A Practical Attack on Some Braid Group Based Cryptographic Primitives

Dennis Hofheinz and Rainer Steinwandt

IAKS, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth
Fakultät für Informatik, Universität Karlsruhe
Am Fasanengarten 5, 76 131 Karlsruhe, Germany

Abstract. A simple heuristic approach to the conjugacy problem in braid groups is described. Although it does not provide a general solution to the latter problem, it demonstrates that various proposed key parameters for braid group based cryptographic primitives do not offer acceptable cryptographic security. We give experimental evidence that it is often feasible to reveal the secret data by means of a normal PC within a few minutes.

Keywords: braid groups, cryptanalysis

1 Introduction

Within the last years various attempts have been made to derive cryptographic primitives from problems originating in combinatorial group theory (see, e. g., [WM85, Wag90, GZ91, AAG99, KLC+00, AAFG01]). One theoretically rather appealing family of schemes is derived from braid groups. These finitely presented groups are well-studied (e. g., [Gar69, Bir74]), and various proposals have been made for deriving cryptographic primitives from the conjugacy problem in these groups.

No general efficient solution for the conjugacy problem in braid groups is known so far, but the attacks from [LL02, Hug02] exhibit weaknesses in the key agreement scheme considered in [AAG99, AAFG01]. The attacks of J. Hughes, S.J. Lee and E. Lee focus on the so-called *multiple simultaneous conjugacy problem* in braid groups, and they propose a modification of the original specification to avoid their attack on the scheme of [AAG99, AAFG01].

In this contribution we demonstrate that the modification considered there is not sufficient for saving the scheme in [AAG99, AAFG01]. Moreover, we demonstrate the vulnerability of the key exchange scheme and the public key cryptosystem from [KLC+00] with respect to a simple heuristic procedure for the conjugacy problem. This procedure does by no means provide a general efficient method for solving the conjugacy problem in braid groups, but our experimental results demonstrate that with the proposed parameter choices both the key agreement and the encryption scheme from [KLC+00] do not offer acceptable cryptographic security.

Y.G. Desmedt (Ed.): PKC 2003, LNCS 2567, pp. 187–198, 2003.
© Springer-Verlag Berlin Heidelberg 2003

In more detail our contribution is organized as follows: in the next section we recall some basic terminology concerning braid groups with the main focus on some algorithmic aspects of the word and conjugacy problem. Thereafter we describe our heuristic approaches for tackling the variants of the conjugacy problem underlying the proposals in [AAG99, KLC+00, AAFG01]. We recall the set-up of these proposals to the extent necessary for describing our attack and give experimental results that illustrate the cryptographic relevance of our approach. Finally, some conclusions are given.

2 Braid Groups

The *braid group* B_n ($n \in \mathbb{N}$) is a finitely presented group that is defined through the following presentation (cf. [Art25]):

$$\left\langle \sigma_1, \ldots, \sigma_{n-1} \middle| \begin{array}{rcll} \sigma_i\sigma_j\sigma_i & = & \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1 \\ \sigma_i\sigma_j & = & \sigma_j\sigma_i & \text{if } |i-j| > 1 \end{array} \right\rangle$$

We refer to $\sigma_1, \ldots, \sigma_{n-1}$ as *(Artin) generators* and to arbitrary elements of B_n as *braids*. If we refer to a concrete representation of a braid in terms of Artin generators, we use the term *braid word*. A braid is said to be *positive* iff it can be written as a product of generators σ_i, i. e., in this case no negative powers of the σ_i are involved. Here the identity $\varepsilon \in B_n$ is also regarded as positive, and it can be shown that the positive braids in B_n form a *monoid* B_n^+ which embeds into B_n (cf. [Gar69]).

2.1 The Δ-Normal Form

Setting inductively $\Delta_1 := \sigma_1$ and $\Delta_i = \sigma_1 \cdots \sigma_i \cdot \Delta_{i-1}$ for $1 < i < n$, we define the *fundamental braid* $\Delta \in B_n$ as $\Delta := \Delta_{n-1}$. Next, we establish a partial ordering $\leq$ on the elements of B_n by setting $v \leq w$ iff there are positive braids $\alpha, \beta \in B_n^+$ such that $w = \alpha v \beta$. Now any braid $\alpha \in B_n$ which satisfies $\varepsilon \leq \alpha \leq \Delta$ is called a *canonical factor*. There is a *canonical homomorphism* $\pi : B_n \to S_n$ from the braid group B_n into the symmetric group S_n which maps σ_i onto the transposition interchanging i and $i+1$ and whose restriction to the set of canonical factors in B_n induces a bijection; see [EM94].

A factorization $\gamma = \alpha\beta$ of a positive braid γ into a canonical factor α and a positive braid β is said to be *left-weighted* iff α has the maximal length among all such decompositions. A *right-weighted* factorization is defined analogously. For any braid $w \in B_n$ we denote the greatest $i \in \mathbb{Z}$ with $\Delta^i \leq w$ by $\inf w$; analogously, $\sup w$ stands for the smallest $i \in \mathbb{Z}$ with $w \leq \Delta^i$. With this notation every braid $w \in B_n$ can be written *uniquely* as

$$w = \Delta^r W_1 \cdots W_s \tag{1}$$

with $r = \inf w$, $s = \sup w - \inf w$ and canonical factors $\varepsilon < W_i < \Delta$ such that $W_i W_{i+1}$ is left-weighted for $1 \leq i < s$ (cf. [Gar69]). In this context, we refer to s as the *canonical length* of w.

The explicit decomposition (1) is called the *Δ-normal form* or simply the *normal form* of w. Note here that the W_i are canonical factors, so each of them can be represented uniquely by the corresponding permutation $\pi(W_i)$ in the symmetric group S_n. For a given braid word $w \in B_n$, its normal form can be computed in time $\mathcal{O}(|w|^2 n \log n)$ with $|\cdot|$ denoting the word length (see [CKL+01] for details).

2.2 The Conjugacy Problem in B_n

For convenience, we introduce the notion of a *tail*: $\gamma \in B_n^+$ is said to be a *tail* of some braid $\alpha \in B_n^+$, iff there is a factorization $\alpha = \beta\gamma$ with $\beta \in B_n^+$. Also we define the automorphism τ of B_n through $\tau(w) := \Delta^{-1} w \Delta$ ($w \in B_n$). With this terminology we can formulate a lemma that is of importance for the heuristic approach to the conjugacy problem discussed below; the first part of this result has already been stated in [LL02]:

Lemma 1. *Let $v, w \in B_n$ be two positively conjugate braid words such that $w = \alpha^{-1} v \alpha$ for some $\alpha \in B_n^+$. Let $\Delta^r W_1 \cdots W_s$ be the normal form of w. Then the following relations hold:*

- *If $\inf w < \inf v$, then the canonical factor $\Delta\tau^r(W_1^{-1})$ is a tail of α.*
- *If $\sup w > \sup v$, then W_s is a tail of α.*

Proof. A proof of the first claim can be found in the proof of Lemma 4.3 in [EM94]; using the terminology from [EM94], the second part follows immediately by noticing that 'reverse cycling' some braid word $w \in B_n$ can be achieved by cycling its inverse w^{-1}. □

Note that the restriction to *positively* conjugated braids is not really a restriction: suppose we have $v, w \in B_n$ with $w = x^{-1} v x$ for some braid word $x \in B_n$, whose normal form is $\Delta^r X_1 \cdots X_s$, with $r \in \mathbb{Z}$ possibly non-zero. Since Δ^2 lies in the centre of B_n (cf. [Gar69]), then also for $\tilde{x} := \Delta^{r \bmod 2} X_1 \cdots X_s$ we find $w = \tilde{x}^{-1} v \tilde{x}$, and $\tilde{x}$ is obviously positive.

Lemma 1 tells us for any given braid word $w \in B_n$ how to eventually find a word with minimal canonical length in w's conjugacy class. (To see this, note that minimal canonical length is equivalent to minimal sup and maximal inf.) The set of all words with minimal canonical length in w's conjugacy class is called the *super summit set* $\mathcal{S}(w)$ of w and can be shown to be finite; moreover, there is an algorithm for computing $\mathcal{S}(w)$ for any given braid word $w \in B_n$ (cf. [EM94]). Of course, two super summit sets $\mathcal{S}(v), \mathcal{S}(w)$ of braid words $v, w \in B_n$ are either equal or disjoint. So we eventually obtain an algorithm for the conjugacy problem in the braid group B_n: given $v, w \in B_n$, we compute $\mathcal{S}(v)$ and *one* element w' of $\mathcal{S}(w)$. Then v and w are conjugate iff $w' \in \mathcal{S}(v)$. This approach cannot only be used to decide whether v and w are conjugated; it can also be used to obtain a positive braid word $\alpha \in B_n^+$ with $w = \alpha^{-1} v \alpha$. Unfortunately, the complexity of this algorithm is not clear, since no precise estimate for the cardinality of $\mathcal{S}(v)$ is known.

Remark There is another presentation for the braid group B_n due to Birman, Ko, and Lee [BKL98]. In this presentation, we may find a unique normal form for any braid word $w \in B_n$ in time $\mathcal{O}(|w|^2 n)$.

3 A Heuristic Approach to the Conjugacy Problem

Given two conjugate braid words $v, w \in B_n$, we would like to find a conjugating braid $\alpha \in B_n$ so that $w = \alpha^{-1} v \alpha$. Actually, for certain proposed parameters $n \in \mathbb{N}$ and specific braids $v, w \in B_n$, this problem is hoped to be computationally intractable and has been suggested as a tool for deriving cryptographic primitives (see, e. g., [KLC+00, AAFG01]). In the sequel, we present an algorithm using Lemma 1 and some additional heuristics which tries to solve the conjugacy problem for the parameter sizes considered in proposals for cryptographic primitives. We do not aim at providing a general efficient solution for the conjugacy problem in braid groups. Nevertheless, the heuristic approach described below exhibits security problems in several cryptographic proposals based on braid groups.

3.1 The Algorithm

Consider Algorithm A (shown in Figure 1) for tackling an instance of the conjugacy problem in B_n (the function `GuessPermutation` is discussed in the next section).

Algorithm A:

- Input: $v, w \in B_n$ with $w = x^{-1} v x$ for some unknown $x \in B_n^+$ with $\inf x = 0$.
- Output: either $\alpha \in B_n^+$ with $w = \alpha^{-1} v \alpha$ or '`failed`'.

```
1. Initialize α as the empty word ε.
2. Put v and w in normal form, so that w = Δ^r W_1 ··· W_s.
3. While inf w < inf v do
       Let γ := Δτ^r(W_1^{-1}), α := γα, w := γwγ^{-1},
       Put w in normal form as in 2.
4. While sup w > sup v do
       Let γ := W_s, α := γα, w := γwγ^{-1},
       Put w in normal form as in 2.
5. Let μ :=GuessPermutation(v, w), α := μα, w := μwμ^{-1}.
6. If v = w, then
       Return α,
     else
       Return 'failed'.
```

Fig. 1. Algorithm A

As mentioned already, the restriction to inputs $v, w \in B_n$ that are *positively* conjugated is not relevant. Requiring the conjugating braid $x = \Delta^r X_1 \cdots X_s$ to fulfill the condition $r(= \inf x) = 0$ is of no practical relevance, either: if $w = x^{-1}vx$ holds, then also $w = \tilde{x}^{-1}v\tilde{x}$ for $\tilde{x} := \Delta^{r \bmod 2} X_1 \ldots X_s$ holds. But either $\tilde{x}$ or $\Delta^{-1}\tilde{x}$ is a positive braid with zero inf. So in the general case it suffices to run Algorithm A *twice* (maybe in parallel): once with inputs v, w and once with inputs $\Delta^{-1}v\Delta, w$.

3.2 Discussion of the Algorithm

Let us take a closer look at the behaviour of Algorithm A: in Step `1` and Step `2` our guess α for the conjugating element x is initialized and the words $v, w \in B_n$ are prepared. Steps `3` and `4` conjugate w in a way that after completion of these steps the canonical length of our new w is not greater than the canonical length of v. Using Lemma 1, at the beginning of Step `5` we therefore have $w = x'^{-1}vx'$ with $x = x'\alpha$ and $x' \in B_n^+$.

Now we assume that in this situation, the 'not yet uncovered' part x' of the conjugating word x is 'essentially determined' by its induced permutation $\pi(x')$. Consequently, in Step `5`, we *guess* the permutation $\pi(x')$, i. e., the function `GuessPermutation(v,w)` is to return a canonical factor μ with $\pi(\mu) = \pi(x')$. To find such a braid word μ in the general case, we would have to solve the conjugacy problem $\pi(w) = \pi(x')^{-1}\pi(v)\pi(x')$ in the symmetric group S_n; also there might be a vast number of possible solutions for $\pi(x')$ when we consider *pure* braids v, i. e., braids $v \in B_n$ satisfying $\pi(v) = \mathrm{id}$. We'll discuss this 'pure case' below. For non-pure braids v, experimentally it turned out that in many cases $\pi(x')$ is a sufficiently 'simple' permutation and that the simple implementation of `GuessPermutation` shown in Figure 2 works fine (it exploits that in cycle notation conjugating a permutation σ by a permutation τ means applying τ to the 'entries' of σ).

Of course, in the case of pure braid words v we have to modify the function `GuessPermutation` accordingly. A promising heuristic approach seems then to be the following: supposing that $v = \Delta^{r_v} V_1 \cdots V_{s_v}$ and $w = \Delta^{r_w} W_1 \cdots W_{s_w}$ are in their normal forms, chances are good that $\mu = \pi^{-1}\left(\pi(V_1 W_1^{-1})\right)$ or $\mu = \pi^{-1}\left(\pi(V_{s_v}^{-1} W_{s_w})\right)$. For our experiments, we optimized this strategy a little: in case neither one of these candidates for μ solves our conjugation problem in B_n, we then guess the *right meet* of these candidates as only a *tail* of μ; here by a *right meet* of two positive braid words α, β we mean a tail of *both* α and β with maximal length in generators.

Back to Algorithm A, in Step `6` we return α in case it actually is a conjugating element.

Note that neither Algorithm A nor the function `GuessPermutation` is probabilistic, so there is no chance that, once one of them has failed, successive applications might improve this result. In particular, there is lots of room for improving and adding heuristics to Algorithm A. E. g., in our experiments with the following variation we were able to solve the conjugacy problem in certain cases in which at the beginning of Step `5` no canonical factor μ alone satisfied

Function `GuessPermutation`**:**

- Input: $v, w \in B_n$ with $w = x^{-1}vx$, for some unknown $x \in B_n^+$ with $\inf x = 0$.
- Output: a canonical factor $\mu \in B_n^+$, such that $\pi(\mu) = \pi(x)$.

```
1. Let τ ∈ S_n be the identity permutation.
2. Let (χ_1, ..., χ_n) := (false, ..., false).
3. For i from n downto 1 do
       Let r := i, s := i,
       While χ_r = false do
           Let χ_r := true,
           Let r := π(v)(r), s := π(w)(s),
           If r ≠ s, then
               Let τ(r) := s.
4. Let (ξ_1, ..., ξ_n) := (false, ..., false).
5. For i from n downto 1 do
       If ξ_i = false, then
           Let ξ_i := true, r := i,
           While ξ_{τ(r)} = false and τ(r) ≠ r do
               Let r := τ(r), ξ_r := true.
           Let τ(r) := i.
6. Return π^{-1}(τ).
```

Fig. 2. The function `GuessPermutation`

$w = \mu^{-1}v\mu$: instead of guessing the *whole* braid word μ in Step `5`, one guesses only a *tail* of some canonical factor μ satisfying $\pi(\mu) = \pi(x)$. Then in Step `6`, if we are not yet finished (i. e., if $v \neq w$), we repeat Steps `3`–`5` until we are.—Of course, here we rely on the assumption that the tails of μ we guess in Step `5` are indeed tails of x.

3.3 The Multiple Simultaneous Conjugacy Problem

Algorithm A can be modified to work for an instance of the *multiple simultaneous conjugacy problem*: given $m \geq 1$ braid words $v_i, w_i \in B_n$ with $w_i = x^{-1}v_ix$ ($1 \leq i \leq m$) for some unknown braid $x \in B_n$, we are looking for some $\alpha \in B_n$ which also satisfies $w_i = \alpha^{-1}v_i\alpha$ for all i. Consider the modification of Algorithm A shown in Figure 3, which can be regarded as an extension to the algorithm used in [LL02, Theorem 2].

Here the function `GuessSimultPerm` takes as input pairs $(v_i, w_i) \in B_n^2$ of braid words such that $w_i = x^{-1}v_ix$ for all i, and yields a guess for a canonical factor β with $\pi(\beta) = \pi(x)$ as a result. For implementing this function, similar ideas as for the function `GuessPermutation` can be used; we skip a detailed discussion of this topic and do not give a sample implementation of `GuessSimultPerm` here.

However, it is worthwhile to remark that—just like in the case of Algorithm A—sometimes it may be helpful to use a variation of Algorithm B which in Step `5` only guesses some *tail* of μ and then repeats Steps `3`–`6` as necessary.

Algorithm B:

- Input: $m \geq 1$ pairs $(v_i, w_i) \in B_n^2$ such that $w_i = x^{-1} v_i x$ $(1 \leq i \leq m)$ for some unknown $x \in B_n^+$ with $\inf x = 0$.
- Output: either $\alpha \in B_n^+$ with $w_i = \alpha^{-1} v_i \alpha$ for all i or '`failed`'.

1. Initialize α as the empty word ε.
2. Put all v_i and all w_i in normal form,
   so that $w_i = \Delta^{r_i} W_1^{(i)} \cdots W_{s_i}^{(i)}$.
3. While $\inf w_j < \inf v_j$ for some $j \in \{1, \ldots, m\}$ do
   Let $\gamma := \Delta\tau^r((W_1^{(j)})^{-1})$, $\alpha := \gamma\alpha$,
   Let $w_i := \gamma w_i \gamma^{-1}$ for all $i \in \{1, \ldots, m\}$,
   Put all w_i in normal form as in 2.
4. While $\sup w_j > \sup v_j$ for some $j \in \{1, \ldots, m\}$ do
   Let $\gamma := W_s^{(j)}$, $\alpha := \gamma\alpha$,
   Let $w_i := \gamma w_i \gamma^{-1}$ for all $i \in \{1, \ldots, m\}$,
   Put all w_i in normal form as in 2.
5. Let $\mu :=$GuessSimultPerm$((v_1, w_1), \ldots, (v_m, w_m))$.
6. Let $\alpha := \mu\alpha$, and $w_i := \mu w_i \mu^{-1}$ for all $i \in \{1, \ldots, m\}$.
7. If $v_i = w_i$ for all i, then
   Return α,
   else
   Return 'failed'.

Fig. 3. Algorithm B

Also another heuristic step turned out to be helpful: after Step 4, it can be worthwhile to 'combine' pairs of braid words v_i, w_i to extend the set of these pairs. Namely, if $w_i = x^{-1} v_i x$ for $i = 1, \ldots, m$, then also $w' = x^{-1} v' x$, where $v' = v_i^{-1} v_j v_i$ and $w' = w_i^{-1} w_j w_i$ for any $i, j \in \{1, \ldots, m\}$. In particular, new found pairs v', w' with $\inf w' < \inf v'$ or $\sup w' > \sup v'$ are interesting in view of a re-application of Steps 3–4.

4 Braid Groups in Cryptography

Several suggestions have been put forward for deriving cryptographic primitives from the conjugacy problem in braid groups. In the sequel we shortly recall the key agreement schemes from [KLC+00, AAFG01] and the public key encryption scheme from [KLC+00]; all of these schemes are based on variants of the conjugacy problem in braid groups. We do not give all details of these proposals and put our main focus on the suggested parameters of the underlying variant of the conjugacy problem. Then we apply the techniques described above to concrete instances of these systems with realistic parameter sizes to get an idea of the practical significance of our approach. For our experiments we used Linux PCs with a clock rate of 1.8 GHz and the `CBraid` package of [Cha01]. The concrete running times varied in dependence on the actual parameter choice and the

type of the conjugacy problem: while for individual conjugacy problems usually a few seconds were sufficient, instances of the multiple simultaneous conjugacy problem typically required some (less than 30) minutes of CPU time.

4.1 Commutator Based Key Agreement Protocol

In [AAG99], a key agreement protocol has been proposed that is based on the multiple simultaneous conjugacy problem. In [AAFG01], the addition of a so-called *key extractor* is suggested. But as the latter modification does not affect our attack, we omit the details on the key extractor in our description:

Public information:
- braid index $n \in \mathbb{N}$
- a subgroup $G_A = \langle a_1, \ldots, a_r \rangle \leq B_n$
- a subgroup $G_B = \langle b_1, \ldots, b_s \rangle \leq B_n$

Private key:
- Alice selects $a \in G_A$
- Bob selects $b \in G_B$

Public key:
- Alice publishes $[a^{-1}b_1a, \ldots, a^{-1}b_sa]$
- Bob publishes $[b^{-1}a_1b, \ldots, b^{-1}a_rb]$

Shared key: derived from $a^{-1}b^{-1}ab$

In [AAG99], no concrete parameter choices for the above scheme are suggested. So for our experiments we used the parameters suggested in [AAFG01]. Namely, we chose the braid index $n = 80$ and public subgroups G_A, $G_B \leq B_n$ with $r = s = 20$ generators each. Each of the generators a_i resp. b_i of the public subgroups was comprised of $5 \leq \ell \leq 10$ Artin generators. The private keys a and b were made up of 100 public generators each.

Using Algorithm B and some of the techniques described in Section 3.3 to attack the multiple simultaneous conjugacy problem with these parameters, we obtained the following success rates for our attack on the secret key of Alice (resp. Bob):

n	$r = s$	ℓ	number of samples	success rate
80	20	5	1000	99.0%
80	20	10	1000	98.9%

To prevent the attack described in [LL02], in [LL02] S. J. Lee and E. Lee suggest to take *pure* braid words a_i, resp. b_i as public generators. More precisely, they suggest to raise any one of the already created generators a_i (resp. b_i) to a suitable power, so that the induced permutation becomes the identity. We implemented this construction and applied Algorithm B with some additional ideas from Sections 3.2 and 3.3. Our results are shown below:

n	$r = s$	ℓ	number of samples	success rate
80	20	5	1000	100.0%
80	20	10	1000	100.0%

Based on a reference to a private communication with one of the authors of [AAFG01], in [Hug02] J. Hughes describes a slightly different parameter choice. Namely, the public generators a_i resp. b_i are words of length 5 comprised of Artin generators and their inverses. Also, when Alice resp. Bob forms the secret word a resp. b, the description of [Hug02] suggests to use a word of length 100 in the public generators a_i resp. b_i and their inverses.

We've also done a few experiments with such a modified parameter choice, but these parameters also turned out to be vulnerable with respect to our attack, and at the moment it is unclear how a practical parameter choice for the schemes from [KLC+00, AAFG01] should look like.

4.2 A Diffie-Hellman Type Key Agreement Protocol and a Public Key Cryptosystem

The following key agreement protocol has been proposed in [KLC+00]. Analogously as in the commutator based scheme just described, in [AAFG01] a modification by means of a so-called key extractor is suggested. This modification does not affect our attack and is therefore not discussed in the sequel.

We denote by LB_ℓ resp. RB_r $(1 < \ell, r < n)$ the subgroup of B_n generated by $\sigma_1, \ldots, \sigma_{\ell-1}$ resp. $\sigma_{n-r+1}, \ldots, \sigma_{n-1}$. Now a key agreement scheme can be described as follows:

Public information:
- braid index $n \in \mathbb{N}$
- integers $\ell, r \in \mathbb{N}$ with $\ell + r = n$
- a braid word $x \in B_n$

Private key:
- Alice selects $a \in LB_\ell$
- Bob selects $b \in RB_r$

Public key:
- Alice publishes $a^{-1}xa$
- Bob publishes $b^{-1}xb$

Shared key: derived from $(ab)^{-1}x(ab)$

The parameters considered in [KLC+00, AAFG01] for this key exchange protocol coincide with those for a public key cryptosystem also described in [KLC+00]. Hence, we recall the description of this encryption scheme before giving our experimental results. For this we denote by $H : B_n \longrightarrow \{0,1\}^k$ an ideal hash function from the braid group B_n to the message space $\{0,1\}^k$.

Public information:
- braid index $n \in \mathbb{N}$
- integers $\ell, r \in \mathbb{N}$ with $\ell + r = n$

Private key: Alice chooses $a \in LB_\ell$

Public key:
- Alice publishes a braid word $x \in B_n$, and
- she publishes $y := a^{-1}xa$

Encryption: To transmit $m \in \{0,1\}^k$ to Alice,
- Bob chooses $b \in RB_r$ at random, and
- he sends $(c,d) := (b^{-1}xb, H(b^{-1}yb) \text{ XOR } m)$ to Alice

Decryption: Alices computes $m = H(a^{-1}ca) \text{ XOR } d$

For our experiments we used the parameters that have also been considered in [KLC+00, AAFG01]. Namely, we used braid groups of index $n \geq 45$ and chose $\ell = \lfloor n/2 \rfloor$, $r = n - \ell$. For x and a (resp. b) we used braid words of canonical length $p \geq 3$. In [KLC+00] the braid word x is required to be 'sufficiently complicated'. Lacking a specification on how to determine such a 'sufficiently complicated' $x \in B_n$, we decided to pick such a braid word in the same manner as proposed in [CKL+01] for generating 'random braid words'. Then we attacked so-obtained instances $y = a^{-1}xa$ of the *generalized conjugacy problem* in B_n by Algorithm A. Below are the results of our experiments for some proposed parameters n and p, where p denotes the canonical length of both x and a (resp. b):

n	p	Total samples	Success rate
45	3	1000	78.1%
50	5	1000	79.1%
70	7	1000	79.0%
90	12	1000	80.0%

Without further specification, in [KLC+00] it is also suggested to use pure braid words x. We decided to generate pure braids as follows: after generating a 'random' braid word x of canonical length p as above, we appended to x the (simple) braid word $\pi^{-1}\left(\pi\left(x^{-1}\right)\right)$, and attacked the so-obtained instances of the system with Algorithm A and the variant of `GuessPermutation` discussed in section 3.2. The figures below show the results of these attacks:

n	p	Total samples	Success rate
45	3	1000	76.9%
50	5	1000	82.4%
70	7	1000	87.5%
90	12	1000	88.9%

So from a cryptographic point of view the parameters in [KLC+00, AAFG01] look rather worrisome, and it remains open how to generate practical instances of the above schemes.

5 Conclusion

We have described a heuristic algorithm for the conjugacy problem in braid groups. This algorithm does not solve the conjugacy problem in the general case, yet it applies in most of the cases considered for public key cryptosystems. We have run various experiments with parameters proposed for braid group cryptosystems to back this result; indeed, we do not know of any public key

cryptosystem based on the conjugacy problem in braid groups whose proposed parameters yield hard instances of this problem with respect to our algorithm.

Furthermore, we believe that our algorithm can be improved to succeed for some parameters not considered yet for cryptographic applications. So it remains unclear how to efficiently find hard instances of the conjugacy problem in braid groups for such purposes.

References

[AAFG01] Iris Anshel, Michael Anshel, Benji Fisher, and Dorian Goldfeld. New Key Agreement Protocols in Braid Group Cryptography. In David Naccache, editor, *"Topics in Cryptology — CT-RSA 2001"*, volume 2020 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 2001. 187, 188, 190, 193, 194, 195, 196

[AAG99] Iris Anshel, Michael Anshel, and Dorian Goldfeld. An Algebraic Method for Public-Key Cryptography. *Mathematical Research Letters*, 6:287–291, 1999. 187, 188, 194

[Art25] Emil Artin. Theorie der Zöpfe. *Hamb. Abh.*, 4:47–72, 1925. 188

[Bir74] Joan S. Birman. *Braids, Links, And Mapping Class Groups*. Number 82 in Annals of Mathematics Studies. Princeton University Press and University of Tokyo Press, Princeton, New Jersey, 1974. 187

[BKL98] Joan S. Birman, Ki Hyoung Ko, and Sang Jin Lee. A new approach to the word and conjugacy problems in the braid groups. *Advances in Mathematics*, 139:322–353, 1998. 190

[Cha01] Jae Choon Cha. CBraid: a C++ library for computations in braid groups, 2001. At the time of writing available electronically at `http://knot.kaist.ac.kr/~jccha/cbraid/`. 193

[CKL+01] Jae Choon Cha, Ki Hyoung Ko, Sang Jin Lee, Jae Woo Han, and Jung Hee Cheon. An Efficient Implementation of Braid Groups. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 144–156. Springer, 2001. 189, 196

[EM94] Elsayed A. Elrifai and H. R. Morton. Algorithms for positive braids. *Quarterly Journal of Mathematics Oxford*, 45:479–497, 1994. 188, 189

[Gar69] F. A. Garside. The Braid Group and Other Groups. *Quarterly Journal of Mathematics Oxford*, 20:235–254, 1969. 187, 188, 189

[GZ91] Max Garzon and Yechezkel Zalcstein. The Complexity of Grigorchuk groups with application to cryptography. *Theoretical Computer Science*, 88:83–98, 1991. 187

[Hug02] Jim Hughes. A Linear Algebraic Attack on the AAFG1 Braid Group Cryptosystem. In Lynn Batten and Jennifer Seberry, editors, *Information Security and Privacy. 7th Australasian Conference, ACISP 2002*, volume 2384 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2002. 187, 195

[KLC+00] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju sung Kang, and Choonsik Park. New Public-Key Cryptosystem Using Braid Groups. In Mihir Bellare, editor, *Advances in Cryptology — CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2000. 187, 188, 190, 193, 195, 196

[LL02] Sang Jin Lee and Eonkyung Lee. Potential Weaknesses of the Commutator Key Agreement Protocol Based On Braid Groups. In Lars Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 2002. 187, 189, 192, 194

[Wag90] Neal R. Wagner. Searching for Public-Key Cryptosystems. In *Proceedings of the 1984 Symposium on Security and Privacy (SSP '84)*, pages 91–98, Los Angeles, Ca., USA, 1990. IEEE Computer Society Press. 187

[WM85] Neal R. Wagner and Marianne R. Magyarik. A Public Key Cryptosystem Based on the Word Problem. In G. R. Blakley and D. Chaum, editor, *Advances in Cryptology. Proceedings of CRYPTO 1984*, volume 196 of *Lecture Notes in Computer Science*, pages 19–36. Springer, 1985. 187