Braid Group Cryptography

Liam Hardiman

March 4, 2019

Finitely Presented Groups

A finitely presented group $G = \langle S|R \rangle$ is specified by two sets, $S = \{x_i\}_{i \in I}$ and $R = \{r_j\}_{j \in J}$.

Finitely Presented Groups

A finitely presented group $G = \langle S | R \rangle$ is specified by two sets, $S = \{x_i\}_{i \in I}$ and $R = \{r_j\}_{j \in J}$.

- S is a set of symbols called **generators**.

Finitely Presented Groups

A finitely presented group $G = \langle S | R \rangle$ is specified by two sets, $S = \{x_i\}_{i \in I}$ and $R = \{r_j\}_{j \in J}$.

- S is a set of symbols called **generators**.
- R is a set of words in S called **relators**. A **word** in S is a finite string consisting of symbols in S and the symbols x_i^{-1}, where $x_i \in S$. The empty string, e, is also a word.

Finitely Presented Groups

A finitely presented group $G = \langle S|R \rangle$ is specified by two sets, $S = \{x_i\}_{i \in I}$ and $R = \{r_j\}_{j \in J}$.

- S is a set of symbols called **generators**.
- R is a set of words in S called **relators**. A **word** in S is a finite string consisting of symbols in S and the symbols x_i^{-1}, where $x_i \in S$. The empty string, e, is also a word.
- We form a group by taking all possible words in S. The inverse of a word w is formed by writing the symbols in w in reverse order and replacing each x_j appearing in w by x_j^{-1}. The group operation is concatenation of words.

Finitely Presented Groups

We form G from S and R by taking all equivalence classes of words in S. Two words v and w are equivalent if v can be transformed into w by a finite sequence of these operations.

Finitely Presented Groups

We form G from S and R by taking all equivalence classes of words in S. Two words v and w are equivalent if v can be transformed into w by a finite sequence of these operations.

1. Replacing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ with e

Finitely Presented Groups

We form G from S and R by taking all equivalence classes of words in S. Two words v and w are equivalent if v can be transformed into w by a finite sequence of these operations.

1. Replacing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ with e
2. Inserting $x_i x_i^{-1}$ or $x_i^{-1} x_i$ at any position

Finitely Presented Groups

We form G from S and R by taking all equivalence classes of words in S. Two words v and w are equivalent if v can be transformed into w by a finite sequence of these operations.

1. Replacing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ with e
2. Inserting $x_i x_i^{-1}$ or $x_i^{-1} x_i$ at any position
3. Replacing r_j with e

Finitely Presented Groups

We form G from S and R by taking all equivalence classes of words in S. Two words v and w are equivalent if v can be transformed into w by a finite sequence of these operations.

1. Replacing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ with e
2. Inserting $x_i x_i^{-1}$ or $x_i^{-1} x_i$ at any position
3. Replacing r_j with e
4. Inserting r_j at any position

Finitely Presented Groups

We form G from S and R by taking all equivalence classes of words in S. Two words v and w are equivalent if v can be transformed into w by a finite sequence of these operations.

1. Replacing $x_i x_i^{-1}$ or $x_i^{-1} x_i$ with e
2. Inserting $x_i x_i^{-1}$ or $x_i^{-1} x_i$ at any position
3. Replacing r_j with e
4. Inserting r_j at any position

Equivalently, G is the quotient of the free group on S by the normal closure of R. We say G is **finitely presented** if S and R are finite sets.

Finitely Presented Groups

Some examples of finitely presented groups include...

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups
- The free group F_n on n generators

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups
- The free group F_n on n generators
- Finitely generated abelian groups

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups
- The free group F_n on n generators
- Finitely generated abelian groups
- The braid group B_n, $n \geq 0$.

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups
- The free group F_n on n generators
- Finitely generated abelian groups
- The braid group B_n, $n \geq 0$.

Nonexamples include

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups
- The free group F_n on n generators
- Finitely generated abelian groups
- The braid group B_n, $n \geq 0$.

Nonexamples include

- Any group with infinitely many generators, e.g. $\mathbb{Z}^{\oplus \mathbb{Z}}$

Finitely Presented Groups

Some examples of finitely presented groups include...

- Finite groups
- The free group F_n on n generators
- Finitely generated abelian groups
- The braid group B_n, $n \geq 0$.

Nonexamples include

- Any group with infinitely many generators, e.g. $\mathbb{Z}^{\oplus \mathbb{Z}}$
- There are finitely generated groups that are not finitely related, e.g. the wreath product of $\mathbb{Z}$ with itself.

The Word Problem

Say we have a finitely presented group G.

The Word Problem

Say we have a finitely presented group G.

The word problem in G

input: *two words v, w in the generators of G*

output: **yes** *if v is equivalent to w.* **no** *otherwise*

The Word Problem

Say we have a finitely presented group G.

The word problem in G

input: *two words v, w in the generators of G*
output: **yes** *if v is equivalent to w.* **no** *otherwise*

Example (The word problem in $F_2 = \langle a, b \rangle$)

Iteratively scan through both words, deleting adjacent inverses.

The Word Problem

Say we have a finitely presented group G.

The word problem in G

input: *two words v, w in the generators of G*

output: **yes** *if v is equivalent to w.* **no** *otherwise*

Example (The word problem in $F_2 = \langle a, b \rangle$)

Iteratively scan through both words, deleting adjacent inverses.
Given $v = aa^{-1}bba$ and $w = babb^{-1}a^{-1}ba$, we have

The Word Problem

Say we have a finitely presented group G.

The word problem in G

input: *two words v, w in the generators of G*

output: **yes** *if v is equivalent to w.* **no** *otherwise*

Example (The word problem in $F_2 = \langle a, b \rangle$)

Iteratively scan through both words, deleting adjacent inverses.
Given $v = aa^{-1}bba$ and $w = babb^{-1}a^{-1}ba$, we have

$$aa^{-1}bba = bba$$
$$babb^{-1}a^{-1}ba = bba.$$

The Word Problem

Say we have a finitely presented group G.

The word problem in G

input:	*two words v, w in the generators of G*
output:	**yes** *if v is equivalent to w.* **no** *otherwise*

Example (The word problem in $F_2 = \langle a, b \rangle$)

Iteratively scan through both words, deleting adjacent inverses. Given $v = aa^{-1}bba$ and $w = babb^{-1}a^{-1}ba$, we have

$$aa^{-1}bba = bba$$
$$babb^{-1}a^{-1}ba = bba.$$

Output **yes**.

The Word Problem

In 1955 Pyotr Novikov showed that there are finitely presented groups in which the word problem is **undecidable** - it is provably impossible to construct an algorithm that always outputs the correct answer.

The Conjugacy Search Problem

Let G be a group.

The Conjugacy Search Problem

Let G be a group.

The Conjugacy Search Problem in G

input: *Two conjugate words u and v in the generators of G.*

output: *A word w such that $u = w^{-1}vw = v^w$*

The Conjugacy Search Problem

Let G be a group.

The Conjugacy Search Problem in G

input: *Two conjugate words u and v in the generators of G.*

output: *A word w such that $u = w^{-1}vw = v^w$*

This is analogous to the discrete logarithm problem in a finite abelian group H.

The Conjugacy Search Problem

Let G be a group.

The Conjugacy Search Problem in G

input: *Two conjugate words u and v in the generators of G.*
output: *A word w such that $u = w^{-1}vw = v^w$*

This is analogous to the discrete logarithm problem in a finite abelian group H.

Discrete Logarithm Problem in H

input: *Elements g, h of H such that $h \in \langle g \rangle$*
output: *An integer k such that $g^k = h$*

The Braid Group

Definition

The braid group on n strands, B_n is defined by the presentation

$$B_n = \langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \mid \sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j, \; |i-j| = 1; \\ \sigma_i \sigma_j = \sigma_j \sigma_i, \; |i-j| > 1 \rangle.$$

The Braid Group

Definition

The braid group on n strands, B_n is defined by the presentation

$$B_n = \langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \mid \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j, \ |i-j| = 1; \\ \sigma_i\sigma_j = \sigma_j\sigma_i, \ |i-j| > 1\rangle.$$

There is, however, a more geometric understanding of the braid group.

The Braid Group

- Arrange two sets of n items in vertical columns on opposite sides of the page. Fasten one end of a string to each item on the left side of the page. To each item on the right side attach the other end of one string. This connection is a **braid**.

The Braid Group

- Arrange two sets of n items in vertical columns on opposite sides of the page. Fasten one end of a string to each item on the left side of the page. To each item on the right side attach the other end of one string. This connection is a **braid**.
- The generator σ_i represents connecting the i-th item on the left to the $i + 1$st on the right and the $i + 1$st on the left to the i-th on the right with the latter string passing over the former.

The Braid Group

- Arrange two sets of n items in vertical columns on opposite sides of the page. Fasten one end of a string to each item on the left side of the page. To each item on the right side attach the other end of one string. This connection is a **braid**.
- The generator σ_i represents connecting the i-th item on the left to the $i+1$st on the right and the $i+1$st on the left to the i-th on the right with the latter string passing over the former.
- Two connections that can be made to look the same by tightening the strings are considered the same braid.

The Braid Group

- Arrange two sets of n items in vertical columns on opposite sides of the page. Fasten one end of a string to each item on the left side of the page. To each item on the right side attach the other end of one string. This connection is a **braid**.
- The generator σ_i represents connecting the i-th item on the left to the $i+1$st on the right and the $i+1$st on the left to the i-th on the right with the latter string passing over the former.
- Two connections that can be made to look the same by tightening the strings are considered the same braid.
- Composing two braids consists of drawing them next to one another, gluing the points in the middle, and connecting the strands.

The Braid Group

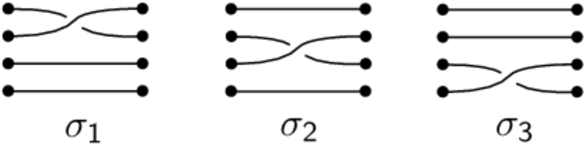
Example

In B_4 the generators σ_1, σ_2, σ_3 can be represented by these braid diagrams.

The Braid Group

Example

In B_4 the generators σ_1, σ_2, σ_3 can be represented by these braid diagrams.



The Braid Group

σ_1^{-1}

is the same as

σ_1^{-1}

The Braid Group

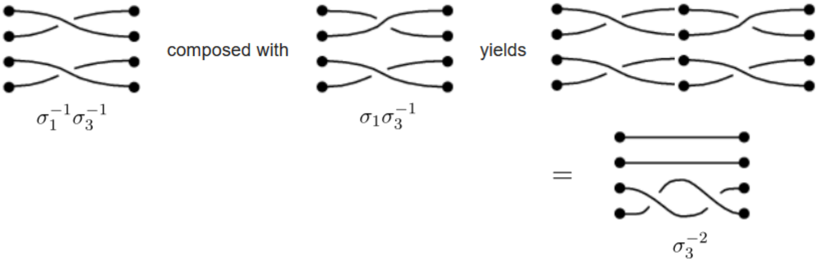
Example

Here's an example of composition of braids in B_4.

The Braid Group

Example

Here's an example of composition of braids in B_4.



Braid Group Facts

- B_1 is the trivial group. $B_2 \cong \mathbb{Z}$. B_n for $n \geq 3$ is infinite and non-commutative.

Braid Group Facts

- B_1 is the trivial group. $B_2 \cong \mathbb{Z}$. B_n for $n \geq 3$ is infinite and non-commutative.
- The $n-1$ transpositions $(i, i+1)$ in the symmetric group S_n obey the braid relations and generate S_n. Consequently, there is a surjective homomorphism $\rho : B_n \to S_n$ that sends σ_i to $(i, i+1)$.

Special Braids

Definition (Permutation Braid)

To each permutation $\tau = b_1 b_2 \cdots b_n \in S_n$, associate an n-braid A by connecting the right i-th point to the left b_i-th point with positive crossings (the strand from i to b_i passes under the one from j to b_j if $i < j$). A braid of this form is called a **permutation braid** or **canonical factor**. The set of all such braids is denoted $\tilde{\Sigma}_n$.

Special Braids

Definition (Permutation Braid)

To each permutation $\tau = b_1 b_2 \cdots b_n \in S_n$, associate an n-braid A by connecting the right i-th point to the left b_i-th point with positive crossings (the strand from i to b_i passes under the one from j to b_j if $i < j$). A braid of this form is called a **permutation braid** or **canonical factor**. The set of all such braids is denoted $\tilde{\Sigma}_n$.

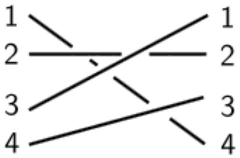


Figure: The braid $A \in \tilde{\Sigma}_4$ corresponding to $\pi = 4213 \in S_4$.

Special Braids

Definition (Fundamental Braid)

The permutation braid corresponding to the permutation $\Omega_n = n(n-1)\cdots(2)1$ is called the **fundamental braid** and is denoted by Δ_n.

Special Braids

Definition (Fundamental Braid)

The permutation braid corresponding to the permutation $\Omega_n = n(n-1)\cdots(2)1$ is called the **fundamental braid** and is denoted by Δ_n.

Figure: The fundamental braid $\Delta_4 \in B_4$ corresponding to the permutation $\Omega_4 = 4321$.

Left-Canonical Form for Braids

Theorem (Elrifai and Morton '94)

For any $W \in B_n$ there is a unique representation called the **left-canonical form** *given by*

$$W = \Delta^u A_1 A_2 \cdots A_p, \quad u \in \mathbb{Z}, A_i \in \tilde{\Sigma}_n \setminus \{e, \Delta\},$$

where $A_i A_{i+1}$ is left-weighted for $1 \leq i \leq p-1$. We call p the **canonical length** *of W.*

Left-Canonical Form for Braids

Theorem (Elrifai and Morton '94)

For any $W \in B_n$ there is a unique representation called the **left-canonical form** *given by*

$$W = \Delta^u A_1 A_2 \cdots A_p, \quad u \in \mathbb{Z}, A_i \in \tilde{\Sigma}_n \setminus \{e, \Delta\},$$

where $A_i A_{i+1}$ is left-weighted for $1 \leq i \leq p-1$. We call p the **canonical length** *of W.*

Note that the correspondence between a permutation $\pi \in S_n$ to its canonical factor $A \in B_n$ is a right inverse of the homomorphism $\rho : B_n \to S_n$, so the cardinality of $\tilde{\Sigma}_n$ is $n!$.

Left-Canonical Form for Braids

Theorem (Ko et al. 2000 using Epstein et al. '92)

1. *Let W be any word in $\sigma_1, \ldots, \sigma_n \in B_n$ with word length ℓ. Then the left-canonical form of W can be computed in time $O(\ell^2 n \log n)$.*

Left-Canonical Form for Braids

Theorem (Ko et al. 2000 using Epstein et al. '92)

1. *Let W be any word in $\sigma_1, \ldots, \sigma_n \in B_n$ with word length ℓ. Then the left-canonical form of W can be computed in time $O(\ell^2 n \log n)$.*
2. *Let $U = \Delta^u A_1 \cdots A_p$ and $V = \Delta^v B_1 \cdots B_q$ be the left-canonical forms of two n-braids. Then we can compute the left canonical form of UV in time $O(pqn \log n)$.*

Left-Canonical Form for Braids

Theorem (Ko et al. 2000 using Epstein et al. '92)

1. *Let W be any word in $\sigma_1, \ldots, \sigma_n \in B_n$ with word length ℓ. Then the left-canonical form of W can be computed in time $O(\ell^2 n \log n)$.*
2. *Let $U = \Delta^u A_1 \cdots A_p$ and $V = \Delta^v B_1 \cdots B_q$ be the left-canonical forms of two n-braids. Then we can compute the left canonical form of UV in time $O(pqn \log n)$.*
3. *If $U = \Delta^u A_1 \cdots A_p$ is the left-canonical form of $U \in B_n$, then we can compute the left-canonical form of U^{-1} in time $O(pn)$.*

Left-Canonical Form for Braids

Theorem (Ko et al. 2000 using Epstein et al. '92)

1. *Let W be any word in $\sigma_1, \ldots, \sigma_n \in B_n$ with word length ℓ. Then the left-canonical form of W can be computed in time $O(\ell^2 n \log n)$.*
2. *Let $U = \Delta^u A_1 \cdots A_p$ and $V = \Delta^v B_1 \cdots B_q$ be the left-canonical forms of two n-braids. Then we can compute the left canonical form of UV in time $O(pqn \log n)$.*
3. *If $U = \Delta^u A_1 \cdots A_p$ is the left-canonical form of $U \in B_n$, then we can compute the left-canonical form of U^{-1} in time $O(pn)$.*

These two theorems show that the word problem in B_n is efficiently solvable, that is, we can efficiently differentiate between any two given elements of B_n.

Diffie-Hellman with Braids: Anshel, Anshel, Goldfeld '99

1. Alice and Bob publicly agree on subgroups B_n, $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.

Diffie-Hellman with Braids: Anshel, Anshel, Goldfeld '99

1. Alice and Bob publicly agree on subgroups B_n, $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. Alice picks a secret word x in the generators of A, $x = x(a_1, \ldots, a_k)$. Bob picks a secret word y in the generators of B, $y = y(b_1, \ldots, b_m)$.

Diffie-Hellman with Braids: Anshel, Anshel, Goldfeld '99

1. Alice and Bob publicly agree on subgroups B_n, $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. Alice picks a secret word x in the generators of A, $x = x(a_1, \ldots, a_k)$. Bob picks a secret word y in the generators of B, $y = y(b_1, \ldots, b_m)$.
3. Alice sends $b_1^x, \ldots, b_m^x$ to Bob, Bob sends $a_1^y, \ldots, a_k^y$ to Alice.

Diffie-Hellman with Braids: Anshel, Anshel, Goldfeld '99

1. Alice and Bob publicly agree on subgroups B_n, $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. Alice picks a secret word x in the generators of A, $x = x(a_1, \ldots, a_k)$. Bob picks a secret word y in the generators of B, $y = y(b_1, \ldots, b_m)$.
3. Alice sends $b_1^x, \ldots, b_m^x$ to Bob, Bob sends $a_1^y, \ldots, a_k^y$ to Alice.
4. Alice computes $x(a_1^y, \ldots, a_k^y) = x^y = y^{-1}xy$. Bob computes $y(b_1^x, \ldots, b_m^x) = x^{-1}yx$.

Diffie-Hellman with Braids: Anshel, Anshel, Goldfeld '99

1. Alice and Bob publicly agree on subgroups B_n, $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. Alice picks a secret word x in the generators of A, $x = x(a_1, \ldots, a_k)$. Bob picks a secret word y in the generators of B, $y = y(b_1, \ldots, b_m)$.
3. Alice sends $b_1^x, \ldots, b_m^x$ to Bob, Bob sends $a_1^y, \ldots, a_k^y$ to Alice.
4. Alice computes $x(a_1^y, \ldots, a_k^y) = x^y = y^{-1}xy$. Bob computes $y(b_1^x, \ldots, b_m^x) = x^{-1}yx$.
5. Alice multiplies on the left by x^{-1}, obtaining $x^{-1}y^{-1}xy$. Bob multiplies on the left by y^{-1} and inverts, $(y^{-1}x^{-1}yx)^{-1} = x^{-1}y^{-1}xy$. Alice and Bob now share $[x, y]$.

Security of Braid Group Diffie-Hellman

An eavesdropper, Eve, wants to derive the shared secret $[x, y]$.
What does she know?

Security of Braid Group Diffie-Hellman

An eavesdropper, Eve, wants to derive the shared secret $[x, y]$. What does she know?

1. The generators of the public subgroups $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.

Security of Braid Group Diffie-Hellman

An eavesdropper, Eve, wants to derive the shared secret $[x, y]$. What does she know?

1. The generators of the public subgroups $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. The conjugates of these generators by secret elements: $a_1^y, \ldots, a_k^y, b_1^x, \ldots, b_m^x$.

Security of Braid Group Diffie-Hellman

An eavesdropper, Eve, wants to derive the shared secret $[x, y]$. What does she know?

1. The generators of the public subgroups $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. The conjugates of these generators by secret elements: $a_1^y, \ldots, a_k^y, b_1^x, \ldots, b_m^x$.

- This looks a lot like the classical Diffie-Hellman problem: find $g^{ab} \pmod{p}$ from g^a, g^b, g, and p. One way to solve the classical DHP is by solving the discrete logarithm problem in $\mathbb{Z}/p\mathbb{Z}$.

Security of Braid Group Diffie-Hellman

An eavesdropper, Eve, wants to derive the shared secret $[x, y]$. What does she know?

1. The generators of the public subgroups $A = \langle a_1, \ldots, a_k \rangle$ and $B = \langle b_1, \ldots, b_m \rangle$.
2. The conjugates of these generators by secret elements: $a_1^y, \ldots, a_k^y, b_1^x, \ldots, b_m^x$.

- This looks a lot like the classical Diffie-Hellman problem: find $g^{ab} \pmod{p}$ from g^a, g^b, g, and p. One way to solve the classical DHP is by solving the discrete logarithm problem in $\mathbb{Z}/p\mathbb{Z}$.
- How about solving the analog of the discrete log problem in B_n: the (simultaneous) conjugacy search problem?

Complications with the Conjugacy Problem

- In the discrete log case, $g^a \equiv g^b \pmod{p}$ if and only if $a \cong b \pmod{p-1}$. In the conjugacy case, if $a_i^y = a_i^{y'}$ for all i then $y = c_a y$ for some c_a in the centralizer of A:

Complications with the Conjugacy Problem

- In the discrete log case, $g^a \equiv g^b \pmod{p}$ if and only if $a \cong b \pmod{p-1}$. In the conjugacy case, if $a_i^y = a_i^{y'}$ for all i then $y = c_a y$ for some c_a in the centralizer of A:

$$\begin{aligned} y^{-1}a_i y = (y')^{-1}a_i y' \iff y'y^{-1}a_i y(y')^{-1} = a_i \\ \iff y'y^{-1} \in C_A. \end{aligned}$$

Complications with the Conjugacy Problem

- That exponents are only defined mod $p-1$ in the discrete log case doesn't matter much. On the other hand, suppose $y' = c_a y$ and $x' = c_b x$ for $c_a \in C_A$ and $c_b \in C_B$. Then

Complications with the Conjugacy Problem

- That exponents are only defined mod $p-1$ in the discrete log case doesn't matter much. On the other hand, suppose $y' = c_a y$ and $x' = c_b x$ for $c_a \in C_A$ and $c_b \in C_B$. Then

$$[x', y'] = (x')^{-1}(y')^{-1}x'y' = x^{-1}c_b^{-1}y^{-1}c_1^{-1}c_b x c_a y.$$

Complications with the Conjugacy Problem

- That exponents are only defined mod $p-1$ in the discrete log case doesn't matter much. On the other hand, suppose $y' = c_a y$ and $x' = c_b x$ for $c_a \in C_A$ and $c_b \in C_B$. Then

$$[x', y'] = (x')^{-1}(y')^{-1}x'y' = x^{-1}c_b^{-1}y^{-1}c_1^{-1}c_b x c_a y.$$

- If $x' \in A$ and $y' \in B$, then $c_b \in A$ and $c_a \in B$, so we can move the c_a's and c_b's around to obtain $[x, y] = [x', y']$.

Complications with the Conjugacy Problem

- That exponents are only defined mod $p-1$ in the discrete log case doesn't matter much. On the other hand, suppose $y' = c_a y$ and $x' = c_b x$ for $c_a \in C_A$ and $c_b \in C_B$. Then

$$[x', y'] = (x')^{-1}(y')^{-1}x'y' = x^{-1}c_b^{-1}y^{-1}c_1^{-1}c_b x c_a y.$$

- If $x' \in A$ and $y' \in B$, then $c_b \in A$ and $c_a \in B$, so we can move the c_a's and c_b's around to obtain $[x, y] = [x', y']$.
- So solving the simultaneous conjugacy problem doesn't seem to be enough for Eve to compute $[x, y]$. She needs to find conjugating elements that lie in the public subgroups A and B.

Membership Search Problem

The Membership Search Problem in G

input: *elements* $x, a_1, \ldots, a_k \in G$

Membership Search Problem

The Membership Search Problem in G

input: *elements* $x, a_1, \ldots, a_k \in G$

output: *an expression (if it exists) of* x *as a word in* $a_1, \ldots, a_k$.

Membership Search Problem

The Membership Search Problem in G

input: *elements* $x, a_1, \ldots, a_k \in G$

output: *an expression (if it exists) of* x *as a word in* $a_1, \ldots, a_k$.

Simultaneous Conjugacy Search Separation Problem (SCSSP)

input: $b_1, \ldots, b_m$ *and* $a_1^y, \ldots, a_k^y$

Membership Search Problem

The Membership Search Problem in G

input: *elements* $x, a_1, \ldots, a_k \in G$
output: *an expression (if it exists) of* x *as a word in* $a_1, \ldots, a_k$.

Simultaneous Conjugacy Search Separation Problem (SCSSP)

input: $b_1, \ldots, b_m$ *and* $a_1^y, \ldots, a_k^y$
output: y' *such that* $a_i^{y'} = a_i^y$ *for all* i *and* y' *is a word in* $b_1, \ldots, b_m$

Security of Braid Group Diffie-Hellman

- Shpilrain and Ushakov ('06) point out that the membership decision problem is unsolvable in B_n for $n \geq 6$ since such groups contain subgroups isomorphic to $F_2 \times F_2$, where the membership decision problem is unsolvable due to Mihailova ('58).

Security of Braid Group Diffie-Hellman

- Shpilrain and Ushakov ('06) point out that the membership decision problem is unsolvable in B_n for $n \geq 6$ since such groups contain subgroups isomorphic to $F_2 \times F_2$, where the membership decision problem is unsolvable due to Mihailova ('58).
- As of 2014 there is no known efficient algorithm for computing the centralizers of arbitrary subsets of braid groups (Kalka, Tsaban, Vinokur).

Security of Braid Group Diffie-Hellman

- Shpilrain and Ushakov ('06) point out that the membership decision problem is unsolvable in B_n for $n \geq 6$ since such groups contain subgroups isomorphic to $F_2 \times F_2$, where the membership decision problem is unsolvable due to Mihailova ('58).
- As of 2014 there is no known efficient algorithm for computing the centralizers of arbitrary subsets of braid groups (Kalka, Tsaban, Vinokur).
- Kotov et al. (2018) describe a heuristic attack on the SCSSP that works experimentally, but they don't appear to provide a bound on the complexity.

Parameter Sizes in Kotov et al. Attack

	B_8	B_{12}	B_{16}	B_{20}
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 20$	2	0.1	0.3	0.2
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 50$	2	1	1	1
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 100$	5	8	6	5
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 200$	42	26	34	16
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 500$	246	387	586	897
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 1000$	1330	3949	5342	9855

Figure: Time required to solve 100 random instances of the SCSSP (in minutes)

Parameter Sizes in Kotov et al. Attack

	B_8	B_{12}	B_{16}	B_{20}
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 20$	3986	6244	9573	12586
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 50$	10005	15842	21968	28576
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 100$	20258	31485	43089	56229
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 200$	41361	64291	86824	110392
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 500$	103100	159551	218255	275611
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 1000$	206835	320829	438238	556195

Figure: Average word length of an instance after "length reduction"

Parameter Sizes in Kotov et al. Attack

	B_8	B_{12}	B_{16}	B_{20}
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 20$	17708	31392	52102	72553
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 50$	44449	79648	119565	164729
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 100$	90000	158295	234520	324138
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 200$	183754	323233	472556	636368
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 500$	458041	802167	1187895	1588793
$\lvert v_i\rvert = \lvert w_i\rvert = \lvert z\rvert \approx 1000$	918904	1613018	2385195	3206254

Figure: Average size (in bits) of an instance after "length reduction"

Quantum?

In 2014, Alagic, Jeffery, and Jordan describe how braid groups can used to **obfuscate** classical and quantum computation.

Quantum?

In 2014, Alagic, Jeffery, and Jordan describe how braid groups can used to **obfuscate** classical and quantum computation.

Definition (Obfuscator)

Loosely speaking, an obfuscator takes in a circuit and outputs a "difficult to understand" but functionally identical circuit.