New Signature Scheme Using Conjugacy Problem

Ki Hyoung Ko, Doo Ho Choi, Mi Sung Cho, and Jang Won Lee

Department of Mathematics, Korea Advanced Institute of Science and Technology, Daejeon, 305-701, Korea
{knot,dhchoi,cms,leejw}@knot.kaist.ac.kr

Abstract. We propose a new digital signature scheme based on a non-commutative group where the conjugacy search problem is hard and the conjugacy decision problem is feasible. We implement our signature scheme in the braid groups and prove that an existential forgery of the implementation under no message attack gives a solution to a variation of conjugacy search problem. Then we discuss performance of our scheme under suggested parameters.

Key words: non-commutative, braid, digital signature, provable, conjugacy problem

1 Introduction

The braid groups were first introduced to construct a key agreement protocol and a public-key encryption scheme [12] presented at CRYPTO 2000. Since then, there have been few other attempts to apply the braid groups or other non-commutative groups to cryptography. Regarded as positive results are the discovery of a hard-core predicate for the conjugacy search problem in the braid group in [13], an implementation of braid computations in [6], and a conversion of the public-key encryption scheme of [12] into a provable one in [11]. Regarded as ambitious but unsuccessful attempts are the pseudorandom generator using the decision problem of Diffie-Hellman type in the braid group in [13] and a public-key cryptosystem using DLP and the conjugacy problem in matrix groups in [16]. But no signature schemes based on non-commutative groups have been proposed so far.

The schemes in [12] are roughly based on the conjugacy problem that is natural and unavoidable on non-commutative groups. More precisely the schemes are based on the conjugacy Diffie-Hellman problem. The role of two problems in the braid cryptosystems is exactly the same as that of the discrete logarithm problem and the Diffie-Hellman problem in the ElGamal scheme on a cyclic group. Besides multiplication among exponents, addition was needed in the ElGamal signature scheme or DSA. But there is no operation corresponding to the addition that is compatible with the conjugation operation in non-commutative groups. This difficulty has been hampering a proposal of a signature scheme on non-commutative algebraic structure.

In this paper we propose a new digital signature scheme based on a variation of the conjugacy problem in non-commutative groups. In fact our signature schemes can be implemented on any non-commutative group where there is a gap between the computational version and the decision version of the conjugacy problem. The philosophy of our scheme is somewhat similar to [15, 5] that are based on a gap between two versions of the Diffie-Hellman problem and this gap became a reality on the elliptic curve cryptography due to the Weil pairing. But the difference is larger. In the discrete logarithm problem, the one-way function taking powers on a fixed generator is onto and so the decision version does not make sense. On the other hand the one-way function taking conjugates on a fixed element in a non-commutative group is far from being onto. Therefore a gap may exists not only between two versions of the

Diffie-Hellman type problem but also between two versions of the conjugacy problem itself. Obviously a gap between the latter implies a gap between the former.

Under our current knowledge, the computational version of conjugacy problem in the braid group can be solved only by an exponential algorithm [9, 7] and so the computational version can be regarded infeasible with the security bound determined by the best solution known. In this paper we propose an oracle to solve the decision version of conjugacy problem in the braid groups and so the braid group has the required gap. This oracle also makes the decision problem of Diffie-Hellman type in the braid group considered in [13] feasible.

Our conjugacy signature scheme is implemented by using the gap in the braid group. This new signature scheme has the following feature and implication:

- It is the first signature scheme based on a non-commutative group.
- The performance and the efficiency is comparable with popular signature schemes based on number theory.
- It is randomized but yet simple so that the provable security of existential forgery under no message attack is immediate and the based problem is closely related to the conjugacy problem.
- Together with the key agreement and the public-key encryption in [12], it demonstrates a usefulness of braid groups in cryptography.
- Various other signature protocols can be designed using the gap in the braid group.

This paper is organized as follows:

In section 2, we will explain two variations of the conjugacy problem in non-commutative group and show their equivalence. Then we propose a general signature scheme on non-commutative groups that have the gap between two versions of conjugacy problem.

In section 3, we give a brief introduction to braid groups and explain the difficulty of the computational version. Then we give an algorithm for the decision version and discuss the security and efficiency of this algorithm.

In section 4, we implement our signature scheme in braid groups. We discuss various methods of random braids including braid hash function and key generation. Then we give a proof of the security and finally we suggest values for system parameters and give a performance table.

2 Conjugacy signature scheme in non-commutative groups

In order to digitize a non-commutative group, we usually describe the group by a presentation or as a multiplicative subgroup of a matrix group over a (polynomial) ring. A presentation of a group is a set of generators and defining relations. A prerequisite to use a presentation is a fast solution to the word problem. On the other hand most of groups given as a subgroup of a matrix group are not useful in cryptography due to plenty of well-developed tools in linear algebra.

2.1 Conjugacy problems in non-commutative groups

In a non-commutative group G, two elements x, y in G are *conjugate* each other, written $x \sim y$ if $y = a^{-1}xa$ for some $a \in G$. Here a or a^{-1} is called a *conjugator* and the pair (x, y) is said to be *conjugate*. Clearly $\sim$ is an equivalence relation. A simple and natural question to ask in a non-commutative group G is the conjugacy problem that can be described as a decision version and a computational version. The *conjugacy decision problem*(**CDP**) asks to determine whether $x \sim y$ for a given instance $(x, y) \in G \times G$. The *conjugator search problem*(**CSP**) asks to find $a \in G$ satisfying $y = a^{-1}xa$ for a given instance $(x, y) \in G \times G$ such that $x \sim y$. We have to be careful when we mention instances in an infinite group G. In the current information theory, it is hard to discuss a uniform distribution in G of elements described by randomly chosen information. To avoid any potential controversy, we always assume that instances to a problem are randomly chosen in a finite subset of an infinite group G restricted by system parameters.

We say a problem is *solvable* (*feasible*, respectively) if there is a deterministic finite (probabilistic polynomial-time, respectively) algorithm that outputs a solution that is accurate (accurate with non-negligible probability, respectively). The solvability is a mathematical notion and the complexity of an algorithm is not an issue as long as it is finite. A solvable problem is not necessarily feasible and vice versa.

The representation theory tells us that for any group G there are homomorphisms from G to rings that are invariant under conjugacy relation. Therefore CDP is always feasible although CDP may not be solvable. But the remaining question concerning CDP is how to construct an efficient algorithm to solve CDP with overwhelming probability.

On the other hand, there are many candidates for non-commutative groups where CSP is infeasible. However there is a normal form (such as Jordan form) of a conjugacy class in many matrix groups and so it is difficult to find a non-commutative group given as a subgroup of a matrix group that has an infeasible CSP. Therefore non-commutative groups with infeasible CSP are usually given by presentations.

We believe that CSP is infeasible in the braid groups B_n even though it is solvable. We will construct an efficient algorithm to give a solution to CDP with overwhelming accuracy. Unfortunately we do not know whether there is a polynomial-time algorithm that decides CDP.

2.2 Matching conjugacy problems in non-commutative groups

For a non-commutative group G, a pair $(x, x') \in G \times G$ is said to be *CSP-hard* if $x \sim x'$ and CSP is infeasible for the instance (x, x'). If (x, x') is CSP-hard, so is clearly (x', x). We now define two matching conjugacy problems in G that are equivalent and provide a foundation of our signature scheme.

Matching Conjugate Search Problem (MCSP)
Instance: A CSP-hard pair (x, x') in G and $y \in G$.
Objective: Find $y' \in G$ such that $y \sim y'$ and $xy \sim x'y'$.

Matching Triple Search Problem (MTSP)
Instance: A CSP-hard pair (x, x') in G and $y \in G$.
Objective: Find a triple $(\alpha, \beta, \gamma) \in G \times G \times G$ such that $\alpha \sim x$, $\beta \sim \gamma \sim y$, $\alpha\beta \sim xy$, and $\alpha\gamma \sim x'y$.

If CSP in G is infeasible, instances of MCSP or MTSP can be given as $x, x', y \in G$ such that $x \sim x'$. In the description of the two matching problems, we do not want to exclude a group where CSP is partially infeasible, that is, the probability that a random conjugate pair (x, x') is CSP-hard is non-negligible. If a conjugate pair (x, x') is not CSP-hard, that is, an element $a \in G$ with $x' = a^{-1}xa$ can be known, then $y' = a^{-1}ya$ is a solution to MCSP and

$(\alpha, \beta, \gamma) = (b^{-1}xb, b^{-1}yb, b^{-1}aya^{-1}b)$ is a solution to MTSP for any $b \in G$ and so the two matching conjugacy problems are feasible. These solutions are said to be *obvious.*

Theorem 1. *In a non-commutative group G, MCSP is feasible if and only if MTSP is feasible.*

Proof. Suppose that MCSP is feasible. Let $\alpha = b^{-1}xb$ and $\beta = b^{-1}yb$ for some $b \in G$, and let γ be a solution to MCSP for the instance (x', α) and y. Then the triple (α, β, γ) is a solution to MTSP.

Suppose MTSP is feasible and MCSP is infeasible. Let (α, β, γ) is a solution to MTSP for a CSP-hard pair (x, x') and y. Since β is a solution to MCSP for a conjugate pair (x, α) and y and MCSP is infeasible, the pair (x, α) is not CSP-hard and so it is feasible to find $b \in G$ such that $\alpha = b^{-1}xb$. Similarly since γ is a solution to MCSP for a conjugate pair (x', α) and y, the pair (x', α) is not CSP-hard and so it is feasible to find $c \in G$ such that $\alpha = c^{-1}x'c$. Then $x' = cb^{-1}xbc^{-1}$ and this contradicts the fact that the pair (x, x') is CSP-hard.

2.3 Description of conjugacy signature scheme

Let G be a non-commutative group where CSP is infeasible and CDP is feasible. We first give a simple conjugacy signature scheme on G and discuss its potential weakness and then we will improve it. Let $h : \{0,1\}^* \to G$ be a hash function, that is, h is a collision-free one-way function that outputs an element of G expressed by a fixed amount of information. For example h can be given by a composition of a usual hash function of bit strings with a conversion from bit strings of a fixed length to elements of G.

Simple conjugacy signature scheme

Key generation: A public key is a CSP-hard pair (x, x') in G and a secret key is a for $x' = a^{-1}xa$.

Signing: Given a message m, a signature σ is given by $\sigma = a^{-1}ya$ for $y = h(m)$.

Verifying: A signature σ is valid if and only if $\sigma \sim y$ and $x'\sigma \sim xy$.

The simple conjugacy signature scheme is a deterministic signature scheme and is clearly based on MCSP. But the secret key a is not zero-knowledge against many known message-signature pairs unless the following problem is infeasible.

k-Simultaneous Conjugator Search Problem (k-SCSP)

Instance: k Pairs $(x_1, x_1'), \ldots, (x_k, x_k') \in G \times G$ such that $x_i' = a^{-1}x_ia$ for all i.

Objective: Find $b \in G$ such that $x_i' = b^{-1}x_ib$ for all i.

It is reasonable to believe that k-SCSP becomes easier as k increases. In particular a solution to CSP is almost unique for the braid groups and so k-SCSP is easier than CSP. We modify the simple conjugacy signature scheme to a randomized signature scheme as follows:

Conjugacy signature scheme

Key generation: A public key is a CSP-hard pair (x, x') in G and a secret key is a for $x' = a^{-1}xa$.

Signing: Given a message m, choose $b \in G$ at random and let $\alpha = b^{-1}xb$ and $y = h(m||\alpha)$, then a signature σ is given by a triple $\sigma = (\alpha, \beta, \gamma)$ where $\beta = b^{-1}yb$ and $\gamma = b^{-1}aya^{-1}b$.

Verifying: A signature σ is valid if and only if $\alpha \sim x$, $\beta \sim \gamma \sim y$, $\alpha\beta \sim xy$, and $\alpha\gamma \sim x'y$.

The conjugacy signature scheme is clearly based on MTSP that is equivalent to MCSP. In the conjugacy signature scheme, the secret key a is zero-knowledge unless 2-SCSP is feasible no matter how many message-signature pairs are known. Indeed b can be known from each message-signature pair if 2-SCSP is feasible and so many (y, aya^{-1}) pairs are known for the secret key a. We propose the conjugacy signature scheme as our new signature scheme on a non-commutative setting and implement it on the braid groups.

3 Gap in braid groups

We will try to convince that CSP and MCSP is hard in the braid groups and construct an efficient oracle for CDP. Consequently we show that the braid groups have a gap between two versions of conjugacy problem.

3.1 Brief introduction to braid group

We introduce necessary terminologies and notations from the braid group rather than to explain the theory of braid. The book [3] seems a good place to start studying braids and the paper [6] contains most of needed implementations of the braid groups.

The n-braid group B_n is presented by the Artin generators $\sigma_1, \ldots, \sigma_{n-1}$ and relations $\sigma_i \sigma_j = \sigma_j \sigma_i$ for $|i-j| > 1$ and $\sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j$ for $|i-j| = 1$. Thus an n-braid b can be written as a word of $\sigma_1, \ldots, \sigma_{n-1}$ and the word-length $|b|$ is the (minimal) number of letters in a word equivalent to b. In particular the word-length of a positive (exponent) word b is the number of letters in b since the relations preserve word-lengths. A positive braid b is a *subword* of a positive braid a, written $b \le a$ if $a = bc$ for some positive or trivial braid c. Clearly '$\le$' gives an partial order on the semigroup of positive braids.

The braid

$$\Delta = (\sigma_1 \cdots \sigma_{n-1})(\sigma_1 \cdots \sigma_{n-2}) \cdots (\sigma_1 \sigma_2)\sigma_1$$

is called the *fundamental braid*. The fundamental braid Δ nearly commutes with any braid b, that is, $\Delta b = \tau(b)\Delta$ where τ is an automorphism of B_n defined by $\tau(\sigma_i) = \sigma_{n-i}$. Since τ^2 is the identity map, Δ^2 truly commutes with any braid. A subword of the fundamental braid Δ is called a permutation braid and the set of all permutation braids is one to one correspondence with the set Σ_n of permutations on $\{0, 1, \ldots, n-1\}$. Thus a permutation braid is denoted by a permutation $\pi : \{0, 1, \ldots, n-1\} \to \{0, 1, \ldots, n-1\}$. For example, Δ is the permutation sending i to $n-i$. The word length of a permutation n-braid is $\le n(n-1)/2$. The *descent set* $D(\pi)$ of a permutation π is defined by $D(\pi) = \{i \mid \pi(i) > \pi(i+1)\}$. Any braid b can be written uniquely as

$$b = \Delta^u \pi_1 \pi_2 \cdots \pi_\ell$$

where u is an integer, π_i are permutation braids such that $\pi_i \neq \Delta$ and $D(\pi_{i+1}) \subset D(\pi_i^{-1})$. This unique decomposition of a braid b is called a *left canonical form*. From the left canonical form of a braid b, the *infimum*, the *supremum*, and the *canonical length* of b are defined as $\inf(b) = u$, $\sup(b) = u + \ell$, and $\ell(b) = \ell$, respectively.

Pseudo-Anosov braids An n-braid can also be thought of as a homeomorphism of n-punctured disk. In this dynamical point of view, braids are classified (up to a multiplication by Δ^2) into the following three types: A braid x is *periodic* if x^m is the trivial braid for some positive integer m and it is *reducible* if a set of more than one strands in x can be separated from the rest of strands by a tube. Finally a braid x is *pseudo-Anosov* if it is neither periodic nor reducible. The following lemma will be useful.

Lemma 1. *If x is pseudo-Anosov and is not a power of another braid then $ax = xa$ implies $a = \Delta^{2i}x^j$ for some non-negative integers i, j.*

Proof. Follows from a standard argument using invariant foliations.

We say a braid is *sufficiently complicated* if it is pseudo-Anosov and it is not a power of another braid. It is not hard to check whether a braid is pseudo-Anosov. A randomly chosen braid of a constant canonical length (not a constant word length) is pseudo-Anosov in a high probability. Since the number of braids is exponential in canonical length and the canonical length of a product of two braids is almost the sum of canonical lengths of two braids, a randomly chosen braid is not a power of another braid. Consequently a randomly chosen braid of a constant canonical length is sufficiently complicated in a high probability.

3.2 Conjugator search problem in braid groups

Although CSP in the braid groups B_n is solvable, it believed to be infeasible as the braid index n increases. It should be noted that CSP may be solved by a polynomial algorithm in n if x is extremely simple (for example, if the word length $|x|$ is a constant smaller than n). A solution for CSP is unique for a sufficiently complicated braid x in the sense that $axa^{-1} = bxb^{-1}$ implies $a = \Delta^{2i}x^jb$ for some non-negative integers i, j by Lemma 1.

For any word W written over k letters, a solution b to k-SCSP satisfies $a^{-1}W(x_1, \ldots, x_k)a = b^{-1}W(x_1, \ldots, x_k)b$ by the homomorphic property of conjugation. If $W(x_1, \ldots, x_k)$ is sufficiently complicated, the converse holds by the uniqueness of solution to CSP. Thus the simultaneity disappears and so CSP is harder than k-SCSP.

If we let $x' = a^{-1}xa$ and $y' = b^{-1}yb$ in MCSP, then MCSP ask to find $b, c \in B_n$ satisfying $a^{-1}xab^{-1}yb = c^{-1}xyc$ for given x, y, a. Since (x, x') is a CSP-hard pair, we have to exclude the case $a = b$. If x and y are sufficiently complicated, we also have to exclude the cases either x commutes with ab^{-1} or y commutes with ab^{-1} since knowing b is equivalent to knowing a up to $\Delta^{2i}x^j$ or $\Delta^{2i}y^j$ by Lemma 1. MCSP seems hard in the remaining possibility for a and b and consequently MCSP seems infeasible as long as an instance (x, x') is CSP-hard and x, y are sufficiently complicated.

Super Summit Set We now discuss a mathematical solution to CSP in B_n given by Garside[9] and improved by Elrifai-Morton[7]. Given $x \in B_n$, let $\mathrm{maxinf}(x)$ be the maximum among infimums of all braids conjugate to x and $\mathrm{minsup}(x)$ be the minimum among supremum of all braids conjugate to x. Clearly maxinf and minsup are conjugacy invariants, that is, if $x \sim y$ then $\mathrm{maxinf}(x) = \mathrm{maxinf}(y)$ and $\mathrm{minsup}(x) = \mathrm{minsup}(y)$. There are two useful conjugations of a braid $x = \Delta^u\pi_1\pi_2\cdots\pi_\ell$ given by its left canonical form. The *cycling* on x is given by $\tau^u(\pi_1)^{-1}x\tau^u(\pi_1)$ and the *decycling* on x is given by $\pi_\ell x\pi_\ell^{-1}$. A series of repeated cycling and repeated decycling on x produces a braid b such that $\inf(b) = \mathrm{maxinf}(x)$ and $\sup(b) = \mathrm{minsup}(x)$.

The super summit set $\mathrm{SSS}(x)$ of x is defined by

$$\mathrm{SSS}(x) = \{b \in B_n \mid b \sim x,\ \inf(b) = \mathrm{maxinf}(x),\ \sup(b) = \mathrm{minsup}(x)\}.$$

Clearly the super summit set itself is a conjugacy invariant. It is not hard to see that $\mathrm{SSS}(x)$ is a finite set with a rough upper bound $(n!)^{\mathrm{minsup}(x)-\mathrm{maxinf}(x)}$. Then a mathematical algorithm to solve CSP for an instance (x, x') can be given by generating $\mathrm{SSS}(x)$ and generating $x'' \in \mathrm{SSS}(x')$ from x' by repeated cycling and repeated decycling and searching x'' in $\mathrm{SSS}(x)$. Obviously the complexity of this algorithm is at least the cardinality $|\mathrm{SSS}(x)|$. None the less this is the best algorithm to solve CSP in the braid group and therefore we use $|\mathrm{SSS}(x)|$ as our security measure for CSP. As mentioned above, MCSP seems hard to solve for a CSP-hard pair (x, x') and so it seems reasonable to use $|\mathrm{SSS}(x)|$ as a security level for MCSP as well.

3.3 Conjugacy decision algorithm in braid groups (BCDA)

A *character* of a non-commutative group G is a function from G to a ring that is invariant under conjugation. Then a character of G will be useful to the conjugacy decision algorithm if we can compute it. A *representation* of G is a homomorphism from G to a matrix group over a ring. Given a matrix M, the characteristic polynomial obtained by the determinant of $M-\lambda I$ is invariant under conjugation on M. Thus for a representation Φ of G, the correspondence $g \mapsto \det(\Phi(g) - \lambda I)$ is a character for any λ in the ring.

Alexander polynomial test Mathematicians and physicists have been interested in the theory of braid for a long time and many representations of the braid groups were found and studied extensively. Good references on representations of the braid groups are the book [3] and the expository article [17]. The most well-understood representation of the n-braid group B_n is the (reduced) Burau representation that sends an n-braid b to an $(n-1)\times(n-1)$ matrix $\Phi(b)$ over the Laurent polynomial ring $\mathbf{Z}[t, t^{-1}]$. Furthermore the Burau representation is one of the fastest to compute. The character of B_n obtained by $\det(\Phi(b) - I)$ is called the *Alexander polynomial* of b, denoted by $P_b(t)$.

For Alexander polynomials $P_a(t), P_b(t)$ of two n-braids a and b, our Alexander polynomial test compares their evaluations at randomly chosen r values $t_1, \ldots, t_r$ of t in a finite field $\mathbf{Z}/p\mathbf{Z}$. The test algorithm outputs "1" if evaluations at all r points agree on two polynomials or outputs "0" otherwise. If the canonical lengths of a, b are $\le \ell$, then the degrees of $P_a(t), P_b(t)$ are $\le \ell n(n-1)/2$ because the degree of the Alexander polynomial is bounded by the word-length. Thus $P_a(t) - P_b(t)$ has at most $\le \ell n(n-1)/2$ roots. Consequently

$$\Pr[P_a(t) \ne P_b(t)\colon \text{Alexander polynomial test outputs } 1] < \left(\frac{\ell n(n-1)}{2p}\right)^r .$$

Therefore the probability that our Alexander polynomial test fails becomes negligible as p and/or r increase.

Given an n-braid b written in $\sigma_1, \ldots, \sigma_{n-1}$, the computation of the Burau matrix is done

by replacing $$\sigma_1 = \begin{bmatrix} -t & & \\ 1 & 1 & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \quad \sigma_i = \begin{bmatrix} \ddots & & & \\ & 1 & t & \\ & & -t & \\ & & 1 & 1 \\ & & & & \ddots \end{bmatrix}, \quad \sigma_{n-1} = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 & t \\ & & & -t \end{bmatrix}.$$

The complexity of this step is $\mathcal{O}(\ell n^3)$ since $|b| = \ell n(n-1)/2$ and each generator corresponds to a column operation involving at most 3 columns. The computation of the determinant can be done in $\mathcal{O}(n^3)$. Thus the complexity of overall Alexander polynomial test is $\mathcal{O}(\ell r n^3)$.

Maxinf-minsup test Given a braid b, $\mathrm{maxinf}(b)$ and $\mathrm{minsup}(b)$ defined in the last section are conjugacy invariants that can be computed efficiently. In fact they can be computed in $\mathcal{O}(\ell^2 n \log n)$ if $\ell = \ell(b)$ is smaller than n according to [4].

Security of conjugacy decision algorithm Our braid conjugacy decision algorithm(BCDA) computes and compares maxinf, minsup, and Alexander polynomials evaluated r random points in $\mathbf{Z}/p\mathbf{Z}$ for r such that $\left(\frac{\ell n(n-1)}{2p}\right)^r$ is smaller than a security requirement.

Alexander polynomials of knots and links are well-understood by knot theorists and $P_b(t)$ agrees with the Alexander polynomial of the link obtained by closing up two ends of the braid b. It is known that many knots or links share the same Alexander polynomial and it is not known that there are two n-braids a and b but one exception such that $\mathrm{Inf}(a) = \mathrm{Inf}(b)$, $\mathrm{Sup}(a) = \mathrm{Sup}(b)$ and $P_a(t) = P_b(t)$. One exceptional case is when a is conjugate to the reverse of b that is the word obtained by flipping the word b written in Artin's generators from left to right. If a is randomly chosen, this exception never occurs.

There are two conceivable attacks to our BCDA. Both attacks are on the Burau representation level. Given a conjugate pair (x, x'), an adversary attempts to find a braid a' such that $\Phi(a')^{-1}\Phi(x)\Phi(a') = \Phi(x')$ and outputs $a'^{-1}ya'$ as a solution to MCSP. Then this adversary will face two obstacles. First the solution space of the $(n-1)^2 \times (n-1)^2$ homogeneous system of linear equations obtained by equating entries in $AX = XB$ for $X = \Phi(a')$, $A = \Phi(x)$, and $B = \Phi(x')$ is of dimension ≥ 1. Since the image of Burau representation does not form a submodule, it is difficult to find the correct image. A more serious difficulty is that there is no known polynomial (in n) algorithm to invert the Burau representation.

The second attack on BCDA is based on the fact that the Burau representation is not injective. Let a' be a braid such that $\Phi(a') = I$. Then y and any conjugate of ya' share the same Alexander polynomial. However ya' likely fails maxinf and minsup test. According to [2], the simplest known example of such a' in B_n for $n \geq 6$ is

$$\begin{aligned} a' = (-8|&543201|251043|542310|031254|145032|215403|542310|103425| \\ &235014|250143|451320|013245|145023|214503|021534|140235) \end{aligned}$$

where the braid given by its left canonical form starts with Δ^{-8} and is followed by 16 permutation n-braids fixing 6 through $n-1$. Since $\mathrm{maxinf}(a') = -8$ and $\mathrm{minsup}(a') = 8$, it is impossible to expect that $\mathrm{maxinf}(ya') = \mathrm{maxinf}(y)$ and $\mathrm{minsup}(ya') = \mathrm{minsup}(y)$.

Also BCDA can solve the decision problem of Diffie-Hellman type in the braid groups considered in [13]. The decision problem of Diffie-Hellman type in the braid groups is to determine whether $y = b^{-1}a^{-1}xab$ for a 4-tuple $(x, a^{-1}xa, b^{-1}xb, y)$ in B_n. Here a is an element of $\lfloor\frac{n}{2}\rfloor$-lower braid subgroup LB_n and b is an element of $\lfloor\frac{n}{2}\rfloor$-upper braid subgroup UB_n in B_n [11, 12]. Using BCDA, an algorithm $\mathcal{D}$ that solves the decision problem can be defined as follows: $\mathcal{D}(x, a^{-1}xa, b^{-1}xb, y) = \mathsf{accept}$ if and only if $x(b^{-1}xb) \sim (a^{-1}xa)y$. If $\mathcal{D}(x, a^{-1}xa, b^{-1}xb, y) = \mathsf{accept}$ but $y \neq a^{-1}b^{-1}xba$, then y is a non-obvious solution of MCSP for the instance $(x, a^{-1}xa)$ and $b^{-1}xb$. Hence the failure probability of this algorithm $\mathcal{D}$ is less than or equal to the success probability of MCSP. As noted in Section 3.2, MCSP seems infeasible as long as an instance $(x, a^{-1}xa)$ is CSP-hard and $x, b^{-1}xb$ are sufficiently complicated. Therefore the success probability of $\mathcal{D}$ seems overwhelming.

4 Implementation of conjugacy signature scheme in braid groups

Since braid groups are infinite and every cryptosystem has to run under finite resources, we first need to establish system parameters to confine the infinite group to a finite environment. Under this restriction, we implement a new signature scheme based on braid groups and prove its equivalence to the based problem. We also give a computational security estimate in terms of system parameters and then discuss efficiency of the scheme under specifically suggested parameters.

4.1 Random braids and braid signature scheme

We first fix positive integers n, ℓ, d as system parameters. Let

$$B_n(\ell) = \{b \in B_n \mid 0 \leq \inf(b), \sup(b) \leq \ell\}.$$

Then $|B_n(\ell)| \leq (n!)^\ell$ and so it is finite. A *random braid generator* produces $b \in_R B_n(\ell)$ in $\mathcal{O}(\ell n)$ time using the random braid generator in [6]. A *bit-string to braid conversion* $c\colon \{0,1\}^N \to B_n(\ell)$ for $N = \ell\lfloor\log_2 n!\rfloor$ can be done in $\mathcal{O}(\ell n)$ time as follows: For a bit string $r \in \{0,1\}^N$, cut r into of ℓ blocks $r_1||r_2||\cdots||r_\ell$ of bit-length $\lfloor\log_2 n!\rfloor$ and then for each $r_i \in [0, n!-1]$, write $r_i = \sum_{k=1}^{n-1} a_k k!$ by recursively dividing r_i by 2 through $n-1$ so that $0 \leq a_k \leq k-1$ and then apply the random braid generator in [6] to the sequence $a_{n-1}, \ldots, a_1$. We think that the values of our random braid generator and bit-string to braid conversion distribute almost uniformly in $B_n(\ell)$ for a small ℓ. We will suggest $\ell = 3$ and so the distribution will not cause much a problem. For a large ℓ, they can be replaced by slower algorithms with better distribution.

For $x, y \in B_n$ such that $x \sim y$, the distance $d(x,y)$ between x and y is defined by $\min\{\ell(b) \mid y = b^{-1}xb\}$. The distance behaves like a metric in a conjugacy class except the fact that $d(x, \tau(x)) = 0$. For example one can show $d(x,y) = d(y,x)$ by using $\inf(b^{-1}) = -\sup(b)$ and $\sup(b^{-1}) = -\inf(b)$.

Random Super Summit Braid Generator From now on, we assume $x \in \mathrm{SSS}(x)$ and $\inf(x) = 0$ and $\sup(x) = \ell$. Then $\mathrm{SSS}(x) \subset B_n(\ell)$. Define the *d-neighborhood* $S(x,d)$ of x in $\mathrm{SSS}(x)$ as follows:

$$S(x,d) = \{y \in \mathrm{SSS}(x) \mid d(x,y) \leq d\}.$$

For a randomly chosen $x' \in S(x,d)$, we will use a conjugate pair (x, x') as a public key. Thus the pair (x,x') must be CSP-hard. The cardinality $|S(x,d)|$ seems an obvious choice for the security level and it will depend on all of n, ℓ, d, x and in particular on the location of x inside $\mathrm{SSS}(x)$. Unfortunately we do not know how to estimate a lower bound for $|S(x,d)|$.

A positive braid a is called a *minimal super summit conjugator* of $x \in \mathrm{SSS}(x)$ if a is minimal among all positive braids b satisfying $b^{-1}xb \in \mathrm{SSS}(x)$ with respect to the partial order '$\leq$'. Since $\Delta^{-1}x\Delta \in \mathrm{SSS}(x)$, a minimal super summit conjugator is a permutation braid. Since any minimal super summit conjugator must be greater than or equal to at least one generator σ_i, there are at most $n-1$ minimal super summit conjugators of a given n-braid x. An algorithm to generate $\mathrm{SSS}(x)$ is proposed in [8] using minimal super summit conjugators. The running time of the algorithm is obviously proportional to the size of $\mathrm{SSS}(x)$.

Consider the directed graph $\Gamma(x)$ where the super summit set $\mathrm{SSS}(x)$ is the set of vertices and there is a directed edge from x_1 to x_2 if $x_2 = a^{-1}x_1a$ for some minimal super summit conjugator a of x_1. We believe that the higher the out-going valency near x in the graph $\Gamma(x)$

is, the larger the d-neighborhood $S(x,d)$ of x is. It is not hard to write an heuristic algorithm to pick a good braid x by investigating valencies.

We now describe how to generate a random braid in $S(x,d)$. This procedure will be called a *random super summit braid generator* denoted by $RSSBG(x,d) = (x',a)$ where $x' \in_R S(x,d)$ and $a \in B_n(d)$ such that $x' = a^{-1}xa$. We first choose $b \in_R B_n(5\ell)$ if $\ell(x) = \ell$. Then we apply a random sequence of cyclings and decyclings to $b^{-1}xb$ until we obtain a braid $a^{-1}xa \in \mathrm{SSS}(x)$. According to [4], the length of this sequence is at most in n^2 and it is much smaller in an average case. If $\ell(a) \le d$, then $a^{-1}xa$ is the output. Otherwise we start over again by choosing new b. Our experiment shows the probability of success on each run is over 70% if $d = \ell + 1$.

We now describe conjugacy signature scheme in braid groups using the system parameters and the random braid generators that we have discussed. Assume that $H : \{0,1\}^* \to B_n(\ell)$ is an one-way hash function obtained by composing an ordinary hash function : $\{0,1\}^* \to \{0,1\}^N$ with the bit-string to braid conversion c.

Braid Signature Scheme (BSS)

Key Generating Algorithm: $Gen(n,\ell,d) = (pk, sk)$:

1. Select a braid $x \in B_n(\ell)$ such that $x \in \mathrm{SSS}(x)$;
2. Choose $(x',a) \underset{R}{\leftarrow} RSSBG(x,d)$;
3. Return $pk = (x,x')$ and $sk = a$.

Signing Algorithm: $Sig(m, pk, sk) = \sigma$:

1. Choose $(\alpha = b^{-1}xb, b) \underset{R}{\leftarrow} RSSBG(x,d)$;
2. Compute $h = H(m||\alpha)$ for a message m and let $\beta = b^{-1}hb$ and $\gamma = b^{-1}aha^{-1}b$;
3. Return a signature $\sigma = (\alpha,\beta,\gamma) \in B_n(\ell) \times B_n(\ell+2d) \times B_n(\ell+4d)$.

Verifying Algorithm: $Ver(pk, m, \sigma) = \{\mathsf{accept}|\mathsf{reject}\}$:

1. Compute $h = H(m||\alpha)$.
2. Return accept if and only if $\alpha \sim x,\ \beta \sim h,\ \gamma \sim h,\ \alpha\beta \sim xh,$ and $\alpha\gamma \sim x'h$.

4.2 Security of braid signature

In this section, we will show the security of the Braid Signature Scheme(BSS). We will adapt the definitions of [1] and [10] where concrete security analysis of digital signatures were performed.

We consider two different scenarios, no-message attack and adaptive chosen-message attack. The forger $\mathcal{F}$'s advantage $\mathsf{Adv}(\mathcal{F})$ in two attack scenarios is defined to be

$$\mathsf{Adv}(\mathcal{F}) = \Pr[(pk,sk) \underset{R}{\leftarrow} Gen(n,k,\ell,r); (m,\sigma) \leftarrow \mathcal{F}^{\mathcal{O}}(pk) : Ver(pk,m,\sigma) = \mathsf{accept}].$$

where $\mathcal{O}$ is the null in no-message attack or is the signing oracle $\mathcal{S}$ in adaptive chosen-message attack. In an adaptive chosen-message attack model, a forger $\mathcal{F}$ may query to the signing oracle. Thus we need to simulate the signing algorithm to complete a security proof. Unfortunately we are able to simulate it only after making BSS inefficient. Since this modification involves other technical ingredients, it seems inappropriate to discuss it in this article.

A forger $\mathcal{F}$ via no-message attack is said to (t, q_H, ε)-break the signature scheme (Gen, Sig, Ver) if $\mathcal{F}$ runs in time at most t, makes at most q_H queries to the hash function, and $\mathsf{Adv}(\mathcal{F}) \ge \varepsilon$. To provide the security proof against no-message attack in the random oracle model, we use the random braid generator and a random hash function H. Here the hash function $H : \{0,1\}^* \to B_n(\ell)$ are random in the sense that all outputs of H are uniformly distributed in $B_n(\ell)$. On the other hand, The advantage of an algorithm $\mathcal{A}$ in solving MTSP is defined as follows:

$$\mathsf{Succ}(\mathcal{A}) = \Pr[x \sim x', y \underset{R}{\leftarrow} B_n(\ell) : \mathcal{A}(x, x', y) = \text{ a solution of MTSP}].$$

We say that an algorithm $\mathcal{A}$ (t, ε)-solves MTSP if $\mathcal{A}$ runs in time at most t and $\mathsf{Succ}(\mathcal{A}) \geq \varepsilon$.

Theorem 2. *If there exists a (t, q_H, ε)-forger $\mathcal{F}$ against no-message attack for BSS, then there exists an (t', ε')-algorithm $\mathcal{A}$ solving MTSP where*

$$\varepsilon' = \frac{1}{q_H}\varepsilon \text{ and } t' = t.$$

Proof. Suppose that a forger $\mathcal{F}$ via no-message attack (t, q_H, ε)-breaks the proposed scheme. We will use $\mathcal{F}$ to construct an algorithm $\mathcal{A}$ that (t', ε')-solves MTSP. Suppose that $\mathcal{A}$ is given a challenge $(x, x' = a^{-1}xa) \in B_n \times B_n$ and $y \in B_n$, where $x, x' \in S(x, d)$ and y is chosen at random in $B_n(\ell)$. Then $\mathcal{A}$ runs $\mathcal{F}$ with public key $pk = (x, a^{-1}xa)$ and a hash function $H : \{0, 1\}^* \to B_n(\ell)$. $\mathcal{F}$ makes hash oracle queries during its execution. Because the hash values $H(m||\alpha)$ depend on the message m and α of the corresponding signature, $\mathcal{F}$ has already compute α when the hash oracle query $m||\alpha$ is requested. $\mathcal{A}$ picks an integer i_0 from $\{1, \cdots, q_H\}$ at random.

Suppose $\mathcal{F}$ makes a hash oracle query on $m_i||\alpha_i$ for $1 \leq i \leq q_H$. If $i = i_0$, then $\mathcal{A}$ returns $h_{i_0} = y$ as a hash value of $m_{i_0}||\alpha_{i_0}$. Otherwise, $\mathcal{A}$ chooses a random braid $k_i \in B_n(\ell)$ returns $h_i = k_i$ as the hash value of $m_i||\alpha_i$. Since the hash values h_i are uniformly distributed in $B_n(\ell)$, h_i's act like random braids.

Eventually $\mathcal{F}$ halts and outputs a message-signature pair $(m^*, (\alpha^*, \beta^*, \gamma^*))$. Without loss of generality we may assume that $\mathcal{F}$ has requested the hash query $m^*||\alpha^*$ before. Therefore $m^*||\alpha^* = m_i||\alpha_i$ for some i. If $i \neq i_0$, then $\mathcal{A}$ outputs $\mathsf{failture}$ and halts. Otherwise, $\mathcal{A}$ outputs $(\alpha^*, \beta^*, \gamma^*)$ as a solution of MTSP given by $x \sim x'$ and y.

If $(m^*, (\alpha^*, \beta^*, \gamma^*))$ is a valid forgery, then it satisfies the following relations:

$$\alpha^* \sim x,\ \beta^* \sim y,\ \gamma^* \sim y,\ \alpha^*\beta^* \sim xy \text{ and } \alpha^*\gamma^* \sim x'y,$$

where $H(m^*||\alpha^*) = y$. Therefore $(\alpha^*, \beta^*, \gamma^*)$ is a solution of MTSP and $\mathsf{Succ}(\mathcal{A}) = \varepsilon' = \frac{1}{q_H}\mathsf{Adv}(\mathcal{F}) = \frac{1}{q_H}\varepsilon$. Since the running time of $\mathcal{A}$ is equal to the running time of $\mathcal{F}$, $t' = t$.

4.3 Parameters suggestion and performance

Among system parameters of BSS, we set $\ell = 3$ and $d = 4$. We use the braid index n as the security parameter. The theoretical security level of BSS is the size of the 4-neighborhood $S(x, 4)$ of x in its super summit set. If an adversary uses a super summit braid generator similar to the one proposed in [8] to recover a secret key by solving CSP, then the search time is proportional to B^D where B is the average of out-going valencies and D is the average radius in the graph of $S(x, 4)$ explained in Section 4.1. For a good choice of x, our experiment shows that $B^D \approx (\frac{n}{4})^{\frac{n(n-1)}{2}}$. The graph of $S(x, 4)$ is never a tree and there will be lots of repetition if we count the number of vertices in $S(x, 4)$ using B^D. We will suggest $B^{D/2}$ as an experimental lower bound of the size of $S(x, 4)$. We recommend the security parameter $n = 20$, $n = 24$ and $n = 28$ depending on a use of BSS.

As explained in Section 3.3, BCDA containing the maxinf-minsup test and the Alexander polynomial test are used to decide conjugacy. In the Alexander polynomial test, the following parameter are suggested: p is a prime such that $2^{31} < p < 2^{32}$, the number of round r is 3. Then the probability that the test fails is much less than $1/2^{80}$.

4.4 Performance table

The following table shows the sizes of a public key and a signature, the time to generate a pair of public and private keys, to sign a message and to verify a signature for the suggested security parameters. The security level is based on our rough estimate $(\frac{n}{4})^{\frac{n(n-1)}{4}}$ for the size of $S(x, 4)$. This experiment was done on a computer with a Pentium III 866MHz processor.

n	Public Key Size	Signature Size	Key Generation Time	Signing Time	Verifying Time	Security Level
20	370 bit	1653 bit	17.82 ms	18.68 ms	30.87 ms	2^{220}
24	478 bit	2138 bit	21.70 ms	22.79 ms	41.75 ms	2^{356}
28	591 bit	2648 bit	24.42 ms	25.77 ms	59.59 ms	2^{530}

Table 1. Performance of Braid Signature Scheme

5 Further study

A good lower bound for the size of the d-neighborhood of x should be found to give a better understanding of the security level of BSS. Our current algorithm to select a good x with a large d-neighborhood is heuristic and so is relatively slow. It takes about 1 second for $n = 20$. Even though x can be generated in advance and can be used for many distinct pairs of public and private keys, we think that there is a lot of room to improve this algorithm.

References

1. M. Bellare and P. Rogaway, *The Exact Security of Digital Signatures-How to Sign with RSA and Rabin*, Proc. of Eurocrypt 96, LNCS **1070**, Springer-Verlag (1996) 399–416.
2. S. Bigelow, *The Burau representation is not faithful for* $n = 5$, Geom. Topol. **3** (1999) 397-404.
3. J. S. Birman, *Braids, links and mapping class groups*, Annals of Math. Study **82**, Princeton University Press (1974).
4. J. S. Birman, K. H. Ko and S. J. Lee, *The infimum, supremum and geodesic length of a braid conjugacy class*, to appear in Advances in Mathematics.
5. D. Boneh, B. Lynn and H. Shacham, *Short signatures from the Weil pairing*, Proc. of Asiacrypt 2001, LNCS **2248**, Springer-Verlag (2001) 533–551.
6. J. C. Cha, K. H. Ko, S. J. Lee, J. W. Han and J. H. Cheon, *An Efficient Implementation of Braid Groups*, Proc. of Asiacrypt 2001, LNCS **2248**, Springer-Verlag (2001) 144–156.
7. E. A. Elrifai and H. P. Morton, *Algorithms for positive braids*, Quart. J. Math. **45** (1994) 479–497.
8. N. Franco and J. Gonzalez-Meneses, *Conjugacy problem for braid groups and Garside groups*, preprint http://www.arxiv.org/abs/math.GT/0112310.
9. F. A. Garside, *The braid group and other groups*, Quart. J. Math. **20** (1969) 235–254.
10. S. Goldwasser, S. Micali and R. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM J. of Computing **17** (1988) 281–308.
11. K. H. Ko, M. S. Cho, D. H. Choi and J. W. Han, *Provably Secure Public-key Encryption Scheme Based on Braid Groups*, Preprint (2001).

12. K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. S. Kang and C. S. Park, *New Public-Key Crytosystem Using Braid Groups*, Proc. of Crypto 2000, LNCS **1880**, Springer-Verlag (2000) 166–183.
13. E. K. Lee, S. J. Lee, S. G. Hahn, *Pseudorandomness from Braid Groups*, Proc. of Crypto 2001, LNCS **2139**, Springer-Verlag (2001) 486–502.
14. D. Pointcheval and J. Stern, *Security Proofs for Signature Schemes*, Proc. of Eurocrypt 96, LNCS **1070**, Springer-Verlag (1996) 387-398.
15. T. Okamoto and D. Pointcheval, *The Gap-Problems: A New Class of Problems for the Security of Cryptographic Schemes*, Proc. of PKC 2001, LNCS **1992**, Springer-Verlag (2001) 104–118.
16. S. H. Paeng, K. C. Ha, J. H. Kim, S. Chee and C. Park, *New Public Key Cryptosystem Using Finite Non Abelian Groups*, Proc. of Crypto 2001, LNCS **2139**, Springer-Verlag (2001) 470–485.
17. V. Turaev, *Faithful linear representations of the braid groups*, www.arxiv.org, math.QA/0103017 (2000).