Potential Weaknesses of the Commutator Key Agreement Protocol based on Braid Groups

Sang Jin Lee[1] and Eonkyung Lee[2]

[1]CMI, Université de Provence, Marseille, France
sjlee@knot.kaist.ac.kr
[2]Korea Information Security Agency, Seoul, Republic of Korea
eonkyung@kisa.or.kr

Abstract. The braid group with its conjugacy problem is one of the recent hot issues in cryptography. At CT-RSA 2001, Anshel, Anshel, Fisher, and Goldfeld proposed a commutator key agreement protocol (KAP) based on the braid groups and their colored Burau representation. Its security is based on the multiple simultaneous conjugacy problem (MSCP) plus a newly adopted key extractor. This article shows how to reduce finding the shared key of this KAP to the list-MSCPs in a permutation group and in a matrix group over a finite field. We also develop a mathematical algorithm for the MSCP in braid groups. The former implies that the usage of colored Burau representation in the key extractor causes a new weakness, and the latter can be used as a tool to investigate the security level of their KAP.

Key words: Key agreement protocol, Braid group, Multiple simultaneous conjugacy problem, Colored Burau matrix.

1 Introduction

Current braid cryptographic protocols are based on the intractability of the conjugacy problem: given two conjugate braids a and b, find a conjugator (i.e., find x such that $b = x^{-1}ax$). Because it is hard to find a trapdoor in this problem, some variants have been proposed for the key exchange purpose [2, 15].

Anshel *et al.* [2] proposed a key agreement protocol (KAP) assuming the intractability of the following problem: given $a_1, \ldots, a_r, x^{-1}a_1x, \ldots, x^{-1}a_rx \in B_n$, find the conjugator x. We call this problem the *multiple simultaneous conjugacy problem* (MSCP). Loosely speaking, their KAP is as follows: given pairs of n braids $(a_1, x^{-1}a_1x), \ldots, (a_r, x^{-1}a_rx)$, $(b_1, y^{-1}b_1y), \ldots, (b_s, y^{-1}b_sy)$, find the commutator $x^{-1}y^{-1}xy$, where x and y are in the subgroup generated by $\{b_1, \ldots, b_s\}$ and $\{a_1, \ldots, a_r\}$, respectively. The first attack on this KAP is the Length Attack by Hughes and Tannenbaum [14]. They showed that this KAP leaks some information about its private keys x and y for some particular choices of parameters.

At CT-RSA 2001, Anshel *et al.* [1] proposed a new version of their KAP. They adopted a new *key extractor* which transforms a braid into a pair of a permutation and a matrix over a finite field. Here the matrix is obtained from a multi-variable matrix, called the *colored Burau matrix*, by evaluating the variables at numbers in a finite field. They recommended parameters so as to defeat the Length Attack, the mathematical algorithm for the conjugacy problem, and a potential linear algebraic attack on the key extractor.

Our Results. This article attacks the KAPs in [1, 2] from two different angles. Our attacks are partially related to the potential ones already mentioned in their paper.

First, we attack the shared key in [1]. The motivation for this attack is that despite the change of variables in the colored Burau matrix by permutations, the matrix in the final output(i.e., the shared key) is more manageable than braids. We show that the security of the key extractor is based on the problems of listing all solutions to some MSCPs in a permutation group and in a matrix group over a finite field. So if both of the two listing problems are feasible, then we can guess correctly the shared key without solving the MSCP in braid groups.

Second, we attack the private keys in [1, 2]. The base problem of these KAPs is different from the standard conjugacy problem in the following two aspects: (i) The conjugation is multiple and simultaneous. That is to say, we have a set of equations $x^{-1}a_ix = c_i$, $i = 1, \ldots, r$, with a single unknown x. On the one hand, the problem is more difficult than the conjugacy problem because we must find a solution which satisfies all the equations simultaneously. On the other hand, the problem is easier because we have multiple equations. (ii) The conjugator x is contained in the subgroup generated by some specific braids $b_1, \ldots, b_s$, which makes the problem easier. We propose a mathematical algorithm for the MSCP in braid groups.

Outline. We review in §2 the braid groups, the canonical form of braid, the colored Burau representation, and the commutator KAP proposed in [1]. We attack the key extractor in §3 and the private key in §4. We close this article with conclusions in §5.

Conventions.

- S_n denotes the n-permutation group. S_n acts on the set $\{1, 2, \ldots, n\}$ from the left so that for $\alpha, \beta \in S_n$ and $1 \le i \le n$, $(\alpha\beta)(i) = \alpha(\beta(i))$. We express a permutation as a product of cycles. A cycle $\alpha = (k_1, k_2, \ldots, k_r)$ means that $\alpha(k_i) = k_{i+1}$ for $i = 1, \ldots, r-1$ and $\alpha(k_r) = k_1$.
- For a prime p, $\mathbf{F}_p$ denotes the field composed of p elements, $\{0, \ldots, p-1\}$.
- $GL_n(R)$, R a ring, denotes the set of all invertible $(n \times n)$-matrices over R.

2 Preliminaries

2.1 Braid Group

Definition 1. *The n-braid group B_n is an infinite non-commutative group defined by the following group presentation*

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{l} \sigma_i\sigma_j = \sigma_j\sigma_i, \quad |i-j| \geq 2 \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}, \quad i = 1, \ldots, n-2 \end{array} \right\rangle.$$

The integer n is called the braid index *and the elements in B_n are called* n-braids*. The generators σ_i's are called the* Artin generators.

We can give braids the following geometric interpretation. An n-braid can be thought as a collection of n horizontal strands intertwining themselves. See Figure 1. Each generator σ_i represents the process of swapping the i^{th} strand with the $(i+1)^{\text{st}}$ one, where the strand from upper-left to lower-right is over the other.

We can cut a long geometric braid into simple pieces so that each piece contains only one crossing. This decomposition gives a word $W = \sigma_{i_1}^{\epsilon_1}\sigma_{i_2}^{\epsilon_2}\cdots\sigma_{i_k}^{\epsilon_k}$, $\epsilon = \pm 1$. k is called the *word length* of W.

There is a natural projection $\pi : B_n \to S_n$, sending σ_i to the transposition $(i, i+1)$. Let's denote $\pi(a)$ by π_a, and call it the *induced permutation* of a. The braids whose induced permutation is the identity are called *pure braids*. Conversely, for a permutation $\alpha \in S_n$, we can make a simple braid A_α, called a *permutation braid*, by connecting the i^{th} point on the right to the $\alpha(i)^{\text{th}}$ point on the left by straight lines, where the strand from upper-left to lower-right is over the other at each crossing.

2.2 Canonical Form

We review the canonical form of braids and the related invariants, inf, len, and sup, which can be used in measuring how complicated the given braid is. This section is needed only for §4.

1. A word $\sigma_{i_1}^{\epsilon_1}\cdots\sigma_{i_s}^{\epsilon_s}$ is called a *positive word* if all the exponents ϵ_i's are positive. Because the relations in the group presentation of B_n are equivalences between positive words with the same word-length, the *exponent sum* $\mathrm{e}(\sigma_{i_1}^{\epsilon_1}\cdots\sigma_{i_s}^{\epsilon_s}) = \epsilon_1 + \cdots + \epsilon_s$ is well-defined and invariant under conjugation. If P is a positive word, then $\mathrm{e}(P)$ is equal to the word-length of P. The *positive braid monoid*, denoted by B_n^+, is the set of all braids which can be represented by positive words.
2. The permutation braid corresponding to the permutation $\alpha(i) = (n - i)$ is called the *fundamental braid* and denoted by Δ. It can be written as $\Delta = (\sigma_1)(\sigma_2\sigma_1)\cdots(\sigma_{n-1}\sigma_{n-2}\cdots\sigma_1)$.

3. **Theorem**(See [9, 10, 4]). Every n-braid a can be decomposed uniquely into
$$a = \Delta^u A_1 A_2 \dots A_k,$$
where each A_i is a permutation braid and for each $1 \le i < k$, $A_i A_{i+1}$ is *left-weighted*. For the definition of 'left-weighted', see Appendix A.
4. The expression above is called the *left-canonical form*. The invariants the *infimum*, the *canonical length*, and the *supremum* are defined by $\inf(a) = u$, $\mathrm{len}(a) = k$, and $\sup(a) = u + k$, respectively. Similarly, we can also define the *right-canonical form*. The two canonical forms give the same inf, len, and sup.
5. Let $a = \Delta^u P$ and $b = \Delta^v Q$ be conjugate, where $P, Q \in B_n^+$. Then $\mathrm{e}(a) = u\mathrm{e}(\Delta) + \mathrm{e}(P) = v\mathrm{e}(\Delta) + \mathrm{e}(Q)$. So in a conjugacy class, a braid with greater inf is simpler than the one with smaller inf. Similarly, we can also say that a braid with smaller sup is simpler than the one with greater sup.

2.3 Colored Burau Representation

Morton [20] introduced the colored Burau matrix which is a generalization of the Burau matrix. It would be helpful to see the Burau matrix first. Let $\mathbf{Z}[t^{\pm 1}]$ be the ring of Laurent polynomials $f(t) = a_k t^k + a_{k+1} t^{k+1} + \cdots + a_m t^m$ with integer coefficients (and possibly with negative degree terms). Let $GL_{n-1}(\mathbf{Z}[t^{\pm 1}])$ be the group of $(n-1) \times (n-1)$ invertible matrices over $\mathbf{Z}[t^{\pm 1}]$. Note that a matrix over $\mathbf{Z}[t^{\pm 1}]$ is invertible if and only if its determinant is $\pm t^m$ for some integer m.

For $1 \le i \le n-1$, let $C_i(t)$ be the matrix which differs from the identity matrix only at the i^{th} row as shown below. When $i = 1$ or $i = n-1$, the i^{th} row vector is truncated to $(-t, 1, 0, \dots, 0)$ or $(0, \dots, 0, t, -t)$.

$$C_i(t) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & t & -t & 1 & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \qquad C_i^{-1}(t) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & 1 & -\frac{1}{t} & \frac{1}{t} & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}.$$

The *Burau representation* $\rho\colon B_n \to GL_{n-1}(\mathbf{Z}[t^{\pm 1}])$ is defined by $\rho(\sigma_i) = C_i(t)$ [3]. The matrices $C_i(t)$'s satisfy the braid relations, i.e., $C_i(t)C_j(t) = C_j(t)C_i(t)$ for $|i-j| \ge 2$ and $C_i(t)C_{i+1}(t)C_i(t) = C_{i+1}(t)C_i(t)C_{i+1}(t)$ for $i = 1, \dots, n-2$, and so ρ is a group homomorphism. The elements in $\rho(B_n)$ are called *Burau matrices*. Here is an example of a Burau matrix.

$$\rho(\sigma_1^{-1}\sigma_2\sigma_1^{-1}) = \begin{pmatrix} -\frac{1}{t} & \frac{1}{t} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t & -t \end{pmatrix} \begin{pmatrix} -\frac{1}{t} & \frac{1}{t} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{t^2} - \frac{1}{t} & -\frac{1}{t^2} + \frac{1}{t} - 1 \\ -1 & 1 - t \end{pmatrix}.$$

Roughly speaking, the *colored Burau matrix* is a refinement of the Burau matrix by assigning σ_i to $C_i(t_{i+1})$ so that the entries of the resulting matrix have several variables. But such a naive construction does not give a group homomorphism. Thus the induced permutations are considered simultaneously. Let's label the strands of an n-braid by $t_1, \ldots, t_n$, putting the label t_j on the strand which starts from the j^{th} point on the right. Figure 1 shows this labelling for the 3-braid $\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2$.

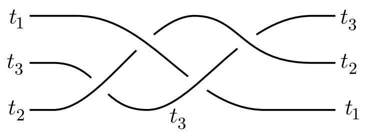

Fig. 1. The labelled braid $\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2$

Definition 2. *Let $a \in B_n$ be given by a word $\sigma_{i_1}^{\epsilon_1}\sigma_{i_2}^{\epsilon_2}\cdots\sigma_{i_k}^{\epsilon_k}$, $\epsilon_j = \pm 1$. Let t_{j_r} be the label of the under-crossing strand at the r^{th} crossing. Then the* colored Burau matrix *$M_a(t_1, \ldots, t_n)$ of a is defined by*

$$M_a(t_1, \ldots, t_n) = \prod_{r=1}^{k} (C_{i_r}(t_{j_r}))^{\epsilon_r}.$$

We can compute the colored Burau matrix by substituting σ_i^{ϵ} by $C_i(\cdot)^{\epsilon}$ and then filling $(\cdot)$ by the variable t_j according to the label of the under-crossing strand. Note that the colored Burau matrix is an invertible matrix over $\mathbf{Z}[t_1^{\pm 1}, \ldots, t_n^{\pm 1}]$, the ring of Laurent polynomials with n variables. In the example $\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2$ shown in Figure 1, the colored Burau matrix is

$$\begin{aligned}
M_{\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2}(t_1, t_2, t_2) &= C_1(t_3)^{-1}C_2(t_2)C_1(t_1)^{-1}C_2(t_3) \\
&= \begin{pmatrix} -t_3^{-1} & t_3^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_2 & -t_2 \end{pmatrix} \begin{pmatrix} -t_1^{-1} & t_1^{-1} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_3 & -t_3 \end{pmatrix} \\
&= \begin{pmatrix} -\frac{1}{t_1} - t_2 + \frac{t_2}{t_1} + \frac{1}{t_1 t_3} - \frac{t_2}{t_1 t_3} & \frac{1}{t_1} + t_2 - \frac{t_2}{t_1} \\ -\frac{t_2}{t_1} - t_2 t_3 + \frac{t_2 t_3}{t_1} & t_2 t_3 - \frac{t_2 t_3}{t_1} \end{pmatrix}.
\end{aligned}$$

Now we describe the colored Burau group as in [1]. We follow the convention of Morton [20], which is a little different from that in [1]. The permutation group S_n acts on $\mathbf{Z}[t_1^{\pm 1}, \ldots, t_n^{\pm 1}]$ from left by changing variables: for $\alpha \in S_n$, $\alpha(f(t_1, \ldots, t_n)) = f(t_{\alpha(1)}, \ldots, t_{\alpha(n)})$. Then S_n also acts on the matrix group $GL_{n-1}(\mathbf{Z}[t_1^{\pm 1}, \ldots, t_n^{\pm 1}])$ entry-wise: for $\alpha \in S_n$ and $M = (f_{ij})$, $\alpha(M) = (\alpha(f_{ij}))$.

Definition 3. *The* colored Burau group *CB_n is $S_n \times GL_{n-1}(\mathbf{Z}[t_1^{\pm 1}, \ldots, t_n^{\pm 1}])$ with multiplication $(\alpha_1, M_1)\cdot(\alpha_2, M_2) = (\alpha_1\alpha_2, (\alpha_2^{-1}M_1)M_2)$. The* colored Burau representation *$C : B_n \to CB_n$ is defined by $C(\sigma_i) = ((i, i+1), C_i(t_{i+1}))$.*

Then it is easy to see the following: (i) CB_n is a group, where the identity is (e, I_{n-1}) and $(\alpha, M)^{-1} = (\alpha^{-1}, \alpha M^{-1})$, (ii) $C(\sigma_i)$'s satisfy the braid relations and so $C : B_n \to CB_n$ is a group homomorphism, and (iii) for $a \in B_n$, $C(a) = (\pi_a, M_a)$, where π_a is the induced permutation and M_a is the colored Burau matrix in Definition 2.

2.4 Commutator Key Agreement Protocol

Recall the commutator KAP of Anshel *et al.* [1]. Fix a (small) prime number p. Let $K_{n,p}$ be the set of pairs $(\alpha, M) \in S_n \times GL_{n-1}(\mathbf{F}_p)$.

Definition 4. *Let $\tau_1, \ldots, \tau_n$ be distinct invertible integers in $\mathbf{F}_p$. The* key extractor $E = E_{p,\tau_1,\ldots,\tau_n} : B_n \to K_{n,p}$ *is defined by*

$$E(a) = (\pi_a, M_a(\tau_1, \ldots, \tau_n) \bmod p),$$

where reduction 'mod p' *means reduction of every entry in the matrix.*

Anshel *et al.* gave a very fast algorithm for computing the key extractor in [1]. The running time is $\mathcal{O}(n\ell(\log p)^2)$, where ℓ is the word-length. The idea is that we can compute $E(\sigma_{i_1}^{\epsilon_1} \cdots \sigma_{i_\ell}^{\epsilon_\ell})$ from $\sigma_{i_1}^{\epsilon_1}$ and $E(\sigma_{i_2}^{\epsilon_2} \cdots \sigma_{i_\ell}^{\epsilon_\ell})$. The commutator KAP using the key extractor is constructed as follows.

Public Information

1. An integer $n > 6$. A prime $p > n$.
2. Distinct and invertible integers $\tau_1, \ldots, \tau_n \in \mathbf{F}_p^*$.
3. $(a_1, \ldots, a_r) \in (B_n)^r$ and $(b_1, \ldots, b_s) \in (B_n)^s$.

Private Key

1. Alice's private key is a word $W(b_1, \ldots, b_s)$.
2. Bob's private key is a word $V(a_1, \ldots, a_r)$.

Public Key

1. Alice's public key is $(x^{-1}a_1x, \ldots, x^{-1}a_rx)$, where $x = W(b_1, \ldots, b_s)$.
2. Bob's public key is $(y^{-1}b_1y, \ldots, y^{-1}b_sy)$, where $y = V(a_1, \ldots, a_r)$.

Shared key

$E(x^{-1}y^{-1}xy) = (\pi_{x^{-1}y^{-1}xy},\ M_{x^{-1}y^{-1}xy}(\tau_1, \ldots, \tau_n) \bmod p)$.

Parameter Recommendation in [1].

– The only restriction on p used in the key extractor is that $p > n$ so that one can choose distinct and invertible elements $\tau_1, \ldots, \tau_n$. One can choose $p < 1000$.
– Take the braid index $n = 80$ or larger and $r = s = 20$. Let each of a_i and b_j be the product of 5 to 10 Artin generators and let each set of public generators involve all the Artin generators of B_n.
– Private keys, x and y, are products of 100 public generators.

3 Linear Algebraic Attack on the Key Extractor

If the KAP [1] is restricted to pure braids, then its key extractor E becomes a group homomorphism. In this case, one can attack the key extractor by linear algebraic methods. To defeat this attack, [1] recommended to choose the private keys x and y such that their induced permutations are sufficiently complex. This section shows that a linear algebraic attack can also be mounted on the KAP even for such parameters. The (list-)MSCP is the following variant of the conjugacy problem.

Definition 5. *Let G be a group. The* MSCP *in G is: given a pair of r-tuples ($r \geq 2$) of elements in G, $(a_1, \dots, a_r)$ and $(x^{-1}a_1x, \dots, x^{-1}a_rx)$, find x in polynomial time (in the input length). In this case, the* list-MSCP *in G is to find the list of all such $x \in G$ in polynomial time.*

Their hardness will be discussed later. Henceforth, we will use the notations in §2.4 if there is no confusion from the context.

Theorem 1. *If the induced permutations of the private keys are known, then we can construct four list-MSCPs in $GL_{n-1}(\mathbf{F}_p)$ such that computing the matrix part of the shared key is reduced to solving all these list-MSCPs.*

Proof. By the definition of the colored Burau representation C, we get the following equation

$$\begin{aligned} C(x^{-1}y^{-1}xy) &= (\pi_x, M_x)^{-1}(\pi_y, M_y)^{-1}(\pi_x, M_x)(\pi_y, M_y) \\ &= (\pi_x^{-1}\pi_y^{-1}\pi_x\pi_y, (\pi_y^{-1}\pi_x^{-1}\pi_y\pi_x M_x^{-1})(\pi_y^{-1}\pi_x^{-1}\pi_y M_y^{-1})(\pi_y^{-1}M_x)M_y). \end{aligned}$$

So the matrix part of the shared key is the product of the following four matrices evaluated at $(t_1, \dots, t_n) = (\tau_1, \dots, \tau_n)$:

$$(\pi_y^{-1}\pi_x^{-1}\pi_y\pi_x M_x^{-1}), \quad (\pi_y^{-1}\pi_x^{-1}\pi_y M_y^{-1}), \quad (\pi_y^{-1}M_x), \quad \text{and} \quad M_y.$$

Now we propose a method of composing MSCPs in $GL_{n-1}(\mathbf{F}_p)$ for these matrices, assuming that we already know the permutations π_x and π_y. Here we consider only $(\pi_y^{-1}M_x)(\tau_1, \dots, \tau_n)$. Similar constructions work for the other three matrices. The following technique makes $(\pi_y^{-1}M_x)(\tau_1, \dots, \tau_n)$ to be a solution to an MSCP with N equations in $GL_{n-1}(\mathbf{F}_p)$ for given N. The basic idea is: if a is a pure braid and $c = x^{-1}ax$, then $(\alpha M_x)(\tau_1, \dots, \tau_n)$ is a solution to $AX = XC$, where α is an arbitrary permutation, $A = (\alpha\pi_x^{-1}M_a)(\tau_1, \dots, \tau_n)$, and $C = (\alpha M_c)(\tau_1, \dots, \tau_n)$.

1. Choose a word $a = U(a_1, \dots, a_r)$ such that a is a pure braid.
2. Compute $c = U(c_1, \dots, c_r)$. Then, $c = x^{-1}ax$ and c is also a pure braid.
3. Compute M_a and M_c. Since $ax = xc$ and

$$C(ax) = (\pi_x, (\pi_x^{-1}M_a)M_x), \quad C(xc) = (\pi_x, M_xM_c),$$

we have $(\pi_y^{-1}\pi_x^{-1}M_a)(\pi_y^{-1}M_x) = (\pi_y^{-1}M_x)(\pi_y^{-1}M_c)$.

4. Compute $(\pi_y^{-1}\pi_x^{-1}M_a)(\tau_1,\ldots,\tau_n)$ and $(\pi_y^{-1}M_c)(\tau_1,\ldots,\tau_n)$. Refer to the resulting matrices as A and C, respectively.
5. Repeat the above steps to get a system of equations $A_jX = XC_j$ for $j = 1,\ldots,N$.

It is easy to see that $(\pi_y^{-1}M_x)(\tau_1,\ldots,\tau_n)$ is a solution to the MSCP in $GL_{n-1}(\mathbf{F}_p)$, $\{A_jX = XC_j\}_{1\le j\le N}$. □

Now we discuss how to construct the algorithm in Theorem 1 practically, the hardness of the list-MSCPs in S_n and in $GL_{n-1}(\mathbf{F}_p)$, and possible fixes.

How to generate a pure braid $a = U(a_1,\ldots,a_r)$ in the first step. An easy construction is to choose a word $V(a_1,\ldots,a_r)$ at random and then take a power $U = V^k$, where k is the order of the induced permutation. However, the order of a permutation is the least common multiple of the lengths of cycles, and so it can be too large. For example, for $n = 87$, the maximal order of n-permutations is greater than 10^7. See [19]. One way to avoid this huge order is to choose $V(a_1,\ldots,a_r)$ as a short word. For example, if V is a product of three a_i's, then its induced permutation is a product of 15 to 30 transpositions and so its order is small. (Note that a_i's are products of 5 to 10 Artin generators.) And once we have a pure braid $a = U(a_1,\ldots,a_r)$, then we can also use $W^{-1}aW$ for any word W on a_i's.

Hardness of the list-MSCPs in permutation group and in matrix group. The MSCP in permutation group is easy. Note that two permutations are conjugate if and only if they have the same cycle decomposition. The MSCP in matrix group is also easy because the equation $AX = XC$ can be considered as a system of homogeneous linear equations in the entries of X. One can use the polynomial time deterministic algorithm by Chistov, Ivanyos, and Karpinski [7].

So the difficulty of the list-MSCP lies only in the number of its solutions. Let G be a group and let $(a_1,\ldots,a_r)$ and $(c_1,\ldots,c_r)\in G^r$ be an instance of a list-MSCP in G. If x_1 and x_2 are two solutions, then $(x_2x_1^{-1})a_i = a_i(x_2x_1^{-1})$ for each i. Hence $x_2 = x_1z$ for some z in $\cap_{i=1}^r \mathrm{Cent}(a_i)$, where $\mathrm{Cent}(a_i) = \{g\in G \mid ga_i = a_ig\}$ is the centralizer of a_i. So the number of the solutions is exactly the cardinality of the subgroup $\cap_{i=1}^r \mathrm{Cent}(a_i)$. We don't have the average cardinality of this subgroup when G is either S_n or $GL_{n-1}(\mathbf{F}_p)$. But it does not seem large for generic a_i's in S_n or in $GL_{n-1}(\mathbf{F}_p)$ from the following observation.

For $a_1,\ldots,a_r\in S_n$, let $z\in\cap_{i=1}^r \mathrm{Cent}(a_i)$. Then for each $1\le i\le r$ and for each $1\le k\le n$, if k lies in an m-cycle of a_i, then so does $z(k)$. Moreover, if $(k_1,\ldots,k_m)$ and $(l_1,\ldots,l_m)$ are m-cycles in a_i such that $z(k_1) = l_1$, then $z(k_j) = l_j$ for all $2\le j\le m$. For example, let $a_1 = (1,2)(4,5)(8,9)$, $a_2 = (3,4)(5,7,8)$, and $a_3 = (1,5,6)(9,10)$ be the cycle decompositions of permutations in S_{10}. Let $z\in\cap_{i=1}^3 \mathrm{Cent}(a_i)$. Then $z(1) = 1$ because $z(1)$ must lie in a 2-cycle of a_1, in an 1-cycle of a_2, and in a 3-cycle of a_3 simultaneously. In addition, $z(k) = k$ for $k = 2,5,6$ because z must fix all the numbers in a cycle of a_i containing 1, for any $i = 1,2,3$. By continuing this argument we can see that z is the identity permutation. The list-MSCP in $GL_{n-1}(\mathbf{F}_p)$ can be discussed similarly using the Jordan canonical form of matrices. See §5.6 in [8].

Possible fix. To defeat our linear algebraic attack, at least one of the list-MSCPs in permutation group and in matrix group must be infeasible. Here, we discuss how to make the induced list-MSCP in permutation group infeasible. One way to do so is to use pure braids. But if all the braids $(a_1, \ldots, a_r)$ and $(b_1, \ldots, b_s)$ are pure, then the induced permutation is nothing more than the identity. Hence, we can consider the following simple cases.

1. Choose $(a_1, \ldots, a_r)$ in B_n and $(b_1, \ldots, b_s)$ in the pure braid group. Then the induced permutation of $x = W(b_1, \ldots, b_s)$ is the identity but it is impossible to list all $y = V(a_1, \ldots, a_r)$ because the equation $y^{-1}b_i y = d_i$ gives no information about π_y.
2. Choose $(a_1, \ldots, a_r)$ and $(b_1, \ldots, b_s)$ so that the induced permutations of a_i's fix $\{1, \ldots, \lfloor\frac{n}{2}\rfloor\}$ and those of b_j's fix $\{\lfloor\frac{n}{2}\rfloor + 1, \ldots, n\}$.

In both cases, the list-MSCP in permutation group is infeasible. But these fixes give disadvantage that the braids a_i's or b_j's become complicated, and so the KAP becomes less secure against the Length Attack.

4 Attack on the Private Key

This section proposes an attack on the private keys of the commutator KAPs in [1, 2] by solving the MSCPs in braid groups; given $(a_1, \ldots, a_r)$ and $(c_1, \ldots, c_r)$ in $(B_n)^r$, find $x \in B_n$ such that $c_i = x^{-1}a_i x$ for all i simultaneously. We start with some discussions.

Uniqueness of the solution to the MSCP in braid groups. For generic choice of $a_1, \ldots, a_r$, the solution x is unique up to a power of Δ^2 (See Appendix B). That is, if x' is another solution such that $x'^{-1}a_i x' = c_i$ for all i, then $x' = \Delta^{2k}x$ for some integer k. Note that $x'^{-1}y^{-1}x'y = x^{-1}y^{-1}xy$ for any y, because Δ^2 is a central element. Therefore, it suffices to find $\Delta^{2k}x$ for any k.

Length Attack. The commutator KAPs in [1, 2] have the following condition in addition to the standard MSCP: x is contained in the subgroup generated by some publicly known braids $b_1, \ldots, b_s$. This fact is crucial to the Length Attack of J. Hughes *et al.* [14]. They showed that the KAP is vulnerable to the Length Attack when b_j's are complicated and x is a product of a small number of $b_j^{\pm 1}$'s. And same for a_i's and y. To defeat the Length Attack, Anshel *et al.* [1] recommended using simple a_i's and b_j's and complicated x and y as mentioned in §2.4. Our attack of this section is strong when a_i's and b_j's are simple and it does not depend on how complicated x and y are.

Which braid is simpler in the conjugacy class? Recall the discussion in §2.2. Let $a = \Delta^u A_1 \dots A_k$ and $c = \Delta^v C_1 \dots C_\ell$ be the left canonical forms of conjugate braids. It is natural to say that a is *simpler* than c if (i) the word-length of $A_1 \dots A_k$ is smaller than that of $C_1 \dots C_\ell$, or (ii) they have same word-length but k is smaller than ℓ. The former is equivalent to $u = \inf(a) > v = \inf(c)$ and the latter is equivalent to $\inf(a) = \inf(c)$ and $\sup(a) < \sup(b)$.

Mathematical algorithm for the conjugacy problem. The conjugacy problem in braid groups is: given $(a, c) \in (B_n)^2$, decide whether they are conjugate and if so, find $x \in B_n$ such that $x^{-1}ax = c$. The algorithm for the conjugacy problem, first proposed by Garside [13] and still being improved [10, 9, 4, 5, 11], works as follows:

1. For each element in B_n, the *super summit set* is defined as the set of all conjugates which have the minimal canonical length. Then it is a finite set and two braids are conjugate if and only if the corresponding super summit sets coincide.
2. Given a braid $a \in B_n$, one can compute an element in the super summit set easily.
3. Given two elements u and v in the same super summit set, there is a chain leading from u to v, where successive elements are conjugated by a permutation braid.

How to develop an algorithm for the MSCP in braid groups. There are several directions in designing an algorithm for the MSCP, depending on the characteristics of the instances. This section focuses on the fact that $(a_1, \dots, a_r)$ is simple because a_i's are products of a few Artin generators but $(c_1, \dots, c_r)$ are usually complicated because $c_i = x^{-1}a_i x$ for a complicated braid x. See [1].

Let $\tau : B_n \to B_n$ be the isomorphism defined by $\tau(\sigma_i) = \sigma_{n-i}$. Hence τ^k, k-composition of τ, is the identity for k even, and τ for k odd.

Proposition 1. *Let $a, c, x \in B_n$ and $c = x^{-1}ax$. Let $c = \Delta^w C_1 \dots C_l$ be the left-canonical form. If $\inf(a) > \inf(c)$, then $x = x_0 H$ for some $x_0 \in B_n$ with $\inf(x) = \inf(x_0)$ and for a permutation braid H determined by $H = \Delta\tau^w(C_1^{-1})$.*

Proof. It is a restatement of the Cycling Theorem in Appendix C (Theorem 4.1 of [9] and Theorem 5.1 of [4]). The statements in [4, 9] look different from the above, but the argument of their proofs is exactly Proposition 1. □

We can understand this proposition in the following way.

- $x = x_0 H$ and $\inf(x) = \inf(x_0)$ means that x_0 is simpler than x: if $x_0 = \Delta^u P$ and $x = \Delta^u Q$ are the left canonical forms, then the word-length of P is smaller than that of Q.
- Let $c' = HcH^{-1}$. Then $x^{-1}ax = c$ implies that $x_0^{-1}ax_0 = c'$. Thus we get a conjugacy problem with simpler solution.

– The condition $\inf(a) > \inf(c)$ means that c is more complicated than a. And H is determined not by x but by $c(= x^{-1}ax)$.
– Consequently, we can interpret Proposition 1 as follows: *if c is more complicated than a, then we can find H such that the solution to the conjugacy problem for (a, c') is simpler than that for (a, c), where $c' = HcH^{-1}$.*

Definition 6. *For $a_1, \ldots, a_r \in B_n$, define $C^{\inf}(a_1, \ldots, a_r)$ as the set of all $(u_1, \ldots, u_r)$ such that $\inf(u_i) \geq \inf(a_i)$ for all i and there exists some $w \in B_n$ satisfying $u_i = w^{-1}a_i w$ for all i simultaneously.*

Theorem 2. *Let $(a_1, \ldots, a_r)$ and $(c_1, \ldots, c_r)$ be an instance of an MSCP in B_n and x a positive braid such that $x^{-1}a_i x = c_i$ for all i. Assume that a_i's and c_i's are already in the left canonical form. Then we can compute positive braid x_0 and $(c'_1, \ldots, c'_r)$ such that $(c'_1, \ldots, c'_r) \in C^{\inf}(a_1, \ldots, a_r)$ and $c'_i = x_0 c_i x_0^{-1}$ for all i, in time proportional to*

$$n(\log n)|x| \left(|x| + \sum_{i=1}^{r} (|a_i| + |c_i|) \right), \tag{1}$$

where $|\cdot|$ denotes the word-length in generators. Moreover $x = x_1 x_0$ for some positive braid x_1, in particular the word-length of x_1 is less than that of x.

Proof. We exhibit an algorithm that computes x_0 and hence $(c'_1, \ldots, c'_r)$.

Input: $(a_1, \ldots, a_r), (c_1, \ldots, c_r) \in (B_n)^r$.
Initialization: $x_0 = e$(identity braid), $c'_i = c_i$ for all i.
Loop:
STEP 1: If $\inf(c_i) \geq \inf(a_i)$ for all i, then STOP
STEP 2: Choose k such that $\inf(c_k) < \inf(a_k)$.
Compute the permutation braid H by applying Proposition 1 to (a_k, c_k).
STEP 3: $x_0 \leftarrow Hx_0$, $c'_i \leftarrow Hc'_i H^{-1}$ for all i. GO TO STEP 1
Output: x_0 and $(c'_1, \ldots, c'_r)$.

Because H in Proposition 1 is a suffix of x, so is x_0 at each step in the above algorithm. Whenever Proposition 1 is applied in the loop, the word-length of x_0 strictly increases and its final length is bounded above by $|x|$. So the algorithm stops in at most $|x|$ repetitions of the loop.

All the computations involved is to compute simple conjugations such as HaH^{-1}, $a \in B_n$ and H a permutation braid, which can be done in time $\mathcal{O}(n(\log n)|a|)$ and simple multiplications of the form Hx_0, which can be done in time $\mathcal{O}(n(\log n)|x_0|)$. So the whole complexity is (1). □

Note that the a_i's are much simpler than c_i's [1] and that the newly obtained braids c'_i's are at least as simple as a_i's in terms of 'inf'.

Now we have simple instance $(a_1, \ldots, a_r)$ and $(c'_1, \ldots, c'_r)$. The natural question is how to solve the MSCP for this new instance. It uses a variant of the Convexity Theorem [4, 9]. See Appendix C.

Theorem 3. *Given* $(c'_1, \ldots, c'_r) \in C^{\inf}(a_1, \ldots, a_r)$*, there exists a chain of elements in* $C^{\inf}(a_1, \ldots, a_r)$ *from* $(a_1, \ldots, a_r)$ *to* $(c'_1, \ldots, c'_r)$*, where successive elements are simultaneously conjugated by a permutation braid. In other words, there is a sequence* $(a_1, \ldots, a_r) \to (a'_1, \ldots, a'_r) \to \cdots \to (a_1^{(k)}, \ldots, a_r^{(k)}) = (c'_1, \ldots, c'_r)$ *such that for each* j*, there is a permutation braid* H_j *satisfying* $a_i^{(j+1)} = H_j^{-1} a_i^{(j)} H_j$ *for all* i *simultaneously.*

Proof. This is a restatement of the Convexity Theorem in Appendix C. □

By Theorem 2 and Theorem 3, we can solve any MSCP in finite time. But the computational complexity of a naive implementation is exponential with respect to the braid index n and involves the cardinality of the set $C^{\inf}(a_1, \ldots, a_r)$. There seems to be no previous result concerning the cardinality of $C^{\inf}(a_1, \ldots, a_r)$.

If the instances are extremely simple, for example when all a_i's are positive braids, then the set $C^{\inf}(a_1, \ldots, a_r)$ will be very small, so that the MSCP is feasible. But the MSCP for generic instances needs more work.

Possible improvement of the algorithm for the MSCP in braid groups. Recently N. Franco and J. González-Meneses [11] improved the algorithm for the conjugacy problem in braid groups. The complexity of their algorithm to compute the super summit set is $\mathcal{O}(N\ell^2 n^4 \log n)$, where N is the cardinality of the super summit set, n is the braid index, and ℓ is the word-length of the given braid. The complexity of the old algorithm was $\mathcal{O}(N\ell^2(n!)n \log n)$. With respect to the braid index n, the complexity was reduced from exponential function to polynomial. We expect that their idea can also be applied to the MSCP and our algorithm can be improved so that the computational complexity is a polynomial in (n, r, ℓ, N), where ℓ is the maximal word-length of a_i's and N is the cardinality of the set $C^{\inf}(a_1, \ldots, a_r)$.

5 Concluding Remarks

For the commutator key agreement protocols of Anshel *et al.* [2, 1], we have proposed two kinds of attacks: a linear algebraic attack on the key extractor and a mathematical algorithm solving the MSCP in braid groups.

Our linear algebraic attack has shown that given the induced permutations of the private keys, computing the matrix part of the shared key $E(x^{-1}y^{-1}xy)$ is reduced to some list-MSCPs in $GL_{n-1}(\mathbf{F}_p)$. So one can compute the entire shared key very efficiently if the list-MSCPs in S_n and in $GL_{n-1}(\mathbf{F}_p)$ are feasible.

On the other hand, we have proposed an algorithm for the MSCP in braid groups that is suitable for the instance, $(a_1, \ldots, a_r)$ and $(c_1, \ldots, c_r)$, where a_i's are simple and c_i's are complicated. It consists of two steps. We first transform $(c_1, \ldots, c_r)$ into $(c'_1, \ldots, c'_r)$ where each c'_i is at least as simple as a_i, and then find the conjugator. The first step is really efficient. However, there is no polynomial time algorithm for the second step.

It is interesting to study the (in-)feasibility of the list-MSCPs in permutation groups and in matrix groups, and to improve the mathematical algorithm for the MSCP in braid groups.

Acknowledgement

The authors are grateful to Rosario Genaro, Dorian Goldfeld, Hi-joon Chae, Gabor Ivanyos, the anonymous referees, and Eurocrypt 2002 program committee for their helpful comments or suggestions. The first author was supported by postdoctoral fellowship program of KOSEF. And the second author was supported by 2002 R&D project, Development of models & schemes for the security analysis of cryptographic protocols, of MIC.

References

1. I. Anshel, M. Anshel, B. Fisher, and D. Goldfeld, *New Key Agreement Protocols in Braid Group Cryptography*, Topics in Cryptology—CT-RSA 2001 (San Francisco, CA), 13–27, Lecture Notes in Comput. Sci., 2020, Springer, Berlin, 2001.
2. I. Anshel, M. Anshel, and D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Res. Lett. **6**(1999), no. 3-4, 287–291.
3. J.S. Birman, *Braids, links and the mapping class group*, Ann. Math. Studies 82, Princeton Univ. Press, 1974.
4. J.S. Birman, K.H. Ko, and S.J. Lee, *A new approaches to the world and conjugacy problem in the braid groups*, Adv. Math. **139**(1998), no. 2, 322–353.
5. J.S. Birman, K.H. Ko, and S.J. Lee, *The infimum, supremum and geodesic length of a braid conjugacy class*, Adv. Math. **164**(2001), no. 1, 41–56.
6. J.C. Cha, K.H. Ko, S.J. Lee, J.W. Han, and J.H. Cheon, *An Efficient Implementation of Braid Groups*, Advances in Cryptology—ASIACRYPT 2001, (Gold Coast, Queensland, Australia), 144–156, Lecture Notes in Comput. Sci., 2248, Springer, Berlin, 2001.
7. A. Chistov, G. Ivanyos, and M. Karpinski, *Polynimial time algorithms for modules over finite dimensional algebras*, Proc. Int. Symp. on Symbolic and Algebraic Computation (ISSAC) '97, ACM. 68–74, 1997.
8. C.G. Cullen, *Matrices and Linear Transformations*, Addison-Wesley, 1972.
9. E.A. Elrifai and H.R. Morton, *Algorithms for positive braids*, Quart. J. Math. Oxford Ser. (2) **45**(1994), no. 180, 479–497.
10. D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, and W. Thurston, *Word processing in groups*, Jones and Bartlett Publishers, Boston, MA, 1992.
11. N. Franco and J. González-Meneses, *Conjugacy problem for braid groups and Garside groups*, arXiv:math.GT/0112310, preprint 2001.
12. A. Fathi, F. Laudenbach, and V. Poénaru, *Travaux de Thurston sur les surfaces*, Astérisque, 66–67, 1979.
13. F.A. Garside, *The braid group and other groups*, Quart. J. Math. Oxford Ser. (2) **20**(1969), 235–254.
14. J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Institute for Mathematics and Its Applications, April, 2000, Minneapolis, MN, Preprint number 1696, URL `http://www.ima.umn.edu/preprints/apr2000/1696.pdf`.
15. K.H. Ko, S.J. Lee, J.H. Cheon, J.H. Han, J.S. Kang, and C. Park, *New public key cryptosystem using braid groups*, Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA), 166–183, Lecture Notes in Comput. Sci., 1880, Springer, Berlin, 2000.

16. E. Lee, S.J. Lee, and S.G. Hahn, *Pseudorandomness from Braid Groups*, Advances in cryptology—CRYPTO 2001 (Santa Barbara, CA), 486–502, Lecture Notes in Comput. Sci., 2139, Springer, Berlin, 2001.
17. J. Los, *Pseudo-Anosov maps and invariant train track in the disc: a finite algorithm*, Proc. Lond. Math. Soc. **66**, 400-430, 1993.
18. J. McCarthy, *Normalizers and centralizers of psuedo-Anosov mapping clases*, Available at `http://www.mth.msu.edu/~mccarthy/research/`.
19. W. Miller, *The maximum order of an element of a finite symmetric group*, Amer. Math. Monthly **94**(1987), no. 6, 497–506.
20. H.R. Morton, *The multivariable Alexander polynomial for a closed braid*, Low-dimensional topology (Funchal, 1998), 167–172, Contemp. Math., 233, Amer. Math. Soc., Providence, RI, 1999.

A Left-weighted

For a positive braid P, the *starting set* $S(P)$ and the *finishing set* $F(P)$ is defined as follows.

$$S(P) = \{i \mid P = \sigma_i Q \quad \text{for some } Q \in B_n^+\},$$
$$F(P) = \{i \mid P = Q\sigma_i \quad \text{for some } Q \in B_n^+\}$$

For positive braids P and Q, we say that PQ is *left-weighted* if $F(P) \supset S(Q)$ and *right-weighted* if $F(P) \subset S(Q)$.

B Uniqueness of the Solution to the MSCP in Braid Groups

Proposition 2. *For generic instances, the MSCP $x^{-1}a_i x = c_i$ for $i = 1, \dots, r$, has unique solution up to power of Δ^2.*

Proof. Let x_1 and x_2 be two solutions, i.e., $x_1^{-1}a_i x_1 = x_2^{-1}a_i x_2$ for each i. Therefore $x_1 x_2^{-1}$ commutes with each a_i and so it commutes with any element in the subgroup generated by $a_1, \dots, a_r$.

If we choose an n-braid at random, then it is pseudo-Anosov [12, 16, 17]. So we may assume that we can choose two pseudo-Anosov braids, a and a', from the subgroup generated by $\{a_1, \dots, a_r\}$ such that they have different invariant measured foliations and that there is no symmetry on the foliations. Then a braid commuting with both of a and a' is of the form Δ^{2k} for some integer k [18]. □

C Algorithm for the Conjugacy Problem

1. The *super summit set* of a is defined by the set of all braids a' such that a and a' are conjugate and $\inf(a')$ is maximal and $\sup(a')$ is minimal in the conjugacy class of a. We can compute the super summit set by two theorems, the Cycling Theorem and the Convexity Theorem.

2. Let $a = \Delta^u A_1 \cdots A_k$ be the left-canonical form of a. The *cycling* $\mathbf{c}(a)$ and the *decycling* $\mathbf{d}(a)$ of a is defined by

$$\mathbf{c}(a) = \Delta^u A_2 \cdots A_k \tau^u(A_1),$$
$$\mathbf{d}(a) = \Delta^u \tau^{-u}(A_k) A_1 \cdots A_{k-1}.$$

In general, the braids on the right hand sides are not in canonical forms, and so must be rearranged into canonical forms before the operations are repeated.
3. **Cycling Theorem** (See [4, 9, 10])
   (i) If $\inf(a)$ is not maximal in the conjugacy class, then $\inf \mathbf{c}^l(a) > \inf(a)$ for some l.
   (ii) If $\sup(a)$ is not minimal in the conjugacy class, then $\sup \mathbf{d}^l(a) < \sup(a)$ for some l.
   (iii) So the maximal value of inf and the minimum value of sup can be achieved simultaneously. In particular, the super summit set is not empty for any braid.
4. **Convexity Theorem** (See [4, 9, 10])
Let $c = x^{-1}ax$, $\inf(a) = \inf(c)$, and $\sup(a) = \sup(c)$. Let $H_1 H_2 \cdots H_k$ be the left-canonical form of x. Then

$$\inf(H_1^{-1} a H_1) \geq \inf(a) \quad \text{and} \quad \sup(H_1^{-1} a H_1) \leq \sup(a),$$

that is, $H_1^{-1} a H_1$ is as simple as a with respect to both of inf and sup.
5. It is clear that $\inf(a) = \inf(\tau(a))$ and $\sup(a) = \sup(\tau(a))$. So by the Convexity Theorem, we know that if both a and c are contained in the same super summit set, then there is a finite sequence $a = a_0 \to a_1 \to \cdots \to a_k = c$ such that for each $i = 1, \ldots, k$, a_i is contained in the super summit set and $a_i = H_i^{-1} a_{i-1} H_i$ for some permutation braid H_i.
So the conjugacy problem can be solved by the following steps: (i) given a and c, compute elements a' and c' contained in the super summit set, by using the Cycling Theorem, and (ii) for all permutation braids A_α, $\alpha \in S_n$, compute $A_\alpha^{-1} a' A_\alpha$ and collect the ones with the same inf and sup as a'. Do the same thing for all the other elements in the super summit set until any new element cannot be obtained.
The first step can be done in polynomial time and the second step is done in exponential time because the number of n-permutations is $n!$.