# A practical attack on a braid group based cryptographic protocol


Alexei Myasnikov[1], Vladimir Shpilrain[2], and Alexander Ushakov[3]

[1] Department of Mathematics, McGill University, Montreal, Quebec H3A 2T5
alexeim@math.mcgill.ca *

[2] Department of Mathematics, The City College of New York, New York, NY 10031
shpilrain@yahoo.com **

[3] Department of Mathematics, CUNY Graduate Center, New York, NY 10016
aushakov@mail.ru ***



**Abstract.** In this paper we present a practical heuristic attack on the Ko, Lee et al. key exchange protocol introduced at Crypto 2000 [11]. Using this attack, we were able to break the protocol in about 150 minutes with over 95% success rate for typical parameters. One of the ideas behind our attack is using Dehornoy's handle reduction method as a counter measure to diffusion provided by the Garside normal form, and as a tool for simplifying braid words. Another idea employed in our attack is solving the *decomposition problem* in a braid group rather than the *conjugacy search problem*.


## 1 Introduction

Braid group cryptography has attracted a lot of attention recently due to several suggested key exchange protocols (see [1], [11]) using braid groups as a platform. We refer to [2], [6] for more information on braid groups.

Here we start out by giving a brief description of the Ko, Lee et al. key exchange protocol (subsequently called just the Ko-Lee protocol).

Let $B_{2n}$ be the group of braids on $2n$ strands and $x_1, \ldots, x_{2n-1}$ its standard generators. Define two subgroups $L_n$ and $R_n$ of $B_{2n}$ as follows:

$$L_n = \langle x_1, \ldots, x_{n-1} \rangle$$

and

$$R_n = \langle x_{n+1}, \ldots, x_{2n-1} \rangle.$$

Clearly, $L_n$ and $R_n$ commute elementwise. The Ko-Lee protocol [11] is the following sequence of operations:


* Partially supported by the NSF grant DMS-0405105.

** Partially supported by the NSF grant DMS-0405105.

*** Partially supported by Umbanet Inc. through an award from the U.S. Department of Commerce NIST, Advanced Technology Program, Cooperative Agreement No. 70NANB2H3012

**(0)** One of the parties (say, Alice) publishes a random element $w \in B_{2n}$ (the "base" word).

**(1)** Alice chooses a word $a$ as a product of generators of $L_n$ and their inverses. The word $a$ is Alice's private key.

**(2)** Bob chooses a word $b$ as a product of generators of $R_n$ and their inverses. The word $b$ is Bob's private key.

**(3)** Alice sends a normal form of the element $a^{-1}wa$ to Bob and Bob sends a normal form of the element $b^{-1}wb$ to Alice.

**(4)** Alice computes a normal form of

$$K_a = a^{-1}b^{-1}wba$$

and Bob computes a normal form of

$$K_b = b^{-1}a^{-1}wab.$$

Since $ab = ba$ in $B_{2n}$, the normal forms of $K_a$ and $K_b$ coincide. Thus Alice and Bob have the same normal form called their *shared secret key.*

We note that a particular normal form used in [11] is called the *Garside normal form* (see our Section 2).

Initially, the security of this problem was claimed to depend on the complexity of the *conjugacy search problem* in $B_{2n}$ which is the following: for a given pair of words $w_1, w_2$ such that $w_1$ is conjugate to $w_2$ in $B_{2n}$, find a particular conjugator, i.e. a word $x$ such that $w_1 = x^{-1}w_2x$. However, it was shown in [13] that solving the conjugacy search problem is not necessary to break the Ko-Lee protocol. More precisely, it was shown that for an adversary to get the shared secret key, it is sufficient to find a pair of words $a_1, a_2 \in L_n$ such that $w_1 = a_1wa_2$. Then $K_a = K_b = a_1b^{-1}wba_2$, where the element $b^{-1}wb$ is public because it was transmitted at step 3. The latter problem is usually called the *decomposition problem.* The fact that it is sufficient for the adversary to solve the decomposition problem to get the shared secret key was also mentioned, in passing, in the paper [11], but the significance of this observation was downplayed there by claiming that solving the decomposition problem does not really give a computational advantage over solving the conjugacy search problem.

In this paper, we show (experimentally) that a particular heuristic attack on the Ko-Lee protocol based on solving the decomposition problem is, in fact, by far more efficient than all known attacks based on solving the conjugacy search problem. With the running time of 150 minutes (on a cluster of 8 PCs with 2GHZ processor and 1GB memory each), the success rate of our attack program was over 96%; see Section 5 for more details.

We note that there is a polynomial-time *deterministic* attack on the Ko-Lee protocol based on solving a variant of the conjugacy search problem [3], but the authors of [3] acknowledge themselves that their attack is not practical and, in fact, has not been implemented.

Another idea employed in our attack is using Dehornoy's forms [4] for recovering words from Garside normal forms and for solving the decomposition problem. In the Ko-Lee protocol, Garside's algorithm for converting braid words into normal forms plays the role of a diffusion algorithm. We show (experimentally) that Dehornoy's algorithm can be used to weaken the diffusion and make the protocol vulnerable to a special kind of length based attack (see [7], [8], [9] for different versions of length based attacks).

To conclude the introduction, we note that several other, less efficient, attacks on the Ko-Lee protocol were suggested before; we refer to [5] for a comprehensive survey of these attacks as well as for suggestions on countermeasures.

*Acknowledgments.* We are grateful to R. Haralick for making a computer cluster in his lab available for our computer experiments.

## 2 Converting Garside normal forms to words

The Garside normal form of an element $a \in B_n$ is the pair $(k,\ (\xi_1, \ldots, \xi_m))$, where $k \in \mathbb{Z}$ and $(\xi_1, \ldots, \xi_m)$ is a sequence of permutations (permutation braids) satisfying certain conditions (see [6] for more information). The braid $a$ can be recovered from its normal form $(k, (\xi_1, \ldots, \xi_m))$ as a product of the $k$th power of the half twist permutation braid $\Delta$ and permutation braids $\xi_1, \ldots, \xi_m$:

$$a = \Delta^k \xi_1 \ldots \xi_m.$$

In this section we describe an algorithm which, given a Garside normal form of an element $a$, tries to find a geodesic braid word representing $a$. (A geodesic braid word of a given braid is a braid word of minimum length representing this braid.) Since all information transmitted by Alice and Bob is in Garside normal forms, we need this algorithm for our attack.

Note that for permutation braids it is easy to find geodesic braid words. Therefore, to convert a given Garside normal form $(k, (\xi_1, \ldots, \xi_m))$ to a word, one can find geodesic braid words $w_\Delta, w_{\xi_1}, \ldots, w_{\xi_m}$ for $\Delta$ and $\xi_1, \ldots, \xi_m$, respectively, and compose a word

$$w = w_\Delta^k w_{\xi_1} \ldots w_{\xi_m}$$

which represents the same word as the given normal form. The length of the obtained word $w$ is

$$|k||w_\Delta| + |w_{\xi_1}| + \ldots + |w_{\xi_m}| = |k|\frac{n(n-1)}{2} + |w_{\xi_1}| + \ldots + |w_{\xi_m}|.$$

If $k \geq 0$ then the given braid $a$ is positive and the word $w$ is geodesic in the Cayley graph of $B_n$.

Before we proceed in the case $k < 0$, recall one property of the elemet $\Delta$ (see [2]). For any braid word $w = x_{i_1}^{\varepsilon_1} \ldots x_{i_k}^{\varepsilon_k}$, one has

$$\Delta^{-1} w \Delta = x_{n-i_1}^{\varepsilon_1} \ldots x_{n-i_k}^{\varepsilon_k}.$$

The result of conjugation of $w$ by $\Delta$ will be denoted by $w^{\Delta}$.

Consider now the case $k < 0$. Denote $-k$ by $p$. One can rewrite the normal form $\Delta^{-p}\xi_1\xi_2\dots\xi_m$ in the following way:

$$\Delta^{1-p}(\Delta^{-1}\xi_1)\Delta^{p-1}\cdot\Delta^{2-p}(\Delta^{-1}\xi_2)\Delta^{p-2}\cdot\Delta^{3-p}(\Delta^{-1}\xi_3)\Delta^{p-3}\cdot\dots \quad (1)$$

Depending on the values of $k$ and $m$ the obtained decomposition (1) will end up either with $\Delta^{m-p}(\Delta^{-1}\xi_m)\Delta^{p-m}$ when $p > m$ or with $\xi_m$ when $p \le m$.

Note that the expressions $\Delta^{-1}\xi_i$ in brackets are inverted permutation braids and the length of a geodesic for $\Delta^{-1}\xi_i$ is $|\Delta|-|\xi_i|$. Compute a geodesic braid word $w_i$ for each $\Delta^{-1}\xi_i$ in (1). Since $\Delta^2$ generates the center of $B_n$, the conjugation by $\Delta^{i-p}$ either does not change the word (when $i-p$ is even) or acts the same way as the conjugation by $\Delta$ does. We have mentioned above that the conjugation by $\Delta$ does not increase the length of the word. Finally, conjugate the obtained words $w_1,\dots,w_k$ by powers of $\Delta$ and denote the results by $w'_1,\dots,w'_k$. Clearly, the product

$$w' = w'_1\dots w'_k$$

defines the same element of $B_n$ as the given normal form does, but the word $w'$ is shorter than $w$:

$$|w'| = \begin{cases} |k|\frac{n(n-1)}{2} - \sum_{i=1}^{m}|w_{\xi_i}|, & \text{if } -k > m \\ |k|\frac{n(n-1)}{2} - \sum_{i=1}^{|k|}|w_{\xi_i}| + \sum_{i=|k|+1}^{m}|w_{\xi_i}| & \text{if } -k \le m \end{cases}$$

We performed a series of experiments in which we generated words of length $l$ in generators of $B_n$ and computed their Garside normal forms $(k,(\xi_1,\dots,\xi_m))$. In the experiments, $l$ was chosen to be sufficiently greater than $n$, e.g. $l > n^2$. In all cases $k$ was approximately $-\frac{3l}{4n}$ while $m$ was approximately $\frac{3l}{2n}$. Thus, almost in all cases the word $w$ is longer than $w'$.

## 3 Minimization of braids

Let $B_n$ be the group of braids on $n$ strands and let

$$\langle x_1,\dots,x_{n-1}\ ;\ [x_i,x_j]=1 \text{ (where } |i-j|>1),\ x_ix_{i+1}x_i = x_{i+1}x_ix_{i+1}\rangle$$

be its standard presentation. Let $w$ be a word in generators of $B_n$ and their inverses. The problem of computing a geodesic word for $w$ in $B_\infty$ was shown to be NP-complete in [12]. It is known however (see e.g. [10], [15]) that many NP-complete problems have polynomial time generic- or average-case solutions, or have good approximate solutions. In this section we present heuristic algorithms for approximating geodesics of braids and cyclic braids.

By Dehornoy's form of a braid we mean a braid word without any "handles", i.e. a completely reduced braid word in the sense of [4]. The procedure that computes Dehornoy's form for a given word chooses a specific ("permitted") handle inside of the word and removes it. This can introduce new handles but the main result about Dehornoy's forms states that any sequence of handle reductions eventually terminates. Of course, the result depends on how one chooses the handles at every step. Let us fix any particular strategy for selecting handles. For a word $w = w(x_1, \ldots, x_{n-1})$ we denote by $D(w)$ the corresponding Dehornoy's form (i.e., the result of handle reductions where handles are chosen by the fixed strategy).

The following algorithm tries to minimize the given braid word. It exploits the property of Dehornoy's form that for a "generic" braid word one has $|D(w)| < |w|$.

**Algorithm 1** *(*Minimization of braids*)*
SIGNATURE. $w' = Shorten(w)$.
INPUT. *A word* $w = w(x_1, \ldots, x_{n-1})$ *in generators of the braid group* $B_n$.
OUTPUT. *A word* $w'$ *such that* $|w'| \le |w|$ *and* $w' = w$ *in* $B_n$.
INITIALIZATION. *Put* $w_0 = w$ *and* $i = 0$.
COMPUTATIONS.

*A. Increment* $i$.
*B. Put* $w_i = D(w_{i-1})$.
*C. If* $|w_i| < |w_{i-1}|$ *then*
  *1) Put* $w_i = w_i^{\Delta}$.
  *2) Goto A.*
*D. If* $i$ *is even then output* $w_{i+1}^{\Delta}$.
*E. If* $i$ *is odd then output* $w_{i+1}$.

The following simple example illustrates why the idea with conjugation by $\Delta$ works. Consider the braid word $w = x_2^{-1}x_1x_2x_1$. This braid is in Dehornoy's form, but the geodesic for the corresponding braid is $x_1x_2$, hence $w$ is not geodesic. Now, the word $w^{\Delta} = x_1^{-1}x_2x_1x_2$ is not in Dehornoy's form. It contains one handle, removing of which results in the word $x_2x_1$ which is shorter than the initial word. If we call handles introduced by Dehornoy *left handles* and define *right handles* as subwords symmetric to left handles with respect to the direction of a braid, then the computation of Dehornoy's form of a word conjugated by $\Delta$ and conjugating the obtained result by $\Delta$ is essentially a process of removing right handles. We note that removing left handles might introduce right handles and vice versa, and the existence of forms without both left and right handles is questionable.

We would like to emphasize practical efficiency of Algorithm 1. We performed a series of experiments to test it; one of the experiments was the following sequence of steps:

1) generate a random freely reduced braid word $w \in B_{100}$ of length 4000;
2) compute its Garside normal form $\xi$;
3) transform $\xi$ back into a word $w'$ as described in Section 2;
4) apply Algorithm 1 to $w'$. Denote the obtained word by $w''$.

In all experiments the length of the obtained words $w''$ varied in the interval $[2500, 3100]$. Thus, the result was shorter than the input. It is possible that for a longer initial word $w$ we would not get the same results, but the length 4000 is more than is used in the Ko-Lee protocol anyway.

The next algorithm is a variation of Algorithm 1 for cyclic braid words.

**Algorithm 2** *(*Minimization of cyclic braids*)*
SIGNATURE. $w' = CycShorten(w)$.
INPUT. *A word* $w = w(x_1, \ldots, x_{n-1})$ *in generators of the braid group* $B_n$.
OUTPUT. *A word* $w'$ *such that* $|w'| \le |w|$ *and* $w' = w$ *in* $B_n$.
INITIALIZATION. *Put* $w_0 = w$ *and* $i = 0$.
COMPUTATIONS.

A. *Increment* $i$.
B. *Put* $w_i = w_{i-1}$.
C. *If* $|D(w_i)| < |w_i|$ *then put* $w_i = D(w_i)$.
D. *If* $w_i = w_i' \circ w_i''$ *(where* $|w_i'| - |w_i''| \le 1$*) and* $|D(w_i'' w_i')| < |w_i|$ *then put* $w_i = D(w_i'' w_i')$.
E. *If* $|w_i| < |w_{i-1}|$ *then Goto A.*
F. *Output* $w_i$.

## 4 The attack

In this section we describe a heuristic algorithm for solving the decomposition problem for a pair of words $w_1$ and $w_2$ as in the Ko-Lee protocol.

First we describe two auxiliary algorithms. The first algorithm decomposes a given word $w$ into a product $usv$, where $u, v \in L_n$, trying to to make $s$ as short as possible.

**Algorithm 3** *(*Decomposition 1*)*
INPUT. *A braid word* $w = w(x_1, \ldots, x_{n-1})$.
OUTPUT. *A triple of words* $(u, s, v)$ *such that* $u, v \in L_n$, $|s| \le |w|$, *and* $usv = w$ *in* $B_n$.
INITIALIZATION. *Put* $u_0 = v_0 = \varepsilon$ *and* $s_0 = w$ *and* $i = 0$.
COMPUTATIONS.

A. *Increment* $i$.
B. *Put* $u_i = u_{i-1}$, $s_i = s_{i-1}$, *and* $v_i = v_{i-1}$.
C. *For each* $j = 1, \ldots, n-1$ *check:*
   1) *If* $|D(x_j s_i)| < |s_i|$ *then*
      – *put* $u_i = u_i x_j^{-1}$;
      – *put* $s_i = D(x_j s_i)$;
      – *goto A.*
   2) *If* $|D(x_j^{-1} s_i)| < |s_i|$ *then*
      – *put* $u_i = u_i x_j$;
      – *put* $s_i = D(x_j^{-1} s_i)$;

   – *goto A.*
   3) *If* $|D(s_i x_j)| < |s_i|$ *then*
      – *put* $v_i = x_j^{-1} v_i$*;*
      – *put* $s_i = D(s_i x_j)$*;*
      – *goto A.*
   4) *If* $|D(s_i x_j^{-1})| < |s_i|$ *then*
      – *put* $v_i = x_j v_i$*;*
      – *put* $s_i = D(s_i x_j^{-1})$*;*
      – *goto A.*

D. *Output the triple* $(u_i, s_i, v_i)$*.*

The next algorithm decomposes two given braid words $w_1$ and $w_2$ into products $us_1v$ and $us_2v$, respectively, where $u, v \in R_n$, trying to make $s_1$ and $s_2$ as short as possible.

**Algorithm 4** *(*Decomposition 2*)*
INPUT. *Braid words* $w_1$ *and* $w_2$*.*
OUTPUT. *A quadruple of words* $(u, s, t, v)$ *such that* $u, v \in R_n$*,* $|s| \leq |w_1|$*,* $|t| \leq |w_2|$*,* $utv = w_2$ *in* $B_n$*, and* $usv = w_1$ *in* $B_n$*.*
INITIALIZATION. *Put* $u_0 = v_0 = \varepsilon$*,* $s_0 = w_1$*,* $t_0 = w_2$*, and* $i = 0$*.*
COMPUTATIONS.

A. *Increment* $i$*.*
B. *Put* $u_i = u_{i-1}$*,* $s_i = s_{i-1}$*,* $t_i = t_{i-1}$*, and* $v_i = v_{i-1}$*.*
C. *For each* $j = n+1, \ldots, 2n-1$ *check:*
   1) *If* $|D(x_j s_i)| < |s_i|$ *and* $|D(x_j t_i)| < |t_i|$ *then*
      – *put* $u_i = u_i x_j^{-1}$*;*
      – *put* $s_i = D(x_j s_i)$*;*
      – *put* $t_i = D(x_j t_i)$*;*
      – *goto A.*
   2) *If* $|D(x_j^{-1} s_i)| < |s_i|$ *and* $|D(x_j^{-1} t_i)| < |t_i|$ *then*
      – *put* $u_i = u_i x_j$*;*
      – *put* $s_i = D(x_j^{-1} s_i)$*;*
      – *put* $t_i = D(x_j^{-1} t_i)$*;*
      – *goto A.*
   3) *If* $|D(s_i x_j)| < |s_i|$ *and* $|D(t_i x_j)| < |t_i|$ *then*
      – *put* $v_i = x_j^{-1} v_i$*;*
      – *put* $s_i = D(s_i x_j)$*;*
      – *put* $t_i = D(t_i x_j)$*;*
      – *goto A.*
   4) *If* $|D(s_i x_j^{-1})| < |s_i|$ *and* $|D(t_i x_j^{-1})| < |t_i|$ *then*
      – *put* $v_i = x_j v_i$*;*
      – *put* $s_i = D(s_i x_j^{-1})$*;*
      – *put* $t_i = D(t_i x_j^{-1})$*;*
      – *goto A.*

D. *Output* $(u_i, s_i, t_i, v_i)$*.*

Now let $w_1$, $w_2$ be braid words in $B_{2n}$ for which there exist words $a_1$, $a_2$ in $L_n$ such that $w_1 = a_1 w_2 a_2$ in $B_{2n}$. Denote by $\overline{S}_{(w_1,w_2)}$ the solution set for the decomposition problem for the pair $(w_1, w_2)$, i.e.,

$$\overline{S}_{(w_1,w_2)} = \{(q_1, q_2) \in L_n \times L_n \mid q_1 w_1 q_2 = w_2\} \text{ in } B_{2n}.$$

Let the triple $(u_i, s_i, v_i)$ be the result of applying Algorithm 3 to the word $w_i$ (where $i = 1, 2$) and $(u, s, t, v)$ the result of applying Algorithm 4 to the pair $(s_1, s_2)$. We will say that the pair $(s, t)$ is a *simplified pair* of $(w_1, w_2)$.

**Lemma 1.** *For a simplified pair $(s,t)$ of $(w_1, w_2)$ the following holds:*

$$\overline{S}_{(w_1,w_2)} = \{(u_2 q_1 u_1^{-1}, v_1^{-1} q_2 v_2) \mid (q_1, q_2) \in \overline{S}_{(s,t)}\}.$$

*Proof.* We have $s = u^{-1} u_1^{-1} w_1 v_1^{-1} v^{-1}$ and $t = u^{-1} u_2^{-1} w_2 v_2^{-1} v^{-1}$, where $u_1$, $u_2$, $v_1$, $v_2 \in L_n$ and $u, v \in R_n$. By the definition of $\overline{S}_{(s,t)}$, one has $(q_1, q_2) \in \overline{S}_{(s,t)}$ if and only if $q_1 s q_2 =_{B_{2n}} t$ in $B_{2n}$, or if and only if

$$q_1 u^{-1} u_1^{-1} w_1 v_1^{-1} v^{-1} q_2 = u^{-1} u_2^{-1} w_2 v_2^{-1} v^{-1}.$$

Since $q_1, q_2 \in L_n$ and $u, v \in R_n$, the last equality holds if and only if

$$q_1 u_1^{-1} w_1 v_1^{-1} q_2 = u_2^{-1} w_2 v_2^{-1},$$

or if and only if

$$u_2^{-1} q_1 u_1^{-1} w_1 v_1^{-1} q_2 v_2^{-1} = w_2,$$

or if and only if $(u_2^{-1} q_1 u_1^{-1}, v_1^{-1} q_2 v_2^{-1}) \in \overline{S}_{(w_1,w_2)}$.

Now represent the set of possible solutions $S = S_{(w_1,w_2)}$ of the decomposition problem for $(w_1, w_2)$ as a directed graph with the vertex set

$$V = L_n \times L_n$$

and the edge set $E$ containing edges of the following two types:

- $(q_1, q_2) \to (q_3, q_4)$ if $q_1 = q_3$ and $q_4 = \overline{q_2' \circ x_j^{\varepsilon} \circ q_2''}$ (where $q_2 = q_2' \circ q_2''$, $j \in \{1, \ldots, n-1\}$, and $\varepsilon = \pm 1$);
- $(q_1, q_2) \to (q_3, q_4)$ if $q_2 = q_4$ and $q_3 = \overline{s_1' \circ x_j^{\varepsilon} \circ q_1''}$ (where $q_1 = q_1' \circ q_1''$, $j \in \{1, \ldots, n-1\}$, and $\varepsilon = \pm 1$).

Define a function $\omega : S \to \mathbb{N}$ as follows:

$$(q_1, q_2) \overset{\omega}{\mapsto} |CycShorten(q_1 w_1 q_2 w_2^{-1})|.$$

(cf. our Algorithm 2).

Let $w_1$ be the base word in the Ko-Lee protocol and $w_2$ a word representing the normal form $\xi = a^{-1} w_1 a$ transmitted by Alice. In this notation we can formulate the problem of finding Alice's keys as a search problem in $S_{(w_1,w_2)}$. Clearly

$$\overline{S}_{(w_1,w_2)} = \{(q_1, q_2) \in S_{(w_1,w_2)} \mid \omega(q_1, q_2) = 0\}$$

and, therefore, the problem is to find a pair of braid words $(q_1, q_2)$ such that $\omega(q_1, q_2) = 0$.

We want to stress that in some cases the set $S_{(w_1,w_2)}$ can be reduced. For example, let $m$ be the smallest index of a generator in both words $w_1$ and $w_2$. Then we can impose a restriction $j \in \{m, \ldots, n-1\}$ and solve the search problem in a smaller space. This situation where $m > 1$ was very often the case in our computations.

The next algorithm is an attack on Alice's private key. The input of the algorithm is the base word $w$ and the Garside normal form $\xi$ of the braid word $a^{-1}wa$ transmitted by Alice. The algorithm finds a pair of words $(\alpha, \beta)$ in generators of $L_n$ such that $\alpha w \beta = u$ in $B_{2n}$, where $u$ is a braid word with the Garside normal form $\xi$. At step A, the algorithm transforms $\xi$ into a word $\overline{w}$. At steps B and C, it computes a simplified pair $(s, t)$ for $(w, \overline{w})$. At steps D-F, the algorithm performs a heuristic search in the key space $S_{(s,t)}$. The search starts at the point $(\varepsilon, \varepsilon)$, where $\varepsilon$ is the empty word. In each iteration we choose an unchecked vertex with the minimum $\omega$ value and construct its neighborhood. The search stops when the point with zero $\omega$ value is found.

**Algorithm 5** *(*Attack on Alice's key*)*
INPUT. *A braid word $w$ and a Garside normal form $\xi$ corresponding to a braid $u$ for which there exists $a \in L_n$ such that $a^{-1}wa = u$ in $B_{2n}$.*
OUTPUT. *A pair of words $\alpha, \beta \in L_n$ such that $\alpha w \beta = u$ in $B_{2n}$.*
INITIALIZATION. *Put $u_0 = v_0 = \varepsilon$, $s_0 = w_1$, and $i = 0$.*
COMPUTATIONS.

*A. Convert a normal form $\xi$ to a word $\overline{w}$.*
*B. Apply Algorithm 1 to words $w$ and $\overline{w}$.*
*B. Let $(u_1, s_1, v_1)$ be the result of applying Algorithm 3 to the word $w$ and $(u_2, s_2, v_2)$ the result of applying Algorithm 3 to the word $\overline{w}$.*
*C. Let $(u, s, t, v)$ be the result of applying Algorithm 4 to the pair of words $(s_1, s_2)$.*
*D. Let $Q = \{(\varepsilon, \varepsilon)\} \subset S_{(s,t)}$.*
*E. Choose an unchecked pair $(q_1, q_2)$ from the set $Q$ with the minimum $\omega$ value.*
*F. For each edge $(q_1, q_2) \to (q_1', q_2') \in S_{(s,st)}$ add a pair $(q_1', q_2')$ to $Q$. If $\omega$-value of some new pair $(q_1', q_2')$ is 0, then output $(u_2 q_1' u_1^{-1}, v_1^{-1} q_2' v_2)$. Otherwise goto E.*

## 5 Experiments and conclusions

We have performed numerous experiments of two types. Experiments of the first type tested security of the original Ko-Lee protocol, whereas experiments of the second type tested security of a protocol similar to that of Ko-Lee, but based on the decomposition problem.

An experiment of the first type is the following sequence of steps:

1) Fix the braid group $B_{100}$.

2) Randomly generate a base word $w$ as a freely reduced word of length 2000 in the generators of $B_{100}$.
3) Randomly generate a word $a = a(x_1, \ldots, x_{49})$ as a freely reduced word of length 1000 in the generators of $B_{100}$.
4) Compute Garside normal forms $\rho_1$ and $\rho_2$ of $w$ and $a^{-1}wa$, respectively.
5) Transform normal forms back into words $w_1$ and $w_2$ (see Section 2).
6) Apply Algorithm 1 to words $w_1$ and $w_2$.
7) Finally, apply Algorithm 5 to the pair $(w_1, w_2)$.

We say that an experiment is successful if all of the above steps were performed in a reasonable amount of time (we allowed 150 minutes); otherwise we stopped the program. We performed 2466 such experiments and had success in 2378 of them, which means the success rate was 96.43%.

Experiments of the second type have different steps 3) and 4). They are as follows:

3') Randomly generate two words $a_1 = a_1(x_1, \ldots, x_{49})$ and $a_2 = a_2(x_1, \ldots, x_{49})$ as freely reduced words of length 1000.
4') Compute Garside normal forms $\rho_1$ and $\rho_2$ of $w$ and $a_1 w a_2$, respectively.

We performed 827 experiments of the second type and had success in 794 of them. This gives the success rate of 96.00%, so that the difference in the success rates of two types of experiments is statistically insignificant.

The conclusion therefore is that we were able to break the Ko-Lee protocol in about 150 minutes with over 95.00% success rate for typical parameters.

Finally, we note that there are several ways to improve the success rate. The easiest way is simply to increase the time allocated to experiments. Also, one can improve the algorithms themselves, in particular, Algorithm 1. With a better minimization algorithm the attack is likely to be more efficient. One can also somewhat narrow down the search space, etc.

## 6 Suggestions on improving the key exchange protocol

In this section, we briefly sketch a couple of ideas that may help to enhance security of the Ko-Lee protocol and, in particular, make it less vulnerable to the attack described in the previous sections.

1) Either increase the length of the private keys and the base or decrease the rank of the group. With the parameters suggested in [11], transmitted braids are sort of "sparse" which allows the adversary to simplify the initial braids substantially. The lengths of the transmitted braids should be at least on the order of $n^2$ (where $n$ is the rank of the braid group) to prevent fast reconstruction of a short braid word from its normal form.
   We note however that increasing the key length is a trade-off between security and efficiency. By comparison, the current key size used in the RSA cryptosystem is 512 bits, whereas to store a braid word of length $l$ from the group $B_n$, $l\lfloor \log_2(2n) \rfloor$ bits are required. This number is approximately 8000 for $l = 1000$ and $n = 100$.

2) Choosing a "base" word $w$ requires special attention. It might be a good idea to generate $w$ as a geodesic in the Cayley graph of $B_{2n}$ starting and terminating with the generator $x_n$ or its inverse (the one which does not belong to $L_n$ or $R_n$) such that any other geodesic representing $w$ starts and terminates with $x_n^{\pm 1}$. Observe that for such $w$ Algorithm 3 stops with the result $(\varepsilon, w, \varepsilon)$. Also, for such $w$ and an arbitrary braid word $w'$, Algorithm 4 applied to $(w, w')$ stops with the result $(\varepsilon, w, w', \varepsilon)$.
3) Choose different commuting subgroups instead of $L_n$ and $R_n$. This looks like the most promising suggestion at the moment; we refer to [14] for more details.

## References


1. I. Anshel, M. Anshel, D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Res. Lett. **6** (1999), 287–291.
2. J. S. Birman, *Braids, links and mapping class groups*, Ann. Math. Studies **82**, Princeton Univ. Press, 1974.
3. J. H. Cheon and B. Jun, *A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem*, Crypto 2003, Lecture Notes in Comput. Sci. **2729** (2003), 212–225 .
4. P. Dehornoy, *A fast method for comparing braids*, Adv. Math. **125** (1997), 200–235.
5. P. Dehornoy, *Braid-based cryptography*, Contemp. Math., Amer. Math. Soc. **360** (2004), 5–33.
6. D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, W. P. Thurston, *Word processing in groups.* Jones and Bartlett Publishers, Boston, MA, 1992.
7. D. Garber, S. Kaplan, M. Teicher, B. Tsaban, U. Vishne, *Probabilistic solutions of equations in the braid group*, preprint. http://arxiv.org/abs/math.GR/0404076
8. D. Hofheinz and R. Steinwandt, *A practical attack on some braid group based cryptographic primitives*, in Public Key Cryptography, 6th International Workshop on Practice and Theory in Public Key Cryptography, PKC 2003 Proceedings, Y.G. Desmedt, ed., Lecture Notes in Computer Science **2567**, pp. 187–198, Springer, 2002.
9. J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Workshop SECI02 Securitè de la Communication sur Intenet, September 2002, Tunis, Tunisia. http://www.storagetek.com/hughes/
10. I. Kapovich, A. Myasnikov, P. Schupp and V. Shpilrain, *Average-case complexity for the word and membership problems in group theory*, Advances in Math. **190** (2005), 343–359.
11. K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, 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.
12. M. S. Paterson, A. A. Razborov, *The set of minimal braids is co-NP-complete*, J. Algorithms **12** (1991), 393–408.

13. V. Shpilrain and A. Ushakov, *The conjugacy search problem in public key cryptography: unnecessary and insufficient*, Applicable Algebra in Engineering, Communication and Computing, to appear.
http://eprint.iacr.org/2004/321/
14. V. Shpilrain and G. Zapata, *Combinatorial group theory and public key cryptography*, Applicable Algebra in Engineering, Communication and Computing, to appear.
http://eprint.iacr.org/2004/242
15. J. Wang, *Average-case computational complexity theory,* Complexity Theory Retrospective, II. Springer-Verlag, New York, 1997, 295–334.