International Journal of Pure and Applied Mathematics
Volume 16 No. 4 2004, 513-531

TWO WAY AUTHENTICATION KEY AGREEMENT BASED ON BRAID GROUPS

Denis Goh Chuan Hu[1], Azman Samsudin[2] §
[1,2]School of Computer Science
University of Science at Malaysia
Penang, 11800, MALAYSIA
[1]e-mail: denis@cs.usm.my
[2]e-mail: azman@cs.usm.my

Abstract: Public Key Cryptosystem (PKC) is an algorithmic method used to transmit private data over an insecure channel in which both ends do not share a common key. The notable PKC algorithms include RSA, Diffe-Hellman and Elliptic Curve Cryptosystem. However, all these algorithms are based on the finite Abelian group theory. Lately, the braid groups theory has also been used in developing PKC. The braid groups theory is based on the concept of intertwining strands extended between two parallel planes and can be represented in terms of braid words. A classical problem that arose from this work is the word conjungacy problem. This paper presents a new authentication and key-exchange protocol suitable for authenticating users and exchanging keys over an insecure communication. This new protocol is based on braid groups. The main objective of this paper is to give the audience an alternative method

Received: September 10, 2004

§Correspondence author

in doing PKC and not just confine themselves to the Abelian group theory.

AMS Subject Classification: 20F36
Key Words: braid groups, key exchange, authentication problems, conjugacy problems

1. Introduction

The main problem of Public Key Cryptography is to create protocols for securely transferring private keys over an insecure channel. For this, we have the Public Key Cryptosystem (PKC). PKC is an algorithmic method used to transmit private data over an insecure channel in which both ends do not share a common key. The most notable PKC algorithm is RSA [13]. Others include Diffe-Hellman Key Exchange [7] and Elliptic Curve Cryptosystem [3, 4]. All these algorithms are used for authentication and common key generation, which will be used to encrypt and decrypt messages in order to ensure they are not being exposed. In other words, an intruder observing the communication will not be able to deduce any useful information that has been transferring between both ends and at the same time we can be assured that the other party is who she claims to be. However, these algorithms are mostly based on the difficulty in factoring prime numbers and discrete logarithms.

In recent years, several researchers are looking into the possibility of using non commutative groups, in particular Artin's braid groups to build a secure cryptosystem [15], [1], [10]. A lot of interest has been focus on incorporating conjugacy problems in braid groups into the cryptosystem as it is algorithmically more difficult and at the same time it provides a one-way functions [4]. However, using braid over finite Abelian group theory in cryptosystem is still in its infancy stage and has not yet proven itself in the enterprise arena. The most notable of these systems has been the Birman-Ko-Lee system [4].

Section 2 gives us a brief introduction on braid groups, followed by a list of hard problems that can be found in braid groups in Section 3. Section 4 provides a review of some of the cryptosystems that use braid groups. Section 5 introduces a new authentication framework. Section 6 discusses the new authentication protocol in mathematic terms and its handshaking process. Section 7 gives us a security analysis on the two way authentication key agreement protocol.

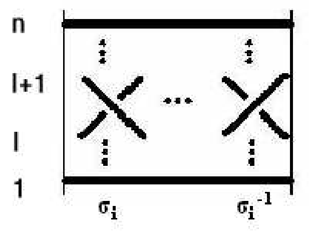

Figure 1: Braid diagram associated with braid generator σ_i and σ_i^{-1}

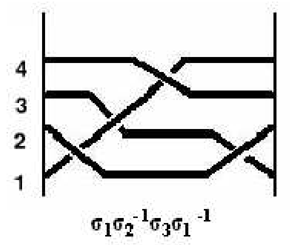

Figure 2: Braid diagram associated with braid generator $\sigma_1\sigma_2^{-1}\sigma_3\sigma_1^{-1}$

2. About Braid Groups

Braid theory was invented in 1925 by Emil Artin [3]. A mathematical braid can be viewed as strands extended between two parallel planes. The strands are interwined with each other. The characteristic of braid is that only adjacent strands may cross each other and only one crossing occurs at a time. Each crossing in the geometrical braid can be denoted by a braid generator (strand is crossing under the $(i+1)$-th strand, we have the braid generator σ_i and for i-th crossing over $(i+1)$-th we have σ_i^{-1}. The resulting string of symbols is called the braid word, see Figure 1 and Figure 2 for illustration.

The number of strands is called the braid index, and the set of all braids on q_n strands is denoted as B_n. This is also sometimes known as Artin's braid group B_n, which got its name after Emil Artin.

Braids have three Reidemeister moves that are ambient isotopic. Ambient isotopic refers to a smooth and continuous deformation of the string and the string is being deformed through the three dimensional space that it sits in, see Figure 3 for illustration.

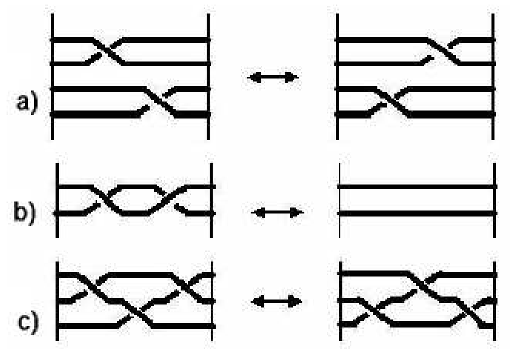

Figure 3: The three Reidemeister moves for braids

a) $\sigma_i\sigma_j = \sigma_j\sigma_i$ if $|i-j| > 1$.

b) $\sigma_i\sigma_i^{-1} = e$.

c) $\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}$.

This leads us to the following definition.

Definition 1. For n ≥ 2, the braid group B_n is defined by the presentation

$$B_n = \langle \sigma_1 \ldots \sigma_{n-1}; \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geq 2, \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ for } |i-j| = 1\rangle .$$

A braid can also be represented using permutation. In this form, given a braid a, let the strand starting from the i-th upper position ends at the i-th lower position. See Figure 4 for illustration of a permutation braid representation. Here, we shall use the horizontal planes as illustration. Note, the following three diagrams were taken from [2].

The multiplication of two braids ab is obtained by positioning braid a on top of braid b. See Figure 5 for illustration of multiplying two braids. An inverse braid a^{-1} is actually the reflection of braid a with respect to the horizontal line. See Figure 6 for illustration of an inverse braid.

3. Hard Problems in Braid Groups

There exist several mathematically hard problems that can be found in braid groups. For this paper, we are only interested in the conjugacy problems that

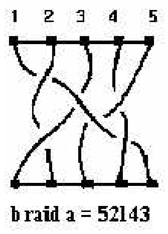

Figure 4: A Permutation Braid Representation

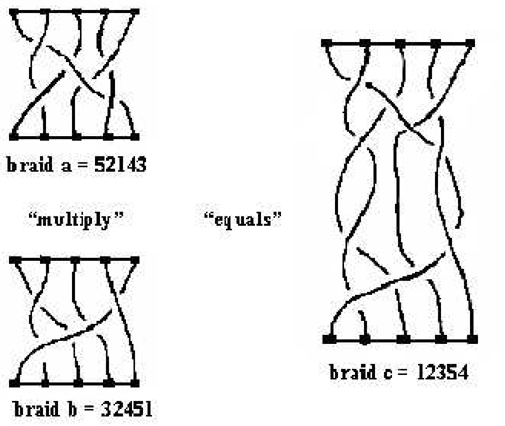

Figure 5: Multiplying two braids

are useful for developing a cryptosystem. We shall only list down four mathematical hard problems namely Conjugacy Decision Problem (CDP), Conjugacy Search Problem (CSP), Generalized Conjugacy Search Problem (GCSP) and Conjugacy Decomposition Problem (CDP). More information on braid groups, word problem and conjugacy problems can be found in [4], [3], [6].

1. CONJUGACY DECISION PROBLEM (CDP)
Instance: $(x, y) \in B_n \times B_n$ such that $y = a \times a^{-1}$ for $a \in B_n$.
Objective: Determine whether x and y are conjugate or not.

2. CONJUGACY SEARCH PROBLEM (CSP)
Instance: $(x, y) \in B_n \times B_n$ such that $y = a \times a^{-1}$ for $a \in B_n$.
Objective: Find $b \in B_n$ such that $y = b \times b^{-1}$.

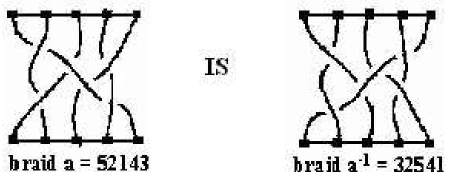

Figure 6: An inverse braid a

3. GENERALIZED CONJUGACY SEARCH PROBLEM (GCSP)
Instance: $(x, y) \in B_n \times B_n$ such that $y = a \times a^{-1}$ for some $a \in B_m$, $m \leq n$.
Objective: Find $b \in B_m$ such that $y = b \times b^{-1}$.

4. CONJUGACY DECOMPOSITION PROBLEM (CDP)
Instance: $(x, y) \in B_n \times B_n$ such that $y = a \times a^{-1}$ for some $a \in B_m$, $m \leq n$.
Objective: Find $b_1, b_2 \in B_m$ such that $y = b_1 \times b_2$.

Our main focus will be on Conjugacy Search Problem and Generalized Conjugacy Search Problem. At present, it is considered infeasible to solve the Conjugacy Search Problem for sufficiently large braid [8]. The Generalized Conjugacy Search Problem is a generalized version of the Conjugacy Problem, which has a restriction on the braid that conjugates x. There is also no known polynomial time algorithm to solve this problem given a sufficiently large braid [8].

4. Related Works

In this section, our focus will be on Ko-Lee [10] work with regards to braid groups' cryptography. This paper gives us a brief explanation on the key agreement as well as the enciphering and deciphering of messages introduced by Ko-Lee using braid groups. On top of that, we shall look at an authentication protocol based on this concept.

4.1. A Diffie-Hellman-Like Key Agreement

In cryptography terms, key agreement is known as having two parties at different ends traditionally called A(lice) and B(ob), being able to compute and derive a common key in both ends in such a way that an intruder observing the communication will not be able to deduce any useful information about the common key. The common key will then be used as the symmetric key to encrypt and decrypt messages between the two ends. The most well known key agreement is the Diffie-Hellman Key Exchange [7]. Figure 7 describes the Diffie-Hellman Key Exchange.

A Diffie-Hellman-Like key agreement based on braid groups was proposed by Ko at al in [10]. It is based on the Generalized Conjugacy Search Problem. In general, braid groups are non commutative, however they contain large subgroups in such a way that each element of the first subgroup commutes with each element of the second subgroup. For instance, there are two subgroups LB_l and RB_r of B_{l+r} for some appropriate pair of integers l, r. LB_l (resp. RB_r) is the subgroup of B_{l+r} consisting of braids made by braiding left l strands (resp. right r strands) among $l+r$ strands. The LB_l is generated by $\sigma_1, \ldots, \sigma_{l-1}$ and RB_r is generated by $\sigma_{l+1}, \ldots, \sigma_{l+r-1}$. It is known that for any $a \in LB_l$ and $b \in RB_r$, $ab = ba$. In other words, every braid in LB_l commutes with every braid in RB_r. Based on this concept, Ko at al in [10] proposed the following one-way function:

$$f : LB_l \times B_{l+r} \to B_{l+r} \times B_{l+r}, \quad f(a,x) = (a \times a^{-1}, x)\,.$$

What this means is that if we are given the pair of (a, x), it is very easy for us to compute $a \times a^{-1}$. However, knowing the pair of $(a \times a^{-1}, x)$, it takes exponential time for us to compute the value a. Based on this fundamental concept, Ko at al in [10] build the Diffie-Hellman-like Key Exchange as illustrated by Figure 8.

Similarly, Ko at al Key Exchange protocol begins with A(lice) chooses a random secret braid s in LB_n and another braid p in B_n. Braid p shall be made public. B(ob) on the other hand will also chooses a random secret braid which acts as his private key r in RB_n. Figure 8 illustrates the steps taken in the Ko at al Key Exchange Agreement. The following mathematics proof the key generated in both ends are identical.

Theorem 2. *t_A is equal to t_B.*

Proof.

$$\begin{aligned} t_A &= sp''s^{-1} \\ &= srpr^{-1}s^{-1} \quad \text{(based on } p'' = rpr^{-1}\text{)} \\ &= srps^{-1}r^{-1} \\ &= rp'r^{-1} \qquad \text{(based on } p' = sps^{-1}\text{)} \\ &= t_B. \end{aligned}$$

The security of this key agreement as stated in [10] is based on the difficulty of the following problem.

Global Public Element
q: prime number
g: $g < q$ and g is a primitive root of q

Alice		**Bob**
1) Select a private key X_a whereby $X_a < q$		
2) Compute public key Y_a $Y_a = g^{X_a}\ mod\ q$	$\xrightarrow{Y_a}$	3) Select a private key X_b whereby $X_b < q$
		4) Compute public key Y_b $Y_b = g^{X_b}\ mod\ q$
6) Compute common key K $K_2 = {Y_b}^{X_a}\ mod\ q$	$\xleftarrow{Y_b}$	5) Compute common key K $K_1 = {Y_a}^{X_b}\ mod\ q$

Figure 7: The Diffie-Hellman Key Exchange Protocol

KO-LEE PROBLEM

Instance: The triple (x, y_1, y_2) of elements in B_{l+r} such that $y_1 = axa^{-1}$ and $y_2 = bxb^{-1}$ for some hidden $a \in LB_l$ and $b \in RB_r$.

Objective: Find $by_1b^{-1}(ay_2a^{-1} = abxb^{-1}a^{-1} = baxa^{-1}b^{-1} = by_1b^{-1})$.

This problem is equivalent to the Generalized Conjugacy Search Problem in B_n and hence it is considered a hard problem. The role of x in KO-LEE Problem is similar to that of g in the Diffie-Hellman problem to find g^{xy} from g^x and g^y. Ko at al in [10] pointed out that in order to make the KO-LEE Problem hard, x must be sufficiently complicated by avoiding the "reducible" braids x_1x_2z, where $x_1 \in LB_l$, $x_2 \in RB_r$ and z is a $(l+r)$-braid that commutes with both LB_l and RB_r as depicted in Figure 9 for $l = r = 3$.

On the other hand, Ko at al in [10] also mentioned that the probability for a randomly chosen $(l + r)$-braid of canonical length q to be reducible is small, that is, roughly $\left(\frac{l! \times r!}{(l+r)!}\right)^q$.

Alice		**Bob**
1) Computes braid		2) Computes braid
$p' = sps^{-1}$	$\xrightarrow{p'}$	$p'' = rpr^{-1}$
4) Computes t_A		3) Computes t_B
$t_A = sp''s^{-1}$	$\xleftarrow{p''}$	$t_B = rp'r^{-1}$

Figure 8: Ko at al Key Exchange Protocol

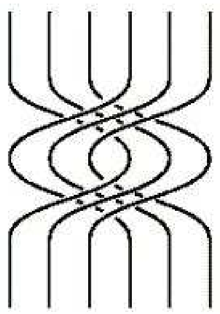

Figure 9: An example of a reducible braid

4.2. Enciphering-Deciphering

In cryptography terms, enciphering and deciphering are known as having two parties at different ends traditionally called A(lice) and B(ob) being able to exchange messages across the insecure communication without worrying an intruder being able to read the messages. Typically, this is done using the public key of the corresponding party to encrypt the messages and the encrypted messages are sent over the insecure communication. The only person who can decrypt back to the original message is the corresponding party and no one else.

Based on the difficulty of the Generalized Conjugacy Search Problem in B_n and the KO-LEE Problem as described in the previous section, Ko at al in [10] came out with a mechanism to encrypt and decrypt messages using braid groups. For this mechanism, we assume that h is a collision-free one-way hash functions of B_n, that is for any given value h, it is computationally infeasible to find x such that $h(x) = h$ (one-way) and for any given x, it is computationally infeasible to find y such that $h(y) = h(x)$ (collision free). A detail explanation on a collision-free one-way hash function of B_n can be found in [9].

The Ko at al enciphering and deciphering protocol begins with one party assuming it is A(lice) generating a pair of public key (p, p') and a secret key s. p lies in B_n and s in LB_n. p' is generated by computing the conjugate, that is $p' = sps^{-1}$. The pair of (p, p') will then be published. Assuming B(ob) would like to send a message M_A to Alice over an insecure communication. The followingsm (Figure 10) are the steps taken to ensure the message, M_A can only be read by Alice.

Alice		**Bob**
		1) Chooses a random braid r in UB_n
		2) Computes $p'' = rpr^{-1}$
4) Computes $M_B = M'' \oplus h(sp''s^{-1})$	$\xleftarrow{M'',p''}$	3) Computes $M'' = M_A \oplus h(rp'r^{-1})$

Figure 10: Ko at al enciphering and deciphering protocol

In order to proof that M_A is actually M_B, we need only to proof that $sp''s^{-1} = rp'r^{-1}$. This is because we are assuming that the hash function h used by both parties is the same.

Theorem 3. *M_A is equal to M_B.*

Proof.

$$\begin{aligned} sp''s^{-1} &= srpr^{-1}s^{-1} && \text{(based on } p'' = rpr^{-1}\text{)} \\ &= rsps^{-1}r^{-1} && \text{(true because braids } r \text{ and } s \text{ commute)} \\ &= rp'r^{-1} && \text{(based on } p' = sps^{-1}\text{).} \end{aligned}$$

The security of this is again based on the difficulty of the generalized Conjugacy Search Problem as well as the KO-LEE Problem in B_n and also owing to the hypotheses on h in [9].

4.3. A Diffie-Hellman-Like Authentication Scheme

In cryptography terms, authentication means having one party or both parties to proof the identity of oneself. This is usually done by forcing one party to

Alice		**Bob**
		1) Chooses a random braid r in UB_n
3) Computes		2) Computes
$y = h(sp''s^{-1})$	$\xleftarrow{p''}$	$p'' = rpr^1$
		4) Verify
	$\xrightarrow{y}$	$y = h(rp'r^1)$

Figure 11: A Diffie-Hellman-like Authentication Protocol

sign a message(s) using some secret information that is only known to that particular party. This secret information is regarded as the private or secret key of that party. An intruder watching the communication should not be able to deduce anything about the private key. In this section, we shall focus on an authentication scheme that has been proposed using the braid groups based on the difficulty of the KO-LEE Problem.

The following challenge-response scheme mentioned in [14] is actually an adaptation of the Diffie-Hellman-like Key Exchange proposed by Ko at al [10]. In the begining, Alice will have to generate a public key pair of (p, p') in B_n. The secret key of Alice is a braid s in LB_n. The relation between p, p' and s is that p is a braid in B_n and s is a braid in LB_n. Braid $p' = sps^{-1}$.

For Bob to authenticate Alice, he would have to choose a random braid r in UB_n, and sends the challenge $p'' = rpr^{-1}$. p is actually part of the public key of Alice (p, p').

Alice will then have to response to Bob's challenge by signing the message with her private key with $h(sp''s^{-1})$. We still use h for a collision-free one way hash functions on B_n. The $h(sp''s^{-1})$ value will then sends back to Bob. Bob would then verify this against his computed results of $h(rp'r^{-1})$. Theorem 4 proofs that $sp''s^{-1}$ is equal to $rp'r^{-1}$. The following Figure 11 illustrates the protocol of a Diffie-Hellman-like authentication.

Theorem 4. *$sp''s^{-1}$ is equal to $rp'r^{-1}$.*

Proof.

$$\begin{aligned} sp''s^{-1} &= srpr^{-1}s^{-1} && \text{(based on } p'' = rpr^{-1}\text{)} \\ &= rsps^{-1}r^{-1} && \text{(true because braids } r \text{ and } s \text{ commute)} \\ &= rp'r^{-1} && \text{(based on } p' = sps^{-1}\text{).} \end{aligned}$$

The security of this is again based on the difficulty of the generalized Conjugacy Search Problem as well as the KO-LEE Problem in B_n and also owing to the hypotheses on h in [9].

5. A New Framework

The objective in this section is to come up with a design that allow us to achieve a two way authentication between the Client (Alice) and the Server (Bob) and at the same time being able to generate a symmetrical key or sometimes known as the session key between both parties. One can view this as an extension to the work carried out by [10], [9], [14]. The proposed design should fulfill the following desirable properties.

— Two Way Authentication
— Two Way Challenge-Response
— Integrity
— Perfect Forward Secrecy
— Resistance to Known-Key Attacks

All of these properties are important in designing our system because it ensures resistance to active attackers whereby an attacker can monitor the communication channel as well as injecting, deleting, altering or replaying messages. The communication channel is deems as open or insecure.

Here, we would like to give some definition and terminology regarding the terms used. Authentication here refers to one party being able to correctly identify the other party. Two way authentication simply means having both parties identifying each other. The assumption made here is that there exists certain information or secret knowledge that is only known to each party. With this, A(lice) is assured that no other party apart from the other party, B(ob) can possibly learn the value of the secret knowledge which is also known as the secret key. Two way challenge-response means both parties challenging each other and at the same time responding to each other's challenge. In our system, challenge means forcing the other party to sign certain information and response means having the other party signs the challenge information using her secret information that is only known to her. Integrity here assumes that A(lice) and B(ob) are two honest parties who execute the steps of the given protocol correctly. A key agreement protocol is a key establishment technique that derives a common shared key among both parties such that an intruder observing the communication will not be able to deduce any useful information about the common key. A protocol is said to have perfect forward secrecy if the condition in which the compromise of a session key or long-term secret key after a given session does not cause the compromise of any earlier session. Forward Secrecy is usually achieved by the two communicating parties agreeing on a temporary session key which is unique for each session. The session key will normally be automatically generated as part of the communication protocol. For Perfect Forward Secrecy (PFS) it must not be possible to determine any previous session keys even if one or both of the private keys used to derive them are known. A protocol is said to be vulnerable to known-key attack if compromise of past session keys allows either a passive adversary to compromise future session keys, or an active adversary to impersonate one of the protocol parties.

6. The Two Way Authentication Key Agreement (2W-A-KA) Protocol

The Two Way Authentication Key Agreement (2W-A-AKA) protocol on braid groups is actually an adaptation of the Ko at al Diffie-Hellman-like Key Agreement and enciphering and deciphering protocol [10]. What follow is a complete description of the entire 2W-A-KA protocol from initialization to the actual handshaking process.

Step 1. (Client key generation, offline or using a secure channel) The 2W-A-KA begins with the Client selecting one braid p in B_n and another braid s in L_n. Braid s will be used as the Client's secret key. Braid p must be sufficiently complicated by avoiding the "reducible" braids as mention in the previous section.

The Client computes the conjugate $p' = sps^{-1}$.

The (p, p') pair shall be used as the Client's public key or can be treated as the Client's password verifier. This information will then be sent to the Server through a secure channel. The Server will then keep the (p, p') pair of the Client and ties it with the Client's user login and store it inside the password verifier database.

User login	Value 1	Value 2
Client	p	p'

Step 2. (Server key generation) Server selects one braid q in B_n and another braid t in LB_n. Braid t will be used as the Server's secret key. Again, braid q must be sufficiently complicated by avoiding the "reducible" braids as mention in the previous section.

Server computes the conjugate $q' = tqt^{-1}$.

The (q, q') pair shall be used as the Server's public key and have the Certificate Authority (CA) certifying it. This information can then be made public. Alternatively, it can also be passed to the client through a secure channel or whatever means as long as the Client can know for sure the public key actually belongs to the Server.

Step 3. (Actual communication) The actual communication begins when the Client connects to the Server. See Figure 12 for the full 2W-A-KA protocol handshaking process.

Client		**Server**
1) User connects to server	$\xrightarrow{User\ login}$	2) Based on user_login, retrieves user's p and p' value
		3) Chooses a random braid r in UB_n
5) Verifies the public key		4) Computes the conjugate $p'' = rpr^{-1}$
of server (q, q') with the Certificate Authority	$\xleftarrow{p'',(q,q')}$	
6) Choose a random braid d in UB_n		
7) Computes the conjugate $q'' = dqd^{-1}$		
8) Computes the conjugate $q''' = dq'd^{-1}$		
9) Computes the conjugate $p''' = sp''s^{-1}$		
10) Computes $M_a = q''' \oplus p'''$		
11) Computes the hash $C_a = h(M_a)$		13) Computes $y = rp'r^{-1}$
12) Computes $z = q'' \oplus p'''$	$\xrightarrow{C_a,z}$	14) Computes $q'' = z \oplus y$
		15) Computes the conjugate $x = tq''t^{-1}$
		16) Computes $M_b = x \oplus y$
		17) Computes the hash $C_b = h(M_b)$
21) Computes the hash $H' = h(q''')$		18) Compares $C_b = C_a$
22) Compare $H = H'$	$\xleftarrow{H}$	19) Computes the hash $H = h(x)$
23) Computes the session key $K_2 = h(p''')$		20) Computes the session key $K_1 = h(y)$

Figure 12: Two Way Authentication Key Agreement Protocol

Step 3a. (Client connects to Server) Client logs into the Server and sends its user login.

Step 3b. (Server receives message from Client) Based on the user login, the Server retrieves the value p and p' from its password verifier database. If the Client is a valid user then the Server will challenge the Client.

— The Server chooses a random braid r in $UB_{n.}$.
— The Server computes the conjugate $p'' = rpr^{-1}$.
— Next, it sends p'' together with it's public key pair (q, q') to the Client.

Step 3c. (Client receives messages from Server) The Client verifies the public key pair (q, q') with the CA to ensure the Server is genuine. This part is optional because the public key of the Server can be passed to the Client through a different means. Next, the Client will challenge the Server as well as responding to the Server's challenge. The Client chooses a random braid d in UB_n. The Client computes the following conjugates:

$$\begin{aligned} q'' &= dqd^{-1}\,, \\ q''' &= dq'd^{-1}\,, \\ p''' &= sp''s^{-1}\,. \end{aligned}$$

Next, the XOR, $\oplus$, operation is performed.

$$\begin{aligned} M_a &= q''' \oplus p'''\,, \\ z &= q'' \oplus p'''\,. \end{aligned}$$

Before sending to the Server, we shall increase the security by hashing the message M_a. The hashing function used is the same as those proposed by Ko at al in [10].

$$C_a = h(M_a)\,.$$

Step 3d. (Client sends C_a and z to Server) Server computes the followings:

$$\begin{aligned} y &= rp'r^{-1}\,, \\ q'' &= z \oplus y\,, \\ x &= tq''t^{-1}\,, \\ M_b &= x \oplus y\,, \\ C_b &= h(M)_b\,. \end{aligned}$$

If $C_a = C_b$ then we have the rightful Client. If Server successfully authenticated the Client then the Server will compute the hash of the value x and sends to the Client. This is done in response to the Client's challenge, $H = h(x)$.

Step 3e. (Server sends H to Client) Client computes $H' = h(q''')$. If the value of H is equal to H' then the Client has successfully authenticated the Server. Now the Client can trust the Server.

Step 3f. (Client and Server compute the common shared key) The Client computes the following hash value: $K_1 = h(p''')$. The Server computes the following hash value: $K_2 = h(y)$. Both K_1 and K_2 will come out to be equal. Lets take a look at the mathematics. Firstly, we need to proof that C_a is equal to C_b.

Theorem 5. *C_a is equal to C_b.*

Proof.

$$\begin{aligned}
C_a &= h(M_a) && \text{(from Step 3c)}\\
&= h(q''' \oplus p''') && \text{(from Step 3c whereby}\\
& && \quad M_a = q''' \oplus p''')\\
&= h(dq'd^{-1} \oplus sp''s^{-1}) && \text{(from Step 3c)}\\
&= h(dtqt^{-1}d^{-1} \oplus srpr^{-1}s^{-1}) && \text{(from Steps 2 and 3b)}\\
&= h(tdqd^{-1}t^{-1} \oplus rsps^{-1}r^{-1}) && \\
&= h(tq''t^{-1} \oplus rp'r^{-1}) && \text{(from Steps 3c and 1)}\\
&= h(x \oplus y) && \text{(from Step 3d)}\\
&= h(M_b) && \\
&= C_b\,. &&
\end{aligned}$$

With this, we have actually proven that C_a is indeed equal to C_b. Hence, the Server has authenticated the Client successfully because only the Client has the knowledge of braid s and no one else does. Next, we need to proof that H is equal to H'.

Theorem 6. *H is equal to H'.*

Proof.

$$\begin{aligned}
H &= h(x) && \text{(from Step 3d)}\\
&= h(tq''t^{-1}) && \text{(from Step 3d)}\\
&= h(tdqd^{-1}t^{-1}) && \text{(from Step 3c)}\\
&= h(dtqt^{-1}d^{-1}) && \\
&= h(dq'd^{-1}) && \text{(from Step 2)}\\
&= h(q''') && \text{(from Step 3c)}\\
&= H' && \text{(from Step 3e)}\,.
\end{aligned}$$

If H is equal to H' then the Client has successfully authenticated the Server because only the Server has the knowledge of braid t. Next we need to ensure that both key generated in both ends are actually identical. For this, we need to proof that K_1 is equal to K_2.

Theorem 7. *K_1 is equal to K_2.*

Proof.

$$\begin{aligned}
K_1 &= h(p''') && \text{(from Step 3f)}\\
&= h(sp''s^{-1}) && \text{(from Step 3c)}\\
&= h(srpr^{-1}s^{-1}) && \text{(from Step 3b)}\\
&= h(rsps^{-1}r^{-1}) && \\
&= h(rp'r^{-1}) && \text{(from Step 1)}\\
&= h(y) && \text{(from Step 3d)}\\
&= K_2 && \text{(from Step 3f)}\,.
\end{aligned}$$

7. Security Analysis

Since the 2W-A-KA is based on the adaptation of Ko at al. Diffie-Hellman-like Key Exchange scheme, the security strength is also based on the difficulty of the Conjugacy Search Problem in B_n, more preciously, it's based on the difficulty of the KO-LEE Problem. Thus, whatever assumption made by Ko at al with regards to his key exchange scheme will also hold true for the 2W-A-KA protocol. Equally true is the hypotheses on the hash function h [9]. In [5], it is suggested that we work in B_{80} with braids specified using sequences of length 12, that is sequences of 12 permutation.

The introduction of the random generation of braid r and d in the 2W-A-KA protocol is to ensure that the protocol is not vulnerable to known-key-attack. Based on the 2W-A-KA protocol, we know that the value of braid d is randomly generated by the Client and braid r is randomly generated by the Server. Assuming there is no flaw in the random generator, we can be sure that every message that flows between the two parties will only be good for that session. On top of that, we are assured that this protocol also fulfill the perfect forward secrecy criteria. This is true because even if the private keys of both the Client and Server are to be exposed at a later time, there is no way for us to determine the session key, that is K_1 or K_2, assuming that we do not have any knowledge of braid r and d.

The 2W-A-KA protocol ensures that upon a successful run, an eavesdropper will not have any useful information about the session key K. Since the derivation of K is based on the KO-LEE problem, which is a hard problem, we can be sure that the session key K is a cryptographically strong key. Hence, we are not concerned about guessing attacks on K, as long as K cannot be computed directly by an intruder.

Assuming there is an intruder which has the ability to create his own messages and make them appear to originate from one of the parties, the 2W-A-KA protocol will still prevent the intruder from gaining access to the host or learning any information about the actual party's secret key. At best, the intruder can only cause a denial-of-service-attack (keep getting an unsuccessful authentication).

Even if the Server's password file is being captured and the intruder learns the value (p, p'), he still could not impersonate as the Client easily. This is back to the difficulty on the Conjugacy Search Problem in B_n, that is, knowing p and p' (where $p' = sps^{-1}$), it is still very difficult to determine s.

8. Conclusion

This paper presents a new authentication and key-exchange protocol suitable for authenticating users and exchanging keys over an insecure communication. This new protocol is based on the braid groups theory. The 2W-A-KA protocol demonstrates to us that we can have an alternative cryptography protocol, which is not based on the Abelian group but instead on braid groups. This further proves that braid groups can provide an ideal platform for an alternative cryptography protocols.

References

[1] I. Anshel, M. Anshel, B. Fisher, D. Goldfeld, New key agreement schemes in braid group cryptography, *RSA* (2001).

[2] I. Anshel, M. Anshel, D. Goldfeld, A linear time matirx key agreement protocol, *Presentation Slides*, CRY2002.

[3] E. Artin, Theory of braid, *Ann. Math.*, **48** (1947), 101-26.

[4] J.S. Birman, K.H. Ko, S.J. Lee, A new approach to the word and conjugacy problems in braid groups, *Adv. Math.* **139** (1998), 322-53.

[5] J. Cha, K. Koh, S. Lee, J. Han, J. Cheon, An efficient implementation of braid groups, *AsiaCrypt 2001*, Spring Verlag Lect. Notes in Computer Sci., **2048** (2001), 145-156.

[6] P. Dehornoy, Braids and self-distributivity, *Progress in Math.*, **192** Birkhäuser (2000).

[7] W. Diffie, M. Hellman, New directions in cryptography, *IEEE Transactions on Information Theory* (November 1976).

[8] J. Hughes, The left SSS attack on Ko-Lee-Cheon-Han-Kang-Park key agreement scheme in B_{45}, *Rump session Crypto* (2000).

[9] P. Hughes, Braid-based cryptography, url: http://www.tcs.hut.fi/helger/crypto/link/public/braid/, *Preprint* (2003).

[10] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, S.J. Kang, C.S. Park, New public-key cryptosystem using braid groups, *CRYPTO 2000, LNCS 1880* (2000), 166-183.

[11] N. Koblitz, Elliptic curve cryptosystems, *Math. Comp.*, **48**, No. 5 (1987), 203-209.

[12] V. Miller, Use of elliptic curves in cryptography, In: *Adcances in Cryptography - Proceedings of CRYPTO'85, Lecture Notes in Computer Science* (Ed. H.C. Williams), **218**, Springer-Verlag (1986), 417-426.

[13] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public key cryptosystems, *Communications of the ACM* (Feb. 1978).

[14] H. Sibert, P. Dehornoy, M. Girault, Entity authentication schemes using braid word reduction, *WCC 2003*, url:http://eprint.iacr.org/2002/187.

[15] V.M. Sidelnikov, M.A. Cherepnev, V.Y. Yashcenko, System of open distribution of keys on the basis of noncommunitative semigroups, *Ross. Acad. Nauk Dokl.* (1993), 332-335.