Introduction to Braid Group Cryptography

Parvez Anandam

March 7, 2006

1 Introduction

Public key cryptosystems rely on certain problems for which no fast algorithms are known. For instance, in Diffie-Hellman, it is the discrete logarithm problem, and in RSA, it is the factoring problem. Most successful public key cryptosystems are based on arithmetic over finite fields (a survey of public key cryptosystems is given by Koblitz and Menezes [1]).

Cryptosystems in current use have a drawback: they succumb easily to quantum computers. Whether or not quantum computing becomes a reality soon does not diminish the imperative of finding other hard problems, to increase the "genetic diversity" of public key cryptography.

The past seven years have seen the rise and wane in popularity of cryptosystems based on the braid groups. As we will see, braid groups have certain properties that make them easily amenable to digital computation. The initial excitement was based on the hardness of the conjugacy search problem (which we describe below) in braid groups. This problem was later shown to be more tractable than originally thought. It is still possible that braid group cryptography is secure for certain choices of parameters, but such parameters have not yet been found.

Nevertheless, the experience gained from studying the use of braid groups in cryptography is valuable. It is conceivable that some nonabelian group will someday play a role in public key cryptography.

2 Braid Groups

The braid group B_n is an infinite, nonabelian group of n braids. A member of the braid group B_n has a simple geometric interpretation. Visualize n strings joining n points at the top to n points at the bottom, not necessarily vertically. Keeping the ends of the strings fixed, imagine crossing these strings zero or more times. An example is shown in Figure 1. Braids can be described using the generators σ_i (Figure 2) of the group B_n.

The braid group B_n is given by the (Artin) presentation

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j = \sigma_j\sigma_i & \text{for } |i-j| \geq 2 \\ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{for } |i-j| = 1 \end{array} \right\rangle \tag{1}$$

A property of braid group elements that makes them easy to digitize is that they can be uniquely represented in a convenient form. We will need to define a few notions to describe this unique representation.

Consider the monoid B_n^+ (a monoid satisfies all the requirements of a group except the existence of inverses). Elements of B_n^+ can be written as words in only the σ_i^{+1} (not the

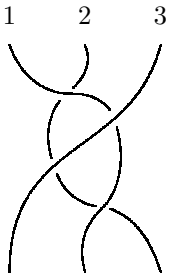

Figure 1: The 3-braid $\sigma_1^{-1}\sigma_2\sigma_1\sigma_2 = \sigma_2\sigma_1$

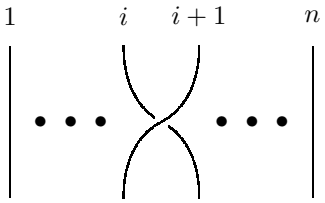

Figure 2: The generator σ_i

σ_i^{-1}) under the same relations as the group B_n shown in Eq. (1). These elements are called *positive* braids and are used to define an order relation between braids:

$$x \leq y \text{ if } y = axb, \text{ with } x, y \in B_n \text{ and } a, b \in B_n^+ \tag{2}$$

We next define the *fundamental braid* Δ of B_n:

$$\Delta = (\sigma_1 \cdots \sigma_{n-1})(\sigma_1 \cdots \sigma_{n-2}) \cdots \sigma_1 \tag{3}$$

A positive braid A satisfying $A \leq \Delta$ is called a *canonical factor*. To each canonical factor A we can associate a permutation π: the upper point i of the braid is connected to the lower point $\pi(i)$, with every crossing being positive. A canonical factor is sometimes also called a *permutation braid*.

A decomposition $P = A_0 P_0$ of a positive braid P into a canonical factor A_0 and a positive braid P_0 is *left-weighted* if A_0 has the maximal length (maximal in "$\leq$") among all such decompositions.

Any braid $b \in B_n$ can be uniquely decomposed into

$$b = \Delta^u A_1 A_2 \ldots A_l \tag{4}$$

with $A_i A_{i+1}$ left-weighted for each i in $1 \leq i \leq l-1$. This unique representation is called the *left-canonical form* [4] or *greedy normal form* [5] of the braid b. The number of canonical factors l is called the *canonical length* of the braid b. The braid index n and the canonical length l are the main security parameters of a braid group cryptosystem.

Finally, we define two commuting subgroups of B_n that will be used in the protocols. Let LB_n be the subgroup of B_n generated by $\sigma_1, \ldots, \sigma_{\lfloor n/2 \rfloor - 1}$. Let UB_n be the subgroup of B_n generated by $\sigma_{\lfloor n/2 \rfloor + 1}, \ldots, \sigma_{n-1}$. Any element $a \in LB_n$ commutes with any element $b \in UB_n$: $ab = ba$.

We now have the algebraic arsenal necessary to construct braid group cryptosystems.

3 Conjugacy search problems

A hard problem is the underpinning of any public key cryptosystem. There are several (apparently) hard problems in braid groups. We will focus on variants of the conjugacy search problem, around which all the braid group cryptosystems proposed to date are built.

3.1 Conjugacy Search Problem

Given $x, y \in B_n$ such that $y = a^{-1}xa$ for some $a \in B_n$.
Find $b \in B_n$ such that $y = b^{-1}xb$.

3.2 Generalized Conjugacy Search Problem

Given $x, y \in B_n$ such that $y = a^{-1}xa$ for some $a \in LB_n$.
Find $b \in LB_n$ such that $y = b^{-1}xb$.
(This problem can also be stated with $a, b \in UB_n$)

3.3 Diffie-Hellman type Generalized Conjugacy Search Problem

Given $x, y_A, y_B \in B_n$ such that $y_A = a^{-1}xa$ and $y_B = b^{-1}xb$ for some $a \in LB_n$ and $b \in UB_n$.
Find $b^{-1}y_Ab (= a^{-1}y_Ba = a^{-1}b^{-1}xab)$.

3.4 Multiple Simultaneous Conjugacy Search Problem

Given $x_i, y_i \in B_n, 1 \leq i \leq t$ such that $y_i = a^{-1}x_ia$ for some $a \in B_n$.
Find $b \in B_n$ such that $y_i = b^{-1}x_ib$ for all i.

4 Commutator based key agreement

In 1999, Anshel, Anshel and Goldfeld proposed [2] a key agreement protocol that is based on the multiple simultaneous conjugacy search problem. This protocol is called the Arithmetica key exchange and is patented [3].

1. Public information
   (a) The braid index n is published.
   (b) A(lice) publishes the subgroup $G_A = \langle x_1, \ldots, x_s \rangle \subseteq B_n$ by specifying the generators $x_1, \ldots, x_s$.
   (c) B(ob) publishes the subgroup $G_B = \langle y_1, \ldots, y_t \rangle \subseteq B_n$ by specifying the generators $y_1, \ldots, y_t$.
2. Key agreement
   (a) A selects a secret word $a = W(x_1, \ldots, x_s) \in G_A$ and sends $a^{-1}y_1a, \ldots, a^{-1}y_ta$ to B.
   (b) B selects a secret word $b = V(y_1, \ldots, y_t) \in G_B$ and sends $b^{-1}x_1b, \ldots, b^{-1}x_sb$ to A.
   (c) A computes the shared key $K = a^{-1}(b^{-1}ab) = a^{-1}W(b^{-1}x_1b, \ldots, b^{-1}x_sb)$, which is the commutator of a and b, $[a, b]$.

(d) B computes the shared key $K = (a^{-1}b^{-1}a)b = V^{-1}(a^{-1}y_1a, \ldots, a^{-1}y_ta)b$, which is the commutator of a and b, $[a, b]$.

This protocol works because the product of conjugates is the conjugation (by the same element) of products: $(a^{-1}xa)(a^{-1}ya) = a^{-1}xya$.

Anshel *et al* suggested using $n = 80$ and $s = t = 20$ generators for each subgroup. Each of these generators are comprised of an average of five Artin generators. The secret words a and b have a length of 128 in the public generators. This choice of parameters was later found to not be secure [10].

5 Diffie-Hellman type key agreement

In 2000, Ko *et al* proposed [4] a key agreement protocol based on the Diffie-Hellman type Generalized Conjugacy Search Problem.

1. Public information
   (a) A sufficiently complicated braid $x \in B_n$ is published, along with the braid index n.
2. Key agreement
   (a) A(lice) selects $a \in LB_n$ (A's private key) and sends $y_A = a^{-1}xa$ (A's public key) to B.
   (b) B(ob) selects $b \in UB_n$ (B's private key) and sends $y_B = b^{-1}xb$ (B's public key) to A.
   (c) A receives y_B and computes the shared key $K = a^{-1}y_Ba = a^{-1}b^{-1}xab$.
   (d) B receives y_A and computes the shared key $K = b^{-1}y_Ab = a^{-1}b^{-1}xab$.

There are strong parallels between this key agreement and the Diffie-Hellman key agreement. The braid x is analogous to the integer g and conjugation $a^{-1}xa$ replaces exponentiation g^a.

Ko *et al* suggested a few instances of the security parameters, e.g. $(n = 50, l = 5), (n = 70, l = 7), (n = 90, l = 12)$, where n is the braid index and l the canonical length of x, a and b . These have unfortunately not stood up to attacks [9, 10].

6 ElGamal type encryption

Let $H : B_n \to \{0, 1\}^k$ be an ideal hash function from the braid group B_n to the message space $\{0, 1\}^k$.

1. Public information
   (a) A sufficiently complicated braid $x \in B_n$ is published, along with the braid index n.
2. Public key
   (a) A(lice) selects $a \in LB_n$ (A's private key) and publishes $y = a^{-1}xa$ (A's public key).
3. Encryption: B(ob) wishes to encrypt a message $m \in \{0, 1\}^k$ to send to A.

(a) B selects $b \in UB_n$ and sends (c, d) to A, where $c = b^{-1}xb$ and $d = H(b^{-1}yb) \oplus m$.

4. Decryption: A wishes to decrypt the ciphertext (c, d).

   (a) A computes $m = H(a^{-1}ca) \oplus d$.

Again, there are obvious parallels between this encryption and the ElGamal encryption over finite fields.

7 Toy example of the Diffie-Hellman type key agreement

It is instructive to examine a toy example of the Diffie-Hellman type key agreement to get a feel for the implementation on a digital computer. We use the C++ `CBraid` library [6] of Cha *et al* [7] to perform the braid group operations.

All braids in this example are in left-canonical form (c.f. Eq. (4)). Each braid is denoted by u followed by the permutations representing the canonical factors A_i.

The security parameters of the key agreement are the braid index n and the canonical length l of x, a and b. For this toy example, we choose $n = 8$ and $l = 4$, i.e. we choose braids x, a and b at random, each with 4 canonical factors expressed as permutations of 8 elements.

The algorithms implemented in `CBraid` have the following complexities:

- Product of two braids: $\mathcal{O}(ln)$.
- Inverse of a braid: $\mathcal{O}(ln)$.
- Rewriting a braid in left-canonical form: $\mathcal{O}(l^2 n \log n)$.
- Comparison of two braids: $\mathcal{O}(l^2 n \log n)$.
- Generating a random braid: $\mathcal{O}(ln)$

Below is the output of our toy program.

```
Braid Index: n = 8
Canonical Length of x, a, b: l = 4

Public x:
(u =  2|4 3 8 2 6 7 1 5|4 2 7 1 3 5 6 8|3 2 5 4 6 8 1 7|1 2 3 5 4 6 7 8|)

A's private key a:
(u =  0|4 1 3 2 5 6 7 8|3 4 2 1 5 6 7 8|2 1 3 4 5 6 7 8|4 1 2 3 5 6 7 8|)

B's private key b:
(u =  0|1 2 3 4 8 7 6 5|1 2 3 4 5 7 8 6|1 2 3 4 5 7 6 8|1 2 3 4 6 7 5 8|)

A's public key y_A = a^-1 * x * a:
(u = -2|7 6 5 8 4 3 2 1|8 7 6 5 4 3 1 2|7 8 6 5 4 3 2 1|8 7 6 5 3 2 1 4|
        3 7 2 8 5 6 1 4|5 2 3 4 6 8 1 7|3 1 2 5 4 6 7 8|1 4 3 5 2 6 7 8|
        3 4 2 1 5 6 7 8|4 1 2 3 5 6 7 8|)

B's public key y_B = b^-1 * x * b:
```

```
(u = -1|8 6 5 7 4 3 2 1|8 7 6 5 4 2 3 1|7 5 8 6 4 3 2 1|4 3 8 2 1 5 6 7|
        4 2 7 1 3 5 6 8|3 2 5 4 7 8 1 6|1 2 3 8 4 6 5 7|1 2 3 4 5 7 8 6|
        1 2 3 4 5 7 6 8|1 2 3 4 6 7 5 8|)

A computes the shared key K = a^-1 * y_B * a:
(u = -2|7 6 5 8 4 3 2 1|8 6 5 7 4 3 1 2|7 8 6 5 4 2 3 1|7 5 8 6 3 2 1 4|
        3 7 2 8 1 4 5 6|5 2 3 4 7 8 1 6|3 1 2 5 4 7 6 8|1 4 3 8 2 5 6 7|
        3 4 2 1 5 7 8 6|4 1 2 3 5 7 6 8|1 2 3 4 6 7 5 8|)

B computes the shared key K = b^-1 * y_A * b:
(u = -2|7 6 5 8 4 3 2 1|8 6 5 7 4 3 1 2|7 8 6 5 4 2 3 1|7 5 8 6 3 2 1 4|
        3 7 2 8 1 4 5 6|5 2 3 4 7 8 1 6|3 1 2 5 4 7 6 8|1 4 3 8 2 5 6 7|
        3 4 2 1 5 7 8 6|4 1 2 3 5 7 6 8|1 2 3 4 6 7 5 8|)
```

We see that both A and B compute the same key K, as expected.

The way a is chosen is to construct a braid with random factors, each of which is a permutation of the lower four elements only (therefore, the upper four elements are always `5 6 7 8`). The braid b is constructed in a similar manner, using permutations of only the upper four elements of the canonical factors (so that the lower four are always `1 2 3 4`). This construction ensures that $a \in LB_n$ and $b \in UB_n$.

8 Attacks

We will not describe in any detail the attacks found against braid group cryptosystems but will merely mention them.

The commutator based key agreement is vulnerable to length-based attacks [8]. The Conjugacy Search Problem is not vulnerable to those attacks.

The Diffie-Hellman type key agreement is vulnerable to a polynomial-time attack that uses the Lawrence-Krammer representation of B_n [9]. The Conjugacy Problem is not vulnerable to those attacks.

There are also heuristic algorithms [10] that attack those two key agreement protocols.

The Conjugacy Search Problem itself is weakened by refinements [11] to the summit set (a finite set of conjugates for every braid) that yield fast probabilistic algorithms for solving the Conjugacy Search Problem.

9 Conclusion

Braid groups provide an elegant framework for designing new public key cryptosystems that can be efficiently implemented on a digital computer. These cryptosystems suffer but a minor drawback: they are not secure!

Even within the confines of braid groups, it may still be possible to construct a secure cryptosystem by an appropriate choice of the security parameters; further investigation is needed. In addition, there are problems besides the conjugacy search problems that could be used to design a secure cryptosystem. The p-th root problem or the Markov problem are candidates.

One may also look for other nonabelian groups where the conjugacy search problem is hard. The word problem (determining whether two elements of the group, expressed as words, are the same element) in such a group has to be easily solvable. Even further, being able to quickly express any word in a canonical form would be greatly desirable. Also, one should be able to describe a group element as a compact string of bits, that a digital

computer can handle. Finally, there should be efficient algorithms to perform the group operations.

One may even go so far as consider using continuous groups for constructing public key cryptosystems [12].

Groups have been extensively studied and it is natural to try to use them. It is abundantly clear, however, that a successful group-based cryptosystem must survive years of intense cryptanalytic effort before it is broadly deployed.

References

[1] Neal Koblitz and Alfred J. Menezes, *A survey of public-key cryptosystems*, SIAM Review **46** (2004) 599-634.

[2] Iris Anshel, Michael Anshel, and Dorian Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters **(**6) (1999) 287-291.

[3] Iris Anshel, Michael Anshel, and Dorian Goldfeld, *Method and apparatus for cryptographically secure algebraic key establishment protocols based on monoids*, United States Patent 6,493,449 (2002).

[4] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju-sung Kang, and Choonsik Park, *New Public-Key Cryptosystem Using Braid Groups*, CRYPTO 2000, 166-184, Springer Lecture Notes in Computer Science 1880 (2000).

[5] Patrick Dehornoy, *Braid-based cryptography*, Contemporary Mathematics **360** (2004) 5-33.

[6] Jae Choon Cha, *CBraid: a C++ library for computations in braid groups*, available at `http://knot.kaist.ac.kr/~jccha/cbraid/` (2001).

[7] Jae Choon Cha, Ki Hyoung Ko, Sang Jin Lee, Jae Woo Han, and Jung Hee Cheon, *An efficient implementation of braid groups*, AsiaCrypt 2001, 144-156, Springer Lecture Notes in Computer Science 2048 (2001).

[8] James Hughes and Allen Tannenbaum, *Length-Based Attacks for certain group based encryption rewriting systems*, Workshop SECI02 SEcurité de la Communication sur Internet (2002).

[9] Jung Hee Cheon and Byungheup Jun, *A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem*, CRYPTO 2003, 212-225, Springer Lecture Notes in Computer Science 2729 (2003).

[10] Dennis Hofheinz and Rainer Steinwandt, *A practical attack on some braid group based cryptographic primitives*, PKC 2003, Springer Lecture Notes in Computer Science 2567 (2003).

[11] Volker Gebhardt, *A new approach to the conjugacy problem in Garside groups*, preprint, `http://arxiv.org/abs/math.GT/0306199` (2003).

[12] Arkady Berenstein and Leon Chernyak, *Geometric Key Establishment*, preprint, `http://eprint.iacr.org/2004/239` (2004).