BRAID GROUP CRYPTOGRAPHY

PRELIMINARY DRAFT

DAVID GARBER

ABSTRACT. In the last decade, a number of public key cryptosystems based on combinatorial group theoretic problems in braid groups have been proposed. Our tutorial is aimed at presenting these cryptosystems and some known attacks on them.

We start with some basic facts on braid groups and on the Garside normal form of its elements. We then present some known algorithms for solving the word problem in the braid group. After that, we present the major public-key cryptosystems based on the braid group. We then discuss some of the known attacks on these cryptosystems. We finish with a discussion of future directions.

CONTENTS

1. The braid group

1.1. **Basic definitions.**

The braid groups were introduced by Artin [3]. There are several definitions for these groups (see [71]), and we need two of them for our purposes.

1.1.1. *Algebraic presentation.*

Definition 1.1. *For $n \geq 2$, the braid group B_n is defined by the presentation*

$$\left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{c} \sigma_i\sigma_j = \sigma_j\sigma_i \textit{ for } |i-j| \geq 2 \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \textit{ for } |i-j| = 1 \end{array} \right\rangle . \tag{1.1}$$

This presentation is called the *Artin presentation* and the generators are called *Artin's generators.*

An element of B_n will be called an *n-braid*. For each n, the identity mapping on $\{\sigma_1, \ldots, \sigma_{n-1}\}$ induces an embedding of B_n into B_{n+1}, so that we can consider an n-braid as a particular $(n+1)$-braid. Using this, one can define the limit group B_∞.

Note that B_2 is an infinite cyclic group, and hence it is isomorphic to the group $\mathbb{Z}$ of integers. For $n \geq 3$, the group B_n is not commutative and its center is an infinite cyclic subgroup.

When a group is specified using a presentation, each element of the group is an equivalence class of words with respect to the congruence generated by the relations of the presentation. Hence, every n-braid is an equivalence class of n-braid words under the congruence $\equiv$ generated by the relations in (1.1).

1.1.2. *Geometric interpretation.* The elements of B_n can be interpreted as geometric n strand braids. One can associate with every braid the planar diagram obtained by concatenating the elementary diagrams of Figure 1 corresponding to the successive letters.

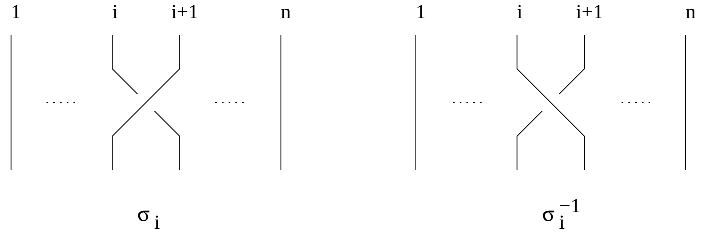

Figure 1. The geometric Artin generators

A braid diagram can be seen as induced by a three-dimensional figure consisting on n disjoint curves connecting the points $(1,0,0),\ldots,(n,0,0)$ to the points $(1,0,1),\ldots,(n,0,1)$ in $\mathbb{R}^3$. Then, the relations in (1.1) correspond to ambient isotopy, that is: to continuously moving the curves without moving their ends and without allowing them to intersect. It is easy to check that each relation in (1.1) corresponds to such an isotopy (see Figures 2 and 3); the converse implication, i.e. the fact that the projections of isotopic 3D figures can always be encoded in words connected by (1.1) was proved by Artin in [3]. Hence, the word problem in the braid group for the presentation (1.1) is also the *braid isotopy problem*, and thus it is closely related to the much more difficult knot isotopy problem.

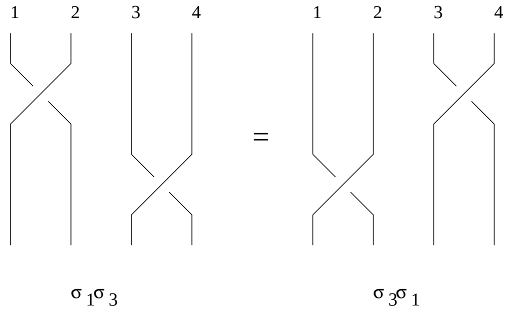

Figure 2. The commutative relation for geometric Artin generators

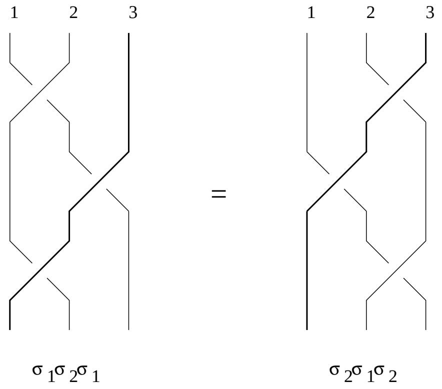

Figure 3. The triple relation for geometric Artin generators

1.2. **Birman-Ko-Lee presentation.** Like Artin's generators, the generators of Birman-Ko-Lee [11] are braids in which exactly one pair of strands crosses. The difference is that the new set of generators includes arbitrary transpositions of strands (i, j) instead of adjacent transpositions $(i, i + 1)$ in the Artin's generators. For each t, s with $1 \leq s < t \leq n$, define the following element of B_n:

$$a_{ts} = (\sigma_{t-1}\sigma_{t-2} \cdots \sigma_{s+1})\sigma_s(\sigma_{s+1}^{-1} \cdots \sigma_{t-2}^{-1}\sigma_{t-1}^{-1})$$

See figure 4 for an example (note that the braid a_{ts} is an elementary interchange of the tth and sth strands, with all other strands held fixed, and with the convention that the strands being interchanged pass in front of all intervening strands). Such an element is called a *band generator.*

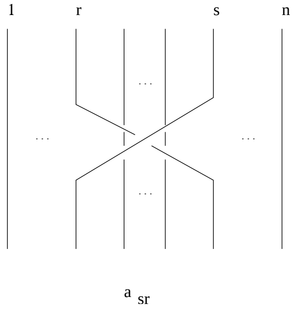

Figure 4. The band generator

Note that the usual Artin generators are the generators $a_{t+1,t}$.

This set of generators satisfies the following relations (see [11, Prop. 2.1] for a proof):

- $a_{ts}a_{rq} = a_{rq}a_{ts}$ if $[s, t] \cap [q, r] = \emptyset$.

- $a_{ts}a_{sr} = a_{tr}a_{ts} = a_{sr}a_{tr}$ for $1 \leq r < s < t \leq n$.

For a geometric interpretation of the second relation, see Figure 5.

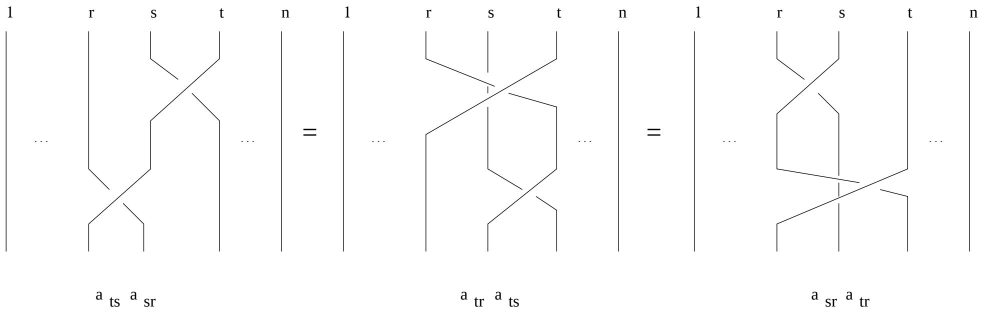

FIGURE 5. The second relation of the Birman-Ko-Lee presentation

2. Normal forms of elements in the braid group

A *normal form* of an element in a group is a unique presentation to each element in the group.

Having a normal form for elements in the group is very useful, since it lets us compare two elements, so it gives a solution for the word problem:

Problem 2.1. *Given a braid w, does $w \equiv \varepsilon$ hold, i.e., does w represent the unit braid ε?*

Since B_n is a group, the above problem is equivalent to the following problem:

Problem 2.2. *Given two braids w, w', does $w \equiv w'$ hold, i.e., do w and w' represent the same braid?*

Indeed, $w \equiv w$ is equivalent to $w^{-1}w' \equiv \varepsilon$, where w^{-1} is the word obtained from w by reversing the order of the letters and exchanging σ_i and σ_i^{-1} everywhere.

Also, the normal form gives a canonical representative of each equivalence class.

We present here some known normal forms of elements in the braid group. For more normal forms, see Bressaud [13], Dehornoy [22] and Dynnikov-Wiest [29].

2.1. **Garside normal form.** We start by defining a *positive braid* which is a braid which can be written as a product of positive powers of Artin generators. We denote the set of positive braids by B_n^+. This set has a structure of a monoid under the operation of braid concatenation.

An important example of a positive braid, which has a central role in the Garside normal form is the *fundamental braid* $\Delta_n \in B_n$:

$$\Delta_n = (\sigma_1 \cdots \sigma_{n-1})(\sigma_1 \cdots \sigma_{n-2}) \cdots \sigma_1$$

Geometrically, Δ_n is the braid on n strands, where any two strands cross positively exactly once (see Figure 6).

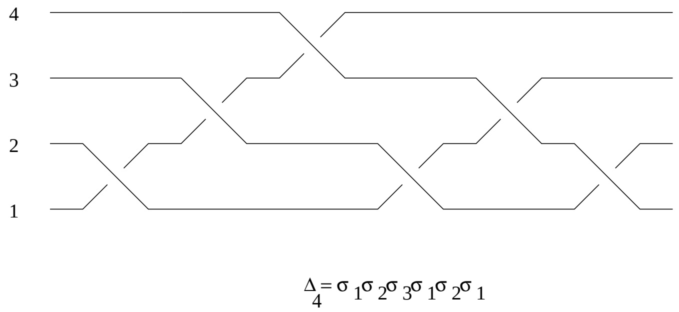
4
3
2
1
$\Delta_4 = \sigma_1 \sigma_2 \sigma_3 \sigma_1 \sigma_2 \sigma_1$

FIGURE 6. The fundamental braid Δ_4

The fundamental braid has several important properties:

(1) For any generator σ_i, we can write $\Delta_n = \sigma_i A = B\sigma_i$ where A, B are positive braids.
(2) For any generator σ_i, the following holds: $\sigma_i \Delta_n = \Delta_n \sigma_{n-i}$.
(3) Δ_n^2 is the generator of the center of B_n.

Now, we introduce *permutation braids*. One can define a partial order on the elements of B_n: for $A, B \in B_n$, we write $A \preceq B$ where $B = AC$ for some C in B_n^+. Its simple properties are:

(1) $B \in B_n^+ \Leftrightarrow \varepsilon \preceq B$
(2) $A \preceq B \Leftrightarrow B^{-1} \preceq A^{-1}$.

$P \in B_n$ is a *permutation braid* (or a *simple braid*) if it satisfies: $\varepsilon \preceq P \preceq \Delta_n$. Its name comes from the fact that there is a bijection between the set of permutation braids in B_n and the symmetric group S_n (there is a natural surjective map from B_n to S_n defined by sending i to the ending place of the strand which starts at position i, and if we restrict ourselves to permutation braids, this map is a bijection). Hence, we have $n!$ permutation braids.

Geometrically, a permutation braid is a braid on n strands, where any two strands cross positively *at most* once.

Given a permutation braid P, one can define a *starting set* $S(P)$ and a *finishing set* $F(P)$ as follows:

$$S(P) = \{i | P = \sigma_i P' \text{ for some } P' \in B_n^+\}$$
$$F(P) = \{i | P = P' \sigma_i \text{ for some } P' \in B_n^+\}$$

The starting set is the indices of the generators which can start a presentation of P. The finishing set is defined similarly. For example, $S(\Delta_n) = F(\Delta_n) = \{1, \ldots, n-1\}$.

A *left-weighted decomposition* of a positive braid $A \in B_n^+$ into a sequence of permutation braids is:

$$A = P_1 P_2 \cdots P_k$$

where P_i are permutation braids, and $S(P_{i+1}) \subset F(P_i)$, i.e. any addition of a generator from P_{i+1} to P_i, will convert P_i into a braid which is not a permutation braid.

The following theorem introduces the *Garside normal form* (or *left canonical form* or *greedy normal form*) and states its uniqueness:

Theorem 2.3. *For every braid $w \in B_n$, there is a unique presentation given by:*

$$w = \Delta_n^r P_1 P_2 \cdots P_k$$

where $r \in \mathbb{Z}$ is maximal, P_i are permutation braids, $P_k \neq \varepsilon$ and $P_1 P_2 \cdots P_k$ is a left-weighted decomposition.

For converting a given braid w into its Garside normal form we have to follow the following steps:

(1) For any negative power of a generator, replace σ_i^{-1} by $\Delta_n^{-1} B_i$ where B_i is a permutation braid.

(2) Move any appearance of Δ_n to the left using the relation: $\sigma_{n-i}\Delta_n = \Delta_n \sigma_i$. So we get: $w = \Delta_n^{r'} A$ where A is a positive braid.

(3) Write A as a left-weighted decomposition of permutation braids. The idea how to do this is as follows: Take A, and break it into permutation braids (i.e. we take the longest possible sequences of generators which are still permutation braids). Then we get: $A = Q_1 Q_2 \cdots Q_j$ where each Q_i is a permutation braid. For each i, we compute the finishing set $F(Q_i)$ and the starting set $S(Q_{i+1})$. In case that the starting set is not contained in the finishing set, we take a generator $\sigma \in S(Q_{i+1}) \backslash F(Q_i)$, and using the relations of the braid group we move it from Q_{i+1} to Q_i. Then, we get the decomposition $A = Q_1 Q_2 \cdots Q_i' Q_{i+1}' \cdots Q_j$. We continue this process till we have $S(Q_{i+1}) \subseteq F(Q_i)$ for every i, and then we have a left-weighted decomposition as needed. For more details, see [30].

Example 2.4. *Let us present the braid $w = \sigma_1\sigma_3^{-1}\sigma_2 \in B_4$ in Garside normal form. First, we should replace σ_3^{-1} by: $\Delta_4^{-1}\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2$, so we get:*

$$w = \sigma_1 \cdot \Delta_4^{-1}\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2 \cdot \sigma_2$$

Now, moving Δ_4 to the left yields:

$$w = \Delta_4^{-1} \cdot \sigma_3\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_2$$

Decomposing the positive part into a left-weighted decomposition, we get:

$$w = \Delta^{-1} \cdot \sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 \cdot \sigma_1\sigma_2$$

The complexity of transforming a word into a canonical form with respect to the Artin presentation is $O(|W|^2 n \log n)$ where $|W|$ is the length of the word in B_n [31, Section 9.5].

In a similar way, one can define a *right normal form.* A *right-weighted decomposition* of a positive braid $A \in B_n^+$ into a sequence of permutation braids is:

$$A = P_k \cdots P_2 P_1$$

where P_i are permutation braids, and $F(P_{i+1}) \subset S(P_i)$, i.e. any addition of a generator from P_{i+1} to P_i, will convert P_i into a braid which is not a permutation braid.

Now, one has the following theorem about the *right normal form* and its uniqueness:

Theorem 2.5. *For every braid $w \in B_n$, there is a unique presentation given by:*

$$w = P_k \cdots P_2 P_1 \Delta_n^r$$

where $r \in \mathbb{Z}$, P_i are permutation braids, and $P_k \cdots P_2 P_1$ is a right-weighted decomposition.

For converting a given braid w into its right normal form we have to follow three steps, similar to those of the Garside normal form: We first replace σ_i^{-1} by $B_i \Delta_n^{-1}$. Then, we move any appearance of Δ_n to the right side. Then, we get: $w = A\Delta_n^{r'}$ where A is a positive braid. The last step is to write A as a right-weighted decomposition of permutation braids.

Now we define the *infimum* and the *supremum* of a braid w: For $w \in B_n$, set $\inf(w) = \max\{r : \Delta^r \preceq w\}$ and $\sup(w) = \min\{s : w \preceq \Delta^s\}$.

One can easily see that if $w = \Delta_n^m P_1 P_2 \cdots P_k$ is the Garside normal form of w, then: $\inf(w) = m, \sup(w) = m + k$.

The *canonical length of* w (or *complexity of* w), denoted by $\operatorname{len}(w)$, is given by $\operatorname{len}(w) = \sup(w) - \inf(w)$. Hence, if w is given in its normal form, the canonical length is the number of permutation braids in the form.

2.2. **Birman-Ko-Lee canonical form.** Based on the presentation of Birman, Ko and Lee [11], they give a new canonical form for elements in the braid group.

They define a new fundamental word:

$$\delta_n = a_{n,n-1}a_{n-1,n-2}\cdots a_{2,1} = \sigma_{n-1}\sigma_{n-2}\cdots\sigma_1$$

See Figure 7 for an example for $n = 4$.

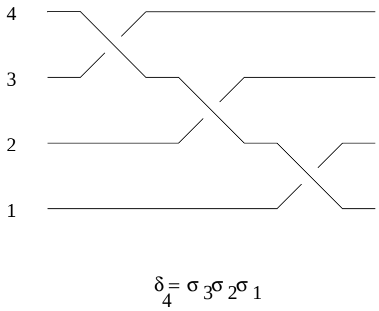

Figure 7. The fundamental braid δ_4

One can easily see the connection between the new fundamental word and Garside's fundamental word Δ_n:

$$\Delta_n^2 = \delta_n^n$$

The new fundamental word δ_n has important properties, similar to Δ_n:

(1) For any generator a_{sr}, we can write $\delta_n = a_{sr}A = Ba_{sr}$ where A, B are positive braids (with respect to the Birman-Ko-Lee generators)

(2) For any generator a_{sr}, the following holds: $a_{sr}\delta_n = \delta_n a_{s+1,r+1}$.

Similar to Garside's normal form of braids, each element of B_n has the following unique form in terms of the band generators:

$$w = \delta_n^j A_1 A_2 \cdots A_k,$$

where $A = A_1A_2\cdots A_k$ is positive, j is maximal and k is minimal for all such representations, also the A_i's are positive braids which are determined uniquely by their associated permutations (see [11, Lemma 3.1]. We will refer to Garside's braids P_i as *permutation braids*, and to the Birman-Ko-Lee braids A_i as *canonical factors*.

Note that there are $C_n = \frac{(2n)!}{n!(n+1)!}$ (the nth Catalan number) different canonical factors for the band-generator presentation [11, Corollary 3.5], whence there are $n!$ different permutation braids for the Artin presentation. Since C_n is much smaller than $n!$, it is sometimes computationally easier to work with the band-generator presentation than the Artin presentation (see also Section 8.3.2).

As in Garside's normal form, there is an algorithmic way to convert any braid to this canonical form: we first convert any negative power of a generator to $\delta_n^{-1}A$ where A is positive. Then, we move all the δ_n to the left, and finally we organize the positive word in a left-weighted decomposition of canonical factors.

The complexity of transforming a word into a canonical form with respect to the Birman-Ko-Lee presentation is $O(|W|^2 n)$, where $|W|$ is the length of the word in B_n [11].

As in Garside normal form, one can define infimum, supremum and canonical length for the canonical form of Birman-Ko-Lee presentation.

3. Algorithms for solving the word problem in braid group

Using ε for the empty word, the *word problem* is the following algorithmic problem:

> Given one braid word w, does $w \equiv \varepsilon$ hold, i.e., does w represent the unit braid ε?

In this section, we will concentrate on some solutions for the word problem in the braid group.

3.1. **Dehornoy's handles reduction.**

The process of *handle reduction* was introduced by Dehornoy [19], and one can see it as an extension of the free reduction process for free groups. Free reduction consists of iteratively deleting all patterns of the form xx^{-1} or $x^{-1}x$: starting with an arbitrary word w of length ℓ, and no matter on how the reductions are performed, one finishes in at most $\ell/2$ steps with a unique reduced word, i.e., a word that contains no xx^{-1} or $x^{-1}x$.

Free reduction is possible for any group presentation, and in particular for B_n, but it does not solve the word problem: there exist words that represent $\varepsilon \in B_n$, but do not freely reduce to the empty word. For example, the word $\sigma_1\sigma_2\sigma_1\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}$ represents the empty word, but free reductions can not reduce it any more.

The handle reduction process generalizes free reduction and involves not only patterns of the form xx^{-1} or $x^{-1}x$, but also more general patterns of the form $\sigma_i \cdots \sigma_i^{-1}$ or $\sigma_i^{-1} \cdots \sigma_i$:

Definition 3.1. A σ_i-handle *is a braid word of the form*

$$w = \sigma_i^e w_0 \sigma_{i+1}^d w_1 \sigma_{i+1}^d \cdots \sigma_{i+1}^d w_m \sigma_i^{-e},$$

with $e, d = \pm 1, m \geq 0$, *and* $w_0, \ldots, w_m$ *containing no* $\sigma_j^{\pm 1}$ *with* $j \leq i+1$.

The reduction of w *is defined as follows:*

$$w' = w_0 \sigma_{i+1}^{-e} \sigma_i^d \sigma_{i+1}^e w_1 \sigma_{i+1}^{-e} \sigma_i^d \sigma_{i+1}^e \cdots \sigma_{i+1}^{-e} \sigma_i^d \sigma_{i+1}^e w_m,$$

i.e., we delete the initial and final letters $\sigma_i^{\pm 1}$, *and we replace each letter* $\sigma_{i+1}^{\pm 1}$ *with* $\sigma_{i+1}^{-e} \sigma_i^{\pm 1} \sigma_{i+1}^e$ *(see Figure 8, taken from* [20]*).*

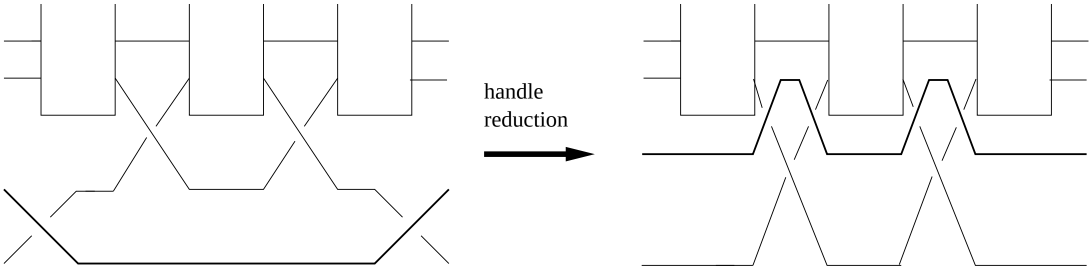

FIGURE 8. An example for a handle reduction (for σ_1)

Note that a braid of the form $\sigma_i \sigma_i^{-1}$ or $\sigma_i^{-1} \sigma_i$ is a handle, and hence we see that handle reduction generalizes free reduction.

Reducing a braid yields an equivalent braid: as illustrated in Figure 8, the $(i+1)$th strand in a σ_i-handle forms a sort of handle, and the reduction consists of pushing that strand so that it passes above the next crossings instead of below. So, as in the case of free reduction, if there is a reduction sequence from a braid w to ε, i.e., a sequence $w = w_0, w_1, \ldots, w_N = \varepsilon$ such that, for each k, w_{k+1} is obtained from w_k by replacing some handle of w_k by its reduction, then w is equivalent to ε, i.e., it represents the empty word ε.

The following result of Dehornoy [19] shows the converse implication and the termination of the process of handle reductions:

Proposition 3.2. *Assume that* $w \in B_n$ *has a length* ℓ. *Then every reduction sequence starting from* w *leads in at most* $2^{\ell^4 n}$ *steps to an irreducible braid (with respect to Dehornoy's reductions). Moreover, the empty word* ε *is the only irreducible word in its equivalence class, hence* w *represents the empty braid if and only if any reduction sequence starting from* w *finishes with the empty word.*

A braid may contain many handles, so building an actual algorithm requires to fix a strategy prescribing in which order the handles will be reduced. Several variants have been considered; as can be expected, the most efficient ones use a Divide-And-Conquer trick.

For our current purpose, the important fact is that, although the proved complexity upper bound of above proposition is very high, handle reduction is extremely efficient in practice, even more than the reduction to a normal form, see [20].

Remark 3.3. *In* [24]*, Dehornoy gives an alternative proof for the convergence of the handle reduction algorithm of braids which is both more simple and more precise than the one in his original paper on handle reductions* [19]*.*

For more solutions for the word problem for the braid groups, see [28] and [33].

4. What is Public Key Cryptography?

The idea of Public Key Cryptography (PKC) was invented by Diffie and Hellman [27]. At the heart of this concept is the idea of using a one-way function for encryption (see the survey paper of Koblitz and Menezes [49]).

The functions used for encryption belong to a special class of *one-way functions* that remain one-way only if some information (the decryption key) is kept secret. If we use informal terminology, we can define a *public-key encryption function* as a map from plain text message units to ciphertext message units that can be feasibly computed by anyone having the public key, but whose inverse function (which deciphers the ciphertext message units) cannot be computed in a reasonable amount of time without some additional information, called the *private key*.

This means that everyone can send a message to a given person using the same enciphering key, which can simply be looked up in a public directory whose contents can be authenticated by some means. There is no need for the sender to have made any secret arrangement with the recipient; indeed, the recipient need never have had any prior contact with the sender at all.

Some of the purposes for which public-key cryptography has been applied are:

- **Confidential message transmission:** Two people want to exchange messages in the open airwaves, in such a way that an intruder observing the communication cannot understand the messages.

- **Key exchange:** Two people using the open airwaves want to agree upon a secret key for use in some symmetric-key cryptosystem. The agreement should be in such a way that an intruder observing the communication cannot deduce any useful information about the common secret.
- **Authentication:** The prover wishes to convince the verifier that he knows the private key without enabling an intruder watching the communication to deduce anything about his private key.
- **Signature:** The target in this part is: The sender of the message has to send the receiver a (clear or ciphered) message together with a signature proving the origin of the message. Each signature scheme may lead to an authentication scheme: in order to authenticate the sender, the receiver can send a message to the sender, and require that the sender signs this message.

Now, we give some examples of the most famous and well-known public-key cryptosystems.

4.1. **Diffie-Hellman.** In 1976, Diffie and Hellamn [27] introduced a key-exchange protocol which is based on the apparent difficulty of computing logarithms over a finite field $GF(q)$ with a one number q of elements and on some commutative property of the exponent.

Their key-exchange protocol works as follows:

Protocol 4.1.

Public keys: *q and a primitive element α.*
Private keys: *Alice: X_i; Bob: X_j.*

Alice: *Sends Bob $Y_i = \alpha^{X_i} \pmod{q}$.*
Bob: *Sends Alice $Y_j = \alpha^{X_j} \pmod{q}$*

Shared secret key: $K_{ij} = \alpha^{X_i X_j} \pmod{q}$

K_{ij} is indeed a shared key since Alice can compute $K_{ij} = Y_j^{X_i} \pmod{q}$ and Bob can compute $K_{ij} = Y_i^{X_j} \pmod{q}$.

This method is secured due to the hardness of the Discrete Logarithm Problem.

4.2. **RSA.** Rivest, Shamir and Adleman [70] introduced one of the most famous and common cryptosystem, which is called RSA. This method is widely used in commerce.

Find two large prime numbers p and q, each about 100 decimal digits long. Let $n = pq$ and $\phi = \phi(n) = (p-1)(q-1)$ (the Euler number). Choose a random integer E between 3 and ϕ that has no common factors with ϕ. It is easy to find an integer D that is the "inverse" of E modulo ϕ, that is, $D \cdot E$ differs from 1 by a multiple of ϕ.

Alice makes E and n public. All the other quantities here are kept secret.

The encryption is done as follows: Bob, who wants to send a plain text message P to Alice, that is an integer between 0 and $n-1$, computes the ciphertext integer $C = P^E \pmod{n}$. (In other words, raise P to the power E, divide the result by n, and let C be the remainder). Then, Bob sends C to Alice.

For decrypting the message, Alice uses the secret decryption number D for finding the plain text P by computing: $P = C^D \pmod{n}$.

This method is currently secure, since in order to determine the secret decryption key D (for decrypting the message), the intruder should factor the 200 or so digit number n, which is a very hard task.

5. Key-exchange protocols based on the braid group

In this section, we present some key-exchange protocols which are based on apparently hard problems in the braid group. After the transmitter and receiver agree on a shared secret key, they can use a symmetric cryptosystem for transmitting messages in the insecure channel.

5.1. Anshel-Anshel-Goldfeld key-exchange protocol.

The following scheme was proposed theoretically by Anshel, Anshel and Goldfeld [2], and implemented in the braid group by Anshel, Anshel, Fisher and Goldfeld [1].

This scheme assumed that the Conjugator Search Problem is difficult enough (so this scheme, as well as the other schemes described below, would keep its interest, even if it turned out that braid groups are not relevant). The Conjugator Search Problem is:

Problem 5.1. *Given two braids p, p' which are conjugate. Find an element s which satisfies: $p' = s^{-1}ps$.*

We start with two public sets of braids, $p_1, \dots, p_k$ and $q_1, \dots, q_m$ in B_n. The secret key of Alice is a word u on alphabet of size k and their inverses, and the secret key of Bob is a word v on a different alphabet of size m and their inverses. We denote by $u(p_1, \dots, p_k)$ the substitution of the ith letter of the alphabet by p_i (for all $1 \leq i \leq k$).

The key-exchange protocol is as follows:

Protocol 5.2.

Public keys: $p_1, \dots, p_k$ *and* $q_1, \dots, q_m$ *in* B_n.

Private keys: *Alice: u; Bob: v.*

Alice: *computes $s = u(p_1, \dots, p_k)$, and sends Bob the conjugates $q'_1 = sq_1s^{-1}, \dots, q'_m = sq_ms^{-1}$.*
Bob: *computes $r = v(q_1, \dots, q_m)$, and sends Alice the conjugates $p'_1 = rp_1r^{-1}, \dots, p'_k = rp_kr^{-1}$.*

Shared secret key: $K = E(su(p'_1, \dots, p'_k)^{-1}) = E(v(q'_1, \dots, q'_m)r^{-1})$ *where E is the colored Burau representation of the braid group defined by Morton* [66] *(see Section 8.4.1 below).*

K is indeed a shared key since Alice can compute $K_A = su(p'_1, \dots, p'_k)^{-1}$ and Bob can compute $K_B = v(q'_1, \dots, q'_m)r^{-1}$, and they are equal since:

$$\begin{aligned} K_A &= su(p'_1, \dots, p'_k)^{-1} = sru(p_1, \dots, p_k)^{-1}r^{-1} = \\ &= srs^{-1}r^{-1} = sv(q_1, \dots, q_m)s^{-1}r^{-1} = v(q'_1, \dots, q'_m)r^{-1} = K_B, \end{aligned}$$

so both of them can compute $K = E(K_A) = E(K_B)$
The security is based on the difficulty of a variant to the Conjugator Search Problem in B_n, namely the *Multiple Conjugator Search Problem*, in which one tries to find a conjugating braid starting not from one single pair of conjugate braids (p, p'), but from a finite family of such pairs $(p_1, p'_1), \dots, (p_k, p'_k)$ obtained using the same conjugating braid. It should be noted that the Multiple Conjugator Search Problem may be easier than the original Conjugator Search Problem.
In [1], it is suggested to work in B_{80} with $k = m = 20$ and short initial braids p_i, q_j of length 5 or 10 Artin generators.

5.2. **Diffie-Hellman-type key-exchange protocol.** Following the commutative idea for achieving a shared secret key of Diffie-Hellman, Ko et al. [48] proposed a key-exchange protocol based on the braid group and some commutative property of some of its elements. Although braid groups are not commutative, we can find large subgroups such that each element of the first subgroup commutes with each element of the second. Indeed, braids involving disjoint sets of strands commute (see also [80]).
Denote by LB_n (resp. UB_n) the subgroup of B_n generated by $\sigma_1, \dots, \sigma_{m-1}$ (resp. $\sigma_{m+1}, \dots, \sigma_{n-1}$) with $m = \lfloor \frac{n}{2} \rfloor$. Then, every braid in LB_n commutes with every braid in UB_n.
Here is Ko et al. key-exchange protocol:

Protocol 5.3.
Public key: *one braid p in B_n.*
Private keys: *Alice: $s \in LB_n$; Bob: $r \in UB_n$.*

Alice: *Sends Bob $p' = sps^{-1}$.*

Bob: *Sends Alice* $p'' = rpr^{-1}$

Shared secret key: $K = srpr^{-1}s^{-1}$

K is a shared key since Alice can compute $K = sp''s^{-1}$ and Bob can compute $K = rp'r{-}1$, and both are equal to K since s and r commute.

The security is based on the difficulty of the Conjugator Search Problem in B_n, or, more exactly, on the difficulty of the following variant, which can be called the Diffie-Hellman-like Conjugacy Problem:

Problem 5.4. *Given a braid* p *in* B_n*, and the braids* $p' = sps^{-1}$ *and* $p'' = rpr^{-1}$*, where* $s \in LB_n$ *and* $r \in UB_n$*, find the braid* $rp'r^{-1}$*, which is also* $sp''s^{-1}$*.*

The suggested parameters are $n = 80$, i.e. to work in B_{80}, with braids specified using (normal) sequences of length 12, i.e., sequences of 12 permutations (see [16]).

6. More cryptology based on the braid group

6.1. **Encryption and decryption.** The following scheme is proposed by Ko et al. [48]. We continue with the same notation of Ko et al. Assume that h is a collision-free one-way hash function of B_n to $\{0,1\}^{\mathbb{N}}$, i.e., a computable function such that the probability of having $h(b_2) = h(b_1)$ for $b_2 \neq b_1$ is negligible (collision-free), and retrieving b from $h(b)$ is infeasible (one-way) (for some examples see Dehornoy [20, Section 4.4]).

We start with $p \in B_n$ and $s \in LB_n$. Alice's public key is the pair (p, p'), with $p' = sps^{-1}$ where s is Alice's private key. For sending the message m_B, which we assume lies in $\{0,1\}^{\mathbb{N}}$, Bob chooses a random braid r in UB_n and he sends the encrypted text $m''_B = m_B \oplus h(rp'r^{-1})$ (using $\oplus$ for the Boolean operation "exclusive-or", i.e. the sum in $\mathbb{Z}/2\mathbb{Z}$), together with the additional datum $p'' = rpr^{-1}$. Now, Alice computes $m_A = m'' \oplus h(sp''s^{-1})$, and we have $m_A = m_B$, which means that Alice retrieves Bob's original message.

Indeed, because the braids r and s commute, we have (as before):

$$sp''s^{-1} = srpr^{-1}s^{-1} = rsps^{-1}r^{-1} = rp'r^{-1},$$

and, therefore, $m_A = m_B \oplus h(rp'r^{-1}) \oplus h(rp'r^{-1}) = m_B$.

The security is based on the difficulty of the Diffie-Hellmann-like Conjugacy Problem in B_n. The recommended parameters are as in Ko et al's exchange-key protocol (see Section 5.2).

6.2. **Authentication schemes.** Three authentication schemes were introduced by Sibert, Dehornoy and Girault [79], which are based on the Conjugacy Search problem and Root Extraction Problem. Concerning the cryptanalysis of the Root Extraction Problem, see [41].

Two more authentication schemes were suggested by Lal and Chaturvedi [51]. Their cryptanalysis were discussed in [81] and [41].

7. Attacks on the conjugacy search problem using Summit Sets

In this section, we explain the algorithms for solving the conjugacy decision and search problems (CDP/CSP) in braid groups that were given in [37, 31, 32, 38] (actually, these algorithms works also in Garside groups, but for our current purposes it suffices).

We follow here the excellent presentation of Birman, Gebhardt and Gonzalez-Meneses [8]. For more details, see their paper.

7.1. **The basic idea.** Given an element $x \in B_n$, the algorithm computes a finite subset I_x of the conjugacy class of x which has the following properties:

(1) For every $x \in B_n$, the set I_x is finite, non-empty and only depends on the conjugacy class of X. It means that two elements $x, y \in B_n$ are conjugate if and only if $I_x = I_y$.
(2) For each $x \in B_n$, one can compute efficiently a representative $\tilde{x} \in I_x$ and an element $a \in B_n$ such that $a^{-1}xa = \tilde{x}$.
(3) There is a finite algorithm which can construct the whole set I_x for any representative $\tilde{x} \in I_x$.

Now, for solving the CDP/CSP for given $x, y \in B_n$ we have to perform the following steps.

(a) Find representatives $\tilde{x} \in I_x$ and $\tilde{y} \in I_y$.
(b) Using the algorithm from property (3), compute further elements of I_x (while keeping track of he conjugating elements), until either:
    (i) $\tilde{y}$ is found as an element of I_x, proving x and y to be conjugate and providing a conjugating element, or
    (ii) the entire set I_x has been constructed without encountering $\tilde{y}$, proving that x and y are not conjugate.

We now survey the different algorithms based on this approach.

In Garside's original algorithm [37], the set I_x is the *Summit Set* of x, denoted $\mathrm{SS}(x)$, which is the set of conjugates of x having maximal infimum.

Remark 7.1. *All the algorithms presented below for the Super Summit Sets and the Ultra Summit Sets work also for Garside groups, which are a generalization of the braid groups. In our survey, for simplification, we present them in the language of braid groups. For more details on the Garside groups and the generalized algorithms, see* [8].

7.2. **The Super Summit Sets.** The Summit Sets are improved by Elrifai and Morton [30], who considered $I_x = \mathrm{SSS}(x)$, the *Super Summit Set* of x, consisting of the conjugates of x having minimal canonical length $\mathrm{len}(x)$. They also show that $\mathrm{SSS}(x)$ is the set of conjugates of x having maximal infimum and minimal supremum, at the same time. In general $\mathrm{SSS}(x)$ is much smaller than $\mathrm{SS}(x)$.

Starting by a given element x, one can find an element $\tilde{x} \in \mathrm{SSS}(x)$ by a sequence of special conjugations, called *cyclings* and *decyclings*:

Definition 7.2. *Let $x = \Delta^p x_1 \cdots x_r \in B_n$ be given in Garside normal form and assume $r > 0$.*

The cycling *of x, denoted by $\mathbf{c}(x)$ is:*

$$\mathbf{c}(x) = \Delta^p x_2 \cdots x_r \tau^{-p}(x_1).$$

where τ is the involution which maps σ_i to σ_{n-i}, for all $1 \leq i \leq n$.

The decycling *of x, denoted by $\mathbf{d}(x)$ is:*

$$\mathbf{d}(x) = x_r \Delta^p x_1 x_2 \cdots x_{r-1} = \Delta^p \tau^{-p}(x_r) x_1 x_2 \cdots x_{r-1}.$$

If $r = 0$, we have $\mathbf{c}(x) = \mathbf{d}(x) = x$.

Note that $\mathbf{c}(x) = (\tau^{-p}(x_1))^{-1} x (\tau^{-p}(x_1))$ and $\mathbf{d}(x) = x_r^{-1} x x_r$. This means that for an element of positive canonical length, the cycling of x is computed by moving the first permutation braid of x to the end, while the decycling of x is computed by moving the last permutation braid of x to the front. Moreover, for every $x \in B_n$, $\inf(x) \leq \inf(\mathbf{c}(x))$ and $\sup(x) \geq \sup(\mathbf{d}(x))$.

Note that the above decompositions of $\mathbf{c}(x)$ and $\mathbf{d}(x)$ are not, in general, Garside normal forms. Hence, if one wants to perform iterated cyclings or decyclings, one needs to compute the left normal form of the resulting element at each iteration.

Given x, one can use cyclings and decyclings to find an element in $\mathrm{SSS}(x)$ in the following way: Suppose that we have an element $x \in B_n$ such that $\inf(x)$ is not equal to the maximal infimum in the conjugacy class of x. Then we can increase the infimum by repeated cycling (due to [30] and [12]): there exists a positive integer k_1 such that $\inf(\mathbf{c}^{k_1}(x)) > \inf(x)$. Therefore, by repeated cycling, we can conjugate x to another element $\hat{x}$ of maximal infimum. Once $\hat{x}$ is obtained, if the supremum is not minimal in the conjugacy class, we can decrease its supremum by repeated decycling. Again, due to [30] and [12], there exists an integer k_2 such that $\sup(\mathbf{d}^{k_2}(\hat{x})) < \sup(\hat{x})$. Hence, using repeated cycling and decycling a finite number of times, one obtains an element in $\mathrm{SSS}(x)$.

If we denote by m the length of x in Artin generators and r is the canonical length of x, then we have (see [30] and [12]):

Proposition 7.3. *A sequence of at most rm cyclings and decyclings applied to x produces a representative $\tilde{x} \in \mathrm{SSS}(x)$.*

Now, we have to explore all the set $\mathrm{SSS}(x)$. We have the following result (see [30]):

Proposition 7.4. *Let $x \in B_n$ and $V \subset \mathrm{SSS}(x)$ be non-empty. If $V \neq \mathrm{SSS}(x)$, then there exist $y \in V$ and a permutation braid s such that $s^{-1}ys \in \mathrm{SSS}(x) \setminus V$.*

Since $\mathrm{SSS}(x)$ is a finite set, the above proposition allows to compute the whole $\mathrm{SSS}(x)$. More precisely, if one knows a subset $V \subset \mathrm{SSS}(x)$ (we start with: $V = \{\tilde{x}\}$), one conjugates each element in V by all permutation braids ($n!$ elements). If one encounters a new element z with the same canonical length as $\tilde{x}$ (which is a new element in $\mathrm{SSS}(x)$), then add z to V and start again. If no new element is found, this means that $V = \mathrm{SSS}(x)$, and we are done.

One important remark is that this algorithm not only computes the set $\mathrm{SSS}(x)$, but it also provides conjugating elements joining the elements in $\mathrm{SSS}(x)$.

Now the checking if x and y are conjugate, is done as follows: Compute representatives $\tilde{x} \in \mathrm{SSS}(x)$ and $\tilde{y} \in \mathrm{SSS}(y)$. If $\inf(\tilde{x}) \neq \inf(\tilde{y})$ or $\sup(\tilde{x}) \neq \sup(\tilde{y})$, then x and y are not conjugate. Otherwise, start computing $\mathrm{SSS}(x)$ as described above. The elements x and y are conjugate if and only if $\tilde{y} \in \mathrm{SSS}(x)$. Note that if x and y are conjugate, an element conjugating x to y can be found by keeping track of the conjugations during the computations of $\tilde{x}$, $\tilde{y}$ and $\mathrm{SSS}(x)$. Hence it solves the Conjugacy Decision Problem and the Conjugacy Search Problem simultaneously.

From the algorithm, we see that the computational cost of computing $\mathrm{SSS}(x)$ depends mainly in two ingredients: the size of $\mathrm{SSS}(x)$ and the number of permutation braids. In B_n, all known upper bounds for the size of $\mathrm{SSS}(x)$ are exponential in n, although it is conjectured that for fixed n, a polynomial bound in the canonical length of x exists [31].

Franco and Meneses [32] reduce the size of the set we have to conjugate with, by the following observation:

Proposition 7.5. *Let $x \in B_n$ and $V \subset \mathrm{SSS}(x)$ be non-empty. If $V \neq \mathrm{SSS}(x)$ then there exist $y \in V$ such that $\sigma_i^{-1} y \sigma_i \in \mathrm{SSS}(x) \setminus V$ for some $1 \leq i \leq n-1$.*

Using this proposition, the $\mathrm{SSS}(x)$ can be computed as in [30], but instead of conjugating each element $y \in \mathrm{SSS}(X)$ by all permutation braids, it suffices to conjugate y by the Artin generators σ_i ($1 \leq i \leq n-1$).

Note that the algorithm computes a directed graph whose vertices are the elements in $\mathrm{SSS}(x)$, and whose arrows are defined as follows: for any two elements $y, z \in \mathrm{SSS}(x)$, there is an arrow labeled by σ_i starting at y and ending at z if $\sigma_i^{-1} y \sigma_i = z$.

Hence, the size of the set of permutation braids is no longer a problem for the complexity of the algorithm (since we can use the Artin generators instead), but there is still a big problem to handle: The size of $\mathrm{SSS}(x)$ is, in general, very big. The next improvement tries to deal with this.

7.3. **The Ultra Summit Sets.**

Gebhardt [38] defines a small subset of $\mathrm{SSS}(x)$ satisfying all the good properties described above, so that a similar algorithm can be used to compute it. The definition of this new subset appeared after observing that the cycling function maps $\mathrm{SSS}(x)$ to itself. As $\mathrm{SSS}(x)$ is finite, iterated cycling of any representative of $\mathrm{SSS}(x)$ must eventually become periodic. Hence it is natural to define the following:

Definition 7.6. *Given $x \in B_n$, we define the* Ultra Summit Set *of x,* $\mathrm{USS}(x)$*, to be the set of elements $y \in \mathrm{SSS}(x)$ such that $\mathbf{c}^m(y) = y$, for some $m > 0$.*

Hence, the Ultra Summit Set $\mathrm{USS}(x)$ consists of a finite set of disjoint orbits, closed under cycling.

Example 7.7. [8] *One has* $\mathrm{USS}(\sigma_1) = \mathrm{SSS}(\sigma_1) = \mathrm{SS}(\sigma_1) = \{\sigma_1, \ldots, \sigma_{n-1}\}$*, and each element corresponds to an orbit under cycling, since* $\mathrm{c}(\sigma_i) = \sigma_i$ *for $i = 1, \ldots, n-1$.*

A more interesting example is given by the element

$$x = \sigma_1\sigma_3\sigma_2\sigma_1 \cdot \sigma_1\sigma_2 \cdot \sigma_2\sigma_1\sigma_3 \in B_4.$$

In this example, $\mathrm{USS}(x)$ *has* 6 *elements, while* $\mathrm{SSS}(x)$ *has* 22 *elements. More precisely, $USS(x)$ consists of 2 closed orbits under cycling:* $\mathrm{USS}(x) = O_1 \cup O_2$*, each one containing* 3 *elements:*

$$O_1 = \{\sigma_1\sigma_3\sigma_2\sigma_1\cdot\sigma_1\sigma_2\cdot\sigma_2\sigma_1\sigma_3, \sigma_1\sigma_2\cdot\sigma_2\sigma_1\sigma_3\cdot\sigma_1\sigma_3\sigma_2\sigma_1, \sigma_2\sigma_1\sigma_3\cdot\sigma_1\sigma_3\sigma_2\sigma_1\cdot\sigma_1\sigma_2\},$$

$$O_2 = \{\sigma_3\sigma_1\sigma_2\sigma_3\cdot\sigma_3\sigma_2\cdot\sigma_2\sigma_3\sigma_1, \sigma_3\sigma_2\cdot\sigma_2\sigma_3\sigma_1\cdot\sigma_3\sigma_1\sigma_2\sigma_3, \sigma_2\sigma_3\sigma_1\cdot\sigma_3\sigma_1\sigma_2\sigma_3\cdot\sigma_3\sigma_2\}.$$

Notice that $O_2 = \tau(O_1)$.

Notice also that the cycling of every element in $\mathrm{USS}(x)$ *gives another element which is already in left normal form, hence iterated cyclings corresponds to cyclic permutations of the factors in the left normal form. Elements which satisfies this property are called* rigid *(see* [8]*).*

Remark 7.8. *The size of the Ultra Summit Set of a generic braid of canonical length* ℓ *is either* ℓ *or* 2ℓ [38]. *This means that, in the generic case, Ultra Summit Sets consist of one or two orbits (depending on whether* $\tau(O_1) = O_1$ *or not), containing rigid braids. But, there are exceptions: for example, the following braid in* B_{12}:

$$\begin{aligned} E &= (\sigma_2\sigma_1\sigma_7\sigma_6\sigma_5\sigma_4\sigma_3\sigma_8\sigma_7\sigma_{11}\sigma_{10}) \cdot (\sigma_1\sigma_2\sigma_3\sigma_2\sigma_1\sigma_4\sigma_3\sigma_{10}) \cdot \\ &\quad \cdot(\sigma_1\sigma_3\sigma_4\sigma_{10}) \cdot (\sigma_1\sigma_{10}) \cdot (\sigma_1\sigma_{10}\sigma_9\sigma_8\sigma_7\sigma_{11}) \cdot (\sigma_1\sigma_2\sigma_7\sigma_{11}) \end{aligned}$$

has an Ultra Summit Set of size 264, *instead of the expected size* 12 *(see* [9, Example 5.1]*).*

In the case of braid groups, the size and structure of the Ultra Summit Sets happen to depend very much on the geometrical properties of the braid, more precisely, on its Nielsen-Thurston type: periodic, reducible or Pseudo-Anosov (see [8, 9]*).*

The algorithm given in [38] to solve the CDP/CSP in braid groups (using Ultra Summit Sets) is analogous to the previous ones, but this time one needs to compute $\mathrm{USS}(x)$ instead of $\mathrm{SSS}(x)$. In order to do this, we first have to obtain an element $\hat{x} \in \mathrm{USS}(x)$. This we do as follows: take an element $\tilde{x} \in \mathrm{SSS}(x)$. Now, start cycling it. Due to the facts that cycling an element is $\mathrm{SSS}(x)$ will result in another element in $\mathrm{SSS}(x)$ and that the Super Summit Set of x is finite, we will have two integers m_1, m_2 $(m_1 < m_2)$, which satisfy:

$$\mathbf{c}^{m_1}(\tilde{x}) = \mathbf{c}^{m_2}(\tilde{x})$$

When having this, the element $\hat{x} = \mathbf{c}^{m_1}(\tilde{x})$ is in $\mathrm{USS}(x)$, since $\mathbf{c}^{m_2-m_1}(\hat{x}) = \hat{x}$.

After finding a representative $\hat{x} \in \mathrm{USS}(x)$, we have to explore all the set $\mathrm{USS}(x)$. This we do using the following results of Gebhardt [38].

Proposition 7.9. *Let* $x \in B_n$ *and* $y \in \mathrm{USS}(x)$. *For every positive braid* u *there is a unique* $\preceq$*-minimal element* $c_y(u)$ *satisfying* $u \preceq c_y(u)$ *and* $(c_y(u))^{-1}y(c_y(u)) \in \mathrm{USS}(x)$.

Definition 7.10. *Given* $x \in B_n$ *and* $y \in \mathrm{USS}(x)$, *we say that a permutation braid* $s \neq 1$ *is a* minimal *for* y *with respect to* $\mathrm{USS}(x)$ *if* $s^{-1}ys \in \mathrm{USS}(x)$, *and no proper prefix of* s *satisfies this property.*

It is easy to see that the number of minimal permutation braids for y is bounded by the number of Artin's generators.

Now, we have:

Proposition 7.11. *Let $x \in B_n$ and $V \subseteq \mathrm{USS}(x)$ be non-empty. If $V \neq \mathrm{USS}(x)$, then there exist $y \in V$ and a generator σ_i such that $c_y(\sigma_i)$ is a minimal permutation braid for y, and $(c_y(\sigma_i))^{-1}y(c_y(\sigma_i)) \in \mathrm{USS}(X) \setminus V$.*

In [38], it is shown how to compute the minimal permutation braids (they are called there *minimal simple elements* in the Garside group's language) corresponding to a given $y \in \mathrm{USS}(x)$ (a further discussion on the minimal simple elements with some examples can be found in [9]). Hence, one can compute the whole $\mathrm{USS}(x)$ starting by a single element $\hat{x} \in \mathrm{USS}(x)$, and then we are done.

As the case of the Super Summit Sets, the algorithm of Gebhardt [38] not only computes $\mathrm{USS}(x)$, but also a graph Γ_x, which determines the conjugating elements. This graph is defined as follows.

Definition 7.12. *Given $x \in B_n$, the directed graph Γ_x is defined by the following data:*

1. *The set of vertices is $\mathrm{USS}(x)$.*
2. *For every $y \in \mathrm{USS}(x)$ and every minimal permutation braid s for y with respect to $\mathrm{USS}(x)$, there is an arrow labeled by s going from y to $s^{-1}ys$.*

Concerning the complexity of this algorithm for solving the Conjugacy Search Problem, the number m_2 of times one needs to apply cycling for finding an element in $\mathrm{USS}(x)$ is not known in general. Nevertheless, in practice, the algorithm based on the Ultra Summit Sets is substantially better for braid groups (see [8]). For more information on the Ultra Summit Sets and its structure, see [9].

Remark 7.13. *One might think that for a given element $x \in B_n$, it is possible that its Ultra Summit Set with respect to the Garside normal form will be different from its Ultra Summit Set with respect to the right normal form (see Section 2.1). If this happens, it is possible that even though one of the Ultra Summit Sets is large, the other will be small.*

Gebhardt and Meneses [39] *shows that at least for* rigid braids*, the size of the above two Ultra Summit Sets is equal, and their associated graphs are isomorphic. A braid w is called* rigid*, if the cycling of w, $\mathbf{c}(w)$ is already given in Garside normal form, with no need for changing the permutation braids (see also* [8, Section 3] *and Example 7.7 here). They conjecture that this is the situation for any braid.*

More information about this sorts of Summit sets can be found in the series of papers [8, 9, 10] and [53, 54, 55].

8. More attacks on the conjugacy search problem

There are some more ways to attack the Conjugacy Search Problem, apart of solving it completely. In this section, we present some techniques to attack the conjugacy search problem without actually solving it.

8.1. **A heuristic algorithm using the Super Summit Sets.**

Hofheinz and Steinwandt [43] use a heuristic algorithm for attacking the Conjugacy Search Problem which is the basis of the cryptosystems of Anshel-Anshel-Goldfeld [1] and Ko et al. [48].

Their algorithm is based on the idea that it is probable that if we start with two elements in the same conjugacy class, their representatives in the Super Summit Set will not be too far away, i.e. one representative is a conjugation of the other by a permutation braid.

So, given a pair (x, x') of braids, where $x' = s^{-1}xs$, we do the following steps:

(1) By a variant of cycling (adding a multiplication by Δ to the first permutation braid, based on Proposition 1 in [52]) and decycling, we find $\tilde{x} \in \mathrm{SSS}(x)$ and $\tilde{x}' \in \mathrm{SSS}(x')$.
(2) Try to find a permutation braid P, such that $\tilde{x}' = P^{-1}\tilde{x}P$.

In case we find such a permutation braid P, since we can follow after the conjugators in the cycling/decycling process, at the end of the algorithm we will have at hand the needed conjugator for breaking the cryptosystem. Note that we do not really need to find exactly s, since each $\tilde{s}$ which satisfies $x' = \tilde{s}^{-1}x\tilde{s}$ will do the job as well and reveal the shared secret key.

Their experiments show that they succeed to reveal the shared secret key in almost 100% of the cases in the Anshel-Anshel-Goldfeld protocol (where the cryptosystem is based on the Multiple Simultaneous Conjugacy Problem) and in about 80% of the cases in the Diffie-Hellman-type protocol.

Note that their attack is special to cryptosystems which are based on the conjugacy problem, since it depends very much on the fact that x and x' are conjugate.

8.2. **Reduction of the Conjugacy Search Problem.**

Maffre [58, 59] presents a deterministic, polynomial algorithm that reduces the conjugacy search problem in braid group.

The algorithm is based on the decomposition of braids into products of canonical factors and gives a partial factorization of the secret: a divisor and a multiple. The tests which were performed on different keys of existing protocols showed that many protocols in their current form are broken and that the efficiency of their attack depends on the random generator used to create the key.

8.3. **Length-based attacks.** A different probabilistic attack on the braid group cryptosystems is the *length-based attack*. In this section, we will sketch its basic idea, and different variants of this attack on the braid group cryptosystems.

8.3.1. *The basic idea.* The basic idea was introduced by Hughes and Tannenbaum [45].

Let ℓ be a length function on the braid group B_n. In the Conjugacy Search Problem, we have an instance of (p, p') where $p' = s^{-1}ps$, and we look for s. The idea of a probabilistic length-based attack to this problem is: if we can write $s = s'\sigma_i$ for a given i, then the length $\ell(\sigma_i s^{-1} p s \sigma_i^{-1})$ should be strictly smaller than the length $\ell(\sigma_j s^{-1} p s \sigma_j^{-1})$ for $j \neq i$.

Thus, for using such an attack, one should choose a good length function on B_n and run it iteratively till we get the correct conjugator.

8.3.2. *Choosing a length function.* In [35], we suggest some length functions for this purposes. The first option is the *Garside length*, which is the length of the Garside normal form by means of Artin generators (i.e. if $w = \Delta_n^r P_1 P_2 \cdots P_k$, then $\ell_{\mathrm{Gar}}(w) = r|\Delta| + |P_1| + |P_2| + \cdots + |P_k|$).

A better length function is the *Reduced Garside length* (which is called *Mixed Garside length* in [31]). The motivation for this length function is that a part of the negative powers of Δ_n can be canceled with the positive permutation braids. Hence, it is defined as follows: if $w = \Delta_n^{-r} P_1 P_2 \cdots P_k$, then:

$$\ell_{\mathrm{RedGar}}(w) = \ell_{\mathrm{Gar}}(w) - \sum_{i=1}^{\min\{r,k\}} |P_i|.$$

This length function is much more well-behaved, and hence it gives better performances. But even this length function did not give a break of the cryptosystems (by the basic length-based attack).

In [42], Hock and Tsaban checked the corresponding length functions for the Birman-Ko-Lee presentation, and they found out that the reduced length with respect to the Birman-Ko-Lee presentation is even better than the reduced Garside length.

8.3.3. *The memory approach.* The main contribution of [34] is new improvements to the length-based attack.

First, it introduces a new approach which uses memory: In the basic length-based attack, we hold each time only the best conjugator so far. The problem with this is that sometimes a prefix of the correct conjugator is not the best conjugator at some iteration and hence it is thrown out. In such a situation, we just miss the correct conjugator in the way, and hence the length-based algorithm fails. Moreover, even if we use a 'look ahead' approach, which means that instead of adding one generator in each iteration we add several generators in each iteration, we still get total failure for the suggested parameters, and some success for small parameters [35].

In the memory approach, we hold each time a given number (which is the size of the memory) of possible conjugators which are the best among all the other conjugators of this length. In the next step, we conjugate all the conjugators in the memory by one more generator, and we choose again only the best ones among all the possibilities. In this approach, in a successful search, we will often have the correct conjugator in the first place of the memory.

The results of [34] show that the length-based attack with memory is applicable to the cryptosystems of Anshel-Anshel-Goldfeld and Ko et al, and hence their cryptosystems are not secure. Moreover, the experiments show that if we increase the size of the memory, the success rate of the length-based attack with memory becomes higher.

8.3.4. *Applicability of the length-based approach.* One interesting point about the length-based approach is that it is applicable not only for the Conjugacy Search Problem, but also for solving equations in groups. Hence, it is a threat also to the Decomposition Problem and for the Shifted Conjugacy Problem which was introduced by Dehornoy (see [21] and Section 10.1.1 here).

Moreover, the length-based approach is applicable in any group which has a reasonable length function, e.g. the Thompson group, as indeed was done by Ruinskiy, Shamir and Tsaban (see [72] and Section 9.2.1 here).

8.4. **Attacks based on linear representations.** A different way to attack these cryptographical schemes is by using linear representations of the braid groups. The basic idea is to map the braid groups into groups of matrices, in which the Conjugacy Search Problem is easy. In this way, we might solve the Conjugacy Search Problem of B_n.

8.4.1. *The Burau representations.* The best known linear representation of the braid group B_n is the Burau representation [14]. We present it here (we partially follow [52]).

The Burau representation is defined as follows. Let $\mathbb{Z}[t^{\pm 1}]$ be the ring of Laurent polynomials $f(t) = a_k t^k + a_{k+1}t^{k+1} + \cdots + a_m t^m$ with integer coefficients (and possibly with negative degree terms). Let $\mathrm{GL}_n(\mathbb{Z}[t^{\pm 1}])$ be the group of $n \times n$ invertible matrices over $\mathbb{Z}[t^{\pm 1}]$. The Burau representation is a homomorphism $B_n \to \mathrm{GL}_n(\mathbb{Z}[t^{\pm 1}])$ which sends a generator $\sigma_i \in B_n$ to the matrix:

$$\begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1-t & t & & \\ & & 1 & 0 & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \in \mathrm{GL}_n(\mathbb{Z}[t^{\pm 1}])$$

where $1-t$ occurs in row and column i of the matrix.

This representation is reducible, since it can be decomposed into the trivial representation of dimension 1 and an irreducible representation $B_n \to \mathrm{GL}_{n-1}(\mathbb{Z}[t^{\pm 1}])$ of dimension $n-1$, called *the reduced Burau representation*, which sends a generator $\sigma_i \in B_n$ to the matrix:

$$C_i(t) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & t & -t & 1 & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \in \mathrm{GL}_{n-1}(\mathbb{Z}[t^{\pm 1}])$$

where t occurs in row i of the matrix. If $i = 1$ or $i = n-1$, the matrix is truncated accordingly (see [52]).

Note that these matrices satisfy the braid group's relations:

$$C_i(t)C_j(t) = C_j(t)C_i(t) \text{ for } |i-j| > 2$$

$$C_i(t)C_{i+1}(t)C_i(t) = C_{i+1}(t)C_i(t)C_{i+1}(t) \text{ for } i = 1, \ldots, n-1$$

The Burau representation of B_n is faithful for $n = 3$ and it is known to be unfaithful for $n \geq 5$ [64, 65, 56, 5] (i.e. the map from B_n to the matrices is not injective). The case of $n = 4$ remains unknown. In the case of $n \geq 5$, the kernel is very small [82], and the probability that different braids admit the same Burau image is negligible.

Here is a variant of the Burau representation introduced by Morton [66]. The *colored Burau matrix* is a refinement of the Burau matrix by assigning σ_i to $C_i(t_{i+1})$, so that the entries of the resulting matrix have several variables. This naive construction does not give a group homomorphism. Thus the induced permutations are considered simultaneously. We label the strands of an n-braid by $t_1, \dots, t_n$, putting the label t_j on the strand which starts from the jth point on the right.

Now we define:

Definition 8.1. *Let $a \in B_n$ be given by a word $\sigma_{i_1}^{e_1} \cdots \sigma_{i_k}^{e_k}$, $e_j = \pm 1$. Let t_{j_r} be the label of the under-crossing strand at the rth crossing. Then* the colored Burau matrix $M_a(t_1, \dots, t_n)$ *of a is defined by*

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

The permutation group S_n acts on $\mathbb{Z}[t_1^{\pm 1}, \dots, t_{n-1}^{\pm 1}]$ from left by changing variables: for $\alpha \in S_n$, $\alpha(f(t_1, \dots, t_n)) = f(t_{\alpha(1)}, \dots, t_{\alpha(n)})$. Then S_n also acts on the matrix group $\mathrm{GL}_{n-1}(Z[t_1^{\pm 1}, \dots, t_n^{\pm 1}])$ entry-wise: for $\alpha \in S_n$ and $M = (f_{ij})$, then $\alpha(M) = (\alpha(f_{ij}))$. Then we have

Definition 8.2. *The* colored Burau group CB_n *is:*

$$S_n \times GL_{n-1}(\mathbb{Z}[t_1^{\pm 1}, \dots, t_n^{\pm 1}])$$

with multiplication $(\alpha_1, M_1) \cdot (\alpha_2, M_2) = (\alpha_1 \alpha_2, (\alpha_2^{-1} M_1) M_2)$. *The* colored Burau representation $C : B_n \to CB_n$ *is defined by* $C(\sigma_i) = ((i, i+1), C_i(t_{i+1}))$.

Then it is easy to see the following:

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

Using the Burau representation, the idea of Hughes [44] to attack the Anshel-Anshel-Goldfeld scheme [2, 1] is as follows: take one or several pairs of conjugate braids (p, p') associated with the same conjugating braids. Now, it is easy to compute their classical Burau image and to solve the Conjugator Search Problem in the linear group.

In general, this is not enough for solving the Conjugator Search Problem in B_n, because there is no reason for the conjugating matrix that has been found to belong to the image of the Burau representation, or that one can find a possible preimage. Since the kernel of the classical Burau representation is small [82], there is a non-negligible probability that we will find the correct conjugator and hence we break the cryptosystem.

In a different direction, Lee and Lee [52] indicate a weakness in the Anshel-Anshel-Goldfeld protocol in a different point. Their shared key is the colored Burau representation of a commutator element.

The motivation for this attack is that despite the change of variables in the colored Burau matrix by permutations, the matrix in the final output, which is the shared key, is more manageable than braids. They show that the security of the key-exchange protocol is based on the problems of listing all solutions to some Multiple Simultaneous Conjugacy Problems in a permutation group and in a matrix group over a finite field. So if both of the two listing problems are feasible, then we can guess correctly the shared key, without solving the Multiple Simultaneous Conjugacy Problem in braid groups.

Note that Lee-Lee attack is special to this protocol, since it uses the colored Burau representation of a commutator element, instead of using the element itself. In case we change the representation in the protocol, this attack is useless.

8.4.2. *The Lawrence-Krammer representation.* The Lawrence-Krammer representation is another linear representation of B_n, which is faithful [6, 50]. It associates with every braid in B_n a matrix of size $\binom{n}{2}$ with entries in a 2-variable Laurent polynomial ring $\mathbb{Z}[t^{\pm 1}, q^{\pm 1}]$.

Cheon and Jun [17] develop an attack against the scheme of Diffie-Hellman-type protocol based on the Lawrence-Krammer representation: as in the case of the Burau representation, it is easy to compute the images of the involved braids in the linear group and to solve the Conjugacy Problem there, but in general, there is no way to lift the solution back to the braid groups.

But, since we only have to find a solution to the derived Diffie-Hellman-like Conjugacy Problem:

Problem 8.3. *Given p, sps^{-1}, and rpr^{-1}, with $r \in LB_n$ and $s \in UB_n$, find $(rs)p(rs)^{-1}$.*

Taking advantage of the particular form of the Lawrence-Krammer matrices, which contain many 0's, Cheon and Jun obtain a solution with a polynomial complexity and they show that, for the parameters suggested by Ko et al. [48], the procedure is doable, and so the cryptosystem is not secure.

9. Public-Key cryptography in the Thompson group

When some of the cryptosystems on the braid groups were attacked, it was natural to look for different groups, with a hope that a similar cryptosystem on a different group will be more secure and more successful. The Thompson group is a natural candidate for such a group: there is a normal form which can computed efficiently, but the decomposition problem seems difficult. On this base, Shpilrain and Ushakov [75] suggest a cryptosystem.

In this section, we will define the Thompson group, the Shpilrain-Ushakov cryptosystem, and we discuss its cryptanalysis.

9.1. **Definitions and the Shpilrain-Ushakov cryptosystem.**

Thompson's group F is the infinite noncommutative group defined by the following generators and relations:

$$F = \langle \quad x_0, x_1, x_2, \dots \quad | \quad x_i^{-1} x_k x_i = x_{k+1} \quad (k > i) \quad \rangle$$

Each $w \in F$ admits a unique *normal form* [15]:

$$w = x_{i_1} \cdots x_{i_r} x_{j_t}^{-1} \cdots x_{j_1}^{-1},$$

where $i_1 \leq \cdots \leq i_r$, $j_1 \leq \cdots \leq j_t$, and if x_i and x_i^{-1} both occur in this form, then either x_{i+1} or x_{i+1}^{-1} occurs as well. The transformation of an element of F into its normal form is very efficient [75].

We define here a natural length function on the Thompson group:

Definition 9.1. The normal form length *of an element* $w \in F$*,* $\mathrm{LNF}(w)$*, is the number of generators in its normal form: If* $w = x_{i_1} \cdots x_{i_r} x_{j_t}^{-1} \cdots x_{j_1}^{-1}$ *is in normal form, then* $\mathrm{LNF}(w) = r + t$*.*

Shpilrain and Ushakov [75] suggest the following key-exchange protocol based on the Thompson group:

Protocol 9.2.

Public subgroups: A, B, W *of* F*, where* $ab = ba$ *for all* $a \in A$*,* $b \in B$
Public key: *a braid* $w \in W$*.*
Private keys: *Alice:* $a_1 \in A, b_1 \in B$*; Bob:* $a_2 \in A, b_2 \in B$*.*

Alice: *Sends Bob* $u_1 = a_1 w b_1$*.*
Bob: *Sends Alice* $u_2 = b_2 w a_2$

Shared secret key: $K = a_1 b_2 w a_2 b_1$

K is a shared key since Alice can compute $K = a_1 u_2 b_1$ and Bob can compute $K = b_2 u_1 a_2$, and both are equal to K since a_1, a_2 commute with b_1, b_2.

Here is a suggestion for implementing the cryptosystem [75]: Fix a natural number $s \geq 2$. Let $S_A = \{x_0 x_1^{-1}, \dots, x_0 x_s^{-1}\}$, $S_B = \{x_{s+1}, \dots, x_{2s}\}$

and $S_W = \{x_0, \dots, x_{s+2}\}$. Denote by A, B and W the subgroups of F generated by S_A, S_B, and S_W, respectively. A and B commute elementwise, as required.

The keys $a_1, a_2 \in A$, $b_1, b_2 \in B$ and $w \in W$ are all chosen of normal form length L, where L is a fixed integer, as follows: Let X be A, B or W. Start with the empty word, and multiply it on the right by a (uniformly) randomly selected generator, inversed with probability $\frac{1}{2}$, from the set S_X. Continue this procedure until the normal form of the word has length L.

For practical implementation of the protocol, it is suggested in [75] to use $s \in \{3, 4, \dots, 8\}$ and $L \in \{256, 258, \dots, 320\}$.

9.2. **Attacks on the cryptosystem.** In this section, we present some attacks on the Ushakov-Shpilrain cryptosystem.

9.2.1. *Length-based attack.* As mentioned before, the length-based attack is applicable for any group with a reasonable length function. Ruinskiy, Shamir and Tsaban [72] applied this attack to the Thompson group.

As before, the basic length-based attack without memory always fails for the suggested parameters. If we add the memory approach, there is some improvement: for a memory of size 1024, there is 11% success. But if the memory is small (up to 64), even the memory approach always fails. They suggest that the reason for this phenomenon (in contrast to a significant success for the length-based attack with memory on the braid group) is that the braid group is much closer to the free group than the Thompson group, which is relatively close to an abelian group.

Their improvement is trying to avoid repetitions. The problem is that many elements return over and over again, and hence the algorithm goes into loops which make its way to the solution much difficult. The solution of this is holding a list of the already-checked conjugators, and when we generate a new conjugator, we check in the list if it has already appeared (this part is implemented by a hash table). In case of appearance, we just ignore it. This improvement increases significantly the success rate of the algorithm: instead of 11% for a memory of size 1024, we now have 49.8%, and instead of 0% for a memory of size 64, we now have 24%.

In the same paper [72], they suggest some more improvements for the length-based algorithm. One of their reasons for continuing with the improvements is the following interesting fact which was pointed out by Shpilrain [74]: there is a very simple fix for key-agreement protocols that are broken in probability less than p: Agree on k independent keys in parallel, and XOR them all to obtain the shared key. The probability of breaking the shared key is at most p^k, which is much smaller.

In a different paper, Ruinskiy, Shamir and Tsaban [78] attack the key agreement protocols based on noncommutative groups from a different direction: by using functions that estimate the distance of a group element to a given subgroup. It is known that in general the Membership Problem is hard, but one can use some heuristic approaches for determining the distance of an element to a given subgroup, e.g., to count the number of generators which are not in the subgroup.

They test it against the Shpilrain-Ushakov protocol, which is based on Thompson's group F, and show that it can break about half the keys within a few seconds.

9.2.2. *Special attack by Matucci.* Some interesting special attack for the Ushakov-Shpilrain cryptosystem can be found in Kassabov and Matucci [62] and Mattuci [61].

10. Future directions

In this section, we will present some future directions for the cryptography based on the braid group.

10.1. **Recent cryptosystems based on the braid group.** In this part, we present recent updates on some problems in the braid group, on which one can construct a cryptosystem.

10.1.1. *A cryptosystem based on the Shifted Conjugacy Search Problem.* Recently, Dehornoy [21] suggested an authentication scheme which is based on the Shifted Conjugacy Search Problem.

Before we describe the scheme, let us define the Shifted Conjugacy Search Problem. Let $x, y \in B_\infty$. We define:

$$x * y = x \cdot \mathrm{d}y \cdot \sigma_1 \cdot \mathrm{d}x^{-1}$$

where $\mathrm{d}x$ is the *shift* of x in B_∞, i.e. d is the injective function on B_∞ which sends the generator σ_i to the generator σ_{i+1} for each $i \geq 1$. In this context, the Shifted Conjugacy Search Problem is:

Problem 10.1. *Let $s, p \in B_\infty$ and $p' = s * p$. Find a braid $\tilde{s}$ satisfying $p' = \tilde{s} * p$.*

Now, the suggested scheme is based on the Fiat-Shamir authentication scheme: We assume that S is a set and $(F_s)_{s \in S}$ is a family of functions of S to itself that satisfies the following condition:

$$F_r(F_s(p)) = F_{F_r(s)}(F_r(p)), \qquad r, s, p \in S$$

Alice is the prover who wants to convince Bob that she knows the secret key s. Then the scheme works as follows:

Protocol 10.2.

Public key: *Two elements* $p, p' \in S$ *such that* $p' = F_s(p)$.

Private keys: *Alice:* $s \in S$.

Alice: *Chooses a random* $r \in S$ *and sends Bob* $x = F_r(p)$ *and* $x' = F_r(p')$.

Bob: *Chooses a random bit* c *and sends it to Alice.*

Alice: *If* $c = 0$, *sends* $y = r$ *(then Bob checks:* $x = F_y(p)$ *and* $x' = F_y(p')$*);*

If $c = 1$, *sends* $y = F_r(s)$ *(then Bob checks:* $x' = F_y(x)$*).*

Dehornoy [21] suggests to implement this scheme on LD-systems. A *LD-system* is a set S with a binary operation which satisfies:

$$r * (s * p) = (r * s) * (r * p).$$

The Fiat-Shamir-type scheme on LD-systems works as follows:

Protocol 10.3.

Public key: *Two elements* $p, p' \in S$ *such that* $p' = s * p$.

Private keys: *Alice:* $s \in S$.

Alice: *Chooses a random* $r \in S$ *and sends Bob* $x = r * p$ *and* $x' = r * p'$.

Bob: *Chooses a random bit* c *and sends it to Alice.*

Alice: *If* $c = 0$, *sends* $y = r$ *(then Bob checks:* $x = y * p$ *and* $x' = y * p'$*);*

If $c = 1$, *sends* $y = r * s$ *(then Bob checks:* $x' = y * x$*).*

Now, one can use the shifted conjugacy operation as the $*$ operation on B_∞ in order to get a LD-system. So, in this way, one can achieve an authentication scheme on the braid group with a non-trivial operation [21].

Remark 10.4. *For attacking the Shifted Conjugacy Search Problem, one cannot use the Summit Sets theory, since it is not a conjugation problem anymore. Nevertheless, one still can apply on it the length-based attack, since it is still an equation with* x. *So it is interesting to check (see also* [21]*):*

1. **Cryptanalysis direction:** *What is the success rate of a length-based attack on this scheme?*
2. **Cryptanalysis direction:** *Can one develop a theory for the Shifted Conjugacy Search Problem which will be parallel to the Summit Sets theory?*

(3) **Cryptosystem direction:** *Can one suggest a LD-system on the braid group, which will be secure for the length-based attack?*
(4) **Cryptosystem direction:** *Can one suggest a LD-system on a different group, which will be secure?*

10.1.2. *A cryptosystem based on the shortest braid problem.* A different type of problem consists in finding the shortest words representing a given braid (see Dehornoy [20, Section 4.5.2]). This problem depends on a given choice of a distinguished family of generators for B_n, e.g., the σ_i's or the band generators of Birman-Ko-Lee.

We consider this problem in B_∞ which is the group generated by an infinite sequences of generators $\{\sigma_1, \sigma_2, \dots\}$ subject to the usual braid relations.

The *Minimal Length Problem* (or *Shortest Word Problem*) is:

Problem 10.5. *Starting with a word w in the $\sigma_i^{\pm 1}$'s, find the shortest word w' which is equivalent to w, i.e., that satisfies $w' \equiv w$.*

This problem is considered to be hard due to the following result of Paterson and Razborov [69]:

Proposition 10.6. *The Minimal Length Problem is co-NP-complete.*

This suggests introducing new schemes in which the secret key is a short braid word, and the public key is another longer equivalent braid word. It must be noted that the NP-hardness result holds in B_∞ only, but it is not known in B_n for fixed n.

The advantage of using an NP-complete problem lies in the possibility of proving that some instances are difficult; however, from the point of view of cryptography, the problem is not to prove that some specific instances are difficult (worst-case complexity), but rather to construct relatively large families of provably difficult instances in which the keys may be randomly chosen.

Based on some experiments, Dehornoy [20] suggests that braids of the form $w(\sigma_1^{e_1}, \sigma_2^{e_2}, \dots, \sigma_n^{e_n})$ with $e_i = \pm 1$, i.e., braids in which, for each i, at least one of σ_i or σ_i^{-1} does not occur, could be relevant.

The possible problem of this approach is that the shortest word problem in B_n for a fixed n is not so hard. In B_3, there is polynomial-time algorithms for the shortest word problem (see [4] and [83] for the presentation by the Artin generators and [84] for the presentation by band generators). Also, this problem was solved in polynomial time in B_4 for the presentation by the band generators [46]. For small fixed n, Wiest [83] conjectures for an efficient algorithm for finding shortest representatives in B_n. Also, an unpublished work [36] indicates that a heuristic algorithm based on a random walk on the Cayley graph of the braid group might give good results in solving the shortest word problem.

In any case, a further research is needed here in several directions:

(1) **Cryptosystem direction**: Can one suggest a cryptosystem based on the shortest word problem in B_∞, for using its hardness due to Paterson-Razborov?
(2) **Cryptanalysis direction**: What is the final status of the shortest word problem in B_n for a fixed n?

10.1.3. *Cycling problem as a potential hard problem.* In their fundamental paper, Ko et al. [48] suggested some problems which can be considered as hard problems, on which one can construct a cryptosystem. One of the problems is the *Cycling Problem*:

Problem 10.7. *Given a braid y and a positive integer t such that y is in the image of the operator $\mathbf{c}^t$. Find a braid x such that $\mathbf{c}^t(x) = y$.*

Maffre, in his thesis [57], shows that the Cycling Problem for $t = 1$ has a very efficient solution. That is, if y is the cycling of some braid, then one can find x such that $\mathbf{c}(x) = y$ very fast.

Following this result, Gebhardt and Gonzales-Meneses [39] has shown that the general Cycling Problem has a polynomial solution. The reason for that is the following result: The cycling operation is surjective on the braid group [39]. Hence, one can easily find the tth preimage of y under this operation.

10.2. **Alternative distributions.** For overcoming some of the attacks, one can try to change the distribution of the generators. For example, one can require that if the generator σ_i appears, then in the next place we give more probability for the appearance of $\sigma_{i\pm 1}$. In general, such a situation is called a *Markov walk*, i.e. the distribution of the choice of the next generator depends on the choice of the current chosen generator.

A work in this direction is the paper of Maffre [59]. After suggesting a deterministic polynomial algorithm that reduces the Conjugacy Search Problem in braid group (by a partial factorization of the secret), he proposes a new random generator of key which is secure against his attack and the one of Hofheinz and Steinwandt [43].

10.3. **Cryptosystems based on different non-commutative groups.** The protocols presented here for the braid groups can be applied to other non-commutative groups, so the natural question here is:

Problem 10.8. *Can one suggest a different non-commutative group where the existing protocols on the braid group can be applied, and the cryptosystem will be secure?*

Some suggestions were given in this direction. As we already saw in the previous section, some cryptosystems and cryptanalysis is based on the Thompson group.

We survey here some more suggestions.

10.3.1. *Miller groups.* For example, Mahalanobis [60] suggested some Diffie-Hellman-type exchange key on Miller Groups [63], which are groups with an abelian automorphism group.

Acknowledgements

First, I wish to thank Patrick Dehornoy who has let me use his survey on braid group cryptography [20]. I have followed his presentation in many places.

I also want to thank Joan Birman, Rainer Steinwandt and Bert Wiest for useful communications. I owe a special thanks to Boaz Tsaban who has helped me in some stages of the preparation of these lecture notes.

References

[1] I. Anshel, M. Anshel, B. Fisher and D. Goldfeld, *New key agreement protocols in braid group cryptography*, CT-RSA 2001 (San Francisco), Springer Lect. Notes in Comp. Sci. **2020** (2001), 1–15.

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

[3] E. Artin, *Theory of Braids*, Ann. Math. **48** (1947), 101–126.

[4] M. A. Berger, *Minimum crossing numbers for 3-braids*, J. Phys. A: Math. Gen. **27** (1994), 6205–6213.

[5] S. Bigelow, *The Burau representation is not faithful for $n = 5$*, Geometry and Topology **3** (1999), 397–404.

[6] S. Bigelow, *Braid groups are linear*, J. Amer. Math. Soc. **14** (2001), 471–486.

[7] J. Birman, *Braids, Links, and Mapping Class Groups*, Annals of Math. Studies **82**, Princeton Univ. Press (1975).

[8] J.S. Birman, V. Gebhardt and J. Gonzalez-Meneses, *Conjugacy in Garside Groups I: Cyclings, Powers, and Rigidity*, Groups, Geometry and Dynamics, to appear.

[9] J.S. Birman, V. Gebhardt and J. Gonzalez-Meneses, *Conjugacy in Garside Groups II: Structure of the Ultra Summit Set*, Groups, Geometry and Dynamics, to appear.

[10] J.S. Birman, V. Gebhardt and J. Gonzalez-Meneses, *Conjugacy in Garside Groups III: periodic braids*, J. Algebra, to appear.

[11] J. Birman, K. Ko, and S. Lee, *A new approach to the word problem in the braid groups*, Adv. Math. **139** (1998), 322–353.

[12] J. Birman, K. Ko, and S. Lee, *The infimum, supremum, and geodesic length of a braid conjugacy class*, Adv. Math. **164** (2001), 41–56.

[13] X. Bressaud, *A normal form for braid groups*, Journal of Knot Theory and its Ramifications, to appear. (Online: http://iml.univ-mrs.fr/~bressaud/math/HDR/Papiers/TressesFinale3Mars2007.pdf).

[14] W. Burau, *Uber Zopfgruppen und gleichsinnig verdrilte Verkettungen*, Abh. Math. Sem. Hanischen Univ. **11** (1936), 171–178.

[15] J.W. Cannon, W.J. Floyd and W.R. Parry, *Introductory notes to Richard Thompson's groups*, L'Enseignement Mathematique **42** (1996), 215–256.

[16] J. Cha, J. Cheon, J. Han, K. Ko, and S. Lee, *An efficient implementation of braid groups*, AsiaCrypt 2001, 144–156, Springer Lect. Notes in Comp. Sci. **2048**, 2001.

[17] J. Cheon and B. Jun, *A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem*, CRYPTO 2003, 212–225, Springer Lect. Notes in Comp. Sci. **2729**, 2003.

[18] M. Cho, D. Choi, K. Ko, and J. Lee, *New signature scheme using conjugacy problem*, (online: http://eprint.iacr.org/2002/168/).

[19] P. Dehornoy, *A fast method for comparing braids*, Adv. Math. **125** (1997), 200–235.

[20] P. Dehornoy, *Braid-based cryptography*, Contemp. Math. **360** (2004), 5–33.

[21] P. Dehornoy, *Using shifted conjugacy in braid-based cryptography*, preprint (2006) (online: CS arXiv: cs.CR/0609091).

[22] P. Dehornoy, *Alternating normal forms for braids and locally Garside monoids monoids*, preprint (2007) (online: Math. arXiv: math.GR/0702592).

[23] P. Dehornoy, *Efficient solutions to the braid isotopy problem*, preprint (2007) (online: Math. arXiv: math.GR/0703666).

[24] P. Dehornoy, *Convergence of handle reduction of braids*, preprint (online: http://www.math.unicaen.fr/~dehornoy/Surveys/Dhn.pdf).

[25] P. Dehornoy, I. Dynnikov, D. Rolfsen and B. Wiest, *Why are braids orderable?*, Panoramas & Synthèses vol. **14**, Soc. Math. France (2002).

[26] P. Dehornoy, M. Girault and H. Sibert, *Entity authentication schemes using braid word reduction*, Proc. Internat. Workshop on Coding and Cryptography, 153–164, Versailles, 2003.

[27] W. Diffie, and M. Hellman, *New directions in cryptography*, IEEE Trans. on Inf. Theory **22** (1976), 644–654.

[28] I. Dynnikov, *On a Yang-Baxter mapping and the Dehornoy ordering*, Uspekhi Mat. Nauk **57**(3) (2002), 151–152; English translation: Russian Math. Surveys, 57-3 (2002).

[29] I. Dynnikov and B. Wiest, *On the complexity of braids*, preprint (online: http://hal.archives-ouvertes.fr/hal-00001267/en/).

[30] E. A. Elrifai and H.R. Morton, *Algorithms for positive braids*, Quart. J. Math. Oxford **45**(2) (1994), 479–497.

[31] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson and W. Thurston, *Word Processing in Groups*, Jones & Bartlett Publ. (1992).

[32] N. Franco and J. Gonzales-Meneses, *Conjugacy problem for braid groups and Garside groups*, J. Algebra **266** (2003), 112–132.

[33] D. Garber, S. Kaplan and M. Teicher, *A new algorithm for solving the word problem in braid groups*, Adv. Math. **167**(1) (2002), 142–159.

[34] D. Garber, S. Kaplan, M. Teicher, B. Tsaban and U. Vishne, *Probabilistic solutions of equations in the braid group*, Adv. Appl. Math. **35** (2005), 323–334.

[35] D. Garber, S. Kaplan, M. Teicher, B. Tsaban and U. Vishne, *Length-based conjugacy search in the braid group*, Contemp. Math. **418** (2006), 75–87.

[36] D. Garber, S. Kaplan and B. Tsaban, *A heuristic approach to the shortest word problem*, unpublished.

[37] F. A. Garside, *The braid group and other groups* Quart. J. Math. Oxford; 20-78; (1969), 235–254.

[38] V. Gebhardt, *A new approach to the conjugacy problem in Garside groups*, J. Algebra **292**(1) (2005), 282–302.

[39] V. Gebhardt and J. Gonzales-Meneses, *On the cycling operation in braid groups*, preprint (2007) (online: http://www.arxiv.org/abs/math.GT/0704.2600).

[40] J. Gonzales-Meneses, *The n-th root of a braid is unique up to conjugacy*, Alg. Geom. Topo. **3** (2003), 1103–1118.

[41] A. Groch, D. Hofheinz and R. Steinwandt, *A practical attack on the root Problem in Braid Groups*, Contemp. Math. **418** (2006), 121–131.

[42] M. Hock and B. Tsaban, *A betterlength function for Artin's braid groups*, preprint (2006) (online: http://www.arxiv.org/abs/math/0611918).

[43] D. Hofheinz and R. Steinwandt, *A practical attack on some braid group based cryptographic primitives*, PKC 2003; Springer Lect. Notes in Comp. Sci. **2567** (2002), 187–198.

[44] J. Hughes, *A linear algebraic attack on the AAFG1 braid group cryptosystem*, ACISP 2002; Springer Lect. Notes in Comp. Sci. **2384**, (2002), 176–189.

[45] J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Inst. for Math. and its Applic. 2000 (online: http://www.ima.umn.edu/preprints/apr2000/1696.pdf).

[46] E. S. Kang, K. H. Ko and S. J. Lee, *Band-generator presentation for the 4-braids*, Topology Appl. **78** (1997), 39–60.

[47] C. Kassel and V. Turaev, *Braid groups*, Springer, 2007.

[48] 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*; Crypto 2000; Springer Lect. Notes in Comp. Sci. **1880** (2000), 166–184.

[49] N. Koblitz and A. Menezes, *A survey of public-key cryptosystems*, SIAM Review **46** (2004), 599–634.

[50] D. Krammer, *Braid groups are linear*, Ann. Math. **151** (2002) 131–156.

[51] S. Lal and A. Chaturvedi, *Authentication schemes using braid groups*, preprint (2005) (online: http://arxiv.org/pdf/cs.CR/0507066).

[52] S.J. Lee and E. Lee, *Potential Weaknesses of the Commutator Key Agreement Protocol Based on Braid Groups*, Springer Lect. Notes in Comp. Sci **2332** (2002), 14–28.

[53] E.K. Lee and S.J. Lee, *Stable super summit sets in Garside groups*, preprint (2006) (online: Math. arXiv: math.GT/0602582).

[54] E.K. Lee and S.J. Lee, *Translation numbers in a Garside group are rational with uniformly bounded denominators*, preprint (2006) (online: Math. arXiv: math.GT/0604061).

[55] E.K. Lee and S.J. Lee, *Some power of an element in a Garside group is conjugate to a periodically geodesic element*, preprint (2006) (online: Math. arXiv: math.GN/0604144).

[56] D.D. Long and M. Paton, *The Burau representation is not faithful for $n \geq 6$*, Topology **32**(2) (1993), 439–447.

[57] S. Maffre, *Conjugaison et Cyclage dans les groupes de Garside, applications cryptographiques*, Ph.D. Lab. LACO, 2005 (online: http://www.unilim.fr/theses/2005/sciences/2005limo0028/maffre_s.pdf)

[58] S. Maffre, *Reduction of conjugacy problem in braid groups, using two Garside structures*, WCC 2005, 214–224.

[59] S. Maffre, *A weak key test for braid-based cryptography*, Designs, Codes and Cryptography **39** (2006), 347–373.

[60] A. Mahalanobis, *Diffie-Hellman Key Exchange Protocol and non-abelian nilpotent groups*, preprint (online: http://eprint.iacr.org/2005/110).

[61] F. Matucci, *The Shpilrain-Ushakov protocol for Thompson's Group F is always breakable*, preprint (2006) (online: http://www.arxiv.org/math/0607184).

[62] F. Matucci and M. Kassabov, *The simultaneous conjugacy problem in Thompson's group F*, preprint (2006) (online: https://www.arxiv.org/math/0607167). .

[63] G.A. Miller, *A non-abelian group whose group of isomorphism is abelian*, Messenger Math. **43** (1913), 124–125.

[64] J.A. Moody, *The Burau representation of the braid group is unfaithful for large n*, Bull. Amer. Math. Soc. New Ser. **25**(2) (1991), 379–384.

[65] J. Moody, *The faithfulness question for the Burau representation*, Proc. Amer. Math. Soc. **119**(2) (1993), 671–679.

[66] H.R. Morton, *The multivariable Alexander polynomial for a closed braid*, Low dimensional topology (Funchal, 1998), 167–172, Contemp. Math. **233**, Amer. Math. Soc., Providence, RI, 1999.

[67] A.G. Myasnikov, V. Shpilrain and A. Ushakov, *A practical attack on some braid group based cryptographic protocols*, Crypto 2005, Springer Lect. Notes in Comp. Sci. **3621** (2005), 86–96.

[68] A.G. Myasnikov, V. Shpilrain and A. Ushakov, *Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol*, in: PKC 2006, Springer Lect. Notes in Comp. Sci. **3958** (2006), 302–314.

[69] M.S. Paterson and A.A. Razborov, *The set of minimal braids is co-NP-complete*, J. Algorithms **12**(3) (1991), 393–408.

[70] R. L. Rivest, A. Shamir, and L. Adleman, *On Digital Signatures and Public Key Cryptosystems*, Commun. Ass. Comp. Mach. **21** (1978), 120–126.

[71] D. Rolfsen, *Minicourse on the braid groups*, preprint (2006). (http://www.math.ubc.ca/~rolfsen/reprints.html)

[72] D. Ruinskiy, A. Shamir and B. Tsaban, *Length-based cryptanalysis: The case of Thompson's group*, J. Math. Crypt., to appear (online: http://www.arxiv.org/cs/0607079).

[73] D. Ruinskiy, A. Shamir and B. Tsaban, *Cryptanalysis of group-based key agreement protocols using subgroup distance functions*, PKC07, Springer Lect. Notes in Comp. Sci. **4450** (2007), 61–75.

[74] V. Shpilrain, *Assessing security of some group based cryptosystems*, Contemporary Mathematics **360** (2004), 167–177.

[75] V. Shpilrain and A. Ushakov, *Thompson's group and public key cryptography*, Lecture Notes Comp. Sc. **3531** (2005), 151–164.

[76] V. Shpilrain and A. Ushakov, *A new key exchange protocol based on the decomposition problem*, Contemp. Math. **418** (2006), 161–167.

[77] V. Shpilrain and G. Zapata, *Combinatorial group theory and public key cryptography*, Applicable Algebra in Engineering, Communication and Computing **17** (2006), 291–302.

[78] V. Shpilrain and G. Zapata, *Using the subgroup membership search problem in public key cryptography*, Contemp. Math. **418** (2006), 169–179.

[79] H. Sibert, P. Dehornoy and M. Girault, *Entity authentication schemes using braid word reduction*, Discrete Appl. Math. **154**(2) (2006), 420–436.

[80] V.M. Sidelnikov, M.A. Cherepnev and V.Y. Yashcenko, *Systems of open distribution of keys on the basis of noncommutative semigroups*, Russ. Acad. Nauk Dokl. **332-5** (1993); English translation: Russian Acad. Sci. Dokl. Math. **48-2** (1994), 384–386.

[81] B. Tsaban, *On an authentication scheme based on the root problem in the braid group*, preprint (2005) (Online: http://arxiv.org/ps/cs.CR/0509059).

[82] V. Turaev, *Faithful linear representations of the braid groups*, Séminaire Bourbaki, Vol. 1999/2000. Astérisque No. **276** (2002), 389–409.

[83] B. Wiest, *An algorithm for the word problem in braid groups*, preprint (2002) (online: http://arXiv.org/abs/math.GT/0211169).

[84] P. Xu, *The genus of closed 3-braids*, Journal of Knot Theory and its Ramifications, **1**(3) (1992), 303–326.

Department of Applied Mathematics, School of Sciences, Holon Institute of Technology, 52 Golomb Street, PO Box 305, 58102 Holon, Israel

E-mail address: garber@hit.ac.il