A New Key Exchange Primitive Based on the Triple Decomposition Problem

Yeşem Kurt [*]

October 31, 2006

Abstract

We present a new key exchange primitive based on the decomposition problem over non-commutative groups. Different from the key establishment schemes that rely on the decomposition problem where the problem is decomposing an element into three parts where the middle piece is known, our scheme relies on decomposing an element into three parts, all unknown. We call this problem "Triple Decomposition Problem". This seems to be a harder problem because it requires quadratic systems to be solved instead of linear systems. We discuss the new primitive over two different protocols. The underlying problems in the two protocols differ slightly. We discuss the system and the underlying problems in one of the protocols in detail over braid groups. We manage to provide a setting which resists against linear algebra attacks and length based attacks.

Keywords: Key Exchange, Cryptographic Protocol, Public Key Cryptography, Non-commutative Cryptography, Braid Group Cryptography, Decomposition Problem

1 Introduction

A key exchange is a protocol by which two parties, commonly named Alice and Bob, agree on a secret key to use in their subsequent private communication. Key exchange is an essential part of a public key system. The first key exchange scheme was introduced by Diffie and Hellman in 1976[7], and independently by Merkle in 1978[17]. The security of the Diffie-Hellman key exchange system relies on the difficulty of the decisional Diffie-Hellman problem over finite fields. The emergence of index calculus attacks against the discrete logarithm problem [1], later the developments in quantum computing, and also the curiosity of mathematicians have led to search for new cryptosystems; cryptosystems relying on hard problems of different natures.

Other than adaptations of the Diffie-Hellman key exchange protocol to different groups, three new schemes have been proposed for key exchange [3, 5, 15]. These schemes work over non-commutative groups, in particular they were proposed to be used over braid groups. The security of two of these systems, namely the security of Commutator Key Exchange [3] and Diffie-Hellman-like key exchange (DH-like KE) [15] depends on variants of the conjugacy search problem over braid groups. The third system is a revision of DH-like KE system, which we call "revised DH-like KE" [5], and its security relies on the difficulty of arranging a braid word in a certain order, or

[*] *Department of Mathematics and Computer Science, Pomona College, Claremont, CA 91711*

the decomposition problem in braid groups. Even though these problems are hard in general, the special choices that had to be made for the systems to be practical allowed some attacks to yield feasible solutions.

In this paper our main concern is the decomposition problem so we will focus on the methods to solve the decomposition problem. One line of attacks against this problem over braid groups used linear representations of braid groups and exploited the linear nature of the equations relating the public key and the private key in the systems [6, 14, 16]. Recently, a length-based probabilistic method to solve equations in certain cases over braid groups has been developed [10]. The primitive we are about to describe is designed to overcome the vulnerabilities of braid-based cryptosystems arising from the linear nature of the relations between the public and private keys. The system parameters need to be chosen carefully to have an immune system against length-based attacks.

In the new primitive the private key has three components. The idea is to hide each of these components by multiplying them by random elements in some subgroups. The crucial part is that one of the components is multiplied by random elements both on the right and on the left. The main difference of this scheme from DH-like KE scheme or its revision is that in latter systems a **known** element is multiplied by elements on both sides whereas in our system an **unknown** element is multiplied by elements on both sides. This destroys the linear relation between the public key and the private key. To find the private key or more accurately a key that works as a private key, an adversary has to decompose an element into three elements satisfying certain conditions. We call this the triple decomposition problem.

In sections 2 and 3 we describe the system over a general platform. The platform needs to be a monoid with subsets satisfying certain commutativity and invertibility conditions. For security analysis, besides vulnerabilities that are specific to the scheme, we consider the attacks against other braid-based cryptosystems and make sure that we avoid them.

In section 2 we describe the new primitive in a classical setting where the commutativity conditions are satisfied by setting commuting subsets in advance and letting users choose from these subsets as was done in DH-like KE scheme and its revision [5, 15]. In this scheme an adversary has to decompose an element into three elements from these subsets in order to get a private key. We also give conditions on the subsets to prevent immediate computation of the shared key from the public keys and to make the system truly rely on the triple decomposition problem. A possible platform and subsets satisfying the necessary conditions are discussed in section 5.

In section 3 we employ the protocol proposed by Shpilrain and Ushakov in [19]. In this protocol the subsets are not set in advance. Centralizers are used to achieve commutativity. The underlying problem in this protocol is different from the one in the previous protocol. Before solving the triple decomposition problem, one has to find the subgroups that the unknowns belong to.

In section 4 we list some necessary properties that a platform should carry for a reliable system. In section 5 we discuss the system over braid groups which have the desirable practical properties. We also give a possible choice of subsets to be used in the protocol that we will discuss in section 2 . It turns out that the choice of these subsets is crucial to the reliability of the system. The final setting we suggest in section 5 offers a system in which the relations between the private and the public keys are quadratic. This renders the linear representation attacks ineffective. Also, the subgroups in the setting are generated by short generators and this causes the length-based attacks fail. So when known attacks against similar systems are considered the system stands strong.

Even though more research is required in order the system to be promising in practice, we think it is a worthwhile direction to pursue as it brings a new perspective in non-commutative cryptography. Even if braid groups turn out to be not suitable as a platform, there may be other groups or monoids on which the new primitive can be explored and hopefully work.

2 The Primitive

In order to describe the system in a more general setting we assume that the underlying structure is a monoid.

Definition. A **monoid** is a set with an associative binary operation and an identity element. It is almost a group except that the elements may not be invertible.

Notation: Let G be a monoid. We write G multiplicatively and use the notation $[A, B] = 1$ for two subsets A, B of G when $ab = ba$ for all $a \in A$ and $b \in B$.

2.1 Protocol I

The system requires a non-commutative monoid G with two sets of subsets of G containing 5 subsets of G each, say $\mathcal{A} = \{A_1, A_2, A_3, X_1, X_2\}$ and $\mathcal{B} = \{B_1, B_2, B_3, Y_1, Y_2\}$, satisfying the following invertibility and commutativity conditions:

i. *(Invertibility conditions)* The elements of X_1, X_2, Y_1, Y_2 are invertible.
ii. *(Commutativity conditions)* $[A_2, Y_1] = 1$, $[A_3, Y_2] = 1$, $[B_1, X_1] = 1$, and $[B_2, X_2] = 1$.

2.1.1 Setting the private and the public keys

Suppose a monoid G and the subsets $\mathcal{A} = \{A_1, A_2, A_3, X_1, X_2\}$ and $\mathcal{B} = \{B_1, B_2, B_3, Y_1, Y_2\}$ satisfying i and ii above are fixed. Alice and Bob carry out the following steps:

1. Alice and Bob agree on who will use which set of subsets; say Alice uses $\mathcal{A}$ and Bob uses $\mathcal{B}$.
2. Alice randomly chooses $a_1 \in A_1$, $a_2 \in A_2$, $a_3 \in A_3$, $x_1 \in X_1$, $x_2 \in X_2$, and computes:

$$u = a_1 x_1, \quad v = {x_1}^{-1} a_2 x_2, \quad \text{and} \quad w = {x_2}^{-1} a_3.$$

Her private key is (a_1, a_2, a_3) and her public key is (u, v, w).

3. Bob randomly chooses $b_1 \in B_1$, $b_2 \in B_2$, $b_3 \in B_3$, $y_1 \in Y_1$, $y_2 \in Y_2$, and computes:

$$p = b_1 y_1, \quad q = {y_1}^{-1} b_2 y_2, \quad \text{and} \quad r = {y_2}^{-1} b_3.$$

His private key is (b_1, b_2, b_3) and his public key is (p, q, r).

2.1.2 Key exchange

To agree on a key Alice and Bob do the following:

1. Alice sends Bob her public key (u, v, w).

2. Bob sends Alice his public key (p, q, r).

3. Alice computes $a_1 p a_2 q a_3 r$.

4. Bob computes $u b_1 v b_2 w b_3$.

Note that Alice computes

$$a_1pa_2qa_3r = a_1(b_1y_1)a_2(y_1^{-1}b_2y_2)a_3(y_2^{-1}b_3) = a_1b_1a_2b_2a_3b_3,$$

and Bob computes

$$ub_1vb_2wb_3 = (a_1x_1)b_1(x_1^{-1}a_2x_2)b_2(x_2^{-1})a_3b_3 = a_1b_1a_2b_2a_3b_3.$$

Hence they agree on

$$\text{shared key} = \quad a_1b_1a_2b_2a_3b_3.$$

The idea is to hide the private key by multiplying each component by some elements from the monoid. These elements are chosen from subsets satisfying the invertibility and commutativity conditions so that both parties compute the same key. Note that there are no conditions on the subsets A_1 and B_3. If security is not sacrificed, they may be chosen in a special way to make the system more practical; for instance to have smaller key sizes.

2.2 Some cases to be avoided

In this section we give some cases in which the shared key can be computed without requiring to solve a quadratic system hence should be avoided. These cases are in a way obvious cases that should be avoided over any platform. There may be other cases that needs to be avoided depending on the platform chosen . One should pay attention to such platform-specific cases.

The cases listed in the remarks 1 and 2 below allow immediate computation of the shared key from the public keys hence should be avoided.

Remark 1. If $[X_1, Y_1] = 1$, $[X_2, Y_1] = 1$, and $[X_2, Y_2] = 1$ then the shared key can be computed from the public keys by

$$\text{shared key} = (a_1x_1)(b_1y_1)(x_1^{-1}a_2x_2)(y_1^{-1}b_2y_2)(x_2a_3)(y_2b_3) = upvqwr.$$

Remark 2. If $[A_2, B_1] = 1$, $[A_3, B_2] = 1$, and $[A_3, B_1] = 1$, then the shared key can be computed from the public keys by

$$\text{shared key} = (a_1a_2a_3)(b_1b_2b_3) = uvwpqr.$$

The following discussion directs us to some necessary conditions for the system to truly rely on the triple decomposition problem.

One way to attack the system is to find a pseudo-key, a key that is not necessarily the private key (of one of the users) but works like one, which amounts to solving

$$a_1x_1 = u \tag{2.1}$$
$$x_1^{-1}a_2x_2 = v \tag{2.2}$$
$$x_2^{-1}a_3 = w \tag{2.3}$$

for a_1, x_1, a_2, x_2, a_3 satisfying the invertibility and commutativity conditions. These conditions are automatically met if it is made sure that x_1, a_2, x_2, a_3 come from X_1, A_2, X_2, A_3 respectively. The problem is then to decompose v and w into elements from the respective subsets in such a way that the inverse of the last component of v is the first component of w. Note that because there are no restrictions on a_1, once x_1 is found, u can be decomposed into a_1x_1 by taking $a_1 = ux_1^{-1}$.

Solving equation 2.2 requires v to be decomposed into three elements, and solving 2.3 requires w to be decomposed into two elements. The former is generally a harder task than the latter. One can think of equation 2.2 as a quadratic equation in terms of the unknowns simply by rewriting it in the form $a_2x_2 = x_1v$ whereas equation 2.3 is considered linear. When there is a unique solution to equation 2.3, the security of the system mainly relies on decomposing an element into two elements: First decompose w into $x_2^{-1}a_3$, then substitute x_2^{-1} in equation 2.2 and decompose vx_2^{-1} into $x_1^{-1}a_2$. So we get

Remark 3. For the system to truly rely on solving equation 2.2 (i.e. a quadratic equation) we need to make sure that there are many solutions to equation 2.3.

Another case that needs to be avoided in order to have a quadratic equation is given in the following remark:

Remark 4. If $[A_2, B_1] = 1$ and $[X_2, B_1] = 1$, or $[A_3, B_2] = 1$ and $[A_3, Y_1] = 1$, then the security of the system relies on the difficulty of decomposing an element into two elements.

When $[A_2, B_1] = 1$, the shared key is $a_1a_2b_1b_2a_3b_3$. Multiplying the first two components of Alice's public key, we get the equation $a_1a_2x_2 = uv$. We need to check if having a_1a_2 together in the shared key and in the equation above conduces any vulnerabilities. We need to investigate if/when decomposing uv into ax_2 with $a \in G$, $x_2 \in X_2$ suffices to compute the shared key. We need to check if

$$apqa_3r = \text{shared key}.$$

Substituting $pq = b_1b_2y_2$ on the left hand side above and then using $[A_3, Y_2] = 1$ and $a = uvx_2{}^{-1}$ for the second equality, and $a_3 = x_2w$ for the third equality we get

$$L.H.S. = a(b_1b_2y_2)a_3y_2^{-1}b_3 = (uvx_2{}^{-1})(b_1b_2)a_3b_3 = (uvx_2{}^{-1})(b_1b_2)x_2wb_3.$$

Note that the shared key is $ub_1vb_2wb_3$. The rightmost expression in the above equation is equal to the shared key if $[X_2, B_1] = 1$, because then we have $vb_1 = b_1v$. This explains the first case in Remark 4. Similar arguments apply for having b_1b_2 together and the case $[A_3, B_2] = 1$ and $[A_3, Y_1] = 1$.

3 Protocol II

In this section we suggest an enhancement to the protocol using a technique suggested by Shpilrain and Ushakov in [19]. In the protocol in the previous section the commutativity conditions are achieved by setting subsets that commute in advance. In this protocol the users pick random elements and publish subsets of centralizers of these elements. When a party needs an element that commutes with one of these random elements, he/she picks an element in the subset corresponding to that random element. We explain the system in detail here.

Let G be a non-commutative monoid with a large number of invertible elements.

Definition. For an element $g \in G$ let $C(g)$ be the set of elements in G that commute with g. i.e. $C(g) = \{h \in G | gh = hg\}$. $C(g)$ is called the **centralizer of** g **in** G.

For a subset $H = \{g_1, \dots, g_k\}$ of G, define $\mathbf{C(H)} = \mathbf{C(g_1, \dots, g_k\}}$ to be the set of elements in G that commute with all g_i for $i = 1, \dots, k$ (hence $C(H) = C(g_1) \cap \dots \cap C(g_k)$).

The protocol goes as follows:

1. Alice picks two invertible elements $x_1, x_2 \in G$, chooses subsets S_{x_1} and S_{x_2} of $C(x_1)$ and $C(x_2)$ respectively, and publishes S_{x_1} and S_{x_2}.

2. Bob picks two invertible elements $y_1, y_2 \in G$, chooses subsets S_{y_1} and S_{y_2} of $C(y_1)$ and $C(y_2)$ respectively, and publishes S_{y_1} and S_{y_2}.

3. Alice chooses random elements $a_1 \in G$, $a_2 \in S_{y_1}$, and $a_3 \in S_{y_2}$. (a_1, a_2, a_3) is her private key. She sends Bob her public key (u, v, w) where $u = a_1x_1$, $v = x_1^{-1}a_2x_2$, $w = x_2^{-1}a_3$.

4. Bob chooses random elements $b_1 \in S_{x_1}$, $b_2 \in S_{x_2}$, and $b_3 \in G$. (b_1, b_2, b_3) is his private key. He sends Alice his public key (p, q, r) where $p = b_1y_1$, $q = y_1^{-1}b_2y_2$, and $r = y_2^{-1}b_3$.

5. Alice computes $a_1pa_2qa_3r = a_1b_1a_2b_2a_3b_3$

6. Bob computes $ub_1vb_2wb_3 = a_1b_1a_2b_2a_3b_3$

3.1 Security

To find the private key of one of the users, say Alice's, or more accurately a key that works as Alice's private key (a pseudo-key), one has to solve the equations (2.1), (2.2), (2.3) given in section 2. Assuming that the cases in remarks in section 2.2 are taken into consideration, in order to find a pseudo-key an adversary needs to solve equation

$$x_1^{-1}a_2x_2 = v \tag{3.1}$$

for x_1, a_2, x_2 satisfying the commutativity conditions. This amounts to decomposing v into three elements x_1^{-1}, a_2, x_2 where x_1 and x_2 commute with elements in S_{x_1} and S_{x_2} respectively, and a_2 is in S_{y_1}. Recall that $S_{x_1}, S_{x_2}, S_{y_1}$ are public.

For the rest of the paper we assume that G is a group. All arguments in the previous sections apply except that we do not need to state the invertibility conditions anymore as all elements in a group are invertible. We also assume that the subsets that are published by the two parties in steps 1 and 2 of the protocol are subgroups that are finitely generated and that users publish the generators of the subgroups.

Solving the following two problems would allow an adversary to find a pseudo-key, so for a reliable system at least these problems should be hard:

Problem 1. Given $g_1, \dots, g_k \in G$ compute $C(g_1, \dots, g_k)$.

Suppose $S_{x_1} =< g_1, \dots, g_k >$ (i.e. S_{x_1} is generated by $g_1, \dots, g_{k_1}$). An adversary trying to find x_1 does not know where to choose x_1 from in the beginning. He knows that it commutes with all elements in S_{x_1}. This implies that $x_1 \in C(g_1) \cap \dots \cap C(g_k) = C(g_1, \dots, g_k)$. Similarly he knows that $x_2 \in C(g'_1, \ldots, g'_{k_2})$ where $S_{x_2} =< g'_1 \ldots g'_{k_2} >$. So if the problem stated above is solved then the subgroups that x_1 and x_2 belong to can be computed. Now the adversary knows where to take x_1 and x_2 from. This leaves him with the following problem.

Problem 2.(Triple Decomposition Problem) Given $v = x_1^{-1}a_2x_2$ find $x_1 \in H$, $a_2 \in A$, and $x_2 \in H'$ where $H = C(g_1, \ldots, g_k)$, $H' = C(g'_1, \ldots, g'_{k_2})$, and A is a subgroup of G given by its generators.

4 The Platform Group G

In this section we state some necessary properties a group G should have in order to have an efficient and secure key establishment protocol. We concentrate on the second protocol. The requirements on the group for Protocol I are the same except for a few of them. We will point them out afterwards. The properties we list here are the same as the ones listed in [19] except for the last one. We give the list here with necessary modifications.

P1 (from [19]) G should be a non-commutative and it should be of exponential growth which means that the number of elements of length k in G is exponential in k. This is needed to prevent attacks by complete exhaustion of the key space.

P2 The word problem in G, namely determining if an element is identity or not should be efficiently solvable. This is needed to make sure that two parties compute the same key.

P3 Elements in G should be efficiently representable on a computer.

P4 Multiplication and inversion of elements should be computationally easy with the representation.

P5 (from [19])It should be computationally easy to generate pairs $(a, \{a_1, \ldots, a_k\})$ such that $aa_i = a_ia$ for each $i = 1, \ldots, k$.

P6 (from [19]) For a generic set $\{g_1, \ldots, g_k\}$ of elements of G it should be difficult to compute

$$C(g_1, \ldots, g_n) = C(g_1) \cap \cdots \cap C(g_k).$$

P7 Even if $H_1 = C(g_1, \ldots, g_{k_1})$, $H_2 = C(g'_1, \ldots, g'_{k_2})$ are computed , it should be hard to find $x_1 \in H_1$, $x_2 \in H_2$, and $a \in H$ such that $x_1ax_2 = v$ where H is some fixed subgroup given by its generating set.

Note for Protocol I For the protocol in section 2.1 we require G to have subsets satisfying the commutativity conditions and avoiding the cases in remarks in section 2.2. This is not sufficient to ensure reliability though; the triple decomposition problem (property 7 above) should be hard with the choice of subsets. In section 5.2 we give a possible choice of subsets which avoids the cases in remarks but does not provide a secure system. We discuss weaknesses of this choice in order to motivate a choice that works. The new setting is given in section 5.2.2.

Properties **P5** and **P6** are not relevant in protocol I in general.

5 Braid Groups

Braid groups have been used in cryptosystems such as Arithmetica [3] and Diffie-Hellman like braid-based systems [5, 15]. Their practicality is well known [5, 8, 9]. Here we give a brief explanation of braid groups that helps to state these practical properties.

Definition. The n-braid group B_n, is an infinite non-commutative group defined by the following group presentation

$$B_n \;=\; < \quad \sigma_1, \sigma_2, \ldots \sigma_{n-1} \quad \Big| \quad \begin{array}{rl} \sigma_i\sigma_j = \sigma_j\sigma_i, & |i-j| \geq 2 \\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j, & |i-j| = 1 \end{array} \quad >$$

The integer n is called the *braid index* and the elements in B_n are called n-braids or simply (braid) words. The generators σ_i are called *Artin generators*.

5.1 Computations with Braids

The following are the properties that make braid groups suitable for applications.

B1. (from [19]) Braid groups B_n are non-commutative groups of exponential growth for $n \geq 3$.

B2. (from [19]) There are several normal forms for elements of B_n including Garside normal form [4]. This normal form is efficiently computable (in quadratic time with respect to the length of a given element) and is unique so it gives a fast solution to the word problem.

B3. The Garside normal form is efficiently representable on a computer ($pn \log n$ where p is the so called canonical length of the word in B_n [15]).

B4. (from [19]) There are quadratic time algorithms (in canonical length) to multiply or invert normal forms of elements of B_n [9].

B5. (from [19]) It is not easy to compute the whole centralizer of an element g of G [13]. The number of steps required to compute $C(g)$ is proportional to the size of the "super summit set" of g which is typically huge. Nevertheless there are approaches to finding "large parts" of $C(g)$ [13].

B6. (from [19])For a generic subgroup A it is hard to compute $C(A)$. The complexity of such computation is proportional to the size of the summit set of A, which is typically huge [12].

B7. Linear representations which were used to attack the DH-like KE and its revised version does not work in this case because it made use of the linear equations and the special choices in the system. Triple decomposition for carefully chosen parameters involves quadratic equations.

Recently a new method using a special length function was introduced to solve the decomposition problem over braid groups [10]. The method is probabilistic and fails for some scenarios. When setting the system parameters we have to make sure that the system one needs to solve in order to get a private key falls in one of those scenarios.

5.2 Subgroups for Protocol I

Commutativity is easily achieved in braid groups by taking generators apart from each other. A possible division of generators that satisfies the commutativity conditions and avoids the cases in remarks 1,2,4 in section 2.2 is given below. Let G_n be the braid group of index n. Let $n-1=3d$ for some positive integer $d \geq 2$. Set

$$\begin{aligned} A_1 &= G_n & & \\ X_1 &=<\sigma_1,\ldots,\sigma_{d-1}> & B_1 &=<\sigma_{d+1},\ldots,\sigma_{n-1}> \\ A_2 &=<\sigma_1,\ldots,\sigma_{d-1}> & Y_1 &=<\sigma_{d+1},\ldots,\sigma_{n-1}> \\ X_2 &=<\sigma_1,\ldots,\sigma_{2d-1}> & B_2 &=<\sigma_{2d+1},\ldots,\sigma_{n-1}> \\ A_3 &=<\sigma_1,\ldots,\sigma_{2d-1}> & Y_2 &=<\sigma_{2d+1},\ldots,\sigma_{n-1}> \\ & & B_3 &= G_n \end{aligned} \tag{5.1}$$

so that X_1 is generated by the first $d-1$ generators and so on. The condition that the equation $x_2^{-1}a_3 = w$ has a large solution space (Remark 3 in section 2.2) is satisfied in this setting, because we have $X_2 = A_3$ which gives $x_2 = x$, $a = xw$ is a solution for any $x \in X_2$.

Even though the above choice of subsets takes the remarks in section 2.2 into consideration, the system can be broken using linear algebra attacks, in particular using Burau representation of braid groups.

5.2.1 Burau Representation

In the Burau representation, the braid group G_n is mapped into $GL_{n-1}(Z[t^{\pm 1}])$, $(n-1)\times(n-1)$ matrices of Laurent polynomials over integers. The entries are quite simple: Image of the generator σ_i is the $(n-1)\times(n-1)$ matrix obtained from the identity matrix by replacing the central $(i,i+1)$ square with $\left(\begin{smallmatrix}1-t & t\\ 1 & 0\end{smallmatrix}\right)$, i.e. image of σ_i is

$$\begin{bmatrix} I_{i-1} & 0 & 0 \\ 0 & \begin{smallmatrix}1-t & t\\ 1 & 0\end{smallmatrix} & 0 \\ 0 & 0 & I_{n-i-1} \end{bmatrix}$$

where I_k is the identity matrix of size k.

The images of the braid words corresponding to the subgroups chosen in setting 5.1 are matrices in certain block forms. Because we assume $n-1=3d$, the matrices are composed of 9 submatrices of size $d\times d$ each. We identify these submatrices by a row and a column number each running from 1 to 3. Now the matrices corresponding to braid words in A_1 and in X_1 are respectively of the form

$$a_1 = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} \qquad x_1 = \begin{pmatrix} X_{11} & 0 & 0 \\ 0 & I_d & 0 \\ 0 & 0 & I_d \end{pmatrix}.$$

Having matrices in block forms conduces a vulnerability: some of the non-trivial entries in the images of private keys get revealed in the public keys when matrices are multiplied. (The entries of the submatrices that are 0 or I_d are called trivial) For example the first component of Alice's public key looks like

$$u = a_1x_1 = \begin{pmatrix} A_{11}X_{11} & A_{12} & A_{13} \\ A_{21}X_{11} & A_{22} & A_{23} \\ A_{31}X_{11} & A_{32} & A_{33} \end{pmatrix}.$$

Notice that the second and third columns of u and a_1 are the same which means those entries of a_1 can be read off from the public key u. Another weakness in this setting is that the cases in remarks in section 2.2 are barely avoided. The shared key is $a_1b_1a_2b_2a_3b_3$ which is equal to $a_1a_2b_1a_3b_2b_3$ in this setting because a_2 and b_1, and a_3 and b_2 commute. b_1 and a_3 do not commute. The common part of subgroups B_1 and A_3 which is generated by σ_{d+1} through σ_{2d-1} is what keeps them from commuting. However some of the entries in the images of b_1 and a_3 corresponding to this common part are revealed in the images of the public keys. These revealed entries and commuting parts in the shared key allow computation of the shared key. In the next section we give a modification on these subsets to counter these weaknesses.

5.2.2 A countermeasure against revealed entries

In this section we modify the subsets in setting 5.1 to fully conceal the private keys and the secret choices (i.e. x_1, x_2, y_1, y_2). The modification is based on the simple observation that if $ab = ba$, then $(sas^{-1})(sbs^{-1}) = (sbs^{-1})(sas^{-1})$ for any $s \in G_n$.

Let $s_1, s_2, s_3, s_4 \in G_n$ be fixed, system wide parameters and let X_1, X_2, A_1, A_2, A_3, Y_1, Y_2, B_1, B_2, B_3 be as in 5.1. Set

$$\begin{aligned} &A_1 = G_n && \\ &X'_1 = \{s_1xs_1^{-1} \mid x \in X_1\} && B'_1 = \{s_1bs_1^{-1} \mid b \in B_1\} \\ &A'_2 = \{s_2as_2^{-1} \mid a \in A_2\} && Y'_1 = \{s_2ys_2^{-1} \mid y \in Y_1\} \\ &X'_2 = \{s_3xs_3^{-1} \mid x \in X_2\} && B'_2 = \{s_3bs_3^{-1} \mid b \in B_2\} \\ &A'_3 = \{s_4as_4^{-1} \mid a \in A_3\} && Y'_2 = \{s_4ys_4^{-1} \mid y \in Y_2\} \\ & && B_3 = G_n. \end{aligned} \tag{5.2}$$

The commutativity conditions are still satisfied with these subsets. With careful choice of s_i's (by careful we mean s_i should include a large number of generators) the block forms are destroyed so no entry gets revealed in the public keys. The other weakness in the previous setting, namely the commutativity of certain parts of the shared key is also avoided. The only case in the remarks in section 2.2 that is not immediately taken care of is remark 3, namely the condition that equation $x_2'^{-1}a'_3 = w'$ has a large solution space for $x'_2 \in X'_2$ and $a'_3 \in A'_3$. Written more explicitly the equation

$$s_3x_2^{-1}s_3^{-1}s_4a_3s_4^{-1} = w'$$

or equivalently

$$x_2^{-1}s_3^{-1}s_4a_3 = s_3^{-1}w's_4 \tag{5.3}$$

should have a large solution space for $x_2 \in X_2$ and $a_3 \in A_3$. We will discuss this issue below. Before, we would like to draw attention to the fact that this equation is similar to the equation that needs to be solved in the revised DH-like KE system over braid groups [5]. Namely find

$x, y \in H \subset G$ given $u = xay$ and $a \in G$. Over braid groups H is generated by either the left or the right half of the generators. Once this problem is solved one has a key that works like a private key. The case in the new system is different in two ways:

1. H corresponds to X_2 in our case and it consists of a larger potion of the generators(two thirds instead of a half).

2. A solution to equation 5.3 does not necessarily lead to a private key. A solution for x_2 has to also satisfy

$$s_1 x_1^{-1} s_1^{-1} s_2 a_2 s_2^{-1} s_3 x_2 s_3^{-1} = v'$$

or equivalently

$$x_1^{-1} s_1^{-1} s_2 a_2 s_2^{-1} s_3 x_2 = s_1^{-1} v' s_3 = v. \tag{5.4}$$

The first point is worth mentioning because the techniques that were able to get solutions for parameters suggested in [15] made use of the special structure of the subgroup H or more accurately the image of H under linear representations of G_n(see [6, 16]). When we have a larger H this structure changes and the same techniques may not work.

More important is the the second point. One way to proceed is to substitute a solution of equation 5.3 into equation 5.4. This gives another equation of similar type. If this new equation has a solution that we can compute then we have a key, if not we try another solution of equation 5.3. This works if the solution space for 5.3 is small. Otherwise one has to deal with equation 5.4 as a whole. The linear algebra methods will not work in this case because the equations that are obtained when the system is mapped into matrices are not linear anymore.

Now the issue is how to make sure 5.3 has a large solution space. Note that if $x_2^{-1} = x$, $a_3 = a$ is a solution to the system, then so is $x_2^{-1} = xz$, $a_3 = z^{-1}a$ for any $z \in C(s_3^{-1}s_4) \cap X_2$. (Recall that in setting 5.1 the subgroups X_2 and A_3 are equal so we have $xz \in X_2$ and $z^{-1}a \in A_3$ as required.) Therefore in order to guarantee a large solution space for 5.3 we can choose s_3 and s_4 so that $C(s_3^{-1}s_4) \cap X_2$ is large. Similarly, for the corresponding party we require $C(s_1^{-1}s_2) \cap Y_1$ to be large. We will discuss the case for equations on Alice's side. The conclusions for Bob's side can be derived similarly. One way of having many elements in $C(s_3^{-1}s_4) \cap X_2$ is for $C(s_3^{-1}s_4)$ and X_2 to have several common generators. Recall that X_2 is generated by $\sigma_1, \ldots, \sigma_{2d-1}$. For example, if s_3 and s_4 are chosen so that $s_3^{-1}s_4$ consists of generators from $\sigma_{d+1}, \ldots, \sigma_{n-1}$, then X_2 and $C(s_3^{-1}s_4)$ will have $\sigma_1, \ldots, \sigma_{d-1}$in common so equation 5.3 will have many solutions. (On Bob's side s_1 and s_2 would be chosen so that $s_1^{-1}s_2$ consist of generators from $\sigma_1, \ldots, \sigma_{2d-1}$).

Now the equation to solve is equation 5.4 which can be restated as: Given $v, s, s' \in G_n$ decompose v into $x_1^{-1} s a_2 s' x_2$ where $x_1 \in X_1$, $a_2 \in A_2$, and $x_2 \in X_2$. This is a difficult problem in general because it involves quadratic relations. The linear algebra attacks that were mainly used to attack the decomposition problem in the previous systems where the relations were linear (see [6, 16]) do not work in this case.

5.2.3 Length-based attacks

In [10] Garber et al give a probabilistic method to solve a system of equations in a random finitely generated subgroup of the braid group. They make use of a "monotonic" length function - a length function which satisfies that the expected length tends to increase with the number of generators (of the subgroup) multiplied. In the analysis they provide, the subgroup is generated by elements that are composed of 10 Artin generators each. They state that if the generators can be written as a product of very few Artin generators then the required monotonicity of the length function gets violated and the algorithm fails. According to setting 5.2 in the previous section and the discussion that follows it the generators of the subgroups that the unknowns belong to are single Artin generators. This protects the system from length-based attacks.

5.2.4 Suggested Parameters

The analysis of length-based attacks given in [10] is for subgroups generated by elements that are products of 10 randomly chosen Artin generators. Most of the analysis were over B_8, braid group of index 8. For larger index, it was concluded that the probability of success decreases but not significantly. The discussion in this paper seemed insufficient to make conclusions for our system so we referred to another paper by the same authors [11] where the conjugacy problem instead of the general decomposition problem is considered. The method applies more effectively to the conjugacy problem because of the special nature of the equations involved. (we have $x^{-1}ax = u$ where x is the unknown.) Some of the conclusions they get apply to any decomposition problem. Two of these conclusions are:

1. The smaller is the size of the generators of the subgroups, the smaller is the probability of success. (Here size of an element is the number of Artin generators in it)

2. The longer is the length of x in terms of the number of generators, the smaller is the probability of success.

Taking these into account we recommend to use the braid group B_{100}. This gives 33 generators in each of the subgroups and each of these generators is a single Artin generator. Choose the secret values to consist of a couple of hundred elements say 300 elements each. We hope to do more research on what would be an optimal choice considering the key lengths and the time complexities for computing and exchanging keys.

6 Conclusion

We have proposed a new way to achieve key exchange in a public key system. The security of the new scheme relies on what we called the triple decomposition problem in a non-commutative group, namely decomposing an element into three pieces satisfying certain properties. We have focused on braid groups as they have the desirable practical properties required by the system. We analyzed the system over a classical protocol in detail and were able to choose a setting so that the system is immune to linear algebra and length-based attacks. The second protocol seems more advantageous as it allows more randomness. There are two problems an adversary has to deal with. The first one is to find the common centralizer of a finite number of elements and the second is the triple decomposition problem. The main security shield is the first problem for now because we have not yet found a way to make sure that the underlying triple decomposition problem is hard.

Further research is required to establish a stronger confidence in the system and to determine concrete parameters for practical purposes. We hope to give more results on practical values in an extended paper.

7 Acknowledgements

We would like to thank Prof. M. Anshel and Prof. V. Shpilrain for their kindness in replying our e-mails and references they have provided.

References

[1] L. Adleman, A subexponential algorithm for the discrete logarithm problem with application to cryptography, *Proceedings IEEE 20th annual symposium on foundations of computer science*, 1979, pp. 55–60.

[2] I. Anshel, M. Anshel, and D. Goldfeld, An algebraic method for public-key Cryptography, *Mathematical Research Letters*, **6** (1999)287-291.

[3] I. Anshel, M. Anshel, B. Fisher, and D. Goldfeld, New Key Agreement Protocols in Braid Group Cryptography, *Topics in Cryptology - CT-RSA 2001*, Lecture Notes in Computer Science **2020**, Springer-Verlag,(2001) 13-27.

[4] J.S. Birman, Braids, Links, and Mapping Class Groups, *Annals of Math. Study* , 82, Princeton University Press (1974).

[5] J. Cha, K. Koh, S. Lee, J.Han, and J. Cheon, An Efficient Implementation of Braid Groups, *Proc. of Asiacrypt 2001*, Lecture Notes in Computer Science, **2248**, Springer-Verlag, (2002), 1-13.

[6] J. Cheon and B. Jun, A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem, *Proc. of Crypto 2003*, Lecture Notes in Computer Science, **2729**, Springer-Verlag, (2003), 212-225.

[7] W. Diffie and M. E. Hellman, New Directions in Cryptography, *IEEE Transactions on Information Theory, IT-22*, **6**, (1976), 664-654.

[8] E.A. Elrifai and H.R. Morton, Algorithms for positive braids, *Quart. J. Math.* , Oxford 45 (1994), 479-497.

[9] D. Epstein, J, Cannon, D. Holt, S. Levy, M. Paterson and W. Thurston, *Word Processing in Groups, Jones & Barlett*, 1992.

[10] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, Probabilistic Solutions of Equations in the Braid Group , *arXiv:math.GR/0404076 v3*, (2005).

[11] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, Length-Based Conjugacy Search in the Braid Group , *arXiv:math.GR/0209267 v1*, (2002).

[12] N. Franco, J. González-Meneses, Computation of Centralizers in Braid Groups and Garside Groups, *Rev. Mat. Iberoamericana*, 19 (2003), 367-384.

[13] J. González-Meneses and B. Wiest, Onthe Centraliser of a braid, *Ann. Sci. École Norm. Sup.*, 37 (5) (2004), 729-757.

[14] J. Hughes, A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem *7th Australasian Conference on Information Security and Privacy -ACISP 2002*, Lecture Notes in Computer Science **2384**, Springer-Verlag, (2002), 176-189.

[15] K. Ko, S. Lee, J. Cheon, J. Han, J. Kang, and C. Park, New Public Key Cryptosystem using Braid Groups, *Proc. of Crypto 2000*, Lecture Notes in Computer Science,**1880**, Springer-Verlag, (2000), 166-183.

[16] E. Lee and J. Park, Cryptanalysis of the Public Key Encryption based on Braid Groups, *Proc. of Eurocrypt 2003*, Lecture Notes in Computer Science **2656**, Springer-Verlag, (2003), 477-490.

[17] R. Merkle, Secure communications over insecure channels, *Communications of the ACM* vol. 21, No. 4 (1978), 294-499.

[18] J. Patarin and L. Goubin, Trapdoor One-Way Permutations and Multivariate Polynomials, *LNCS 1334, Proceedings of ICICS'97*, Springer-Verlag, (1987), 356-368.

[19] V.Shilparin, A. Ushakov , A new key exchange protocl based on the decomposition problem, *http://eprint.iacr.org/2005/447.pdf*