Contemporary Mathematics

Braid-based cryptography

Patrick DEHORNOY

Abstract. We survey some of the recently developed cryptographic schemes involving Artin's braid groups, as well as the attacks against these schemes. We also point out some hints for future work.

Contents:

Braid-based cryptography appeared recently: its birthdate can be traced back to the pioneering papers [**3**] and [**42**] published by Anshel, Anshel & Goldfeld in 1999 and by Ko, Lee, Cheon, Han, Kang and Park in 2000. The subject has met with

2000 *Mathematics Subject Classification.* 20F36, 94A60, 94A62, 68P25.

Key words and phrases. braid group, normal form, handle reduction, word problem, conjugacy problem.

The author thanks Hervé Sibert for his help; many ideas, facts, and statistical data come from the latter's Ph.D. thesis [**52**].

a quick success, probably due in part to the intuitive and appealing character of braid groups. Recently, several attacks lowered the initial enthousiasm, and some authors even announced the premature death of the subject. The aim of this text is to give a sketchy description of braid-based cryptography, including a discussion of the attacks, and an analysis of why, in our opinion, these attacks do not condemn the subject, but only show that further investigation is needed.

1. Background on braids

Artin's braid groups are infinite non-commutative groups. They are eligible for applications because there exist efficient ways of specifying braids and computing with them.

1.1. Braid groups. Braid groups appear in several *a priori* unrelated frameworks, and they admit many equivalent definitions. For our current purpose, it is convenient to start with an explicit presentation.

DEFINITION 1.1. For $n \geqslant 2$, the braid group B_n is defined by the presentation

$$\langle \sigma_1, \dots, \sigma_{n-1} \,;\, \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geqslant 2,\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ for } |i-j| = 1 \rangle. \tag{1.1}$$

For each n, the identity mapping on $\{\sigma_1, \dots, \sigma_{n-1}\}$ induces an embedding of B_n into B_{n+1}, so that the groups B_n naturally arrange into an inductive system of groups with increasing complexity. Note that B_2 is an infinite cyclic group, *i.e.*, is isomorphic to the group of integers: the "arithmetic" of braids can be seen as an extension of the usual arithmetic of integers equipped with addition.

We refer the reader to any textbook about braids, for instance [**8**] or [**15**], for a geometric interpretation for the elements of B_n as n-strand braids in the usual sense. The principle is to associate with every word in the letters $\sigma_i^{\pm 1}$ the plane diagram obtained by concatenating the elementary diagrams of Figure 1 corresponding to the successive letters. Such a diagram can be seen as a plane projection of a three-dimensional figure consisting on n disjoint curves connecting the points $(1,0,0), \dots, (n,0,0)$ to the points $(1,0,1), \dots, (n,0,1)$ in $\mathbb{R}^3$, and, then, the relations (1.1) are a translation of ambient isotopy, *i.e.*, the result of continuously moving the curves without moving their ends and without allowing them to intersect. It is easy to check on Figure 2 that each relation in (1.1) corresponds to such an isotopy; the converse implication, *i.e.*, the fact that the projections of isotopic three-dimentional geometric braids always can be encoded in words connected by (1.1) was proved by E. Artin in [**4**].

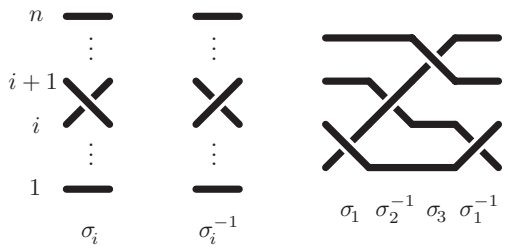

FIGURE 1. Braid diagrams associated with σ_i, σ_i^{-1}, and with $\sigma_1\sigma_2^{-1}\sigma_3\sigma_1^{-1}$

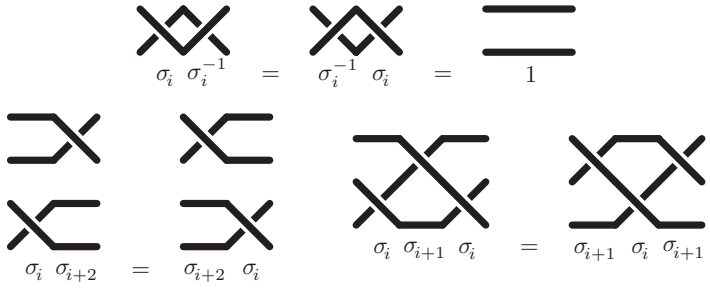

FIGURE 2. Geometric interpretation of the braid relations

The geometric interpretation makes it clear that mapping the braid σ_i to the transposition that exchanges i and $i+1$ induces a surjective homomorphism of the braid group B_n onto the symmetric group S_n. Under this homomorphism, here denoted π, a braid b is mapped to the permutation f of $1, \dots, n$ such that the strand that finishes at position i in a diagram associated with b begins at position $f(i)$. The kernel of π is a normal subgroup of B_n generated by the braids σ_i^2 and their conjugates, which corresponds to the fact that the symmetric group S_n admits the Coxeter presentation obtained from (1.1) by adding the relations $\sigma_i^2 = 1$.

1.2. The greedy normal form. 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. In the current case, a word on the letters $\sigma_1^{\pm 1}, \dots, \sigma_{n-1}^{\pm 1}$ will be called an *n-strand braid word*. So, by definition, every n-strand braid is an equivalence class of n-strand braid words under the congruence denoted $\equiv$ generated by Relations (1.1). If the braid b is the equivalence class of the braid word w, we say that w is a *representative* of b. Note that a braid always admits infinitely many representative braid words, since all words $w\sigma_1^k\sigma_1^{-k}$ represent the same braid.

A good solution for working with a presented group consists in selecting a distinguished word in each equivalence class, *i.e.*, in defining a *normal form*. Then working with normal words is exactly equivalent to working with the elements of the groups themselves. In the case of braid groups, there exists such a normal form, namely the *greedy normal form* of [**23**]. It grew up from the work of Garside [**29**], and several variants have been described in several partly independent papers [**20, 1, 56, 22, 23**]. Normal braid words are as follows. Let us denote by B_n^+ the submonoid of B_n generated by $\sigma_1, \dots, \sigma_{n-1}$. The elements of B_n^+ are called *positive* braids. Garside proved that B_n^+ admits, as a monoid, the presentation (1.1), that B_n is a group of fractions of B_n^+, *i.e.*, every braid in B_n can be expressed as $b_1^{-1}b_2$ with b_1, b_2 in B_n^+, and that B_n^+ equipped with the (left) divisibility relation is a lattice, *i.e.*, any two positive braids admit a greatest common left divisor (gcd) and a least common right multiple (lcm)—as well as a right gcd and a left lcm.

Let Δ_n denote the positive braid of B_n^+ inductively defined by

$$\Delta_1 = 1, \qquad \Delta_{n+1} = \Delta_n\sigma_n\sigma_{n-1}\dots\sigma_1. \tag{1.2}$$

Then Δ_n is a (left and right) common multiple for all generators σ_i in the monoid B_n^+, and it follows that the powers of Δ_n can be used as universal denominators: every

braid in B_n can be written as $\Delta_n^k b$ with k in $\mathbb{Z}$ and b in B_n^+, a decomposition that is unique when we require k to be maximal.

Let us say that a positive braid b in B_n^+ is *simple* if it is a left divisor of Δ_n, *i.e.*, if we have $\Delta_n = bc$ for some positive braid c. If b is any positive braid, then there exists a maximal simple braid b_1 that divides b, namely the left gcd of b and Δ_n. We can then write $b = b_1 b'$, and repeat the operation, *i.e.*, look for the maximal simple left divisor b_2 of b', *etc.*

Let us say that a finite sequence $(k; b_1, \dots, b_r)$, with k an integer and $b_1, \dots, b_r$ some simple braids, is *normal* if, for each i, the braid b_i is distinct of 1 and Δ_n and it is the maximal simple left divisor of $b_i b_{i+1} \dots b_r$. Then the result is:

PROPOSITION 1.2. *Every braid b in B_n admits a unique decomposition of the form $\Delta_n^k b_1 \cdots b_r$ with $(k; b_1, \dots, b_r)$ a normal sequence.*

In this situation, we say that $(k; b_1, \dots, b_r)$ is the normal form of b. The parameter r is called the *complexity* (or *canonical length*) of b, and denoted by $\mathrm{cpty}(b)$.

In order to obtain a unique braid word representative for every braid, it now suffices to fix, for every simple braid, a positive word representing it. One shows that the restriction of the mapping π to simple braids is a bijection, *i.e.*, the simple braids of B_n are in one-to-one correspondence with the permutations of $1, \dots, n$. In order to select one distinguished braid word representative for each simple braid, one can iteratively construct for each permutation f a positive braid word $\widehat{f}$ so that $\pi(\widehat{f})$ equals f, and then, for each simple braid b, use $\widehat{\pi(b)}$ as a distinguished word representing b. The conclusion is that every braid admits a unique representative of the form $(\widehat{\Delta_n})^k \widehat{\pi(b_1)} \dots \widehat{\pi(b_r)}$ with $(k; b_1, \dots, b_r)$ a normal sequence: this word will be called the *normal word* representing the considered braid.

EXAMPLE 1.3. Let us consider the braid of Figure 1, namely $b = \sigma_1 \sigma_2^{-1} \sigma_3 \sigma_1^{-1}$. To find the normal form of b, we first have to find the maximal k such that $\Delta_4^k b$ belongs to B_4^+. As b admits a decomposition with as many negative and positive letters, b does not belong to B_4^+, *i.e.*, $k = 0$ is impossible. Now, one has $\Delta_4 b = \sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_3$, hence $b = \Delta_4^{-1}\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_3$, and $k = -1$ works. Then, the maximal simple left divisor of $\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_3$, *i.e.*, the left gcd of that braid and Δ_4, is $\sigma_1\sigma_3\sigma_2\sigma_3$, and we have $\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_3 = (\sigma_1\sigma_3\sigma_2\sigma_3)(\sigma_2\sigma_3)$. As $\sigma_2\sigma_3$ is simple, the normal form of b is $(-1; \sigma_1\sigma_3\sigma_2\sigma_3, \sigma_2\sigma_3)$, and its complexity is 2. Finally, assuming that $\sigma_1\sigma_3\sigma_2\sigma_3$ and $\sigma_3\sigma_2\sigma_1\sigma_3\sigma_2\sigma_3$ are the chosen normal words representing the simple braids $\sigma_1\sigma_3\sigma_2\sigma_3$ and Δ_4, respectively, $\sigma_3^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_3^{-1}\sigma_1\sigma_3\sigma_2\sigma_3\sigma_2\sigma_3$ is the unique normal word representing b.

1.3. Implementation of braids. In order to implement integers, one needs to specify them using numbers, *i.e.*, finite sequences of digits. Similarly, braids have to be encoded in finite sequences of letters in order to be implemented. The normal braid words of Section 1.2 are a natural solution, but it is not the only possible one. The question of how braids are specified is important in practice, because it has a direct influence on what we call the size of the braid and on the notion of a random braid drawing—as well as on the practical efficiency of the procedures.

1.3.1. *Normal braid words.* The first, natural possibility is to use normal braid words. The practical benefit of working with unique representatives is that establishing a possible braid equality is algorithmically easy: if we assume that two braids b_1, b_2 are represented by normal words w_1, w_2 respectively, then the braid equality $b_1 = b_2$ is equivalent to the word equality $w_1 = w_2$.

On the other hand, algebraic operations in B_n have a non-negligible algorithmic cost. With the same notations as above, the normal word representing the braid $b_1 b_2$ is not the word $w_1 w_2$ in general, but it is the unique normal word that is equivalent to $w_1 w_2$. So we need an algorithm taking an arbitrary braid word w as an input and returning the unique normal word that is equivalent to w. In the current case, such an algorithm exists and, for the typical braid size that will be considered in the sequel (a few dozens strands, a few hundreds or thousands of crossings), computing the normal form has a moderate cost (typically one second). The normal form is computed iteratively: assuming that w is a normal word, one computes the normal form of $w\sigma_i^{\pm 1}$—or, more generally, of $wu^{\pm 1}$ where u represents a simple braid—by pushing to the left the new letter $\sigma_i^{\pm 1}$ through w (in the process, $\sigma_i^{\pm 1}$ or $u^{\pm 1}$ may change and be replaced with another simple words $v^{\pm 1}$). The point is that the greedy normal form is associated with an automatic structure [**23, 56, 16**], which implies that adding one more letter—or one more simple word—can be done in a time cost that is linear in the length of w. So, finally, the normal word equivalent to a word w can be determined in a time cost that is quadratic in the length of w.

1.3.2. *Normal braid sequences.* We mentioned that there exists a bijection between the simple braids of B_n and S_n. It follows that, instead of selecting a distinguished braid word for each simple braid, we can also use a permutation. Thus, in order to specify the braid whose normal decomposition is $(k; b_1, \dots, b_r)$, we can use the sequence $(k; \pi(b_1), \dots, \pi(b_r))$ consisting of one integer and a finite sequence of permutations. For simplicity, such data will be called a *braid sequence* in the sequel, and we shall naturally say that a braid sequence $(k; f_1, \dots, f_r)$ is normal if the corresponding sequence $(k; \widehat{f_1}, \dots, \widehat{f_r})$ is.

Then, by construction, every braid is represented by a unique normal braid sequence. For instance, we have seen that the braid b of Example 1.3 admits the normal decomposition $(-1; \sigma_1\sigma_3\sigma_2\sigma_3, \sigma_2\sigma_3)$. The permutations associated with the simple braids $\sigma_1\sigma_3\sigma_2\sigma_3$ and $\sigma_2\sigma_3$ are $(2, 4, 3, 1)$ and $(1, 3, 4, 2)$ respectively—we denote by $(f(1), \dots, f(n))$ the permutation f—so the normal braid sequence associated with b is $(-1; (2, 4, 3, 1), (1, 3, 4, 2))$.

Using normal braid sequences is similar to using normal braid words: checking equality is straightforward, but computing product or inverse requires being able to determine the unique normal sequence that is equivalent to an arbitrary sequence. The algorithm to do this is analogous to the one used for braid words: as mentioned above, in the case of words, the principle is to incorporate one after the other the subwords representing simple braids, so, in the current framework, to incorporate one after the other the permutations. A practical implementation is given in [**12**].

1.3.3. *Arbitrary braid words.* Still another possibility is to use arbitrary braid words—or braid sequences—instead of normal ones. The advantage of using arbitrary braid words is clear: if we no longer require that all braid words are normal, the cost of computing the product becomes negligible, as the product of the braids represented by two words w_1, w_2 can be represented by the concatenated word $w_1 w_2$, without further reduction to the normal form. On the other hand, the price to pay is equally clear: when we have to compare two braids, it no longer suffices to compare letter by letter the unique braid words or sequences that represent them, but we need a specific algorithm to decide whether the braid word equivalence $w_1 \equiv w_2$ holds. Of course, checking whether the normal form of $w_1^{-1} w_2$ is empty is one

possibility—but, then, the benefit with respect to using normal words everywhere is not obvious.

We shall return to this option in Section 4.3 below.

1.3.4. *Data size and random drawing.* Although using (normal) words and using (normal) sequences are equivalent approaches in theory, they are not completely equivalent in practice, because they lead to different evaluations of the size of the data, and to different notions of random drawing. For instance, the braids σ_1^k and Δ_n^k are represented by the normal sequences

$$(0; (2\ 1\ 3 \dots n), \dots, (2\ 1\ 3 \dots n)) \text{ and } (0; (n\ n{-}1\ \dots 2\ 1), \dots, (n\ n{-}1\ \dots 2\ 1))$$

respectively, each of length k, hence by data of global size kn (if we do not take into account the size of the integers, otherwise a factor $\log n$ is to be added). On the other hand, these braids are represented by normal words of length k and $kn(n-1)/2$, respectively. We see that the evaluations may differ by a large constant factor, typically $4,950$ for $n = 100$ (even more if we take into account the size of the integers and add $\log n$ factors).

This phenomenon induces some ambiguity about what randomly drawing a braid means. A most natural choice is to draw random braid words, or random braid sequences, and then compute the associated normal word or sequence. One problem is that, as Table 1 shows, when we draw random words of length ℓ, the average complexity of the normal words equivalent to these words is not even proportional to their length, implying a possible bias in the statistical results that can be deduced from such drawings. Table 2 shows that the bias may be smaller with braid sequences, as, in the average, putting to normal form a random braid sequence does not change its length significantly.

	$\ell = 100$	$\ell = 200$	$\ell = 500$	$\ell = 1,000$	$\ell = 2,000$
$n = 10$	15.2	28.3	67	132	265
$n = 20$	10.1	17.2	28.2	73.2	142
$n = 50$	6.8	10.2	19.6	34.9	63.4
$n = 100$	5.43	7.62	13.2	21.2	36.4
$n = 200$	4.59	6.12	9.63	14.1	23.1

TABLE 1. Average complexity of the braid specified by a random n-strand braid word of length ℓ

	$r = 10$	$r = 20$	$r = 50$	$r = 100$	$r = 200$
$n = 10$	9.01	17.4	42.6	84.6	168
$n = 20$	9.99	19.99	49.96	99.92	199.81
$n = 50$	10	20	50	100	200
$n = 100$	10	20	50	100	200
$n = 200$	10	20	50	100	200

TABLE 2. Average complexity of the braid specified by a random n-strand braid sequence of length r: one almost always finds r.

Another possibility is to generate normal braid words or normal braid sequences exclusively. This is not difficult in practice: indeed, for a finite sequence of simple braids $(b_1, \dots, b_r)$ to be normal turns out to be a local property, in the sense that it only requires that each one of the pairs (b_1, b_2), (b_2, b_3), $\dots$, (b_{r-1}, b_r) be normal. Thus one can directly draw normal words (or sequences) by restricting the drawing process after each factor b_i to those simple braids b such that (b_i, b) is normal—which is easy. Of course, the resulting distribution of normal words or sequences need not coincide with the one originating from normalizing random words or random sequences.

REMARK 1.4. The family $\{\sigma_1, \dots, \sigma_{n-1}\}$ is not the only interesting family of generators for the group B_n. In [**9**] and [**10**], Birman, Ko, and Lee consider a larger family of generators for B_n, namely the $\binom{n}{2}$ braids a_{rs} defined for $1 \leqslant r < s \leqslant n$ by

$$a_{rs} = \sigma_{s-1} \cdots \sigma_{r+1} \sigma_r \sigma_{r+1}^{-1} \cdots \sigma_{s-1}^{-1}. \tag{1.3}$$

Most of the Garside theory for the submonoid B_n^+ of B_n generated by the σ_i's also works for the submonoid generated by the a_{rs}'s. Again there exists a unique normal form defined in terms of the divisors of some distinguished element. The point is that, in the case of B_n^+, we have $n-1$ generators and a distinguished element Δ_n whose length in the generators σ_i is $\binom{n}{2}$, while, in the case of the Birman–Ko–Lee monoid, we have $\binom{n}{2}$ generators and a distinguished element whose length in terms of the generators a_{rs} is $n-1$. This explains why th Birman–Ko–Lee monoid is often called the *dual* monoid for B_n. So other possibilities for implementing braids consist in using normal words in the generators a_{rs}'s or using the associated sequences of permutations, which turn out to make a proper subset of S_n (non-overlapping permutations). Also see [**5, 19, 16, 17, 34**] for more about the possible Garside structures of the braid groups.

2. Braid-based schemes

We now review (some of) the existing braid-based cryptographical schemes. Nearly all schemes proposed so far relie on the supposed difficulty of problems related with conjugacy in B_n, and, in particular, of what will be called the Conjugator Search Problem in B_n. For $n \geqslant 3$, the braid group B_n is non-commutative, as, *e.g.*, σ_1 and σ_2 do not commute; it is even strongly non-commutative, in that its centre is very small, namely cyclic, generated by the unique element Δ_n^2. Moreover, the centraliser of a braid b is typically generated by b and Δ_n^2, *i.e.*, it has the least possible value. It follows that the conjugacy operation is not trivial in B_n. One says that two braids p, p' are *conjugate* if we have $p' = sps^{-1}$ for some braid s. The *Conjugacy Problem* is the question of algorithmically recognizing whether two braids p, p' are conjugate, and the *Conjugator Search Problem* is the related question of finding a conjugating braid for a pair (p, p') of conjugate braids, *i.e.*, finding s satisfying $p' = sps^{-1}$.

As will be explained in Section 3, the Conjugacy Problem and the Conjugator Search Problem in B_n are decidable, but the only solutions proposed so far for the latter have a high algorithmic complexity, at least in the worst case. Typically, in the state-of-the-art, it seems infeasible to solve the Conjugator Search Problem for appropriately chosen pairs of 50-strand braid words of length 1,000: as it is

easy to work with such data, in particular to put them into normal form, starting from the Conjugator Search Problem appears a reasonable basis for designing cryptographical schemes.

2.1. Key exchange. Various cryptographical problems have been addressed. The key exchange problem is as follows: two entities, traditionally called A(lice) and B(ob), wish to agree on a common secret, in such a way that an intruder observing the communication cannot deduce any useful information about the common secret.

2.1.1. *The Anshel-Anshel-Goldfeld scheme.* The following scheme was proposed by Anshel & *al.* in [**3**] and [**2**]. Note that the scheme can be used in any group where 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 public key consists of two sets of braids, say $p_1, \dots, p_\ell$, $q_1, \dots, q_m$, in B_n. The secret key of Alice is a word u on ℓ letters and their inverses, the secret key of Bob is a word v on m letters and their inverses. The exchanges are as follows:

• A computes the braid $s = u(p_1, \dots, p_\ell)$, and uses it to compute the conjugates $q'_1 = sq_1s^{-1}$, ... , $q'_m = sq_ms^{-1}$; she sends $q'_1, \dots, q'_m$;
• B computes the braid $r = v(q_1, \dots, q_m)$, and uses it to compute the conjugates $p'_1 = rp_1r^{-1}$, ... , $p'_\ell = rp_\ell r^{-1}$; he sends $p'_1, \dots, p'_\ell$;
• A computes $t_{\text{A}} = s\, u(p'_1, \dots, p'_\ell)^{-1}$;
• B computes $t_{\text{B}} = v(q'_1, \dots, q'_m)\, r^{-1}$.
The common key is $t_{\text{A}} = t_{\text{B}}$.

Indeed, we find:

$$t_{\text{A}} = s\, u(p'_1, \dots, p'_\ell)^{-1} = s\, r\, u(p_1, \dots, p_\ell)^{-1}\, r^{-1}$$
$$= s\, r\, s^{-1}\, r^{-1} = s\, v(q_1, \dots, q_m)\, s^{-1}\, r^{-1} = v(q'_1, \dots, q'_m) r^{-1} = t_{\text{B}}$$

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 $(p_1, p'_1), \dots, (p_\ell, p'_\ell)$ 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. Provided this problem is hard enough for the considered braids, knowing the pairs $(p_1, p'_1), \dots, (p_\ell, p'_\ell)$ does not enable an intruder to find the value of the secret key r, and, similarly, knowing the pairs $(q_1, q'_1), \dots, (q_m, q'_m)$ does not enable him to find s. In [**2**], it is suggested to work in B_{80} with $\ell = m = 20$ and short initial braids p_i, q_j of length 5 or 10.

2.1.2. *A Diffie–Hellman-like scheme.* Although braid groups are not commutative, they contain 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. So, if we 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 $m = \lfloor n/2 \rfloor$, every braid in LB_n commutes with every braid in UB_n.

This observation is exploited in the following Diffie–Hellman-like key exchange scheme proposed by Ko & *al.* in [**42**]. This scheme also had been proposed independently by Sidelnikov & *al.* in [**54**] in the context of a general, unspecified

noncommutative semigroup with difficult conjugacy problem, but the authors do not mention braid groups explicitly.

Here the public key consists in one braid p in B_n. The secret key of Alice is a braid s in LB_n; the secret key of Bob is a braid r in UB_n. The exchanges are as follows:

- A computes the conjugate $p' = sps^{-1}$, and sends it to B;
- B computes the conjugate $p'' = rpr^{-1}$, and sends it to A;
- A computes $t_{\text{A}} = s\,p''\,s^{-1}$;
- B computes $t_{\text{B}} = r\,p'\,r^{-1}$.

The common key is $t_{\text{A}} = t_{\text{B}}$.

Indeed, because s and r commute, we find:

$$t_{\text{A}} = s\,p''\,s^{-1} = s\,r\,p\,r^{-1}\,s^{-1} = r\,s\,p\,s^{-1}\,r^{-1} = r\,p'\,r^{-1} = t_{\text{B}}.$$

Here also 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–Hellmann-like Conjugacy Problem*: Given a braid p in B_n, and the braids $p' = sps^{-1}$ and $p'' = rpr^{-1}$, where s and r lie in LB_n and UB_n respectively, find the braid $rp'r^{-1}$, which is also $sp''s^{-1}$. The connection between the Diffie–Hellmann-like Conjugacy Problem and the variant of the Conjugator Search Problem where the conjugating braid is to be found in LB_n is similar to the one between the Diffie-Helmann Problem and the Discrete Log Problem. Provided that problem is difficult enough, knowing the pairs (p, p') and (p, p'') is not sufficient to find the mixed intermediate conjugate $rp'r^{-1} = sp''s^{-1}$. In [**12**], it is suggested to work in B_{80} with braids specified using (normal) sequences of length 12, *i.e.*, sequences of 12 permutations.

2.2. Enciphering–deciphering. Here the problem is as follows: Bob wishes to send Alice a message m, and he can use Alice's public key to encipher his message. Alice must be able to retrieve Bob's original message using her private key, but an intruder watching the communication should not.

The following scheme is proposed by Ko & *al.* in [**42**]. The notations are those of the previous key exchange scheme. In addition, we assume that h is a *collision-free one-way hash function* of B_n to $\{0,1\}^N$, *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)—see Section 4.4 below.

We start with p in B_n and s in LB_n. Alice's public key is the pair (p, p'), with $p' = sps^{-1}$; Alice's private key is s. In order to send the message m_{B}, which we assume lies in $\{0,1\}^N$, using $\oplus$ for the Boolean operation "exclusive-or" (*i.e.*, the sum in $\mathbb{Z}/2\mathbb{Z}$):

- B chooses a random braid r in UB_n and he sends the encrypted text $m'' = m_{\text{B}} \oplus h(rp'r^{-1})$ together with the auxiliary datum $p'' = rpr^{-1}$;
- A computes $m_{\text{A}} = m'' \oplus h(sp''s^{-1})$.

Then we have $m_{\text{A}} = m_{\text{B}}$, *i.e.*, Alice retrieves Bob's original message.

Indeed, because the braids r and s commute, we have

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

and, therefore, $m_{\text{A}} = m_{\text{B}} \oplus h(rp'r^{-1}) \oplus h(rp'r^{-1}) = m_{\text{B}}$. Here also the security is based on the difficulty of the Diffie–Hellmann-like Conjugacy Problem in B_n: owing to the hypotheses on h, being able to break the system means being able to retrieve the common value of $rp'r^{-1}$ and $sp''s^{-1}$ from the pairs (p, p') and (p, p''). The recommended parameters are the same as in Section 2.1.2.

2.3. Authentication. Now the problem is: Alice (the prover) wishes to prove her identity to Bob (the verifier), *i.e.*, she wishes to prove that she knows some private (secret) key without enabling an intruder watching the communication to deduce anything about her private key.

2.3.1. *A Diffie–Hellman-like scheme.* The following challenge–response scheme, mentioned in [**53**], is a direct adaptation of the previous scheme by Ko & *al.* [**42**].

The keys are as above: the public key is a pair of conjugate braids (p, p') in B_n with $p' = sps^{-1}$, while Alice's private key is the braid s used to conjugate p into p'; we assume that s belongs to LB_n. We still use h for a collision-free one-way hash function on B_n. The exchanges are as follows:

- B chooses a random braid r in UB_n, and he sends the challenge $p'' = rpr^{-1}$ to A;
- A sends the response $y = h(sp''s^{-1})$;
- B checks $y = h(rp'r^{-1})$.

A correct answer of the prover A leads to acceptation by the verifier B because, as above, the braids r and s commute and, therefore, we have $rp'r^{-1} = sp''s^{-1}$. On the other hand, acceptation occurs only if the response y of A satisfies $y = h(rp'r^{-1})$, hence, by the hypotheses about h, only if A has been able to find the value of $rp'r^{-1}$ from the knowledge of p, p', p'', *i.e.*, only if A has been able to solve the corresponding instance of the Diffie–Hellmann-like Conjugacy Problem in B_n. Once again, the security of the scheme relies on the difficulty of that problem.

2.3.2. *A Fiat-Shamir-like scheme.* Another authentication scheme is proposed by Sibert & *al.* in [**53**]. This scheme is reminiscent of the Fiat–Shamir scheme, and it involves repeating several times a three-pass challenge–response step.

As before, the public keys are a pair of conjugate braids (p, p') with $p' = sps^{-1}$, while s, the conjugating braid, is Alice's private key. In contrast to the previous schemes, both p and s lie in B_n, *i.e.*, we do not assume that s lies in any particular subgroup such as LB_n or UB_n. We still assume that h is a collision-free one-way hash function on B_n. The authentication procedure consists in repeating k times the following three exchanges:

- A chooses a random braid r in B_n, and she sends the *commitment* $x = h(rp'r^{-1})$;
- B chooses a random bit c and sends it to A;
- For $c = 0$, A sends $y = r$, and B checks $x = h(yp'y^{-1})$;
- For $c = 1$, A sends $y = rs$, and B checks $x = h(ypy^{-1})$.

If A knows s and answers correctly, she is accepted by B: for $c = 0$, we have $x = h(yp'y^{-1})$ directly, while, for $c = 1$, we have $ypy^{-1} = (rs)p(rs)^{-1} = rp'r^{-1}$, hence $x = h(ypy^{-1})$. On the other hand, if A is dishonest, she can cheat and send a correct answer in both cases: in the case $c = 0$, it suffices that A keeps track of r

and sends coherent answers; in the case $c = 1$, it suffices that the commitment x is chosen so as to anticipate the equality $x = h(ypy^{-1})$, which is easy as well. But a cheater cannot choose his commitment so as to answer correctly in both cases: if he anticipates $c = 0$, the probability of answering correctly for $c = 1$ is negligible, and, symmetrically, if he anticipates $c = 1$, the probability of answering correctly for $c = 0$ is negligible. So, globally, a cheater has no more than 0.5 chance to be accepted. Thus, by repeating the exchanges k times, we can make the probability that a cheater be accepted as small as $1/2^k$.

The security of the scheme relies on the difficulty of the original Conjugator Search Problem in B_n. Indeed, for A to be accepted with a probability higher than 0.5 means being able to answer in both cases $c = 0$ and $c = 1$, hence knowing y, y' satisfying $x = h(ypy^{-1})$ and $x = h(y'p'y'^{-1})$. As h is supposed to be collision-free, *i.e.*, virtually injective, this gives $ypy^{-1} = y'p'y'^{-1}$, hence $(y'^{-1}y)p(y'^{-1}y)^{-1} = p'$: so A must know a solution to the Conjugator Search Problem for (p, p'). It is suggested to work in B_{50} (as no subgroup is involved) with braids specified by words of length 512—with possible additional requirements on p (see Section 4.1).

2.4. Signature. The last problem we consider is signature. The problem is for A to send B a (clear or ciphered) message together with a signature proving the origin of the message. Note that each signature scheme leads to an authentication scheme: in order to authenticate A, B can send her a message and require that A signs this message.

Two braid-based signature schemes are introduced by Ko & *al.* in [**41**]: the second one is the scheme recommended by the authors, but the first is simpler and the common principle is more easily readable. These schemes use the supposed complexity gap between the Conjugacy Problem and the Conjugator Search Problem: as will be seen in Section 3.3 below, for medium values of n (typically $n = 20$), using a linear representation makes it possible to decide whether two braids in B_n are conjugate without enabling one to determine a conjugating braid when it exists.

As before, the public keys are a pair of conjugate braids (p, p') with $p' = sps^{-1}$, while s, the conjugating braid, is Alice's private key; the braids p and s belong to B_n. We use H for a one-way collision-free hash function from $\{0, 1\}^*$ to B_n—so a function going in the converse direction of the hash functions involved in Sections 2.2 and 2.3; we use $\sim$ for conjugacy in B_n.

The first scheme is as follows:

- A signs the message m with $q' = sqs^{-1}$, where $q = H(m)$;
- B checks $q' \sim q$ and $p'q' \sim pq$.

If A uses the secret key s, we have $q' = sqs^{-1}$ and $p'q' = spqs^{-1}$, so the signature is accepted. Conversely, the security of the scheme relies on the difficulty of the following *Matching Conjugate Search Problem*: Assuming $p' \sim p$ and q in B_n, find q' satisfying both $q' \sim q$ and $p'q' \sim pq$. It is shown in [**41**] that the problem is at least as difficult as the Conjugator Search Problem for the pair (p, p').

A possible weakness of the previous scheme lies in that repeated uses disclose many conjugate pairs (q_i, q'_i) associated with the common conjugator s, leading to possible attacks (*cf.* Section 3). To avoid this, the authors of [**41**] modify the

scheme by incorporating an additional random braid—here we again use h for a one-way hash function from B_n to $\{0,1\}^*$:

- A chooses a random braid r in B_n;
- A signs the message m with the triple (p'', q'', q'), where $p'' = rpr^{-1}$, $q = H(mh(p''))$, $q'' = rqr^{-1}$, and $q' = rs^{-1}qsr^{-1}$;
- B checks $p'' \sim p$, $q'' \sim q' \sim q$, $p''q'' \sim pq$, and $p''q' \sim p'q$.

The security analysis is similar to that of the first scheme. The authors recommend to work in B_{20} with keys specified by braid sequences of length 3.

The previous schemes are elegant and the arguments developed in [**41**] provide a convincing evidence of their security. However, it has been noted that it could be theoretically possible to generate fake signatures: even if the latter are associated with no message, this is usually considered to be a potential weakness.

3. Attacks against the braid schemes

Several attacks against the braid-based schemes were proposed recently [**36, 38, 28, 37, 35, 47, 13**], and we shall now sketch some of them. As the security of the schemes depends on the difficulty of the Conjugator Search Problem in B_n and its variants, it is not surprising that the attacks mainly aim at solving this problem. At least three different strategies can be considered: (*i*) using a solution to the Conjugacy Problem; (*ii*) using a probabilistic approach inside B_n; (*iii*) using auxiliary groups, typically linear representations.

3.1. Solutions to the Conjugacy Problem. The most obvious way to attack the braid schemes is to solve the Conjugacy Problem in B_n, which is known to be possible after Garside's seminal work [**29**]. Successive refinements of the method have greatly improved its algorithmic efficiency.

3.1.1. *The Super Summit Set.* Garside's method for solving the Conjugacy Problem in B_n consists in associating with every braid b a distinguished finite set of conjugates of b called its *Summit Set*. El-Rifai and Morton showed in [**22**] how to replace the Summit Set with one of its subsets called the *Super Summit Set* (*SSS*), which is smaller and therefore easier to determine.

Definition 3.1. The *super summit set* $SSS(b)$ of a braid b is the set of all conjugates of b with the minimal possible complexity.

Proposition 3.2. [**22**] *For every braid b, the set $SSS(b)$ is finite, and it is algorithmically computable.*

By construction, two braids b, b' are conjugate if and only if their *SSS*'s coincide—or, equivalently, if and only if these sets intersect. So, the previous result implies the decidability of the Conjugacy Problem in B_n. Actually, more precise results are known.

Definition 3.3. Assume that b is a braid in B_n, and that $(k; b_1, \dots, b_r)$ is its normal form. Then the braids $\partial_+(b)$ and $\partial_-(b)$ are defined by

$$\partial_+(b) = \Delta_n^k b_2 \dots b_r \phi_n^k(b_1), \quad \partial_-(b) = \Delta_n^k \phi_n^k(b_r) b_1 \dots b_{r-1}, \tag{3.1}$$

where ϕ_n is the flip automorphism that maps σ_i to σ_{n-i} for each i; we say that $\partial_+(b)$ (*resp.* $\partial_-(b)$) is obtained by *cycling* (*resp. decycling*) from b.

By construction, the braids $\partial_+(b)$ and $\partial_-(b)$ are conjugates of b. The point is that, if b is a braid in B_n that does not belong to its *SSS*, *i.e.*, that does not have the minimal complexity in its conjugacy class, then by cycling or decycling b at most $n(n-1)/2$ times, one can find a conjugate of b with a strictly smaller complexity. So, by repeating the operation, we obtain after finitely many steps a conjugate b^* of b lying in $SSS(b)$.

The next result is that, if b lies in its *SSS*, then the whole set $SSS(b)$ can be obtained by saturating $\{b\}$ under conjugation by simple braids: starting with $\{b\}$, we successively consider all conjugates sbs^{-1} where s is a simple braid; we keep those conjugates b' that have the same complexity as b, and throw the others away, until no more conjugate can be added.

A complete procedure for deciding whether two braids b, b' are conjugate is therefore (see Figure 3):

- Using cycling and decycling, find a conjugate b^* of b lying in $SSS(b)$;
- Using cycling and decycling, find a conjugate b'^* of b' lying in $SSS(b')$;
- Determine $SSS(b)$ by saturating $\{b^*\}$ under simple conjugation;
- Then b and b' are conjugate if and only if b'^* belongs to $SSS(b)$.

By keeping track of the conjugating braids used at each step, one does not only decide whether b and b' are conjugate, but one also obtains a conjugator if it exists, *i.e.*, if b and b' are conjugate. So, in this way, one solves both the Conjugacy Problem and the Conjugator Search Problem of B_n.

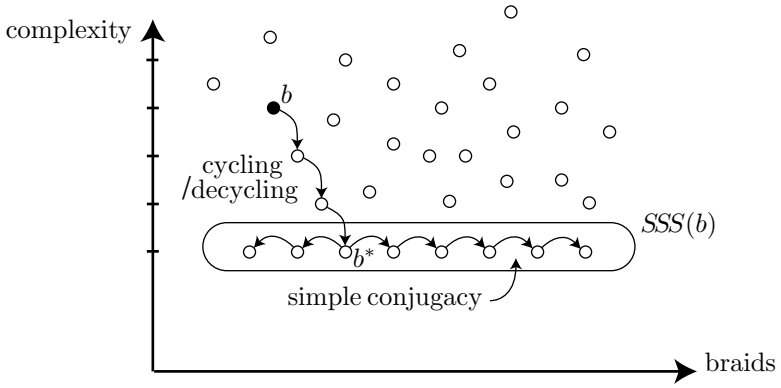

FIGURE 3. Solving the conjugacy problem: going to the *SSS* and then enumerating it (the points represent the conjugates of b)

As for complexity, since cycling and decycling a constant number of times guarantees that the normal length will decrease if it can, finding a conjugate in the *SSS* has a linear cost with respect to the complexity of the initial braid. Then there remains the cost of enumerating the set $SSS(b)$. It is shown in [**27**] that this can be done in a number of steps which is linear in the size of $SSS(b)$, and with a reasonable constant coefficient. The problem is that there exist $n!$ simple braids in B_n so, for typical values like $n = 50$, it is hopeless to enumerate all of them. What Franco and Gonzales-Meneses show in [**27**] is that, if conjugating b by simple braids s_1, s_2

gives elements in the *SSS*, so does conjugating b by the gcd of s_1 and s_2 and, as a consequence, it is sufficient to consider conjugation by what they call *minimal* simple elements, which are at most n in number in the case of B_n.

3.1.2. *The Ultra Summit Set.* Quite recently, V. Gebhardt proposed a new refinement in [**30**]. This refinement consists in replacing the *SSS* with a still smaller set called the Ultra Summit Set (*USS*).

Let us consider the action of cycling on the *SSS*: starting with a braid b in its *SSS*, iterated cycling in *SSS* need not return to the initial b, but it certainly becomes eventually periodic. We can therefore partition the *SSS* into several orbits, each of which consists of a cyclic part, and of tails (Figure 4). In [**30**], Gebhardt defines the *Ultra Summit Set* (*USS*) to be the union of the cyclic parts of the orbits. By definition, the *USS* is a subset of the *SSS*, and Gebhardt shows that the *USS* can be used instead of the *SSS*: as with the *SSS*, it is easy to find an element of the *USS*, and, then, the whole *USS* can be computed using minimal simple elements. The point is that the size of the *USS* is often much smaller than that of the *SSS*, typically its size can be linear w.r.t. the length of the initial braid, while that of the *SSS* is exponential. In such cases, the *USS* can be determined quickly, and the Conjugacy Problem is solved. Nothing is proved so far, but the average complexity of the method might turn to be polynomial.

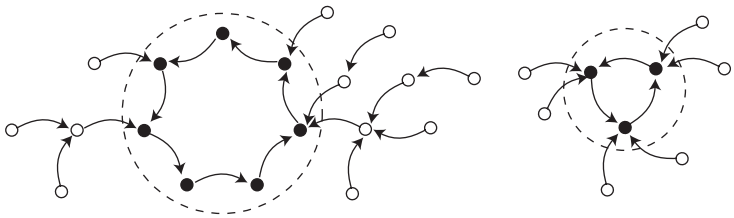

FIGURE 4. Action of cycling inside the *SSS*; the elements of the *USS* are in black

3.1.3. *Derived attacks.* The previous exact solutions to the Conjugacy Problem can be used to directly attack the cryptographical schemes of Section 2.

As for the solution based on the *SSS*, statistics suggest that, for 5 strands and more, the size of the latter set is exponential in the length (and the complexity) of the braid. The typical size of the *SSS* of a 80-strand braid specified by a braid sequence of length 12—the size considered in [**42**]—is (much) more than 2^{100}, which makes an exhaustive search infeasible.

The recent improvement of the method based on the *USS* might change the situation: Gebhardt reports that the *USS* of 100-strand braids specified by random braid words of length 1,000 can be computed effectively, and the associated Conjugator Search Problem can therefore be solved. Here the significant parameter is the size of the *SSS* or of the *USS*, and whether the attack is feasible directly depends on this size.

REMARK 3.4. In the schemes we described, the secret key is a braid s conjugating p to p'. We know that, for a given pair (p, p') of conjugate braids, the conjugating braid s is not unique: any other braid s' such that $s^{-1}s'$ belongs to the commutator of p is also a solution, and, when we solve the Conjugator Search Problem, there is no reason to find the precise s that was used. However, this is

sufficient to break the schemes: in each case, an intruder only needs to know one braid s' conjugating p to p', no matter whether it is the one initially used.

3.2. Attacks based on length. Besides using an exact solution to the Conjugator Search Problem, one can also attack the braid schemes using a probabilistic heuristic approach: whenever the probability of success is more than negligible, this may be enough to endanger the security of the scheme. The length-based attacks of [**28, 45, 32, 35**] belong to this family.

The common principle of these attacks is to try to retrieve a conjugator for a pair (p, p') by starting with p', which is supposed to be derived from p, and iteratively conjugating p' into a new braid $tp't^{-1}$ so that the length [**28**] or the complexity [**45, 32, 35**] of $tp't^{-1}$ is minimal.

In [**28, 45, 32**], one simply checks whether the new conjugate $tp't^{-1}$ happens to be equal to p. The attack is specially active against the key exchange scheme of [**3, 2**] based on the Multiple Conjugator Search Problem, because, in this case, one knows several pairs of conjugate braids associated with the same conjugating braid. Non-negligible success rates are reported in the cases of 50-strand braids and more, at least for some choices of the number of initial braids p_i's and q_j's.

The attack of [**35**] is similar, but it includes one more step and it is therefore more powerful. As above, we start with a conjugate p' of p, and the first step is to find a conjugate $tp't^{-1}$ of p' with minimal complexity. Then, instead of checking whether, by chance, $tp't^{-1}$ is merely equal to p, one looks whether the "permutation distance" between $tp't^{-1}$ and p is at most 1, *i.e.*, one tries to find a permutation f such that $tp't^{-1}$ is equal to the simple conjugate $\widehat{f}p\widehat{f}^{-1}$. Finding the possible permutation is easy, as this amounts to solving the Conjugator Search Problem in the symmetric group S_n. With this improvement, the success rate of the attack reaches 99% in the case of the Anshel & *al.* key agreement scheme of [**2**] in B_{80} with $\ell = m = 20$ and initial braids p_i, q_j of length 5 or 10, even when on restricts to pure braids—braids such that the associated permutation is the identity—as was suggested in [**45**] to improve security. The success rate against the key exchange scheme of [**42**] is slightly lower, but it remains about 80% for (pure) braids in B_{80} and a secret conjugator s obtained by drawing 12 random permutations—as suggested in [**42**].

3.3. Attacks based on linear representations. A third way to attack the braid-based cryptographical schemes is to use linear representations of the braid groups, *i.e.*, to map the braid groups into groups of matrices. As the Conjugacy Problem in a linear group is easy, we may think of solving the Conjugacy Problem of B_n in this way.

3.3.1. *The Burau representation.* The best known linear representation of the braid group B_n is the Burau representation, a linear representation with values in $GL_n(\mathbb{Z}[t, t^{-1}])$. The image of σ_i is the $n \times n$ matrix obtained from the identity matrix by replacing the central $(i, i+1)$-square with $\left(\begin{smallmatrix} 1-t & t \\ 0 & 1 \end{smallmatrix}\right)$. The Burau representation of B_n is known to be unfaithful for $n \geqslant 5$ [**6**], but the kernel is very small, and the probability that different braids admit the same Burau image is negligible.

In [**36**] and [**45**], the Anshel & *al.* scheme of [**2**] is attacked using the Burau representation: starting with one or several pairs of conjugate braids (p, p') associated with the same conjugating braid s, it is easy to compute their 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. But, in [**36**], one takes advantage of the fact that, in the scheme of [**2**], the secret data are the words u and v, which can be guessed in the linear groups as well.

In [**41**], a similar approach is developed not to attack any scheme, but, on the contrary, to solve the Conjugacy Problem of B_n for medium values of n (typically $n = 20$) and implement the signature schemes of Section 2.4. The solution is probabilistic, and it relies on evaluating the Burau matrices at a convenient number of values of the indeterminate t chosen in $\mathbb{Z}/N\mathbb{Z}$ with N large enough.

3.3.2. *The Lawrence–Krammer representation.* The Lawrence–Krammer representation is another linear representation of B_n, which is faithful [**7, 43**]. It associates with every n strand braid a matrix of size $\binom{n}{2}$ with entries in a 2-variable Laurent polynomial ring $\mathbb{Z}[t^{\pm1}, q^{\pm1}]$. In [**13**], Cheon and Jun develop an attack against the scheme of [**42**] based on the Lawrence–Krammer representation: as above, it is easy to compute the images of the involved braids in the linear group and to solve the Conjugacy Problem there, but there is no way to lift the solution back to the braid groups in general. But, once again, what is needed here is not a complete solution to the Conjugator Search Problem, but only a solution to the derived Diffie–Hellman-like Conjugacy Problem: knowing 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, the authors obtain a solution with a polynomial complexity and they show that, for the sizes considered in [**42**], the procedure is not infeasible.

3.4. Are these attacks dangerous? As they stand, the previous attacks seem very dangerous, and nearly all schemes considered so far have been attacked, for typical values of the parameters which were previously suggested to be secure. It is certainly possible to increase complexity, and therefore to improve security, by increasing the size of the parameters, but, then, the computation times will become very large, and the benefit of using braids will quickly vanish.

However, it is not certain that the attacks are really so dangerous, and that they virtually dismiss braid-based cryptography. Indeed, it seems easy to understand two specific reasons that make these attacks work, namely the use of variants of the Conjugator Search Problem, and the way the keys are generated.

3.4.1. *Variants of the Conjugator Search Problem.* The previous attacks work mainly because the involved schemes use specific variants of the Conjugator Search Problem, namely the Multiple Conjugator Search Problem and the Diffie–Hellman-like Conjugacy Problem, rather than the problem itself: except for the Hofheinz–Steinwandt and for the direct attack using the *USS*, the attacks do not work against the Conjugator Search Problem, and, *e.g.*, the authentication scheme of [**53**] is not threatened. It is true that, as it stands, the scheme of [**2**] seems to be the most seriously endangered, but, in any case, the conclusion could be that this specific scheme is to be modified or rejected, but not that the whole braid-based cryptography is endangered.

3.4.2. *Why are length-based attacks efficient?* Length-based attacks, and, in particular, the one of [**35**], seem very efficient, and the latter works against instances of the Conjugator Search Problem that *a priori* seem generic. Actually, this is not the case, and the success of this attack only reflects the way keys are generated.

It is easy to understand how length-based attacks work. For all braids b_1, b_2, we have the relation

$$0 \leqslant \mathrm{cpty}(b_1 b_2) \leqslant \mathrm{cpty}(b_1) + \mathrm{cpty}(b_2) \tag{3.2}$$

between the complexity of $b_1 b_2$ and those of b_1 and b_2. These inequalities are optimal: for instance, for $b_2 = b_1^{-1}$, the complexity of $b_1 b_2$ is 0. Now, Table 3 shows that, with probability virtually 1, we have the equality

$$\mathrm{cpty}(b_1 b_2) = \mathrm{cpty}(b_1) + \mathrm{cpty}(b_2). \tag{3.3}$$

This is natural, as the normal form of $b_1 b_2$ can be obtained from that of b_1 by successively appending the factors of the normal form of b_2: the factor Δ^k coming from b_2 moves across the whole normal form of b_1, but, then, the simple factors from b_2 have little chance to merge with those coming from b_1. So, in general, the normal form of $b_1 b_2$ is not obtained by simply appending the normal form of b_2 after that of b_1, but the cascades arising from pushing to the factors coming from b_2 quickly stop: typically, only the last 3 or 4 factors in the normal form of b_1 are modified, and the overall number of simple factors does not change.

	$r = 10$	$r = 20$	$r = 50$	$r = 100$	$r = 200$
$n = 10$	17.4	34.2	84.6	169	337
$n = 20$	19.99	39.97	99.91	199.8	399.7
$n = 50$	20	40	100	200	400
$n = 100$	20	40	100	200	400
$n = 200$	20	40	100	200	400

TABLE 3. Complexity of the product of two random n-strand braid sequences of length r: one almost always finds $2r$.

As a consequence, if p and s are random braids, one has, with a high probability

$$\mathrm{cpty}(sps^{-1}) = \mathrm{cpty}(p) + 2\,\mathrm{cpty}(s)\,: \tag{3.4}$$

the complexity of a random conjugate of p is higher than that of p, a paradoxical situation as conjugacy is a symmetric relation!

Let us consider the Heinzhof–Steinwandt attack. Assume that we start with a braid p lying in its *SSS* (a statistically almost sure case), and we generate p' by conjugating p using a random braid s. According to the previous observation, the complexity of the conjugates of p obtained by conjugating by the succesive letters of s is likely to increase at each step, and p' is likely to have a high complexity. Now, in the attack, we start from p' and lower its complexity so as to return to the *SSS*: the successive descending steps need not coincide with the ascending steps leading from p to p', and there is no reason that we finish with p exactly, but it is not surprising that the first element p'' of $SSS(p)$ on which we land often lies at a distance of at most one simple conjugacy from p, the hypothesis under which the attack works.

It is then clear that one can defeat the previous attack by requiring that p' has the same complexity as p does, *i.e.*, that p' lies in $SSS(p)$, and then checking that the distance between p' and p is more than one simple.

We see that, in this case, the crucial point is how keys are constructed. The situation is similar in the case of the recent results by Gebhardt on the Ultra

Summit Set: here an attack is possible only when the *USS* happens to be small enough, what need not always be the case (see below), and, again, the attack can be defeated by a proper choice of the keys.

So it seems that the proper conclusion to draw from the attacks is simply that all instances of the Conjugator Search Problem in B_n are not equally secure, and, that, in particular, it is not a good idea to generate a pair of conjugate braids by starting with a random braid p, and then deducing p' by drawing a random conjugator s and computing $p' = sps^{-1}$. Such a conclusion should not come as a surprise, as prescribing some constraints on the keys is a quite common situation: there exist few cryptosystems where keys can be chosen at random. So, even if some authors argued that the existing attacks definitely condemn braid-based cryptography, it seems more reasonable at the moment to only conclude that still more work is needed, in particular in the direction of constructing provably hard instances of the underlying problems—or in giving some evidence that such instances cannot be constructed.

4. Clues for further research

In this more prospective section, we mention some possible further lines of research. Some of them already received preliminary developments, but others seem to remain completely open.

4.1. Key generation. We saw that most of the attacks against the braid-based schemes take advantage of the way the keys are generated. So the main problem of current braid-based cryptography seems to construct provably hard instances of the involved problems, hence, at the moment, of the Conjugator Search Problem. Although this would be the only way of generating keys with convenient security properties, the question seems to have been little investigated so far, and we can only mention a few preliminary observations.

The first remark is that, even if the attacks of Section 3 did not exist, a random choice of the keys is certainly not possible: if we start with a braid p and conjugate it by s, then Table 4 shows that it is likely that a large prefix of the normal form of s remains directly readable in the normal form of sps^{-1}. This obvious point seems quite significant, and it is rather surprising that it is not addressed more systematically in the existing literature.

	$r = 10$	$r = 20$	$r = 50$	$r = 100$	$r = 200$
$n = 10$	3.0	8.8	27.5	58.5	121.5
$n = 20$	6.1	16.0	45.6	94.0	186.0
$n = 50$	7.4	17.4	47.4	97.5	197.0
$n = 100$	7.7	17.7	47.7	97.5	197.5
$n = 200$	7.8	17.8	47.8	98.0	198.0

TABLE 4. Number of initial simple factors in the normal form of s remaining unchanged in the normal form of sp when s, p are n-strand braids specified by r random permutations.

As explained above, a sufficient condition for generating instances of the Conjugator Search Problem that defeat the Hofheinz–Steinwandt attack of [**35**], and,

more generally, all length based attacks, is to choose the pair (p, p') in such a way that p and p' cannot be distinguished using the length or the complexity. The question of generating such pairs is discussed in [**52**], where Sibert proposes a symmetric construction in which p and p' are conjugates of a common third braid. Unfortunately, it seems difficult to design a similar construction for the Multiple Conjugator Search Problem or the Diffie–Hellman-like Conjugacy Problem: while it is easy to choose the conjugator s so that two conjugate braids p, sps^{-1} have the same complexity, doing it with several pairs simultaneously, or with s in some prescribed subgroup LB_n or UB_n, seems uneasy.

Next, it is natural to choose the conjugate braids p, p' so that they lie in their *SSS*, or, better *USS*, and such that the "simple distance" between p and p', *i.e.*, the minimal number of simple conjugacy steps inside *USS* needed to conjugate p to p', is large enough (in any case, at least 2 to defeat the Hofheinz–Steinwandt attack). This leads to the condition that the set $USS(p)$ itself be large enough, which is also required to defeat the direct attack consisting in enumerating $USS(p)$. So we need families of braids with a large *USS*. The work of Gonzales-Meneses in [**33**] suggests a possible connection with the Nielsen–Thurston classification of braids: if p is a pseudo-Anosov braid, then the sets $SSS(p)$ and $USS(p)$ seem to be (much) smaller than in the case of periodic and reducible braids. So choosing the initial braid p to be reducible could be a promising approach.

In a similar direction—but no precise connection with the size of the *USS* is known so far—experiments suggest that a good way of generating hard instances of the Conjugator Search Problem could be to use braids whose normal sequence involves short permutations, *i.e.*, permutations with a small number of inversions. While the average number of inversions in a random permutation of n objects is $n(n-1)/4$, it is suggested in [**52**] to use permutations with a typical number of $O(n)$ inversions. Such a choice appears as intermediate between starting with braid words, *i.e.*, with the braids σ_i, which correspond to permutations with one inversion, and starting with random braid sequences, which corresponds to permutations with an average number of $O(n^2)$ inversions.

We conclude with a minor remark whose only interest is to emphasize the role possibly played by the way braids are specified. Using braids with a sufficiently high complexity is a usual requirement, in particular for defeating attacks like the one of [**36**]. If the instances of the Conjugacy Problem are generated using normal sequences, the complexity can be given a prescribed value. On the other hand, if we start with arbitrary words of moderate size, we have no direct control of the complexity, which may be too small. Then, some tricks may be used to guarantee a high complexity, *e.g.*, requiring that, after each occurence of $\sigma_i^{\pm 1}$, the next letter is $\sigma_{i-1}^{\pm 1}, \sigma_i^{\pm 1}$, or $\sigma_{i+1}^{\pm 1}$. Also, randomly replacing some letters σ_i with σ_i^2 dramatically increases the complexity.

At least, these heuristic remarks should show how little is known, and how large is the work that remains to be done about these questions.

4.2. Random drawing and security proofs. The question of how to perform a random drawing in the group B_n, or in some prescribed subset of B_n relevant for a specific scheme, is both a practical and a theoretical issue, as it underlies all security arguments. We refer to [**11**] for a general study.

The main problem is that the group B_n is infinite, and, contrary to the case of integers, there is no known way of replacing B_n with a convenient finite quotient

as is done with $\mathbb{Z}$ and $\mathbb{Z}/N\mathbb{Z}$. It is not difficult to introduce probability measures on B_n by resorting to the notion of a random walk [**57**], but the notion of a uniform probability measure over B_n remains problematic. The natural requirement would be to resort to a probability measure μ on the subsets of B_n that is left-invariant, *i.e.*, such that $\mu(bA) = \mu(A)$ holds for every braid b and every subset A of B_n. A group G is said to be *amenable* if there exists at least one such measure on G. Now, for $n \geqslant 3$, the group B_n is not amenable: indeed, the subgroup of B_3 generated by σ_1^2 and σ_2^2 is free, and it is known that a group admitting at least one free subgroup of rank 2 cannot be amenable. So there exists no measure on B_n that is invariant under left (or right) multiplication.

As braids can be specified by (normal) words or sequences, we can easily fix an upper bound on the words or the codes we consider, and then define the notion of a random drawing over the braid words of length at most ℓ or over the braid sequences of length at most r. The problem is that the probability measures induced on the group B_n itself are not known, and there is no reason why these measures should be invariant under multiplication. In particular, there is no reason why randomly drawing braid words should induce the same probability distribution as randomly drawing braid sequences.

These questions are crucial in order to prove possible security results. For instance, it is considered desirable that an authentication scheme be zero-knowledge, this meaning that, using public data only, one can construct a probabilistic Turing machine able to simulate the instances of the communication between A and B in a way that cannot be distinguished from the real communication [**31, 26**]: if this can be done, no information about the secret data can be extracted from the exchanges performed in the scheme. Let us consider tthe Fiat–Shamir-like authentication scheme of Section 2.3.2. The question is to design a Turing machine generating the triples $(h(rp'r^{-1}), 0, r)$ (communication in the case $c = 0$), and $(h(rp'r^{-1}), 1, rs)$ (communication in the case $c = 1$), with the same probability distribution as in the real exchanges. Using a probability measure μ, we could argue as follows: for the triples of the first type, the Turing machine draws r μ-randomly, then it computes $x = h(rp'r^{-1})$ and outputs $(x, 0, r)$; for the triples of the second type, the machine draws r and s' μ-randomly, it computes $x = h(rs'p(rs')^{-1})$ and outputs $(x, 1, rs')$. Now we can claim that the distributions of rs (real communication) and rs' (simulated communication) coincide only if the measure μ is right-invariant. Unfortunately, no such μ exists, so, whatever the chosen probability is, the distributions of rs and rs' need not coincide, and the communication between A and B need not be correctly simulated in the case $c = 1$.

Failing theoretical results, it is sometimes possible to establish statistical evidence instead. It is shown in [**52**] that, for the scheme above, one can construct a probabilistic Turing machine simulating the exchanges between A and B in a way that is not theoretically indistinguishable from the real communication, but, at least, is computationally indistinguishable in some convenient sense. The criterion chosen by Sibert is not to compare the distributions of rs and rs' directly, but to investigate instead some parameters deduced from these distributions, namely the distance to the braid $p' = sps^{-1}$, which is the only public datum where s is involved. Here, the distance between two braids is defined to be the length of the maximal common subword in their normal forms. The results show that, both when braids are drawn using random braid words, and when they are drawn using random sequences of short permutations, the distances between rs (the "real" braid)

and sps^{-1} on the one hand, and between rs' (the simulated braid) and sps^{-1} on the other hand have the same distribution. This supports the opinion that the Fiat–Shamir-like scheme is close to zero-knwoledge in practice.

However, such statistical results are partial, and they cannot replace genuine proofs. The general question of what is a good random drawing on B_n remains widely open; see [**41**] for further investigation.

4.3. Using braid words and braid reduction. The greedy normal form of braids provides a convenient way for working with braids. However, we can also think of another way of implementing braids, namely by using arbitrary words (or sequences). As was explained in Section 1.3.3, the possible benefit would lie in the possibility of using direct comparison algorithms deciding $w \equiv w'$ more quickly than computing the normal form of $w^{-1}w'$ does. Such efficient algorithms do exist. Here we shall mention one of them, namely braid word reduction, but still other candidates can be found in [**59**] and in [**21**] (see [**18**]).

4.3.1. *Handle reduction.* Braid word reduction—also called *handle reduction*—is introduced in [**14**], and it can be seen as an extension of the free reduction process relevant for free groups. Free reduction consists in 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, thus in particular for the Artin presentation of B_n, but, as B_n is not a free group for $n \geqslant 3$, it does not solve the word problem: if freely reducing a braid word w leads to an empty word, then w represents 1 in B_n, but, conversely, there exist words that represent 1 in B_n but do not freely reduce to the empty word, *e.g.*, the freely reduced word $\sigma_1\sigma_2\sigma_1\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}$.

Handle reduction generalizes free reduction and involves not only patterns of the form xx^{-1} or $x^{-1}x$—*i.e.*, in our framework, $\sigma_i\sigma_i^{-1}$ or $\sigma_i^{-1}\sigma_i$—but also more general patterns of the form $\sigma_i \dots \sigma_i^{-1}$ or $\sigma_i^{-1} \dots \sigma_i$ with intermediate letters between the letters σ_i and σ_i^{-1}.

Definition 4.1. A *σ_i-handle* is a braid word of the form

$$w = \sigma_i^e \, w_0 \, \sigma_{i+1}^d \, w_1 \, \sigma_{i+1}^d \dots \sigma_{i+1}^d \, w_m \, \sigma_i^{-e}, \tag{4.1}$$

with $e, d = \pm 1$, $m \geqslant 0$, and $w_0, \dots, w_m$ containing no $\sigma_j^{\pm 1}$ with $j \leqslant i+1$. Then the *reduct* of w is defined to be

$$w' = w_0 \, \sigma_{i+1}^{-e}\sigma_i^d\sigma_{i+1}^e \, w_1 \, \sigma_{i+1}^{-e}\sigma_i^d\sigma_{i+1}^e \dots \sigma_{i+1}^{-e}\sigma_i^d\sigma_{i+1}^e \, w_m, \tag{4.2}$$

i.e., we delete the initial and final letters $\sigma_i^{\pm 1}$, and we replace each letter $\sigma_{i+1}^{\pm 1}$ with $\sigma_{i+1}^{-e}\sigma_i^{\pm 1}\sigma_{i+1}^e$.

Note that a braid word of the form $\sigma_i\sigma_i^{-1}$ or $\sigma_i^{-1}\sigma_i$ is a handle, and that reducing it means deleting it: handle reduction generalizes free reduction.

Reducing a braid word yields an equivalent braid word: as illustrated in Figure 5, the $(i+1)^{\text{th}}$ strand in a σ_i-handle forms a sort of handle, and reduction consists in pushing that strand so that it skirts above the next crossings instead of below. So, as in the case of free reduction, if there is a reduction sequence from a braid word w to the empty word, *i.e.*, a sequence $w = w_0, w_1, \dots, w_N = \varepsilon$ such that, for each k, the word w_{k+1} is obtained from w_k by replacing some handle of w_k by its reduct, then w is equivalent to the empty word, *i.e.*, it represents the unit

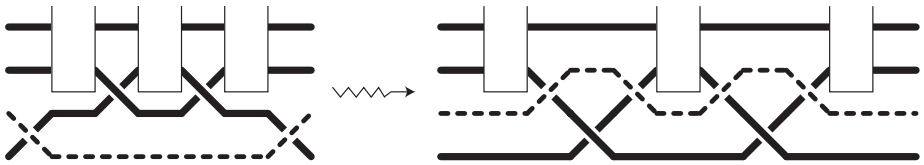

FIGURE 5. Reducing a σ_1-handle

braid. What is not obvious is the converse implication, as well as the fact that handle reduction must terminate in any case. This is what the following result from [**14**] claims:

PROPOSITION 4.2. *Assume that w is an n-strand braid word of length ℓ. Then every reduction sequence starting from w leads in at most $2^{\ell^4 n}$ steps to an irreducible word. Moreover, the empty word is the only irreducible word in its equivalence class, hence w represents the unit braid if and only if some—or, equivalently, any—reduction sequence starting from w finishes with the empty word.*

A braid word 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 Proposition 4.2 is very high, handle reduction is extremely efficient in practice: Table 5 shows that, for various sizes, handle reduction is always much quicker than reduction to the normal form. Also, reduction being a local procedure, the amount of memory needed to implement it is essentially what is needed to just store the braid under reduction. So, using arbitrary words together with handle reduction instead of normal words could be interesting in cryptographical applications when the computing resources of one of the entities are limited.

	$\ell = 64$	$\ell = 256$	$\ell = 1,024$	$\ell = 4,096$
$n = 4$	0.20 *vs.* 5.36	2.71 *vs.* 77.4	54.5 *vs.* 1,526	1,560 *vs.* 29,900
$n = 16$	0.03 *vs.* 8.65	0.45 *vs.* 105	10.2 *vs.* 1,378	1,635 *vs.* 21,990
$n = 64$	0.016 *vs.* 23.1	0.14 *vs.* 194	1.56 *vs.* 1,899	33 *vs.* 23,640

TABLE 5. Handle reduction *vs.* normal form: comparison of average CPU time in millisec. for random n-strand braid words of length ℓ; C++ implementation on AMD Duron 750 MHz by H. Sibert

It can be noted that braid reduction is specially efficient on certain families of words. For instance, we mentioned in Section 4.1 that, starting with a random word, we can increase the complexity by replacing some letters σ_i with their square σ_i^2. In doing so, the cost of reduction almost remains the same, so, for such words, the benefit of using reduction is obvious.

4.3.2. *Scrambling functions.* A new problem arises when we use arbitrary braid words. As was said above, the main advantage of the method is that the product of braids is implemented by word concatenation. But, for instance, if we have to conjugate (the braid represented by) w by (the braid represented by) u and we transmit the word uwu^{-1}, then the word u, and therefore the braid it represents, is immediately readable as a prefix of the word uwu^{-1}.

The problem is not new, as we mentioned in Section 4.1 that a similar phenomenon occurs when the normal form is used: in general, at least the beginning of the normal form of s is likely to be readable on the normal form of sps^{-1}.

As for words, a solution is to introduce some *scrambling process* that replaces the word uwu^{-1} by an equivalent word $f(uwu^{-1})$ from which u cannot be recovered. Some suggestions are formulated in [**53**]. One possibility consists in using handle reduction itself as a scrambling device, *i.e.*, to define $f(w)$ to be $\mathrm{red}(w)$, the result of applying handle reduction according to some fixed strategy until an irreducible word is obtained. In particular, it is proved that the words u and w can be chosen so that u and $f(uwu^{-1})$ have no prefix in common and, more precisely, no prefix of the word $f(uwu^{-1})$ represents a braid equal to one represented by a prefix of u.

A better choice could be to use the scrambling function defined by

$$f'(w) = \mathrm{red}(\sigma_n^{-1} \dots \sigma_1^{-1}\, \mathrm{red}(\sigma_1 \dots \sigma_n w \sigma_n^{-1} \dots \sigma_1^{-1})\, \sigma_1 \dots \sigma_n).$$

The scrambling effect of the double reduction is remarkable, as the expectation for the length of the longest common subword between a random word w and its image $f'(w)$ given in Table 6 is exactly the same as for independent random words.

	$n = 32$ $\ell = 4,096$	$n = 64$ $\ell = 8,192$
average length	4	4
maximal length	5	5

TABLE 6. Length of the longest common subword between w and $f'(w)$ for w a random n-strand braid word of length ℓ

Still another possibility is to resort to the braid word transformation called reversing [**17**]. One says that a braid word w *reverses* to w' if w' can be obtained from w by iteratively applying the following transformations:

- Replace some subword $\sigma_i^e \sigma_j^{e'}$ with $|i-j| \geqslant 2$ and $e, e' = \pm 1$ by $\sigma_j^{e'} \sigma_i^e$;
- Replace some subword $\sigma_i^e \sigma_j^e \sigma_i^e$ with $|i-j| = 1$ and $e = \pm 1$ by $\sigma_j^e \sigma_i^e \sigma_j^e$;
- Replace some subword $\sigma_i^e \sigma_j^{-e}$ with $|i-j| = 1$ and $e = \pm 1$ by $\sigma_j^{-e} \sigma_i^{-e} \sigma_j^e \sigma_i^e$;
- Delete some subword $\sigma_i \sigma_i^{-1}$ or $\sigma_i^{-1} \sigma_i$.

These reversing transformations map a braid word to an equivalent braid word, and handle reduction turns out to be a special case of reversing. A possible scrambling function is obtained by defining $g(w)$ to be the result of randomly reversing w until the length of the maximal common subword between w and $g(w)$ reaches some prescribed (small) value.

REMARK 4.3. Using a scrambling function to avoid that u be read from uwu^{-1} does not solve the problem that the normal word equivalent to u can be possibly

read from the normal word equivalent to uwu^{-1}: one is a word problem, while the other is a braid problem. In general, using arbitrary braid words may be a good choice for practical efficiency, but one should not forget that computing the normal form always remains possible, in particular for attacking the schemes.

4.4. Hash functions. In several schemes, a hash function is invoked. This means a function h from the considered braid group B_n to another set, *e.g.*, the set $\{0,1\}^*$ of all finite sequences of 0's and 1's. A hash function is called *ideal* if its values of h cannot be distinguished from the values given by a random oracle. Ideal hash functions are often required for security proofs. Constructing an ideal hash function is connected with the general question of finding a so-called hard-core predicate, *i.e.*, in the current case, of finding a function of B_n to $\{0,1\}$ whose value provably gives no information about its argument (in connection with some fixed difficult problem, here the Conjugator Search Problem). In [**46**], it is argued that the function mapping a braid b of B_n to the parity of the exponent of Δ_n in the normal form of b has such a property.

In practice, it is sufficient to work with hash functions h that are one-way and collision-free, *i.e.*, such that recovering a braid from its image under h is infeasible and such that the probability for different braids to have the same image under h is negligible. There exist several ways of conceiving such functions. In [**12**], Cha & *al.* propose to resort to the (colored) Burau representation. As was mentioned in Section 3.3, the Burau representation of B_n is known to be unfaithful for $n \geqslant 5$, but the kernel is very small, and the probability that different braids admit the same Burau image is negligible. Also, there is no known way of recovering a braid from its Burau image. So a hash function can be obtained by encoding the Burau image, for instance in a sequence of 0's and 1's.

Other linear representations could be used similarly. One could also use non-linear representations, *i.e.*, homomorphisms whose target group is not a linear group. For instance, there exists a well known embedding, due to Artin, of B_n into $\mathrm{Aut}(F_n)$, the group of automorphisms of a rank n free group—see also [**58, 49**] for variants. However, Artin's representation cannot be used directly to define a hash function, as its image is precisely known and there exists an algorithmic way to invert it, *i.e.*, to go back from an automorphism to a braid.

Another type of hash function is considered in [**53**], where $h : B_n \to B_n$ is defined by $h(x) = xpx^{-1}$, with p a fixed public braid for which the Conjugator Search Problem is hard enough. Using such a function h in the Fiat-Shamir-like Scheme of Section 2.3.2 could be convenient, as one knows no nontrivial solution to the *Double Conjugator Search Problem* consisting in finding x witnessing that a braid p' is equal to $(xsx^{-1})p(xs^{-1}x^{-1})$ assuming that there exists one.

Completely different hash functions could be defined using Dynnikov's formulas of [**21**]—see also [**18**]. The principle is to encode every braid of B_n into a sequence of $2n$ integers, and it originates from an action of braids on laminations of a punctured disk by counting the intersections with some fixed lines. The sequence associated with the unit braid is $(0,1,0,1,\dots,0,1)$, and the formulas for deducing the sequence $(\alpha'_1,\beta'_1,\dots,\alpha'_n,\beta'_n)$ associated with the braid $b\sigma_i$ from the sequence $(\alpha_1,\beta_1,\dots,\alpha_n,\beta_n)$ associated with the braid b are

$$\begin{aligned} \alpha'_i &= \alpha_i + (\delta^+ + \beta_i)^+, & \alpha'_{i+1} &= \alpha_{i+1} - (\delta^+ - \beta_{i+1})^+, \\ \beta'_i &= \beta_i - (-\delta')^+ + \delta^+, & \beta'_{i+1} &= \beta_{i+1} + (-\delta')^+ - \delta^+, \end{aligned} \tag{4.3}$$

with $\delta = \alpha_{i+1} - \alpha_i$, $\delta' = \alpha'_{i+1} - \alpha'_i$, and x^+ standing for $\max(x, 0)$. The sequence associated with the braid $b\sigma_i^{-1}$ is obtained by successively applying τ, σ_i, and τ to the sequence associated with b, where $\tau(\alpha_1, \beta_1, \ldots, \alpha_n, \beta_n)$ is $(-\alpha_1, \beta_1, \ldots, -\alpha_n, \beta_n)$. This representation of B_n is known to be faithful, and it is easy to compute the image of large braids provided large integer arithmetic is available: the integers arising in the encoding of a length ℓ braid word are ℓ digit numbers.

Remark 4.4. Dynnikov's formulae (4.3) can also be used to efficiently solve of word problem of B_n: they even provide the only known solution whose complexity is proved to be quadratic in the length ℓ of the input word independently of the number of strands—the solution associated with the greedy normal form is quadratic in the length only for a fixed value of the number of strands, otherwise its complexity is in $O(\ell^3 \log \ell)$ [**23**]. This solution could be used as an alternative to braid word reduction in the approach of Section 4.3.

4.5. Other braid problems. Another direction of research is to investigate new primitive problems possibly replacing the Conjugator Search Problem.

4.5.1. *Root problems.* The braid groups are torsion-free, *i.e.*, if b is a non-trivial braid, then b^2, and, more generally, b^e for $e \geqslant 2$, is not trivial. Then two natural new problems arise, namely the Root Existence Problem (for exponent e): Starting with a braid b in B_n, decide whether there exists c satisfying $c^e = b$, and the Root Extraction Problem (for exponent e): Assuming that the braid b is an e^{th} power in B_n, find c satisfying $c^e = b$.

Styshnev proved in [**55**] that the previous problems are decidable—see also [**51**]—but the only known algorithm consists in explicitly enumerating several conjugacy classes related with the initial braid, a process which seems exponential in essence, and therefore infeasible when braids of a sufficient size are considered. In practice, root problems appear more difficult than conjugacy problems. Let us also mention a result by Gonzales-Meneses, who proves in [**33**] using the Nielsen–Thurston classification of braids that the e^{th} root in B_n is unique up to conjugacy when it exists.

The following authentication scheme of [**53**] relies on the Root Extraction Problem. Here Alice's private key is a braid s in B_n, while $p = s^2$ is public—replacing the exponent 2 by another fixed exponent is also possible. The authentication procedure consists in repeating k times the following three exchanges—h is again supposed to be a collision-free one-way hash function:

- A chooses a random braid r in B_n, and sends the commitment $x = h(rpr^{-1})$;
- B chooses a random bit c and sends it to A;
- For $c = 0$, A sends $y = r$, and B checks $x = h(ypy^{-1})$;
- For $c = 1$, A sends $y = rsr^{-1}$, and B checks $x = h(y^2)$.

If A knows s, the verification for $c = 1$ is $y^2 = rs^2s^{-1} = rpr^{-1}$, so an honest prover is accepted. On the other hand, a dishonest prover can always cheat B, but, because of the commitment, the probability is not more than $1/2$ for one three-pass exchange, hence not more than $1/2^k$ when the exchanges are repeated k times. The security of the scheme relies both on the difficulty of the Conjugator Search Problem and on the difficulty of the Root Extraction Problem. Its weakness lies in that, in the case $c = 1$, a conjugate of the secret key s is communicated.

No scheme relying on the Root Extraction Problem alone has been proposed so far.

4.5.2. *The minimal length problem.* A different type of problem consists in finding short words representing a given braid: this problem is not a problem inside the group B_n, but rather a problem for B_n together with the choice of a distinguished family of generators, *e.g.*, the σ_i's. Here we consider the problem for an arbitrary number of strands, *i.e.*, in the direct limit B_∞ of all groups B_n, which is the group generated by an infinite sequences of generators $\sigma_1, \sigma_2, \dots$ subject to the braid relations of (1.1). So the precise problem we address is the following *Minimal Length Problem*: Starting with a word w in the $\sigma_i^{\pm 1}$'s, find the shortest word w' that is equivalent to w, *i.e.*, that satisfies $w' \equiv w$.

The potential cryptographic interest of this problem lies in the following result of Paterson and Razborov:

PROPOSITION 4.5. [**48**] *The Minimal Length Problem is 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, *i.e.*, when there is no bound on the number of strands of the considered braids: inside B_n, no such result is known.

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. Preliminary experiments suggest 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 specified by braid words in which, for each i, at least one of σ_i or σ_i^{-1} does not occur, could be relevant. Also, the recent relaxation algorithm of [**59**] seems efficient for finding short word representatives in B_n for small values of n.

4.5.3. *More exotic operations.* Finally, the group operations of B_n are not the only possible ones, and we could also think of using other algebraic operations. Let sh denote the shift endomorphism of the group B_∞ that maps every generator σ_i to the corresponding σ_{i+1}—here working with B_∞ is essential. Then several skew versions of conjugacy may be defined. In particular, the operation defined on B_∞ by

$$s * p = s\,\mathrm{sh}(p)\,\sigma_1\,\mathrm{sh}(s^{-1}) \tag{4.4}$$

turns out to have many remarkable properties [**15**], and it could possibly be used as an alternative to conjugacy. Let us also mention that the function

$$h : x \mapsto x * 1 = x\,\sigma_1\,\mathrm{sh}(x^{-1}) \tag{4.5}$$

is known to be injective, while no practical way of finding preimages is known. This suggests that h could lead to a valuable hash function. Many variants are possible.

5. Conclusion

This survey is far from complete, but its aim is to give an introduction rather than a comprehensive description. We would like to convince the reader that braid cryptography is still at a very preliminary stage, and that much more work is needed before it can be claimed to be a valuable alternative to the cryptosystems based on

modular integers or on elliptic curves. In particular, the question of key generation seems crucial, and, as was emphasized in Section 4.1, very little is known about it.

More specifically, the complexity status of the Conjugator Search Problem in B_n remains unclear: it might turn out in the future that the worst-case complexity is polynomial—according to some weak prediction of [**56**]—or that the worst-case complexity is exponential, but the generic-case or the average-case complexity [**39, 40**] is polynomial. Such questions are important, but, in any case, the main issue for cryptography is the existence of definable families of instances for which the problem is provably difficult, and this is a different question. Also, we have mentioned that using conjugacy problems and their variants could be just a first step: completely new schemes could appear, and the future of braid cryptography does not necessarily reduce to the single question of conjugacy in B_n.

Braid groups are equipped with a number of different structures originating from various points of view: algebra, combinatorics, geometry, topology. It was argued recently [**50**] that the multiplicity of the possible approaches are a potential weakness, because many different attacks can appear and defeating them will perhaps be possible, but at the expense of changing the schemes and contradicting the necessary stability requirements. But, on the other hand, it may also be argued that the multiplicity of approaches is a guarantee of depth and richness, and, at this point, we see no serious reason for doubting that braid groups are and will remain a promising platform for cryptography.

References

A general web site with references in braid cryptology is maintained by H. Lipmaa at the address http://www.tcs.hut.fi/∼helger/crypto/link/public/braid/

[1] S.I. Adjan, *Fragments of the word Delta in a braid group*, Mat. Zam. Acad. Sci. SSSR **36-1** (1984) 25–34; translated Math. Notes of the Acad. Sci. USSR; 36-1 (1984) 505–510.

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

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

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

[5] D. Bessis, F. Digne, & J. Michel, *Springer theory in braid groups and the Birman-Ko-Lee monoid*, Pacific J. Math. **205-2** (2002) 287–309.

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

[7] S. Bigelow, *Braid groups are linear*, J. Amer. Math. Soc. **14-2** (2001) 471–486.

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

[9] J. Birman, K.H. Ko & S.J. Lee, *A new approach to the word problem in the braid groups*, Advances in Math. **139-2** (1998) 322–353.

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

[11] A.V. Borovik, A.G. Myasnikov, & V. Shpilrain, *Measuring sets in infinite groups*, Contemp. Math. **298** (2002) 21–42.

[12] J.C. Cha, K.H. Ko, S.J. Lee, J.W. Han, J.H. Cheon, *An efficient implementation of braid groups*, AsiaCrypt 2001, Springer Lect. Notes in Comput. Sci., 2048 (2001) 144–156.

[13] J.H. Cheon & B. Jun, *A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem*, Crypto 2003, to appear.

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

[15] P. Dehornoy, *Braids and Self-Distributivity*, Progress in Math. vol. 192, Birkhäuser (2000).

[16] P. Dehornoy, *Groupes de Garside*, Ann. Scient. Ec. Norm. Sup. **35** (2002) 267–306.

[17] P. Dehornoy, *Complete positive group presentations*, J. of Algebra **268** (2003) 156–197.
[18] P. Dehornoy, I. Dynnikov, D. Rolfsen, & B. Wiest, *Why are braids orderable?*, Panoramas & Synthèses vol. 14, Soc. Math. France (2002).
[19] P. Dehornoy & L. Paris, *Gaussian groups and Garside groups, two generalizations of Artin groups*, Proc. London Math. Soc. **79-3** (1999) 569–604.
[20] P. Deligne, *Les immeubles des groupes de tresses généralisés*, Invent. Math. **17** (1972) 273–302.
[21] 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).
[22] E. A. El-Rifai & H. R. Morton, *Algorithms for positive braids*, Quart. J. Math. Oxford **45-2** (1994) 479–497.
[23] D. Epstein, J. Cannon, D. Holt, S. Levy, M. Paterson, & W. Thurston, *Word Processing in Groups*, Jones & Bartlett Publ. (1992).
[24] E. Feder, *Algorithmic problems in the braid group*, PhD Thesis, City Univ. New York; http://arXiv.org/abs/math.GR/0305205 (2003).
[25] U. Feige, A. FIat, & A. Shamir, *Zero–knowledge proofs of identity*, J. Cryptology **1** (1988) 77–94.
[26] R. Fenn, M.T. Greene, D. Rolfsen, C. Rourke, & B. Wiest, *Ordering the braid groups*, Pacific J. of Math. **191** (1999) 49–74.
[27] N. Franco & J. Gonzales-Meneses, *Conjugacy problem for braid groups and Garside groups*, J. Algebra, to appear; http://xxx.lanl.gov/abs/math.GT/0112310 (2001).
[28] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, & U. Vishne, *Length-based conjugacy search in the braid group*, Preprint, http://arxiv.org/abs/math.GR/0209267 (2002).
[29] F. A. Garside, *The braid group and other groups*, Quart. J. Math. Oxford **20-78** (1969) 235–254.
[30] V. Gebhardt, *A new approach to the conjugacy problem in Garside groups*, Preprint, http://arxiv.org/abs/math.GT/0306199 (2003).
[31] S. Goldwasser, S. Micali, & C. Rackoff, *Knowledge complexity of interactive proof systems*, Proc. 17th STOC (1985) 291–304.
[32] J. Gonzales-Meneses, *Improving an algorithm to solve the Multiple Simultaneous Conjugacy Problems in braid groups*, Preprint, http://arxiv.org/abs/math.GT/0212150 (2002).
[33] J. Gonzales-Meneses, *The n-th root of a braid is unique up to conjugacy*, Preprint (2003).
[34] K.H. Han & J.W. Ko, *Positive presentations of the braid groups and the embedding problem*, Preprint.
[35] D. Hofheinz & R. Steinwandt, *A practical attack on some braid group based cryptographic primitives*, PKC 2003; Springer Lect. Notes in Comput. Sci. 2567 (2002) 187–198.
[36] J. Hughes, *The left SSS attack on Ko-Lee-Cheon-Han-Kang-Park key agreement scheme in* B_{45}, Rump session Crypto 2000.
[37] J. Hughes, *A linear algebraic attack on the AAFG1 braid group cryptosystem*, ACISP 2002; Springer Lect. Notes in Comput. Sci. 2384 (2002) 176–189.
[38] J. Hughes & A Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems, Inst. for Mathematics and Its Applic. (Minneapolis MN) 2000*, http://www.ima.umn.edu/preprints/apr2000/1696.pdf.
[39] I. Kapovich, A. Myasnykov, P. Schupp, & V. Shpilrain, *Generic-case complexity, decision problems in group theory and random walks*, J. Algebra **264** (2003) 665–694.
[40] I. Kapovich, A. Myasnykov, P. Schupp, & V. Shpilrain, *Average-case complexity and decision problems in group theory*, Adv. Math., to appear; http://arXiv.org/abs/math.GR/0206273 (2002).
[41] K.H. Ko, D.H. Choi, M.S. Cho, & J.W. Lee, *New signature scheme using conjugacy problem*, Preprint; http://eprint.iacr.org/2002/168.
[42] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, & C. Park, *New public-key cryptosystem using braid groups*, Crypto 2000; Springer Lect. Notes in Comput. Sci., 1880 (2000) 166–184.
[43] D. Krammer, *Braid groups are linear*, Ann. Math. **151-1** (2002) 131–156.
[44] S.J. Lee & E.K. Lee, *Overview of the cryptosystels using braid groups*, ASCOF 2001, pp. 41–50; http://www.kisa.or.kr/technology/sub1/data/ASCoF2001.pdf
[45] S.J. Lee & E.K. Lee, *Potential weakness of the commutator key agreement protocol based on braid groups*, Eurocrypt 2002, Springer Lect. Notes in Comput. Sci. 2332 (2002) 14–28.

[46] E.K. Lee, S.J. Lee, S.G. Hahn, *Pseudorandomness from braid groups*, Crypto 2001; Springer Lect. Notes in Comput. Sci., 2139 (2001) 486–502.
[47] E. Lee & J.H. Park, *Cryptanalysis of the public-key encryption based on braid groups*, Eurocrypt 2003, to appear.
[48] M.S. Paterson & A. A. Razborov, *The set of minimal braids is co-NP-complete*, J. of Algorithms **12** (1991) 393–408.
[49] W. Shpilrain, *Representing braids by automorphisms*, Intern. J. of Algebra & Comput **11-6** (2001) 773–777.
[50] V. Shpilrain, *Assessing security of some group based cryptosystems*, Preprint (2003).
[51] H. Sibert, *Extraction of roots in Garside groups*, Comm. in Algebra **30-6** (2002) 2915–2927.
[52] H. Sibert, *Algorithmique des groupes de tresses*, Thèse de doctorat, Université de Caen (2003).
[53] H. Sibert, P. Dehornoy, & M. Girault, *Entity authentication schemes using braid word reduction*, WCC 2003, to appear; http://eprint.iacr.org/2002/187.
[54] V.M. Sidelnikov, M.A. Cherepnev, & V.Y. Yashcenko, *Systems of open distribution of keys on the basis of noncommutative semigroups*, Ross. Acad. Nauk Dokl. **332-5** (1993) English translation: Russian Acad. Sci. Dokl. Math. 48-2 (1994) 384–386.
[55] V.B. Styshnev, *The extraction of a root in a braid group (English)*, Math. USSR Izv. **13** (1979) 405–416.
[56] W. Thurston, *Finite state algorithms for the braid group*, Circulated notes (1988).
[57] A. Vershik, S. Nechaev, & R. Bibkov, *Statistical properties of locally free groups with applications to braid groups and growth of random heaps*, Comm. Math. Phys. **212** (2000) 469–501.
[58] M. Wada, *Groups invariants of links*, Topology **31-2** (1992) 399–406.
[59] B. Wiest, *An algorithm for the word problem in braid groups*, Preprint; http://arXiv.org/abs/math.GT/0211169 (2002).

Laboratoire de Mathématiques Nicolas Oresme, UMR 6139 CNRS, Université de Caen BP 5186, 14032 Caen, France
E-mail address: dehornoy@math.unicaen.fr
URL: //www.math.unicaen.fr/∼dehornoy