Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Magnus Ringerud

WalnutDSA: Another attempt at braid group cryptography

Master's thesis in Mathematical Sciences
Supervisor: Kristian Gjøsteen
May 2019

NTNU
Norwegian University of Science and Technology

Magnus Ringerud

WalnutDSA: Another attempt at braid group cryptography

Master's thesis in Mathematical Sciences
Supervisor: Kristian Gjøsteen
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Abstract

The main purpose of this thesis is to study the WalnutDSA digital signature scheme proposed for the American National Institute of Standards and Technology's (NIST) Post-Quantum Cryptography Standardization process. In order to understand the scheme, one chapter is devoted to develop the necessary theory for braid groups. We then briefly look into some of the older schemes involving braid groups, and why they are insecure. After presenting the Walnut scheme, we study attacks designed to break it, before giving comments which may explain why NIST chose not to include the scheme in the second round of the standardization process.

Sammendrag

Hovedformålet med denne oppgaven er å studere det digitale signatursystemet WalnutDSA, som ble foreslått til det amerikanske National Institute of Standards and Technology (NIST) sin "Post-Quantum Cryptography Standardization" prosess. For å forstå systemet er ett kapittel satt av til å utlede den nødvendige teorien for flettegrupper. Vi ser raskt på noen eldre kryptosystemer som involverer flettegrupper, og hvorfor de er usikre. Etter å ha presentert Walnut-systemet ser vi på forskjellige angrep konstruert for å knekke det, før vi gir noen kommentarer som kan forklare hvorfor NIST valgte å ikke inkludere systemet i runde to av standardiseringsprosessen.

Preface

This thesis was written under the supervision of Professor Kristian Gjøsteen, and marks the end of my time as a student for the degree of Master of Science in Mathematics at NTNU.
I thank Gjøsteen for his suggestions of different topics, and for guiding me after the topic was chosen. Without the weekly meetings pushing me forward, I am quite sure that writing this thesis would not have been as enjoyable as it has been. I found the topic at hand remarkably elegant, and I am glad I got the opportunity to study it.

I want to thank Linjeforeningen Delta and it's members, for giving me a place to grow, and making these past five years the best years of my life. Special thanks to my friends Didrik, Eiolf, Filip, Morten and Peter for being my mathematical sparring partners. Gratitudes to Eiolf Kaspersen and Ole Martin Kringlebotn for proofreading and feedback. Finally, I want to thank my family for always supporting me.

There is nothing groundbreaking in this thesis, but rather a compilation of the works of many researchers. When the topic was chosen, the system of study was still highly relevant, but since it did not make it to the second round of the NIST standardization process, few relevant papers were written after January 2019. Thus the ending might seem somewhat abrupt, and while some earlier sections could have been fleshed out in more detail, I felt that adding more content would only serve to add the sake of adding, and not improve the thesis in a meaningful way. Therefore I am quite satisfied with the end product.

Magnus Ringerud
Trondheim, May 2019

Table of Contents

Chapter 1

Introduction

The security of most of the public key cryptosystems being used today is based on discrete logarithm problems, or the problem of factoring large integers. These problems can be solved using Peter Shor's quantum algorithm [32] on a large enough quantum computer. People worry that such a quantum computer may one day be built.

To establish a standard for post-quantum cryptography, the National Institute of Standards and Technology (NIST) launched the "Post-Quantum Cryptography Standardization", and the call for proposals ended on November 30, 2017. The first round of testing ended January 30, 2019, and the second round is still in progress at the time of writing.

In Chapter 2 we give a short introduction to digital signature schemes, their various properties, different attacks against them, and various notions of security.

The main purpose of this thesis is to study the WalnutDSA digital signature scheme, proposed to NIST by SecureRF [4]. The scheme is based on braid groups - non-abelian groups which was first proposed for cryptography two decades ago - and we will study it in Chapter 5.

While braid groups are non-abelian, they possess a remarkable amount of structure. In order to understand and appreciate the mathematics of the scheme, we will study braid groups in Chapter 3. We then quickly look at some of the previous braids group constructions in Chapter 4. These systems have elegant descriptions and properties, but have all been proven insecure.

In Chapter 6 we study different attacks made against Walnut, and the consequences they had for the implementation and security of the scheme, before we give concluding remarks in Chapter 7.

Chapter 2

Digital signatures

2.1 Description

Suppose Alice wants to send a message to Bob, via some channel. An eavesdropper Eve has access to the channel, and can edit any message she sees, even to the extent of changing the message for one of her own. Alice wants her message to arrive without modification, and if it has been tampered, Bob should notice. Since many people might want to send messages to each other, a symmetric key approach is not favourable due to the need for storing keys for each correspondent. There are multiple solutions for this problem, but we will only give a description of *digital signatures*.

Definition 2.1 (Digital signature scheme)**.** A digital signature scheme is a public key cryptosystem, which consists of three algorithms $\mathcal{K}, \mathcal{S}, \mathcal{V}$:

- The *key generation* algorithm $\mathcal{K}$, which takes as input a security parameter k, and returns a *signing key* sk and a *verification key* vk. For every key pair there is an associated message set denoted $\mathcal{M}_{sk}$ and $\mathcal{M}_{vk}$. Note that $\mathcal{K}$ must be a probabilistic algorithm.
- The *signing* algorithm $\mathcal{S}$, which takes as input a signing key sk and a message $m \in \mathcal{M}_{sk}$, and outputs a signature σ. This algorithm may be probabilistic.
- The *verification* algorithm, which takes as input a verification key vk, a message $m \in \mathcal{M}_{vk}$ and a signature σ, and outputs either 0 or 1. The algorithm need in general not be probabilistic.

We require that for any key pair (vk, sk) output by $\mathcal{K}$ and any message m, we have

$$\Pr[\mathcal{V}(vk, m, \mathcal{S}(sk, m)) = 1] = 1. \tag{2.1}$$

When the output of $\mathcal{V}$ is 1, we have a *valid* signature. Otherwise, the signature is invalid.

We often call the signing key sk for the private key, and the verification key vk for the public key. We define a *forgery* as a valid signature created without using the private key.

When using a digital signature scheme to authenticate a message, one sends both m and σ across the channel, and as such anyone with access to said channel will be able to see both. From the equations above, it is clear that anyone with access to the channel who knows the parameters of the scheme and the public key of the signer, can verify a signature.

2.1.1 Properties of digital signatures

Following the motivational part at the beginning of this chapter, there are additional benefits to gain from using digital signatures. Apart from being able to move away from paper documents, applying a digital signature to communications has the following benefits:

- *Authenticity:* A valid signature on a message sent by Alice shows that the message was in fact sent by Alice. This is important for example when asking a bank to transfer funds; you do not want anyone other than yourself to be able to withdraw money from your account.
- *Integrity:* Integrity means confidence in that the message has not been tampered with. While encryption might hide the content of a message, it could be possible to change an encrypted message without knowing the contents or understanding what change you made. If a message is digitally signed, any change to the message after the signature is generated will make the signature invalid, and hence any tampering will be noticed.
- *Non-repudiation:* If someone has signed a document, they cannot at a later time deny having signed it.

2.2 Security

To specify what it means for a signature scheme to be secure, we follow the description given in [20]. We describe different kinds of attacks and notions of security.

2.2.1 Description of attacks

We first and foremost distinguish between two kinds of attacks:

- *Key-only attacks*, where the adversary knows nothing but the public key of the signer.
- *Message attacks*, where the adversary can examine signatures of either known or specified messages before attempting to break the scheme.

We can specify four different message attacks, which are characterized by how the list of messages the adversary can see are chosen. Let A denote the honest signer whose scheme the adversary is trying to break.

- *Known-message attack:* The adversary is given access to signatures of a known set of messages $m_1, \ldots, m_l$. The messages are known to the adversary, but not chosen by him.

- *Generic chosen-message attack:* The adversary can obtain valid signatures from A for a chosen list of messages $m_1, \ldots, m_l$. The messages are chosen by the adversary before he can see A's public key. Hence this attack is independent of the public key, and is generic. The attack is non-adaptive, as the entire list of messages is created before any signature is seen.
- *Directed chosen-message attack:* Like in the generic attack, the adversary can obtain valid signatures from A for a chosen list of messages $m_1, \ldots, m_l$, but this time he can create the list of messages after seeing the public key. The attack is thus directed against a particular A. The attack is still non-adaptive, as the list is created before any signatures are seen.
- *Adaptive chosen-message attack:* The adversary can use A as an "oracle", not only can he request signatures of chosen messages that depend on A's public key, he may also request signatures of messages that depend on the previously obtained signatures.

The above attacks are listed in order of increasing severity. To see that the adaptive chosen-message attack is a natural one, consider a bank who must sign more or less random documents created by someone else.

Since the adaptive chosen-message attack is the most powerful attack, it is this attack we want to make sure our scheme can withstand. When speaking of it we often drop the "adaptive" part, and only refer to it as the chosen-message attack, or CMA for short.

2.2.2 Security notions

To "break" a scheme could mean different things for different schemes. In general, we might say that an adversary has broken our scheme if he is able to do any of the following with non-negligible probability.

- A *total break:* Recover A's signing key.
- *Universal forgery:* Forge a signature for any message.
- *Selective forgery:* Forge a signature for a message of the adversary's choice.
- *Existential forgery:* Forge a signature for at least one message.

Note that a forgery has to be a new signature, simply claiming that a signature received from A is a forgery is not the same as being able to create a new one. Note also that the above list is not exhaustive, there are other ways of "breaking" a signature scheme. The severity of the "breaks" are listed in decreasing severity, and the easiest task the adversary can undertake is to find an existential forgery. We say that a scheme is *totally breakable, universally forgeable, selectively forgeable* or *existentially forgeable* if it is breakable in one of the above senses.

To summarize, we say that the strongest notion of security for a digital signature scheme is security against existential forgery under an adaptive chosen-message attack, which we will refer to as EUF-CMA (existentially unforgeable under chosen-message attack) security. To formalize, we play a *game* which is illustrated in Figure 2.1.

For any polynomial time adversary $\mathcal{A}$ and a signer $\mathcal{S}$, the process proceeds as follows:

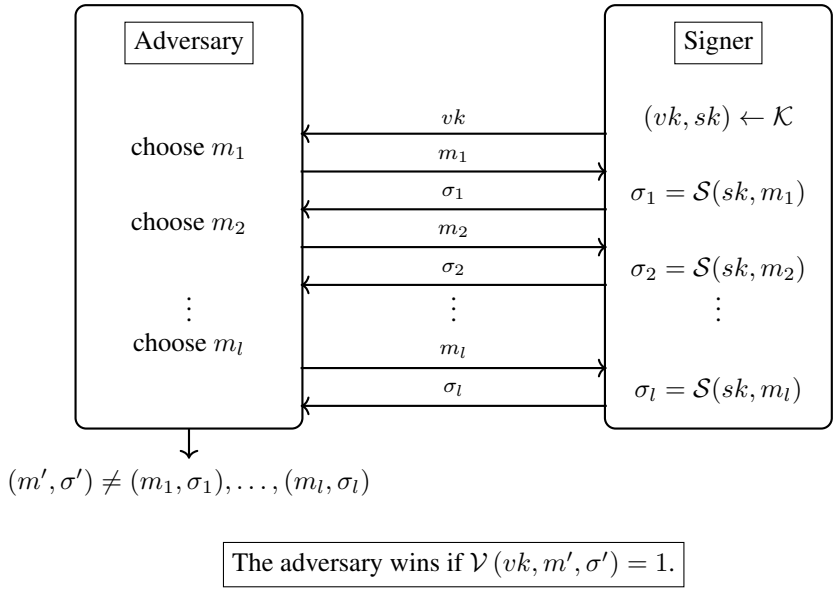


Figure 2.1: Illustration of CMA-game.

1. The signer provides his public key vk to the adversary.
2. The adversary chooses a message m_1 to receive a valid signature for, and sends a signature query.
3. The signer computes the signature σ_1, and sends it back.
4. After receiving σ_1, the adversary chooses a message m_2 and queries it.
5. The process continues until the adversary has obtained l signatures, for some polynomial time bound l.
6. The adversary now attempts to create a forgery, and outputs (m', σ') where

$$(m', \sigma') \neq (m_1, \sigma_1), \ldots, (m_l, \sigma_l).$$

He wins if

$$\mathcal{V}(vk, m', \sigma') = 1. \tag{2.2}$$

In some cases, we may also require that $m' \neq m_1, \ldots, m_l$.

We denote the probability of the adversary $\mathcal{A}$ winning the game by $\Pr[\mathcal{A}]$, and a scheme is considered broken if this probability is not negligible.

Chapter 3

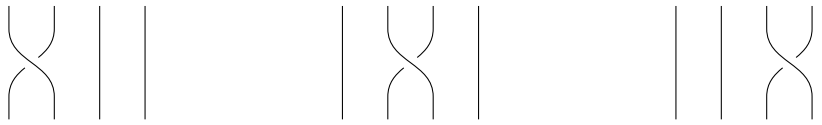
Braid groups

3.1 Description and properties

An N*-braid* can be viewed geometrically as a collection of N strands crossing each other a finite number of times. The *braid group on* N *strands* $\mathcal{B}_N$ is the collection of all such N-braids, with the group operation being composition of braids. The identity element is the trivial braid e with no crossings, and the group itself is infinite and non-abelian for $N \geq 2$. We can visualize the braids in three dimensions as follows: consider a braid b and let every strand have a starting point $(x, y, z) = (i, 0, 1)$ for $1 \leq i \leq N$, and an end point $(\lambda(i), 0, 0)$.

There are several ways to formalize this, one of them being through the concept of homotopy from topology. With this idea, we can view the braid group as the set of equivalence classes of N-braids, in which we can think of two braids as being equivalent if they become the same braid when we "pull" the strands.

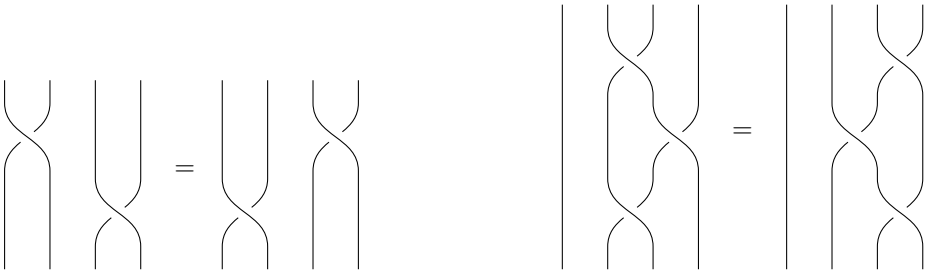
More suited for our purpose is the algebraic representation of the group given by Emil Artin [6]. He showed that the braid group $\mathcal{B}_N$, and the free group on $N - 1$ generators with two given relations, are isomorphic. We will work in the Artin-representation, but it is useful to think of the elements as elements of the braid group. The representation uses the generators b_i for $1 \leq i \leq N - 1$, which represents the braids where strand i crosses over strand $i + 1$. We refer to these as the Artin generators, and the inverse of a generator braid is the braid b_i^{-1} where strand i crosses under strand $i + 1$. To be precise, an N-braid



(a) b_1 **(b)** b_2 **(c)** b_3

Figure 3.1: The three generators of $\mathcal{B}_4$.

is an equivalence class $w \in \mathcal{B}_N$, while a representation of the braid in the generators is called a *braid word*. The terminology is often clear from the context, and we will freely



(a) $b_1 b_3 = b_3 b_1$.

(b) $b_2 b_3 b_2 = b_3 b_2 b_3$.

Figure 3.2: Illustration of the braid relations.

interchange the two terms.

Theorem 3.1 (Artin)**.** The braid group on N strands can be written as

$$\mathcal{B}_N = \langle b_1, b_2, \ldots, b_{N-1} \rangle \,, \tag{3.1}$$

with relations

$$b_i b_j = b_j b_i \text{ for } |i-j| > 1 \tag{3.2}$$

$$b_i b_j b_i = b_j b_i b_j \text{ for } |i-j| = 1. \tag{3.3}$$

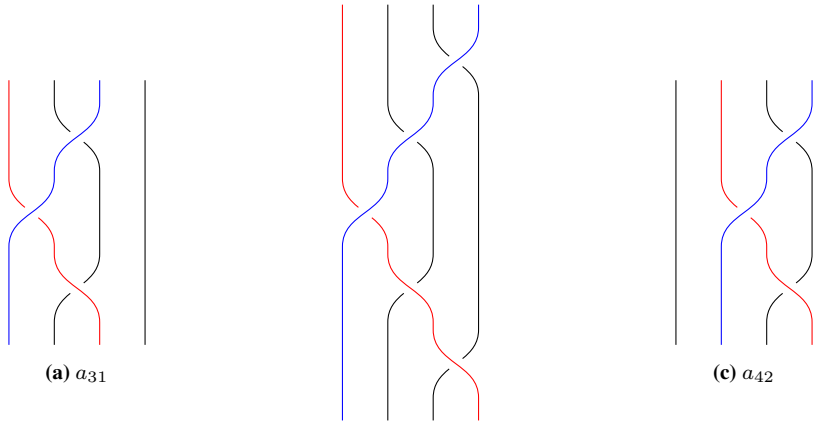
Thus every N-braid can be written as a word in these generators and their inverses.

The first relation has a simple geometric interpretation: it says that braids where no common strands cross are commutative. The second, while more troublesome to imagine, can be seen from an illustration of the resulting braids. Notice that in Figure 3.2b, all the crossings in the two braids are the same, they just happen at different heights.

3.2 Normal form of braids

As in many areas of mathematics, we would like to uniquely decompose our group elements into factors. It turns out that there are multiple such decompositions for braid groups, and we call them *normal forms*. The best known examples are the Garside normal form and the Birman-Ko-Lee normal form (the latter will be abbreviated as the BKL normal form). While the BKL normal form is the representation used for braids in the WalnutDSA cryptosystem which we will study later (Chapter 5), we will not go into details here. For the full description of the BKL normal form, see the description given by Birman, Ko and Lee in [9].

An interesting difference between the two is the use of a different set of generators in the BKL normal form. Given that a braid has infinitely many representations ($w = wb_i b_i^{-1}$ for all braids w), it is not surprising that there exists multiple generating sets, but they are still worth studying. Like Artin's generators b_i, which swap strand i and $i+1$, the new generators swap one pair of strands, but they need not be adjacent. With the notation we



(a) a_{31} **(b)** a_{41} **(c)** a_{42}

Figure 3.3: Examples of band generators.

have used so far, where b_i swaps strand i over strand $i+1$, we define the *band generator* a_{ts} for $1 \leq s < t \leq N$ as

$$a_{ts} := (b_{t-1}^{-1} b_{t-2}^{-1} \cdots b_{s+1}^{-1}) b_s^{-1} (b_{s+1} b_{s+2} \cdots b_{t-1}). \tag{3.4}$$

This is slightly different from the presentation given in [9], as their generator b_i passes strand i under strand $i+1$. The braid a_{ts} swaps strand t and s, with the convention that the strands being crossed pass in front of the others. Note that the Artin generators form a subset of the band generators, as $b_1^{-1} = a_{21}, b_2^{-1} = a_{32}$ and so forth (we could change the inverses here by letting the crossings happen "behind" all the other strands).

Many of the properties are analogous to the Garside normal form, and since Garside presented the first treatment of the subject, we will start by following his presentation [17]. Garside's results was later improved by Thurston [16], and E. A. Elrifai and H. R. Morton [15].

3.2.1 Positive braids

We start off by introducing *positive braids*. We say that a crossing is positive if strand i crosses over strand $i+1$. In particular, all the generators b_i are positive. We generalize this to braids by defining a positive braid as one that can be written as a product of the generators b_i, and we denote the set of positive braids as $\mathcal{B}_N^+$. This has the structure of a monoid, since we inherit all the properties except the existence of inverses from the group.

We can use the positive braids to define a partial order on the braid group: for $A, B \in \mathcal{B}_N$ we say that $A \leq B \Leftrightarrow B = AC$ for some $C \in \mathcal{B}_N^+$. We thus have the equivalent statements

$$e \leq B \Leftrightarrow B \in \mathcal{B}_N^+.$$

One particularly useful positive braid is the *fundamental braid*:

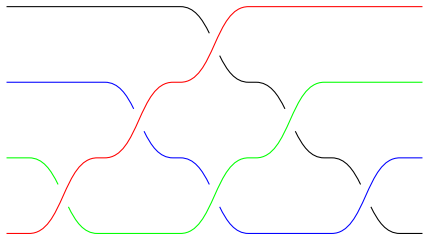


Figure 3.4: The fundamental braid $\Delta_4 = (b_1b_2b_3)(b_1b_2)b_1$.

Definition 3.2. The fundamental braid Δ_N of $\mathcal{B}_N$ is the braid

$$\Delta_N = (b_1b_2 \dots b_{N-1})(b_1 \dots b_{N-2}) \cdots (b_1b_2)b_1. \tag{3.5}$$

Geometrically, the fundamental braid is the braid where every pair of strands cross exactly once, and every crossing is positive. If we let

$$\Pi_r = b_1b_2 \dots b_r, \tag{3.6}$$

then we can write the fundamental braid as

$$\Delta_N = \Pi_{N-1}\Pi_{N-2} \cdots \Pi_1 = \Pi_{N-1}\Delta_{N-1}. \tag{3.7}$$

In particular we have $\Delta_1 = \Pi_0 = e$, the trivial braid, represented as an empty string which we call the *nullstring*. Where there is no ambiguity, we will often write $\Delta = \Delta_N$.

The fundamental braid has a number of properties, and we will derive some of them in the following pages.

Lemma 3.3. For $1 < s \leq t < N$ we have

$$b_s\Pi_t = \Pi_t b_{s-1}. \tag{3.8}$$

Proof. We use the braid relations

$$\begin{aligned} b_s\Pi_t &= b_s(b_1b_2 \dots b_t) \\ &= (b_1 \dots b_{s-2})b_sb_{s-1}b_s(b_{s+1} \dots b_t) \\ &= (b_1 \dots b_{s-2})b_{s-1}b_sb_{s-1}(b_{s+1} \dots b_t) \\ &= (b_1 \dots b_{s-2})b_{s-1}b_s(b_{s+1} \dots b_t)b_{s-1} \\ &= \Pi_t b_{s-1}. \end{aligned}$$

□

Lemma 3.4. In $\mathcal{B}_N$ we have

(i) $b_1\Delta_t = \Delta_t b_{t-1}$ for $t = 2, \dots, N$.

For $j = 1, 2 \dots, N-1$ we have

(ii) $b_j\Delta = \Delta b_{N-j}$

(iii) $b_j^{-1}\Delta^{-1} = \Delta^{-1}(b_{N-j})^{-1}$

(iv) $b_j^{-1}\Delta = \Delta(b_{N-j})^{-1}$

(v) $b_j\Delta^{-1} = \Delta^{-1}b_{N-j}$.

Proof.

(i) For $t = 2$,

$$b_1\Delta_2 = b_1b_1 = \Delta_2 b_1 \quad \text{as required.}$$

For $2 < t \leq N$,

$$\begin{aligned} b_1\Delta_t &= b_1\Pi_{t-1}(b_1 \cdots b_{t-2})\Delta_{t-2} \\ &= b_1(b_2 \cdots b_{t-1})\Pi_{t-1}\Delta_{t-2} && \text{by Lemma 3.3} \\ &= \Pi_{t-1}(\Pi_{t-2}b_{t-1})\Delta_{t-2} \\ &= \Pi_{t-1}\Pi_{t-2}\Delta_{t-2}b_{t-1} && \text{by (3.2)} \\ &= \Delta_t b_{t-1}. \end{aligned}$$

Notice that the direct proof for $t = 2$ is necessary, as we cannot apply Lemma 3.3 in this situation.

(ii) For $j = 1$ we get

$$b_1\Delta = \Delta b_{N-1} \quad \text{by (i).}$$

For $1 < j \leq N-1$ we compute

$$\begin{aligned} b_j\Delta &= b_j\Pi_{N-1}\Pi_{N-2}\cdots\Pi_{N-j+1}\Delta_{N-j+1} \\ &= \Pi_{N-1}\Pi_{N-2}\cdots\Pi_{N-j+1}b_1\Delta_{N-j+1} && \text{by Lemma 3.3} \\ &= \Pi_{N-1}\Pi_{N-2}\cdots\Pi_{N-j+1}\Delta_{N-j+1}b_{N-j} && \text{by (i)} \\ &= \Delta b_{N-j}. \end{aligned}$$

(iii)

$$b_j^{-1}\Delta^{-1} = (\Delta b_j)^{-1} = (b_{N-j}\Delta)^{-1} = \Delta^{-1}(b_{N-j})^{-1} \quad \text{by (ii).}$$

(iv) Starting from the point above we have $b_k^{-1}\Delta^{-1} = \Delta^{-1}(b_{N-k})^{-1}$. We multiply with Δ from the left and the right to get

$$\Delta b_k^{-1} = (b_{N-k})^{-1}\Delta.$$

Setting $j = N - k$ gives the desired result.

(v)
$$b_j\Delta^{-1} = (\Delta b_j^{-1})^{-1} = ((b_{N-j})^{-1}\Delta)^{-1} = \Delta^{-1}b_{N-j} \quad \text{by (iv).}$$

□

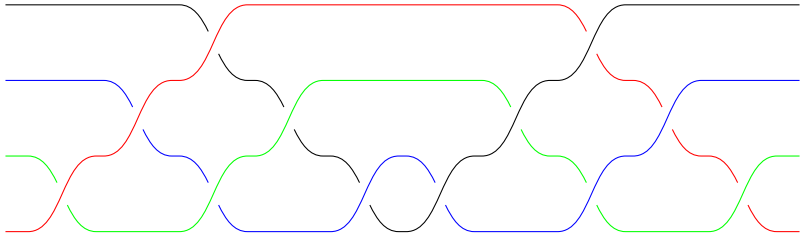
In $\mathcal{B}_N$, let τ be the map $\tau(b_i) = b_{N-i}$. We call this the *reflection map*, and we can extend it to an automorphism of the group. We can also write this as conjugation by Δ;

$$\begin{aligned} \tau(b_i) &= \Delta b_i \Delta^{-1} = \Delta\Delta^{-1}b_{N-i} = b_{N-i} \\ \tau(b_i^{-1}) &= \Delta b_i^{-1}\Delta^{-1} = \Delta\Delta^{-1}b_{N-i}^{-1} = b_{N-i}^{-1} \end{aligned} \tag{3.9}$$

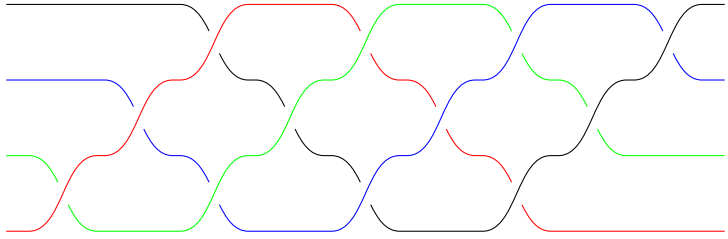
by Lemma 3.4. Notice that

$$\begin{aligned} \tau^2(b_i) &= \tau(\tau(b_i)) = \tau(b_{N-i}) = b_{N-(N-i)} = b_i \\ \tau^2(b_i^{-1}) &= \tau(\tau(b_i^{-1})) = \tau(b_{N-i}^{-1}) = b_{N-(N-i)}^{-1} = b_i^{-1}, \end{aligned}$$

which means that $\tau^2(B) = B$ for any word B. This also means that Δ^2 is in the centre of the group, and we often call this the *full-twist braid*, because we can visualize it as grabbing the ends of the identity braid e and rotating them $360°$, see Figure 3.5.



(a) The representation $b_1b_2b_3b_1b_2b_1b_1b_2b_3b_1b_2b_1$.



(b) The more visually pleasing representation $b_1b_2b_3b_1b_2b_3b_1b_2b_3b_1b_2b_3$.

Figure 3.5: The full-twist braid Δ^2.

We get the following result.

Theorem 3.5. For any word $B \in \mathcal{B}_N$ and $m \in \mathbb{Z}$ we have

$$B\Delta^{2m} = \Delta^{2m}B, \quad B\Delta^{2m+1} = \Delta^{2m+1}\tau(B).$$

Proof. Repeated use of Lemma 3.4 imply that for any word B we have

$$B\Delta = \Delta\tau(B)$$
$$B\Delta^{-1} = \Delta^{-1}\tau(B),$$

which combined with the observation above gives the result. □

Let $\operatorname{rev} B$ denote the *reverse* of the word B, meaning that if $B = x_1x_2\dots x_m$ is a word where x_i is either a generator or its inverse, $\operatorname{rev} B = x_m x_{m-1}\dots x_1$.

Lemma 3.6. In $\mathcal{B}_N$,

$$\tau(\Delta) = \Delta, \quad \text{and} \quad \operatorname{rev}\Delta = \Delta.$$

Proof. By Theorem 3.5 we have

$$\tau(\Delta)\Delta = \Delta\tau(\tau(\Delta)) = \Delta^2 \implies \tau(\Delta) = \Delta.$$

For the second point we proceed by induction. Obviously we have

$$\operatorname{rev}\Delta_2 = \operatorname{rev} b_1 = b_1 = \Delta_2,$$

so the statement is true for $N = 2$. Assume now that it holds for some r. Then

$$\begin{aligned}\operatorname{rev}\Delta_{r+1} &= \operatorname{rev}((b_1\dots,b_r)\Delta_r)\\ &= \operatorname{rev}\Delta_r \operatorname{rev}(b_1\dots,b_r)\\ &= \Delta_r(b_r,\dots,b_1) \quad \text{by the induction hypothesis}\\ &= \Pi_{r-1}\Pi_{r-2}\cdots\Pi_1(b_r,\dots,b_1).\end{aligned}$$

Now b_r commutes with $\Pi_{r-2}, \Pi_{r-3}, \dots,$ b_{r-1} commutes with $\Pi_{r-3}, \Pi_{r-4}, \dots$ and so forth. Thus we have

$$\begin{aligned}\operatorname{rev}\Delta_{r+1} &= \Pi_{r-1}b_r\Pi_{r-2}b_{r-1}\cdots\Pi_1 b_2 b_1\\ &= \Pi_r\Pi_{r-1}\cdots\Pi_2 b_1\\ &= \Delta_{r+1}.\end{aligned}$$

□

We require one more lemma before we can start working towards the normal form we sought out to obtain.

Lemma 3.7. There exists positive words X_t, Y_t such that for $1 \le t < N$ we have

$$b_t X_t = \Delta = Y_t b_t. \tag{3.10}$$

Proof. To start off, note that since

$$\Delta = \Pi_{N-1}\cdots\Pi_1 = Y_1 b_1,$$

the first case is trivial. If we let $f(b_2, \ldots, b_t)$ denote any positive word in the generators $b_2, \ldots, b_t$, by Lemma 3.3 we get that

$$\Pi_t f(b_1, \ldots, b_{t-1}) = f(b_2, \ldots, b_t)\Pi_t.$$

Let $b_t \in \{b_2, \ldots, b_{N-1}\}$. If we let $\Pi_{t-1}\Pi_{t-2}\cdots\Pi_1 = f(b_1, \ldots, b_{t-1})$, we have

$$\begin{aligned}\Delta &= \Pi_{N-1}\cdots\Pi_{t+1}\Pi_t f(b_1, \ldots, b_{t-1}) \\ &= \Pi_{N-1}\cdots\Pi_{t+1} f(b_2, \ldots, b_t)\Pi_t \\ &= \Pi_{N-1}\cdots\Pi_{t+1} f(b_2, \ldots, b_t)(b_1\cdots b_{t-1})b_t \\ &= Y_t b_t.\end{aligned}$$

This shows that Y_t exists for $t = 1, 2, \ldots, N-1$. To finish the proof, let $X_t = \operatorname{rev} Y_t$ and see that

$$\Delta = \operatorname{rev}\Delta = \operatorname{rev}(Y_t b_t) = \operatorname{rev}(b_t)\operatorname{rev}(Y_t) = b_t X_t.$$

□

3.2.2 Permutation braids

A braid P which satisfies $e \leq P \leq \Delta$ is called a *permutation braid*. The name reflects the fact that there is a bijection from the set of permutation braids in $\mathcal{B}_N$, to the set S_N. The bijection can be constructed by sending i to $\lambda(i)$ (the ending position of the strand beginning at i), and letting each crossing be positive. Geometrically, a permutation braid is a braid where any pair of strands cross positively at most once. We denote the set of permutation braids by S_N^+.

For a permutation braid P we can define a *starting set* $S(P)$ and a *finishing set* $F(P)$ as follows:

$$\begin{aligned}S(P) &= \{i \mid P = b_i P' \text{ for some } P' \in \mathcal{B}_N^+\} \\ F(P) &= \{i \mid P = P' b_i \text{ for some } P' \in \mathcal{B}_N^+\}.\end{aligned}$$

Note that we can actually define these sets for any braid, but that the sets then may be empty. The starting set is the set of indices of the generators which can start a representation of a braid, and the finishing set is similarly defined. Notice that we can write $F(P) = S(\operatorname{rev} P)$. As a concrete example, we look at Δ_4:

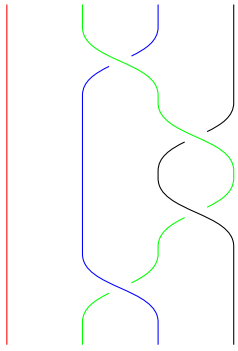
$$\begin{aligned}\Delta_4 &= b_1 b_2 b_3 b_1 b_2 b_1 \\ &= b_1 b_2 b_3 b_2 b_1 b_2 \\ &= b_1 b_3 b_2 b_3 b_1 b_2 \\ &= b_3 b_1 b_2 b_3 b_1 b_2 \\ &= b_3 b_2 b_1 b_2 b_3 b_2 \qquad (3.11) \\ &= b_3 b_2 b_1 b_3 b_2 b_3 \\ &= b_2 b_3 b_2 b_1 b_2 b_3. \qquad (3.12)\end{aligned}$$

This calculation shows that $S(\Delta_4) = F(\Delta_4) = \{1, 2, 3\}$. Notice that by using Lemma 3.7, we can generalize this to get $S(\Delta) = F(\Delta) = \{1, 2, \ldots, N-1\}$.

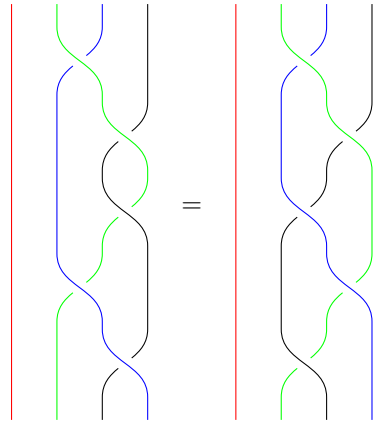
3.2.3 Decomposition into normal form

For a positive braid B, we may decompose it into a product $B = P_1 P_2 \cdots P_k$ of permutation braids by letting each P_i be the longest possible sequence of generators that still form a permutation braid. Note that every generator is a permutation braid by Lemma 3.7, so we can always create a "trivial" decomposition where every generator forms its own permutation braid. We say that a decomposition $B = P_1 P_2 \cdots P_k$ into permutation braids is *left-weighted* if $S(P_{i+1}) \subseteq F(P_i)$. This means that for any $j \in S(P_{i+1})$, if we "move" b_j from P_{i+1} to P_i, we can write the new braid P_i' as $P_i' = P_i b_j = P' b_j b_j$ for some positive braid P'. This new braid is no longer a permutation braid, since the strands in position j and $j+1$ at the end of P' now cross twice.

As an example we look at the braid $(b_2 b_3)(b_3 b_2) \in \mathcal{B}_4$ in Figure 3.6a.
Equations (3.12) and (3.11) show that $P_1 = b_2 b_3 \leq \Delta_4$ and $P_2 = b_3 b_2 \leq \Delta_4$, hence they



(a) The left-weighted braid $(b_2 b_3)(b_3 b_2)$.



(b) The *not* left-weighted braid $(b_2 b_3)(b_3 b_2 b_3)$, and its left-weighted form $(b_2 b_3 b_2)(b_3 b_2)$.

are both permutation braids. We cannot apply the braid relations to change these braids in any way, and hence we have $S(P_2) = F(P_1) = \{3\}$, and the braid is left-weighted.

For a braid that is not left-weighted, look at $(b_2 b_3)(b_3 b_2 b_3)$ in Figure 3.6b. For a geometric argument, we see that the last crossing is between strands 3 and 4, and that both of them lie "on top" of strand 2, and thus we can "move" the crossing to the middle of the braid. More formally, notice that we can write $P_2 = b_3 b_2 b_3 = b_2 b_3 b_2$, so $S(P_2) = \{2, 3\}$. Meanwhile we have that for $P_1 = b_2 b_3$ we have $F(P_1) = \{3\}$, and as such $S(P_2) \nsubseteq F(P_1)$. Finally, doing a straightforward computation yields

$$(b_2 b_3)(b_3 b_2 b_3) = (b_2 b_3)(b_2 b_3 b_2) = (b_2 b_3 b_2)(b_3 b_2),$$

where all the terms are permutation braids by the equations leading up to (3.12). We will later generalize the last computation, and show that when moving a generator like this we will always get new permutation braids.

The next lemma is useful when determining starting sets.

Lemma 3.8. Let $A \in S_N^+$ have permutation σ. The following are equivalent:

(i) $i \in S(A)$,

(ii) strands i and $i+1$ cross in A,

(iii) $\sigma(i+1) < \sigma(i)$.

Proof. If $A = b_i A'$, then (i) implies (ii) since the strands cross in b_i. Point (ii) and (iii) are equivalent since the strands cross positively at most once. If we have (ii), we can draw a diagram of the permutation in which this crossing is the first to happen. Constructing the braid from this diagram gives a word starting with b_i, which implies (i). □

Lemma 3.9. Let $A \in S_N^+$. Then $b_i A \in S_N^+$ if and only if $i \notin S(A)$.

Proof. Assume that $b_i A \in S_N^+$. If $i \in S(A)$, then we can write $b_i A = b_i b_i A'$, which is not a permutation braid, contradicting the assumption, and thus $i \notin S(A)$. For the other direction, note by the previous lemma that if $i \notin S(A)$, strands i and $i+1$ do not cross in A, which means that they cross once in $b_i A$. Every other pair of strands cross at most once, and hence $b_i A \in S_N^+$. □

The above lemmas are important for the following reason: If in our decomposition we encounter $P_i P_{i+1}$ where $S(P_{i+1}) \nsubseteq F(P_i)$, we can find a $j \in S(P_{i+1}) \setminus F(P_i)$, and write $P_i P_{i+1} = P_i b_j b_j^{-1} P_{i+1}$. Let $C_i = P_i b_j$ and $C_{i+1} = b_j^{-1} P_{i+1}$. Since $P_i, P_{i+1} \in S_N^+$, we can write

$$\begin{aligned} \Delta &= P_{i+1} B \\ \implies b_j^{-1}\Delta &= C_{i+1} B \\ \implies \Delta &= C_{i+1} B b_{N-j}, \end{aligned}$$

showing that $C_{i+1} \in S_N^+$. For C_i we use the lemmas:

$$j \notin F(P_i) \implies j \notin S(\operatorname{rev} P_i) \implies b_j \operatorname{rev} P_i \in S_N^+,$$

which again means that

$$\begin{aligned} \Delta &= b_j \operatorname{rev} P_i Q \\ \implies \operatorname{rev}\Delta &= \operatorname{rev}(b_j \operatorname{rev} P_i Q) \\ \implies \Delta &= \operatorname{rev} Q P_i b_j \\ \implies (\operatorname{rev} Q)^{-1}\Delta &= P_i b_j \\ \implies \Delta &= P_i b_j \tau(\operatorname{rev} Q), \end{aligned}$$

showing that $P_i b_j = C_i \in S_N^+$.

Due to a rather lengthy proof, we refer to Lemma 2.2 in [15] for the proof of the next lemma.

Lemma 3.10. Every $P \geq e$ has a unique left-weighted decomposition $P = P_1 A_1$ with $P_1 \in S_N^+$. Every other positive factorisation $P = AB$ with $A \in S_N^+$ satisfies $P_1 = AQ$ for some $Q \geq e$.

Thus we can say that the permutation braid P_1 has maximal length, since for any other decomposition $P = AB$ with $A \in S_N^+$, we have $A \leq P_1$. The lemma implies the following corollary, which will be crucial for the proof of the main result.

Corollary 3.11. Let P have a left-weighted factorisation $P = P_1 A_1$, with $P_1 \in S_N^+$. Then $S(P) = S(P_1)$.

Proof. Clearly $S(P_1) \subseteq S(P)$. Let $i \in S(P)$. Then $P = b_i B$ with $B \geq e$, and by Lemma 3.10 we then have $P_1 = b_i Q$ for some $Q \geq e$, and hence $i \in S(P_1)$. □

We now state the main result.

Theorem 3.12 (Garside normal form)**.** Any word W in $\mathcal{B}_N$ can be written uniquely as

$$W = \Delta^r P_1 P_2 \cdots P_k, \tag{3.13}$$

where $r \in \mathbb{Z}$, $P_i \in S_N^+$, and $P_i P_{i+1}$ is left-weighted for $1 \leq i \leq k-1$. We call r the *infimum* of W, and k the *canonical length* of W.

Proof. We can write W as

$$W = W_1(x_1)^{-1} W_2(x_2)^{-1} \cdots (x_s)^{-1} W_{s+1},$$

where every W_j is a positive word of length ≥ 0, and the x_j are generators. By Lemma 3.7, for each x_j there exists a positive word X_j such that $x_j X_j = \Delta$, which implies $(x_j)^{-1} = X_j \Delta^{-1}$, and thus we can write

$$W = W_1 X_1 \Delta^{-1} W_2 X_2 \Delta^{-1} \cdots W_s X_s \Delta^{-1} W_{s+1}.$$

By using Theorem 3.5 to move any appearance of Δ^{-1} to the left, we get $W = \Delta^{-s} P$, where P is a positive word.

We now want to express P as a left-weighted decomposition. Write $P = P_1 Q_1$, where P_1 is the longest sequence of generators that form a permutation braid. If $S(Q_1) \subseteq F(P_1)$, i.e $P = P_1 Q_1$ is a left-weighted decomposition, use the same technique to write $Q_1 = P_2 Q_2$. If not, simply move generators from Q_1 to P_1 by writing $P_1 Q_1 = P_1 b_j b_j^{-1} Q_1 = C_1 Q'$. By a previous argument we know that C_1 is still a permutation braid. By repeating this process, we can make sure that $P = P_1 Q_1$ is left-weighted. Remember that by Lemma 3.10, this decomposition is unique. We now proceed by induction. Assume that $P = P_1 \cdots P_i Q_i$ is left-weighted. If we use the method above to decompose $Q_i = P_{i+1} Q_{i+1}$ into a left-weighted decomposition, we have $S(P_{i+1}) = S(Q_i) \subseteq F(P_i)$ by Corollary 3.11, and hence the decomposition $P = P_1 P_2 \cdots P_i P_{i+1} Q_{i+1}$ is left-weighted and unique. Since P has finite length, the process must end, and by induction we have reached a left-weighted form $P = P_1 \cdots P_k$. This is called the *left-canonical form* of P.

Now all that remains is to show uniqueness. Assume that

$$\Delta^r P = \Delta^s Q,$$

where P, Q are shorthand for left-weighted decompositions. We assume that $\Delta \not\leq Q$, as then we would define $m = s + 1$ and look at the expression $\Delta^m Q'$ instead. Suppose that $r > s$, and let $a = r - s$. We then get that

$$\Delta^r P = \Delta^s Q \implies \Delta^a P = Q,$$

which means that $\Delta \leq Q$, contrary to our assumption. By a mirrored argument we get $r = s$, which in turn implies $P = Q$. By Lemma 3.10 these must have the same decomposition, and hence we have shown uniqueness and the proof is complete. □

We finish this section with the algorithm of Elrifai and Morton for computing the normal form [15]. First assume that we have written $P = \Delta^r P'$ for some P'. Assume that P' is factored into permutation braids $P' = P_1 P_2 \cdots P_k$. Again, this can always be done by taking the naive approach and letting each b_i form a permutation braid. Compute the sets $F(P_i)$ and $S(P_i)$. If $S(P_{i+1}) \subseteq F(P_i)$ for each i, then the decomposition is left-weighted. If not we can find a $j \in S(P_{i+1}) \setminus F(P_i)$, and write $P_i P_{i+1} = P_i b_j b_j^{-1} P_{i+1} = C_i C_{i+1}$. By the discussion following Lemma 3.9, both C_i and C_{i+1} are still permutation braids, and we use them to replace $P_i P_{i+1}$ in the decomposition.

Because $S(P_i)$ might not equal $S(P_i b_j)$, we cannot simply move to the next permutation braid, but must start again by finding the first i where $S(P_{i+1}) \subseteq F(P_i)$. However, because the permutation braids will increase in length and there is a finite set of letters in a braid word, the process must terminate.

If we let P be a word of length $|P|$ in the Artin generators, i.e $P = b_{i_1} \cdots b_{i_{|P|}}$, then every generator can move at most $|P|$ steps, while comparing starting and finishing sets and updating the permutations has complexity $\mathcal{O}(N)$. Thus the normal form can be computed in time $\mathcal{O}\left(|P|^2 N\right)$.

Thurston [16] gave a different algorithm for computing the Garside normal form of a positive braid word given as a product of permutation braids $P = P_1 P_2 \cdots P_k$. The complexity of the algorithm is $\mathcal{O}\left(k^2 N \log N\right)$, which might be faster than the previous algorithm, as the permutation braids might be products of multiple generators.

For the sake of completeness, from [9] we note that the BKL normal form of a braid w with length $|w|$ in the band generators ((3.4)) can be computed in time $\mathcal{O}\left(|w|^2 N\right)$.

All of this work will come in handy in the next chapter, where we discuss the word problem for braid groups, and in Section 6.6 when we look at an attack on the WalnutDSA signature scheme using the Garside normal form.

Chapter 4

Cryptographic protocols based on braid groups

4.1 Mathematical problems

In order for us to construct a cryptosystem, we require a hard mathematical problem to build upon. We will first introduce a few selected problems, which some of the older braid group cryptosystems are built upon.

4.1.1 The word problem

The word problem asks if two given words in some generating set represent the same element. In our case, this is equivalent to asking whether two braid words represent the same braid. This problem can be solved through the use of the normal forms from Section 3.2, since every braid has a unique normal form.

Notice that there is a surjective homomorphism $\phi\colon \mathcal{B}_N \to S_N$, which sends a braid b to its permutation $\phi(b) = \sigma$, where $\sigma(i) = \lambda(i)$. This morphism ignores whether crossings are positive or not, thus $\phi(b) = \phi(b^{-1})$ as they swap the same strands. If the resulting permutation is the identity permutation, we say that b is a *pure braid*, and the set $\mathcal{PB}_N$ of pure braids forms a subgroup. More precisely, we define

$$\mathcal{PB}_N := \ker(\phi). \tag{4.1}$$

In our case, since we are working in a group, we can reduce the word problem in the braid group to the word problem in the pure braid subgroup $\mathcal{PB}_N$: the question

$$w \stackrel{?}{=} w'$$

can be written equivalently as

$$w(w')^{-1} \stackrel{?}{=} e,$$

and thus we have reduced the problem of checking whether two braids are equivalent to checking whether the above product is equivalent to the nullstring. The first thing we do is

compute the permutation of $w(w')^{-1}$; if it is not trivial, the braids are not equivalent. If it is trivial, we can compute the normal form of the left hand side. Since it is unique, the normal form of $w(w')^{-1}$ is equal to the nullstring if the two braids are equivalent. Thus the word problem can be solved by the algorithm of Elrifai and Morton in time $\mathcal{O}\left(\left|w(w')^{-1}\right|^2 N\right)$, or even faster if we choose one of the other approaches.

Another approach is Dehornoys handle reduction algorithm presented in [14], which is an extension of free reductions, where we delete patterns of the form xx^{-1} or x^{-1}, that deals with subwords of the form $b_i^{e_i} v b_i^{-e_i}$. We will not go into the algorithm here, but remark that the algorithm is deterministic, and that a braid is equivalent to the trivial braid if the result of the handle reduction is the nullstring.

4.1.2 Conjugacy search problems

- **The conjugacy search problem:** In its simplest form, the conjugacy search problem is that given two braids x, y where it is known that $y = axa^{-1}$ for some $a \in \mathcal{B}_N$, find a $b \in \mathcal{B}_N$ such that $y = bxb^{-1}$.

- **Generalized conjugacy search problem:** A more general form of the previous problem is the following: given $x, y \in \mathcal{B}_N$ such that $y = axa^{-1}$ for some $a \in \mathcal{B}_m, m \leq N$, find a $b \in \mathcal{B}_m$ such that $y = bxb^{-1}$.

- **Multiple simultaneous conjugacy search problem:** A version of the conjugacy search problem is called the *multiple simultaneous conjugacy search problem*, which is as follows: given $y_i, x_i \in \mathcal{B}_N$ such that $y_i = ax_ia^{-1}$ for $1 \leq i \leq t$, find a $b \in \mathcal{B}_N$ such that $y_i = bx_ib^{-1}$ for all i.

4.2 Commutator based key exchange

The following protocol was proposed in [3, 5]. The protocol assumes nothing about the group other than that the conjugacy search problem is believed to be hard, and thus is still interesting even though the protocol in the braid group case has been proven insecure.

The public keys consist of two subgroups generated by the braids $p_1, p_2, \ldots, p_l$ and $q_1, q_2, \ldots, q_m$ in $\mathcal{B}_N$. Alice chooses her secret key as a word u on l letters and their inverses, and Bob chooses a secret key which is a word v on m letters and their inverses. The protocol is shown in Protocol 1.

Protocol 1 The Anshel-Anshel-Goldfeld protocol

Public information: The braid index N.
Key agreement:
Alice computes $a = u(p_1, \dots, p_l)$
Bob computes $b = v(q_1, \dots, q_m)$
Alice uses a to compute $q'_1 = aq_1a^{-1}, \dots, q'_m = aq_ma^{-1}$, and sends $q'_1, \dots q'_m$ to Bob
Bob similarly computes $p'_1, \dots p'_l$ and sends them to Alice
Alice computes $t_A = au(p'_1, \dots, p'_l)^{-1}$
Bob computes $t_B = v(q'_1, \dots, q'_m)b^{-1}$
The secret key is $t_A = t_B$.

The protocol works because

$$u(p'_1, \dots, p'_l)^{-1} = (bu(p_1, \dots, p_l)b^{-1})^{-1} = b(u(p_1, \dots, p_l))^{-1}b^{-1} = ba^{-1}b^{-1},$$

and hence

$$\begin{aligned} t_A &= au(p'_1, \dots, p'_l)^{-1} \\ &= ab(u(p_1, \dots, p_l))^{-1}b^{-1} \\ &= aba^{-1}b^{-1} \\ &= av(q_1, \dots, q_m)a^{-1}b^{-1} \\ &= v(q'_1, \dots, q'_m)b^{-1} \\ &= t_B. \end{aligned}$$

In [23], an attack against the multiple simultaneous conjugacy search problem was presented with very high success rate against the parameters $N = 80$ and $l = m = 20$, where every generator of the public subgroups consisted of 5 to 10 Artin generators. The different algorithms and techniques presented render this protocol useless with these parameters. The writers also mention having experimented with different versions of the protocol, including one with secret words a, b of length 100 in the public generators, but all of them turned out to be vulnerable, and thus braid groups seem unsuitable for this protocol.

4.3 Diffie-Hellman key agreement

We define two commuting subgroups of $\mathcal{B}_N$, which will be used in a Diffie-Hellman like scheme. The subgroups themselves are not commutative, but the elements from the different groups commute with each other. We define $L\mathcal{B}_N$ and $U\mathcal{B}_N$ as the subgroups generated by $b_1, \dots, b_{\lfloor N/2 \rfloor - 1}$ and $b_{\lfloor N/2 \rfloor + 1}, \dots, b_{N-1}$. Because of the first group relation ((3.2)), we see that for any $a \in L\mathcal{B}_N$ and $b \in U\mathcal{B}_N$ we have $ab = ba$.

Using these subgroups we construct a Diffie-Hellman type conjugacy search problem: given $x, y_a, y_b \in \mathcal{B}_N$ such that $y_a = a^{-1}xa$ and $y_b = b^{-1}xb$ for some $a \in L\mathcal{B}_N$ and $b \in U\mathcal{B}_N$, find

$$b^{-1}y_ab = b^{-1}a^{-1}xab = a^{-1}b^{-1}xba = a^{-1}y_ba.$$

We can see that this problem is a variant of the generalized conjugacy search problem. In [24], this problem was used to build the following key agreement protocol.

Protocol 2 Diffie-Hellman type key agreement.

Public information: The braid index N and a sufficiently complicated braid $x \in \mathcal{B}_N$.
Key agreement:
Alice chooses a random secret braid $a \in L\mathcal{B}_N$ and sends $y_1 = axa^{-1}$ to Bob
Bob chooses a random secret braid $b \in U\mathcal{B}_N$ and sends $y_2 = bxb^{-1}$ to Alice
Alice computes $K = ay_2a^{-1}$
Bob computes $K = by_1b^{-1}$.

We see that

$$K = ay_2a^{-1} = abxb^{-1}a^{-1} = baxa^{-1}b^{-1} = by_1b^{-1}$$

since the subgroups commute, and hence Alice and Bob compute the same secret key. Details concerning "sufficiently complicated braids" can be found in [24]. We can draw parallels to the regular Diffie-Hellman scheme. In this case x takes the place of the generator g, and conjugation axa^{-1} replaces exponentiation g^a.

4.4 A braid group public key encryption scheme

Along with the previous protocol, an encryption scheme using the same ideas was also presented in [24]. Let $H\colon \mathcal{B}_N \to \{0,1\}^k$ be a public, collision free, one way hash function. This means that the probability of having $H(b_1) = H(b_2)$ when $b_1 \neq b_2$ is negligible, and retrieving b from $H(b)$ is infeasible.

Protocol 3 Braid group encryption scheme

Public information: A collision free, one way hash function H, and the braid index N.
Key generation:
Alice chooses a private key $s \in L\mathcal{B}_N$, and a braid $p \in \mathcal{B}_N$. She computes $p' = sps^{-1}$ and publishes her public key (p, p').
Bob wants to send a message $m_B \in \{0,1\}^k$ to Alice. He picks a random braid $r \in U\mathcal{B}_N$ and computes $p'' = rpr^{-1}$.
Encryption:
Bob sends the pair (m''_B, p''), where $m''_B = m_B \oplus H(rp'r^{-1})$. Here $\oplus$ is addition in $\mathbb{Z}/2\mathbb{Z}$.
Decryption:
Alice computes $m_A = m''_B \oplus H(sp''s^{-1})$, and we get $m_A = m_B$.

Because the braids s and r commute, we get

$$sp''s^{-1} = srpr^{-1}s^{-1} = rsps^{-1}r^{-1} = rp'r^{-1},$$

and thus

$$m_A = m''_B \oplus H(sp''s^{-1}) = m_B \oplus H(rp'r^{-1}) \oplus H(sp''s^{-1}) = m_B$$

as claimed.

An attack against the Diffie-Hellman type problem from before was also presented in [23], and a variety of different choices for the index N and the length of the words p, s, r all seemed to indicate that the scheme was vulnerable to their attack.

Chapter 5

WalnutDSA

In this chapter we study the WalnutDSA$^{\text{TM}}$signature scheme, submitted to the NIST post quantum cryptography standardization process. We follow the presentation given in [4], and later revisions of this article. The scheme involves a representation of the braid group called the *colored Burau representation*, and a group action called *E-multiplication*.

5.1 Colored Burau representation

Let $\mathbb{F}_q$ denote the field with q elements, and let $\mathbb{F}_q[t_1, t_1^{-1}, \ldots, t_N, t_N^{-1}]$ be the ring of Laurent polynomials in N variables. We want to introduce the *colored Burau representation*

$$\Pi_{CB} \colon \mathcal{B}_N \to GL\left(N, \mathbb{F}_q[t_1, t_1^{-1}, \ldots, t_N, t_N^{-1}]\right) \rtimes S_N. \tag{5.1}$$

The symbol $\rtimes$ is notation for a semidirect product, which is a generalization of a direct product. For two groups G and H, their semidirect product with respect to a homomorphism $\varphi \colon H \to \mathrm{Aut}(G)$ is the group with $G \times H$ as the underlying set, and the group operation $*$ defined as

$$(g_1, h_1) * (g_2, h_2) = (g_1 \varphi(h_1)(g_2), h_1 h_2).$$

For the generators b_i, we create a colored Burau matrix as follows: If $i = 1$, we set

$$CB(b_1) = \begin{pmatrix} -t_1 & 1 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & \vdots \\ \vdots & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}, \quad CB(b_1^{-1}) = \begin{pmatrix} -\frac{1}{t_2} & \frac{1}{t_2} & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & \vdots \\ \vdots & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}.$$

For $2 \leq i < N$ we define

$$CB(b_i) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & t_i & -t_i & 1 & & \\ & & & \ddots & & \\ & & & & & 1 \end{pmatrix}, \quad CB(b_i^{-1}) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & 1 & -\frac{1}{t_{i+1}} & \frac{1}{t_{i+1}} & & \\ & & & \ddots & & \\ & & & & & 1 \end{pmatrix}.$$

The variables here appear in row i. Recall that there is a surjection $\phi\colon \mathcal{B}_N \to S_N$ which sends a braid b to its corresponding permutation σ. We can use this to associate to each generator braid b_i a colored Burau matrix/permutation pair $(CB(b_i), \phi(b_i))$. We now want to define a multiplication of such pairs, in order to form the *colored Burau group*. To begin, we notice that given a Laurent polynomial $f(t_1, \ldots, t_N)$ in N variables, a permutation $\sigma \in S_N$ can act on it by permuting the indices of the variables. We denote this action as $f \mapsto {}^{\sigma}f$:

$$ {}^{\sigma}f(t_1, \ldots, t_N) = f(t_{\sigma(1)}, \ldots, t_{\sigma(N)}). \tag{5.2}$$

Note that the action of σ is really a group action on the natural numbers, and we stress that even though it is natural to think of the permutation as acting from left to right, similarly to how the braids permute from top to bottom, the action starts on the right and moves left. For example, $(1,2)(2,3) = (1,2,3)$ instead of $(1,2)(2,3) = (1,3,2)$.

We then extend this action to matrices over the ring of Laurent polynomials by letting the permutation act on each entry, and we denote this action by $M \mapsto {}^{\sigma}M$. For ease of notation, let σ_i be the i-th transposition $(i, i+1)$. Multiplication of two pairs of Burau matrices and permutations is then given as follows: given two generator braids $b_i^{\pm}, b_j^{\pm}$ with permutations σ_i, σ_j, the pair associated with $b_i^{\pm} \cdot b_j^{\pm}$ is

$$\left(CB(b_i^{\pm}), \sigma_i\right) \cdot \left(CB(b_j^{\pm}), \sigma_j\right) = \left(CB(b_i^{\pm}) \cdot {}^{\sigma_i}CB(b_j^{\pm}), \sigma_i\sigma_j\right). \tag{5.3}$$

We can extend this definition to general braids inductively: given $b = b_{i_1}^{e_1} b_{i_2}^{e_2} \cdots b_{i_k}^{e_k}$, the matrix part $CB(b)$ of the associated pair is

$$\left(CB(b_{i_1}^{e_1}) \cdot {}^{\sigma_{i_1}}CB(b_{i_2}^{e_2}) \cdot {}^{\sigma_{i_1}\sigma_{i_2}}CB(b_{i_3}^{e_3}) \cdots {}^{\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_{k-1}}}CB(b_{i_1}^{e_1}) \cdot CB(b_{i_2}^{e_2}) \cdots CB(b_{i_k}^{e_k})\right). \tag{5.4}$$

The *colored Burau representation* is then given as

$$\begin{aligned} \Pi_{CB}\colon \mathcal{B}_N &\to GL\left(N, \mathbb{F}_q[t_1, t_1^{-1}, \ldots, t_N, t_N^{-1}]\right) \rtimes S_N \\ b &\mapsto \left(CB(b), \sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_k}\right). \end{aligned}$$

Notice that $\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_k}$ is just $\phi(b)$, the permutation of the braid, and so we can write this as $\Pi_{CB}(b) = (CB(b), \sigma_b)$. One now checks that the braid relations hold for the representation. For $|i-j| > 1$, this is easy because the transpositions will be disjoint and hence $\sigma_i\sigma_j = \sigma_j\sigma_i$ and ${}^{\sigma_i}CB(b_j^{\pm}) = CB(b_j^{\pm})$. Finally, we get two block matrices which will commute.

For the other relation, the equations we have to check are

$$\begin{aligned} CB(b_i) \cdot {}^{(i,i+1)}CB(b_{i+1}) \cdot {}^{(i,i+1)(i+1,i+2)}CB(b_i) = \\ CB(b_{i+1}) \cdot {}^{(i+1,i+2)}CB(b_i) \cdot {}^{(i+1,i+2)(i,i+1)}CB(b_{i+1}) \end{aligned}$$

and

$$(i,i+1)(i+1,i+2)(i,i+1) = (i+1,i+2)(i,i+1)(i+1,i+2).$$

It is fairly straightforward to see that the permutations are the same, and we have

$$(i,i+1)(i+1,i+2)(i,i+1) = (i,i+2) = (i+1,i+2)(i,i+1)(i+1,i+2),$$

which matches the permutation of the braids in Figure 3.2b.

For the matrix part we begin by writing the matrices in block form:

$$CB(b_i) = \left(\begin{array}{c|cccc|c} \mathbf{I}_{i-2} & \multicolumn{4}{c|}{\mathbf{0}_{i-2,4}} & \mathbf{0}_{i-2,N-i-2} \\ \hline & 1 & 0 & 0 & 0 & \\ & t_i & -t_i & 1 & 0 & \\ \mathbf{0}_{4,i-2} & 0 & 0 & 1 & 0 & \mathbf{0}_{4,N-i-2} \\ & 0 & 0 & 0 & 1 & \\ \hline \mathbf{0}_{N-i-2,i-2} & \multicolumn{4}{c|}{\mathbf{0}_{N-i-2,4}} & \mathbf{I}_{N-i-2} \end{array}\right)$$

$$CB(b_{i+1}) = \left(\begin{array}{c|cccc|c} \mathbf{I}_{i-2} & \multicolumn{4}{c|}{\mathbf{0}_{i-2,4}} & \mathbf{0}_{i-2,N-i-2} \\ \hline & 1 & 0 & 0 & 0 & \\ & 0 & 1 & 0 & 0 & \\ \mathbf{0}_{4,i-2} & 0 & t_{i+1} & -t_{i+1} & 1 & \mathbf{0}_{4,N-i-2} \\ & 0 & 0 & 0 & 1 & \\ \hline \mathbf{0}_{N-i-2,i-2} & \multicolumn{4}{c|}{\mathbf{0}_{N-i-2,4}} & \mathbf{I}_{N-i-2} \end{array}\right).$$

We see that in order to check the first equation, it suffices to check the middle block, since the rest will automatically commute. In the calculations, we will therefore abuse notation by writing $CB(b_i)$ while referring to the middle block. For $2 \leq i \leq N-2$ we compute

$$CB(b_i) \cdot^{(i,i+1)} CB(b_{i+1}) \cdot^{(i,i+1)(i+1,i+2)} CB(b_i) =$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i & -t_i & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & t_i & -t_i & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_{i+1} & -t_{i+1} & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i & 0 & -t_i & 1 \\ t_i t_{i+1} & -t_i t_{i+1} & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$CB(b_{i+1}) \cdot^{(i+1,i+2)} CB(b_i) \cdot^{(i+1,i+2)(i,i+1)} CB(b_{i+1}) =$$
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & t_{i+1} & -t_{i+1} & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i & -t_i & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & t_i & -t_i & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i & 0 & -t_i & 1 \\ t_i t_{i+1} & -t_i t_{i+1} & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

and see that the relation is satisfied. For $i = 1$, the computations become

$$CB(b_1) \cdot^{(1,2)} CB(b_2) \cdot^{(1,2)(2,3)} CB(b_1) =$$
$$\begin{pmatrix} -t_1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ t_1 & -t_1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -t_2 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -t_2 & 1 \\ -t_1 t_2 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$CB(b_2) \cdot {}^{(2,3)}CB(b_1) \cdot {}^{(2,3)(1,2)}CB(b_2) = \\ \begin{pmatrix} 1 & 0 & 0 \\ t_2 & -t_2 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -t_1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ t_1 & -t_1 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -t_2 & 1 \\ -t_1 t_2 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

and thus all of the relations are satisfied, and we have a valid representation of the group. Since the relations are satisfied, and any occurrence of $b_i b_i^{-1}$ in the braid word disappears in the representation, it is independent of the expression of b in the generators.

5.2 E-multiplication

The group action *E-multiplication* was first introduced in [2] as a one way function. Formally, it is a right action of the colored Burau group, henceforth denoted as $\mathcal{CB}_N$, on the set $GL(N, \mathbb{F}_q) \times S_N$. We denote an ordered list of non-zero elements in the finite field as *t-values*:

$$\{\tau_1, \tau_2, \ldots, \tau_N\} \subset \mathbb{F}_q^{\times}.$$

For any set of t-values, i.e a vector $\vec{\tau}$, we can evaluate a Laurent polynomial $f(t_1, t_2, \ldots, t_N)$ to obtain an element of $\mathbb{F}_q$:

$$f(t_1, t_2, \ldots, t_N)(\vec{\tau}) := f(\vec{\tau}).$$

Like before, we extend this action to matrices over Laurent polynomials. We now define E-multiplication. It takes as input two ordered pairs

$$(M, \sigma_0) \in GL(N, \mathbb{F}_q) \times S_N, \quad (CB(b), \sigma_b) \in \mathcal{CB}_N,$$

and returns a new pair $(M', \sigma') \in GL(N, \mathbb{F}_q) \times S_N$. We denote the operation of E-multiplication with a $\star$, and write

$$(M', \sigma') = (M, \sigma_0) \star (CB(b), \sigma_b). \tag{5.5}$$

As for the colored Burau representation, we define the operation inductively. When the braid $b = b_i^{\pm}$ is a single generator or its inverse, we set

$$(M, \sigma_0) \star (CB(b_i^{\pm}), \sigma_i) = \left(\left(M \cdot {}^{\sigma_0}CB(b_i^{\pm}) \right)(\vec{\tau}), \sigma_0 \sigma_i \right). \tag{5.6}$$

We see that the operation is the same as doing a regular multiplication in the colored Burau group, and then evaluating the matrix part in a set of t-values. In the general case where $b = b_{i_1}^{e_1} b_{i_2}^{e_2} \cdots b_{i_k}^{e_k}$, we set

$$(M, \sigma_0) \star (CB(b), \sigma_b) = (M, \sigma_0) \star \left(CB(b_{i_1}^{e_1}), \sigma_{i_1} \right) \star \left(CB(b_{i_2}^{e_2}), \sigma_{i_2} \right) \star \cdots \star \left(CB(b_{i_k}^{e_k}), \sigma_{i_k} \right), \tag{5.7}$$

where we associate from the left.

For ease of notation, and to write the operation as an action by the braid group, for a pair $(M, \sigma) \in GL(N, \mathbb{F}_q) \times S_N$ and a braid b with permutation σ_b, we write

$$(M, \sigma) \star b := (M, \sigma) \star (CB(b), \sigma_b). \tag{5.8}$$

It is worth noting that for a pair (M, σ) and two generator braids $b_i^{\pm}, b_j^{\pm}$, we have

$$\begin{aligned}
\big((M, \sigma) \star b_i^{\pm}\big) \star b_j^{\pm} &= \big((M, \sigma) \star (CB(b_i^{\pm}), \sigma_i)\big) \star b_j^{\pm} \\
&= ((M \cdot {}^{\sigma}CB(b_i^{\pm}))(\vec{\tau}), \sigma\sigma_i) \star b_j^{\pm} \\
&= (((M \cdot {}^{\sigma}CB(b_i))(\vec{\tau}) \cdot {}^{\sigma\sigma_i}CB(b_j^{\pm}))(\vec{\tau}), \sigma\sigma_i\sigma_j) \\
&= ((M \cdot {}^{\sigma}CB(b_i^{\pm}) \cdot {}^{\sigma\sigma_i}CB(b_j^{\pm}))(\vec{\tau}), \sigma\sigma_i\sigma_j) \\
&= ((M \cdot {}^{\sigma}\ (CB(b_i^{\pm}) \cdot {}^{\sigma_i}CB(b_j^{\pm})))(\vec{\tau}), \sigma\sigma_i\sigma_j) \\
&= (M, \sigma) \star (CB(b_i^{\pm}) \cdot {}^{\sigma_i}CB(b_j^{\pm}), \sigma_i\sigma_j) \\
&= (M, \sigma) \star \big((CB(b_i^{\pm}), \sigma_i) \cdot (CB(b_j^{\pm}), \sigma_j)\big) \\
&= (M, \sigma) \star (b_i^{\pm} \cdot b_j^{\pm}),
\end{aligned} \tag{5.9}$$

where the final multiplication happens in the colored Burau group. This is a requirement for E-multiplication to formally be a group action, but it is a nice result which we can extend to get

$$(M, \sigma) \star b = (M, \sigma) \star (CB(b), \sigma_b) = ((M \cdot {}^{\sigma}CB(b))(\vec{\tau}), \sigma\sigma_b). \tag{5.10}$$

Note however that when we actually compute an E-multiplication, we do it generator by generator, meaning that we evaluate the matrices in every step instead of computing with polynomials.

The calculations above can be done since σ is a bijection, and since for Laurent polynomials f and g, we can extend $(f \cdot g)(\vec{\tau}) = f(\vec{\tau})g(\vec{\tau})$ to show that the evaluation map

$$\begin{aligned}
\phi_{\vec{\tau}} \colon GL(N, \mathbb{F}_q[t_1, t_1^{-1}, \ldots, t_N, t_N^{-1}]) &\to GL(N, \mathbb{F}_q) \\
M &\mapsto M(\vec{\tau})
\end{aligned}$$

is a homomorphism.

Finally we note that since the colored Burau representation is independent of the expression of the braid and evaluation is a homomorphism, E-multiplication is also independent of the expression of the braid.

5.3 Cloaking elements

The security of WalnutDSA is based on certain braids which hide and obscure our messages. We call these braids *cloaking elements*.

Definition 5.1 (Cloaking element)**.** Let $(M, \sigma) \in GL(N, \mathbb{F}_q) \times S_N$. A braid v in the pure braid subgroup $\mathcal{PB}_N$ is called a cloaking element if

$$(M, \sigma) \star v = (M, \sigma). \tag{5.11}$$

We denote the set of cloaking elements for the pair (M, σ) as $\mathrm{Cloak}_{(M,\sigma)}$.

Lemma 5.2. The set of cloaking elements form a subgroup, and is called the stabilizer of (M, σ).

Proof. Let $v, w \in \mathrm{Cloak}_{(M,\sigma)}$. By the calculations in 5.9 we have

$$\begin{aligned}(M,\sigma) \star (v \cdot w) &= (M,\sigma) \star ((CB(v),\sigma_v) \cdot (CB(w),\sigma_w)) \\ &= ((M,\sigma) \star (CB(v),\sigma_v)) \star (CB(w),\sigma_w) \\ &= (M,\sigma) \star (CB(w),\sigma_w) \\ &= (M,\sigma),\end{aligned}$$

and thus $v \cdot w \in \mathrm{Cloak}_{(M,\sigma)}$. □

We remark that whether or not a braid is a cloaking element depends on the t-values. The authors present the following way to generate cloaking elements.

Proposition 5.3. Fix integers $N \geq 2$ and $1 < a < b \leq N$. Assume that the t-values τ_a, τ_b satisfy $\tau_a \tau_b = -1$. Let $M \in GL(N, \mathbb{F}_q)$ and $\sigma \in S_N$. A cloaking element v of (M, σ) is then given by $v = w b_i^4 w^{-1}$ where b_i is any Artin generator, and the permutation σ_w satisfies

$$\sigma_w(i) = \sigma^{-1}(a), \quad \sigma_w(i+1) = \sigma^{-1}(b).$$

Proof. As in the proof of the group relations for $\mathcal{CB}_N$, we only look at the central block matrix of $CB(b_i)$ in the following computation, and we abuse the notation in the same way by referring to $CB(b_i)$ as the middle block of the full matrix:

$$\begin{aligned}CB(b_i^2) &= \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i & -t_i & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot^{\sigma_i} \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i & -t_i & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathrm{Id}_{S_N} \right) \\ &= \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i - t_i t_{i+1} & t_i t_{i+1} & 1 - t_i & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathrm{Id}_{S_N} \right).\end{aligned}$$

This gives the following central block for $CB(b_i^4)$

$$\begin{aligned}CB(b_i^4) &= \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i - t_i t_{i+1} & t_i t_{i+1} & 1 - t_i & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot^{\mathrm{Id}_{S_N}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ t_i - t_i t_{i+1} & t_i t_{i+1} & 1 - t_i & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathrm{Id}_{S_N} \right) \\ &= \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ (t_i - t_i t_{i+1}) + t_i t_{i+1}(t_i - t_i t_{i+1}) & (t_i t_{i+1})^2 & t_i t_{i+1}(1 - t_i) + (1 - t_i) & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \mathrm{Id}_{S_N} \right)\end{aligned}$$

Using the assumptions $\sigma\sigma_w(i) = a, \sigma\sigma_w(i+1) = b$ and $\tau_a \tau_b = -1$, we get

$$\begin{aligned}(M,\sigma) \star v &= (M,\sigma) \star ((CB(w),\sigma_w) \cdot (CB(b_i^4), \mathrm{Id}_{S_N}) \cdot (CB(w^{-1}),\sigma_w)) \\ &= ((M \cdot^{\sigma} (CB(w) \cdot^{\sigma_w} CB(b_i^4) \cdot^{\sigma_w \mathrm{Id}_{S_N}} CB(w^{-1})))(\vec{\tau}), \sigma\sigma_w \,\mathrm{Id}_{S_N}\, \sigma_w^{-1}) \\ &= ((M \cdot^{\sigma} (CB(w) \cdot^{\sigma_w} CB(w^{-1})))(\vec{\tau}), \sigma) \\ &= (M,\sigma).\end{aligned}$$

□

Notice that in the construction above, the matrix M is arbitrary, and we can thus say without ambiguity that v cloaks σ. We apply this in the following way. Fix a braid $b = b_{i_1}^{e_1} b_{i_2}^{e_2} \cdots b_{i_l}^{e_l}$, and choose a point $1 \leq k \leq l$. Write $b = x_1 \cdot x_2$, where $x_1 = b_{i_1}^{e_1} \cdots b_{i_{k-1}}^{e_{k-1}}$ and $x_2 = b_{i_k}^{e_k} \cdots b_{i_l}^{e_l}$. For a pair (m_0, σ_0) we have

$$(m_0, \sigma_0) \star b = ((m_0, \sigma_0) \star x_1) \star x_2).$$

Using Proposition 5.3 we can generate a cloaking element v for the permutation $\sigma_0\sigma_{x_1}$. By construction, for any matrix M we then have that $(M, \sigma_0\sigma_{x_1}) \star v = (M, \sigma_0\sigma_{x_1})$. Since $(m_0, \sigma_0) \star x_1$ is on the form $(M, \sigma_0\sigma_{x_1})$ for some matrix M, we get

$$\begin{aligned}(m_0, \sigma_0) \star b &= ((m_0, \sigma_0) \star x_1) \star x_2) \\ &= (M, \sigma_0\sigma_{x_1}) \star x_2 \\ &= (M, \sigma_0\sigma_{x_1}) \star v \star x_2 \\ &= ((m_0, \sigma_0) \star x_1) \star v \star x_2 \\ &= (m_0, \sigma_0) \star x_1 \star v \star x_2.\end{aligned}$$

Thus we have created a new braid $b' = x_1 \cdot v \cdot x_2$ which contains v and has the property that $(m_0, \sigma_0) \star b = (m_0, \sigma_0) \star b'$. We call the inserted cloaking element v a *concealed cloaking element*. This process can be iterated, and we have the following definition.

Definition 5.4. Given a braid $b \in \mathcal{B}_N$, the output of κ iterations of randomly inserting cloaking elements is denoted $\kappa(b)$ and called a κ-cloaking of b.

5.4 Cryptographic notation

Following the main article, we present some notation for cryptographic protocols.

- Let S be a set

  $\langle S \rangle$ denotes a unique encoding of S as a binary string.

  $s \xleftarrow{\$} S$ denotes the operation of randomly choosing $s \in S$.

- Let $A(*; \rho)$ be a randomized algorithm with randomness based on a coin ρ.

  $A(y_1, \ldots, y_q; \rho)$ denotes the output of algorithm A on input $y_1, \ldots, y_q$ and coin ρ.

  $z \xleftarrow{\$} A(y_1, \ldots, y_q)$ denotes choosing ρ at random and letting $z = A(y_1, \ldots, y_q; \rho)$.

- Let $b \in \mathcal{B}_N$. We define $\mathcal{P}(b) := (\mathrm{Id}_N, \mathrm{Id}_{S_N}) \star b$.

The security of Walnut is based on the following problem:

Definition 5.5 (The REM problem)**.** The REM-problem is: consider the braid group $\mathcal{B}_N$ for $N \geq 10$. Given a pair $(M, \sigma) \in GL(N, \mathbb{F}_q) \times S_N$ where it is stipulated that $(M, \sigma) = \mathcal{P}(b)$ for some unknown braid b with sufficiently long BKL normal form [9], then it is infeasible to find a braid b' such that $(M, \sigma) = \mathcal{P}(b')$. Informally, this means that reversing E-multiplication is hard.

5.5 Key generation

The Walnut protocol allows a signer with a private/public key pair to create a signature for a given message that can be authenticated by anyone who knows the public key and the verification protocol. The system wide parameters, denoted par, are generated by a central authority via a parameter generation algorithm denoted Pg, so $par \xleftarrow{\$} \mathrm{Pg}$. A signer S generates its own key pair, denoted $(\mathrm{Pub}(S), \mathrm{Priv}(S))$, via a key generation algorithm Kg. More concretely: $(\mathrm{Pub}(S), \mathrm{Priv}(S)) \xleftarrow{\$} \mathrm{Kg}(par)$.

5.5.1 Public system wide parameters:

- An integer $N \geq 10$ and the associated braid group $\mathcal{B}_N$.
- An integer $\kappa > 1$ which is chosen to meet the security level. The signature will use κ concealed cloaking elements.
- A rewriting algorithm $\mathcal{R}\colon \mathcal{B}_N \to \mathcal{B}_N$ which uses the braid relations to rewrite a word, such as the normal forms from Section 3.2 or Dehornoys handle reduction (page 20).
- A finite field $\mathbb{F}_q$.
- Two integers $1 < a < b < N$.
- t-values $\{\tau_1, \ldots, \tau_N\}$, where $\tau_i \in \mathbb{F}_q^\times$ and $\tau_a \cdot \tau_b = -1$.

5.5.2 Private key:

The signer's private key is a pair of reduced (no subwords $x \cdot x^{-1}$ or $x^{-1} \cdot x$ appear for any generator) braids:

$$\mathrm{Priv}(S) = (w, w') \in \mathcal{B}_N \times \mathcal{B}_N,$$

where the braids w, w' and $w \cdot w'$ are not in the pure braid group.

5.5.3 Public key:

The signer's public key consists of two matrix and permutation pairs which are generated from the public key:

$$\mathrm{Pub}(S) = (\mathcal{P}(w), \mathcal{P}(w')).$$

5.6 Encoder

In order to use the protocol, we require a method to turn a message $m \in \{0,1\}^*$ into a braid word. Let $H\colon \{0,1\}^* \to \{0,1\}^{2\eta}$ be a cryptographically secure 2η-bit hash function for $\eta \geq 1$. In the main article, it is formally defined what "cryptographically secure" means, but other papers have assumed that the intended meaning is that of a "random oracle". In the verification step of the signature scheme, it can be shown that the permutation of the

encoded message has to be trivial, and therefore the output braid must be a pure braid. We thus want to construct an injective function $E\colon \{0,1\}^{2\eta} \to H_N$, where H_N is a subgroup of the pure braid group. This will make sure that two distinct words will give distinct encodings. The encoding algorithm is then presented as follows: the collection of pure braids given by

$$\begin{aligned} g_{(N-1),N} &= b_{N-1}^2 \\ g_{(N-2),N} &= b_{N-1} \cdot b_{N-2}^2 \cdot b_{N-1}^{-1} \\ g_{(N-3),N} &= b_{N-1} \cdot b_{N-2} \cdot b_{N-3}^2 \cdot b_{N-2}^{-1} \cdot b_{N-1}^{-1} \\ &\vdots \\ g_{1,N} &= b_{N-1} b_{N-2} \cdots b_2 b_1^2 b_2^{-1} \cdots b_{N-2}^{-1} b_{N-1}^{-1} \end{aligned} \tag{5.12}$$

will generate a free subgroup $H_N \leq \mathcal{PB}_N$. For simplicity, we write $g_i = g_{i,N}$ when N is clear from context.

5.6.1 Encoding algorithm:

We create the braid $E(H(m)) \in \mathcal{B}_N$ as follows. The hash $H(m)$ of the message consists of η 2-bit blocks. Fix a collection S of η tuples S_k, where each S_k is a four-tuple of generators from the set above:

$$S_k = (g_{k_1}, g_{k_2}, g_{k_3}, g_{k_4}).$$

For the k-th block of $H(m)$, there is a bijective mapping to its corresponding tuple S_k, and the output of the encoder is the product of the generators obtained from going through all the blocks of $H(m)$ in order. The encoding function is injective, because given $E(M_1) = g_{i_1} g_{i_2} \cdots g_{i_l} = E(M_2)$ for two message digests M_1 and M_2, each g_{i_k} must come from the same tuple S_k. Since the correspondence between message-block and tuple is bijective, the k-th block of the message digest must be the same for both messages. Since this holds for all k, we have that $M_1 = M_2$, which shows that the encoder is injective.

5.7 Signature generation and verification

5.7.1 Signature generation:

To sign a message $m \in \{0,1\}^*$, the signer does as follows:

1. Compute $H(m)$.
2. Generate cloaking elements v, v_1, v_2 where v cloaks $(\mathrm{Id}_N, \mathrm{Id}_{S_N})$, v_1 cloaks $\mathcal{P}(w)$ and v_2 cloaks $\mathcal{P}(w')$.
3. Generate the encoded message $E(H(m))$.
4. Compute $\mathrm{Sig} = \mathcal{R}(\kappa(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2))$, which is now a rewritten braid. We generate the κ concealed cloaking elements from the element $\mathcal{P}(w)$.

5. The signature is the pair $(H(m), \mathrm{Sig})$.

The cloaking elements v, v_1, v_2 disappear under the correct E-multiplication, as does the κ concealed cloaking elements.

5.7.2 Signature verification:

A signature (m, Sig) is verified as follows:

1. Generate the encoded message $E(H(m))$.

2. Evaluate $\mathcal{P}(E(H(m)))$.

3. Evaluate $\mathcal{P}(w) \star \mathrm{Sig}$.

4. Test the equality

$$\mathrm{Matrix}\big(\mathcal{P}(w) \star \mathrm{Sig}\big) \stackrel{?}{=} \mathrm{Matrix}\big(\mathcal{P}\big(E\big(H(m)\big)\big)\big) \cdot \mathrm{Matrix}\big(\mathcal{P}(w')\big), \tag{5.13}$$

   where Matrix denotes the matrix part of the pair, and the multiplication on the right is ordinary matrix multiplication.

5. Reject signatures longer than 2^{14} Artin generators[1].

6. Accept the signature if 4 and 5 hold.

We show the computations involved in the verification process. Since v cloaks $(\mathrm{Id}_N, \mathrm{Id}_{S_N})$, v_1 cloaks $\mathcal{P}(w)$ and v_2 cloaks $\mathcal{P}(w')$, we have that $\sigma_v = \sigma_{v_1} = \sigma_{v_2} = \mathrm{Id}_{S_N}$, and

$$\begin{aligned}
(\mathrm{Id}_N, \mathrm{Id}_{S_N}) \star v &= (CB(v)(\vec{\tau}), \mathrm{Id}_{S_N}) = (\mathrm{Id}_N, \mathrm{Id}_{S_N}) \\
\mathcal{P}(w) \star v_1 &= ((CB(w) \cdot {}^{\sigma_w}CB(v_1))(\vec{\tau}), \sigma_w) = (CB(w)(\vec{\tau}), \sigma_w) \\
\mathcal{P}(w') \star v_2 &= ((CB(w') \cdot {}^{\sigma_{w'}}CB(v_2))(\vec{\tau}), \sigma_{w'}) = (CB(w')(\vec{\tau}), \sigma_{w'}).
\end{aligned} \tag{5.14}$$

Given the fact that E-multiplication is independent of the expression of braids, and how we defined a κ-cloaking of a braid, we have that

$$\begin{aligned}
\mathcal{P}(w) \star \mathrm{Sig} &= \mathcal{P}(w) \star \mathcal{R}(\kappa(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2)) \\
&= \mathcal{P}(w) \star (v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2).
\end{aligned}$$

By construction, $\sigma_{E(H(m))} = \mathrm{Id}_{S_N}$. Note also that because the map ϕ, which sends a braid b to its permutation σ_b, is a homomorphism, $\sigma_{w^{-1}} = \sigma_w^{-1}$. Combining all of this,

[1] This is to block certain length based attacks on the scheme.

the matrix part of $\mathcal{P}(w) \star \mathrm{Sig}$ becomes

$$\begin{aligned}
&\Big(CB(w) \cdot {}^{\sigma_w}\Big(CB(v_1) \cdot CB(w^{-1}) \\
&\qquad \cdot {}^{\sigma_w^{-1}}CB(v) \cdot {}^{\sigma_w^{-1}}CB(E(H(m))) \cdot {}^{\sigma_w^{-1}}CB(w') \cdot {}^{\sigma_w^{-1}\sigma_{w'}}CB(v_2)\Big)\Big)\,(\vec{\tau}) \\
&= \Big(CB(w) \cdot {}^{\sigma_w}\Big(CB(v_1) \cdot CB(w^{-1}) \\
&\qquad \cdot {}^{\sigma_w^{-1}}\Big(CB(v) \cdot CB(E(H(m))) \cdot CB(w') \cdot {}^{\sigma_{w'}}CB(v_2)\Big)\Big)\Big)\,(\vec{\tau}) \\
&= \Big(CB(w) \cdot {}^{\sigma_w}\Big(CB(v_1) \cdot CB(w^{-1})\Big) \\
&\qquad \cdot {}^{\sigma_w\sigma_w^{-1}}\Big(CB(v) \cdot CB(E(H(m))) \cdot CB(w') \cdot {}^{\sigma_{w'}}CB(v_2)\Big)\Big)\,(\vec{\tau}) \\
&\overset{*}{=} \big(CB(w) \cdot {}^{\sigma_w}CB(w^{-1}) \cdot CB(E(H(m))) \cdot CB(w')\big)\,(\vec{\tau}) \\
&= \big(CB(E(H(m))) \cdot CB(w')\big)\,(\vec{\tau}) \\
&= \mathrm{Matrix}\Big(\mathcal{P}(E(H(m)))\Big) \cdot \mathrm{Matrix}\Big(\mathcal{P}(w')\Big),
\end{aligned}$$

where $*$ follows from the identities derived in (5.14). In order to save space, we have skipped some steps in the equations while evaluating the matrices.

5.8 Security proof

We present a security proof that is virtually identical to the one in [4]. We will present a slightly modified version of the above scheme called WalnutDSA-I, and show that it is existentially unforgeable under adaptive chosen-message attacks in the random oracle model. To show this, we assume that if a forger has the ability to forge valid signatures of a specific type with a non-negligible probability, he is able to break the REM-problem with non-negligible probability. We define the set

$$\mathrm{Cloak} := \left\{(v, v_1, v_2) \mid v \in \mathrm{Cloak}_{\mathrm{Id}}, v_1 \in \mathrm{Cloak}_{\mathcal{P}(w)}, v_2 \in \mathrm{Cloak}_{\mathcal{P}(w')}\right\},$$

where $\mathrm{Id} = (\mathrm{Id}_N, \mathrm{Id}_{S_N})$. The system wide parameters and key generation algorithm are the same as in the original scheme, and are given by

$$\begin{aligned}
par &\xleftarrow{\$} \mathrm{Pg} \\
(\mathrm{Pub}(S), \mathrm{Priv}(S)) &\xleftarrow{\$} \mathrm{Kg}(par).
\end{aligned}$$

To sign a message m we use two hash functions $H, G\colon \{0,1\}^* \to \{0,1\}^{2\eta}$, and we use the following protocol

1. $(v, v_1, v_2) \xleftarrow{\$} \mathrm{Cloak}$, $V = \langle (v, v_1, v_2) \rangle$, so V is a string encoding of (v, v_1, v_2).
2. Compute $E(H(m||G(V)))$, where $||$ means string concatenation.
3. Compute
$$\mathrm{Sig} = \mathcal{R}\Big(\kappa\big(v_1 w^{-1} v \cdot E\big(H\big(m||G(V)\big)\big) \cdot w' v_2\big)\Big). \tag{5.15}$$
The signature is $(m, H(m), G(V), \mathrm{Sig})$.

To validate the signature one checks that

$$\mathrm{Matrix}\Big(\mathcal{P}(w) \star \mathrm{Sig}\Big) \stackrel{?}{=} \mathrm{Matrix}\Big(\mathcal{P}\big(E\big(H\big(m||G(V)\big)\big)\big)\Big) \cdot \mathrm{Matrix}\Big(\mathcal{P}(w')\Big). \quad (5.16)$$

Notice that all signatures on a message m created by an honest signer will lie in the double coset

$$\mathrm{DC}_{m,V,H,G} := \left\{ \mathcal{R}\Big(X \cdot E\big(H\big(m||G(V)\big)\big) \cdot Y\Big) \,\middle|\, X, Y \in \mathcal{B}_N \right\}, \quad (5.17)$$

but there might be valid signatures lying outside of this set.

Security proof

We assume the existence of a forger $\mathcal{F}$, that when given a public key $(\mathrm{Pub}(S))$ and a message m can produce a valid signature lying in $\mathrm{DC}_{m,V,H,G}$. This is fairly restrictive, but as noted by the authors it is important to rule out certain kinds of attacks.

More precisely we define $\mathcal{F}$ to be a randomized algorithm that can make hash queries to a random oracle.

- Hash query: let $\mathcal{O}_\rho$ denote a random oracle that depends on a coin ρ. The oracle responds to a hash query, which is a string $str \in \{0,1\}^*$, with the hash of the string.

To describe the forger $\mathcal{F}$, we study WalnutDSA-I with parameters and keys given by

$$par \xleftarrow{\$} \mathrm{Pg}, \quad (\mathrm{Pub}(S), \mathrm{Priv}(S)) \xleftarrow{\$} \mathrm{Kg}(par).$$

We assume that the hash function H is fixed and multi-collision-resistant, while the hash function $G = G_\rho$ is given by the oracle $\mathcal{O}_\rho$, which depends on a coin ρ. The forger is defined to be a randomized algorithm that when given a message $m \in \{0,1\}^*$, a public key $\mathrm{Pub}(S)$ and a coin ρ, outputs a tuple (m, h, g_ρ, s), where $h = H(M), g_\rho = G_\rho(V)$, and $V \xleftarrow{\$} \mathrm{Cloak}, s \xleftarrow{\$} \mathrm{DC}_{m,V,H,G}$. We assume that the probability that (m, h, g_ρ, s) is a valid signature is non-negligible.

The proof is based on the *forking lemma*, and we will present the needed result without a proof. For details, see [7, 29].

Lemma 5.6. Let $\mathcal{F}$ be run twice with input

$$(m, \mathrm{Pub}(S), \rho), \quad (m, \mathrm{Pub}(S), \rho').$$

Then, with non-negligible probability, $\mathcal{F}$ will output two valid signatures

$$(m, h, g_\rho, s), \quad (m, h, g_{\rho'}, s'),$$

such that $g_\rho \neq g_{\rho'}$.

A non-trivial application of the lemma provides us with a method to run the forger twice with the same cloaking elements V, which we require for the next step. Using the lemma we can show that the forger can break the REM-problem from Definition 5.5 with non-negligible probability, provided there is a polynomial time solution to the conjugacy

search problem in braid groups. We recall the problem here for simplicity. The conjugacy search problem is: given two braids x, y where it is known that $y = axa^{-1}$ for some $a \in \mathcal{B}_N$, find a $b \in \mathcal{B}_N$ such that $y = bxb^{-1}$. Such polynomial time solutions are conjectured to exist, and there exist experimental results that support the claim, see [18, 21].

Theorem 5.7. Assume that the conjugacy search problem can be solved in polynomial time in braid groups, and that two WalnutDSA-I signatures

$$(m, H(m), G_\rho(V), s), \quad (m, H(m), G_{\rho'}(V), s')$$

with $G_\rho(V) \neq G_{\rho'}(V)$ are known to an adversary. Then the adversary can solve the REM-problem in polynomial time with non-negligible probability.

Proof. Let

$$s = \mathcal{R}\Big(X\big(E(H\big(m||G_\rho(V)\big)\big)Y\Big) = X \cdot E\big(H\big(m||G_\rho(V)\big)\big) \cdot Y$$
$$s' = \mathcal{R}\Big(X\big(E(H\big(m||G_{\rho'}(V)\big)\big)Y\Big) = X \cdot E\big(H\big(m||G_{\rho'}(V)\big)\big) \cdot Y,$$

where $=$ means equivalent in the braid group. Note that the braids X, Y only depend on the cloaking elements V. We get that

$$s \cdot (s')^{-1} = X \cdot \Big(E\big(H\big(m||G_\rho(V)\big)\big)\big(E\big(H\big(m||G_{\rho'}(V)\big)\big)\big)^{-1}\Big) \cdot X^{-1}, \tag{5.18}$$

and by our assumptions it is possible to solve for X, which means we can also solve for Y. By (5.14) we have

$$\begin{aligned}(\mathrm{Id}_N, \mathrm{Id}_{S_N}) \star Y &= (\mathrm{Id}_N, \mathrm{Id}_{S_N}) \star w' \star v_2 \\ &= \mathcal{P}(w') \star v_2 \\ &= \mathcal{P}(w'),\end{aligned}$$

which means that $\mathcal{P}(w') = \mathcal{P}(Y)$, and hence we have reversed E-multiplication. □

The forking lemma is crucial in this proof, since it allows us to use the same elements X, Y for both signatures, which enables us to reduce the problem to the conjugacy search problem.

5.8.1 Strong existential forgery

Strong existential forgery is the situation where an attacker is able to forge a second signature for a message that is different from a previously obtained signature of the same message. Since a signature

$$\mathrm{Sig} = \mathcal{R}(\kappa(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2))$$

for a message m can be multiplied by an additional cloaking element on the right to create a new valid signature for the message m, WalnutDSA is subject to strong existential forgery. However, this is not an issue if we require a forgery to be a pair (m, s) in which m has not been signed before.

Chapter 6

Attacks on Walnut

From the first publication of Walnut, many people researched and proposed attacks against the protocol, causing it to be modified several times. We will give a short presentation of some of these attacks, following roughly the descriptions given in [28], the security discussion chapter of [4], and the authors' own webpage dedicated to this purpose [31].

Definition 6.1 (Security level)**.** A secret is said to have *security level* k over a finite field $\mathbb{F}$ if the best known attack for obtaining the secret involves running an algorithm that requires at least 2^k elementary operations (addition, subtraction, multiplication, division) in the field.

We note that in the first submission to NIST, the parameter q was set to $q = 2^5 = 32$ for the 128-bit version, and $q = 2^8 = 256$ for the 256-bit version. It was specified that $\tau_1 = \tau_2 = 1$, instead of $\tau_a \tau_b = -1$. For both versions, $N = 8$ was used.

6.1 Factorization attack

The first attack against the scheme was a universal forgery attack that was published by Hart et al. in [22]. In this version of Walnut, the private key was a single braid w, and the public key was $\mathcal{P}(w)$. This turns the signature into a pure braid. The attack proceeds by collecting a set of message/signature pairs (m_i, s_i) for i in some finite indexing set I, and then use this set to create a valid signature for a message m. Specifically, we want to find a short expression in $GL(N, \mathbb{F}_q)$ for the element $h = \mathrm{Matrix}\,(\mathcal{P}(E(H(m))))$. Let $g_i = \mathrm{Matrix}(\mathcal{P}(E(H(m_i))))$ and assume that

$$h = \prod_{j=1}^{L} g_{i_j}^{e_{i_j}}, \quad i_j \in I. \tag{6.1}$$

We then get that

$$s = \prod_{j=1}^{L} s_{i_j}^{e_{i_j}},$$

which is the concatenation of the corresponding signatures, is a valid signature for m. Thus an attacker can create a forgery if they can solve a factorization problem over $GL(N, \mathbb{F}_q)$. An algorithm to solve this problem was also presented in [22], with time complexity $\mathcal{O}\left(q^{(N-1)/2}\right)$.

The authors provided several countermeasures to their attack. First of all, an increase of parameters could thwart the attack. Furthermore, they note that the signatures produced are extremely long, and that a limit on the length of the signatures would block their attack.

To counter the attack, the creators of WalnutDSA changed the private (and thus also the public) key into two different braids with nontrivial permutations. This was done to prevent the concatenation of two signatures from still beeing a valid signature. However, Ward Buellens proved in [11] that despite these changes, by doubling the size of the signatures the attack in [22] could still be used. In response, the authors of WalnutDSA set an explicit length limit on the signatures to block the attack.

6.2 Collision search attack

In [10], Simon Blackburn proposed a Pollard-rho type attack to show that the parameters in the WalnutDSA submission were slightly too small to achieve the required security levels. Specifically, he showed that in order to achieve k-bit security, we need to have

$$q^{N(N-3)-1} > 2^{2k}.$$

Blackburn presented a collision search algorithm that finds a collision for the function $\mathcal{P}\colon B_N \to GL(N, \mathbb{F}_q) \times S_N$ in $\sqrt{|GL(N, \mathbb{F}_q) \times S_N|}$ number of operations. Using this he was able to find an equivalent private key. The number of possible elements in each part of the public key is bounded above by $N!q^{N(N-1)}$, as there are $N!$ permutations in S_N, and the number of matrices over $\mathbb{F}_q$ with $(0, 0, \ldots, 0, 1)$ in the last row is $q^{N(N-1)}$. Since $N = 8$ was used for both the 128-bit and 256-bit version, we get Table 6.1. The key

Security level	q	$N!q^{N(N-1)}$	$\sqrt{\lvert N!q^{N(N-1)}\rvert}$
128	2^5	$\approx 2^{295.3}$	$\approx 2^{147.65}$
256	2^8	$\approx 2^{463.3}$	$\approx 2^{231.65}$

Table 6.1: Upper bounds on the number of elements of $\mathcal{P}(B_N)$.

part is that for the 256-bit version, the upper bound on the number of elements is not high enough to block the collision search. Blackburn also uses the estimate $q^{N(N-3)}$ for the number of possible matrices for each part of the public key, provided in an earlier version of WalnutDSA, as a lower bound, and notes that the 128-bit version might also be broken by the collision search.

In response, the authors changed the parameters from $N = 8$ to $N = 10$ for both the 128-bit and the 256-bit versions.

6.3 Encoder issues and collision search

Ward Buellens showed in [12] that the original 4-bit encoder, which used two bits to decide which braid from the set $\{g_1, g_2, g_3, g_4\}$ from (5.12) to map to, and two bits to choose an exponent, was not injective. In [8], Buellens and Blackburn also showed that it mapped to a set of braids where the matrix parts of $\mathcal{P} \circ E$ were lying in a space of only dimension 13 over $\mathbb{F}_q$. This meant that they could find messages m_1 and m_2 such that $\mathcal{P}\big(E(H(m_1))\big) = \mathcal{P}\big(E(H(m_2))\big)$ by a regular collision search algorithm. They implemented an algorithm based on an algorithm of Oorschot and Wiener [33], which takes an expected $\big|\mathcal{P}\big(E(H(\{0,1\}^*))\big)\big|^{1/2}$ evaluations of $\mathcal{P} \circ E \circ H$. Since the cardinality of the matrix part of the output space of $\mathcal{P} \circ E \circ H$ is bounded above by q^{dim}, we have an upper bound of $\big(q^{\text{dim}}\big)^{1/2}$ evaluations of $\mathcal{P} \circ E \circ H$, which for the suggested parameters is $q^{6.5}$, way below the required security levels.

To defend against the attack they proposed two solutions; either significantly increasing the parameters, or to change the encoder into an injective one along with a slight adjustment of the parameters. The first solution has the drawback that it increases the size of the private and public keys, and it also impacts the verification algorithm. Thus they recommended the second option, which has several other benefits. While not only removing the potential hazard of encoding two different messages as the same braid, changing the encoder would bump the dimension of $\mathcal{P}\big(E\big(H(\{0,1\}^*)\big)\big)$ up to $(N-2)^2+1$, which gives the collision finding algorithm an upper bound of $q^{((N-2)^2+1)^{1/2}}$ evaluations of $\mathcal{P} \circ E$. Note that the authors never give any estimate on the number of elementary operations it takes to evaluate $\mathcal{P} \circ E$, but given that this step is a part of the verification step of the scheme, and where WalnutDSA shines is in the signature verification [4], we can assume that this is fast. The experimental results also show that the attack is practical. The equation to obtain the required security level is

$$q^{\text{dim}} \geq 2^{2k}, \tag{6.2}$$

and the required parameters for 256-bit security level would then be $q = 2^8$ and $N = 10$.

In response to this attack, the encoder was changed into the one described in Section 5.6.

6.4 Subgroup chain attack

The final attack given in [8] solves the REM problem by exploiting the fact that $\mathcal{P}$ restricted to pure braids is a homomorphism $\mathcal{P}\colon P_N \to GL(N, \mathbb{F}_q)$. For $1 < k \leq N$, we define A_k to be the group of invertible $N \times N$ matrices of the form

$$A_k = \left\{ \begin{pmatrix} X & Y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \mathrm{Id}_{N-k} \end{pmatrix} \mid X \in GL(k-1, \mathbb{F}_q), Y \in \mathbb{F}_q^{k-1} \right\}.$$

We then get the following commutative diagram

$$\begin{array}{ccccccccc} \{e\} & \hookrightarrow & P_2 & \hookrightarrow & \dots & \hookrightarrow & P_N & \hookrightarrow & B_N \\ \downarrow \mathcal{P} & & \downarrow \mathcal{P} & & & & \downarrow \mathcal{P} & & \vdots \mathcal{P} \\ \{(\mathrm{Id}_N, \mathrm{Id}_{S_N})\} & \hookrightarrow & A_2 & \hookrightarrow & \dots & \hookrightarrow & A_N & \hookrightarrow & A_N \times S_N \end{array} .$$

Using this subgroup structure, the REM problem can be solved by successively reducing the problem to a collision search in a smaller subgroup. The running time of the attack is dominated by the first step, which is estimated to require $q^{N-5/2}$ E-multiplications. The authors also give a faster version of the attack, which creates longer signatures. In order to not reach the upper limit on the signature length, they suggest to only apply this method to solve the most expensive steps. In this version they use a finer chain of subgroups, which gets the expected running time down to $q^{N/2-1}$.

To block the attack, the authors have no other suggestion than to increase the parameters such that $q^{N/2-1}$ is higher than the desired security level. This would imply that for $N = 10$, q have to be increased from 2^5 to 2^{32} for 128-bit security.

In response to the attack, the authors changed q to $M31 = 2^{31} - 1$ for 128-bit, and $q = M61 = 2^{61} - 1$ for 256-bit security. This also has the advantage that $\mathbb{F}_q$ is now a prime field, which makes it possible to create cloaking elements without requiring that $\tau_1 = \tau_2 = 1$. When $\tau_a \tau_b = -1$, the running time increases by at least a factor of $\sqrt{q}\sqrt{x}$, where x is a parameter that was set to 60 for $N = 8$ in the attack code, while the Walnut designers expect $x = 96$ for $N = 10$ [1]. This brings the expected running time up to $\sqrt{x}q^{(N-1)/2}$.

6.5 Removing cloaking elements

An attack by Kotov, Menshov and Ushakov [25] on an older version of Walnut showed how to remove the cloaking elements from a signature. Their attack works purely on braids, and as such is independent of the size of q. In the old version of Walnut, there were no concealed cloaking elements, and as such the only cloaking elements present in the signature

$$\mathrm{Sig} = \mathcal{R}\Big(v_1 w^{-1} v \cdot E\big(H\big(m||G(V)\big)\big) \cdot w' v_2\Big)$$

were v, v_1 and v_2. The authors noticed that since all of these elements were of the form $w b_i^{\pm 2} w^{-1}$, they twisted the strands $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$. Since the t-values are public, an attacker could trace the strands to find all "critical letters", braids $b_i^{\pm 1}$ which swap $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$. One would then flip the power of the exponent at one of the critical letters and minimize the result (i.e removing trivial braids), since there might be a cloaking element there. Note that for an element $w b_i^{e_i} b_i^{e_i} w^{-1}$, if we replace one of the e_i with $-e_i$, we would get a trivial braid. After minimizing the new braid, one would check the length; if the new braid was significantly shorter, one would have evidence that a cloaking element was removed. For cloaking elements of the form $w b_i^{\pm 4} w^{-1}$, one would swap $b_i^{\pm 1}$ for $b_i^{\mp 3}$.

Given the extremely naive explanation above, the attack proceeds as follows: the attacker would collect k pairs (m_i, s_i), where s_i is a signature for m_i created with the

signer's private key (w, w'). One would then remove the cloaking elements from the signatures to get a set of braids

$$P_i = w^{-1} \cdot E\big(H(m_i)\big) \cdot w', \quad 1 \leq i \leq k.$$

We obtain the following equations:

$$\begin{aligned} P_1 P_2^{-1} &= w^{-1} E\big(H(m_1)\big) \left(E\big(H(m_2)\big)\right)^{-1} w \\ &\vdots \\ P_{k-1} P_k^{-1} &= w^{-1} E\big(H(m_{k-1})\big) \left(E\big(H(m_k)\big)\right)^{-1} w, \end{aligned}$$

where we know both $P_{i-1}P_i^{-1}$ and $E\big(H(m_{i-1})\big)\left(E\big(H(m_i)\big)\right)^{-1}$ for all i. Thus we have an instance of the multiple conjugacy search problem, which we have discussed before. Because the uncloaking procedure might not produce an element of the form $w^{-1}E\big(H(m_i)\big)(w')$, there might be errors in the given set. The authors describe a method to deal with this, and one would use this approach to attempt to solve the system to obtain a solution x, and compute x' as

$$x' = E\big(H(m_i)\big)^{-1} x P_i$$

for some i. The authors show that when using (x, x') to sign a message m, the signature coincides with one obtained using (w, w'), even though $(x, x') \neq (w, w')$.

The authors suggested introducing critical letters at random places in the private braids to block the attack, but noted that this approach could also fail due to their attack on the Kayawood key agreement protocol [26], also designed by SecureRF.

To patch Walnut, κ concealed cloaking elements were introduced. The designers of Walnut suggested that to achieve security level k, one would need

$$\kappa > k/\log_2(N!), \tag{6.3}$$

but due to certain assumptions that do not hold, it has been debated whether this countermeasure is effective [27].

6.6 Decomposition of products in braid groups

The last attack is given by Merz and Petit [28]. Like the previous attack, theirs work purely on the braids, and is therefore independent of q. Their approach is to find common subsequences of permutation braids in the Garside normal forms (Theorem 3.12) of the braids ABC and B when B is known. They then use these to produce braids A' and C' such that

$$A' \equiv A \pmod{\Delta^2}, \quad C' \equiv C \pmod{\Delta^2}, \quad \text{and} \quad A'C' = AC, \tag{6.4}$$

which can be used to break WalnutDSA.

Given the beauty of the mathematics, and the required countermeasures, we will work more thoroughly through the details of this attack than we did for the previous ones.

6.6.1 Normal form of products

Recall from (3.9) that for a generator braid b_i we have $b_i = \Delta b_{N-i}\Delta^{-1} = \tau(b_{N-i})$, which gives us

$$b_i\Delta = \Delta\tau(b_i).$$

We can extend this to every braid P to get $P\Delta = \Delta\tau(P)$, since τ is an automorphism, and we call $\tau(P)$ a reflection of P. We recall that $\tau^2 = \mathrm{Id}$, which implies that that Δ^2 is in the centre of the braid group. We also recall from Theorem 3.12 that the Garside normal form of a braid A is of the form

$$A = \Delta^a \cdot A_1 A_2 \cdots A_n,$$

where a is called the infimum of A, and n is the canonical length. We are interested in the normal form of the product AB, i.e the normal form of

$$\begin{aligned} AB &= \Delta^a \cdot A_1 \cdots A_n \Delta^b \cdot B_1 \cdots B_m \\ &= \Delta^{a+b} \cdot \tau^b(A_1) \cdots \tau^b(A_n) B_1 \cdots B_m \\ &= \Delta^{a+b} \cdot \tau^{b'}(A_1) \cdots \tau^{b'}(A_n) B_1 \cdots B_m, \end{aligned} \tag{6.5}$$

where $b' \equiv b \pmod 2$. This is a product of permutation braids by the following lemma.

Lemma 6.2. A braid B is a permutation braid if and only if $\tau(B)$ is.

Proof. Let B be a permutation braid. By definition we have $\Delta = BP$ for some positive braid P. We compute

$$\begin{aligned} \Delta &= BP \\ \implies \tau(\Delta) &= \tau(BP) \\ \Delta\Delta\Delta^{-1} &= \tau(B)\tau(P) \\ \Delta &= \tau(B)\tau(P), \end{aligned}$$

which shows that $\tau(B)$ is a permutation braid.

Now assume that $\tau(B)$ is a permutation braid. Then, by the point above, $\tau(\tau(B))$ is also a permutation braid. However, since $\tau^2 = \mathrm{Id}$, we get that B is a permutation braid. □

For the next lemma, recall that by the definition of the starting set $S(A)$, if $i \in S(A)$ then A can be written as $A = b_i A'$. We then have that if $i \in S(A)$, then $\tau(A) = \tau(b_i A') = \tau(b_i)\tau(A') = b_{N-i}\tau(A')$, and hence $(N-i) \in S(\tau(A))$. Recall that a product A_1A_2 being left-weighted means that $S(A_2) \subseteq F(A_1)$.

Lemma 6.3. A product A_1A_2 is left-weighted if and only if $\tau(A_1)\tau(A_2)$ is.

Proof. Assume that A_1A_2 is left-weighted, and that $\tau(A_1)\tau(A_2)$ is not. Then there exists a $j \in S(\tau(A_2)) \setminus F(\tau(A_1))$. We write

$$\begin{aligned} \tau(A_1A_2) &= \tau(A_1)\tau(A_2) \\ &= \tau(A_1) b_j b_j^{-1} \tau(A_2). \end{aligned}$$

By Lemma 6.2 and the discussion preceding Lemma 3.10, we know that $\tau(A_1)b_j$ is a permutation braid. However, because $\tau(A_1)b_j = \tau(A_1 b_{N-j})$, this means that $A_1 b_{N-j}$ must also be permutation braid, which implies $(N-j) \notin F(A_1)$. Note that $j \in S(\tau(A_2))$ implies $(N-j) \in S(A_2)$ by the discussion above. From the assumption that A_1A_2 is left-weighted we have $(N-j) \in S(A_2) \subseteq F(A_1)$, and we have reached a contradiction. Hence $\tau(A_1)\tau(A_2)$ is left-weighted.

If $\tau(A_1)\tau(A_2)$ is left-weighted, then $\tau(\tau(A_1))\tau(\tau(A_2))$ is left-weighted by the arguments above. Again, using $\tau^2 = \mathrm{Id}$, this implies that A_1A_2 is left-weighted. □

Therefore, (6.5) is a product of permutation braids, but in general not left-weighted. However, by Lemma 6.3, $\tau^{b'}(A_1)\cdots\tau^{b'}(A_n)$ is left-weighed and we get the following corollary.

Corollary 6.4. Let $\Delta^a \cdot A_1 \cdots A_n$ and $\Delta^b \cdot B_1 \cdots B_m$ be the normal forms of A and B, and let $b' \equiv b \mod 2$. Then the product $\tau^{b'}(A_1)\cdots\tau^{b'}(A_n)B_1\cdots B_m$ is left-weighted if and only if $\tau^{b'}(A_n)B_1$ is.

Since this condition will not be met for most $A, B \in \mathcal{B}_N$, the normal form of a product is not necessarily the product of the normal forms. When computing the normal form of AB, there might be created more Δ's in the process of computing the left-weighted product of $\tau^{b'}(A_1)\cdots\tau^{b'}(A_n)B_1\cdots B_m$. Moving these all the way to the left results in reflections of all the leftward permutation braids. We say that a change in a braid that is not a reflection is a *non-trivial* change.

When $\tau^{b'}(A_n)B_1$ is not left-weighted, we can find a $j \in S(B_1)\setminus F(A_n)$ and exchange A_n for $A'_n = A_n b_j$, which will still be a permutation braid. This would be a non-trivial change to A_n. For such a change to happen to A_{n-1}, we require $S(A_n) \neq S(A_n b_j)$, and that there is a $j \in S(A_n b_j) \setminus F(A_{n-1})$. This process continues inductively until some braid, say A_{n-c}, is not changed.

Since we moved a generator from B_1 to start the process, the finishing set $F(B_1)$ might differ from the finishing set of the new braid. A similar process to the one above must then be applied to the braids on the right. The discussion above gives us the following proposition.

Proposition 6.5. Let $A, B \in B_N$, and let $\Delta^a \cdot A_1 \cdots A_n$ and $\Delta^b \cdot B_1 \cdots B_m$ be their normal forms. The normal form of AB is then of the form

$$\Delta^{a+b+k} \cdot \tau^{b+k}(A_1)\cdots\tau^{b+k}(A_{n-c})X_1\cdots X_l, \tag{6.6}$$

for some integer $0 \leq c \leq n$ and permutation braids $X_1, \ldots X_l$, where $k \in \mathbb{Z}$ is the number of Δ's created when computing the left-weighted form of $\tau^b(A_1)\cdots\tau^b(A_n)\cdot B_1\cdots B_m$.

Note that

$$\Delta^k \cdot X_1\cdots X_l = \tau^b(A_{n-c+1})\cdots\tau^b(A_n)\cdot B_1\cdots B_m. \tag{6.7}$$

The point of Proposition 6.5 is that if c is less than n, the permutation braids in the normal forms of A and AB coincide up to reflection for the $n-c$ leftmost factors.

The authors of [28] rigorously prove an analogous statement for the right side of the normal forms. The details are not very important in this context, and we state the proposition without proof.

Proposition 6.6. Let $A, B \in \mathcal{B}_N$, and let $\Delta^a \cdot A_1 \cdots A_n$ and $\Delta^b \cdot B_1 \cdots B_m$ be their normal forms. The normal form of AB is then of the form

$$\Delta^{a+b+k} \cdot Y_1 \cdots Y_l \cdot B_{d+1} \cdots B_m, \tag{6.8}$$

for some integer $0 \leq d \leq m$ and permutation braids $Y_1, \ldots Y_l$, where $k \in \mathbb{Z}$ is the number of Δ's created when computing the left-weighted form of $\tau^b(A_1) \cdots \tau^b(A_n) \cdot B_1 \cdots B_m$.

Again, the point of the proposition is to notice that if $d < m$, then the normal forms of B and AB coincide on the $m - d$ rightmost factors.

Penetration distance

In [28], the authors provide experimental data for an estimate for the value c. Its numerical value is not too interesting; the key part is that for a given N, their data suggests that there exists an upper bound for the expected value of c. More precisely, they follow the work done in [19] and define the *penetration-distance* as follows.

Definition 6.7 (Penetration distance)**.** For two braids A, B, the penetration distance $\mathrm{pd}(A, B)$ for the product AB is the number of permutation braids at the end of A which undergo a non-trivial change in the normal form of the product.

They then find the penetration distances of random braids of a given length, and visualize the results. Based on the experiments, the authors propose the following conjecture for the expected penetration distance of two random braids.

Conjecture 6.8. Let $A, B \in \mathcal{B}_N$ be braid words that are picked uniformly at random from all freely reduced braid words of length k. Then there exists a $C_N \in \mathbb{N}$ such that for all k, we have

$$\mathbb{E}(\mathrm{pd}(A, B)) < C_N, \tag{6.9}$$

The conjecture implies that there is a bound on the penetration distance, regardless of the length of A and B. This has a significant impact on Propositions 6.5 and 6.6. Let A and B be braid words of canonical length n and m. If we assume that the conjecture is true, then we expect at least the $n - C_N$ leftmost permutation braids of A and AB to coincide up to reflection, as long as $n \geq C_N$. In the proof of Proposition 6.6, the authors show that C_N is also a bound for d. Because of this, we expect that at least the $m - C_N$ rightmost permutation braids of B and AB coincide whenever $m \geq C_N$.

6.6.2 Algorithm

Using the propositions above, we will show how to find braids A', C' with $A' \equiv A \pmod{\Delta^2}$ and $C' \equiv C \pmod{\Delta^2}$, such that $A'C' = AC$.

Let $A = \Delta^a \cdot A_1 \cdots A_n, B = \Delta^b \cdot B_1 \cdots B_m$ and $C = \Delta^c \cdot C_1 \cdots C_r$ be the normal forms of random, freely reduced braid words in $\mathcal{B}_N$. Assume that $m \geq C_N$, where C_N is the constant from Conjecture 6.8. From Proposition 6.5, we expect the normal form of BC to be of the form

$$\Delta^{b+c+k} \cdot \tau^{c+k}(B_1) \cdots \tau^{c+k}(B_{m-C_N}) Y_1 \cdots Y_l \tag{6.10}$$

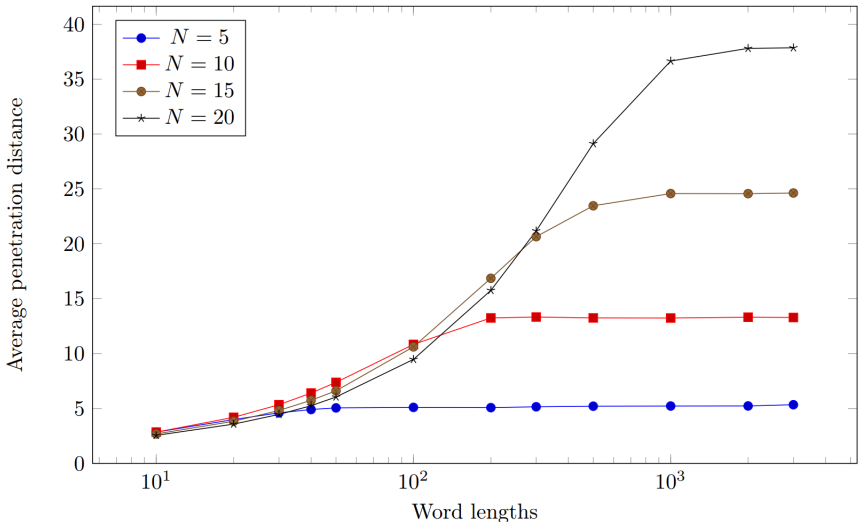


Figure 6.1: Average penetration distance after multiplication with a braid of given length on the right hand side. This figure is copied from Fig. 3 in [28].

for some permutation braids $Y_1, \ldots, Y_l$ such that

$$\Delta^k \cdot Y_1 \cdots Y_l = \tau^c(B_{j+1}) \cdots \tau^c(B_m) \cdot C_1 \cdots C_r, \tag{6.11}$$

where k is the number of Δ's that are created when computing the left-weighted form of $\tau^c(B_{j+1}) \cdots \tau^c(B_m) \cdot C_1 \cdots C_r$. The j in this equation is the index of the rightmost braid in the product $B_1 \cdots B_m$ that does not undergo a non-trivial change. By our assumptions, $j > m - C_N$.

If the part of B that was preserved into BC is of length greater than $C_N + 1$, which we can expect for $m - C_N > C_N + 1 \implies m > 2C_N + 1$, we expect by Proposition 6.6 that the normal form of $A(BC)$ is of the form

$$\Delta^{a+b+c+k+k'} \cdot X_1 \cdots X_s \cdot \tau^{c+k}(B_{C_N+1}) \cdots \tau^{c+k}(B_{m-C_N}) \cdot Y_1 \cdots Y_l. \tag{6.12}$$

Here

$$\Delta^{k'} \cdot X_1 \cdots X_s = \tau^{b+c+k}(A_1) \cdots \tau^{b+c+k}(A_n) \cdot \tau^{c+k}(B_1) \cdots \tau^{c+k}(B_{i-1}), \tag{6.13}$$

where i is the index of the leftmost braid in the product $B_1 \cdots B_m$ that does not undergo a non-trivial change. Note that the indexes $C_N + 1$ and $m - C_N$ are not strict bounds, given that the expected penetration distance is strictly less than C_N, but the expressions given here correspond better with the propositions used to derive the result.

Putting all this together lets us write the normal form of a given ABC as

$$\Delta^u \cdot X_1 \cdots X_s \cdot \tau^{c+k}(B_i) \cdots \tau^{c+k}(B_j) \cdot Y_1 \cdots Y_l, \tag{6.14}$$

where $u = a+b+c+k+k'$. As discussed above, we expect $|i-j| > 0$ when $m > 2C_N+1$. The algorithm to find A' and C' is then as follows:

1. Compute the normal forms of B and ABC.

2. Check if there is a common subsequence of permutation braids $B_{i_1} \cdots B_{i_2}$ in the normal form of B for some $1 \leq i \leq i_1 < i_2 \leq j \leq m$, in the normal form of ABC . If one is found, save the location of B_{i_1} in B and ABC and move to 3. If no sequence is found, do the same search for a sequence $\tau(B_{i-1}) \cdots \tau(B_{i-2})$ of $\tau(B)$ in the normal form of ABC. If no common subsequence is found in either of the braids, we terminate the process, and cannot find the factors. If multiple sequences are found, we do the rest of the algorithm for all of them.

3. Split B or $\tau(B)$ at B_{i_1} or $\tau(B_{i_1})$ respectively. Do the same for ABC, and denote the parts at B_I, B_{II}, ABC_I and ABC_{II}. Note that $\tau^{c+k}(B_{i_1}) \cdots \tau^{c+k}(B_{i_2})$ belong to B or $\tau(B)$ depending on whether $c+k$ leaves residue 0 or 1 modulo 2, since τ^2 is the identity. Thus, without knowing either c or k, we know the residue of $c+k$, and we write $(c+k)' \equiv c+k \pmod 2$. We now have

$$\begin{aligned}
B_I &= \Delta^b \cdot \tau^{c+k}(B_1) \cdots \tau^{c+k}(B_{i_1}) \\
B_{II} &= \tau^{c+k}(B_{i_1+1}) \cdots \tau^{c+k}(B_m) \\
ABC_I &= \Delta^{a+b+c+k+k'} \cdot X_1 \cdots X_s \cdot \tau^{c+k}(B_i) \cdots \tau^{c+k}(B_{i_1}) \\
ABC_{II} &= \tau^{c+k}(B_{i_1+1}) \cdots \tau^{c+k}(B_j) \cdot Y_1 \cdots Y_l.
\end{aligned}$$

4. Compute

$$\begin{aligned}
A' :&= ABC_I \cdot B_I^{-1} \cdot \Delta^{-(c+k)'} \\
&= \Delta^{a+b+c+k+k'} \cdot X_1 \cdots X_s \cdot \tau^{c+k}(B_{i_1-1})^{-1} \cdots \tau^{c+k}(B_1)^{-1} \cdot \Delta^{-b-(c+k)'} \\
&= \Delta^{a+b+c+k} \cdot \tau^{b+c+k}(A_1) \cdots \tau^{b+c+k}(A_n) \cdot \Delta^{-b-(c+k)'} \quad \text{by (6.13)} \\
&= \Delta^{a+c+k-(c+k)'} \cdot A_1 \cdots A_n
\end{aligned}$$

and

$$\begin{aligned}
C' :&= \Delta^{(c+k)'} \cdot B_{II}^{-1} \cdot ABC_{II} \\
&= \Delta^{(c+k)'} \cdot \tau^{c+k}(B_m)^{-1} \cdots \tau^{c+k}(B_{i_1+1})^{-1} \cdot \tau^{c+k}(B_{i_1+1}) \cdots \tau^{c+k}(B_j) \cdot Y_1 \cdots Y_l \\
&\overset{(*)}{=} \Delta^{(c+k)'} \cdot \tau^{c+k}(B_m)^{-1} \cdots \tau^{c+k}(B_{j+1})^{-1} \cdot \Delta^{-k} \tau^c(B_{j+1}) \cdots \tau^c(B_m) \cdot C_1 \cdots C_r \\
&= \Delta^{(c+k)'-k} \cdot C_1 \cdots C_r,
\end{aligned}$$

where $(*)$ is true by (6.11).

Since $a + c + k - (c + k)' \equiv a \pmod 2$ and $(c + k)' - k \equiv c \pmod 2$, we have

$$A' \equiv A \pmod{\Delta^2} \quad \text{and} \quad C' \equiv C \pmod{\Delta^2}.$$

Furthermore,

$$\begin{aligned} A'C' &= \Delta^{a+c+k-(c+k)'} \cdot A_1 \cdots A_n \cdot \Delta^{(c+k)'-k} \cdot C_1 \cdots C_r \\ &= \Delta^{a+c} \cdot \tau^{((c+k)'-k)}(A_1) \cdots \tau^{((c+k)'-k)}(A_n) \cdot C_1 \cdots C_r \\ &= \Delta^{a+c} \cdot \tau^{c}(A_1) \cdots \tau^{c}(A_n) \cdot C_1 \cdots C_r \\ &= AC. \end{aligned}$$

6.6.3 Cracking Walnut

Since a legitimately produced signature of a message m is of the form

$$\text{Sig} = W_1 \cdot E\big(H(m)\big) \cdot W_2,$$

and since every attacker can compute $E\big(H(m)\big)$, an attacker only has to collect a single valid signature/message pair before being able to create braids W_1', W_2' that satisfy (6.4). Note that given

$$W_1' \equiv W_1 \pmod{\Delta^2} \quad \text{and} \quad W_2' \equiv W_2 \pmod{\Delta^2},$$

the equation $W_1' \cdot W_2' = W_1 \cdot W_2$ implies that

$$\begin{aligned} W_1' \cdot W_2' &= W_1 \left(\Delta^2\right)^s \cdot W_2 \left(\Delta^2\right)^r \\ &= W_1 \cdot W_2 \left(\Delta^2\right)^{r+s} \implies r + s = 0, \end{aligned} \tag{6.15}$$

since Δ^2 is in the center of the braid group.

Given that $\text{Sig} = W_1 \cdot E\big(H(m)\big) \cdot W_2$ is a valid signature for the message m, we have that $\text{Sig}' = W_1 \cdot E\big(H(m')\big) \cdot W_2$ is a valid signature for the message m'. Furthermore, from (6.15) we get

$$\begin{aligned} W_1' \cdot E\big(H(m')\big) W_2' &= W_1 \left(\Delta^2\right)^s \cdot E\big(H(m')\big) \cdot W_2 \left(\Delta^2\right)^r, \\ &= W_1 \cdot E\big(H(m')\big) \cdot W_2 \left(\Delta^2\right)^{r+s} \\ &= W_1 \cdot E\big(H(m')\big) \cdot W_2, \end{aligned}$$

and thus the attacker can use W_1', W_2' to forge valid signatures for any message m'.

In the algorithm, one has to compute the normal form of Sig and $E(H(m))$. Using the algorithm of Thurston in [16], this is done in time $\mathcal{O}\left(|P|^2 N \log N\right)$, where $|P|$ is the number of permutation braids in a given decomposition of a braid, not necessarily in normal form. As explained in [28], the running time of the entire algorithm is dominated by this step, and since $|E(H(m))| \leq |\,\text{Sig}\,|$, the running time of the algorithm is $\mathcal{O}\left(|\,\text{Sig}\,|^2 N \log N\right)$.

As a countermeasure to the attack, the authors propose two changes. The first one is to simply increase the parameter N. However, from their experimental data it seems like we

require $N \approx 50$ for 128-bit, and $N \approx 85$ for 256-bit security, a massive increase from the current parameters.

Rather than doing the above, they suggest putting concealed cloaking elements into the encoding $E\big(H(m)\big)$, as this would impact the number of common subsequences in the encoding and in the signature. They suggest adding as many as 30 and 60 elements for the two security levels, but note that there might be a need for more given the uncloaking procedure discussed in the previous section.

This solution was also found independently by the Walnut designers, and in [1] they found that inserting a concealed cloaking element every 7-12 generators will block the attack.

Chapter 7

Conclusion

On January 30, 2019, NIST announced the candidates that were selected for round two of the standardization process, and WalnutDSA was not one of them. Though no reason was given for whether a scheme was chosen or not, we can make some educated guesses as to why Walnut did not make the cut.

7.1 Poor design choices

First off, the parameters were too small. Not being able to block against the collision attack presented by Blackburn [10] does not give a lot of credibility to your product.

Similarly, the message encoder not being injective seems like something the team should have realized from the start. The dimension problem exploited by Buellens and Blackburn [8], while somewhat harder to discover, was also a major weakness.

After the attack presented in [22], both the private and the public key were changed from one to two braids. Ward Buellens then showed in [11] that this change was not enough to block the attack, which prompts a discussion of whether the change should be reverted or not, as it also doubled the size of the keys.

In an early version, an estimate on the number of possible matrices in each part of the public key was $q^{N(N-3)}$. This was changed to $q^{N(N-1)+1}$ in a later version of the Walnut documentation, where it was stated that "The search space for all such matrices is again the square of this lower bound". Given how the colored Burau representation and E-multiplication is designed, the bottom row of a matrix in the image will always equal $(0, 0, \ldots, 0, 1)$. This gives a strict upper bound of $q^{N(N-1)}$ possible matrices in each part of the public key. Thus their stated lower bound is greater than the actual upper bound. It is mentioned several times in [4] that the bottom row is all zero except for the last element, but they do not remark that the last element is always equal to 1. Again, this mistake does not bring credibility to the authors.

While none of the points above provide a clear reason as to why Walnut did not get chosen, they give an impression of a team that did not fully understand the implications of the choices they have made during the construction of the scheme, or when designing the countermeasures.

7.2 Flaws in security proof

A more concrete point is the lack of a description for the simulator in the security proof given in [4]. The only description given for the simulator is that it answers a signature query with a valid signature, without specifying how this would be done. In the security proof it is noted that the simulator does not know the private key, but for this to actually mean something there has to be a difference in how a signing query is answered when one has the private key, and when one does not. Therefore the proof only works for key-only attacks.

This was first pointed out by Buellens [13], and discussed over several of the following pages. The authors refer to the article "Security Proofs for Signature Schemes" by Pointcheval and Stern [30] as a model for their security proof. However, this article has an entire page devoted to describing the simulator. Buellens also remarks the following:

1. Given a hash function H with too short output length l, we know that the scheme is insecure, as one might be able to find collisions $H(m) = H(m')$ where $m \neq m'$. This would imply that a valid signature for m is also a valid signature for m'.

2. There has to be an upper limit to the length of signatures, or else the scheme is insecure by an attack by Buellens [8].

Since the scheme is not secure if these considerations are not taken into account, the security proof should fail somewhere, but neither of these quantities are mentioned in the proof.

To further add to this, the security proof makes a rather strong assumption that the signatures produced by the forger $\mathcal{F}$ lie in the set

$$\mathrm{DC}_{m,V,H,G} := \left\{ \mathcal{R}\Big(X \cdot E\big(H\big(m||G(V)\big)\big) \cdot Y \Big) \,\middle|\, X, Y \in B_N \right\},$$

instead of dealing with signatures of any form.

Bibliography

[1] I. Anshel et al. *Defeating the Hart et al, Beullens-Blackburn, Kotov-Menshov-Ushakov, and Merz-Petit Attacks on WalnutDSA(TM)*. Cryptology ePrint Archive, Report 2019/472. https://eprint.iacr.org/2019/472. 2019.

[2] I. Anshel et al. "Key agreement, the Algebraic EraserTM, and lightweight cryptography". In: *Contemporary Mathematics* 418 (2007), pp. 1–34.

[3] I. Anshel et al. "New Key Agreement Protocols in Braid Group Cryptography". In: *Topics in Cryptology — CT-RSA 2001*. Ed. by D. Naccache. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 13–27. ISBN: 978-3-540-45353-6.

[4] I. Anshel et al. *WalnutDSA(TM): A Quantum-Resistant Digital Signature Algorithm*. Cryptology ePrint Archive, Report 2017/058. 2017. URL: https://eprint.iacr.org/2017/058.

[5] M. Anshel, M. Anshel, and D. Goldfeld. "An Algebraic Method For Public-Key Cryptography". In: *Mathematical Research Letters* 6 (1999), pp. 287–291.

[6] E. Artin. "Theory of braids". In: *Ann. of Math* 48.2 (1947), pp. 101–126.

[7] M. Bellare and G. Neven. "Multi-signatures in the Plain public-Key Model and a General Forking Lemma". In: *Proceedings of the 13th ACM Conference on Computer and Communications Security*. CCS '06. Alexandria, Virginia, USA: ACM, 2006, pp. 390–399. ISBN: 1-59593-518-5. DOI: 10.1145/1180405.1180453. URL: http://doi.acm.org/10.1145/1180405.1180453.

[8] W. Beullens and S. R. Blackburn. "Practical attacks against the Walnut digital signature scheme". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2018, pp. 35–61.

[9] J. Birman, K. H. Ko, and S. J. Lee. "A new approach to the word and conjugacy problems in the braid groups". In: *Advances in Mathematics* 139.2 (1998), pp. 322–353.

[10] S. R. Blackburn. *OFFICIAL COMMENT: WalnutDSA*. Page 7-12. Jan. 2018. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/WalnutDSA-official-comment.pdf (visited on 04/17/2019).

[11] W. Buellens. *OFFICIAL COMMENT: WalnutDSA*. Page 2-4. Jan. 2018. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/WalnutDSA-official-comment.pdf (visited on 04/12/2019).

[12] W. Buellens. *OFFICIAL COMMENT: WalnutDSA*. Page 15. Jan. 2018. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/WalnutDSA-official-comment.pdf (visited on 04/12/2019).

[13] W. Buellens. *OFFICIAL COMMENT: WalnutDSA*. Page 1. Jan. 2018. URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/WalnutDSA-official-comment.pdf (visited on 04/12/2019).

[14] P. Dehornoy. "A fast method for comparing braids". In: *advances in mathematics* 125.2 (1997), pp. 200–235.

[15] E. Elrifai and H. Morton. "Algorithms for positive braids". In: *Quarterly Journal of Mathematics* 45 (Dec. 1994). DOI: 10.1093/qmath/45.4.479.

[16] D. B. Epstein. *Word processing in groups*. AK Peters/CRC Press, 1992.

[17] F. A. Garside. "THE BRAID GROUP AND OTHER GROUPS". In: *The Quarterly Journal of Mathematics* 20.1 (1969), pp. 235–254. DOI: 10.1093/qmath/20.1.235.

[18] V. Gebhardt. "A new approach to the conjugacy problem in Garside groups". In: *Journal of Algebra* 292.1 (2005), pp. 282–302.

[19] V. Gebhardt and S. Tawn. "Normal forms of random braids". In: *Journal of Algebra* 408 (2014), pp. 115–137.

[20] S. Goldwasser, S. Micali, and R. L. Rivest. "A digital signature scheme secure against adaptive chosen-message attacks". In: *SIAM Journal on Computing* 17.2 (1988), pp. 281–308.

[21] A. Groch, D. Hofheinz, and R. Steinwandt. "A practical attack on the root problem in braid groups". In: *Contemporary Mathematics* 418 (2006), p. 121.

[22] D. Hart et al. “A Practical Cryptanalysis of WalnutDSA”.
In: *Public-Key Cryptography – PKC 2018*. Ed. by M. Abdalla and R. Dahab.
Cham: Springer International Publishing, 2018, pp. 381–406.
ISBN: 978-3-319-76578-5.

[23] D. Hofheinz and R. Steinwandt.
“A Practical Attack on Some Braid Group Based Cryptographic Primitives”.
In: *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*. PKC ’03.
London, UK, UK: Springer-Verlag, 2003, pp. 187–198. ISBN: 3-540-00324-X.
URL: http://dl.acm.org/citation.cfm?id=648120.746922.

[24] K. H. Ko et al. “New public-key cryptosystem using braid groups”.
In: *Annual International Cryptology Conference*. Springer. 2000, pp. 166–183.

[25] M. Kotov, A. Menshov, and A. Ushakov.
“An attack on the Walnut digital signature algorithm”.
In: *Designs, Codes and Cryptography* (2018), pp. 1–20.

[26] M. Kotov, A. Menshov, and A. Ushakov.
“Attack on Kayawood Protocol: Uncloaking Private Keys.”
In: *IACR Cryptology ePrint Archive* 2018 (2018), p. 604.

[27] A. Menshov. *OFFICIAL COMMENT: WalnutDSA*. Page 41.
URL: https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/WalnutDSA-official-comment.pdf (visited on 04/12/2019).

[28] S.-P. Merz and C. Petit. “Factoring Products of Braids via Garside Normal Form”.
In: *IACR International Workshop on Public Key Cryptography*. Springer. 2019, pp. 646–678.

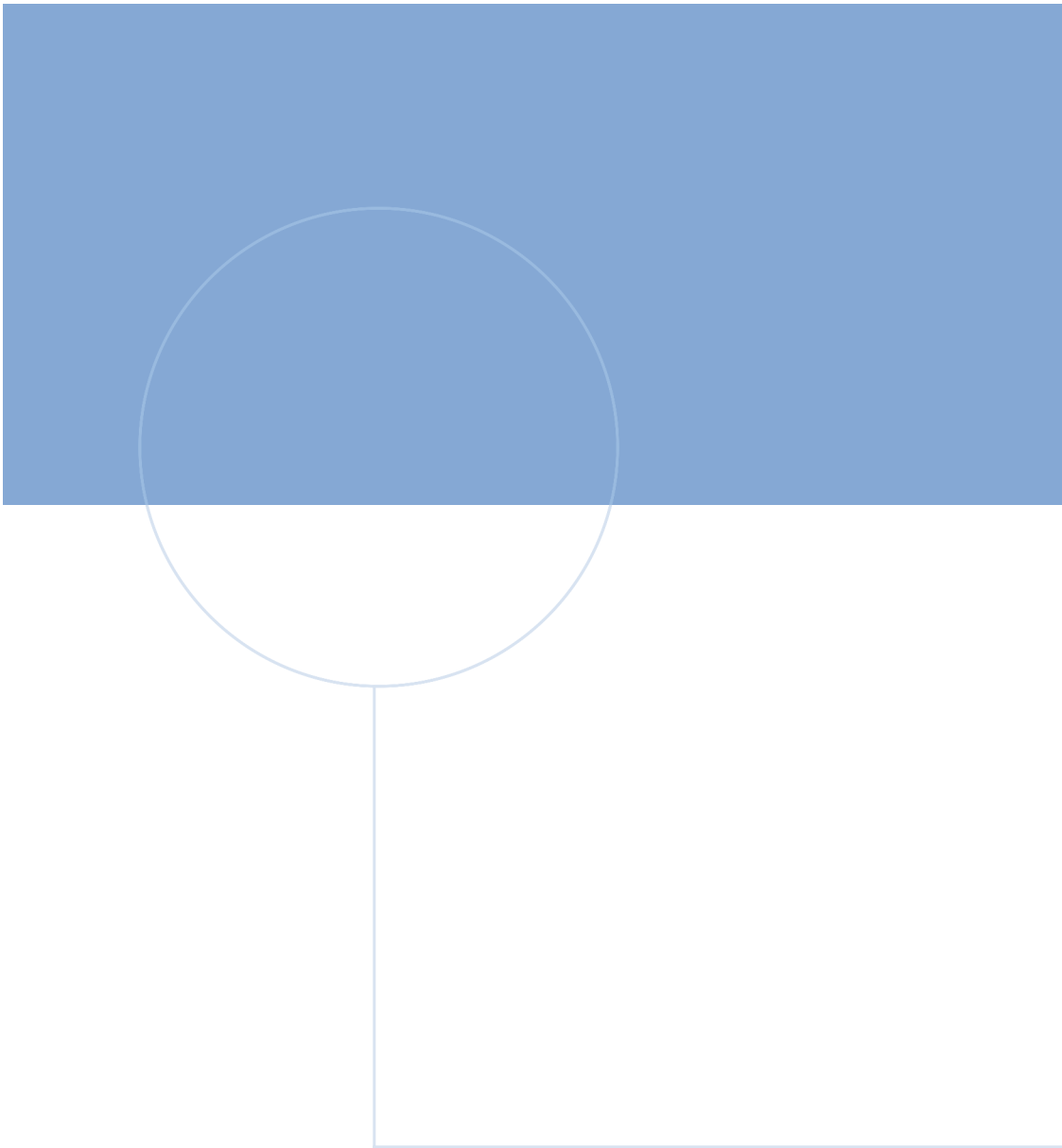
[29] D. Pointcheval and J. Stern.
“Security arguments for digital signatures and blind signatures”.
In: *Journal of cryptology* 13.3 (2000), pp. 361–396.

[30] D. Pointcheval and J. Stern. “Security proofs for signature schemes”.
In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1996, pp. 387–398.

[31] SecureRF. *NIST Community Analysis with SecureRF Responses*. 2019.
URL: https://www.securerf.com/nist-community-analysis.

[32] P. W. Shor.
“Algorithms for quantum computation: Discrete logarithms and factoring”.
In: *Proceedings 35th annual symposium on foundations of computer science*.
Ieee. 1994, pp. 124–134.

[33] P. C. Van Oorschot and M. J. Wiener.
“Parallel collision search with cryptanalytic applications”.
In: *Journal of cryptology* 12.1 (1999), pp. 1–28.



NTNU
Norwegian University of
Science and Technology