Cryptanalysis of the Public-Key Encryption Based on Braid Groups

Eonkyung Lee[1] and Je Hong Park[2]

[1]Cryptographic Technology Team, KISA, Seoul, 138-803, South Korea
eonkyung@kisa.or.kr
[2]Department of Mathematics, KAIST, Taejon, 305-701, South Korea
arttex@cais.kaist.ac.kr

Abstract. At CRYPTO 2000, a new public-key encryption based on braid groups was introduced. This paper demonstrates how to solve its underlying problem using the Burau representation. By this method, we show that the private-key can be recovered from the public-key for several parameters with significant probability in a reasonable time. Our attack can be mounted directly on the revised scheme mentioned at ASIACRYPT 2001 as well. On the other hand, we give a new requirement for secure parameters against our attack, which more or less conflicts with that against brute force attack.

Key words: Cryptanalysis, Public-key encryption, Braid group, Burau representation.

1 Introduction

Most current public-key cryptosystems (PKCs) are based on number-theoretic problems such as factoring integers or finding discrete logarithms. The first PKC that is not based on these problems is the Merkle-Hellman encryption based on knapsack problem. Despite the NP-completeness of this problem, all knapsack-based encryptions have been broken.

By Shor's algorithms [16], quantum computers can solve integer factorization problems and discrete logarithm problems efficiently. Although practical quantum computers are at least 10 years away, their potential will soon create distrust in current cryptographic methods [17]. In order to enrich cryptography as well as not to put all eggs in one basket, there have been continuous attempts to develop alternative PKCs based on different kinds of problems. In the later 1990s, lattice cryptography received wide attention. Among the proposed schemes, NTRU, introduced by Hoffstein et al. [7], has been improved to be secure by Nguyen and Pointcheval [14].

On the other hand, there have been efforts in the area of noncommutative groups [2, 11, 12, 15]. As a candidate for cryptographically promising noncommutative groups, braid groups have attracted many cryptographers' attention due to efficient implementation and several hard problems. The braid group B_n is a torsion-free noncommutative group naturally arising from geometric braids with

n strands. Anshel et al. [2, 1] proposed key agreement protocols in braid groups, on which a couple of attacks have been mounted by Lee and Lee [13], Hughes and Tannenbaum [9, 10], and Hofheinz and Steinwandt [8].

A practical public-key encryption scheme in braid groups (BPKE) was introduced by Ko et al. at CRYPTO 2000 [11], which appeared to be based on a Diffie-Hellman(DH) like problem: given $(a, x^{-1}ax, y^{-1}ay)$ for $a \in B_n$, $x \in G_1$, $y \in G_2$, find $y^{-1}x^{-1}axy$. Here, G_1 and G_2 are proper subgroups of B_n commuting each other. Hofheinz and Steinwandt [8] proposed an algorithm for this problem. Cha et al. [3] mentioned a revised version, a more generalized encryption scheme, to which Hofheinz and Steinwandt's algorithm cannot be applied. The underlying problem of this generalized version could be roughly stated as: given (a, x_1ax_2), find (z_1, z_2) such that $z_1az_2 = x_1ax_2$, where $a \in B_n$ and $x_1, x_2, z_1, z_2 \in G_1$. There have been attempts in analyzing braid cryptosystem via representation. As a first step, Hahn et al. [6] analyzed the problem for particular case, $x_1 = x_2^{-1}$, not in braid group but in general linear group using the Burau representation. Hughes [9] proposed a heuristic method of recovering a braid from its Burau matrix in order to attack Anshel et al.'s key agreement protocol [1].

Our Results This paper proposes two improvements of Hughes algorithm. Comparing with his algorithm, one is more efficient with the same success rate, and the other has higher success rate with less efficiency. Using these algorithms and Hahn et al.'s idea, we attempt to solve the underlying problem of the revised BPKE as well as the original scheme. Using this method, we recover the private-key from the public-key for seven out of the nine parameters suggested at [11] with significant probability in a reasonable time. On the other hand, we give a new requirement for secure parameters against our attack, which more or less conflicts with that against brute force attack.

Outline Section 2 reviews necessary material for this paper. Section 3 describes the revised BPKE [3], and shows that its security can be reduced to the aforementioned problem for a particular type of instances. Section 4 proposes a new method of solving that problem using the Burau representation, and shows that the private-key can be recovered from the public-key with significant probability in a reasonable time for some parameters proposed at [11].

2 Preliminaries

2.1 Basic Notations

For a positive integer N, let $\mathbb{Z}_N$ denote the integers modulo N. For a ring R, $\mathrm{Mat}_{n,m}(R)$ stands for the set of all $(n \times m)$-matrices over R, and $\mathrm{GL}_n(R)$ the general linear group which is the set of all $(n \times n)$-invertible matrices over R. For simplicity, $\mathrm{Mat}_n(R)$ means $\mathrm{Mat}_{n,n}(R)$. I_n is the $(n \times n)$-identity matrix. $R[t_1, \ldots, t_n]$ is the ring composed of polynomials in $(t_1, \ldots, t_n)$ with coefficients in R.

2.2 Braid Group

Here, we have a quick review of the braid groups. There is a detailed explanation in [11] for beginners.

The n-braid group B_n is presented by the $(n-1)$ Artin generators $\sigma_1, \ldots, \sigma_{n-1}$ and two kinds of relations $\sigma_i\sigma_j = \sigma_j\sigma_i$ for $|i-j| > 1$ and $\sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j$ for $|i-j| = 1$. The monoid[1] B_n^+ can be regarded as the set of all positive words with the identity e_n. Let LB_n and UB_n be the subgroups of B_n generated by $\sigma_1, \ldots, \sigma_{\lfloor n/2 \rfloor -1}$ and $\sigma_{\lfloor n/2 \rfloor +1}, \ldots, \sigma_{n-1}$, respectively. Thus, LB_n could be regarded as $B_{\lfloor n/2 \rfloor}$, and LB_n^+ as $LB_n \cap B_n^+$.

A fundamental braid of B_n means $(\sigma_1 \cdots \sigma_{n-1})(\sigma_1 \cdots \sigma_{n-2}) \cdots (\sigma_1)$ denoted by Δ_n. For $x \in B_n$, the supremum of x, $\sup(x)$, indicates the smallest integer such that there is $P \in B_n^+$ satisfying $\Delta_n^{\sup(x)} = xP$. The infimum of x, $\inf(x)$, indicates the largest integer such that there is $Q \in B_n^+$ satisfying $\Delta_n^{\inf(x)} Q = x$. The canonical-length of x, $\mathrm{len}(x)$, means $\sup(x) - \inf(x)$. Each of them is an invariant of x. In addition, the word-length of x, $\mathrm{wlen}(x)$, indicates the shortest word length of x.

3 Security of the BPKE

The BPKE was proposed in two versions [11, 3]. We consider the revised scheme described in [3] which is of more general form than the original scheme [11]. A private-key is given as a pair (x_1, x_2) where x_1 and x_2 are in LB_n, and the associated public-key is a pair (a, b) where $a \in B_n$ and $b = x_1 a x_2$. The encryption and decryption is as follows.

Encryption Given a message $M \in \{0,1\}^k$,

1. Choose $y_1, y_2 \in UB_n$ at random.
2. Ciphertext is $(y_1 a y_2, h(y_1 b y_2) \oplus M)$.

Decryption Given a ciphertext (c, d) and the private-key (x_1, x_2), compute $M = h(x_1 c x_2) \oplus d$.

Here, $h : B_n \to \{0,1\}^k$ is a collision-free hash function. We remark that the original BPKE in [11] is obtained with the condition $x_1 = x_2^{-1}$ and $y_1 = y_2^{-1}$.

For a given public-key (a, b), there can be multiple private-keys. If we could find any $z_1, z_2 \in LB_n$ such that $z_1 a z_2 = b$, then M is always recovered by computing $h(z_1 c z_2) \oplus d$ regardless of the considered BPKE being the original version (i.e., $x_1 = x_2^{-1}$) or the revised one (i.e., x_1 and x_2 are independent). So, the problem of recovering private-key is reduced to the following problem.

> Given $a, b \in B_n$, find $x_1, x_2 \in LB_n$ satisfying $b = x_1 a x_2$, provided that such braids exist.

[1] A group-like object which fails to be a group because elements need not have an inverse within the object. A monoid S must also be associative and have an identity element $e \in S$ such that for all $x \in S$, $ex = xe = x$.

We note that if such x_1 is found then x_2 is automatically derived, and vice versa.

This paper considers the above problem for particular instances. Hereinafter, we call this problem the *P-BPKE problem* for our convenience.

P-BPKE problem: Given $a, b \in B_n^+$, find $x_1, x_2 \in LB_n^+$ satisfying $b = x_1 a x_2$, provided that such braids exist.

Since the BPKE is described in terms of left-canonical form in [11, 3], we should discuss it also wherein. In implementing the BPKE, the ranges of a, x_1, and x_2 are specified. This means that the bounds of their supremums and infimums are finite and given. Let Δ_L denote the fundamental braid within LB_n, i.e., $\Delta_L = (\sigma_1 \cdots \sigma_{\lfloor n/2 \rfloor -1})(\sigma_1 \cdots \sigma_{\lfloor n/2 \rfloor -2}) \cdots (\sigma_1)$. We note that (i) if $y \in B_n$, then $\Delta_n^k y = y \Delta_n^k$ for any even integer k, and (ii) if $y \in B_n$, $k \le \inf(y)$, and $k < 0$, then both $\Delta_n^{-k} y$ and $y \Delta_n^{-k}$ are positive n-braids. Now we show that the security of the BPKE depends on the intractability of the P-BPKE problem.

Proposition 1. *If the P-BPKE problem is solved, then the private-key of the BPKE can be recovered from its associated public-key.*

Proof. Let (a, b) be a given public-key and (x_1, x_2) be its private-key. Since $\inf(a)$ is known, we can choose an *even* integer u so that $\Delta_n^u a$ is a positive braid. Let $a' = \Delta_n^u a$. The lower bound of $\inf(x_1)$ and $\inf(x_2)$ in LB_n is publicly known, say v. For $v < 0$, let $b' = \Delta_L^{-v} \Delta_n^u b \Delta_L^{-v}$. Since $a' \in B_n^+, \Delta_L^{-v} x_1, x_2 \Delta_L^{-v} \in LB_n^+$, and $b' = \Delta_L^{-v} x_1 a' x_2 \Delta_L^{-v}$, we can obtain $P_1, P_2 \in LB_n^+$ from (a', b') such that $b' = P_1 a' P_2$ by hypothesis. Let $z_1 = \Delta_L^v P_1$ and $z_2 = P_2 \Delta_L^v$. Then, $z_1, z_2 \in LB_n$ and $z_1 a z_2 = \Delta_L^v P_1 a P_2 \Delta_L^v = \Delta_n^{-u} \Delta_L^v P_1 a' P_2 \Delta_L^v = \Delta_n^{-u} \Delta_L^v b' \Delta_L^v = b$. Therefore, (z_1, z_2) is another private-key corresponding to the public-key (a, b).

For $v \ge 0$, (z_1, z_2) is recovered from (a, b) in a similar and simpler way by letting $b' = \Delta_n^u b$. □

4 The P-BPKE Problem

This section shows how to solve the P-BPKE problem. A natural way to solve it in B_n is via another group G as follows: (i) to transform the problem in B_n into the one in G, (ii) to solve the problem in G, and (iii) to lift the solution back into B_n. Representations of braid groups—a representation of a group is a homomorphism from that group to $\mathrm{GL}_k(R)$ for some integer k and domain R of characteristic zero—appear to provide such a group G. In order to perform (iii) above, a faithful representation, Krammer representation, can be considered. However, in this case, an n-braid is transformed into an $\frac{n(n-1)}{2} \times \frac{n(n-1)}{2}$-matrix. Since manipulating these matrices, especially in (ii) above, seems unpractical for large braid index and word length, we consider a more practical one, Burau representation. The Burau representation $\rho_n : B_n \to \mathrm{GL}_n(\mathbb{Z}[t, t^{-1}])$ is defined by

$$\rho_n(\sigma_i) = \left[\begin{array}{c|c|c} I_{i-1} & 0 & 0 \\ \hline 0 & \begin{matrix} 1-t & t \\ 1 & 0 \end{matrix} & 0 \\ \hline 0 & 0 & I_{n-i-1} \end{array}\right].$$

4.1 Computing $\rho_m(x_1)$ or $\rho_m(x_2)$ from (a, x_1ax_2)

From a pair of n-braids (a, b) such that $b = x_1ax_2$ for $a \in B_n^+$ and $x_1, x_2 \in LB_n^+$, $(\rho_n(a), \rho_n(b))$ can be computed easily. Let $m = \lfloor \frac{n}{2} \rfloor$. Then, there exists (X_1', X_2') such that (i) $X_i' = \begin{bmatrix} X_i & 0 \\ 0 & I_{n-m} \end{bmatrix}$ for $i = 1, 2$, where $X_i \in \mathrm{GL}_m(\mathbb{Z}[t])$, and (ii) $\rho_n(b) = X'_1\rho_n(a)X_2'$. This section computes X_1 or X_2.

Finding $X_i|_{t=t_0}$ Let

$$\rho_n(a) = \begin{bmatrix} A_1 \, A_2 \\ A_3 \, A_4 \end{bmatrix} \qquad \text{and} \quad \rho_n(b) = \begin{bmatrix} B_1 \, B_2 \\ B_3 \, B_4 \end{bmatrix},$$

where $A_1, B_1 \in \mathrm{Mat}_m(\mathbb{Z}[t])$, $A_2, B_2 \in \mathrm{Mat}_{m,n-m}(\mathbb{Z}[t])$, $A_3, B_3 \in \mathrm{Mat}_{n-m,m}(\mathbb{Z}[t])$, and $A_4, B_4 \in \mathrm{Mat}_{n-m}(\mathbb{Z}[t])$.

The equation $\rho_n(b) = X_1'\rho_n(a)X_2'$ yields the following four types of equations.

$$\begin{array}{ll} \mathsf{e1} : B_1 = X_1A_1X_2, & \mathsf{e2} : B_2 = X_1A_2, \\ \mathsf{e3} : B_3 = A_3X_2, & \mathsf{e4} : B_4 = A_4. \end{array}$$

Now we consider e2 and e3. If either A_2 or A_3 is of full rank, then X_1 or X_2 is determined uniquely.

We experiment how often a has full rank A_i for $i = 2$ or 3 by Maple. On input (n, l, t_0), our program chooses at random 100 positive n-braid a's with word length l, checks the ranks of $A_i|_{t=t_0}$ for $i = 2, 3$, and then outputs the number of a's for which at least one of $A_i|_{t=t_0}$'s has full rank. Figure 1 shows these numbers for various (n, l)'s at $t_0 = 2$. For several other random integer t_0's, the results were very similar. The reason is that the degree of independency of the column vectors of A_i generally remains the same after substituting randomly chosen integer t_0 for the variable t. For instance, in the case of $n = 2m$, $\mathrm{rank}(A_i) = m$ if and only if $\det(A_i) \neq 0$. Here, $\det(A_i)$ is a polynomial in t, and many of the roots of $\det(A_i) = 0$ lie beyond integers. Note that the existence of t_0 such that $A_2|_{t=t_0}$ (resp. $A_3|_{t=t_0}$) has full rank implies that A_2 (resp. A_3) has also full rank. It is observed that the longer the word length of a is, the higher is the probability that A_2 or A_3 has full rank. One can heighten a little the probability by simultaneously solving three restated linear (in X_1 and X_2^{-1}) equations: $B_1X_2^{-1} = X_1A_1$, $B_2 = X_1A_2$, $B_3X_2^{-1} = A_3$ at $t = t_0$.

Finding X_i For $a \in B_n^+$ with full rank A_2 or A_3, X_1 or X_2 can be computed by solving e2 or e3, respectively. Without loss of generality, we assume that $\mathrm{rank}(A_3) = m$. Every entry of X_2 is expressed as a polynomial in $\mathbb{Z}[t]$. The highest degree of entries in X_2 is at most $\mathrm{wlen}(x_2)$. In fact, it is far smaller than $\mathrm{wlen}(x_2)$ for random x_2. We note that since $x_2 \in LB_n^+$, the upper bound of $\mathrm{wlen}(x_2)$ can be computed because $\mathrm{wlen}(x_2) \leq \sup(x_2)\frac{m(m-1)}{2} \leq \frac{sm(m-1)}{2}$ where $m = \lfloor n/2 \rfloor$ and s is the publicly known upper bound of $\sup(x_2)$ in LB_n. Recalling that $\det(\rho_m(\sigma_j)) = -t$ for every j, $\mathrm{wlen}(x_2)$ can be derived from $X_2|_{t=t_0}$ by $\det(X_2|_{t=t_0}) = (-t_0)^{\mathrm{wlen}(x_2)}$ in at most $\frac{sm^2}{2}$ multiplications.

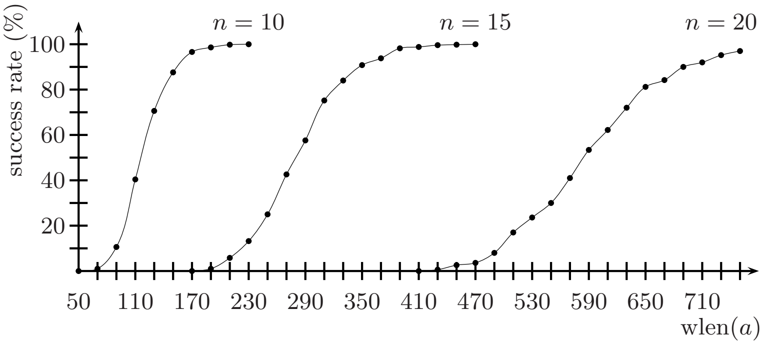

Fig. 1. Probability that A_2 or A_3 is of full rank for a random $a \in B_n^+$

We then evaluate (A_3, B_3) at $\mathrm{wlen}(x_2)+1$ different t_0's such that $\mathrm{rank}(A_3|_{t=t_0}) = m$. From each $(A_3|_{t=t_0}, B_3|_{t=t_0})$, computing $X_2|_{t=t_0}$ takes time $O(m^3)$ in the Gauss-Jordan elimination. From these $X_2|_{t=t_0}$'s, X_2 can be computed by Lagrange interpolating polynomial method with complexity $O(m^2 \mathrm{wlen}(x_2)^2)$ [4].

Hence, the total evaluation of $\rho_m(x_2)$ from those $(A_3|_{t=t_0}, B_3|_{t=t_0})$'s takes time $O(m^2 \cdot \mathrm{wlen}(x_2)^2)$ regarding $\mathrm{wlen}(x_2)$ as greater than m.

4.2 Recovering x from $\rho_m(x)$

Hughes [9] proposed an algorithm recovering a private key of AAFG1 key agreement protocol [1] from its Burau matrix. His main idea is to reconstruct the private key, say x, from $\rho_m(x)$ generator by generator from right to left by assuming that the first column with the highest degree polynomial entry in $\rho_m(x)$ indicates the last generator of x. The Burau representation is not faithful for $n \geq 5$. Namely, no algorithm can recover the designated preimage from the Burau matrix. Hughes' heuristic algorithm behaves much better than expected. It succeeds with 96% for AAFG1 scheme with suggested parameter.

A natural question is what is the fundamental reason for this high probability. For Hughes' experiment, we analyze mathematically how that column is connected to the last generator. Using this result, we improve his algorithm.

For $x \in B_m$, let $c(\rho_m(x))$ indicate the first column containing the highest degree entry in $\rho_m(x)$ among the first $(m-1)$ columns. If there is no confusion from the context, it is interchangeably written as $c(x)$.

If $x \in B_m^+$ has $1 \leq \mathrm{wlen}(x) \leq 2$, then it is clear that $c(x)$ indicates the last generator of x. What happens for longer word lengths? We investigate in which cases $c(x)$ fails to indicate the last generator of x.

Theorem 1. *Suppose that $c(x)$ indicates the last generator of x whenever $x \in B_m^+$ and $\mathrm{wlen}(x) \leq l$ for some positive integer $l \geq 2$. For any $z \in B_m^+$ with $\mathrm{wlen}(z) = l + 1$, $c(z)$ indicates the last generator of z with probability more than $1 - \frac{1}{m-1}$.*

Proof. z can be expressed as $z = x\sigma_i$ for some $x \in B_m^+$ and $1 \le i < m$. Then,

$$\rho_m(z)_k = \begin{cases} (1-t)\rho_m(x)_i + \rho_m(x)_{i+1} & \text{if } k = i, \\ t\rho_m(x)_i & \text{if } k = i+1, \\ \rho_m(x)_k & \text{otherwise.} \end{cases} \tag{1}$$

Hereinafter, for a matrix M, let ${}_iM_j$, ${}_iM$, and M_j denote the (i,j)-th entry, the i-th row vector, and the j-th column vector of M, respectively. And, let $\mathrm{Hdeg}(M)$ denote the highest degree of entries in M. Let c_k and d_k denote $\mathrm{Hdeg}(\rho_m(z)_k)$ and $\mathrm{Hdeg}(\rho_m(x)_k)$, respectively. Then

$$c_i = \mathrm{Hdeg}((1-t)\rho_m(x)_i + \rho_m(x)_{i+1}), \tag{2}$$

$$c_{i+1} = d_i + 1, \tag{3}$$

$$c_k = d_k \qquad \text{for } k \ne i, i+1. \tag{4}$$

Let $c(x) = j$. We remark that since $\mathrm{wlen}(x) = l$ and $l \ge 2$, there exists $w \in B_m^+ \backslash \{e_m\}$ such that $x = w\sigma_j$.

Case 1. $|i - j| > 1$ or $i = j$

In this case, $z = w\sigma_j\sigma_i = w\sigma_i\sigma_j$. So it suffices to show that $c(z) = i$ or j. Here, we consider only the case of $i < j-1$ (For the cases of $i > j+1$ and $i = j$, the same results are obtained in a similar and easier way.).

Since $c(x) = j$ and $i+1 < j$, $d_j > d_{i+1}$. If $d_i \ge d_{i+1}$, then $c_i = d_i + 1$ from Eq.(2). Thus $c_i \le d_j$ because $c(x) = j$ and $i < j$. And $c_{i+1} = c_i$ by Eq.(3). Thus, by Eq.(4), $c(z) = i$ or j according to $c_i = c_j$ or $c_i < c_j$. Otherwise(i.e. if $d_i < d_{i+1}$), $c_i \le d_{i+1}$ from Eq.(2). $c_{i+1} \le d_{i+1}$ by Eq.(3), and hence $c_i, c_{i+1} < c_j$ because $d_{i+1} < d_j$ and $d_j = c_j$. Thus $c(z) = j$ by Eq.(4).

Case 2. $i = j+1$

If $d_i < d_{i+1}$, then $c_i \le d_{i+1}$ from Eq.(2), and $c_{i+1} \le d_{i+1}$ from Eq.(3). Thus, by Eq.(4), $c(z) = j$ because $d_{i+1} \le d_j$. Otherwise(i.e. if $d_i \ge d_{i+1}$), $c_i = d_i + 1$ from Eq.(2), and hence $c_{i+1} = c_i$ from Eq.(3). Thus, $c(z) = j$ or i by Eq.(4). Since σ_i is clearly the last generator of z, it suffices to show that if $c(z) = j$, then σ_j is another last generator of z. We do this in two steps. First, we prove that $c(z) = j$ implies $c(w) = j+1$. Then we prove that if $c(w) = j+1$, σ_j is the last generator of z.

Proof of the first step: Since $z = w\sigma_j\sigma_{j+1}$,

$$\rho_m(z)_k = \begin{cases} (1-t)\rho_m(w)_j + \rho_m(w)_{j+1} & \text{if } k = j, \\ t(1-t)\rho_m(w)_j + \rho_m(w)_{j+2} & \text{if } k = j+1, \\ t^2\rho_m(w)_j & \text{if } k = j+2, \\ \rho_m(w)_k & \text{otherwise.} \end{cases}$$

With this, $c(z) = j$ implies the followings.

- $\mathrm{Hdeg}(\rho_m(w)_{j+1}) \ge \mathrm{Hdeg}(\rho_m(w)_j) + 2$ because $\mathrm{Hdeg}(t^2\rho_m(w)_j) \le \mathrm{Hdeg}((1-t)\rho_m(w)_j + \rho_m(w)_{j+1}))$. So $\mathrm{Hdeg}(\rho_m(z)_j) = \mathrm{Hdeg}(\rho_m(w)_{j+1})$.

· $\mathrm{Hdeg}(\rho_m(w)_{j+1}) \geq \mathrm{Hdeg}(\rho_m(w)_{j+2})$ because $\mathrm{Hdeg}(t(1-t)\rho_m(w)_j + \rho_m(w)_{j+2}) \leq \mathrm{Hdeg}(\rho_m(w)_{j+1})$.
· $\mathrm{Hdeg}(\rho_m(w)_{j+1}) > \mathrm{Hdeg}(\rho_m(w)_k)$ for all $k < j$.
· $\mathrm{Hdeg}(\rho_m(w)_{j+1}) \geq \mathrm{Hdeg}(\rho_m(w)_k)$ for all $k > j+2$.

From these inequalities, we get $c(w) = j+1$.

Proof of the second step: Since $\mathrm{wlen}(w) < l$, there exists $w' \in B_m^+$ such that $w = w'\sigma_{j+1}$. Since $z = w\sigma_j\sigma_i = w'\sigma_{j+1}\sigma_j\sigma_{j+1} = w'\sigma_j\sigma_{j+1}\sigma_j$, $c(z)$ turns out to indicate the last generator of z.

Case 3. $i = j-1$

In this case, $c_q < d_{i+1}$ for all $q < i$, and $c_q \leq d_{i+1}$ for all $q > i+1$. Therefore, we check c_i and c_{i+1} comparing with d_{i+1}.

Since $c(x) = i+1$, we get $d_i + 1 \leq d_{i+1}$. If $d_i + 1 < d_{i+1}$, then $c_i = d_{i+1}$ from Eq.(2), and hence $c_{i+1} < c_i$ from Eq.(3). Thus, we get the desired $c(z) = i$.

Now we consider the other case, $d_i + 1 = d_{i+1}$, by comparing $\rho_m(z)_i$ and $\rho_m(z)_{i+1}$ entry by entry. For $1 \leq p \leq m$, define $S_p = \{k \in \{1, \ldots, m\} \mid \deg({}_k\rho_m(x)_p) = d_p\}$. Clearly, $S_p \neq \emptyset$ for any p.

· If $k \in S_{i+1} - S_i$, then $\deg({}_k\rho_m(z)_{i+1}) < \deg({}_k\rho_m(z)_i) = d_{i+1}$ because $\deg({}_k\rho_m(z)_{i+1}) = \deg({}_k\rho_m(x)_i) + 1 < d_i + 1 = d_{i+1} = \deg({}_k\rho_m(x)_{i+1}) = \deg({}_k\rho_m(z)_i)$.
· If $k \in S_i - S_{i+1}$, then $\deg({}_k\rho_m(z)_{i+1}) = \deg({}_k\rho_m(z)_i) = d_{i+1}$ because $\deg({}_k\rho_m(z)_{i+1}) = \deg({}_k\rho_m(x)_i) + 1 = \deg({}_k\rho_m(z)_i)$.
· If $k \notin S_i \cup S_{i+1}$, then we cannot compare $\deg({}_k\rho_m(z)_i)$ and $\deg({}_k\rho_m(z)_{i+1})$. However, it is easy to check that each of them is less than d_{i+1}.
· If $k \in S_i \cap S_{i+1}$, then $\deg({}_k\rho_m(z)_i) \leq \deg({}_k\rho_m(z)_{i+1}) = d_{i+1}$. More precisely, let α_k and β_k be the coefficients of the highest degree entry in $\deg({}_k\rho_m(x)_i)$ and $\deg({}_k\rho_m(x)_{i+1})$, respectively. In this case, $\alpha_k \neq \beta_k$ implies $\deg({}_k\rho_m(z)_i) = d_{i+1}$, and $\alpha_k = \beta_k$ implies $\deg({}_k\rho_m(z)_i) < d_{i+1}$.

So we can see that if $S_i \neq S_{i+1}$, then $c(z) = i$ because $c_{i+1} \leq c_i = d_{i+1}$. Otherwise, there are two cases: (i) if there is $k \in S_i$ such that $\alpha_k \neq \beta_k$, then $c(z) = i$ because $c_i = c_{i+1} = d_{i+1}$, and (ii) if $\alpha_k = \beta_k$ for all $k \in S_i$, then $c(z) = i+1$ because $c_i < c_{i+1} = d_{i+1}$.

The last case, (ii), points out the unique case when $c(z)$ may fail to indicate the last generator of z. So the probability that this case happens is much lower than $\frac{1}{m-1}$. □

Improved algorithms Theorem 1 shows that $c(x\sigma_i)$ always succeeds in indicating the last generator of $x\sigma_i$ except in the case that $c(x) = i+1$, $d_i + 1 = d_{i+1}$, $S_i = S_{i+1}$, and $\alpha_k = \beta_k$ for all $k \in S_i$. So the failure probability appears very low under the condition stated in Theorem 1. Its loose upper bound is $\frac{1}{m-1}$. Therefore, for a random positive braid with fixed word-length, the success probability appears to become high as the braid index increases. And, for fixed braid index, it appears to become low as the word-length increases.

Combining the results of Theorem 1 with Hughes algorithm, this section proposes two algorithms recovering x from $\rho_m(x)$. The first algorithm arises from the basic property of braids we are dealing with. Being compared to Hughes algorithm, it is faster and its success probability is almost the same. The second algorithm comes from Theorem 1. Its success probability is higher and it is slower than Hughes'. The objective of these algorithms is to seek an unknown sequence $(A[1], \ldots, A[\mathrm{wlen}(x)])$ satisfying $x = \sigma_{A[1]} \cdots \sigma_{A[\mathrm{wlen}(x)]}$, where $\mathrm{wlen}(x)$ was already obtained in the process of computing $\rho_m(x)$ as mentioned in section 4.1. We describe more about these algorithms in Algorithms 1 and 2, and give experimental results in Table 1 and 2. Our experiment was performed on a computer with a Pentium IV 1.7 GHz processor using Maple 7. On input (m, l), our program chooses at random 500 x's from B_m^+ with $\mathrm{wlen}(x) = l$, computes $\rho_m(x)$ from x, computes z from $\rho_m(x)$ by each algorithm, and then determines whether or not z is equal to x by comparing their left-canonical forms.

Algorithm 1. As we see in Eq.(1), if σ_j is the last generator of x, then every entry in $\rho_m(x)_{j+1}$ is always in $t\mathbb{Z}[t]$ because we deal with only positive braids here. Let $c'(\rho_m(x))$ indicate the first column containing the highest degree entry in $\rho_m(x)$ among the columns whose next column's entries are all in $t\mathbb{Z}[t]$. Like $c(\cdot)$, let $c'(\rho_m(x))$ and $c'(x)$ be used interchangeably.

Algorithm 1 Finding x from $\rho_m(x)$ without self-correction

1: **for** i from l to 1 by -1 **do**
2:    Compute $j_c = c'(x)$.
3:    **if** there does not exist such j_c **then**
4:       break
5:    **else**
6:       $A[i] \leftarrow j_c;\ \rho_m(x) \leftarrow \rho_m(x)\rho_m(\sigma_{j_c})^{-1}$
7:    **end if**
8: **end for**
9: $z \leftarrow \sigma_{A[i+1]} \cdots \sigma_{A[l]}$
10: Return(z)

Although Algorithm 1 has an additional task to Hughes algorithm like checking whether or not ${}_k\rho_m(x)_{j+1} \in t\mathbb{Z}[t]$ for all $k = 1, \ldots, m$ at Step 2, it is eventually more efficient than his as in Table 1. In our experiment, due to this task, Algorithm 1 stops on the way to building z usually much before $\mathrm{wlen}(z) = l$ if $z \neq x$. Namely, $i >> 0$ at Step 9. Whereas, Hughes algorithm goes on until $\mathrm{wlen}(z) = l$ even if $z \neq x$. So the time gap between these two algorithms grows as $\mathrm{wlen}(x)$ increases.

Algorithm 2. Algorithm 2 is gained from comparing the experiments of Algorithm 1 and Hughes algorithm, and from Theorem 1. The motivation is as follows. Given a positive braid x, let $x_1 = \sigma_{c'(x)}$, $x_2 = \sigma_{c'(xx_1^{-1})}, \ldots, x_k = \sigma_{c'(xx_1^{-1} \cdots x_{k-1}^{-1})}$ for $k < \mathrm{wlen}(x)$, and let $x' = xx_1^{-1} \cdots x_k^{-1}$. In our experiences, if $c'(x') \neq c(x')$,

$(m, \mathrm{wlen}(x))$	(7, 40)	(7, 55)	(7, 70)	(10, 60)	(10, 80)	(10, 100)
(a) Algorithm 1	252	324	414	487	651	790
(b) Hughes algorithm	260	368	484	515	745	1060

Table 1. Elapsed time in recovering x from $\rho_m(x)$ (unit: millisecond)

then $c'(x'x_k \cdots x_i)$ does not indicate the last generator of $x'x_k \cdots x_i \stackrel{\text{let}}{=} y$ for some $1 \le i \le k$. For this y, if $c'(y) > 1$ and every entry in the $c'(y)$-th column of $\rho_m(y)$ is in $t\mathbb{Z}[t]$, then $c'(y)-1$ has a good possibility to indicate the last generator of y from Case 3 of Theorem 1. As Table 2 shows, the success probability of Algorithm 2 is higher than Algorithm 1. However, it is slower due to this self-correction process.

Algorithm 2 Finding x from $\rho_m(x)$ with self-correction

Let $M[0], M[1], \ldots, M[l]$ be $(m \times m)$-matrices such that $M[i] = \rho_m(\sigma_{A[1]} \cdots \sigma_{A[i]})$ and $M[0] = \rho_m(e_m)$.
for i from l to 1 by -1 **do**
  Compute $j_a = c(M[i])$ and $j_c = c'(M[i])$.
  if there exists such j_c and $j_c = j_a$ **then**
    $A[i] \leftarrow j_c;\ M[i-1] \leftarrow M[i]\rho_m(\sigma_{A[i]})^{-1}$
  else
    if $i = l$ **then**
      break
    end if
    if there is $k(>i)$ such that $j_c = j_a > 1$ for $M[k]$ and every entry of $M[k]_{j_c}$ is in $t\mathbb{Z}[t]$ **then**
      $i \leftarrow k;\ A[i] \leftarrow A[i] - 1;\ M[i-1] \leftarrow M[i]\rho_m(\sigma_{A[i]})^{-1}$
    else
      break
    end if
  end if
end for
if $i = l$ **then**
  $z \leftarrow e_m$
else
  $z \leftarrow \sigma_{A[i+1]} \cdots \sigma_{A[l]}$
end if
Return(z)

4.3 Attack on the BPKE

Using the methods in section 4.1 and 4.2, this section attempts to attack the BPKE for several parameters suggested by the inventors as in Table 3, where a

m	5			7			10		
wlen(x)	30	40	50	40	55	70	60	80	100
(a) Algorithm 1	96	83	76	91	76	64	87	67	42
(b) Algorithm 2	100	99	97	99	97	82	99	90	69

Table 2. Success rate of recovering x from $\rho_m(x)$ (unit: %)

n	50	70	90	50	70	90	50	70	90
len(a)	5	5	5	7	7	7	12	12	12
len(x)	3	3	3	5	5	5	10	10	10
wlen(x)	440	884	1471	735	1487	2468	1497	2955	4924

Table 3. All parameters at [11] and wlen(x) for random x

and x are assumed to be of $\inf(a) = \inf(x) = 0$. Since any attack on the revised BPKE is effective in the original scheme, and not vice versa, we consider the revised one here. Each wlen(x) in the last row of Table 3 is the average length of ten random x's given a braid index and a canonical-length. We recall that for a braid index n, $a \in B_n, m = \lfloor \frac{n}{2} \rfloor, x_1, x_2 \in B_m$.

Computing $\boldsymbol{\rho_m(x_1)}$ or $\boldsymbol{\rho_m(x_2)}$ from $\boldsymbol{(a, x_1 a x_2)}$ From Figure 1, we know that the probability of computing $\rho_m(x_1)$ or $\rho_m(x_2)$ from $(a, x_1 a x_2)$ becomes high as len(a) increases for any n. In numerous experiments, these probabilities for the smallest len(a) (i.e., $n = 50, 70, 90$ and len(a) $= 5$) are 90% or so. And its running time is $O(m^2 \mathrm{wlen}(x_i)^2)$ as mentioned in section 4.1. Therefore, for all the parameters in Table 3, either $\rho_m(x_1)$ or $\rho_m(x_2)$ is computed from $(a, x_1 a x_2)$ with very high probability.

Recovering $\boldsymbol{x}$ from $\boldsymbol{\rho_m(x)}$ Here, $x = x_1$ or x_2 in the above paragraph. In attacking the BPKE in practice, Algorithm 2 will have much higher success probability than Algorithm 1 as in Table 2, however, it took too long time to be used on PC for large parameters. So, we used only Algorithm 1 due to its efficiency.

Table 4 shows that for the first seven parameters in Table 3 (i.e. $(n, \mathrm{len}(x)) = (50, 3), (70, 3), (90, 3), (50, 5), (70, 5), (90, 5), (50, 10))$, x is recovered from $\rho_m(x)$ with significant probability within dozens of minutes.

For the last two parameters, we could not recover any x from $\rho_{35}(x)$ (resp. $\rho_{45}(x)$) for 500 x's in B_{35} (resp. B_{45}) with len(x) $= 10$.

Compare the two cases: $(n, \mathrm{len}(x)) = (70, 5)$ and $(50, 10)$. They have similar word-lengths, see Table 3. The probability of recovering is 12% for the former and 0.2% for the latter, see Table 4. With this observation, Theorem 1 implies that our attack gives an interesting condition contrary to brute force attack for the BPKE to be secure. When the word-length of private-key is fixed, the braid

len(x)	3			5			10		
n	50	70	90	50	70	90	50	70	90
Success rate(%)	100	100	100	24	12	4	0.2	–	–
Elapsed time(min)	0.7	3.7	12.2	1	4.6	15	2.3	–	–

Table 4. Recovering x from $\rho_m(x)$

index should be large so that the BPKE is secure against brute force attack. However, it should be small against our attack.

Remark. Consider the so called generalized conjugacy problem (GCP) [11]: given $(a, x^{-1}ax)$, find x, where $a \in B_n$, $x \in LB_n$. This is one of the problems on which our attack can be mounted by restating the equations $\mathtt{e1}, \mathtt{e2}, \mathtt{e3}$ in section 4.1 as $\mathtt{e1}' : XB_1 = A_1X$, $\mathtt{e2}' : XB_2 = A_2$, $\mathtt{e3}' : B_3 = A_3X$, where $X = \rho_m(x)$. Clearly, all procedures are the same as those described in section 4.1 and 4.2. Hofheinz and Steinwandt [8] proposed an algorithm which solves the GCP more successfully and efficiently than ours. Thus, it can be used to solve the DH problem in B_n, and hence to break the original BPKE. However, since their algorithm strongly depends on the conjugacy relation between public braids, it cannot be applied to the revised BPKE.

5 Conclusions

The security of the revised BPKE as well as the original one is based on the following computational problem: given (a, x_1ax_2), find (z_1, z_2) such that $z_1az_2 = x_1ax_2$, where $a \in B_n^+$, $x_1, x_2, z_1, z_2 \in LB_n^+$. Using the Burau representation, this paper proposes a new method of solving this problem. For seven out of the nine parameters suggested at [11], we demonstrated the effectiveness of this method by recovering the private-key from the public-key with significant probability in a reasonable time (dozens of minutes plus low degree polynomial time).

On the other hand, we give a new requirement for secure parameters against our attack, which more or less conflicts with that against brute force attack. When we select private-key given word-length, its braid index should be large to defeat brute force attack. However, it should be small against our attack.

Hofheinz and Steinwandt's algorithms seem to be very efficient for large parameters, whereas they cannot be applied to the revised scheme at all. Our algorithms are effective in the revised scheme as well as the original scheme, whereas they are inefficient for very large parameters like those at [3]. Therefore, the revised BPKE with sufficiently large parameters appears to be secure at the current state of knowledge.

Acknowledgement

We would like to thank the anonymous referees for valuable comments and suggestions. And we are very grateful to Sang Geun Hahn for motivating us to make this line of research, and Jim Hughes and Sang Jin Lee for helpful discussions. The first author was supported by R&D project 2002-S-073 of KISA. And the second author was supported in part by grant no. R01-2002-000-00151-0 from the Basic Research Program of the Korea Science and Engineering Foundation.

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*, Lecture Notes in Computer Science **2020**, Springer-Verlag, pp. 13–27, 2001.
2. I. Anshel, M. Anshel, and D. Goldfeld, An algebraic method for public-key cryptography. *Mathematical Research Letters* **6**, pp. 287–291, 1999.
3. 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*, Lecture Notes in Computer Science **2248**, Springer-Verlag, pp.144–156, 2001.
4. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd Edition, The MIT Press, 2001.
5. R. Gennaro and D. Micciancio, Cryptanalysis of a pseudorandom generator based on braid groups. *Advances in Cryptology - EUROCRYPT 2002*, Lecture Notes in Computer Science **2332**, Springer-Verlag, pp. 1-13, 2002.
6. S.G. Hahn, E. Lee, and J.H. Park, Complexity of the Generalized Conjugacy Problem. To appear in *Discrete Applied Mathematics*. Available at `http://crypt.kaist.ac.kr/pre_papers/DA5640.ps`.
7. J. Hoffstein, J. Pipher, and J.H. Silverman, NTRU: a Ring based Publick Key Cryptosystem. *Algorithmic Number Theory Symposium - ANTS III*, Lecture Notes in Computer Science **1423**, Springer-Verlag, pp. 267–288, 1999.
8. D. Hofheinz and R. Steinwandt, A Practical Attack on Some Braid Group Based Cryptographic Primitives, *Public Key Cryptography - PKC 2003*, Lecture Notes in Computer Science **2567**, Springer-Verlag, pp. 187–198, 2003.
9. J. Hughes, A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem. *7th Australasian Conference of Information Security and Privacy - ACISP 2002*, Lecture Notes in Computer Science **2384**, Springer-Verlag, pp. 176–189, 2002.
10. J. Hughes and A. Tannenbaum, Length-Based Attacks for Certain Group Based Encryption Rewriting Systems, *Workshop SECI02 SÉcurité de la Communication sur Intenet, September, 2002*. Available at `http://www.network.com/hughes/SECI02.pdf`.
11. K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, and C. Park, New Public-key Cryptosystem Using Braid Groups. *Advances in Cryptology - Crypto 2000*, Lecture Notes in Computer Science **1880**, Springer-Verlag, pp. 166–183, 2000.
12. E. Lee, S.J. Lee, and S.G. Hahn, Pseudorandomness from Braid Groups. *Advances in Cryptology - CRYPTO 2001*, Lecture Notes in Computer Science **2139**, Springer-Verlag, pp. 486–502, 2001.
13. S. J. Lee and E. Lee, Potential Weaknesses of the Commutator Key Agreement Protocol based on Braid Groups. *Advances in Cryptology - EUROCRYPT 2002*, Lecture Notes in Computer Science **2332**, Springer-Verlag, pp. 14–28, 2002.
14. P.Q. Nguyen and D. Pointcheval, Analysis and Improvements of NTRU Encryption Paddings. *Advances in Cryptology - CRYPTO 2002*, Lecture Notes in Computer Science **2442**, Springer-Verlag, pp. 210–215, 2002.

15. S.H. Paeng, K.C. Ha, J.H. Kim, S. Chee, and C. Park, New Public Key Cryptosystem Using Finite Non Abelian Groups. *Advances in Cryptology - CRYPTO 2001*, Lecture Notes in Computer Science **2139**, Springer-Verlag, pp. 470–485, 2001.
16. P.W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal of Computing* **26**, pp. 1484–1509, 1997.
17. V. Wheatman and L. McRory, Quantum Computers: The End of Public-Key Cryptography? *Gartner research note*, January 2002.