McMASTER UNIVERSITY

Braid Group Cryptography

by

Chris Gatopoulos

July 2014

Abstract

We give an overview of public-key cryptographic schemes based on non-commutative groups with special consideration to braid groups. We discuss some attacks on these schemes, and some thoughts about future work.

Contents

List of Figures

List of Tables

Chapter 1

The Braid Group & Cryptography

1.1 Introduction

Alice has sensitive information that she needs to tell Bob and no one else. The problem is that anything she sends Bob is out in the open. Anyone can listen and eavesdroppers are everywhere. She even cannot always be sure her message can get to Bob without a risk of it being tampered with. Sometimes someone else will try to impersonate Alice, and send Bob a counterfeit message that sender wants Bob to believe came from Alice. How can we keep our secrets safe and trust in their integrity? Sometimes we just do not want to make it easy for a casual snooper to pry into our business. Other times security is serious business, and it is crucial that we keep our sensitive information from those who should not see it.

People have been keeping secrets for a long time. The earliest secret writings have been found in 4000-year-old Egyptian hieroglyphics. These do not seem to have been a serious attempt to hide information, but rather a puzzle for readers to solve. The Romans used a substitution cipher, the Caesar cipher. In this scheme each letter of the alphabet is replaced with another. By today's standard this is almost a trivial scheme to crack. A stronger version of a substitution cipher was developed in the 16^{th} Century, the Vigenère cipher. By the 20^{th} Century machines like Enigma made ciphers that were extremely difficult for a human to break.

All of the older cryptosystems were "private key". In other words, the same secret key is used to decrypt as it is to encrypt. This poses the problem of transmitting the key securely itself. In 1976 the first "public key" protocol was developed by Diffie and Hellman. This was an important development, as now secrets can be kept hidden without ever physically sharing keys.

Today, security is more important than it has ever been. The internet has left us vulnerable in ways that did not exist before. Everything is potentially out in the open. There is banking information, business secrets, health-care records, credit-card purchases, emails, telephone calls, national security, your secrets – all of that subject to attack. Just this

year, in 2014, it is projected that eCommerce will hit 1.5 trillion dollars! There are hackers, criminals, extortionists, and others with malicious intent that want your information.

The good news is that cryptography is keeping us safe.

Or is it?

We just don't know. Unfortunately, we do not know that any public-key cryptosystems are provably secure. This is a major problem. For our own security we need to develop cryptosystems that we can prove are secure.

Why the braid groups? The braid groups are subset of non-commutative groups. Non-commutative groups in general are interesting to study as a basis for cryptology. Current cryptosystems rely on just a few computational assumptions. These may prove to be false some day. Non-commutative groups are a rich source of new problems; there are a lot of unsolved and intractible problems. It may be possible that with non-commutative groups we will be able to develop a provably secure system. It could be that these systems are more efficient so that we can get away with having smaller keys and less computation. The braid groups have some nice properties among these groups that make them a natural choice as a basis for a cryptographic system.

In this paper, we first give an overview of current and potential public-key cryptosystems. Afterwards we discuss the properties of the braid group, and then the potential of braids to be a secure platform for cryptology.

Chapter 2

Cryptography

2.1 The Basics

Alice and Bob want to exchange information without it being understood or corrupted by a third party. The channel itself is assumed to be open. That is, anyone may be eavesdropping.

An encrypted message, the ***ciphertext*** message, is sent with the intention that only those authorized to understand it can. A message in clear text, the ***plaintext*** message, is a message that is not encrypted. ***Encryption***, or ***enciphering***, is the transformation of a plaintext message into a ciphertext, while ***decryption***, or ***deciphering***, is the inverse transformation of a ciphertext back to plaintext. Classically, ***cryptography*** is the study of encryption schemes, or ***cryptosystems***, ***cryptoanalysis*** is the study of breaking cryptosystems, and ***cryptology*** encompasses both. However, in the modern literature cryptography is usually synonymous with cryptology.

A protocol to exchange a message without a third party being able to understand or deduce any information about the message, except, perhaps, its length is a ***confidential message exchange***. An unauthorized attempt, especially one more sophisticated than random brute-force, to decrypt or corrupt a ciphertext is an ***attack***. Modern cryptosystems assume that the attacker knows what cryptosystem is being used, and do not assume security through obscurity, or ignorance of how a message is encrypted. Instead, security is provided by some secret knowledge that only those who are authorized have. The knowledge required to encrypt or decrypt is a referred to as a ***key***. In a ***private key*** or ***symmetric*** cryptosystem, the same secret key is used to encrypt and decrypt. A protocol to exchange or establish secret keys securely is a ***key exchange***. In a ***public key*** cryptosystem or ***asymmetric*** cryptosystem, there is a ***public*** key that is public knowledge, and a ***private*** key that is secret. Typically, the key used to encrypt is in the open, while the key used to decrypt is kept hidden.

A problem is ***easy*** if it admits an algorithmic solution of at most ***polynomial time*** with respect to the size of its input. A ***hard*** problem is a problem that requires greater than polynomial time. Hard problems are deemed to be ***intractible***, while easy problems

are usually considered ***tractible***. Polynomial time is also called ***polytime***. ***P*** is the ***complexity class*** of easy problems. The ***NP*** class is the set of problems that can be verified in polynomial time. **P** is contained in **NP**, but it is unknown if **P** and **NP** coincide. Public key cryptosystems assume the existence of ***one-way functions***. These are functions that are easy to compute, but difficult to invert, given only the information required to compute the function. If there exists auxiliary information, called the ***key*** or ***trapdoor***, that makes it easy to calculate the inverse, then this one-way function is a ***trapdoor*** function. That one-way functions exist is conjecture and is a long-standing open problem. Because the solution to the inverse of a one-way function is easy to verify, the existence of one-way functions would also prove that $\mathbf{P} \neq \mathbf{NP}$, although the reverse is not the case.

If by having a solution to a problem B, we can solve the problem A in polytime, then solving the problem B can be ***polytime reduced*** to solving A. A cryptosystem is ***secure*** if decryption is a hard problem Otherwise it is ***insecure***. An assumption A is ***stronger*** than assumption B if solving the problem relying on B can be polytime ***reduced*** to solving the problem relying on A. (If we can solve B in polytime, we can solve A in polytime.) It is equivalent to say that assumption B is ***weaker*** than A. The definition may sound counterintuitive. If the assumption is stronger, then we make a greater assumption about how hard a problem is, so that the problem is at least as easy. Note that the implication for polytime solvability is one-way. If B is polytime, so is A. However, A may be polytime solvable, while B is not. That is, B may be a hard problem, while A is not. A cryptosystem with weaker assumptions will be at least as secure as one with stronger assumptions.

A ***private-key*** cryptosystem is symmetric in the sense that the key used to decrypt the message is the same, or is derivable, from the key used to encrypt. In a typical private-key cryptosystem Alice sends Bob a message as follows:

- Alice and Bob agree to a shared secret key, K, beforehand in some secured manner.

  This determines the encryption and decryption maps, f_K and f_K^{-1}, respectively.
- Alice wishes to send Bob the message m.

  She sends him the message $f_K(m)$.
- Bob computes $f_K^{-1} \circ f_K = m$.

  Bob has recovered the message m.

A ***public-key*** cryptosystem is asymmetric in that from the knowledge of the key used to either encrypt or decrypt the message, it is infeasible to find the other key. In a typical public-key cryptosystem without authentication (see below) Alice sends Bob a message as follows:

- Bob chooses an encryption map f_B with its inverse, the decryption map f_B^{-1}.

  He makes f_B public, while he keeps f_B^{-1} secret.

- Alice wishes to send Bob the message m.

  She sends him the message $f_B(m)$.

- Bob computes $f_B^{-1} \circ f_B = m$.

  Bob has recovered the message m.

In ***authentication*** Alice (the prover) wishes to verify to Bob (the verifier) that she knows some private key without a third party being able to understand or deduce any information about the key, except, perhaps, its length.

- Bob chooses an encryption map f_B with its inverse, the decryption map f_B^{-1}.

  He makes f_B public, while he keeps f_B^{-1} secret.

- Alice chooses an encryption map f_A with its inverse, the decryption map f_A^{-1}.

  She makes f_A public, while she keeps f_A^{-1} secret. f_A^{-1} is her private key.

- Alice wishes to send Bob the message m.

  She sends him the message $f_B \circ f_A^{-1}(m)$.

- Bob notes Alice's public encryption map f_A and his own private map f_B^{-1}.

  Bob computes $f_A \circ f_B^{-1} \circ \big(f_B \circ f_A^{-1}(m)\big) = m$.

  Bob has recovered the message m.

Alice sends Bob a message along with a ***signature***, with the intention that this signature proves that only Alice herself could have originated the message. The message itself may be encrypted or sent as clear text.

- Alice chooses an encryption map f_A with its inverse, the decryption map f_A^{-1}.

  She makes f_A public, while she keeps f_A^{-1} secret.

- Alice wishes to send Bob the message $f(m)$ with her signature s. $f(m)$ may be an encryption that Bob can decrypt, or just the identity that leaves a plaintext.

  She sends him the message $f(m)$ with $f \circ f_A^{-1}(s)$ appended.

- Bob notes Alice's public encryption map f_A.

  Bob recovers the message m and finds $f_A^{-1}(s)$ appended.

  Bob computes $f_A \circ f_A^{-1}(s) = s$.

  Bob has recovered the signature s appended to the message m.

A ***hash function*** takes in a, possibly, arbitrarily large input and returns a, typically, much smaller output, usually of fixed length. Let $f : m \mapsto h$ be a map from the message space to a hash value. Then, f is a hash function if it is infeasible to find both an m' such that $f(m') = f(m)$, and, knowing m, another $m'' \neq m$ such that $f(m'') = f(m)$. Among its uses, a hash function can be used to verify some secret knowledge without revealing the secret, or prove the integrity of a sent message.

2.2 Some Number Theory

Classical cryptosystems rely on some key number-theoretic results.

Definition 2.1. $a \in \mathbf{Z}$ is a ***divisor***, or ***factor***, of an integer n if $\exists b \in \mathbf{Z}$ such that $ab = n$. We write $a|n$ and say a divides n, n is a multiple of a, or, sometimes, a factors n.

Definition 2.2. $\mathbf{Z}^+$ denotes the set of integers greater than 0. An integer $c \in \mathbf{Z}^+$ is a ***common divisor*** of a and b if $c|a$ and $c|b$. An integer $d \in \mathbf{Z}^+$ is the ***greatest common divisor*** of a and b if d is a common divisor of a and b, and for any other common divisor, c, of a and b, then $c|d$. We write $\gcd(a, b) = d$. Two positive integers a, b are ***coprime*** if $\gcd(a, b) = 1$. It is also common to use ***factor*** in place of "divisor" in all of the above.

Definition 2.3. A ***totative*** of a given positive integer n is an integer m such that $0 < m \leq n$ and m and n are coprime.

Definition 2.4. $\phi(n)$, the ***Euler totient function*** or the ***phi function*** of a positive integer n counts the number of totatives of n. This is also called the ***Euler function*** or the ***Euler number***.

For p prime, $\phi(p) = p - 1$. If a and b are coprime, then $\phi(a) = \phi(a)\,\phi(b)$. For a positive power, k, of a prime p, $\phi\left(p^k\right) = p^k - p^{k-1} = p^{k-1}\,(p-1) = p^k\,(1 - 1/p)$. In general then, if

$$n = p_1^{k_1} p_2^{k_2} \cdots p_\ell^{k_\ell}$$

is a decomposition of n into powers of primes, then

$$\begin{aligned}
\phi(n) &= \phi\left(p_1^{k_1}\right)\phi\left(p_2^{k_2}\right)\cdots\phi\left(p_\ell^{k_\ell}\right) \\
&= p_1^{k_1}\left(1 - \frac{1}{p_1}\right) p_2^{k_2}\left(1 - \frac{1}{p_2}\right)\cdots p_\ell^{k_\ell}\left(1 - \frac{1}{p_\ell}\right) \\
&= p_1^{k_1} p_2^{k_2}\cdots p_\ell^{k_\ell}\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_\ell}\right) \\
&= n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_\ell}\right)
\end{aligned}$$

Definition 2.5. Let $a, b, m \in \mathbf{Z}$. Then a is ***congruent*** to b ***modulo*** m if $m|\,(a - b)$. We write $a \equiv b \mod m$.

Theorem 2.6 (Lagrange's Theorem)**.** *Let G be a finite group with order n. Then the order of any subgroup of G divides n.*

Proof. Let H be a subgroup of G. Then the (say) left cosets of H in G form a set of equivalence classes on G. In other words, two elements of G, a, b are in the same equivalence class if $a = bh$ for some $h \in H$. Let cH and dH be two left cosets of H.

TABLE 2.1: Number-Theoretic Problems

	Number-Theoretic Problems
RSA Problem	Let $m, e, n \in \mathbf{Z}$, with $m < n$ and $e < n$. $e, n, (m^e \mod n)$ is public. Find m.
Integer Factorization Problem	Given $n \in \mathbf{Z}$ factor n into primes.
Discrete Logarithm Problem (DLP)	Let GF(q) be a given, known, finite field of order q, with p a primitive element of that field. Given $p, r \in \mathrm{GL}(q)$, find $c \in \mathbf{Z}$ such that $p^c = r$.
Diffie-Hellman Problem	Let GF(q) be a given, known, finite field of order q, with p a primitive element of that field. Let $A, B \in \mathbf{Z}$. Given $p, p^A, p^B \in \mathrm{GL}(q)$, find p^{AB}.

Then there is a bijective map $f : cH \longrightarrow dH$ defined by $f : a \mapsto dc^{-1}a$ with inverse $f^{-1} : b \mapsto cd^{-1}b$. Thus, cH and dH, and therefore every coset of H, have the same number of elements. The cosets are disjoint. Therefore the order of G is the product of the order of H with the number of cosets of H. □

Theorem 2.7 (Euler's Theorem)**.** *Let $a, n \in \boldsymbol{Z}^+$ be coprime. Then,*

$$a^{\phi(n)} \equiv 1 \mod n$$

Proof. The residue classes of modulo n that are coprime to n form a group of order $\phi(n)$ under multiplication. Let a be any element of this group and let k be its order. Then the elements $a, a^2, \cdots, a^{k-1}, a^k = 1$ form a cyclic subgroup of order k. By Lagrange's theorem, the order of any subgroup divides $\phi(n)$, and we can write $\phi(n) = k\ell$. Then,

$$a^{\phi(n)} = a^{k\ell} = \left(a^k\right)^\ell \equiv 1^\ell = 1 \mod n$$

□

2.3 Some Group Theory

Definition 2.8. If G is a group, with $p, q \in G$, then the ***conjugate*** of p by the ***conjugator*** q is given by

$$p^q \equiv qpq^{-1}$$

The notation p^q is deliberately suggestive of exponentiation as

$$p^{qr} = (p^q)^r$$

holds for $p, q, r \in G$. If $n \notin G,\ n \in \mathbf{Z}$, then the notation p^n denotes the usual group-theoretic exponentiation and not conjugation. It is immediately clear that

$$(p^n)^q = (p^q)^n$$

It is then consistent if we use the notation

$$p^{qn} = p^{nq} \equiv (p^n)^q = (p^q)^n$$

Conjugacy in a group is an equivalence relation. p and p^q are in the same ***conjugacy class*** and are conjugates of each other. In particular, $(p^q)^{q^{-1}} = p$. Conjugacy is, of course, trivial in an abelian group.

TABLE 2.2: Group-Theoretic Problems

	Group-Theoretic Problems
Conjugacy Decision Problem (CDP)	Also called the ***Conjugacy Problem***, the ***Conjugacy Decision Problem*** asks whether two given elements in a nonabelian group are conjugate. That is, given G, a nonabelian group, with $p, p' \in G$, does there exist $q \in G$ such that $$p' = p^q \quad ?$$
Conjugator Search Problem (CSP)	Given G, a nonabelian group, and that $p, p' \in G$ are conjugate, find the conjugating element. That is, find $q \in G$ such that $$p' = p^q$$
Multiple Simultaneous Conjugator Search Problem(MSCSP)	Given multiple conjugate pairs in a nonabelian group G with the same conjugator, $$(p_1, p_1'), (p_2, p_2'), \ldots, (p_m, p_m'),$$ find the conjugating element. Typically, this will be an easier problem than the Conjugator Search Problem.
Diffie-Hellman type Generalized Conjugacy	Given G, a nonabelian group, $p \in G$, and subgroups $G_a, G_b \subset G$, such that elements of G_a commute with elements of G_b. Given $p \in G$ and conjugates p^r, p^s with both $r \in G_a$ and $s \in G_b$ unknown, find p^{rs}. Note that since r and s commute p^{sr} will be equal to p^{rs}.
Decomposition Search Problem (DSP)	Given G, a nonabelian group, and $p, p' \in G$, find elements r, s in some subset $H \subset G$ such that $rps = p'$, given that at least one such pair r, s exists. If H is a subgroup, then this is also called the ***double coset problem***. The decomposition search problem reduces to the conjugator search problem when $s = r^{-1}$.
Factorization Search Problem	Given G, a nonabelian group, $p \in G$, and subgroups $G_a, G_b \subset G$, such that elements of G_a commute with elements of G_b, find elements $r \in G_a$ and $s \in G_b$ such that $rs = p$, given that at least one such pair r, s exists.

2.4 Cryptosystems

The following cryptosystems are public key, and assume the existence of one-way functions for their security. Because a trapdoor problem is polytime reducible to the cryptosystem problem, it is a necessary, though not sufficient condition, that the trapdoor problems are hard ones.

The ***RSA*** scheme was introduced by Rivest, Shamir, and Adleman in Rivest et al. [24]. It is illustrated in Table 2.3.

TABLE 2.3: The RSA Scheme

	RSA Scheme
Problem	Let $m, e, n \in \mathbf{Z}$, with $m < n$ and $e < n$. $e, n, (m^e \mod n)$ is public. Find m.
Trapdoor	$d \in \mathbf{Z}$ such that $m^{ed} \mod n \equiv m$. This condition on d is equivalent to $d \cdot e \equiv 1 \mod \phi(n)$.
Trapdoor Creation	Choose two large primes p, q. Let $n = pq$ and calculate $\phi(n) = \phi(pq) = (p-1)(q-1)$. Choose e such that $\gcd(e, \phi) = 1$. Find d such that $d \cdot e \equiv 1 \mod \phi$. This can be done through the Euclidean algorithm.
Trapdoor Problem	The Integer Factorization Problem: Given $n \in \mathbf{Z}$ factor n into primes.
Cryptosystem Protocol	Alice chooses two large prime numbers, p and q. She calculates $n = pq$. She calculates the Euler number of n, $\phi = \phi(pq) = (p-1)(q-1)$. Alice chooses a random number e between 3 and ϕ that is relatively prime to ϕ. She finds a number d that is the inverse of e modulo ϕ. In other words, $d \cdot e \equiv 1 \mod \phi$. She makes e and n public. Bob wishes to send the message m_B to Alice. The cryptosystem is set up so that m_B is an integer less than p or q so that $\gcd(m_B, n) = 1$. He computes the ciphertext $c \equiv (m_B)^e \mod n$. Alice computes $m_A \equiv c^d \mod n$. $m_A = m_B$, the original message.

To show that Bob retrieves Alice's original message, note that by Theorem 2.7, Euler's theorem, $(m_B)^{\phi} \equiv 1 \mod n$. Then, we have

$$\begin{aligned} m_A &\equiv c^d \mod n \\ &= \left((m_B)^e\right)^d \mod n \\ &= (m_B)^{e \cdot d} \mod n \\ &= (m_B)^{(1+k\phi)} \mod n, \quad k \in \mathbf{Z} \\ &= m_B \cdot \left((m_B)^{\phi}\right)^k \mod n \\ &\equiv m_B \cdot 1^k \mod n \\ &= m_B \mod n \\ &= m_B \end{aligned}$$

The RSA problem is no harder than the integer factorization problem. Assuming that they are equally hard, the RSA protocol relies on the assumption of the trapdoor that given a number that is the product of two large primes, it is infeasible to factor that number.

The ***ElGamal*** scheme is illustrated in Table 2.4. For the protocol to be secure it is required that the trapdoor problem, the discrete logarithm problem, is hard.

TABLE 2.4: The ElGamal Scheme

	ElGamal Scheme
Problem	Let GF(q) be a finite field, with $p \in \text{GF}(q)$ a primitive element. Let $A, B, m_B \in \mathbf{Z}$. $p, \text{GF}(q), p^A, p^B, \left(m_B \cdot p^{AB}\right)$ is public. Find m_B.
Trapdoor	$A \in \mathbf{Z}$.
Trapdoor Creation	A is randomly chosen.
Trapdoor Problem	Given $p, p^A \in \text{GF}(q)$ find A.
Cryptosystem Protocol	Alice chooses public keys q and p, such that GF(q) is a finite field, and p is a primitive element in that field. Alice chooses a secret key $A \in \mathbf{Z}$. She sends p^A to Bob. Bob wishes to send the message m_B to Alice. Bob chooses a secret key $B \in \mathbf{Z}$ and computes $m_B \cdot p^{AB} = m_B \cdot \left(p^A\right)^B$. He sends $\left(m_B \cdot p^{AB}\right)$ and p^B to Alice. Alice computes $\left(p^{AB}\right)^{-1} = \left(p^{BA}\right)^{-1} = \left(\left(p^B\right)^A\right)^{-1}$. She recovers $m_B = \left(m_B \cdot p^{AB}\right) \cdot \left(p^{AB}\right)^{-1}$.

The ***Diffie-Hellman Key Exchange*** is shown in table 2.5. The trapdoor problem is the discrete logarithm problem, while the underlying problem for the cryptosystem is the ***Diffie-Hellman Problem***. It is common to use cyclic fields, so that q is prime. Then, all the calculations would be carried out ($\mod q$) and, for example, Alice would send Bob $\left(p^A \mod q\right)$. The Diffie-Hellman protocol relies on the assumption that the Diffie-Hellman Problem is a hard one, so that if one is given only p^A and p^B, then it is infeasible to compute p^{AB}.

TABLE 2.5: The Diffie-Hellman Key Exchange

	Diffie-Hellman Key Exchange
Problem	Let GF(q) be a finite field, with $p \in$ GF(q) a primitive element. Let $A, B \in \mathbf{Z}$. $p, \mathrm{GF}(q), p^A, p^B$ is public. Find p^{AB}.
Trapdoor	$A \in \mathbf{Z}$
Trapdoor Creation	A is randomly chosen.
Trapdoor Problem	Given $p, p^A \in$ GF(q) find A.
Cryptosystem Protocol	Alice chooses public keys q and p, such that GF(q) is a finite field, and p is a primitive element in that field. Alice chooses a secret key $A \in \mathbf{Z}$. She sends p^A to Bob. Bob chooses a secret key $B \in \mathbf{Z}$. He sends p^B to Alice. Alice computes $k_A = \left(p^B\right)^A = p^{BA} = p^{AB}$. Bob computes $k_B = \left(p^A\right)^B = p^{AB}$. $k_A = k_B$. This is a common key.

The ***Ko-Lee Key Exchange*** is a porting of the Diffie-Hillman Key Exchange to non-abelian groups (although with the braid groups in mind) and introduced in Ko et al. [16]. It is described in Table 2.6. The underlying problem is the group-theoretic analog to the Diffie-Hillman, while the trapdoor problem is the conjugacy search problem.

TABLE 2.6: The Ko-Lee Key Exchange

	Ko-Lee Key Exchange
Problem	Let G be a nonabelian group, with subgroups G_a, G_b such that elements of G_a commute with G_b. Let $p \in G,\ A \in G_a,\ B \in G_b$. p, G, G_a, G_b, p^A, p^B is public. Find p^{AB}.
Trapdoor	$A \in G_a$.
Trapdoor Creation	A is randomly chosen from G_a.
Trapdoor Problem	Given $p, p^A \in G$ find $A \in G_a$.
Cryptosystem Protocol	G is a non-abelian group with subgroups G_a, G_b such that elements of G_a commute with G_b. Alice chooses public key $p \in G$. Alice chooses a secret key $A \in G_a$. She sends p^A to Bob. Bob chooses a secret key $B \in G_b$. He sends p^B to Alice. Alice computes $k_A = \left(p^B\right)^A = p^{BA} = p^{AB}$. Bob computes $k_B = \left(p^A\right)^B = p^{AB}$. $k_A = k_B$. This is a common key.

The Decomposition Search Key Exchange is illustrated in 2.7. The trapdoor relies on the hardness of the Decomposition Search Problem for its security.

TABLE 2.7: The Decomposition Search Key Exchange

	Decomposition Search Key Exchange
Problem	Let G be a nonabelian group, with subgroups G_a, G_b such that elements of G_a commute with G_b. Let $p \in G,\ A_1, A_2 \in G_a,\ B_1, B_2 \in G_b$. $p, G, G_a, G_b, A_1pA_2, B_1pB_2$ is public. Find $A_1B_1pB_2A_2$.
Trapdoor	$A_1, A_2 \in G_a$.
Trapdoor Creation	A_1, A_2 are randomly chosen from G_a.
Trapdoor Problem	Given $p, A_1pA_2 \in G$ find $A_1, A_2 \in G_a$.
Cryptosystem Protocol	G, G_a, and G_b are public. G is a non-abelian group with subgroups G_a, G_b such that elements of G_a commute with G_b. Alice chooses public key $p \in G$. Alice chooses secret keys $A_1, A_2 \in G_a$. She sends A_1pA_2 to Bob. Bob chooses a secret key $B_1, B_2 \in G_b$. He sends B_1pB_2 to Alice. Alice computes $k_A = A_1B_1pB_2A_2$. Bob computes $k_B = B_1A_1pA_2B_2 = A_1B_1pB_2A_2$. $k_A = k_B$. This is a common key.

The ***Shpilrain-Ushakov Decomposition Search Key Exchange*** is shown in Table 2.8. This variation of the previous protocol is found in Shpilrain and Ushakov [26].

TABLE 2.8: The Shpilrain-Ushakov Decomposition Search Key Exchange

	Shpilrain-Ushakov Decomposition Search Key Exchange
Problem	Let G be a nonabelian group, with subgroups G_a, G_b such that elements of G_a commute with G_b. Let $p \in G,\ A_1, A_2 \in G_a,\ B_1, B_2 \in G_b$. $p, G, G_a, G_b, A_1pB_1, B_2pA_2$ is public. Find $A_1B_2pA_2B_1$.
Trapdoor	$A_1 \in G_a,\ B_1 \in G_b$.
Trapdoor Creation	A_1, B_1 are randomly chosen from $G_a,\ G_b$ respectively.
Trapdoor Problem	Given $p, A_1pB_1 \in G$ find $A_1 \in G_a$ and $B_1 \in G_b$.
Cryptosystem Protocol	G is a non-abelian group with public subgroups G_a, G_b such that elements of G_a commute with G_b. Alice chooses public key $p \in G$. Alice chooses secret keys $A_1 \in G_a$ and $B_1 \in G_b$. She sends A_1pB_1 to Bob. Bob chooses a secret key $B_2 \in G_b$ and $A_2 \in G_a$. He sends B_2pA_2 to Alice. Alice computes $k_A = A_1 (B_2pA_2) B_1$. Bob computes $k_B = B_2 (A_1pB_1) A_2 = A_1B_2pA_2B_1$. $k_A = k_B$. This is a common key.

The ***Stickel Key Exchange*** is illustrated in Table 2.9. This protocol does not require us to find a pair of subgroups that commute with members of the other subgroup.

TABLE 2.9: The Stickel Key Exchange

	Stickel Key Exchange
Problem	Let G be a nonabelian semigroup, with $a, b, g \in G$ and n_a, n_b the order of a, b respectively in G. Let $\ell, m, r, s \in \mathbf{Z}$ with $0 < \ell, r < n_a$ and $0 < m, s < n_b$. $a, b, g, G, a^\ell g b^m, a^r g b^s$ is public. Find $a^{\ell+r} g b^{m+s}$.
Trapdoor	$\ell, m \in \mathbf{Z}$.
Trapdoor Creation	ℓ, m are randomly chosen subject to the conditions $0 < \ell < n_a$ and $0 < m < n_b$.
Trapdoor Problem	Given semigroup elements $a, b, g, a^\ell g b^m$, find $\ell, m \in \mathbf{Z}$.
Cryptosystem Protocol	Alice chooses public keys a, with order n_a, and b, with order n_b, in some finite non-abelian semigroup G such that $ab \neq ba$. Although G may be a group, this protocol does not require G to have the identity element or for elements to have inverses. As a variation of the protocol, she also chooses some element $g \in G$. She makes a, b, G public. g is made public if it is used; otherwise it is ignored below. Alice chooses $\ell \in \mathbf{Z}$ with $0 < \ell < n_a$, and $m \in \mathbf{Z}$ with $0 < m < n_b$. She sends $u = a^\ell g b^m$ to Bob. Bob chooses $r \in \mathbf{Z}$ with $0 < r < n_a$, and $s \in \mathbf{Z}$ with $0 < s < n_b$. He sends $v = a^r g b^s$ to Alice. Alice computes $k_A = a^\ell v b^m = a^\ell a^r g b^s b^m = a^{\ell+r} g b^{m+s}$. Bob computes $k_B = a^r v b^s = a^r a^\ell g b^m b^s = a^{\ell+r} g b^{m+s}$. $k_A = k_B$.

The following protocol, the ***Factorization Search Problem Scheme***, is found in Myasnikov et al. [21]. It is illustrated in Table 2.9.

TABLE 2.10: The Factorization Search Problem Scheme

	Factorization Search Problem Scheme
Problem	Let G be a nonabelian group, with subgroups G_a, G_b such that elements of G_a commute with G_b. Let $A_1, A_2 \in G_a,\ B_1, B_2 \in G_b$. $G, G_a, G_b, A_1B_1, A_2B_2$ is public. Find $A_2A_1B_1B_2$.
Trapdoor	$A_1 \in G_a$ and $B_1 \in G_b$.
Trapdoor Creation	A_1 is randomly chosen from G_a and B_1 is randomly chosen from G_b.
Trapdoor Problem	Given group element A_1B_1, find element A_1 in subgroup G_a and element B_1 in subgroup G_b.
Cryptosystem Protocol	G is a non-abelian group with public subgroups G_a, G_b such that elements of G_a commute with G_b. Alice chooses secret keys $A_1 \in G_a$ and $B_1 \in G_b$. She sends A_1B_1 to Bob. Bob chooses a secret key $B_2 \in G_b$ and $A_2 \in G_a$. He sends B_2A_2 to Alice. Alice computes $k_A = B_1 (B_2A_2) A_1 = A_2A_1B_1B_2$. Bob computes $k_B = A_2 (A_1B_1) B_2$. $k_A = k_B$. This is a common key.

Definition 2.9. The ***commutator*** of group elements x, y, is the product

$$[x, y] \equiv xyx^{-1}y^{-1} = y^x y^{-1} = xx^{-y}$$

The ***Anshel-Anshel-Goldfeld protocol*** was introduced in Anshel et al. [1]. It is described in Table 2.11. This protocol does not require us to find a pair of subgroups that commute with members of the other subgroup.

TABLE 2.11: The Anshel-Anshel-Goldfeld Key Exchange

	Anshel-Anshel-Goldfeld Key Exchange
Problem	Let G be a nonabelian group. Let $p_1, \cdots, p_r \in G$ and $q_1, \cdots, q_s \in G$ be two sets of letters on G. $A = \alpha(p_1, \cdots, p_r)$ and $B = \beta(q_1, \cdots, q_s)$ are words on G. $G,\ p_1, \cdots, p_r,\ q_1, \cdots, q_s,\ p_1^B, \cdots, p_r^B,\ q_1^A, \cdots, q_s^A$ is public. Find $[A, B]$, the commutator of A and B.
Trapdoor	$A = \alpha(p_1, \cdots, p_r)$.
Trapdoor Creation	A is a randomly chosen word on $p_1, \cdots, p_r \in G$.
Trapdoor Problem	Given sets of group elements $q_1, \cdots, q_s$ and $q_1^A, \cdots, q_s^A$, find element $A \in G$.
Cryptosystem Protocol	G is a nonabelian group. The public key is composed of two sets of keys in G: $p_1, \cdots, p_r$ and $q_1, \cdots, q_s$. There is no commutation requirement of any kind. The alphabet $p_1, \cdots, p_r, q_1, \cdots, q_s$ is made public. Alice constructs the word $A = \alpha(p_1, \cdots, p_r)$, and keeps it private. Alice computes $q_1^A, \cdots, q_s^A$. Alice sends Bob $q_1^A, \cdots, q_s^A$. Bob constructs the word $B = \beta(q_1, \cdots, q_s)$. Bob computes $p_1^B, \cdots, p_n^B$. Bob sends Alice $p_1^B, \cdots, p_r^B$. Alice computes $t_A = A\,\alpha\left(p_1^B, \cdots, p_r^B\right)^{-1}$. Bob computes $t_B = \beta\left(q_1^A, \cdots, q_s^A\right)\ B^{-1}$. $t_A = t_B$. This is the common key $[A, B]$, the commutator of A and B.

Alice and Bob both compute the common key $[A, B]$. We have,

$$\begin{aligned} t_A &= A\,\alpha\left(p_1^B, \cdots, p_r^B\right)^{-1} \\ &= A\left((\alpha(p_1, \cdots, p_r))^B\right)^{-1} \\ &= A\left(A^B\right)^{-1} \\ &= AA^{-B} \\ &= [A, B] \\ &= B^A B^{-1} \\ &= \left(\beta(q_1, \cdots, q_s)\right)^A B^{-1} \\ &= \beta\left(q_1^A, \cdots, q_s^A\right)\ B^{-1} \\ &= t_B \end{aligned}$$

2.5 Platform Requirements

Shpilrain [25] gives the following requirements for platforms:

- (P0) The group must be well known and well understood. In particular, the conjugacy search problem must be well studied, or reduced to a well-known problem.

  This, of course, is a practical requirement that measures our current understanding of a potential platform and not necessarily its intrinsic suitability and usefulness at a later time. This requirement reduces the number of candidates substantially.

- (P1) The word problem must be solvable, with a deterministic algorithm at most quadratic time.

  This is required by legitimate users to extract keys.

- (P2) The conjugacy search problem should not have a solution with a subexponential-time deterministic algorithm.

  This is required for the security of the platform. In practice, that the platform satisfies this requirement will be conjecture. It may be very difficult or impossible to prove that no such algorithm exists.

- (P3) It must not be possible to find x by inspection given a^x.

Another nice property for a group to have is the commutation property:

- Given a group G, there are easily found subgroups $G_a, G_b \in G$ such that $\forall A \in G_a, \forall B \in G_b$ then $AB = BA$.

Chapter 3

Braids

3.1 Introduction

The braid groups present an attractive choice for a cryptographic platform. These groups are finitely generated but have infinite order, so that keys of arbitrary length can be constructed from a limited set of primitive elements. The word problem on the braid group of n-strands and word length ℓ is solvable in polynomial time with a known algorithm having run time $O(\ell^2 n)$. On the other hand, the algorithm with the best known running time to solve the conjugacy problem in general is at least exponential [1].

A visual realization of the braid groups is helpful for intuition. The identity braid on n strands can be pictured as n parallel line segments equally spaced apart, with starting and ending points fixed and aligned along axes perpendicular to the line segments. In an arbitrary braid a strand is allowed to cross over or under a strand adjacent to it, so that the strands exchange their order. We label a strand by its current order among the n strands. We will consider braids as group elements. Composing braids by concatenation is the right-acting group action. That is, for braids r, s, the composite rs consists of the braid r followed by braid s, joined together in that order to form a new braid. Two braids are considered equivalent if they are ambient isotopic in $\mathbf{R}^3$ while fixing the start and end points. That is, the strands are not allowed to be cut, intersect each other, or pass through each other, but they can be displaced and stretched or shrunk otherwise.

We will draw braids from left to right, so that the diagrams will follow the way they are composed as group actions. The strands will be labeled in order from the bottom of the diagram to the top.

FIGURE 3.1: The identity braid on 6 strands. The orientation is from left to right.

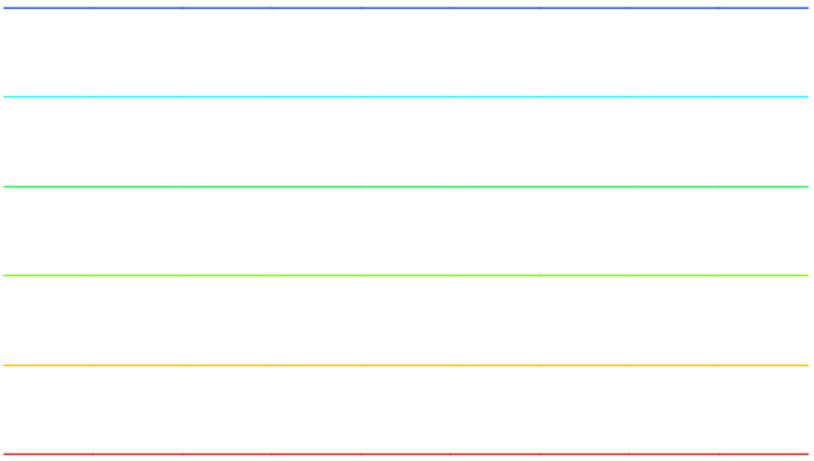


FIGURE 3.2: An arbitrary braid on 6 strands.

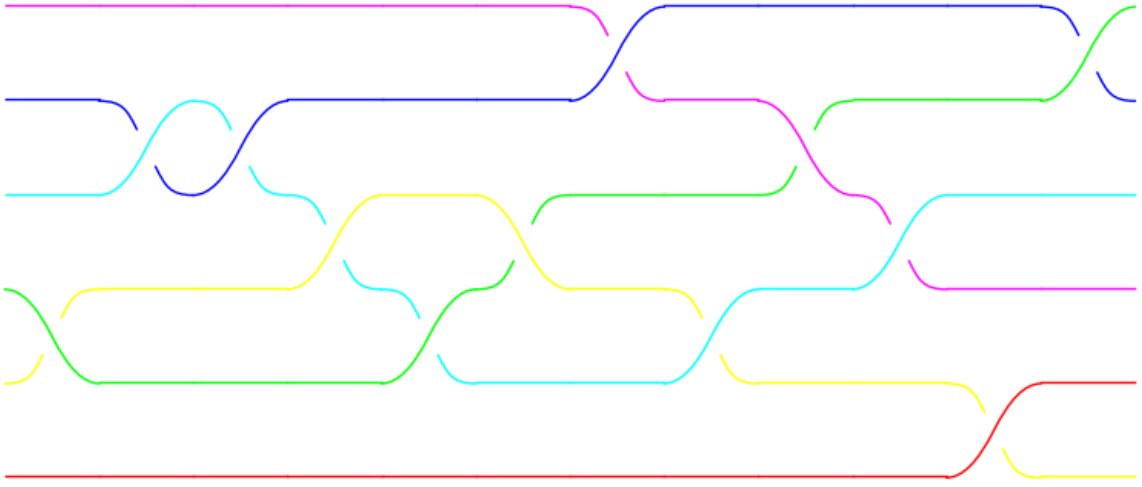


FIGURE 3.3: r on the left and s on the right are arbitrary braids on 6 strands.

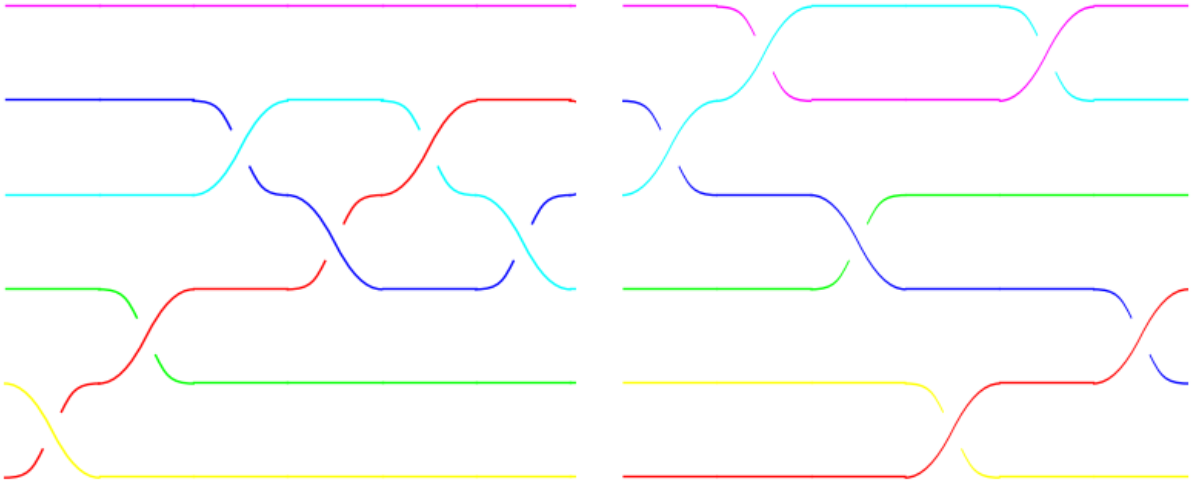
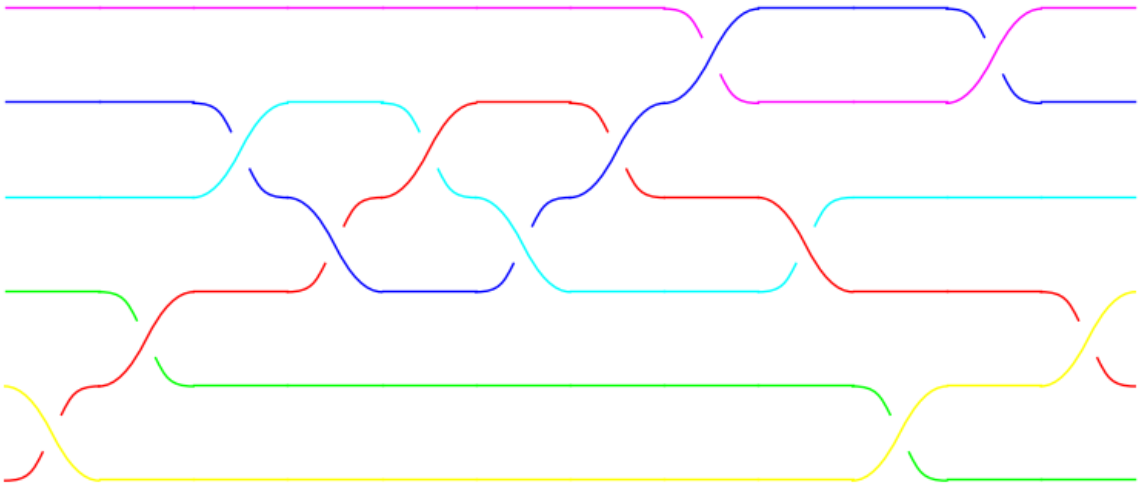


FIGURE 3.4: rs, the product of r and s, is the braid formed by concatenating r on the left with s on the right.



3.2 Presentations

Definition 3.1. The ***Artin presentation*** of the n^{th} braid group is given by

$$\mathrm{B}_n \equiv \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| \geq 2 \\ \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1} & \text{if } |i-j| = 1 \end{array} \right\rangle$$

σ_i is a braid of one crossing in which band i crosses band $i+1$. σ_i^{-1} is the corresponding inverse in which band i crosses band $i+1$ with the opposite orientation. The generators σ_i were introduced in Artin [3]. While in this paper σ_i signifies that strand i passes over strand $i+1$, the other convention in which strand $i+1$ passes over strand i is also used elsewhere.

Definition 3.2. The ***identity element*** of B_n, is the element corresponding to the ***identity braid*** in which no strand crosses any other strand. We denote the identity by e_n or normally just e. This is also called the ***trivial*** element or braid.

This presentation is an example of a more general concept of Artin and Coxeter groups, in which the relations are of the form

$$\langle a_i a_j \rangle^{m_{ij}} = \langle a_j a_i \rangle^{m_{ji}}$$

for generators $\{a_i\}_{i \in I}$. The Coxeter group associated to an Artin group has the added relation $a_i^2 = 1$. S_n, the n^{th} symmetric group, is generated by transpositions on n objects. With the added relation $a_i^2 = 1$ of the Coxeter group we recover the symmetric group by identifying a_i with the transposition exchanging i with $i+1$. In other words, there is a homomorphism from the braid group B_n to the symmetric group S_n, with the set of σ_i^2s as the kernel. Intuitively this means that the homomorphism from B_n to S_n only cares about the starting and ending points of the strands, and forgets about everything in between.

FIGURE 3.5: The braid $\sigma_2\sigma_4^{-1}\sigma_5\sigma_5^{-1}$: σ_2 followed by σ_4^{-1} followed by σ_5 followed by σ_5^{-1}. The orientation is from left to right and the strands are numbered from bottom to top. σ_2 is the generator that passes the third strand over the second, while σ_4^{-1} passes the fifth strand under the fourth. σ_5 passes the sixth strand over the fifth and σ_5^{-1} passes the sixth strand under the fifth. A strand is labeled according to its position at the time of application of the group element.

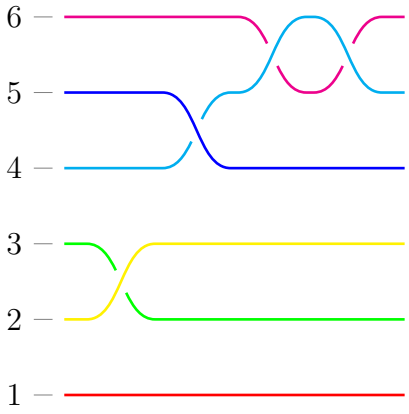
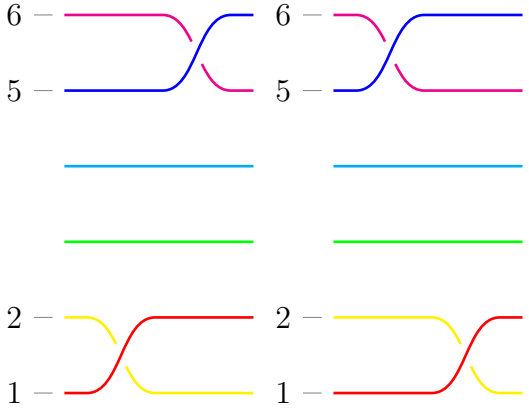


FIGURE 3.6: The braid $\sigma_1\sigma_5$ on the left and $\sigma_5\sigma_1$ on the right, illustrating the relation $\sigma_i\sigma_j = \sigma_j\sigma_i$ for $|i-j| \geq 2$. The braids are equivalent up to ambient isotopy.



An alternate presentation is the ***Birman-Ko-Lee Presentation***, also called the ***Band Generator Presentation***. The generators of the Birman-Ko-Lee presentation satisfy

$$a_{ts}a_{rq} = a_{rq}a_{ts}, \text{ if } [s,t] \cap [q,r] = 0$$

and

$$a_{ts}a_{sr} = a_{tr}a_{ts} = a_{sr}a_{tr}, \text{ for } 1 \leq r < s < t \leq n$$

In terms of the Artin generators,

$$a_{ts} = (\sigma_{t-1}\sigma_{t-2}\cdots\sigma_{s+1})\,\sigma_s\,(\sigma_{s+1}^{-1}\cdots\sigma_{t-2}^{-1}\sigma_{t-1}^{-1})$$

FIGURE 3.7: The braid $\sigma_3\sigma_4$ on the left and $\sigma_4\sigma_3$ on the right, illustrating that $\sigma_i\sigma_j$ is not equal to $\sigma_j\sigma_i$ for $|i-j|=1$. On the left, the strand that starts at position 5 ends at position 4, while on the right, the strand that begins at position 5 terminates at postion 3.

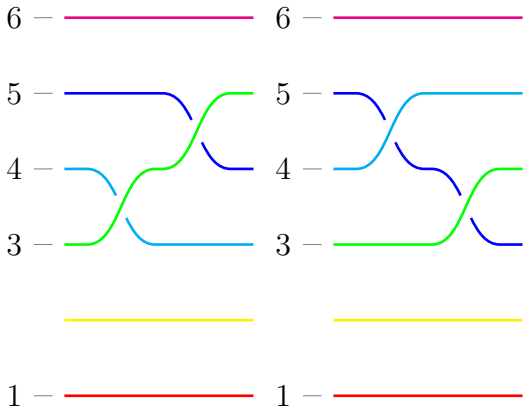
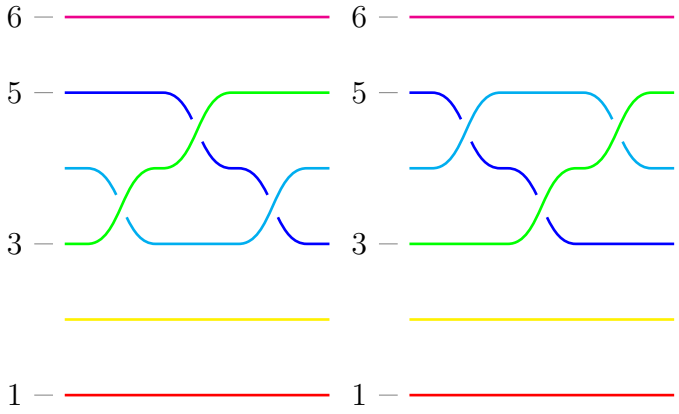


FIGURE 3.8: The braid $\sigma_3\sigma_4\sigma_3$ on the left and $\sigma_4\sigma_3\sigma_4$ on the right, illustrating the commutation relation $\sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}$. The braids are equivalent up to ambient isotopy.



a_{ts} passes the strand t over the strand s, exchanging t with s. While the Artin generators exchange only neighboring strands, the two strands that the Birman-Ko-Lee generators exchange are arbitrary. The strands not involved in the exchange are passed over. The Artin generator σ_i is identified with the Birman-Ko-Lee generator $a_{i+1,i}$.

3.3 Canonical Forms

Definition 3.3. A ***braid word***, or just ***word***, on some subset $B' \in \mathrm{B}_n$ is an explicitly written product of elements in B'. The set B' is called the ***alphabet***, while the elements of B' are referred to as ***letters***.

Definition 3.4. $\tau : \mathrm{B}_n \longrightarrow \mathrm{B}_n$ is the involution defined by $\tau : \sigma_i \mapsto \sigma_{n-i}$.

FIGURE 3.9: The braid $\sigma_2\sigma_2^{-1}$ on the left and $\sigma_4\sigma_4$ on the right. Except for the trivial braid, no braid is its own inverse. $\sigma_2\sigma_2^{-1}$ is equivalent to the trivial braid, while $\sigma_4\sigma_4$ is not, even though the strands that start at positions 4 and 5 terminate at the same positions, 4 and 5. If we take the quotient of the braid group B_n with the relations $\sigma_i\sigma_i$ we recover the symmetric group S_n.

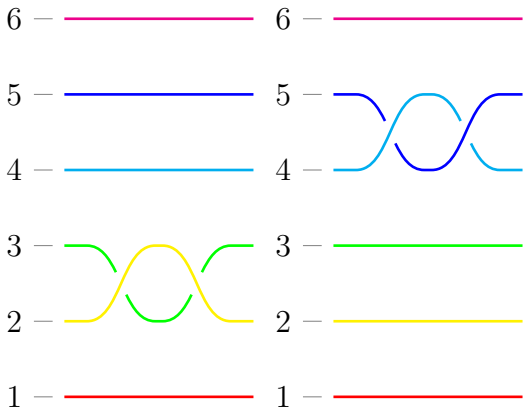
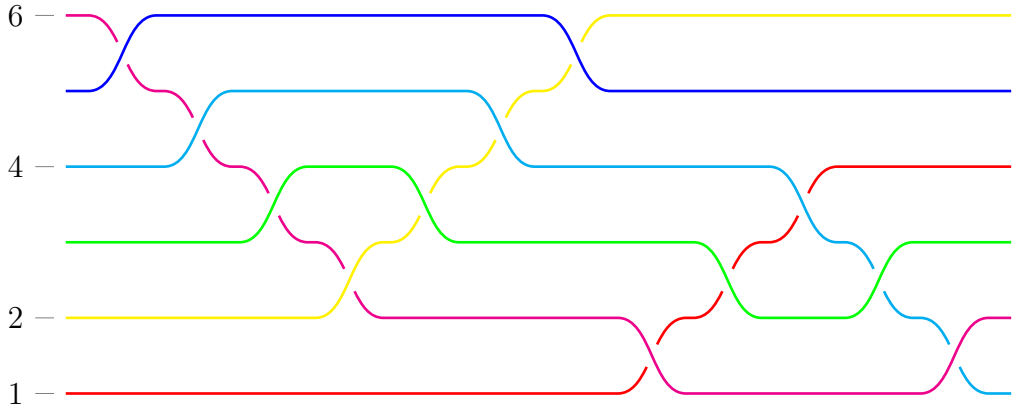


FIGURE 3.10: Birman-Ko-Lee generators in terms of Artin generators. The braid $a_{6,2}a_{4,1}^{-1}$: $a_{6,2}$ followed by $a_{1,4}^{-1}$. Strand 6 passes over strand 2, followed by strand 4 passing under strand 1.



τ flips over the braid about its central (horizontal) axis.

3.4 Normal (Canonical) Forms

In this section, two canonical forms for the braid groups will be defined. The first will be the ***Garside greedy normal form*** in Proposition 3.25, the the other will be the ***Birman-Ko-Lee canonical form***.

Definition 3.5. The ***positive braids*** are the elements of B_n^+, the positive submonoid of B_n generated by $\sigma_1, \sigma_2, \cdots, \sigma_{n-1}$.

The trivial braid, e_n, is a member of B_n^+, as it serves as the identity of the monoid.

Definition 3.6. A ***positive braid word*** is a braid word on B_n^+.

Definition 3.7. The ***weight*** of a braid word $w \in \mathrm{B}'_n$, written $\mathrm{wt}(w)$, is the abelianization map $\mathrm{wt} : \mathrm{B}_n \longrightarrow \mathbf{Z}$, with $\mathrm{wt}(\sigma'_i) = 1,\ \sigma'_i \in \mathrm{B}'_n$. This is also referred to as $\mathrm{len}(w)$, the ***length*** of w.

In our diagrams positive braids are ones in which strands have a negative slope when they cross over other strands. That is, a strand moves to the right and downwards when passing over another strand. A partial ordering on B_n through B_n^+ is defined in Elrifai and Morton [8].

Definition 3.8. If for some pair $A, B \in \mathrm{B}_n$, $\exists P_\ell, P_r \in \mathrm{B}_n^+$ such that $B = P_\ell A P_r$, then we write $A \leq B$.

For any *positive* braid $B \in \mathrm{B}_n^+$ we immediately obtain

Proposition 3.9. *Let $B \in B_n^+$, then*

$$e \leq B$$

Proof. The identity braid on B_n satisfies $\mathrm{e}\mathrm{e}B = B$. □

and

Proposition 3.10. *If either $B = a_\ell A$ or $B = A a_r$ for some $a_\ell \in B_n^+$ or some $a_r \in B_n^+$, then*

$$A \leq B$$

Proof. If $B = a_\ell A$, then $B = a_\ell A \mathrm{e}$ and $A \leq B$. If $B = A a_r$, then $B = \mathrm{e} A a_r$ and $A \leq B$. □

Taking the inverse of a braid is contravariant with respect to this partial ordering since

Proposition 3.11.

$$A \leq B \iff B^{-1} \leq A^{-1}$$

Proof. ($\Longrightarrow$) If $A \leq B$ then for some $a_\ell, a_r \in \mathrm{B}_n^+$, $B = a_\ell A a_r \implies B^{-1} = a_r^{-1} A^{-1} a_\ell^{-1} \implies A^{-1} = a_\ell B^{-1} a_r$. Then, $B^{-1} \leq A^{-1}$. ($\Longleftarrow$) This is the same case, with a change of variable, and noting that taking the inverse is an involution. □

The braids Δ_m are defined inductively,

$$\begin{aligned} \Delta_1 &= 1 \\ \Delta_m &= \Delta_{m-1} \sigma_{m-1} \sigma_{m-2} \cdots \sigma_1 \end{aligned}$$

Definition 3.12. The ***fundamental braid on*** $\boldsymbol{B_n^+}$ is the braid Δ_n. We usually omit the subscript and write Δ for the fundamental braid.

That the Δ_ms are positive braids is clear from the definition. The pattern given through induction is

$$\begin{aligned}
\Delta_2 &= \Delta_1\sigma_1 = \sigma_1 = (\sigma_1)\\
\Delta_3 &= \Delta_2\sigma_2\sigma_1 = \Delta_1\sigma_1\sigma_2\sigma_1 = \sigma_1\sigma_2\sigma_1 = (\sigma_1)\,(\sigma_2\sigma_1)\\
\Delta_4 &= \Delta_3\sigma_3\sigma_2\sigma_1 = \sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1 = (\sigma_1)\,(\sigma_2\sigma_1)\,(\sigma_3\sigma_2\sigma_1)\\
\Delta_5 &= \Delta_4\sigma_4\sigma_3\sigma_2\sigma_1 = \sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_1\sigma_4\sigma_3\sigma_2\sigma_1 = (\sigma_1)\,(\sigma_2\sigma_1)\,(\sigma_3\sigma_2\sigma_1)\,(\sigma_4\sigma_3\sigma_2\sigma_1)\\
&\vdots\\
\Delta_m &= \Delta_{m-1}\sigma_{m-1}\sigma_{m-2}\cdots\sigma_1 = (\sigma_1)\,(\sigma_2\sigma_1)\,(\sigma_3\sigma_2\sigma_1)\cdots(\sigma_{m-1}\sigma_{m-2}\cdots\sigma_1)
\end{aligned}$$

There are equivalent formulae for Δ_m, and for any $i \leq m-2$,

$$\begin{aligned}
\Delta_m &= (\sigma_1)\,(\sigma_2\sigma_1)\,(\sigma_3\sigma_2\sigma_1)\cdots(\sigma_{m-1}\sigma_{m-2}\cdots\sigma_1)\\
&= (\sigma_1\sigma_2\cdots\sigma_{m-1})\cdots(\sigma_1\sigma_2)(\sigma_1)\\
&= (\sigma_i)(\sigma_{i+1}\sigma_i)\cdots(\sigma_{n-1}\cdots\sigma_i)(\sigma_{i-1}\sigma_i\cdots\sigma_{m-1})\cdots(\sigma_1\cdots\sigma_{m-1})\\
&= (\sigma_1\cdots\sigma_{m-1})\cdots(\sigma_1\cdots\sigma_{i+1})(\sigma_i\cdots\sigma_1)\cdots(\sigma_i\sigma_{i-1})(\sigma_i)\\
&= (\sigma_{m-1})\,(\sigma_{m-2}\sigma_{m-1})\,(\sigma_{m-3}\sigma_{m-2}\sigma_{m-1})\cdots(\sigma_1\sigma_2\cdots\sigma_{m-1})\\
&= (\sigma_{m-1}\sigma_{m-2}\cdots\sigma_1)\cdots(\sigma_{m-1}\sigma_{m-2})(\sigma_{m-1})
\end{aligned}$$

The commutation relation for Δ_m is

$$\Delta_m\sigma_{m-i} = \sigma_i\Delta_m$$

Or as a conjugation,

$$\Delta_m^{-1}\sigma_i\Delta_m = \sigma_{m-i}$$

In the case of the fundamental braid, where $m = n$, this can be written as

$$\Delta^{-1}\sigma_i\Delta = \sigma_{n-i} = \tau(\sigma_i)$$

and see that

$$\tau : \sigma_i \mapsto \Delta^{-1}\sigma_i\Delta$$

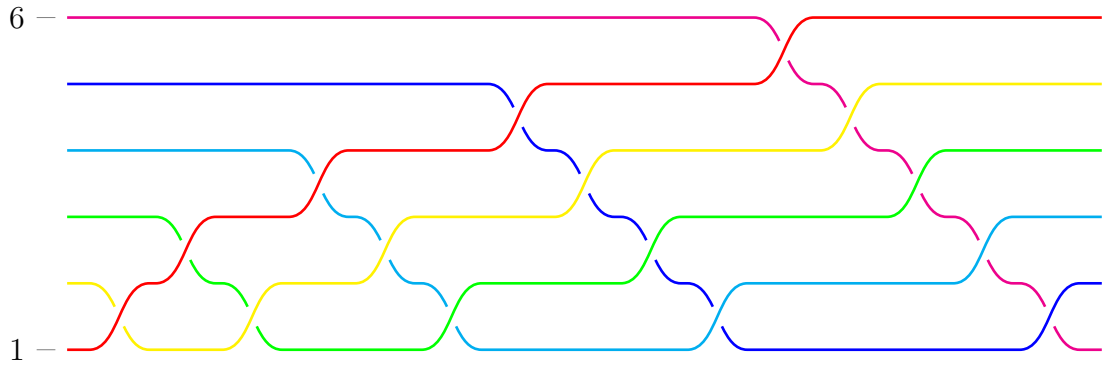
maps σ_i to its conjugation by Δ. τ is an involution, so that the relation generalizes immediately for integral powers of Δ,

$$\Delta^{-2k}\sigma_i\Delta^{2k} = \sigma_i$$

while

$$\Delta^{-(2k+1)}\sigma_i\Delta^{2k+1} = \tau(\sigma_i)$$

FIGURE 3.11: $\Delta_6 \in B_6^+$ as generated by our derived formula. The geometric picture is that of a half twist of positive orientation over all of the strands. The orientation is from left to right. The strands are numbered from bottom to top.



The center of B_n, is generated by Δ^2. This is the braid in which every strand crosses every other strand exactly twice, once as the overpassing and once as the underpassing strand. The braid group, in fact, is highly non-commutative. The centralizer for a typical braid, $b \in \mathrm{B}_n$, is usually generated by Δ^2 and b itself.

Definition 3.13. A positive braid $b \in \mathrm{B}_n^+$ is called a ***left divisor*** of $p \in \mathrm{B}_n^+$ if $\exists b' \in \mathrm{B}_n^+$ such that $p = bb'$. The ***right divisor*** is defined analogously.

Definition 3.14. A positive braid $b \in \mathrm{B}_n^+$ is called ***simple*** if it is a left divisor of Δ. That is, $\Delta = bb'$ for some $b' \in \mathrm{B}_n^+$. A simple braid is also referred to as a ***positive permutation*** braid.

Proposition 3.15. *If b is a simple braid, then*

$$e \leq b \leq \Delta$$

Proof. b is a simple braid, so that for some $b' \in \mathrm{B}_n^+$, $\Delta = bb' = \mathrm{e}bb'$, and so, $b \leq \Delta$. $b = \mathrm{e}eb$, so $\mathrm{e} \leq b$. Hence $\mathrm{e} \leq b \leq \Delta$. □

Δ_n is the positive braid of n strands in which every strand crosses every other strand exactly once; it can be visualized as a set of n strands given a half twist. A simple braid is a positive braid in which a strand crosses any other strand at most once.

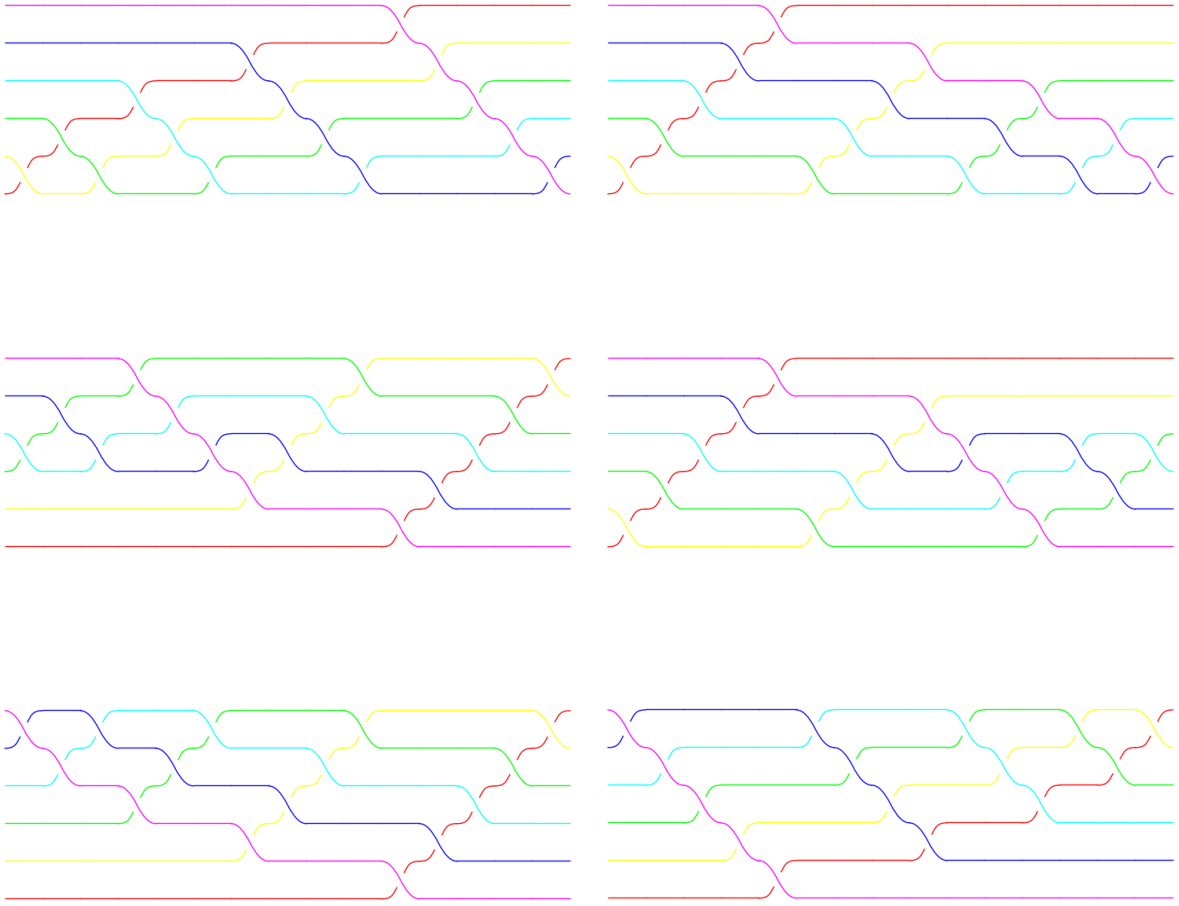
Definition 3.16. A positive braid g is called the ***greatest common divisor*** of the positive braids a and b if g is a left divisor of both a and b, and any other left divisor of both a and b is a left divisor of g. We write $g = \gcd(a, b)$.

Proposition 3.17. *For any generator σ_i*

$$e \leq \sigma_i \leq \Delta$$

Proof. $\Delta = (\sigma_i)(\sigma_{i+1}\sigma_i)\cdots(\sigma_{n-1}\cdots\sigma_i)(\sigma_{i-1}\sigma_i\cdots\sigma_{m-1})$, so that σ_i is simple. The result follows by Proposition 3.15. □

FIGURE 3.12: The six representations of Δ_6 corresponding to the six formulae given. These are all the same braid up to ambient isotopy in $\mathbf{R}^3$. The six diagrams are sequenced left to right and then down the page, and are given in the same order as the formulae in the text. i is 3.



It follows from Proposition 3.17 that Δ is a common left multiple of all generators σ_i of B_n^+. A similar line of reasoning shows that Δ is also a common right multiple, so that

$$\Delta = p_\ell \sigma_i = \sigma_i p_r$$

for some positive braids $p_\ell, p_r \in \mathrm{B}_n^+$.

Proposition 3.18. $\Delta^s \leq A \iff A = a_\ell \Delta^s = \Delta^s a_r$, *for some* $a_\ell, a_r \in B_n^+$.

Proof. ($\Longrightarrow$) If $\Delta^s \leq A$, then for some $a'_\ell, a'_r \in \mathrm{B}_n^+$, $A = a'_\ell \Delta^s a'_r = a'_\ell \tau^s(a'_r)\, \Delta^s = \Delta^s \tau^s(a'_\ell)\, a'_r$. For $b \in \mathrm{B}_n^+$, then $\tau^s(b) \in \mathrm{B}_n^+$, since $\tau(b) \in \mathrm{B}_n^+$. Let $a_\ell = a'_\ell \tau^s(a'_r)$ and $a_r = \tau^s(a'_\ell)\, a'_r$, and the result follows. ($\Longleftarrow$) If $A = a_\ell \Delta^s = \Delta^s a_r$ then $\Delta^s \leq A$ by Proposition 3.10. □

Proposition 3.19. $B \leq \Delta^t \iff \Delta^t = d_\ell B = B d_r$, *for some* $d_\ell, d_r \in B_n^+$.

Proof. ($\Longrightarrow$) If $B \leq \Delta^t$ then, for some $c_\ell, c_r \in \mathrm{B}_n^+$, $\Delta^t = c_\ell B c_r \implies c_\ell^{-1} \Delta^t = B c_r \implies \Delta^t \tau^t\left(c_\ell^{-1}\right) = \Delta^t \left(\tau^t(c_\ell)\right)^{-1} = B c_r \implies \Delta^t = B c_r \tau^t(c_\ell) = B d_r$, where $d_r = c_r \tau^t(c_\ell)$.

Similarly, $\Delta^t = c_\ell B c_r \implies \Delta^t c_r^{-1} = c_\ell B \implies \tau^t(c_r^{-1})\,\Delta^t = (\tau^t(c_r))^{-1}\,\Delta^t = c_\ell B \implies \Delta^t = \tau^t(c_r)\,c_\ell B = d_\ell B$, where $d_\ell = \tau^t(c_r)\,c_\ell$. ($\Longleftarrow$) $\Delta^t = d_\ell B = B d_r \implies \Delta^t = d_\ell B \mathrm{e} = \mathrm{e} B d_r \implies B \leq \Delta^t$. □

Proposition 3.20. *For $\Delta^{s_1} \leq C \leq \Delta^{t_1}$ and $\Delta^{s_2} \leq D \leq \Delta^{t_2}$,*

$$\Delta^{s_1+s_2} \leq CD \leq \Delta^{t_1+t_2}$$

Proof. $\Delta^{s_1} \leq C \implies C = c_\ell \Delta^{s_1} c_r$ for some $c_\ell, c_r \in \mathrm{B}_n^+$. $\Delta^{s_2} \leq D \implies D = d_\ell \Delta^{s_2} d_r$ for some $d_\ell, d_r \in \mathrm{B}_n^+$. Then, $CD = c_\ell \Delta^{s_1} c_r d_\ell \Delta^{s_2} d_r = \tau^{s_1}(c_r)\,c_\ell \Delta^{s_1}\Delta^{s_2} d_r \tau^{s_2}(d_\ell) = a_\ell \Delta^{s_1+s_2} a_r$ for some $a_\ell, a_r \in \mathrm{B}_n^+$. Thus, $\Delta^{s_1+s_2} \leq CD$. $C \leq \Delta^{t_1} \implies \Delta^{t_1} = c'_\ell C c'_r$ for some $c'_\ell, c'_r \in \mathrm{B}_n^+$. Then, $C = c_r'^{-1}\Delta^{t_1} c_\ell'^{-1} = c_r'^{-1}\tau^{t_1}\left(c_\ell'^{-1}\right)\Delta^{t_1} = c_r'^{-1}\,(\tau^{t_1}(c'_\ell))^{-1}\,\Delta^{t_1}$. $D \leq \Delta^{t_2} \implies \Delta^{t_2} = d'_\ell D d'_r$ for some $d'_\ell, d'_r \in \mathrm{B}_n^+$. Then, $D = d_r'^{-1}\Delta^{t_2} d_\ell'^{-1} = \Delta^{t_2}\tau^{t_2}\left(d_r'^{-1}\right) d_\ell'^{-1} = \Delta^{t_2}\,(\tau^{t_2}(d'_r))^{-1}\,d_\ell'^{-1}$. Then, $CD = c_r'^{-1}\,(\tau^{t_1}(c'_\ell))^{-1}\,\Delta^{t_1}\Delta^{t_2}\,(\tau^{t_2}(d'_r))^{-1}\,d_\ell'^{-1}$. Thus, $\Delta^{t_1}\Delta^{t_2} = \Delta^{t_1+t_2} = d'_\ell \tau^{t_2}(d'_r)\,CD\tau^{t_1}(c'_\ell)\,c'_r = b_\ell CD b_r$ for some $b_\ell, b_r \in \mathrm{B}_n^+$. Therefore, $CD \leq \Delta^{t_1+t_2}$. □

Proposition 3.21. *$\forall B \in B_n$,*

$$\Delta^s \leq B \leq \Delta^t$$

for some $s, t \in \boldsymbol{Z}$.

Proof. Note that $\mathrm{e} = \Delta^0$, $\mathrm{e} \leq \sigma_i \leq \Delta$ and $\Delta^{-1} \leq \sigma_i^{-1} \leq \mathrm{e}$. Expressing B as a word in $\left\{\sigma_i^{\pm 1}\right\}$ and applying Proposition 3.20 repeatedly gives $\Delta^{-m} \leq B \leq \Delta^p$, where m is the number of σ_i^{-1} and p is the number of σ_i occurring in B. □

Definition 3.22. $[s, t]$ is the subset of B_n

$$\left\{B \mid \Delta^s \leq B \leq \Delta^t,\ B \in \mathrm{B}_n\right\}$$

Definition 3.23. For $B \in \mathrm{B}_n$, $\sup(B) = \min\{t \mid B \leq \Delta^t\}$, while $\inf(B) = \max\{s \mid \Delta^s \leq B\}$.

Definition 3.24. For $B \in \mathrm{B}_n$, $\ell(B) = \sup(B) - \inf(B)$ is called the ***canonical length*** of B.

It follows that every braid in B_n can be written in the form $\Delta^k b$ where $k \in \mathbf{Z}$ and $b \in \mathrm{B}_n^+$. If $\Delta^k = \inf(B)$, then k is maximal, and this form is unique. We have,

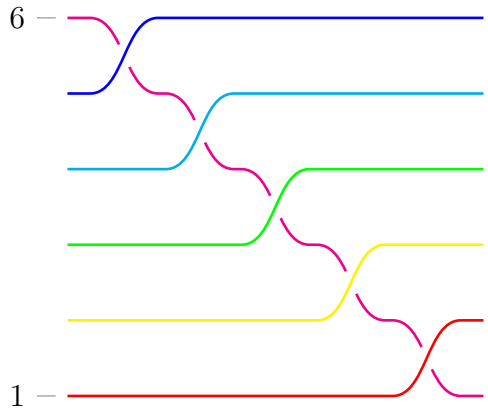
Proposition 3.25. *For each braid b in B_n there is a unique decomposition of the form*

$$b = \Delta^k b_1 \cdots b_r$$

*with each b_i in B_n^+. This decomposition is called the **greedy normal form** of the braid b.*

Proof. Let $\Delta^k = \inf(B)$. Then, $b = \Delta^k b'$ for some $b' \in \mathrm{B}_n^+$. If $b_1 = \gcd(\Delta, b')$ then b_1 is simple and it is a left divisor of b'. We can then write $\Delta^k b = \Delta^k b_1 b''$ for some $b'' \in \mathrm{B}_n^+$. We let $b_2 = \gcd(\Delta, b'')$ and repeat the decomposition. This process continues while each b_i is distinct of the trivial braid, e. □

FIGURE 3.13: δ_6. δ_n takes strand n to position 1 and shifts all the other strands up one position. If we apply δ_n n times each strand returns to the position it started at, so that we have a full twist on n strands.



Definition 3.26. The set $\{B \mid \Delta^s \leq B \leq \Delta^t,\ B \in \mathrm{B}_n\}$ is denoted $[s,t]$.

Theorem 3.27. *For $A, B \in B_n$, $\ell(AB) \leq \ell(A) + \ell(B)$. This is equivalent to*

$$[s_1,t_1]\,[s_2,t_2] \subset [s_1+s_2, t_1+t_2]$$

and

$$\inf(AB) \geq \inf(A) + \inf(B)\,, \quad \sup(AB) \leq \sup(A) + \sup(B)$$

Proof. This follows from Proposition 3.20. □

In fact, using Proposition 3.25, it is shown in Elrifai and Morton [8] that the inclusion is an equality.

Definition 3.28 (Birman-Ko-Lee Normal Form). The fundamental braid for the ***Birman-Ko-Lee canonical form*** is

$$\begin{aligned}\delta_n &= a_{n,n-1}a_{n-1,n-2}\cdots a_{3,2}a_{2,1}\\ &= \sigma_{n-1}\sigma_{n-2}\cdots\sigma_2\sigma_1\end{aligned}$$

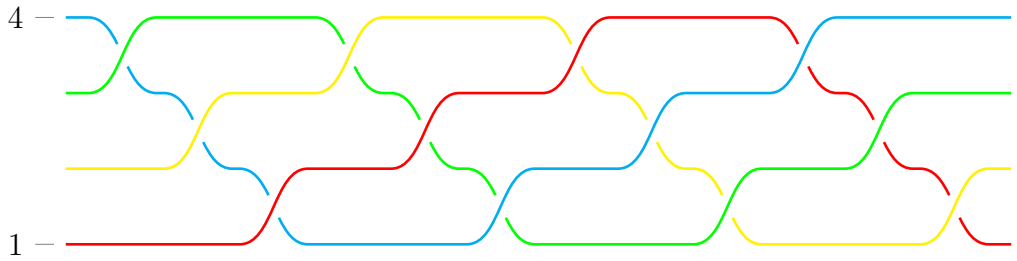
The n^{th} power of δ_n is equivalent to a full twist of the braid strands, so that the relationship between the Garside fundamental form and the Birman-Ko-Lee is

$$\delta_n^n = \Delta_n^2$$

and, so, δ_n^n generates the center of B_n. δ_n is a left and right multiple of any Birman-Ko-Lee generator, so that

$$\delta_n = p_\ell a_{s,t} = a_{s,t} p_r$$

FIGURE 3.14: δ_4^4: A full twist on 4 strands.



for some positive braids $p_\ell, p_r \in \mathrm{B}_n^+$. The commutation relation is

$$a_{s,t}\delta_n = \delta_n a_{s+1,t+1}$$

3.5 Linear Representations

In this section we will describe three linear representations of the braid groups, the ***Burau***, the ***Colored Burau***, and the ***Lawrence-Krammer*** representations.

The (unreduced) ***Burau representation*** is the mapping $\rho_B : \mathrm{B}_n \longrightarrow \mathrm{GL}(n, \mathbf{Z}[t, t^{-1}])$ given by

$$\rho_B(\sigma_i)(t) \equiv \mathrm{I}_{i-1} \oplus \begin{pmatrix} -t & t \\ 1 & 0 \end{pmatrix} \oplus \mathrm{I}_{n-i-1}$$

with inverse

$$\rho_B\left(\sigma_i^{-1}\right)(t) \equiv \mathrm{I}_{i-1} \oplus \begin{pmatrix} 0 & 1 \\ 1/t & 1 \end{pmatrix} \oplus \mathrm{I}_{n-i-1}$$

The Burau representation can be reduced by one dimension to give an $(n-1)$-dimensional representation, the reduced Burau representation. The mapping $\rho_{RB} : \mathrm{B}_n \longrightarrow \mathrm{GL}(n-1, \mathbf{Z}[t, t^{-1}])$ is given by

$$\rho_{RB}(\sigma_i)(t) \equiv \begin{cases} \begin{pmatrix} -t & 1 \\ 0 & 1 \end{pmatrix} \oplus \mathrm{I}_{n-3}, & \text{if } i = 1 \\ \mathrm{I}_{i-2} \oplus \begin{pmatrix} 1 & 0 & 0 \\ t & -t & 1 \\ 0 & 0 & 1 \end{pmatrix} \oplus \mathrm{I}_{n-i-2}, & \text{if } 1 < i < n-1 \\ \mathrm{I}_{n-3} \oplus \begin{pmatrix} 1 & 0 \\ t & -t \end{pmatrix}, & \text{if } i = n \end{cases}$$

with inverse

$$\rho_{RB}\left(\sigma_i^{-1}\right)(t) \equiv \begin{cases} \begin{pmatrix} -1/t & 1/t \\ 0 & 1 \end{pmatrix} \oplus \mathrm{I}_{n-3}, & \text{if } i = 1 \\ \mathrm{I}_{i-2} \oplus \begin{pmatrix} 1 & 0 & 0 \\ 1 & -1/t & 1/t \\ 0 & 0 & 1 \end{pmatrix} \oplus \mathrm{I}_{n-i-2}, & \text{if } 1 < i < n-1 \\ \mathrm{I}_{n-3} \oplus \begin{pmatrix} 1 & 0 \\ 1 & -1/t \end{pmatrix}, & \text{if } i = n \end{cases}$$

From the block form of the Burau, it is easily seen that $\rho(\sigma_i)$ commutes with $\rho(\sigma_j)$ for $|i-j| > 1$. The matrices also obey $\rho(\sigma_i)\,\rho(\sigma_{i+1})\,\rho(\sigma_i) = \rho(\sigma_{i+1})\,\rho(\sigma_i)\,\rho(\sigma_{i+1})$, so that all the braid relations are satisfied.

The Burau representation is known to be faithful for $n = 2$ and $n = 3$ [20]. It is not faithful for $n \geq 5$ [4]. The kernel, however, is very small, so that the chance of collision between matrices is minute. For $n = 5$ the shortest known element of the kernel is a word of length 120 in the generators $\sigma_1, \ldots, \sigma_4$ [28]. Whether the Burau representation is faithful for $n = 4$ is an open question. While there are heuristic algorithms that have a high success rate in finding the inverse mapping, from a Burau matrix to an element of the braid group, there is no known deterministic algorithm to do so.

Definition 3.29. Let $b = \sigma_{i_1}^{\epsilon_1}\sigma_{i_2}^{\epsilon_2}\cdots\sigma_{i_n}^{\epsilon_n}$ be a word in B_n with $\epsilon_j = \pm 1$. Let t_k be a labeling of the strands in the braid with t_k the strand that starts at position k. Let t_{j_r} be the label of the underpassing strand that is at the r^{th} crossing. Then the ***colored Burau matrix*** is given by

$$C_b(t_1, \ldots, t_n) = \prod_{r=1}^{n} \left(\rho_{RB}(\sigma_{i_r})(t_{j_r})\right)^{\epsilon_r}$$

The ***colored Burau group***, CB_n is the set

$$\mathrm{S}_n \times \mathrm{GL}\left(n-1, \mathbf{Z}\left[t_1^{\pm 1}, \ldots, t_n^{\pm 1}\right]\right)$$

with multiplication

$$(\alpha_1, C_1) \cdot (\alpha_2, C_2) = \left(\alpha_1\alpha_2, \left(\alpha_2^{-1}C_1\right) C_2\right)$$

and representation $\rho_{CB} : \mathrm{B}_n \longrightarrow \mathrm{CB}_n$ given by

$$\rho_{CB}(\sigma_i) = \left((i, i+1), \rho_{RB}(\sigma_i(t_i))\right)$$

The identity braid has representation

$$\rho_{CB}(\mathrm{e}) = \left(() , \mathrm{I}_{n-1}\right)$$

and the inverse by

$$(\alpha, C)^{-1} = \left(\alpha^{-1}, \alpha C^{-1}\right)$$

The ***Lawrence-Krammer*** representation is a mapping $\rho : \mathrm{B}_n \longrightarrow \mathrm{GL}(n(n-1)/2, \mathbf{Z}[t^{\pm}, q^{\pm}])$. It was shown in Krammer [18] to be faithful for B_4. Bigelow [5] showed that the Lawrence-Krammer representation is faithful for all n. Since there is an injective mapping from the braid group to a matrix group, the braid group is, in fact, linear.

3.6 Summit Sets

Summit sets are used in a direct approach to solve the conjugacy problem. Let x be an arbitrary element of B_n: We consider the set, I_x, of x with the following properties:

- I_x is finite and non-empty.
- I_x only depends on the conjugacy class of x.

  That is, $x, y \in \mathrm{B}_n$ are conjugate iff $I_x = I_y$.
- There is an efficient algorithm to compute a representative $\tilde{x} \in I_x$ and a conjugating element $a \in \mathrm{B}_n$ such that $axa^{-1} = \tilde{x}$.
- The whole set I_x can be computed from $\tilde{x}$ with a finite algorithm.

To solve the CDP/CSP for $\forall x, y \in \mathrm{B}_n$:

- Find the representatives $\tilde{x}$ of I_x and $\tilde{y}$ of I_y.
- Compute each further element of I_x using the algorithm mentioned above and note the conjugating element.

  If this element is found to be $\tilde{y}$, then x and y are conjugates, and we also have the conjugating element.
- If the entire set I_x has been constructed, and $\tilde{y}$ has still not been found among the elements of I_x, then x and y are not conjugate.

We call this set the ***Summit Set***, $SS(x)$, of x, when this is the set of conjugates of x having maximal infimum.

The ***Super Summit Set***, $\mathrm{SSS}(x)$, of x is the set of conjugates of x with minimal possible complexity. These are the conjugates of x having minimal canonical length, $\ell(x)$; it is the set having both maximal infimum and minimal supremum. $\mathrm{SSS}(x)$ is a much smaller set than $\mathrm{SS}(x)$.

Definition 3.30. Let $x = \Delta^p x_1 \cdots x_r \in \mathrm{B}_n$ be in Garside normal form with $r > 0$. Then $\mathbf{c}(x)$, the cycling of x is given by

$$\mathbf{c}(x) = \Delta^p x_2 \cdots x_r \tau^p(x_1)$$

where τ, defined previously, is the involution defined by $\tau : \sigma_i \mapsto \sigma_{n-i}$.

Cycling moves the first simple element of x, x_1, from the first position after Δ^p to the end of the word, where it is replaced by $\tau^p(x_1)$.

Proposition 3.31. *With x, r as in Definition 3.30, $\mathbf{c}(x)$ is conjugate to x, and* $\inf(\mathbf{c}(x)) \geq \inf(x)$.

Proof. Note that $\forall x_i \in \mathrm{B}_n$, $\tau^{-p}(x_i) = \tau^p(x_i)$ and $\tau\left(x_i^{-1}\right) = (\tau(x_i))^{-1}$. The definition of $\mathbf{c}(x)$ gives

$$\begin{aligned}
\mathbf{c}(x) &= \Delta^p x_2 \cdots x_r \tau^p(x_1) \\
&= \Delta^p x_1^{-1} x_1 x_2 \cdots x_r \tau^p(x_1) \\
&= \tau^p\left(x_1^{-1}\right) \Delta^p x_1 x_2 \cdots x_r \tau^p(x_1) \\
&= (\tau^p(x_1))^{-1} (\Delta^p x_1 x_2 \cdots x_r) \, (\tau^p(x_1)) \\
&= (\tau^p(x_1))^{-1} \, x \, (\tau^p(x_1))
\end{aligned}$$

so that $\mathbf{c}(x)$ is conjugation of x by $\tau^p(x_1)$. Since the x_i are simple, $\inf(x) = \inf(\Delta^p x_1 \cdots x_r) = p$. Similarly it follows that $\inf(\Delta^p x_2 \cdots x_r \tau^p) = p$ and $\inf(\tau(x_1)) = 0$. From Theorem 3.27, $\inf(c) = \inf(\Delta^p x_2 \cdots x_r \tau^p \cdot x_1) \geq \inf(\Delta^p x_2 \cdots x_r \tau^p) + \inf(\tau(x_1)) = p = \inf(x)$. □

Definition 3.32. The decycling of x, $\mathbf{d}(x)$, is given by

$$\mathbf{d}(x) = x_r \Delta^p x_1 \cdots x_{r-1} = \Delta^p \tau^p(x_r) \, x_1 \cdots x_{r-1}$$

Deycling moves the last braid of x, x_r, from the end of the word to the start of the word. When $r = 0$, $\mathbf{c}(x) = \mathbf{d}(x) = x$.

Proposition 3.33. *With x, r as in Definition 3.32, $\boldsymbol{d}(x)$ is conjugate to x, and* $\sup(\boldsymbol{d}(x)) \leq \sup(x)$.

Proof. The definition of $\mathbf{d}(x)$ gives

$$\begin{aligned}
\mathbf{d}(x) &= x_r \Delta^p x_1 \cdots x_{r-1} \\
&= x_r \Delta^p x_1 \cdots x_{r-1} x_r x_r^{-1} \\
&= x_r (\Delta^p x_1 \cdots x_{r-1} x_r) \, x_r^{-1} \\
&= x_r x x_r^{-1}
\end{aligned}$$

so that $\mathbf{d}(x)$ is conjugation of x by x_r^{-1}. The rest of the proof is similar to the proof of Proposition 3.31. □

Repeated cycling will achieve the maximum value for inf without increasing the sup, while repeated decycling will achieve the minimum value for sup without increasing the inf, so that both bounds can be achieved simultaneously. If m is the length of Δ as a word in the Artin generators, and ℓ is the canonical length of x, then after at most $m\ell$ cyclings and decyclings, the maximum value for inf and the minimum value for sup will be achieved [6].

To solve the CDP/CSP for $\forall x, y \in \mathrm{B}_n$:

- Find a conjugate $\tilde{x}$ of x in $\mathrm{SSS}(x)$ using cycling and decycling.
- Find a conjugate $\tilde{y}$ of y in $\mathrm{SSS}(y)$ using cycling and decycling.
- Starting with $\tilde{x}$ compute each further element of $\mathrm{SSS}(x)$ using simple conjugation and note the conjugating element.

  If this element is found to be $\tilde{y}$, then x and y are conjugates, and we also have the conjugating element.
- If the entire set I_x has been constructed, and $\tilde{y}$ has still not been found among the elements of I_x, then x and y are not conjugate.

Prior to each cycling or decycling the Garside normal form needs to be calculated since, in general, since neither $\mathbf{c}(x)$ or $\mathbf{d}(x)$ is in Garside normal form.

The ***Ultra Summit Set***, introduced in Gebhardt [11], is a refinement of the Super Summit Set.

Definition 3.34. Let x be an arbitrary element of B_n. The Ultra Summit Set of x, $\mathrm{USS}(x)$, is the set of conjugates of x such that for $y \in \mathrm{USS}(x)$ and $m > 0$, $\mathbf{c}^m(y) = y$.

The members of the USS are the elements of the SSS that are in closed orbits under cycling. This is normally a much smaller set than the SSS. Typically, the size of the SSS is exponential with respect to the length of the initial braid, while for the USS it may be linear. If so, it is possible that the CSP can be solved in polynomial time.

3.7 Other Braid Group Problems

There are some other problems that may be hard problems in the braid groups.

The ***Root Existence Problem*** asks if given $b \in \mathrm{B}_n$ and $a \in \mathbf{Z}$, does there exist a $c \in \mathrm{B}_n$ such that

$$c^a = b$$

The ***Root Extraction Problem*** asks if given $b \in \mathrm{B}_n$ and $a \in \mathbf{Z}$, find $c \in \mathrm{B}_n$ such that

$$c^a = b$$

The ***Minimal Length Problem*** is the problem that if given a word $w \in \mathrm{B}_n$ over the $\sigma_i^{\pm 1}$s, find the shortest word $w' \in \mathrm{B}_n$ such that $w' \equiv w$. It is known that the Minimum Length Problem is NP-complete [23].

Chapter 4

Braid Group Cryptography

4.1 Introduction

Braid groups provide a natural choice among non-commutative groups as a cryptographic platform. The word problem is known to have a polynomial time algorithmic solution, while the conjugacy search problem is believed to be hard. It is easy to find subgroups that commute with members of other subgroups, since, in the Artin presentation, the generators commute with all but the immediately neighboring generator. This is helpful for some protocols.

These schemes rely on the difficulty of solving the conjugacy problem in the braid group. There are three main approaches that can be taken. We can consider an exact solution, a probabilistic strategy, or try to solve in a different representation.

4.2 Braid Group Platform

All of the cryptosystems in Table 2.2 are easily ported to the braid groups.

The Anshel-Anshel-Goldfeld scheme can be ported to the braid groups as follows:

- We choose parameters $L_1, L_2, n, r, s \in \mathbf{Z}^+$ with $1 \leq L_1 \leq L_2$.
- Let B_n be the braid group on n strands.
- Let $p_1, \cdots, p_r \in \mathrm{B}_n$ and $q_1, \cdots, q_s \in \mathrm{B}_n$ be two sets of letters on B_n, with each p_i and q_i of length between L_1 and L_2 in the Artin generators of B_n.
- $A = \alpha(p_1, \cdots, p_r)$ and $B = \beta(q_1, \cdots, q_s)$ are words on B_n, where A has length L in the p_i and where B has length L in the q_i.
- $\mathrm{B}_n,\ p_1, \cdots, p_r,\ q_1, \cdots, q_s,\ p_1^B, \cdots, p_r^B,\ q_1^A, \cdots, q_s^A$ is public.

- Find $[A, B]$, the commutator of A and B. This is the common key.

The Ko-Lee cryptosystem is implemented as follows:

Let $m = \lfloor n/2 \rfloor$. We denote the subgroup of B_n generated by $\sigma_1, \cdots, \sigma_{m-1}$ by LB_n, and the subgroup of B_n generated by $\sigma_{m+1}, \cdots, \sigma_{n-1}$ by UB_n.

- Alice chooses a public key p in B_n.
- Alice chooses a secret key A in LB_n.

  She sends the conjugate p^A to Bob.
- Bob chooses a secret key B in UB_n.

  He sends the conjugate p^B to Alice.
- Alice computes $t_A = \left(p^B\right)^A = p^{BA} = p^{AB}$.
- Bob computes $t_B = \left(p^A\right)^B = p^{AB}$.
- $t_A = t_B$. This is a common key.

Note that $B \in \mathrm{UB}_n$ and $A \in \mathrm{LB}_n$, and so they commute.

A public key cryptosystem is proposed in Ko et al. [16]:

Bob wishes to send Alice a message $m_B \in \{0,1\}^N$. We let $h : \mathrm{B}_n \longrightarrow \{0,1\}^N$ be a one-way collision-free secure hash function to the message space. Collision free means that the probability that $h(m_1) = h(m_2)$ for $m_1 \neq m_2$ is negligible. $\oplus$ denotes "exclusive or" (addition in $\mathbf{F}_2$). Note that $h(m) \oplus h(m)$ is always equal to the identity for the $\oplus$ operator.

- Alice picks a sufficiently complicated braid $p \in \mathrm{B}_n$.

  She then picks a braid $A \in \mathrm{LB}_n$.

  Alice's public key is $\left(p, p^A\right)$ and her secret key is A.
- Bob picks an arbitrary braid B in UB_n.

  He computes the cipher text $m' = m_B \oplus h\left(\left(p^A\right)^B\right)$ with the auxiliary datum p^B.

  He sends $\left(m', p^B\right)$ to Alice.
- Alice computes $m_A = m' \oplus h\left(\left(p^B\right)^A\right)$.
- Then $m_A = m_B$ and Alice gets back Bob's original message.

B and A commute so,

$$\begin{aligned}\left(p^A\right)^B &= p^{AB}\\ &= p^{BA}\\ &= \left(p^B\right)^A\end{aligned}$$

and

$$m_A = m' \oplus h\Big(\left(p^B\right)^A\Big) = m' \oplus h\left(p^{BA}\right) = m' \oplus h\left(p^{AB}\right) = m_B \oplus h\left(p^{AB}\right) \oplus h\left(p^{AB}\right) = m_B$$

Two signature schemes are proposed in Ko et al. [17]. In the first, we let $H : \{0,1\}^N \longrightarrow \mathrm{B}_n$ be a one-way collision-free secure hash function from the message space to the braid group. Note that this is in the opposite direction of the previous hash function h.

- Alice picks a sufficiently complicated braid $p \in \mathrm{B}_n$.

  She then picks a braid $A \in \mathrm{B}_n$.

  Alice's public key is $\left(p, p^A\right)$ and her secret key is A.
- Alice computes $q = H(m)$.

  She signs the message m with q^A.
- Bob checks that $q^A \sim q$ and that $p^A q^A \sim pq$.

The other is as follows,

- Alice picks a random braid B in B_n.
- Alice computes p^B, $q = H\left(mh\left(p^B\right)\right)$, q^{AB} and q^B

  She signs m with the triple $\left(p^B, q^A, q^{AB}\right)$.
- Bob checks that $p^B \sim p$, $q^B \sim q$, $p^B q^B \sim pq$ and $p^B q^{AB} \sim -p^A q$.

Note that this system works only if the conjugacy decision problem (CDP) is feasible to solve.

In an authentication scheme Bob wants to verify Alice's identity. Alice complies by showing she knows a private key to Bob without an intruder deducing anything about the key. The following Diffie-Hellman-like authentication scheme is proposed in Sibert et al. [27]. $h : \mathrm{B}_n \longrightarrow \{0,1\}^N$ is a collision-free secure hash function to the message space.

- Alice picks a sufficiently complicated braid $p \in \mathrm{B}_n$.

  She then picks a braid $A \in \mathrm{LB}_n$.

  Alice's public key is (p, p_A) and her secret key is A.

- Bob picks an arbitrary braid B in UB_n.

  He sends the challenge p^B to Alice.

- Alice sends the response $y = h\Big(\big(p^B\big)^A\Big) = h\big(p^{BA}\big)$.

- Bob checks that $y = h\Big(\big(p^A\big)^B\Big) = h\big(p^{AB}\big) = h\big(p^{BA}\big)$.

In the same paper another authentication scheme is also proposed.

- Alice picks a sufficiently complicated braid $p \in \mathrm{B}_n$.

  She then picks a braid $A \in \mathrm{B}_n$ (not necessarily in LB_n).

  Alice's public key is $\big(p, p^A\big)$ and her secret key is A.

The next exchanges are repeated k times, where Bob wishes that the probability of a false authentication be $1/2^k$.

- Alice picks an arbitrary braid B in B_n.

  She sends the commitment $x = h\Big(\big(p^A\big)^B\Big)$.

- Bob picks a random bit c and sends it to Alice.

  If $c = 0$, Alice sends $y = B$.

  Bob checks that $x = h\big(\big(p^A\big)^y\big) = h\big(p^{AB}\big)$.

  If $c = 1$, Alice sends $y = BA$.

  Bob checks that $x = h(p^y) = h\big(p^{AB}\big)$.

4.3 Direct Attacks

Summit sets are described in Section 3.6. The Summit-Set approach is used in an attempt to solve the CSP exactly. Because the number of cyclings and decyclings required by the SSS algorithm to solve the conjugacy problem is proportional to the canonical length of the braid, the number of steps in this algorithm is linear in the complexity of the braid. However, the number of simple braids of length n is $n!$, so that the cost to enumerate all simple braids grows faster than exponential. This number quickly becomes unmanageable. For example, for $n = 60$, which would give a reasonably-sized braid, $n!$ is close to the number of atoms in the universe. Franco and Gonzalez-Meneses [9] approach this issue by conjugating by ***minimal simple elements***. These are elements that are the gcd of simple braids. There are no more than n such braids in B_n. Using Ultra Summit Sets is another approach. It seems that the average complexity using USS may be polynomial.

4.4 Length-Based Attacks

This is a probabilistic attack. A brute-force search may succeed solving the conjugacy problem, given enough computational power coupled with non-negligible probabilities of success. Assuming p_A is a conjugate that was derived from conjugating p, then a simple method to find a conjugator for a pair (p, p_A) is to begin iteratively conjugating p_A until either the length or the complexity of the conjugate is minimal. If this new conjugate is equal to p, then there is success. The Multiple Simultaneous CSP is more susceptible to this attack because there is more than one pair with the same conjugator.

A length-based attack on the AAG protocol was first described in Hughes and Tannenbaum [15] and then implemented in Myasnikov and Ushakov [22] for the braid groups. Let G be a group with some canonical representation. Then some length function, ℓ, on the elements of G is defined. Let $A, B \in G$. B is a reducing element of A if $\ell\big(A^B\big) < \ell(A)$. It is critical for this attack that

$$\ell\big(A^B\big) > \ell(A)\,, \quad \text{for the vast majority of } A \text{ and } B, \tag{4.1}$$

so that B is very rarely a reducing agent of A. Now, $A = p_{t_1}^{\epsilon_1} \dots p_{t_L}^{\epsilon_L}$ for some ϵ_i, t_i, so that a sequence to calculate q_i^A is as follows:

$$\begin{gathered} q_i \\ p_{t_L}^{\epsilon_L} q_i p_{t_L}^{-\epsilon_L} \\ p_{t_{L-1}}^{\epsilon_{L-1}} p_{t_L}^{\epsilon_L} q_i p_{t_L}^{-\epsilon_L} p_{t_{L-1}}^{-\epsilon_{L-1}} \\ \vdots \\ p_{t_1}^{\epsilon_1} \dots p_{s_{L-1}}^{\epsilon_{L-1}} p_{t_L}^{\epsilon_L} q_i p_{t_L}^{-\epsilon_L} p_{t_{L-1}}^{-\epsilon_{L-1}} \dots p_{t_1}^{-\epsilon_1} = q_i^B \end{gathered}$$

For the attack, we reverse this sequence and start from the bottom, at q_i^A. We find a reducing agent at each step, and apply the reducing agent the number of times that minimizes the length. That is, we apply some optimal power of a reducing agent at each step. If we are successful, we arrive at q_i, and can read off A. The AAG protocol is particularly vulnerable to this attack when the public keys p_i or q_i are complicated and secret keys A or B is the product of a small number of distinct p_i or q_i. Because of this in Anshel et al. [2] it is suggested that simple public keys and complicated secret keys be used. Finding a suitable length function, and in particular, one in which Equation 4.1 holds, is the main obstacle to implementing a successful attack. There is no guarantee that a suitable length function can be found. There were no successful implementations of the length-based attack of Hughes and Tannenbaum [15], up to Myasnikov and Ushakov [22]. The parameters here were set to

$$n = 80,\ r = s = 20,\ L_1 = 10, 20, 30, 40,\ L_2 = L_1 + 3,\ L = 50$$

The experimental attack and variations had success rates

L_1, L_2	10,13	20,23	30,33	40,43
Original Attack	00 %	51%	97%	96%
Variation 2	00	05	45	60
Variation 4	00	51	80	64
Variation 5	00	30	97	96

where it is seen that increasing the length of the keys makes this scheme less secure under this attack.

In Garber et al. [10] the length functions used in the attack are variations of the Garside length, based on the Garside normal form. They found modest success for low n. For example, out of 500 tries for $n = 4$ and $r = 2$, there were 429 successes. For $n = r = 9$, there were 116, while for $n = 20$, $r = 18$, there were 2.

Lee and Lee [19] gives an algorithm for a length-based attack on the MSCP. While the attack in Hughes [14] suggests that simple public keys be used, in this paper the claim is that this attack is strongest with simple public keys, and is insensitive to how complicated the private keys are.

Gonzalez-Meneses [12] claims to have developed an algorithm that is an improvement to the one in Lee and Lee [19].

A more powerful method is to first include an additional step of finding a conjugate of p_A, say p'_A, with minimal complexity. Then we assume that there is a simple conjugation of p, that is, one by a permutation, that is equal to p'_A. Then this is just solving the CSP in the symmetric group S_n. This is the approach taken in Hofheinz and Steinwandt [13]. The found for $n = 80$ a successful attack rate of $78 - 89\%$ against the Hiffie-Dellman protocol and $99 - 100\%$ against the AAG.

It seems reasonable to believe that how effective a length-based attack is dependent on the complexity of the braids, and how randomly correlated the generating braids truly are to each other.

4.5 Linear-Representation Attacks

The strategy here is to use a linear representation. The CSP is an easy problem in the matrix group. A linear representation of a braid group is one that we can find a matrix representation for. The Burau representation for the braid groups B_n is known to be faithful for $n < 4$ and its status is unknown for $n = 4$. It is unfaithful for $n > 4$. However, the kernel is very small, so that there is a negligible probability that two distinct braids will have the same representation.

The Anshel-Anshel-Goldfeld Scheme relies on the difficulty of the Conjugator Search Problem. An approach to solve the CSP or the Multiple Simultaneous CSP is to take a pair or pairs of conjugate braids, find their classical Burau images, and then solve the problem as a linear group. However, the matrix that is found to be a conjugator may or may not be a Burau matrix. It may also be difficult to find preimage in the braid group,

as there is no general algorithm to do so. The original paper Anshel et al. [1] suggests the parameters

$$n = 80,\ r = s = 20,\ L_1 = 5,\ L_2 = 8,\ L = 100$$

It is now strongly believed that these parameters do not provide a secure cryptosystem as they are susceptible to attack by summit sets, linear attacks, and the attack in Hofheinz and Steinwandt [13].

In Hughes [14] the attack on the cryptosystem using these parameters (except that $L_2 = 5$) is through a linear representation via the Burau representation. Through experimental results they conclude that with modest resources ($\approx$ 410 processors with 5 hr wall time) that it is possible to break the system with 99% success.

Lee and Lee [19] proposes a linear algebraic attack through the colored Burau. A weakness through the key extraction algorithm in Anshel et al. [2] is employed.

The Lawrence-Krammer representation is faithful. The approach here is to also convert a problem in the braid group to a linear one in the matrix group. Here, as well, though, it may be difficult to find a preimage in the braid group.

Cheon and Jun [7] propose a polynomial time algorithm for the Diffie-Hellman conjugacy problem in the braid groups based on the Lawrence-Krammer representation. The authors conclude that while the complexity of the encryption is too great to break [at the time] in real time with the parameters given in Ko et al. [17], the cryptosystem is insecure since increasing the key size increases the security by only a small amount. However, the conjugacy problem is still proposed to be a hard problem, which this algorithm does not solve, and it is suggested that the full difficulty of the CP be used to develop a cryptosystem.

Chapter 5

Conclusion

It does look that the attacks on these braid group cryptosystems are effective. The important question to ask is why do these systems seem breakable. It may be that the keys that are being generated are insecure. It is strongly suspected that the parameters originally thought to generate strong keys do not. Perhaps we need more complex keys. There is a point however, where the keys are so long and complex, that they are no longer attractive to use or are even practical. It may be though that we just do not know how to generate keys that are secure at this time, regardless of the parameters we are using. The keys we generate may be susceptible to attacks such as the length-based, but that properly-generated keys are not.

The specific protocols we are using may be weak. The conjugacy search problem may be a hard problem, but no protocol uses the CSP directly. It is used as a trapdoor in some, but a hard trapdoor problem does not guarantee that the underlying problem is hard. Most of these attacks, except the direct attacks, such as the USS, are not able to crack the conjugacy search problem.

Work in the future could be to study the generation of the keys, what makes a strong key, and how to generate such a key. Strong protocols that are based on the CSP itself, or other problems that are believed to be hard, could be studied.

The braid groups are not the only non-commutative group that are attractive as a platform, and there are others that fit the Shpilrain criteria. There are also other problems that are conjectured to be hard that can be explored.

At this point, we are far from being able to say whether braid groups are a viable or even ideal platform for cryptosystems or not. There is still a lot of work to be done.

Bibliography

[1] Iris Anshel, Michael Anshel, and Dorian Goldfeld. An algebraic method for public-key cryptography. *Mathematical Research Letters*, 6(3/4):287–291, 1999.

[2] Iris Anshel, Michael Anshel, Benji Fisher, and Dorian Goldfeld. New key agreement protocols in braid group cryptography. In David Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 13–27. Springer, 2001. ISBN 3-540-41898-9. URL http://dblp.uni-trier.de/db/conf/ctrsa/ctrsa2001.html#AnshelAFG01.

[3] E. Artin. Theory of braids. *Ann. Math. (2)*, 48:101–126, 1947. ISSN 0003-486X; 1939-8980/e. doi: 10.2307/1969218.

[4] S. Bigelow. The Burau representation is not faithful for n = 5. *ArXiv Mathematics e-prints*, April 1999.

[5] S. J. Bigelow. Braid Groups are Linear. *ArXiv Mathematics e-prints*, May 2000.

[6] J. S. Birman, K. H. Ko, and S. J. Lee. The infimum, supremum and geodesic length of a braid conjugacy class. *ArXiv Mathematics e-prints*, March 2000.

[7] Jung Hee Cheon and Byungheup Jun. A polynomial time algorithm for the braid diffie-hellman conjugacy problem. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 212–225. Springer, 2003. ISBN 3-540-40674-3. URL http://dblp.uni-trier.de/db/conf/crypto/crypto2003.html#CheonJ03.

[8] Elsayed A Elrifai and HUGH R Morton. Algorithms for positive braids. *Quarterly Journal of Mathematics*, 45(180):479–498, 1994.

[9] N. Franco and J. Gonzalez-Meneses. Conjugacy problem for braid groups and Garside groups. *ArXiv Mathematics e-prints*, December 2001.

[10] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne. Length-based conjugacy search in the Braid group. *ArXiv Mathematics e-prints*, September 2002.

[11] V. Gebhardt. A New Approach to the Conjugacy Problem in Garside Groups. *ArXiv Mathematics e-prints*, June 2003.

[12] J. Gonzalez-Meneses. Improving an algorithm to solve Multiple Simultaneous Conjugacy Problems in braid groups. *ArXiv Mathematics e-prints*, December 2002.

[13] Dennis Hofheinz and Rainer Steinwandt. A practical attack on some braid group based cryptographic primitives. In *Public Key Cryptography*, pages 187–198, 2003.

[14] James Hughes. A linear algebraic attack on the aafg1 braid group cryptosystem. In Lynn Margaret Batten and Jennifer Seberry, editors, *ACISP*, volume 2384 of *Lecture Notes in Computer Science*, pages 176–189. Springer, 2002. ISBN 3-540-43861-0. URL http://dblp.uni-trier.de/db/conf/acisp/acisp2002.html#Hughes02.

[15] James Hughes and Allen Tannenbaum. Length-based attacks for certain group based encryption rewriting systems. *CoRR*, cs.CR/0306032, 2003.

[16] Ki Hyoung Ko, Sangjin Lee, Jung Hee Cheon, Jae Woo Han, Ju-Sung Kang, and Choonsik Park. New public-key cryptosystem using braid groups. In Mihir Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 166–183. Springer, 2000. ISBN 3-540-67907-3. URL http://dblp.uni-trier.de/db/conf/crypto/crypto2000.html#KoLCHKP00.

[17] Ki Hyoung Ko, Doo Ho Choi, Mi Sung Cho, and Jang Won Lee. New signature scheme using conjugacy problem. Technical report, 2002.

[18] Daan Krammer. The Braid Group B_4 is Linear. *Inventiones mathematicae*, 142(3): 451–486, 2000. ISSN 0020-9910. doi: 10.1007/s002220000088. URL http://dx.doi.org/10.1007/s002220000088.

[19] Sangjin Lee and Eonkyung Lee. Potential weaknesses of the commutator key agreement protocol based on braid groups. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 14–28. Springer, 2002. ISBN 3-540-43553-0. URL http://dblp.uni-trier.de/db/conf/eurocrypt/eurocrypt2002.html#LeeL02.

[20] W Magnus and A Peluso. On a Theorem of V.I. Arnold. *Comm. Pure Appl. Math.*, (22):683–692, 1969.

[21] A. Myasnikov, V. Shpilrain, and A. Ushakov. *Group-based Cryptography*. Advanced Courses in Mathematics - CRM Barcelona. Springer Basel AG, 2008. ISBN 9783764388263. URL http://books.google.ca/books?id=AbPZJf1wcKcC.

[22] Alex D. Myasnikov and Alexander Ushakov. Length based attack and braid groups: Cryptanalysis of anshel-anshel-goldfeld key exchange protocol. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 76–88. Springer, 2007. ISBN 978-3-540-71676-1. URL http://dblp.uni-trier.de/db/conf/pkc/pkc2007.html#MyasnikovU07.

[23] Mike Paterson and Alexander A. Razborov. The set of minimal braids is co-np-complete. *J. Algorithms*, 12(3):393–408, 1991. URL http://dblp.uni-trier.de/db/journals/jal/jal12.html#PatersonR91.

[24] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978. ISSN 0001-0782. doi: 10.1145/359340.359342. URL http://doi.acm.org/10.1145/359340.359342.

[25] V. Shpilrain. Assessing security of some group based cryptosystems. *ArXiv Mathematics e-prints*, November 2003.

[26] V. Shpilrain and A. Ushakov. Thompson's group and public key cryptography. *ArXiv Mathematics e-prints*, May 2005.

[27] Herv Sibert, Patrick Dehornoy, and Marc Girault. Entity authentication schemes using braid word reduction. *IACR Cryptology ePrint Archive*, 2002:187, 2002. URL http://dblp.uni-trier.de/db/journals/iacr/iacr2002.html#SibertDG02.

[28] V. Turaev. Faithful Linear Representations of the Braid Groups. *ArXiv Mathematics e-prints*, June 2000.