Entity Authentication Schemes Using Braid Word Reduction

Hervé SIBERT, Patrick DEHORNOY, Marc GIRAULT

Abstract

Artin's braid groups provide a promising background for cryptographical applications. Several braid based key agreement protocols have been described, but few authentication or signature schemes have been proposed. We introduce three authentication schemes based on braids. Two of them are zero-knowledge interactive proofs of knowledge. Then we discuss their possible implementations, involving normal forms or an alternative braid algorithm, called handle reduction, which can achieve good efficiency under specific requirements.

1 Introduction

In the recent years, beginning with [14], several authors proposed to build cryptographical schemes using noncommutative groups, in particular Artin's braid groups [1, 2, 13, 4] where the word problem is easy and the conjugacy problem is algorithmically difficult, thus providing one-way functions. The aim of this paper is twofold. Firstly, we propose three authentication schemes designed for braid groups: the braid schemes considered so far dealt with key exchange and confidentiality, thus not providing means of authentication. Secondly, we propose a new way of implementing braid operations, namely using braid words and the so-called handle reduction algorithm. Not only is such an implementation very efficient in practice, but it is also well suited for the schemes we shall describe, and, more generally, for all braid schemes where using unique representatives of the braids is not necessary.

The paper is organized as follows. In Section 2, we state the difficult problems based on braid groups that we shall rely on. In Section 3, we describe our authentication schemes and study their theoretical security. In Section 4, we present classical ways of computing with braids, and analyze the security of the associated implementations of our protocols. In Section 5, we introduce a second, radically new approach consisting in using non-normal braid words together with a reduction algorithm, and we discuss its efficiency and security.

2 Difficult braid problems

For $n \geqslant 2$, Artin's braid group B_n is defined to be the group with presentation

$$\langle \sigma_1, \ldots, \sigma_{n-1} \,;\, \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geqslant 2,\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ for } |i-j| = 1 \rangle. \tag{2.1}$$

We refer the reader to any textbook about braids, for the usual geometrical interpretation of each element of the group B_n by an n-strand braid: the idea is that every n-strand braid diagram can be encoded in a word in the letters $\sigma_i^{\pm 1}$ by slicing it into a concatenation of elementary diagrams with one crossing, and using σ_i for the diagram where the ith strand crosses over the $(i+1)$st one. Then, the relations of (2.1) correspond to the notion of ambient isotopy,

as shown by E. Artin. Notice the sum of the exponents ± 1 of a braid word in the letters $\sigma_i^{\pm 1}$ depends only on the braid b it represents. We call this sum $e(b)$ the *exponent sum* of b.

The first important point here is that braid groups are infinite noncommutative groups which are eligible for practical computations: there exist in particular efficient ways of specifying a braid and of computing with braids (see Section 4 below). The second important point is the existence of difficult problems for which no feasible solution is known. The main problem suitable for cryptography is the following:

Conjugacy Search Problem(CSP): *Given a braid b in B_n, and a conjugate $b' = sbs^{-1}$ of b, find a witness, i.e., find s' satisfying $b' = s'bs'^{-1}$.*

In [10], Garside gives an algorithm solving the CSP, and improvements of his algorithm have been proposed [6, 9]. Nevertheless, the CSP has no polynomial-time solution up to now, and it is considered infeasible to solve the CSP for sufficiently large braids [12].

Note that only variants of this problem, all reducing to the CSP, have been used in braid-based cryptography so far. In our first scheme we use the following variant:

Diffie-Helmann-like Conjugacy Problem (DHCP): *Given a braid b in B_n, and the braids $b' = sbs^{-1}$ and $b'' = rbr^{-1}$, where s and r lie in two subgroups of B_n that commute one with the other, find the braid $sb''s^{-1}$ $(= rb'r^{-1})$.*

The connection between this problem, explicitly introduced in [13], and the CSP is similar to the one between the Diffie-Helmann Problem and the Discrete Log Problem. The DHCP is obviously reducible to the CSP, but it is expected to be as hard. A last problem is :

Root Problem (RP) (for exponent e): *Assuming that the braid b' is an e-th power in B_n, find an e-th root of b', i.e., find b satisfying $b^e = b'$.*

It is proved in [15] that the RP is decidable, but the only known algorithm consists in enumerating several conjugacy classes related with the initial braid b', a process which is exponential in essence. In practice, the RP looks even more difficult than the CSP.

3 Three authentication schemes

We present three public-key entity authentication schemes. Scheme I is a two-pass scheme based on the DHCP and is perfectly honest-verifier zero-knowledge. The other two schemes are iterated three-pass protocols. One is based on the CSP only, and the other on the CSP and the RP. Both are zero-knowledge in a theoretical infinite framework. Notice that our second scheme is the first braid-based scheme relying only on the CSP, so it resists the existing attacks against variants of the CSP. The level of security of the schemes can be parameterized by modifying the size of the braid specifiers in use, in particular the braid index n.

3.1 Scheme I

Scheme I is derived from the key agreement scheme proposed in [13], and relies on the difficulty of the DHCP. It uses the trick that braids involving disjoint families of strands commute. The data consist of a public key, which is a pair of braids, and of A's private key, also a braid.

We denote by LB_n (*resp.* UB_n) the subgroup of B_n generated by the σ_i's with $i < \lfloor n/2 \rfloor$ (*resp.* $i > \lfloor n/2 \rfloor$). The point is that the subgroups LB_n and UB_n commute. We assume that H is a fixed collision–free hash function, thus thwarting active attacks (practical choices for H, using the Burau representation or braid operations, are in the full version of the paper).

• Phase 1. **Key generation**
- (i) Choose a public braid p in B_n such that the DHCP for p is hard enough;
- (ii) A(lice) chooses a secret braid s in LB_n, her private key; she publishes $p' = sps^{-1}$; the pair (p, p') is the public key.

• Phase 2. **Authentication phase**
- (i) B(ob) chooses a braid r in UB_n, and sends the challenge $x = rpr^{-1}$ to A;
- (ii) A sends the response $y = H(sxs^{-1})$ to B, and B checks $y = H(rp'r^{-1})$.

Proposition 3.1. *Scheme I is a perfect honest-verifier ZK interactive proof of knowledge of s.*

The proof is detailed in the full version of the paper.

3.2 Scheme II

We describe now two authentication schemes belonging to the family of zero-knowledge schemes [8, 11]; they consist in repeating a three-pass process several times so as to guarantee the required level of security. The probability that a dishonest prover is authenticated is $1/2^k$, where k is the number of repetitions. The first scheme is based only on the difficulty of the CSP, thus avoiding various existing attacks of the variants of the CSP, namely against the DHCP and the Multiple Conjugacy Problem [1].

• Phase 1. **Key generation**:
- (i) Choose a public braid p in B_n, so that the CSP for p is hard;
- (ii) A(lice) chooses a secret braid s in B_n, her private key; she publishes $p' = sps^{-1}$; the pair (p, p') is the public key.

• Phase 2. **Authentication phase**: Repeat the following exchanges k times, with k a polynomial function of the size of the braid specifiers:
- (i) A chooses a random braid r, and sends $x = rpr^{-1}$ to B(ob);
- (ii) B sends a random bit ϵ to A;
- (iii) For $\epsilon = 0$, A sends $y = r$ to B, and B checks $x = ypy^{-1}$;
- (iii') For $\epsilon = 1$, A sends $y = rs^{-1}$ to B, and B checks $x = yp'y^{-1}$.

Remark 3.2. Scheme II can be used without change when the braid group B_n is replaced with another (finite or infinite) noncommutative group G in which the CSP is difficult.

In the general framework of a group G—so, in particular, in the case of B_n—we have the natural notion of a right-invariant probability measure, namely a measure P satisfying $P(A) = P(\{xa; x \in A\})$ for each a in G, uniform probability measure being a special case.

Proposition 3.3. *Whenever the probability distribution of r at Step 2(i) is right-invariant, Scheme II (in B_n, or, more generally, in any group G where the CSP is difficult) is a zero-knowledge interactive proof of knowledge of s.*

Proof (sketch). – Completeness: In Step 2(iii), we have $y = r$, so $x = ypy^{-1}$. In Step 2(iii'), using $y = rs^{-1}$ and $p' = sps^{-1}$, we get $x = rpr^{-1} = (rs^{-1})(sps^{-1})(rs^{-1})^{-1} = yp'y^{-1}$. So Bob accepts a correct answer at each step, so he accepts Alice's proof of identity with probability 1.

– Soundness: The proof follows the scheme of [8]. If an entity A' is authenticated with nonnegligible probability, then some level of the truncated execution tree of (A', B) has at least half of its vertices having two sons, *i.e.*, vertices where A' is able to answer both questions of B. This yields a polynomial algorithm finding a vertex with two sons. Hence, from the

knowledge of A', one can extract in polynomial time a braid behaving like the private key of Alice, so Scheme II is sound.

– *Zero-knowledge:* We consider the following probabilistic Turing machine M:

- Step 1: M randomly selects a bit ε and a braid y;
- Step 2: For $\varepsilon = 0$, M computes $x = ypy^{-1}$; for $\varepsilon = 1$, it computes $x = yp'y^{-1}$;
- Step 3: M initiates a protocol with B, sends x to B; B replies with the bit ε';
- Step 4: For $\varepsilon = \varepsilon'$, M outputs the triple (x, ε, y), otherwise it resets to Step 1.

Since the probability distribution of r in the authentication protocol is assumed to be right-invariant, we obtain the same probability distribution for the y's generated by M as for Alice's ones. Moreover, since in case $\varepsilon = 1$ we have $x = ysps^{-1}y^{-1}$, *i.e.*, $x = (ys)p(ys)^{-1}$, using the same assumption, we obtain the same probability distribution for the x's arising for $\varepsilon = 0$ and those arising for $\varepsilon = 1$. As a consequence, B cannot distinguish the two cases, and the probability to have $\varepsilon = \varepsilon'$ is equal to $1/2$. This implies that the computation cost of the machine M is in average $2k$ operations, and, therefore, that M is a polynomial time machine provided that k itself is polynomial in the size of the initial data. □

3.3 Scheme III

Our second ZK scheme involves the RP together with the CSP for the same braid. Here we give the description for exponent $e = 2$; the other cases are similar.

• Phase 1. **Key generation**:

- A(lice) chooses a secret braid s in B_n and computes $p = s^2$, so that the CSP for s and p, and the RP for p are difficult; public key is p, private key is s.

• Phase 2. **Authentication phase**: Repeat the following exchanges k times:

- (i) A chooses a random braid r, and sends $x = rpr^{-1}$ to B(ob);
- (ii) B sends a random bit ϵ to A;
- (iii) For $\epsilon = 0$, A sends $y = r$ to B; then B checks $x = ypy^{-1}$;
- (iii') For $\epsilon = 1$, A sends $y = rsr^{-1}$ to B; then B checks $x = y^2$.

Proposition 3.4. *Scheme III is a zero–knowledge interactive proof of knowledge of s, provided an element of the conjugacy class of the secret key is published and the probability distribution of r at Step 2(i) is right-invariant.*

In the proof, which is detailed in the full paper, we assume that a braid s' of the conjugacy class of s is published at Phase 1, as conjugates of s would be disclosed in any case.

4 Implementations using normal forms

In practical implementations, we have to replace the infinite braid group B_n by some finite subset, and it is unclear how a right-measure could be defined on such a subset. So, in such implementations, Schemes II and III do not remain zero-knowledge. However, we shall see here how to adapt the schemes so as to retrieve practical indistinguishability.

4.1 Specifying a braid

Before discussing implementations, we first need to know how braids are represented. Several solutions exist. In this section, we consider the most common one, which consists in choosing

a normal form, *i.e.*, choosing one distinguished representative for every braid. However, we can also work with arbitrary representatives, as we will see in Section 5.

• **Greedy normal form.** When the braid group B_n is introduced as in Section 2, a braid is an equivalence class of words with respect to the equivalence relation $\equiv$ generated by the relations (2.1). Every braid can be specified by different braid words, but we can avoid ambiguities by resorting to a normal form, *i.e.*, by defining distinguished words, called *normal*, so that each braid is represented by a unique normal word. In terms of implementation, the most practical normal form is the *greedy normal form* of [7]. It is associated with an automatic structure, which in particular implies that the normal form of a length ℓ braid word can be computed in $\mathcal{O}(\ell^2)$ steps [7]. The greedy normal form is the way braids are specified in [13].

• **Permutations.** There exists a natural surjective mapping of B_n to the symmetric group S_n. This mapping is non-injective, but there exists a canonical way to choose for every permutation π a braid word $\widehat{\pi}$ that projects onto π. Now, let Δ_n be Garside's fundamental braid word $\Delta_n = (\sigma_1)(\sigma_2\sigma_1)\dots(\sigma_{n-1}\sigma_{n-2}\dots\sigma_2\sigma_1)$. The normal n-strand braid words considered above happen to have the form $\Delta_n^k\widehat{\pi_1}\dots\widehat{\pi_\ell}$, where k is an integer, and $(\pi_1,\dots,\pi_\ell)$ is a sequence of permutations in S_n. Hence we can encode a normal braid word by the associated sequence $(k,\pi_1,\dots,\pi_\ell)$. Conversely, we can associate to every sequence $(k,\pi_1,\dots,\pi_\ell)$ as above the braid word $\Delta_n^k\widehat{\pi_1}\dots\widehat{\pi_\ell}$. If this word is normal, we naturally say that the sequence $(k,\pi_1,\dots,\pi_\ell)$ is normal. The point is that there exists an efficient algorithm [7] which, starting with an arbitrary sequence $(k,\pi_1,\dots,\pi_\ell)$, computes the unique normal sequence $(k',\pi'_1,\dots,\pi'_{\ell'})$ representing $\Delta_n^k\widehat{\pi_1}\dots\widehat{\pi_\ell}$. Therefore, sequences of permutations also provide a good solution for working with braids.

If a braid b is represented by the normal sequence $(k,\pi_1,\dots,\pi_\ell)$, we define the *infimum*, *supremum*, and *normal lengths* of b by $\inf(b) = k$, $\sup(b) = k+\ell$ and $\lg(b) = \ell$, respectively.

4.2 Implementations of Schemes II and III

When it comes to implementation in a finite background, Schemes II and III lose their theoretical zero-knowledge properties. Indeed, we use some finite alphabet $\mathcal{A}$ (consisting of the letters $\sigma_i^{\pm 1}$, or of permutations plus one integer), and we draw words in the set $\mathcal{A}^\ell$ of all words over $\mathcal{A}$ of length ℓ. This yields a random draw in the finite set $\mathcal{B}_n^\ell$ of all braids in B_n that can be specified (in at least one way) by a word in $\mathcal{A}^\ell$, *i.e.*,

$$\mathcal{B}_n^\ell = \{[w]\,;\, w \in \mathcal{A}^\ell\},$$

where $[w]$ denotes the braid represented by w. There is no known draw over $\mathcal{A}^\ell$ that would induce the right invariant distribution over $\mathcal{B}_n^\ell$ required for Proposition 3.3, so we cannot design a probabilistic Turing machine simulating the exchanges in Scheme II for $\varepsilon = 1$.

4.3 Scheme II'

The problem lies in the probability distributions of the instances of the form $(x,1,y)$: indeed, the distribution of the braids rs^{-1} with r in $\mathcal{B}_n^\ell$ can be in general distinguished from that of braids r chosen randomly in $\mathcal{B}_n^\ell$. To avoid this problem, we consider a variant of Scheme II, called Scheme II'. We show that Scheme II' is, in fact, as secure as Scheme II itself, and we give statistical arguments supporting practical indistinguishability.

Let us assume here that braids are specified using normal sequences of permutations. So $\mathcal{A}$ denotes the set of all permutations of n objects, and Alice is supposed to choose random words

n	$\sup(r) = \sup(s)$	$\sup(p)$	$\sup(rpr^{-1})$/ awaited value	$\sup(rs^{-1})$/ awaited value
20	10	20	29.98 / 30	9.99 / 10
30	10	20	30.00 / 30	10.00 / 10
50	15	30	45.00 / 45	15.00 / 15

Table 1: Supremum length of rpr^{-1} and rs^{-1} when r, p, s are random positive (*i.e.*, satisfying $\inf r = \inf p = \inf s = 0$) braids of given sup (samples of 10,000 braids).

uniformly in $\mathcal{A}^\ell$. We define Scheme II' by introducing the following changes in Scheme II:
- the private key s satisfies $\inf(s) = 0$, *i.e.*, there is no Δ_n in the normal word expansion of s, and $\sup(s) = \ell$; these values are public, and so is the exponent sum $\mathrm{e}(s)$.
- the public key p' is chosen in $\mathcal{B}_n^\ell$, and p is deduced using $p = s^{-1}p's$.
The difference with Scheme II is that, in Scheme II', the characteristics of p and p' are exchanged. This change induces no problem, as p and p' play symmetric roles:

Proposition 4.1. *Scheme II' and Scheme II have the same security.*

The proof is in the full version of the paper, the point being that the instances of the CSP Scheme II' relies on are just as difficult as the corresponding instances in Scheme II.

By construction, Scheme II' has the same ideal zero knowledge properties (Proposition 3.3) as Scheme II. We now give a statistical argument based on a length analysis supporting the claim that, for Scheme II', the distinguishability introduced by the restriction to a finite set of braids provides no useful information. Some other arguments are in the full paper.

Proposition 4.2. *For every prover A, there exists a probabilistic polynomial-time Turing machine, which recreates for each B its view of the communication between A and B with a probability distribution that is indistinguishable through statistical length analysis of the data.*

The proof is detailed in the full version of the paper. The point is that, for an overwhelming proportion of pairs of braids (r, a), we have $\inf(ra) = \inf r + \inf a$ and $\sup(ra) = \sup r + \sup a$, and similar relations for conjugacy (see Table 1). Straight computation shows that the braids effectively transmitted by A, and those generated by the Turing machine, have the same infimum and supremum lengths and exponent sum for $n \geqslant 30$, so they are indistinguishable by length analysis, which is the main statistical mean of distinction for braids.

We thus have discussed the security for an implementation of Scheme II when braids are specified using sequences of permutations, or, equivalently, of normal words. Similar security results are obtained when braids are specified using arbitrary braid words. Finally, analogous results hold when Scheme II is replaced with Scheme III.

4.4 Choice of the parameters

When braids are represented using a normal form, the choice of the public and private keys ensuring security of the schemes depends only on the bounds making the CSP, the DHCP and the RP difficult. From the considerations in [13], it arises that, when braids are specified using sequences of permutations, parameters ensuring sufficient security and efficiency are $n = 30$ ($n = 60$ in the case of Scheme I) and products of 15 random permutations. When

braids are represented using greedy normal braid words, for the same values of n, choosing braid words of length at least $1,000$ guarantees at least the same level of security as the values given for products of permutations.

5 Implementations using braid reduction

We describe now implementations of a new type, based on a braid algorithm called *handle reduction*. They are very flexible, but require specifications to ensure a good level of security.

5.1 Non unique specifiers.

The implementations of Section 4 involve unique expressions of the braids: in each case, a braid is encoded into a unique object, and it is specified by a *unique* word over some alphabet, whose letters may be σ_i's (in the case of normal words), or permutations (as in 4.1).

At least in some cases, this uniqueness is not needed. This is especially the case when braids are used for authenticating entities, as in the schemes described here: authentication requires to check equivalence only, hence it may be suitable to use non-unique representatives.

5.2 Handle reduction of braids

The previous principles can be implemented by using an efficient solution for the word problem of B_n involving no normal form, namely the handle reduction method of [5]. Handle reduction is an algorithmic procedure that takes a braid word w as input and returns an equivalent braid word $\mathrm{red}(w)$. From an algorithmic point of view, reducing a braid word is simpler than computing its greedy normal form (at least 20 times faster for the sizes under consideration), but, on the other hand, reduction does not yield a normal form, as $w \equiv w'$ does not imply $\mathrm{red}(w) = \mathrm{red}(w')$ in general. We refer to [5] for a description of the method, also included in the full version of the paper. For our current purpose, the point is the following:

Proposition 5.1. *Let w, w' be braid words. Then w and w' are equivalent if and only if the braid word $w^{-1}w'$ reduces to the empty word.*

Table 2 shows some statistics for the number of reduction steps and the average time needed to reduce a random braid in terms of the braid index (*i.e.*, the size of the alphabet) and the number of crossings (*i.e.*, the length of the words).

	$n = 8$	$n = 16$	$n = 32$	$n = 64$
$\ell = 256$	0.85 (88)	0.35 (22)	0.18 (6.4)	0.07 (2.1)
$\ell = 1,024$	28 (1,974)	7 (385)	2.5 (81)	1.3 (21)
$\ell = 4,096$	1,333 (35,966)	621 (18,617)	88 (2,188)	24 (358)

Table 2: Statistics for handle reduction: average CPU time in millisec. and (bracketed) average number of reduction steps in terms of the braid index n and the length ℓ of the words

Let $\mathcal{A}$ denote here the alphabet $\{\sigma_1^{\pm 1}, \ldots, \sigma_{n-1}^{\pm 1}\}$, and let $\mathcal{B}_n$ denote the set of all n-strand braid words, *i.e.*, the set of all words over $\mathcal{A}$. As previously, for w in $\mathcal{B}_n$, we denote by $[w]$ the braid represented by w. Let us consider Scheme II. The data used by Alice and Bob are now braid words, and equality is replaced by equivalence everywhere. So, Phase 1 becomes

• Phase 1. **Key generation**:
- (*i*) Choose a public braid word p;
- (*ii*) A(lice) chooses a secret braid word s, her private key; she publishes $p' \equiv sps^{-1}$; the pair of braid words (p, p') is the public key.

We must specify the choice of the public key p'. With unique braid specifiers there is one possible choice for p' only by definition. Now we use arbitrary braid words, so we must explain how to choose p' among the braid words equivalent to sps^{-1}. This means we have to choose some function S of $\mathcal{B}_n$ to itself mapping every braid word to an equivalent braid word. We call S the *scrambling* function, as the security of the exchanges will rely on the impossibility of recovering the word w from the word $S(w)$. So, our implementation of Scheme II becomes

• Phase 1. **Key generation**:
- (*i*) Choose a public braid word p;
- (*ii*) A(lice) chooses a secret braid word s, and publishes $p' = S(sps^{-1})$; public key is the pair of words (p, p'); private key is the word s.

• Phase 2. **Authentication phase**: Repeat the following exchanges k times:
- (*i*) A chooses a random braid word r, and sends $x = S(rpr^{-1})$ to B(ob);
- (*ii*) B sends a random bit ϵ to A;
- (*iii*) For $\epsilon = 0$, A sends $y = r$ to B, and B checks that $x^{-1}ypy^{-1}$ reduces to ε;
- (*iii'*) For $\epsilon = 1$, A sends $y = S(rs^{-1})$, and B checks that $x^{-1}yp'y^{-1}$ reduces to ε.

5.3 Choice of a scrambling function

When using non-unique specifiers, the security of our schemes do not rely only on the instances of the difficult braid problems which are published/transmitted, but also on the words chosen to publish/transmit these instances. Hence, we have to introduce requirements about how the instances of the involved braid problems are specified using non-unique words:

$$\text{Recovering the braid } [s] \text{ from the words } p \text{ and } S(sps^{-1}) \text{ must be infeasible.} \tag{5.1}$$

$$\text{Recovering the braid } [s] \text{ from the word } S(s^2) \text{ must be infeasible.} \tag{5.2}$$

A first, natural solution is to define the scrambling function S to be a unique, distinguished braid representative, for instance

$$\text{Define } S(w) \text{ to be the normal form of } w.$$

For such a choice, (5.1) and (5.2) are merely equivalent to their braid counterparts, and the security of our schemes is guaranteed for typical values of the parameters similar to those considered in [13] and mentioned above.

We argued that the advantage of using arbitrary braid words and handle reduction is to avoid computing normal forms, so using normal form for the scrambling process may appear paradoxical. Let us observe that such a choice could be relevant, especially in the case of the non-symmetric Schemes II and III. Here Alice (the prover) would have to compute normal forms—which we said has a non-negligible computation cost—but Bob (the verifier) only has to check braid equivalences, what he can easily do using braid word reduction, which, in particular, requires very little memory.

A second, different, solution is to construct a scrambling function using handle reduction. What makes Requirement (5.1) more difficult than its braid counterpart is that the

word $S(sps^{-1})$ may contain information about the prefixes of sps^{-1}, so in particular about s or its equivalence class $[s]$. Now assume that we can define S so that the following condition holds:

$$\text{No proper prefix of } S(w) \text{ is equivalent to a (proper) prefix of } w. \tag{5.3}$$

Then the only piece of information we can extract from, say, the words p and $S(sps^{-1})$ is the braid $[p]$ and the braid $[sps^{-1}]$, and we are back to the CSP for finding $[s]$. The argument is similar for $S(s^2)$, so we can consider that, if a function S satisfying Condition (5.3) is available, then the requirements of Section 3 are satisfied under the same theoretical assumptions as above about the CSP and the RP.

We claim this can be (sufficiently) achieved using handle reduction for scrambling, namely

- Define $S(w)$ to be $\mathrm{red}(w)$, *i.e.*, the result of reducing w.

This definition makes sense, as the braid word $\mathrm{red}(w)$ is equivalent to w in every case. Condition (5.3) is not fulfilled in general: if the initial word w is reduced, *i.e.*, contains no handle, then $\mathrm{red}(w)$ is equal to w, so, in particular, w and $\mathrm{red}(w)$ have many prefixes in common. However, we detail in the full version of the paper how to choose the parameters so as to avoid the problem, namely by defining the public key p to be a word of the form $p_1^{-1}p_2$, where p_1 and p_2 are positive braid words (no letter σ_i^{-1}) of equal length and, moreover, their classes are large with respect to some linear ordering of B_n.

There is still another possible choice of the scrambling function involving another braid algorithm called word reversing, which mainly consists in replacing some very short subwords in the braid word we are scrambling by equivalent, but different, subwords.

5.4 Zero knowledge property

In Section 4, we adapted Scheme II and defined Scheme II' so as to reach practical indistinguishability in the zero-knowledge part of the proof. Scheme II' is still relevant when non-unique specifiers are used, and the result about the indistinguishability of the braids is still valid. However, in the current case, we also have to obtain a similar result for the words that are exchanged, and not only for the braids they represent, *i.e.*, we have to check that the exchanged reduced (or scrambled) words are indistinguishable in terms of length and of common subwords. The first point follows from the fact that the braid word s' used in the proof of Prop. 4.2 has the same length as the private key s; the second point is true provided the private key is chosen as explained above. Thus Scheme II' satisfies the same zero knowledge properties in the framework of non-unique specifiers as in that of normal forms.

5.5 Normal form vs. reduction

Comparing the different implementations we have proposed is not obvious, as random draws over products of permutations or generators $\sigma_i^{\pm 1}$ give radically different distributions on B_n. In the full paper, we put the emphasis on the use of braid word reduction, as implementations using normal forms already appear in literature. Namely, we justify the fact that using braid words together with handle reduction is an efficient solution, as it avoids attacks based on the fact that the permutational length is small, like the one considered in [12].

6 Conclusion

In this paper, we have proposed the first authentication schemes specially designed for braid groups. The first of them is a two-pass protocol relying on the Diffie-Hellman variant of the CSP, while the two other ones are iterated three-pass protocols based on the CSP and/or RP. We have discussed the security of these schemes and given evidence that none of them leaked any useful information on the secret key, even though traditional zero knowledge models cannot apply to the braid groups, which are infinite. Finally, we have addressed in a detailed manner the issue of implementing these protocols. We have pointed out the relevance of the so-called handle reduction for authentication schemes, as this method allows one to efficiently verify the equivalence of two braid words without requiring to compute a normal form.

References

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

[2] I. Anshel, M. Anshel, B. Fisher & D. Goldfeld, *New key agreement schemes in braid group cryptography*, RSA 2001.

[3] 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.

[4] J.C. Cha, K.H. Ko, S.J. Lee, J.W. Han & J.H. Cheon, *An efficient implementation of braid groups*, AsiaCrypt 2001, LNCS **2248** (2001) 157–174.

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

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

[7] D. Epstein & *al.*, *Word Processing in Groups*, Jones & Bartlett Publ. (1992).

[8] U. Feige, A. Fiat & A. Shamir, *Zero–knowledge proofs of identity*, J.of Cryptology **1** (1988) 77–94.

[9] N. Franco & J. Gonzales-Meneses, *Conjugacy problem for braid groups and Garside groups*, http://xxx.lanl.gov/abs/math.GT/0112310 (2001).

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

[11] S. Goldwasser, S. Micali & C. Rackoff, *Knowledge complexity of interactive proof systems*, Proc. 17th STOC (1985) 291–304.

[12] J. Hughes, *The left SSS attack on Ko-Lee-Cheon-Han-Kang-Park key agreement scheme in* B_{45}, Rump session Crypto 2000.

[13] 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, LNCS **1880** (2000) 166–184.

[14] 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.

[15] V.B. Styshnev, *The extraction of a root in a braid group (English)*, Math. USSR Izv. **13** (1979) 405–416.