Braid Group Cryptography

May 5, 2009

Practical Public-key Cryptosystems

- Diffie-Hellman
- RSA
- ElGamal
- Elliptic Curve Cryptosystems, etc

Quantum computers

- The speed of quantum computers threatens the security of the encryption schemes.

Quantum computers

- The speed of quantum computers threatens the security of the encryption schemes.
- Shor's [1994] algorithm for factoring and solving dicrete log.

Quantum computers

- The speed of quantum computers threatens the security of the encryption schemes.
- Shor's [1994] algorithm for factoring and solving dicrete log.
- Chuang et al [2001] implemented Shor's algorithm on 7-qubit quantum computer.

Quantum computers

- The speed of quantum computers threatens the security of the encryption schemes.
- Shor's [1994] algorithm for factoring and solving dicrete log.
- Chuang et al [2001] implemented Shor's algorithm on 7-qubit quantum computer.
- NIST [2004] built a 10-qubit quantum computer.

Quantum computers

- The speed of quantum computers threatens the security of the encryption schemes.
- Shor's [1994] algorithm for factoring and solving dicrete log.
- Chuang et al [2001] implemented Shor's algorithm on 7-qubit quantum computer.
- NIST [2004] built a 10-qubit quantum computer.
- QCL (Oemer 2003), quantum programming language.

Quantum computers

- The speed of quantum computers threatens the security of the encryption schemes.
- Shor's [1994] algorithm for factoring and solving dicrete log.
- Chuang et al [2001] implemented Shor's algorithm on 7-qubit quantum computer.
- NIST [2004] built a 10-qubit quantum computer.
- QCL (Oemer 2003), quantum programming language.

What kinds of group-theoretic problems can be used that the current classical computers and the future quantum computers can not solve?

Some alternative basis for public key cryptography

- Conjugacy search problem and related problems in braid groups.

Some alternative basis for public key cryptography

- Conjugacy search problem and related problems in braid groups.
- Problem of solving multivariate systems of polynomials in finite fields

Some alternative basis for public key cryptography

- Conjugacy search problem and related problems in braid groups.
- Problem of solving multivariate systems of polynomials in finite fields

It is unclear when these or other algebraic problems will be well enough understood to produce practical public key cryptographic primitives with reliable security estimates.

What is a group?

A group is a set, G, together with an operation $*$ that combines any two elements a and b to form another element denoted $a * b$. To qualify as a group, the set and operation, $(G, *)$, must satisfy four requirements

What is a group?

A group is a set, G, together with an operation $*$ that combines any two elements a and b to form another element denoted $a * b$. To qualify as a group, the set and operation, $(G, *)$, must satisfy four requirements

- For all $a, b \in G$, $a * b \in G$.

What is a group?

A group is a set, G, together with an operation $*$ that combines any two elements a and b to form another element denoted $a * b$. To qualify as a group, the set and operation, $(G, *)$, must satisfy four requirements

- For all $a, b \in G$, $a * b \in G$.
- For all $a, b, c \in G$, the equation $(a * b) * c = a * (b * c)$ holds.

What is a group?

A group is a set, G, together with an operation $*$ that combines any two elements a and b to form another element denoted $a * b$. To qualify as a group, the set and operation, $(G, *)$, must satisfy four requirements

- For all $a, b \in G$, $a * b \in G$.
- For all $a, b, c \in G$, the equation $(a * b) * c = a * (b * c)$ holds.
- There exists an element $e \in G$, such that for all elements $a \in G$, the equation $e * a = a * e = a$ holds.

What is a group?

A group is a set, G, together with an operation $*$ that combines any two elements a and b to form another element denoted $a * b$. To qualify as a group, the set and operation, $(G, *)$, must satisfy four requirements

- For all $a, b \in G$, $a * b \in G$.
- For all $a, b, c \in G$, the equation $(a * b) * c = a * (b * c)$ holds.
- There exists an element $e \in G$, such that for all elements $a \in G$, the equation $e * a = a * e = a$ holds.
- For each $a \in G$, there exists an element $b \in G$ such that $a * b = b * a = e$, where e is the identity element.

What is a generator set of a group?

A subset H of G is a subgroup of $(G, *)$ if H is a group under the operation $*$.

What is a generator set of a group?

A subset H of G is a subgroup of $(G, *)$ if H is a group under the operation $*$.

If S is a subset of a group G, then $< S >$, the subgroup generated by S, is the smallest subgroup of G containing every element of S.

What is a generator set of a group?

A subset H of G is a subgroup of $(G, *)$ if H is a group under the operation $*$.

If S is a subset of a group G, then $< S >$, the subgroup generated by S, is the smallest subgroup of G containing every element of S.

If $G =< S >$, then we say S generates G; and the elements in S are called generators.

What is a generator set of a group?

A subset H of G is a subgroup of $(G, *)$ if H is a group under the operation $*$.

If S is a subset of a group G, then $< S >$, the subgroup generated by S, is the smallest subgroup of G containing every element of S.

If $G =< S >$, then we say S generates G; and the elements in S are called generators.

If S is one element set, then we say that $G =< S >$ is a cyclic group.

What is a generator set of a group?

A subset H of G is a subgroup of $(G, *)$ if H is a group under the operation $*$.

If S is a subset of a group G, then $< S >$, the subgroup generated by S, is the smallest subgroup of G containing every element of S.

If $G =< S >$, then we say S generates G; and the elements in S are called generators.

If S is one element set, then we say that $G =< S >$ is a cyclic group.

Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

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

Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

$$\left\langle \sigma_1, \dots, \sigma_{n-1} \;\middle|\; \begin{array}{l} \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geq 2, \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } |i-j| = 1 \end{array} \right\rangle$$

- The presentation is called the *Artin presentation*.

Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

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

- The presentation is called the *Artin presentation*.
- The generators are called the *Artins generators*.

Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

$$\left\langle \sigma_1, \dots, \sigma_{n-1} \;\middle|\; \begin{array}{l} \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geq 2, \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } |i-j| = 1 \end{array} \right\rangle$$

- The presentation is called the *Artin presentation*.
- The generators are called the *Artins generators*.
- An element of B_n will be called an n-braid.

Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

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

- The presentation is called the *Artin presentation*.
- The generators are called the *Artins generators*.
- An element of B_n will be called an n-braid.
- B_2 is an infinite cyclic group.

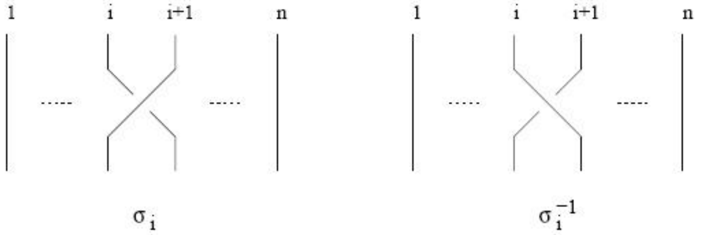
Algebraic presentation

For $n > 1$, the braid group B_n is defined by the presentation:

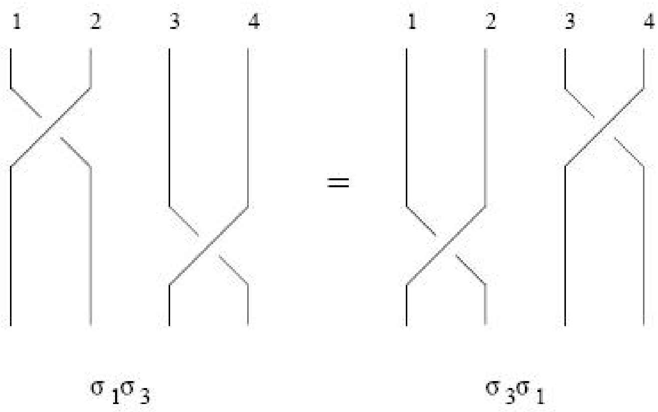
$$\left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{l} \sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| \geq 2, \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} \text{ for } |i-j| = 1 \end{array} \right\rangle$$

- The presentation is called the *Artin presentation*.
- The generators are called the *Artins generators*.
- An element of B_n will be called an n-braid.
- B_2 is an infinite cyclic group.
- For $n > 2$, the group B_n is not commutative and it contains an infinite cyclic subgroup.

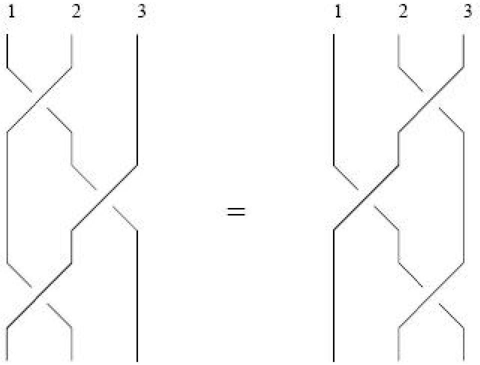
Geometric presentation



Geometric presentation



Geometric presentation



$\sigma_1 \sigma_2 \sigma_1$ $\qquad$ $\sigma_2 \sigma_1 \sigma_2$

- Burau representation

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
  2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
  2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.
  3. There are a few heuristic algorithms to calculate a preimage with empirically high success rate.

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
  2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.
  3. There are a few heuristic algorithms to calculate a preimage with empirically high success rate.
  4. It has lower order than other representations which makes it more popular in some cryptanalytic attacks against the braid group cryptosystems.

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
  2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.
  3. There are a few heuristic algorithms to calculate a preimage with empirically high success rate.
  4. It has lower order than other representations which makes it more popular in some cryptanalytic attacks against the braid group cryptosystems.
- Lawrence-Krammer representation (always faithful).

- Burau representation
    1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
    2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.
    3. There are a few heuristic algorithms to calculate a preimage with empirically high success rate.
    4. It has lower order than other representations which makes it more popular in some cryptanalytic attacks against the braid group cryptosystems.
- Lawrence-Krammer representation (always faithful).
- Gassner representation.

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
  2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.
  3. There are a few heuristic algorithms to calculate a preimage with empirically high success rate.
  4. It has lower order than other representations which makes it more popular in some cryptanalytic attacks against the braid group cryptosystems.
- Lawrence-Krammer representation (always faithful).
- Gassner representation.
- Morton representation

- Burau representation
  1. Faithful for $n \leq 3$, not faithful for $n \geq 5$ and unknown for $n = 4$.
  2. There is no deterministic algorithm which would return some preimage of the given Burau matrix.
  3. There are a few heuristic algorithms to calculate a preimage with empirically high success rate.
  4. It has lower order than other representations which makes it more popular in some cryptanalytic attacks against the braid group cryptosystems.
- Lawrence-Krammer representation (always faithful).
- Gassner representation.
- Morton representation

Burau representation

Standard Burau representation

Burau representation

Standard Burau representation

$$\sigma_i \mapsto \begin{pmatrix} I_{i-1} & 0 & 0 & 0 \\ 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & I_{n-i-1} \end{pmatrix}$$

Burau representation

Standard Burau representation

$$\sigma_i \mapsto \begin{pmatrix} I_{i-1} & 0 & 0 & 0 \\ 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & I_{n-i-1} \end{pmatrix}$$

Reduced Burau representation

$$\sigma_i \mapsto \begin{pmatrix} I_{i-2} & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & t & -t & 1 & 0 \\ 0 & 0 & 0 & 0 & I_{n-i-2} \end{pmatrix}$$

Hard problems in Braid groups

Hard problems in Braid groups

1. **Conjugacy Search Problem:** Assuming that the braid w is a conjugate of the braid u, find a witness, i.e., find a braid x satisfying $w = xux^{-1}$.

Hard problems in Braid groups

1. **Conjugacy Search Problem:** Assuming that the braid w is a conjugate of the braid u, find a witness, i.e., find a braid x satisfying $w = xux^{-1}$.
2. **Word Problem:** Given a braid w, does w represent the unit braid?

Hard problems in Braid groups

1. **Conjugacy Search Problem:** Assuming that the braid w is a conjugate of the braid u, find a witness, i.e., find a braid x satisfying $w = xux^{-1}$.
2. **Word Problem:** Given a braid w, does w represent the unit braid?
3. **Root Problem:** Assuming that the braid w is an k-th power in B_n,find a k-th root of w, i.e., find a braid u satisfying $u^k = w$.

Hard problems in Braid groups

1. **Conjugacy Search Problem:** Assuming that the braid w is a conjugate of the braid u, find a witness, i.e., find a braid x satisfying $w = xux^{-1}$.
2. **Word Problem:** Given a braid w, does w represent the unit braid?
3. **Root Problem:** Assuming that the braid w is an k-th power in B_n,find a k-th root of w, i.e., find a braid u satisfying $u^k = w$.
4. **Decomposition Problem:** Assuming that for the braids u and w there are braids a_1 and a_2 such that $w = a_1 u a_2$ find witnesses, i.e., find braids x and y satisfying $w = xuy$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \dots, s_{m-1}$ (resp. $s_{m+1}, \dots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \dots, s_{m-1}$ (resp. $s_{m+1}, \dots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.
2. *Private keys:* Alice: $a \in LB_n$; Bob: $b \in UB_n$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.
2. *Private keys:* Alice: $a \in LB_n$; Bob: $b \in UB_n$.
3. Alice sends to Bob $p_1 = apa^{-1}$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \dots, s_{m-1}$ (resp. $s_{m+1}, \dots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.
2. *Private keys:* Alice: $a \in LB_n$; Bob: $b \in UB_n$.
3. Alice sends to Bob $p_1 = apa^{-1}$.
4. Bob sends to Alice $p_2 = bpb^{-1}$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.
2. *Private keys:* Alice: $a \in LB_n$; Bob: $b \in UB_n$.
3. Alice sends to Bob $p_1 = apa^{-1}$.
4. Bob sends to Alice $p_2 = bpb^{-1}$.
5. Alice computes $K_1 = ap_2s^{-1}$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.
2. *Private keys:* Alice: $a \in LB_n$; Bob: $b \in UB_n$.
3. Alice sends to Bob $p_1 = apa^{-1}$.
4. Bob sends to Alice $p_2 = bpb^{-1}$.
5. Alice computes $K_1 = ap_2 s^{-1}$.
6. Bob computes $K_2 = bp_1 b^{-1}$.

The Ko at al. key - exchange protocol (2000)

Let LB_n (resp. UB_n) be a subgroup of B_n generated by $s_1, \ldots, s_{m-1}$ (resp. $s_{m+1}, \ldots, s_{m-1}$) with $m = \lfloor n/2 \rfloor$.
Then, every braid in LB_n commutes with every braid in UB_n.

PROTOCOL:

1. *Public key:* one braid p in B_n.
2. *Private keys:* Alice: $a \in LB_n$; Bob: $b \in UB_n$.
3. Alice sends to Bob $p_1 = apa^{-1}$.
4. Bob sends to Alice $p_2 = bpb^{-1}$.
5. Alice computes $K_1 = ap_2s^{-1}$.
6. Bob computes $K_2 = bp_1b^{-1}$.

$$\text{KEY} : K_1 = K_2$$

Ko at al. encryption scheme (2000)

Key generation algorithm:

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.
Alice's public key: p and p'.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.
Alice's public key: p and p'.
Encryption algorithm:

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.
2. Bob computes $p'' = rpr^{-1}$.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.
2. Bob computes $p'' = rpr^{-1}$.
3. Bob computes $m''_B = m_B \oplus h(rp'r^{-1})$.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.
2. Bob computes $p'' = rpr^{-1}$.
3. Bob computes $m''_B = m_B \oplus h(rp'r^{-1})$.
4. Bob sends to Alice p'' and m''_B.

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.
2. Bob computes $p'' = rpr^{-1}$.
3. Bob computes $m''_B = m_B \oplus h(rp'r^{-1})$.
4. Bob sends to Alice p'' and m''_B.

Decryption algorithm:

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.
2. Bob computes $p'' = rpr^{-1}$.
3. Bob computes $m''_B = m_B \oplus h(rp'r^{-1})$.
4. Bob sends to Alice p'' and m''_B.

Decryption algorithm:

1. Alice computes $m_A = m''_B \oplus h(sp''s^{-1})$

Ko at al. encryption scheme (2000)

Key generation algorithm:

1. Alice chooses $p \in B_n$ and $s \in LB_n$.
2. Alice computes $p' = sps^{-1}$.

Alice's private key: s.

Alice's public key: p and p'.

Encryption algorithm:

1. Bob chooses a random braid $r \in UB_n$.
2. Bob computes $p'' = rpr^{-1}$.
3. Bob computes $m''_B = m_B \oplus h(rp'r^{-1})$.
4. Bob sends to Alice p'' and m''_B.

Decryption algorithm:

1. Alice computes $m_A = m''_B \oplus h(sp''s^{-1})$