A DYNAMIC ASYMMETRIC KEY AGREEMENT FOR BROADCAST ENCRYPTION BASED ON BRAID GROUPS

Norranut Saguansakdiyotin

A Dissertation Submitted in Partial
Fulfillment of the Requirements for the Degree of
Doctor of Philosophy (Computer Science)
School of Applied Statistics
National Institute of Development Administration
2012

A DYNAMIC ASYMMETRIC KEY AGREEMENT FOR BROADCAST ENCRYPTION BASED ON BRAID GROUPS

Norranut Saguansakdiyotin

School of Applied Statistics

Associate Professor.....P. Hiranvanichakorn.....Advisor

(Pipat Hiranvanichakorn, D.E.)

The Examining Committee Approved This Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science).

Assistant Professor.....Committee Chairperson

(Ohm Sornil, Ph.D.)

Associate Professor.....P. Hiranvanichakorn.....Committee

(Pipat Hiranvanichakorn, D.E.)

Assistant Professor.....Committee

(Pramote Kuacharoen, Ph.D.)

Kanchana Silawarawet.....Committee

(Kanchana Silawarawet, Ph.D.)

Associate Professor.....Acting Dean

(Raweewan Auepanwiriyakul, Ph.D)

November 2012

ABSTRACT

Title of Dissertation	A Dynamic Asymmetric Key Agreement for Broadcast Encryption Based on Braid Groups
Author	Norranut Saguansakdiyotin
Degree	Doctor of Philosophy (Computer Science)
Year	2012

Broadcast encryption is the scheme that a sender encrypts messages for a designated group of receivers, and sends the ciphertexts by broadcasting over the networks. Many research papers have done it using elliptic curve cryptosystem. This research proposes a dynamic asymmetric key agreement protocols for the broadcast encryption which is based on braid groups cryptosystem, an alternative method in the public key cryptography in which it can reduce the computational cost. The proposed scheme is also used the concept of an identity-based cryptosystem in order to reduce a system complexity and cost for establishing and managing a public key authentication framework. The scheme has some advantages over the other scheme in that the proposed scheme does not require a center for group key management, thus it is suitable for dynamic networks like mobile ad hoc networks in which group members can be dynamically changed all the time. The scheme also has a low computation cost in encryption and decryption processes and it makes a constant ciphertext.

ACKNOWLEDGEMENTS

First of all, I would like to express my very great appriciation to my advisor, Associate Professor Dr. Pipat Hiranvanichakorn for providing me with very valuable advice in this research dissertation. I also would like to extend my thanks to all of the committee members; Assistant Professor Dr. Pramote Kuacharoen, Assistant Professor Dr. Ohm Sornil and Dr. Kanchana Silawarawet for their comments and suggestions. Finally thanks are for my wife for supporting me during my study.

Norranut Saguansakdiyotin

November 2012

TABLE OF CONTENTS

LIST OF TABLES

Tables **Page**

LIST OF FIGURES

Figures **Page**

CHAPTER 1

INTRODUCTION

1.1 Background

Broadcast encryption is a scheme that allows a sender to send a ciphertext to some designated groups whose members of the group can decrypt it with his or her private key. However, nobody outside the group can decrypt the message. Broadcast encryption is widely used in the present day in many aspects, such as VoIP, TV subscription services over the Internet, communication among group members or from someone outside the group to the group members. This type of scheme also can be extended in networks like mobile multi-hop networks, which each node in these networks has limitation in computing and storage resources. The broadcast encryption can be divided into two categories from a relation of receivers. In the first category, a sender can randomly designate several receivers. Users in this category may be no relation between each other. For the second category, a sender can encrypt a message to a designated group in which each user in the group can use his private key independently to decrypt the ciphertext. Users can contact with other users in the group and all users in the group are listening on a broadcast channel. Usually the first category has lots of advantages. It is more flexible than the second category and sender can randomly designated a subset of receivers**.** However, these advantages make the first category much more complicated. It is very difficult to make the scheme satisfy so many advantages while keep the ciphertext and keys constant size. For a network like a mobile ad hoc network, the complex in computation and the need for large memory make it inefficient. The proposed scheme for broadcast encryption is in the second category.

There are several group key management protocols, and typically they are divided into three categories; centralized group key distribution, decentralized group

key management, and contributory group key agreement. In the centralized group key distribution, there is a group controller. This group controller is responsible for distributing a group key. The advantages in this category are that it minimizes storage requirements, computational power on both client and server sides, and bandwidth utilization. There exist some drawbacks in this category such as the performance bottleneck, the central point of failure, and the requirement of trusted authority. In the decentralized group key management, a group is divided into several subgroups. Each subgroup has a subgroup controller and it is responsible for key management. There exist the same problems as in the centralized group key distribution. The last category is the contributory group key agreement. In this category, a shared group key is generated via the cooperation of all members and there is no central management, so it is more suitable for ad hoc networks and a group which has small group members.

1.2 Related Works

Fiat and Naor (1993) proposed the concept of broadcast encryption. In this scheme, sender allows to send a ciphertext to a designated group whose members of the group can decrypt it with his or her private key. However, nobody outside the group can decrypt the message. Their solution was secure against m collusion users and the length of the ciphertext is $O(m\log^2 m\log n)$, where n is the number of users and m is the number of colluders. Further research by Naor, Naor, and Lotspiech (2001) proposed a solution with ciphertext and keys do not rely on the m. The scheme has private key size of $O(\log^2 n)$. There also are many research papers about broadcast group-oriented encryption as in Ma, Wu, and Li (2006) and Ma and Ao (2009). The former proposes a novel broadcast encryption used in the group communication. It is an asymmetric group key agreement scheme achieved a broadcast message with constant ciphertexts and private keys. The later proposes the improved version by including the identity of users to the previous scheme, and it is secure against chosen ciphertext attack and the key generation withstands collude attack from the users of the group. Both schemes are the centralized group key agreement schemes, because they need a private key generator for generating member's private keys. These

schemes as mentioned before are implemented using bilinear pairing in elliptic curve cryptosystem.

There are also some research papers in doing asymmetric group key agreement as in Wu, Mu, Susilo, Qin, and Domingo-Ferrer (2009) and Zhao, Zhang and Tian (2011). Both schemes are the contributory group key agreement schemes. The former scheme is constructed on one round asymmetric group key agreement (ASGKA) based on the concept of aggregatable signature based broadcast (ASBB) by using bilinear pairings. Typically in an ASGKA protocol, it has two keys; one is a public group key, which is used as an encryption key for a message to a group and another is a private key, which a group member can use it individually as a decryption key. An ASBB is the scheme that the public key can be used to verify signatures as well as to encrypt messages, and any valid signature can be used to decrypt the ciphertexts. As mentioned in Wu, Mu, Susilo, Qin, and Domingo-Ferrer (2009), their scheme does not improve in communication overhead for one-time group applications in which the members of the group are about fully dynamic as in ad hoc networks, because their scheme has heavy communication overhead in key establishment. The later scheme is a dynamic asymmetric group key agreement (DASGKA) combining a conventional authenticated group key agreement, a public key encryption and a multi-signature. This scheme is implemented using elliptic curve cryptosystem. In the dynamic asymmetric group key agreement, a group of users can form a temporary group and agree to share a public encryption key. Users can join or leave the group without running a completely new key agreement protocol.

The concept of an identity-based cryptosystem was proposed by Shamir (1984). An idea of this new paradigm is to use user's identifier information such as email address or IP address as a public key for encryption or signature verification. An identity-based cryptosystem can reduce a system complexity and cost for establishing and managing a public key authentication framework known as the public key infrastructure (PKI). The first identity-based cryptosystem scheme which was proposed by Shamir is an identity-based signature scheme (IBS). An identity-based encryption scheme (IBE) became an opening problem until 2001. In 2001, Shamir's open problem was solved by Boneh and Franklin (2001) as well as Cook (2001). An identity-based cryptosystem has mostly been implemented using bilinear pairing which has exponential cost in pairing operation.

1.3 Motivation

Braid groups were introduced by Artin (1947) and first used to construct a Diffie-Hellman type key agreement protocol and a public key encryption scheme by Ko , Lee , Cheon , Han , Kang , Park (2000). In Karu and Loikkanen work (2000), the comparison of fast public key cryptosystems; elliptic curve cryptography (ECC), NTRU, and braid groups has been made. The result shown that the braid groups based efficient cryptosystem can be implemented, and it is faster than RSA and elliptic curve cryptography. For very limited environments like personal digital assistant (PDA)'s, smart cards, and mobile phones they require faster cryptosystem, therefore braid groups based cryptosystem can be one of the choices. The braid groups can be used in a symmetric group key agreement protocol as in Thanongsak Aneksrup and Pipat Hiranvanichakorn (2011). The motivation for this research is to create an asymmetric broadcast encryption scheme based on an identity-based cryptosystem and braid group concepts in contributory group key agreement manner. The proposed scheme starts with building an asymmetric broadcast encryption scheme based on braid groups and then applies the identity-based cryptosystem to it.

CHAPTER 2

LITERATURE REVIEW

This chapter gives an overview of the concepts relating in the proposed scheme on a dynamic asymmetric key agreement for broadcast encryption such as the concept of broadcast encryption, identity-based cryptosystem, braid groups based cryptosystem, some related researches in asymmetric group key agreement scheme for broadcast encryption, and identity-based broadcast encryption.

2.1 Broadcast Encryption

Fiat and Naor (1993) proposed the concept of broadcast encryption in 1993. In this scheme, sender allows to send a ciphertext to a designated group whose members of the group can decrypt it with his or her private key. However, nobody outside the group can decrypt the message. Broadcast encryption is widely used in the present day in many aspects, such as VoIP, TV subscription services over the Internet, communication among group members or from someone outside the group to the group members. This type of scheme also can be extended in networks like mobile multi-hop networks, which each node in these networks has limitation in computing and storage resources.

The original scheme which is proposed by Fiat and Naor was to prove that two devices which were not known each other could agree on a common key for secure communications over a one-way communication. This is different from a traditional secure transmission of information using public key cryptography in which devices must know about each other and agree on encryption keys before transmission. Broadcast encryption allows devices which may not have existed, when a group was firstly formed, can join into the group and communicate securely.

In the original broadcast encryption scheme proposed by Fiat and Naor, there exists a key distribution center. The center allocates predefined keys for all of the

users in a group. It was also a zero-message scheme in which the broadcast center did not have to broadcast a message for the members to be able to compute the key. It could be computed from information that the members receives from the center, and from other members. The scheme is a *k*-resilient broadcast encryption scheme in which it is secure against a coalition of at most *k* non-privileged users.

There are many research papers about broadcast group-oriented encryption as in Ma, Wu, and Li (2006) and Ma and Ao (2009). The former proposes a novel broadcast encryption used in the group communication. It is an asymmetric group key agreement scheme achieved a broadcast message with constant ciphertexts and private keys. The later proposes the improved version by including the identity of users to the previous scheme, and it is secure against chosen ciphertext attack and the key generation withstands collude attack from the users of the group. Because a member's identity is included in a private key generation, two or more members cannot forge a new private key to the other. The review of these papers can be found in related works section in this chapter.

2.2 Identity-Based Cryptosystem

The concept of an identity-based cryptosystem was proposed by Shamir (1984). There are two schemes in an identity-based cryptosystem. The first one is an identity-based encryption scheme (IBE) and the second one is an identity-based signature scheme (IBS). This section reviews a concept of both the identity-based encryption scheme (IBE) and the identity-based signature scheme (IBS). It also gives an example of an identity-based cryptosystem using bilinear paring.

2.2.1 Basic Concept of Identity-Based Encryption and Signature

This subsection describes the concept of identity-based encryption (IBE) and identity-based Signature (IBS) schemes from Baek, Newmarch, Safavi-Naini and Susilo (2004) paper.

2.2.1.1 Identity-Based Encryption

In an identity-based encryption (IBE) scheme, if Alice needs to send a ciphertext to Bob, she can encrypt it using Bob's identity information such as his

email address or his IP address as well as a public key of a trusted third party called a private key generator. Upon receiving the ciphertext, Bob can decrypt it using his private key which is associated with Bob's identity and generated by the private key generator as shown in Figure 2.1.

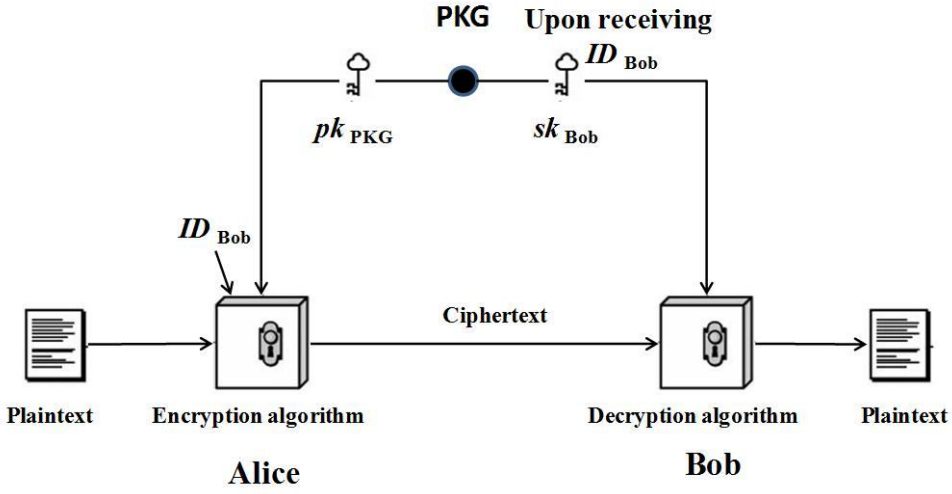

Figure 2.1 Identity-Based Encryption

We can describe an identity-based encryption scheme in four steps; setup, private key extraction, encryption and decryption as the following.

1) Setup: The private key generator creates its private and public key pair denoted as sk_{PKG} and pk_{PKG} respectively.

2) Private Key Extraction: The receiver Bob authenticates himself to the private key generator and obtains his private key sk_{Bob} which is associated with his identity ID_{Bob}.

3) Encryption: Alice uses the receiver Bob's identity ID_{Bob} and the public key of the private key generator pk_{PKG} to encrypt a message.

4) Decryption: the receiver, Bob, uses his private key sk_{Bob} to decrypt the ciphertext.

2.2.1.2 Identity-Based Signature

In an identity-based signature (IBS) scheme, If Alice wants to sign a message to Bob, she can sign it using her private key which is associated with her

identifier information and obtained from a private key generator. Upon receiving the message Bob can verify it using Alice's identifier information as well as a public key of the private key generator as shown in Figure 2.2.

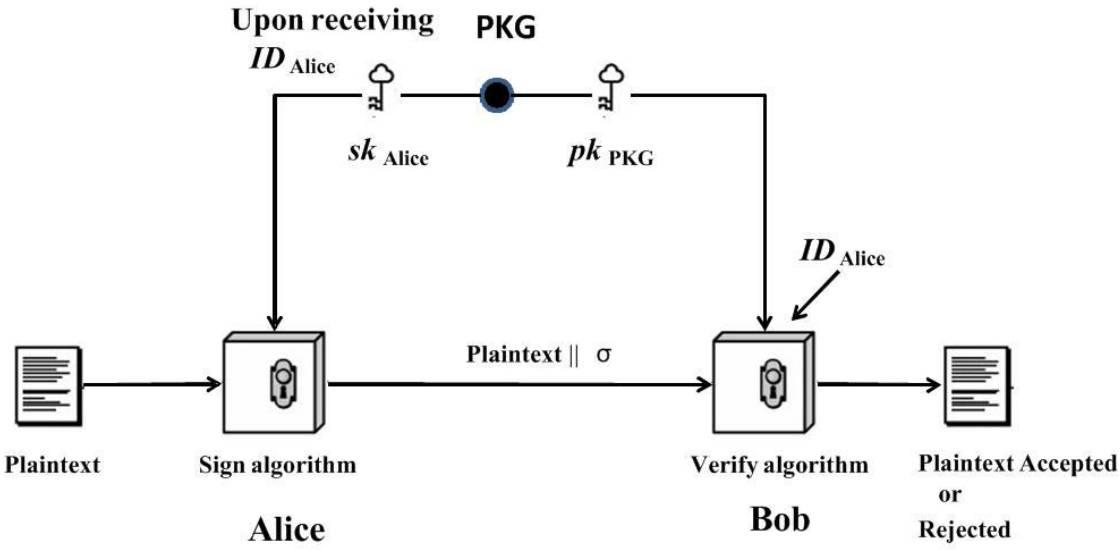

Figure 2.2 Identity-Based Signature

We can describe an identity-based signature scheme in four steps; setup, private key extraction, encryption and decryption as the following.

1) Setup: The private key generator creates its private and public key pair denoted as sk_{PKG} and pk_{PKG} respectively.

2) Private Key Extraction: The signer, Alice, authenticates herself to the private key generator and obtains her private key sk_{Alice} which is associated with her identity ID_{Alice}.

3) Signature Generation: Alice uses her private key sk_{Alice} to generate signature σ and send it with a message.

4) Signature Verification: the verifier, Bob, checks whether σ is a genuine signature on the message using Alice's identity and the private key generator's public key pk_{PKG}. If it is, then accepts the message. Otherwise rejects the message.

2.2.2 Bilinear Pairing

This subsection gives some preliminaries of bilinear pairing and its properties. Let G and G_τ be two groups of prime order q and let P be a generator of G, where G is additively represented and G_τ is multiplicatively. A map e: $G \times G \rightarrow G_\tau$ is said to be a bilinear pairing and the group G is called a bilinear group, if the following three properties hold:

2.2.2.1 Bilinearity: $e(aP, bP) = e(P, P)^{ab} = e(P, abP) = e(abP, P)$ for all a, b $\in Z_q^*$;

2.2.2.2 Non-degeneracy: this means that there exists $P \in G$ such that $e(P, P) \neq 1$ where 1 is an identity of G_τ;

2.2.2.3 Computability: this means that there exists an efficient algorithm to compute $e(P, P)$ $\forall$ $P \in G$.

2.2.3 Related Complexity Assumptions

Consider the following problems in the group G_1 of prime order q, generated by P.

2.2.3.1 The Decisional Bilinear Diffie-Hellman problem (DBDHP) is, given a generator P of a group G_1, (aP, bP, cP) and an element $h \in G_2$, to decide whether $h = e(P, P)^{abc}$.

2.2.3.2 Given a generator P of a group G_1 and (aP, bP), the Computational Diffie-Hellman problem (CDHP) is to compute abP.

2.2.4 An Example of Identity-Based Encryption

This subsection gives an example of an identity-based encryption scheme using pairings on elliptic curves. This overview comes from Hoffstein, Pipher, and Silverman (2008). When Bob wants to send Alice a message, he uses private key generator's public key pk_{PKG} and Alice's identity information ID_{Alice} to encrypt his message. In the meantime, private key generator uses its private key sk_{PKG} and Alice's identity information ID_{Alice} to create a private key of Alice sk_{Alice}. Alice then uses sk_{Alice} to decrypt and read Bob's message.

This example is explained in four steps; setup, private key extraction, encryption and decryption as the following.

2.2.4.1 Setup: In this step the private key generator prepares public parameters and creates its private and public keys. There are some public parameters such as; a finite field F_q, an elliptic curve E, a point on the elliptic curve $P \in E(F_q)$ of prime order l. There are also two hash functions H_1 and H_2 where H_1: $\{ID_s\} \rightarrow E(F_q)$ and H_2: $F_q^* \rightarrow \{0,1\}^B$. The first hash function maps a user identity to a point on E. The second hash function maps each element in F_q^* to a binary string of length B.

The private key generator chooses a secret integer s modulo m and publishes the point $P_{PKG} = sP \in E(F_q)$. The P_{PKG} is a public key of private key generator and s is its private key.

2.2.4.2 Private Key Extraction: Alice chooses identity information. The private key generator computes the point $P_{Alice} = H_1(ID_{Alice}) \in E(F_q)$ where P_{Alice} is Alice's public key. The private key generator sends the point $Q_{Alice} = s\ P_{Alice} \in E(F_q)$ to Alice. This Q_{Alice} is Alice's private key.

2.2.4.3 Encryption: Bob chooses a plaintext M and a random number r modulo q-1. Bob computes the point $P_{Alice} = H_1(ID_{Alice}) \in E(F_q)$. Bob's ciphertext is the pair $C = (rP, M \oplus H_2(\hat{e}_l(P_{Alice}, P_{PKG})^r))$.

2.2.4.4 Decryption: Alice decrypts the ciphertext (C_1, C_2) by computing $C_2 \oplus H_2(\hat{e}l(Q_{Alice}, C_1))$.

The correctness of this scheme can be shown as the following.

$$\hat{e}_l(Q_{Alice}, C_1) = \hat{e}_l(sP_{Alice}, rP) = \hat{e}_l(P_{Alice}, P)^{rs}$$
$$= \hat{e}_l(P_{Alice}, sP)^r = \hat{e}_l(P_{Alice}, P_{PKG})^r$$

2.3 Braid Groups

The braid group was first introduced by Artin (1947). It is a "non-commutative" group which can be used in cryptography because its computations can be performed efficiently, but it is strong enough against attacks. For the geometric presentation of the braid group, a braid B_n is a set of disjoint n strands all of which are attached to two horizontal bars at the top and the bottom, and between the top and the bottom bars, one strand crosses any one horizontal line only once. We call n is the braid index. A braid can be represented by a sequence of generator σ_n which is called the Artin

generator as proposed by Artin. If the strand i^{th} passes under the strand $i+1^{th}$, it denotes σ_i. Corresponding if the strand i^{th} passes over the strand $i+1^{th}$ it denotes σ_i^{-1} as shown in Figure 2.3. The multiplication of two braids with the same braid index, xy comes from concatenating the ends of the strands of the first braid with the beginnings of the strands of the second braid, e.g., $x = \sigma_1^{-1}$and $y = \sigma_2\sigma_1$, so $xy = \sigma_1^{-1}\sigma_2\sigma_1$ as shown in Figure 2.4. The identity braid is the braid consisting of strands with no crossings as shown in Figure 2.5. The inverse of a braid is the mirror image of that braid with respect to the horizontal line, e.g. from the previous example, $y^{-1} = (\sigma_2\sigma_1)^{-1} = \sigma_1^{-1}\sigma_2^{-1}$.

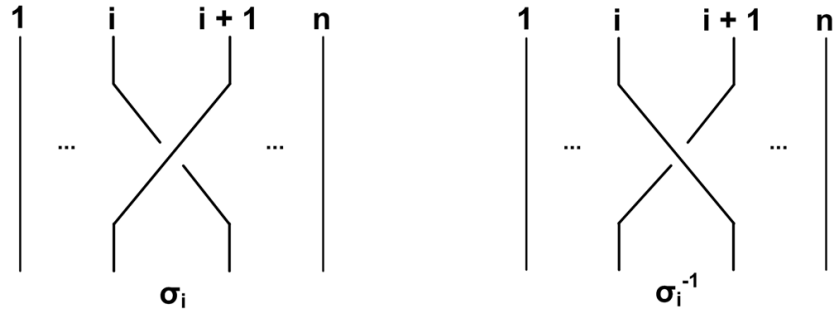

Figure 2.3 Artin Generator of σ_i and σ_i^{-1}

As we have seen that any braid B_n can be expressed as a braid word which is a sequence of generator, e.g., $\sigma_1^{-1}\sigma_2\sigma_1$, and it has the following relation;

(1) $\sigma_i\sigma_j = \sigma_j\sigma_i$ where $|i-j| > 1$
e.g., $\sigma_1\sigma_3 = \sigma_3\sigma_1$

(2) $\sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j$ where $|i-j| = 1$
e.g., $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$

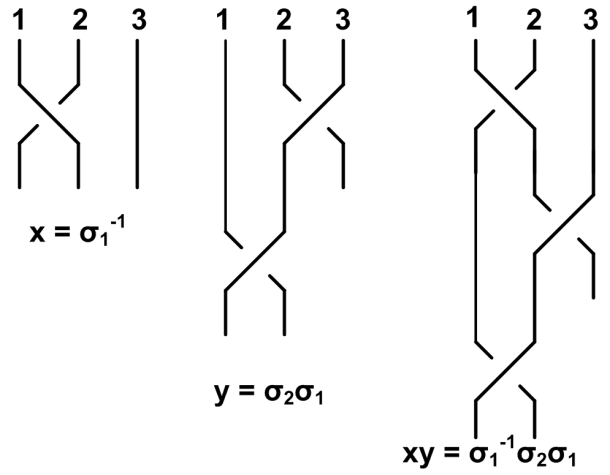

Figure 2.4 Artin Generator for Some Braids

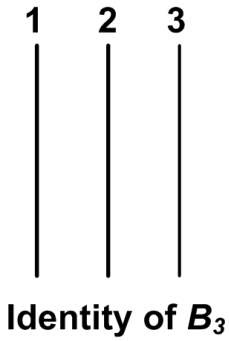

Figure 2.5 Identity Braid

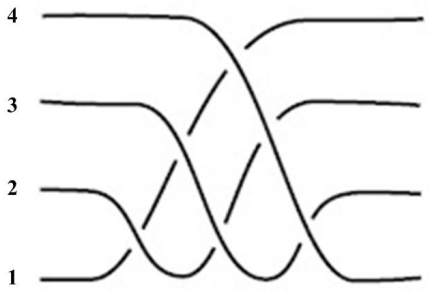

Figure 2.6 Fundamental Braid of B_4

2.3.1 Properties of Braid Groups

This subsection reviews some notations as well as properties of braid groups as mentioned in Cha, Ko, Lee, Han and Cheon (2001).

2.3.1.1 Elements in B_n^+ are called positive braids or positive words in which every generator in the braid word has no negative power.

2.3.1.2 There is a fundamental braid D or Δ where $\Delta = (\sigma_1 \ldots \sigma_{n-1})(\sigma_1 \ldots \sigma_{n-2}) \ldots \sigma_1$ in the Artin presentation as shown in Figure 2.6. In the fundamental braid, each pair of strand crosses exactly once. The fundamental braid D can be written in many different ways as a positive word. It has two properties:

1) For each generator a, $D = aA = Ba$ for some $A, B \in B_n^+$.
2) For each generator a, $aD = DT(a)$ and $Da = T^{-1}(a)D$ where

T (tau) is the automorphism of B_n defined by $T(i) = \sigma_{i-1}$ for Artin presentation .

2.3.1.3 There are partial orders '$\leq$', '$\leq_L$' and '$\leq_R$' in B_n. For two words V and W in B_n, we say that $V \geq W$ (respectively $V \geq_L W$, $V \geq_R W$) if $V = PWQ$ (resp. $V = WP$, $V = PW$) for some $P,Q \in B_n^+$. If a word is compared against either an empty word e or a power of D, all three orders are equivalent due to the property 2) above. Note that the partial orders depend on a presentation of B_n and W is a positive word if and only if $W \geq e$.

2.3.1.4 For two elements V and W in a partial order set, the *meet* $V \wedge W$ (resp. join $V \vee W$) denotes the largest (resp. smallest) element among all elements smaller (resp. larger) than V and W. When we want to distinguish the meet and join for left and right versions, we will use '$\wedge_L$', '$\wedge_R$', '$\vee_L$' and '$\vee_L$'.

2.3.1.5 A braid satisfying $e \leq A \leq D$ is called a canonical factor and $[0, 1]_n$ denotes the set of all canonical factors in B_n. The cardinality of $[0, 1]_n$ is $n!$ for the Artin presentation.

2.3.1.6 For a positive braid P, a decomposition $P = A_0P_0$ is left-weighted if $A_0 \in [0, 1]_n$, $P_0 \geq e$, and A_0 has the maximal length (or maximal in '$\leq_L$') among all such decompositions. A left-weighted decomposition $P = A_0P_0$ is unique. A_0 is called the maximal head of P. The notion 'right-weighted' can be also defined similarly.

2.3.1.7 Any braid W given as a word can be decomposed uniquely into $W = D^u A_1A_2 \ldots A_k$, where $e < A_i < D$, $u \in Z$, where the decomposition A_iA_{i+1} is left-

weighted for each $1 \le i \le k$ - 1. This decomposition, called the left canonical form of W, is unique and so it solves the word problem. The right canonical form of W can be also defined similarly.

2.3.2 Operations in Braid Groups

This subsection give an overview of braid groups operations as mentioned in the paper entitled "An Efficient Implementation of Braid Groups" by Cha, Ko, Lee, Han, and Cheon (2001). First they write a given braid in the form as $\beta = D^q A_1 A_2 \ldots A_l$, where q is an integer and each A_i is a canonical factor, they represent the braid as a pair $\beta = (q, (A_i))$ of an integer q and a list of l canonical factors (A_i). They note that this representation is not necessarily the left canonical form of β, and hence l may be greater than the canonical length of β.

A braid given as a word in generators can converted into the above form by rewriting each negative power σ^{-1} of generators as a product of D^{-1} and a canonical factor $D\sigma^{-1}$ and collecting every power of D at the left end using the fact $(\prod A_i)D^{+/-1} = D^{+/-1}(\prod T^{+/-1}(A_i))$ for any sequence of canonical factors A_i.

2.3.2.1 Multiplication

The multiplication of two braids is just the juxtaposition of two lists of permutation and applying τ as the following equation.

$(D^p A_1 \ldots A_l)(D^q B_1 \ldots B_{l'}) = D^{p+q}\, T^q(A_1) \ldots T^q(A_l) B_1 \ldots B_{l'}$

The inverse of a braid can be computed using the following equation.

$(D^q A_1 \ldots A_l)^{-1} = D^{-q-l}\, T^{-q-l}(B_l) \ldots T^{-q-1}(B_1)$ where $B_i = A_i^{-1} D$, viewing A_i and D as permutations.

2.3.2.2 Left Canonical Form

Given a positive braid $P = A_1 \ldots A_l$, where A_i is a canonical factor, the algorithm computes the maximal heads of $A_{l-1}A_l$, $A_{l-2}A_{l-1}A_l$, …, $A_1 \ldots A_l = P$ sequentially using the following facts.

1) For any positive braid A and P, the maximal head of AP is the maximal head of the product of A and the maximal head of P.

2) For two canonical factors A and B, the maximal head of AB is $A((DA^{-1}) \wedge_L B)$, where the inverse is taken in the permutation group.

2.3.2.3 Left Canonical Form by Hands

Given a positive braid *P*, it can be easy to use visual approach suggested by ElriFai and Morton (1994) to convert to its left canonical form as the following. We can partition a braid words into permutation braids (or canonical braids) by scanning a braid words from the highest to the lowest braid words and partition it if a pair of strings are about to cross for the second time, and then start a new permutation braid. Then look at adjacent pairs of permutation braids, and see if any adjacent pair of strings crosses in the lower but not the upper braid. If so, move it up and continue, otherwise stop.

2.3.3 Example in Braid Groups Operation

This subsection gives an example of how to convert a braid word into the left canonical form by hands. For example a braid word $\sigma_3\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_1$ in B_4 can be converted into the left canonical form as the following steps.

First change a negative braid into a positive braid, for this example, change σ_1^{-1} and σ_3^{-1} to positive braid word as the following.

$\sigma_1^{-1} = D^{-1}D\sigma_1^{-1} = D^{-1}(\sigma_1\sigma_2\sigma_3)(\sigma_1\sigma_2)(\sigma_1)\sigma_1^{-1} = D^{-1}(\sigma_1\sigma_2\sigma_3)(\sigma_1\sigma_2)$ and

$\sigma_3^{-1} = D^{-1}D\sigma_3^{-1} = D^{-1}(\sigma_1\sigma_2\sigma_3)(\sigma_1\sigma_2)(\sigma_1)\sigma_3^{-1} = D^{-1}(\sigma_1\sigma_2\sigma_3)(\sigma_2\sigma_1)(\sigma_2)\sigma_3^{-1}$

$= D^{-1}(\sigma_1\sigma_3\sigma_2)(\sigma_3\sigma_1)(\sigma_2)\sigma_3^{-1} = D^{-1}(\sigma_1\sigma_3\sigma_2)(\sigma_1\sigma_3)(\sigma_2)\sigma_3^{-1}$

$= D^{-1}(\sigma_3\sigma_1\sigma_2)(\sigma_1\sigma_3)(\sigma_2)\sigma_3^{-1} = D^{-1}(\sigma_3\sigma_2\sigma_1)(\sigma_2\sigma_3)(\sigma_2)\sigma_3^{-1}$

$= D^{-1}(\sigma_3\sigma_2\sigma_1)(\sigma_3\sigma_2)(\sigma_3)\sigma_3^{-1}$

$= D^{-1}(\sigma_3\sigma_2\sigma_1)(\sigma_3\sigma_2)$

Thus $\sigma_3\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_1 = \sigma_3\sigma_2D^{-1}(\sigma_1\sigma_2\sigma_3)(\sigma_1\sigma_2)D^{-1}(\sigma_3\sigma_2\sigma_1)(\sigma_3\sigma_2)\sigma_2\sigma_1$, Now use the fact $(\prod A_i)D^{+/-1} = D^{+/-1}(\prod T^{+/-1}(A_i))$ for any sequence of canonical factors A_i

$\sigma_3\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_1 = (\sigma_3\sigma_2D^{-1})(\sigma_1\sigma_2\sigma_3\sigma_1\sigma_2D^{-1})\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-1}T(\sigma_3\sigma_2)\ D^{-1}T(\sigma_1\sigma_2\sigma_3\sigma_1\sigma_2)\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-1}(\sigma_1\sigma_2)\ D^{-1}T(\sigma_1\sigma_2\sigma_3\sigma_1\sigma_2)\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-1}(\sigma_1\sigma_2)\ D^{-1}(\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2)\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-1}(\sigma_1\sigma_2D^{-1})\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-1}D^{-1}T(\sigma_1\sigma_2)\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-2}T(\sigma_1\sigma_2)\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-2}(\sigma_3\sigma_2)\ \sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2\sigma_1$

$= D^{-2}(\sigma_3\sigma_2)\ (\sigma_3\sigma_2\sigma_1)\ (\sigma_3\sigma_2)\ (\sigma_3\sigma_2\sigma_1)\ (\sigma_3\sigma_2)\ (\sigma_2\sigma_1)$

From this point, we can use algorithm proposed by ElriFai and Morton (1994) in subsection 2.3.2.3 to partition the braid words above into permutation braids as the following and Figure 2.7.

$D^{-2}(\sigma_3\sigma_2)\ (\sigma_3\sigma_2\sigma_1)\ (\sigma_3\sigma_2)\ (\sigma_3\sigma_2\sigma_1)\ (\sigma_3\sigma_2)\ (\sigma_2\sigma_1)$

$= D^{-2}(\sigma_3\sigma_2\sigma_3)(\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

Use algorithm proposed by ElriFai and Morton (1994) again to move up an adjacent pairs of permutation braids, which no crossing in the above as in Figure 2.7 and we got the sequence of a permutation braids as the following.

$D^{-2}(\sigma_3\sigma_2\sigma_3)(\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

$= D^{-2}(\sigma_3\sigma_2\sigma_3\sigma_1)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

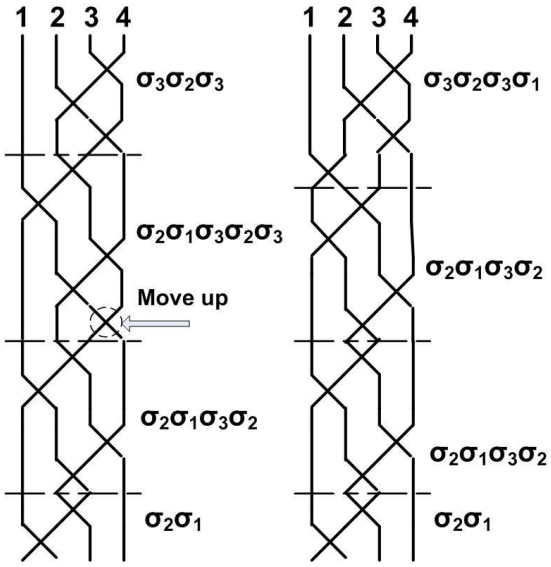

Figure 2.7 An Example of Permutation Braids

Continue with the braid words above, we can find the left canonical form as the following.

$D^{-2}(\sigma_3\sigma_2\sigma_3\sigma_1)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

$= D^{-2}(\sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_1)(\sigma_1\sigma_2)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

$= D^{-2}D(\sigma_1\sigma_2)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

$= D^{-1}(\sigma_1\sigma_2)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

The left canonical form of the braid word $\sigma_3\sigma_2\sigma_1^{-1}\sigma_3^{-1}\sigma_2\sigma_1$ is $D^{-1}(\sigma_1\sigma_2)(\sigma_2\sigma_1\sigma_3\sigma_2)(\sigma_2\sigma_1)$

2.3.4 Implementation

There have to be efficient ways to represent and perform operations on braid groups in a computer in order to do cryptography based on them. Because a braid has a unique decomposition, this decomposition provides a nice way to store a particular braid in the memory of a computer. In order to store a braid in a computer's memory, the power of the fundamental braid u as in 2.3.1.7 must be stored, as well as the sequence of canonical factors following it (A_i). The power of the fundamental braid can be stored as an integer. The canonical factors are the permutation braids. Thus to store a canonical factor, an array representing the permutation can be used. Let A be an array representing a permutation table. If a permutation sends i to $\pi(i)$, then $A(i) = \pi(i)$. For example braid $y \in B_3$ in Figure 2.4 can be written in permutation as $\pi(2,3,1)$.

For example braid words x and $y \in B_3$ in Figure 2.4, when they are putted into a program, they can be kept in memory as the sign integer like x = 10000001 and y = 0000001000000001. When the program wants to multiply them together, it can read and then concatenate them as a braid word "-1 2 1". Then the program performs a left-canonical form operation. The result is as the following.

$x = \sigma_1^{-1} = D^{-1}D\sigma_1^{-1} = D^{-1}(\sigma_1\sigma_2)(\sigma_1)\sigma_1^{-1} = D^{-1}(\sigma_1\sigma_2)$

$y = \sigma_2\sigma_1$

$xy = D^{-1}(\sigma_1\sigma_2)\ \sigma_2\sigma_1$

Then keeps $u = -1$, $A_1 = (\sigma_1\sigma_2)$, and $A_2 = (\sigma_2\sigma_1)$ for the braid xy.

2.3.5 Hard Problems in Braid Groups

Recently there are some mathematically hard problems in braid groups as a candidate for cryptographic one way function, but the famous one is the generalized conjugacy search problem (GCSP) which we apply it in our protocol to maximize strength of the key. In the generalized conjugacy search problem, it states that x and y are conjugate if there exist an element a such that $y = a\ x\ a^{-1}$ for $m < n$, where B_m is a subgroup of B_n generated by $\sigma_1, \sigma_2, \ldots, \sigma_{m-1}$. The hardness in GCSP is as following;

Given a pair $(x,y) \in B_n \times B_n$ such that $y = a\ x\ a^{-1}$ for some $a \in B_m$

The objective is to find $b \in B_m$ such that $y = b\ x\ b^{-1}$ for $m \leq n$.

Thus we can conclude that x and y are conjugate. It is able to compute y easily when we are known both a and x, but needs exponential time to compute b from bxb^{-1} when known x and y.

2.3.6 Public key Cryptography Based on Braid Groups

This section gives an example in using the braid groups in public key cryptography. We show a simple example protocol, according to Dehornoy (2004).

Here we say that Alice needs to send message m_A to Bob. Alice uses her private key and Bob's public key to encrypt the message and Bob can use his private key and Alice's public key to decrypt it.

1) Alice computes the conjugate $p' = sps^{-1}$, s is Alice's private key and (p,p') is her public key;

2) Bob computes the conjugate $p'' = rpr^{-1}$, r is Bob's private key and (p, p'') is his public key;

3) Alice sends a ciphertext $m'' = m_A \oplus H(sp''s^{-1})$ together with her public key p' to Bob;

4) Bob computes

$$m_A = m'' \oplus H(r\ p'\ r^{-1})$$
$$= m'' \oplus H(r\ sps^{-1}\ r^{-1})$$
$$= m'' \oplus H(srpr^{-1}s^{-1})$$
$$= m'' \oplus H(s\ p''\ s^{-1})$$
$$= m_A$$

As shown above, braid r and s commutes with each other, thus $sr = rs$. This property of the braid groups is true when we carefully select the braids. Suppose we have n subgroups $B_{g1}, B_{g2}, B_{g3}, \ldots, B_{gn}$ of g-braid groups where $g = g_1 + g_2 + g_3 + \ldots + g_n$. For any braid $s_l \in B_{gl}$ and $s_m \in B_{gm}$ with $l \neq m$, It is true that $s_l\ s_m = s_m\ s_l$.

2.4 Asymmetric Group Key Agreement Schemes

This section gives an overview of related researches on broadcast encryption using asymmetric group key agreement such as proposed by 1) Ma and Ao (2009),

2) Wu, Mu, Susilo, Qin, and Domingo-Ferrer (2009) and 3) Zhao, Zhang and Tian (2011).

2.4.1 Improved Group-Oriented Encryption for Group Communication

This scheme was proposed by Ma and Ao (2009) and improved from the scheme proposed by Ma, Wu, and Li (2006). They improve the security of Ma et al.'s encryption scheme and make it withstand collude attack by using the identities of the users in designing the group–oriented encryption scheme. In this scheme a sender is allowed to encrypt a message using a group's public key. Any user in the group can independently decrypt the ciphertext using his private key.

This subsection describes the group oriented encryption scheme as the following. There are five phases in this scheme; Initialize, KeyGen, KeyVer, Encrypt, and Decrypt.

2.4.1.1 Initialize

Let G_1 be a cyclic multiplicative group generated by g, whose order is a prime q and G_2 be a cyclic multiplicative group of the same order q. A bilinear pairing is a map: e: $G_2 \times G_1 \rightarrow G_2$ which can be efficiently computed. There are three cryptographic hash functions: 1) H:$\{0,1\}^* \rightarrow Z_q$ 2) G: $G_2 \rightarrow \{0,1\}^l$ 3) H_1: $G_1 \rightarrow Z_q^*$. A private key generator chooses $a \in Z_q^*$ and $g_2 \in G_1$ uniformly at random, and computes $g_1=g^a$. The master private key is a, and the master public keys are (g_1,g_2).

2.4.1.2 KeyGen

The private key generator chooses $k \in Z_q^*$ uniformly at random for a group A, and then publishes $PK_A = g^k$ and $VK_A=g^{a^2k}$ as group A's public key. The member p_i's private key can be generated as follows:

1) The private key generator chooses $r_i \in Z_q^*$ uniformly at random.

2) Computes and outputs $d_{i1} = g^{H(ID_i)r_i} g_2^{ar_i}$, $d_{i2} = g^{ar_i}$, and $d_{i3} = g^{ak} g^{H(ID_i)r_i}$.

The member p_i's private key is $d_i = \{d_{i1}, d_{i2}, d_{i3}\}$, where ID_i denotes the identity of member p_i.

2.4.1.3 KeyVer

After receiving the private key distributed from the private key generator, the member p_i verifies the validity of the key by the following equation.

$e(d_{i3},g^a) = e(VK_A, g)e(g^{H(ID_i)}, d_{i2})$ or not

If above equation holds, p_i accepts the private keys, otherwise outputs ERROR message. They say the member p_i can verify the key because

$$e(d_{i3},g^a) = e(g^{ak}g^{H(ID_i)ri}, g^a)$$
$$= e(g,g^{a^2k})e(g^{H(ID_i)}, g^{ari})$$
$$= e(VK_A, g)e(g^{H(ID_i)}, d_{i2})$$

2.4.1.4 Encryption

In order to encrypt a message $M \in \{0,1\}^l$ for the group A, the sender first chooses $s \in Z_q^*$ uniformly at random, and computes the ciphertext

$c_1 = G(e(g_1,PK_A)^s) \oplus M$

$c_2 = g^s$

$c_3 = g_2^s$

$c_4 = H_1(g^{ks})$

$c_5 = g^{(s+h)^{-1}}$

The ciphertext for message M is $c = (c_1,c_2,c_3,c_4,c_5)$, where $h = H(c_1||c_2||c_3||c_4)$. The sender sends the ciphertext to all the members in the group A by broadcasting over the Internet.

2.4.1.5 Decrypt

After receiving the ciphertext $c = (c_1,c_2,c_3,c_4,c_5)$, a user $p_i \in$ A can decrypt it as follows:

1) Computes $T = e(c_2,d_{i3})e(c_3,d_{i2}) / e(c_2,d_{i1})$.
2) Computes $M = c_1 \oplus G(T)$.

The Decrypt is correct, because

$$T = e(c_2,d_{i3})e(c_3,d_{i2}) / e(c_2,d_{i1})$$
$$= e(g^s, g^{ak}g^{H(ID_i)ri})e(g_2^s, g^{ari}) / e(g^s, g^{H(ID_i)ri}g_2^{ari})$$
$$= e(g, g)^{aks}$$

Then p_i gets the message $M = c_1 \oplus G(T)$.

This scheme is an asymmetric group key agreement scheme. The scheme has advantage in that it can withstand adaptively chosen ciphertext and colluded attacks, but it has some drawbacks in computation complexity, and no supported protocol for dynamic group.

2.4.2 Asymmetric Group Key Agreement

This scheme was proposed by Wu, Mu, Susilo, Qin, and Domingo-Ferrer (2009). This scheme is an asymmetric group key agreement (ASGKA) protocol. They proposed a generic construction of one-round asymmetric group key agreement protocol based on a new primitive referred to as aggregatable signature based broadcast (ASBB), in which the public key can be simultaneously used to verify signatures and encrypt messages and signature can be used to decrypt ciphertext. A round means that each party sends one message and can broadcast simultaneously. This scheme was also implemented using bilinear pairings.

From a generic construction, to realize one-round ASGKA protocol, they need only to implement a secure ASBB scheme. They construct an ASBB scheme secure in the random oracle model using bilinear pairing techniques.

2.4.2.1 An Efficient ASBB Scheme

Let PairGen be an algorithm taking on input a security parameter 1^{λ}, and outputs a tuple $\Upsilon = (p, G, G_\tau, e)$ where G and G_τ have the same prime order p, and $e: G \times G \rightarrow G_\tau$ is an efficient non- degenerate bilinear map such that $e(g, g) \neq 1$ for any generator g of G, and for all $u, v \in Z$, it holds that $e(g^u, g^v) = e(g, g)^{uv}$.

1) Public parameters: Let $\Upsilon = (p, G, G_\tau, e) \leftarrow$ PairGen (1^{λ}), $G=\langle g \rangle$. Let $H:\{0,1\}^* \rightarrow G$ be a cryptographic hash function. The system parameters are $\pi = (\Upsilon, g, H)$.

2) Public/secret keys: Select at random $r \in Z_p^*$. $X \in G \setminus \{1\}$. Compute $R = g^{-r}$, $A = e(X, g)$. The public key is $pk = (R, A)$ and the secret key is $sk = (r, X)$.

3) Sign: The signature of any string $s \in \{0,1\}^*$ under the public key pk is $\sigma = X H(s)^r$.

4) Verify: Given a message-signature pair (s,σ), the verification equation is $e(\sigma,g)e(H(s),R) = A$. If the output is 1, it means that purported signature is valid.

5) Encryption: For a plaintext $m \in G_\tau$, randomly select $t \in Z_p^*$ and compute $c_1 = g^t$, $c_2 = R^t$, $c_3 = mA^t$.

6) Decryption: After receiving a ciphertext (c_1, c_2, c_3), anyone who has a valid message-signature pair (s,σ) can extract $m = c_3 / e(\sigma,c_1)e(H(s), c_2)$.

2.4.2.2 Concrete One-Round ASGKA Protocol

The following is a concrete one-round ASGKA protocol which using the instantiated ASBB scheme.

1) Public parameters generation: It is the same as ASBB scheme.

2) Group setup: Decide a group of players $P = \{U_1,\ldots, U_n\}$ Randomly select $h_i \in G$ for $i = 1,\ldots,n$. h_i can map to U_i in a natural way, e.g., according to the dictionary order in their binary representation.

3) Group key agreement: U_i randomly select $X_i \in G$, $r_i \in Z_p^*$ and publishes $\{\sigma_{i,j}, R_i, A_i\}_{i\neq j}$, where $\sigma_{i,j} = X_i h_j^{r_i}$, $R_i = g^{-r_i}$, $A_i = e(X_i, g)$.

4) Group encryption key derivation: The player share the same group encryption key(R,A); $R = \prod^n_{j=1} R_j = g^{-\Sigma^n_{j=1} r_j}$, $A = \prod^n_{j=1} A_j = e(\prod^n_{j=1} X_j, g)$.

5) Decryption key derivation: Using the private input (X_i, r_i) during the protocol execution phase, player U_i can calculate its secret decryption key from the public communication;

$$\sigma_i = X_i h_j^{r_i} \prod^n_{j=1,j\neq i} \sigma_{j,i} = \prod^n_{j=1} X_j h_i^{r_j} = (\prod^n_{j=1} X_j) h_i^{\Sigma^n_{j=1} r_j}.$$

6) Encryption: For a plaintext $m \in G_\tau$, anyone who knows the public parameters and the group encryption key can produce the ciphertext $c = (c_1, c_2, c_3)$, where $t \leftarrow Z_p$, $c_1 = g^t$, $c_2 = R^t$, *and* $c_3 = mA^t$.

7) Decryption: $e(\sigma_i,g)e(h_i, R) = A$, each player U_i can decrypt $m = c_3 / e(\sigma_i,c_1)e(h_i, c_2)$.

This scheme is an asymmetric group key agreement scheme. The scheme has advantage in that it uses an aggregatable signature based broadcast in

which a public key can be used to verify a message, but it also has some drawbacks in computation complexity, and no supported protocol for dynamic group.

2.4.3 Dynamic Asymmetric Group Key Agreement

This scheme was proposed by Zhao, Zhang and Tian (2011). They describe a dynamic asymmetric group key agreement (DASGKA) protocol. The protocol is not required central management. The protocol combines the concepts of a conventional group key agreement, a public key encryption and a multi-signature. Their construction is similar to an authenticated group key agreement for dynamic group. After a shared private key is computed, a corresponding public key is published to outsiders. In order for outsiders to trust the public key, a multi-signature is attached. Their scheme is an asymmetric group key agreement, and it is a dynamic scheme in which it allows users to join or leave a group efficiently without triggering a completely new key agreement protocol.

2.4.3.1 An Instance

This subsection gives an overview of an instance for the dynamic asymmetric group key agreement.

Definition A prime order group G is a group Diffie-Hellman (GDH) if there exists an efficient algorithm VDDH( ) which solves the Decisional Diffie-Hellman problem in G and there is no polynomial-time algorithm which solves the Computational Diffie-Hellman problem.

1) DASGKA.Setup

On input a security parameter 1^{λ}, a cyclic GDH group $G = <g>$ of some large prime order q is chosen. Two cryptographic hash functions H: $\{0, 1\}^* \rightarrow Z_p^*$ and F: $\{0, 1\}^* \rightarrow G$ are needed.

2) DASGKA.KeyGen

User U_i chooses randomly $x_i \in Z_q^*$ and computes $PK_i = g^{x_i}$. U_i keeps $SK_i = x_i$ secret as the private key, and publishes PK_i as public key.

3) DASGKA.KeyAgree

A group of players S = $\{U_1,..., U_n\}$ agree to trigger the protocol on time T. They require that there exists only one session for the same group and the same T. All players form a circle structure, with $U_{n+1} = U_1$ and $U_0 = U_n$.

(1) Round 1

Each U_i chooses $a_i \in Z_q^*$ and computes $K_i = g^{a_i}$. U_i sets $M_i = U_i \| T \| H(S) \| K_i$, and signs it as $\sigma_i(M_i)$ = MS.Sign(SK_i,M_i) = $F(M_i)^{x_i}$ where MS denoted as multi-signature scheme functions. U_i broadcasts (M_i,$\sigma_i(M_i)$) to others.

(2) Round 2

Each U_i checks the validation of data from his neighbors, namely (M_{i-1},$\sigma_{i-1}(M_{i-1})$) and (M_{i+1},$\sigma_{i+1}(M_{i+1})$). For instance, he checks the structure of M_{i-1} and runs VDDH(g,$F(M_{i-1})$,PK_{i-1}, $\sigma_{i-1}(M_{i-1})$) to see if (g,$F(M_{i-1})$,PK_{i-1}, $\sigma_{i-1}(M_{i-1})$ form a valid DDH tuple. If both are valid, U_i calculates his shared key with neighbors $K_{i,i+1}$ and $K_{i-1,i}$, and circle calculations X_i as follows.

$K_{i,i+1} = (K_{i+1})^{a_i} = g^{a_i a_{i+1}}$, $K_{i-1,i} = (K_{i-1})^{a_i} = g^{a_{i-1} a_i}$,

$X_i = H(K_{i,i+1}) \oplus H(K_{i-1,i})$,

$M'_i = U_i \| T \| H(S) \| X_i$;

$\sigma_i(M'_i)$ = MS.Sign (SK_i,M'_i) = $F(M'_i)^{x_i}$.

U_i broadcasts the message and the signature (M'_i, $\sigma_i(M'_i)$) to others.

(3) Key computing

After receiving all the messages, U_i checks the validation of all (M'_j, $\sigma_j(M'_j)$), $j \in \{1,\ldots,n\}$, $j \neq i$. If valid, U_i obtains all the circle calculations X_j and checks whether $X_1 \oplus \ldots \oplus X_n = 0$. If so, the handshake is accepted and shared session key is $SK = H(H(K_{1,2}),\ldots,H(K_{i,i+1}),\ldots,H(K_{n,1}),T)$, where $H(K_{i-j,i-j+1}) = H(K_{i,i+1}) \oplus X_{i-1} \oplus \ldots \oplus X_{i-j}$, with $j = 1,\ldots,n-1$.

4) DASGKA.PkGen

With the shared secret key $SK \in Z_q^*$, anyone in the group can generate a common public key $PK = g^{SK}$. If the group wants to publish the public key to outsiders, each player needs additional round for generating a multi-signature as follows. Each player U_i sets $M = T \| H(S) \| PK$, $\sigma_i(M) = F(M)^{x_i}$ and broadcasts $\sigma_i(M)$ to others, which can be verified by using VDDH( ). Then $\sigma_{1,\ldots,n}(M) = \prod_{i=1}^{n} \sigma_i(M) = F(M)^{\sum_{i=1}^{n} x_i}$ is a multi-signature for PK and the group descriptions S. It can be verified by using VDDH( ) to decide whether $(g, F(M), \prod_{i=1}^{n} PK_i, \sigma_{1,\ldots,n}(M))$ form

a valid DDH tuple. The public key PK, time stamp T, group description S and the signature $\sigma_{1,\ldots,n}(M)$ are made public.

5) DASGKA.Enc

Anyone who wants to send message $m \in G$ to the group, can select randomly $k \in Z_q^*$ and compute $c = (c_1, c_2) = (g^k, m \oplus PK^k)$. The ciphertext c is sent to the group.

6) DASGKA.Dec

On receiving the ciphertext c, any group member with the shared private key SK can calculate $m = c_2 \oplus c_1^{SK}$ to recover the message.

7) DASGKA.Join

We suppose $S = \{U_1, \ldots, U_n\}$ to be the current group and $J = \{U_{n+1}, \ldots, U_{n+n'}\}$ ($n' \geq 1$) to be a set of outsiders hoping to join the group. The agreed joining time is T'. They form a new circle structure among the members $S' = \{U_1, \ldots, U_{n+n'}\}$, with $U_{n+n'+1} = U_1$. The neighborhood changing concerns only U_1 and U_n among the old members.

(1) U_1, U_n and $\{U_{n+1}, \ldots, U_{n+n'}\}$ interacts as Round 1 of DASGKA.KeyAgree on the new time stamp T' and the new group S'. U_1 uses the previous K_1 and U_n uses previous K_n.

(2) U_1, U_n and $\{U_{n+1}, \ldots, U_{n+n'}\}$ interacts as Round 2 of DASGKA.KeyAgree. The shared key between U_1 and U_2, as well as the shared key between U_{n-1} and U_n, remains unchanged. U_i broadcasts its previous circle calculation value X_i, $2 \leq i \leq n-1$. If anyone finds that some player outputs a value different from the previous one, he alerts all players and returns reject.

(3) All players in S' run as Key Computing of DASGKA.KeyAgree to obtain a new shared private key, except for that they only need to check the validation of $n'+2$ messages from U_1, U_n and $\{U_{n+1}, \ldots, U_{n+n'}\}$

(4) All players in S' run DASGKA.PkGen to refresh the public key and the multi-signature.

8) DASGKA.Leave

For convenience of explanation, we assume that only a member $U_i \in S$ leaves the group S. Member leaving in bulk can be done simultaneously. The

remainders form a new circle structure among the members $S'= \{U_1, \ldots, U_{i-1}, U_{i+1}, \ldots, U_n\}$. The neighborhood changing concerns only U_{i-1} and U_{i+1}. New time stamp T' and the new group S' are used during the protocol.

(1) U_{i-1} and U_{i+1} interact as Round 1 of DASGKA.Key-Agree, using the previous K_{i-1} and K_{i+1}.

(2) U_{i-1} and U_{i+1} interact as Round 2 of DASGKA.Key-Agree. The shared key between U_{i-2} and U_{i-1}, as well as the shared key between U_{i+1} and U_{i+2}, remains unchanged.

(3) All players in S' run as Key Computing of DASGKA.KeyAgree to obtain a new shared private key, except for that they only need to check the validation of two messages from U_{i-1} and U_{i+1}.

(4) All players in S' run DASGKA.PkGen to refresh the public key and the multi-signature.

This scheme is an asymmetric group key agreement scheme. The scheme has advantage in that it has protocols; join and leave protocols in which they support for a dynamic group, but it also has some drawbacks in computation complexity.

2.5 Identity-Based Broadcast Encryption Scheme

This section gives an overview of a research in an identity-based broadcast encryption scheme as proposed by Du, Wang, Ge, and Wang (2005). In this scheme, an identity-based broadcast encryption is used to distribute a key over a network, so that each member can compute a specified key. Then a conventional private key cryptosystem such as data encryption standard (DES) can be used to encrypt subsequent messages.

Their scheme consists of a center and a set of users $U = \{ID_i \mid i = \{1,\ldots,n\}\}$, where ID_i is a unique identifier of user i. Each user has a public/private key pair (Q_i, S_i). The broadcast encryption is as the following.

2.5.1 Algorithms

2.5.1.1 Setup

A private key generator chooses a random number $s \in Z_q^*$ and set $P_{pub} = sP$. Then the private key generator publishes system parameters $params = \{G_1, G_2, q, P, P_{pub}, H_1, H_2\}$, and keep s as a master key.

2.5.1.2 Private Key Extraction

A user submits his identity information *ID* to private key generator. Then private key generator computes the user's public key as $Q_{ID} = H_1(ID)$, and returns his private key $S_{ID} = sQ_{ID}$.

2.5.1.3 Encryption

The center computes $Q_{V_1} = \sum_{i=1}^{n} Q_i$ and a (n-1 × n) matrix which is defined as the following.

$$\begin{pmatrix} a_2 \\ a_3 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

The center also form n-1 auxiliary keys

$Q_{V_i} = (Q_1, Q_2, \ldots, Q_n) \times a_i'$ where $2 \le i \le n$ which means that

$Q_{V_2} = Q_1 + Q_2$

$Q_{V_3} = Q_1 + Q_3$

$Q_{V_n} = Q_1 + Q_n$.

The cryptogram is then formed by computing,for some random $r \in Z_q^*$

$U_1 = rP$, $U_i = r\,Q_{V_i}$ where $2 \le i \le n$

$V = k \oplus H_2(e(P_{\text{pub}}, r\,Q_{V_1}))$

The center outputs the ciphertext (U_i, $1 \le i \le n$, V) and broadcast it to the set of users U.

2.5.1.4 Decryption

The recipient ID_i set a vector $a_1 = (0,\ldots 0,1,0,\ldots 0)$ with n elements, and only the i^{th} element is 1. Then A is a $n \times n$ matrix

$$A = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

The recipient ID_i can solve the following system of equations $(x_1, x_2, \ldots, x_n) \times A =$ (1 1 … 1). With $(x_1, x_2, \ldots, x_n)$ they can get

$$(x_1, x_2, \ldots, x_n) \times \begin{pmatrix} Q_i \\ Q_{V_2} \\ \vdots \\ Q_{V_n} \end{pmatrix} = Q_{V_1}$$

To decrypt the ciphertext, the recipient ID_i needs to compute $e(P_{pub}, r\,Q_{V_1})$, which with knowledge of the private key S_i it can do via:

$$e(P_{pub}, r\,Q_{V_1})$$
$$= e(P_{pub}, r(x_1 Q_i + x_1\,Q_{V_2} + \ldots + x_n\,Q_{V_n}))$$
$$= e(P_{pub}, rx_1Q_i)\; e(P_{pub}, r(x_2\,Q_{V_2} + \ldots + x_n\,Q_{V_n}))$$
$$= e(rP, x_1sQi)\; e(P_{pub}, x_2rQ_{V_2} + \ldots + x_n\,rQ_{V_n})$$
$$= e(U_1, x_1Si)\; e(P_{pub}, x_2U_2 + \ldots + x_n\,U_n)$$

Then the recipient can compute

$$K = V \oplus H_2(e(U_1, x_1Si)\; e(P_{pub}, \textstyle\sum^{n}_{i=2} x_iU_i))$$

2.5.2 Analysis

In that paper, the authors also gives analyze of the identity-based broadcast encryption scheme both in computation and communication costs.

2.5.2.1 The computation cost for encryption by the center is as the following

1) $2n$-2 additions in the group G_1.
2) n +1 scalar multiplications in the group G_1.
3) One pairing computation.
4) One hashing computation.
5) One XOR operation.

2.5.2.2 The computation cost for decryption per user is as the following

1) Solving a set of linear equations with n variables using Cramer's Rule.
2) n-1 additions in the group G_1.

3) n scalar multiplications in the group G_1.
4) Two pairing computations.
5) One hashing computation.
6) One XOR operation.

The communication cost of this scheme is one broadcast, which includes n elements in the group G_1 and a message $V \in \{0,1\}^*$.

This paper proposes an identity-based broadcast encryption scheme for distributing a group key. Members can use this group key in symmetric way to encrypting or decrypting a message. The scheme itself is an asymmetric key distribution. The disadvantage in this scheme is that it needs a trust centralized management. When group members are changed, the center must compute a new series of member's public keys. This can be occurred when a new member needs to join a group or an existing member has left the group.

CHAPTER 3

BROADCAST ENCRYPTION BASED ON BRAID GROUPS

A proposed broadcast encryption scheme is made up of three phases; setup, encryption, and decryption. In the setup phase, when any user needs to join a group, he sends a join request message to a director. The director is one of the group members and everyone knows published braids denoted as g_i of others. Each user can compute his own public keys PK_i from his private key K_i, the published braid g_o of another node at the same level in a key tree and his published braid g_i. The proposed scheme uses the key tree, mentioned in Norranut Saguansakdiyotin and Pipat Hiranvanichakorn (2012), to construct a public group key. The public group key PK_{Group} can be computed individually from a user private key K_i and other public key according to i^{th} position of the user node in the tree. This chapter first mentions the notation of key tree and then states the detail of algorithms. In the encryption phase, it shows that anyone outside a group can send encrypted message to the group members. This chapter also demonstrates the decryption method in the decryption phase. At the end of this chapter, it states complexity of the proposed scheme by comparing with other schemes in broadcast encryption.

3.1 Key Tree Notation

A key tree was earliest proposed by Wallner, Harder, and Agee (1997) as a tool in centralized group key distribution systems and was adapted by Kim, Perrig, and Tsudik (2000) for using in fully distributed, contributory key agreement. Figure 3.1 shows an example of key tree mentioned in Norranut Saguansakdiyotin and Pipat Hiranvanichakorn (2012). It is a binary tree which has only left subtree. The tree composes of both intermediate and leaf nodes. The root node is located at level 0 and the lowest leaf is at level h. Each node is represented as $<l,v>$ where l and v are

denoted as v^{th} node at level l in a tree. As shown in Figure 3.1, a member node M_i where $i \in (1 \ldots N)$ is located only at a leaf of the tree. Each member node is associated with a private keys pair ($K_{<l,v>}$, $K^{-1}_{<l,v>}$) and a published braid $g_{<l,v>}$. A public key of each member node $PK_{<l,v>} = K_{<l,v>}\, g_{<l,v'>}\, g_{<l,v>}\, K^{-1}_{<l,v>}$ where v' is another node at the same level. For an intermediate node, which is not a member node $K_{<l,v>} = K_{<l+1,\ 2v>}\ PK_{<l+1,\ 2v+1>}\ K^{-1}_{<l+1,\ 2v>}$ or $K_{<l,v>} = K_{<l+1,\ 2v+1>}\ PK_{<l+1,\ 2v>}\ K^{-1}_{<l+1,\ 2v+1>}$. A key $K_{<l,v>}$ and a public key $PK_{<l,v>}$ of an intermediate node is computed independently from the values of key and public key of child nodes to achieve a subgroup key.

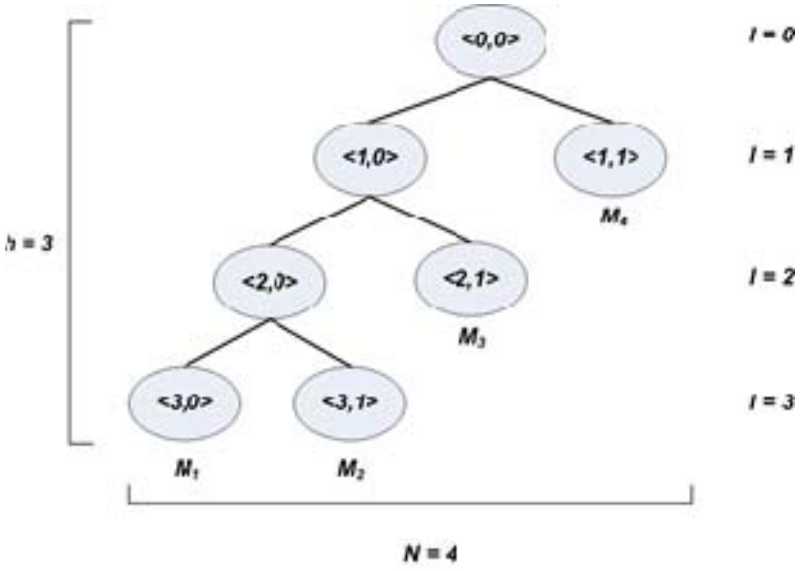

Figure 3.1 Notation of Key Tree

3.2 Setup

Assume that users A, B, and C join a group simultaneously. These users can be ordered according to some criteria such as MAC address or IP address. The first member of a group is the director and is located at the left highest level node in a key tree. Each user has his private key K_i, which is a braid in the different braid groups of each other. In order to setup a group each user needs to compute their own public keys PK_i. The public key of user i^{th} is $K_i\, g_o g_i\, K_i^{-1}$ where g_o is a published braid of

another node at the same level in a key tree and g_i is his own published braid. The published braid g_i must be in the same braid group of K_i. The following shows an example of how to compute a public group key in setup phase;

User A:	$K_A = a$ where $a \in B_a$	(A's private key)
	$g_A \in B_a$	(A's published braid)
	$PK_A = ag_Bg_Aa^{-1}$	(A's public key)
User B:	$K_B = b$ where $b \in B_b$	(B's private key)
	$g_B \in B_b$	(B's published braid)
	$PK_B = bg_Ag_Bb^{-1}$	(B's public key)
User C:	$K_C = c$ where $c \in B_c$	(C's private key)
	$g_C \in B_c$	(C's published braid)
	$PK_C = cg_Ag_Bg_Cc^{-1}$	(C's public key)

Assume that the order of a group member is users A, B, and C respectively, so user A is the director of the group. At this time a key tree is formed as shown in Figure 3.2. The director can compute key tree consisting of member public keys PK_i and public group keys PK_{Group}. In a key tree, a member node has public key and published braid, but an intermediate node has only public subgroup key. In this case, the key tree consisting of the values of PK_A, PK_B, PK_{A}B, PK_C, and PK_{ABC} as well as published braids of the members. The director, user A, must broadcast this key tree to all members.

When every member in the group receives the key tree, they can use this key tree information in the future in order to compute a new public group key PK_{Group} if they are selected to be a director. This is because every member can also compute a group key K_{Group} by using information in a key tree. From the previous scenario, user B and C can also compute the group key K_{ABC} as the following;

For user B's point of view;

$K_{ABC} = K_{AB}\, PK_C\, (K_{AB})^{-1}$ where

$K_{AB} = b\, PK_A\, b^{-1}$

For user C's point of view;

$K_{ABC} = c\, PK_{AB}\, c^{-1}$

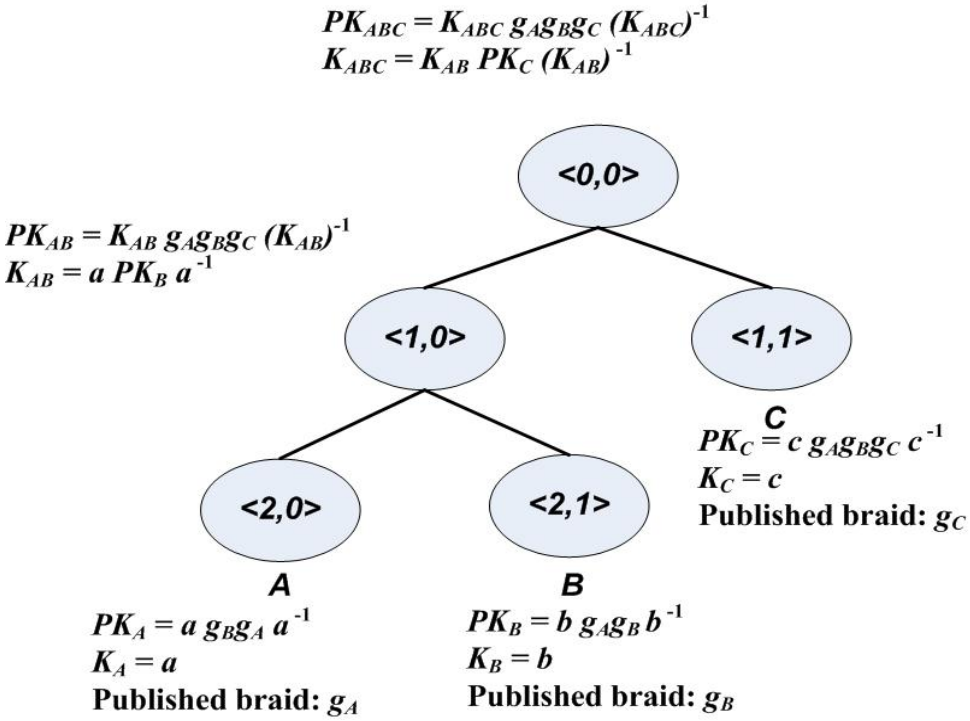

Figure 3.2 Group Key at Setup Phase Computed by User A

Next is an example of how to define values of braids for each user. In this scenario, there are four users; A, B, C, and D, so the braid group B_{20} can be used. Each user's private key must be on different braid groups, so the braid indexing 1 to 5 can be defined for user A, braid index 6 to 10 for user B, and so on. The published braid g_i must be in the same braid group with K_i. The following gives an example of how to compute key tree.

Published braids:

g_A: $\sigma_2\sigma_2$ g_B: $\sigma_9\sigma_6$ g_C: $\sigma_{11}\sigma_{14}$

For user A:

A's private key: $\sigma_1\sigma_4\sigma_3$

A's public key: $(\sigma_1\sigma_4\sigma_3)\, \sigma_9\sigma_6\sigma_2\sigma_2\, (\sigma_1\sigma_4\sigma_3)^{-1}$

For user B:

B's private key: $\sigma_8\sigma_7\sigma_9$

B's public key: $(\sigma_8\sigma_7\sigma_9)\ \sigma_2\sigma_2\sigma_9\sigma_6\ (\sigma_8\sigma_7\sigma_9)^{-1}$

For user C:

C's private key: $\sigma_{11}\sigma_{12}\sigma_{13}$

C's public key: $(\sigma_{11}\sigma_{12}\sigma_{13})\ \sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\ (\sigma_{11}\sigma_{12}\sigma_{13})^{-1}$

The director, user A, can compute key tree as the following;

$K_{AB} = a\ PK_B\ a^{-1}$

$= (\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_2\sigma_2\sigma_9\sigma_6\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$ (1)

$PK_{AB} = K_{AB}\ g_A g_B g_C\ K_{AB}^{-1}$

$= (\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_2\sigma_2\sigma_9\sigma_6\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\ \sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\ (\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$

The value of K_{AB} can also be computed by user B as the following;

$K_{AB} = b\ PK_A\ b^{-1}$

$= (\sigma_8\sigma_7\sigma_9)\ (\sigma_1\sigma_4\sigma_3)\ \sigma_9\sigma_6\sigma_2\sigma_2\ (\sigma_1\sigma_4\sigma_3)^{-1}\ (\sigma_8\sigma_7\sigma_9)^{-1}$ (2)

We can see that (1) = (2)

$K_{ABC} = K_{AB}\ PK_C\ K_{AB}^{-1}$

$= (\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_2\sigma_2\sigma_9\sigma_6\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})\ \sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\ \sigma_{14}\ (\sigma_{11}\sigma_{12}\sigma_{13})^{-1}(\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$ (3)

$PK_{ABC} = K_{ABC}\ g_A g_B g_C\ K_{ABC}^{-1}$

$= (\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_2\sigma_2\sigma_9\sigma_6\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})\ \sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\ \sigma_{14}\ (\sigma_{11}\sigma_{12}\sigma_{13})^{-1}(\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\ \sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\ (\sigma_1\sigma_4\sigma_3)(\sigma_8\sigma_7\sigma_9)\ \sigma_2\sigma_2\sigma_9\sigma_6\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})\ \sigma_{14}^{-1}\sigma_{11}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\ (\sigma_{11}\sigma_{12}\sigma_{13})^{-1}(\sigma_1\sigma_4\sigma_3)\ (\sigma_8\sigma_7\sigma_9)\ \sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\ (\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$

The value of K_{ABC} can be computed by user B and C as the following;

For user B:

$K_{ABC} = K_{AB}\, PK_C\, K_{AB}^{-1}$

$= (\sigma_8\sigma_7\sigma_9)\,(\sigma_1\sigma_4\sigma_3)\,\sigma_9\sigma_6\sigma_2\sigma_2\,(\sigma_1\sigma_4\sigma_3)^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})\,\sigma_2\sigma_2\sigma_9\sigma_6$
$\sigma_{11}\sigma_{14}\,(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}\,(\sigma_8\sigma_7\sigma_9)\,(\sigma_1\sigma_4\sigma_3)\,\sigma_2^{-1}\sigma_2^{-1}\sigma_6^{-1}\sigma_9^{-1}\,(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}$ (4)

For user C:

$K_{ABC} = c\, PK_{AB}\, c^{-1}$

$= (\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}$
$(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,\,(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}$ (5)

We can see that (3) = (4) = (5)

In the proposed scheme, each user needs to send his or her join request message to the group. These users know the order among them in a tree, so they can compute their public keys and send them to a director. Then the director computes a public group key and broadcasts it to all members, thus the total communication messages in key agreement and public group key generation are n+1 multicast and n unicast messages. The n multicast messages are from sending join request, and one multicast message is from sending updated key tree. The n unicast messages are from sending user's public keys. The total computation costs are n + (n-1) serial numbers of braid group multiplication. The n serial numbers of braid group multiplication are for the public key computation of the n members, and the n-1 serial numbers of braid group multiplication are for the public group key computation by director.

3.3 Encryption

In this phase, a user outside the group can send a ciphertext to the group members by encrypting it with the sender private key and the group public key. The receivers, which are the group members, can decrypt it using their own private keys and the sender's public key.

The following example occurs when user D, which is not the group member referred from the previous subsection, sends a ciphertext to the group members. In order to encrypt the message m_D, user D computes the ciphertext m' by encrypting it with user D's private key and the group public key, then he sends m' together with his public key PK_D to the group members as the following;

$PK_D = d\, g_A g_B g_C\, d^{-1}$

$m' = m_D \oplus H\,(R\ d\ PK_{ABC}\ d^{-1})$ and sends m', a random braid R and PK_D to members of the group. The random braid R can be changed in every message that sent.

3.4 Decryption

Each member of the group can decrypt the ciphertext with the group key. For short, this section gives an example of user A that is the group member wants to decrypt the message as the following;

$m_D = m' \oplus H\,(R\ K_{ABC}\ PK_D\ (K_{ABC})^{-1})$

$m_D = m' \oplus H\,(R\ ((a\ PK_B\ a^{-1})PK_C\ (a\ PK_B\ a^{-1})^{-1})\ PK_D\ ((a\ PK_B\ a^{-1})\ PK_C\ (aPK_B\ a^{-1})^{-1})^{-1})$

From the above example, user A can use his private key a in the term K_{ABC} and $(K_{ABC})^{-1}$ as in the user A's view as mentioned in the previous subsection.

3.5 Correctness

This section shows the correctness of the proposed algorithms as the following;

Theorem: $ab = ba$ where $a \in B_{g_1}$, and $b \in B_{g_2}$

$m_D = m' \oplus H\,(R\ K_{ABC}\ PK_D\ (K_{ABC})^{-1})$

$= m' \oplus H(R\ K_{ABC}\ d\ g_A g_B g_C\ d^{-1}\ (K_{ABC})^{-1})$

$= m' \oplus H(R\ d\ K_{ABC}\ g_A g_B g_C\ (K_{ABC})^{-1}\ d^{-1})$

$= m' \oplus H(R\ d\ PK_{ABC}\ d^{-1})$

As examples shown above, any user which is not the group member cannot decrypt the ciphertext because he does not know the value of K_{ABC}.

3.6 Key Secrecy

The key secrecy is the concept related to the membership changes. Typically there are two types of the key secrecy; backward and forward secrecy. The backward secrecy prevents a new member joining the group to know the previous ciphertext of the group. A new group key distributing to the group members when a new member joins the group cannot be used to decrypt the previous ciphertext. The forward secrecy is used to prevent a left member to use the previous key to decrypt a ciphertext. The proposed scheme fulfils the concept of both backward secrecy and forward secrecy. It is shown by using two protocols; join and leave protocols. The join protocol is operated when a new member needs to join a group, on the other hand the leave protocol is operated when a member needs to leave the group.

3.7 Join Protocol

In the proposed scheme, when a new member needs to join a group, he will send a request to join a group message containing his published braid to a director. The director can be anyone in the existing group members. After the director receives the join request message, he sends a sequence of published braids of all members to the new member. Then the new member computes his public key and sends it back to the director. The director can generate a new key tree including the new member's public key and new public group key in the tree and then broadcasts this new key tree to all members. The insertion point of a new member in a key tree is at a new root node. From section 3.2, the setup phase, an example is continued with the scenario when user D needs to join the group as shown in Figure 3.3. User C as a director

sends a sequence of published braids of all existing members $g_A g_B g_C$ to a new member. User D can compute his public key and then send it back to the director. The director can compute a new group key $K_{ABCD} = K_{ABC}\, PK_D\, K_{ABC}^{-1}$ and a new public group key PK_{ABCD}. The new member can also compute the new public group key after getting an updated key tree but he cannot compute the previous group key K_{ABC} or K_{AB} because he does not know each user's private key. Thus the proposed scheme complies with the concept of backward secrecy.

Next is an example of join protocol that continued from the setup phase as the following.

For user D:

D's private key: $\sigma_{19}\sigma_{17}$

D's published braids: $\sigma_{18}\sigma_{16}$

D's public key: $(\sigma_{19}\sigma_{17})\ \sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\sigma_{18}\sigma_{16}\ (\sigma_{19}\sigma_{17})^{-1}$

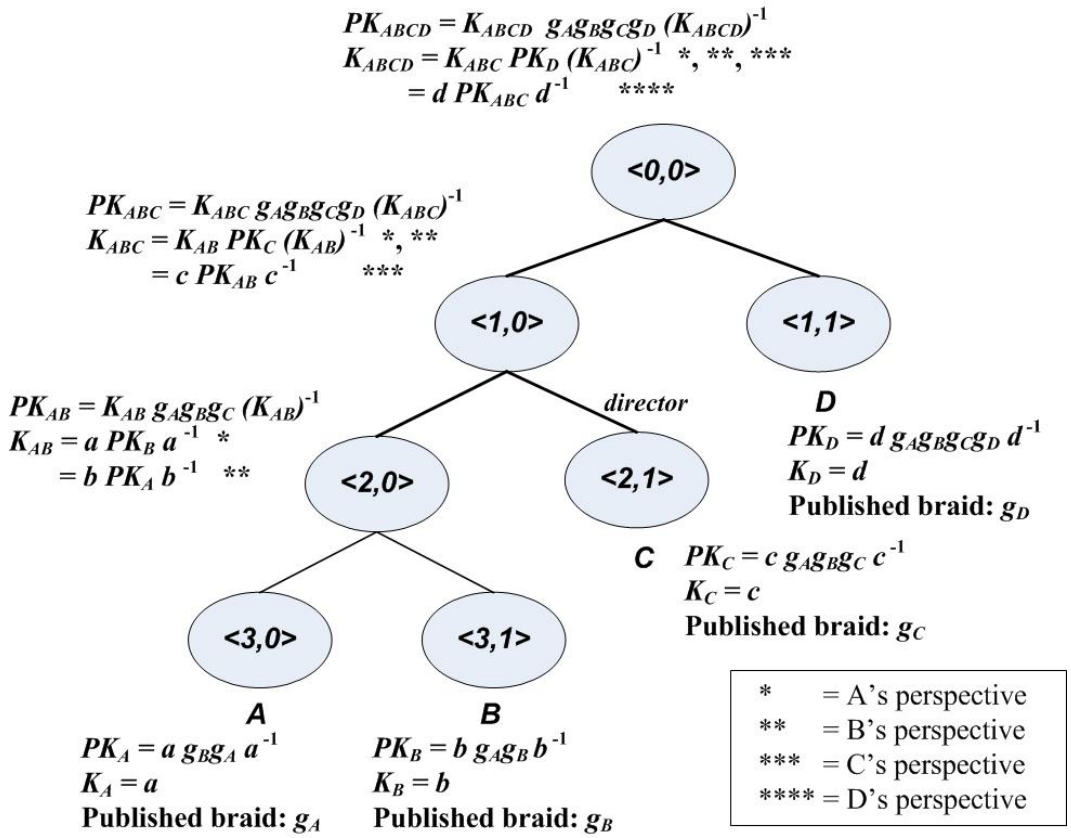

Figure 3.3 Key Tree After User D Joins the Group

The director, user C, can compute key tree as the following;

$K_{ABCD} = K_{ABC}\, PK_D\, K_{ABC}^{-1}$

$= (\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}$
$(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}\,(\sigma_{19}\sigma_{17})\,\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}$
$\sigma_{14}\sigma_{18}\sigma_{16}\,(\sigma_{19}\sigma_{17})^{-1}\,(\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,\sigma_{14}^{-1}$
$\sigma_{11}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}(\sigma_1\sigma_4\sigma_3)\;\;(\sigma_8\sigma_7\sigma_9)\;\;\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\;\;(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}$
(6)

$PK_{ABCD} = K_{ABCD}\, g_A g_B g_C \underline{g_D}\, K_{ABCD}^{-1}$

$= (\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,\sigma_2\sigma_2\sigma_9\sigma_6$
$\sigma_{11}\sigma_{14}(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\;\;(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}(\sigma_{19}\sigma_{17})\,\sigma_2\sigma_2$
$\sigma_9\sigma_6\sigma_{11}\sigma_{14}\sigma_{18}\sigma_{16}\,(\sigma_{19}\sigma_{17})^{-1}\,(\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$
$\sigma_{14}^{-1}\sigma_{11}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}(\sigma_1\sigma_4\sigma_3)(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}$
$\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\sigma_{18}\sigma_{16}\,(\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$
$\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}$
$(\sigma_{19}\sigma_{17})\,\sigma_{16}^{-1}\sigma_{18}^{-1}\sigma_{14}^{-1}\sigma_{11}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_{19}\sigma_{17})^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)$
$\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,\sigma_{14}^{-1}\sigma_{11}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}$
$(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}$

The value of K_{ABCD} can be computed by user D as the following;

$K_{ABCD} = d\, PK_{ABC}\, d^{-1}$

$= (\sigma_{19}\sigma_{17})\,(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})$
$\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\,(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$
$\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{11}\sigma_{14}\,\sigma_{18}\sigma_{16}\,(\sigma_1\sigma_4\sigma_3)(\sigma_8\sigma_7\sigma_9)\,\sigma_2\sigma_2\sigma_9\sigma_6\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})\,\sigma_{14}^{-1}$
$\sigma_{11}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}(\sigma_{11}\sigma_{12}\sigma_{13})^{-1}(\sigma_1\sigma_4\sigma_3)\,(\sigma_8\sigma_7\sigma_9)\,\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}$
$(\sigma_{19}\sigma_{17})^{-1}$ (7)

We can see that (6) = (7)

The total communication messages for join protocol are two multicast and two unicast messages. The first multicast message is a join request message from new member to a group. The second multicast message is an updated key tree message sending by a director. Two unicast messages are from sending a sequence of

published braids of all existing members by a director and sending a new member's public key. The total computation costs are three serial numbers of braid group multiplications. One braid multiplication is at new member to compute his public key, and two braid multiplications are at director to compute new subgroup key and new public group key.

The multiple join occurs when any m users want to join a group simultaneously. In this case, the joining users can know the order among them, so the total communication messages are m+2 multicast and m unicast messages. The m multicast messages are from sending join request by m joining users. Two multicast messages are from sending a sequence of published braids of all members and sending updated key tree by a director. The m unicast messages are from sending each user's public key. The total computation cost is $2m$+1 serial numbers of braid group multiplication.

3.8 Leave Protocol

The leave protocol operates when any user needs to leave the group. A leaving member sends a leave request message to a director. For the proposed scheme, the director is designed to be a member below the leaving node in a tree in order to minimize the computation. In a case that the leaving node is the first member in an existing tree, the director can be the second member in the tree. The director has to compute a new key tree, and then broadcasts it to all members. The next example is continued with the scenario from the join protocol in section 3.7. In this leave protocol scenario, it is assumed that user C is going to leave the group. User B is going to be a director of the group and responsible to compute a new key tree as shown in Figure 3.4. In this case, user B computes a new public key PK_{AB}, a new group key $K_{ABD} = K_{AB}\ PK_D\ K_{AB}^{-1}$, and a new public group key PK_{ABD}. Then user B broadcasts new key tree to all members. The proposed scheme designed to comply with the concept of forward secrecy as stated above. For example, the leaving user C cannot know the value of the new group key K_{ABD} and he cannot use his private key to decrypt messages.

Next is an example that continued from the join operation as the following.

The director, user B can compute key tree as the following;

$K_{ABD} = K_{AB}\, PK_D\, K_{AB}^{-1}$

$= (\sigma_8\sigma_7\sigma_9)(\sigma_1\sigma_4\sigma_3)\,\sigma_9\sigma_6\sigma_2\sigma_2\,(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_{19}\sigma_{17})\,\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{18}\sigma_{16}$
$(\sigma_{19}\sigma_{17})^{-1}(\sigma_8\sigma_7\sigma_9)(\sigma_1\sigma_4\sigma_3)\,\sigma_2^{-1}\sigma_2^{-1}\sigma_6^{-1}\sigma_9^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}$ (8)

$PK_{ABD} = K_{ABD}\, g_A g_B g_D\, K_{ABD}^{-1}$

$= (\sigma_8\sigma_7\sigma_9)(\sigma_1\sigma_4\sigma_3)\,\sigma_9\sigma_6\sigma_2\sigma_2\,(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_{19}\sigma_{17})\,\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{18}\sigma_{16}$
$(\sigma_{19}\sigma_{17})^{-1}(\sigma_8\sigma_7\sigma_9)(\sigma_1\sigma_4\sigma_3)\,\sigma_2^{-1}\sigma_2^{-1}\sigma_6^{-1}\sigma_9^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}\,\sigma_2\sigma_2\sigma_9\sigma_6\,\sigma_{18}\sigma_{16}\,(\sigma_8\sigma_7\sigma_9)$
$(\sigma_1\sigma_4\sigma_3)\,\sigma_9\sigma_6\sigma_2\sigma_2\,(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_{19}\sigma_{17})\,\sigma_{16}^{-1}\sigma_{18}^{-1}\sigma_6^{-1}\sigma_9^{-1}\sigma_2^{-1}\sigma_2^{-1}(\sigma_{19}\sigma_{17})^{-1}(\sigma_8\sigma_7\sigma_9)$
$(\sigma_1\sigma_4\sigma_3)\,\sigma_2^{-1}\sigma_2^{-1}\sigma_6^{-1}\sigma_9^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}\,(\sigma_8\sigma_7\sigma_9)^{-1}$

The value of K_{ABD} can be computed by user D as the following;

$K_{ABD} = d\, PK_{AB}\, d^{-1}$

$= (\sigma_{19}\sigma_{17})\,(\sigma_8\sigma_7\sigma_9)\,(\sigma_1\sigma_4\sigma_3)\,\sigma_9\sigma_6\sigma_2\sigma_2\,(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}\,\sigma_2\sigma_2\sigma_9\sigma_6\sigma_{18}\sigma_{16}$
$(\sigma_8\sigma_7\sigma_9)(\sigma_1\sigma_4\sigma_3)\,\sigma_2^{-1}\sigma_2^{-1}\sigma_6^{-1}\sigma_9^{-1}(\sigma_1\sigma_4\sigma_3)^{-1}(\sigma_8\sigma_7\sigma_9)^{-1}(\sigma_{19}\sigma_{17})^{-1}$ (9)

We can see that (8) = (9)

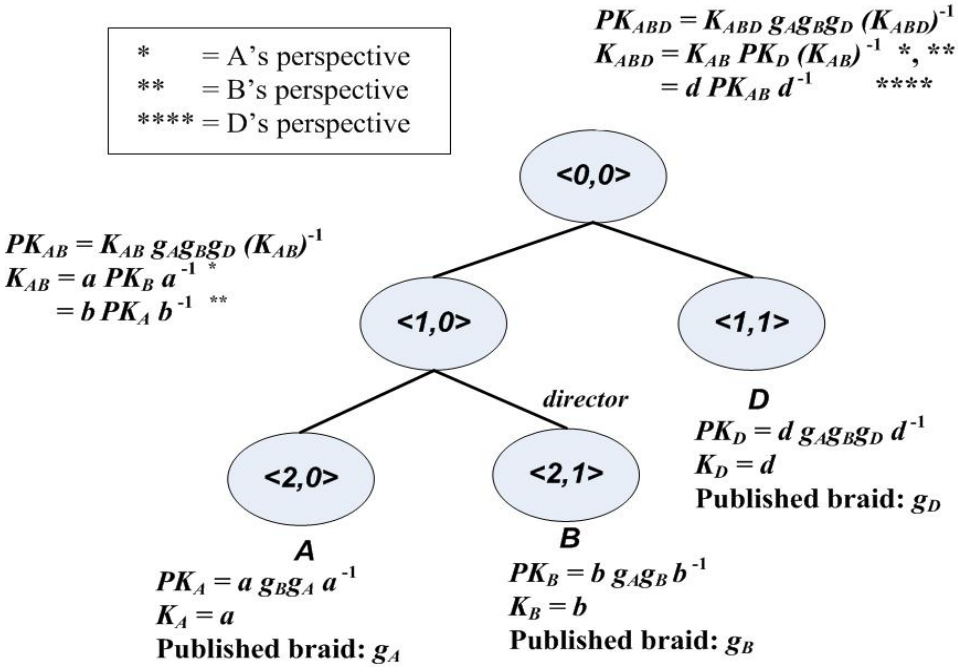

Figure 3.4 Key Tree After User C Leaves the Group

The total communication messages for leave protocol are ≤ (n-1) + 1 unicast and one multicast messages. One unicast message comes from sending leave request to the director. In worse case, the remaining members must compute their new public keys and send them to a director, so this requires (n-1) unicast messages. The one multicast message is for sending updated key tree by the director. In worse case, the number of computation costs in this protocol is equal to (n-1) + (n-2) when the leaving user is the first one that has joined the group.

The multiple leave occurs when any m users want to leave a group simultaneously. In this case the total communication messages are m + (n-m) unicast in worse case and one multicast messages. The m unicast messages are from m living member in sending leave request and (n-m) unicast messages are from the remaining member in sending new public keys to a director as in worse case. The only one multicast message is for sending updated key tree by a director. The total computation cost is (n-m) + (n-m-1) serial numbers of braid group multiplication.

3.9 Collude Attack

A collude attack can be occurred when two or more users work together and they can forge a valid private key which it will be given to anyone. The proposed scheme can resist a collude attack like this. The first reason is that in the proposed scheme a private key of user comes from user itself, so it is not distributed from private key generator. Users can produce their own private keys and then publish public keys to others. The second is that public and private keys of user are related together e.g. $PK_A = (K_A)g_O g_A(K_A)^{-1}$. If someone forges a private key of anyone, so his private and public keys are not related then he can know it.

3.10 Complexity

This section gives the comparison on the proposed broadcast group-oriented encryption scheme with the scheme proposed by Ma et al., Wu et al. and Zhao et al. in both communication and computation costs.

3.10.1 Communication Cost

The communication cost is shown in Table 3.1. The communication costs are analyzed by comparing both unicast and multicast messages for every member in the system. For join and leave operations, the assumptions that there are n existing members in a group and m members need to join or leave the group are used. For Ma et al., they do not state how to publish the public group key and send private key to each member, so it supposes to used unicast message and is written down with remark. Another notation in this table is that the join and leave operations were not proposed in Ma et al. and Wu et al. scheme. For Zhao et al. (2011) the process for generating a multi-signature is omitted because the comparison in the same condition with the others is needed.

3.10.2 Computation Cost

The computation cost is shown in Table 3.2. The values in the table are measured in Big-O notation. The proposed scheme has only multiplication in braid groups while the others have both multiplication in G or G_τ (where G_τ comes from e: $G \times G \rightarrow G_\tau$), and also exponentiation.

Table 3.1 Communication Cost of Broadcast Encryption Schemes

Scheme	Operation	Message	Unicast Message	Multicast Message
Ma, Wu, Li	KeyAgree and PKgen	n *	n *	-
	Join	-	-	-
	Leave	-	-	-
Wu, Mu, Susilo, Qin, Domingo-Ferrer	KeyAgree and PKgen	n	-	n
	Join	-	-	-
	Leave	-	-	-
Zhao, Zhang, Tian	KeyAgree and PKgen	$2n$	-	$2n$
	Join	$2m+4$	-	$2(m+2)$
	Leave	$2m$	-	$2m$
Proposed Scheme	KeyAgree and PKgen	$2n+1$	n	$n+1$
	Join	$2m+2$	m	$m+2$
	Leave	$n+1$	n	1

Note: * Does not Mention Clearly

Table 3.2 Computation Cost of Broadcast Encryption Schemes

Scheme	Operation	Computation
Ma, Wu, Li	KeyAgree and PKgen	$O(n)E$
	Join	-
	Leave	-
Wu, Mu, Susilo, Qin, Domingo-Ferrer	KeyAgree and PKgen	$O(n)M + O(n)M_\tau + O(n^2)E + O(n)E_\tau$
	Join	-
	Leave	-
Zhao, Zhang, Tian	KeyAgree and PKgen	$O(n)E$
	Join	$O(n+m)E$
	Leave	$O(n+m)E$
Proposed Scheme	KeyAgree and PKgen	$O(n)Mul$
	Join	$O(m)Mul$
	Leave	$O(n-m)Mul$

Note: n: the total number of members in the protocol,
m: the number of members who want to join/leave the group,
G: element in G, G_τ: element in G_τ ,
M: multiplication (or division) in G,
E: exponentiation in G,
M_τ: multiplication (or division) in G_τ,
Mul: multiplication in braid groups.

CHAPTER 4

IDENTITY BASED BROADCAST ENCRYPTION BASED ON BRAID GROUPS

This chapter shows the way to apply the concept of the identity based encryption scheme to the proposed scheme in broadcast encryption based on braid groups mentioned in chapter 3 and it is called an identity based broadcast encryption scheme. First of all, an idea of how to apply the concept of braid groups to the identity based encryption is expressed. The reason in applying braid groups to the identity based encryption is to reduce the exponential cost in bilinear pairing operation. This chapter also gives an example in applying braid group to the identity based encryption, then gives an example of identity based broadcast encryption scheme based on braid groups. The proposed scheme is designed to support for a dynamic group, which new users can join the group or existing members can leave the group. The final section shows the comparison result with another scheme on an identity based broadcast encryption. The proposed scheme needs a private key generator only in private key extraction operation in the beginning, but for group operations such as in setup, join and leave phases it does not require the private key generator.

4.1 Identity Based Encryption Based on Braid Groups

The braid groups concept can be applied to the identity based encryption scheme, but some hard problems such as conjugacy search problem in braid groups makes it to be nontrivial problem. This section gives an example that Alice, Bob and Charlie need to send ciphertext to each other using the identity based encryption based on braid groups. The identity based encryption based on braid group also has four steps; setup, private key extraction, encryption and decryption as stated in the following.

4.1.1 Setup

In the proposed scheme on the identity based encryption based on braid groups, each user needs to send their identities to a private key generator. The private key generator encodes their identities into braids, and then divides each braid of each user into two braids. For example an identity of a user i^{th} can be encoded into a braid g_i, then this braid is divided into left and right braids; g_{i_l} , g_{i_r}. The private key generator then prepares private braid groups; g, z_1, z_2, z_3, and z_4. Each user has two private keys; $SK1_i$ and $SK2_i$ where $SK1_i$ is equal $z_1\, g_{i_l}\, z_2$ and $SK2_i$ is equal $z_4\, g_{i_r}\, z_3$. The braid groups g, g_1,…, g_n have to be on different braid groups. The braid z_1, z_2, z_3 and z_4 must have the index expanded on all braid groups; g_1 to g_n. Because braids g_1,…, g_n, z_1, z_2, z_3 and z_4 are also in the same group, the conjugacy search problem can be applied. The private key generator also computes public braids; $Z_1 = z_2^{-1} g$, and $Z_2 = g z_4^{-1}$ and publishes them for using in an encryption and decryption phases. These two values, Z_1 and Z_2 are secure because braid z_2, z_4, and g are secret braids.

4.1.2 Encoding Method

An identity can be written as an IP address which is 32 bits long. A number in each octet ranges from 0 to 255. These 256 numbers must be mapped into braid words for each octet.

Normally a braid $a \in B_n$ can be written in at most $(n\text{-}1)^1 + \ldots + (n\text{-}1)^{n-1}$ positive braids as shown in the following;

1-word long; $(n\text{-}1)^1$ words, for example; $\{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$

2-word long; $< (n\text{-}1)^2$ words, for example; $\{\sigma_1\sigma_1, \sigma_1\sigma_2, \sigma_1\sigma_3, \sigma_1\sigma_4, \sigma_2\sigma_1, \sigma_2\sigma_2, \sigma_2\sigma_3, \sigma_2\sigma_4, \sigma_3\sigma_2, \sigma_3\sigma_3, \sigma_3\sigma_4, \sigma_4\sigma_3, \sigma_4\sigma_4\}$

3-word long; $< (n\text{-}1)^3$ words, for example; $\{\sigma_1\sigma_1\sigma_1, \sigma_1\sigma_1\sigma_2, \sigma_1\sigma_1\sigma_3, \ldots, \sigma_4\sigma_4\sigma_4\}$

4-word long; $< (n\text{-}1)^4$ words, for example; $\{\sigma_1\sigma_1\sigma_1\sigma_1, \sigma_1\sigma_1\sigma_1\sigma_2, \sigma_1\sigma_1\sigma_1\sigma_3, \ldots, \sigma_4\sigma_4\sigma_4\sigma_4\}$

But using the above method, care must be concerned in that two braid words can produce the same value such as $\sigma_1\sigma_2\sigma_1 = \sigma_2\sigma_1\sigma_2$ or $\sigma_1\sigma_3 = \sigma_3\sigma_1$.

For ease of implementation, my suggestion is that each octet in an IP address can be mapped by using B_5 which has 8-word long as shown in Table 4.1.

For example IP address in the following can be mapped into braid words;

IP address 10.5.32.255 mapped into $\sigma_4\sigma_4\sigma_4\sigma_4\sigma_1\sigma_4\sigma_1\sigma_4$. $\sigma_4\sigma_4\sigma_4\sigma_4\sigma_4\sigma_1\sigma_4\sigma_1$. $\sigma_4\sigma_4\sigma_1\sigma_4\sigma_4\sigma_4\sigma_4\sigma_4$. $\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1$.

Thus $g_{i_l} = \sigma_4\sigma_4\sigma_4\sigma_4\sigma_1\sigma_4\sigma_1\sigma_4\ \sigma_4\sigma_4\sigma_4\sigma_4\sigma_4\sigma_1\sigma_4\sigma_1$ and $g_{i_r} = \sigma_4\sigma_4\sigma_1\sigma_4\sigma_4\sigma_4\sigma_4\sigma_4\ \sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1\sigma_1$.

In the Table 4.1, a braid word representing each octet contains only σ_1 and σ_4 in it. In case of the proposed scheme, the private key g_i of each user is in the different braid groups, so it must be mapped into the different braid groups.

Table 4.1 Mapping of an Octet in IPv4 into a Braid Word

Number in Each Octet in Binary	Braid Words
0000 0000	$\sigma_4\sigma_4\sigma_4\sigma_4\ \sigma_4\sigma_4\sigma_4\sigma_4$
0000 0001	$\sigma_4\sigma_4\sigma_4\sigma_4\ \sigma_4\sigma_4\sigma_4\sigma_1$
0000 0010	$\sigma_4\sigma_4\sigma_4\sigma_4\ \sigma_4\sigma_4\sigma_1\sigma_4$
0000 0011	$\sigma_4\sigma_4\sigma_4\sigma_4\ \sigma_4\sigma_4\sigma_1\sigma_1$
1111 1111	$\sigma_1\sigma_1\sigma_1\sigma_1\ \sigma_1\sigma_1\sigma_1\sigma_1$

4.1.3 Private Key Extraction

Each user obtains the private keys; $SK1_i$ and $SK2_i$ from a private key generator. Figure 4.1 shows the value of braids in the setup and private key extraction phases. The detail of the values prepared by the private key generator for Alice, Bob, and Charlie is shown in the following.

For Alice:

1) The private key generator encodes Alice's identity ID_A into a braid in group g_1, then divides this braid into left and right braids; g_{1_l} and g_{1_r}

2) The private key generator prepares private keys $SK1_A$ and $SK2_A$ for Alice where $SK1_A = z_1\, g_{1_l}\, z_2$ and $SK2_A = z_4\, g_{1_r} z_3$

For Bob:

1) The private key generator encodes Bob's identity ID_B into a braid in group g_2, then divides this braid into left and right braids; g_{2_l} and g_{2_r}

2) The private key generator prepares private keys $SK1_B$ and $SK2_B$ for Bob where $SK1_B = z_1\, g_{2_l}\, z_2$ and $SK2_B = z_4\, g_{2_r} z_3$

For Charlie:

1) The private key generator encodes Charlie's identity ID_C into a braid in group g_3, then divides this braid into left and right braids; g_{3_l} and g_{3_r}

2) The private key generator prepares private keys $SK1_C$ and $SK2_C$ for Bob where $SK1_C = z_1\, g_{3_l}\, z_2$ and $SK2_C = z_4\, g_{3_r}\, z_3$

For the private key generator:

1) The private braid groups; g, z_1, z_2, z_3, and z_4

2) The public braid groups; g_1, g_2, g_3, …, g_n are for users 1^{st} to n^{th} where g, g_1, g_2, g_3, …, g_n must be on different braid groups.

3) The private key generator prepares public braids; $Z_1 = z_2^{-1} g$, and $Z_2 = g z_4^{-1}$

4.1.4 Encryption

This subsection gives an example that Alice needs to send a message M to Bob. She can encrypt it with her private key; $SK1_A$, $SK2_A$ and an identity of Bob, ID_B as the following. The value of Z_1 and Z_2 are given from the setup phase.

$$M' = M \oplus H\{\, SK1_A\, Z_1(g_2)\, Z_2\, SK2_A \,\}$$

4.1.5 Decryption

Bob can decrypt the message sent from Alice by using his private key; $SK1_B$, $SK2_B$ and an identity of Alice, ID_A as the following.

$$M = M' \oplus H\{\, SK1_B\, Z_1\,(g_1)\, Z_2 SK2_B \,\}$$

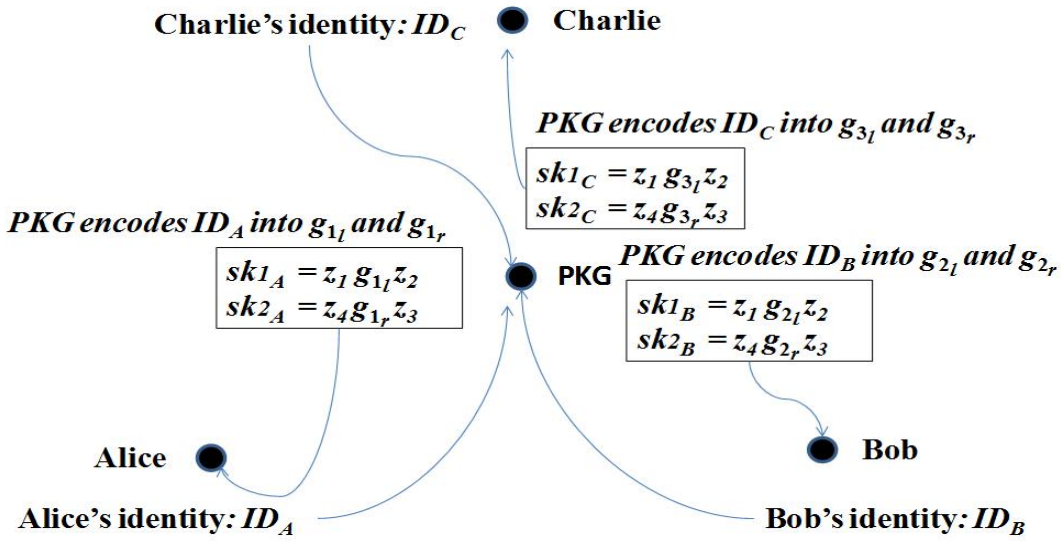

Figure 4.1 Setup and Private Key Extraction Phases for an Identity Based Encryption Based on Braid Groups

4.1.6 Correctness

This subsection shows the correctness of encryption and decryption processes as in the following. The property of commutation braids in different braid groups is used for example braid *a* and *b* commutes with each other, thus *ab* = *ba*. This property of the braid groups is true, when *a* and *b* are on different braid groups.

Theorem: $ab = ba$ where $a \in B_{g_1}$, and $b \in B_{g_2}$

Thus: $M = M' \oplus H\{ SK1_B\, Z_1\, (g_1)\, Z_2\, SK2_B\}$

$= M' \oplus H\{(z_1 g_{2_l}\, z_2)(\, z_2^{-1} g)\, (g_1)\, (g z_4^{-1})(\, z_4 g_{2_r} z_3)\}$

$= M' \oplus H\{(z_1 g_{2_l}\, z_2)(\, z_2^{-1} g)\, (g_{1_l}\, g_{1_r})\, (g z_4^{-1})(\, z_4 g_{2_r} z_3)\}$

$= M' \oplus H\{(z_1 g_{2_l})(\, g)\, (g_{1_l}\, g_{1_r})(g)(g_{2_r} z_3)\}$

$= M' \oplus H\{(z_1 g_{1_l})(g)\, (g_{2_l} g_{2_r})\, (g)(g_{1_r} z_3)\}$

$= M' \oplus H\{(z_1 g_{1_l} z_2)(\, z_2^{-1} g)\, (g_{2_l} g_{2_r})\, (g z_4^{-1})(\, z_4 g_{1_r} z_3)\, \}$

$= M' \oplus H\{ SK1_A\, Z_1\, (g_2)\, Z_2\, SK2_A\}$

$= M$

We can see that the ciphertext, sent by Alice to Bob, cannot decrypt it by Charlie or anybody because they do not know a value of $SK1_B$ *and* $SK2_B$.

4.1.7 Example

This subsection shows an example of the identity based encryption based on braid groups. This example is tested by braiding program implemented by Cha, Ko, Lee, Han, and Cheon (2001).

Given:

$g = \sigma_1\sigma_2\sigma_3$

$z_1 = \sigma_7\sigma_{12}\sigma_{17}$, $z_2 = \sigma_6\sigma_{11}\sigma_{19}$

$z_3 = \sigma_9\sigma_{15}\sigma_{18}$, $z_4 = \sigma_8\sigma_{13}\sigma_{18}$

For Alice: $g_1 = \sigma_9\sigma_7\sigma_8\sigma_6$ so $g_{1l} = \sigma_9\sigma_7$ and $g_{1r} = \sigma_8\sigma_6$

For Bob: $g_2 = \sigma_{11}\sigma_{14}\sigma_{14}\sigma_{12}$ so $g_{2l} = \sigma_{11}\sigma_{14}$ and $g_{2r} = \sigma_{14}\sigma_{12}$

For Charlie: $g_3 = \sigma_{18}\sigma_{19}\sigma_{16}\sigma_{17}$ so $g_{3l} = \sigma_{18}\sigma_{19}$ and $g_{3r} = \sigma_{16}\sigma_{17}$

A key that Alice encrypts a message M to Bob is $SK1_A\ Z_1(g_2)\ Z_2\ SK2_A$ as shown in Figure 4.2 in the left normal form.

In this case:

$SK1_A = z_1\ g_{1_l}\ z_2 = (\sigma_7\sigma_{12}\sigma_{17})(\sigma_9\sigma_7)(\sigma_6\sigma_{11}\sigma_{19})$

$Z_1 = z_2^{-1}g = (\sigma_6\sigma_{11}\sigma_{19})^{-1}(\sigma_1\sigma_2\sigma_3)$

$g_2 = (\sigma_{11}\sigma_{14}\sigma_{14}\sigma_{12})$

$Z_2 = gz_4^{-1} = (\sigma_1\sigma_2\sigma_3)(\sigma_8\sigma_{13}\sigma_{18})^{-1}$

$SK2_A = z_4\ g_{1_r}z_3 = (\sigma_8\sigma_{13}\sigma_{18})(\sigma_8\sigma_6)(\sigma_9\sigma_{15}\sigma_{18})$

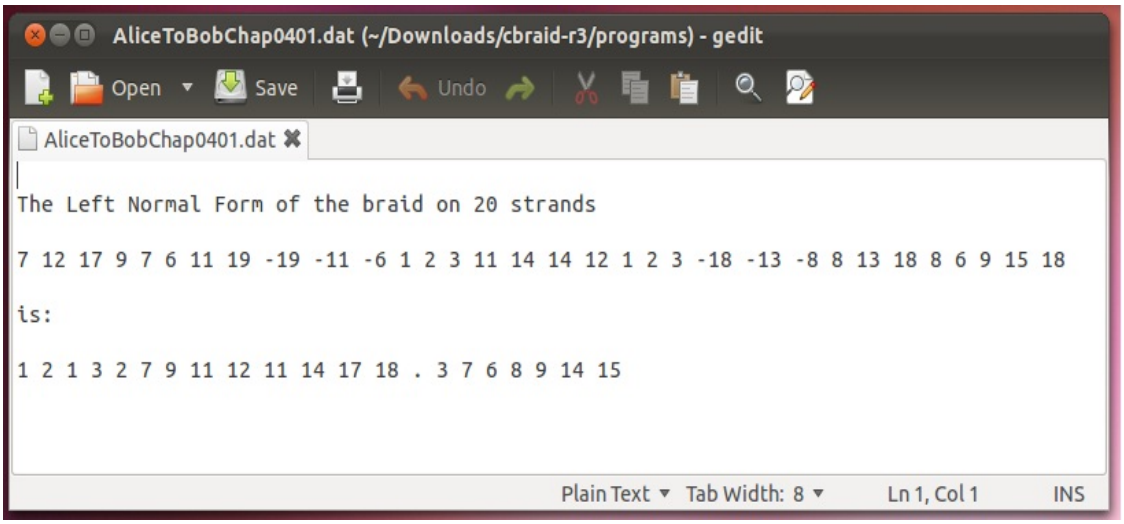

Figure 4.2 Encryption Key Computed by Braiding Program

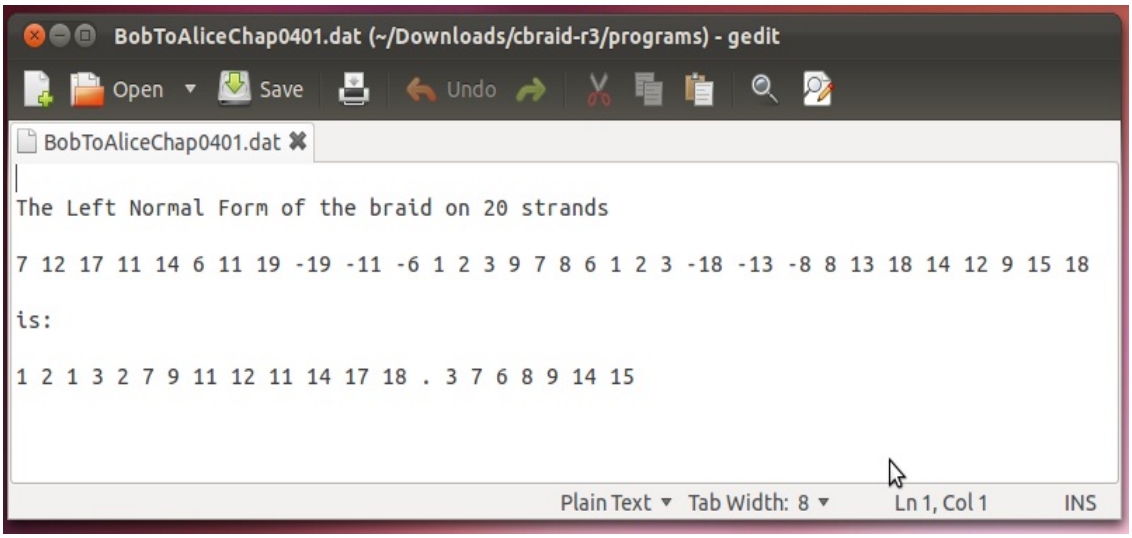

Figure 4.3 Decryption Key Computed by Braiding Program

A key that Bob decrypts a ciphertext M' is $SK1_B\ Z_1\ (g_1)\ Z_2 SK2_B$ as shown in Figure 4.3 in the left normal form.

In this case:

$SK1_B = z_1\ g_{2_l}\ z_2 = (\sigma_7\sigma_{12}\sigma_{17})(\sigma_{11}\sigma_{14})(\sigma_6\sigma_{11}\sigma_{19})$

$Z_1 = z_2^{-1} g = (\sigma_6\sigma_{11}\sigma_{19})^{-1}(\sigma_1\sigma_2\sigma_3)$

$g_1 = (\sigma_9\sigma_7\sigma_8\sigma_6)$

$Z_2 = gz_4^{-1} = (\sigma_1\sigma_2\sigma_3)(\sigma_8\sigma_{13}\sigma_{18})^{-1}$

$SK2_A = z_4\, g_{2_r} z_3 = (\sigma_8\sigma_{13}\sigma_{18})(\sigma_{14}\sigma_{12})(\sigma_9\sigma_{15}\sigma_{18})$

We can see that $SK1_A\, Z_1(g_2)\, Z_2\, SK2_A = SK1_B\, Z_1\,(g_1)\, Z_2 SK2_B$.

4.1.8 Complexity

This section expresses complexity in both communication and computation costs and compares them with the identity-based encryption scheme used bilinear pairing as in the following.

4.1.8.1 Communication Cost

For communication cost, the proposed scheme has the same cost as an identity-based encryption (IBE) scheme using bilinear pairing. Both schemes use two unicast messages in setup and private key extraction phases; one is for sending an identity from a user to a private key generator and another is for the private key generator to send a private key to user. Only one unicast message is in the encryption phase for both schemes. Table 4.2 is a summarization of these costs.

Table 4.2 Communication Cost of Identity Based Encryption Schemes

Scheme	Operation	Unicast Message
IBE Bilinear pairing	Setup and Private Key Extraction	2
	Encryption	1
	Decryption	-
IBE Braid groups	Setup and Private Key Extraction	2
	Encryption	1
	Decryption	-

Table 4.3 Computation Cost of Identity Based Encryption Schemes

Scheme	Operation	Computation
IBE Bilinear pairing	Setup	1*M*
	Private Key Extraction	1*H* + 1*M*
	Encryption	1*M* + 1*P* + 1*H* + 1*XOR*
	Decryption	1*P* + 1*H* + 1*XOR*
IBE Braid groups	Setup	2*Mul*
	Private Key Extraction	1*Enc* + 2*Mul*
	Encryption	1*H* + 1*Mul* + 1*XOR*
	Decryption	1*H* + 1*Mul* + 1*XOR*

Note: *M*: scalar multiplication (point multiplication) in *G*, *H*: hashing operation, *P*: pairing operation, *XOR*: XOR operation, *Mul*: serial multiplication in braid groups, *Enc*: encoding operation $\{ID\} \rightarrow$braid.

4.1.8.2 Computation Cost

For the identity-based encryption scheme using bilinear pairing, there are some operations involved such as scalar multiplication or point multiplication in *G*, pairing operation, hashing operation and XOR operation, but the pairing operation is the most dominating an execution time. For IBE using bilinear pairing, one point multiplication is from generating the private key generator's public key at setup phase. One hashing operation and one point multiplication are from generating a user's private key at the private key extraction phase. In the encryption phase, one point multiplication, one pairing operation, one hashing operation and one XOR operation are used. In the decryption phase, one pairing operation, one hashing operation and one XOR operation are used. For the identity-based encryption scheme using braid groups, there are two serial numbers of braid group multiplication in setup phase for generating public braids Z_1 and Z_2. One encoding operation is for converting string of an identity into two braids, and two serial numbers of braid multiplication for generating two private keys of a user at the private key extraction phase. In the both encryption and decryption phases, one serial number of braid multiplication, one

hashing operation and one XOR operation are used. Table 4.3 is a summarization of these costs.

4.2 Identity Based Broadcast Encryption Based on Braid Groups

A concept of an identity based broadcast encryption comes from applying the concept of an identity based encryption based on braid groups scheme from the previous section to the broadcast encryption scheme, mentioned in the previous chapter. The previous section gives the example that Alice encrypts message to Bob using the identity based encryption scheme. For the identity based broadcast encryption scheme, we can change Bob to be any group and do the same procedures as an individual person. The group also can do the same thing as in the broadcast encryption scheme in that the group can form a group public key used by others to encrypt messages for that group. This group public key can be group identity containing each user identities. Anyone needing to send encrypted message to group members, can encrypt that message using a group identity, so group members can decrypt it using their own private key. By comparing the proposed scheme on broadcast encryption with this proposed scheme on the identity based broadcast encryption, we are going to see that the later scheme does not required to setup a key tree to manage a group public key. This can reduce the cost in group key management. This section gives an example that Eve wants to send a ciphertext to a group which has Alice, Bob and Charlie as group members. This section demonstrates how to build a group by using identity based encryption scheme. Demonstration is divided into three phases; setup, encryption, and decryption.

4.2.1 Setup

The setup phase continues with an example that a group has three members; Alice, Bob, and Charlie. This subsection shows how to build a group using the identity based cryptography. An operation starts from each member must contact to a private key generator to obtain their private keys. Each member identity is contained in the group identity. For ease of explanation, the following example gives a group public key as $PK_{Group} = (g_1)(g_2)(g_3)$ as shown in Figure 4.4.

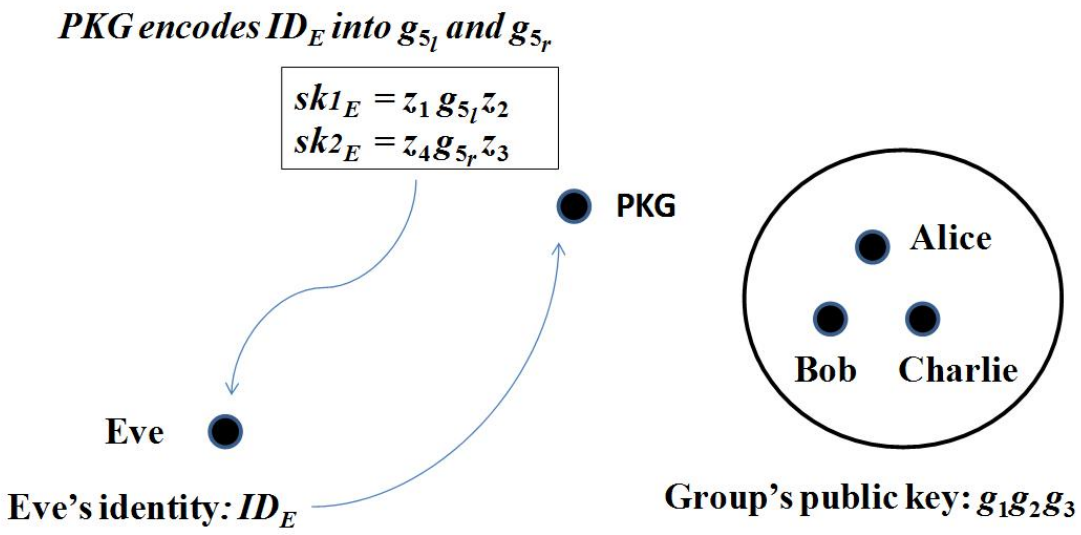

Figure 4.4 Setup and Private Key Extraction Phases for an Identity Based Broadcast Encryption Based on Braid Groups

For this example, Eve can encrypt a message to group by using the group public key and every group member can decrypt it using his/her own private key. The total communication messages in key agreement and public group key generation are 2*n* unicast messages where *n* is a number of users in the system. This comes from each user uses two unicast messages; one for a user to send identity information to a private key generator and another for private key generator to send back private keys. The value of the PK_{Group} needs to be known to all members in some ways depending on a situation; for example it can be sent together with a ciphertext to all members in case that a center needs to encrypt a secret shared key to the group members, or it can be delivered to all members during a process for creating a group. The computation cost is 2*n* + 3 serial numbers of braid group multiplication because three serial numbers of braid group multiplication are for creating public braids Z_1, Z_2 and PK_{Group}. The 2*n* comes from each user needs two private keys. This phase also needs *n* encoding operations for converting string of identity information into braid.

4.2.2 Encryption

This section gives an example that Eve wants to encrypt message to a group and the detail of this procedure is shown in the following.

For Eve:

1) The private key generator encodes Eve's identity ID_E into a braid in group g_5, then divides this braid into left and right braids; g_{5_l} and g_{5_r}

2) The private key generator prepares private keys $SK1_E$ and $SK2_E$ for Eve where $SK1_E = z_1 g_{5_l} z_2$ and $SK2_E = z_4 g_{5_r} z_3$

For Group:

1) The group public key $PK_{Group} = g_1 g_2 g_3$

When Eve sends a message M to the group, she can encrypt it with her private key; $SK1_E$, $SK2_E$ and an group identity, ID_{Group} (or PK_{Group}) as the following. The value of Z_1 and Z_2 are given from the setup phase.

$$M' = M \oplus H\{ SK1_E \, Z_1(PK_{Group}) \, Z_2 \, SK2_E \}$$

4.2.3 Decryption

Group member can decrypt the message sent from Eve by using his/her private key as the following.

For Alice:

$$M = M' \oplus H\{SK1_A \, Z_1(g_2) \, (g_3)(\, g_5) \, Z_2 \, SK2_A \}$$

For Bob:

$$M = M' \oplus H\{SK1_B \, Z_1(g_1) \, (g_3)(\, g_5) \, Z_2 \, SK2_B \}$$

For Charlie:

$$M = M' \oplus H\{SK1_C \, Z_1(g_1) \, (g_2)(\, g_5) \, Z_2 \, SK2_C \}$$

4.2.4 Correctness

This section shows the correctness of encryption and decryption processes as in the following. The property of commutation braids in different braid groups is used

for example braid a and b commutes with each other, thus $ab = ba$. This property of the braid groups is true, when a and b are on different braid groups.

Theorem: $ab = ba$ where $a \in B_{g_1}$, and $b \in B_{g_2}$

Thus for Alice:

$M = M' \oplus H\{SK1_A\ Z_1(g_2)(g_3)(\ g_5)\ Z_2\ SK2_A\}$

$= M' \oplus H\{\ (z_1 g_{1_l} z_2)(\ z_2^{-1} g)(g_2)(g_3)(\ g_5)\ (g z_4^{-1})(\ z_4 g_{1_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{1_l} z_2)(\ z_2^{-1} g)(g_2)(g_3)(\ g_{5_l}\ g_{5_r})\ (g z_4^{-1})(\ z_4 g_{1_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{1_l})(g)(g_2)(g_3)(\ g_{5_l}\ g_{5_r})(g)(g_{1_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{5_l})(g)(g_2)(g_3)(\ g_{1_l}\ g_{1_r})(g)(g_{5_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{5_l})(g)(g_2)(g_3)(g_1)(g)(g_{5_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{5_l} z_2)(\ z_2^{-1} g)(g_1)(g_2)(g_3)(g z_4^{-1})(z_4 g_{5_r} z_3)\ \}$

$= M$

Thus for Bob:

$M = M' \oplus H\{SK1_B\ Z_1(g_1)(g_3)(\ g_5)\ Z_2\ SK2_B\}$

$= M' \oplus H\{\ (z_1 g_{2_l} z_2)(\ z_2^{-1} g)(g_1)(g_3)(g_5)\ (g z_4^{-1})(\ z_4 g_{2_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{2_l} z_2)(\ z_2^{-1} g)(g_1)(g_3)(\ g_{5_l}\ g_{5_r})\ (g z_4^{-1})(\ z_4 g_{2_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{2_l})(g)(g_1)(g_3)(\ g_{5_l}\ g_{5_r})(g)(g_{2_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{5_l})(g)(g_1)(g_3)(\ g_{2_l}\ g_{2_r})(g)(g_{5_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{5_l})(g)(g_1)(g_2)(g_3)(g)(g_{5_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{5_l} z_2)(\ z_2^{-1} g)(g_1)(g_2)(g_3)(g z_4^{-1})(z_4 g_{5_r} z_3)\ \}$

$= M$

Thus for Charlie:

$M = M' \oplus H\{SK1_C\ Z_1(g_1)(g_2)(\ g_5)\ Z_2\ SK2_C\}$

$= M' \oplus H\{\ (z_1 g_{3_l} z_2)(\ z_2^{-1} g)(g_1)(g_2)(\ g_5)\ (g z_4^{-1})(\ z_4 g_{3_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{3_l} z_2)(\ z_2^{-1} g)(g_1)(g_2)(\ g_{5_l}\ g_{5_r})\ (g z_4^{-1})(\ z_4 g_{3_r} z_3)\ \}$

$= M' \oplus H\{\ (z_1 g_{3_l})(g)(g_1)(g_2)(\ g_{5_l}\ g_{5_r})(g)(g_{3_r} z_3)\ \}$

$= M' \oplus H\{ (z_1 g_{5_l})(g)(g_1)(g_2)(g_{3_l} g_{3_r})(g)(g_{5_r} z_3) \}$

$= M' \oplus H\{ (z_1 g_{5_l})(g)(g_1)(g_2)(g_3)(g)(g_{5_r} z_3) \}$

$= M' \oplus H\{ (z_1 g_{5_l} z_2)(z_2^{-1} g)(g_1)(g_2)(g_3)(g z_4^{-1})(z_4 g_{5_r} z_3) \}$

$= M$

We can see that the ciphertext, sent by Eve to the group, cannot decrypt it by others outside the group because they do not know any member private keys.

4.2.5 Example

This subsection shows an example of the identity based broadcast encryption based on braid groups. This example is tested by braiding program implemented by Cha, Ko, Lee, Han, and Cheon (2001).

Given $g = \sigma_1\sigma_2\sigma_3$

$z_1 = \sigma_7\sigma_{12}\sigma_{17}$, $z_2 = \sigma_6\sigma_{11}\sigma_{19}$

$z_3 = \sigma_9\sigma_{15}\sigma_{18}$, $z_4 = \sigma_8\sigma_{13}\sigma_{18}$

For Alice: $g_1 = \sigma_9\sigma_7\sigma_8\sigma_6$

For Bob: $g_2 = \sigma_{11}\sigma_{14}\sigma_{14}\sigma_{12}$

For Charlie: $g_3 = \sigma_{18}\sigma_{19}\sigma_{16}\sigma_{17}$

For Eve: $g_5 = \sigma_{22}\sigma_{21}\sigma_{23}\sigma_{24}$

For Group: $PK_{Group} = g_1 g_2 g_3$

The key that Eve encrypts a message M to group is $SK1_E\ Z_1(PK_{Group})\ Z_2\ SK2_E$ as shown in Figure 4.5 in the left normal form.

In this case:

$SK1_E = z_1\ g_{5_l}\ z_2 = (\sigma_7\sigma_{12}\sigma_{17})(\sigma_{22}\sigma_{21})(\sigma_6\sigma_{11}\sigma_{19})$

$Z_1 = z_2^{-1} g = (\sigma_6\sigma_{11}\sigma_{19})^{-1}(\sigma_1\sigma_2\sigma_3)$

$PK_{Group} = g_1 g_2 g_3 = (\sigma_9\sigma_7\sigma_8\sigma_6)(\sigma_{11}\sigma_{14}\sigma_{14}\sigma_{12})(\sigma_{18}\sigma_{19}\sigma_{16}\sigma_{17})$

$Z_2 = g z_4^{-1} = (\sigma_1\sigma_2\sigma_3)(\sigma_8\sigma_{13}\sigma_{18})^{-1}$

$SK2_E = z_4\ g_{5_r} z_3 = (\sigma_8\sigma_{13}\sigma_{18})(\sigma_{23}\sigma_{24})(\sigma_9\sigma_{15}\sigma_{18})$

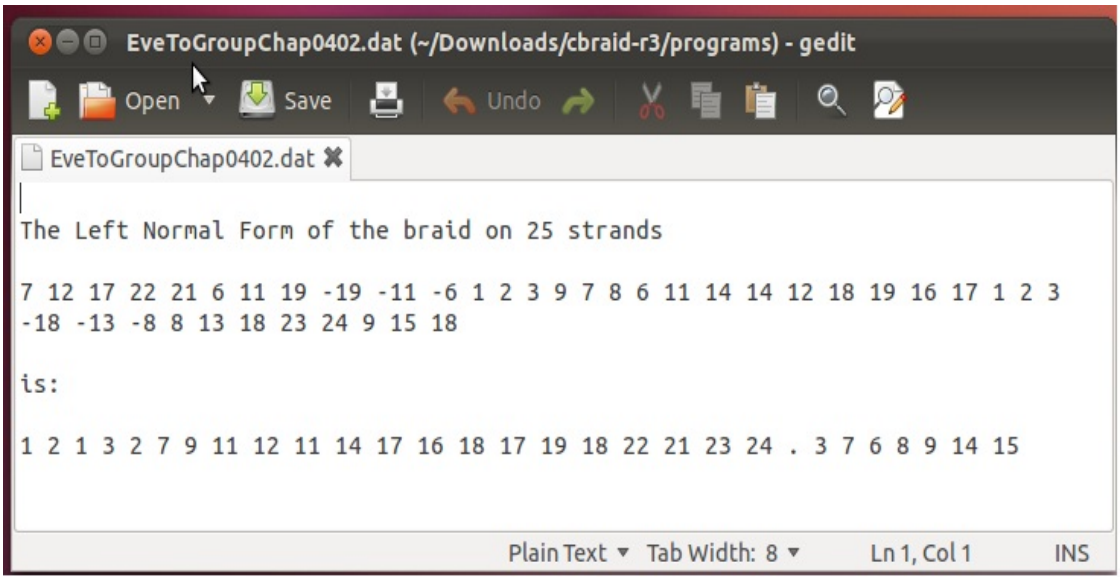

Figure 4.5 Eve's Encryption Key Computed by Braiding Program

The key that Alice decrypts a ciphertext M' is $SK1_A Z_1(g_2)(g_3)(g_5)Z_2 SK2_A$ as shown in Figure 4.6, and the key that Bob decrypt a ciphertext M' is $SK1_B\ Z_1(g_1)(g_3)(g_5)Z_2\ SK2_B$ as shown in Figure 4.7. Both are in the left normal form.

In Alice case:

$SK1_A = z_1\ g_{1_l}\ z_2 = (\sigma_7\sigma_{12}\sigma_{17})(\sigma_9\sigma_7)(\sigma_6\sigma_{11}\sigma_{19})$

$Z_1 = z_2^{-1}g = (\sigma_6\sigma_{11}\sigma_{19})^{-1}(\sigma_1\sigma_2\sigma_3)$

$g_2g_3g_5 = (\sigma_{11}\sigma_{14}\sigma_{14}\sigma_{12})(\sigma_{18}\sigma_{19}\sigma_{16}\sigma_{17})(\sigma_{22}\sigma_{21}\sigma_{23}\sigma_{24})$

$Z_2 = gz_4^{-1} = (\sigma_1\sigma_2\sigma_3)(\sigma_8\sigma_{13}\sigma_{18})^{-1}$

$SK2_A = z_4\ g_{1_r} z_3 = (\sigma_8\sigma_{13}\sigma_{18})(\sigma_8\sigma_6)(\sigma_9\sigma_{15}\sigma_{18})$

In Bob case:

$SK1_B = z_1\ g_{2_l}\ z_2 = (\sigma_7\sigma_{12}\sigma_{17})(\sigma_{11}\sigma_{14})(\sigma_6\sigma_{11}\sigma_{19})$

$Z_1 = z_2^{-1}g = (\sigma_6\sigma_{11}\sigma_{19})^{-1}(\sigma_1\sigma_2\sigma_3)$

$g_1g_3g_5 = (\sigma_9\sigma_7\sigma_8\sigma_6)(\sigma_{18}\sigma_{19}\sigma_{16}\sigma_{17})(\sigma_{22}\sigma_{21}\sigma_{23}\sigma_{24})$

$Z_2 = gz_4^{-1} = (\sigma_1\sigma_2\sigma_3)(\sigma_8\sigma_{13}\sigma_{18})^{-1}$

$SK2_B = z_4\ g_{2_r} z_3 = (\sigma_8\sigma_{13}\sigma_{18})(\sigma_{14}\sigma_{12})(\sigma_9\sigma_{15}\sigma_{18})$

We can see that $SK1_E\ Z_1(PK_{Group})\ Z_2\ SK2_E = SK1_A Z_1(g_2)(g_3)(g_5) Z_2 SK2_A$

$= SK1_B\ Z_1(g_1)(g_3)\ (g_5) Z_2\ SK2_B$

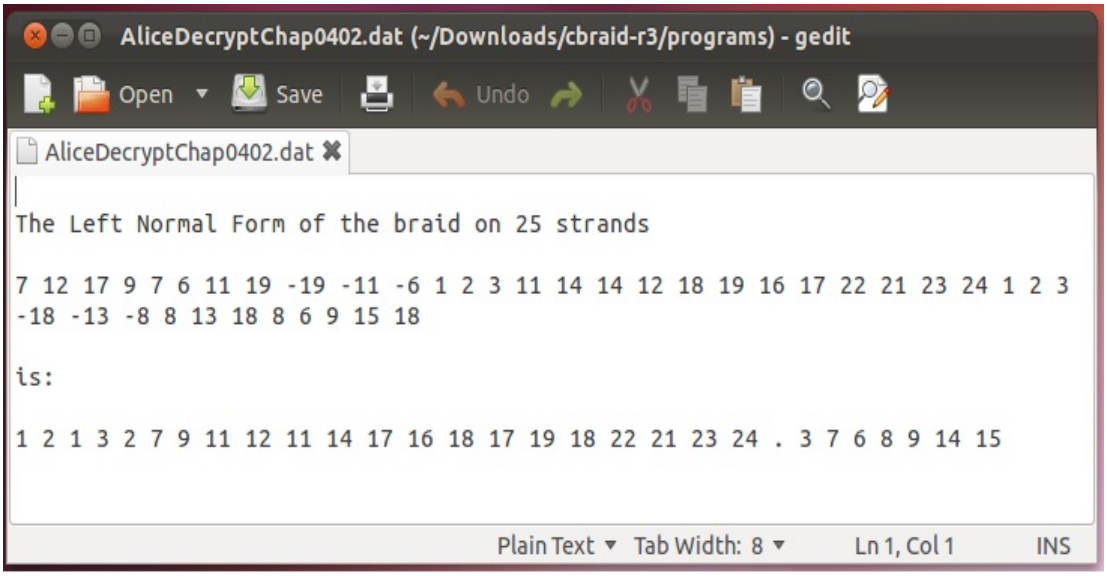

Figure 4.6 Alice's Decryption Key Computed by Braiding Program

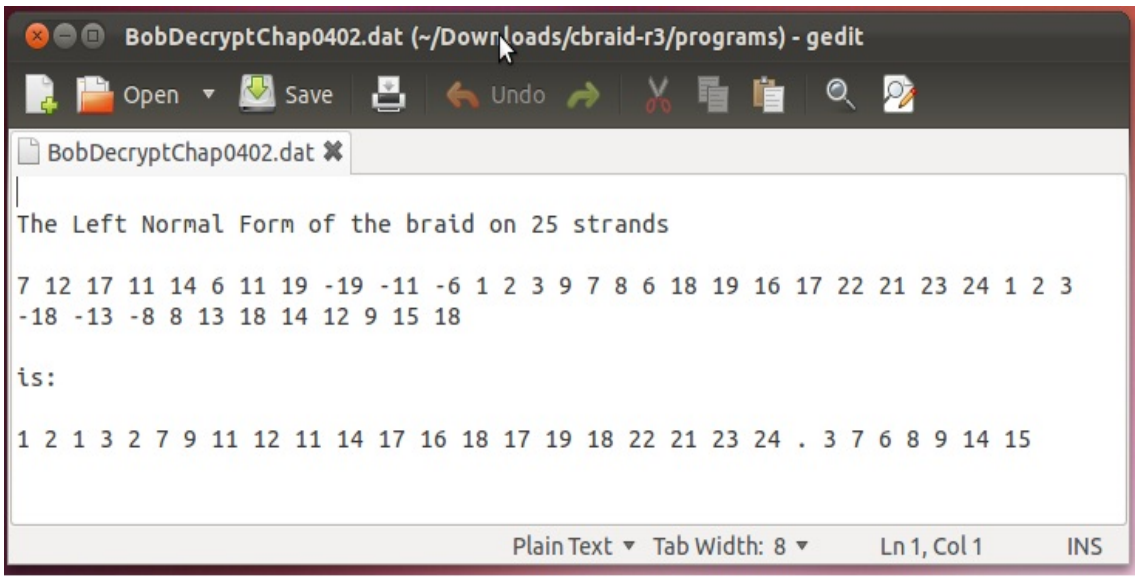

Figure 4.7 Bob's Decryption Key Computed by Braiding Program

4.3 Key Secrecy

For backward secrecy, a new member joining the group cannot use his private key to decrypt the previous ciphertext for that group. This is true because the previous public group key does not contain the new member's identity. Only a way he can decrypt the ciphertext is to know one of the previous group member's private key. For the forward secrecy, when one group member leaves a group, a group public key is changed, so a leaving member cannot use his private key to decrypt a ciphertext. The proposed scheme fulfils the concept of both backward secrecy and forward secrecy. This can be shown by using two protocols; join and leave protocols. The join protocol is operated when a new member needs to join a group, on the other hand the leave protocol is operated when a member needs to leave the group.

4.4 Join Protocol

In the proposed scheme, we assume that each new member have already contacted a private key generator in order to get his private key. When the new member needs to join a group, he can send a request to join message to the group. This message contains the new member's public key. Every member can compute a new group public key. From subsection 4.2.1, the setup phase, this example continues with the scenario when user named "Delta" needs to join the group as shown in the following.

For Delta:

1) The private key generator encodes Delta's identity ID_D into a braid in group g_4, then divides this braid into left and right braids; g_{4_l} and g_{4_r}

2) The private key generator prepares private keys $SK1_D$ and $SK2_D$ for Delta where $SK1_D = z_1 g_{4_l} z_2$ and $SK2_D = z_4 g_{4_r} z_3$

For Group:

The group public key $PK_{ABCD} = (g_1)\ (g_2)\ (g_3)\ (g_4)$

The total communication messages for join protocol are two unicast messages and one multicast message; these two unicast messages for obtaining a private key from private key generator, and one multicast message for sending a join request. The total computation cost are three serial numbers of braid group multiplication (one for generate new public group key and two for generate private keys for new member), and one encoding operation to convert identity into braid.

The multiple join occurs when any m users want to join a group simultaneously. In this case the total communication messages are $2m$ unicast messages and m multicast messages, and the total computation cost is $2m+1$ serial numbers of braid group multiplication, and m encoding operation.

4.5 Leave Protocol

The leave protocol operates when an existing member needs to leave the group. A leaving member sends a leave request message to a group. All members can compute a new public group key by themselves. The next example continues with the scenario from the join protocol in section 4.4. In this leave protocol scenario, it is assumed that user Charlie is going to leave the group. In this case, all members can compute a new public group key $PK_{ABD} == (g_1)\ (g_2)\ (g_4)$. The proposed scheme designed to comply with the concept of forward secrecy as mentioned above. For example, the leaving user Chalie cannot decrypt a ciphertext intended for the current group because she does not know the value of any member's private key.

The total communication message for leave protocol is one multicast message. The number of computation cost in this protocol is equal one serial number of braid group multiplication.

The multiple leave occurs when any m users want to leave a group simultaneously. In this case the total communication messages are m multicast messages, and the total computation cost is one serial numbers of braid group multiplication.

4.6 Complexity

This section gives a summarization in both communication and computation costs from comparing with the scheme proposed by Du, Wang, Ge, and Wang (2005) as in the following. In Du, Wang, Ge and Wang's scheme, the center encrypts a key to group members and then group member can use this key as a symmetric group key. Their scheme also needs a center to manage member's public keys, but the proposed scheme does not need a center to manage member's public keys. The comparison takes on all operation; setup, private key extraction, encryption and decryption. The costs for join and leave operations also are expressed in the next subsection, but these operations do not state in the Du, Wang, Ge and Wang's scheme. For the setup, private key extraction, join, and leave phases, the calculation of both costs are on per system, but for the encryption and decryption phases, the calculation are based on one sender (a center) and one recipient. For the join and leave phases, the assumption is that there are n existing group members and m members need to join or leave a group simultaneously. The comment for this comparison is that in the proposed scheme, the communication and computation costs are included the costs in contacting a private key generator and computing at the private key generator in setup and private key extraction, and join phases.

4.6.1 Communication Cost

The communication cost is shown in Table 4.4. The communication cost is analyzed by expressing both unicast and multicast messages for every member in the system. The communication cost in setup and private key extraction phase is $O(n)$ for both schemes. For the proposed scheme, the communication costs are $O(m)$ in both join and leave phases.

4.6.2 Computation Cost

The computation cost is shown in Table 4.5. The computation cost in setup phase are $O(1)$ on point multiplication for Du, Wang, Ge and Wang's scheme, and $O(1)$ on braid multiplication for the proposed scheme**.** The computation cost in private key extraction phase are $O(n)$ on both point multiplication and hashing operations for Du, Wang, Ge and Wang's scheme, and $O(n)$ on both braid multiplication and

encoding operations for the proposed scheme**.** The computation cost in encryption phase are $O(n)$ on both addition in G and point multiplication, and $O(1)$ on pairing, hashing, and XOR operations for Du, Wang, Ge and Wang's scheme, but $O(1)$ on braid multiplication, hashing and XOR operations for the proposed scheme. The computation cost in decryption phase is the same as in encryption phase in term of Big-O notation. For the proposed scheme, the computation costs are $O(m)$ on braid group multiplication and encoding operations in join phase and $O(1)$ on braid group multiplication in leave phase.

4.6.3 Message Size

A ciphertext size of the Du, Wang, Ge and Wang's scheme is depended on the number of group members. A reason for this is that the value of $U_i \in G$ where $2 \leq i \leq n$ is not identity information, thus they must be sent to all members. In the proposed scheme, a group identity, ID_{Group} or PK_{Group}, can be known by all group members, so no need to send it together with a ciphertext in some situation. Thus a ciphertext size in the proposed scheme is constant.

Table 4.4 Communication Cost of Identity Based Broadcast Encryption Schemes

Scheme	Operation	Message	Unicast Message	Multicast Message
Du, Wang, Ge and Wang's Scheme	Setup and Private Key Extraction (per system)	$2n$	$2n$	-
	Encryption (one sender)	1	-	1
	Decryption	-	-	-
	Join	-	-	-
	Leave	-	-	-

Table 4.4 (Continued)

Scheme	Operation	Message	Unicast Message	Multicast Message
The Proposed Scheme	Setup and Private Key Extraction (per system)	2*n*	2*n*	-
	Encryption (one sender)	1	-	1
	Decryption	-	-	-
	Join	3*m*	2*m*	*m*
	Leave	*m*	-	*m*

Note: *n*: the total number of members in the protocol,
m: the number of members who want to join/leave the group.

Table 4.5 Computation Cost of Identity Based Broadcast Encryption Schemes

Scheme	Operation	Computation
Du, Wang, Ge and Wang's Scheme	Setup (per system)	1*M*
	Private Key Extraction (per system)	*(n)H* + (*n)M*
	Encryption (one sender)	(2*n*-2)*A* + (*n*+1)*M* + 1*P* + 1*H* + 1*XOR*
	Decryption (per member)	(*n*-1)*A* + (*n*)*M* + 2*P* + 1*H* + 1*XOR*
	Join	-
	Leave	-

Table 4.5 (Continued)

Scheme	Operation	Computation
The Proposed Scheme	Setup (per system)	3*Mul*
	Private Key Extraction (per system)	*(n)Enc* + (2*n*)*Mul*
	Encryption (one sender)	1*H* + 1*Mul* + 1*XOR*
	Decryption (per member)	1*H* + 1*Mul* + 1*XOR*
	Join	(2*m*+1)*Mul* + (*m*)*Enc*
	Leave	1*Mul*

Note: *n*: the total number of members in the protocol,
m: the number of members who want to join/leave the group, A: addition in *G*,
M: scalar multiplication (point multiplication) in *G*, *H*: hashing operation,
P: pairing operation, *XOR*: XOR operation, *Mul*: serial multiplication in braid groups,
Enc: encoding {ID}$\rightarrow$ braid.

CHAPTER 5

IMPLEMENTAION

This chapter shows how to implement the broadcast encryption scheme based on braid groups mention in chapter 3. The implementation is on Ubuntu 11.10 operating systems. The program is developed in C++ programming language. The program also uses additional braiding libraries, developed by Cha, Ko, Lee, Han, and Cheon (2001). This chapter also states how to calculate public key for each user and the public group key. It also expresses the way to encrypt and decrypt message and how to build a hash function for braid group.

5.1 Program Design

The program is implemented by using multi-threading concept. A main thread is responsible for receiving commands from user. The user commands can be join, leave or quit. The join command uses when user needs to join a group. The leave command uses when he needs to leave a group, and the quit command is used when he wants to quit a program.

5.1.1 Main Thread

The subsection shows a flowchart for main thread as in Figure 5.1. In this main thread, it creates a new thread used for receiving a packet. The broadcast socket in main thread is used for sending join request (JREQ) or leave request (LREQ) messages. The program also records node's own IP address and joining group in a join request list which is implemented using link list data structure.

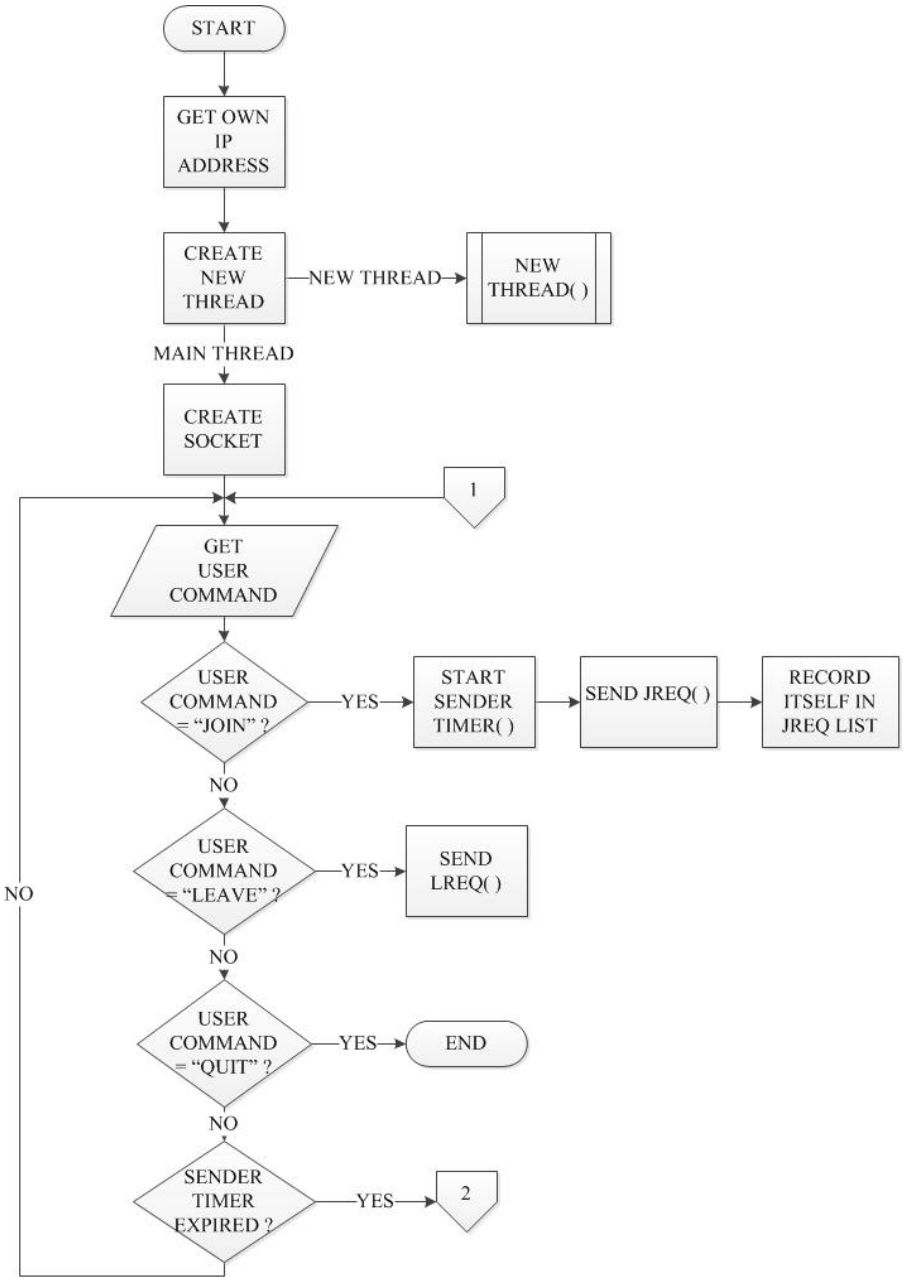

Figure 5.1 Flowchart for Main Program

5.1.2 New Thread

This new thread is created to handle the receiving packets such as join request, leave request, or update key tree packets. A flowchart of this new thread is shown in Figure 5.2. The program also records a sender's IP address and joining group of the sender in a join request list.

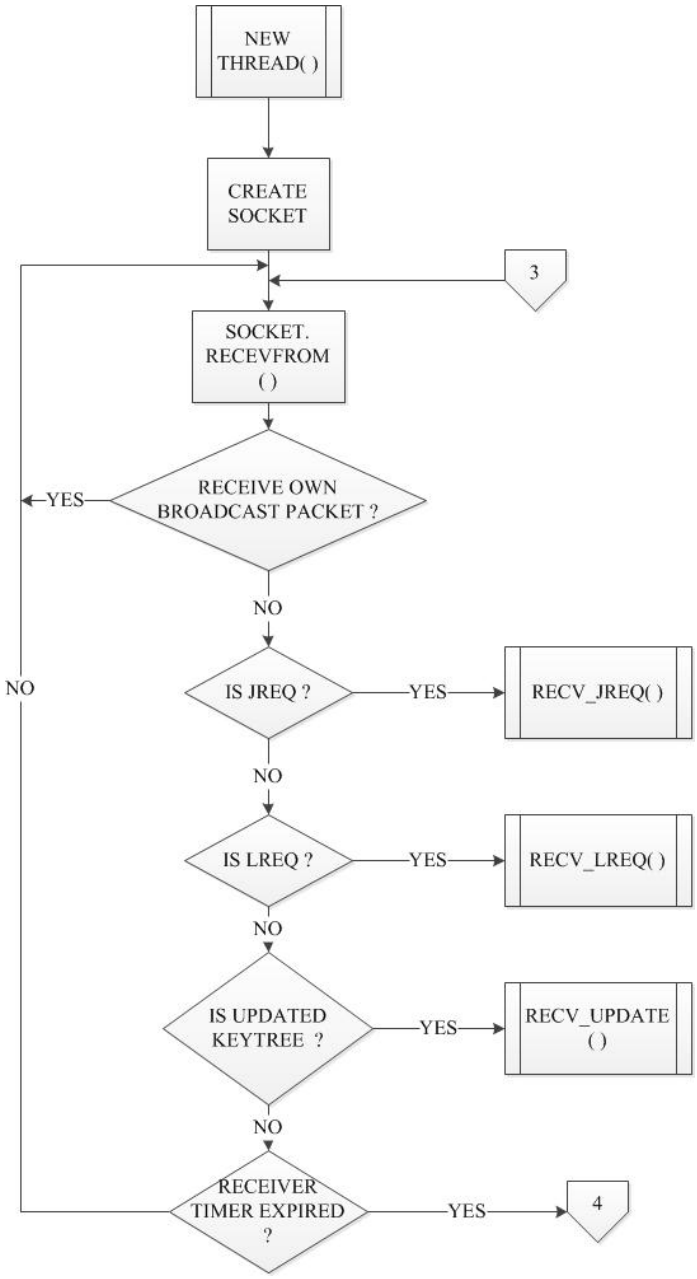

Figure 5.2 Flowchart for New Thread

5.1.3 Sender Timer

The sender timer is useful for a situation that there exists only one sender node broadcasting a join request message. No node replies for this request. The flowchart is shown in Figure 5.3. When a sender's timer is expired, the sender checks whether it is the lowest IP address node. If it has the lowest IP address, it sets itself as a director of a group. Then it creates key tree.

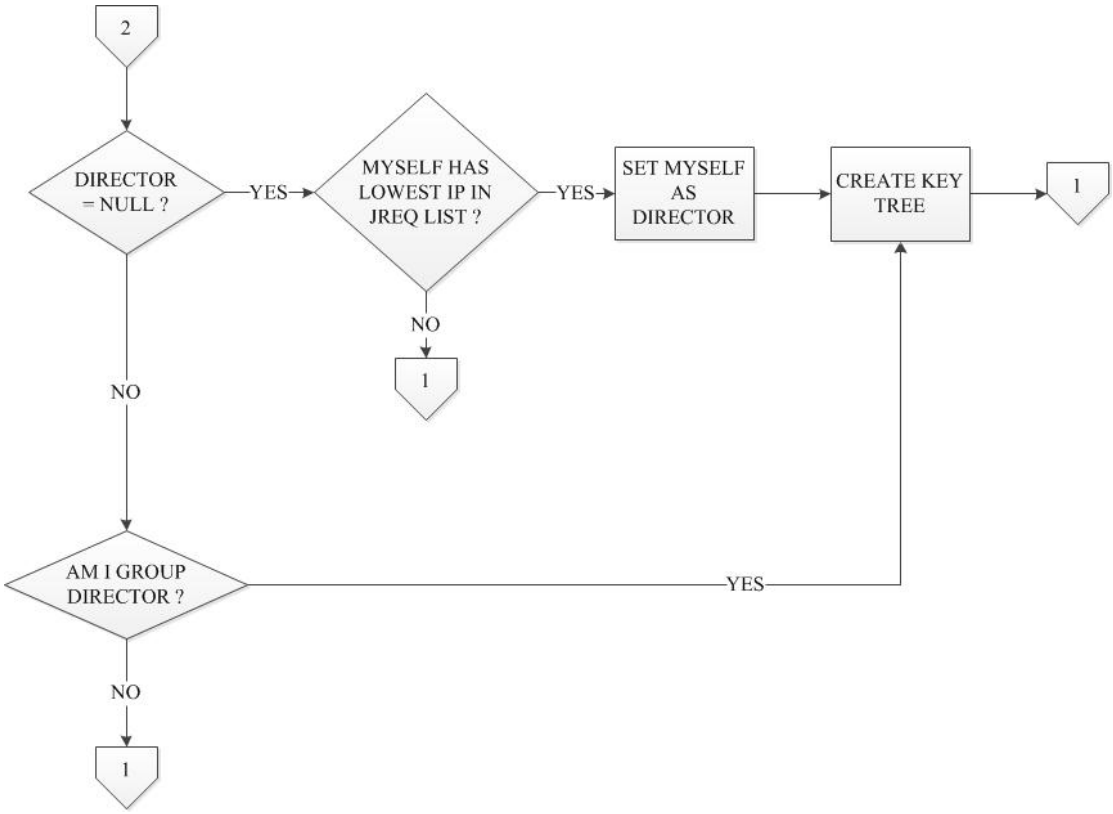

Figure 5.3 Flowchart in Handling Sender Time Out

5.1.4 Receive JREQ Function

This function handles a join request packet at a destination node. The flowchart is shown in Figure 5.4. First of all, it records a sender's IP address and a group to join in join request list. If a receiving node is a director of the group, it updates key tree by adding a new member in the key tree and then broadcast an updated key tree to others. In case that the receiving node has no information about a director of the group and it also participates in the group, it starts a receiver timer, and then goes to a process for selecting a group director after timer is expired.

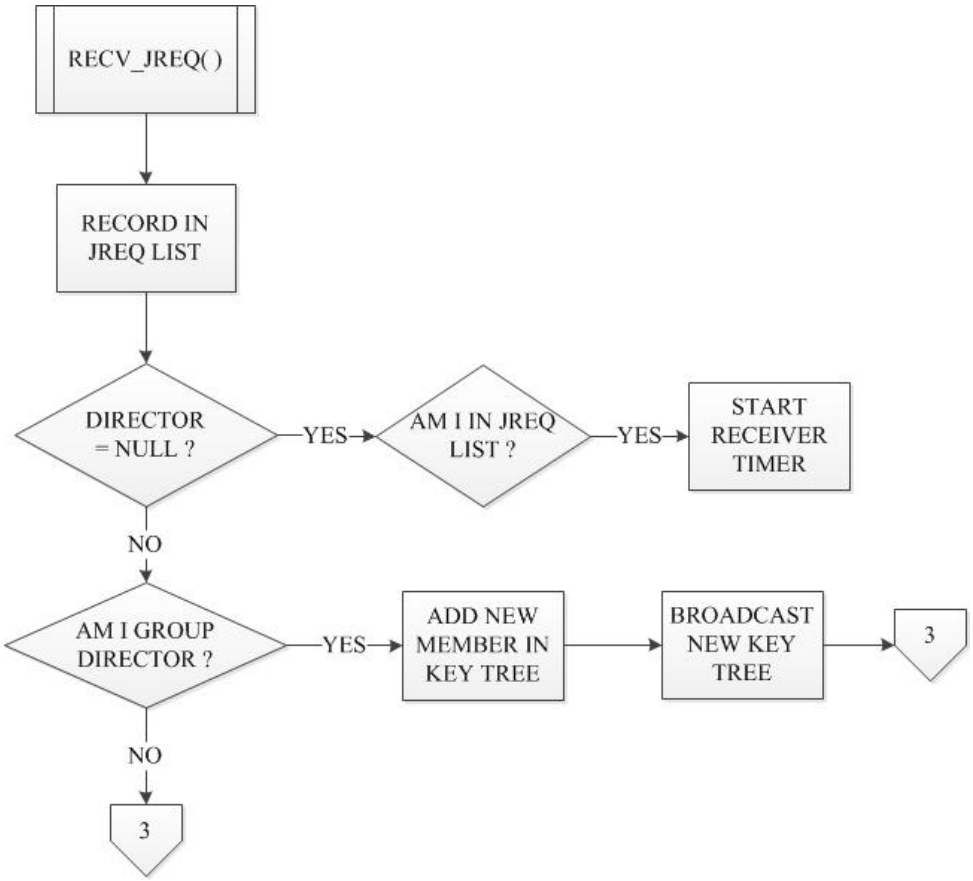

Figure 5.4 Flowchart for Recv_JREQ Function

5.1.5 Receiver Timer

The receiver timer is used to waiting for a while to correct join request messages from others. The flowchart is shown in Figure 5.5. When a receiver timer is expired, the receiver checks whether it is the lowest IP address node in join request list. If it has the lowest IP address, it sets itself as a director of a group. Then it create key tree and broadcast this keytree.

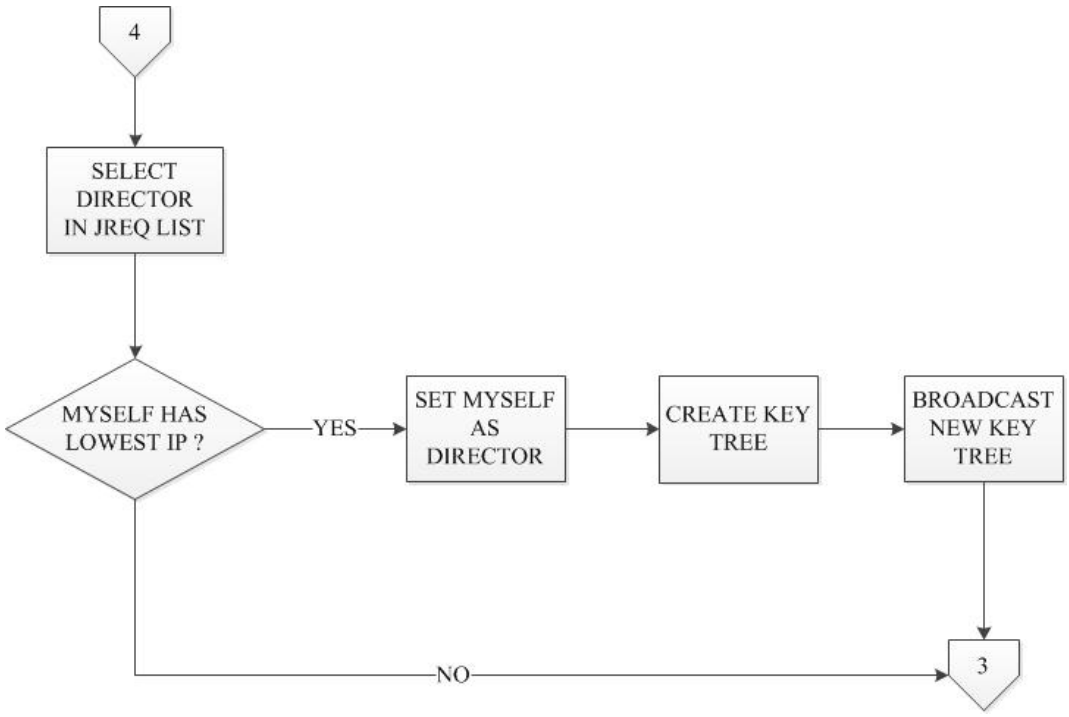

Figure 5.5 Flowchart in Handling Receiver Time Out

5.1.6 Receive LREQ Function

This function handles a leave request packet at a destination node. The flowchart is shown in Figure 5.6. If a receiving node is a director of the group, it updates key tree by removing a requested member from the key tree and then broadcast key tree to others.

5.1.7 Receive UPDATED_TREE Function

This function handles an updated key tree packet at a destination node. The flowchart is shown in Figure 5.7. First, it checks that it has sent join request message for this group or not. If a receiving node has sent a join request for this group, it stops a receiver timer and then updates new key tree.

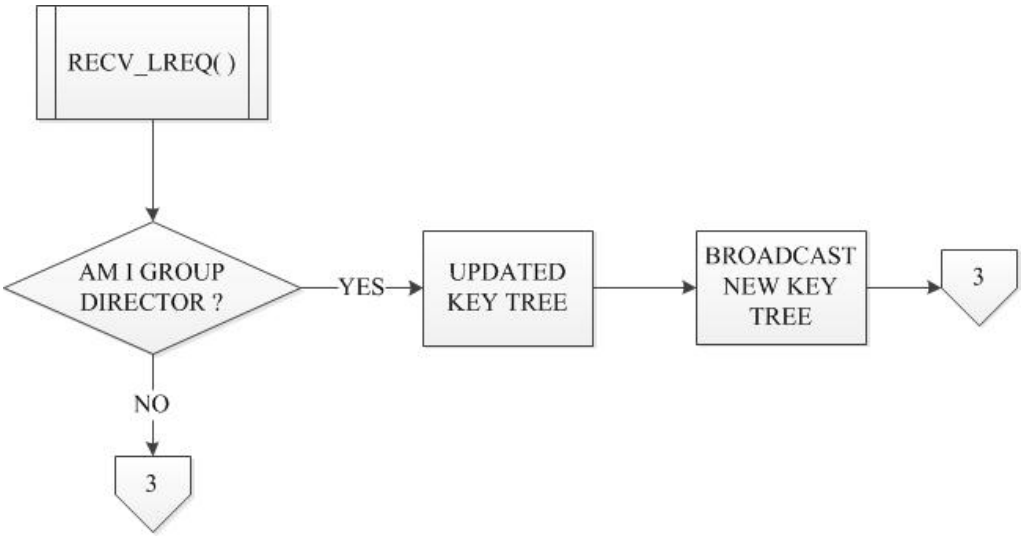

Figure 5.6 Flowchart for Recv_LREQ Function

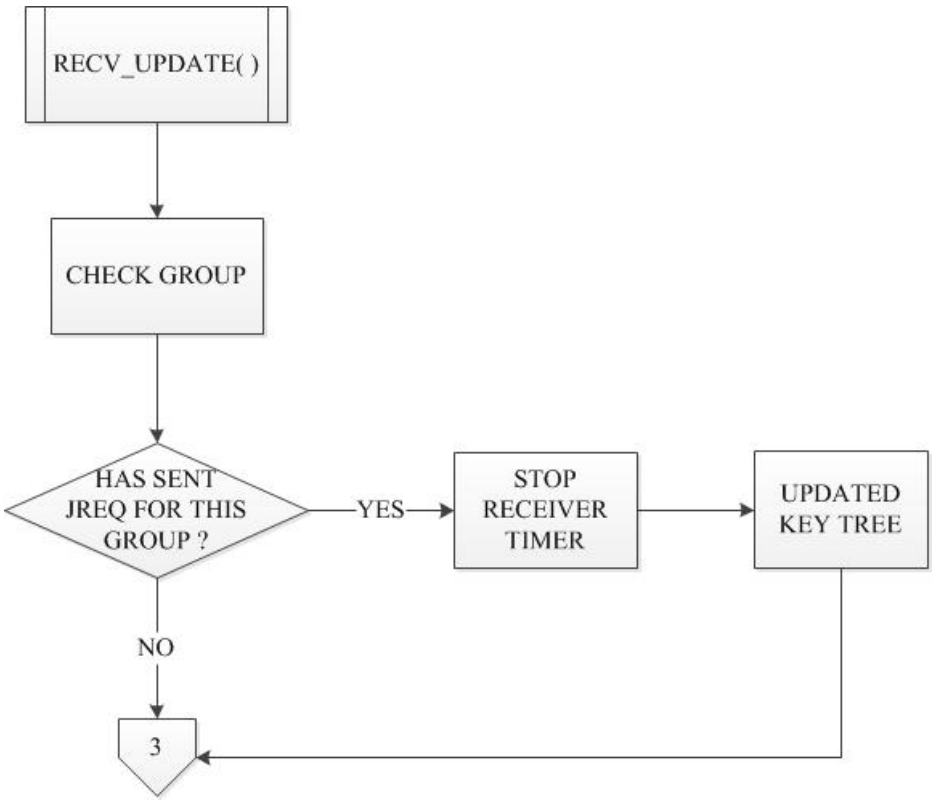

Figure 5.7 Flowchart for Recv_UPDATED Function

5.2 Key Tree Calculation

The implementation in the proposed scheme uses only positive braid word. Each user has private key *i* and *j*, he/she needs to calculate his/her public key PK_U. This PK_U has value equal to $i\ g_o g_u\ j$ where g_u is his/her published braid and g_o is the published braid of another node at the same level in a key tree. The steps for computing this public key PK_U are as the following;

1) Read the value of *i* and *j* in C++ string format.

For example; an input string for *i* is 1 4 3, an input string for *j* is 2 1 1, and assume that $g_o g_u$ is 2 7

2) Convert *i* and *j* to the Artin braid by using the function. WordToBraid(list<sint16> w, sint16 n) where w is a list of characters in string and n is braid index. This function returns an Artin braid object.

For example; a braid for *i* is $\sigma_1\sigma_4\sigma_3$ and a braid for *j* is $\sigma_2\sigma_1\sigma_1$

1) Right multiply the braid *i* with the braid $g_o g_u$ and then make the value of this braid $i\ g_o g_u$ in left canonical form by using function MakeLCF(const Braid&).

For example; a left canonical form of braid $i\ g_o g_u$ is $\sigma_1\sigma_4\sigma_3\sigma_2\sigma_7$

2) Right multiply the braid $i\ g_o g_u$ with the braid *j* and then make the value of this braid in left canonical form. This is a braid PK_U.

For example; a left canonical form of braid PK_U is $\sigma_1\sigma_4\sigma_3\sigma_2\sigma_7 . \sigma_2\sigma_1 . \sigma_1$

Each member in a group can send his/her public key PK_U in left canonical form to a director. In this case, user U sends a value 1 4 3 2 7 . 2 1 . 1 to the director. The director can use the function RightMultiply(const Factor<P>&), got a canonical factor form as an argument to the function, and multiply with other braid to calculate a public group key.

In this case; P_1 is 1 4 3 2 7, P_2 is 2 1, and P_3 is 1. This sequence of factor can be multiplied with other braids.

5.3 Encryption and Decryption

This subsection considers the braid cryptosystem proposed in chapter 3. The encryption and decryption scheme is as follows.

Encryption:

$m' = m_U \oplus H\,(R\ u\ PK_{Group}\ u^{-1})$ where m', $m_U \in \{0, 1\}^M$, and sends m', R and PK_U to members of the group.

Decryption:

$m_D = m' \oplus H\,(R\ K_{ABC}\ PK_D\ (K_{ABC})^{-1})$

In the above, the H: $B_n \rightarrow \{0, 1\}^M$ is a collision-free hash function. The H can be obtained by composing a collision free hash function of bitstrings into $\{0, 1\}^M$. This needs a function to convert braids into bitstrings. A braid written as left canonical form $D^u A_1 \ldots A_l$ can be converted into a bitstring by dumping the integer u and permutation tables of A_i as binary digits for $i = 1, \ldots, l$ sequentially. Because different braids are converted into different bitstrings, this conversion can be used as a part of the hash H.

For example, consider B_3 which have (3! = 6) possible numbers of canonical factors such as $\pi(1, 2, 3)$, $\pi(1, 3, 2)$, $\pi(2, 1, 3)$, $\pi(2, 3, 1)$, $\pi(3, 1, 2)$, $\pi(3, 2, 1)$. Each possible canonical factor can be represented by using three binary digits from 000 to 111. If a key's length is p bits, a message block can be made in p bit block and put both bit blocks into an exclusive-or operation.

CHAPTER 6

CONCLUSION

6.1 Conclusion

The research proposes two schemes; the first scheme is the broadcast encryption based on braid groups and the second scheme is the identity based broadcast encryption based on braid groups. Both schemes are the dynamic asymmetric key agreement scheme for broadcast encryption. The braid group concept was applied in these schemes to reduce the computation complexity, and the identity-based cryptosystem was applied to reduce a system complexity and cost for establishing and managing a public key authentication framework known as the public key infrastructure (PKI). By comparing the first proposed scheme and other schemes in an asymmetric broadcast encryption scheme, the first proposed scheme can support dynamic networks like mobile ad hoc networks. Comparing with another dynamic scheme as in Zhao, Zhang, and Tian (2011), the computation costs in the first proposed scheme are in braid multiplication, but the other are the exponentiation in G. The communication costs for both schemes are not too different.

By comparing the second proposed scheme on the identity based broadcast encryption with the similar scheme as in Du, Wang, Ge and Wang's scheme (2005), we can see that the second proposed scheme can support well in dynamic networks. This is because the second proposed scheme does not require a center for group key management. The communication cost of the second proposed scheme is the same as the other. This is $O(n)$ unicast message in setup and private key extraction phases. The second proposed scheme also includes the join protocol which is taken $O(m)$ in both unicast and multicast messages, and the leave protocol which is taken $O(m)$ in multicast message . The computation cost in both encryption and decryption phases for the second proposed scheme is better than the other in which it uses $O(1)$ in braid group multiplication while the other uses $O(n)$ in both point addition and

multiplication as well as $O(1)$ in paring operation. The second proposed scheme also has an advantage in that it produces a constant ciphertext.

The comparison also was taken between the first proposed scheme in broadcast encryption with the second proposed scheme in the identity based broadcast encryption. Both schemes are asymmetric contributory key agreement scheme. The former scheme requires a key tree for managing a group public key, but the later scheme does not.

6.2 Future Works

There are some points to be concerned for the proposed scheme in the identity based broadcast encryption to make it more efficient. The first one is that the group public key is getting bigger when a group number are increased. The second comes from using the identity based cryptosystem. The identity based cryptosystem has disadvantage in the privacy in distributing user's private key by a private key generator. This can be eliminated by applying the other schemes like a certificateless cryptosystem in the proposed scheme.

BIBLIOGRAPHY

Artin, E. 1947. Theory of Braids. **The Annals of Mathematics.** 48, 1 (January): 101-126.

Boneh, D. and Franklin, M. 2001. **Identity-Based Encryption from the Weil Pairing.** Retrieved March 20, 2011 from http://crypto.stanford.edu/~dabo/papers/bfibe.pdf

Baek , J.; Newmarch, J.; Safavi-naini, R. and Susilo, W. 2004. **A Survey of Identity-Based Cryptography.** Retrieved March 20, 2011 from http://jan.newmarch.name/publications/auug_id_survey.pdf

Cocks, C. 2001. **An Identity Based Encryption Scheme Based on Quadratic Residues, Cryptography and Coding-Institute of Mathematics and Its Applications International Conference on Cryptography and Coding.** Proceedings of IMA, Springer-Verlag. Pp. 360-363.

Cha, J. C.; Ko, K. H.; Lee, S.; Han, J. W. and Cheon, J. H. 2001. **An Efficient Implementation of Braid Groups.** Retrieved October 20, 2011 from http://www.iacr.org/archive/asiacrypt2001/22480144.pdf

Dehornoy, P. 2004. Braid-Based Cryptography. **Contemporary Mathematics.** 360: 5-33.

Du, X.; Wang, Y.; Ge, J. and Wang, Y. 2005. An ID-Based Broadcast Encryption Scheme for Key Distribution. **IEEE Transactions on In Broadcasting.** 51 (2): 264-266.

Elrifai, E. A. and Morton, H. R. 1994. Algorithms for Positive Braids. **Quarterly Journal of Mathematics.** 45 (4): 479-497.

Fiat, A. and Naor, M. 1993. Broadcast Encryption. **Advances in Cryptology-CRYTO 1993.** Springer-Verlag. Lecture Notes in Computer Science. 773: 480-491.

Hoffstein, J.; Pipher, J. and Silverman, J. H. 2008. **An Introduction to Mathematical Cryptography.** New York: Springer.

Kim, Y.; Perrig, A. and Tsudik, G. 2000. **Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups.** Retrieved June 12, 2011 from http://delivery.acm.org/10.1145/360000/352638/p235-kim.pdf?ip=202.29.108.22&acc=ACTIVE%20SERVICE&CFID=136034141&CFTOKEN=73071967&_acm_=1351842178_4f6ffbd0e3bcdb27093427464c56bce6-244.

Ko, K. H.; Lee, S. J.; Cheon, J. H.; Han, J. W.; Kang, J. and Park, C. 2000. **New Public-Key Cryptosystem Using Braid Groups.** Retrieved August 23, 2011 from http://pdf.aminer.org/000/120/173/new_public_key_ cryptosystem_ using_braid_groups.pdf

Karu, P. and Loikkanen, J. 2000. **Practical Comparison of Fast Public-Key Cryptosystems.** Retrieved October 12, 2011 from http://www.tml.hut.fi/Opinnot/Tik-110.501/2000/papers.html

Ma, C. and Ao, J. 2009. **Improved Group-Oriented Encryption for Group Communication.** Retrieved October 12, 2011 from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5194037

Ma, C.; Wu, Y. and Li, J. 2006. **Broadcast Group-oriented Encryption for Group Communication.** Retrieved October 12, 2011 from http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4064210

Naor, D.; Naor, M. and Lotspiech J. 2001. **Revocation and Tracing Schemes for Stateless Receivers.** Retrieved November 12, 2011 from http://www.wisdom.weizmann.ac.il/~naor/PAPERS/2nl.pdf

Norrarut Saguansakdiyotin and Pipat Hiranvanichakorn 2012. Broadcast Encryption Based on Braid Groups. **International Journal of Computer Science and Network Security.** 12 (February): 12-19.

Shamir, A. 1984. **Identity-Based Cryptosystems and Signature Schemes.** Retrieved November 12, 2011 from http://discovery.csc.ncsu.edu/ Courses/csc774-S07/shamir84.pdf

Thanongsak Aneksrup and Piapt Hiranvanichakorn. 2011. Efficient Group Key Agreement on Tree-based Braid Groups. **Computer and Information Science.** 4 (January): 14-27.

Wallner, D.; Harder, E. and Agee, R. 1997. **Key Management for Multicast: Issues and Architecture.** Retrieved November 23, 2011 from http://tools.ietf.org/html/draft-wallner-key-arch-00.

Wu, Q.; Mu, Y.; Susilo, W.; Qin, B. and Domingo-Ferrer; J. 2009. **Asymmetric Group Key Agreement.** Retrieved November 23, 2011 from http://www.iacr.org/archive/eurocrypt2009/54790154/54790154.pdf

Zhao, X.; Zhang, F. and Tian, H. 2011. Dynamic Asymmetric Group Key Agreement for ad Hoc Networks. **Ad Hoc Networks.** 9, 5 (July): 928-939.

BIOGRAPHY

NAME	Norranut Saguansakdiyotin
ACADEMIC BACKGROUND	M.S. (Software Engineering), West Virginia University, USA. M.S. (Computer Science), Rangsit University, Thailand**.** B.S., Chulachomklao Royal Military Academy, Thailand.
PRESENT POSITION	Lecturer in Computer Engineering Department, Siam University.
EXPERIENCES	Network Engineer, Olympiathai Co., Ltd.
PUBLICATION	1. Norranut Saguansakdiyotin and Pipat Hiranvanichakorn. 2011. Broadcast Group-Oriented Encryption based on Braid Groups. In 2011 the 7th IMT-GT International Conference on Mathematics, Statistics and its Applications (ICMSA2011). July 21-23, Bangkok Thailand.

2. Norranut Saguansakdiyotin and Pipat Hiranvanichakorn. 2012. Broadcast Encryption Based on Braid Groups. In 2012 International Journal of Computer Science and Network Security (IJCSNS). Volume 12, Number 2, February 2012, Page 12-19.