numbers (for instance 1024 bit is a typical cryptographic size for reasonable security), but it is totally impossible (at the current state of art), except by some incredible stroke of luck (or a bad choice of the prime) to factor the product of two primes of this size (a 2048-bit number).

The first usable public key cryptosystem, introduced by Rivest, Shamir and Adleman in 1978, was the RSA cryptosystem [88]. The RSA cryptosystem is based on the hardness of integer factoring problem.

Another type of public key cryptosystem is based on discrete logarithm problem (DLP) in cyclic groups of prime order. In short, if G is a large cyclic group with a generator g, then given any element $y = g^a$ in the group, computing a is called the DLP over G. The Diffie-Hellman key exchange protocol derives its security from the hardness of DLP. Later, several variants of DLP has been proposed (for example Computational Diffie-Hellman (CDH) problem, Decision Diffie-Hellman (DDH) problem). Groups on which one of these problems is computationally hard is used for implementing various cryptographic protocols.

Since the discovery of Diffie-Hellman [49], many groups have been identified on which the DLP is believed to be hard. These include the multiplicative group of a finite field of characteristic 2, the group of units of Z_n where n is a composite integer, the multiplicative subgroup of the prime field Z_p, the group of points on an elliptic curve defined over a finite field, the Jacobian of an hyperelliptic curve over a finite field. Elliptic curve cryptosystem is based on the DLP in cyclic groups of prime order that are embedded in elliptic curves defined over finite fields. Cryptosystems using DLP in this group have two potential advantages over systems based on the multiplicative group of finite field (and also over systems based on RSA):

1. the great diversity of elliptic curves available to provide the groups; and
2. for suitably chosen security parameters (like the curve, underlying field, the group order *etc.*), there is no known subexponential algorithm (such as those of "index calculus" type) to solve the DLP.

This leads to the fact that a high level of security can be achieved in elliptic curve cryptosystem over a group of computationally much smaller order. Having groups of smaller order means shorter key lengths which in turn leads to faster communication, bandwidth saving and smaller chip size for a hardware design. Thus elliptic curve cryptography can be a crucial factor in some applications with restricted environments, such as the design of smart cards and other handhold and mobile devices with small silicon space.

Typically, to have the same security as a 2048-bit RSA, one estimates that an elliptic curve (or the Jacobian of a curve of small genus) over a finite field should have number of points equal to a small multiple of a prime of 224 or perhaps 250 bits. Even though the basic operations on elliptic curves and on Jacobians are more complicated than in $(Z/nZ)^*$, the small key size largely compensates for this. The elliptic and hyperelliptic curve cryptography and their secure and efficient implementation have been receiving considerable attention in the research community.

Earlier bilinear pairings, namely Weil pairing and Tate pairing of algebraic curves were used in cryptography to reduce the discrete logarithm problem on some elliptic or hyperelliptic curves to the discrete logarithm problem in a finite field. In recent years, bilinear pairings have found positive application in cryptography to construct new cryptographic primitives mainly because the following results.

(a) *Key Agreement* : Key agreement is required in situations where two or more parties want to communicate securely among themselves. The situation where three or more parties share a secret key is often called conference keying. In this situation, the parties can securely send and receive message from each other. An adversary not having access to the secret key will not be able to decrypt the message. Joux [66] in 2000 showed that the Weil pairing can be used for "good" by using it in a protocol to construct three-party one-round Diffie-Hellman key agreement. This was one of the breakthroughs in key agreement protocols.

(b) *ID-Based Cryptosystem* : The concept of identity-based cryptosystem is due to Shamir [105]. Such a scheme has the property that a user's public key is an easily calculated function of his identity, while a user's private key can be calculated for him by a trusted authority, called private key generator (PKG). The ID-based public key cryptosystem can be an alternative for certificate-based public key infrastructure (PKI), especially when efficient key management and moderate security are required. Since the problem was proposed in 1984, no satisfactory scheme was obtained until Boneh and Franklin [24] in 2001 proposed a feasible functional ID-based encryption using Weil pairing.

• **Objective of the Thesis:**

The main goal of the thesis is to obtain efficient and provably secure group key agreement protocols and their dynamic versions.

• **Our Contribution in the Thesis:**

The thesis is based on our following seven articles.

1. R. Barua, R. Dutta, P. Sarkar.
   Extending Joux Protocol to Multi Party Key Agreement.
   In proceedings of Indocrypt 2003, LNCS 2904, pp. 205-217, Springer-Verlag, 2003. Also available at http://eprint.iacr.org/2003/062.

2. R. Dutta, R. Barua and P. Sarkar.
   Pairing Based Cryptographic Protocols : A Survey.
   Manuscript 2004. Available at http://eprint.iacr.org/2004/064.

3. R. Dutta, R. Barua and P. Sarkar.
   Authenticated Multi-party Key Agreement : A Provably Secure Tree Based Scheme using Pairing.
   In proceedings of National Workshop on Cryptology 2004, Kerala, India, October 2004.

4. R. Dutta, R. Barua and P. Sarkar.
   Provably Secure Authenticated Tree Based Group Key Agreement.
   In proceedings of ICICS 2004, LNCS 3269, pp. 92-104, Springer-Verlag,2004. Also available at http://eprint.iacr.org/2004/090.

5. R. Dutta and R. Barua.
   Dynamic Group Key Agreement in Tree-based Setting.
   In proceedings of ACISP 2005, LNCS 3574, pp. 101-112, Springer-Verlag, 2005. Also available at http://eprint.iacr.org/2005/131.

6. R. Dutta and R. Barua.
   Constant Round Dynamic Group Key Agreement.
   In proceedings of ISC 2005, LNCS 3650, pp. 74-88, Springer-Verlag, 2005. Also available at http://eprint.iacr.org/2005/221.

7. R. Dutta and R. Barua.
   Overview of Key Agreement Protocols.
   Manuscript 2005. Available at http://eprint.iacr.org/2005/289.

We provide a brief overview of previous works on key agreement covering most of the major works in this area. We include this survey in the thesis to put our contribution in the right

perspective with respect to other group key agreement protocols. The first part of the survey looks into the chronological development of the protocols, followed by description of protocols arranged according to the following three categories:

1. Two-party key agreements
2. Three-party key agreements
3. Multi-party key agreements

We construct a new pairing-based unauthenticated group key agreement protocol that extends Joux [66] tripartite single round key agreement protocol to multi-party setting using a ternary tree structure. Achieving security of the scheme even against passive adversary is not so trivial. We provide a rigorous security analysis of our scheme against passive adversary under the assumption that Decisional Hash Bilinear Diffie-Hellman (DHBDH) problem is hard.

Then we achieve authentication of our tree-based protocol by modifying the Katz-Yung [70] technique suitable for our tree-based protocol. The main component of Katz-Yung protocol is the use of a digital signature scheme. However, in keeping with our tree-structure we require an aggregate signature scheme or a multisignature scheme. In each round, each user sends the signed message to his/her representative. The representative combines these signatures and sends the aggregate signature or multisignature to the other users. This process is repeated until the last round. Moreover, a similar construction can be used to modify any tree-based unauthenticated group key agreement to an authenticated group key agreement protocol. We provide a concrete security analysis of our authenticated protocol against active adversary in the security framework formalized by Bresson *et al.* [32].

Next we extend our tree-based group key agreement protocol into dynamic scenario where a user may join or leave the group at any time. We provide a proof of security of the scheme in the security model of Bresson *et al.* [32]. Our protocol is designed to ensure a minimum modification to the computation already precomputed when a user leaves or joins. This is because we need to maintain the structure of the tree and a person may leave/join disturbing this structure. So one needs to "relocate" using minimal computation. Retaining the tree structure is an important issue of our protocol while such dynamic operations (join or leave) are concerned.

We also present and analyse a simple and elegant constant round group key agreement protocol and enhance it to the dynamic scenario where a group of users can join or leave at any time during protocol execution with updated keys. The unauthenticated protocol is a variant of the Burmester-Desmedt [36, 70] (BD) group key agreement. It differs from the BD protocol in two respects. It has a provision for checking whether all users honestly follow the protocol – any deviation from the protocol would be detected by each participant. Secondly, our computation of the session key is simple using lesser number of operations. Also this protocol is dynamic. As in the tree-based case, we provide a concrete security analysis of our protocol under the assumption that Decisional Diffie-Hellman (DDH) problem is hard. The proof is in the security model of Bresson *et al.* [32] adapting Katz-Yung [70] technique. However, unlike Katz-Yung compiler, we have done away with

random nonces and thereby able to reduce the number of rounds by one – a gain in communication complexity. The reduction proof remains as tight as in Katz-Yung compiler except for a small additional (negligible) term.

We have made an attempt to survey on pairing-based cryptographic protocols. Our original goal was to obtain some new pairing-based protocols. So we have included the survey in this thesis. The main purpose of this survey is not only to show the importance of pairing in modern day cryptography, but also to present the variety of cryptographic primitives that can be defined using pairing. We cover in this survey the basic pairing-based protocols that have proper security proofs and will continue to be referred in the future. We left out some other protocols due to the non-availability of proper security proofs. We have also not provided any proofs of important protocols like the Boneh-Franklin ID-based encryption scheme, mainly due to lack of space. Although, this survey is not exhaustive and up-to-date, we have included it in the hope that it will serve as a motivation to a beginner. In fact, this has been used as a reading material by Prof. Ron Rivest of Massachusetts Institute of Technologies (MIT) in his course on "Selected Topics In Cryptography (6.897)".

The subsequent section includes organization of the thesis.

1.1 Synopsis

Our contribution, as mentioned earlier, is in designing efficient and secure group key agreement protocols. This work can be broadly classified into two parts. The first part (Chapter 4) consists of designing a group key agreement protocol using bilinear pairing. The second part of the thesis (Chapter 5) is devoted to design constant round group key agreement protocol. Here is a brief summery of the chapters.

Chapter 1 is a fairly elementary introduction to private and public key cryptography. The material of the section is intended to serve as a motivation for the rest of the thesis.

In Chapter 2, we cover a summary of the relevant theory of elliptic curves over finite fields and the necessary preliminary material required in the later chapters. Recently, bilinear pairings such as Weil pairing or Tate pairing on elliptic and hyperelliptic curves have been found applications in design of cryptographic protocols. We first study in great detail the Elliptic Curve Cryptography (ECC) and pairings for a better understanding of the subject from mathematical point of view. We provide a brief description of group operations, group structure, group order of ECC. We also introduce bilinear pairing (*cf.* Weil pairing). One section of this chapter is devoted in describing several variants of Diffie-Hellman problems while another section discusses the security notions and security models of certain basic cryptographic primitives.

Chapter 3 is devoted to realization of key agreement. An overview of key agreement protocols is presented in this chapter. We have included a comprehensive treatment of describing the most important key agreement protocols and provided an account of chronological developments connected with key agreement.

In Chapter 4, we are mainly interested in designing a pairing-based protocol. We design a dynamic group key agreement in tree-based setting that uses a different arrangement of participants. The protocol has an in-built hierarchical structure that makes it desirable for certain applications. We prove that the scheme is provably secure under the assumption that DHBDH problem is hard. The bilinear pairing based Joux [66] protocol and multi-signature by Boldyreva [18] are used. The protocol is proven to be secure in the security model formalized by Bresson *et al.* [32] by appropriately modifying the Katz-Yung [70] technique to tree-based setting. One can be benefited in both communication and computation power by combining our protocol with an efficient constant round protocol. Designing such hybrid group key agreement protocols may be desirable for certain applications, in particular when large number of users are concerned.

We present and analyze in Chapter 5 a simple and elegant constant round group key agreement protocol and enhance it to dynamic setting where a set of users can leave or join the group at any time during protocol execution with updated keys. This scheme may be viewed as a variant of Burmester-Desmedt [36, 70] group key agreement protocol (BD) with considerably better efficiency and flexibility. In contrast to BD protocol, let us refer to our protocol as DB protocol. Although the DB protocol is similar to BD protocol, there are subtle differences between them.

1. Key computation in DB protocol is different and is more efficiently done than in BD protocol.
2. Number of rounds, point-to-point communication, signature verifications required in DB protocol are less as compared to BD protocol and number of modular multiplications reduces from $O(n^2)$ to $O(n)$ with the same number of modular exponentiations.
3. DB protocol is more flexible than BD protocol in the sense that DB protocol is dynamic.
4. DB protocol has the ability to detect the presence of corrupted group members, although one can not detect who among the group members are behaving improperly.

The emphasis of this work is to achieve provable security of the scheme DB under DDH assumption. We provide a concrete security analysis of this protocol against active adversary in the security model of Bresson *et al.* [32] adapting Katz-Yung [70] technique. The protocol is forward secure, efficient and fully symmetric.

Finally, we conclude in Chapter 6.

We have attempted to provide a brief survey on pairing-based cryptographic protocols in the Appendix. This is just to show the depth and breadth of the applications of bilinear maps to cryptography. We have included this material in our thesis in the hope that this will provide others a ready reference to applications of bilinear maps. We discuss in this survey several cryptographic primitives using pairings. Some other primitives have been left out, mainly due to the non-availability of proper security proofs. The area is still growing and almost each conference proceedings include some new proposals. On the other hand, we have covered the basic schemes which will continue to be referred in the future. Thus we believe that our survey will provide both an introduction to the area as well as serve as a ready reference to the area in the next few years.

Chapter 2

Preliminary Background and Definitions

2.1 Introduction

One of the main goal of this thesis is to make some contribution in designing cryptographic protocols based on bilinear pairings. Since the only known bilinear maps are the Weil and Tate pairings, we first include some mathematical background on elliptic curves. We make no attempt to be complete in the presentation. For an elementary introduction to elliptic curves, we recommend Koblitz's book [77] and the notes by Charlap and Robbins [39]. The proofs of the results stated in this chapter can be found in the book by J. Silverman [110]. We then concentrate on specifying several versions of Diffie-Hellman problems. Security of various protocols included in this thesis are based on the hardness of these problems. This chapter also deals with the security notions and security models for different cryptographic primitives.

2.2 Elementary Concepts on Elliptic Curves

Let K be a field and $\overline{K}$ its algebraic closure. An elliptic curve over K is defined by a Weierstrass equation

$$E/K : y^2 + a_1xy + a_3y \quad = \quad x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in K$ and there are no "singular points" (singular points for a curve $f(x, y) = 0$ are those points where both the partial derivatives of f vanish.) If $L \supset K$, then the set of L-rational points on E is

$$E(L) = \{(x, y) \in L \times L : y^2 + a_1xy + a_3y \quad = \quad x^3 + a_2x^2 + a_4x + a_6\} \cup \{\mathcal{O}\}.$$

Here $\mathcal{O}$ is an identified element, called point at infinity. If $L \supset K$, then $E(L) \supset E(K)$. We denote $E(\overline{K})$ by E. Simplified Weierstrass equation is as follows.

Case 1: If $\text{char}(K) \neq 2, 3$, then the equation simplifies to $y^2 = x^3 + ax + b$, $a, b \in K$ and $4a^3 + 27b^2 \neq 0$.

Case 2: If $\text{char}(K) = 2$, then the equation simplifies to

$$y^2 + xy = x^3 + ax^2 + b,\ a, b \in K,\ b \neq 0, \text{ non-supersingular, or}$$
$$y^2 + cy = x^3 + ax + b,\ a, b, c \in K,\ c \neq 0, \text{ supersingular.}$$

For any $L \supset K$, the set $E(L)$ is an abelian group under the "chord-and-tangent law" [77] explained below: If $P \neq \mathcal{O}$, $Q \neq \mathcal{O}$, $Q \neq -P$, then $P + Q = -R$, where R is the third point of intersection of the line PQ (or tangent PQ in case $P = Q$) with the curve E.

Consider $E/K : y^2 = x^3 + ax + b$. Addition formulae are as follows:

1. $P + \mathcal{O} = \mathcal{O} + P = P$, for all $P \in E(L)$.
2. $-\mathcal{O} = \mathcal{O}$.
3. If $P = (x, y) \in E(L)$, then $-P = (x, -y)$.
4. If $Q = -P$, then $P + Q = \mathcal{O}$.
5. If $P = (x_1, y_1) \in E(L)$, $Q = (x_2, y_2) \in E(L)$, $P \neq -Q$, then $P + Q = (x_3, y_3)$, where $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, and

$$\begin{aligned} \lambda &= \frac{y_2 - y_1}{x_2 - x_1} \quad \text{if } P \neq Q; \\ \lambda &= \frac{3x_1^2 + a}{2y_1} \quad \text{if } P = Q. \end{aligned}$$

Note that if $P \neq \mathcal{O}$, $Q \neq \mathcal{O}$, $Q = -P$, then $P + Q = \mathcal{O}$, *i.e.* $\mathcal{O}$ is the third point of intersection of any vertical line through P (or Q) with the curve E. Any vertical line through P (or Q) meets the curve E at infinity. This is why $\mathcal{O}$ is called point at infinity. $\mathcal{O}$ serves as the identity of the abelian group $E(L)$.

For the purpose of cryptography, assume henceforth that $K = \mathbb{F}_q$, *i.e.* the finite field of characteristic p and of order q and $\overline{K} = \cup_{m \geq 1} \mathbb{F}_{q^m}$. The following are three important results on the group order of elliptic curve groups.

Theorem 2.2.1 *(Hasse's Theorem)* $\#E(\mathbb{F}_q) = q + 1 - t$, $|t| \leq 2\sqrt{q}$. *Consequently,* $\#E(\mathbb{F}_q) \approx q$.

Theorem 2.2.2 *(Schoof's Algorithm)* $\#E(\mathbb{F}_q)$ *can be computed in polynomial time.*

Theorem 2.2.3 *(Weil Theorem) Let* $t = q+1-\#E(\mathbb{F}_q)$. *Let* α, β *be complex roots of* $T^2-tT+q \in Z[T]$ *(where* $Z[T]$ *is the ring of polynomials in* T *with integer coefficients). Then* $\#E(\mathbb{F}_{q^k}) = q^k + 1 - \alpha^k - \beta^k$ *for all* $k \geq 1$.

The structure of elliptic curve groups is summarized by the following results.

– Let E be an elliptic curve defined over $\mathbb{F}_q$. Then $E(\mathbb{F}_q) \cong Z_{n_1} \oplus Z_{n_2}$, where $n_2|n_1$ and $n_2|(q-1)$.

– $E(\mathbb{F}_q)$ is cyclic if and only if $n_2 = 1$.

– $P \in E$ is an n-torsion point if $nP = \mathcal{O}$ and $E[n]$ is the set of all n-torsion points.

– If $\gcd(n, q) = 1$, then $E[n] \cong Z_n \oplus Z_n$.

2.2.1 Supersingular Elliptic Curves

An elliptic curve $E/\mathbb{F}_q$ is supersingular if $p|t$ where $t = q + 1 - \#E(\mathbb{F}_q)$.

Theorem 2.2.4 *(Waterhouse)* $E/\mathbb{F}_q$ *is supersingular if and only if* $t^2 = 0, q, 2q, 3q$ *or* $4q$. *The group structure is given by the following result.*

Theorem 2.2.5 *(Schoof) Let* $E/\mathbb{F}_q$ *be supersingular with* $t = q + 1 - \#E(\mathbb{F}_q)$. *Then*

1. *If* $t^2 = q, 2q$ *or* $3q$, *then* $E(\mathbb{F}_q)$ *is cyclic.*
2. *If* $t^2 = 4q$ *and* $t = 2\sqrt{q}$, *then* $E(\mathbb{F}_q) \cong Z_{\sqrt{q}-1} \oplus Z_{\sqrt{q}-1}$.
3. *If* $t^2 = 4q$ *and* $t = -2\sqrt{q}$, *then* $E(\mathbb{F}_q) \cong Z_{\sqrt{q}+1} \oplus Z_{\sqrt{q}+1}$.
4. *If* $t = 0$ *and* $q \not\equiv 3 \bmod 4$, *then* $E(\mathbb{F}_q)$ *is cyclic.*
5. *If* $t = 0$ *and* $q \equiv 3 \bmod 4$, *then* $E(\mathbb{F}_q)$ *is cyclic or* $E(\mathbb{F}_q) \cong Z_{\frac{q+1}{2}} \oplus Z_2$.

2.2.2 Divisors and Weil Pairing

Let E be an elliptic curve defined over a field $\mathbb{F}_q$ and given by the equation $C(x, y) = 0$, where $C(x, y)$ is the polynomial defining E. We consider the set of $\mathbb{F}_{q^n}$-rational points of E. The group of divisors of $E(\mathbb{F}_{q^n})$ is the free abelian group generated by the points of $E(\mathbb{F}_{q^n})$. Thus any divisor D is of the form

$$D = \sum_{P \in E(\mathbb{F}_{q^n})} n_P \langle P \rangle.$$

where $n_P \in Z$ and $n_P = 0$ except for finitely many P's. The support of a divisor D is the set of points $\{P \in E | n_P \neq 0\}$. We will only consider *zero divisors*, i.e., divisors where $\sum n_P = 0$.

A rational function f on E is an element of the field of fractions of the ring $\mathbb{F}_{q^n}[x,y]/(C(x,y))$. If $P=(x,y)$, then by $f(P)$ we mean $f(x,y)$. The divisor of a rational function f is defined by

$$\mathrm{div}(f) \quad = \quad \sum_{P \in E(\mathbb{F}_{q^n})} \mathrm{ord}_P(f)\langle P \rangle$$

where $\mathrm{ord}_P(f)$ is the order of the zero/pole that f has at P. A divisor D is said to be *principal* if $D = \mathrm{div}(f)$, for a rational function f.

Theorem 2.2.6 *A divisor $D = \sum_{P \in E(\mathbb{F}_{q^n})} n_P \langle P \rangle$ is principal if and only if*

1. $\sum n_P = 0$ *and*
2. $\sum n_P P = \mathcal{O}$.

Two divisors D_1 and D_2 are said to be *equivalent* ($D_1 \sim D_2$) if $D_1 - D_2$ is principal.

Theorem 2.2.7 *Any zero divisor $D = \sum n_P \langle P \rangle$ is equivalent to a (unique) divisor of the form $\langle Q \rangle - \langle \mathcal{O} \rangle$ for some $Q \in E(\mathbb{F}_{q^n})$.*

Given a rational function f and a zero divisor $D = \sum n_P \langle P \rangle$, define

$$f(D) \quad = \quad \prod_{P \in E(\mathbb{F}_{q^n})} f(P)^{n_P}.$$

Now we define the important notion of *Weil pairing.*

Definition 2.2.8 *Let $P, Q \in E[n]$, the subset of all n-torsion points of the $\mathbb{F}_q$-rational points of an elliptic curve E defined over $\mathbb{F}_q$. Let D_P be a divisor which is equivalent to $\langle P \rangle - \langle \mathcal{O} \rangle$. Then nD_P is principal and hence there exists a rational function f_P such that $div(f_P) = nD_P$. Define D_Q and f_Q analogously. Let D_P and D_Q have disjoint support. Then* Weil pairing *$\widehat{e}(P,Q)$ is defined as*

$$\widehat{e}(P,Q) \quad = \quad \frac{f_P(D_Q)}{f_Q(D_P)}.$$

(One can define another pairing function, called *Tate pairing* [8, 59], which is very similar to the Weil pairing).

2.2.3 Weil Pairing and Bilinear Map

Let E be an elliptic curve defined over $\mathbb{F}_q$. Let n be an integer with $\gcd(n,q)=1$ and $\mathbb{F}_{q^k}$ be the smallest extension of $\mathbb{F}_q$ such that $E[n] \subseteq E(\mathbb{F}_{q^k})$. (This implies that $n^2 | \#E(\mathbb{F}_{q^k})$ and $n | (q^k - 1)$.) Let μ_n be the subgroup of order n in $\mathbb{F}_{q^k}^*$. Then Weil pairing $\widehat{e}$ is a map $\widehat{e} : E[n] \times E[n] \rightarrow \mu_n$ with the following properties.

1. *(Bilinearity)* : For all $R, S, T \in E[n]$,
   (a) $\widehat{e}(S+R,T) = \widehat{e}(S,T) \cdot \widehat{e}(R,T)$,
   (b) $\widehat{e}(S,T+R) = \widehat{e}(S,T) \cdot \widehat{e}(S,R)$.
2. *(Non-degeneracy)* : Let $S \in E[n]$. If $\widehat{e}(S,T) = 1$ for all $T \in E[n]$, then $S = \mathcal{O}$.
3. *(Computability)* : $\widehat{e}$ can be computed efficiently.
4. *(Identity)* : $\widehat{e}(S,S) = 1$ for all $S \in E[n]$.
5. *(Alternation)* : $\widehat{e}(S,T) = \widehat{e}(T,S)^{-1}$.

The property of bilinearity defined above implies the following: $\widehat{e}(aS,bT) = \widehat{e}(S,T)^{ab}$ for all $S, T \in G_1$ and $a, b \in Z$.

Lemma 2.2.9 *Let $P \in E[n]$ have order n. Then $P_1, P_2 \in E[n]$ are in the same coset of $\langle P \rangle$ within $E[n]$ if and only if $\widehat{e}(P,P_1) = \widehat{e}(P,P_2)$.*

Theorem 2.2.10 *Let $P \in E[n]$ have order n. Let $R \in E[n]$ be such that $\widehat{e}(P,R)$ has order n. Then $f : \langle P \rangle \to \mu_n$ defined by $f(Q) = \widehat{e}(Q,R)$ is a group isomorphism.*

As a consequence of the above, we have the following important fact.

$$E[n]/\langle P \rangle \cong Z_n \cong \mu_n.$$

We next define bilinear pairing or bilinear map.

Definition 2.2.11 *Let G_1, G_2 be two groups of the some large prime order q. We view G_1 as an additive group and G_2 as a multiplicative group. Let P be an arbitrary generator of G_1. (aP* denotes P added to itself a times*). Assume that discrete logarithm problem (DLP) is hard in both G_1 and G_2. A mapping $e : G_1^2 \to G_2$ satisfying the following properties is called a* cryptographic bilinear map.

(Bilinearity) : $e(aP,bQ) = e(P,Q)^{ab}$ for all $P, Q \in G_1$ and $a, b \in Z_q^*$.
(Non-degeneracy): $e(P,P) \neq 1$. i.e. if P is a generator of G_1, then $e(P,P)$ is a generator of G_2.
(Computability) : There exists an efficient algorithm to compute $e(P,Q)$ for all $P, Q \in G_1$.

Modified Weil Pairing [24] and Tate Pairing [8], [59] are examples of cryptographic bilinear maps. Currently, active research is being carried out to obtain efficient algorithms to compute pairings. Our thesis excludes this area.

Following are some important properties of bilinear pairings.

1. $e(S, \mathcal{O}) = 1$ and $e(\mathcal{O}, S) = 1$ for all $S \in G_1$.
2. $e(S, -T) = e(-S, T) = e(S, T)^{-1}$ for all $S, T \in G_1$.
3. $e(S, T) = e(T, S)$ for all $S, T \in G_1$.
4. Let $S \in G_1$. If $e(S, T) = 1$ for all $T \in G_1$, then $S = \mathcal{O}$.

Note that for a bilinear pairing e, $e(P, P) \neq 1$ for all $P \in G_1$, but for Weil pairing $\widehat{e}$, $\widehat{e}(P, P) = 1$ for all $P \in E[n]$. One can use Weil pairing to check the linear independency of two points as follows:
If two elliptic curve points P_1, P_2 are linearly dependent, then $P_2 = aP_1$ for some $a \in \mathbb{F}_q$. Consequently, we have

$$\widehat{e}(P_1, P_2) = \widehat{e}(P_1, cP_1) = \widehat{e}(P_1, P_1)^a = 1.$$

For two linearly independent points P_1, P_2, $\widehat{e}(P_1, P_2) \neq 1$.

Let P be a point of order n on an elliptic curve $E/\mathbb{F}_q$. Then one can always find a point Q of order n on an elliptic curve $E/\mathbb{F}_{q^k}$ for some $k > 1$ such that P, Q are linearly independent. *i.e.* $\widehat{e}(P, Q) \neq 1$. For supersingular elliptic curve, this is done by means of a distortion map ϕ which is an automorphism on $E/\mathbb{F}_{q^k}$ (*i.e.* $P, \phi(P)$ are linearly independent).

Example [24]: We now illustrate by an example how bilinear map can be derived from Weil pairing. We fix a supersingular curve E over $\mathbb{F}_p$, $p > 3$ and $p \equiv 2 \bmod 3$, given by $y^2 = x^3 + 1$. $E(\mathbb{F}_p)$ contains $p + 1$ points. Let $P \in E(\mathbb{F}_p)$ be a point of order n where $n|(p + 1)$. Since $\gcd(n, p) = 1$, the set of all n-torsion points $E[n]$ is isomorphic to Z_{n^2}. The field $\mathbb{F}_{p^2}$ is the smallest extension of the field $\mathbb{F}_p$ such that $E[n]$ is contained in $E(\mathbb{F}_{p^2})$ and the set $E(\mathbb{F}_{p^2})$ contains a point Q of order n which is linearly independent of the points in $E(\mathbb{F}_p)$. Let G_1 be the subgroup of points generated by P. Let $\zeta \in \mathbb{F}_{p^2}$ be a non-trivial root of $x^3 + 1 \equiv 0 \bmod p$. Then $\phi(x, y) = (\zeta x, y)$ is an automorphism on $E(\mathbb{F}_{p^2})$. The map ϕ is called a distortion map.
Note that $E[n]$ is the group generated by P and $\phi(P)$ which are linearly independent and both are of order n.

Let G_2 be a subgroup of $\mathbb{F}_{p^2}^*$ of order n and $\widehat{e} : E[n] \times E[n] \to G_2$ be the Weil pairing map. Then the *modified Weil pairing* $e : G_1 \times G_1 \to G_2$ is defined by

$$e(P, Q) \quad = \quad \widehat{e}(P, \phi(Q)).$$

It is not hard to check that e is a cryptographic bilinear pairing. (e is bilinear and computable, also note that $e(P, P) = \widehat{e}(P, \phi(P)) \neq 1$ as P, $\phi(P)$ are linearly independent.)

Henceforth, we take G_1, G_2, e as defined in Definition 2.2.11 for the rest of the thesis unless mentioned otherwise. This unification of parameters is required for a smooth and unambiguous presentation of the thesis.

2.2.4 Notations and Terminologies

For a set S, we use the notation $a \in_R S$ or $a \longleftarrow S$ to mean that a is randomly chosen from S and the notation $|$ to denote concatenation of data items (*cf.* $A|B|C$). Unless otherwise stated, we assume that the messages are arbitrary length finite binary strings and the above setup holds for the cryptographic protocols throughout the presentation. We define a function $f(m)$ to be *negligible* if it is less than $\frac{1}{m^l}$ for every fixed $l > 0$ and sufficiently large integer m.

In the subsequent discussion, we specify some versions of Diffie-Hellman problems. Each problem comes in two flavors : computational followed by decisional. We define the following two terms.

- advantage : When an algorithm has to distinguish between two probability distribution.

- success probability : When an algorithm has to find an object of interest.

We formalize advantage of DDH and success probability of CDH problems and describe the corresponding assumptions. For each of the other problems, there is a corresponding assumption which can be formalized in a way similar to the DDH and CDH problems.

We say that a problem is easy if there exists an efficient algorithm (polynomial time in the size of input) to solve the problem. On the contrary, a problem is hard if there is no polynomial time algorithm (in the size of input) to solve it. Sometimes, to solve a problem, an algorithm may have access to oracles which are considered as black boxes. The algorithm may make queries to the oracles at any time and will receive the corresponding responses.

We provide the following classification of the problems.

2.3 Diffie-Hellman Problems

1. Computational Diffie-Hellman (CDH) problem in G_1 :
   Instance : (P, aP, bP) for some $a, b \in Z_q^*$.
   Output : abP.

   The success probability of any probabilistic, polynomial-time algorithm $\mathcal{A}$ in solving CDH problem in G_1 is defined to be :

   $$\mathsf{Succ}^{\mathsf{CDH}}_{\mathcal{A},G_1} = \mathsf{Prob}[\mathcal{A}(P, aP, bP) = abP : a, b \in_R Z_q^*].$$

   CDH assumption : For every probabilistic, polynomial-time algorithm $\mathcal{A}$, $\mathsf{Succ}^{\mathsf{CDH}}_{\mathcal{A},G_1}$ is negligible.
   (See sections 3.3.1, 3.3.2, 3.6.2, 3.6.4, A.3.9, A.3.12).

2. Decisional Diffie-Hellman (DDH) problem in G_1 :
   Instance : (P, aP, bP, cP) for some $a, b, c \in Z_q^*$.

Output : yes if $c = ab \bmod q$ and output no otherwise.
Comments : DDH problem in G_1 is easy. DDH problem in G_1 can be solved in polynomial time by verifying $e(aP, bP) = e(P, cP)$. This is the well known MOV reduction [24] : The DLP in G_1 is no harder than the DLP in G_2.

The advantage of any probabilistic, polynomial-time, 0/1-valued algorithm $\mathcal{A}$ in solving DDH problem in G_1 is defined to be :

$$\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},G_1} = |\mathsf{Prob}[\mathcal{A}(P, aP, bP, cP) = 1] - \mathsf{Prob}[\mathcal{A}(P, aP, bP, abP) = 1] : a, b, c \in_R Z_q^*|.$$

DDH assumption : For every probabilistic, polynomial-time, 0/1-valued algorithm $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{A},G_2}$ is negligible.
(See sections 3.2.4, 3.5.1, 3.5.2, 3.5.3, 3.5.4, 3.5.6, Chapter 5).

Gap Diffie-Hellman (GDH) group : A prime order group G_1 is a GDH group if there exists an efficient polynomial-time algorithm which solves the DDH problem in G_1 and there is no probabilistic polynomial-time algorithm which solves the CDH problem with non-negligible probability of success. The domains of bilinear pairings provide examples of GDH groups. The MOV reduction essentially introduces a method to solve DDH in G_1, whereas there is no known efficient algorithm for solving CDH in G_1, in general.
(See sections A.3.1, A.3.3, A.3.11, A.5.1, A.6.4).

3. Weak Diffie-Hellman (W-DH) problem in a group G_1 :
Instance : (P, Q, sP) for $P, Q \in G_1$ and for some $s \in Z_q^*$.
Output : sQ.
Comments : W-DH problem is equivalent to CDH problem.
(See sections 3.4.2, A.3.10).

4. Reversion of CDH (RCDH) problem in G_1 :
Instance : (P, aP, rP) for some $a, r \in Z_q^*$.
Output : $bP, b \in Z_q^*$ satisfying $a = rb \bmod q$.
Comments : RCDH problem is equivalent to CDH problem in G_1 [44].

5. $(k+1)$-exponent problem ($(k+1)$-EP) in G_1:
Instance : $(P, yP, y^2P, \ldots, y^kP)$for a random $y \in Z_q^*$.
Output : $y^{k+1}P$.
Comments : $(k+1)$-EP is no harder than CDH problem.
(See section A.3.7).

6. k-Diffie-Hellman Inversion (k-DHI) problem in G_1 :
Instance : $(P, yP, y^2P, \ldots, y^kP)$ for a random $y \in Z_q^*$.
Output : $\frac{1}{y}P$.
Comments : k-DHI problem is polynomially equivalent to $(k+1)$-EP.

7. k-Strong Diffie-Hellman (k-SDH) problem in G_1 :
   Instance : $(P, yP, y^2P, \ldots, y^kP)$ for a random $y \in Z_q^*$.
   Output : $(c, \frac{1}{y+c}P)$ where $c \in Z_q^*$.
   Comments : k-SDH problem is a stronger version of k-DHI problem. When c is pre-specified, k-SDH problem is polynomially equivalent to k-DHI. k-SDH problem has a simple random self reduction in G_1.
   (See section A.3.13).

8. Collusion Attack Algorithm with k-traitors (k-CAA) :
   Instance : $(P, yP, h_1, \ldots, h_k \in Z_q^*, \frac{1}{h_1+y}P, \ldots, \frac{1}{h_k+y}P)$ for a random $y \in Z_q^*$.
   Output : $\frac{1}{h+y}P$ for some $h \notin \{h_1, \ldots, h_k\}$.
   Comments : k-CAA is polynomially equivalent to $(k-1)$-DHI problem which in turn is polynomially equivalent to k-EP.

9. l- Many Diffie-Hellman problem in G_1:
   Oracle : $\mathcal{O}_{P,\tilde{y}}(J) = (\prod_{j \in J} y_j)P \in G_1$ where vector $\tilde{y} = (y_1, y_2, \ldots, y_l) \in_R (Z_q^*)^l$ and J is any proper subset of $\{1, 2, \ldots, l\}$.
   Instance : $(P, \mathcal{O}_{P,\tilde{y}}, J)$ for any vector $\tilde{y} = (y_1, y_2, \ldots, y_l) \in_R (Z_q^*)^l$ and for all $J \subset \{1, 2, \ldots, l\}$.
   Output: $(\prod_{j=1}^{l} y_j)P$.
   Comments : $(l-1)$-DHI assumption implies l-Many-DH assumption. This reduction is also valid for the decision version of DHI and Many-DH problems. l-DHI assumption is easier to state than l-Many-DH assumption since there is no need for an oracle. Many Diffie-Hellman problem is sometimes referred to as Generalized Computational Diffie-Hellman (GCDH) problem.
   (See sections 3.6.1, A.3.11).

10. Chosen-target CDH problem in G_1 :
    Let s be a random element of Z_q^* and $Q = sP$.
    Oracles : (a) A target oracle $\mathcal{T}_{G_1}$ that returns a random element $U_i \in G_1$. (b) A helper oracle $s(\cdot)$ that returns sU on a randomly chosen input $U \in G_1$.
    Instance : (q, P, Q, H_1) where $H_1 : \{0,1\}^* \rightarrow G_1^*$ is a cryptographic hash function and access to the target and helper oracles with at most $q_{\mathcal{T}}$ and $q_{\mathcal{H}}$ queries respectively.
    Output : A set V of, say l pairs $((V_1, j_1), (V_2, j_2), \ldots, (V_l, j_l))$, where for all $i, 1 \leq i \leq l$, there exists $j_i, 1 \leq j_i \leq q_{\mathcal{T}}$ such that $V_i = sU_{j_i}$ where all V_i are distinct and $q_{\mathcal{H}} < q_{\mathcal{T}}, l$.
    (See section A.3.2).

11. Chosen-target Inverse CDH problem in G_1 [122] :
    Let s be a random element of Z_q^* and $Q = sP$.
    Oracles : (a) A target oracle $\mathcal{T}_{G_1}$ that returns a random element $U_i \in G_1$. (b) A helper oracle $\mathsf{Inv-cdh}-s(\cdot)$ that computes $s^{-1}U$ for a randomly chosen input $U \in G_1$.
    Instance : (q, P, Q, H_1) where $H_1 : \{0,1\}^* \rightarrow G_1^*$ is a cryptographic hash function and access to the target and helper oracles with at most $q_{\mathcal{T}}$ and $q_{\mathcal{H}}$ queries respectively.
    Output : A set V of, say l pairs $((V_1, j_1), (V_2, j_2), \ldots, (V_l, j_l))$, where for all $i, 1 \leq i \leq l$, there exists $j_i, 1 \leq j_i \leq q_{\mathcal{T}}$ such that $V_i = s^{-1}U_{j_i}$ where all V_i are distinct and $q_{\mathcal{H}} < q_{\mathcal{T}}, l$.

2.4 Bilinear Diffie-Hellman Problems

1. Bilinear Diffie-Hellman (BDH) problem in (G_1, G_2, e) :
   Instance : (P, aP, bP, cP) for some $a, b, c \in Z_q^*$.
   Output : $e(P, P)^{abc}$.
   (See sections 3.4.1, 3.4.2, A.2.1-A.2.4, A.5.2, A.5.3, A.6.1, A.6.3, A.3.3).

2. Decisional Bilinear Diffie-Hellman (DBDH) problem in (G_1, G_2, e) :
   Instance : (P, aP, bP, cP, r) for some $a, b, c \in_R Z_q^*$, $r \in_R G_2$.
   Output : yes if $r = e(P, P)^{abc}$ and output no otherwise.
   (See sections A.2.6, A.6.3).

3. Decisional Hash Bilinear Diffie-Hellman (DHBDH) problem in (G_1, G_2, e) :
   Instance : (P, aP, bP, cP, r) for some $a, b, c, r \in Z_q^*$ and a one way hash function $H : G_2 \rightarrow Z_q^*$.
   Output : yes if $r = H(e(P, P)^{abc}) \bmod q$ and output no otherwise.
   Comments : The DHBDH problem in (G_1, G_2, e) is a hash version of the decisional BDH problem in (G_1, G_2, e) .
   (See Chapter 4).

4. k-Bilinear Diffie-Hellman Inversion (k-BDHI) problem in (G_1, G_2, e) :
   Instance : $(P, yP, y^2P, \ldots, y^kP)$ for some $y \in Z_q^*$.
   Output : $e(P, P)^{\frac{1}{y}} \in G_2$.
   Comments : 1-BDHI assumption is polynomially equivalent to the standard BDH assumption. It is not known if the k-BDHI assumption, for $k > 1$, is polynomially equivalent to BDH.
   (See section A.3.4)

5. k-Decisional Bilinear Diffie-Hellman Inversion (k-DBDHI) problem in (G_1, G_2, e) :
   Instance : $(P, yP, y^2P, \ldots, y^kP, r)$ for some $y \in Z_q^*$, $r \in_R G_2$.
   Output : yes if $r = e(P, P)^{\frac{1}{y}} \in G_2$ and output no otherwise.
   (See section A.2.5).

2.5 Miscellaneous Problems

1. ROS problem [103, 120]:[1]
   Oracle : A random function $F : Z_q^l \rightarrow Z_q$.
   Instance : A system of t equations in l unknowns $c_1, c_2, \ldots, c_l$ over Z_q^* : $a_{k,1}c_1 + \cdots + a_{k,l}c_l = F(a_{k,1}, \ldots, a_{k,l})$ for $k = 1, 2, \ldots, t$, $t \geq l + 1$.
   Output : Co-efficient $a_{k,i} \in Z_q^*$ and a solvable subsystem of $l + 1$ equations in the unknowns $c_1, c_2, \ldots, c_l$.
   (See section A.3.8).

[1] The term ROS was first introduced in [103] by Schnorr and the term is not explained. But it may stand for Random Overdetermined Solvable system of linear equations.

2. Co-Gap Diffie-Hellman (Co-GDH) group : Consider a cryptographic bilinear map in the following setup :
   a) G_1, G_2 are two additive groups and G_T is a multiplicative group of prime order q;
   b) P_1 is a generator of G_1 and P_2 is a generator of G_2;
   c) ψ is a computable isomorphism from G_1 to G_2, with $\psi(P_1) = P_2$; and
   d) e is an efficiently computable bilinear map $e : G_1 \times G_2 \rightarrow G_T$ satisfying the following properties :

   – *Bilinearity :* For all $Q_1 \in G_1, Q_2 \in G_2$ and $a, b \in Z_q^*$, $e(aQ_1, bQ_2) = e(Q_1, Q_2)^{ab}$.
   – *Non-degeneracy :* $e(P_1, P_2) \neq 1$.

   These properties imply one more : for any $Q_1, Q_2 \in G_1$, $e(Q_1, \psi(Q_2)) = e(Q_2, \psi(Q_1))$. (Such bilinear maps can be derived from Weil pairing and Tate pairing; for simplicity the reader may assume $G_1 = G_2$). We refer this setup as the Co-GDH setup. With this setup, we obtain natural generalizations of the CDH and DDH problems :

   Computational Co-Diffie-Hellman (Co-CDH) problem :
   Instance : (P_1, P_2, aP_1, bP_2) for some $a, b \in Z_q^*$.
   Output : $abP_2 \in G_2$.

   Decisional Co-Diffie-Hellman (Co-DDH) problem :
   Instance : $(P_1, P_2, aP_1, bP_2, cP_2)$ for some $a, b, c \in Z_q^*$.
   Output : yes if $c = ab \bmod q$ and output no otherwise.

   When $G_1 = G_2$ and $P_1 = P_2$, these problems reduced to the standard CDH and DDH problems respectively.

   Groups G_1, G_2 are said to be Co-GDH groups if there exists an efficient algorithm to solve the Co-DDH problem and there is no polynomial-time (in $|q|$) algorithm to solve the Co-CDH problem. The existence of a cryptographic bilinear map ensures the existence of Co-GDH groups. (See sections A.3.4, A.3.5, A.3.6).

2.6 Security Models

It is extremely important to correctly define the security model as well as to prove the security of any proposed protocol into that well defined model. Broadly speaking, the process of providing security in a protocol comes in five stages.

1. Specification of model;
2. Definition of goals within this model;
3. Statement of assumption;
4. Description of protocol;

5. Proof that the protocol meets its goals within the model.

In this section, we briefly review the security models for the three fundamental cryptographic primitives: Encryption, Digital Signature and Key Agreement. There are several variants of public key encryption: ID-based encryption (IBE), Searchable Public Key Encryption (SPKE), Hierarchical ID-based Encryption (HIDE); and depending on the nature and requirement of practical applications, there are a wide variety of signature schemes: Blind signature, Multi-signature, Aggregate signature, Verifiably encrypted signature, Ring signature, Group signature, Unique signature *etc.* Apart from these three basic primitives, there are protocol designs for Signcryption, Threshold decryption, Key sharing, Identification schemes, Chameleon hashes *etc.* Describing the security notions and the security models of each of them is beyond the scope of this thesis.

2.6.1 Security Model for Encryption Schemes

The strongest notion of security for public key encryption scheme is the indistinguishability of encryptions against adaptive chosen ciphertext attack (IND-CCA) [12, 100].

An encryption scheme consists of three algorithms.

1. Setup : Generates system parameters and for each entity a pair of encryption and decryption key. The system parameters and the encryption key are public whereas the decryption key corresponding to the encryption key is kept private to the corresponding entity.
2. Encrypt : Encrypts message using the public encryption key.
3. Decrypt : Decrypts the message using the private decryption key of the corresponding public encryption key.

We describe below the IND-CCA security for encryption scheme. Consider the following game between the challenger and an adversary $\mathcal{A}$.

Setup: The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary the resulting system parameters params and a public encryption key PK. The corresponding decryption key SK is kept secret to itself.

Phase 1: The adversary $\mathcal{A}$ is given access to the decryption oracle corresponding to the private decryption key SK. $\mathcal{A}$ issues decryption queries on ciphertexts $C_1, \ldots, C_m$ (polynomially many) to the challenger. The challenger responds by running algorithm $\mathsf{Decrypt}$ to decrypt the ciphertext C_i, $1 \leq i \leq m$ using the private decryption key SK. It sends the resulting plaintext M_i, $1 \leq i \leq m$ to the adversary $\mathcal{A}$.

These queries may be asked adaptively, that is, each query C_i may depend on the replies to $C_1, \ldots, C_{i-1}$.

Challenge: Once the adversary decides that Phase 1 is over, it outputs two equal length plaintexts M_0, M_1. The challenger picks a random bit $b \in \{0,1\}$ and sets $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{PK}, M_b)$. It sends C as the challenge to the adversary.

Phase 2: The adversary issues more decryption queries $C_{m+1}, \ldots, C_n$ (polynomially many) to the challenger with the restriction that $C_i \neq C$ for $m+1 \leq i \leq n$. Challenger responds as in Phase 1.

Guess: Finally, the adversary $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-CCA adversary. The advantage for the adversary $\mathcal{A}$ in attacking the scheme is defined as

$$\mathsf{Adv}(\mathcal{A}) = |\mathsf{Prob}[b = b'] - \frac{1}{2}|.$$

We say that an identity based scheme is semantically secure against an adaptive chosen ciphertext attack (IND-CCA) if no polynomially bounded adversary $\mathcal{A}$ has non-negligible advantage against the challenger.

Boneh and Franklin [24] *strengthened* the IND-CCA model to IND-ID-CCA model which is the standard notion of security for ID-based encryption schemes. In an ID-based encryption scheme there are four algorithms.

1. Setup : Creates system parameters and *master key*.
2. Extract : Uses master key to generate the private key corresponding to an arbitrary public key string ID.
3. Encrypt : Encrypts messages using the public key ID.
4. Decrypt : Decrypts the message using the corresponding private key of ID.

The IND-ID-CCA model deals with an adversary who possesses private keys corresponding to identities of its choice $\mathsf{ID}_1, \ldots, \mathsf{ID}_n$ and attacks an identity ID in an ID-based system. Consider the following game between the challenger and an adversary $\mathcal{A}$.

Setup: The challenger takes a security parameter k and runs the Setup algorithm. It gives the adversary the resulting system parameters params and keeps the master key secret to itself.

Phase 1: The adversary issues queries $q_1, \ldots, q_m$ where q_i is one of the following two queries.

– Extraction query $\langle \mathsf{ID}_i \rangle$. The challenger responds by running algorithm $\mathsf{Extract}$ to generate the private key d_i corresponding to the public key $\langle \mathsf{ID} \rangle$. It then sends d_i to the adversary.

– Decryption query $\langle \mathsf{ID}_i, C_i \rangle$. The challenger responds by running algorithm Extract to generate the private key d_i corresponding to ID_i. It then runs algorithm Decrypt to decrypt the ciphertext C_i using the private key d_i. It sends the resulting plaintext to the adversary.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \ldots, q_{i-1}$.

Challenge: Once the adversary decides that Phase 1 is over, it outputs two equal length plaintexts M_0, M_1, an identity ID on which it wishes to be challenged. The only constraint is that ID did not appear in any private key extraction queries in Phase 1. The challenger picks a random bit $b \in \{0, 1\}$ and sets $C = \mathsf{Encrypt}(\mathsf{params}, \mathsf{ID}, M_b)$. It sends C as the challenge to the adversary.

Phase 2: The adversary issues more queries $q_{m+1}, \ldots, q_n$ where q_i is one of the following two queries.

– Extraction query $\langle \mathsf{ID}_i \rangle$ where $\mathsf{ID}_i \neq \mathsf{ID}$. Challenger responds as in Phase 1.

– Decryption query $\langle \mathsf{ID}_i, C_i \rangle \neq \langle \mathsf{ID}, C \rangle$. Challenger responds as in Phase 1.

Guess: Finally, the adversary outputs a guess $b' \in \{0, 1\}$ and wins the game if $b = b'$.

We refer to such an adversary $\mathcal{A}$ as an IND-ID-CCA adversary. The advantage for the adversary $\mathcal{A}$ in attacking the scheme is defined as

$$\mathsf{Adv}(\mathcal{A}) = |\mathsf{Prob}[b = b'] - \frac{1}{2}|.$$

We say that an identity based scheme is semantically secure against an adaptive chosen ciphertext attack (IND-ID-CCA) if no polynomially bounded adversary $\mathcal{A}$ has non-negligible advantage against the challenger.

The IND-ID-CCA model is the *strongest* acceptable notion of security and the security model for other public key encryption schemes (*e.g.* SPKE, HIDE *etc.*) are based on it. A less stronger security notion of a public key encryption is chosen plaintext attack (IND-CPA) where the adversary is not allowed to perform Phase 2.

Boneh and Boyen [21] gave Selective ID model, which is slightly weaker than the model described above. In this model the adversary must commit ahead of the time to the identity that it intends to attack, whereas in the standard model described above, the adversary is allowed to choose this identity adaptively.

The security notion for (ID-based) threshold decryption is a modified extension of the security model described above in the threshold setting. More details can be found in [6].

2.6.2 Security Model for Digital Signature Schemes

Security against existential forgery under adaptive chosen message attack is the strongest notion of security for digital signature schemes. This was defined by Goldwasser, Micali and Rivest [62].

A standard digital signature scheme $\mathsf{DSig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ consists of three algorithms.

1. $\mathcal{K}$: the key generation algorithm that generates randomly public system parameters params and public/secret key pair PK, SK of a signer.
2. $\mathcal{S}$: signature generation algorithm that generates a signature on a given message m using the secret key SK of a signer.
3. $\mathcal{V}$: signature verification algorithm that checks the validity of a signature on a given message using the public key of a signer.

Consider the following game between the challenger and a forger $\mathcal{F}$.

Step 1: The challenger takes a security parameter k as input and runs the key generation algorithm $\mathcal{K}$ to generate respectively the system parameters params and public/secret key pair PK, SK of a signer. It gives the forger $\mathcal{F}$ the resulting system parameters params and the public (verification) key PK, and keeps the secret key SK to itself.

Step 2: The forger $\mathcal{F}$ has the access to the signing oracle corresponding to the public key PK. $\mathcal{F}$ issues signature queries on messages $m_1, \ldots, m_n$ (polynomially many queries) to the signing oracle and receives respective valid signatures $\sigma_1, \ldots, \sigma_n$. $\mathcal{F}$ can make these queries adaptively (*i.e.* after seeing the signature σ_i for the message m_i, it decides the message m_{i+1} for the next signature query).

Step 3: Finally, $\mathcal{F}$ outputs a message, signature pair (m, σ) where $m \neq m_1, \ldots, m_n$.

The forger $\mathcal{F}$ wins the game if σ is a valid signature on message m under the public key PK with the restriction that $\mathcal{F}$ did not submit m to the signing oracle.

We say that the signature scheme DSig is existentially unforgeable under an adaptive chosen message attack if it is infeasible for a forger who only knows the public key to produce a valid message-signature pair after obtaining polynomially many signatures on messages of its choice from the signer. The advantage in existentially forging a signature of a forger algorithm $\mathcal{F}$, given access of a signing oracle $\mathcal{S}$, is defined to be

$$\mathsf{ADV}_{\mathsf{DSig}}(\mathcal{F}) := \mathsf{Prob}[\mathcal{V}(\mathsf{PK}, m, \sigma) = 1 : (\mathsf{PK}, \mathsf{SK}) \stackrel{R}{\leftarrow} \mathcal{K}, (m, \sigma) \stackrel{R}{\leftarrow} \mathcal{F}^{\mathcal{S}}(\mathsf{PK})].$$

The probability is taken over the coin tosses of the key generation algorithm and of the forger. Here the forger $\mathcal{F}$ is allowed to query the signing oracle adaptively: any of its query may depend on previous answers, but it may not emit a signature for a message on which it had previously

queried the oracle. The forger may additionally have access of a hash oracle, which can be used as a random oracle.

Formally a signature scheme is secure against existential forgery on adaptive chosen-message attacks if for every probabilistic polynomial-time forger algorithm $\mathcal{F}$, there does not exist a non-negligible probability ϵ such that $\mathsf{ADV}_{\mathsf{DSig}}(\mathcal{F}) \geq \epsilon$.

We now describe the security notion of Aggregate signature scheme and Multisignature scheme based on the basic digital signature scheme DSig. We use aggregate signature and multi signature in our articles [52, 53] (included in Chapter 4).

Security of Aggregate Signature:

Formally the aggregate signature scheme $\mathsf{ASig} = (\mathcal{K}, \mathcal{S}, \mathcal{V}, \mathcal{AS}, \mathcal{AV})$ consists of five algorithms, where $\mathsf{DSig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is a standard digital signature scheme, called the base signature scheme. Here $\mathcal{K}$ is the randomized key generation algorithm, $\mathcal{S}$ is the (possibly) randomized signing algorithm and $\mathcal{V}$ is the deterministic verification algorithm. The aggregation algorithm and the aggregation verification algorithm are respectively $\mathcal{AS}$ and $\mathcal{AV}$.

Let $\mathsf{PK}_i, \mathsf{SK}_i$ be the public and private key respectively for player $U_i, 1 \leq i \leq n$, for the scheme DSig. The aggregate signature is generated as follows :

$$\mathcal{AS}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m_1, \ldots, m_n, \sigma_1, \ldots, \sigma_n) = \sigma$$

Each signature σ_i is valid for message m_i relative to public key PK_i and σ is the single aggregate signature. The messages m_i are all distinct. The verification is done by checking whether
$\mathcal{AV}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m_1, \ldots, m_n, \mathcal{AS}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m_1, \ldots, m_n, \mathcal{S}(\mathsf{SK}_1, m_1), \ldots, \mathcal{S}(\mathsf{SK}_n, m_n)))$
$= 1$.

Note that the aggregate signature verification holds if and only if the individual signature verification in DSig holds. *i.e.* $\mathcal{V}(\mathsf{PK}_i, m_i, \mathcal{S}(\mathsf{SK}_i, m_i)) = 1$ holds for all user U_i, $1 \leq i \leq n$.

Next we describe the security model of aggregate signature : the *aggregate chosen key model*. Informally, the goal of the aggregate forger $\mathcal{F}$ in attacking ASig is to existentially forge an aggregate signature given access to a single challenge public key and a signing oracle on this challenge key. The other public keys are of adversary's choice. To formalize the idea, consider the following game between the challenger and a forger $\mathcal{F}$.

Step 1: A public key PK_1, generated at random, is provided to the aggregate forger $\mathcal{F}$ which in addition has also given the power to access the signing oracle corresponding to PK_1.

Step 2: $\mathcal{F}$ requests adaptively signatures with PK_1 on messages of his choice.

Step 3: At the end, $\mathcal{F}$ outputs $n-1$ additional public keys $\mathsf{PK}_2, \ldots, \mathsf{PK}_n$ where $n \geq 1$ is the number of users participating in the generation of aggregate signature, a sequence of distinct messages $m_1, \ldots, m_n$ and an aggregate signature σ for these messages.

The forger $\mathcal{F}$ wins the game if σ is a valid aggregate signature on messages $m_1, \ldots, m_n$ under public keys $\mathsf{PK}_1, \ldots, \mathsf{PK}_n$ with the restriction that $\mathcal{F}$ did not submit m_1 to the signing oracle. The advantage of the forger $\mathcal{F}$ is defined to be the probability of success of $\mathcal{F}$ in the above game where probability is over coin tosses of the key generation and of $\mathcal{F}$. The aggregate signature scheme ASig is said to be secure if there is no probabilistic, polynomial-time forger with non-negligible advantage.

Security of Multi Signature:

Formally the multisignature scheme $\mathsf{MSig} = (\mathcal{K}, \mathcal{S}, \mathcal{V}, \mathcal{MS}, \mathcal{MV})$ consists of five algorithms, where $\mathsf{DSig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ is a standard digital signature scheme, called the base signature scheme. Here $\mathcal{K}$ is the randomized key generation algorithm, $\mathcal{S}$ is the (possibly) randomized signing algorithm and $\mathcal{V}$ is the deterministic verification algorithm. The multisignature generation algorithm and the multisignature verification algorithm are respectively $\mathcal{MS}$ and $\mathcal{MV}$.

Let $\mathsf{PK}_i, \mathsf{SK}_i$ be the public and private key respectively for player $U_i, 1 \leq i \leq n$, for the scheme DSig. The multisignature on a message m is generated as follows :

$$\mathcal{MS}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m, \sigma_1, \ldots, \sigma_n) = \sigma$$

Each signature σ_i is valid for message m relative to public key PK_i and σ is the single multisignature. The verification is done by checking whether

$$\mathcal{MV}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m, \mathcal{MS}(\mathsf{PK}_1, \ldots, \mathsf{PK}_n, m, \mathcal{S}(\mathsf{SK}_1, m), \ldots, \mathcal{S}(\mathsf{SK}_n, m))) = 1.$$

Note that the multisignature verification holds if and only if the individual signature verification in DSig holds. *i.e.* $\mathcal{V}(\mathsf{PK}_i, m, \mathcal{S}(\mathsf{SK}_i, m)) = 1$ holds for all user U_i, $1 \leq i \leq n$.

Next we describe the security model of multisignature. Informally, the goal of the forger $\mathcal{F}$ in attacking MSig is to existentially forge a multisignature given access to a single challenge public key and a signing oracle on this challenge key. The other public keys are of adversary's choice. To formalize the idea, consider the following game between the challenger and a forger $\mathcal{F}$.

Step 1: A public key PK_1, generated at random, is provided to the forger $\mathcal{F}$ which in addition has also given the power to access the signing oracle corresponding to PK_1.

Step 2: $\mathcal{F}$ requests adaptively signatures with PK_1 on messages of his choice.

Step 3: At the end, $\mathcal{F}$ outputs $n-1$ additional public keys $\mathsf{PK}_2, \ldots, \mathsf{PK}_n$ where $n \geq 1$ is the number of users participating in the generation of multisignature, a message m and a multisignature σ for the message m.

The forger $\mathcal{F}$ wins the game if σ is a valid multisignature on message m under public keys $\mathsf{PK}_1, \ldots, \mathsf{PK}_n$ with the restriction that $\mathcal{F}$ did not submit m to the signing oracle. The advantage of the forger $\mathcal{F}$ is defined to be the probability of success of $\mathcal{F}$ in the above game

where probability is over coin tosses of the key generation and of $\mathcal{F}$. The multisignature scheme MSig is said to be *existentially unforgeable under chosen message attack* if there is no probabilistic, polynomial-time forger with non-negligible advantage.

2.6.3 Security Attributes and Model for Key Agreement Schemes

In this section, we review the security goals and attributes that a key agreement protocol should posses and describe the adversarial model for key agreement.

(a) Protocol Goals and Attributes

It is necessary to identify what attacks a protocol should withstand and what attributes are desirable for a protocol to have. The goals of authenticated key agreement protocols, unlike other primitives, such as encryption or digital signatures, lacked formal definition for a long time. Here we discuss the various desirable attributes and goals that one may wish a key agreement protocol to posses. The following definitions are taken from [4, 3, 16].

First we identify two types of attacks:

1. *Passive attacks*: Here an adversary attempts to prevent a protocol achieving its goal by simply observing an honest execution of the protocol among entities and therefore analyzing the recorded data (the transcript of the protocol execution).
2. *Active attacks*: Here an adversary additionally subverts the communication themselves in any way possible: by injecting messages, intercepting messages, replaying messages, altering messages, and the like.

Clearly, it is essential for any secure protocol to withstand both passive and active attacks, since an adversary can reasonably be assumed to have these capabilities in a distributed network.

Now we present the fundamental security goals and desirable security and performance attributes for key agreement protocols.

- **Fundamental Security Goals:**

  The fundamental security goals for a protocol described below are defined to be design goals that are independent of the protocol details and considered to be vital in any application.

  1. *Implicit Key Authentication*: This goal, if met, assures an entity A that only the intended other entities with whom A wants to agree upon a joint key may be able to compute a particular key. This level of authentication results in what is known as an Authenticated Key Agreement (AK) protocol.

2. *Key Confirmation*: Here an entity is assured that one or more other entities actually computed the shared session key.
3. *Explicit Key Authentication*: This goal is met when both implicit key authentication and key confirmation goals are met. This creates an Authenticated Key Agreement with Key Confirmation (AKC) protocol.
4. *Good Key*: This goal states that the key is selected uniformly at random from the key space, so that no adversary has an information-theoretic advantage when mounting a guessing strategy to determine the key.

- **Desirable Security Attributes:**

Note that a key agreement protocol may be executed several times (sessions) among any subset of n entities and a common (session) key is established in each session. Each entity may have a long-term secret key (issued by a certifying authority/ public key generator). While executing the key agreement protocol, a participant chooses a random ephemeral key (or a short-term secret key).
A number of desirable security attributes of key agreement protocol have been identified [4, 3, 16] which can be vital in excluding realistic attacks depending on the application scenario.

1. *Known Session Key Security*: A protocol is known session key secure if an adversary, having obtained some previous session keys, still cannot get the session key of the current run of the protocol.
2. *(Perfect) Forward Secrecy*: A protocol enjoys forward secrecy if compromise of the long-term secret keys of one or more entities does not affect the security of the previous session keys. If this property holds even when the private keys of all the participating entities are compromised, we say that the protocol achieves perfect forward secrecy.
3. *No Key Compromise Impersonation*: A protocol resists key-compromise impersonation when the compromise of one entity's long-term secret key does not imply that the private keys of other users will also be compromised. The adversary may impersonate the compromised entity in the subsequent protocol executions, but cannot impersonate other entities.
4. *No Unknown Key-Share*: We say that a protocol is subjected to unknown key-share attack if an adversary convinces a group of parties that they share a key with the adversary, whereas in fact the key is shared between the group and another party.
5. *No Key Control*: A protocol is said to have no key control if it is not possible by any participant (or adversary) to force the session key to a preselected value or predict the value of the session key.

- **Desirable Performance Attributes:**

These attributes include:

1. Number of messages exchanged among entities and number of rounds required.
2. Bandwidth required by message, *i.e.* total number of bits transmitted.
3. Complexity of computation by each entity as it affects execution time.
4. Possibility of pre-computation to reduce on-line computational complexity.

Also the protocol is called role symmetry if the messages transmitted have the same structure and message-independent if the messages transmitted among entities are independent of each other.

(b) Security Model for Key Agreement

A sound formalization of security model for the authenticated key agreement is introduced by Bresson *et al.* [32, 33, 35]. These works are based on the initial work of Bellare and Rogaway [15] and Bellare, Canetti and Krawczyk [11]. We describe below the adversarial model following Bresson *et al.*'s [32] formal security model which is more general in the sense that it covers authenticated key agreement in group setting and suited for dynamic groups.

Let $\mathcal{P} = \{U_1, \ldots, U_n\}$ be a set of n (fixed) users or participants. At any point of time, any subset of $\mathcal{P}$ may decide to establish a session key. Thus a user can execute the protocol for group key agreement several times with different partners, can join or leave the group at his desire by executing the protocols for Join or Leave. We identify the execution of protocols for key agreement, member(s) join and member(s) leave as different sessions. The adversarial model consists of allowing each user an unlimited number of instances with which it executes the protocol for key agreement or inclusion or exclusion of a user or a set of users. We assume adversary never participates as a user in the protocol. This adversarial model allows concurrent execution of the protocol. The interaction between the adversary $\mathcal{A}$ and the protocol participants occur only via oracle queries, which model the adversary's capabilities in a real attack. Let S, S_1, S_2 be three sets defined as:

$$S = \{(V_1, i_1), \ldots, (V_l, i_l)\}, S_1 = \{(V_{l+1}, i_{l+1}), \ldots, (V_{l+k}, i_{l+k})\}, S_2 = \{(V_{j_1}, i_{j_1}), \ldots, (V_{j_k}, i_{j_k})\}$$

where $\{V_1, \ldots, V_l\}$ is any non-empty subset of $\mathcal{P}$. We will require the following notations.

Π_U^i : i-th instance of user U.
sk_U^i : session key after execution of the protocol by Π_U^i.
sid_U^i : session identity for instance Π_U^i. We set $\mathsf{sid}_U^i = S = \{(U_1, i_1), \ldots, (U_k, i_k)\}$ such that $(U, i) \in S$ and $\Pi_{U_1}^{i_1}, \ldots, \Pi_{U_k}^{i_k}$ wish to agree upon a common key.
pid_U^i : partner identity for instance Π_U^i, defined by $\mathsf{pid}_U^i = \{U_1, \ldots, U_k\}$, such that $(U_j, i_j) \in \mathsf{sid}_U^i$ for all $1 \leq j \leq k$.
acc_U^i : 0/1-valued variable which is set to be 1 by Π_U^i upon normal termination of the session and 0 otherwise.

We assume that the adversary has complete control over all communications in the network. All information that the adversary gets to see is written in a transcript. So a transcript consists of all the public information flowing across the network. The following oracles model an adversary's interaction with the users in the network:

- $\mathsf{Send}(U, i, m)$: This query models an active attack, in which the adversary may intercept a message and then either modify it, create a new one or simply forward it to the intended participant. The output of the query is the reply (if any) generated by the instance Π_U^i upon receipt of message m. The adversary is allowed to prompt the unused instance Π_U^i to initiate the protocol with partners $U_2, \ldots, U_l, l \leq n$, by invoking $\mathsf{Send}(U, i, \langle U_2, \ldots, U_l \rangle)$.
- $\mathsf{Execute}(S)$: This query models passive attacks in which the attacker eavesdrops on honest execution of group key agreement protocol among unused instances $\Pi_{V_1}^{i_1}, \ldots, \Pi_{V_l}^{i_l}$ and outputs the transcript of the execution. A transcript consists of the messages that were exchanged during the honest execution of the protocol.
- $\mathsf{Join}(S, S_1)$: This query models the insertion of user instances $\Pi_{V_{l+1}}^{i_{l+1}}, \ldots, \Pi_{V_{l+k}}^{i_{l+k}}$ in the group $\{\Pi_{V_1}^{i_1}, \ldots, \Pi_{V_l}^{i_l}\}$ for which $\mathsf{Execute}$ have already been queried. The output of this query is the transcript generated by the invocation of algorithm Join. If $\mathsf{Execute}(S)$ has not taken place, then the adversary is given no output.
- $\mathsf{Leave}(S, S_2)$: This query models the removal of user instances $\Pi_{V_{j_1}}^{i_{j_1}}, \ldots, \Pi_{V_{j_k}}^{i_{j_k}}$ from the group $\{\Pi_{V_1}^{i_1}, \ldots \Pi_{V_l}^{i_l}\}$. If $\mathsf{Execute}(S)$ has not taken place, then the adversary is given no output. Otherwise, algorithm Leave is invoked. The adversary is given the transcript generated by the honest execution of procedure Leave.
- $\mathsf{Reveal}(U, i)$: This unconditionally outputs session key sk_U^i if it has previously been accepted by Π_U^i, otherwise a value NULL is returned. This query models the misuse of the session keys, *i.e* known session key attack.
- $\mathsf{Corrupt}(U)$: This outputs the long-term secret key (if any) of player U. The adversarial model that we adopt is a weak-corruption model in the sense that only the long-term secret keys are compromised, but the ephemeral keys or the internal data of the protocol participants are not corrupted. This query models (perfect) forward secrecy.
- $\mathsf{Test}(U, i)$: This query is allowed only once, at any time during the adversary's execution. A bit $b \in \{0, 1\}$ is chosen uniformly at random. The adversary is given sk_U^i if $b = 1$, and a random session key if $b = 0$. This oracle computes the adversary's ability to distinguish a real session key from a random one.

An adversary which has access to the $\mathsf{Execute}$, Join, Leave, Reveal, $\mathsf{Corrupt}$ and Test oracles, is considered to be passive while an active adversary is given access to the Send oracle in addition. (For static case, there are no Join or Leave queries as a group of fixed size is considered.)

The adversary can ask Send, $\mathsf{Execute}$, Join, Leave, Reveal and $\mathsf{Corrupt}$ queries several times, but Test query is asked only once and on a fresh instance. We say that an instance Π_U^i is *fresh*

unless either the adversary, at some point, queried $\mathsf{Reveal}(U, i)$ or $\mathsf{Reveal}(U', j)$ with $U' \in \mathsf{pid}_U^i$ or the adversary queried $\mathsf{Corrupt}(V)$ (with $V \in \mathsf{pid}_U^i$) before a query of the form $\mathsf{Send}(U, i, *)$ or $\mathsf{Send}(U', j, *)$ where $U' \in \mathsf{pid}_U^i$.

Finally adversary outputs a guess bit b'. Such an adversary is said to win the game if $b = b'$ where b is the hidden bit used by the Test oracle.

Let Succ denote the event that the adversary $\mathcal{A}$ wins the game for a protocol XP. We define

$$\mathsf{Adv}_{\mathcal{A},\mathsf{XP}} := |2\ \mathsf{Prob}[\mathsf{Succ}] - 1|$$

to be the advantage of the adversary $\mathcal{A}$ in attacking the protocol XP.

The protocol XP is said to be a *secure unauthenticated group key agreement* (KA) protocol if there is no polynomial time *passive* adversary with non-negligible advantage. In other words, for every probabilistic, polynomial-time, 0/1 valued algorithm $\mathcal{A}$, $\mathsf{Adv}_{\mathcal{A},\mathsf{XP}} < \frac{1}{M^L}$ for every fixed $L > 0$ and sufficiently large integer M. We say that protocol XP is a *secure authenticated group key agreement* (AKA) protocol if there is no polynomial time *active* adversary with non-negligible advantage.

Next we define

$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{KA}}(t, q_E)$	:=	the maximum advantage of any passive adversary attacking protocol XP, running in time t and making q_E calls to the $\mathsf{Execute}$ oracle.
$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{AKA}}(t, q_E, q_S)$	:=	the maximum advantage of any active adversary attacking protocol XP, running in time t and making q_E calls to the $\mathsf{Execute}$ oracle and q_S calls to the Send oracle.
$\mathsf{Adv}_{\mathsf{XP}}^{\mathsf{AKA}}(t, q_E, q_J, q_L, q_S)$	:=	the maximum advantage of any active adversary attacking protocol XP, running in time t and making q_E calls to the $\mathsf{Execute}$ oracle, q_J calls to Join oracle, q_L calls to the Leave oracle and q_S calls to the Send oracle.

Remark 2.6.1 *Session identity is required to identify a session uniquely and all participants executing a session should hold the same session identity. Conventionally, session identity sid_U^i for an instance Π_U^i is set to be the concatenation of all (broadcasted) messages sent and received by Π_U^i during its course of execution. This essentially assumes that all the partners of Π_U^i hold the same concatenation value of sent and received messages which may not be the case in general. Our definition of session identity is different and can be applied for more general protocols.*

Remark 2.6.2 *We will make the assumption that in each session at most one instance of each user participates. Further, an instance of a particular user participates in exactly one session. This is not a very restrictive assumption, since a user can spawn an instance for each session it participates in. On the other hand, there is an important consequence of this*

assumption. Suppose there are several sessions which are being concurrently executed. Let the session ID's be $\mathsf{sid}_1, \ldots, \mathsf{sid}_k$. *Then for any instance* Π^i_U, *there is exactly one* j *such that* $(U, i) \in \mathsf{sid}_j$ *and for any* $j_1 \neq j_2$, *we have* $\mathsf{sid}_{j_1} \cap \mathsf{sid}_{j_2} = \emptyset$. *Thus at any particular point of time, if we consider the collection of all instances of all users, then the relation of being in the same session is an equivalence relation whose equivalence classes are the session IDs. Moreover, an instance* Π^i_U *not only knows* U, *but also the instance number* i – *this being achieved by maintaining a counter.*

Note : The security model for key agreement protocols that we describe here does not offer any protection against the influence of malicious insiders or users who do not follow the protocol honestly. Since the model gives the adversary *complete* control over all communication in the network, the adversary can modify messages or delay to deliver messages or refuse to deliver them at all. Thus the adversary can mount "denial of service" attacks, and as a consequence of its ability to refuse delivering messages, can cause an honest instance to "hang" indefinitely. Furthermore, if the adversary is allowed to corrupt *all* parties (adopting strong corruption model where all internal data together with the long-term secret key (if any) of participants get revealed), full-fledge agreement is clearly impossible.

However, some of these concerns can be addressed partially within the framework above. For example, key confirmation can be introduced in *any* authenticated group key agreement protocol by incorporating an additional round as follows [70]: after computing key sk, each user U_i computes $x_i = H(\mathsf{sk}|U_i)$, signs x_i, broadcast x_i and the corresponding signature, and computes the "actual" session key $\mathsf{sk}' = H(\mathsf{sk}|\perp)$. Here H is modeled as a random oracle and $\perp$ represents some distinguishing string. Other players check the validity of the broadcast values in the usual manner.

It is indeed an interesting challenge to specify meaningful and achievable goals that take into account malicious insiders. To design tools to explore capabilities of malicious insiders in a group key agreement protocol and providing security of protocols into that framework is an interesting direction for future research.

2.7 Conclusion

This chapter is motivated from providing the basic notations and definitions that are required in the discussion of subsequent chapters. In this chapter, we introduce some background materials on elliptic curves and bilinear pairings. We discuss several variants of Diffie-Hellman problems. Finally we concentrate on describing the security notions and security models for certain basic cryptographic primitives.

Chapter 3

Overview of Key Agreement Protocols

The emphasis of this chapter is to focus on key agreement. To this aim, we address a self-contained, up-to-date presentation of key agreement protocols at high level. We have attempted to provide a brief but fairly complete survey of all these schemes.

3.1 Introduction

Security of many cryptographic protocols hinges on efficient key distribution and key management mechanism. Key distribution scheme enables a distinguished set of users, to obtain a common secret key. If the group key is generated and distributed by a central trusted party via secure channels, then it involves not only the participation of trusted authority, but also requires avialbility of secure channel between the trusted party and each user. This will give rise to complex key management and key distribution mechanism, which may be impractical or costly. In this chapter, we are concerned with the efficient provably secure contributory group key distribution via public open network, called key agreement, which interactively establish a group key such that each group member makes a contribution to the group key.

Key agreement is one of the fundamental cryptographic primitive after encryption and digital signature. Such protocols allow two or more parties to exchange information among themselves over an adversarially controlled insecure network and agree upon a common session key, which may be used for later secure communication among the parties. Thus, secure key agreement protocols serve as basic building block for constructing secure, complex, higher-level protocols.

The problem of designing efficient key agreement protocols, both in the two party and multi party (*i.e.* group) setting with lower computation and communication cost and round complexity have received much attention. The first pioneering work for key agreement is the Diffie-Hellman protocol given in their seminal paper [49] that invents the public key cryptography and revolutionizes the field of modern cryptography . However, the basic Diffie-Hellman protocol does not authenticate the two communication entities in the sense that an active adversary who has control

over the channel can mount a man-in-the-middle attack to agree upon two separate keys with the users without the users being aware of this.

Authenticated Diffie-Hellman key agreement allows a pool of users within a large and completely insecure public network to establish a common secret key and ensures each user that no other principal aside from these specifically identified group of users can possibly learn the value of a particular secret key. This is *implicit key authentication* and the protocol is called authenticated key agreement (AK) protocol. Additionally, the authenticated key agreement protocols are designed to ensure the entities that they are indeed sharing this secret key with each other. This property is called *explicit key authentication* and the protocol is said to be authenticated key agreement with key confirmation (AKC) protocol. Over the years, a number of security properties have been seen to be important in key agreement protocols and different approaches have been developed to solve the problem. Standard key derivation, message authentication code (MAC), digital signature scheme *etc.* are basic tools used to authenticate a key agreement.

There are a very few key agreement protocols that have concrete security proofs against active adversaries in a well defined security model. We can classify the key agreement protocols into two categories:

– Certificate-based and

– ID-based.

The certificate-based protocols work by assuming that each entity has a static (long term) public/private key pair, and each entity knows the public key of each other entity. The static public keys are authenticated via certificates issued by a certifying authority (CA) by binding users' identities to static keys. When two entities wish to establish a session key, a pair of ephemeral (short term) public keys are exchanged between them. The ephemeral and static keys are then combined in a way so as to obtain the agreed session key. The authenticity of the static keys provided by signature of CA assures that only the entities who posses the static keys are able to compute the session key. Thus the problem of authenticating the session key is replaced by the problem of authenticating the static public keys which is solved by using CA, a traditional approach based on a public key infrastructure (PKI).

However, in a certificate-based system, the participants must first verify the certificate of the user before using the public key of the user. In 1984, Shamir [105] proposed the idea of ID-based cryptosystem where the identity information of a user functions as his public key. A private key generator (PKG), sometimes also referred to as key generation center (KGC), which is trusted by all users is responsible for the generation of users' corresponding private keys. Shamir gave a practical ID-based signature scheme and asked for ID-based encryption to simplify key management procedures in certificate-based public key infrastructure. A few key agreement protocols have been developed based on Diffie-Hellman and Shamir's key setup idea [61, 97]. Recently, Cocks [47] and Boneh and Franklin [24] have proposed two ID-based encryption schemes which potentially allow the replacement of a PKI with a system where ones identity becomes the public key and a trusted PKG helps to generate users' private key. Cocks' scheme is based on the Quadratic Residuosity

problem, whilst that of Boneh and Franklin relies on the Weil Pairing. Shortly after that, many ID-based cryptographic protocols were developed (see [51] for a survey) based on pairings and is currently an area of very active research.

Several variations of the Diffie-Hellman [49] protocol and Joux [66] protocol have been suggested to incorporate authentication and a trial and error approach has been adopted to provide informal security analysis of the key agreement protocols. However, most of these protocols were broken and some of these protocols have flaws that came to light years after its proposal. The main problem were that appropriate threat models and the goals of secure AK and AKC protocols lacked formal definitions. It is extremely important both to correctly define the security model and to prove the security of any proposed implementation in that model.

Bellare and Rogaway [15] first consider a formal treatment for provable security of protocols in two party setting. Adapting their work, Blake-Wilson, Johnson and Menezes [17] developed a security model for distributed computing and provided rigorous definitions of the goals of secure AK and AKC protocols within this model. They proposed concrete AK and AKC protocols that were proven to be secure within this framework in the random oracle model.

Bellare, Canetti and Krawczyk [11] introduced a modular approach to design and analyze key agreement protocols. They achieved the modularity by applying a protocol translation tool, called an authenticator/compiler to protocols proven secure in a much simplified adversarial setting where authentication of communication links is not required.

Based on these works, Bresson *et al.* [32, 33, 35] introduced further refinements and defined a sound formalization for the authenticated key agreement and provided provably secure protocols within this model. This is an important step and has been used to analyze key agreement protocols, both in the two party and multi party setting.

We provide below a survey of previous work on key agreement protocols arranged according to the following three categories.

1. Two-party key agreements
2. Three-party key agreements
3. Multi-party key agreements

3.1.1 Two-party key agreement

Numerous Diffie-Hellman based AK and AKC protocols have been designed to add authentication (and key confirmation) to the Diffie-Hellman protocol; however, many have subsequently been found to have flaws. One of the well-known authenticated key agreement (AK) protocol in the Diffie-Hellman family is MTI protocol by Matsumoto, Takashima and Imai [85]. They designed three infinite families of key agreement protocols to provide implicit key authentication in the classical Diffie-Hellman key agreement protocol. However, the security analysis against active adversary is only heuristic.

Law *et al.* [79] found small subgroup attack and unknown key share attack on these protocols. Small subgroup attack forces the shared secret session key to be one of a small and known subset of points of the group. If the order n of the underlying group is not prime, say $n = mt$ where $t > 1$ is small, then small subgroup attack can be launched. Unknown key share attack enables two entities A, B to share a session key K even though B believes that he share the secret K with another entity, say, E while E does not know the value K. Law *et al.* [79] presented an efficient authenticated key agreement protocol, often called MQV protocol. The security analysis of MQV protocol against active adversary is also heuristic. Both MTI and MQV family of protocols are certificate-based.

There are many ID-based key agreement protocols based on pairing. Scott [104] proposed an ID-based key agreement protocol where each user selects his own personal identity number (PIN) and a trusted PKG issues each user an individual secret associated with the identity of corresponding user. A value is calculated from both the individual secret and PIN number and placed inside a hardware token. The individual secret can be reconstructed from their memorized PIN number, identity and token.

Another ID-based authenticated key agreement was proposed by Smart [111] that combines the idea of Boneh and Franklin [24] with the tripartite Diffie-Hellman protocol of Joux [66]. The scheme uses weil pairing and requires all users involved in the key agreement to be clients of the same PKG. The protocol allows efficient ID-based escrow facility for sessions that enables low enforcement agencies to decrypt messages encrypted with the session keys, after having obtained the necessary warrants.

Chen and Kudla [43] developed an ID-based authenticated key agreement protocol more efficient than Smart's protocol [111]. They have suggested a mechanism to turn escrow off which can also be applied to Smart's protocol [111] (the escrow-free environment may be desirable for personal communications the users wish to keep confidential even from the PKG). They also provided a modification that allows key agreement between users under different PKGs.

None of the two party key agreement protocols by Scott [104], Smart [111] and Chen and Kudla [43] were broken, although heuristic arguments are adopted to prove their security against active adversary. Shim [106] presented an ID-based key agreement protocol. However, Sun and Heish [115] showed that Shim's key agreement protocol is insecure against the man-in-the-middle attack.

Another efficient ID-based authenticated key agreement protocol was proposed by McCullagh and Barreto [86] that can be used in either escrow or escrow-free mode. They also developed a scheme for key agreement between clients of different PKGs. The scheme is twice as efficient as the scheme in [43] without precomputation. Later, Xie [118] pointed out a flaw in it and removed this flaw by suggesting modifications for the protocol. Recently, Choo [46] showed that both the scheme and its modified variant are not secure if the adversary is allowed to reveal non-partner players who had accepted the same session key.

Jeong *et al.* [65] proposed three simple single-round two-party key agreement protocols with detail security analysis in the security model of [32].

Some important two-party key agreement protocols are as follows.

1. Diffie-Hellman Key Agreement
2. MTI Key Agreement
3. MQV Key Agreement
4. Jeong, Katz and Lee's Key Agreement
5. Smart's ID-Based Key Agreement
6. Scott's ID-Based Key Agreement
7. Chen and Kudla's ID-Based Key Agreement
8. McCullagh and Berreto's ID-Based Key Agreement

The first proposal for unauthenticated two-party key agreement is the Diffie-Hellman (DH) key agreement [49]. To provide authentication in this protocol, MTI key agreements [85] are proposed. Later, flaws have been detected in MTI protocols and MQV [79] key agreements are introduced. Both MTI and MQV protocols use certificate-based authentication mechanism. However, security analysis is only heuristic mainly because of the lack of proper security model at that time. Recently, Jeong, Katz and Lee [65] proposed three simple variant of Diffie-Hellman key agreement protocol and provide a concrete security analysis in a standard security framework. Their schemes use message authentication code (MAC) for providing the authentication in the original DH protocol.

Among the ID-based two-party key agreement, the notable works are done by Scott [104], Smart [111] and Chen and Kudla [43], none of which are broken, although the security analysis are based on heuristic arguments. In Scott [104]'s ID-based scheme, each user selects his own PIN number and obtains an individual secret corresponding to his PIN from a trusted PKG. PKG stores inside a hardware token a value derived from both PIN number and the individual secret. Each user can reconstruct his individual secret from the memorized PIN, identity and token. Smart [111]'s protocol allows efficient ID-based escrow facilities for sessions that enables law enforcement agencies to decrypt encrypted messages with the session keys, after obtaining the necessary warrants. The scheme requires that all the uses involved in the key agreement, are clients of the same PKG. Chen and Kudla [43]'s ID-based authenticated protocol is more efficient than Smart's protocol. They have suggested a mechanism to turn escrow off and also provided modifications that enable key agreement between users under different PKGs. Security of both the Smart's protocol and Scott's protocol are based on the hardness on DL and CDH problem, whereas that of Chen and Kudla's protocol is based the hardness of BDH problem. In a more recent work, McCullagh and Barreto [86] provide an ID-based two-party key agreement which is more efficient than the protocol by Chen and Kudla. Security of this protocol relies on the hardness assumption on BDHI problem.

Details are in Section 3.2.

3.1.2 Three-party key agreement

In one of the breakthroughs in key agreement, Joux [66] proposed a three party single round key agreement protocol using pairings on elliptic curve. This was the first positive application of bilinear pairings in cryptography. However, just like Diffie-Hellman, Joux's protocol is unauthenticated and is susceptible to the man-in-the-middle attacks. This original scheme is not ID-based.

Al-Riyami and Paterson [3] proposed four tripartite authenticated key agreement protocols to provide implicit key authentication in Joux's protocol by incorporating certified public keys using ideas from MTI [85] and MQV [79] protocols. They argued heuristically that these protocols achieve some desirable security attributes. Later, Shim [107] made some cryptanalysis on these protocols and found that one of these protocols is insecure against man-in-the-middle attack.

In [90, 91], Nalla *et al.* proposed authenticated tripartite ID-based key agreement schemes that were broken by Chen [42] and Shim [108]. Zhang, Liu and Kim [124] developed an ID-based single round authenticated tripartite key agreement protocol, the authenticity of which is assured by Hess' [63] ID-based signature scheme and provided heuristic security analysis of the protocol against active adversary.

Some important three-party key agreement protocols are as follows.

1. Joux Key Agreement
2. Zhang, Liu and Kim's ID-Based Key Agreement

Joux's tripartite single round key agreement is important because it is the first positive application of bilinear pairing in cryptographic protocol design. Zhang, Liu and Kim enhance it to ID-based setting and provide a security analysis using heuristic arguments. Details are in Section 3.3.

3.1.3 Group key agreement

Another direction of research on key agreement is to generalize the two party key agreement to multi party setting and consider the dynamic scenario where participants may join or leave a multicast group at any given time. As a result of the increased popularity of group oriented applications, the design of an efficient authenticated group key agreement protocol has recently received much attention in the literature.

A comprehensive treatment have been made to extend the two party (and three party) key agreement protocols to multi party setting. Notable solutions have been suggested by Ingemerson *et al.* [64], Burmester and Desmedt [36], Steiner *et al.* [113] and Becker and Willie [10]. All these works assume a passive (eavesdropping) adversary, and the last three provide rigorous proofs of security.

For practical applications, efficiency is a critical concern in designing group key agreement in addition to provable security. In particular, number of rounds may be crucial in an environment

where number of group members are quite large and the group is dynamic. Handling dynamic membership changes get much attention to the current research community. A group key agreement scheme in a dynamic group must ensure that the session key is updated upon every membership change so t hat subsequent communication sessions are protected from leaving members and previous communication sessions are protected from joining members. Although this can be achieved by running any authenticated group key agreement protocol from scratch whenever group membership changes, alternative approaches to handle this dynamic membership more effectively would be clearly preferable in order to minimize cost of the rekeying operations associated with group updates.

The problem of key agreement in Dynamic Peer Groups (DPG) were studied by Steiner *et al.* [113]. They proposed a class of "generic n-party Diffie-Hellman protocols". Atenise *et al.* [4, 5] introduced authentication into the class of protocols and heuristically analyze their security against active adversary. Steiner *et al.* [114] consider a number of different scenario of group membership changes and introduced a complete key management suite CLIQUES studied specifically for DPGs which enable addition and exclusion of group members as well as refreshing of the keys. The security analysis of these schemes are heuristic against active adversaries. However, Pereira and Quisquater [99] have described a number of potential attacks, highlighting the need for ways to obtain greater assurance in the security of these protocols.

Bresson *et al.* [32, 33, 35] have recently given a formal security model for group authenticated key agreement. They provided the first provably secure protocols based on the protocols of Steiner *et al.* [113]for this setting which requires $O(n)$ rounds to establish a key among a group of n users. The initial works [33], [35] respectively consider the static and dynamic case, the security of both of which are in random oracle model following the formalized security model introduced by themselves under the computational Diffie-Hellman (CDH) assumption. They further refine in [32] the existing security model to incorporate major missing details, (*e.g.* strong corruption and concurrent sessions) and proposed an authenticated dynamic group Diffie-Hellman key agreement proven secure under the DDH assumption within this model. Their security result holds in the standard model instead of random oracle model.

Tree-based group key agreement. A different arrangement of participants for key agreement is to consider tree-based setting which requires $\log n$ rounds and has some computational advantages. Group key agreements in tree based setting are typically essential while the users are grouped into a hierarchical structure. The leaves of the tree denote individual users and each internal node corresponds to a user who acts as a representative for the set of users in the subtree rooted at that node. The representative users may have more computational resources than other users in the subtree.

There have been quite a number of tree based key agreement protocols. Kim, Perrig, Tsudik [74] extends the 2-party DH protocol to binary tree-based setting that yields a secure protocol suite, called Tree-based Group Diffie-Hellman (TGDH) which is both simple and fault tolerant. They have considered the dynamic scenario where a group of users can join or leave the group and introduced four protocols: Join, Leave, Merge and Partition. However, the security analysis against

active adversary is completely heuristic.

Nalla and Reddy [89] extends Smart's ID-based two party single round authenticated protocol to multi-party ID-based key agreement using a binary tree structure and made heuristic arguments to prove that the protocol achieves some desirable security attributes against active adversary.

In [9], we present a ternary tree based unauthenticated key agreement protocol by extending the basic Joux's protocol to multi-party setting and provide a proof of security against passive adversaries. We have further proposed in [53] a provably secure authenticated tree based group key agreement from the unauthenticated protocol of [9] and analyze the security in the model formalized by Bresson *et al.* [32]. We consider in [54] the dynamic case of the scheme in [53] that enables a user to join or leave the group at his desire retaining the tree structure with minimum key updates. We will discuss these schemes in Chapter 4 in great detail.

Constant round group key agreement. Recently, Katz and Yung [70] presented a detailed security analysis of a variant of two round unauthenticated group key agreement of Burmester and Desmedt [36](BD) in the standard model under decision Diffie-Hellman (DDH) assumption. They also provide a compiler construction, application of which makes the unauthenticated BD protocol to a provably secure three round authenticated group key agreement. Their security analysis is in the security model formalized by Bresson *et al.* [32]. The protocol achieves the nice property of forward secrecy where compromise of the long term secrets of one or more entities does not affect the security of previous session keys. However, this approach does not prevent attacks from malicious insiders as described in [68], *e.g.* existence of dishonest entities who deviates from the protocol – such as refusing to deliver messages or giving a valid signature on an incorrect message – can make the system insecure.

Choi, Hwang and Lee [45] extends the BD protocol in bilinear pairing-based setting, security of which relies on the hardness of CDH problem in the random oracle model. They have also constructed an ID-based authenticated group key agreement under Decision Hash Bilinear Diffie-Hellman (DHBDH) assumption in the random oracle model. Both the protocols achieve forward secrecy.

Becker and Willie [10] introduced the octopus protocols and the cube protocols in order to minimize the number of exchanges. They studied lower bounds for the communication complexity of contributory key distribution and established lower bounds for the total number of messages, the total number of exchanges and the total number of necessary rounds. They derived a lower bound of only one round for multi-party group key agreement protocols and leave as an open question whether any group key agreement scheme can meet this bound.

Boyd and Nieto [29] proposed a constant round authenticated group key agreement with a security proof in the random oracle model that meets Becker-Willie's lower bound of one round. However, the protocol does not provide forward secrecy. Furthermore, the protocol is computationally asymmetric as it requires a "group leader" to perform $O(n)$ encryption and $O(n)$ communication each time a group key is established.Another provably authenticated static group key agreement based on standard secret sharing techniques combined with ElGamal encryption scheme is proposed

by Bresson and Catalano [30] using asynchronous network. The security is in the standard model under DDH assumption. However, this protocol is inefficient from point of view of the computation rate and suffers from a significant communication overhead both in terms of the number of messages sent by all members during the protocol execution and in terms of the number of bits communicated throughout the protocol execution.Bresson, Chevassut, Essiari and Pointcheval [34] introduced a very efficient provably secure group key agreement in dynamic scenario suitable for restricted power devices and wireless environments. The protocol requires two rounds and is proven to be secure in the random oracle model. However, their exists a base station as a trustee.

Later, Nam, Kim, Won [94] demonstrate certain flaws in the basic setup protocol of [34] and proposed a modified version of the scheme to remove these flaws.

Nam, Kim, Yang, Won [95] investigate the problem of contributory group key agreement over combined wired/wireless networks, consisting of arbitrary number of mobile devices with limited computational resources and general-purpose stationary high-performance computer. They have designed a 3-round generalized protocol which take advantage of the difference in computing power among users and uses a 2-round unauthenticated protocol, introduced by them, as a basic building block. Their 2-round basic protocol is proven to be secure against passive adversary under DDH assumption.

A communication-efficient dynamic group key agreement protocol well suited for a lossy and high-delay unbalanced network is developed by Nam *et al.* [93]. Their protocol enables conference key agreement in an environment that consists of mobile hosts with restricted computational resources and stationary hosts with relatively high computational capabilities. They analyze their scheme in the random oracle model and prove that it is secure under factoring assumption.

More recently, Kim *et al.* [76] proposed a very efficient constant round dynamic authenticated group key agreement protocol and provide a security analysis under CDH assumption in the random oracle model.

In [55], we present a constant round group key agreement protocol which may be viewed as a variant of Burmester-Desmedt [36] protocol with better efficiency and flexibility. Chapter 5 deals with this work in details.

Some important multi-party key agreement protocols are as follows.

1. Ingemarson, Tang and Wong's Group Key Agreement
2. Steiner, Tsudik and Waidner's Group Key Agreement
3. Burmester-Desmedt Constant Round Group Key Agreement
4. Octopus Protocol and Cube Protocol
5. Boyd and Nieto's Constant Round Group Key Agreement
6. Bresson and Catalano's Constant Group Key Agreement
7. Bresson, Chevassut and Pointcheval's Dynamic Group Key Agreement

8. Bresson, Chevassut, Essiari and Pointcheval's Dynamic Group Key Agreement
9. Nam, Kim, Kim and Won's Dynamic Group Key Agreement
10. Kim, Lee and Lee's Dynamic Group Key Agreement

There are many proposals that extend the two-party DH key agreement to multi-party setting. Notable solutions have been suggested by Ingemarson, Tang and Wong [64], Steiner, Tsudik and Waidner [113] and Burmester and Desmedt [36, 37]. All these protocols have rigorous proof of security against passive adversary. The security analysis against active adversary are only heuristic mainly due to the lack of proper security model for key agreement at that time. Becker and Willie [10] proposed Octopus and Cube Protocols and study lower bounds for communication complexity for group key agreement. Boyd and Nieto [29] proposed a single round authenticated group key agreement with a security proof in the random oracle model. This protocol meets Becker and Willie's lower bound for a single round protocol. A constant round protocol is proposed by Bresson and Catalano [30] that uses secret sharing technique and ElGamal encryption scheme. Security analysis of the scheme is in a standard security model.

Another important area of research is to handle dynamic membership changes in a group key agreement protocol. The dynamic group key agreement protocols of Bresson, Chevassut and Pointcheval [33], Bresson, Chevassut, Essiari and Pointcheval [34] are supported by proper security analysis against active adversary in a standard security framework. Nam, Kim, Kim and Won's [93] dynamic group key agreement is well suited for unbalanced network environment, consisting of arbitrary number of mobile hosts with limited computational resources and stationary hosts with relatively high computational capacities. Another interesting dynamic group key agreement is proposed by Kim, Lee and Lee [76] which is very efficient. The protocol is proven to be secure in the random oracle model.

The details are in Section 3.4.

3.1.4 Organization

The rest of the chapter is organized as follows. Section 3.2 focuses on two party key agreement. Section 3.3 deals with three party key agreement. We devote Section 3.4 for key agreement in multi party scenario. Finally, we conclude in Section 3.5.

3.2 Two Party Key Agreement

3.2.1 Diffie-Hellman Key Agreement

(Diffie, Hellman [49], 1976)

Diffie-Hellman (DH) proposed the first two-party single-round key agreement protocol in their seminal paper [49] that enables the users to compute a common key from a secret key and publicly exchanged information. No user is required to hold secret information before entering the protocol and each member makes an independent contribution to the common agreed key. This work invents the revolutionary concept of public-key cryptography and is the most striking development in the history of cryptography.

- Protocol Description :

  Setup : Let G be a finite multiplicative group of some large prime order q and g be a generator of G.

  Key Agreement : Assume that two entities A and B want to decide upon a common key. They perform the following steps.

  1. User A chooses a random $a \in Z_q^*$, computes $T_A = g^a$ and sends T_A to B.
  2. User B chooses a random $b \in Z_q^*$, computes $T_B = g^b$ and sends T_B to A.
  3. User A computes $K_A = T_B^a$ and similarly user B computes $K_B = T_A^b$.

  If A and B execute the above steps honestly, they will agree upon a common key $K_{\mathsf{AB}} = K_A = K_B = g^{ab}$.

- Assumption :
  DLP is hard.

- Security :
  The protocol is unauthenticated in the sense that it is secure against passive adversaries. An active adversary can mount man-in-the-middle attack.

- Efficiency :

  Communication : Round required is 1 and group element (of G) sent per user is 1.

  Computation : Each user computes 2 exponentiations.

Note : Kim, Perrig, Tsudik [74] extends this 2-party DH protocol to binary tree-based setting that yields a secure protocol suite, called Tree-based Group Diffie-Hellman (TGDH) which is both simple and fault tolerant. They have considered the dynamic scenario where a group of users can join or leave the group and introduced four protocols: Join, Leave, Merge and Partition. However, the security analysis against active adversary is completely heuristic.

3.2.2 MTI Key Agreement

(Matsumoto, Takashima, Imai [85], 1986)

In an attempt to provide implicit key authentication in the classical Diffie-Hellman [49] key agreement protocol, Matsumoto *et al.* [85] designed three infinite families of key agreement protocols. The MTI/A0 and MTI/C0 are two special cases of these families that are much studied in the literature. Here we describe these two protocols.

- Protocol Description :

  Setup : Let G be an elliptic curve additive group of some large prime order q and P be a generator of G. A certifying authority (CA) is used in the initial setup stage to provide certificates which bind users' identities to long-term secret keys. The certificate for entity A will be of the form $\mathsf{Cert}_A = (\mathcal{I}_A|W_A|P|\mathcal{S}_{\mathsf{CA}}(\mathcal{I}_A|W_A|P))$. Here $\mathcal{I}_A$ denotes the identity string of A, $|$ denotes concatenation of data items, $\mathcal{S}_{\mathsf{CA}}$ denotes the CA's signature and w_A, $W_A = w_A P$ are respectively the long term private key, public key of A.

  Key Agreement : Two entities A and B with respective certificates Cert_A, Cert_B, long term public/private key pairs (W_A, w_A) and (W_B, w_B) perform the following steps to decide upon a common agreed key.

  (a) Protocol MTI/A0

  1. User A generates $r_A \in_R Z_q^*$, computes $R_A = r_A P$ and sends (R_A, Cert_A) to B.
  2. User B generates $r_B \in_R Z_q^*$, computes $R_B = r_B P$ and sends (R_B, Cert_B) to A.
  3. User A computes $K_A = r_A W_B + w_A R_B$. Similarly user B computes $K_B = r_B W_A + w_B R_A$.

  After an honest execution of the protocol, A and B will agree upon a common secret key $K_{\mathsf{AB}} = K_A = K_B = (w_A r_B + w_B r_A)P$.

  (b) Protocol MTI/C0

  1. User A generates $r_A \in_R Z_q^*$, computes $T_A = r_A W_B$ and sends T_A to B.
  2. User B generates $r_B \in_R Z_q^*$, computes $T_B = r_B W_A$ and sends T_B to A.
  3. User A computes $K_A = w_A^{-1} r_A T_B$ and similarly user B computes $K_B = w_B^{-1} r_B T_A$.

  After an honest execution of the protocol, A and B decide upon the common secret key $K_{\mathsf{AB}} = K_A = K_B = r_A r_B P$.

  Assumption :
  DLP is hard.

  Security :
  It is heuristically argued that the protocols achieve implicit key authentication. Law *et al.* [79] pointed out flaws in the protocols. They proved that the MTI/A0 and MTI/C0 families of protocols respectively are vulnerable to the small subgroup attack and unknown key share attack and presented an efficient authenticated key agreement protocol, often called MQV protocol that withstands these attacks. MTI/A0 protocol does not

provide forward secrecy since an adversary who learns w_A, w_B can compute all session keys established by A and B.

Efficiency :

Communication : Round required is 1, group element (of G) sent per user is 1.

Computation : In MTI/A0 protocol, each user computes 3 scalar multiplications and 1 addition in G. In MTI/C0 protocol, each user computes 2 scalar multiplications in G, 1 inverse in Z_q^* and 1 multiplication in Z_q^*.

3.2.3 MQV Key Agreement

(Law, Menezes, Qu, Solinas, Vanstone [79], 1998)

- Protocol Description :

Setup : The setup is same as in 3.2.2 for MTI protocol. We denote by f the bit length of q, *i.e.* $f = \lfloor \log_2 q \rfloor + 1$. For a finite elliptic curve point $Q \in G$, $\overline{Q}$ is defined as follows. Let x be the x-coordinate of Q, and $\overline{x}$ be the integer obtained from the binary representation of x. Then $\overline{Q}$ is defined to be the integer $(\overline{x} \bmod 2^{\lceil f/2 \rceil}) + 2^{\lceil f/2 \rceil}$. Observe that $(\overline{Q} \bmod q) \neq 0$.

Key Agreement : Two entities A and B with respective certificates Cert_A, Cert_B, long term public/private key pairs (W_A, w_A) and (W_B, w_B) perform the following steps to decide upon a common agreed key.

1. User A generates $r_A \in_R Z_q^*$, computes $R_A = r_A P$ and sends (R_A, Cert_A) to B.
2. User B generates $r_B \in_R Z_q^*$, computes $R_B = r_B P$ and sends (R_B, Cert_B) to A.
3. User A computes $s_A = r_A + \overline{R}_A w_A \bmod q$ and $K_A = s_A(R_B + \overline{R}_B W_B)$.
4. User B computes $s_B = r_B + \overline{R}_B w_B \bmod q$ and $K_B = s_B(R_A + \overline{R}_A W_A)$.

If A and B follow the protocol, they will agree upon a common secret key $K_{\mathsf{AB}} = K_A = K_B = s_A s_B P = (r_A r_B + r_A w_B \overline{R}_B + r_B w_A \overline{R}_A + w_A w_B \overline{R}_A \overline{R}_B)P$.

Assumption :

DLP is hard.

Security :

The protocol possess the security attributes of known key security, forward secrecy, key compromise impersonation and key control. However, the security analysis is only heuristic. Later, Kaliski [69] observed that the protocol does not possess the unknown key share attribute.

Efficiency :

Communication : Round required is 1, group element (of G) sent per user is 1.

Computation : Each user computes 3 scalar multiplications and 1 addition in G. Since the expression for $\overline{R}_A$ uses half the bits of the x-coordinate of R_A, the scalar multiplication $\overline{R}_A w_A$ can be done in half the time of a full scalar multiplication. Hence

the work required by each entity is 2.5 full scalar multiplications. The on-line work required by each entity is only 1.5 scalar multiplications as $r_A P$ can be computed off-line. These result increased efficiency in key computation without affecting the security of the protocol.

3.2.4 Jeong, Katz and Lee's Key Agreement

(Jeong, Katz, Lee [65], 2004)

Jeong *et al.* [65] proposed three single-round key agreement schemes $\mathcal{TS}1$, $\mathcal{TS}2$, $\mathcal{TS}3$ which are simple variants of DH key agreement. They proved that the security of the scheme $\mathcal{TS}1$ and $\mathcal{TS}2$ are based on CDH assumption in the random oracle model whereas the security of the scheme $\mathcal{TS}3$ is based on DDH assumption in the standard model. The security analysis is in the security model as defined in [13, 15, 32]. The scheme $\mathcal{TS}1$ does not provide forward secrecy whilst both the schemes $\mathcal{TS}2$, $\mathcal{TS}3$ provide forward secrecy as well as key independence. We describe the scheme $\mathcal{TS}3$ below.

- Protocol Description :

  Setup : Let $G = \langle g \rangle$ be a multiplicative group of some large prime order q and $H : \{0,1\}^* \rightarrow \{0,1\}^k$ $(k = |q|)$ be a cryptographic hash function. We assume that $x_i, y_i = g^{x_i}$ are respectively the private, public key pair of an entity P_i. We also assume that the entities can be ordered by their names (*e.g.* lexicographically) and write $P_i < P_j$ to denote this ordering.

  Key Agreement : Assume that two entities P_i, P_j wants to establish a session key and $P_i < P_j$. They perform the following steps.

  1. User P_i first computes $K_{i,j} = y_j^{x_i}$ that it will use as a key for a message authentication code ($K_{i,j}$ may need to be hashed before being used). Then P_i chooses an ephemeral key $\alpha_i \in Z_q^*$ at random, computes a tag $\tau_i \leftarrow \mathsf{Mac}_{k_{i,j}}(i|j|g^{\alpha_i})$ and sends $g^{\alpha_i}|\tau_i$ to P_j.
  2. Similarly, user P_j computes a key $K_{j,i} = y_i^{x_j}$ for a message authentication code, chooses an ephemeral key $\alpha_j \in Z_q^*$ at random, computes a tag $\tau_j \leftarrow \mathsf{Mac}_{k_{j,i}}(j|i|g^{\alpha_j})$ and sends $g^{\alpha_j}|\tau_j$ to P_i.
  3. User P_i, on receiving the message, verifies the tag using $k_{i,j}$. If verification fails, no session key is computed. Otherwise, P_i computes a session key $K_i = (g^{\alpha_j})^{\alpha_i}$ with session identifier $\mathsf{sid}_i = g^{\alpha_i}|\tau_i|g^{\alpha_j}|\tau_j$.
  4. Similarly, P_j verifies the tag of the received message using $k_{j,i}$. If verification fails, no session key is computed. Otherwise, P_j computes a session key $K_j = (g^{\alpha_i})^{\alpha_j}$ with session identifier $\mathsf{sid}_j = g^{\alpha_j}|\tau_j|g^{\alpha_i}|\tau_i$.

  If P_i, P_j follow the above steps, they will agree upon a common secret key $K_{i,j} = K_i = K_j = g^{\alpha_i \alpha_j}$ with a common session identifier $\mathsf{sid}_i = \mathsf{sid}_j$.

- Assumption :
  DDH problem is hard and the message authentication code (MAC) used in the protocol is strongly unforgeable.

- Security :
  The protocol is proven to be secure in the standard model using the security model as defined in [13, 32] instead of using heuristic arguments. The protocol provides forward secrecy and key independence assuming that the MAC is secure and DDH problem is hard.

- Efficiency :

  Communication : Round required is 1 and total message size communicated per user is $|q| + |\mathsf{Mac}|$.

  Computation : Each user computes 3 modular exponentiations and 1 MAC computation.

3.2.5 Smart's ID-Based Key Agreement

(Smart [111], 2002)

Smart proposed an ID-based authenticated key agreement protocol by combining the ideas from [24, 66, 79]. The scheme requires that all users involved in the key agreement are clients of the same Private Key Generator (PKG).

- Protocol Description :

  Setup : Suppose G_1, G_2, e are same as defined in Definition 2.2.11 of cryptographic bilinear maps. The PKG chooses a secret key $s \in Z_q^*$ and sets $P_{pub} = sP$. Let $H_1 : \{0,1\}^* \rightarrow G_1^*$ be a Map-to-point hash function. The master key of PKG is s and the global public key is P_{pub}. The system parameters and master public key are distributed to the users through a secure authenticated channel.

  Extract : Given a public identity $\mathsf{ID} \in \{0,1\}^*$, the PKG computes the public key $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathsf{G_1}$ and generates the associated private key $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$.

  Key Agreement : Let two users A and B with public keys respective $Q_A = H_1(\mathsf{ID}_A)$ and $Q_B = H_1(\mathsf{ID}_B)$ decide to agree upon a common secret key. They perform the following operations.

  1. User A chooses an ephemeral key $a \in_R Z_q^*$, computes $T_A = aP$ and sends T_A to B.
  2. User B chooses an ephemeral key $b \in_R Z_q^*$, computes $T_B = bP$ and sends T_B to A.
  3. User A computes $K_A = e(aQ_B, P_{pub})\ e(S_A, T_B)$ where $S_A = sQ_A$ is the long term secret key of A sent by the PKG on submitting A's public identity.
  4. User B computes $K_B = e(bQ_A, P_{pub})\ e(S_B, T_A)$ where $S_B = sQ_B$ is the long term secret key of B sent by the PKG on submitting B's public identity.

5. After an honest execution of the above steps, both A and B will share the common agreed key $K_{\mathsf{AB}} = K_A = K_B = e(aQ_B + bQ_A, P_{pub})$.

- Assumption :
  The classical DLP and CDH problem are hard.

- Security :
  It is heuristically argued that the protocol posses the security properties: mutual implicit key authentication, known key security, partial forward secrecy, imperfect key control, key compromise impersonation and unknown key-share resilience. Smart also proposed in the paper an ID-based authenticated key agreement protocol with key confirmation property. Shim [106] discussed that Smart's protocol does not posses perfect forward secrecy and proposed a modified scheme which in turn is proven to be insecure against man-in-the-middle attack by Sun and Hsieh [115].

- Efficiency :

  Communication : Round required is 1, group element (of G_1) sent per user is 1.

  Computation : Each user computes 1 scalar multiplication in G_1, 2 pairing computations, 1 multiplication in G_2 and 1 Map-to-point hash operation. Additionally, the PKG requires to compute 1 Map-to-point hash operation, 1 scalar multiplication per client and also 1 scalar multiplication to generate P_{pub}.

Note : The protocol allows efficient ID-based escrow facility for sessions that enables law enforcement agencies to decrypt messages encrypted with the session keys, after having obtained the necessary warrants. Nalla and Reddy [89] extends this protocol to multi-party ID-based key agreement using a binary tree structure and made heuristic arguments to prove that the protocol achieves some desirable security attributes.

3.2.6 Scott's ID-Based Key Agreement

(Scott [104], 2002)

Scott proposed an ID-based scheme where each user selects their own PIN number and a trusted PKG issues each user an individual secret associated with the identity of corresponding user. A value is calculated from both the individual secret and PIN number and placed inside a hardware token. The individual secret can be reconstructed from their memorizes PIN, identity and token.

- Protocol Description :

  Setup : Same as in section 3.2.5 for Smart's protocol.

  Extract : For individual clients to register with the PKG, they must prove their identity. Given the public identity $\mathsf{ID}_A \in \{0,1\}^*$ of an user A, the PKG computes the public key

$Q_A = H_1(\mathsf{ID}_A) \in G_1$, generates the associated private key $S_A = sQ_A$. After authenticating himself, the user A receives S_A, calculates $\alpha_A Q_A$ where α_A is the desired secret PIN of A, subtracts the two and places the value $(s-\alpha_A)Q_A$ inside a hardware token. A memorizes α_A and then discard the secret S_A which it can reconstruct using the token, PIN and identity.

Key Agreement : Let two users A, B with respective public keys Q_A, Q_B want to agree upon a common session key. They executes the following steps.

1. A picks $a \in_R Z_q^*$, computes $T_A = e((s-\alpha_A)Q_A + \alpha_A Q_A, Q_B)^a$ and sends T_A to B.
2. B picks $b \in_R Z_q^*$, computes $T_B = e((s-\alpha_B)Q_B + \alpha_B Q_B, Q_A)^b$ and sends T_B to A.
3. A computes $K_A = T_B^a$ and similarly user B computes $K_B = T_A^b$.

If both A and B follow the protocol, they will agree upon a common key $K_{\mathsf{AB}} = K_A = K_B = e(Q_A, Q_B)^{sab}$. (Scott used Tate pairing of order r in their protocol and the ephemeral keys a, b chosen respectively by A, B are less than r.)

- Assumption :
  The classical DLP and CDH problem are hard.

- Security :
  The author informally argued that the scheme is secure against impersonation attack.

- Efficiency :

  Communication : Round required is 1, group element (of G_2) sent per user is 1.

  Computation : Each user computes 1 scalar multiplication in G_1, 1 pairing computation, 2 exponentiation in G_2, 1 Map-to-point hash operation and 1 subtraction in G_1. Additionally, the PKG requires to compute 1 Map-to-point hash operation and 1 scalar multiplication per client and also 1 scalar multiplication to generate P_{pub}.

3.2.7 Chen and Kudla's ID-Based Key Agreement

(Chen, Kudla [43], 2002)

In this work, Chen and Kudla presented an identity-based authenticated key agreement protocol more efficient than Smart's protocol [111] and analyzed the security using formal security model of [13, 14]. They have suggested a mechanism to turn escrow off which can also be applied to Smart's protocol [111] (the escrow-free environment may be desirable for personal communications the users wish to keep confidential even from the PKG). They also provided another modification that allows key agreement between users under different PKGs.

- Protocol Description :

  Setup : Same as for Smart's protocol in section 3.2.5.

Extract : Same as in section 3.2.5

Key Agreement : Users A, B with public keys Q_A, Q_B respectively performs the following steps to decide upon a common secret key.

1. User A chooses an ephemeral key $a \in_R Z_q^*$, computes $T_A = aQ_A$ and sends T_A to B.
2. User B chooses an ephemeral key $b \in_R Z_q^*$, computes $T_B = bQ_B$ and sends T_B to A.
3. User A computes $K_A = e(S_A, T_B + aQ_B)$ and similarly user B computes $K_B = e(S_B, T_A + bQ_A)$.

After an honest execution of the protocol, A and B agree upon a common session key $K_{\mathsf{AB}} = K_A = K_B = e(Q_A, Q_B)^{s(a+b)}$.

- Assumption :
  BDH problem is hard.

- Security :
  The authors adopt the security model of [13, 14] and prove the security of their protocol in the random oracle model assuming that the adversary makes no Reveal query. It is heuristically argued that the protocol achieves the security properties: partial forward secrecy, imperfect key control, unknown key share resilience and key compromise impersonation.

- Efficiency :

  Communication : Round required is 1, group element (of G_1) sent per user is 1.

  Computation : Each user computes 2 scalar multiplications in G_1, 1 pairing computation, 2 exponentiation in G_2 and 2 Map-to-point hash operation. Additionally, the PKG requires to compute 1 Map-to-point hash operation and 1 scalar multiplication per user and 1 scalar multiplication to generate P_{pub}.
  Clearly, the scheme is efficient compared to Smart's protocol [111].

3.2.8 McCullagh and Barreto's ID-Based Key Agreement

(McCullagh, Barreto [86], 2004)

McCullagh and Barreto [86] designed an efficient ID-based authenticated key agreement protocol that can be used in either escrow or escrow-free mode and also a scheme for key agreement between clients of different PKGs. The scheme is twice as efficient as the scheme in [43] without precomputation. We describe below the key agreement scheme with escrow.

- Protocol Description :

  Setup : The setup is same as in Smart's protocol in section 3.2.5.

  Extract : The PKG verifies the on line public identity ID_A of A and computes $a = H_1(\mathsf{ID}_A)$ and $Q_A = (a+s)P$. Q_A is the public key of A, which can also be computed as $aP + P_{pub}$. The PKG then calculates A's private key as $S_A = (a+s)^{-1}P$.

Key Agreement : Two entities A, B perform the following steps to agree upon a common key.

1. User A chooses an ephemeral key $x_a \in_R Z_q^*$, computes $T_A = x_a Q_B$ and sends T_A to B.
2. User B chooses an ephemeral key $x_b \in_R Z_q^*$, computes $T_B = x_b Q_A$ and sends T_B to A.
3. User A computes $K_A = e(T_B, S_A)^{x_a}$ and similarly user B computes $K_B = e(T_A, S_B)^{x_b}$.

If A and B follow the protocol, they will the same shared secret key $K_{\mathsf{AB}} = K_A = K_B = e(P, P)^{x_a x_b}$.

- Assumption :
  BDHI problem is hard.

- Security :
  The security analysis of the protocol is in the security model of [13, 14] assuming that the adversary makes no Reveal query and using random hash oracle. Heuristic arguments show that the protocol achieves the security properties: known key security, key compromise impersonation, forward secrecy, unknown key share resilience and key control. Later, Xie [118] pointed out that a malicious adversary can successfully launch a key compromise attack. He removed this flaw by suggesting modifications for the protocol. Recently, Choo [46] showed that both the scheme and its modified variant are not secure if the adversary is allowed to reveal non-partner players who had accepted the same session key.

- Efficiency :

  Communication : Round required is 1 and group element (of G_1) sent per user is 1.

  Computation : Each user computes 1 scalar multiplication in G_1, 1 pairing computation, 1 exponentiation in G_2 and 1 hash (H_1) operation. Additionally, the PKG requires to compute 1 hash operation and 1 scalar multiplication per user and also 1 scalar multiplication to generate P_{pub}.
  The scheme is efficient than the schemes in [43, 111].

3.3 Three Party Key Agreement

3.3.1 Joux Key Agreement

(Joux [66], 2000)

Joux introduced a very simple and elegant tripartite key agreement protocol which makes use of bilinear pairing on elliptic curves that requires just one broadcast per entity. This was a major breakthrough in key agreement and was the first positive application of pairing in cryptography. Following this work, a number of pairing-based protocols were proposed.

- Setup : Let G_1, G_2, e be as defined in Smart's protocol in section 3.2.5. P is a generator of the additive group G_1 of order q, G_2 is a multiplicative group of same order q and e is the bilinear map from $G_1 \times G_1 \rightarrow G_2$.

- Protocol Description :

  Key Agreement : Consider three entities A, B, C decide to agree upon a common secret key. They perform the following steps.

  1. User A chooses $a \in_R Z_q^*$, computes aP and sends aP to both B and C.
  2. User B chooses $b \in_R Z_q^*$, computes bP and sends bP to both A and C.
  3. User C chooses $c \in_R Z_q^*$, computes cP and sends cP to both A and B.
  4. User A computes $K_A = e(bP, cP)^a$, user B computes $K_B = e(aP, cP)^b$ and user C computes $K_C = e(aP, bP)^c$.

  If A, B, C execute the above steps honestly, then they will agree upon a common key $K_{\mathsf{ABC}} = K_A = K_B = K_c = e(P, P)^{abc}$.

- Assumption :
  BDH problem is hard.

- Security :
  Joux's protocol is unauthenticated in the sense that it is secure against a passive adversary and suffers from the man-in-the-middle attack in presence of an active adversary.

- Efficiency :

  Communication : Round required is 1, group element (of G_1) sent per entity is 1.

  Computation : Each entity computes 1 scalar multiplication in G_1, 1 pairing computation and 1 exponentiation in G_2.

Note : Al-Riyami and Paterson [3] proposed four tripartite authenticated key agreement protocols to provide implicit key authentication in Joux's protocol by incorporating certified public keys using ideas from MTI [85] and MQV [79] protocols. They argued heuristically that these protocols achieve some desirable security attributes. Later, Shim [107] made some cryptanalysis on these protocols and found that one of these protocols is insecure against man-in-the-middle attack.

3.3.2 Zhang, Liu and Kim's ID-Based Key Agreement

(Zhang, Liu, Kim [124], 2002)

In this work, an ID-based one round authenticated tripartite key agreement protocol is proposed by incorporating Hess' [63] ID-based signature.

- Protocol Description :

Setup : Let G_1, G_2 be two groups of some large prime order q. We take G_1 to be an additive group and G_2 to be a multiplicative group. It is assumed that DL problem is hard in both G_1, G_2. Let P be a generator of G_1. We also consider a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The PKG chooses a secret key $s \in Z_q^*$ and sets $P_{pub} = sP$. Let $H_1 : \{0,1\}^* \rightarrow G_1^*$ be a Map-to-point hash function. We also consider a cryptographic hash function $H : G_1 \rightarrow Z_q^*$. The master key of PKG is s and the global public key is P_{pub}.

Extract : Given a public identity $\mathsf{ID} \in \{0,1\}^*$, the PKG computes the public key $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in \mathsf{G_1}$ and generates the associated private key $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$.

Key Agreement : Three entities A, B, C with respective static (or long term) public keys $Q_A = H_1(\mathsf{ID}_A)$, $Q_B = H_1(\mathsf{ID}_B)$, $Q_C = H_1(\mathsf{ID}_C)$ and respective static (or long term) private keys $S_A = sQ_A$, $S_B = sQ_B$, $S_C = sQ_C$ perform the following steps to agree upon a common key.

1. User A chooses an ephemeral key $a \in Z_q^*$ at random, computes $P_A = aP$, $T_A = H(P_A)S_A + aP$ and sends (P_A, T_A) to both B and C.
2. User B chooses an ephemeral key $b \in Z_q^*$ at random, computes $P_B = bP$, $T_B = H(P_B)S_B + bP$ and sends (P_B, T_B) to both A and C.
3. User C chooses an ephemeral key $c \in Z_q^*$ at random, computes $P_C = cP$, $T_C = H(P_C)S_C + cP$ and sends (P_C, T_C) to both A and B.
4. User A verifies $e(T_B + T_C, P) = e(H(P_B)Q_B + H(P_C)Q_C, P_{pub})\ e(P_B, P_B)\ e(P_C, P_C)$ and computes $K_A = e(P_B, P_C)^a$ only if the verification succeeds.
5. User B verifies $e(T_A + T_C, P) = e(H(P_A)Q_A + H(P_C)Q_C, P_{pub})\ e(P_A, P_A)\ e(P_C, P_C)$ and computes $K_B = e(P_A, P_C)^b$ only if the verification succeeds.
6. User C verifies $e(T_B + T_A, P) = e(H(P_B)Q_B + H(P_A)Q_A, P_{pub})\ e(P_B, P_B)\ e(P_A, P_A)$ and computes $K_C = e(P_B, P_A)^c$ only if the verification succeeds.
   If the entities A, B, C follow the protocol, they will agree upon a common session key $K_{\mathsf{ABC}} = K_A = K_B = K_C = e(P, P)^{abc}$.

- Assumption :
  BDH problem and Weak-DH are hard (Hess' ID-based signature is secure under Weak-DH assumption).

- Security :
  Heuristic arguments shows that the protocol achieves the security attributes: implicit key authentication, known session key security, perfect forward secrecy, no key compromise impersonation, no unknown key share and no key control assuming that the underlying signature scheme (Hess's signature) is secure and BDH problem is hard.

- Efficiency :

  Communication : Round required is 1 and group elements (of G_1) sent per user is 2.

Computation : Each user computes 5 scalar multiplications in G_1, 5 pairing computations, 2 multiplications in G_2, 2 Map-to-point hash operation (H_1) and 2 hash function (H) evaluation. Additionally, the PKG requires to compute 1 Map-to-point hash operation and 1 scalar multiplication per user and also 1 scalar multiplication to generate P_{pub}.

Note : We present in [9] a ternary tree based unauthenticated key agreement protocol by extending the basic Joux's protocol to multi-party setting and provide a proof of security against passive adversaries. We have further proposed in [53] a provably secure authenticated tree based group key agreement from the unauthenticated protocol of [9] and analyze the security in the model formalized by Bresson *et al.* [32]. We consider in [54] the dynamic case of the scheme in [53] that enables a user to join or leave the group at his desire retaining the tree structure with minimum key updates. We demonstrate these schemes in Chapter 4 covering rigorous security analysis of each of the schemes.

3.4 Multi Party Key Agreement

3.4.1 Ingemarsson, Tang and Wong's Group Key Agreement

(Ingemarsson, Tang, Wong [64], 1982)

Since the publication of 2-party DH key exchange in 1976, various solutions have been proposed to extend DH key exchange to multi-party key distribution. Notable solutions have been proposed by Ingemarsson *et al.* [64] in 1982. We describe one of the families of protocols proposed by them.

- Protocol Description :

  Setup : Let $G = \langle g \rangle$ be a multiplicative group of some large prime order q.

  Key Agreement : Assume that n participants $U_1, \ldots, U_n$ want to agree upon a common key. The participants must be arranged in a logical ring. In a given round, every participant raises the previously received intermediate key value to the power of its own exponent and forwards the result to the next participant. The actual protocol is as follows.

  1. In round 1, user U_i, $1 \leq i \leq n$, chooses a random $x_i \in Z_q^*$, computes g^{x_i} and forwards it to $U_{(i+1) \bmod n}$.
  2. In round $k \in [1, n-1]$, user U_i, $1 \leq i \leq n$ computes $g^{\Pi\{x_j | j \in [(i-k) \bmod n, i]\}}$ and forwards it to $U_{(i+1) \bmod n}$.
  3. After $n-1$ rounds, all user agree upon a common session key $K = g^{x_1 x_2 \ldots x_n}$.

- Assumption :
  DDH problem is hard.

- Security :
  This protocol falls into the class of "natural" extensions of DH 2-party protocol and is secure against a passive adversary.

- Efficiency :

  Communication : Rounds required are $n-1$, messages sent per user are $n-1$.

  Computation : Each user computes n modular exponentiations.

3.4.2 Steiner, Tsudik and Waidner's Group Key Agreement

(Steiner, Tsudik, Waidner [113], 1996)
Steiner *et al.* [113] defined a class of "generic n-party DH protocols" for which they have showed that security against passive adversaries is based on the intractability of the DDH problem. We describe three Group Diffie-Hellman (GDH) protocols: GDH.1, GDH.2 and GDH.3 introduced by them.

- Protocol Description :

  Setup : Let $G = \langle g \rangle$ be a cyclic group of a large prime order q.

  Key Agreement : Let users $U_1, \ldots U_n$ wishing to agree upon a common session key. They proceed the following steps.

(a) Protocol GDH.1

The protocol executes in $2(n-1)$ rounds and consists of two stages: up flow and down flow. The purpose of up flow stage is to collect contributions from all group members. The actual protocol execution is as follows.

1. *(Up flow)* In round i, $1 \leq i \leq n-1$, user U_i selects a random $x_i \in Z_q^*$ and sends $\{g^{\Pi(x_k|k\in[1,j])} \mid j \in [1,i]\}$ to U_{i+1}.
2. *(Down flow)* In round $(n-1+i)$, $i \in [1, n-1]$, user U_{n-i} sends $\{g^{\Pi(x_k|k\notin[i,j])} \mid j \in [1,i]\}$ to user U_{n-i+1}.

If all the users follow the above steps, they will agree upon a common secret key $K = g^{x_1 x_2 \ldots x_n}$.

(b) Protocol GDH.2

The protocol executes in n rounds and consists of two stages. In the first stage ($n-1$ rounds) contributions are collected from individual group members and then, in the second stage (n-th round), the group keying material is broad-casted. The actual protocol is as follows.

1. *(Up flow)* In round i, $1 \leq i \leq n$, user U_i selects a random $x_i \in Z_q^*$ and sends $\{g^{\frac{x_1 x_2 \ldots x_i}{x_j}} | j \in [1, j]\}$ and $g^{x_1 x_2 \ldots x_i}$ to U_{i+1}.
2. *(Broadcast)* In round n, U_n selects a random $x_n \in Z_q^*$ and broadcasts $\{g^{\frac{x_1 x_2 \ldots x_n}{x_i}} | i \in [1, n[\}$ to the rest of the users.

If all the users follow the protocol, they will agree upon a common secret key $K = g^{x_1 x_2 \ldots x_n}$.

(c) Protocol GDH.3

This protocol consists of four stages. The protocol execution among n users $U_1, \ldots, U_n$ requires $n + 1$ rounds and proceeds as follows.

1. *(Up flow)* In the first stage, user U_i in the i-th round, $1 \leq i \leq n - 2$, selects a random $x_i \in Z_q^*$, computes $g^{\Pi\{x_k | k \in [1,i]\}}$ and sends it to U_{i+1}.
2. *(Broadcast)* After processing the up flow message, U_{n-1} obtains $g^{\Pi\{x_k | k \in [1,n-1]\}}$ and broadcasts this value in the second stage to the rest of the participants. This is $n - 1$-th round.
3. *(Response)* In the n-th round, each user U_i $(i \neq n)$, factors out its own exponent x_i and forwards the result to U_n. Thus user U_n receives the value $g^{\Pi\{x_k | k \in [1,n-1] \wedge k \neq i\}}$ in the n-th round. Note that factoring out x_i by U_i requires to compute its inverse x_i^{-1} which is always possible since the underlying group is of prime order.
4. *(Broadcast)* In the final stage, U_n collects all inputs from the previous stage, raises every one of them to the power x_n and broadcasts the resulting $n - 1$ values $\{g^{\Pi\{x_k | k \in [1,n] \wedge k \neq i\}} | i \in [1, n - 1]\}$ to the rest of the users.

Note that every user U_i now as a value of the form $g^{\Pi\{x_k | k \in [1,n] \wedge k \neq i\}}$ and can easily generate the intended group key $K = g^{x_1 x_2 \ldots x_n}$.

- Assumption :
  DDH problem is hard.

- Security :
  The above class of protocols are proven to be secure against passive adversaries under DDH assumption. Ateniese, Steiner, Tsudik [4, 5] studied these protocols for Dynamic Peer Groups (DPG) and provided an authentication mechanism to protocol GDH.2 with key confirmation and integrity. They adopt heuristic arguments to show that their authenticated protocols achieve certain desirable security attributes. The problem of key agreement in DPG is also studied by Steiner, Tsudik, Waidner [114]. They considered all group key agreement operations (member addition, member exclusion, mass join, mass leave) and present a concrete protocol suite, CLIQUES, which offers complete key agreement services. The security analysis against active adversary is only heuristic. However, Pereira and Quisquater [99] have described a number of potential attacks, highlighting the need for ways to obtain greater assurance in the security of these protocols.

- Efficiency :

  Communication : For GDH.1, GDH.2, GDH.3, rounds required are respectively $2(n-1)$, n, $n+1$ and the respective messages sent per user are at most 2, 1, 2. For GDH.1, each of U_1, U_n sends only 1 message.

  Computation : Modular exponentiations computed by user U_i is $i+1$ for GDH.1 and $i+1$ for GDH.2. For GDH.3, modular exponentiations computed by user U_{n-1} are 2, user U_n are $n-1$ and for all other users are 4.

3.4.3 Burmester and Desmedt's Constant Round Group Key Agreement

(Burmester, Desmedt [36], 1994)
Burmester and Desmedt presented a much more efficient key agreement protocol (BD) in group setting that requires only two rounds.

- Protocol Description :

  Setup : Let $G = \langle g \rangle$ be a multiplicative group of some large prime order q.

  Key Agreement : When n users $U_1, \ldots, U_n$ wish to establish a session key, they proceed as follows where the indices are taken modulo n so that user U_0 is U_n and user U_{n+1} is U_1.

  1. Each user U_i, $1 \leq i \leq n$, choose a random $x_i \in Z_q^*$ and broadcasts $Z_i = g^{x_i}$.
  2. Each user U_i broadcasts $X_i = \left(\frac{Z_{i+1}}{Z_{i-1}}\right)^{x_i}$.
  3. Each user U_i computes their session key as

  $$K_i = (Z_{i-1})^{nx_i} \; X_i^{n-1} \; X_{i+1}^{n-2} \; \ldots \; X_{i+n-2} \bmod q.$$

  If all the users U_i for $1 \leq i \leq n$ follow the above steps, they will agree upon the same key $K = g^{x_1x_2 + x_2x_3 + \cdots + x_nx_1}$.

- Assumption :
  DDH problem is hard.

- Security :
  The protocol is unauthenticated in the sense that it is secure against passive adversaries. The authors provide the security proof later in [37]. In [70], Katz and Yung investigate the security of a variant of BD protocol for unauthenticated group key agreement in detail and proposed a scalable compiler which transforms a secure unauthenticated group key agreement protocol into a secure authenticated group key agreement protocol preserving forward secrecy of the original protocol. They adopt the security model as formalized by Bresson *et al* [32] for security analysis.

- Efficiency :

Communication : Rounds required are 2, messages sent per user are 2.

Computation : Each user computes at most 3 (full length) modular exponentiations and $(\frac{n^2}{2} + \frac{3n}{2} - 3)$ modular multiplications.
Katz-Yung modification to BD protocol adds one more round and performs additionally 2 signature generations and $(2n - 2)$ signature verifications.

Note : Choi, Hwang and Lee [45] proposed a group key agreement protocol which is a bilinear version of the BD protocol and security of which relies on the hardness of CDH problem in the random oracle model. They have also constructed an ID-based authenticated group key agreement under Decision Hash Bilinear Diffie-Hellman (DHBDH) assumption in the random oracle model. Both the protocols achieve forward secrecy.

3.4.4 Octopus Protocol and Cube Protocol

(Becker, Willie [10], 1998)

In this work, Becker and Willie attempted to study lower bounds for the communication complexity of contributory key distribution. Their objective was to minimize the number of exchanges and to this aim, they introduced the basic octopus protocol without broadcasting which requires $2n - 4$ exchanges. They have formally described the 2^d-cube protocol with d rounds and developed 2^d-octopus protocols with $\lceil \log_2 n \rceil + 1$ rounds that makes use of the basic octopus protocol. Both these protocols use no broadcasting.

- Protocol Description :

  Setup : Let G be a finite cyclic group of some large prime order q and g be a generator of G. We further assume a bijective mapping $\phi : G \rightarrow Z_q^*$.

  Key Agreement : Suppose the participants $U_1, \ldots, U_n$ want to agree on a common key. They perform the following steps.

  (a) Octopus Protocol

  Before introducing this protocol, first consider the following Diffie-Hellman key exchange among four user A, B, C, D. Users A and B and users C and D perform a Diffie-Hellman key exchange generating keys g^{ab} and g^{cd} respectively. Subsequently, $A(B)$ sends $g^{\phi(g^{ab})}$ to $C(D)$ and $C(D)$ sends $g^{\phi(g^{cd})}$ to $A(B)$. Hence, A and C (B and D) can generate the joint key $g^{\phi(g^{ab})\phi(g^{cd})}$.
  In the octopus protocol, the participants $U_1, \ldots, U_n$ are partitioned into five groups. Four users U_{n-3}, U_{n-2}, U_{n-1} and U_n take charge of the central control. We denote these users by A, B, C, D respectively. The remaining users are distributed in four groups: $\{U_i | i \in I_A\}$, $\{U_i | i \in I_B\}$, $\{U_i | i \in I_C\}$ and $\{U_i | i \in I_D\}$ where I_A, I_B, I_C, I_D

are possibly of equal size, pairwise disjoint and $I_A \cup I_B \cup I_C \cup I_D = \{1, \ldots, n-4\}$. Now $U_1, \ldots, U_n$ can generate a group key as follows.

1. Each user $X \in \{A, B, C, D\}$ generates a joint key k_i with user U_i for all $i \in I_X$.
2. The users A, B, C, D perform the four party key exchange described above using the respective secret value $a = K(I_A)$, $b = K(I_B)$, $c = K(I_C)$ and $d = K(I_D)$, where $K(J) := \Pi_{i \in J} \phi(k_i)$ for $J \subseteq \{1, \ldots, n-4\}$. Thereafter, A, B, C, D hold the joint and later group key
$$K := g^{\phi(g^{K(I_A \cup I_B)})\phi(g^{K(I_C \cup I_D)})}.$$
3. We describe this step only for user A. The users B, C, D act correspondingly. For all $j \in I_A$, A sends $g^{K(I_B \cup I_A \setminus \{j\})}$ and $g^{\phi(g^{(I_C \cup I_D)})}$ to U_j. Now U_j calculates
$$(g^{K(I_B \cup I_A \setminus \{j\})})^{\phi(k_j)} = g^{K(I_A \cup I_B)}$$
and then generates the group key
$$K = (g^{\phi(g^{(I_C \cup I_D)})})^{\phi(g^{(K(I_A \cup I_B)})}.$$

(b) 2^d-Cube Protocol

In the cube protocol for 2^d-participants, the 2^d participants are identified with the vectors in the d-dimensional vector space $GF(2)^d$ and a basis $\vec{b}_1, \ldots, \vec{b}_d$ of $GF(2)^d$ is chosen. The protocol may be performed in d rounds as follows.

1. In the first round, every participant $\vec{v} \in GF(2)^d$ generates a random number $r_{\vec{v}}$ and performs a Diffie-Hellman key exchange with participant $\vec{v} + \vec{b}_1$ using the values $r_{\vec{v}}$ and $r_{\vec{v}+\vec{b}_1}$.
2. In the i-th round, every participant $\vec{v} \in GF(2)^d$ performs a Diffie-Hellman key exchange with participant $\vec{v} + \vec{b}_i$, where both parties use the value generated in round $i-1$ as the secret value for the key exchange.

In every round i, $1 \leq i \leq d$, the participants communicate on a maximum number of parallel edges of the d-dimensional cube in the direction $\vec{b}_i$, Thus every party is involved in exactly one Diffie-Hellman exchange per round. Furthermore, all the parties share a common key at the end of this protocol because the vectors $\vec{b}_1, \ldots, \vec{b}_d$ form a basis of the vector space $GF(2)^d$.

(c) 2^d-Octopus Protocol

In the 2^d-octopus protocol, participants act as in the octopus protocol with the only difference that 2^d instead of four users are distinguished to take charge of the central control, whereas remaining $n - 2^d$ users are partitioned into 2^d groups, *i.e.* in steps 1 and 3 of the octopus protocol, 2^d participants manage communication with the rest and in step 2 thus 2^d participants perform the cube protocol for 2^d participants.

- Assumption :
  DDH problem is hard.

- **Security :**
  It is proved that 2^d-cube protocol is secure against passive adversary. A very similar security analysis applies to 2^d-octopus protocol.

- Efficiency :

  Communication : For octopus protocol, 2^d-cube protocol and 2^d-octopus protocol, rounds required are respectively $2\lceil\frac{n-4}{4}\rceil + 2$, d and $2\lceil\frac{n-2^d}{2^d}\rceil + d$, messages sent per user are respectively $3n-4$, nd and $3(n-2^d)+2^d d$.

  Computation : Both bijection operation (ϕ) and modular exponentiation required per user are at most $3n-4$ for octopus protocol, nd for 2^d-cube protocol and $3(n-2^d)+2^d d$ for 2^d-octopus protocol.

3.4.5 Boyd and Nieto's Constant Round Group Key Agreement

(Boyd, Nieto [29], 2003)

Boyd and Nieto [29] proposed a single-round authenticated static group key agreement that meets the bound of Becker and Wille [10] for a single-round protocol and have proved the security in the random oracle model following the model established by Bellare *et al.* [11, 13, 14]. The protocol does not provide forward secrecy.

- Protocol Description :

  Setup : Consider a secure public key encryption scheme $\mathcal{PE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ where $\mathcal{K}$ is the key generation algorithm and $\mathcal{E}, \mathcal{D}$ are respectively the encryption and decryption algorithms. Let $\Sigma = (\overline{K}, \mathcal{S}, \mathcal{V})$ be a secure signature scheme with $\overline{\mathcal{K}}$ the key generation algorithm, $\mathcal{S}$ the signing algorithm and $\mathcal{V}$ the verification algorithm. The key distribution algorithm $\mathcal{G}_L$ assigns to each user U_i an encryption/decryption key pair $(e_i, d_i) \leftarrow K(1^k)$ and a signing/verification key pair $(\overline{e}_i, \overline{d}_i) \leftarrow \overline{K}(1^k)$ where k is a security parameter. Each user is provided by $\mathcal{G}_L$ an authenticated copy of the public keys of all other user. We also consider a one way hash function $h : \{0,1\} \rightarrow \{0,1\}^k$.

  Key Agreement : Let $\mathcal{U} = \{U_1, \ldots, U_n\}$ be a set of n users wishing to establish a session key. The group members $U_1, \ldots, U_n$ consists of one distinguishing member, say, U_1, called initiator and all the other members are called responders. The users perform the following steps in order to agree upon a common key.

  1. Each user U_i, chooses a random nonce $N_i \in \{0,1\}^k$.
  2. Each responder U_i, $2 \leq i \leq n$, broadcasts $U_i | N_i$ to the rest of the users.

3. The initiator U_1 encrypts N_1 for each other user U_i in $\mathcal{U}$ using public encryption key e_i and generates $\mathcal{E}_{e_i}(N_1)$ for $2 \leq i \leq n$. Then U_1 signs $\mathcal{U}|\mathcal{E}_{e_2}(N_1)|\mathcal{E}_{e_3}(N_1)|\cdots|\mathcal{E}_{e_n}(N_1)$ to compute signature $\mathcal{S}_{d_1}(\mathcal{U}|\mathcal{E}_{e_2}(N_1)|\mathcal{E}_{e_3}(N_1)|\cdots|\mathcal{E}_{e_n}(N_1))$. U_1 broadcasts $\mathcal{U}|\{\mathcal{E}_{e_i}(N_i)|2 \leq i \leq n\}|\mathcal{S}_{d_1}(\mathcal{U}|\mathcal{E}_{e_2}(N_1)|\mathcal{E}_{e_3}(N_1)|\cdots|\mathcal{E}_{e_n}(N_1))$.
4. Each user computes the conference key $K_{\mathcal{U}} = h(N_1|N_2|\cdots|N_n)$.

- Assumption :
  The public key encryption scheme and the signature scheme are secure.

- Security :
  The protocol is proven to be secure in the random oracle model following the security model of [11, 13, 14]. However, the protocol does not provide forward secrecy.

- Efficiency :

  Communication : Round required is 1, initiator's broadcast constitutes $n+1$ messages and responder's broadcast constitutes only 2 messages.

  Computation : Each responder performs only 1 signature verification, 1 decryption in a public key cryptosystem and 1 operation of one-way hash function. The initiator has a heavy burden caused by $(n-1)$ encryptions in a public key cryptosystem and 1 signature generation. The computational burden of U_1 can be reduced substantially by careful choice of public key cryptosystem.
  The computation required are substantially lower than in the proven secure generalized Diffie-Hellman protocol of Bresson *et al.* [32, 33, 35] which require for user U_i to compute $i+1$ exponentiation in addition to generating and verifying a signature.

3.4.6 Bresson and Catalano's Constant Round Group Key Agreement

(Bresson, Catalano [30], 2004)

A constant round provably authenticated static group key agreement protocol is introduced by Bresson and Catalano [30] which is based on secret sharing techniques combined with the El-Gamal encryption scheme and uses asynchronous network. Their security analysis is in the standard model under DDH assumption.

- Protocol Description :

  Setup : Let p, q be two primes such that $q|p-1$. Suppose $G\langle g\rangle$ is a subgroup of order q, $\mathcal{H}$ is a hash function modeled as a random oracle and sid be the current session identity. Let users $U_1, \ldots, U_n$ want to agree upon a common session key. Each user U_i, for $1 \leq i \leq n$, has a public key h_i and a private key x_i such that $h_i = g^{x_i} \bmod p$. We also consider a

secure signature scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ where $\mathcal{K}, \mathcal{S}, \mathcal{V}$ are respectively the key generation, signing and verification algorithm.

In a preprocessing stage, each user U_i runs the key generation algorithm $\mathcal{K}$ to obtain a couple of matching signing and verification key $(\mathsf{SK}_i, \mathsf{PK}_i)$.

Key Agreement :

1. In the first round, each user U_i chooses randomly $a_i \in G$, $r, b_{i,1}, \ldots, b_{i,n-1} \in Z_q$ and define $f_i(z) = r_i + b_{i,1}z + b_{i,2}z^2 + \cdots + b_{i,n-1}z^{n-1} \bmod q$. Now for each j, $1 \leq j \leq n$, $j \neq i$, U_i chooses $k \in Z_q$ and sets $C_{i,j} = (A_{i,j}, B_{i,j}) = (g^k \bmod p, h_j^k a_i \bmod p)$. User U_i sends U_j the value $C_{i,j}$, $f_i(j)$ and $\sigma_{i,j} = \mathcal{S}_{\mathsf{SK}_i}(C_{i,j}|f_i(j)|\mathsf{sid})$.
2. In the second round, each user U_i, on receiving all the values above, first checks the authentication (signature) of all received values. If check fails, the user aborts the protocol (U_i also aborts the protocol in case he receives less than $n-1$ tuple $(C_{j,i}, f_j(i), \sigma_{j,i})$). U_i then multiplies the received cipher texts. Let $A_i = \Pi_{j \neq i} A_{j,i} \bmod p$ and $B_i = a_i \ \Pi_{j \neq i} B_{j,i} \bmod p$. U_i decrypts the result to define the value $a_{(i)} = B_i / A_i^{x_i}$ and computes $f_i = f_i(i) + \Sigma_{j \neq i} f_j(i) \bmod q$ as his share of a $(n-1)$-degree polynomial $f(z)$ whose free term is indicated by r. User U_i sends to other users the value f_i and $w_i = \mathcal{S}_{\mathsf{SK}_i}(f_i|\mathsf{sid})$.
3. In the third round, the users interpolate $f(z)$ and retrieve r. User U_i defines its session seed as
$$\mathsf{sk}_{(i)} = a_{(i)} \ g^r \bmod p.$$
4. For confirmation, each user U_i computes $s_i = \mathcal{H}(\mathsf{sk}_{(i)}|\mathsf{sid})$ and broadcasts this value together with its signature $\gamma_i = \mathcal{S}_{\mathsf{SK}_i}(s_i|\mathsf{sid})$. If the n broad-casted values are all the same, set the final key as
$$\mathsf{sk} = \mathcal{H}(\mathsf{sk}_{(i)}).$$

- Assumption :
  DDH problem is hard and the signature scheme is secure.
- Security :
  The protocol achieves provable security in the standard model under the well known DDH assumption.
- Efficiency :

  Communication : Rounds required is 2 (plus a confirmation additional round).

  Computation : Each user performs more than $3n$ modular exponentiations, $3n$ modular multiplications, n signature generations and n signature verifications.
  The protocol is inefficient from point of view of computation rate.

3.4.7 Bresson, Chevassut and Pointcheval's Dynamic Group Key Agreement

(Bresson, Chevassut, Pointcheval [33], 2001)

Bresson *et al.* [33] provided a formal treatment of the authenticated group Diffie-Hellman key exchange problem in a scenario in which the membership is dynamic rather than static.

- Protocol Description :

Setup : We consider a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$ where l is a security parameter. The session key SK associated to the protocol is $\{0,1\}^l$ equipped with a uniform distribution. Let $G = \langle g \rangle$ be a finite cyclic group of a k-bit prime (large) order q. This group could be a prime subgroup of Z_p^* or it could be an (hyper)-elliptic curve based group. We view G as a multiplicative group.

– *Key Agreement* : The authenticated dynamic group key agreement scheme consists of three protocols SETUP1, REMOVE1 and JOIN1. Suppose a multi-cast group of users $\mathcal{I} = \{U_1, \ldots, U_n\}$ wish to agree upon a common key. They are arranged in a ring. Each user saves the set of values he receives in the down-flow of SETUP1, REMOVE1 and JOIN1. These values are needed to execute REMOVE1 for a subsequent removal of a user from $\mathcal{I}$. Any user from $\mathcal{I}$ could be selected as a group controller U_{GC} trusted to initialize the dynamic operations. In this protocol, we consider the user with the highest index in $\mathcal{I}$ as the group controller, the flows of a user U are signed using its long-lived key LL_U, the names of the users are in the protocol flows, and the session key sk is

$$\mathsf{sk} = \mathcal{H}(\mathcal{I}|\mathsf{Fl}_{\mathsf{max}(\mathcal{I})}|g^{x_1 \ldots x_{\mathsf{max}(\mathcal{I})}})$$

where $\mathsf{Fl}_{\mathsf{max}(\mathcal{I})}$ is the down-flow, session identities SIDS and partner identities PIDS are appropriately defined.

(a) Protocol SETUP1

The protocol consists of two stages: up-flow and down-flow. The multi-cast group $\mathcal{I}$ is set to $\mathcal{J}$. In the up-flow, the user U_i receives a set (Y, Z) of intermediate value with

$$Y = \cup_{0<m<i}\{Z^{\frac{1}{x_m}}\}$$

and Z, where $Z = g^{\Pi_{0<t<i}x_t}$. User U_i chooses at random a private value x_i, raises the value in Y to the power of x_i and then concatenates with Z to obtain his intermediate value

$$Y' = \cup_{0<m\leq i}\{Z'^{\frac{1}{x_m}}\},$$

where $Z' = Z^{x_i} = g^{\Pi_{0<t\leq i}x_t}$. User U_i then forwards the value (Y', Z') to the next user in the ring. The down-flow occurs when $U_{\mathsf{max}(\mathcal{I})}$ receives the last up-flow. At that point $U_{\mathsf{max}(\mathcal{I})}$ performs the same steps as a user in the up-flow, but broadcasts the set of intermediate value Y' only. In effect, the value Z' computed by $U_{\mathsf{max}(\mathcal{I})}$ will lead to the session key sk, since $Z' = g^{\Pi_{0<t\leq n}x_t}$. Users in $\mathcal{I}$ compute sk and accept.

(b) Protocol REMOVE1

Suppose a set of users $\mathcal{J} \subset \mathcal{I}$ want to leave the multi-cast group $\mathcal{I}$. The multi-cast group $\mathcal{I}$ is first set to be $\mathcal{I} \setminus \mathcal{J}$. This protocol consists of a down-flow only. The group controller U_{GC} (*i.e.* with the highest-index in $\mathcal{I} \setminus \mathcal{J}$) generates a random value x'_{GC} and removes from the saved previous broadcast the values destinated to the users in $\mathcal{J}$. U_{GC} then raises all the remaining values in which x_{GC} appeared to the power of $(x_{\mathsf{GC}}^{-1}\, x_{\mathsf{GC}})$ and broadcasts the rest (x_{GC} is U_{GC}'s previous secret value). Users in $\mathcal{I}$ compute session key sk and accept. Users in $\mathcal{J}$ erase any internal data, user U_{GC} erases x_{GC} and $x_{\mathsf{GC}^{-1}}$ while internally saving x'_{GC}.

(c) Protocol JOIN1

Suppose a set of new users $\mathcal{J}$ want to join the multi-cast group $\mathcal{I}$. The protocol consists of two stages: up-flow and down-flow. The group controller U_{GC} (*i.e.* user with highest-index in $\mathcal{I}$) generates a random value x'_{GC}, raises the value from the saved previous broadcast in which x_{GC} appears to the power of $(x_{\mathsf{GC}}^{-1}\, x'_{\mathsf{GC}})$ and obtains a set of values Y' (x_{GC} is U_{GC}'s previous secret exponent). U_{GC} also computes the value Z' by raising the last value in Y' to x'_{GC}. U_i then forwards the values (Y', Z') to the first joining user in $\mathcal{J}$. From that point, JOIN1 will work as the SETUP1 protocol. Upon receiving the broadcast flow, users in $\mathcal{I} \cup \mathcal{J}$ erase previous session keys, compute sk and accept. The multi-cast group $\mathcal{I}$ is then set to $\mathcal{I} \cup \mathcal{J}$.

- Assumption :
  Generalized DH (or Many DH) problem is hard and the signature scheme is secure.
- Security :
  The protocol is proven to be secure in the random oracle model. The authors precisely define a security model for dynamic authenticated group key agreement with "implicit" authentication as the fundamental goal and the entity authentication as well and provide a concrete security analysis of their scheme in this model.
- Efficiency : Suppose n is the number of group members, j and l are respectively the number of joining and leaving users, e, v, s are respectively the cost of modular exponentiation, signature verification and signature generation.

  Communication : For protocols SETUP1, REMOVE1 and JOIN1, rounds required are respectively $n, 1, O(j)$ and maximum bits sent per user are respectively $n|q| + |\sigma|$, $(n - l)|q| + |\sigma|$ and $(n + j)|q| + |\sigma|$.

  Computation : For protocols SETUP1, REMOVE1 and JOIN1, computations required per user are respectively $ne + s + v$, $(n - l)e + s$ and $(n + j)e + 2s + jv$.

Note : Bresson, Chevassut and Pointcheval presented the static authenticated protocol in [35] and considered the dynamic scenario in [33]. The security analysis of both protocols are in a security

model formalized by themselves in the random oracle model under the CDH assumption. To incorporate major missing details (*e.g.* strong-corruption and concurrent sessions), they further refine the existing security model and proposed an authenticated dynamic group Diffie-Hellman key exchange and show that it is provably secure under the DDH assumption within this model. Their security result holds in the standard model instead of random oracle model.

3.4.8 Bresson, Chevassut, Essiari and Pointcheval's Dynamic Group Key Agreement

(Bresson, Chevassut, Essiari, Pointcheval [34], 2003)

A very efficient provably secure group key agreement is introduced in dynamic scenario which is suitable for restricted power devices and wireless environments. The scheme consists of three protocols: the setup protocol GKE.Setup, the remove protocol GKE.Remove and the join protocol GKE.Join. The main GKE.Setup protocol allows a cluster of mobile users (also called clients) and a wireless gateway (also called server) to agree on a session key. The other two protocols of the scheme handle efficiently the dynamic membership changes of clients in one wireless domain.

- Protocol Description :

  Setup : Let G be a finite multiplicative group of some large prime order q and g be a generator of G. Let $l = |q|$. We consider three hash functions $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$, $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^{l_0}$, $\mathcal{H}_1 : \{0,1\}^{l_1} \times G \rightarrow \{0,1\}^{l_0}$, where l_1 is the maximum bit length of a counter c used to prevent replay attacks. Let C be the set of all potential clients and S be the server. Before the protocol is run for the first time, an initialization phase occurs during which the following steps are performed.

  1. Each client $U_i \in C$ generates a pair of signing private/public keys $(\mathsf{SK}_i, \mathsf{PK}_i)$ by running the key generation algorithm of a signature scheme.
  2. The server S sets its private/public keys to be $(\mathsf{SK}_S, \mathsf{PK}_S) = (x, y)$, where $x \in_R Z_q^*$ and $y = g^x$.

  Key Agreement : The protocol executes in two rounds. In the first round, S collects contributions from individual clients and then, in the second round, it sends the group keying material to the clients.

  (a) Protocol GKE.Setup

  The algorithm, on input a set of client devices $\mathcal{J}$, performs the following steps.

  1. Set the wireless group $\mathcal{G}_c$ to be the input set $\mathcal{J}$.
  2. Each client $U_i \in \mathcal{G}_c$ chooses $x_i \in Z_q$ at random and precomputes $y_i = g^{x_i}$, $\alpha_i = \mathsf{PK}_S^{x_i} = y_i^{x_i}$ as well as a signature σ_i of y_i, under the private key SK_i.
  3. Each client U_i sends (y_i, σ_i) to S.

4. For each $U_i \in \mathcal{G}_c$, the server S checks the signature σ_i using PK_i, and if they are all correct, computes the value $\alpha_i = y_i^x$.
5. The server S initializes the counter $c = 0$ as a bit-string of length l_1 and computes the shared secret value $K = \mathcal{H}_0(c|\{\alpha_i\}_{i\in\mathcal{G}_c})$ and sends to each client U_i the value c and $K_i = K \oplus \mathcal{H}_1(c|\alpha_i)$.
6. Each client U_i (and S) recovers the shared secret value $K = K_i \oplus \mathcal{H}_1(c|\alpha_i)$ and the session key $\mathsf{sk} = \mathcal{H}(K|\mathcal{G}_c|S)$.

(b) Protocol GKE.Remove

The algorithm on input the set $\mathcal{J}$ of leaving client devices, performs the following steps.

1. Update the wireless client group $\mathcal{G}_c = \mathcal{G}_c \setminus \mathcal{J}$.
2. The server S operates as in the GKE.Setup phase. It increases the counter c and computes the shared secret value $K = \mathcal{H}_0(c|\{\alpha_i\}_{i\in\mathcal{G}_c})$. item3. Then S sends to each client $U_i \in \mathcal{G}_c$ the value c and $K_i = K \oplus \mathcal{H}_1(c|\alpha_i)$.
4. Each client $U_i \in \mathcal{G}_c$ already holds the value $\alpha_i = g^{xx_i}$, and the old counter value. So it first checks that the new counter is greater than the old one and recovers the secret shared value $K = K_i \oplus \mathcal{H}_1(c|\alpha_i)$ and the session key $\mathsf{sk} = \mathcal{H}(K|\mathcal{G}_c|S)$.

(c) Protocol GKE.Join

The algorithm, on input the set of joining client devices $\mathcal{J}$, performs the following steps.

1. Update the wireless client group $\mathcal{G}_c = \mathcal{G}_c \cup \mathcal{J}$.
2. Each joining client $U_j \in \mathcal{J}$ chooses at random a value $x_j \in Z_q^*$ and precomputes $y_j = g^{x_j}$, $\alpha_j = \mathsf{PK}_S^{x_j}$ as well as a signature σ_j of y_j, under the private key SK_j
3. Each joining client $U_j \in \mathcal{J}$ sends the value (y_j, σ_j) to the device server S.
4. The server S checks the incoming signatures and if correct, operates as in the GKE.Setup phase, with an increased counter c and computes the shared value $K = \mathcal{H}_0(c|\{\alpha_i\}_{i\in\mathcal{G}_c})$.
5. Then it sends to each client $U_i \in \mathcal{G}_c$ the value c and $K_i = K \oplus \mathcal{H}(c|\alpha_i)$.
6. Each client $U_i \in \mathcal{G}_c$ already holds the value $\alpha_i = g^{xx_i}$ and the old counter value (set to be zero for the new ones). So it first checks the new counter is greater than the old one, and recovers the secret shared value $K = K_i \oplus \mathcal{H}_1(c|\alpha_i)$ and the session key $\mathsf{sk} = \mathcal{H}(K|\mathcal{G}_c|S)$.

- Assumption :
  CDH problem is hard and the signature scheme is secure.
- Security :
  The protocol is proven to be secure in the random oracle model. They have formalized the

adversarial model giving the adversary an enormous capabilities to closely model its abilities in the real life. They assume that the adversary never participates in the protocol execution (neither as a client, nor as a server). The security analysis of their scheme is in this security model and they claim that the protocol achieves partial forward secrecy. However, Nam, Kim, Won [94] demonstrate that the basic setup protocol GKE.Setup is insecure against known key security and if an active adversary is ever allowed to participate in the protocol as a client, the protocol does not even provide the fundamental security attribute implicit key authentication and perfect forward secrecy. They have proposed a modified version of the scheme which satisfies all the security properties: implicit key authentication, forward secrecy and known key security.

- Efficiency :

  Communication : Rounds required is 2 for each of GKE.Setup, GKE.Remove and GKE.Join.

  Computation : The protocol uses a base station as a trustee who has the special role of additing or removing clients from the group. This server computes n modular exponentiations, 1 signature generation, n signature verification, n one-way hash function operation and n XOR operations.

3.4.9 Nam, Kim, Kim and Won's Dynamic Group Key Agreement

(Nam, Kim, Kim, Won [93], 2004)

Nam *et al.* [93] presented a communication-efficient dynamic group key agreement protocol well suited for a lossy and high-delay unbalanced network environment consisting of mobile hosts with restricted computational resources and stationary hosts with relatively high computational capabilities. They analyzed their scheme in the random oracle model and proved that it is secure under factoring assumption.

- Protocol Description :

  Setup : Let $N = pq$ where p, q are large distinct primes, $|p| = |q|$ such that $p = 2p' + 1$ and $q = 2q' + 1$ where p', q' are also prime integers. Then such an N is a Blum integer since $p = q = 3(\text{mod}4)$. We denote by Z_N^* the multiplicative group modulo N. We choose a quadratic residue $g \neq 1$ uniformly at random from the set of quadratic residues in Z_N^* generated by g. Suppose U_c is the controller in a multi-cast group $\mathcal{MG}$, and $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$ is a hash function modeled as a random oracle in the security proof of the scheme. In initialization phase, each user U_i is assigned a long-term public/private key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$ by running a key generation algorithm. The public keys of all users are assumed to be known a priori to all parties including the adversary.

  Key Agreement : The scheme consists of three algorithms IKA1, LP1 and JP1 for initial group formation, user leave and user join respectively.

(a) Protocol IKA1

Assume a multi-cast group $\mathcal{MG} = \{U_1, \ldots, U_n\}$ of n users wish to establish a session key by participating in initial group formation protocol IKA1 which runs in 2 rounds, one with $n-1$ unicast and the other with a single broadcast. They perform the following steps.

1. Each user U_i picks a random $r_i \in [1, N]$ and computes $z_i = g^{r_i} \bmod N$. Then $U_i (\neq U_c)$ signs $U_i|z_i$ to obtain signature σ_i and sends $m_i = U_i|z_i|\sigma_i$ to the controller U_c.
2. The controller U_c, on receiving each message m_i, verifies the correctness of m_i and computes $y_i = z_i^{r_c} \bmod N$. After receiving all the $n-1$ messages, U_c computes Y as

$$Y = \begin{cases} \Pi_{i \in [1,n] \setminus \{c\}} y_i \bmod N & \text{if } n \text{ is even} \\ \Pi_{i \in [1,n]} y_i \bmod N & \text{if } n \text{ is odd} \end{cases}$$

   U_c also computes the set $\mathcal{T} = \{T_i | i \in [1, n]\}$ where $T_i = Y\ y_i^{-1}$. Let $\mathcal{Z} = \{z_i | i \in [1, n]\}$. Then, U_c signs $\mathcal{MG}|\mathcal{Z}|\mathcal{T}$ to obtain signature σ_i and broadcasts $m_c = \mathcal{MG}|\mathcal{Z}|\mathcal{T}|\sigma_i$ to the entire group.
3. Each user $U_i (\neq U_c)$, on receiving the broadcast message m_c, verifies the correctness of m_c and computes $Y = z_c^{r_i} T_i \bmod N$. All users in $\mathcal{MG}$ compute their session key as $K = \mathcal{H}(\mathcal{T}|Y)$, and store their random exponent r_i and the set $\mathcal{Z}$ for future use.

(b) Protocol LP1

Consider a scenario where a set of user $\mathcal{L}$ leaves a multi-cast group $\mathcal{MG}_p$. Then protocol LP1 is executed to provide each user a new multi-cast group $\mathcal{MG}_n = \mathcal{MG}_p \setminus \mathcal{L}$ with a new session key. Any remaining user can act as the controller in the new multi-cast group $\mathcal{MG}_n$. The protocol LP1 requires only one round with a single broadcast and proceeds as follows.

1. The group controller U_c picks a new random $r_c \in [1, N]$ and computes $z'_c = g^{r'_c} \bmod N$. Using r'_c, z'_c and the saved set $\mathcal{Z}$, U_c then proceeds exactly as in IKA1, except that it broadcasts $m_c = \mathcal{MG}|z_c|z'_c|\mathcal{T}|\sigma_c$ where z'_c is the random exponential from the previous controller.
2. Each user $U_i (\neq U_c)$, on receiving the broadcast message m_c, verifies that $\mathcal{V}(\mathcal{MG}_n|z_c|z'_c|\mathcal{T}, \sigma_c, \mathsf{PK}_c) = 1$ and the received z_c is equal to the one that is received in the previous session. All users in $\mathcal{MG}_n$ then compute their session key as $K = \mathcal{H}(\mathcal{T}|Y)$ and update the set $\mathcal{Z}$.

(c) Protocol JP1

Assume a scenario in which a set of j new users, $\mathcal{J}$, joins a multi-cast group $\mathcal{MG}_p$ to form a new multi-cast group $\mathcal{MG}_n = \mathcal{MG}_p \cup \mathcal{J}$. Then the user join protocol JP1 is executed to provide the users of $\mathcal{MG}_n$ with a new session key. The group controller in the new multi-cast group $\mathcal{MG}_n$ can be played by any user from the previous multi-cast group $\mathcal{MG}_p$. The protocol JP1 takes two rounds, one with j unicasts and the other with a single broadcast and proceeds as follows.

1. Each user $U_i \in \mathcal{J}$ picks a random $r_i \in [1, N]$ and computes $z_i = g^{r_i} \bmod N$. U_i then generates signature σ_i on $U_i|z_i$, sends $m_i = U_i|z_i|\sigma_i$ to U_c and stores r_i.
2. The group controller U_c proceeds in the usual way choosing a new random $r'_c \in [1, N]$, computes z'_c, Y, $\mathcal{T}$ and $K = \mathcal{H}(\mathcal{T}|Y)$, updating the set $\mathcal{Z}$ with new z_i's and then broadcasting $m_c = \mathcal{MG}_n|z_c|\mathcal{Z}|\mathcal{T}|\sigma_i$.
3. Each user $U_i \in \mathcal{MG}_p \setminus \{U_c\}$ verifies that the received z_c is equal to the one received in the previous session and user $U_i \neq U_c$ verifies the correctness of m_c. Then $U_i \neq U_c$ proceeds as usual, computing $Y = z_c'^{r_i} T_i \bmod N$ and $K = \mathcal{H}(\mathcal{T}|Y)$. All users in $\mathcal{MG}_n$ store or update the set $\mathcal{Z}$.

- Assumption :
  The factoring problem is hard and the signature scheme is existentially unforgeable.

- Security :
  A rigorous proof of security of the IKA1 protocol is provided in the security model as formalized by Bresson *et al.* [33, 32, 35]. The protocol is proven to be secure in the random oracle model under factoring assumption and provides perfect forward secrecy.

- Efficiency :

  Communication : For IKA1, JP1 and LP1 protocols, rounds required are respectively 2, 2, 1 and total messages sent are respectively n, $j + 1$ and 1 where j is the number of joining users. IKA1 requires $n - 1$ unicasts and 1 broadcast, JP1 requires j unicasts and 1 broadcast and LP1 requires 1 broadcast.

  Computation : For each of IKA1, JP1 and LP1 protocols, total modular exponentiation computed is $O(n)$ and total signature verification is $O(n)$.

Note : Nam, Kim, Yang, Won [95] investigate the problem of contributory group key agreement over combined wired/wireless networks, consisting of arbitrary number of mobile devices with limited computational resources and general-purpose stationary high-performance computer. They have designed a 3-round generalized protocol which take advantage of the difference in computing power among users and uses a 2-round unauthenticated protocol (introduced by them) very similar to the protocol IKA1 as a basic building block. Their 2-round basic protocol is proven to be secure against passive adversary under DDH assumption.

3.4.10 Kim, Lee and Lee's Dynamic Group Key Agreement

(Kim, Lee, Lee [76], 2004)
A very efficient authenticated group key exchange (AGKE) scheme is developed by Kim *et al.* [76] for dynamically changing groups in *ad hoc* networks, where a member of a group may join and/or leave at any given time and a group key is exchanged without the help of any central server. Their proposed scheme is provably secure and requires constant rounds.

- Protocol Description :

Setup : The protocol is based on the CDH assumption and a secure signature scheme $\Sigma = (\mathcal{K}, \mathcal{S}, \mathcal{V})$. We take the key space to be $\{0,1\}^l$ where l is a security parameter. Let G be a multiplicative group of some large prime order q ($l \leq |q|$) and g be a generator of G. We also consider a one-way hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$. In an initialization phase, each user U_i is provided a public/private key pair $(\mathsf{PK}_{U_i}, \mathsf{SK}_{U_i})$ for verifying/signing. The list of public keys is published to all users.

Key Agreement : The protocol consists of three algorithms GKE.Setup, GKE.Join and GKE.Leave for initial setup, user join and user leave respectively.

(a) Protocol GKE.Setup

Let $G_0 = \{U_1, \ldots, U_n\}$ be an initial group and ID_{U_i} be the identity of user U_i. We consider a ring structure among the members of G_0 and let $I_0 = \mathsf{ID}_{U_1}|\mathsf{ID}_{U_2}|\ldots|\mathsf{ID}_{U_n}$. The protocol executes in two rounds as follows.

1. In round 1, each user U_i randomly chooses $k_i \in \{0,1\}^l$ and $x_i \in Z_q^*$, computes $y_i = g^{x_i}$ and keeps k_i secret. The last user U_n computes $\mathcal{H}(k_n|0)$. Each user U_i generates a signature $\sigma_i^{(1)} = \mathcal{S}_{\mathsf{SK}_{U_i}}(M_i^{(1)}|I_0|0)$ where $M_i^{(1)} = y_i$ for $1 \leq i \leq n-1$ and $M_n^{(1)} = \mathcal{H}(k_n|0)|y_n$, and broadcasts $M_i^{(1)}|\sigma_i^{(1)}$.
2. In round 2, each user U_i, on receiving $M_j^{(1)}|\sigma_j^{(1)}$, verifies $\sigma_j^{(1)}$ on $M_j^{(1)}$ for all $j \neq i$. If some signatures are not valid, the process fails and halts. Otherwise, user U_i computes $t_i^L = \mathcal{H}(y_{i-1}^{x_i}|I_0|0)$, $t_i^R = \mathcal{H}(y_{i+1}^{x_i}|I_0|0)$ and generates $T_i = t_i^L \oplus t_i^R$. The last user U_n additionally computes $\widehat{T} = k_n \oplus t_n^R$. Each user U_i now generates signature $\sigma_i^{(2)} = \mathcal{S}_{\mathsf{SK}_{U_i}}(M_i^{(2)}|I_o|0)$ where $M_i^{(2)} = k_i|T_i$ for $1 \leq i \leq n-1$ and $M_n^{(2)} = \widehat{T}|T_n$, and broadcasts $M_i^{(2)}|\sigma_i^{(2)}$ for $1 \leq n$.
3. In the key computation phase, each user U_i verifies signature $\sigma_j^{(2)}$, $j \neq i$. If all signatures are valid, U_i computes $\widetilde{t}_{i+1}^R, \widetilde{t}_{i+2}^R, \ldots, \widetilde{t}_{i+(n-1)} (= \widetilde{t}_i^L)$ by using t_i^R as follows.

$$\widetilde{t}_{i+1}^R = T_{i+1} \oplus t_i^R, \widetilde{t}_{i+2}^R = T_{i+2} \oplus \widetilde{t}_{i+1}^R, \ldots, \widetilde{t}_{i+(n-1)}^R = T_{i+(n-1)} \oplus \widetilde{t}_{i+(n-2)}^R.$$

Then U_i checks if $t_i^L = \widetilde{t}_i^L$ holds. Note that the above check enables honest users to notice the errors caused by system faults or wrong messages broadcasted

by illegal users and halts the protocol. However, it is not easy to find who transmitted illegal messages. Finally, each user obtains $\widetilde{k}_n$ from $\widehat{T}$ and checks if $\mathcal{H}(\widetilde{k}_n|0) = \mathcal{H}(k_n|0)$. This check value guarantees key control. All users then compute a session key $\mathsf{sk}_0 = \mathcal{H}(k_1|k_2|\ldots|k_{n-1}|k_n|0)$.
4. Each user U_i additionally computes $h_i^L = \mathcal{H}(y_{i-1}^{x_i}|\mathsf{sk}_0|0)$, $h_i^R = \mathcal{H}(y_{i+1}^{x_i}|\mathsf{sk}_0|0)$ and $X = \mathcal{H}(k_n|\mathsf{sk}_0|0)$, saves $(h_i^L, h_i^R, X, \mathsf{sk}_0)$ secretly and erases other ephemeral data. These post computed values will be used for subsequent join or leave operation.

(b) Protocol GKE.Join

Let $G_{v-1} = \{U_1, \ldots, U_n\}$ $(v \geq 1)$ be the current group and $\mathcal{J} = \{U_{n+1}, \ldots, U_{n+n'}\}$ $(n' \geq 1)$ be a set of new users wishing to join the group G_{v-1}. The group G_{v-1} is divided into three parts: $\{U_1\}$, $\{U_2, \ldots, U_{n-1}\}$ and $\{U_n\}$. We consider U_2 as a representative of $\{U_2, \ldots, U_{n-1}$ and for convenience of explanation, we allow that $U_{n+n'+1}$, $U_{n+n'+2}$ and $U_{n+n'+3}$ denote U_1, U_2, U_3 respectively. In this algorithm, a ring structure is considered among the users $U_{n+1}, \ldots, U_{n+n'+3}$. Let $\mathcal{G}$ be the set $\{U_{n+1}, \ldots, U_{n+n'+3}\}$ and $I_v = \mathsf{ID}_{U_1}|\ldots|\mathsf{ID}_{U_{n+n'}}$. The users in $\mathcal{G}$ proceed as follows.

1. In round 1, each user $U_{n+i}\mathcal{G}$ randomly chooses $k_{n+i} \in \{0,1\}^l$ and $x_{n+i} \in Z_q^*$, computes $y_{n+i} = g^{x_{n+i}}$ and keeps k_{n+i} secretly. The user $U_{n+n'+2}(= U_2)$ computes $y_{n+n'+2} = g^X$ by using the secret value X instead of $x_{n+n'+2}$ and the user $U_{n+n'+3}(= U_3)$ computes $\mathcal{H}(k_{n+n'+3}|v)$. Each user U_{n+i} generates signature $\sigma_{n+i}^{(1)} = \mathcal{S}_{\mathsf{SK}_{U_{n+i}}}(M_{n+i}^{(1)}|I_v|v)$ where $M_{n+i}^{(1)} = y_{n+i}$ for $1 \leq i \leq n'+2$ and $M_{n+n'+3}^{(1)} = \mathcal{H}(k_{n+n'+3}|v)|y_{n+n'+3}$ and broadcasts $M_{n+i}^{(1)}|\sigma_{n+i}^{(1)}$.
2. In round 2, all users, on receiving $M_{n+i}^{(1)}|\sigma_{n+i}^{(1)}$, verifies $\sigma_{n+i}^{(1)}$'s. Each user U_{n+i} computes $t_{n+i}^L = \mathcal{H}(y_{L(n+i)}^{x_{n+i}}|I_v|v)$, $t_{n+i}^R = \mathcal{H}(y_{R(n+i)}^{x_{n+i}}|I_v|v)$ and generates $T_{n+i} = t_{n+i}^L \oplus t_{n+i}^R$ where $L(n+i)$ and $R(n+i)$ respectively means the left and right index of $n+i$ on the ring for $i \in \{1, \ldots, n'+3\}$. The user $U_{n+n'3}$ additionally computes $\widehat{T} = k_{n+n'+3} \oplus t_{n+n'+3}^R$. Each user U_{n+i} generates signature $\sigma_{n+i}^{(2)} = \mathcal{S}_{\mathsf{SK}_{U_{n+i}}}(M_{n+i}^{(2)}|I_v|v)$ and broadcasts $M_{n+i}^{(2)}|\sigma_{n+i}^{(2)}$ where $M_{n+i}^{(2)} = k_{n+i}|T_{n+i}$ for $1 \leq i \leq n'+2\}$ and $M_{n+n'+3}^{(2)} = \widehat{T}|T_{n+n'+3}$. All users of $\{U_3, \ldots, U_{n-1}\}$ compute $t_{n+n'+2}^L$ and $t_{n+n'+2}^R$ by using X.
3. In the key computation phase, all users verify the signatures $\sigma_{n+i}^{(2)}$. If all signatures are valid, each user U_{n+i} computes $\widetilde{t}_{n+i+1}^R, \ldots, \widetilde{t}_{n+i+n'-1}(= \widetilde{t}_{n+i}^L)$ by using t_{n+i}^R and checks if $t_{n+i}^L = \widetilde{t}_{n+i}^L$ holds. Also, users $U_3, \ldots, U_{n-1}$ can check it by using $t_{n+n'+2}^L$ and $t_{n+n'+2}^R$. Finally, all users recover $k_{n+n'+3}$ for $\widehat{T}$ and computes a new session key $\mathsf{sk}_v = \mathcal{H}(k_{n+1}|\ldots|k_{n+n'+3}|v)$.
4. Each new user U_{n+i} $(1 \leq i \leq n')$ computes $h_{n+i}^L = \mathcal{H}(y_{L(n+i)}^{x_{n+i}}|\mathsf{sk}_v|v)$ and $h_{n+i}^R = \mathcal{H}(y_{R(n+i)}^{x_{n+i}}|\mathsf{sk}_v|v)$. Users U_1 and U_n respectively compute $h_1^L = \mathcal{H}(y_{n+n'}^{x_1}|\mathsf{sk}_v|v)$

and $h_n^R = \mathcal{H}(y_{n+1}^{x_n}|\mathsf{sk}_v|v)$ instead of the previous value $h_i^L (= h_n^R)$. All users compute a new value $X = \mathcal{H}(k_n|\mathsf{sk}_v|v)$. Each user U_i then saves h_i^L, h_i^R, X and sk_v secretly and erases all other ephemeral data. These post computation is required for any subsequent join or leave operation.

(c) Protocol GKE.Leave

Let $G_{v-1} = \{U_1, \ldots, U_n\}$ be the current group and let $\mathcal{R} = \{U_{l_1}, \ldots, U_{l_{n''}}\}$ where $\{l_1, \ldots, l_{n''}\} \subset \{1, \ldots, n\}$, be a set of leaving users. We denote by $\mathcal{N}(\mathcal{R})$ the set of all left/right neighbors of leaving users on the ring. *i.e.*

$$\mathcal{N}(\mathcal{R}) = \{U_{L(l_1)}, U_{R(l_1)}, \ldots, U_{L(l_{n''})}, U_{R(l_{n''})}\}.$$

For the ease of discussion, let $U_{L(l_i)} = U_{l_i-1}$ and $U_{R(l_i)} = U_{l_i+1}$. Then $\mathcal{N}(\mathcal{R}) = \{U_{l_1-1}, U_{l_1+1}, \ldots, U_{l_{n''}-1}, U_{l_{n''}+1}\}$. To generate a new group $G_v = G_{v-1}\mathcal{R}$ with a new session key sk_v, a new Diffie-Hellman value should be shared between the left and right neighbors U_{l_j-1} and U_{l_j+1} $(1 \le j \le n'')$ respectively of the leaving user U_{l_j} on the ring. In this Leave algorithm, we consider a ring structure among members of G_v and we newly index the members as $G_v = \{U_1, \ldots, U_{n-n''}\}$. Let $I_v = \mathsf{ID}_{n_1}| \ldots |\mathsf{ID}_{n-n''}$. The algorithm runs in two rounds as follows.

1. In round 1, each user U_w of $\mathcal{N}(\mathcal{R})$ randomly chooses $k_w \in \{0,1\}^l$ and $x_w \in Z_q^*$, computes $y_w = g^{x_w}$ and keeps k_w secretly. Then user $U_{l_{n''}+1}$ computes $\mathcal{H}(k_{l_{n''}+1}|v)$. User U_w generates signature $\sigma_w^{(1)} = \mathcal{S}_{\mathsf{SK}_{U_w}}(M_w^{(1)}|I_v|v)$ where $M_w^{(1)} = y_w$ with $w \in \{l_1 - 1, l_1 + 1, \ldots, l_{n''} - 1\}$ and $M_{l_{n''}+1}^{(1)} = \mathcal{H}(k_{l_{n''}+1}|v)|y_{l_{n''}+1}$ and broadcasts $M_w^{(1)}|\sigma_w^{(1)}$.
2. All users of G_v verify signatures $\sigma_w^{(1)}$'s in the second round. If all signatures are valid, each user U_{l_j-1} (respectively U_{l_j+1}) of $\mathcal{N}(\mathcal{R})$ regenerates $h_{l_j-1}^R = y_{l_j+1}^{x_{l_j-1}}$ (respectively, $h_{l_j+1}^L = y_{l_j-1}^{x_{l_j+1}}$). Then each user U_i of G_v computes $t_i^L = \mathcal{H}(h_i^L|I_v|v)$, $t_i^R = \mathcal{H}(h_i^R|I_v|v)$ and $T_i = t_i^L \oplus t_i^R$. The user $U_{l_{n''}+1}$ generates a signature $\widehat{T} = k_{l_{n''}+1} \oplus t_{l_{n''}+1}^R$. Each user $U_i^{l_{n''}+1}$ generates a signature $\sigma_i^{(2)} = \mathcal{S}_{\mathsf{SK}_{U_i}}(M_i^{(2)}|I_v|v)$ and broadcasts $M_i^{(2)}|\sigma_i^{(2)}$ where $M_{l_{n''}+1}^{(2)} = \widehat{T}|T_{l_{n''}+1}$, $M_i^{(2)} = k_i|T_i$ for other users except $U_{l_{n''}+1}$ of $\mathcal{N}(\mathcal{R})$ and $M_i^{(2)} = T_i$ for users of $G_v \backslash \mathcal{N}(\mathcal{R})$.
3. In the session key computation phase, all users verify signatures $\sigma_i^{(2)}$'s. If all signatures are valid, each user U_i computes $\widetilde{t}_{i+1}^R, \widetilde{t}_{i+2}, \ldots, \widetilde{t}_{i+(n-n''-1)} (= \widetilde{t}_i^L)$ by using t_i^R and checks if $t_i^L = \widetilde{t}_i^L$ holds. Finally, all users compute a session key
$$\mathsf{sk}_v = \mathcal{H}(k_{l_1-1}|k_{l_1+1}| \ldots |k_{l_{n''}-1}|k_{l_{n''}+1}|v).$$
4. Each user U_i regenerates $h_i^L = \mathcal{H}(h_i^L|\mathsf{sk}_v|v)$, $h_i^R = \mathcal{H}(h_i^R|\mathsf{sk}_v|v)$ and $X = \mathcal{H}(k_{l_{n''}+1}|\mathsf{sk}_v|v)$ and saves h_i^L, h_i^R, X and the session key sk_v secretly. These post computation is required for any subsequent join or leave operation.

- Assumption :
  CDH problem is hard and the signature scheme is secure.

- Security :
  The protocol is proven to achieve provable security in the random oracle model. The proof technique is similar to that in [34].

- Efficiency :
  Suppose n is the number of group members, j and l are respectively the number of joining and leaving users, h, x, e, v, s are respectively the cost of hash function operation, XOR operation, modular exponentiation, signature verification and signature generation.

  Communication : For each of the algorithms GKE.Setup, GKE.Join and GKE.Leave, rounds required is 2, the maximum bits sent per user is $|q| + 3|h| + 2|\sigma|$ where $|q|, |h|, |\sigma|$ respectively denote the length of q, the order of the cyclic group G, the length of hash function h and the length of a signature.

  Computation : The computations required for algorithms GKE.Setup, GKE.Join and GKE.Leave are respectively $3e+4h+(n+1)x+2s+O(n)v$, $3e+4h+(j+1)x+2s+O(j)v$ and $3e + 4h + (n-1)x + 2s + (l+n)v$.

3.5 Conclusion

This survey is devoted to realization of key agreement. We have included a comprehensive treatment of describing the most important key agreement protocols and provided an account of chronological developments connected with key agreement.

Chapter 4

Tree-Based Group Key Agreement using Pairing

In this chapter, we are mainly interested in designing a pairing-based protocol. We present a provably secure tree based authenticated group key agreement protocol in dynamic scenario. The material that we provide here are basically related to our works in [9, 52, 53, 54]. Bilinear pairing and multi-signature are at the heart of our protocol. We prove that our protocol is provably secure in the security model of Bresson *et al.* An appropriate modification of Katz-Yung approach to tree based setting is adopted while proving its security against active adversaries. The protocol has an in-built hierarchical structure that makes it desirable for certain applications.

4.1 Introduction

Key agreement is one of the fundamental cryptographic primitives. This is required in situations where two or more parties want to communicate securely among themselves. The situation where three or more parties share a secret key is often called *conference keying*. In this situation, the parties can securely send and receive messages from each other. An adversary not having access to the secret key will not be able to decrypt the message.

A group key agreement protocol allows a group of users to exchange information over public network to agree upon a common secret key from which a session key can be derived. This common session key can later be used to achieve desirable security goals, such as authentication, confidentiality and data integrity.

Key agreement protocols fall naturally into two classes – authenticated and unauthenticated. Unauthenticated key agreement protocols consider the adversary to be passive, i.e., the adversary can listen to the traffic on the network, but cannot alter it. On the other hand, authenticated key agreement protocols consider the adversary to be active, i.e., the adversary can alter/replace the

messages flowing through the network. Thus the security requirements for authenticated key agreement is more stringent. Some of the desirable properties in authenticated key agreement are mutual implicit key authentication, known key security, forward secrecy, key compromise impersonation and key control.

Diffie-Hellman proposed the first two-party single-round key agreement protocol in their seminal paper [49]. In one of the breakthroughs in key agreement protocols, Joux [66] proposed a single-round three party key agreement protocol that uses bilinear pairings. Earlier, Burmester and Desmedt [36] had proposed a multi-party two-round key agreement protocol. They proved in [37] that this protocol is secure against a passive adversary in the standard model under decision Diffie-Hellman assumption. All these protocols are unauthenticated in the sense that an active adversary who has control over the channel can mount a man-in-the-middle attack to agree upon separate keys with the users without the users being aware of this.

Authenticated key agreement protocols allow two or more parties to agree upon a common secret key even in the presence of active adversaries. Authenticated key agreement protocols are the basic tools for group-oriented and collaborative applications such as, distributed simulation, multi-user games, audio or video-conferencing, and also peer-to-peer applications that are likely to involve a large number of users.

Apart from authentication, the other aspects of key agreement protocols are computation and communication efficiency. When the number of participants is more than two, the protocol is usually a multi-round protocol. In each round some or all of the users send messages to the other users and at the end of all the rounds, all the users should agree upon a common key. The total number of bits exchanged in the protocol is a crucial parameter in judging the efficiency of the protocol. Further, in each round, each user has to perform some computation like an exponentiation or a scalar multiplication. The total amount of computation required by all the users is another measure of efficiency of the protocol.

Designing scalable constant round protocols get much attention due to its increasing applicability in modern computing environments. Bresson and Catalano [30] proposed a constant round provably secure authenticated static group key agreement protocol and proved its security in the standard model using standard secret sharing and ElGamal encryption scheme. However, with increasing number of users, the complexity of the protocol becomes very high. More recently, Kim, Lee, Lee [76] presented a constant round authenticated key agreement for dynamic group secure in random oracle model. A secure constant round dynamic group key agreement protocol suitable for wired/wireless network is proposed by Nam *et al.* [92]. The security of this scheme is also in random oracle model.

Tree based group key agreement protocols are typically essential while the users are grouped into a hierarchical structure. The leaves of the tree denote individual users and each internal node corresponds to a user that represents the set of users in the subtree rooted at that node. The representative users have more computational resources than other users in the subtree. In a tree based group key agreement protocol, the set of all users in each subtree agree upon a common secret key. While a new user joins a group or a old user leaves a group in dynamic situation with large

group size, then it is worth to make an optimal use of the data precomputed in the previous session (retaining of course the desirable security attributes), instead of executing the whole protocol among the new group. Making optimal use of precomputed values in the previous session, a group of users can save computation and communication in subsequent sessions in which users join or leave the group. Besides, the in-built hierarchical structure of a tree-based protocol enables some subclass of users in the group to agree upon multiple common keys after a single session execution. This facilitates a particular subclass of users of the group to securely communicate among themselves. These features make tree based key agreement protocols useful for certain applications and there have been quite a number of tree based key agreement protocols [5, 10, 73, 74, 113].

Although designing constant round provably secure group key agreement [30, 76, 92] received much attention in the current research, it does not eliminate the need of tree based key agreement. We can combine constant round protocols and tree based protocols to get hybrid group key agreement which are efficient in terms of both computation and communication power. Consider the situation where there are collection of user sets, each having a common key agreed upon by executing an efficient constant round protocol among the users in that user set. Now executing a constant round protocol among these user sets may not always be desirable and executing the protocol among all the users may be expensive from computation or communication point of view when number of users in each subgroup is large. For instance, if the number of groups is about 20 and each group is large, then a constant round key agreement using the protocol of [76] will involve a large number of computations as well as communications. In contrast, the tree based protocol (with the representatives of each group) will compute the common key in 3 rounds with lesser number of communications and verifications. Thus, a tree based scheme can be incorporated among these user sets to get an efficient multi-party key agreement protocol.
Our Contribution: The objective of this work is to design a multi-party key agreement protocol using bilinear pairing. We obtain a provably secure tree based authenticated group key agreement in the dynamic scenario where a user can join or leave the group as his desire with updating sets of keys. The content of this chapter is based on our works in [9, 52, 53, 54].

First we present an unauthenticated multi-party key agreement protocol secure against passive adversary. For n parties, $n > 2$, the number of rounds required for our protocol is $\lceil \log_3 n \rceil$. The total number of messages exchanged and the total number of scalar multiplications (over a suitable elliptic curve) is less than $\frac{5}{2}(n-1)$. Additionally, $n\lceil \log_3 n \rceil$ pairings have to be computed.

Our protocol is based on bilinear maps. In one of the breakthroughs in key agreement protocols, Joux [66] proposed a three party, single round key agreement protocol. An authenticated version of this protocol has been proposed in [3], [124]. The basic Joux protocol is at the heart of our (unauthenticated) protocol. We group the users into three user sets using top down recursive procedure so that user sets with two users appear only at the first round. The Joux protocol is invoked for each set of three users to agree upon a common key. For the set of two users (if any), one of them generates randomly another short term private key to have two keys. These two keys and the short term private key of the other user is used in another invocation of Joux's protocol. A set of one user is kept unchanged. Our protocol then proceeds over several rounds and each round consists of several invocations of Joux's protocol. Ultimately, we are left with three user sets and

Joux's protocol is invoked once more to complete the protocol.

Thus our multi party protocol is essentially a combination of Joux's tripartite Diffie-Hellman protocol and tree based group key agreement using ternary tree structure. Even though the idea of combining Joux's protocol and tree based group key agreement is a natural extension of Joux's original protocol, it is nontrivial to obtain a security proof for the protocol against passive adversary. One of our major contribution is to provide such a proof using techniques from [10], [74], [113]. In fact, any secure two or three party protocol can be used with the ternary tree structure to obtain a secure multi party protocol. Our security analysis against passive adversaries is also a kind of reduction. We argue that if the underlying two and three party protocols are secure, then our multi party protocol is also secure.

We then obtain a provably secure authenticated tree based group key agreement protocol from our unauthenticated protocol. The security of the authenticated protocol is considered in the model formalized by Bresson *et al* [32]. Authentication is achieved by modifying the Katz-Yung [70] technique to tree based protocol. The main component of the Katz-Yung authenticated protocol is the use of a digital signature scheme. We also use a signature scheme, where each user signs every message that (s)he sends. However, in keeping with the tree structure of the protocol, we require that in each round each user sends the signed message to his/her representative. The representative then combines these signatures and sends the aggregate signature [52] or multi-signature [53] to the other users. This results in savings in the total amount of communication. (We will concentrate on presenting our authenticated protocol [53] which uses multi-signature and is more efficient than an earlier version of the protocol [52] that uses aggregate signature.)

To a certain extent, this work completes the task of extending Joux's three-party key agreement protocol to multi-party setting with a concrete security analysis against active adversaries in a formal security model. Moreover, a similar construction can be used to modify any tree based unauthenticated group key agreement protocol to an authenticated group key agreement protocol.

Finally, we further extends this static authenticated protocol to a dynamic authenticated protocol and provide a proof of security in the security model of Bresson *et al.* [32]. Our protocol is designed to ensure minimum modification to the computation already precomputed when a user leaves or joins the group. Besides, if the tree structure is not maintained after a join or leave operation, then the subsequent join or leave operations cannot be performed anymore. Thus retaining the tree structure is another important issue of our protocol while such dynamic operations (join or leave) are concerned.

4.2 Dynamic Group Key Agreement Protocol

We start by describing the requirements of the protocol, follow it up with a description of our unauthenticated protocol as well as our authenticated protocol and finally propose our authenticated dynamic protocol. Our dynamic group key agreement protocol is designed to ensure minimum modification to the computations already precomputed when a user leaves or joins the group. By appropriately modifying the proof in [53], the security of our dynamic authenticated protocol can be

reduced to that of the unauthenticated protocol which is a provably secure scheme under DHBDH assumption.

Suppose a set of n users $\mathcal{P} = \{U_1, U_2, \ldots, U_n\}$ wish to agree upon a secret key. Let US be a subset of users. Quite often, we identify a user with its instance during the execution of a protocol. In case US is a singleton set, we will identify US with the instance it contains. Each user set US has a representative $\mathsf{Rep}(\mathsf{US})$ and for the sake of concreteness we take $\mathsf{Rep}(\mathsf{US}) = U_j$ where $j = \min\{k : \Pi_{U_k}^{d_k} \in \mathsf{US}\}$. We use the notation $A[1, \ldots, n]$ for an array of n elements $A_1, \ldots, A_n$ and write $A[i]$ or A_i to denote the i-th element of array $A[\,]$. Let $G_1 = \langle P \rangle, G_2$ (groups of prime order q) and $e(,)$ be as described in Section 2.2.11 of Chapter 2. We choose a hash function $H : G_2 \to Z_q^*$. The public parameters are $\mathsf{params} = (G_1, G_2, e, q, P, H)$. Each user $U_i \in \mathcal{P}$ chooses $s_i \in Z_q^*$ at random which it uses as its ephemeral key. These keys are session specific and determine the final common key for the users in a session.

4.2.1 Unauthenticated Key Agreement Protocol

First we describe the idea behind our n-party key agreement protocol which is an extension of Joux's three-party single-round key agreement protocol to multi-party setting. *We partition the set of users into three user sets using top down recursive procedure so that user sets with two users appear only at the first round. We invoke Joux protocol for each set of three users to agree upon a common key. For the set of two users, one of them generates randomly another short term secret key to have two keys. These two keys and the short term secret key of the other user is used in another invocation of Joux's protocol. We keep a set consisting of one user unchanged. Our protocol then proceeds over several rounds and each round consists of several invocation of Joux's protocol. Ultimately, we are left out with three user sets and Joux's protocol is invoked once more to complete the protocol.*

Let $p = \lfloor \frac{n}{3} \rfloor$ and $r = n \bmod 3$. The set of users participating in a session is partitioned into three user sets $\mathsf{US}_1, \mathsf{US}_2, \mathsf{US}_3$ with respective cardinalities being p, p, p if $r = 0$; $p, p, p+1$ if $r = 1$; and $p, p+1, p+1$ if $r = 2$. This top down recursive procedure is invoked for further partitioning to obtain a ternary tree structure (*cf.* Figure 4.2). The lowest level 0 consists of singleton users having a secret key. We invoke $\mathsf{CombineTwo}$, a key agreement protocol for two user sets and $\mathsf{CombineThree}$, a key agreement protocol for three user sets in the key tree thus obtained. We demonstrate these two procedure in Figure 4.1.

All communications are done by representatives and users in each user set have a common agreed key. In $\mathsf{CombineThree}$, a, b, c respectively are the common agreed key of user sets A, B, C. Representative of user set A sends aP to both the user sets B, C. Similarly, representative of B sends bP to both A, C and representative of C sends cP to both A, B. After these communications, each user can compute the common agreed key $H(e(P,P)^{abc})$. In $\mathsf{CombineTwo}$, users in user set A have common agreed key a, users in user set B have common agreed key b. Representative of A sends aP to user set B and representative of B sends bP to user set A. Moreover, representative of user set A generates a random key $\hat{a} \in Z_q^*$ and sends $\hat{a}P$ to all the users in both A, B. After these communications, each user can compute the common agreed key $H(e(P,P)^{a\hat{a}b})$. Finally,

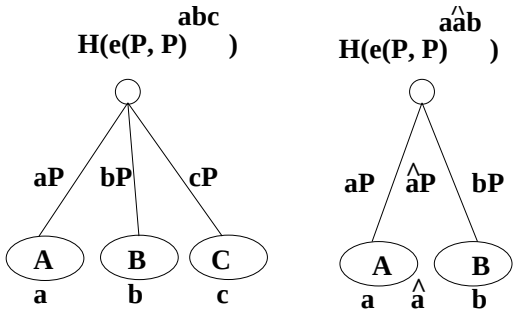

Figure 4.1: **procedure** CombineThree and **Procedure** CombineTwo

in KeyAgreement, we partition the group of users to obtain a ternary tree structure and invoke procedures CombineTwo and CombineThree to agree upon a common key among the group of users.

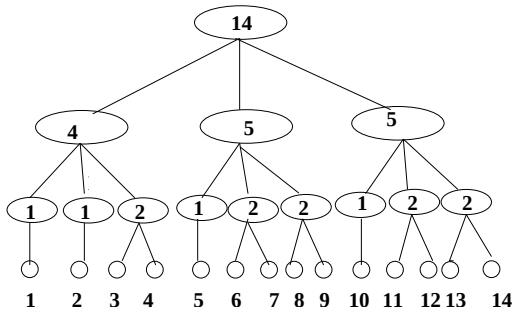

Figure 4.2: **procedure** KeyAgreement for $n = 14$

The formal description of the protocol is given below.

procedure KeyAgreement$(l, \mathsf{US}[i+1, \ldots, i+l], \mathsf{S}[i+1, \ldots, i+l])$
1. **if** $(l = 2)$ **then**
2. **call** CombineTwo$(\mathsf{US}[i+1, i+2], \mathsf{S}[i+1, i+2])$;
3. return;
4. **end if**
5. **if** $(l = 3)$ **then**
6. **call** CombineThree$(\mathsf{US}[i+1, i+2, i+3], \mathsf{S}[i+1, i+2, i+3])$;
7. return;
8. **end if**
9. $p_0 = 0$; $p_1 = \lfloor l/3 \rfloor$; $p_3 = \lceil l/3 \rceil$; $p_2 = l - p_1 - p_3$;
10. $n_0 = 0$; $n_1 = p_1$; $n_2 = p_1 + p_2$;
11. **for** $j = 1$ to 3 **do in parallel**
12. $\widehat{\mathsf{US}}_j = \mathsf{US}[i + n_{j-1} + 1, \ldots, i + n_{j-1} + p_j]$;
13. **if** $p_j = 1$, **then** $\widehat{\mathsf{S}}_j = S[i + n_{j-1} + 1]$;
14. **else**

15. **call** KeyAgreement$(p_j, \widehat{\mathsf{US}}_j, \mathsf{S}[i+n_{j-1}+1,\ldots,i+n_{j-1}+p_j])$;
16. Let $\widehat{\mathsf{S}}_j$ be the common agreed key among all members of $\widehat{\mathsf{US}}_j$;
17. **end if;**
18. **end for**;
19. **call** CombineThree$(\widehat{\mathsf{US}}[1,2,3], \widehat{\mathsf{S}}[1,2,3])$;

end KeyAgreement

procedure CombineTwo$(\mathsf{US}[1,2], \mathsf{S}[1,2])$
1. **do** Steps 2 and 3 **in parallel**
2. US_1 generates $\mathsf{S} \in_R Z_q^*$ and sends $\mathsf{S}P$ and S_1P to US_2;
3. US_2 sends S_2P to US_1;
4. **end do**;
5. **do** steps 6 and 7 **in parallel**
6. US_1 computes $H(e(\mathsf{S}_2P, SP)^{\mathsf{S}_1})$;
7. US_2 computes $H(e(\mathsf{S}_1P, SP)^{\mathsf{S}_2})$;
8. **end do**;

end CombineTwo

procedure CombineThree$(\mathsf{US}[1,2,3], \mathsf{S}[1,2,3])$
1. **for** $i = 1$ to 3 **do in parallel**
2. Let $\{j,k\} = \{1,2,3\} \setminus \{i\}$;
3. Rep(US_i) sends S_iP to all members $\mathsf{US}_j \cup \mathsf{US}_k$;
4. **end for**;
5. **for** $i = 1$ to 3 **do in parallel**
6. let $\{j,k\} = \{1,2,3\} \setminus \{i\}$;
7. each member of US_i computes $H(e(\mathsf{S}_jP, \mathsf{S}_kP)^{\mathsf{S}_i})$;
8. **end for**;

end CombineThree

The start of the recursive protocol KeyAgreement is made by the following statements:

start main
1. $\mathsf{US}_j = \{U_j\}$ for $1 \le j \le n$;
2. User j chooses a secret $s_j \in_R Z_q^*$;
3. User j sets $\mathsf{S}[j] = s_j$;
4. **call** KeyAgreement$(n, \mathsf{US}[1,\ldots,n], \mathsf{S}[1,\ldots,n])$.

end main

The values $s_1, \ldots, s_n$ are session specific and determine the final common key for the users. Note that CombineTwo is invoked only for two individual users, (i.e. $|\mathsf{US}_1| = |\mathsf{US}_2| = 1$), whereas CombineThree is invoked for three individual users as well as for three groups of users. In CombineThree

the common agreed key of user sets $\mathsf{US}_1, \mathsf{US}_2, \mathsf{US}_3$ is $H(e(P,P)^{\mathsf{S}_1\mathsf{S}_2\mathsf{S}_3})$ and in CombineTwo the common agreed key of the two users in the singleton sets $\mathsf{US}_1, \mathsf{US}_2$ is $H(e(P,P)^{\mathsf{S}_1\mathsf{S}_2\mathsf{S}})$.

The protocol described above allows $U_1, \ldots, U_n$ to agree upon a common key. The same protocol can be used by an arbitrary subset of $\{U_1, \ldots, U_n\}$ to agree upon a common key. This feature will be explained in more details for the authenticated protocol.

The algorithm KeyAgreement is recursive and goes through several levels starting with level 0. In each level, there are sets of users who have (or agree upon) a common secret key. In level 0, each user is in a set by himself/ herself and the secret key of the singleton set is the secret key of the concerned user. For n users, let the levels be numbered $0, \ldots, R(n)$. In level i, let the number of user sets be n_i. Thus $n_0 = n$, and $n_k = 1$, where $k = R(n)$. We identify the rounds of the algorithm as follows. There are $R(n)$ rounds of computation. The i^{th} round of computation takes the state of the algorithm from level $i-1$ to level i for $i = 1, \ldots, R(n)$. We introduce some notations for convenience of analyzing the algorithm.
–$U_j^{(i)}$: the j-th user set at level $i, 0 \leq i \leq R(n), 1 \leq j \leq n_i$,
–$s_j^{(i)}$: common secret key agreed upon by users in the user set $U_j^{(i)}$,
–$P_j^{(i)}$: i-th level j-th public key, *i.e.* $P_j^{(i)} = s_j^{(i)} P$.
For $n = 10$, the working of the algorithm (unauthenticated version) is shown in Figure 4.3.

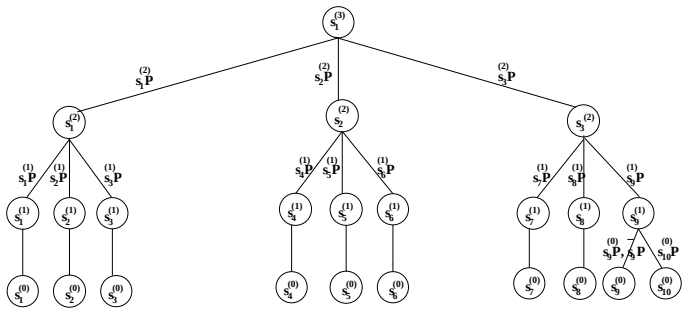

Figure 4.3: Key agreement among $n = 10$ parties (user U_9 generates $\overline{s}_9$ randomly and sends $\overline{s}_9 P$ to user U_{10} at the first level)

Lemma 4.2.1 *The final agreed key KEY among n users in the subroutine* KeyAgreement *is*

$$KEY = s_1^{(k)} = H(e(P,P)^{s_1^{(k-1)} s_2^{(k-1)} s_3^{(k-1)}})$$

where $k = R(n)$, $n > 2$.

Lemma 4.2.2 *Each member of $U_j^{(i)}$ can compute $s_j^{(i)}$ for $1 \leq j \leq n_i$, $i \leq k$ where $k = R(n)$. Consequently, all users are able to compute the common key $KEY = s_1^{(k)}$.*

Proof : We prove this lemma by induction on i.
Base Step : Initially $U_j^{(0)} = U_j$ and private key $s_j^{(0)}$ is assigned to each user for $1 \leq j \leq n_0$. So each

member of $U_j^{(1)}, 1 \leq j \leq n_1$ can compute $s_j^{(1)}$.
Induction Step: Let for $2 \leq i \leq k, 1 \leq j \leq n_{i-1}$, each member of user set $U_j^{(i-1)}$ has computed $s_j^{(i-1)}$ at the $(i-1)^{th}$ round. Note that KeyAgreement never calls CombineTwo for round $i \geq 2$. Now at the i-th round, for $j = 1, \ldots, n_i$, KeyAgreement calls CombineThree in which user set $U_j^{(i)}$ computes $s_j^{(i)}$ using the corresponding s of the $(i-1)^{th}$ round and user set $U_j^{(i)}$ consists of all members of $U_{3j-2}^{(i-1)}, U_{3j-1}^{(i-1)}$ and $U_{3j}^{(i-1)}$, all of who know the corresponding s by induction hypothesis. So each member of $U_j^{(i)}$ can compute the required $s_j^{(i)}$. Hence the proof follows. ■

One question arises here: How does a user U_u $(1 \leq u \leq n)$, in a certain round $i \geq 1$ determine to which user set he belongs and whether he is the representative of that user set? Using the above recursions, user u can easily compute all the user sets $U_j^{(i)}, 1 \leq j \leq n_i$, of this round and can efficiently check its position. Note that each $U_j^{(i)}$ is a subset of $\{U_1, U_2, \ldots, U_n\}$. So if $U_u \in U_j^{(i)}$ for some j, $1 \leq j \leq n_i$, user U_u verifies whether u equals $\min\{k : U_k \in U_j^{(i)}\}$ and if so, U_u is the representative of the user set $U_j^{(i)}$.

4.2.2 Authenticated Key Agreement Protocol

In this section, we describe our authenticated protocol. Authentication is obtained by incorporating signature schemes into the unauthenticated protocol described above. The idea is to authenticate each message by appending a signature with it. In keeping with our tree-structure, we require a multi-signature scheme. Each user set has a representative. For a user set, each user computes a signature on the message to be transmitted by using a digital signature scheme and sends the signature to the representative of that user set, which in turn computes a multi-signature on that message and sends the message, multi-signature pair to the other user sets. More specifically, we use the Boneh-Lynn-Shacham (BLS) [26] short signature and the Boldyreva [18] multi-signature scheme based on it. A description of BLS short signature scheme is given in Section A.3.1 and that of Boldyreva's multi-signature scheme is provided in Section A.3.3.

As part of the two signature schemes each user U_i chooses a signing and a verification key sk_i(or sk_{U_i}) and pk_i(or pk_{U_i}) respectively.

An important issue in authenticating the protocol is that of session ID. Recall that Π_U^i is the ith instance of user U. Suppose instances $\Pi_{U_{i_1}}^{d_1}, \ldots, \Pi_{U_{i_k}}^{d_k}$ wish to agree upon a common key. According to our definition, $\mathsf{sid}_{U_{i_j}}^{d_j} = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ (see section 2.6.3 of Chapter 2). At the start of a session, $\Pi_{U_{i_j}}^{d_j}$ need not know the entire set $\mathsf{sid}_{U_{i_j}}^{d_j}$. This set is built up as the protocol proceeds. Of course, we assume that $\Pi_{U_{i_j}}^{d_j}$ knows the pair (U_{i_j}, d_j). Clearly, $\Pi_{U_{i_j}}^{d_j}$ knows U_{i_j}. Knowledge of d_j can be maintained by U_{i_j} by keeping a counter which is incremented when a new instance is created. Each instance keeps the partial information about the session ID in a variable psid_U^i. Before the

start of a session an instance $\Pi_{U_{i_j}}^{d_j}$ sets $\mathsf{psid}_{U_{i_j}}^{d_j} = \{(U_{i_j}, d_j)\}$. On the other hand, we do assume that any instance $\Pi_{U_{i_j}}^{d_j}$ knows its partner ID $\mathsf{pid}_{U_{i_j}}^{d_j}$, i.e., the set of users with which it is partnered in the particular session.

Another issue is the numbering of the messages. Any instance Π_U^i sends out a finite number of messages, which can be uniquely numbered by Π_U^i based on their order of occurrence. The number of instances participating in a session determines the leaf level of the ternary key agreement tree which in turn uniquely determines the ternary tree structure. Once the ternary tree structure is defined, the numbering of the messages is also defined uniquely. More specifically, if US is a user set at some intermediate point in the execution of a session and $U = \mathsf{Rep}(\mathsf{US})$, then all instances in US knows the number of the next message to be sent out by U.

We now explain the signing part of the protocol. All messages are exchanged as part of CombineTwo and CombineThree and hence signing is also introduced as part of CombineTwo and CombineThree. The call to CombineTwo and CombineThree in Steps 2 and 6 of KeyAgreement involves individual users whereas the call to CombineThree in Step 19 of KeyAgreement is on three sets of users. In the first case, we require only the BLS signature scheme, whereas in the second case we require the multi-signature scheme.

We now describe the modified versions of CombineTwo and CombineThree. In these algorithms, US will represent sets of instances (as opposed to sets of users as in Section 4.2.1). In case US is a singleton set (as in AuthCombTwo and AuthCombThree-A) we will identify US with the instance it contains. We also define $\mathsf{Rep}(\mathsf{US}) = \Pi_{U_j}^{d_j}$ where$j = \min\{k : \Pi_{U_k}^{d_k} \in \mathsf{US}\}$.

AuthCombTwo(US[1,2],S[1,2])
1. **perform** Steps 2,3 and 4 **in parallel** with Steps 5,6, and 7;
2. $\mathsf{US}_1 = \Pi_{U_1}^{d_1}$ generates $\mathsf{S} \in_R Z_q^*$ and forms $m_1 = (\mathsf{S}P, \mathsf{S}_1 P)$;
3. US_1 computes $\sigma_1 = \mathcal{S}(sk_{U_1}, \mathsf{US}_1|1|m_1)$;
4. US_1 sends $\mathsf{US}_1|1|m_1|\sigma_1$ to US_2;
5. $\mathsf{US}_2 = \Pi_{U_2}^{d_2}$ sets $m_2 = \mathsf{S}_2 P$;
6. US_2 computes $\sigma_2 = \mathcal{S}(sk_{U_2}, \mathsf{US}_2|1|m_2)$;
7. US_2 sends $\mathsf{US}_2|1|m_2|\sigma_2$ to US_1;
8. **end**;
9. **for** $i = 1, 2$ **do in parallel**
10. set $i' = 3 - i$; let $\mathsf{US}_i = \Pi_{U_i}^{d_i}$; $\mathsf{US}_{i'} = \Pi_{U_{i'}}^{d_{i'}}$;
11. US_i verifies $\sigma_{i'}$ on $\mathsf{US}_{i'}|1|m_{i'}$ using $pk_{U_{i'}}$ and algorithm $\mathcal{V}$;
12. **if** verification fails, **then** US_i sets $\mathsf{acc}_{U_i}^{d_i} = 0$, $\mathsf{sk}_{U_i}^{d_i} = \mathsf{NULL}$ and aborts;
13. **else** US_i sets $\mathsf{psid}_{U_i}^{d_i} = \mathsf{psid}_{U_i}^{d_i} \cup \{(U_{i'}, d_{i'})\}$;
14. **end for**;
15. US_1 computes $H(e(\mathsf{S}_2 P, \mathsf{S}P)^{\mathsf{S}_1}$;
16. US_2 computes $H(e(\mathsf{S}_1 P, \mathsf{S}P)^{\mathsf{S}_2}$;

end AuthCombTwo

In AuthCombTwo, user instance $\Pi_{U_1}^{d_1}$ has a secret key s_1 with partial session-identity $\mathsf{psid}_{U_1}^{d_1} = \{(U_1, d_1)\}$. Besides it generates a random key $s \in Z_q^*$, computes a signature σ_1 using basic signature scheme DSig on $m_1 = (s_1 P, sP)$ and sends $U_1|1|m_1|\sigma_1$ to user instance $\Pi_{U_2}^{d_2}$. Similarly, user instance $\Pi_{U_2}^{d_2}$ with secret key s_2 and partial session-identity $\mathsf{psid}_{U_2}^{d_2} = \{(U_2, d_2)\}$ computes a basic signature σ_2 on $m_2 = s_2 P$ and sends $U_2|1|m_2|\sigma_2$ to user instance $\Pi_{U_1}^{d_1}$. On receiving the message $U_2|1|m_2|\sigma_2$, user U_1 verifies σ_2 on $U_2|1|m_2$ according to the verification algorithm $\mathcal{V}$ of DSig. If verification fails, it sets $\mathsf{acc}_{U_1}^{d_1} = 0$, $\mathsf{sk}_{U_1}^{d_1} = \mathsf{NULL}$ and aborts. Otherwise it sets $\mathsf{psid}_{U_1}^{d_1} = \mathsf{psid}_{U_1}^{d_1} \cup \{(U_2, d_2)\}$ and computes the key $H(e(P,P)^{ss_1s_2})$. In a similar way, user U_2 verifies σ_1 on $U_1|1|m_1$ and computes the key only if verification succeeds. Note that at the end, user instances $\Pi_{U_1}^{d_1}, \Pi_{U_2}^{d_2}$ have the same partial session-identity, namely $\mathsf{psid}_{U_1}^{d_1} \cup \mathsf{psid}_{U_2}^{d_2}$.

There are two versions of CombineThree for the authenticated protocol: AuthCombThree-A and AuthCombThree-B. The description of algorithm AuthCombThree-A is analogous to that of AuthCombTwo.

AuthCombThree-A(US[1, 2, 3],S[1, 2, 3])
1. **for** $i = 1, 2, 3$ **do in parallel**
2. set $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$;
3. $\mathsf{US}_i (= \Pi_{U_i}^{d_i})$ computes $\sigma_i = \mathcal{S}(sk_{U_i}, \mathsf{US}_i|1|\mathsf{S}_i P)$;
4. US_i sends $\mathsf{US}_i|1|\mathsf{S}_i P|\sigma_i$ to US_j and US_k;
5. **end for**;
6. **for** $i = 1, 2, 3$ **do in parallel**
7. set $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$; let $\mathsf{US}_i = \Pi_{U_i}^{d_i}$; $\mathsf{US}_j = \Pi_{U_j}^{d_j}$; $\mathsf{US}_k = \Pi_{U_k}^{d_k}$;
8. US_i verifies
9. • σ_j on $\mathsf{US}_j|1|\mathsf{S}_j P$ using pk_{U_j};
10. • σ_k on $\mathsf{US}_k|1|\mathsf{S}_k P$ using pk_{U_k};
11. **if** any of the above verification fails **then**
12. US_i sets $\mathsf{acc}_{U_i}^{d_i} = 0$, $\mathsf{sk}_{U_i}^{d_i} = \mathsf{NULL}$ and aborts;
13. **else**
14. US_i sets $\mathsf{psid}_{U_i}^{d_i} = \mathsf{psid}_{U_i}^{d_i} \cup \{(U_j, d_j), (U_k, d_k)\}$;
15. US_i computes $H(e(\mathsf{S}_j P, \mathsf{S}_k P)^{\mathsf{S}_i})$;
16. **end if**;
17. **end for**;

end AuthCombThree-A

The next authenticated variation of CombineThree uses multi-signature. We need a notation to describe the algorithm. Suppose $\mathsf{psid}_U^i = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ is the (partial) session ID for instance Π_U^i. Then $\mathsf{First}(\mathsf{psid}_U^i)$ is defined to be the set $\{U_{i_1}, \ldots, U_{i_k}\}$. In the next algorithm, we use the notation $\Pi_V^{d_V}$ to denote an instance of a user V. The value of the instance number d_V is not uniquely determined by V. However, it is unique for one particular session. Since one particular invocation of the next algorithm will involve only one session, there will be no confusion in the use of the notation d_V.

AuthCombThree-B(US[1,2,3],S[1,2,3])
1. **for** $i = 1, 2, 3$ **do in parallel**
2. set $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$; $\Pi_U^{d_U} = \mathsf{Rep}(\mathsf{US}_i)$;
3. **for** each $\Pi_V^{d_V} \in \mathsf{US}_i$ **do in parallel**
4. $\Pi_V^{d_V}$ computes $\sigma_V = \mathcal{S}(sk_V, \mathsf{psid}_V^{d_V} | t_i | \mathsf{S}_i P)$, where t_i is number of next message to be sent by $\Pi_U^{d_U}$;
5. $\Pi_V^{d_V}$ sends (V, σ_V) to U;
6. **end for**;
7. $\Pi_U^{d_U}$ computes the multi-signature $\sigma_{\mathsf{US}_i} = \mathcal{MS}(\{\sigma_V : V \in \mathsf{US}_i\})$;
8. $\Pi_U^{d_U}$ sends $\mathsf{psid}_U^{d_U} | t_i | \mathsf{S}_i P | \sigma_{\mathsf{US}_i}$ to $\mathsf{US}_j \cup \mathsf{US}_k$;
9. **end for**;
10. **for** $i = 1, 2, 3$ **do in parallel**
11. set $\{j, k\} = \{1, 2, 3\} \setminus \{i\}$; $\Pi_{W_j}^{d_{W_j}} = \mathsf{Rep}(\mathsf{US}_j)$; $\Pi_{W_k}^{d_{W_k}} = \mathsf{Rep}(\mathsf{US}_k)$;
12. **for** each $\Pi_V^{d_V} \in \mathsf{US}_i$ **do in parallel**
13. $\Pi_V^{d_V}$ receives $\mathsf{psid}_{W_j}^{d_{W_j}} | t_j | \mathsf{S}_j P | \sigma_{\mathsf{US}_j}$ from W_j and $\mathsf{psid}_{W_k}^{d_{W_k}} | t_k | \mathsf{S}_k P | \sigma_{\mathsf{US}_k}$ from W_k;
14. Let $F_j = \mathsf{First}(\mathsf{psid}_{W_j}^{d_{W_j}})$ and $F_k = \mathsf{First}(\mathsf{psid}_{W_k}^{d_{W_k}})$;
15. $\Pi_V^{d_V}$ verifies
16. • F_j and F_k are subsets of $\mathsf{pid}_V^{d_V}$;
17. • t_j and t_k are next expected message numbers from $\Pi_{W_j}^{d_{W_j}}$ and $\Pi_{W_k}^{d_{W_k}}$ respectively;
18. • σ_{US_j} is the multi-signature of F_j on $\mathsf{psid}_{W_j}^{d_{W_j}} | t_j | \mathsf{S}_j P$;
19. • σ_{US_k} is the multi-signature of F_k on $\mathsf{psid}_{W_k}^{d_{W_k}} | t_k | \mathsf{S}_k P$;
20. **if** any of the above verification fails, **then**
21. $\Pi_V^{d_V}$ sets $\mathsf{acc}_V^{d_V} = 0$, $\mathsf{sk}_V^{d_V} = \mathsf{NULL}$ and aborts;
22. **else**
23. $\Pi_V^{d_V}$ computes $H(e(\mathsf{S}_j P, \mathsf{S}_k P)^{\mathsf{S}_i})$;
24. $\Pi_V^{d_V}$ sets $\mathsf{psid}_V^{d_V} = \mathsf{psid}_V^{d_V} \cup \mathsf{psid}_{W_j}^{d_{W_j}} \cup \mathsf{psid}_{W_k}^{d_{W_k}}$;
25. **end if**;
26. **end for**;
27. **end for**;

end AuthCombThree-B

The algorithm AuthCombThree-B performs key agreement among three user sets $\mathsf{US}_1, \mathsf{US}_2, \mathsf{US}_3$ as follows: Suppose $\Pi_{U_1}^{d_1}$ be the representative of user set US_1 and users in this set have a common agreed key s_1. Similarly, $\Pi_{U_2}^{d_2}$, s_2 are those for user set US_2 and $\Pi_{U_3}^{d_3}$, s_3 for user set US_3. Let $m_1 = \mathsf{psid}_{U_1}^{d_1} | t_1 | s_1 P$, where t_1 is the next expected message number to be sent by $\Pi_{U_1}^{d_1}$. For each user V in user set US_1, $\mathsf{psid}_V^{d_V} | t_1 | s_1 P$ is same as m_1. Each user $V \in \mathsf{US}_1$ computes a basic signature σ_V on m_1 using the scheme DSig and sends (V, σ_V) to $\Pi_{U_1}^{d_1}$. After accumulating these basic signatures, the representative $\Pi_{U_1}^{d_1}$ constructs the multi-signature msig_1 on message m_1 using

the scheme MSig and sends $m_1|\mathsf{msig}_1$ to $\mathsf{US}_2 \cup \mathsf{US}_3$. Similarly, representative $\Pi_{U_2}^{d_2}$ of user set US_2 sends $m_2|\mathsf{msig}_2$ to $\mathsf{US}_1 \cup \mathsf{US}_3$ and representative $\Pi_{U_3}^{d_3}$ of user set US_3 sends $m_3|\mathsf{msig}_3$ to $\mathsf{US}_1 \cup \mathsf{US}_2$ where $m_2 = \mathsf{psid}_{U_2}^{d_2}|t_2|s_2P$ and $m_3 = \mathsf{psid}_{U_3}^{d_3}|t_3|s_3P$, t_2, t_3 being the next expected message number to be sent by $\Pi_{U_2}^{d_2}$, $\Pi_{U_3}^{d_3}$ respectively.

We define a variable $\mathsf{First}(\mathsf{psid}_U^d)$ for the ease of discussion. If $\mathsf{psid}_U^i = \{(U_{i_1}, d_{i_1}), \ldots, (U_{i_k}, d_{i_k})\}$, then $\mathsf{First}(\mathsf{psid}_U^i) = \{U_{i_1}, \ldots, U_{i_k}\}$. Now on receipt of messages $m_2|\mathsf{msig}_2$ and $m_3|\mathsf{msig}_3$, each user instance $\Pi_V^{d_V} \in \mathsf{US}_1$ (1) checks $\mathsf{First}(\mathsf{psid}_{U_2}^{d_2}) \subseteq \mathsf{pid}_V^{d_V}$ and $\mathsf{First}(\mathsf{psid}_{U_3}^{d_3}) \subseteq \mathsf{pid}_V^{d_V}$; (2) verifies t_2, t_3 are the next expected message number to be sent by $\Pi_{U_2}^{d_2}$, $\Pi_{U_3}^{d_3}$ respectively; (3) verifies $\mathsf{msig}_2, \mathsf{msig}_3$ are multi–signatures on m_2, m_3 respectively. If any of these verification fails, $\Pi_V^{d_V}$ sets $\mathsf{acc}_V^{d_V} = 0$ and $\mathsf{sk}_V^{d_V} = \mathsf{NULL}$ and aborts. Otherwise computes the common key $H(e(P,P)^{s_1 s_2 s_3})$ and sets $\mathsf{psid}_V^{d_V} = \mathsf{psid}_V^{d_V} \cup \mathsf{psid}_{U_2}^{d_2} \cup \mathsf{psid}_{U_3}^{d_3}$.

Similar verifications are done by each user in sets US_2 and US_3. Common key $H(e(P,P)^{s_1 s_2 s_3})$ is computed only if verification holds. Note that at the end of an honest execution of this protocol, each user in the group $\mathsf{US}_1 \cup \mathsf{US}_2 \cup \mathsf{US}_3$ has a common partial session-identity.

The calls of CombineTwo and CombineThree from KeyAgreement are modified as follows:

Step 2 : change to **call** AuthCombTwo($\mathsf{US}[i+1, i+2], \mathsf{S}[i+1, i+2]$);
Step 6 : change to **call** AuthCombThree-A($\mathsf{US}[i+1, i+2, i+3], \mathsf{S}[i+1, i+2, i+3]$);
Step 19 : change to **call** AuthCombThree-B($\widehat{\mathsf{US}}[1,2,3], \widehat{\mathsf{S}}[1,2,3]$);

The start of the protocol between instances $\Pi_{U_1}^{d_1}, \ldots, \Pi_{U_k}^{d_k}$ is made as follows:

start session($\Pi_{U_{i_1}}^{d_1}, \ldots, \Pi_{U_{i_k}}^{d_k}$)
1. **for** $j = 1$ to k
2. instance $\Pi_{U_{i_j}}^{d_j}$ chooses a secret key $s_j \in_R Z_q^*$ and sets $\mathsf{S}[j] = s_j$;
3. set $\mathsf{US}_j = \{\Pi_{U_{i_j}}^{d_j}\}$; $\mathsf{psid}_{U_{i_j}}^{d_j} = \{(U_{i_j}, d_j)\}$;
4. **end for**;
5. **call** KeyAgreement($k, \mathsf{US}[1, \ldots, k], \mathsf{S}[1, \ldots, k]$).

end session

Figure 4.4 illustrates the working of the procedures AuthCombTwo, AuthCombThree-A and AuthCombThree-B. The following property is easy to verify from the description of the protocol.

Proposition 4.2.3 *Suppose* US *is a set of instances occurring in* AuthCombThree-B *in some session. Then for any two instances* $\Pi_{U_1}^{i_1}$ *and* $\Pi_{U_2}^{i_2}$ *participating in the session with* $\Pi_{U_1}^{i_1}, \Pi_{U_2}^{i_2} \in \mathsf{US}$, *we have* $\mathsf{psid}_{U_1}^{i_1} = \mathsf{psid}_{U_2}^{i_2}$. *Moreover, after the completion of a session in which* Π_U^i *participates, we have* $\mathsf{psid}_U^i = \mathsf{sid}_U^i$.

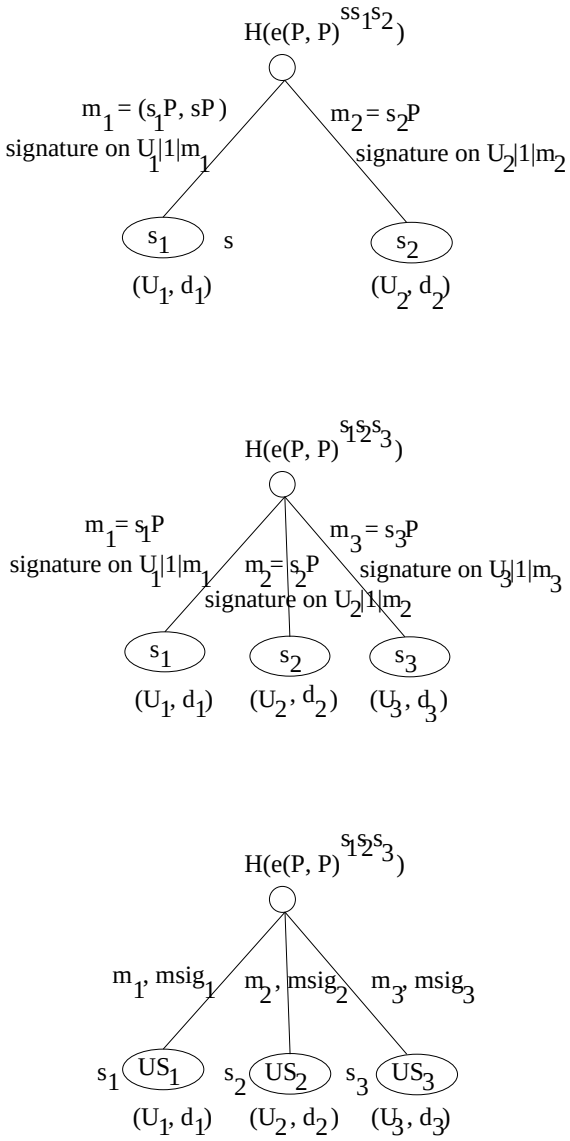

Figure 4.4: **procedure** AuthCombTwo, AuthCombThree-A and AuthCombThree-B

One consequence of the first statement of Proposition 4.2.3 is that in AuthCombThree-B, each user in a user set computes the signature on the same message. Further, the second statement assures us that the partial session IDs finally "grow" into full session IDs.

Note that our protocol does not include key verification between users. This can be achieved at extra computational and communication cost.

4.2.3 Dynamic Key Agreement Protocol

Dynamic key agreement consists of a key agreement protocol together with two additional algorithms, Join and Leave. The procedure Join enables a user to join a group. A user can leave a group by invoking the procedure Leave. In this subsection, we describe protocols for join and leave for the above static tree-based authenticated protocol. Our protocol design makes an optimal use of the data precomputed in the procedure KeyAgreement. When a user joins or leaves a group, the structure of the key tree is disturbed and requires to be updated for any subsequent join or leave operation. Maintaining the tree structure of the key agreement protocol is a crucial part of our scheme. We refer this as the preservation of the structure of the procedure KeyAgreement.

Suppose we have a key tree T of n users $\{1, 2, \ldots, n\}$ according to the key agreement described in Section 4.2.1 with $k = R(n)$ rounds. For the sake of easy description, we take the user set $\mathcal{P} = \{1, \ldots, n\}$ instead of the set $\{U_1, \ldots, U_n\}$ and introduce some more notations. For $1 \leq l \leq k$, let $U_i^{(l)}$ be the i-th user set at level l and $s_i^{(l)}$ be the common agreed key of users in the user set $U_i^{(l)}$ at level l. Initially, $U_i^{(0)} = \{i\}$, $s_i^{(0)}$ is the private key randomly chosen by user i from Z_q^*.

Join

Suppose a new user $\{u\}$ with private key x requests to join the group $\{1, \ldots, n\}$. He joins the tree in such a way that the structure of KeyAgreement is still preserved and updating of key path is optimal in the sense that minimum updating or recomputation is required. For instance, consider a group key agreement with $n = 10$ members. In this case, the root node will have 3 subtrees, with the left and middle subtrees having 3 leaves each and the right subtree having 4 leaves. Now suppose a new user wants to join this group. He cannot join the first subgroup (of subusers) since this is contrary to the way we partition the user sets. So the entire group of users will have to be repartitioned. Similarly, he cannot join the third subgroup (of 4 users) without causing repartitioning. But if he joins the second subtree, then there is no need of repartitioning and so key updating is minimal and is only along a single path. This is illustrated in Figure 4.5.

We will show in Lemma 4.2.4 that such a path can be uniquely determined. We call this unique path the optimal path of joining of the new user.

We now describe our protocol Join. This algorithm invokes a procedure FindKeyPath to find the optimal key path path of joining of a new member and updates path according to the algorithm UpdateKeyPath. When a new user wants to join a group, it should join in such a way that the tree structure is preserved and minimum key updates are required. The key updates along the unique

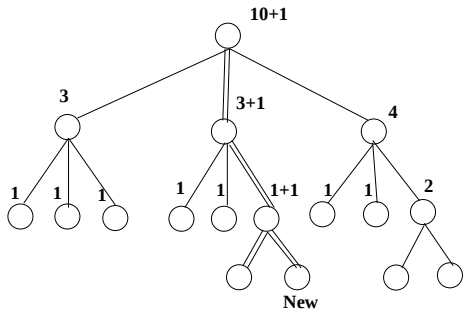

Figure 4.5: **procedure** Join

optimal key path ensures the minimum modification to the computation already precomputed when a new user joins the group, retaining of course the structure of the tree. As mentioned earlier, maintaining the tree structure is a crucial part of our scheme for any subsequent join or leave operation.

The new user who has permission to join the group for a group key agreement computes its optimal key path of joining $i_k, \ldots, i_1; N_k, \ldots, N_1$ by using algorithm FindKeyPath, communicate this to all other members of the group and invokes algorithm UpdateKeyPath to update this path. *We assume that the new user as well as others behave honestly and do not deviate from the protocol execution.*

The algorithm UpdateKeyPath works as follows to update keys in level 1 on joining of the new user. In the key tree T with n users, the number of children of the node i_1 (node at level 1) in the optimal key path is either 1 or 2 or 3. If i_1 has 1 leaf node, then the user corresponding to this leaf node chooses a new private key, agree upon a common key with the new user by invoking algorithm AuthCombTwo and the corresponding user set for level 1 is modified to a set that includes these two users. In case i_1 has 2 leaf nodes, the users corresponding to these leaves choose new private keys, a new user set for level 1 is constructed that contains these two users and the new user and algorithm AuthCombThree-A is invoked to agree upon a common key among them. If i_1 has 3 leaves, then the users corresponding to these leaves choose new private keys. The rightmost user agree upon a common key with the new user by invoking algorithm AuthCombTwo and constructs a new user set that consists of the new user and itself. Then AuthCombThree-B is invoked for this new user set and the other two leaves of i_1 to agree upon a common key. Finally the corresponding user set for level 1 is modified to a set that includes these users.

The subsequent user sets are accordingly changed by algorithm UpdateKeyPath and key updates in level $l+1$ ($1 \leq l \leq k-1$) are done by invoking algorithm AuthCombThree-B among the three user sets which are subtrees of node i_{l+1}. The modified user set corresponding to the node i_l invokes AuthCombThree-B to agree upon a common key with the user sets corresponding to the other two subtrees (siblings of i_l) of node i_{l+1} and a new user set for level $l+1$ is constructed that is the union of these three user sets. We proceed in this way and finally a common key is agreed among all the $n+1$ users. At the end, we newly index the members (leaf nodes) as $\{1, 2, \ldots, n+1\}$.

Leave

Suppose key tree T with n leaf nodes $\{1, 2, \ldots, n\}$ has the tree structure used in the procedure KeyAgreement and suppose a member j_0, $1 \leq j_0 \leq n$, wants to leave the group. On removal of the leaving node j_0, the tree structure will be disturbed. To retain the tree structure, algorithm Leave first invokes a procedure FindExtractNode to find a suitable leaf node, say l_0 (we call it extracted leaf node) in the key tree such that removal of this extracted leaf node from the key tree and adding it as the j_0-th leaf yields a new key tree where tree structure is still preserved. While finding the extracted leaf node l_0, we keep in mind the fact that we require minimal key updates.

We take a designated user, called group controller (GC) to initiate the operation Leave. Any user other than the leaving user can be GC. To be specific, we take one sibling of the node leaving the group as the GC. So when a member leaves the group, it shares a subkey with GC. However, this subkey will be recalculated by GC and the group key updated accordingly. This extra knowledge will no way enable the leaving member to gain any information of the updated common key. *We assume that all the users together with the* GC *are honest and do not deviate from the protocol execution.*

The procedure FindExtractNode uses a subroutine Extract that outputs the identity or index of a leaf node in T such that the structure of KeyAgreement is preserved in the tree after removal of this node. The procedure FindExtractNode outputs the index of the extracted leaf node l_0 to GC which will be inserted as the j_0-th leaf node in the key tree. It also outputs to GC the index of the internal node j_i which is the root of the highest level subtree with disturbed tree structure on removal of j_0.

Finally, after getting the index l_0 of the leaf node to be extracted (if required), the algorithm Leave updates the two key paths, one from root to the parent of the leaving leaf node j_0 and another from root to the parent of the extracted leaf node l_0.

The procedure Extract works as follows. If all the three subtrees T_L, T_M, T_R of the tree T have equal number of leaf nodes, then removal of a leaf node from the leftmost subtree T_L will not disturb the tree structure and so we can extract a leaf node from T_L. Similarly, if the number of leaves in both T_L, T_M are same, say p, and that of right subtree T_R is $p + 1$, then we can extract a leaf from T_R without disturbing the tree structure. If the number of leaves in T_M, T_R are same, say, $p + 1$ and that of T_L is p, then we can extract a leaf from T_M retaining the tree structure. We recursively apply this procedure on T_L, T_R or T_M chosen in this manner and finally reach to a leaf node. The index of the user corresponding to this leaf node is outputted to the GC.

Now a user corresponding to leaf j_0 leaves the group, the tree structure is disturbed. We first find the highest level, say i, for which the subtree rooted at the internal node at, say j_i, lacks the tree structure. Consequently, j_i is the root of the highest level subtree with disturbed tree structure on removal of j_0. To retain the tree structure, we may need to extract suitably a leaf node l_0 from the tree T in such a way that the key updates required are minimal. We do this by using an algorithm FindExtractNode that uses the procedure Extract as a subroutine and removes the leaving member j_0, still preserving the structure of KeyAgreement in the resulting key tree. This algorithm outputs

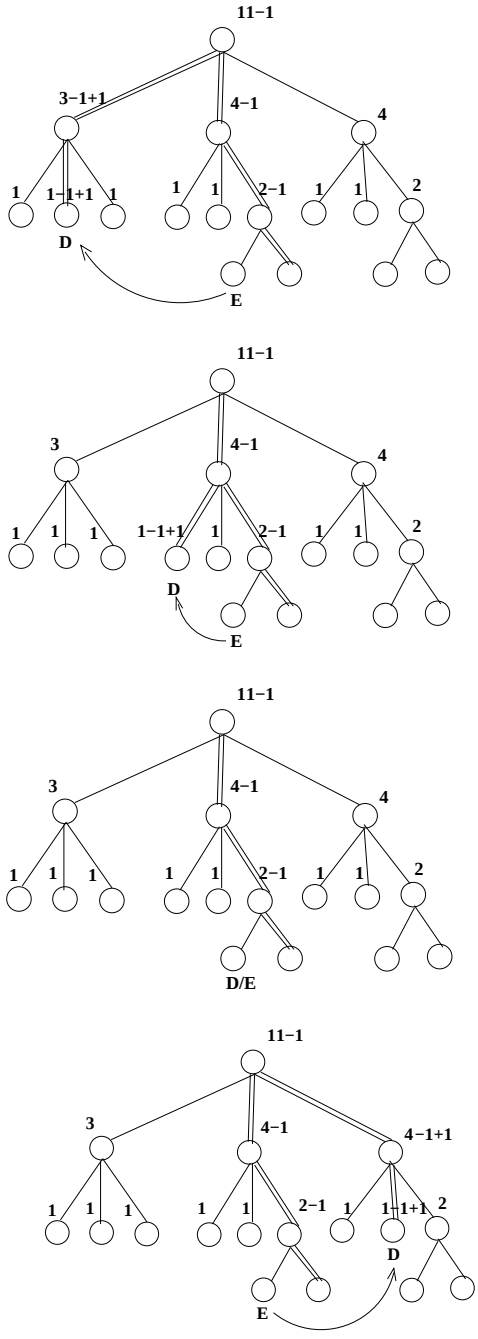

Figure 4.6: Different cases of **procedure** Leave with $n = 11$ (D denotes the leaf node to be removed and E denotes the node to be extracted to maintain the key structure)

the index of the extracted leaf node l_0 to GC and also the index of the internal node j_i.

Next we describe the algorithm Leave below. Our algorithm Leave invokes FindExtractNode to obtain the index l_0 of the node to be extracted (if required), finds from the tree T the path from root to the parent of the leaving leaf node j_0 and also the path from root to the parent of the extracted leaf node l_0. Two new user sets are constructed in level 1: one user set includes the user corresponding to l_0 together with the users corresponding to brothers of j_0 and another user set includes only the users corresponding to brothers of l_0. All the users in these two new user sets choose new private keys. The subsequent higher level user sets are modified accordingly and appropriate algorithms AuthCombTwo, AuthCombThree-A or AuthCombThree-B are invoked for successive key agreements in the key tree. We invoke AuthCombTwo for two party authenticated key agreement, AuthCombThree-A for three party authenticated key agreement and AuthCombThree-B for authenticated key agreement among three user sets. At the end of the procedure Leave, we newly index the users (leaf nodes) as $\{1, 2, \ldots, n-1\}$. Some particular cases are shown in Figure 4.6.

Next we state and prove Lemma 4.2.4 which uniquely determines the optimal path of joining of a new user.

Lemma 4.2.4 *For $1 \leq l \leq k$, $k = R(n)$, define the following:*
$i_l :=$ *the index of the node at level l whose subtree will contain the new user $\{u\}$ as a leaf,*
$\eta_l :=$ *the number of leaf nodes in the subtree at i_l,*
$N_l :=$ *number of leaf nodes to the left of node i_l and*
$r_l = \eta_l \bmod 3$.
Clearly $i_k = 1; \eta_k = n; r_k = n \bmod 3; N_k = 0$. Then, for $2 \leq l \leq k$, we have $i_{l-1} = 3i_l - r_l$; $\eta_{l-1} = \lfloor \frac{\eta_l}{3} \rfloor$; and $N_{l-1} = N_l + (2 - r_l)\eta_{l-1}$. Thus user u joins the subgroup at i_1 and $i_k, i_{k-1}, \ldots, i_1$ determine the optimal path along which the subgroup keys needs to be updated or recomputed.

Proof : By induction. The case $l = k$ is straightforward. Suppose the results hold for $l < k$. The node i_{l-1} is the $(3 - r_l)$-th child of the subtree T at i_l and there are $3(i_l - 1)$ nodes at level $(l - 1)$ to the left of subtree T in the original tree. Hence $i_{l-1} = 3(i_l - 1) + (3 - r_l) = 3i_l - r_l$.

The second one is easy to check. For the third one, we argue as follows: There are η_l leaf nodes in the subtree T at i_l and i_l has exactly three off-springs with subtrees T_L (left), T_M (middle) and T_R (right). If $r_l = 0$, *i.e.* if $|T_L| = |T_M| = |T_R|$, then i_{l-1} is the third child; if $r_l = 1$, then i_{l-1} is the second child and if $r_l = 2$, then i_{l-1} is the first child. In any case, the number of leaf nodes in the subtree at i_{l-1} is η_{l-1}.

In case 1 above, there are $2\eta_{l-1}$ leaf nodes to the left of node i_{l-1} in the subtree T at i_l. In case 2, there are η_{l-1} leaf nodes to the left and in case 3, there are zero leaf nodes to the left. In each case, there are $(2-r_l)\eta_{l-1}$ leaf nodes to the left of i_{l-1} in T. Since there are N_l leaf nodes to the left of i_l in the original tree, the total number of leaf nodes to the left of i_{l-1} in $N_l + (2-r_l)\ \eta_{l-1} = N_{l-1}$, by definition of N_{l-1}. This completes the proof. ■

The formal description of the procedures Join, FindKeyPath and UpdateKeyPath are given below.

procedure Join($[1,\ldots,n],\{u\}$)
1. **call** FindKeyPath(n);
2. Let path be the optimal key path of joining of the new member $\{u\}$.
3. **call** UpdateKeyPath(path);

end Join

procedure FindKeyPath(n)
1. $k = R(n); i_k = 1; N_k = 0; \eta_k = n; r_k = n \bmod 3;$
2. $l = k$ downto 2 **do**
3. $\quad i_{l-1} = 3i_l - r_l;\ \eta_{l-1} = \lfloor \frac{\eta_l}{3} \rfloor;\ r_{l-1} = \eta_{l-1} \bmod 3;\ N_{l-1} = N_l + (2 - r_l)\eta_{l-1};$
4. **end do**
5. **return** $(i_k,\ldots,i_1;\eta_k,\ldots,\eta_1;N_k,\ldots,N_1);$

end FindKeyPath

procedure UpdateKeyPath(path)
1. parse path $= (i_k,\ldots,i_1;\eta_k,\ldots,\eta_1;N_k,\ldots,N_1);$
2. The node i_1 at level 1 has η_1 children which are clearly leaf nodes. Note that $\eta_1 \le 3$.
3. **if** $\eta_1 = 1$ **then**
4. $\quad$ user $N_{i_1}+1$ chooses a new private key $s^{(0)}_{N_{i_1}+1} \in_R Z_q^*$;
5. $\quad \hat{U}_1 = U^{(0)}_{N_{i_1}+1}; \hat{s}_1 = s^{(0)}_{N_{i_1}+1};\ \hat{U}_2 = \{u\}; \hat{s}_2 = x;$
6. $\quad$ **call** AuthCombTwo($\hat{U}[1,2], \hat{s}[1,2]$);
7. $\quad s^{(1)}_{i_1}$ is assigned the agreed key between user sets $\hat{U}_1, \hat{U}_2$ and $\hat{U}^{(1)}_{i_1} = \hat{U}[1,2];$
8. **end if**
9. **if** $\eta_1 = 2$ **then**
10. $\quad$ users $N_{i_1}+1, N_{i_1}+2$ choose new private keys $s^{(0)}_{N_{i_1}+1}, s^{(0)}_{N_{i_1}+2} \in_R Z_q^*$ respectively;
11. $\quad \hat{U}_1 = U^{(0)}_{N_{i_1}+1}; \hat{s}_1 = s^{(0)}_{N_{i_1}+1};\ \hat{U}_2 = U^{(0)}_{N_{i_1}+2}; \hat{s}_2 = s^{(0)}_{N_{i_1}+2};\ \hat{U}_3 = \{u\}; \hat{s}_3 = x;$
12. $\quad$ **call** AuthCombThree $-$ A($\hat{U}[1,2,3], \hat{s}[1,2,3]$);
13. $\quad s^{(1)}_{i_1}$ is assigned the agreed key between user sets $\hat{U}_1, \hat{U}_2, \hat{U}_3$ and $\hat{U}^{(1)}_{i_1} = \hat{U}[1,2,3];$
14. **end if**
15. **if** $\eta_1 = 3$ **then**
16. $\quad$ users $N_{i_1}+1, N_{i_1}+2, N_{i_1}+3$ choose new private keys $s^{(0)}_{N_{i_1}+1}, s^{(0)}_{N_{i_1}+2}, s^{(0)}_{N_{i_1}+3}$ respectively;
17. $\quad \hat{U}_1 = U^{(0)}_{N_{i_1}+3}; \hat{s}_1 = s^{(0)}_{N_{i_1}+3};\ \hat{U}_2 = \{u\}; \hat{s}_2 = x;$
18. $\quad$ **call** AuthCombTwo($\hat{U}[1,2], \hat{s}[1,2]$);
19. $\quad s^{(0)}_{N_{i_1}+3}$ is assigned the agreed key between user sets $\hat{U}_1, \hat{U}_2$
20. $\quad \hat{U}_3 = \hat{U}[1,2]; \hat{s}_3 = s^{(0)}_{N_{i_1}+3};\ \hat{U}_2 = U^{(0)}_{N_{i_1}+2}; \hat{s}_2 = s^{(0)}_{N_{i_1}+2};\ \hat{U}_1 = U^{(0)}_{N_{i_1}+1}; \hat{s}_1 = s^{(0)}_{N_{i_1}+1};$
21. $\quad$ **call** AuthCombThree $-$ B($\hat{U}[1,2,3], \hat{s}[1,2,3]$);
22. $\quad s^{(1)}_{i_1}$ is assigned the agreed key between user sets $\hat{U}_1, \hat{U}_2, \hat{U}_3$ and $\hat{U}^{(1)}_{i_1} = \hat{U}[1,2,3];$

23. **end if**
24. $l = 1$ to $k-1$ **do**
25. $j = i_l$;
26. **if** $i_l = 3i_{l+1} - 2$ *i.e.* $r_{l+1} = 2$ **then**
27. **call** AuthCombThree $-$ B$(U^{(l)}[j, j+1, j+2], s^{(l)}[j, j+1, j+2])$;
28. $s_{i_{l+1}}^{(l+1)}$ is assigned the agreed key among user sets $\hat{U}_j^{(l)}, \hat{U}_{j+1}^{(l)}, \hat{U}_{j+2}^{(l)}$;
$U_{i_{l+1}}^{(l+1)} = U^{(l)}[j, j+1, j+2]$;
29. **end if**
30. **if** $i_l = 3i_{l+1} - 1$ *i.e.* $r_{l+1} = 1$ **then**
31. **call** AuthCombThree $-$ B$(U^{(l)}[j-1, j, j+1], s^{(l)}[j-1, j, j+1])$;
32. $s_{i_{l+1}}^{(l+1)}$ is assigned the agreed key among user sets $\hat{U}_{j-1}^{(l)}, \hat{U}_j^{(l)}, \hat{U}_{j+1}^{(l)}$;
$U_{i_{l+1}}^{(l+1)} = U^{(l)}[j-1, j, j+1]$;
33. **end if**
34. **if** $i_l = 3i_{l+1}$ *i.e.* $r_{l+1} = 0$ **then**
35. **call** AuthCombThree $-$ B$(U^{(l)}[j-2, j-1, j], s^{(l)}[j-2, j-1, j])$;
36. $s_{i_{l+1}}^{(l+1)}$ is assigned the agreed key among user sets $\hat{U}_{j-2}^{(l)}, \hat{U}_{j-1}^{(l)}, \hat{U}_j^{(l)}$;
$U_{i_{l+1}}^{(l+1)} = U^{(l)}[j-2, j-1, j]$;
37. **end if**
38. **end do**

end UpdateKeyPath

Remark 4.2.5 *The join algorithm presented above is an authenticated algorithm. If we invoke* CombineTwo *in lines 6, 18 instead of* AuthCombTwo *and* CombineThree *in lines 12, 21, 27, 31, 35 instead of* AuthCombThree-A *and* AuthCombThree-B*, we get an unauthenticated version of the join algorithm. We require* $k = \lfloor \log_3 n \rfloor + 1$ *key updates in the execution of* Join*. After joining of the new member,* $R(n+1) = k+1$ *if* $n = 3^k$*, and* k *otherwise.*

The formal description of the procedures Extract, FindExtractNode and Leave are given below.

procedure Extract (T, n)
1. **if** $n = 1$ **then return** the index of the leaf node to GC;
2. **if** $n = 2$ **then return** the index of the left leaf node to GC;
3. Let T_L, T_M, T_R be respectively the left, middle and right subtree of T;
4. $p = \lfloor \frac{n}{3} \rfloor$; $r = n \bmod 3$;
5. **while** $(p \geq 1)$ **do**
6. **if** $r = 0$ *i.e.* $|T_L| = |T_M| = |T_R| = p$ **then**
7. **call** Extract (T_L, p);
8. **end if**
9. **if** $r = 1$ *i.e.* $|T_L| = |T_M| = p, |T_R| = p+1$ **then**
10. **call** Extract $(T_R, p+1)$;

11. **end if**
12. **if** $r = 2$ *i.e.* $|T_L| = p, |T_R| = |T_M| = p + 1$ **then**
13. **call** Extract $(T_M, p + 1)$;
14. **end if**
15. **end do**

end Extract

procedure FindExtractNode (T, n, j_0)

1. $k = R(n)$;
2. After removing node j_0, let $i(\leq k)$ be the highest level for which the subtree rooted at j_i lacks the structure of KeyAgreement. (In case $i < k$, this structure is retained in the other two siblings of j_i);
3. Let the left, middle and right subtrees of off-springs of the node j_i be T_L, T_M, T_R respectively;
4. **Case 1 :** Before removing j_0, let $|T_L| = |T_M| = |T_R| = p$; remove node j_0;
5. **if** (j_0 is leaf node of T_M or T_R) **then**
6. **call** Extract (T_L, p);
7. **end if**
8. **Case 2 :** Before removing j_0, let $|T_L| = |T_M| = p; |T_R| = p + 1$; remove node j_0;
9. **if** (j_0 is leaf node of T_L or T_M) **then**
10. **call** Extract $(T_R, p + 1)$;
11. **end if**
12. **Case 3 :** Before removing j_0, let $|T_L| = p$; $|T_M| = |T_R| = p + 1$; remove node j_0;
13. **if** (j_0 is leaf node of T_L or T_R) **then**
14. **call** Extract $(T_M, p + 1)$;
15. **end if**
16. Let l_0 be the extracted node to be inserted as the j_0-th leaf in the tree T;
17. **return** (l_0, j_i);

end FindExtractNode

procedure Leave (T, n, j_0)

1. Let $k = R(n)$.
2. **call** FindExtractNode(T, n, j_0);
3. Let (l_0, j_i) be its output.
4. **if** $l_0 = j_0$ **then**
5. let $\mathsf{path}_D = (j_k, \ldots, j_1)$ be path from root to parent of j_0. The siblings of user j_0 choose random new keys and form a new user set $U_{j_1}^{(1)}$. The subsequent user sets $U_{j_t}^{(t)}$ are modified replacing old $U_{j_1}^{(1)}$ by the new one and keys updated as in UpdateKeyPath.
6. **end if**
7. **else**
8. let $\mathsf{path}_D = (j_k, \ldots, j_i, j_{i-1}, \ldots, j_1)$ and $\mathsf{path}_E = (j_k, \ldots, j_i, l_{i-1}, \ldots, l_1)$ be the paths from

root to the parent of the leaving leaf node j_0 and extracted leaf node l_0 respectively, where j_t denotes the index of the node at level t whose subtree contains the leaving node j_0 and l_t denotes the index of the node at level t whose subtree contains the extracted node l_0.

9. User l_0 and its brothers choose random new keys; also brothers of user j_0 choose random new keys;
10. User l_0 and brothers of j_0 construct new user set $U_{j_1}^{(1)}$; brothers of user l_0 constructs new set $U_{l_1}^{(1)}$.
11. The subsequent user sets $U_{j_t}^{(t)}$, $U_{l_t}^{(t)}$, for $2 \leq t \leq k$, are modified accordingly replacing the old $U_{j_1}^{(1)}$ and $U_{l_1}^{(1)}$ values by their new values. Note that $|U_{l_1}^{(1)}| \neq 0$.
12. $t = 1$ to $i - 1$ **do in parallel**
13. • user sets of level $t - 1$ in $U_{j_t}^{(t)}$ agree upon a common key by invoking appropriate subroutine for authenticated key agreement depending upon the cardinality of $U_{j_t}^{(t)}$.
14. • user sets of level $t - 1$ in $U_{l_t}^{(t)}$ agree upon a common key by invoking appropriate subroutine for authenticated key agreement depending upon the cardinality of $U_{l_t}^{(t)}$.
15. **end do**;
16. $t = i$ to k **do**
17. user sets of level $t - 1$ in $U_{j_t}^{(t)}$ agree upon a common key by invoking appropriate subroutine for authenticated key agreement depending upon the cardinality of $U_{j_t}^{(t)}$.
18. **end do**;
19. **end else**

end Leave

Remark 4.2.6 *The invocation of* AuthCombTwo, AuthCombThree-A *and* AuthCombThree-B *in the above procedure make the protocol* Leave *authenticated. If instead, we use unauthenticated* CombineTwo *and* CombineThree, *an unauthenticated version of the leave algorithm is obtained. At most* $2\lfloor \log_3(n+1) \rfloor$ *key updates are required in the execution of* Leave. *After removal of a member,* $R(n-1) = k+1$ *if* $n = 3^k + 1$, *and* k *otherwise.*

4.3 Security Analysis

4.3.1 Security of the Unauthenticated Protocol

A secure key agreement protocol should withstand both passive and active attacks. In this subsection we define the **Decisional ternary tree group key agreement (DTGKA) problem** for our unauthenticated protocol and show that this problem is hard for the passive adversary by reducing the hardness of this problem to the hardness of DHBDH problem following the technique used in [10], [74], [113].

We use the following notations for convenience of analyzing the security.

–$U_j^{(i)}$: the j-th user set at level $i, 0 \leq i \leq R(n), 1 \leq j \leq n_i$,

–$s_j^{(i)}$: common secret key agreed upon by users in the user set $U_j^{(i)}$,

–$P_j^{(i)}$: i-th level j-th public key, *i.e.* $P_j^{(i)} = s_j^{(i)}P$.

Given a ternary tree T of height at most k with n leaf nodes ($n > 2$) and $X = (s_1, s_2, \ldots, s_n)$ for $s_i \in Z_q^*$, the public and secret values are collectively defined as follows :

– $vw(k, X, T) := \{P_j^{(i)}$ where j and i are defined according to $T\}$

– $K(k, X, T) := KEY = H(e(P, P)^{s_1^{(k-1)} s_2^{(k-1)} s_3^{(k-1)}})$

Here $vw(k, X, T)$ is exactly the view of the passive adversary in the ternary tree T where final key is $K(k, X, T)$. We call the key $K(k, X, T)$ a DHBDH key. Our goal is to show that this DHBDH key generated by the unauthenticated protocol can not be distinguished by a polynomial time algorithm from a random number if all the transmitted values during a protocol run are known.

Suppose $\mathcal{T}_k$ is the set of all ternary trees of height k having structure of KeyAgreement. A tree T of height k is chosen randomly from $\mathcal{T}_k$ and let $X \in_R (Z_q^*)^n$ ($n \leq 3^k$) be the labels of (short term private keys associated with) the leaf nodes of T. Then k is the number of rounds with n users. Let us define two random variables $A_k, \widehat{A}_k$ as follows :
–$A_k := (vw(k, X, T), y), y \in_R Z_q^*$
–$\widehat{A}_k := (vw(k, X, T), K(k, X, T))$

Let $\mathcal{S}_k = \{(T, X) : T \in_R \mathcal{T}_k$ and $X \in_R (Z_q^*)^n$, where n is the number of leaf level nodes in $T\}$. Let B_T be the number of edges in the ternary tree T. For $(T, X) \in \mathcal{S}_k$, define $\Gamma(T, X)$ to be the ordered tuple of all public information along the arcs of T. Clearly, $\Gamma(T, X) \subseteq (G_1)^{B_T}$. Then the random variable A_k takes values from the sample space $(G_1)^{B_T} \times Z_q^*$ according to the uniform probability distribution and $\widehat{A}_k$ takes values from the sample space $\Gamma(T, X) \times Z_q^* \subseteq (G_1)^{B_T} \times Z_q^*$ with the uniform probability distribution.

Definition 4.3.1 *Consider* (G_1, G_2, e) *. Let* $n > 2$ *be a positive integer,* $X = (s_1, s_2, \ldots, s_n)$ *for* $s_i \in Z_q^*$ *and* T *be a ternary tree of height* k *with* n *leaf nodes labeled by* X*, and* $A_k, \widehat{A}_k$ *are defined as above. A* **Decisional ternary tree group key agreement (DTGKA) algorithm** $\mathcal{F}$ *for* (G_1, G_2, e) *is a probabilistic polynomial time algorithm that outputs either* 0 *or* 1*, satisfying, for some fixed* $l > 0$ *and sufficiently large* m *:*

$$|Prob[\mathcal{F}(A_k) = 1] - Prob[\mathcal{F}(\widehat{A}_k) = 1]| > \frac{1}{m^l}.$$

We call $\mathcal{F}$ *a polynomial time distinguisher that distinguishes* A_k *and* $\widehat{A}_k$*.*

The **DTGKA problem** is to find a polynomial time distinguisher $\mathcal{F}$ for A_k and $\widehat{A}_k$ defined above.

Theorem 4.3.2 *If the DHBDH problem in (G_1, G_2, e) is hard, then A_k and $\widehat{A}_k$ are polynomially indistinguishable.*

Proof : First let us provide a plan of the proof. The proof is by induction on k.

Base Step : $k = 1$: Distinguishing A_1 and $\widehat{A}_1$ implies "solving" DHBDH problem in (G_1, G_2, e) . Thus it is not possible to distinguish A_1 and $\widehat{A}_1$ assuming DHBDH problem is hard in (G_1, G_2, e) .
Induction Hypothesis : Assume that for some $k \geq 2$, it is not possible to distinguish A_{k-1} and $\widehat{A}_{k-1}$.
Induction Step : We show that the ability to distinguish A_k and $\widehat{A}_k$ implies either (a) "solving" DHBDH problem in (G_1, G_2, e) or (b) ability to distinguish A_{k-1} and $\widehat{A}_{k-1}$. Since (a) is given to be hard and (b) is hard by induction hypothesis, it follows that it is not possible to distinguish between A_k and $\widehat{A}_k$.

Now we turn to a proof of the induction step. Let $T \in_R \mathcal{T}_k$ be a ternary tree of height k with n leaf nodes. Let $X \in_R (Z_q^*)^n$ be the labels of the leaf nodes of T. Let T_1, T_2, T_3 respectively be the left, middle and right subtree of height at most $k-1$ of the tree T. This implies that at least one of these subtrees has height exactly $k-1$. If $X_1 = (s_1, \ldots, s_l), X_2 = (s_{l+1}, \ldots, s_m)$ and $X_3 = (s_{m+1}, \ldots, s_n)$, where s_1 through s_l are associated with T_1, s_{l+1} through s_m with T_2 and s_{m+1} through s_n with T_3, then A_k and $\widehat{A}_k$ can be rewritten as :
$A_k := (vw(k, X, T), y)$ for random $y \in Z_q^*$
$= (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), P_1^{(k-1)}, P_2^{(k-1)}, P_3^{(k-1)}, y)$
$= (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), s_1^{(k-1)}P, s_2^{(k-1)}P, s_3^{(k-1)}P, y)$

$\widehat{A}_k := (vw(k, X, T), K(k, X, T))$
$= (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), P_1^{(k-1)}, P_2^{(k-1)}, P_3^{(k-1)}, KEY)$
$= (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), s_1^{(k-1)}P, s_2^{(k-1)}P, s_3^{(k-1)}P, KEY)$

Let us consider the following random variables:
$A_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), s_1^{(k-1)}P, s_2^{(k-1)}P, s_3^{(k-1)}P, y)$
$B_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), r_1P, s_2^{(k-1)}P, s_3^{(k-1)}P, y)$
$C_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), r_1P, r_2P, s_3^{(k-1)}P, y)$
$D_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), r_1P, r_2P, r_3P, y)$
$\widehat{D}_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), r_1P, r_2P, r_3P, K_1)$
$\widehat{C}_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), r_1P, r_2P, s_3^{(k-1)}P, K_2)$
$\widehat{B}_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), r_1P, s_2^{(k-1)}P, s_3^{(k-1)}P, K_3)$
$\widehat{A}_k := (vw(k-1, X_1, T_1), vw(k-1, X_2, T_2), vw(k-1, X_3, T_3), s_1^{(k-1)}P, s_2^{(k-1)}P, s_3^{(k-1)}P, KEY)$

where $r_1, r_2, r_3 \in_R Z_q^*$ and $K_1 = H(e(P,P)^{r_1 r_2 r_3})$, $K_2 = H(e(P,P)^{r_1 r_2 s_3^{(k-1)}})$ and $K_3 = H(e(P,P)^{r_1 s_2^{(k-1)} s_3^{(k-1)}})$.

Claim : If $A_k, \widehat{A}_k$ are distinguishable in polynomial time, then at least one of the followings can be distinguished : $(A_k, B_k), (B_k, C_k), (C_k, D_k), (D_k, \widehat{D}_k), (\widehat{D}_k, \widehat{C}_k), (\widehat{C}_k, \widehat{B}_k)$ or $(\widehat{B}_k, \widehat{A}_k)$.
Proof of the Claim : Let $a_1 = Prob[\mathcal{F}(A_k) = 1]$, $a_2 = Prob[\mathcal{F}(B_k) = 1]$, $a_3 = Prob[\mathcal{F}(C_k) = 1]$, $a_4 = Prob[\mathcal{F}(D_k) = 1]$, $a_5 = Prob[\mathcal{F}(\widehat{D}_k) = 1]$, $a_6 = Prob[\mathcal{F}(\widehat{C}_k) = 1]$, $a_7 = Prob[\mathcal{F}(\widehat{B}_k) = 1]$, $a_8 = Prob[\mathcal{F}(\widehat{A}_k) = 1]$. Since A_k and $\widehat{A}_k$ are distinguishable in polynomial time, $|a_1 - a_8| > \frac{1}{m^l}$ for sufficiently large m and for a fixed $l > 0$. Now we will show that at least one of the followings must hold : $|a_i - a_{i+1}| > \frac{1}{m^{l+1}}$ for $i = 1, \ldots, 7$. If not, let $|a_i - a_{i+1}| \leq \frac{1}{m^{l+1}}$ for all $i = 1, \ldots, 7$.
Then $|a_1 - a_8| \leq |a_1 - a_2| + \cdots + |a_7 - a_8| \leq \frac{7}{m^{l+1}} \leq \frac{1}{m^l}$, a contradiction, if $m \geq 7$.

We shall show that the ability to distinguish any one of (A_k, B_k), (B_k, C_k), (C_k, D_k), $(\widehat{D}_k, \widehat{C}_k)$, $(\widehat{C}_k, \widehat{B}_k)$ or $(\widehat{B}_k, \widehat{A}_k)$ reduces to solving DTGKA problem with height $k-1$ and ability of distinguishing $(D_k, \widehat{D}_k)$ reduces to solving the DHBDH problem in (G_1, G_2, e) . The proof of the two cases : (A_k, B_k) and $(D_k, \widehat{D}_k)$ are discussed here in details. A proof similar to the case (A_k, B_k) follows for others.

Case : Distinguish (A_k, B_k) : Suppose $\mathcal{F}_{AB_k}$ is a polynomial time distinguisher that can distinguish A_k and B_k in polynomial time. We will show that $\mathcal{F}_{AB_k}$ can be used to solve DTGKA problem with height $k-1$. We construct a polynomial time distinguisher $\mathcal{F}_{A\widehat{A}_{k-1}}$ that distinguishes A_{k-1} and $\widehat{A}_{k-1}$ in polynomial time as follows :
Let $V_{k-1}^* = (vw(k-1, X^*, T^*), r^*)$ where T^* is a ternary tree of height $k-1$ with $|X^*| = n$ having structure of KeyAgreement. The distinguisher $\mathcal{F}_{A\widehat{A}_{k-1}}$ first constructs two ternary trees $\widetilde{T}$ and $\overline{T}$ with leaf level secret key distribution $\widetilde{X}$ and $\overline{X}$ respectively in the following manner : if $n = 3^k$, then take $|\widetilde{X}| = |\overline{X}| = n$, else take either $|\widetilde{X}| = |\overline{X}| = n$ or $|\widetilde{X}| = n, |\overline{X}| = n+1$ or $|\widetilde{X}| = |\overline{X}| = n+1$. Then $\mathcal{F}_{A\widehat{A}_{k-1}}$ constructs a tree of height k with $|X^*| + |\widetilde{X}| + |\overline{X}|$ users and $T^*, \widetilde{T}, \overline{T}$ as the left, middle and right subtree respectively. The resulting tree is clearly a random member of $\mathcal{T}_k$. Next $\mathcal{F}_{A\widehat{A}_{k-1}}$ sets :
$V_k^* = (vw(k-1, X^*, T^*), vw(k-1, \widetilde{X}, \widetilde{T}), vw(k-1, \overline{X}, \overline{T}), r^*P, \widetilde{K}P, \overline{K}P, y)$, where $\widetilde{K} = K(k-1, \widetilde{X}, \widetilde{T}), \overline{K} = K(k-1, \overline{X}, \overline{T})$, $y \in_R Z_q^*$ and runs $\mathcal{F}_{AB_k}$ on input V_k^*. Now $Prob[\mathcal{F}_{AB_k}(A_k = V_k^*) = 1] = Prob[\mathcal{F}_{A\widehat{A}_{k-1}}(\widehat{A}_{k-1} = V_{k-1}^*) = 1]$ and $Prob[\mathcal{F}_{AB_k}(B_k = V_k^*) = 1] = Prob[\mathcal{F}_{A\widehat{A}_{k-1}}(A_{k-1} = V_{k-1}^*) = 1]$.
Consequently, $|Prob[\mathcal{F}_{A\widehat{A}_{k-1}}(\widehat{A}_{k-1} = V_{k-1}^*) = 1] - Prob[\mathcal{F}_{A\widehat{A}_{k-1}}(A_{k-1} = V_{k-1}^*) = 1]|$
$= |Prob[\mathcal{F}_{AB_k}(A_k = V_k^*) = 1] - Prob[\mathcal{F}_{AB_k}(B_k = V_k^*) = 1]|$. Hence if $\mathcal{F}_{AB_k}$ can distinguish between A_k and B_k, then $\mathcal{F}_{A\widehat{A}_{k-1}}$ can distinguish between A_{k-1} and $\widehat{A}_{k-1}$.

Case : Distinguish $(D_k, \widehat{D}_k)$: Suppose $\mathcal{F}_{D\widehat{D}_k}$ is a polynomial time distinguisher that can dis-

tinguish D_k and $\widehat{D}_k$ in polynomial time. We shall show that $\mathcal{F}_{D\widehat{D}_k}$ can be used to construct a polynomial time algorithm $\mathcal{A}$ that solves the DHBDH problem in (G_1, G_2, e) . Note that r_1P and r_2P are independent variables from $vw(k-1, X_1, T_1)$ and $vw(k-1, X_2, T_2)$.
Given $V^*_{k-1} = (P, r^*P, \widetilde{r}P, \overline{r}P, H(e(P,P)^r))$, the algorithm $\mathcal{A}$ has to decide whether $r = r^*\widetilde{r}\overline{r} \bmod q$ ($\mathcal{A}$ outputs *yes* in this case) or r is random ($\mathcal{A}$ outputs *no* in this case) where $r^*, \widetilde{r}, \overline{r} \in_R Z^*_q$. For this, $\mathcal{A}$ first generates a tree of height k having structure of KeyAgreement with three subtrees T^*, $\widetilde{T}$ and $\overline{T}$. The leaf level secret key distribution of these subtrees are X^*, $\widetilde{X}$ and $\overline{X}$ respectively. Then $\mathcal{A}$ sets :
$V^*_k = (vw(k-1, X^*, T^*), vw(k-1, \widetilde{X}, \widetilde{T}), vw(k-1, \overline{X}, \overline{T}), r^*P, \widetilde{r}P, \overline{r}P, H(e(P,P)^r))$ and runs $\mathcal{F}_{D\widehat{D}_k}$ on input V^*_k. Now $Prob[\mathcal{A}$ outputs *no* on input $V^*_{k-1}] = Prob[\mathcal{F}_{D\widehat{D}_k}(D_k = V^*_k) = 1]$
and $Prob[\mathcal{A}$ outputs *yes* on input $V^*_{k-1}] = Prob[\mathcal{F}_{D\widehat{D}_k}(\widehat{D}_k = V^*_k) = 1]$.
Consequently, $|Prob[\mathcal{A}$ outputs *no* on input $V^*_{k-1}] - Prob[\mathcal{A}$ outputs *yes* on input $V^*_{k-1}]|$
$= |Prob[\mathcal{F}_{D\widehat{D}_k}(D_k = V^*_k) = 1] - Prob[\mathcal{F}_{D\widehat{D}_k}(\widehat{D}_k = V^*_k) = 1]|$. Hence if $\mathcal{F}_{D\widehat{D}_k}$ can distinguish between D_k and $\widehat{D}_k$, then $\mathcal{A}$ solves DHBDH problem in (G_1, G_2, e) . ■

The above discussion yields the following theorem.

Theorem 4.3.3 *The group key agreement protocol* UP *described in Section 4.2.1 is secure against passive adversaries under the assumption that DHBDH problem is hard.*

Remark 4.3.4 *The unauthenticated protocol* UP *executes a single* Execute *oracle. Since it does not involve any long term public/private keys,* Corrupt *oracles may simply be ignored and thus the protocol achieves forward secrecy. We have proved in Theorem 4.3.3 that the protocol* UP *is secure against passive adversary under DHBDH assumption for a single* Execute *query. This proof can be extended for the case of multiple* Execute *queries by using standard hybrid argument techniques: If* $\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t, 1)$ *is the advantage of the protocol* UP *for a single query to the* Execute *oracle, then with* q_E *queries to the* Execute *oracle, the advantage of* UP *is*

$$\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t, q_E) \leq q_E \ \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t, 1).$$

4.3.2 Security of the Authenticated (Static) Protocol

The goal is to show that the modification described in Section 4.2.2, converts the protocol of Section 4.2.1 into an authenticated key agreement protocol. The idea behind the proof is the following. Assuming that DSig and MSig are secure, we can convert any adversary attacking the authenticated protocol into an adversary attacking the unauthenticated protocol. We follow the reduction proof technique used by Katz and Yung [70]. However, there are some technical differences between our proof and that of [70].

1. The Katz-Yung technique is a generic technique for converting *any* unauthenticated protocol into an authenticated protocol. On the other hand, we concentrate on one particular protocol. Hence we can avoid some of the complexities of the Katz-Yung proof.

2. In keeping with the tree structure, our protocol involves a multi-signature scheme whereas Katz-Yung requires only a signature scheme. Multi-signatures allow a group of users to sign a message, such that a verifier can verify that all users indeed signed the message. For our application, we require the signing protocol to be *non-interactive* as in [18]. In fact, any non-interactive multi-signature scheme can be used with our protocol though we note that currently the only such known scheme has been presented by Boldyreva [18] and is based on the Boneh-Lynn-Shacham [26](BLS) pairing based short signature scheme. Both the BLS short signature scheme and the Boldyreva multi-signature scheme have been proved to be secure in the random oracle model. Since we use both the signature schemes in our authenticated protocol, the complete security of our protocol is also in the random oracle model. However, we do not actually require the random oracle in our security proof. Thus if one can replace the BLS signature scheme and the Boldyreva multi-signature scheme with schemes which are secure in the standard model, then the security of our protocol is also obtained in the standard model.

3. Katz-Yung define session identity for a participant in a session to be the concatenation of all sent and received messages by that participant in the session. They assume that this concatenation value is same for all participants involved in the session. We made no such assumption in our protocol. Consequently, we define session identity in a different way than Katz-Yung. (see Section 2.6.3). *Note that session identity is session specific. For different sessions, these session identities are different and all users participating in a particular session will have the same session identity (*cf. *Remark 2.6.1). Also by our assumption, for any instance* Π_U^i*, there is exactly one* j *such that* $(U, i) \in \mathsf{sid}_j$ *where* sid_j *is the session identity for the* j*-th session and for any* $j_1 \neq j_2$*, we have* $\mathsf{sid}_{j_1} \cap \mathsf{sid}_{j_2} = \emptyset$*. We bind the session identity with the message transmitted during our protocol execution. Since session identities uniquely identifies a session and all users in a particular session hold the same session identity, such an inclusion of session identity in transmitted messages prevents replay attack. In replay attack, adversary uses messages transmitted in a previous session in current session to obtain some information.The use of previously transmitted message in the current session will not be valid since the session identities are different.* Session-identity has an important role in our security proof as we will see below.

4. Katz-Yung protocol uses random nonces whereas our protocol does not. Thus we do not need any extra round for communication of random nonces. The security implication of our protocol remains as tight as the Katz-Yung reduction proof. The advantage in breaking the security of our authenticated protocol is upper bounded by the followings: the advantage of breaking the security of our unauthenticated protocol, the success probability of forging the signature scheme DSig and the success probability of forging the multisignature scheme MSig. So if we assume that DSig and MSig are secure, then given an active adversary to attack our authenticated protocol with non-negligible probability, we can construct a passive adversary to attack our unauthenticated protocol with non-negligible probability. But we have shown in Section 4.3.1 that our unauthenticated protocol is secure against passive adversary under the DHBDH assumption. Hence our authenticated protocol is secure against active adversary

under the same assumption.

5. *Note that a* Corrupt *query outputs the long-term secret key (if any) of a user. In our unauthenticated protocol, there are no long term secret keys. Thus* Corrupt *queries are ignored (see Remark 4.3.4). Consequently, our unauthenticated protocol trivially achieves forward secrecy. We follow the proof provided by Katz and Yung for the security of their authenticated protocol. They proved that if an unauthenticated group key agreement protocol is secure achieving forward secrecy, then the authenticated protocol obtained by applying their compiler, is also a secure group key agreement achieving forward secrecy.* Since our unauthenticated protocol achieves forward security, by Katz-Yung theorem, our authenticated protocol also achieves forward secrecy.

Theorem 4.3.5 *The protocol* AP *described in Section 4.2.2 satisfies the following:*

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{AP}}(t, q_E, q_S) \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_S/2) + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t') + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{MSig}}(t')$$

where $t' \leq t + (|\mathcal{P}|q_E + q_S)t_{\mathsf{AP}}$*, where* t_{AP} *is the time required for execution of* AP *by any one of the users.*

Proof: Let $\mathcal{A}'$ be an adversary which attacks the authenticated protocol AP. Using this we construct an adversary $\mathcal{A}$ which attacks the unauthenticated protocol UP. We first have the following claim.

Claim : Let Forge be the event that a signature (either of DSig or of MSig) is forged by $\mathcal{A}'$. Then

$$\mathsf{Prob}[\mathsf{Forge}] \leq |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{MSig}}(t') + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t').$$

Proof of Claim: Let E_1 be the event that $\mathcal{A}'$ forges a multi-signature, i.e., it makes a query of the type $\mathsf{Send}(V, i, \mathsf{psid}^k_U|j|\mathsf{SP}|\sigma)$ such that user V uses algorithm $\mathcal{MV}$ to verify σ to be the multi-signature of $\mathsf{First}(\mathsf{psid}^k_U)$ on $\mathsf{psid}^k_U|j|\mathsf{SP}$ and this message-signature pair was not previously returned by $\mathcal{A}$.

Let E_2 be the event that $\mathcal{A}'$ makes a query of the type $\mathsf{Send}(V, i, Y)$ where Y has the form $Y = \Pi^{d_k}_{U_k}|1|\mathsf{SP}|\sigma_{U_k}$ with $\mathcal{V}(\mathsf{pk}_{U_k}, \Pi^{d_k}_{U_k}|1|\mathsf{SP}, \sigma_{U_k}) = 1$. Then clearly $\mathsf{Forge} = E_1 \vee E_2$.

First consider the event E_2. Using $\mathcal{A}'$, we construct an algorithm $\mathcal{F}$ that forges a signature for DSig as follows: Given a public key pk, algorithm $\mathcal{F}$ chooses a random $U \in \mathcal{P}$ and sets $pk_U = pk$. The other public keys and private keys for the system are generated honestly by $\mathcal{F}$. The forger $\mathcal{F}$ simulates all oracle queries of $\mathcal{A}'$ by executing protocol AP itself, obtaining the necessary signatures with respect to pk_U, as needed, from its signing oracle. Thus $\mathcal{F}$ provides a perfect simulation for $\mathcal{A}'$. If $\mathcal{A}'$ ever outputs a new valid message/signature pair with respect to $pk_U = pk$, then $\mathcal{F}$ outputs this pair as its forgery. The success probability of $\mathcal{F}$ is equal to $\frac{\mathsf{Prob}[E_2]}{|\mathcal{P}|}$ and hence $\mathsf{Prob}[E_2] \leq |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t')$.

Next consider the event E_1. Using $\mathcal{A}'$, we may construct an algorithm $\mathcal{F}$ that forges a multi-signature for the scheme MSig as follows: Given a public key pk_1, algorithm $\mathcal{F}$ chooses a random $U \in \mathcal{P}$, sets $pk_U = pk_1$ and honestly generates all other public/private keys for the system and

outputs them. The forger $\mathcal{F}$ simulates all oracle queries of $\mathcal{A}'$ in the natural way by executing protocol AP itself, obtaining the necessary signatures with respect to pk_U, as needed, from its signing oracle and thus providing a perfect simulation for $\mathcal{A}'$. Now, if $\mathcal{A}'$ ever outputs a valid multi-signature σ on a message m for a set of users $L = \{U_{i_1}, \ldots, U_{i_k}\}$, which was not obtained from the signing oracle, then $\mathcal{F}$ outputs the message m, σ and L as its forgery. The success probability of $\mathcal{F}$ is equal to $\frac{\mathsf{Prob}[E_1]}{|\mathcal{P}|}$ and hence $\mathsf{Prob}[E_1] \leq |\mathcal{P}|\ \mathsf{Succ}^{\mathsf{MSig}}(t')$.

Then $\mathsf{Prob}[\mathsf{Forge}] = \mathsf{Prob}[E_1 \vee E_2] \leq \mathsf{Prob}[E_1] + \mathsf{Prob}[E_2] \leq |\mathcal{P}|\ \mathsf{Succ}^{\mathsf{MSig}}(t') + |\mathcal{P}|\ \mathsf{Succ}^{\mathsf{DSig}}(t')$, yielding the result of the Claim. ■(of Claim)

Now we describe the construction of the passive adversary $\mathcal{A}$ attacking UP that uses adversary $\mathcal{A}'$ attacking AP. Adversary $\mathcal{A}$ uses a list tlist. It stores pairs of session IDs and transcripts in tlist.

Adversary $\mathcal{A}$ generates the verification/signing keys pk_U, sk_U for each user $U \in \mathcal{P}$ and gives the verification keys to $\mathcal{A}'$. If ever the event Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}$ outputs whatever bit is eventually output by $\mathcal{A}'$. Note that since the signing and verification keys are generated by $\mathcal{A}$, it can detect occurrence of the event Forge. $\mathcal{A}$ simulates the oracle queries of $\mathcal{A}'$ using its own queries to the Execute oracle. We provide details below.

Execute queries: Suppose $\mathcal{A}'$ makes a query $\mathsf{Execute}((U_{i_1}, d_1), \ldots, (U_{i_k}, d_k))$. This means that instances $\Pi^{d_1}_{U_{i_1}}, \ldots, \Pi^{d_k}_{U_{i_k}}$ are involved in this session. $\mathcal{A}$ defines $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ and sends the execute query to its Execute oracle. It receives as output a transcript T of an execution of UP. It appends (S, T) to tlist. Adversary $\mathcal{A}$ then expands the transcript T for the unauthenticated protocol into a transcript T' for the authenticated protocol according to the modification described in Section 4.2.2. It returns T' to $\mathcal{A}'$.

Send queries: The first send query that $\mathcal{A}'$ makes to an instance is to start a new session. We will denote such queries by Send_0 queries. To start a session between unused instances $\Pi^{d_1}_{U_{i_1}}, \ldots, \Pi^{d_k}_{U_{i_k}}$, the adversary has to make the following send queries.

$$\begin{gathered}\mathsf{Send}_0(U_{i_1}, d_1, \langle U_{i_2}, \ldots, U_{i_k}\rangle);\\ \mathsf{Send}_0(U_{i_2}, d_2, \langle U_{i_1}, U_{i_3} \ldots, U_{i_k}\rangle);\\ \vdots\\ \mathsf{Send}_0(U_{i_k}, d_k, \langle U_{i_1}, \ldots, U_{i_{k-1}}\rangle).\end{gathered}$$

Note that the above queries may be made in any order. When all the above queries have been made, $\mathcal{A}$ sets $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ and makes an Execute query to its own execute oracle. It receives a transcript T in return and stores (S, T) in the list tlist.

Assuming that signatures (both DSig and MSig) cannot be forged, any subsequent Send query (i.e., after a Send_0 query) to an instance Π^i_U is a properly structured message with a valid signature. For any such Send query, $\mathcal{A}$ verifies the query according to the algorithm of Section 4.2.2. If the verification fails, $\mathcal{A}$ sets $\mathsf{acc}^i_U = 0$ and $\mathsf{sk}^i_U = \mathsf{NULL}$ and aborts Π^i_U. Otherwise, $\mathcal{A}$ performs the action to be done by Π^i_U in the authenticated protocol. This is done in the following manner: $\mathcal{A}$ first finds the unique entry (S, T) in tlist such that $(U, i) \in S$. From T, it finds the message which corresponds to the message sent by $\mathcal{A}'$ to Π^i_U. From the transcript T, adversary $\mathcal{A}$ finds the next

public information to be output by Π_U^i. If this involves computation of a multi-signature and all the individual signatures have not yet been received by Π_U^i (using Send queries from $\mathcal{A}'$), then there is no output to this Send query. In all other cases, $\mathcal{A}$ returns the next public information to be output by Π_U^i to $\mathcal{A}'$.

Reveal/Test queries : Suppose $\mathcal{A}'$ makes the query $\mathsf{Reveal}(U,i)$ or $\mathsf{Test}(U,i)$ for an instance Π_U^i for which $\mathsf{acc}_U^i = 1$. At this point the transcript T' in which Π_U^i participates has already been defined. Now $\mathcal{A}$ finds the unique pair (S,T) in tlist such that $(U,i) \in S$. Assuming that the event Forge does not occur, T is the unique unauthenticated transcript which corresponds to the transcript T'. Then $\mathcal{A}$ makes the appropriate Reveal or Test query to one of the instances involved in T and returns the result to $\mathcal{A}'$.

As long as Forge does not occur, the above simulation for $\mathcal{A}'$ is perfect. Whenever Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. So $\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Forge}] = \frac{1}{2}$. Now

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} &:= 2\,|\mathsf{Prob}_{\mathcal{A},\mathsf{UP}}[\mathsf{Succ}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Forge}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Forge}]\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + (1/2)\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}] - \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Forge}] + (1/2)\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 1/2| \\
&\geq |2\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}] - 1| - |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 2\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Forge}]| \\
&\geq \mathsf{Adv}_{\mathcal{A}',\mathsf{AP}} - \mathsf{Prob}[\mathsf{Forge}]
\end{aligned}$$

The adversary $\mathcal{A}$ makes an Execute query for each Execute query of $\mathcal{A}'$. Also $\mathcal{A}$ makes an Execute query for each session started by $\mathcal{A}'$ using Send queries. Since a session involves at least two instances, such an Execute query is made after at least two Send queries of $\mathcal{A}'$. Hence the number of such Execute queries is at most $q_S/2$, where q_S is the number of Send queries made by $\mathcal{A}'$. Hence the total number of Execute queries made by $\mathcal{A}$ is at most $q_E + q_S/2$, where q_E is the number of Execute queries made by $\mathcal{A}'$. Also since $\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_S/2)$ by assumption, we obtain:

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{AP}} \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_S/2) + \mathsf{Prob}[\mathsf{Forge}].$$

This yields the statement of the theorem. ■

4.3.3 Security of the Dynamic Authenticated Protocol

We will show that our dynamic authenticated key agreement protocol DAP is secure in the model as described in Chapter 2. In fact, we can convert any active adversary attacking the protocol DAP into a passive adversary attacking the unauthenticated protocol UP assuming that both DSig and MSig are secure and DHBDH problem is hard. No Corrupt query appears since long term secret keys are not used. So our protocol trivially achieves forward secrecy. We prove our result in Theorem 4.3.6 that states the security of our dynamic authenticated key agreement protocol DAP.

Theorem 4.3.6 *The dynamic group key agreement protocol* DAP *described in Section 4.2.3 satisfies the following:*

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{DAP}}(t, q_E, q_J, q_L, q_S) \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + (q_J + q_L + q_S)/2) + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t') + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{MSig}}(t')$$

where $t' \leq t + (|\mathcal{P}|q_E + q_J + q_L + q_S)t_{\mathsf{DAP}}$*, where* t_{DAP} *is the time required for execution of* DAP *by any one of the users,* q_E, q_J, q_L *and* q_S *are respectively the maximum number of execute, join, leave and send queries that an adversary can make.*

Proof (Sketch) : Let $\mathcal{A}'$ be an adversary which attacks the dynamic authenticated protocol DAP. Using this we construct an adversary $\mathcal{A}$ which attacks the unauthenticated protocol UP. As in the proof of the previous Theorem 4.3.5, we have the following claim.
Claim : Let Forge be the event that a signature (either of DSig or of MSig) is forged by $\mathcal{A}'$. Then $\mathsf{Prob}[\mathsf{Forge}] \leq |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{MSig}}(t') + |\mathcal{P}|\ \mathsf{Succ}_{\mathsf{DSig}}(t')$.

Adversary $\mathcal{A}$ maintains a list Tlist to store pairs of session IDs and transcripts. It also uses two lists Jlist and Llist to be specified later. Adversary $\mathcal{A}$ generates the verification/signing keys pk_U, sk_U for each user $U \in \mathcal{P}$ and gives the verification keys to $\mathcal{A}'$. If ever the event Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}$ outputs whatever bit is eventually output by $\mathcal{A}'$. Note that since the signing and verification keys are generated by $\mathcal{A}$, it can detect occurrence of the event Forge. $\mathcal{A}$ simulates the oracle queries of $\mathcal{A}'$ using its own queries to the Execute oracle. We provide details below.
Execute and Send queries: These queries are simulated as in [53]. Apart from the usual send queries, there are two special type of send queries, Send_J and Send_L. If an unused instance Π^d_U wants to join the group $\Pi^{d_1}_{U_{i_1}}, \ldots, \Pi^{d_k}_{U_{i_k}}$, then $\mathcal{A}'$ will make $\mathsf{Send}_J(U, d, \langle U_{i_1}, \ldots, U_{i_k}\rangle)$ query. This query initiates $\mathsf{Join}(\{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}, (U, d))$ query . $\mathcal{A}$ first finds a unique entry of the form (S, T) in Tlist with $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$. If no such entry, $\mathcal{A}$ makes an execute query to its own execute oracle on S and gets a transcript T. $\mathcal{A}$ then stores $(S, U|d, T)$ in Jlist. Similarly, when $\mathsf{Send}_L(U, d, \langle U_{i_1}, \ldots, U_{i_k}\rangle)$ query is made, $\mathcal{A}$ stores $(S, U|d, T)$ in Llist.
Join queries : Suppose $\mathcal{A}'$ makes a query $\mathsf{Join}(\{(U_{i_1}, d_1), \ldots, , (U_{i_k}, d_k)\}, (U, d))$. $\mathcal{A}$ finds an entry of the form $(S, U|d, T)$ in Jlist where $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$. If no such entry, then the adversary $\mathcal{A}'$ is given no output. Otherwise, $\mathcal{A}$ modifies T as follows: $\mathcal{A}$ can find the path of joining of U in the key tree with leafs $U_{i_1}, \ldots, U_{i_k}$ and detect the positions in T where the new messages are to be injected or where the old messages are to be replaced by new messages. $\mathcal{A}$ does these modifications in T according to the unauthenticated version of the (see Remark 4.2.5) of algorithm Join described in Section 4.2.3 and gets a modified transcript T_M. It then patches appropriate basic signatures and multi-signatures with each message in T_M according to the modifications described in Section 4.2.2. Thus $\mathcal{A}$ expands transcript T_M into a transcript T' for DAP and returns T' to $\mathcal{A}'$.
Leave queries : These queries are simulated as Join queries with modified transcript T_M obtained from unauthenticated transcript T according to the algorithm Leave in Section 4.2.3.
Reveal/Test queries : Suppose $\mathcal{A}'$ makes the query $\mathsf{Reveal}(U, i)$ or $\mathsf{Test}(U, i)$ for an instance Π^i_U for which $\mathsf{acc}^i_U = 1$. At this point the transcript T' in which Π^i_U participates has already been

defined. Now $\mathcal{A}$ finds the unique pair (S,T) in Tlist such that $(U,i) \in S$. Assuming that the event Forge does not occur, T is the unique unauthenticated transcript which corresponds to the transcript T'. Then $\mathcal{A}$ makes the appropriate Reveal or Test query to one of the instances involved in T and returns the result to $\mathcal{A}'$.
As long as Forge does not occur, the above simulation for $\mathcal{A}'$ is perfect. Whenever Forge occurs, $\mathcal{A}$ aborts and outputs a random bit. So $\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Forge}] = \frac{1}{2}$. Using this, one can show

$$\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} \geq \mathsf{Adv}_{\mathcal{A}',\mathsf{DAP}} - \mathsf{Prob}[\mathsf{Forge}]$$

The adversary $\mathcal{A}$ makes an $\mathsf{Execute}$ query for each $\mathsf{Execute}$ query of $\mathcal{A}'$. $\mathcal{A}'$ makes q_J Join queries and q_L Leave queries. These queries are initialized respectively by Send_J and Send_L queries of $\mathcal{A}'$. Now each of Send_J and Send_L query of $\mathcal{A}'$ makes at most one $\mathsf{Execute}$ query of $\mathcal{A}$. Thus there are at most $q_J + q_L$ execute query made by $\mathcal{A}$ to respond all the Send_J and Send_L queries of $\mathcal{A}'$. Also $\mathcal{A}$ makes an $\mathsf{Execute}$ query for each session started by $\mathcal{A}'$ using Send queries. Since a session involves at least two instances, such an $\mathsf{Execute}$ query is made after at least two Send queries of $\mathcal{A}'$. Thus there are $(q_S - q_J - q_L)/2$ execute queries of $\mathcal{A}$ to respond all other Send queries of $\mathcal{A}'$, where q_S is the number of Send queries made by $\mathcal{A}'$. Hence the total number of $\mathsf{Execute}$ queries made by $\mathcal{A}$ is at most $q_E + q_J + q_L + (q_S - q_J - q_L)/2 = q_E + (q_J + q_L + q_S)/2$, where q_E is the number of $\mathsf{Execute}$ queries made by $\mathcal{A}'$. Also since $\mathsf{Adv}_{\mathcal{A},\mathsf{UP}}(t, q_E, q_J, q_L, q_S) \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_J/2 + q_L/2 + q_S/2)$ by assumption, we obtain:

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{DAP}} \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + (q_J + q_L + q_S)/2) + \mathsf{Prob}[\mathsf{Forge}].$$

This yields the statement of the theorem. ■

4.4 Efficiency

This involves the communication and computation efficiency. In each round, a user may have to transmit publicly an element of G_1 to some or all the other users. Also it has to perform some operations like scalar multiplications, pairing computations. The number of rounds, total group elements sent, total messages exchanged provides the communication overhead in the protocol whereas total pairing computation, total scalar multiplications used incurs the computation costs.

First consider the unauthenticated version. Let $R(n)$ denote the total number of rounds, $P(n)$ the total pairings computed, $B(n)$ the combined message size and $E(n)$ the total number of scalar multiplications. If a user sends publicly an element of G_1 to some or all remaining users, then this counts one to $B(n)$. Thus $B(n)$ is the total number of such group elements. Now one can prove the Lemma stated below by induction.

Lemma 4.4.1 *For $n > 2$, the following recursions and bounds hold for $R(n), B(n), E(n)$ and $P(n)$:*

1. *$R(3n) = 1 + R(n)$; $R(3n+1) = 1 + R(n+1)$; $R(3n+2) = 1 + R(n+1)$; with initial conditions $R(1) = 0, R(2) = 1$. Consequently, $R(n) = \lceil \log_3 n \rceil$ for all n.*

2. $B(3n) = 3 + 3B(n)$; $B(3n+1) = 3 + 2B(n) + B(n+1)$; $B(3n+2) = 3 + B(n) + 2B(n+1)$; *with initial conditions* $B(1) = 0, B(2) = 3$. *Consequently,* $B(n) < \frac{5}{2}(n-1)$ *for* $n > 2$.

3. $E(3n) = 3 + 3E(n)$; $E(3n+1) = 3 + 2E(n) + E(n+1)$; $E(3n+2) = 3 + E(n) + 2E(n+1)$; *with initial conditions* $E(1) = 0, E(2) = 3$. *Consequently,* $E(n) < \frac{5}{2}(n-1)$ *for* $n > 2$.

4. $P(3n) = 3n + 3P(n)$; $P(3n+1) = 3n + 1 + 2P(n) + P(n+1)$; $P(3n+2) = 3n + 2 + P(n) + 2P(n+1)$; *with initial conditions* $P(1) = 0, P(2) = 2$. *Consequently,* $P(n) \leq n\lceil \log_3 n \rceil$.

Table 1 compares the unauthenticated version of our protocol with protocol BD of Burmester and Desmedt [36, 37], protocol GDH-3 of Steiner *et al.* [113] and protocol TGDH of Kim *et al.* [74]. These protocols are the most efficient protocols that have proper security analysis against passive adversary. So we choose these protocols to analyse the efficiency of our unauthenticated protocol. Both the BD protocol and GDH-3 protocol are provably secure under DDH assumption. A description of these schemes are provided in Section 3.4.3, and Section 3.4.2 respectively. The protocol TGDH is the Tree-Based Group Diffie-Hellman key agreement that extends the basic 2-party DH key agreement to binary tree-based setting. This scheme is secure against passive adversary under the assumption that decision version of Hash Diffie-Hellman problem is hard. The security of our protocol relies on the hardness assumption on decision version of Hash Bilinear Diffie-Hellman problem. (In the following table, Inv stands for total number of modular inversions and Mul stands for total number of multiplications used).

	$R(n)$	$B(n)$	$E(n)$	$P(n)$	Inv
BD [36]	2	$2n$	$n(n+1)$	–	n
GDH-3 [113]	$n+1$	$3(n-1)$	$5n-6$	–	–
TGDH [74]	$\lceil \log_2 n \rceil$	$n\lceil \log_2 n \rceil$	$n\lceil \log_2 n \rceil$	–	–
Our Protocol	$\lceil \log_3 n \rceil$	$< \frac{5}{2}(n-1)$	$< \frac{5}{2}(n-1)$	$\leq n\lceil \log_3 n \rceil$	–

Table 1 : Protocol Comparison (unauthenticated versions).

For a group of n participants, our unauthenticated protocol [9] requires $R(n) = \lceil \log_3 n \rceil$ rounds and total messages sent is $< \frac{5}{2}(n-1)$. The computation cost of this protocol is as follows. Total number of scalar multiplications in G_1 is $< \frac{5}{2}(n-1)$, total pairings required is $nR(n)$ and total group exponentiations in G_2 is $nR(n)$.

Points to note for unauthenticated protocols :

1. The underlying group of GDH-3 and BD protocol is a multiplicative subgroup of Z_p^* of order q where p and q both are prime.

2. The communication complexity is measured by $R(n)$ and $B(n)$ and our protocol achieves the minimum for both among all known protocols, except BD protocol. The computation

complexity of our protocol consists of two parts – exponentiation and pairing. The number of exponentiation is less than all other protocols, but additionally $n\lceil \log_3 n \rceil$ pairings are required. Assuming that each pairing computation is approximately equal to three exponentiations [8], [59], the TGDH and GDH-3 algorithms are more efficient than ours. This is based on the current state of the art in the algorithm for computing pairings. Any improvement in pairing computation algorithms will improve the efficiency of our protocol with respect to both TGDH and GDH-3.

3. Moreover, all the above protocols give DH-key except BD protocol.

Next we describe the efficiency of our authenticated (static) protocol. Let Y be the total number of singleton user sets in level 1. We set $R(n) = \lceil \log_3 n \rceil$.

Note that in our authenticated protocol, the representative of a user set with more than one user creates a multi-signature after it collects the basic (BLS) signatures from the other users in that user set. This makes the representative to wait for accumulating the basic signatures. In the first round of the authenticated protocol, no multi-signature is required because each user set is a singleton set with a user itself being the representative and only a basic signature on the transmitted message is sent by the representative. Our authenticated protocol additionally requires the followings :

1. The number of rounds increases by $R(n) - 1$.
2. Total number of basic (BLS) signatures computed is $nR(n)$.
3. Total number of additional messages (basic signatures) communicated is
$$n[R(n) - 1] - \frac{3}{2}(3^{R(n)-1} - 1).$$
4. Total bilinear aggregate signatures computed, communicated and verified is $\frac{3}{2}(3^{R(n)-1}-1)-Y$.
5. Total number of basic signatures verified is n.

This protocol involves no basic signature verification above level 1, only verification of multi-signatures are required. If any of the basic signature used in multi-signature creation on a message is improper, then this can be detected during multi-signature verification. This feature reduces the cost of verification for the authenticated protocol. Moreover, only representatives are required to have more computation power than other users. They are allowed to create and communicate multi-signatures which saves the total amount of communications.

Finally we consider the dynamic scenario. When a group of users want to join/leave, the member inclusion/member exclusion are done one by one in our dynamic protocol to retain the ternary tree structure. See Remarks 4.2.5 and 4.2.6.

4.5 Conclusion

In this chapter, we study the problem of constructing a new pairing-based protocol. This chapter deals with our works in [9, 52, 53, 54]. We design a dynamic group key agreement in tree-based setting that uses a different arrangement of participants. We prove that the scheme is provably secure under the assumption that DHBDH problem is hard. The bilinear pairing based Joux [66] protocol and multi-signature by Boldyreva [18] are used. The protocol is proven to be secure in the security model formalized by Bresson *et al.* [32] by appropriately modifying the Katz-Yung [70] technique to tree-based setting. One may benefit in both communication and computation power by combining our protocol with an efficient constant round protocol. Designing such hybrid group key agreement protocols may be desirable for certain applications, in particular when large number of users are concerned.

Chapter 5

Constant Round Dynamic Group Key Agreement

This chapter covers our article [55] where we concentrate on designing constant round group key agreement protocol. We present a fully symmetric constant round authenticated group key agreement protocol in dynamic scenario. Our proposed scheme achieves forward secrecy and is provably secure under DDH assumption in the security model of Bresson *et al.* providing, we feel, better security guarantee than previously published results. The protocol is efficient in terms of both communication and computation power.

5.1 Introduction

The main contribution of this chapter is to obtain a provably secure constant round authenticated group key agreement protocol in dynamic scenario where a user can join or leave the group at his desire with updated key. This work is based on our article [55]. We propose a scheme that is proven to be secure against passive adversary assuming the intractability of decision Diffie-Hellman (DDH) problem. This unauthenticated protocol can be viewed as a variant of the unauthenticated protocol (BD) of Burmester and Desmedt [36] although the session key computation is done differently that makes our protocol computationally more efficient than BD protocol. We incorporate authentication in our protocol using digital signature with appropriate modifications in the Katz-Yung [70] technique. Our authenticated protocol is a simplification of the protocol of [70] in which we remove the first round. Finally, we extend this static authenticated protocol to dynamic setting by introducing algorithms for join and leave. We prove that the security of both the static and dynamic authenticated protocols rely on that of the unauthenticated protocol. The security analysis against active adversary is in the model as formalized in section 2.6.3 of Chapter 2. Our protocol is forward secure, fully symmetric and being of constant round, is more efficient as compared to the protocol of Bresson *et al.* [32] (whose round complexity is linear in the number of group members).

Our security result holds in the standard model and thus provides better security guarantees than previously published results in the random oracle model.

More recently, Kim *et al.* [76] proposed a very efficient constant round dynamic authenticated group key agreement protocol and provide a security analysis of their static authenticated protocol which is shown to be secure under Computational Diffie-Hellman (CDH) assumption using random hash oracle. They did not consider the security analysis of their dynamic authenticated protocol. Unlike [76], we have achieved the security of our dynamic scheme in the standard model under standard DDH assumption without using any random oracle. We separately analyze the security of our static unauthenticated protocol, static authenticated protocol and dynamic authenticated protocol and reduce the security of the static authenticated protocol and dynamic authenticated protocol to that of the unauthenticated protocol.

Our proposed scheme considers the users $U_1, U_2, \ldots, U_n$ participating in the protocol on a ring where U_{i-1}, U_{i+1} are respectively the left and right neighbors of U_i for $1 \leq i \leq n$ with $U_0 = U_n, U_{n+1} = U_1$. Only 2 rounds are required in our protocol which makes our protocol efficient from communication point of view. User $U_i, 1 \leq i \leq n$, sends a message in first round only to its neighbors U_{i-1}, U_{i+1} and a message in second round to the rest of the $n-1$ users. Each user sends one message in each round with bit length at most $2|q| + 2|s|$ where $|q|$ is the length of q, the order of the underlying group on which DDH problem is assumed to be hard and $|s|$ is the length of signature. Each group member computes at most 3 modular exponentiations (1 in round 1 and 2 in round 2), $2n-2$ modular multiplications ($n-1$ multiplications for recovery of all right keys and $n-1$ multiplications for session key computation), 1 division, 2 signature generation and $n+1$ signature verification.

We emphasize that our protocol is dynamic and computationally more efficient as compared to the protocol of Burmester and Desmedt [36]. In contrast to the authenticated BD protocol (introduced by Katz-Yung [70]) that requires 3 rounds, our protocol completes in only two rounds. Additionally, our protocol differs from the BD protocol in the way the session key is computed after the rounds are over. Each user computes $\frac{n^2}{2} + \frac{3n}{2} - 3$ modular multiplications in BD protocol. On a more positive note, each user in our protocol requires to compute at most $2n$ modular multiplications. This makes our protocol much more efficient as compared to BD protocol. Besides, our protocol has the ability to detect the presence of a corrupted group member, although we cannot detect who among the group members are behaving improperly. If an invalid message is sent by a corrupted member, then this can be detected by all legitimate members of the group and the protocol execution may be stopped instantly. This feature makes our protocol interesting when the adversarial model no longer assumes that the group members are honest.

5.2 Protocol

We start with a description of our unauthenticated protocol. Following it, we describe our authenticated protocol. Finally, we propose our dynamic group key agreement.

Suppose a set of n users $\mathcal{P} = \{U_1, \ldots, U_n\}$ wish to establish a common session key among

themselves. We consider the users $U_1, \ldots, U_n$ participating in the protocol are on a ring and U_{i-1}, U_{i+1} are respectively the left and right neighbors of U_i for $1 \leq i \leq n$, $U_0 = U_n$ and $U_{n+1} = U_1$. Quite often, we identify a user U_i with his instance $\Pi_{U_i}^{d_i}$ (for some integer d_i that is session specific) during a protocol execution. We consider a multiplicative group G of some large prime order q with g as a generator. We also consider a hash function $\mathcal{H} : \{0,1\}^* \rightarrow Z_q^*$.

5.2.1 Unauthenticated Key Agreement Protocol

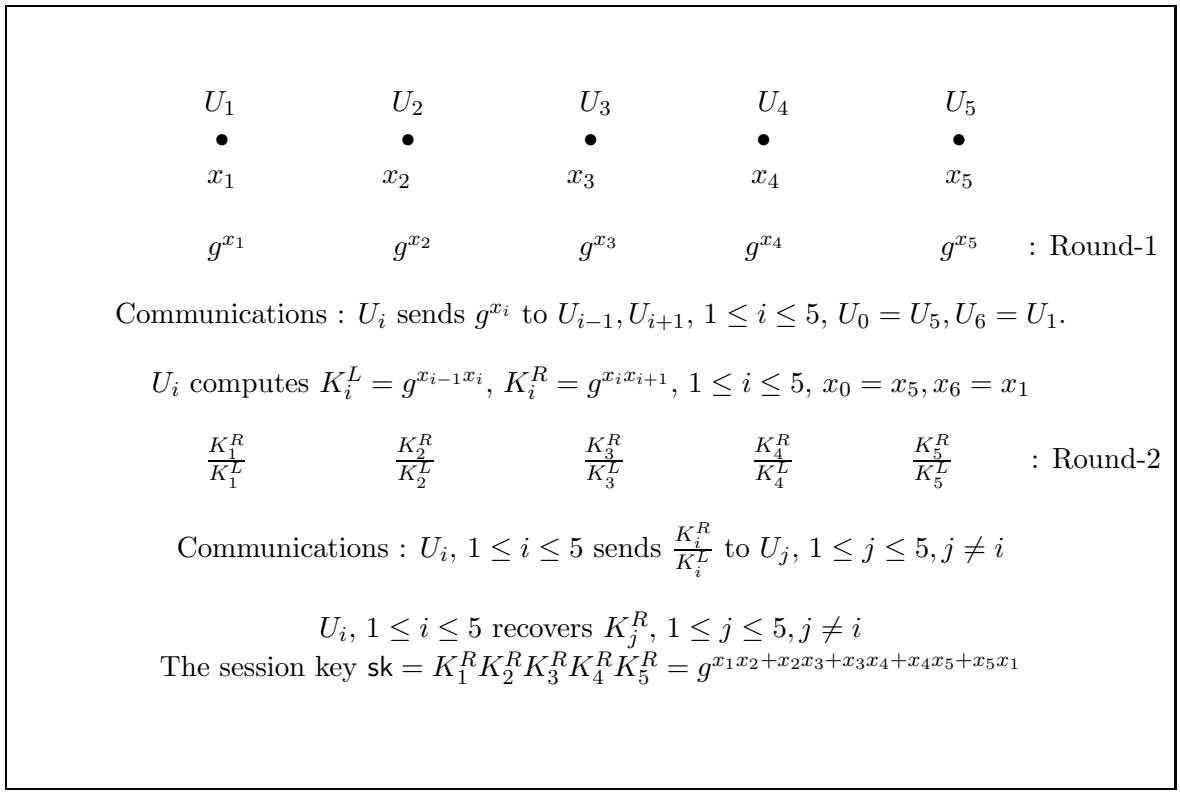

Figure 5.1: The unauthenticated group key agreement among $n = 5$ users.

First we informally describe our unauthenticated protocol KeyAgree that involves two rounds and a key computation phase. At the start of the session, each user $U_i = \Pi_{U_i}^{d_i}$ chooses randomly a private key $x_i \in Z_q^*$. In the first round, U_i computes $X_i = g^{x_i}$ and sends X_i to his neighbors U_{i-1}, U_{i+1}. After this communication round is over, U_i receives X_{i-1} from U_{i-1} and X_{i+1} from U_{i+1}. User U_i then computes his left key $K_i^L = X_{i-1}^{x_i}$, right key $K_i^R = X_{i+1}^{x_i}$, $Y_i = K_i^R / K_i^L$ and sends Y_i to the rest of the users in the second round. Finally in the key computation phase, U_i computes $\overline{K}_{i+1}^R, \overline{K}_{i+2}^R, \ldots, \overline{K}_{i+(n-1)}^R$ as follows making use of his own right key K_i^R:

$$\overline{K}_{i+1}^R = Y_{i+1} K_i^R, \overline{K}_{i+2}^R = Y_{i+2} \overline{K}_{i+1}^R, \ldots, \overline{K}_{i+(n-1)}^R = Y_{i+(n-1)} \overline{K}_{i+(n-2)}^R.$$

Then U_i verifies if $\overline{K}^R_{i+(n-1)}$ is same as that of his left key $K^L_i (= K^R_{i+(n-1)})$. If verification fails, then U_i aborts. Otherwise, U_i has correct right keys of all the users. U_i computes the session key $\mathsf{sk}^{d_i}_{U_i} = \overline{K}^R_1 \overline{K}^R_2 \ldots \overline{K}^R_n$ which is equal to $g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$. U_i also computes and stores $x = \mathcal{H}(\mathsf{sk}^{d_i}_{U_i})$ for a join operation and stores his left key and right key K^L_i, K^R_i respectively for a leave operation as we will see in the subsequent subsections. We refer to x as the seed which is common to all users involved in the session. Figure 5.1 illustrates the protocol with $n = 5$ users.

Observe that each user computes 3 exponentiations (1 in round 1 and 2 in round 2) and at most $2n-2$ multiplications ($n-1$ multiplications for recovery of all right keys and $n-1$ multiplications for session key computation). The formal description of the protocol is given below.

procedure KeyAgree$(U[1,\ldots,n], x[1,\ldots,n])$
- **(Round 1):**

1. **for** $i = 1$ to n **do in parallel**
2. $\quad U_i (= \Pi^{d_i}_{U_i})$ computes $X_i = g^{x_i}$ and sends X_i to U_{i-1} and U_{i+1} ;
3. **end for**
4. Note that $X_0 = X_n$ and $X_{n+1} = X_1$.

- **(Round 2):**

5. **for** $i = 1$ to n **do in parallel**
6. $\quad U_i$ computes the left key $K^L_i = X^{x_i}_{i-1}$, the right key $K^R_i = X^{x_i}_{i+1}$ and $Y_i = K^R_i / K^L_i$;
7. $\quad U_i$ sends Y_i to the rest of the users;
8. **end for**
9. Note that $K^R_i = K^L_{i+1}$ for $1 \le i \le n-1$, $K^R_n = K^L_1$ and $K^R_{i+(n-1)} = K^L_i$.

- **(Key Computation)**:

10. **for** $i = 1$ to n **do in parallel**
11. U_i computes $\overline{K}^R_{i+1} = Y_{i+1} K^R_i$;
12. $\quad$ **for** $j = 2$ to $n-1$ **do**
13. $\quad\quad U_i$ computes $\overline{K}^R_{i+j} = Y_{i+j} \overline{K}^R_{i+(j-1)}$;
14. $\quad$ **end for**
15. $\quad U_i$ verifies if $K^R_{i+(n-1)} = \overline{K}^R_{i+(n-1)}$ (*i.e.* if $K^L_i = \overline{K}^R_{i+(n-1)}$);
16. $\quad$ **if** verification fails, **then** U_i sets $\mathsf{acc}^{d_i}_{U_i} = 0$, $\mathsf{sk}^{d_i}_{U_i} = \mathsf{NULL}$ and aborts the protocol;
17. $\quad$ **else** U_i computes the session key $\mathsf{sk}^{d_i}_{U_i} = \overline{K}^R_1 \overline{K}^R_2 \ldots \overline{K}^R_n$, the seed $x = \mathcal{H}(\mathsf{sk}^{d_i}_{U_i})$ and stores K^L_i, K^R_i;
18. $\quad$ **end if**
19. **end for**

end KeyAgree

5.2.2 Authenticated Key Agreement Protocol

We now describe our authenticated protocol. The main underlying idea behind the protocol is to authenticate each message via a signature during each protocol execution. We authenticate the

unauthenticated protocol of Section 5.2.1 by incorporating a standard digital signature scheme $\mathsf{DSig} = (\mathcal{K}, \mathcal{S}, \mathcal{V})$ where $\mathcal{K}$ is the key generation algorithm, $\mathcal{S}$ is the signature generation algorithm and $\mathcal{V}$ is the signature verification algorithm. As part of this signature scheme, $\mathcal{K}$ generates a signing and a verification key sk_i (or sk_{U_i}) and pk_i (or pk_{U_i}) respectively for each user U_i. Session identity is an important issue of our authentication mechanism which uniquely identifies the session and is same for all instances participating in the session.

Suppose instances $\Pi_{U_{i_1}}^{d_1}, \ldots, \Pi_{U_{i_k}}^{d_k}$ wish to agree upon a common key in a session. Then according to our definition in section 2.6.3 of Chapter 2, $\mathsf{sid}_{U_{i_j}}^{d_j} = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$. Note that the instance numbers can be easily generated using counter. We make the assumption that in each session at most one instance of each user participates and an instance of a particular user participates in exactly one session. As mentioned earlier, this is a reasonable assumption to avoid collisions in the session identities (See Remark 2.6.2).

At the start of the session, $\Pi_{U_{i_j}}^{d_j}$ need not to know the entire set $\mathsf{sid}_{U_{i_j}}^{d_j}$. This set is built up as the protocol proceeds. We use a variable partial session-identity psid_U^d for instance Π_U^d involved in a session to keep the partial information about his session identity. Initially, $\mathsf{psid}_{U_{i_j}}^{d_j}$ is set to be $\{(U_{i_j}, d_j)\}$ by $\Pi_{U_{i_j}}^{d_j}$ and finally after completion of the session, $\mathsf{psid}_{U_{i_j}}^{d_j}$ grows into full session identity $\mathsf{sid}_{U_{i_j}}^{d_j}$. We assume that any instance $\Pi_{U_{i_j}}^{d_j}$ knows his partner identity $\mathsf{pid}_{U_{i_j}}^{d_j}$ *i.e.* the set of users with which it is partnered in the particular session. We describe below the algorithm AuthKeyAgree that is obtained by modifying algorithm KeyAgree by introducing signatures in the communication.

procedure AuthKeyAgree$(U[1, \ldots, n], x[1, \ldots, n])$

- **(Round 1):**

1. **for** $i = 1$ to n **do in parallel**
2. $\quad U_i (= \Pi_{U_i}^{d_i})$ sets its partial session-identity $\mathsf{psid}_{U_i}^{d_i} = \{(U_i, d_i)\}$;
3. $\quad U_i$ chooses randomly $x_i \in Z_q^*$ and computes $X_i = g^{x_i}$ and $\sigma_i = \mathcal{S}(sk_{U_i}, M_i)$ where $M_i = U_i|1|X_i$;
4. $\quad U_i$ sends $M_i|\sigma_i$ to U_{i-1} and U_{i+1};
5. **end for**
6. Note that $M_0|\sigma_0 = M_n|\sigma_n$ and $M_{n+1}|\sigma_{n+1} = M_1|\sigma_1$.

- **(Round 2):**

7. **for** $i = 1$ to n **do in parallel**
8. $\quad U_i$, on receiving $M_{i-1}|\sigma_{i-1}$ from U_{i-1} and $M_{i+1}|\sigma_{i+1}$ from U_{i+1}, verifies σ_{i-1} on M_{i-1} and σ_{i+1} on M_{i+1} using the verification algorithm $\mathcal{V}$ and the respective verification keys $pk_{U_{i-1}}$, $pk_{U_{i+1}}$;
9. $\quad$ **if** verification fails, **then** U_i sets $\mathsf{acc}_{U_i}^{d_i} = 0$, $\mathsf{sk}_{U_i}^{d_i} = \mathsf{NULL}$ and aborts;
10. $\quad$ **else** U_i computes the left key $K_i^L = X_{i-1}^{x_i}$, the right key $K_i^R = X_{i+1}^{x_i}$, $Y_i = K_i^R / K_i^L$ and signature $\overline{\sigma}_i = \mathcal{S}(sk_{U_i}, \overline{M}_i)$ where $\overline{M}_i = U_i|2|Y_i|d_i$;
11. $\quad U_i$ sends $\overline{M}_i|\overline{\sigma}_i$ to the rest of the users;
12. $\quad$ **end if**

13.**end for**
14.Note that $K_i^R = K_{i+1}^L$ for $1 \leq i \leq n-1$, $K_n^R = K_1^L$ and $K_{i+(n-1)}^R = K_i^L$.
• **(Key Computation)**:
15.**for** $i = 1$ to n **do in parallel**
16. **for** $j = 1$ to n, $j \neq i$ **do**
17. U_i, on receiving $\overline{M}_j|\overline{\sigma}_j$ from U_j verifies $\overline{\sigma}_j$ on $\overline{M}_j$ using the verification algorithm $\mathcal{V}$ and the verification key pk_{U_j};
18. **if** verification fails, **then** U_i sets $\mathsf{acc}_{U_i}^{d_i} = 0$, $\mathsf{sk}_{U_i}^{d_i} = \mathsf{NULL}$ and aborts;
19. **else** U_i extracts d_j from $\overline{M}_j$ and sets $\mathsf{psid}_{U_i}^{d_i} = \mathsf{psid}_{U_i}^{d_i} \cup \{(U_j, d_j)\}$;
20. **end for**
21. U_i computes $\overline{K}_{i+1}^R = Y_{i+1}K_i^R$;
22. $j = 2$ to $n-1$ **do**
23. U_i computes $\overline{K}_{i+j}^R = Y_{i+j}\overline{K}_{i+(j-1)}^R$;
24. **end for**
25. U_i verifies if $K_{i+(n-1)}^R = \overline{K}_{i+(n-1)}^R$ (*i.e.* if $K_i^L = \overline{K}_{i+(n-1)}^R$);
26. **if** verification fails, **then** U_i sets $\mathsf{acc}_{U_i}^{d_i} = 0$, $\mathsf{sk}_{U_i}^{d_i} = \mathsf{NULL}$ and aborts;
27. **else** U_i computes the session key $\mathsf{sk}_{U_i}^{d_i} = \overline{K}_1^R\overline{K}_2^R \ldots \overline{K}_n^R$, the seed $x = \mathcal{H}(\mathsf{sk}_{U_i}^{d_i})$ and stores K_i^L, K_i^R;
28. **end if**
29. **end if**
30.**end for**
end AuthKeyAgree

5.2.3 Dynamic Key Agreement Protocol

Our dynamic key agreement consists of the key agreement protocol together with two additional algorithms, Join and Leave. A set of users can join in a group by the procedure Join whereas the procedure Leave enables a set of users to leave a group.

Join

The key agreement in Join *algorithm is done in such a way that the new users are unable to know the previous session keys. When a set of new users $U_{n+1}, \ldots, U_{n+m}$ with respective secret keys $x_{n+1}, \ldots, x_{n+m}$ want to join a group of users $U_1, \ldots U_n$ with respective secret keys $x_1, \ldots, x_n$, we invoke* KeyAgree *or* AuthKeyAgree *(in case of authenticated version) among the users $U_1, U_2, U_n, U_{n+1}, \ldots, U_{n+m}$ with secret keys $x_1, x, x_n, x_{n+1}, \ldots, x_{n+m}$ considering a ring of $m+3$ users. Here x is the common seed value that users $U_1, U_2, \ldots, U_n$ agree upon in the previous session before users $U_{n+1}, \ldots, U_{n+m}$ have joined. In addition, from the transmitted messages, users $U_3, \ldots, U_{n-1}$ are able to compute the current session key as they know the common seed value x, although they do not participate in the communication during the protocol execution, however they*

receive all the messages they due to receive.

More formally, suppose $U[1,\ldots,n]$ be a set of users with respective secret keys $x[1,\ldots,n]$ and an execution of AuthKeyAgree among the instances $\Pi_{U_1}^{t_1},\ldots,\Pi_{U_n}^{t_n}$ has already been done. So all these instances $\Pi_{U_i}^{t_i}, 1 \leq i \leq n$, have a common session key and also a common seed $x \in Z_q^*$ resulting from this execution of AuthKeyAgree. Let the set of users $U[n+1,\ldots,n+m]$ with secret keys $x[n+1,\ldots,n+m]$ want to join the group $U[1,\ldots,n]$. The new instances involved in the procedure Join are $\Pi_{U_1}^{d_1},\ldots,\Pi_{U_{n+m}}^{d_{n+m}}$.

We consider a ring of $l = m+3$ users $V_1 = U_1$, $V_2 = U_2$, $V_3 = U_n$, $V_i = U_{n+i-3}$ for $4 \leq i \leq l$ with V_2 now using the seed x as its private key. We set $y_1 = x_1, y_2 = x, y_3 = x_n$, $y_i = x_{n+i-3}$ and $\hat{d}_1 = d_1, \hat{d}_2 = d_2, \hat{d}_3 = d_n$, $\hat{d}_i = d_{n+i-3}$. The left and right neighbors of V_i are respectively V_{i-1} and V_{i+1} for $1 \leq i \leq l$ with $V_0 = V_l$ and $V_{l+1} = V_1$. We take V_{l+i} to be V_i and V_2 is the representative of the set of users $U[2,\ldots,n-1]$. We invoke KeyAgree (for unauthenticated version of join algorithm) or AuthKeyAgree (for authenticated version of join algorithm) for l users $V[1,\ldots,l]$ with respective keys $y[1,\ldots,l]$. For simplicity, we describe the unauthenticated version of the procedure Join and mention the additional modifications required for its authenticated version.

Let for $1 \leq i \leq l$, $\hat{X}_i = g^{y_i}$; $\hat{X}_0 = \hat{X}_l, \hat{X}_{l+1} = \hat{X}_1$; $\hat{K}_i^L = \hat{X}_{i-1}^{y_i}; \hat{K}_i^R = \hat{X}_{i+1}^{y_i}; \hat{Y}_i = \hat{K}_i^R / \hat{K}_i^L$. In round 1, V_i sends $\hat{X}_i$ to both V_{i-1} and V_{i+1}. Additionally, V_1 sends $\hat{X}_1$ and V_3 sends $\hat{X}_3$ to all users $U[3,\ldots,n-1]$ in this round. In the second round, V_i computes its left key $\hat{K}_i^L$, right key $\hat{K}_i^R$ and sends $\hat{Y}_i$ to the rest of the users in $V[1,\ldots,l]$. Additionally, V_i sends $\hat{Y}_i$ to all users in $U[3,\ldots,n-1]$. If the protocol does not abort, V_i computes the session key $\mathsf{sk}_{V_i}^{\hat{d}_i}$ in the key computation phase which is the product of l right keys corresponding to l users $V[1,\ldots,l]$. V_i also computes the seed $\mathcal{H}(\mathsf{sk}_{V_i}^{\hat{d}_i})$ and stores $\hat{K}_i^L, \hat{K}_i^R$ that can be used for subsequent dynamic operations. Although active participation of the users $U[3,\ldots,n-1]$ are not required during the protocol execution, these users should be able to compute the common session key, the seed, the left key and the right key. Fortunately, these users have x, $\hat{X}_1 = g^{y_1}$ and $\hat{X}_3 = g^{y_3}$. So each can compute and store U_2's left key $\hat{K}_2^L = g^{y_1 x}$, right key $\hat{K}_2^R = g^{y_3 x}$ and proceeding in the same way as V_2 does, recover right keys of l users $V[1,\ldots,l]$, computes the session key and the common seed. The joining algorithm Join is formally described below.

procedure Join$(U[1,\ldots,n+m], x[1,\ldots,n+m])$

1. Set $l = m+3$; $V_1 = U_1, V_2 = U_2, V_3 = U_n; \hat{d}_1 = d_1, \hat{d}_2 = d_2, \hat{d}_3 = d_n; y_1 = x_1, y_2 = x, y_3 = x_n$; and for $4 \leq i \leq l$, $V_i = U_{n+i-3}; \hat{d}_i = d_{n+i-3}; y_i = x_{n+i-3}$;
2. We consider a ring of l users $V[1,\ldots,l]$ with respective instance numbers $\hat{d}[1,\ldots,l]$ and secret keys $y[1,\ldots,l]$;
3. **call** KeyAgree$(V[1,\ldots,l], y[1,\ldots,l])$;
4. Let for $1 \leq i \leq l$, $\hat{X}_i = g^{y_i}$; $\hat{X}_0 = \hat{X}_l, \hat{X}_{l+1} = \hat{X}_1$; $\hat{K}_i^L = \hat{X}_{i-1}^{y_i}; \hat{K}_i^R = \hat{X}_{i+1}^{y_i}; \hat{Y}_i = \hat{K}_i^R / \hat{K}_i^L$;
5. V_1 and V_3, in round 1, additionally send $\hat{X}_1$ and $\hat{X}_3$ respectively to all users in $U[3,\ldots,n-1]$;
6. V_i, in round 2, additionally sends $\hat{Y}_i$ to all users in $U[3,\ldots,n-1]$;
7. **for** $i = 3$ to $n-1$ **do**
8. $\quad U_i$ computes $\hat{K}_3^R = \hat{Y}_3 K_2^R$;

9. $j = 2$ to $l - 1$ **do**
10. U_i computes $\hat{K}^R_{2+j} = \hat{Y}_{2+j}\hat{K}^R_{2+(j-1)}$;
11. **end do**
12. U_i computes $\mathsf{sk}^{d_i}_{U_i} = \hat{K}^R_1\hat{K}^R_2 \ldots \hat{K}^R_l$;
13. **end for**

end Join

If we invoke procedure AuthKeyAgree instead of KeyAgree in line 3 of the above algorithm, then messages transmitted during the protocol execution are properly structured with signatures appended to them generated and verified according to the algorithm AuthKeyAgree. At the end of the session, if the protocol terminates normally without abort, then each user V_i, $1 \leq i \leq l$, additionally has a common session identity $\mathsf{sid}^{\hat{d}_i}_{V_i} = \{(V_1, \hat{d}_1), \ldots, (V_l, \hat{d}_l)\}$ apart from the common session key, the seed, the left and the right keys. Users $U[3, \ldots, n-1]$ are also able to compute this session identity from the messages received by them during the protocol execution.

Leave

The key agreement in Leave *algorithm is done in such a way that the set of revoked users in a session are not able to know the current session key. We drop the leaving users from the ring. The left and right neighbors of each leaving user choose new secret keys. Then the key agreement is done as usual in the resulting ring of users with the respective secret keys by invoking* KeyAgree *or* AuthKeyAgree *(in case of authenticated version of the protocol).*

Suppose $U[1, \ldots, n]$ is a set of users with respective secret keys $x[1, \ldots, n]$ and an execution of AuthKeyAgree among the instances $\Pi^{t_1}_{U_1}, \ldots, \Pi^{t_n}_{U_n}$ has already been done. Let K^L_i, K^R_i, $1 \leq i \leq n$ be the left and right keys respectively of U_i computed and stored in this session. Suppose users $U_{l_1}, \ldots, U_{l_m}$ want to leave the group $U[1, \ldots, n]$. Then the new user set is $U[1, \ldots, l_1 - L] \cup U[l_1 + R, \ldots, l_2 - L] \cup \ldots \cup U[l_m + R, \ldots, n]$ where $U_{l_i - L}$ and $U_{l_i + R}$ are respectively the left and right neighbors of the leaving user U_{l_i}, $1 \leq i \leq m$. Then for any leaving user U_l, $l - L = l - i$ if the consecutive users $U_l, U_{l-1}, \ldots, U_{l-(i-1)}$ are all leaving and U_{l-i} is not leaving the group. Similarly, $l + R = l + i$ if consecutive users $U_l, U_{l+1}, \ldots, U_{l+(i-1)}$ are all leaving and U_{l+i} is not leaving the group. We reindex these $n - m$ remaining users and denote the new user set by $V[1, \ldots, n-m]$. We also reindex the left and right keys and denote by two arrays $\hat{K}^L[1, \ldots, n-m]$ and $\hat{K}^R[1, \ldots, n-m]$ respectively the left and right keys of users $V[1, \ldots, n-m]$. Let the new instances involved in the procedure Leave are $\Pi^{d_1}_{V_1}, \ldots, \Pi^{d_{n-m}}_{V_{n-m}}$.

We consider a ring of $n - m$ users $V[1, \ldots, n-m]$. For a leaving user U_{l_i}, his left neighbor $U_{l_i - L}$ and right neighbor $U_{l_i + R}$ respectively choose new secret keys $x_{j_1}, x_{j_2} \in Z^*_q$ where $j_1 = l_i - L$ and $j_2 = l_i + R$, computes $X_{j_1} = g^{x_{j_1}}, X_{j_2} = g^{x_{j_2}}$. Note that in the ring, the left and right neighbors of U_{j_1} are respectively $U_{j_1 - 1}$ and U_{j_2} and that of U_{j_2} are respectively U_{j_1} and U_{j_2+1}. U_{j_1} sends X_{j_1} (properly structured with corresponding signature as in AuthKeyAgree) to his neighbors U_{j_1-1}, U_{j_2} and U_{j_2} sends X_{j_2} (properly structured) to his neighbors U_{j_1}, U_{j_2+1}. This is the first round. In the second round, each user V_i, after proper verification of the received messages, computes

$Y_i = \hat{K}_i^R / \hat{K}_i^L$ and sends Y_i (properly structured associating signature) to the rest of the users in $V[1, \ldots, n-m]$. The key computation phase is exactly the same as in the procedure AuthKeyAgree among $n-m$ users $V_1, \ldots . V_{n-m}$. The algorithm Leave is formally described below.

procedure Leave$(U[1, \ldots, n], x[1, \ldots, n], \{U_{l_1}, \ldots, U_{l_m}\})$

- **(Round 1):**

Let K_i^L, K_i^R be respectively the left and right keys of user U_i, $1 \leq i \leq n$, computed and stored in a previous session among instances $\Pi_{U_1}^{t_1}, \ldots, \Pi_{U_n}^{t_n}$.

1. **for** $i = 1$ to m **do in parallel**
2. Let $j_1 = l_i - L; j_2 = l_i + R$;
3. U_{j_1}, U_{j_2} respectively choose randomly new secret keys $x_{j_1}, x_{j_2} \in Z_q^*$ and computes $X_{j_1} = g^{x_{j_1}}$, $X_{j_2} = g^{x_{j_2}}$ and $\sigma_{j_1} = \mathcal{S}(sk_{U_{j_1}}, M_{j_1})$, $\sigma_{j_2} = \mathcal{S}(sk_{U_{j_2}}, M_{j_2})$ where $M_{j_1} = U_{j_1}|1|X_{j_1}$, $M_{j_2} = U_{j_2}|1|X_{j_2}$;
4. U_{j_1} sends $M_{j_1}|\sigma_{j_1}$ to U_{j_1-1} and U_{j_2};
5. U_{j_2} sends $M_{j_2}|\sigma_{j_2}$ to U_{j_1} and U_{j_2+1} $(U_{n+1} = U_1)$;
6. **end for**

- **(Round 2):**

7. **for** $i = 1$ to m **do in parallel**
8. Let $j_1 = l_i - L, j_2 = l_i + R$;
9. We set $W = \{j_1 - 1, j_1, j_2, j_2 + 1\}$;
10. U_{j_1-1}, U_{j_2} on receiving $M_{j_1}|\sigma_{j_1}$ from U_{j_1} verifies σ_{j_1} on M_{j_1} using the verification key $pk_{U_{j_1}}$;
11. U_{j_1}, U_{j_2+1} on receiving $M_{j_2}|\sigma_{j_2}$ from U_{j_2} verifies σ_{j_2} on M_{j_2} using the verification key $pk_{U_{j_2}}$;
12. **if** these verifications fail, **then** U_w, $w \in W$, sets $\mathsf{acc}_{U_w}^{d_w} = 0$, $\mathsf{sk}_{U_w}^{d_w} = \mathsf{NULL}$ and aborts;
13. **else**
14. U_{j_1} modifies its left key $K_{j_1}^L = X_{j_1-1}^{x_{j_1}}$ and right key $K_{j_1}^R = X_{j_2}^{x_{j_1}}$;
15. U_{j_2} modifies its left key $K_{j_1}^L = X_{j_1}^{x_{j_2}}$ and right key $K_{j_2}^R = X_{j_2+1}^{x_{j_2}}$;
16. U_{j_1-1} modifies its right key $K_{j_1-1}^R = X_{j_1}^{x_{j_1-1}}$;
17. U_{j_2+1} modifies its left key $K_{j_2+1}^L = X_{j_2}^{x_{j_2+1}}$;
18. **end if**
19. **end for**

We reindex $n-m$ users $U[1 \ldots n] \setminus \{U_{l_1}, \ldots, U_{l_m}\}$. Let $U[1 \ldots n-m]$ be the new user set and $\hat{K}^L[1 \ldots n-m]$, $\hat{K}^R[1 \ldots n-m]$ respectively be the set of corresponding left, right keys.

20. **for** $i = 1$ to $n-m$ **do in parallel**
21. V_i computes $Y_i = \hat{K}_i^R / \hat{K}_i^L$ and signature $\hat{\sigma}_i = \mathcal{S}(sk_{V_i}, \hat{M}_i)$ where $\hat{M}_i = V_i|2|Y_i|d_i$;
22. V_i sends $\hat{M}_i|\hat{\sigma}_i$ to the rest of the users in $V[1, \ldots, n-m]$;
23. **end for**
24. Note that $\hat{K}_i^R = \hat{K}_{i+1}^L$ for $1 \leq i \leq n-m-1$, $\hat{K}_n^R = \hat{K}_1^L$ and $\hat{K}_{i+(n-m-1)}^R = \hat{K}_i^L$.

- **(Key Computation)**:

25. **for** $i = 1$ to $n-m$ **do in parallel**
26. **for** $j = 1$ to $n-m$, $j \neq i$ **do**
27. V_i, on receiving $\overline{M}_j|\overline{\sigma}_j$ from V_j verifies $\overline{\sigma}_j$ on $\overline{M}_j$ using the verification algorithm $\mathcal{V}$ and the verification key pk_{V_j};

28. **if** verification fails, **then** V_i sets $\mathsf{acc}_{V_i}^{d_i} = 0$, $\mathsf{sk}_{V_i}^{d_i} = \mathsf{NULL}$ and aborts;
29. **else** V_i extracts d_j from $\overline{M}_j$ and sets $\mathsf{psid}_{V_i}^{d_i} = \mathsf{psid}_{V_i}^{d_i} \cup \{(V_j, d_j)\}$;
30. **end for**
31. V_i computes $\overline{K}_{i+1}^R = Y_{i+1}\hat{K}_i^R$;
32. $j = 2$ to $n - m - 1$ **do**
33. V_i computes $\overline{K}_{i+j}^R = Y_{i+j}\overline{K}_{i+(j-1)}^R$;
34. **end for**
35. V_i verifies if $\hat{K}_{i+(n-m-1)}^R = \overline{K}_{i+(n-m-1)}^R$ (*i.e.* if $\hat{K}_i^L = \overline{K}_{i+(n-m-1)}^R$);
36. **if** verification fails, **then** V_i sets $\mathsf{acc}_{V_i}^{d_i} = 0$, $\mathsf{sk}_{V_i}^{d_i} = \mathsf{NULL}$ and aborts;
37. **else** V_i computes the session key $\mathsf{sk}_{V_i}^{d_i} = \overline{K}_1^R\overline{K}_2^R \ldots \overline{K}_{n-m}^R$, the seed $x = \mathcal{H}(\mathsf{sk}_{V_i}^{d_i})$ and stores $\hat{K}_i^L, \hat{K}_i^R$;
38. **end if**
39. **end if**
40. **end for**

end Leave

5.3 Security Analysis

We first define the Decision Diffie-Hellman (DDH) problem and refer section 2.6.3 of Chapter 2 for the security model and security notions that a group key agreement protocol should achieve. We use the notation $a \longleftarrow S$ to denote that a is chosen randomly from S.

5.3.1 Decision Diffie-Hellman (DDH) problem

Let $G = \langle g \rangle$ be a multiplicative group of some large prime order q. Then Decision Diffie-Hellman (DDH) problem on G is defined as follows:

Instance : (g^a, g^b, g^c) for some $a, b, c \in Z_q^*$.
Output : yes if $c = ab \bmod q$ and output no otherwise.

We consider two distributions as:

$$\Delta_{\mathsf{Real}} = \{a, b \longleftarrow Z_q^*, A = g^a, B = g^b, C = g^{ab} : (A, B, C)\}$$
$$\Delta_{\mathsf{Rand}} = \{a, b, c \longleftarrow Z_q^*, A = g^a, B = g^b, C = g^c : (A, B, C)\}.$$

The advantage of any probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$ in solving DDH problem on G is defined to be : $\mathsf{Adv}_{\mathcal{D},G}^{\mathsf{DDH}} = |\mathsf{Prob}[(A, B, C) \longleftarrow \Delta_{\mathsf{Real}} : \mathcal{D}(A, B, C) = 1] - \mathsf{Prob}[(A, B, C) \longleftarrow \Delta_{\mathsf{Rand}} : \mathcal{D}(A, B, C) = 1]|$.

The probability is taken over the choice of $\log_g A, \log_g B, \log_g C$ and $\mathcal{D}$'s coin tosses. $\mathcal{D}$ is said to be a (t, ϵ)-DDH distinguisher for G if $\mathcal{D}$ runs in time at most t such that $\mathsf{Adv}_{\mathcal{D},G}^{\mathsf{DDH}}(t) \geq \epsilon$.
DDH assumption : There exists no (t, ϵ)-DDH distinguisher for G. In other words, for every

probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$, $\mathsf{Adv}^{\mathsf{DDH}}_{\mathcal{D},G} \leq \epsilon$ for sufficiently small $\epsilon > 0$. *Note* : This definition of the DDH problem is essentially same as that defined in section 2.3 of Chapter 2. The only difference is in the way it is presented here.

5.3.2 Security of the Unauthenticated Protocol

We will show (Theorem 5.3.1) that our unauthenticated protocol UP is secure against passive adversary under DDH assumption. The proof, although not exactly same, is quite similar to Katz-Yung's proof [70] of security against passive adversary of the unauthenticated BD [36] protocol under DDH assumption.

Theorem 5.3.1 *The unauthenticated protocol* UP *described in Section 5.2.1 is secure against passive adversary under DDH assumption, achieves forward secrecy and satisfies the following:*

$$\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t, q_E) \leq 4\ \mathsf{Adv}^{\mathsf{DDH}}_{G}(t') + \frac{8q_E}{|G|}$$

where $t' = t + O(|\mathcal{P}|\ q_E\ t_{exp})$, t_{exp} *is the time required to perform exponentiation in* G *and* q_E *is the number of* $\mathsf{Execute}$ *query that an adversary may ask.*

Proof : Let $\mathcal{A}$ be an adversary for the unauthenticated protocol UP. Using this, we can construct an algorithm $\mathcal{D}$ which solves the DDH problem with non-negligible advantage. We first consider that the adversary $\mathcal{A}$ makes a single $\mathsf{Execute}$ query. The number of parties n (≥ 3) among which the adversary $\mathcal{A}$ asks $\mathsf{Execute}$ query is chosen by $\mathcal{A}$ itself. Moreover, since we do not use any long term secret key in our protocol UP, $\mathsf{Corrupt}$ query may simply be ignored for $\mathcal{A}$ and the protocol trivially achieves forward secrecy. The adversary $\mathcal{A}$ has access to three oracles: $\mathsf{Execute}$, Reveal and Test. To deal with the $\mathsf{Execute}$ and Reveal query, we define distributions Real and Fake' for transcript, session key pair (T, sk) as follows where Real is the real execution scenario of the protocol UP and prove the Claim 1 stated below.

$$\mathsf{Real} := \left\{ \begin{array}{ll} x_1, \ldots x_n \longleftarrow Z_q^*; & \\ X_1 = g^{x_1}, X_2 = g^{x_2}, \ldots, X_n = g^{x_n}; & \\ K_1^R = K_2^L = g^{x_1 x_2}, K_2^R = K_3^L = g^{x_2 x_3}, \ldots, K_n^R = K_1^L = g^{x_n x_1}; & : (T, \mathsf{sk}) \\ Y_1 = K_1^R / K_1^L, Y_2 = K_2^R / K_2^L, \ldots, Y_n = K_n^R / K_n^L; & \\ T = (X_1, \ldots, X_n; Y_1, \ldots, Y_n); \mathsf{sk} = K_1^R K_2^R \ldots K_n^R & \end{array} \right\}$$

$$\mathsf{Fake}' := \left\{ \begin{array}{ll} x_1, \ldots x_n \longleftarrow Z_q^*; & \\ X_1 = g^{x_1}, X_2 = g^{x_2}, \ldots, X_n = g^{x_n}; & \\ K_1^R = K_2^L = g^{x_1 x_2}, K_2^R = K_3^L = g^{x_2 x_3}, \ldots, K_{n-1}^R = K_n^L = g^{x_{n-1} x_n}; & : (T, \mathsf{sk}) \\ K_n^R = K_1^L \longleftarrow G; & \\ Y_1 = K_1^R / K_1^L, Y_2 = K_2^R / K_2^L, \ldots, Y_n = K_n^R / K_n^L; & \\ T = (X_1, \ldots, X_n; Y_1, \ldots, Y_n); \mathsf{sk} = K_1^R K_2^R \ldots K_n^R & \end{array} \right\}$$

Claim 1 : For any algorithm $\mathcal{A}$ running in time t, we have $|\mathsf{Prob}[(T,\mathsf{sk}) \longleftarrow \mathsf{Real} : \mathcal{A}(T,\mathsf{sk}) = 1] - \mathsf{Prob}[(T,\mathsf{sk}) \longleftarrow \mathsf{Fake}' : \mathcal{A}(T,\mathsf{sk}) = 1]| \leq \mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{1}{|G|}$.

Proof: We construct a distinguisher $\mathcal{D}$ for DDH problem using $\mathcal{A}$, which on an input $(A,B,C) \in G^3$, first generates a pair (T,sk) according to the distribution Dist' described below (which depends on A,B,C), then runs $\mathcal{A}$ on (T,sk) and outputs whatever $\mathcal{A}$ outputs.

$$\mathsf{Dist}' := \left\{ \begin{array}{l} x_1,\ldots x_n \longleftarrow Z_q^*; \\ X_1 = A^{x_1}, X_2 = g^{x_2}, \ldots, X_{n-1} = g^{x_{n-1}}, X_n = B^{x_n}; \\ K_1^R = K_2^L = A^{x_1 x_2}, K_2^R = K_3^L = g^{x_2 x_3}, \ldots, K_{n-2}^R = K_{n-1}^L = g^{x_{n-2} x_{n-1}}; \quad : (T,\mathsf{sk}) \\ K_{n-1}^R = K_n^L = B^{x_{n-1} x_n}, K_n^R = K_1^L = C^{x_n x_1}; \\ Y_1 = K_1^R/K_1^L, Y_2 = K_2^R/K_2^L, \ldots, Y_n = K_n^R/K_n^L; \\ T = (X_1,\ldots,X_n;Y_1,\ldots,Y_n); \mathsf{sk} = K_1^R K_2^R \ldots K_n^R \end{array} \right\}$$

The distribution Real and the distribution $\{a,b \longleftarrow Z_q^*, A = g^a, B = g^b, C = g^{ab}; (T,\mathsf{sk}) \longleftarrow \mathsf{Dist}' : (T,\mathsf{sk})\}$ are statistically equivalent as long as the exponents x_j used in Dist' are random. On the other hand, the distribution Fake' and the distribution $\{a,b \longleftarrow Z_q^*, c \longleftarrow Z_q^* \setminus \{ab\}, A = g^a, B = g^b, C = g^c; (T,\mathsf{sk}) \longleftarrow \mathsf{Dist}' : (T,\mathsf{sk})\}$ are statistically equivalent but for a factor of $\frac{1}{|G|}$. In distribution Fake', the value of $K_n^R (= K_1^L)$ is chosen uniformly at random from G whereas in Dist', this value is chosen uniformly from $G \setminus \{g^{ab}\}$. These two distributions are statistically equivalent by the self reducibility property of DDH problem. Hence $|\mathsf{Prob}[(T,\mathsf{sk}) \longleftarrow \mathsf{Real} : \mathcal{A}(T,\mathsf{sk}) = 1] - \mathsf{Prob}[(T,\mathsf{sk}) \longleftarrow \mathsf{Fake}' : \mathcal{A}(T,\mathsf{sk}) = 1]| \leq |\mathsf{Prob}[a,b \longleftarrow Z_q^* : \mathcal{D}(g^a,g^b,g^{ab}) = 1] - \mathsf{Prob}[a,b \longleftarrow Z_q^*, c \longleftarrow Z_q^* \setminus \{ab\} : \mathcal{D}(g^a,g^b,g^c) = 1]| + \frac{1}{|G|} \leq \mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{1}{|G|}$ as the time of $\mathcal{D}$ is dominated by the time t'' of $\mathcal{A}$. ■(of Claim 1)

Next we define the final distribution Fake as follows and prove the Claim 2 stated below:

$$\mathsf{Fake} := \left\{ \begin{array}{l} x_1,\ldots x_n \longleftarrow Z_q^*; \\ X_1 = g^{x_1}, X_2 = g^{x_2}, \ldots, X_n = g^{x_n}; \\ K_1^R = K_2^L, K_2^R = K_3^L, K_3^R = K_4^L, \ldots, K_n^R = K_1^L \longleftarrow G; \quad : (T,\mathsf{sk}) \\ Y_1 = K_1^R/K_1^L, Y_2 = K_2^R/K_2^L, \ldots, Y_n = K_n^R/K_n^L; \\ T = (X_1,\ldots,X_n;Y_1,\ldots,Y_n); \mathsf{sk} = K_1^R K_2^R \ldots K_n^R \end{array} \right\}$$

Claim 2: For any algorithm $\mathcal{A}$ running in time t, we have $|\mathsf{Prob}[(T,\mathsf{sk}) \longleftarrow \mathsf{Fake}' : \mathcal{A}(T,\mathsf{sk}) = 1] - \mathsf{Prob}[(T,\mathsf{sk}) \longleftarrow \mathsf{Fake} : \mathcal{A}(T,\mathsf{sk}) = 1]| \leq \mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{1}{|G|}$.

Proof : Given an adversary $\mathcal{A}$, we construct an algorithm $\mathcal{D}$ that takes $(A,B,C) \in G^3$ as input, generates a pair (T,sk) according to the distribution Dist described below (which depends on A,B,C), runs $\mathcal{A}$ on (T,sk) and outputs whatever $\mathcal{A}$ outputs.

$$
\mathsf{Dist} := \left\{
\begin{array}{l}
x_1, \ldots x_n \longleftarrow Z_q^*; \\
\textbf{if } n \text{ is even } \textbf{then} \\
\quad \textbf{for } i = 1 \ (2) \ n \ \textbf{do} \\
\quad\quad X_i = A^{x_i}, X_{i+1} = B^{x_{i+1}}; \\
\quad \textbf{end for} \\
\textbf{end if} \\
\textbf{if } n \text{ is odd } \textbf{then} \\
\quad \textbf{for } i = 1 \ (2) \ n-2 \ \textbf{do} \\
\quad\quad X_i = A^{x_i}, X_{i+1} = B^{x_{i+1}}; \\
\quad \textbf{end for} \\
\quad X_n = A^{x_n} \\
\textbf{end if} \\
\textbf{for } i = 1 \ (2) \ n-2 \ \textbf{do} \\
\quad K_i^R = K_{i+1}^L = C^{x_i x_{i+1}}, K_{i+1}^R = K_{i+2}^L = C^{x_{i+1} x_{i+2}}; \\
\textbf{end do} \\
K_{n-1}^R = K_n^L = C^{x_{n-1} x_n}, K_n^R = K_1^L \longleftarrow G; \\
Y_1 = K_1^R / K_1^L, Y_2 = K_2^R / K_2^L, \ldots, Y_n = K_n^R / K_n^L; \\
T = (X_1, \ldots, X_n; Y_1, \ldots, Y_n); \mathsf{sk} = K_1^R K_2^R \ldots K_n^R
\end{array}
\right\} : (T, \mathsf{sk})
$$

The distribution Fake' and the distribution $\{a, b \longleftarrow Z_q^*, A = g^a, B = g^b, C = g^{ab}; (T, \mathsf{sk}) \longleftarrow \mathsf{Dist} : (T, \mathsf{sk})\}$ are statistically equivalent as long as the exponents x_j used in Dist are random. On the other hand, the distribution Fake and the distribution $\{a, b \longleftarrow Z_q^*, c \longleftarrow Z_q^* \setminus \{ab\}, A = g^a, B = g^b, C = g^c; (T, \mathsf{sk}) \longleftarrow \mathsf{Dist} : (T, \mathsf{sk})\}$ are statistically equivalent but a factor of $\frac{1}{|G|}$. In distribution Fake, the values of $K_i^R (= K_{i+1}^L)$ for $1 \leq i \leq n$ are chosen uniformly at random from G and in Dist, these value are chosen uniformly from $G \backslash \{g^{ab}\}$. Then by the self reducibility property of DDH problem, we have $|\mathsf{Prob}[(T, \mathsf{sk}) \longleftarrow \mathsf{Fake}' : \mathcal{A}(T, \mathsf{sk}) = 1] - \mathsf{Prob}[(T, \mathsf{sk}) \longleftarrow \mathsf{Fake} : \mathcal{A}(T, \mathsf{sk}) = 1]| \leq |\mathsf{Prob}[a, b \longleftarrow Z_q^* : \mathcal{D}(g^a, g^b, g^{ab}) = 1] - \mathsf{Prob}[a, b \longleftarrow Z_q^*, c \longleftarrow Z_q^* \setminus \{ab\} : \mathcal{D}(g^a, g^b, g^c) = 1]| + \frac{1}{|G|} \leq \mathsf{Adv}_G^{\mathsf{DDH}}(t'') + \frac{1}{|G|}$ as the time of $\mathcal{D}$ is dominated by that of $\mathcal{A}$ (which is t''). ■(of Claim 2)

Now we provide the proof of the following claim which deals with the Test query of $\mathcal{A}$.

Claim 3: For any computationally-unbounded adversary $\mathcal{A}$, we have $\mathsf{Prob}[(T, \mathsf{sk}_0) \longleftarrow \mathsf{Fake}; \mathsf{sk}_1 \longleftarrow G; b \longleftarrow \{0, 1\} : \mathcal{A}(T, \mathsf{sk}_b) = b] = \frac{1}{2}$.

Proof : In Fake, let $v_i^R := \log_g K_i^R, 1 \leq i \leq n$. Then we have the following system of equations: $\log_g Y_1 = -v_n^R + v_1^R; \log_g Y_2 = -v_1^R + v_2^R; \ldots; \log_g Y_n = -v_{n-1}^R + v_n^R$. Furthermore, $\mathsf{sk} = g^{v_1^R + v_2^R + \cdots + v_n^R}$ gives the equation $\log_g \mathsf{sk} = v_1^R + v_2^R + \cdots + v_n^R$ which is linearly independent from the above system of equations. This implies that the session key sk is independent of the transcript T in Fake. Hence for any computationally unbounded adversary $\mathcal{A}$, $\mathsf{Prob}[(T, \mathsf{sk}_0) \longleftarrow \mathsf{Fake}; \mathsf{sk}_1 \longleftarrow G, b \longleftarrow \{0, 1\} : \mathcal{A}(T, \mathsf{sk}_b) = b] = \frac{1}{2}$. ■(of Claim 3)

Now $\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP},\mathcal{A}}(t,1) := |2\mathsf{Prob}[\mathsf{Succ}] - 1| = 2|\mathsf{Prob}[(T,\mathsf{sk}_0) \longleftarrow \mathsf{Real}, \mathsf{sk}_1 \longleftarrow G, b \longleftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) = b] - \frac{1}{2}| = 2|\mathsf{Prob}[(T,\mathsf{sk}_0) \longleftarrow \mathsf{Real}, \mathsf{sk}_1 \longleftarrow G, b \longleftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) = b] - \mathsf{Prob}[(T,\mathsf{sk}_0) \longleftarrow \mathsf{Fake}, \mathsf{sk}_1 \longleftarrow G, b \longleftarrow \{0,1\} : \mathcal{A}(T,\mathsf{sk}_b) = b]|$ by Claim 3 and using Claim 1 and Claim 2, we obtain $\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t,1) \leq 4\mathsf{Adv}^{\mathsf{DDH}}_{G}(t'') + \frac{4}{|G|}$. Then by applying the self-reducibility property of DDH problem, we get the result stated in the Theorem.

Consider the case for $q_E(> 1)$ $\mathsf{Execute}$ query. The adversary first generates q_E tuples (A_i, B_i, C_i), $1 \leq i \leq q_E$ with the following properties from the tuple $(A, B, C) \in G^3$ given to the adversary.

1. If $(A,B,C) \longleftarrow \Delta_{\mathsf{Real}}$, then $(A_i, B_i, C_i) \longleftarrow \Delta_{\mathsf{Real}}$ for all i, $1 \leq i \leq q_E$ with (A_i, B_i) randomly distributed in G^2 (independently of anything else).
2. If $(A,B,C) \longleftarrow \Delta_{\mathsf{Rand}}$, then $(A_i, B_i, C_i) \longleftarrow \Delta_{\mathsf{Rand}}$ for all i, $1 \leq i \leq q_E$ (independently of anything else) with all but a probability $\frac{q_E}{|G|}$ it will be the case that $\log_g C_i \neq \log_g A_i \log_g B_i$ for all i.

Then proceeding in the similar way as above of defining distributions Real, $\mathsf{Fake'}$, $\mathsf{Dist'}$, Fake, Dist, we may define distributions Real_{q_E}, Fake'_{q_E}, Dist'_{q_E}, Fake_{q_E} and Dist_{q_E}which simply consist of q_E independent copies of each of the corresponding distributions. In case of Dist'_{q_E} and Dist_{q_E}, we use the corresponding tuple (A_i, B_i, C_i) for the i-th copy. We use notation $(\vec{T}, \vec{\mathsf{sk}})$ to denote the transcript/session key pair generated by these distributions. Then similar to the claims 1, 2 and 3, we can prove the following claims:

Claim 4 : For any algorithm $\mathcal{A}$ running in time t, we have $|\mathsf{Prob}[(\vec{T}, \vec{\mathsf{sk}}) \longleftarrow \mathsf{Real}_{q_E} : \mathcal{A}(\vec{T}, \vec{\mathsf{sk}}) = 1] - \mathsf{Prob}[(\vec{T}, \vec{\mathsf{sk}}) \longleftarrow \mathsf{Fake}'_{q_E} : \mathcal{A}(\vec{T}, \vec{\mathsf{sk}}) = 1]| \leq \mathsf{Adv}^{\mathsf{DDH}}_{G}(t') + \frac{2q_E}{|G|}$, where t' is as in the statement of the Theorem.

Claim 5 : For any algorithm $\mathcal{A}$ running in time t, we have $|\mathsf{Prob}[(\vec{T}, \vec{\mathsf{sk}}) \longleftarrow \mathsf{Fake}'_{q_E} : \mathcal{A}(\vec{T}, \vec{\mathsf{sk}}) = 1] - \mathsf{Prob}[(\vec{T}, \vec{\mathsf{sk}}) \longleftarrow \mathsf{Fake}_{q_E} : \mathcal{A}(\vec{T}, \vec{\mathsf{sk}}) = 1]| \leq \mathsf{Adv}^{\mathsf{DDH}}_{G}(t') + \frac{2q_E}{|G|}$, where t' is as in the statement of the Theorem.

Claim 6: For any computationally-unbounded adversary $\mathcal{A}$, we have $\mathsf{Prob}[(\vec{T}, \vec{\mathsf{sk}}_0) \longleftarrow \mathsf{Fake}; \vec{\mathsf{sk}}_1 \longleftarrow G^{q_E}; b \longleftarrow \{0,1\} : \mathcal{A}(\vec{T}, \vec{\mathsf{sk}}_b) = b] = \frac{1}{2}$.

These three claims yield the result stated in the Theorem. ■

Note : If n is even, then we need not to define the intermediate (T, sk) distribution Fake'. In this case, we can obtain a smaller upper bound of $\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(1, q_E)$ considering only the distributions Real and Fake and defining Dist as in the proof of Claim 2. Consequently, we get a more tighter upper bound for $\mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t, q_E)$.

5.3.3 Security of the Authenticated (Static) Protocol

We prove that the security of our static authenticated protocol AP (subsection 5.2.2) relies on that of UP under the assumption that the underlying signature scheme DSig is secure. In fact, given any active adversary attacking AP, we can construct a passive adversary attacking UP of subsection 5.2.1. We state the security result of AP below in Theorem 5.3.2. Similar to the security proof of our tree-based static authenticated protocol in section 4.3.2 of Chapter 4, our proof technique is based on the proof technique used by Katz and Yung [70]. However, unlike the Katz-Yung compiler, we have done away with the random nonces and hence are able to reduce the number of rounds by one. Thus we gain in communication complexities. The security implication remains as in the Katz-Yung BD protocol. The way we bind session identity with each message in our protocol run, enables to handle replay attacks without using random nonces as in Katz-Yung. The session identity has a very important role in the protocol as we will show shortly. In our unauthenticated protocol, there are no long term secret keys. Thus we can avoid the Corrupt oracle queries and the unauthenticated protocol trivially achieves forward secrecy. Then following Katz-Yung, we can avoid Corrupt query for authenticated protocol as well, because this query for the authenticated protocol outputs long-term secret key (if any), defined as part of the unauthenticated protocol.

Theorem 5.3.2 *The authenticated protocol* AP *described in section 5.2.2 is secure against active adversary under DDH assumption, achieves forward secrecy and satisfies the following:*

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{AP}}(t, q_E, q_S) \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + \frac{q_S}{2}) + |\mathcal{P}|\ \mathsf{Adv}_{\mathsf{DSig}}(t')$$

where q_E and q_S are respectively the maximum number of Execute *and* Send *query an adversary may ask.*

Proof : Let $\mathcal{A}'$ be an adversary which attacks the authenticated protocol AP. Using this we construct an adversary $\mathcal{A}$ which attacks the unauthenticated protocol UP. We first have the following claim.

Claim : Let Forge be the event that a signature of DSig is forged by $\mathcal{A}'$. Then $\mathsf{Prob}[\mathsf{Forge}] \leq |\mathcal{P}|\ \mathsf{Adv}_{\mathsf{DSig}}(t')$.

Proof of Claim: Suppose the event Forge occurs. Then $\mathcal{A}'$ makes a query of the type $\mathsf{Send}(V, i, Y)$ where Y is either of the form $Y = U_k|1|X_k|\sigma_k$ with $\mathcal{V}(pk_{U_k}, U_k|1|X_k, \sigma_k) = 1$ or of the form $Y = U_k|2|X_k|d_k|\sigma_k$ with $\mathcal{V}(pk_{U_k}, U_k|2|X_k|d_k|, \sigma_k) = 1$ for some instance $\Pi^{d_k}_{U_k}$ with $X_k \in G$ and σ_k was not output by any instance of U_k on the respective messages. Using $\mathcal{A}'$, we construct an algorithm $\mathcal{F}$ that forges a signature for DSig as follows: Given a public key pk, algorithm $\mathcal{F}$ chooses a random $U \in \mathcal{P}$ and sets $pk_U = pk$. The other public keys and private keys for the system are generated honestly by $\mathcal{F}$. The forger $\mathcal{F}$ simulates all oracle queries of $\mathcal{A}'$ by executing protocol AP itself, obtaining the necessary signatures with respect to pk_U, as needed, from its signing oracle.

Thus $\mathcal{F}$ provides a perfect simulation for $\mathcal{A}'$. If $\mathcal{A}'$ ever outputs a new valid message/signature pair with respect to $pk_U = pk$, then $\mathcal{F}$ outputs this pair as its forgery. The success probability of $\mathcal{F}$ is equal to $\frac{\mathsf{Prob[Forge]}}{|\mathcal{P}|}$ and hence $\mathsf{Prob[Forge]} \leq |\mathcal{P}|\ \mathsf{Adv}_{\mathsf{DSig}}(t')$. ■(of Claim)

Now we describe the construction of the passive adversary $\mathcal{A}$ attacking UP that uses adversary $\mathcal{A}'$ attacking AP. Adversary $\mathcal{A}$ uses a list tlist. It stores pairs of session IDs and transcripts in tlist.

Adversary $\mathcal{A}$ generates the verification/signing keys pk_U, sk_U for each user $U \in \mathcal{P}$ and gives the verification keys to $\mathcal{A}'$. If ever the event Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}$ outputs whatever bit is eventually output by $\mathcal{A}'$. Note that since the signing and verification keys are generated by $\mathcal{A}$, it can detect occurrence of the event Forge.

$\mathcal{A}$ simulates the oracle queries of $\mathcal{A}'$ using its own queries to the Execute *oracle.* The idea is that the adversary $\mathcal{A}$ queried its Execute oracle to obtain a transcript T of UP for each Execute query of $\mathcal{A}'$ and also for each initial send query $\mathsf{Send}_0(U, i, *)$ of $\mathcal{A}'$. $\mathcal{A}$ then patches appropriate signatures with the messages in T to obtain a transcript T' of AP and uses T' to answer queries of $\mathcal{A}'$. Since by assumption, $\mathcal{A}'$ can not forge, $\mathcal{A}'$ is 'limited' to send messages already contained in T'. This technique provides a good simulation. We discuss details below.

Execute queries: Suppose $\mathcal{A}'$ makes a query $\mathsf{Execute}((U_{i_1}, d_1), \ldots, (U_{i_k}, d_k))$. This means that instances $\Pi^{d_1}_{U_{i_1}}, \ldots, \Pi^{d_k}_{U_{i_k}}$ are involved in this session. $\mathcal{A}$ defines $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ and sends the execute query to its Execute oracle. It receives as output a transcript T of an execution of UP. It appends (S, T) to tlist. Adversary $\mathcal{A}$ then expands the transcript T for the unauthenticated protocol into a transcript T' for the authenticated protocol according to the modification described in Section 5.2.2. It returns T' to $\mathcal{A}'$.

Send queries: The first send query that $\mathcal{A}'$ makes to an instance is to start a new session. We will denote such queries by Send_0 queries. To start a session between unused instances $\Pi^{d_1}_{U_{i_1}}, \ldots, \Pi^{d_k}_{U_{i_k}}$, the adversary has to make the send queries: $\mathsf{Send}_0(U_{i_j}, d_j, \langle U_{i_1}, \ldots, U_{i_k}\rangle \setminus U_{i_j})$ for $1 \leq j \leq k$. Note that these queries may be made in any order. When all these queries have been made, $\mathcal{A}$ sets $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ and makes an Execute query to its own execute oracle. It receives a transcript T in return and stores (S, T) in the list tlist.

Assuming that signatures cannot be forged, any subsequent Send query (i.e., after a Send_0 query) to an instance Π^i_U is a properly structured message with a valid signature. For any such Send query, $\mathcal{A}$ verifies the query according to the algorithm of Section 5.2.2. If the verification fails, $\mathcal{A}$ sets $\mathsf{acc}^i_U = 0$ and $\mathsf{sk}^i_U = \mathsf{NULL}$ and aborts Π^i_U. Otherwise, $\mathcal{A}$ performs the action to be done by Π^i_U in the authenticated protocol. This is done in the following manner: $\mathcal{A}$ first finds the unique entry (S, T) in tlist such that $(U, i) \in S$. Such a unique entry exists for each instance by assumption (see Remark 2.6.2). Now from T, $\mathcal{A}$ finds the appropriate message which corresponds to the message sent by $\mathcal{A}'$ to Π^i_U. From the transcript T, adversary $\mathcal{A}$ finds the next public information to be output by Π^i_U and returns it to $\mathcal{A}'$.

Reveal/Test queries : Suppose $\mathcal{A}'$ makes the query $\mathsf{Reveal}(U, i)$ or $\mathsf{Test}(U, i)$ to an instance Π^i_U for which $\mathsf{acc}^i_U = 1$. At this point the transcript T' in which Π^i_U participates has already been

defined. Now $\mathcal{A}$ finds the unique pair (S,T) in tlist such that $(U,i) \in S$. Assuming that the event Forge does not occur, T is the unique unauthenticated transcript which corresponds to the transcript T'. Then $\mathcal{A}$ makes the appropriate Reveal or Test query to one of the instances involved in T and returns the result to $\mathcal{A}'$.

As long as Forge does not occur, the above simulation for $\mathcal{A}'$ is perfect. Whenever Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. So $\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Forge}] = \frac{1}{2}$. Now

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} &:= 2\,|\mathsf{Prob}_{\mathcal{A},\mathsf{UP}}[\mathsf{Succ}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Forge}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Forge}]\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \overline{\mathsf{Forge}}] + (1/2)\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 1/2| \\
&= 2\,|\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}] - \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Forge}] + (1/2)\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 1/2| \\
&\geq |2\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}] - 1| - |\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Forge}] - 2\ \mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ} \wedge \mathsf{Forge}]| \\
&\geq \mathsf{Adv}_{\mathcal{A}',\mathsf{AP}} - \mathsf{Prob}[\mathsf{Forge}]
\end{aligned}$$

The adversary $\mathcal{A}$ makes an $\mathsf{Execute}$ query for each $\mathsf{Execute}$ query of $\mathcal{A}'$. Also $\mathcal{A}$ makes an $\mathsf{Execute}$ query for each session started by $\mathcal{A}'$ using Send queries. Since a session involves at least two instances, such an $\mathsf{Execute}$ query is made after at least two Send queries of $\mathcal{A}'$. The total number of such $\mathsf{Execute}$ queries is at most $q_S/2$, where q_S is the number of Send queries made by $\mathcal{A}'$. The total number of $\mathsf{Execute}$ queries made by $\mathcal{A}$ is at most $q_E + q_S/2$, where q_E is the number of $\mathsf{Execute}$ queries made by $\mathcal{A}'$.
Also since $\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_S/2)$ by assumption, we obtain:

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{AP}} \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_S/2) + \mathsf{Prob}[\mathsf{Forge}].$$

This yields the statement of the theorem. ■

5.3.4 Security of the Dynamic Authenticated Protocol

In this subsection, we will show that the modifications described in Section 5.2.3 converts the protocol UP of Section 5.2.1 into a secure dynamic authenticated key agreement protocol DAP. Assuming that the signature scheme DSig is secure, we can convert any adversary attacking the protocol DAP into an adversary attacking the protocol UP. We ignore $\mathsf{Corrupt}$ queries since our protocol DAP does not use any long-term secret keys. Thus the protocol DAP trivially achieves forward secrecy. We state below our security result.

Theorem 5.3.3 *The dynamic authenticated key agreement protocol* DAP *described in Section 5.2.3 satisfies the following:*

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{DAP}}(t, q_E, q_J, q_L, q_S) \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + (q_J + q_L + q_S)/2) + |\mathcal{P}|\ \mathsf{Adv}_{\mathsf{DSig}}(t')$$

where $t' \leq t + (|\mathcal{P}|q_E + q_J + q_L + q_S)t_{\mathsf{DAP}}$*, where* t_{DAP} *is the time required for execution of* DAP *by any one of the users.*

Proof : Let $\mathcal{A}'$ be an adversary which attacks the dynamic authenticated protocol DAP. Using this we construct an adversary $\mathcal{A}$ which attacks the unauthenticated protocol UP. As in the previous proof, we have the following claim.
Claim : Let Forge be the event that a signature is forged by $\mathcal{A}'$. Then $\mathsf{Prob}[\mathsf{Forge}] \leq |\mathcal{P}|\ \mathsf{Adv}_{\mathsf{DSig}}(t')$.

Now we describe the construction of the passive adversary $\mathcal{A}$ attacking UP that uses adversary $\mathcal{A}'$ attacking DAP. Adversary $\mathcal{A}$ can execute the unauthenticated protocol UP several times among any subset of $\mathcal{P}$ and also can obtain the session key of the protocol execution by making Reveal queries to any instances involved in the session. *We will show that $\mathcal{A}$ itself simulates the* Join *and* Leave *queries of $\mathcal{A}'$ using its own* Execute *and* Reveal *oracle.* Adversary $\mathcal{A}$ maintains a list Tlist to store pairs of session IDs and transcripts. It also uses two lists Jlist and Llist to be specified later.

Adversary $\mathcal{A}$ generates the verification/signing keys pk_U, sk_U for each user $U \in \mathcal{P}$ and gives the verification keys to $\mathcal{A}'$. If ever the event Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. Otherwise, $\mathcal{A}$ outputs whatever bit is eventually output by $\mathcal{A}'$. Note that since the signing and verification keys are generated by $\mathcal{A}$, it can detect occurrence of the event Forge. $\mathcal{A}$ simulates the oracle queries of $\mathcal{A}'$ using its own queries to the Execute and Reveal oracles. We provide details below.

Execute queries: These queries are simulated as in the proof of Theorem 5.3.2.

Send queries: Apart from the usual send queries, there are two special type of send queries, Send_J and Send_L.

If the set $S_1 = \{(U_{i_{k+1}}, d_{k+1}), \ldots, (U_{i_{k+l}}, d_{k+l})\}$ of unused instances wants to join the group $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$, then $\mathcal{A}'$ will make $\mathsf{Send}_J(U_{i_j}, d_j, \langle U_{i_1}, \ldots, U_{i_k}\rangle)$ query for all j, $k+1 \leq j \leq k+l$. These queries initiate $\mathsf{Join}(S, S_1)$ query . Note that the instances in S might have already executed either the unauthenticated (a) key agreement protocol or (b) join protocol or (c) leave protocol. Accordingly, $\mathcal{A}$ first finds any one of the following form of a unique entry: (1) (S, T) in Tlist or (2) (S', S'', T) in Jlist with $S = S' \cup S''$ or (3) (S', S'', T) in Llist with $S = S' \setminus S''$. If no such entry, $\mathcal{A}$ makes an execute query to its own execute oracle on S, gets a transcript T and stores (S, T) in Tlist.

In case $(S, T) \in$ Tlist, $\mathcal{A}$ first makes a Reveal query to any instance in S to obtain the session key sk corresponding to T, computes the seed $x = \mathcal{H}(\mathsf{sk})$ and simulates the algorithm for Join by querying its Execute oracle (making appropriate changes). Then patching up signature in each message, $\mathcal{A}$ obtains a transcript T' and stores (S, S_1, T') in Jlist. $\mathcal{A}$ thus simulates the transcript T' of Join using its own Execute and Reveal oracles. In the remaining cases (2) and (3), T is generated by $\mathcal{A}$ itself and so $\mathcal{A}$ can simulate transcript T' of Join from T.

Similarly, when a set $S_2 = \{(U_{l_1}, d_{l_1}), \ldots, (U_{l_m}, d_{l_m})\}$ of unused instances wants to leave the group $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$, then $\mathcal{A}'$ will make $\mathsf{Send}_L(U_{i_j}, d_j, \langle U_{i_1}, \ldots, U_{i_k}\rangle)$ query for all $j \in \{l_1, \ldots, l_m\}$. These queries initiate $\mathsf{Leave}(S, S_2)$ query. As mentioned above in case of member join, $\mathcal{A}$ first finds a unique entry of the form (S, T) in Tlist or a unique entry of the form (S', S'', T) in Jlist with $S = S' \cup S''$ or a unique entry of the form (S', S'', T) in Llist with $S = S' \setminus S''$. If no such entry, then $\mathcal{A}$ makes a query to its own execute oracle on S, gets a transcript T and stores (S, T) in Tlist.

$\mathcal{A}$ then simulates the algorithm for Leave by itself and gets a modified transcript T' from T as follows: A first detects the positions in T where the new messages are to be injected or the old messages are to be replaced by new messages. $\mathcal{A}$ do these modifications in T according to the algorithm Leave described in Section 5.2.3 and gets a modified transcript T' by patching up appropriate signature with each message. Thus $\mathcal{A}$ expands T into a transcript T' for Leave algorithm. $\mathcal{A}$ stores (S, S_2, T') in Llist.

Send_0 queries are answered as in Theorem 5.3.2. The usual send queries are simulated as in Theorem 5.3.2 with the following modifications.

Suppose $\mathcal{A}'$ makes a Send query to instance Π_U^i. After proper verification, $\mathcal{A}$ finds a unique entry (S, T) in Tlist such that $(U, i) \in S$. The answer to this query is as in Theorem 5.3.2. If no such entry is found, then $\mathcal{A}$ finds a unique entry (S, S_1, T') in Jlist such that $(U, i) \in S_1$. This means that the session for Join has already been initiated. $\mathcal{A}$ then obtains the next public information for T' to be output by Π_U^i (provided all necessary information has been received by Π_U^i by send queries from $\mathcal{A}'$) and sends it to $\mathcal{A}'$. If $\mathcal{A}$ finds a unique entry (S, S_2, T') in Llist such that $(U, i) \in S_2$, then as above, the appropriate answer to the query is found from T'.

Join queries : Suppose $\mathcal{A}'$ makes a query $\mathsf{Join}(S, S_1)$ where $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, d_k)\}$ and $S_1 = \{(U_{i_{k+1}}, d_{k+1}), \ldots, (U_{i_{k+l}}, d_{k+l})$. The instances $\Pi_{U_{i_{k+1}}}^{d_{k+1}}, \ldots, \Pi_{U_{i_{k+l}}}^{d_{k+l}}$ want to join in the group $\Pi_{U_{i_1}}^{d_1}, \ldots \Pi_{U_{i_k}}^{d_k}$. $\mathcal{A}$ finds an entry of the form (S, S_1, T') in Jlist. If no such entry, then the adversary $\mathcal{A}'$ is given no output. Otherwise, $\mathcal{A}$ returns T' to $\mathcal{A}'$

Leave queries : Suppose $\mathcal{A}'$ makes a query $\mathsf{Leave}(S, S_2)$ where $S = \{(U_{i_1}, d_1), \ldots, (U_{i_k}, , d_k)\}$ and $S_2 = \{(U_{l_1}, d_{l_1}), \ldots, (U_{l_m}, d_{l_m})\}$ where $(U_{l_j}, d_{l_j}) \in S$ for $1 \leq j \leq m$. The instance $\Pi_{U_{l_1}}^{d_{l_1}}, \ldots \Pi_{U_{l_m}}^{d_{l_m}}$ want to leave the group $\Pi_{U_{i_1}}^{d_1}, \ldots, \Pi_{U_{i_k}}^{d_k}$ where $U_{l_j} \in \{U_{i_1}, \ldots, U_{i_k}\}$ for $1 \leq j \leq m$. $\mathcal{A}$ finds an entry of the form (S, S_2, T') in Llist. If no such entry, then the adversary $\mathcal{A}'$ is given no output. Otherwise, $\mathcal{A}$ returns T' to $\mathcal{A}'$.

Reveal/Test queries : Suppose $\mathcal{A}'$ makes the query $\mathsf{Reveal}(U, i)$ or $\mathsf{Test}(U, i)$ for an instance Π_U^i for which $\mathsf{acc}_U^i = 1$. At this point the transcript T' in which Π_U^i participates has already been defined. If T' corresponds to the transcript of the authenticated protocol, then $\mathcal{A}$ finds the unique pair (S, T) in Tlist such that $(U, i) \in S$. Assuming that the event Forge does not occur, T is the unique unauthenticated transcript which corresponds to the transcript T'. Then $\mathcal{A}$ makes the appropriate Reveal or Test query to one of the instances involved in T and returns the result to $\mathcal{A}'$. Otherwise, T' is the transcript for Join or Leave, as the case may be. Since T' has been simulated by $\mathcal{A}$, $\mathcal{A}$ is able to compute the modified session key and hence send an appropriate reply to $\mathcal{A}'$.

As long as Forge does not occur, the above simulation for $\mathcal{A}'$ is perfect. Whenever Forge occurs, adversary $\mathcal{A}$ aborts and outputs a random bit. So $\mathsf{Prob}_{\mathcal{A}',\mathsf{AP}}[\mathsf{Succ}|\mathsf{Forge}] = \frac{1}{2}$. Using this, one can show

$$\mathsf{Adv}_{\mathcal{A},\mathsf{UP}} \quad \geq \quad \mathsf{Adv}_{\mathcal{A}',\mathsf{DAP}} - \mathsf{Prob}[\mathsf{Forge}]$$

The adversary $\mathcal{A}$ makes an Execute query for each Execute query of $\mathcal{A}'$. $\mathcal{A}'$ makes q_J Join queries and q_L Leave queries. These queries are initialized respectively by Send_J and Send_L queries of $\mathcal{A}'$. Now each of Send_J and Send_L query of $\mathcal{A}'$ makes at most one Execute query of $\mathcal{A}$. Thus there are at most $q_J + q_L$ execute query made by $\mathcal{A}$ to respond all the Send_J and Send_L queries of $\mathcal{A}'$.

Also $\mathcal{A}$ makes an Execute query for each session started by $\mathcal{A}'$ using Send queries. Since a session involves at least two instances, such an Execute query is made after at least two Send queries of $\mathcal{A}'$. Thus there are $(q_S - q_J - q_L)/2$ execute queries of $\mathcal{A}$ to respond all other Send queries of $\mathcal{A}'$, where q_S is the number of Send queries made by $\mathcal{A}'$. Hence the total number of Execute queries made by $\mathcal{A}$ is at most $q_E + q_J + q_L + (q_S - q_J - q_L)/2 = q_E + (q_J + q_L + q_S)/2$, where q_E is the number of Execute queries made by $\mathcal{A}'$. Also since $\mathsf{Adv}_{\mathcal{A},\mathsf{UP}}(t, q_E, q_J, q_L, q_S) \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + q_J/2 + q_L/2 + q_S/2)$ by assumption, we obtain:

$$\mathsf{Adv}^{\mathsf{AKA}}_{\mathsf{DAP}} \leq \mathsf{Adv}^{\mathsf{KA}}_{\mathsf{UP}}(t', q_E + (q_J + q_L + q_S)/2) + \mathsf{Prob}[\mathsf{Forge}].$$

This yields the statement of the theorem. ■

5.4 Efficiency

Efficiency of a protocol is related to the costs of communication and computation. Communication cost involves counting total number of rounds and total messages transmitted through the network during a protocol execution. Number of rounds is a critical concern in practical environments where number of group members is large. Table 1 analyzes the efficiency of our authenticated (static) protocol with the two most efficient protocols – (1) authenticated BD protocol KY, introduced by Katz-Yung in [70], and (2) the group key agreement protocol KLL proposed by Kim *et al.* [76]. Both the protocol KY and our scheme are forward secure, achieve provable security under DDH assumption in standard model, whereas the protocol KLL is secure in the random oracle model under CDH assumption. We use the following notations:

n	total number of users in a group
R	total number of rounds
PTP	maximum number of point-to-point communication per user
Exp	maximum number of modular exponentiations computed per user
Mul	maximum number of modular multiplications computed per user
Xor	maximum number of modular XOR computed per user
Hash	maximum number of hash functions computed per user
Div	maximum number of divisions computed per user
Sig	maximum number of signatures generated per user
Ver	maximum number of signature verification per user
r.o.m.	stands for random oracle model

Protocol	Communication		Computation							Security model	Hardness assumption	Remarks
	R	PTP	Exp	Mul	Div	Sig	Ver	Xor	Hash			
KLL [76]	2	$n+1$	3	0	0	2	$n+1$	$n+1$	4	r.o.m.	CDH	dynamic
KY [70]	3	$3(n-1)$	3	$\frac{n^2}{2}+\frac{3n}{2}-3$	1	2	$2(n-1)$	0	0	standard	DDH	static
Our protocol	2	$n+1$	3	$2n-2$	1	2	$n+1$	0	0	standard	DDH	dynamic

Table 1: Protocol comparison

A description of the Burmester-Desmedt [36, 37] can be found in Section 3.4.3. This protocol is unauthenticated in the sense that it is proven to be secure against passive adversary. Katz-Yung [70] provided a compiler construction and reduce the unauthenticated BD protocol into a 3-round authenticated group key agreement. Security of this protocol is based on DDH assumption and the security analysis is in the standard model, no random oracle is used.

In a recent work, Kim *et al.* [76] proposed a constant round group key agreement, security of which is under the CDH assumption in the random oracle model. They also extend this protocol into dynamic setting where a set of users can join or leave a group. However, no security proof is provided for the dynamic variant, only the security analysis of the static authenticated protocol is given. A detail description of this scheme can be found in Section 3.4.10.

Unlike Kim *et al.* [76], our protocol is an improvement on the Katz-Yung [70] BD protocol. Our security analysis is in the standard model without using any random oracle and the scheme is provably secure under DDH assumption. We also consider the dynamic scenario and provide concrete security analysis of our dynamic variant separately.

Our protocol is more efficient as compared to the protocol of Burmester and Desmedt [36] in terms of computation power. The authentication in the BD protocol was introduced by Katz and Yung [70] that requires 3 rounds. In each round, a user sends message to the rest of the users. In contrast, each user in our protocol sends a message only to it's two neighbors in the first round and a message to the rest of the users in the second round. Our protocol differs from the BD protocol in the way the session key is computed after the rounds are over. Each user computes $\frac{n^2}{2}+\frac{3n}{2}-3$ modular multiplications in BD protocol while in our protocol each user requires to compute at most $2n$ modular multiplications. This makes our protocol much more efficient as compared to BD protocol. Besides, our protocol has the ability to detect the presence of a corrupted group member, although we cannot detect who among the group members are behaving improperly. If an invalid message is sent by a corrupted member, then this can be detected by all legitimate members of the group and the protocol execution may be stopped instantly. This feature makes our protocol interesting when the adversarial model no longer assumes that the group members are honest.

5.5 Conclusion

We present and analyze a simple and elegant constant round group key agreement protocol and enhance it to dynamic setting where a set of users can leave or join the group at any time during protocol execution with updated keys. The emphasis of this work is to achieve provable security of our scheme under DDH assumption. We provide a concrete security analysis of our protocol against

active adversary in the security model of Bresson *et al.* [32] adapting Katz-Yung [70] technique. The protocol is forward secure, efficient and fully symmetric.

Chapter 6

Concluding Remarks

Earlier, bilinear pairings, namely Weil and Tate pairings of algebraic curves have been used to reduce the Discrete Log Problem of some elliptic or hyperelliptic curve groups to the Discrete Log Problem on the multiplicative group of some finite fields. However, in recent years, bilinear pairings have found positive applications in cryptography in constructing new cryptographic primitives.

Even though a pairing-based crytographic primitive can be fully understood without any knowledge of elliptic curves, any implementation of such primitives will almost certainly involve the (modified) Weil or Tate pairing. (In fact, our tree-based GKE has been implemented using Tate-pairing as part of a project under Department of IT, Government of India.) In the preliminaries, we have, therefore, included a brief introduction to elliptic curves that quickly leads to the definition of Weil pairing. We have also included in this part, the various Diffie-Hellman problems and their variants and analogues involving pairings. The different security models for the cryptographic primitives used later have also been included.

Our major contribution in this thesis is in designing efficient and secure dynamic group key exchange(GKE) protocols. To put our contribution in the right perspective with respect to other GKE protocols, we have first included a brief survey of GKE protocols covering most of the major works in this area. This is the contribution of our paper entitled *Overview of Key Agreement Protocols* [56]. The first part of the survey gives an overview of the protocols and looks at their historical development. This is followed by descriptions of some important protocols arranged according to the following three categories.

1. Two -party key agreements
2. Three -party key agreements
3. Multi-party key agreements

In our contribution, we first construct a new paring-based GKE protocol that extends the Joux's 3-party key exchange protocol to the multi-party case. To extend Joux's protocol, we need

to consider ternary trees with certain structures. The tree-based protocol was first shown to be secure against passive adversary under the assumption that the Decisional Hash Bilinear Diffie-Hellman problem is hard. This is the content of our article entitled *Extending Joux Protocol to Multi Party Key Agreement* [9]. Authentication is then achieved by modifying the Katz-Yung technique suitable to our tree-based protocol. The main component of the Katz-Yung protocol is the use of a digital signature scheme. However, in keeping with our tree-structure we require a multisignature scheme. In each round, each user sends the signed message to the representative, who then sends the multisignature to the other users. This process is then repeated until the last round. It should be emphasized that a similar construction can be used to modify any tree-based unauthenticated GKE to an authenticated GKE protocol. This is the contribution of our article entitled *Provably Secure Authenticated Tree Based Group Key Agreement* [53].

Our tree-based GKE protocol is then extended to a dynamic authenticated protocol and we provide a proof of its security. Our protocol is designed to ensure a minimum modification to the computation already precomputed when a user leaves or joins. This is because we need to maintain the structure of the tree and a person leaving or joining may disturb this structure. So one needs to "relocate" using minimal computation. This is the content of our paper entitled *Dynamic Group Key Agreement in Tree-based Setting* [54].

A natural real life scenario is the case when a group of users leave or join. So the natural question is, can one obtain a dynamic tree-based protocol that allows a group of users to leave or join with a minimal amount of computation? This could be a further topic of research and we would like to have an elegant and secure protocol that allows simultaneous join and leave.

Next we analyse and present a simple and elegant constant round group key agreement protocol and enhance it to the dynamic setting where a group of users can leave and join. The unauthenticated protocol is, off course, a variant of the Burmester-Desmedt protocol(EUROCRYPT94). It differs from the BD protocol in two respects. It has a provision for checking whether all users honestly follow the protocol–any deviation from the protocol would be detected. Secondly, our computation of the session key is simple using lesser number of operations. As in the tree-based case, we have applied the the Katz-Yung compiler to obtain the authenticated protocol. However, unlike the Katz-Yung compiler, we have done away with the random nonces and hence we are able to reduce the number of rounds by one. Thus we gain in communication complexities. The security implication remains as in the Katz-Yung BD protocol i.e. the reduction proof remains as tight as in the Katz-Yung compiler except for a small additional (negligible) term.

Furthermore, we have obtained a dynamic version of this protocol where a group of users may join or leave. By extending the model in a natural way, we have been able to prove the security of the dynamic protocol under the assumption that the corresponding static protocol and the signature scheme are secure. Consequently, under DDH, using a secure signature scheme we have been able to convert our unauthenticated protocol to a secure dynamic GKE protocol.This summarizes our work in the article entitled *Constant Round Dynamic Group Key Agreement* [55]

One issue that has not been addressed(in this thesis) is security against malicious insiders. The models used do not provide any protection against malicious insiders. An important direction of

research would be to obtain from the protocols presented those that would address this and similar issues also.

Finally, a brief survey of pairing-based cryptography has been included in this thesis (in the appendix). Most of the important protocols based on pairings that have proofs of security, have been covered. The main purpose of this survey is not only to show the importance of pairing in modern day cryptography, but also to present the variety of cryptographic primitives that can be defined using pairing. The survey shows the depth and breadth of the applications of bilinear mappings to cryptography. Since our original purpose was to obtain some new pairing-based protocols we have included the survey in this thesis. *No claim what-so-ever is being made that our survey is exhaustive, nor do we claim that it is up-to-date.* In fact, any survey on pairing-based cryptography is bound to be out-dated once it is published–given the spurt of activities in pairing-based cryptography. So we have put up the survey on the web with the intention of updating it periodically. We have also not provided any proofs of important protocols like the Boneh-Franklin ID-based encryption scheme, mainly due to lack of space. We have also left out some other protocols due to non-availability of proper security proofs. On the other hand, we have covered all the basic schemes which will continue to be referred in the future. We believe that the survey will provide both as an introduction to the area as well as serve as a ready reckoner for those working in pairing-based cryptography. We have included it in the hope that it will serve as a motivation to a beginner. (In fact, this has been used as a reading material by Prof. Ron Rivest of MIT in his course on "Selected Topics in Cryptography (6.897)".) The above survey is part of our article entitled *Pairing Based Cryptographic Protocols : A Survey* [51].

Bibliography

[1] M. Abdalla, M. Bellare and P. Rogaway. *DHIES: An Encryption Scheme Based on the Diffie-Hellman Problem.* In proceedings of CT-RSA 2001, LNCS 2020, pp. 143-158, Springer-Verlag, 2001.

[2] M. Abdalla, P. A. Fouque and D. Pointcheval. *Password-Based Authenticated Key Exchange in the Three-Party Setting.* In proceedings of PKC 2005, LNCS 3386, Springer-Verlag, 2004.

[3] S. Al-Riyami and K. G. Paterson. *Tripartite Authenticated Key Agreement Protocols from Pairings.* In proceedings of IMA Conference of Cryptography and Coding, LNCS 2898, pp. 332-359. Also available at http://eprint.iacr.org/2002/035.

[4] G. Ateniese, M. Steiner, and G. Tsudik. *Authenticated Group Key Agreement and Friends.* In proceedings of ACM CCS 1998[1], pp. 17-26, ACM Press, 1998.

[5] G. Ateniese, M. Steiner, and G. Tsudik. *New Multi-party Authenticated Services and Key Agreement Protocols.* In Journal of Selected Areas in Communications, 18(4), pp. 1-13, IEEE, 2000.

[6] J. Baek, Y. Zheng. *Identity-Based Threshold Decryption.* In proceedings of PKC 2004, LNCS 2947, pp. 262-276, Springer-Verlag, 2004. Also available at http://eprint.iacr.org/2003/164.

[7] P. S. L. M. Barreto. *Pairing Based Crypto Lounge.* Available at http://planeta.terra.com.br/informatica/paulobarreto/pblounge.html

[8] P. S. L. M. Barreto, H. Y. Kim and M. Scott. *Efficient Algorithms for Pairing Based Cryptosystems.* In proceedings of Crypto 2002, LNCS 2442, pp. 354-368, Springer-Verlag, 2002. Also available at http://www.iacr.org/2002/008.

[9] R. Barua, R. Dutta, P. Sarkar. *Extending Joux Protocol to Multi Party Key Agreement.* In proceedings of Indocrypt 2003, LNCS 2904, pp. 205-217, Springer-Verlag, 2003. Also available at http://eprint.iacr.org/2003/062.

[10] K. Becker and U. Wille. *Communication Complexity of Group Key Distribution.* In proceedings of ACM CCS 1998, pp. 1-6, ACM Press, 1998.

[11] M. Bellare, R. Canetti, and H. Krawczyk. *A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols*. In proceedings of the 30th Annual Symposium on the Theory of Computing, pp. 419-428. ACM Press, 1998. Also available at http://www.cs.edu/users/mihir/papers/key-distribution.html/.

[12] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. *Relations Among Notions of Security for Public-key Encryption Schemes*. In proceedings of Crypto 1998, LNCS 1462, pp. 26-45, Springer-Verlag, 1998.

[13] M. Bellare and P. Rogaway. *Entity Authentication and Key Distribution*. In proceedings of Crypto 1993, LNCS 773, pp. 231-249, Springer-Verlag, 1994.

[14] M. Bellare and P. Rogaway. *Provably Secure Session Key Distribution: The Three-party Case*. In proceedings of STOC 1995, pp. 57-66, ACM Press, 1995.

[15] M. Bellare and P. Rogaway. *Random Oracles are Practical : A Paradigm for Designing Efficient Protocols*. In proceedings of ACM CCS 1993, pp. 62-73, ACM Press, 1993.

[16] S. Blake-Wilson and A. Menezes. *Security Proofs for Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques*. In proceedings of the 5th International Workshop on Security Protocols, LNCS 1361, pp. 137-158, Springer-Verlag, 1997.

[17] S. Blake-Wilson, D. Johason and A. Menezes. *Key Agreement Protocols and Their Security Analysis*. In proceedings of the sixth IMA International Conference on Cryptography and Coding, LNCS 1355, pp. 30-45, Springer-Verlag, 1997.

[18] A. Boldyreva. *Efficient Threshold Signature, Multi-signature and Blind Signature Schemes Based on the Gap-Diffie-Hellman-Group Signature Scheme*. In proceedings of PKC 2003, LNCS 2139, pp. 31-46, Springer-Verlag, 2003. Also available at http://www.iacr.org/2002/118.

[19] A. Boldyreva, A. Palacio and B. Warinschi. *Secure Proxy Signature Schemes for Delegation of Signing Rights*. Available at http://eprint.iacr.org/2003/096.

[20] D. Boneh, X. Boyen. *Short Signatures Without Random Oracles*. In proceedings of Eurocrypt 2004, LNCS 3027, pp. 56-73, Springer-Verlag, 2004.

[21] D. Boneh, X. Boyen. *Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles*. In proceedings of Eurocrypt 2004, LNCS 3027, pp. 223-238, Springer-Verlag, 2004.

[22] D. Boneh, X. Boyen. *Secure Identity-Based Encryption Without Random Oracles*. In proceedings of Crypto 2004, LNCS 3152, pp. 443-459, Springer-Verlag, 2004.

[23] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano. *Public Key Encryption with Keyword Search*. In proceedings of Eurocrypt 2004, LNCS 3027, pp. 506-522, Springer-Verlag, 2004. Also available at http://eprint.iacr.org/2003/195.

[24] D. Boneh and M. Franklin. *Identity-Based Encryption from Weil Pairing.* In proceedings of Crypto 2001, LNCS 2139, pages 213-229, Springer-Verlag, 2001.

[25] D. Boneh, C. Gentry, B. Lynn and H. Shacham. *Aggregate and Verifiably Encrypted Signature from Bilinear Maps.* In proceedings of Eurocrypt 2003, LNCS 2248, pp. 514-532, Springer-Verlag, 2003.

[26] D. Boneh, B. Lynn, and H. Shacham. *Short Signature from Weil Pairing.* In proceedings of Asiacrypt 2001, LNCS 2248, pp. 213-229, Springer-Verlag, 2001.

[27] D. Boneh, I. Mironov, V. Shoup. *A Secure Signature Scheme from Bilinear Maps.* In proceedings of CT-RSA 2003, LNCS 2612, pp. 98-110, Springer-Verlag, 2003.

[28] D. Boneh and A. Silverberg. *Applications of Multi-linear forms to Cryptography.* In Contemporary Mathematics, 324, American Mathematical Society, pp. 71-90, 2003. Also available at http://eprint.iacr.org/2002/080.

[29] C. Boyd and J. M. G. Nieto. *Round-optimal Contributory Conference Key Agreement.* In proceedings of PKC 2003, LNCS 2567, pp. 161-174, Springer-Verlag, 2003.

[30] E. Bresson and D. Catalano. *Constant Round Authenticated Group Key Agreement via Distributed Computing.* In proceedings of PKC 2004, LNCS 2947, pp. 115-129, Springer-Verlag, 2004.

[31] E. Bresson, O. Chevassut and D. Pointcheval. *The Group Diffie-Hellman Problems.* In proceedings of SAC 2002, LNCS 2595, pp. 325-338, Springer-Verlag, 2002.

[32] E. Bresson, O. Chevassut, and D. Pointcheval. *Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions.* In proceedings of Eurocrypt 2002, LNCS 2332, pp. 321-336, Springer-Verlag, 2002.

[33] E. Bresson, O. Chevassut, and D. Pointcheval. *Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case.* In proceedings of Asiacrypt 2001, LNCS 2248, pp. 290-309, Springer-Verlag, 2001.

[34] E. Bresson, O. Chevassut, A. Essiari and D. Pointcheval. *Mutual Authentication and Group Key Agreement for Low-power Mobile Devices.* Computer Communication, 27(17), pp. 1730-1737, 2004. A preliminary version appeared in proceedings of the 5th IFIP-TC6/IEEE , MWCN 2003, pp. 59-62, 2003. Full version available at http://www.di.ens.fr/ bresson.

[35] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater. *Provably Authenticated Group Diffie-Hellman Key Exchange.* In proceedings of ACM CCS 2001, pp. 255-264, ACM Press, 2001.

[36] M. Burmester and Y. Desmedt. *A Secure and Efficient Conference Key Distribution System.* In proceedings of Eurocrypt 1994, LNCS 950, pp. 275-286, Springer-Verlag, 1995.

[37] M. Burmester and Y. Desmedt. *A Secure and Scalable Group Key Exchange System.* In Information Processing Letters, 94(3), pp. 137-143, 2005.

[38] R. Canetti and H. Krawczyk. *Analysis of Key-exchange Protocols and Their Use for Building Secure Channels.* In proceedings of Cryptographic Techniques, LNCS 2045, pp. 453-474. Springer-Verlag, 2001.

[39] L. Charlap and D. Robbins. *An Elementary introduction to elliptic curves.* CRD Expository Report No. 31, Institute for Defence Analysis, Princeton, December, 1988.

[40] D. Chaum, H. V. Antwerpen. *Undeniable Signatures.* In proceedings of Crypto 1989, LNCS 435, pp. 212-217, Springer-Verlag, 1989.

[41] J. Cha and J. Cheon. *An Identity-based Signature from Gap Diffie-Hellman Groups.* In proceedings of PKC 2003, LNCS 2567, pp. 18-30, Springer-Verlag, 2003. Also available at http://eprint.iacr.org/2002/018.

[42] Z. Chen. *Security Analysis on Nalla-Reddy's ID-based Tripartite Authenticated Key Agreement Protocol.* Available at http://eprint.iacr.org/2003/103.

[43] L. Chen and C. Kudla. *Identity Based Authenticated Key Agreement Protocols from Pairings.* Available at http://eprint.iacr.org/2002/184.

[44] X. Chen, F. Zhang, K. Kim. *A New ID-based Group Signature Scheme from Bilinear Pairings.* In proceedings of WISA 2003, pp. 585-592, August 2003, Jeju Island (KR). Also available at http://eprint.iacr.org/2003/116.

[45] K. Y. Choi, J. Y. Hwang and D. H. Lee. *Efficient ID-based Group Key Agreement with Bilinear Maps.* In proceedings of PKC 2004, LNCS 2947, Springer-Verlag, 2004.

[46] K. -K. R. Choo. *Revisit of McCullagh-Barreto Two-Party ID-Based Authenticated Key Agreement Protocols.* Available at http://eprint.iacr.org/204/343.

[47] C. Cocks. *An Identity Based Encryption Scheme based on Quadratic Residues.* In Cryptography and Coding, LNCS 2260, pp. 360-363, Springer-Verlag, 2001. http://www.cesg.gov.uk/site/ast/idpkc/media/ciren.pdf.

[48] R. Cramer and V. Shoup. *A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Cipher text Attack.* In proceedings of Crypto 1998, LNCS 1462, pp. 13-25, Springer-Verlag, 1998.

[49] W. Diffie, M. Hellman. *New Directions in Cryptography.* In IEEE Transaction on Information Theory, IT-22 (6), pp. 644-654, 1976.

[50] Y. Dodis, L. Reyzin. *Breaking and Repairing Optimistic Fair Exchange from PODC 2003.* In proceedings of ACM Workshop on Digital Rights Management 2003, pp. 47-54, ACM Press.

[51] R. Dutta, R. Barua and P. Sarkar. *Pairing Based Cryptographic Protocols : A Survey.* Manuscript 2004. Available at http://eprint.iacr.org/2004/064.

[52] R. Dutta, R. Barua and P. Sarkar. *Authenticated Multi-party Key Agreement : A Provably Secure Tree Based Scheme using Pairing.* In proceedings of National Workshop on Cryptology 2004, Kerala, India, October 2004.

[53] R. Dutta, R. Barua and P. Sarkar. *Provably Secure Authenticated Tree Based Group Key Agreement.* In proceedings of ICICS 2004, LNCS 3269, pp. 92-104, Springer-Verlag, 2004. Also available at http://eprint.iacr.org/2004/090.

[54] R. Dutta and R. Barua. *Dynamic Group Key Agreement in Tree-based Setting.* In proceedings of ACISP 2005, LNCS 3574, pp. 101-112, Springer-Verlag, 2005. Also available at http://eprint.iacr.org/2005/131.

[55] R. Dutta and R. Barua. *Constant Round Dynamic Group Key Agreement.* In proceedings of ISC 2005, LNCS 3650, pp. 74-88, Springer-Verlag, 2005.

Also available at http://eprint.iacr.org/2005/221.

[56] R. Dutta and R. Barua. *Overview of Key Agreement Protocols.* Manuscript 2005. Available at http://eprint.iacr.org/2005/289.

[57] G. Frey, H. Ruck. *A Remark Concerning m-divisibility and The Discrete Logarithm in the Divisor Class Group of Curves.* In Mathematics of Computation, 62, pp. 865-874, 1994.

[58] E. Fujisaki, T. Okamoto. *Secure Integration of Asymmetric and Symmetric Encryption Schemes.* In proceedings of Crypto 1999, LNCS 1666, pp. 577-554, Springer-Verlag, 1999.

[59] S. Galbraith, K. Harrison and D. Soldera. *Implementing the Tate Pairing.* In proceedings of Algorithm Number Theory Symposium - ANTS V, LNCS 2369, pp. 324-337, Springer-Verlag, 2002.

[60] C. Gentry and A. Silverberg. *Hierarchical ID-based Cryptography.* In proceedings of Asiacrypt 2002, LNCS 2501, pp. 548-566, Springer-Verlag, 2002.

[61] M. Girault and J. C. Paillers. *An Identity Based Scheme Providing Zero-knowledge Authenticated Key Exchange.* In proceedings of ESORICS 1990, pp. 173-184, 1990.

[62] S. Goldwasser, S. Micali and R. Rivest. *A Digital Signature Scheme Secure against Adaptive Chosen -Message Attacks.* In SIAM Journal of Computing, 17(2), pp. 281-308, April 1998.

[63] F. Hess. *Efficient Identity Based Signature Schemes Based on Pairings.* In proceedings of SAC 2002, LNCS 2595, pp. 310-324, Springer-Verlag, 2002.

[64] I. Ingemarsson, D. T. Tang, and C. K. Wong. *A Conference Key Distribution System.* In IEEE Transactions on Information Theory 28(5), pp. 714-720, 1982.

[65] I. R. Jeong, J. Katz and D. H. Lee. *One-Round Protocols for Two-Party Authenticated Key Exchange.* In proceedings of ACNS 2004, LNCS 3089, pp. 220-232, Springer-Verlag, 2004.

[66] A. Joux. *A One Round Protocol for Tripartite Diffie-Hellman.* In proceedings of ANTS 4, LNCS 1838, pp. 385-394, Springer-Verlag, 2000.

[67] A. Joux, K. Nguyen. *Separating DDH from DH in Cryptographic Groups.* Available at http://eprint.iacr.org/2001/003.

[68] M. Just and S. Vaudenay. *Authenticated Multi-Party Key Agreement.* In proceedings of Asiacrypt 1996, LNCS 1163, pp. 36-49, Springer-Verlag, 1996.

[69] B. Kaliski. Contribution to ANSI X9F1 and IEEE P1363 working group, June 1998.

[70] J. Katz and M. Yung. *Scalable Protocols for Authenticated Group Key Exchange.* In proceedings of Crypto 2003, LNCS 2729, pp. 110-125, Springer-Verlag, 2003.

[71] M. Kim, K. Kim. *A New Identification Scheme Based on the Gap Diffie-Hellman Problem.* In SCIS 2002, 1/2, pp. 349-352, Shirahama, Japan, 2003.

[72] M. Kim, K. Kim. *A New Identification Scheme Based on the Bilinear Diffie-Hellman Problem.* In proceedings of ACISP 2002, LNCS 2384, pp. 368-378, Springer-Verlag, 2002.

[73] Y. Kim, A. Perrig, and G. Tsudik. *Simple and Fault-tolerant Key Agreement for Dynamic Collaborative Groups.* In proceedings of ACM CCS 2000, pp. 235-244, ACM press, 2000.

[74] Y. Kim, A. Perrig, and G. Tsudik. *Tree Based Group Key Agreement.* Available at http://eprint.iacr.org/2002/009.

[75] Y. Kim, A. Perrig, and G. Tsudik. *Communication-efficient Group Key Agreement.* In proceedings of the 17th International Information Security Conference, IFIP SEC 2001, pp. 229-244, 2001.

[76] H. J. Kim, S. M. Lee and D. H. Lee. *Constant-Round Authenticated Group Key Exchange for Dynamic Groups.* In proceedings of Asiacrypt 2004, LNCS 3329, pp. 245-259, Sringer-Verlag, 2004.

[77] N. Koblitz. *A Course in Number Theory and Cryptography.* Springer-Verlag, 1987.

[78] H. Krawczyk. *SIGMA: The "SIGn-and-MAc" Approach to Authenticate Diffie-Hellman and its use in the Ike Protocols.* In proceedings of Crypto 2003, LNCS 2729, pp. 400-425, Springer-Verlag, 2003.

[79] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. *An Efficient Protocol for Authenticated Key Agreement.* Technical Report CORR 98-05, Department of C & O, University of Waterloo, 1998. Also available at http://citeseer.nj.nec.com/law98efficient.

[80] H. K. Lee, H. S. Lee and Y. R. Lee. *Multi-party Authenticated Key Agreement Protocols from Multi linear Forms*. Available at http://eprint.iacr.org/2002/166.

[81] B. Libert, J. J. Quisquater. *Efficient Revocation and Threshold Pairing Based Cryptosystems*. In proceedings of PODC 2003, pp. 163-171, ACM Press.

[82] B. Libert, J. J. Quisquater. *New Identity Based Signcryption Schemes from Pairings*. Available at http://eprint.iacr.org/2003/023.

[83] A. Lysyanskaya. *Unique Signatures and Verifiable Random Functions from DH-DDH Separation*. In proceedings of Crypto 2002, LNCS 2442, pp. 597-612, Springer-Verag, 2002.

[84] J. Malone-Lee. *Identity-Based Signcryption*. Available at http://eprint.iacr.org/2002/098.

[85] T. Matsumoto, Y. Takashima and H. Imai. *On Seeking Smart Public-key Distribution Systems*. In Transactions of the IECE of Japan, E69, pp. 99-106, 1986.

[86] N. McCullagh and P. S. L. M. Barreto. *A New Two-Party Identity-Based Authenticated Key Agreement*. In proceedings of CT-RSA 2005, LNCS 3376, pp. 262-274, Springer-Verlag, 2005. Also available at http://eprint.iacr.org/2004/122.

[87] A. Menezes, T. Okamoto, and S. Vanstone. *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field*. In IEEE Transaction on Information Theory, 39, pp. 1639-1646, 1993.

[88] A. Menezes, P. C. Van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, 1997. Also available at http://cacr.math.uwaterloo.ca/hac.

[89] D. Nalla and K. C. Reddy. *Identity Based Authenticated Group Key Agreement Protocol*. In proceedings of Indocrypt 2002, LNCS 2551, pp. 215-233, Springer-Verlag, 2002.

[90] D. Nalla. *ID-Based Tripartite Key Agreement with Signature*. Available at http://eprint.iacr.org/2003/144.

[91] D. Nalla and K. C. Reddy. *ID-Based Tripartite Authenticated Key Agreement Protocols from Pairings*. Available at http://eprint.iacr.org/2003/004.

[92] J. Nam, J. Lee, S. Kim and D. Won. *DDH-based Group Key Agreement for Mobile Computing*. Available at http://eprint.iacr.org/2004/127.

[93] J. Nam, S. Kim, S. Kim and D. Won. *Provably-Secure and Communication-Efficient Scheme for Dynamic Group Key Exchange*. Available at http://eprint.iacr.org/2004/115.

[94] J. Nam, S. Kim and D. Won. *Attacks on Bresson-Chevassut-Essiari-Pointcheval's Group Key Agreement Scheme for Low-Power Mobile Devices*. Available at http://eprint.iacr.org/2004/251.

[95] J. Nam, S. Kim, H. Yang and D. Won. *Secure Group Communications over Combined Wired/Wireless Network*. Available at http://eprint.iacr.org/2004/260.

[96] NIST. AES, December 2000. Available at http://www.nist.gov/aes.

[97] E. Okamoto. *Proposal for Identity Based Key Distribution System*. In Electronic Letters, 22, pp. 1283-1284, 1986.

[98] K. G. Paterson. *ID-based Signatures from Pairings on Elliptic Curves*. In Electronic Letters, 38(18), pp. 1025-1026, 2002. Also available at http://eprint.iacr/org/2002/004.

[99] O. Pereira and J.J. Quisquater. *A Security Analysis of the Cliques Protocol Suite*. In Computer Security Foundations Workshop (CSFW 2001), pp. 73-81, IEEE Computer Society Press, 2001.

[100] C. Racoff and D. Simon. *Noninteractive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack*. In proceedings of Crypto 1991, LNCS 576, pp. 433-444, Springer-Verlag, 1991.

[101] R. Sakai and M. Kasahara. *Cryptosystems Based on Pairing over Elliptic Curve*. In SCIS 2003, 8c-1, Jan. 2003. Japan. Also available at http://eprint.iacr.org/2003/054.

[102] R. Sakai, K. Ohgishi and M. Kasahara. *Cryptosystems Based on Pairing*. In SCIS 2000-c20, Okinawa, Japan, January 2000.

[103] C. P. Schnorr. *Security of Blind Discrete Log Signatures Against Interactive Attacks*. In proceedings of ICICS 2001, LNCS 2229, pp. 1-12, Springer-Verlag, 2001.

[104] M. Scott. *Authenticated ID-based Key Exchange and Remote Log-in with Insecure Token and PIN Number*. Available at http://eprint.iacr.org/2002/164.

[105] A. Shamir. *Identity-based Cryptosystems and Signature Schemes*. In proceedings of Crypto 1984, LNCS 196, pp. 47-53, Springer-Verlag, 1984.

[106] K. Shim. *Efficient ID-based Authenticated Key Agreement Protocol Based on the Weil Pairing*. In Electronic Letters, 39(8), pp. 653-654, 2003.

[107] K. Shim. *Cryptanalysis of Al-Riyami-Paterson's Authenticated Three Party Key Agreement Protocols*. Available at http://eprint.iacr.org/2003/122.

[108] K. Shim. *Cryptanalysis of ID-Based Tripartite Authenticated Key Agreement Protocol*. Available at http://eprint.iacr.org/2003/115.

[109] V. Shoup. *On Formal Models for Secure Key Exchange*. In IBM Technical Report RZ 3120, 1999. Also available at http://shoup.net/papers.

[110] J. Silverman. *The Arithmetic of Elliptic Curves*. Springer-Verlag, 1986.

[111] N. P. Smart. *An Identity-based Authenticated Key Agreement Protocol Based on the Weil Pairing.* In Electronic Letters, 38, pp. 630-632, 2002. Also available at http://www.iacr.org/2001/111.

[112] B. Song and K. Kim. *Two-pass Authenticated Key Agreement Protocol with Key Confirmation.* In proceedings of Indocrypt 2000, LNCS 1977, pp. 237-249, Springer-Verlag, 2000.

[113] M. Steiner, G. Tsudik, M. Waidner. *Diffie-Hellman Key Distribution Extended to Group Communication.* In proceedings of ACM CCS 1996, pp. 31-37, ACM Press, 1996.

[114] M. Steiner, G. Tsudik and M. Waidner. *Cliques : A New Approach to Group Key Agreement.* In IEEE Conference on Distributed Computing Systems, May 1998, pp. 380.

[115] H. M. Sun and B. T. Hsieh. *Security Analysis of Shim's Authenticated Key Agreement Protocols from Pairings.* Available at http://eprint.iacr.org/2003/113.

[116] W. Susilo, Y. Mu. *Non-Interactive Deniable Ring Authentication.* In proceedings of ICISC 2003, LNCS 2971, pp. 397-412, Springer-Verlag, 2003.

[117] B. Waters. *Efficient Identity-Based Encryption Without Random Oracles.* In proceedings of Eurocrypt 2005, LNCS 3494, pp. 114-127, Springer-Verlag, 2005.

[118] G. Xie. *Cryptanalysis of Noel McCullagh and Paulo S. L. M. Barreto's Two-party Identity-Based Key Agreement.* Available at http://eprint.iacr.org/2004/308.

[119] G. Xie. *An ID-Based Key Agreement Scheme from Pairing.* Available at http://eprint.iacr.org/2005/093.

[120] F. Zhang, K. Kim. *ID-Based Blind Signature and Ring Signature from Pairings.* In proceedings of Asiacrypt 2002, LNCS 2501, pp. 533-547, Springer-Verlag, 2002.

[121] F. Zhang, R. Safavi-Naini and W. Susilo. *An Efficient Signature Scheme from Bilinear Pairings and its Applications.* In proceedings of PKC 2004, LNCS 2947, pp. 277-290, Springer-Verlag, 2004.

[122] F. Zhang, R. Safavi-Naini and W. Susilo. *Efficient Verifiably Encrypted Signature and Partially Blind Signature from Bilinear Pairings.* In proceedings of Indocrypt 2003, LNCS 2904, pp. 191-204, Springer-Verlag, 2003. Also available at http://www.iacr.org/2004/004.

[123] F. Zhang, R. Safavi-Naini and W. Susilo. *ID-Based Chameleon Hashes from Bilinear Pairings.* Available at http://eprint.iacr.org/2003/208.

[124] F. Zhang, S. Liu and K. Kim. *ID-based One Round Authenticated Tripartite Key Agreement Protocol with Pairings.* Available at http://eprint.iacr.org/2002/122.

Appendix A

Overview of Pairing-Based Cryptographic Protocols

The bilinear pairing such as Weil pairing or Tate pairing on elliptic and hyperelliptic curves have recently been found applications in design of cryptographic protocols. In this survey, we have tried to cover different cryptographic protocols based on bilinear pairings which possess, to the best of our knowledge, proper security proofs in the existing security models. We make no attempt to be complete in the presentation and included it here in the hope that it could serve as a motivation for a beginner. (In fact, this has been used as a reading material by Prof. Ron Rivest of Massachusetts Institute of Technologies (MIT) in his course on "Selected Topics In Cryptography (6.897)".).

A.1 Introduction

The concept of identity-based cryptosystem is due to Shamir [105]. Such a scheme has the property that a user's public key is an easily calculated function of his identity, while a user's private key can be calculated for him by a trusted authority, called private key generator (PKG). The ID-based public key cryptosystem can be an alternative for certificate-based public key infrastructure (PKI), especially when efficient key management and moderate security are required.

Earlier bilinear pairings, namely Weil pairing and Tate pairing of algebraic curves were used in cryptography for the MOV attack [88] using Weil pairing and FR attack [57] using Tate pairing. These attacks reduce the discrete logarithm problem on some elliptic or hyperelliptic curves to the discrete logarithm problem in a finite field. In recent years, bilinear pairings have found positive application in cryptography to construct new cryptographic primitives. The current work is an attempt to survey this field.

Protocols from pairings can be broadly classified into two types:

– Construction of primitives which apparently cannot be constructed using other techniques (*e.g.*

ID-based encryption, non-trivial aggregate signature *etc*).

– Construction of primitives which can be constructed using other techniques, but for which pairings provide improved functionality (*e.g.* Joux's three-party key agreement, threshold scheme, searchable public key encryption *etc*).

Joux [66], in 2000, showed that the Weil pairing can be used in a protocol to construct three-party one-round Diffie-Hellman key agreement. This was one of the breakthroughs in key agreement protocols. After this, Boneh and Franklin [24] presented in Crypto 2001 an ID-based encryption scheme based on properties of bilinear pairings on elliptic curves which is the first fully functional, efficient and provably secure identity-based encryption scheme. In Asiacrypt 2001, Boneh, Lynn and Shacham proposed a basic signature scheme using pairing, the BLS [26] scheme, that has the shortest length among signature schemes in classical cryptography. Subsequently numerous cryptographic schemes based on BLS signature scheme were proposed.

Apart from the three fundamental cryptographic primitives: encryption, signature and key agreement, there are protocol designs for signcryption, threshold decryption, key sharing, identification scheme, chameleon hashes *etc*. We provide the following classification of the protocols:

1. Encryption: Encryption schemes are used for the purpose of achieving privacy and confidentiality. In recent years, pairings made ID-based public key encryption feasible. In identity-based public key encryption, the public key distribution problem is eliminated by making each user's public key derivable from some known aspect of his identity, such as his email address. When Alice wants to send a message to Bob, she simply encrypts her message using Bob's public key which she merely derives from Bob's identifying information. Bob, on receiving the encrypted message, obtains his private key from a third party called a Private Key Generator (PKG) after authenticating himself to PKG and decrypts the message. The private key that PKG generates on Bob's query is a function of its master key and Bob's identity.

2. Signature:

   – *Short Signature*: These are required in environments with space and bandwidth constraints. When a human is asked to manually key in the signature, the shortest possible signature is desirable.

   – *Blind Signature*: Blind signatures play a central role in digital cash schemes. A user can obtain from a bank a digital coin using a blind signature protocol. The coin is essentially a token properly signed by the bank. The blind signature protocols enable a user to obtain a signature from a signer so that the signer does not learn any information about the message it signed and so that the user cannot obtain more than one valid signature after one interaction with the signer. The concept of blind signatures provides anonymity of users in applications such as electronic voting, electronic payment systems *etc*.

   – *Multisignature*: Multisignature scheme allows any subgroup of a group of users to jointly sign a message such that a verifier is convinced that each member of the subgroup participated in signing. The goal of multisignature is to prove that each member of the

stated subgroup signed the message and the size of this subgroup can be arbitrary. It is up to a particular application to decide what subgroup is required to sign a message. A verifier might reject a multi-signature not because it's invalid, but because the verifier in not satisfied with the subgroup which signed the message. Multi-signatures can be applied to provide efficient batch verification of several signatures of the same message under different public keys.

- *Aggregate Signature*: Consider n users $U = \{1, 2, \ldots, n\}$. Each user $i \in U$ has a public-private key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$. User i signs message M_i and outputs signature σ_i. A public aggregation algorithm outputs a compressed short signature σ on input all of $\sigma_1, \sigma_2, \ldots, \sigma_n$. This aggregation of n signatures can be done by anyone. Additionally, there is an aggregate verification algorithm that takes $\mathsf{PK}_1, \mathsf{PK}_2, \ldots, \mathsf{PK}_n$, $M_1, M_2, \ldots, M_n$, and σ as input and decides whether the aggregate signature σ is valid. Aggregate signature scheme has use in the secure border gateway protocol for compressing the list of signatures on distinct messages issued by distinct parties.
- *Verifiably Encryption Signature*: These signatures enable user Alice to give Bob a signature on a message M encrypted using a third party's public key and Bob to verify off-line that the encrypted signature is valid. Bob can verify that Alice has signed the message, but cannot deduce any information about her signature. To enable fair exchange, verifiably encrypted signatures are used in optimistic contract signing protocols.
- *Ring Signature*: Consider a set of n users $U = \{1, 2, \ldots, n\}$. Each user $i \in U$ has a public-private key pair $(\mathsf{PK}_i, \mathsf{SK}_i)$. A ring signature on U is a signature that is constructed using all the public keys of the users in U, and a single private key of any user in U. A ring signature protects the anonymity of a signer since the verifier knows that the signature is from a member of the ring U, but does not know exactly who the signer is. There is also no way to revoke the anonymity of the signer. Ring signatures have applications in authenticated (yet repudiable) communication and leaking secrets.
- *Group Signature*: Group signatures permits any member of a group to sign on behalf of the group. Anyone can verify the signature with a group public key while no one can know the identity of the signer except the group manager. Group signature provides anonymity of users with the property that group manager can identify the signer. In group signature, it is computationally hard to decide whether two different signatures were issued by the same member.
- *Proxy Signature*: A proxy signature allows an entity, called the delegator to delegate its signing rights to another entity, called a proxy signer. The proxy signer signs messages on behalf of the delegator, in case of say, temporal absence, lack of time or computational power, *etc.* Proxy signatures have found numerous practical applications where delegation of rights is quite common, particularly in distributed systems, Grid Computing, mobile agent applications, distributed shared object systems and mobile communications [19].
- *Unique Signature*: Unique signature schemes are secure signature schemes where the signature is hard-to-compute function of the public key and the message. Unique signature

schemes, also known as invariant signature schemes, are desirable in cryptography and have an important application to construct verifiable random functions (VRFs). VRFs are objects that combine the properties of pseudo-random functions with the verifiability property and can be viewed as a commitment to an arbitrary number of bits.

3. Key Agreement: Key agreement is required in situations where two or more parties want to communicate securely among themselves. The situation where three or more parties share a secret key is often called conference keying. In this situation, the parties can securely send and receive message from each other. An adversary not having access to the secret key will not be able to decrypt the message. A detailed discussion on key agreement protocols is provided in Chapter 3.

4. Threshold: Threshold cryptography approach is useful to remove single point failure. When the centralization of the power is a concern, threshold decryption can be used in particular. In the (t, n)-threshold scheme, $t \leq n$, there are n users. A secret information is distributed among these n-users. Any subset of more than t users are allowed to reconstruct the secret. The computation is performed preserving security even in the presence of an active adversary that can corrupt up to t users.

5. Miscellaneous:

   – *Chameleon Hash*: Chameleon hashing is basically non-interactive commitment scheme. A chameleon hash function is associated with a pair of public-private keys. Anyone who knows the public key, can compute the associated hash function. Without the knowledge of associated trapdoor, the chameleon hash function is collision resistant. However, the trapdoor information holder can easily find collisions for every given input. Chameleon hashes have applications in constructing chameleon signatures. The recipient can verify that the signature of a certain message m is valid, but cannot prove others that the signer actually signed m and not another message. These are closely related to undeniable signature [40].

   – *Signcryption*: A signcryption scheme is a scheme that provides private and authenticated delivery of messages between two parties in a more efficient manner than a straightforward composition of an encryption scheme with a signature scheme. It combines the functionality of signature and encryption. The idea of signcryption scheme is to perform encryption and signature in a single logical step in order to obtain confidentiality, integrity, authentication and non-repudiation more efficiently than the sign-then-encrypt approach.

   – *Identification*: Identification scheme is another important and useful cryptographic tool where a prover $\mathcal{P}$ interacts with a verifier $\mathcal{V}$ to convince him of his identity. Only $\mathcal{P}$ knows the secret value corresponding to his public one, and this secret information permits to convince $\mathcal{V}$ of his identity.

In this Chapter, we have tried to survey different cryptographic primitives and include only those schemes which have, to the best of our knowledge, concrete security proofs in the existing

adversarial models. Barreto's pairing based crypto lounge [7] is an excellent compilation of existing work on pairing based cryptography. This survey does not consider algebraic theory of pairings nor algorithms to compute them.

A.2 Encryption Schemes

In identity-based public key encryption, the public key distribution problem is eliminated by making each user's public key derivable from some known aspect of his identity, such as his email address. When Alice wants to send a message to Bob, she simply encrypts her message using Bob's public key which she derives from Bob's identifying information. Bob, after receiving the encrypted message, obtains his private key from a third party called a Private Key Generator (PKG) after authenticating himself to PKG and can then decrypt the message. The private key that PKG generates on Bob's query is a function of its master key and Bob's identity.

Shamir [105] introduced this concept of identity-based cryptosystem. The first ID-based encryption was proposed by Boneh and Franklin [24] in 2001 that uses bilinear pairing.

The advantage of ID-based encryption are compelling. It makes maintaining authenticated public key directories unnecessary. Instead, a directory for authenticated public parameters of PKG's is required which is less burdensome than maintaining a public key directory since there are substantially fewer PKGs than total users. In particular, if everyone uses a single PKG, then everyone in the system can communicate securely and users need not perform on-line lookup of public keys or public parameters.

Some disadvantages of ID-based system are : (1) the PKG knows Bob's private key, *i.e.* key escrow is inherent in the system which for some applications may be a serious problem, (2) Bob has to authenticate himself to its PKG in the same way as he would authenticate himself to a certifying authority (CA), (3) Bob's PKG requires a secure channel to send Bob his private key, (4) Bob has to publish his PKG's public parameters and Alice must obtain these parameters before sending an encrypted message to Bob.

Let G_1, G_2, e be the same as in Definition 2.2.11 of cryptographic bilinear pairings. P is a generator of the additive group G_1 of order q (a large prime), G_2 is a multiplicative group of same order q and e is the bilinear map from $G_1 \times G_1 \rightarrow G_2$. We use these notations throughout this chapter.

A.2.1 ID-Based Encryption

(Boneh, Franklin, [24], 2001)

- Protocol Description :

  Setup : Choose $s \in_R Z_q^*$ and set $P_{pub} = sP$. Choose cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow G_1^*$ and $H_2 : G_2 \rightarrow \{0,1\}^n$, n is the bit length of messages. The master key is

s and the global public key is P_{pub}.

Extract : Given a public identity $\mathsf{ID} \in \{0,1\}^*$, compute the public key $Q_{\mathsf{ID}} = H_1(\mathsf{ID}) \in G_1$ and the private key $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. The computation $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ maps an arbitrary string to a point of the group G_1. This operation is called Map-to-point and is more expensive than computation of usual message digest.

Encrypt : Choose a random $r \in Z_q^*$, set the cipher text for the message M to be $C = \langle rP, M \oplus H_2(g_{\mathsf{ID}}^r) \rangle$ where $g_{\mathsf{ID}} = e(Q_{\mathsf{ID}}, P_{pub})$.

Decrypt : Given $C = \langle U, V \rangle$, compute $V \oplus H_2(e(S_{\mathsf{ID}}, U))$.

- Assumption :
  BDH problem is hard.

- Security :
  This is the basic scheme. Security against adaptive chosen cipher text attack in the random oracle model under the BDH assumption is obtained after the Fujisaki-Okamoto [58] transformation.

- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Encrypt : 1 Map-to-point hash operation; 1 scalar multiplication in G_1; 1 hash function (H_2) evaluation; 1 XOR operation; 1 pairing computation; 1 group exponent in G_2.

  Decrypt : 1 hash function (H_2) evaluation; 1 XOR operation; 1 pairing computation.

A.2.2 Searchable Public Key Encryption

(Boneh, Crescenzo, Ostrovsky, Persiano, [23], 2003)

Suppose Alice wishes to read her email on a number of devices : laptop, desktop, pager, etc. Alice's mail gateway is supposed to route email to the appropriate device based on the keywords in the email. Suppose Bob sends an email with keyword "urgent". The gateway routes the email to Alice's pager, after testing whether the email contains this keyword "urgent" without learning anything else about the mail. This mechanism is referred to as *Searchable Public Key Encryption* (SPKE).

To send a message M with keywords $W_1, \ldots, W_n$, Bob sends

$$E_{A_{pub}}(M) | \mathsf{SPKE}(A_{pub}, W_1) | \ldots | \mathsf{SPKE}(A_{pub}, W_n)$$

where $E_{A_{pub}}(M)$ is the encryption of M using Alice's public key A_{pub}. The point of searchable encryption is that given $\mathsf{SPKE}(A_{pub}, W')$ and a certain trapdoor T_W (that is given to the gateway by Alice), the gateway can test whether $W = W'$. If $W \neq W'$ the gateway learns nothing more

about W'.

A SPKE scheme using bilinear map :

- Protocol Description :

  KeyGen : Choose $s \in_R Z_q^*$ and set $P_{pub} = sP$. The secret key is s and the public key is P_{pub}. Let K be the set of all keywords and $H_1 : K \rightarrow G_1$, $H_2 : G_2 \rightarrow Z_q^*$ be two hash functions.

  SPKE : Given a keyword W and the public key P_{pub}, choose a random $r \in Z_q^*$ and output $\langle rP, H_2(e(H_1(W), P_{pub})^r)\rangle$.

  Trapdoor : Given a keyword W and the secret key s, output $T_W = sH_1(W)$.

  Test : Given Trapdoor T_W, a SPKE $S = \langle U, V\rangle$ and the public key P_{pub}, test if $V = H_2(e(T_W, U))$. If true, output yes, else output no.

- Assumption :
  BDH problem is hard.

- Security :
  Semantically secure against a chosen keyword attack in the random oracle model assuming BDH problem is intractable.

- Efficiency :

  KeyGen : 1 scalar multiplication in G_1.

  SPKE : 1 Map-to-point hash operation; 1 scalar multiplication in G_1; 1 hash function (H_2) evaluation; 1 pairing computation; 1 group exponent in G_2.

  Trapdoor : 1 scalar multiplication in G_1.

  Test : 1 pairing computation; 1 hash function (H_2) evaluation.

A.2.3 Hierarchical ID-Based Encryption (HIDE)

(Gentry, Silverberg, [60], 2002)

Although having a single private key generator (PKG) would completely eliminate on-line lookup, it is undesirable for a large network because the PKG has a burdensome job. Not only is private key generation computationally expensive, but also the PKG must verify proofs of identity and must establish secure channels to transmit private keys. HIDE allows a root PKG to distribute the workload by delegating private key generation and identity authentication to lower-level PKGs. In a HIDE scheme, a root PKG need only generate private keys for domain-level PKGs, who in turn generate private keys for users in their domains in the next level. Authentication and private key transmission can be done locally. To encrypt a message to Bob, Alice only needs to obtain the

public parameters of Bob's parent PKG (and Bob's identifying information); there are no "lower-level parameters". HIDE has the advantage of damage control : disclosure of a domain PKG's secret does not compromise the secrets of higher-level PKGs.

- Protocol Description : (BasicHIDE)
  The entities in the tree (other than the root) are the users of the tree. Let Level_i be the set of entities at level i, where $\mathsf{Level}_0 = \{\mathsf{Root\ PKG}\}$.

  Root Setup : The root PKG chooses an arbitrary generator $P_0 \in G_1$, picks a random $s_0 \in Z_q^*$ and sets $Q_0 = s_0 P_0$. Let $H_1 : \{0,1\}^* \rightarrow G_1$ and $H_2 : G_2 \rightarrow \{0,1\}^n$ be two cryptographic hash functions. The message space is $\mathcal{M} = \{0,1\}^n$. The cipher-text space is $\mathcal{C} = G_1^t \times \{0,1\}^n$ where t is the level of the recipient. The root PKG's secret is $s_0 \in Z_q^*$ and global public key is (P_0, Q_0).

  Lower-level Setup : Entity $E_t \in \mathsf{Level}_t$ picks a random $s_t \in Z_q^*$ which it keeps secret.

  Extract : Let E_t be an entity in Level_t with ID-tuple $(\mathsf{ID}_1, \ldots, \mathsf{ID}_t)$, where $(\mathsf{ID}_1, \ldots, \mathsf{ID}_i)$ for $1 \leq i \leq t$ is the ID-tuple of E_t's ancestor at Level_i. Set S_0 to be the identity element of G_1.
  Then E_t's $(t \geq 1)$ parent :
  1. computes $P_t = H_1(\mathsf{ID}_1, \ldots, \mathsf{ID}_t) \in G_1$,
  2. sets E_t's secret point S_t to be $S_{t-1} + s_{t-1} P_t = \sum_{i=1}^{t} s_{i-1} P_i$ and
  3. also gives E_t the values $Q_i = s_i P_0$ for $1 \leq i \leq t-1$.

  Encrypt : To encrypt $M \in \mathcal{M}$ with the ID-tuple $(\mathsf{ID}_1, \ldots, \mathsf{ID}_t)$, do the following :
  1. compute $P_i = H_1(\mathsf{ID}_1, \ldots, \mathsf{ID}_i) \in G_1$ for $1 \leq i \leq t$,
  2. choose a random $r \in Z_q^*$ and
  3. set the cipher text to be $C = \langle rP_0, rP_2, \ldots, rP_t, M \oplus H_2(g^r) \rangle$ where $g = e(Q_0, P_1) \in G_2$.

  Decrypt : Let $C = \langle U_0, U_2, \ldots, U_t, V \rangle \in \mathcal{C}$ be the cipher text encrypted using the ID-tuple $(\mathsf{ID}_1, \ldots, \mathsf{ID}_t)$. To decrypt C, E_t computes :

  $$V \oplus H_2 \left(\frac{e(U_0, S_t)}{\prod_{i=2}^{t} e(Q_{i-1}, U_i)} \right) = M.$$

  Note : The scheme is derived from Boneh-Franklin [24] scheme. An interesting fact is that lower-level PKGs need not always use the same s_t for each private key extraction. Rather, s_t could be generated randomly for each of the PKG's children. Another fact is that H_1 can be chosen to be an iterated hash function, for example, P_i may be computed as $H_1(P_{i-1}, \mathsf{ID}_i)$ rather than $H_1(\mathsf{ID}_1, \ldots, \mathsf{ID}_i)$.

- Assumption :
  BDH problem is hard.

- Security :
  Chosen cipher text security of this basic scheme is obtained by using Fujisaki-Okamoto [58] padding in the random oracle model under the assumption that BDH problem is hard.

- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 Map-to-point hash operation; 2 scalar multiplications in G_1; 1 addition in G_1.

  Encrypt : For an identity at level t, t scalar multiplications in G_1; 1 Map-to-point hash operation; 1 hash function (H_2) evaluation; 1 group exponent in G_2; 1 XOR operation; 1 pairing computation.

  Decrypt : For an identity at level t, t pairing computations; 1 hash function (H_2) evaluation; 1 XOR operation. The bit-length of the cipher text and the complexity of decryption grow linearly with the level of the message recipient.

A.2.4 Dual-HIDE : Dual-Hierarchical-Identity-Based Encryption

(Gentry, Silverberg [60], 2002)

- Protocol Description :
  Suppose two users, Alice and Bob, have the ID-tuples $(\mathsf{ID}_{y_1}, \ldots, \mathsf{ID}_{y_l}, \ldots, \mathsf{ID}_{y_m})$ and $(\mathsf{ID}_{z_1}, \ldots, \mathsf{ID}_{z_l}, \ldots, \mathsf{ID}_{z_n})$ respectively, where $(\mathsf{ID}_{y_1}, \ldots, \mathsf{ID}_{y_l}) = (\mathsf{ID}_{z_1}, \ldots, \mathsf{ID}_{z_l})$. In other words, Alice is in Level_m, Bob is in Level_n and their common ancestor upto Level_l are same. Alice may use Dual-HIDE to encrypt a message to Bob as follows :

  Encrypt : To encrypt $M \in \mathcal{M}$, Alice :

  1. computes $P_{z_i} = H_1(\mathsf{ID}_{z_1}, \ldots, \mathsf{ID}_{z_i}) \in G_1$ for $l+1 \leq i \leq n$,
  2. chooses a random $r \in Z_q^*$ and
  3. sets the cipher text to be $C = \langle rP_0, rP_{z_{l+1}}, \ldots, rP_{z_n}, M \oplus H_2(g_{y_l}^r) \rangle$ where

  $$g_{y_l} = \frac{e(P_0, S_y)}{\prod_{i=l+1}^{m} e(Q_{y_{(i-1)}}, P_{y_i})} = e(P_0, S_{y_l}).$$

  S_y is Alice's secret point, S_{y_l} is the secret point of Alice's and Bob's common ancestor at level l and $Q_{y_i} = s_{y_i} P_0$ where s_{y_i} is the secret number chosen by Alice's ancestor at level i.

  Decrypt : Let $C = \langle U_0, U_{l+1}, \ldots, U_n, V \rangle$ be the cipher text. To decrypt C, Bob computes :

  $$V \oplus H_2 \left(\frac{e(U_0, S_z)}{\prod_{i=l+1}^{n} e(Q_{z_{(i-1)}}, U_i)} \right) = M.$$

- Assumption :
  BDH problem is hard.

- Security :
  Secure in the random oracle model assuming the hardness of BDH problem.

- Efficiency :

  Encrypt : 1 Map-to-point hash operation; $(n - l + 1)$ scalar multiplications in G_1; 1 hash function (H_2) evaluation; 1 XOR operation; $(m - l + 1)$ pairing computation; 1 group exponent in G_2.

  Decrypt : $(n-l+1)$ pairing computation; 1 hash function (H_2) evaluation; 1 XOR operation. If Alice and Bob have a common ancestor below the root PKG, then the cipher text is shorter than for normal HIDE. Further, using Dual HIDE, the encrypter Alice computes $(m - l + 1)$ pairings while the decrypter Bob computes $(n - l + 1)$ pairings. In the non-dual HIDE scheme, the encrypter computes one pairing while the decrypter computes n pairings. Thus when $m < (2l - 1)$, the total work is less with Dual-HIDE than with non-dual HIDE. Dual-HIDE also makes domain-specific broadcast encryption possible. Furthermore, one can restrict key escrow using Dual-HIDE.

A.2.5 ID-Based Encryption Without Random Oracle

(Boneh, Boyen [21], 2004)

- Protocol Description :

  Setup : The public keys (ID) are assumed to be elements of Z_q^* and messages are elements of G_2. Select random elements $x, y \in Z_q^*$ and set $U = xP, V = yP$. The public parameters are (U, V) and the master key is (x, y).

  Extract : Given a public key $\mathsf{ID} \in Z_q^*$, choose a random $r \in Z_q^*$ and compute $K = \frac{1}{\mathsf{ID}+x+ry}P \in G_1$. Output the private key $S_{\mathsf{ID}} = (r, K)$.

  Encrypt : To encrypt a message $M \in G_1$ under public key $\mathsf{ID} \in Z_q^*$, pick a random $s \in Z_q^*$ and output the cipher text $C = \langle s(\mathsf{ID})P + sU, sV, e(P, P)^s \; M\rangle$.

  Decrypt : To decrypt a cipher text $C = \langle X, Y, Z\rangle$ using the private key $S_{\mathsf{ID}} = (r, K)$, output $Z/e(X + rY, K)$.

- Assumption :
  q-DBDHI problem is hard.

- Security :
  Secure against selective-ID adaptive chosen cipher text attack without random oracles under q-DBDHI assumption.

- Efficiency :

  Setup : 2 scalar multiplications.

Extract : 1 inversion in Z_q^*; 1 scalar multiplication in G_1.

Encrypt : 4 scalar multiplications in G_1; 1 group exponent in G_1; 1 multiplication in G_2. Note that $e(P,P)$ can be precomputed once and for all so that encryption requires no pairing computation.

Decrypt : 1 scalar multiplication in G_1; 1 addition in G_1; 1 inversion in G_2.

A.2.6 Hierarchical ID-Based Encryption (HIBE) Without Random Oracle

(Boneh, Boyen [21], 2004)

- Protocol Description :

  Setup : The public keys (ID) of depth l are assumed to be vectors of elements in Z_q^l. The j-th component for an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_l) \in Z_q^l$ corresponds to the identity at level j. The system parameters for an HIBE of maximum depth l is generated as follows: Choose a random $\alpha \in Z_q^*$ and set $P_1 = \alpha P \in G_1$. Choose random elements $h_1, \ldots h_l \in G_1$ and another generator $P_2 \in G_1^*$. The public parameters are $(P, P_1, P_2, h_1, \ldots, h_l)$ and master key is αP_2. For $j = 1, \ldots, l$, define $F_j(x) = xP_1 + h_j$. The messages are assumed to be elements of G_2.

  Extract: For an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_j) \in Z_q^j$ of depth $j \leq l$, pick random $r_1, \ldots, r_j \in Z_q$ and set the private key $S_{\mathsf{ID}} = (\alpha P_2 + \sum_{k=1}^{j} r_k F_k(\mathsf{ID}_k), r_1 P, \ldots, r_j P)$. Note that, if at depth $(j-1)$ the private key for identity $\mathsf{ID}_{|j-1} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_{j-1}) \in Z_q^{j-1}$ is $S_{\mathsf{ID}|j-1} = (d_0, \ldots, d_{j-1})$, then the private key S_{ID} for ID is generated by choosing randomly $r_j \in Z_q$ and setting $S_{\mathsf{ID}} = (d_0 + r_j F_j(\mathsf{ID}_j), d_1, \ldots, d_{j-1}, r_j P)$.

  Encrypt : To encrypt a message $M \in G_2$ under the public key $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_j) \in Z_q^j$, pick randomly $s \in Z_q^*$ and output $C = \langle e(P_1, P_2)^s \; M, sP, sF_1(\mathsf{ID}_1), \ldots, sF_j(\mathsf{ID}_j) \rangle$.

  Decrypt : Consider an identity $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_j)$. To decrypt a cipher text $C = \langle A, B, C_1, \ldots, C_j \rangle$ using the private key $S_{\mathsf{ID}} = (d_0, d_1, \ldots, d_j)$, output

  $$A \prod_{k=1}^{j} e(C_j, d_j) / e(B, d_0) = M.$$

- Assumption :
  DBDH problem is hard.

- Security :
  Secure against selective-ID adaptive chosen cipher text attack without random oracles under DBDH assumption.

- Efficiency :

  Setup : 2 scalar multiplications in G_1.

Extract : For an identity at depth j, $(2j+1)$ scalar multiplications in G_1; $(j+1)$ additions in G_1.

Encrypt : 1 group exponent in G_2; 1 multiplications in G_2; $(j-1)$ scalar multiplications in G_1. Note that encryption does not require any pairing computation as $e(P_1, P_2)$ can be precomputed once and included in the system parameters.

Decrypt : For an identity at depth j, j multiplications in G_2; j pairing computations; 1 inversion in G_2.

A.2.7 Another ID-Based Encryption Without Random Oracle

(Waters [117], 2005)

- **Protocol Description** :

  Setup : Choose a secret $\alpha \in Z_q^*$ at random and set $P_1 = \alpha P$. Choose P_2 randomly in G_1. Additionally, choose a random value $Q' \in G_1$ and a random n-length vector $U = (Q_i)$, whose elements are chosen at random from G_1. The published public parameters are P, P_1, P_2, Q' and U and the master secret is αP_2.

  Extract : Let $\mathsf{ID} \in \{0,1\}^n$ be an n-bit string representing a public identity. Let v_i be the i-th bit of ID and $\mathcal{V} \subseteq \{1, \ldots, n\}$ be the set of all i for which $v_i = 1$. Select a random $r \in Z_q^*$. Output the private key

  $$S_{\mathsf{ID}} = (\alpha P_2 + r(Q' + \sum_{i \in \mathcal{V}} Q_i), rP).$$

  Encrypt : To encrypt a message $M \in G_2$ under public identity ID, pick a random $t \in Z_q^*$ and output the cipher test

  $$C = (e(P_1, P_2)^t M, tP, t(Q' + \sum_{i \in \mathcal{V}} Q_i)).$$

  Decrypt : To decrypt a cipher text $C = (C_1, C_2, C_3)$ using the private key $S_{\mathsf{ID}} = (A, B)$, output

  $$C_1 \frac{e(A, C_3)}{e(B, C_2)} = M.$$

- **Assumption :**
  DBDH problem is hard.

- **Security :**
  Secure against adaptive chosen ciphertext attacks without the random oracle model.

- **Efficiency :**
  If the value $e(P_1, P_2)$ is cached then encryption requires on average $\frac{n}{2}$ (and at most n) group

operations in G_1, 2 exponentiations in G_1, 1 exponentiation in G_2 and 1 group operation in G_2. Decryption requires 2 bilinear map computation, 1 group operation in G_2 and 1 inversion in G_2.

Note : Boneh and Boyen [21] presented an identity-based encryption scheme that is provably secure in the selective-ID model without random oracle. In [22], Boneh and Boyen describe a scheme that is fully secure without random oracles, but is too inefficient to be of practical use. The construction of Waters [117] presented here is a modified version of Boneh and Boyen's scheme in [21] that makes the identity-based scheme fully secure and is efficient as compared to the scheme in [22].

A.3 Signature Schemes

Digital signatures are one of the most important cryptographic primitives. In traditional public key signature algorithms, the binding between the public key and the identity of the signer is obtained via a digital certificate. Shamir [105] first noticed that it would be more efficient if there was no need for such bindings, in that case given the user's identity, the public key could be easily derived using some public deterministic algorithm. This makes efficient ID-based signature schemes desirable. In ID-based signature schemes, verification function is easily obtained from the identity, possibly the same key and the same underlying computation primitives can be used. Boneh, Lynn, Shacham [26] proposed a pairing based short signature scheme in 2001. This was followed by a large number of pairing based signature schemes for different applications.

A.3.1 BLS Short Signature

(Boneh, Lynn, Shacham, [26], 2001)
Short signatures are needed in environments with space and bandwidth constraints. For example, when a human is asked to type in a digital signature the shortest possible signatures are desired. Two most frequently used signature schemes are RSA and DSA. If one uses 1024 bit modulus, RSA signatures are 1024 bit long and standard DSA or ECDSA (elliptic curve DSA) signatures are 320 bit long. These signatures are too long to be keyed. The following signature scheme provides short signature of length approximately 160 bits with a level of security similar to 320 bit DSA signatures.

- Protocol Description :

  KeyGen : Let $H : \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function. The secret key is $x \in_R Z_q^*$ and the public key is $P_{pub} = xP$ for a signer.

  Sign : Given secret key x and a message $m \in \{0,1\}^*$, compute the signature $\sigma = xH(m)$.

  Verify : Given public key $P_{pub} = xP$, a message m and a signature σ, verify $e(P, \sigma) = e(P_{pub}, H(m))$.

- Assumption :
  Existence of GDH group.
- Security :
  Secure against existential forgery under adaptive chosen message attack in the random oracle model assuming CDH problem is hard in G_1.
- Efficiency :

  KeyGen : 1 scalar multiplication in G_1.

  Sign : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Verify : 1 Map-to-point hash operation; 2 pairing computations.

A.3.2 Blind Signature

(Boldyreva [18], 2003)

Blind signatures are the basic tools of digital cash schemes. The goal of a blind signature protocol is to enable a user to obtain a signature from a signer so that the signer does not learn any information about the message it signed and so that the user can not obtain more than one valid signature after one interaction with the signer.

- Protocol Description :

  KeyGen : Let $H : \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function. The secret key is $x \in_R Z_q^*$ and the public key is $P_{pub} = xP$ for a signer.

  Blind Signature Issuing Protocol : Given secret key x and a message $m \in \{0,1\}^*$,

  (Blinding) The user chooses randomly $r \in Z_q^*$, computes $M' = rH(m)$ and sends M' to signer.

  (Signing) The signer computes $\sigma' = xM'$ and sends back σ' to the user.

  (Unblinding) The user then computes the signature $\sigma = r^{-1}\sigma'$ and outputs (m, σ).

  Verify : Given public key P_{pub}, a message m and a signature σ, verify $e(P_{pub}, H(m)) = e(P, \sigma)$.
- Assumption :
  Chosen-target CDH problem is hard.
- Security :
  Secure against one more forgery under chosen message attack assuming the hardness of chosen-target CDH problem.
- Efficiency :

KeyGen : 1 scalar multiplication in G_1.

Blind Signature Issuing Protocol : 1 Map-to-point hash operation; 3 scalar multiplications in G_1.

Verify : 2 pairing computations; 1 Map-to-point hash operation.

A.3.3 Multi signature

(Boldyreva, [18], 2003)

A multi-signature scheme allows any subgroup of a group of users to jointly sign a document such that a verifier is convinced that each member of the subgroup participated in signing.

- Protocol Description :

  KeyGen : Let $H; \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function. Consider a set U of n users. The secret key is $x_i \in_R Z_q^*$ and the public key is $P_{pub_i} = x_i P$, for user $u_i \in U, 1 \leq i \leq n$.

  Multi-signature Creation : Any user $u_i \in U$ with secret key x_i that wishes to participate in signing a message $m \in \{0,1\}^*$, computes $\sigma_i = x_i H(m)$ and sends it to a designated signer D (which can be implemented by any user). Let $L = \{u_{i_1}, \ldots u_{i_l}\} \subseteq U$ be a subset of users contributed to the signing. After getting all the σ_j for $j \in J = \{i_1, \ldots, i_l\}$, D computes the multi-signature $\sigma = \sum_{j \in J} \sigma_j$ and outputs (m, L, σ).

  Multi-signature Verification : Given $T = (m, L, \sigma)$ and the list of public keys of the users in L: $P_{pub_j} = x_j P, j \in J = \{i_1, \ldots, i_l\}$, the verifier computes $P_{pub_L} = \sum_{j \in J} P_{pub_j} = \sum_{j \in J} x_j P$ and verifies $e(P, \sigma) = e(P_{pub_L}, H(m))$.

- Assumption :
  Existence of GDH group.

- Security :
  Secure against existential forgery under chosen message attack in the random oracle model under the assumption that the CDH problem is hard in G_1.

- Efficiency :

  KeyGen : n scalar multiplications in G_1.

  Multi-signature Creation : If $l \leq n$ users are participating in signing, then 1 Map-to-point hash operation; l scalar multiplications in G_1; $(l-1)$ additions in G_1.

  Multi-signature Verification : If number of users in the list L is l, then $(l-1)$ additions in G_1; 2 pairing computations.

A.3.4 Aggregate Signature

(Boneh, Gentry, Lynn, Shacham [25], 2003)

An aggregate signature scheme is a digital signature that supports aggregation : Given n signatures on n distinct messages m_i from n distinct users i, $1 \leq i \leq n$, it is possible to aggregate all these signatures into a single short signature. This single signature and the n original messages $m_i, 1 \leq i \leq n$ will convince the verifier that user i indeed signed message m_i , $1 \leq i \leq n$.

- Protocol Description :

  KeyGen : Consider the Co-GDH setup. Let U be a set of n users and $H : \{0,1\}^* \rightarrow G_2$ be a Map-to-point hash function. The secret key is $x_i \in_R Z_q^*$ and the public key is $P_{pub_i} = x_i P_1$ for user $u_i \in U, 1 \leq i \leq n$.

  Aggregation : User $u_i \in U$ signs message $m_i \in \{0,1\}^*$ to generate BLS signature $\sigma_i = x_i H(m_i)$, $1 \leq i \leq n$. The messages m_i must be all distinct. The aggregate signature is $\sigma = (\sigma_1 + \sigma_2 + \cdots + \sigma_n) \in G_2$.

  Aggregate verification : Given public keys P_{pub_i}, distinct messages $m_i, 1 \leq i \leq n$ and an aggregate signature σ, verify $e(P_1, \sigma) = \prod_{i=1}^{n} e(P_{pub_i}, H(m_i))$.

- Assumption :
  Existence of Co-GDH group and a bilinear map.

- Security :
  Secure against existential forgery in the aggregate chosen key model assuming that the Co-CDH problem is hard in (G_1, G_2).

- Efficiency :

  KeyGen : n scalar multiplications in G_1.

  Aggregation : n Map-to-point hash operations; n scalar multiplications in G_2; $(n-1)$ additions in G_2.

  Aggregate verification : n Map-to-point hash operations; $(n+1)$ pairing computations.

A.3.5 Bilinear Verifiably Encrypted Signature

(Boneh, Gentry, Lynn, Shacham [25], 2003)

When Alice wants to sign a message for Bob but does not want Bob to possess her signature on the message immediately, Alice encrypts her signature using the public key of a trusted third party (adjudicator), and sends the result to Bob along with a proof that she has given him a valid encryption of her signature. Bob can verify that Alice has signed the message but can not deduce

any information about her signature. Later in the protocol, Bob can either obtain the signature from Alice or resort to the adjudicator who can reveal Alice's signature.

- Protocol Description :

  KeyGen : Consider the Co-GDH setup. Let $H : \{0,1\}^* \rightarrow G_2$ be a Map-to-point hash function. Choose $x, x' \in_R Z_q^*$ and set $P_{pub} = xP_1, P'_{pub} = x'P_1$. The private/public key pair for signer is (x, P_{pub}) and that of the adjudicator is (x', P'_{pub}).

  Sign, Verify : For a message $m \in \{0,1\}^*$, the signature of a signer with private key x is $\sigma = xH(m) \in G_2$ and the verification is $e(P_1, \sigma) = e(P_{pub}, H(m))$.

  Verifiably Encrypted Signature Creation : Given a secret key $x \in Z_q^*$, a message $m \in \{0,1\}^*$ and an adjudicator's public key $P'_{pub} \in G_1$, compute $h = H(m) \in G_2$ and $\sigma = xh$. Select a random $r \in Z_q^*$ and set $\mu = r\psi(P_1)$ and $\sigma' = r\psi(P'_{pub})$. Aggregate σ, σ' as $w = (\sigma + \sigma') \in G_2$ and output the pair (w, μ).

  Verifiably Encrypted Signature Verification : Given a public key P_{pub}, a message m, an adjudicator's public key P'_{pub} and a verifiably encrypted signature (w, μ), set $h = H(m)$; accept if $e(P_1, w) = e(P_{pub}, h)\ e(P'_{pub}, \mu)$ holds.

  Adjudication : Given an adjudicator's public key P'_{pub} and corresponding private key $x' \in_R Z_q^*$, a public key P_{pub} and a verifiably encrypted signature (w, μ) on some message m, ensure the verifiably encrypted signature is valid; then compute $\sigma = w - x'\mu$. (Before giving the signature, the adjudicator must perform the validity test to prevent a malicious user from tricking him into signing arbitrary messages under his adjudication key). No involvement of adjudicator during generation of encrypted signature or its verification. Adjudicator involves only during signature revelation phase.

- Assumption :
  Existence of Co-GDH group and a bilinear map.

- Security :
  Secure against existential forgery and aggregate extraction assuming that Co-GDH [26] signature scheme is secure against existential forgery and extraction respectively. Co-GDH signature scheme is in fact the BLS signature scheme in Co-GDH setup.

- Efficiency :

  KeyGen : 2 scalar multiplications in G_1.

  Sign : 1 Map-to-point hash operation; 1 scalar multiplication in G_2.

  Verify : 1 Map-to-point hash operation; 2 pairing computations.

  Verifiably Encrypted Signature Creation : 1 Map-to-point hash operation; 3 scalar multiplications in G_2; 1 addition in G_2;

  Verifiably Encrypted Signature Verification : 1 Map-to-point hash operation; 3 pairing computations; 1 multiplication in G_T.

  Adjudication : 1 scalar multiplication in G_2 + 1 inversion in G_2.

A.3.6 Bilinear Ring Signature

(Boneh, Gentry, Lynn, Shacham [25], 2003)

Consider a set U of n users each having a public/private key pair. Ring signature on U is a signature that is constructed using all these public keys of the users in U, and a single private key of any user in U. A ring signature has the property of signer-ambiguity : a verifier is convinced that the signature was produced using one of the private keys of U, but is not able to determine which one.

- Protocol Description :

  KeyGen : Consider the Co-GDH setup. Let $H : \{0,1\}^* \rightarrow G_2$ be a Map-to-point hash function. The secret key is $x_i \in_R Z_q^*$ and the public key is $P_{pub_i} = x_i P_1$ for user $u_i \in U$.

  Ring Signing : Given public keys $P_{pub_1}, \ldots, P_{pub_n} \in G_1$, a message $m \in \{0,1\}^*$, and a private key x_s for a certain $s, 1 \leq s \leq n$, choose $a_i \in_R Z_q$ for all $i \neq s$, compute $h = H(m) \in G_2$ and set

  $$\sigma_s = \frac{1}{x_s}(h - \psi(\sum_{i \neq s} a_i P_{pub_i})).$$

  For all $i \neq s$, let $\sigma_i = a_i P_2$. Output the ring signature $\sigma = (\sigma_1, \ldots, \sigma_n) \in G_2^n$.

  Ring Verification : Given public keys $P_{pub_1}, \ldots, P_{pub_n} \in G_1$, a message $m \in \{0,1\}^*$, and a ring signature σ, compute $h = H(m)$ and verify $e(P_1, h) = \prod_{i=1}^{n} e(P_{pub_i}, \sigma_i)$.

- Assumption :
  Existence of Co-GDH group and a bilinear map.

- Security :
  The identity of the signer is unconditionally protected and the scheme is resistant to forgery in the random oracle model assuming that the Co-CDH problem is hard in (G_1, G_2).

- Efficiency :

  KeyGen : n scalar multiplications in G_1.

  Ring Signing : 1 inversion in Z_q^*; 1 Map-to-point hash operation; $(n-1)$ scalar multiplications in G_2; $(n-1)$ scalar multiplications in G_1; 1 inversion in G_2.

  Ring Verification : 1 Map-to-point hash operation; $(n+1)$ pairing computations.

A.3.7 ZSS Short Signature

(Zhang, Safavi-Naini, Susilo, [121], 2004)

- Protocol Description :

KeyGen : Let $H : \{0,1\}^* \rightarrow Z_q^*$ be a hash function. The secret key is $x \in_R Z_q^*$ and the public key is $P_{pub} = xP$ for a signer.

Sign : Given a secret key x and a message $m \in \{0,1\}^*$, compute signature $S = \frac{1}{H(m)+x}P$.

Verify : Given a public key P_{pub}, a message m and a signature S, verify $e(H(m)P+P_{pub}, S) = e(P,P)$.

- Assumption :
  $(k+1)$-exponent problem is hard.

- Security :
  Existentially unforgeable under an adaptive chosen message attack in the random oracle model assuming that $(k+1)$-exponent problem is hard.

- Efficiency :

  KeyGen : 1 scalar multiplication in G_1.

  Sign : 1 inversion in Z_q^*; 1 hash function (H) evaluation; 1 scalar multiplication in G_1.

  Verify : 2 pairing computation (one of which, $e(P,P)$ can be precomputed); 1 scalar multiplication in G_1; 1 hash function (H) evaluation; 1 addition in G_1.
  This scheme is more efficient than BLS scheme as it requires less pairing computation and no computation of the expensive special hash function Map-to-point that encodes finite strings to elements of group G_1.

A.3.8 ID-Based Blind Signature (Schnorr type)

(Zhang, Kim [120], 2002)

- Protocol Description :

  Setup : Let $H : \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function. Consider another hash function $H_1 : \{0,1\}^* \times G_2 \rightarrow Z_q$. Choose $s \in_R Z_q^*$ and set $P_{pub} = sP$. The master key is s and the global public key is P_{pub}.

  Extract : Given signer's public identity $\mathsf{ID} \in \{0,1\}^*$, compute the public key $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ and the private key $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$.

  Blind Signature Issuing Protocol : Given a signer's private key S_{ID} and a message $m \in \{0,1\}^*$,

  (Initialization) The signer randomly chooses a number $r \in Z_q$, computes $R = rP$ and sends R to the user as a commitment.

  (Blinding) The user randomly chooses $a, b \in Z_q^*$ as blinding factors, computes $c = H(m, e(bQ_{\mathsf{ID}} + R + aP, P_{pub})) + b$ and sends c to the signer.

  (Signing) The signer sends back S, where $S = cS_{\mathsf{ID}} + rP_{pub}$.

(Unblinding) The user computes $S' = S + aP_{pub}$ and $c' = c - b$ and outputs (m, S', c'). Then (S', c') is the blind signature of the message m.

Verification : Accept if and only if $c' = H(m, e(S', P)e(Q_{\mathsf{ID}}, P_{pub})^{-c'})$.

- Assumption :
  ROS-problem is hard.

- Security :
  Secure against one more forgery in the random oracle model under the assumption that ROS problem is hard.

- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Blind Signature Issuing Protocol : 6 scalar multiplications in G_1; 1 pairing computation; 1 hash function (H) evaluation; 1 addition in Z_q; 4 additions in G_1; 1 inversion in Z_q.

  Verification : 1 hash (H) function evaluation; 2 pairing computations; 1 exponentiation in G_2.

A.3.9 ID-Based Ring Signature

(Zhang, Kim [120], 2002)

- Protocol Description :

  Setup : Let $H_1 : \{0,1\}^* \to G_1$ be a Map-to-point hash function and $H : \{0,1\}^* \to Z_q^*$ be another hash function. Choose $s \in_R Z_q^*$ and set $P_{pub} = sP$. The master key is s and the global public key is P_{pub}.

  Extract : Given public identity $\mathsf{ID} \in \{0,1\}^*$, compute the public key $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ and the secret key $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. Let ID_i be a user's identity and S_{ID_i} be the private key associated with ID_i for $i = 1, \ldots, n$. Let $L = \{\mathsf{ID}_i : 1 \leq i \leq n\}$ be the set of identities. The real signer's identity ID_k is listed in L.

  Signing : Given signer's private key S_{ID_k} and a message $m \in \{0,1\}^*$,

  (Initialization) : Choose an element $A \in_R G_1$ and compute $c_{k+1} = H(L||m||e(A, P))$.

  (Generate forward ring sequence) : For $i = k+1, \ldots n-1, 0, 1, \ldots, k-1$, choose randomly $T_i \in G_1$ and compute $c_{i+1} = H(L||m||e(T_i, P)e(c_i H_1(\mathsf{ID}_i), P_{pub}))$.

  (Forming the ring) : Compute $T_k = A - c_k S_{\mathsf{ID}_k}$.

  (Output the ring signature) : The resulting signature for m and L is the $(n+1)$-tuple : $(c_0, T_0, T_1, \ldots, T_{n-1})$.

Verification : Given $(c_0, T_0, T_1, \ldots, T_{n-1})$, m and L, compute $c_{i+1} = H(L||m||e(T_i, P)e(c_i H_1(\mathsf{ID}_i), P_{pub}))$ for $i = 0, 1, \ldots n-1$. Accept if $c_n = c_0$ and reject otherwise.

- Assumption :
  CDH problem is hard.
- Security :
  The scheme is unconditionally signer-ambiguous and non-forgeable in the random oracle model under the assumption that CDH problem is hard.
- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Signing : n hash function (H) evaluation; $(2n-1)$ pairing computations.

  Verification : $2n$ pairing computations; n hash function (H) evaluation.

A.3.10 ID-Based Signature from Pairing

(Hess, [63], 2002)

- Protocol Description :

  Setup : Choose $s \in_R Z_q^*$ and set $P_{pub} = sP$. The master key is s and the global public key is P_{pub}. Let $H_1 : \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function and $H : \{0,1\}^* \times G_2 \rightarrow Z_q^*$ be another hash function.

  Extract : Given a public identity $\mathsf{ID} \in \{0,1\}^*$, compute the public identity $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$ and the secret key $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$.

  Sign : Given a secret key S_{ID} and a message $m \in \{0,1\}^*$, the signer chooses an arbitrary $P_1 \in G_1^*$ and a random $k \in Z_q^*$ and computes

  1. $r = e(P_1, P)^k$,
  2. $v = H(m, r)$,
  3. $u = vS_{\mathsf{ID}} + kP_1$.

  The signature is then the pair $(u, v) \in G \times Z_q^*$.

  Verify : Given a public key Q_{ID}, a message m and a signature (u, v) the verifier computes :

  1. $r = e(u, P)e(Q_{\mathsf{ID}}, -P_{pub})^v$
  2. Accept the signature if and only if $v = H(m, r)$.

  Assumption :
  Weak-DH problem is hard.

Security :

Secure against existential forgery under adaptive chosen message attack in the random oracle model assuming Weak-DH problem is hard.

Efficiency :

Setup : 1 scalar multiplication in G_1.

Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

Sign : The signing operation can be optimized by the signer pre-computing $e(P_1, P)$ for P_1 of his choice, for example $P_1 = P$, and storing this value with the signing key. This means that the signing operation involves one exponentiation in the group G_2, one hash function (H) evaluation and one simultaneous multiplication in the group G_1.

Verify : The verification operation requires one exponentiation in G_2, one hash function (H) evaluation and two evaluations of the pairing. One of the pairing evaluation can be eliminated, if a large number of verifications are to be performed for the same identity, by pre-computing $e(Q_{ID}, -P_{pub})$.
This scheme is very efficient in terms of communication requirements. One needs to transmit one element of the group G_1 and one element of Z_q^*.

A.3.11 Unique Signature Without Random Oracle

(Lysyanskaya [83], 2002)

Unique signature schemes, also known as invariant signature schemes, are secure signature schemes where the signature is a hard-to-compute function of the public key and the message. One must verify a signature again even if it has been accepted before. Because this time the signature may come from an unauthorized party. If a signature scheme allows the signer to easily generate many signatures on the same message, then it simply leads to denial-of-service attack on a verifier who is forced to verify many signatures on the same message. This illustrates that intuitively unique signatures are desirable. Boneh and Silverberg [28] proposed a unique signature scheme based on the existence of multi-linear maps. Currently, no such suitable maps are known and the existence of such maps is presently a research problem [28]. Lysyanskaya proposed a unique signature scheme based on this idea while making use of bilinear pairing. This scheme is proven to be secure in the standard model under Many-DH assumption.

- Protocol Description :

KeyGen : Choose n pairs of random elements in Z_q^* : $(a_{1,0}, a_{1,1}), (a_{2,0}, a_{2,1}), \ldots, (a_{n,0}, a_{n,1})$. This is the secret key for a signer. Compute $A_{i,b} = a_{i,b}P, 1 \leq i \leq n, b \in \{0, 1\}$. The public key for the signer is $P_{pub} = \{A_{i,0}, A_{i,1} | 1 \leq i \leq n\}$.

Sign : Assume that the messages being signed are n-bit codewords of a code of distance Cn, where $0 < C \leq 1/2$ is a constant. Given the secret key and an n-bit codeword

$m = m_1 \circ m_2 \circ \ldots \circ m_n$, output the signature

$$\sigma = \{s_{m,i} = (\prod_{j=1}^{i} a_{j,m_j})P : 1 \leq i \leq n\}.$$

Verify : Let $s_{m,0} = 1$. Given the public key P_{pub}, verify that, for all $i, 1 \leq i \leq n$, $e(P, s_{m,i}) = e(s_{m,i-1}, A_{i,m_i})$.

Graphically, we view the message space as the leaves of a balanced binary tree of depth n. Each internal node of the tree is assigned a label, as follows : the label of the root is P. The label of a child, denoted by l_c is obtained from the label of its parent, denoted by l_p as follows : if the depth of the child is i, and it is the left child, then its label is $l_c = a_{i,0}l_p$ while if it is the right child, its label will be $l_c = a_{i,1}l_p$. The signature on an n-bit message consists of all the labels on the path from the leaf corresponding to this message all the way to the root. To verify the correctness of a signature, the fact that Decision Diffie-Hellman is easy in G_1 is used.

- Assumption :
  Existence of GDH group, Many-DH problem is hard.
- Security :
  Provably secure against existential forgery under adaptive chosen message attack in the standard model assuming the underlying group is a GDH group and the hardness of Many-DH problem.
- Efficiency :

  KeyGen : $2n$ scalar multiplications in G_1.

  Sign : n scalar multiplications in G_1, $(n-1)$ multiplications in Z_q^*.

  Verify : $2n$ pairing computations.

A.3.12 Authentication-Tree Based Signature Without Random Oracle

(Boneh, Mironov, Shoup [27], 2003)

In an authentication-tree based scheme, signatures are produced that represent paths connecting messages and the root of the tree. Messages are usually placed in the very bottom level of the tree. The authentication mechanism works inductively : the root authenticates its children, they authenticate their children, and so on, down to the message authenticated by its parent.

- Protocol Description :

KeyGen : Consider a keyed family of collision resistant hash functions $\mathcal{H}_k : \mathcal{M} \rightarrow \{0,1\}^s$ where $\mathcal{M}$ is the message space. The signature scheme allows signing l^n messages, where l and n are arbitrary positive integer, n is the branching factor of the authentication tree.

1. Pick randomly $\alpha_i \in Z_q^*, 1 \leq i \leq n$ and $Q \in_R G_1$. Choose a random key k for the collision resistant hash function $\mathcal{H}_k$. Compute $Q_1 = (1/\alpha_1)Q, \ldots, Q_n = (1/\alpha_n)Q \in G_1$.
2. Pick randomly $R \in G_1$. Compute $y = e(R, Q)$.
3. Pick randomly $\beta_0 \in Z_q$. Compute $x_0 = y^{\beta_0}$.
4. The public key for a signer is $(k, Q, Q_1, \ldots, Q_n, y, x_0)$ and the corresponding private key is $(\alpha_1, \alpha_2, \ldots, \alpha_n, \beta_0, R)$.

Sign : Each node in the tree is authenticated with respect to its parent; messages to be signed are authenticated with respect to the leaves, which are selected in sequential order and never reused. To sign i-th message $m \in \mathcal{M}$, the signer generates the i-th leaf of the authenticated tree together with a path from the leaf to the root. Denote the path from leaf to root by $(x_l, i_l, x_{l-1}, i_{l-1}, \ldots, i_1, x_0)$:
x_j is the i_j-th child of $x_{j-1} (i_j \in \{1, \ldots, n\})$.

1. $x_j = y^{\beta_j}$ for some $\beta_j \in_R Z_q^*, 1 \leq j \leq l$. The secret β_j is stored for as long as node x_j is an ancestor of the current signing leaf.
2. Compute $f_j = \alpha_{i_j}(\beta_{j-1} + \mathcal{H}_k(x_j))R$. This is the authenticated value of x_j, the i_j-th child of x_{j-1}.
3. Compute $f = (\beta_l + \mathcal{H}_k(m))R$. This is the authenticated value of m.
4. The signature on m is $(f, f_l, i_l, \ldots, f_1, i_1)$.

Verify : Given a signature $(\hat{f}, \hat{f}_l, \hat{i}_l, \ldots, \hat{f}_1, \hat{i}_1)$ on a message m, do the followings :

1. Compute $\hat{x}_l = e(\hat{f}, Q)y^{-\mathcal{H}_k(m)}$.
2. Compute $\hat{x}_{j-1} = e(\hat{f}_j, Q_{i_j})y^{-\mathcal{H}_k(\hat{x}_j)}$ for $l \leq j \leq 1$.
3. Accept the signature if $\hat{x}_0 = x_0$.

- Assumption :
  CDH problem is hard.

- Security :
  Provably secure against existential forgery against adaptive chosen message attack assuming that the CDH problem is hard.

- Efficiency :

  KeyGen : n scalar multiplications in G_1; 1 pairing computation; 1 exponentiation in G_2.

  Sign : l exponentiations in G_2; $(l+1)$ hash function $(\mathcal{H}_k)$ evaluations; $(l+1)$ additions in Z_q^*; l multiplications in Z_q^*, l scalar multiplications in G_1.

  Verify : $(l+1)$ pairing computations; $(l+1)$ hash function $(\mathcal{H}_k)$ evaluations; $(l+1)$ exponentiations in G_2; $(l+1)$ multiplications in G_2.

A.3.13 Short Signature Without Random Oracle

(Boneh, Boyen [20], 2004)

- Protocol Description :

  KeyGen : The secret key is $(x, y) \in_R Z_q^* \times Z_q^*$ and the public key is (P, U, V) where $U = xP$ and $V = yP$ for a signer. The messages are assumed to be elements of Z_q^*.

  Sign : Given a secret key (x, y), a message $m \in Z_q^*$, choose a random $r \in Z_q^*$ and compute $\sigma = \frac{1}{x+m+yr}P$. Here $\frac{1}{x+m+yr}$ is computed modulo q and the unlikely event $x+m+yr = 0$ is avoided by choosing a different r. The signature is (σ, r).

  Verify : Given a public key (P, U, V), a message $m \in Z_q^*$ and a signature (σ, r), verify $e(\sigma, U + mP + rV) = e(P, P)$.

- Assumption :
  q-SDH problem is hard.

- Security :
  Secure against existential forgery under chosen message attack under SDH assumption and without using the random oracle model.

- Efficiency :

  KeyGen : 2 scalar multiplications in G_1.

  Sign : 1 inversion in Z_q^*; 1 scalar multiplication in G_1.

  Verify : 2 scalar multiplication in G_1; 2 additions in G_1; 2 pairing computations one of which, $e(P, P)$ can be precomputed.

Note : Recently, Waters [117] proposed an efficient signature scheme that depends only upon the CDH assumption in the standard model (*i.e.* without using any random oracle).

A.4 Key Agreement Schemes

Key agreement is one of the fundamental cryptographic primitives. This is required when two or more parties want to communicate securely. In one of the breakthroughs in key agreement, Joux [66] proposed a three party single round key agreement protocol using pairing. This was the first positive application of bilinear pairing in cryptography. Afterwards, pairings were used widely to get a large number of cryptographic protocols some of which have been previously mentioned. Several key agreement protocols were proposed that prevents man-in-the-middle attack against a passive adversary. These protocols are called unauthenticated. The protocols for authenticated key agreement enables a group of parties within a large and completely insecure public network to establish a common secret key and furthermore ensures that they are indeed sharing this key

with each other. Achieving authenticated key agreement are crucial for allowing symmetric-key encryption/authentication of data among the parties. Authenticated key agreement protocols are the basic tools for group-oriented and collaborative applications such as, distributed simulation, multi-user games, audio or video-conferencing, and also peer-to-peer application that are likely to involve a large number of users. These are used to construct secure channels which are the base for designing, analyzing and implementing higher-level protocols in a modular approach. A formal model of security for group authenticated key agreement can be found in [32]. Much research work remains to be done in this area. We devote Chapter 3 for a comprehensive treatment of describing the most important key agreement protocols.

A.5 Threshold Schemes

The idea behind the (t, n)-threshold cryptosystem approach is to distribute secret information (*i.e.* the secret key) and computation (*i.e.* signature generation or decryption) among n parties in order to remove single point failure. The goal is to allow a subset of more than t players to jointly reconstruct a secret and perform the computation while preserving security even in the presence of an active adversary which can corrupt up to t (a threshold) parties. The secret key is distributed among n parties with the help of a trusted dealer or without it by running an interactive protocol among all parties.

A.5.1 Threshold Signature

(Boldyreva, [18], 2003)

- Protocol Description :

  KeyGen : Let $H : \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function. Suppose there are n servers $u_i, 1 \leq i \leq n$. The private key $x \in Z_q^*$ is shared among these users using Shamir's secret sharing scheme such that any subset S of $t+1$ servers can reconstruct x using Lagrange interpolation : $x = \sum_{i \in S} L_i x_i$, where $L_i = \prod_{j \in S} \frac{-x_j}{(x_i - x_j)}$ is the Lagrange co-efficient, x_i is the private key share and $P_{pub_i} = x_i P$ is the public key share of user u_i.

  Signature Share Generation : To sign a message $m \in \{0,1\}^*$, user u_i outputs $\sigma_i = x_i H(m)$.

  Signature Share Verification : Given m, σ_i, P_{pub_i}, anyone can check whether user u_i is honestly behaving in giving it's share σ_i of signature by checking $e(P, \sigma_i) = e(P_{pub_i}, H(m))$. If σ_i passes through this test, call it an acceptable share.

  Signature Reconstruction : Suppose a set S of $(t+1)$ honest servers are found and accordingly $(t+1)$ acceptable shares $\sigma_i, i \in S$. The resulting signature on m is $\sigma = \sum_{i \in S} L_i \sigma_i$. The correctness of the scheme is easy to verify since $e(P, \sigma) = e(H(m), xP)$.

- Assumption :
  Existence of GDH group.
- Security :
  Secure in the random oracle model against an adversary which is allowed to corrupt any $t < n/2$ players under the assumption that the underlying group is GDH.
- Efficiency :

  KeyGen : n scalar multiplications in G_1.

  Signature Share Generation : For each user, 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Signature Share Verification : 2 pairing computations; 1 Map-to-point hash operation.

  Signature Reconstruction : $(t+1)$ scalar multiplications in G_1; t additions in G_1; $(t+1)$ Lagrange co-efficient (L_i) computations.

A.5.2 Pairing Based (t, n)-Threshold Decryption

(Libert, Quisquater [81], 2003)

The following scheme is a threshold adaption of the Boneh-Franklin IBE scheme where a fixed PKG plays the role of a trusted dealer.

- Protocol Description:

  KeyGen : Choose a $(t-1)$-degree polynomial $f(x) = s + a_1 x + \cdots + a_{t-1}x^{t-1}$ for random $a_1, \ldots, a_{t-1} \in Z_q^*$. For $i = 1, 2, \ldots, n$, compute $P_{pub}^{(i)} = f(i)P \in G_1$ and $P_{pub} = sP$.
  Let $H_1 : \{0,1\}^* \rightarrow G_1$ be a Map-to-point hash function and $H_2 : G_2 \rightarrow \{0,1\}^l$ be another hash function. Before requesting his private share, each player can check that $\sum_{i \in S} L_i P_{pub}^{(i)} = P_{pub}$ for any subset $S \subset \{1, \ldots, n\}$ such that $|S| = t$ where L_i denotes the appropriate Lagrange co-efficient explicitly given by the formula $L_i = \prod_{j \in S} \frac{-x_j}{(x_i - x_j)}$.
  Given a user's identity $\mathsf{ID} \in \{0,1\}^*$, the PKG plays the role of the trusted dealer. For $i = 1, \ldots, n$, it delivers $d_{\mathsf{ID}_i} = f(i)Q_{\mathsf{ID}} \in G_1$ to player i. After receiving d_{ID_i}, player i checks
  $$e(P_{pub}^{(i)}, Q_{\mathsf{ID}}) = e(P, d_{\mathsf{ID}_i}).$$
  If verification fails, he complains to the PKG which then issues a new share.

  Encrypt : Given message $m \in \{0,1\}^l$ and identity ID, compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$. Choose a random $r \in Z_q^*$ and set the cipher text to be $C = \langle rP, m \oplus H_2(e(P_{pub}, Q_{\mathsf{ID}})^r) \rangle$.

  Decryption Share Generation : When receiving $\langle U, V \rangle$, player i computes his decryption share $e(U, d_{\mathsf{ID}_i})$ and gives it to the re-combiner who may be a designated player.

Recombination : The re-combiner selects a set $S \subset \{1, \ldots, n\}$ of t acceptable share $e(U, d_{\mathsf{ID}_i})$ and computes $g = \prod_{i \in S} e(U, d_{\mathsf{ID}_i})^{L_i}$. Once he has g, he recovers the plain text $m = V \oplus H_2(g)$.

Correctness of the scheme is easy to verify since $g = e(rP, \sum_{i \in S} L_i d_{\mathsf{ID}_i}) = e(rP, sQ_{\mathsf{ID}}) = e(P_{pub}, Q_{\mathsf{ID}})^r$. To check publicly whether the share of a player is acceptable or not, do the following : Each player chooses a random $R \in G_1$ and computes $w_1 = e(P, R), w_2 = e(U, R)$ and $h = H(e(U, d_{\mathsf{ID}_i}), e(P_{pub}, Q_{\mathsf{ID}}), w_1, w_2)$. Next, player i computes $V = R + hd_{\mathsf{ID}_i} \in G_1$ and joins the tuple (w_1, w_2, h, V) to it's share. The other players can check that $e(P, V) = e(P, R)e(P_{pub}^{(i)}, Q_{\mathsf{ID}})^h$, $e(U, V) = e(U, R)e(U, d_{\mathsf{ID}_i})^h$. If this test fails, player i is a dishonest player.

- Assumption :
  BDH problem is hard.

- Security :
  This threshold IBE scheme is provably secure against chosen plain text attacks in the ID-based setting under the BDH assumption.

- Efficiency :

  KeyGen : n function (f) evaluation; $(2n + 1)$ scalar multiplications in G_1; $2n$ pairing computations.

  Encrypt : 1 Map-to-point hash operation; 1 hash function (H_2) evaluation; 1 XOR operation; 1 exponentiation in G_2; 1 scalar multiplication in G_1.

  Decryption Share Generation : For each share holder, 1 pairing computation.

  Recombination : $|S|$ pairing computations; $(|S| - 1)$ multiplications in G_2; $|S|$ Lagrange co-efficient computations.

A.5.3 ID-based (t, n)-Threshold Decryption

(Baek, Zheng [6], 2003)

Consider the following scenario : Alice wishes to send a confidential message to a committee in an organization. She can first encrypt the message using the identity of the committee and then send over the cipher text. Suppose Bob who is the committee's president has created the identity and has obtained a matching private decryption key from the PKG. Preparing for the time when Bob is away, he can share his private key out among a member of decryption server in such a way that any committee member can successfully decrypt the cipher text if and only if the committee member obtains a certain number of decryption shares from the decryption servers. *i.e.* Bob himself plays the role of a trusted dealer.

The following scheme provides the feature that a user who obtained a private key from the PKG can share the key among decryption servers at will. After key generation, the PKG can be closed. Also this protocol achieves chosen cipher text security under BDH assumption in random oracle model.

- Protocol Description :

  KeyGen : PKG chooses $x \in_R Z_q^*$ and computes $P_{pub} = xP$. The master key of PKG is x and the public key is P_{pub}. Consider four hash functions : $H_1 : G_2 \to \{0,1\}^l$, $H_2 : G_1 \times \{0,1\}^l \to G_1$, $H_3 : \{0,1\}^* \to G_1$, $H_4 : G_2^3 \to Z_q^*$. H_3 is a Map-to-point hash function.

  Extract : Given an identity $\mathsf{ID} \in \{0,1\}^*$, compute $Q_{\mathsf{ID}} = H_3(\mathsf{ID}), D_{\mathsf{ID}} = xQ_{\mathsf{ID}}$ and returns D_{ID}.

  Private Key Distribution : Given a private key D_{ID}, n decryption shares and a threshold parameter $t \leq n$, pick randomly $R_1, R_2, \ldots, R_{t-1} \in G_1^*$ and compute $F(u) = D_{\mathsf{ID}} + uR_1 + u^2R_2 + \ldots + u_{t-1}R_{t-1}$ for $u \in \{0\} \cup N$. Compute $S_i = F(i), y_i = e(S_i, P), 1 \leq i \leq n$ and sends (S_i, y_i) secretly to server $\Gamma_i, 1 \leq i \leq n$. Γ_i then keeps S_i as secret while it publishes y_i.

  Encrypt : Given a plain text $m \in \{0,1\}^l$, identity $\mathsf{ID} \in \{0,1\}^*$, choose $r \in Z_q^*$ at random and set $U = rP$. Compute $Q_{\mathsf{ID}} = H_3(\mathsf{ID}), d = e(Q_{\mathsf{ID}}, P_{pub})$, $\kappa = d^r$, $V = H_1(\kappa) \oplus m$, $W = rH_2(U,V)$ and set the cipher text to be $C = (U,V,W)$.

  Decryption Share Generation : Given a cipher text $C = (U,V,W)$, decryption server Γ_i with secret key S_i computes $H_2 = H_2(U,V)$ and checks if $e(P,W) = e(U,H_2)$. If the test holds then compute $\kappa_i = e(S_i,U)$, $\widetilde{\kappa}_i = e(Q_i,U)$, $\widetilde{y}_i = e(Q_i,P)$, $\lambda_i = H_4(\kappa_i, \widetilde{\kappa}_i, \widetilde{y}_i)$, $L_i = Q_i + \lambda_i S_i$, where Q_i is chosen randomly from G_1. Output $\delta_i = (i, \kappa_i, \widetilde{\kappa}_i, \widetilde{y}_i, L_i)$.

  Decryption Share Verification : Given a cipher text $C = (U,V,W)$ and a decryption share $\delta_i = (i, \kappa_i, \widetilde{\kappa}_i, \widetilde{y}_i, L_i)$, compute $\lambda_i = H_4(\kappa_i, \widetilde{\kappa}_i, \widetilde{y}_i)$. Check if

  $$\frac{e(L_i,U)}{\kappa_i^{\lambda_i}} = \widetilde{\kappa}_i, \quad \frac{e(L_i,P)}{y_i^{\lambda_i}} = \widetilde{y}_i.$$

  If the above test holds, then share δ_i of server Γ_i is an acceptable share. Given acceptable shares $S_j, j \in S \subseteq \{1,\ldots,n\}$ where $|S| \geq t$, D_{ID} can be recovered as follows : $D_{\mathsf{ID}} = F(0) = \sum_{j \in S} c_{oj} S_j$, c_{0j} are appropriate Lagrange co-efficients.

  Share Combining : Given a cipher text $C = (U,V,W)$ and a set of decryption shares $\{\delta_j\}_{j \in S \subseteq \{1,2,\ldots,n\}}$ where $|S| \geq t$, compute $H_2 = H_2(U,V)$, check if $e(P,W) = e(U,H_2)$. If C passes this test (*i.e.* C is a valid cipher text), compute $\kappa = \prod_{j \in S} \kappa_j^{c_{0j}}$ and $m = H_1(\kappa) \oplus V$. Output m.

  The correctness of the scheme is easy to verify since

  $$\prod_{j \in S} \kappa_j^{c_{0j}} = \prod_{j \in S} e(S_j,U)^{c_{0j}} = e(\sum_{j \in S} c_{0j} S_j, U) = e(\sum_{j \in S} c_{0j} S_j, rP) = e(D_{\mathsf{ID}}, P)^r.$$

- Assumption :
  BDH problem is hard.
- Security :
  This protocol achieves chosen cipher text security in the random oracle model under BDH assumption.
- Efficiency :

  KeyGen : 1 scalar multiplication.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Private Key Distribution : For each share holder, $(t-1)$ scalar multiplications in G_1; $(t-1)$ additions in G_1; 1 pairing computation.

  Encrypt : 1 scalar multiplication; 1 Map-to-point hash operation; 1 pairing computation; 1 exponentiation in G_2.

  Decryption Share Generation : For each share holder, 1 hash function H_2 evaluation, 5 pairing computations; 1 hash function H_4 evaluation; 1 scalar multiplication in G_1; 1 addition in G_1.

  Decryption Share Verification : 1 hash function H_4 evaluation; 2 pairing computations; 2 exponentiations in G_2.

  Share Combining : 1 hash function H_2 evaluation; 2 pairing computations; $|S|$ Lagrange co-efficient computations; $(|S|-1)$ multiplications in G_2; 1 hash function H_1 evaluation; 1 XOR operation.

A.6 Miscellaneous Applications

A.6.1 Key Sharing Scheme

(Sakai, Ohgishi, Kasahara [102], 2000)

- Protocol Description :

  Key Generation : Let $H_:\{0,1\}^*$ be a Map-to-point hash function.
  The idea of Key Sharing Scheme is quite simple : Suppose a PKG has a master key s, and it issues private keys to users of the form sP_y, where $P_y = H_1(\mathsf{ID}_y)$ and $\mathsf{ID}_y \in \{0,1\}^*$ is the identity of user y. Then users y and z have a shared secret that only they (and the PKG) may compute, namely

  $$e(sP_y, P_z) = e(P_y, P_z)^s = e(P_y, sP_z).$$

  They may use this shared secret to encrypt their communications. This key sharing scheme is non-interactive and can be viewed as a type of "dual-identity-based encryption", where the word "dual" indicates that the identities of both the sender and the

recipient (rather than just the recipient) are required as input into the encryption and decryption algorithm.

- Assumption :
  BDH problem is hard.

- Efficiency :
  For each party, 1 pairing computation for key sharing; 1 scalar multiplication in G_1; 1 Map-to-point hash operation for private key extraction.

A.6.2 ID-Based Chameleon Hashes from Bilinear Pairings

(Zhang, Safavi- Naini, Susilo [123], 2003)

A chameleon hash function is a trapdoor one-way hash function : without knowledge of the associated trapdoor, the chameleon hash function is resistant to the computation of pre-images and collisions. However, with the knowledge of the trapdoor, collisions are efficiently computable.
Applications : ID-based chameleon hash functions can be used to construct ID-based chameleon signature schemes which achieves the goal of ID-based undeniable signature and is non-interactive. An ID-based chameleon signature scheme is an ID-based signature computed over the ID-based chameleon hash of a certain message m under the identity of the intended recipient. The recipient can verify that the signature of the message m is valid, but can not prove to others that the signer actually signed m and not another message. Indeed, the recipient can find collisions of the chameleon hash function, thus finding a message different from m which would pass the signature verification procedure.

(a) Scheme 1

- Protocol Description :

  Setup : PKG chooses a random number $s \in Z_q^*$ and sets $P_{pub} = sP$. The master key of PKG is s and the public key is P_{pub}. Consider a Map-to-point hash function $H_0 : \{0,1\}^* \rightarrow G_1$ and another hash function $H_1 : \{0,1\}^l \rightarrow Z_q^*$.

  Extract : A user submits his identity $\mathsf{ID} \in \{0,1\}^*$ to PKG which computes the public key as $Q_{\mathsf{ID}} = H_0(\mathsf{ID})$ and returns $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$ to the user as his private key.

  Hash : Given a message $m \in \{0,1\}^l$, choose a random element R from G_1, define the hash as $\mathsf{Hash}(\mathsf{ID}, m, R) = e(R, P)e(H_1(m)H_0(\mathsf{ID}), P_{pub})$.

  Forge : $\mathsf{Forge}(\mathsf{ID}, S_{\mathsf{ID}}, m, R, m') = R' = (H_1(m) - H_1(m'))S_{\mathsf{ID}} + R$.
  The forgery is correct because

  $$\mathsf{Hash}(\mathsf{ID}, m', R')$$

  $$= \quad e(R', P)\ e(H_1(m')H_0(\mathsf{ID}), P_{pub})$$

$$
\begin{aligned}
&= e((H_1(m) - H_1(m'))S_{\mathsf{ID}} + R, P)\ e(H_1(m')H_0(\mathsf{ID}), P_{pub}) \\
&= e((H_1(m) - H_1(m'))S_{\mathsf{ID}}, P)\ e(R, P)\ e(H_1(m')H_0(\mathsf{ID}), P_{pub}) \\
&= e((H_1(m) - H_1(m'))H_0(\mathsf{ID}), P_{pub})\ e(R, P)\ e(H_1(m')H_0(\mathsf{ID}), P_{pub}) \\
&= e(R, P)\ e(H_1(m)H_0(\mathsf{ID}), P_{pub}) \\
&= \mathsf{Hash}(\mathsf{ID}, m, R)
\end{aligned}
$$

- Assumption :
  BLS signature scheme is secure.
- Security :
  Semantically secure and resistant to collision forgery under active attacks provided BLS signature scheme is secure.
- Efficiency :

  Setup : 1 scalar multiplication.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Hash : 2 pairing computations; 1 scalar multiplication in G_1; 1 Map-to-point hash operation; 1 hash function H_1 evaluation. Using precomputation for $a = e(P, P)$ and $b = e(H_0(\mathsf{ID}), P_{pub})$, to compute the chameleon hash of a message m, the sender requires only 1 EC scalar multiplication of G_1 + 2 group exponentiation in G_2. *i.e.* $R = rP, \mathsf{Hash}(\mathsf{ID}, m, R) = a^r b^{H_1(m)}$.

  Forge : 2 hash function H_1 evaluation; 1 scalar multiplication in G_1; 1 subtraction in Z_q^*; 1 addition in G_1.

(b) Scheme 2

- Protocol Description:

  Setup : PKG chooses a random number $s \in Z_q^*$ and sets $P_{pub} = sP$. The master key of PKG is s and the public key is P_{pub}. Consider two hash functions $H_0 : \{0,1\}^* \to Z_q^*$ and $H_1 : \{0,1\}^l \to Z_q^*$.

  Extract : Given an identity $\mathsf{ID} \in \{0,1\}^*$, compute $S_{\mathsf{ID}} = \frac{1}{s+H_0(\mathsf{ID})}P$. S_{ID} is the private key corresponding to the public identity ID.

  Hash : Given a message $m \in \{0,1\}^l$, an identity $\mathsf{ID} \in \{0,1\}^*$ and a random element $R \in G_1$, define $\mathsf{Hash}(\mathsf{ID}, m, R) = e(P, P)^{H_1(m)} e(H_0(\mathsf{ID}) + P_{pub}, R)^{H_1(m)}$

  Forge : $\mathsf{Forge}(\mathsf{ID}, S_{\mathsf{ID}}, m, R, m') = R' = H_1(m')^{-1}((H_1(m) - H_1(m'))S_{\mathsf{ID}} + H_1(m)R)$.

  The forgery is correct because

  $$
  \begin{aligned}
  &\mathsf{Hash}(\mathsf{ID}, m', R') \\
  &= e(P, P)^{H_1(m')}\ e(H_0(\mathsf{ID}) + P_{pub}, R')^{H_1(m')} \\
  &= e(P, H_1(m')P)\ e(H_0(\mathsf{ID}) + P_{pub}, H_1(m')H_1(m')^{-1}((H_1(m) - H_1(m'))S_{\mathsf{ID}} + H_1(m)R))
  \end{aligned}
  $$

$$= e(P, H_1(m')P)\ e(H_0(\mathsf{ID}) + P_{pub}, (H_1(m) - H_1(m'))S_{\mathsf{ID}})e(H_0(\mathsf{ID}) + P_{pub}, H_1(m)R)$$
$$= e(P, H_1(m')P)\ e(P, (H_1(m) - H_1(m'))P)\ e(H_1(\mathsf{ID}) + P_{pub}, H_1(m)R))$$
$$= e(P, P)^{H_1(m)}\ e(H_1(\mathsf{ID}) + P_{pub}, R)^{H_1(m)}$$
$$= \mathsf{Hash}(\mathsf{ID}, m, R).$$

- Assumption :
  ZSS signature scheme is secure.
- Security :
  Semantically secure and resistant to collision forgery under active attacks, provided ZSS signature scheme is secure.
- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 hash function H_0 evaluation; 1 addition in Z_q^*; 1 multiplicative inverse in Z_q^*; 1 scalar multiplication in G_1.

  Hash : 2 pairing computations; 1 hash function H_0 evaluation; 2 exponentiations in G_2; 1 addition in G_1. Precomputing $a = e(P, P)$, to compute the chameleon hash of a message m, the sender only needs to compute 1 EC scalar multiplication of G_1 + 1 group exponentiation in G_2. *i.e.* $R = rS_{\mathsf{ID}}, \mathsf{Hash}(\mathsf{ID}, m, R) = a^{(r+1)H_1(m)}$.

  Forge : 2 hash function H_1 evaluations; 2 scalar multiplications in G_1; 1 subtraction in Z_q^*; 1 multiplicative inverse in Z_q^*.

A.6.3 Signcryption Schemes

The idea of this primitive is to perform encryption and signature in a single logical step in order to obtain confidentiality, integrity, authentication and non-repudiation more efficiently than the sign-then-encrypt approach. The drawback of this latter situation is to expand the final cipher text size and increase the sender's and receiver's computing time which may be impractical for low bandwidth network. Malone-Lee [84] defines extended security notions for ID-based signcryption schemes.

(a) Identity-Based Signcryption
(Malone-Lee [84], 2003)

- Protocol Description

  Setup : Choose $s \leftarrow^R Z_q^*$ and set $P_{\mathsf{Pub}} = sP$. The master key generated by the trusted party is s and the public key is P_{pub}. Consider three hash functions : $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : \{0,1\}^* \rightarrow Z_q^*$ and $H_3 : G_2 \rightarrow \{0,1\}^l$.

  Extract(ID) : Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$, $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. The secret key corresponding to identity $\mathsf{ID} \in \{0,1\}^*$ is S_{ID} and the public key is Q_{ID}.

  Signcrypt($S_{\mathsf{ID}_a}, \mathsf{ID}_b, m$) : For a message $m \in \{0,1,\}^l$, compute $Q_{\mathsf{ID}_b} = H_1(\mathsf{ID}_b)$. Choose $x \leftarrow^R Z_q^*$ and set $U = xP$. Compute $r = H_2(U||m)$, $W = xP_{pub}$, $V = rS_{\mathsf{ID}_a} + W$, $y = e(W, Q_{\mathsf{ID}_b})$, $\kappa = H_3(y)$, $c = \kappa \oplus m$. Send $\sigma = (c, U, V)$

Unsigncrypt $(\mathsf{ID}_a, S_{\mathsf{ID}_b}, \sigma)$: Compute $Q_{\mathsf{ID}_a} = H_1(\mathsf{ID}_a)$. Parse σ as (c, U, V). Compute $y = e(S_{\mathsf{ID}_b}, U)$, $\kappa = H_3(y)$, $m = \kappa \oplus c$, $r = H_2(U||m)$. Return m if and only if $e(V, P) = e(Q_{\mathsf{ID}_a}, P_{pub})^r \; e(U, P_{pub})$.
Consistency constraint :
if $\sigma = \mathsf{Signcrypt}(S_{\mathsf{ID}_a}, \mathsf{ID}_b, m)$, then $m = \mathsf{Unsigncrypt}(\mathsf{ID}_a, S_{\mathsf{ID}_b}, \sigma)$. This scheme is the result of a combination of the simplified version of Boneh and Franklin's IBE cryptosystem with a variant of Hess's identity based signature.

- Assumption :
  BDH problem is hard.
- Security :
  This protocol achieves the security IND-ISC-CCA (indistinguishability of identity-based signcryptions under chosen cipher text attack) and also the security EF-ISC-ACMA (existentially unforgeability of identity-based signcryptions under adaptive chosen message attack) in the random oracle model assuming BDH problem is hard.
- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Signcrypt : 1 Map-to-point hash operation; 3 scalar multiplications in G_1; 1 hash function H_2 evaluation; 1 pairing computation; 1 hash function H_3 evaluation; 1 XOR operation, 1 addition in G_1.

  Unsigncrypt : 1 Map-to-point hash operation; 4 pairing computations; 1 hash function H_3 evaluation; 1 XOR operation; 1 hash function H_2 evaluation; 1 exponentiation in G_2. The size of the cryptogram is $n + 2|G_1|$ when a message of n-bit is sent.

(b) Another Identity-Based Signcryption
(Libert, Quisquater [82], 2003)

- Protocol Description :

  Setup : Choose $s \leftarrow^R Z_q^*$ and set $P_{pub} \leftarrow sP$. The secret key is s and the public key is P_{pub}. Choose a secure symmetric cipher (E, D) with key space K_s and cipher text space C_s. Also consider three hash functions : $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : G_2 \rightarrow K_s$ and $H_3 : C_s \times G_2 \rightarrow Z_q^*$. H_1 is a Map-to-point hash function.

  Extract(ID) : Compute $Q_{\mathsf{ID}} = H_1(\mathsf{ID})$, $S_{\mathsf{ID}} = sQ_{\mathsf{ID}}$. The secret key corresponding to the identity $\mathsf{ID} \in \{0,1\}^*$ is S_{ID} and the public key is Q_{ID}.

  Signcrypt$(S_{\mathsf{ID}_a}, \mathsf{ID}_b, m)$: For a message $m \in \{0,1\}^l$, compute $Q_{\mathsf{ID}_b} = H_1(\mathsf{ID}_b)$. Choose $x \leftarrow^R Z_q^*$ and set $\kappa_1 = e(P, P_{pub})^x$, $\kappa_2 = H_2(e(P_{pub}, Q_{\mathsf{ID}_b})^x)$. Compute $c = E_{\kappa_2}(m)$, $r = H_3(c, \kappa_1)$, $S = xP_{pub} - rS_{\mathsf{ID}_a}$. Send $\sigma = (c, r, S)$.

  Unsigncrypt$(\mathsf{ID}_a, S_{\mathsf{ID}_b}, \sigma)$: Compute $Q_{\mathsf{ID}_a} = H_1(\mathsf{ID}_a)$. Parse σ as (c, r, S) and set $\kappa_1 = e(P, S) \; e(P_{pub}, Q_{\mathsf{ID}_a})^r$, $\tau = e(S, Q_{\mathsf{ID}_b}) \; e(Q_{\mathsf{ID}_a}, S_{\mathsf{ID}_b})^r$, $\kappa_2 = H_2(\tau)$, $m = D_{\kappa_2}(c)$. Accept if and only if $r = H_3(c, \kappa_1)$.

- Assumption :
  DBDH problem is hard.
- Security :
  This protocol achieves IND-ISC-CCA security for confidentiality and also EF-ISC-ACMA security for unforgeability in the random oracle model assuming DBDH problem is hard.
- Efficiency :

  Setup : 1 scalar multiplication in G_1.

  Extract : 1 Map-to-point hash operation; 1 scalar multiplication in G_1.

  Signcrypt : 1 Map-to-point hash operation; 2 pairing computations; 1 hash function H_2 evaluation; 1 exponentiation in G_2; 1 symmetric key encryption; 1 hash function H_3 evaluation; 2 scalar multiplications in G_1; 1 inversion in G_1.

  Unsigncrypt : 1 Map-to-point hash operation; 4 pairing computations; 2 exponentiations in G_2; 1 hash function H_2 evaluation; 1 symmetric key decryption; 1 hash function H_3 evaluation.

A.6.4 Identification Scheme based on GDH

(Kim, Kim [71], 2002)

Identification scheme is a very important and useful cryptographic tool. It is an interactive protocol where a prover $\mathcal{P}$ tries to convince a verifier $\mathcal{V}$ of his identity. Only $\mathcal{P}$ knows the secret value corresponding to his public one and the secret value allows to convince $\mathcal{V}$ of his identity.

- Protocol Description :

  KeyGen : Choose randomly $a, b, c \in Z_q^*$ and compute aP, bP, cP, $v = e(P,P)^{abc}$. The secret key is (a, b, c) and make aP, bP, cP, v public.

  Protocol actions between $\mathcal{P}$ and $\mathcal{V}$: This scheme consists of several rounds, each of which is performed as follows :

  1. $\mathcal{P}$ chooses randomly $r_1, r_2, r_3 \in Z_q^*$ and computes $x = e(P,P)^{r_1 r_2 r_3}$, $Q_1 = r_1 P$, $Q_2 = r_2 P$ and $Q_3 = r_3 P$ and sends $\langle x, Q_1, Q_2, Q_3 \rangle$ to $\mathcal{V}$.
  2. $\mathcal{V}$ picked $w \in Z_q^*$ at random and sends w to $\mathcal{P}$.
  3. $\mathcal{P}$ computes $y = e(wP, P)^{abc}\ e(P,P)^{r_1 r_2 r_3}$ and sends to $\mathcal{V}$; $\mathcal{V}$ accepts if $y = v^w\ x$ and rejects otherwise.

- Assumption : Existence of GDH group.
- Security :
  Secure against active attacks assuming that the underlying group is a GDH group.
- Efficiency :

KeyGen : 3 scalar multiplications in G_1; 1 pairing computations.

Protocol actions between $\mathcal{P}$ *and* $\mathcal{V}$: 3 pairing computations and 4 scalar multiplications in G_1 for $\mathcal{P}$; 1 exponentiation in G_2 and 1 multiplication in G_2 for $\mathcal{V}$.

A.6.5 Other Signature Schemes

There are a large number of cryptographic protocols that uses pairings. Discussing every protocol is beyond the scope of the paper. This subsection includes a list of few other interesting signature schemes that have various cryptographic applications in digital world.

1. Optimistic Fair Exchange [50].
2. Non-Interactive Deniable Ring Authentication [116].
3. Another Varifiably Encrypted Signature [122].
4. Partially Blind Signature [122].
5. ID-Based Group Signature [44].
6. Delegation-By-Certificate Proxy Signature [19].
7. Hierarchical ID-Based Signatures (HIDS) [60].

We refer Barreto's Pairing-Based Crypto Lounge [7] for an overview (references) of recent developments in cryptosystems based on pairings.

A.7 Conclusion

Several cryptographic primitives using pairings have been described in this survey. Some others have been left out, mainly due to the non-availability of proper security proofs. The area is still growing and almost each conference proceedings include some new proposals. On the other hand, we have covered the basic schemes which will continue to be referred in the future. Thus we believe that our survey will provide both an introduction to the area as well as serve as a ready reference to the area in the next few years.