Factoring Products of Braids via Garside Normal Form*

Simon-Philipp Merz[1] and Christophe Petit[2]

[1] Royal Holloway, University of London, UK
[2] University of Birmingham, UK

Abstract. Braid groups are infinite non-abelian groups naturally arising from geometric braids. For two decades they have been proposed for cryptographic use. In braid group cryptography public braids often contain secret braids as factors and it is hoped that rewriting the product of braid words hides individual factors. We provide experimental evidence that this is in general not the case and argue that under certain conditions parts of the Garside normal form of factors can be found in the Garside normal form of their product. This observation can be exploited to decompose products of braids of the form ABC when only B is known.
Our decomposition algorithm yields a universal forgery attack on WalnutDSA$^{\text{TM}}$, which is one of the 20 proposed signature schemes that are being considered by NIST for standardization of quantum-resistant public-key cryptography. Our attack on WalnutDSA$^{\text{TM}}$ can universally forge signatures within seconds for both the 128-bit and 256-bit security level, given one random message-signature pair. The attack worked on 99.8% and 100% of signatures for the 128-bit and 256-bit security levels in our experiments.
Furthermore, we show that the decomposition algorithm can be used to solve instances of the conjugacy search problem and decomposition search problem in braid groups. These problems are at the heart of other cryptographic schemes based on braid groups.

1 Introduction

Continuous progress in quantum computing and the prospect of large scale quantum computers necessitate the development of quantum-resistant cryptographic algorithms. Currently, the security of most widespread algorithms relies on the hardness of the discrete logarithm problem, the elliptic-curve discrete logarithm problem or the integer factorization problem. All of these mathematical problems can be solved using Shor's quantum algorithm [42]. Even though quantum computers with sufficient processing power to pose a threat to current cryptographic applications presumably do not yet exist, researchers, intelligence agencies and the industry aspire to develop cryptographic systems that remain safe once such devices come into being. Current approaches to attain quantum-resistance include cryptography based on codes, isogenies, lattices and multivariate polynomials over finite fields [19, 36, 38, 44]. Another approach are cryptographic systems based on non-abelian groups [22]. Indeed no quantum algorithm to solve the hidden subgroup problem (the core problem solved by Shor's algorithm for finite abelian groups) is known for general non-abelian groups.

The conjugacy search problem is a fundamental decision problem in combinatorial group theory.

Definition 1. Given two braids $X, Y \in B_N$ where $Y = C \cdot X \cdot C^{-1}$ for some $C \in B_N$, the *conjugacy search problem* (CSP) in braid groups is to find $\tilde{C} \in B_N$ such that $Y = \tilde{C} \cdot X \cdot \tilde{C}^{-1}$.

* The full version can be found in the IACR eprint archive as article 2018/1142.

The asserted computational difficulty of the CSP and its variations has inspired many cryptographic primitives on non-abelian groups such as [3, 33].

To establish standards for quantum-secure cryptography [40] the National Institute of Standards and Technology (NIST) is currently evaluating public-key algorithms [40]. One of the 20 signature schemes being considered for standardisation is WalnutDSA$^{\text{TM}}$ [6] operating on braid groups.

NIST's ongoing standardization project and thus the potential for widespread use of WalnutDSA$^{\text{TM}}$ and other braid group algorithms make a thorough security analysis and understanding of the braid group vital. WalnutDSA$^{\text{TM}}$ has been analysed before [10, 29, 35] bringing some weaknesses of the signature scheme to light. However, the attacks could be thwarted by increasing parameters and slightly changing the protocol [41]. A fundamental assumption underlying the security of WalnutDSA$^{\text{TM}}$ is that individual factors in a product of three braids are "obfuscated" when they are presented in some normal form.

Our contribution: In this paper, we describe how the Garside normal forms of factors relate to the Garside normal form of their product. Together with an observation based on experiments, we use this to locate single factors in a product of braids and to decompose certain products in braid groups. More precisely, we give an algorithm that can recover the factors of a product $ABC \in B_N$ up to the centre of the group when only B is known.

Signatures of WalnutDSA$^{\text{TM}}$ can be written as a braid word $W_1 \cdot E \cdot W_2$, where W_1 and W_2 are secret braids and E is a deterministic encoding of the message. The product is presented rewritten, e.g. in normal form, with the explicit aim of obfuscating individual factors. Our observations imply that W_1 and W_2 can in fact be efficiently recovered up to the centre of the group. Replacing E by the encoding of any other message yields a new universal forgery attack that works within seconds on most random message-signature pairs.

Related work: Braid groups have been suggested for cryptographic purposes for two decades [22] and protocols such as the Anshel-Anshel-Goldfeld key exchange [4] and Ko et al.'s protocol [33] have been studied extensively. A newer protocol sharing some design components with WalnutDSA$^{\text{TM}}$ is the Algebraic Eraser [5]. This scheme and the Anshel-Anshel-Goldfeld key exchange have been subject to numerous attacks which were mostly based on representation theory [8, 9, 31] or on a length-based approach [30, 39]. Yet, the same cryptanalytic techniques do not seem to apply to WalnutDSA$^{\text{TM}}$.

Considerable work has been devoted towards a solution of the conjugacy search problem (CSP) in braid groups. Apart from heuristic approaches such as the previously mentioned length-based attacks, the most successful approaches use summit sets [13, 24, 25].

Responsible Disclosure Process: We provided the designers of WalnutDSA$^{\text{TM}}$ with the details of our attack on the 20th of August. They acknowledged that the attack works. To prevent malicious use of our attack on the signature scheme or similar products by SecureRF, we agreed to postpone the publication of our findings until the 21st of November.

Outline: In Section 2, we provide preliminary results on braid groups and the Garside normal form. In Section 3 we present the current instantiation of WalnutDSA$^{\text{TM}}$ and how it was modified to thwart previous attacks. Section 4 gives our algorithm to recover the factors of a braid ABC presented by its normal form when the braid B is known. In Section 5 we describe our attack on

WalnutDSA$^{\text{TM}}$ and discuss potential countermeasures. Section 6 shows how the decomposition algorithm can be used to solve instances of the conjugacy search problem. We conclude our work in Section 7.

2 Braid Groups

This section provides preliminary mathematical background on braid groups. In Section 2.1 we define braid groups and provide their algebraic presentation. Section 2.2 defines the colored Burau representation of braid groups which is needed to explain WalnutDSA$^{\text{TM}}$ but not essential for the understanding of our contribution. In Section 2.3 we define the Garside normal form. A reader familiar with braid groups and the Garside normal form may proceed to Section 3.

2.1 Artin Presentation

Let N be a positive integer and let B_N denote the braid group on N strands introduced by Emil Artin [7]. Geometrically, the elements of a braid group are the equivalence classes of N strands under ambient isotopy, i.e. we consider two braids the same if we can distort one into the other without breaking any strand. Artin proved that B_N is indeed a group with presentation

$$B_N = \left\langle b_1, \ldots, b_{N-1} \;\middle|\; \begin{array}{l} b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1} \\ b_i b_j = b_j b_i \text{ for } |i-j| \geq 2 \end{array} \right\rangle, \tag{1}$$

where the group operation is given by concatenation of the strings. Thus, we can represent any braid of B_N as a finite, non-unique word in the so called *Artin generators* b_i. Imagining our strands lying in a plane and numbering the strands from left to right, the generator b_i corresponds to the i-th strand crossing over the $(i+1)$-th strand.

Figures 1 and 2 illustrate the relations given in Presentation (1).

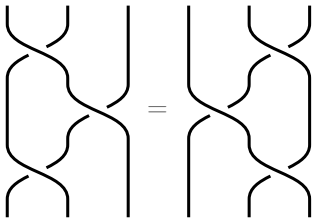

Fig. 1: $b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1}$

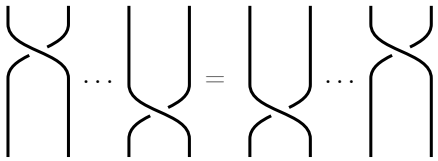

Fig. 2: $b_i b_j = b_j b_i$, if $|i-j| \geq 2$

Note that there is a natural homomorphism sending elements of B_N to the induced permutations in the symmetric group S_N. More precisely, each Artin generator b_i is sent to the transposition $\pi_i := (i, i+1)$. For some braid word $b_{i_1}^{\epsilon_1} \ldots b_{i_k}^{\epsilon_k}$ the induced permutation is $\pi_{i_1}^{\epsilon_1} \ldots \pi_{i_k}^{\epsilon_k}$. Since the corresponding permutations respect the relations in Presentation (1), sending braids to their induced permutations is a well-defined homomorphism. Clearly, this homomorphism from B_N to S_N is surjective. Braids in the kernel, i.e. braids inducing the identity permutation, are called *pure braids*.

It is well known that the group of *pure* braids can be generated by g_{ij}, $1 \leq i < j \leq N$ [12], where

$$g_{ij} := b_{j-1} \cdot b_{j-2} \cdot \cdots \cdot b_{i+1} \cdot b_i^2 \cdot b_{i+1}^{-1} \cdot \cdots \cdot b_{j-2}^{-1} \cdot b_{j-1}^{-1}. \tag{2}$$

The generator g_{ij} may be depicted geometrically as braid where the j-th string passes behind the strings $(j-1),\dots,(i+1)$, in front of the i-th string and then behind the strings $i,\dots,j-1$ back to the j-th position.

2.2 Colored Burau Representation

The colored Burau representation of braid groups which we will describe in this section is used to define WalnutDSA$^{\text{TM}}$ and its underlying problem. A reader who is mainly interested in the structure being preserved in products of Garside normal forms may want to skip this section.

Let q be the power of a prime and let $\mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}]$ be the ring of Laurent polynomials with coefficients in the finite field $\mathbb{F}_q$ with q elements. There exists an action of S_N on $\mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}]$, where a permutation acts on the indices of the variables of the Laurent polynomial. That is, for every $\sigma \in \mathsf{S}_N$ and $f \in \mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}]$

$$f(t_1,\dots,t_N) \mapsto {}^{\sigma}f = f(t_{\sigma(1)},\dots,t_{\sigma(N)})$$

The action of S_N extends to $\mathsf{GL}_N(\mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}])$ by applying it entry-wise. For $\sigma \in \mathsf{S}_N$ and $M \in \mathsf{GL}_N(\mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}])$, we denote the action by $M \mapsto {}^{\sigma}M$.

The colored Burau matrices of each Artin generator are defined as follows [6]:

$$\mathsf{CB}(b_1) := \begin{pmatrix} -t_1 & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix} \quad \text{and} \quad \mathsf{CB}(b_i) := \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & t_i & -t_i & 1 & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$

where the t_i are written in the i-th row for $2 \le i \le N-1$. Equipping the semidirect product $\mathsf{GL}_N(\mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}]) \rtimes \mathsf{S}_N$ with the operation

$$(M_1,\sigma_1)\cdot(M_2,\sigma_2) = (M_1 \cdot {}^{\sigma_1}M_2, \sigma_1\sigma_2),$$

one obtains a group and one can check that the map

$$\begin{aligned} \Phi : B_N &\to \mathsf{GL}_N\big(\mathbb{F}_q[t_1^{\pm 1},\dots,t_N^{\pm 1}]\big) \rtimes \mathsf{S}_N \\ b_i &\mapsto \big(\mathsf{CB}(b_i), \pi_i\big), \end{aligned} \tag{3}$$

where π_i denotes the transposition $(i,i+1) \in \mathsf{S}_N$, extends to a group homomorphism. This group homomorphism is called *colored Burau representation* of B_N [16].

2.3 Garside Normal Form

A *normal form* in a group is a canonical way to represent the elements and thus it provides an opportunity to compare them.

Garside was the first to develop a normal form for braid groups [23] which was improved most notably by Thurston [21] and Elrifai and Morton [20] leading to what is known as the *Garside*

normal form today. For further normal forms in braid groups see [11, 15, 18].

In this section we reproduce some results that led to the development of the Garside normal form to introduce terminology necessary for the explanation of our observation in Section 4.

Let B_N^+ denote the monoid of *positive braids* in B_N which are the braids that can be written as a product of positive powers of Artin generators. This is a well-defined monoid as all the defining relations in the presentation of braid groups (1) contain only positive powers of Artin generators.

We denote Garside's "*fundamental braid*" [23] by Δ. Recall that this braid is the unique positive braid in which any two strands cross exactly once and it is of central importance in the Garside normal form. We recall some properties of the fundamental braid due to Garside [23].

Proposition 2. *Let B_N be the braid group on N strands. For $i = 1, \dots, N-1$, we have*

$$b_i \Delta = \Delta b_{N-i}.$$

In particular Δ^2 commutes with every generator and lies in the centre of B_N. In fact, the centre of B_N is cyclic and generated by Δ^2.

Remark 3. Let τ be the inner automorphism of B_N conjugating elements with Δ, i.e.

$$\begin{aligned} \tau : B_N &\to B_N \\ \beta &\mapsto \Delta \beta \Delta^{-1} \end{aligned}$$

Let $W = b_{i_1}^{\epsilon_1} \dots b_{i_k}^{\epsilon_k} \in B_N$ with $\epsilon_j \in \{0, 1\}$. Then the previous Proposition implies

$$\tau(W) = \Delta W \Delta^{-1} = \tau(b_{i_1})^{\epsilon_1} \dots \tau(b_{i_k})^{\epsilon_k} = b_{N-i_1}^{\epsilon_1} \dots b_{N-i_k}^{\epsilon_k}.$$

In particular, τ^2 is the identity automorphism. We will continue to denote this automorphism by τ and call it the *reflection in B_N* throughout this paper.

Proposition 4. *[23] For any generator b_i, $i = 1, \dots, N-1$, we can find positive braids x_i and $y_i \in B_N^+$ such that*

$$b_i x_i = \Delta = y_i b_i.$$

An explicit description of the braids x_i, y_i is given at the same place. Together with Proposition 2 this observation can be used to rewrite any representation of an element of B_N efficiently in the form $\Delta^r P$, where $r \in \mathbb{Z}$ and P is a positive braid that cannot be written as a positive word containing Δ as a subword. Listing all possible words P and choosing the lexicographically minimal one for P yields the initial normal form due to Garside. This algorithm has exponential running time in the number of strands N and the braid length, so it is not completely satisfactory from a computational point of view. However, we have the following natural partial order in the monoid of positive braids.

Definition 5. Let $a, b \in B_N^+$. We write $a \leq b$ if $ac = b$ for some $c \in B_N^+$. We say a is a *prefix* of b. This is a partial order invariant under left multiplication, i.e. $a \leq b$ implies $da \leq db$ for all $d \in B_N^+$.

Let 1 denote the identity in B_N. We see that $1 \leq A$ if and only if $A \in B_N^+$.

Given a partial order as in Definition 5 one may wonder whether there is a greatest common prefix in some sense.

Proposition 6. *[23] For any two elements $a, b \in B_N^+$ there exists a unique element d such that $d \leq a$, $d \leq b$ and that $d' \leq d$ for every common prefix d' of a and b.*

Definition 7. Using the same notation as in the previous proposition, we call d the *greatest common divisor (gcd)* of a and b and we write $d = a \wedge b$.

Elrifai and Morton [20] and Thurston [21] independently developed two different algorithms to compute the normal form of a braid in polynomial time building on top of Garside's results. The centrepiece of their work is to consider the following braids.

Definition 8. The positive prefixes of Δ are called *permutation braids*, i.e. $A \in B_N$ is a permutation braid if and only if $1 \leq A \leq \Delta$.

Permutation braids are exactly those positive braids with any pair of strands crossing at most once and thus uniquely determined by the permutation they induce.

Instead of listing exponentially many representatives and choosing the lexicographically minimal one, the idea of Thurston, Elrifai and Morton was to write a braid word β as a product of permutation braids

$$\beta = \Delta^r A_1 \cdots A_k,$$

where uniqueness is achieved by requiring each letter to appear as far to the left as possible.

Definition 9. A product of permutation braids $A_i A_{i+1}$ is called *left-weighted* if $A_i A_{i+1} \wedge \Delta = A_i$.

That is, if we move any crossing from A_{i+1} to A_i the resulting braid would not be a permutation braid anymore. This allows us to formulate the Garside left normal form.

Theorem 10 (Garside left normal form). *Every braid β can be represented uniquely by a braid word*

$$\Delta^r A_1 \cdots A_k,$$

where $r \in \mathbb{Z}$, $1 < A_i < \Delta$ and $A_i A_{i+1}$ is a left-weighted product for $1 \leq i \leq k$.

Definition 11. Consider the notation of the preceding Theorem 10. We call the integer k the *canonical length* of β and the integer r the *infimum* of β.

For details on the algorithms to compute the Garside left normal form we refer to [20, 21]. Using the approach of Elrifai and Morton the normal form of some given positive braid word $b_{i_1} \dots b_{i_k}$ can be computed in time $\mathcal{O}(k^2 N)$, where k is the number of Artin generators of the braid word given. Thurston's alternative but equivalent solution computes the left normal form of a positive braid word given as a product of permutation braids $A_1 \dots A_{k'}$ with time complexity $\mathcal{O}(k'^2 N \log N)$ [21]. Note, this might be faster than the previous algorithm as most permutation braids are a product of multiple Artin generators.

We want to point out that similar observations as the ones we will state in Section 4 for the Garside normal form hold for other normal forms such as the Birman-Ko-Lee (BKL) normal form as well. In particular, structure in the BKL normal form can be exploited directly to attack WalnutDSA$^{\text{TM}}$ or solve instances of the conjugacy search problem too. However, using the Garside normal form turned out to be slightly more efficient in our experiments which is why we will mean the Garside left normal form when talking about left normal forms for the remainder of this paper.

3 WalnutDSA™

WalnutDSA™ is a digital signature scheme operating on braid groups. It was proposed by Anshel, Atkins, Goldfeld and Gunnels [6]. This Section summarizes the newest version of the signature scheme. In Section 3.1 we define E-Multiplication and cloaking elements and state the underlying hardness assumption of WalnutDSA™. The section is not necessary to understand our attack, but these basic building blocks are needed to define the signature scheme itself. Section 3.2 provides details about parameters used and the signature generation and validation. Finally, we will give a brief overview of previous work on WalnutDSA™ showing that our approach is fundamentally disparate in Section 3.3.

3.1 E-Multiplication™ and Cloaking Elements

E-Multiplication was first introduced as a one-way function [5] and it is a foundation of WalnutDSA™. Let $\mathbb{F}_q^\times$ denote the non-zero elements of the finite field $\mathbb{F}_q$. An N-tuple of the form

$$\tau = (\tau_1, \ldots, \tau_N) \in (\mathbb{F}_q^\times)^N$$

will be called "*T-values*" in the following. Given such a tuple, we can evaluate any Laurent polynomial $\mathbb{F}_q[t_1^{\pm 1}, \ldots, t_N^{\pm 1}]$ at τ, denoted $\downarrow_\tau$:

$$\begin{aligned} \downarrow_\tau \colon \mathbb{F}_q[t_1^{\pm 1}, \ldots, t_N^{\pm 1}] &\to \mathbb{F}_q \\ f &\mapsto f(\tau_1, \ldots, \tau_N) \end{aligned}$$

Similarly, we can evaluate any matrix $M \in \mathsf{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \ldots, t_N^{\pm 1}])$ to $M \downarrow_\tau$ by doing so entrywise.

E-Multiplication is a right action of the colored Burau group $\mathsf{GL}_N(\mathbb{F}_q[t_1^{\pm 1}, \ldots, t_N^{\pm 1}]) \rtimes \mathsf{S}_N$ on $\mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N$. We will follow the notation of [6] denoting E-Multiplication with a star: $\star$.

For a single Artin generator b_i, E-Multiplication is defined as

$$(M, \sigma) \star \Phi(b_i) := \Big(M \cdot {}^{\sigma}(\mathsf{CB}(b_i)) \downarrow_\tau,\ \sigma \cdot \pi_i \Big),$$

where $\pi_i = (i, i+1) \in \mathsf{S}_N$ and Φ is the map given in Equation (3). For a general braid β represented by $b_{i_1}^{\epsilon_1} \ldots b_{i_k}^{\epsilon_k}$ with $\epsilon_j \in \{-1, 1\}$, we define E-Multiplication inductively left-to-right as

$$(M, \sigma) \star \Phi(\beta) := (M, \sigma) \star \left(\mathsf{CB}(b_{i_1})^{\epsilon_1}, \pi_{i_1}^{\epsilon_1}\right) \star \cdots \star \left(\mathsf{CB}(b_{i_k})^{\epsilon_k}, \pi_{i_k}^{\epsilon_k}\right).$$

Remark 12. Following the notation of [6], we write $(M, \sigma) \star \beta$ instead of $(M, \sigma) \star \Phi(\beta)$ for $\beta \in B_N$. Moreover, we denote by $\mathcal{P}$ the map

$$\begin{aligned} \mathcal{P} : B_N &\to \mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N \\ \beta &\mapsto (\mathrm{Id}, \mathrm{id}) \star \beta. \end{aligned} \tag{4}$$

The security of WalnutDSA™ is based on the computational hardness assumption of the *reversing E-Multiplication* (REM) problem.

Definition 13. Given an ordered pair $(M, \sigma) \in \mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N$ such that $(M, \sigma) = (\mathrm{Id}, \mathrm{id}) \star \beta$ for some braid $\beta \in B_N$. The *reversing E-Multiplication* (REM) problem is to find a braid β' such that $(\mathrm{Id}, \mathrm{id}) \star \beta' = (M, \sigma)$.

In particular inverting the map given in (4) is assumed to be hard. Reversing E-Multiplication is enough to break WalnutDSA$^{\text{TM}}$, indeed we will see that the ability to solve the REM problem allows to forge the signature of one message and that solving two instances of the REM problem allows the recovery of the private key from the public key.

However, our attack on WalnutDSA$^{\text{TM}}$ bypasses the problem of reversing E-Multiplication. We will see that our attack works solely on braids and is therefore independent of the colored Burau representation and of the size q of the underlying field $\mathbb{F}_q$.

Another basic building block of WalnutDSA$^{\text{TM}}$ are certain braids termed cloaking elements.

Definition 14. A braid is called *cloaking element* of $(M, \sigma) \in \mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N$, if it stabilizes (M, σ) under the right action of the braid group via E-Multiplication.

In WalnutDSA$^{\text{TM}}$ cloaking elements of the following form are generated [41].

Proposition 15. *Let $(M, \sigma) \in \mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N$. Assume $\tau_a = -\tau_b^{-1}$ for two T-values with indices $1 \leq a < b \leq N$. Let σ_w denote the permutation induced by some braid $w \in B_N$ and let b_i be an Artin generator for $1 \leq i \leq N-1$. If*

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

the braid $w \cdot b_i^{\pm 4} \cdot w^{-1}$ cloaks (M, σ).

Proof. This is an immediate consequence of $\left(\mathsf{CB}_i(\tau_a) \cdot \mathsf{CB}_i(-\tau_a^{-1})\right)^2 = \mathsf{Id}_N$.

Remark 16. Cloaking elements as proposed by the designers of WalnutDSA$^{\text{TM}}$ depend only on the permutation σ and not on the matrix M of the element they are stabilizing. Therefore, we will say that some braid cloaks a permutation σ.

For further details on the generation of cloaking elements in WalnutDSA$^{\text{TM}}$ we refer interested readers to the original implementation by SecureRF [1] or our implementation in MAGMA [14](see [37]). However, our attack will be independent of the way cloaking elements are generated.

Concealed cloaking elements are cloaking elements for which the cloaked permutation is not public. Given a braid word W, concealed cloaking elements are added to the word by splitting W into two braid words W_1 and W_2 at a random location and inserting a braid cloaking the permutation induced by W_1 in between.

3.2 The Signature Scheme

Key Generation and Parameter Values Before any message can be signed, the following system wide public parameters need to be fixed:

- The rank N of the braid group B_N.
- A rewriting algorithm $\mathcal{R} : B_N \to B_N$, i.e. an algorithm transforming a braid word w into an equivalent braid word $\mathcal{R}(w)$. For example, one can use algorithms computing normal forms [11, 23], Dehornoy's handle reduction [17] or the stochastic rewriting algorithm introduced in [2].
- A finite field $\mathbb{F}_q$.

- T-values $= \{\tau_1, \tau_2, \ldots, \tau_N\} \in (\mathbb{F}_q^\times)^N$, such that $\tau_a = -\tau_b^{-1}$ for some publicly known integers $1 \leq a < b \leq N$.
- The number of concealed cloaking elements that will be added.
- A hash function $H : \{0,1\}^* \to \{0,1\}^{2k}$ for some k. Our attack will not depend on any weaknesses of the hash function and therefore we can treat H as a random oracle.

Next, the signer chooses braid words w and w' by choosing uniformly at random l Artin generators or their inverses. The secret key of the signer is the pair (w, w'), while the public key is $(\mathcal{P}(w), \mathcal{P}(w'))$ where $\mathcal{P}$ is the map given in Remark 12. Note, the length of the private braids w and w' is chosen large enough to prevent brute force attacks from being effective.
Later, we will see that the success of our new attack is independent of all parameters but N.

As of the 21st of November 2018, the use of the following parameters is suggested for WalnutDSA$^{\text{TM}}$:

claimed security level	128-bit security level	256-bit security level
N	10	10
q	$2^{31} - 1$	$2^{61} - 1$
l	132	287
concealed cloaking elts	12	24
H	SHA2-256	SHA2-512

Message Encoding In order for signatures to provide integrity and authenticity, a signer must encode the message that is to be signed into the signature. The Walnut digital signature algorithm requires the message to be mapped onto a pure braid.

To encode a message in WalnutDSA$^{\text{TM}}$ it is hashed using the publicly known hash function H. Then every two bits of the output specify one pure braid generator (see (2)) and the encoding $E(H(m))$ of a message m is the product of all pure braid generators selected. As the exact choice of pure braid generators is irrelevant for our attack we refer to [41] for a full description.

Signature Generation A signer needs to perform the following steps to generate a signature.

1. Compute the encoded message $E\big(H(m)\big)$.
2. Generate cloaking elements v, v_1, v_2 as given by Proposition 15 for the identity and the permutations induced by the private braids w, w', respectively.
3. Add the required number of concealed cloaking elements in randomly chosen locations in the braid words $W_1 := v_1 \cdot w^{-1} \cdot v$ or $W_2 := w' \cdot v_2$.
4. Use a rewriting algorithm $\mathcal{R}$ to obtain a rewritten braid word

$$\mathsf{Sig} := \mathcal{R}\big(W_1 \cdot E(H(m)) \cdot W_2\big),$$

which is the signature for m.

Signature Verification To verify a signature, a receiver computes $E(H(m))$ and checks whether

$$\mathrm{Matrix}\big(\mathcal{P}(w) \star \mathsf{Sig}\big) = \mathrm{Matrix}\big(\mathcal{P}(E(H(m)))\big) \cdot \mathrm{Matrix}\big(\mathcal{P}(w')\big) \tag{5}$$

comparing the matrix parts of $\mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N$. If both sides of the equation are equal, the receiver accepts the signature as valid. It is easy to check that legitimately produced signatures satisfy (5).

3.3 Previous Work on WalnutDSA™

We want to give a brief overview of previous attacks on the Walnut digital signature algorithm [10, 29, 35] and the changes they have triggered in the scheme to patch the weaknesses. Moreover, this section shows that our attack uses a completely different approach.

Factorization Attacks The first attack on a previous version of WalnutDSA™ was published by Hart et al. [29]. In the previous version both secret braids were equal and the public key only consisted of the image of this one secret braid under the map $\mathcal{P} : B_N \to \mathsf{GL}_N(\mathbb{F}_q) \times \mathsf{S}_N$ (see Remark 12).

The attack exploited a malleability property of the signatures, enabling an attacker to forge a signature by solving a factorization problem in a group of matrices. Trying to destroy the malleability property, the designers of Walnut started using two different private braids. However, Beullens [10, 41] showed that the following malleability property holds in this case too.

Theorem 17. *[10] Let m, m_1 and m_2 be messages and let h, h_1 and h_2 be the matrix parts of $\mathcal{P}\big(E(H(m))\big)$, $\mathcal{P}\big(E(H(m_1))\big)$ and $\mathcal{P}\big(E(H(m_2))\big)$, respectively.*
For braids $w_1, w_2, w_3 \in B_N$, we have

i) If $h = h_1^{-1}$ and Sig_1 is a valid signature for m_1 under the public key $\big(\mathcal{P}(w_1), \mathcal{P}(w_2)\big)$, then Sig_1^{-1} is a valid signature for m under the public key $\big(\mathcal{P}(w_2), \mathcal{P}(w_1)\big)$.
ii) If $h = h_1 \cdot h_2$ and $\mathsf{Sig}_1, \mathsf{Sig}_2$ are valid signatures for m_1 and m_2 under the public keys $\big(\mathcal{P}(w_1), \mathcal{P}(w_2)\big)$ and $\big(\mathcal{P}(w_2), \mathcal{P}(w_3)\big)$ respectively, then $\mathsf{Sig}_1 \cdot \mathsf{Sig}_2$ is a valid signature for m under the public key $\big(\mathcal{P}(w_1), \mathcal{P}(w_3)\big)$.

Suppose, an attacker wants to forge a signature for the message m under the public key $\big(\mathcal{P}(w), \mathcal{P}(w')\big)$. Clearly, they can compute the matrix $h = \mathrm{Matrix}\big(\mathcal{P}\big(E(H(m))\big)\big)$. Next, the attacker collects pairs of messages and signatures (m_i, Sig_i) that are valid under the same public key. By the malleability properties, it suffices to find a factorization $h = h_{i_1} \cdot h_{i_2}^{-1} \cdot h_{i_3} \ldots h_{i_{m-1}}^{-1} \cdot h_{i_m}$ to get a valid signature for m, where h_i denotes the matrix part of $\mathcal{P}\big(E(H(m_i))\big)$.

Such a factorization can be obtained by writing $h \cdot h_1^{-1}$ as a product of elements of the set

$$\left\{h_i h_j^{-1} \mid i \neq j;\ 1 \leq i, j \leq k\right\} \subseteq \left\{ \begin{pmatrix} X & Y \\ 0 & 1 \end{pmatrix} \mid X \in \mathsf{GL}_{N-1}(\mathbb{F}_q),\ Y \in \mathbb{F}_q^{N-1} \right\}. \tag{6}$$

An algorithm to solve this factorization problem with time complexity $\mathcal{O}\big(q^{\frac{N-1}{2}}\big)$ was proposed by Hart et al. [29]. However, the factorizations contained roughly 2^{25} elements of the set given in (6) and consequently the forged signature

$$\mathsf{Sig} = \mathsf{Sig}_{i_1} \cdot \mathsf{Sig}_{i_2}^{-1} \cdots \mathsf{Sig}_{i_{m-1}}^{-1} \cdot \mathsf{Sig}_{i_m} \cdot \mathsf{Sig}_1^{-1}$$

satisfies the verification equation, but can be easily detected due to its enormous length. By imposing an upper limit on the length of valid signatures as was done in the implementation submitted to NIST, the attack was blocked. In contrast, the forgeries produced by our attack will be of the same length as legitimately produced signatures.

Collision Search Attack Beullens and Blackburn [10, 41] realized that the originally proposed 4-bit encoder was not injective and that it mapped to a set of braids where the matrix parts under the function $\mathcal{P}$ were lying in a surprisingly low dimensional, 13 dimensional, affine subspace over $\mathbb{F}_q$. This made the scheme susceptible to a generic collision search attack. More precisely, it was possible to find pairs of distinct messages m_1 and m_2 such that $\mathcal{P}(E(H(m_1))) = \mathcal{P}(E(H(m_2)))$ for sufficiently small q using a generic collision search algorithm. Beullens and Blackburn implemented the collision search due to van Oorschot and Wiener [45] which takes $|\mathcal{P}\big(E(H(\{0,1\}))\big)|^{\frac{1}{2}} \leq q^{6.5}$ evaluations of $\mathcal{P} \circ E \circ H$.

Recall that a signature is accepted as valid if (5) is satisfied. Given a collision of m_1 and m_2, an attacker can query a signature for m_1 and gets automatically a valid signature for m_2. Consequently, the signature scheme was not existentially unforgeable [28].

To counter the attack the designers of WalnutDSA$^{\text{TM}}$ changed the encoder to the 2-bit version described previously, where $\mathcal{P}\big(E(H(\{0,1\}^*))\big)$ lies in an affine subspace of dimension $(N-2)^2+1$ [41] over $\mathbb{F}_q$, which is greater than 13 for $N \geq 6$. Together with a significant raise of the parameters N and q, the generic collision search attack became ineffective. Our attack will be independent of q, but we will see that it can be defeated to some extend by further increasing the parameter N.

Reversing E-Multiplication The last attack presented in [10] solves the underlying problem of WalnutDSA$^{\text{TM}}$, reversing E-Multiplication (REM) [see Definition 13], directly.

Note, it suffices to solve a single instance of the REM problem to forge a signature of a freely chosen message or solve two instances of the REM problem to obtain an equivalent pair of secret braids from the public key. Thus, the hardness of this problem is crucial for the security of WalnutDSA$^{\text{TM}}$.

The attack exploits that E-Multiplication restricted to pure braids is a group homomorphism which maps the chain of subgroups

$$\{e\} = P_1 \subset P_2 \subset \cdots \subset P_N \subset B_N$$

to a nice chain of subgroups in $\mathsf{GL}_N(\mathbb{F}_q)$. Here, $P_i \subset B_N$ denotes the subgroup of pure braids on N strings that can be identified with the pure braids of B_i or, formulated differently, the pure braids that can be written in the generators $b_1, \ldots, b_{i-1}$. Exploiting this subgroup structure, the REM problem can be solved by successively reducing the problem to a smaller subgroup using collision searches. The authors of [10] suggest moreover a slightly finer chain of subgroups for the first reductions which are the most costly ones to improve the performance of the algorithm further.

The resulting attack requires $\mathcal{O}(q^{\frac{N}{2}-1})$ E-Multiplications, and was blocked by a significant increase in the parameters q and N. As mentioned before, our attack will be independent of q and can only be defeated to some extent by increasing N significantly.

Uncloaking Signatures The most recent attack is due to Kotov, Menshov and Ushakov [35]. They give a heuristic attack which operates purely on braids. The attack removes cloaking elements of a previous version of the Walnut digital signature algorithm without concealed cloaking elements.

The authors observed that cloaking elements in WalnutDSA$^{\text{TM}}$ are always generated in such a way that the strands corresponding to the inverse T-values cross each other (see Proposition 15). Since T-values are public, an attacker can trace all strands and find "critical positions" in a signature

where there might be a cloaking element. This allows a length-based attack: Note that untwisting the middle part of cloaking elements produces a trivial braid. An attacker guesses the location of cloaking elements and tries to remove them by untwisting the critical position. When multiplying signatures with removed cloaking elements together, more precisely one such signature multiplied with the inverse of another, further elements cancel out. If the remaining word is of significantly shorter length, one has heuristic evidence that the cloaking elements have been removed successfully.

The uncloaking procedure on multiple signatures leads to a system of conjugacy equations in B_N (potentially with errors). Once again this can be heuristically solved using a length-based approach. For earlier work about length-based attacks we refer amongst others to [30, 39].

To patch Walnut, concealed cloaking elements, i.e. cloaking elements that are inserted in random locations before and after the encoded message, were introduced. Removing multiple concealed cloaking elements that are not inserted consecutively into the signature appears to be more difficult.

The designers of WalnutDSA$^{\text{TM}}$ suggested to insert

$$\kappa \geq \frac{2 \cdot (\text{security level in bits})}{\log_2(N!)} \tag{7}$$

concealed cloaking elements [41]. For $N = 10$ this yields the values given in the table in Section 3.2. However, the number κ was estimated under the assumption that one needs to know the permutation of a cloaking element in order to remove it. As this does not hold, the efficacy of this countermeasure has been disputed [41].

We will see that the success of our attack is independent of the number of concealed cloaking elements inserted to the signature the way it was suggested by the designers of WalnutDSA$^{\text{TM}}$. However, we will discuss in Section 5.3 that adding a significant number of concealed cloaking elements to the encoded message might thwart our attack at the cost of enlarging signatures and slowing down the signature generation and verification.

4 Decomposition of Products in Braid Groups

The use of normal forms as "obfuscation procedures" in cryptographic schemes such as WalnutDSA$^{\text{TM}}$ suggests that properties of single braids are well hidden in the normal form of their product. In this section, we will see that this is in general not the case. More precisely, we will argue that we can expect some (potentially reflected) permutation braids of factors with sufficiently large canonical length to appear in the normal form of their product.

In Section 4.1 we prove how the permutation braids of factors relate to the permutation braids of their product. Together with the experimental results of Section 4.2 this yields the observation stated in the previous paragraph. In Section 4.3 we show how the observation can be exploited under certain conditions to recover the factors of products of the form $ABC \in B_N$ up to the centre $\langle \Delta^2 \rangle$, when B is known. The algorithm to decompose products of braids will be at the heart of our cryptanalysis of WalnutDSA$^{\text{TM}}$ in Section 5 and our new solutions to the conjugacy and decomposition search problems in Section 6.

4.1 Garside Normal Form of Products

Recall that $b_i = \Delta b_{N-i} \Delta^{-1} = \tau(b_{N-i})$ for $i = 1, \ldots, N-1$ by Proposition 2 and Remark 3. Let $\Delta^a \cdot A_1 \ldots A_n$ and $\Delta^b \cdot B_1 \ldots B_m$ be the normal forms of two elements $A, B \in B_N$ respectively.

Pushing all Δ's in the product AB to the front yields

$$\begin{aligned} AB &= \Delta^a \cdot A_1 \dots A_n \cdot \Delta^b B_1 \dots B_m = \Delta^{a+b} \cdot \tau^b(A_1) \dots \tau^b(A_n) B_1 \dots B_m \\ &= \Delta^{a+b} \cdot \tau^{b'}(A_1) \dots \tau^{b'}(A_n) B_1 \dots B_m, \end{aligned} \tag{8}$$

for $b' \equiv b \pmod 2$ since τ^2 is the identity map. This is a product of permutation braids by the following Lemma of which we will omit the straightforward proof.

Lemma 18. *Let $1 \leq A_1, A_2 \leq \Delta$ be elements of B_N. Then $1 \leq \tau(A_1), \tau(A_2) \leq \Delta$ too. Furthermore, $A_1 A_2$ is a left-weighted product if and only if $\tau(A_1)\tau(A_2)$ is left-weighted.*

Thus, (8) is a product of permutation braids but in general not left-weighted. However, we see that $\tau(A_1) \dots \tau(A_n)$ is a left-weighted product by Lemma 18 and thus the following Lemma is an immediate consequence.

Lemma 19. *Let $\Delta^a \cdot A_1 \dots A_n$ and $\Delta^b \cdot B_1 \dots B_m$ be the left normal forms of the braids $A, B \in B_N$ respectively. Let $b' \equiv b \pmod 2$, then*

$$\Delta^{a+b} \cdot \tau^{b'}(A_1) \cdots \tau^{b'}(A_n) B_1 \cdots B_m$$

is the left normal form of AB if and only if $\tau^{b'}(A_n)B_1$ is a left-weighted product.

Clearly, the condition will not be met for most $A, B \in B_N$. When computing the left normal form of AB in general, new Δ's might be created in the process of computing the left-weighted product of $\tau^b(A_1) \dots \tau^b(A_n) B_1 \dots B_m$. Moving these Δ's to the front results in reflections of *all* leftward permutation braids, which yields the following proposition.

Proposition 20. *Let A, $B \in B_N$ and let $\Delta^a \cdot A_1 \dots A_n$ and $\Delta^b \cdot B_1 \dots B_m$ be their left normal form respectively. The left normal form of AB is*

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

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

Note that we have $\Delta^k \cdot X_1 \dots X_l = \tau^b(A_{n-C_N+1}) \dots \tau^b(A_n) \cdot B_1 \dots B_m$. The algorithms to compute the Garside left normal form visualize the previous proposition quite well. If $A_1 \cdots A_n$ is a left normal form and we multiply with an Artin generator b_i on the right, this modifies the last permutation braid if $A_n b_i \wedge \Delta \neq A_n \wedge \Delta$. If A_n is not changed all leftward permutation braids are still in left normal form and we are done. If A_n is changed two conditions must be met for A_{n-1} to be changed as well. First, $A_n b_i \wedge \Delta$ must contain another Artin generator b_j in the set of all Artin generators the word can start with compared to $A_n \wedge \Delta$. And second, $A_{n-1} b - j \wedge \Delta \neq A_{n-1} \wedge \Delta$. This process continues inductively to the left until some permutation braid is not changed anymore. If one of the changed permutation braids becomes Δ during this process, it is moved to the front by reflecting all leftward permutation braids.

Remark 21. It is not hard to find particular braids for which the previous proposition does not contain a lot of information as $c = n$. This happens for example, if $B = A^{-1}$ when the product vanishes or if A and B are braids that do not share common strands and thus commute. However, in the next section we will see that for every N and randomly chosen braids $A, B \in B_N$ the expected value for c is bounded independently of n.

Clearly, if c is smaller than n the permutation braids in the left normal forms of A and AB coincide on the left hand side up to reflection. Next, we show that the rightmost permutation braids of the left normal forms of B and AB coincide too. The following Proposition due to Elrifai and Morton provides us with a link between multiplication of a braid on the left and on the right.

Proposition 22. *[20] Let $\Delta^u \cdot x_1 \dots x_m$ be the left normal form of X. Then the left normal form of X^{-1} is $X^{-1} = \Delta^{-u-m} \cdot x'_m \dots x'_1$, where $x'_i := \tau^{-u-i}(x_i^{-1}\Delta)$ for $i = 1, \dots, m$.*

The braid $x_i^{-1}\Delta$ is called the *right complement* of x_i. Let δ denote the map sending permutation braids to their right complement. It is easy to check that δ induces a bijection on the permutation braids and $\delta^2 = \tau$.

Proposition 23. *Let $A, B \in B_N$ and let $\Delta^a \cdot A_1 \dots A_n$ and $\Delta^b \cdot B_1 \dots B_m$ be their left normal form respectively. The left normal form of AB is*

$$\Delta^{a+b+k} \cdot Y_1 \dots Y_l \cdot B_{c+1} \dots B_m,$$

for some integer $0 \leq c \leq m$ and permutation braids $Y_1, \dots, Y_l$, where $k \in \mathbb{Z}$ is the number of Δ's that are created when computing the left normal form of $\tau^b(A_1) \dots \tau^b(A_n)B_1 \dots B_m$.

Proof. Clearly, we can show the proposition for A^{-1} and B^{-1} instead of A and B. More precisely, we show that the permutation braids on the right hand side of $A^{-1}B^{-1}$ coincide with the ones of B^{-1}.

By Proposition 20 we know that the left normal form of $B_1 \dots B_m A_1 \dots A_n$ is

$$\Delta^k \tau^k(B_1) \dots \tau^k(B_{m-c}) X_1 \dots X_l, \tag{9}$$

for some $0 \leq c \leq m$, $k \in \mathbb{Z}$ and permutation braids $X_1, \dots, X_l$. Proposition 22 implies that the left normal form of $(B_1 \dots B_m A_1 \dots A_n)^{-1} = A_n^{-1} \dots A_1^{-1} B_m^{-1} \dots B_1^{-1}$ is

$$\begin{aligned} &\Delta^{-k-(m-c+l)} \cdot X'_l \dots X'_1 \cdot \big(\tau^k(B_{m-c})\big)' \dots \big(\tau^k(B_1)\big)' \\ &= \Delta^{-k-(m-c+l)} \cdot X'_l \dots X'_1 \cdot \tau^{-m+c}\big(\delta(B_{m-c})\big) \dots \tau^{-1}\big(\delta(B_1)\big), \end{aligned} \tag{10}$$

using

$$\big(\tau^k(B_i)\big)' = \tau^{k-i}\big(\delta(\tau^k(B_i))\big) = \delta^{2(k-i)}\big(\delta^{2k+1}(B_i)\big) = \tau^{-i}\big(\delta(B_i)\big).$$

Simultaneously, the left normal form of $B_m^{-1} \dots B_1^{-1}$ is $\tau^{-m}(\delta(B_m)) \dots \tau^{-1}(\delta(B_1))$ by Proposition 22. Comparing with (10), we see that the left normal forms of $B_m^{-1} \dots B_1^{-1}$ and $A_n^{-1} \dots A_1^{-1} B_m^{-1} \dots B_1^{-1}$ coincide on the rightmost $m - c$ permutation braids. This finishes the proof.

4.2 Penetration Distance

In this section we provide experimental results to estimate the size of the parameter c in Propositions 20 and 23 for "randomly" chosen braids A and B. We will find that for every N this expectation is uniformly bounded independently of the canonical lengths of the factors A and B.

Since the braid group B_N is infinite for $N \geq 2$, choosing braids at random is a non-trivial task. In practice, there are various ways to choose braids of B_N in a randomized manner. However, different methods result in different probability distributions on B_N.

Recall that every braid word can be rewritten as an element of the monoid of positive braids B_N^+ which we introduced in Section 2.3. Let $|x|$ denote the *length* of a positive braid $x \in B_N^+$, i.e. the number of Artin generators occurring in any positive braid word representing x. Since the defining relations of the braid group (and the braid monoid) are homogeneous, this is well-defined.

We start by recalling some results due to Gebhardt and Tawn [27] who studied the Garside normal forms of random braids. They analysed statistical properties of the normal forms of positive braids of length k generated using two methods:

i) Choose uniformly at random k Artin generators $b_i \in \{b_1, \ldots, b_{N-1}\}$ and concatenate them, i.e. choose uniformly at random a braid word from the set of all positive braid words of B_N^+ of length k. We say that we generate *positive words* of length k uniformly at random.
ii) Consider the set of all braids that can be represented by a braid word of length k and choose uniformly at random one braid from this set. We say that we generate uniformly at random *positive braids* of length k.

Note, the number of words representing the same element of B_N^+ depends on the element. Therefore, both variants yield different probability distributions on the set of all braids that can be represented by positive braid words of length k.

However, the implementation of the second method is significantly more difficult in practice (see [26] for an algorithm that runs polynomially in N and k) which is why most (cryptographic) applications generate "random braids" similarly to the first method.

Following the terminology of Gebhardt and Tawn, we call conjugation with Δ, i.e. a reflection, of a permutation braid a *trivial change*. We define the *penetration distance* as follows.

Definition 24. *[27]* For two braids A and B, the *penetration distance* $\mathsf{pd}(A, B)$ for the product AB is the number of permutation braids at the end of the normal form of A which undergo a non-trivial change in the normal form of the product. I.e.

$$\mathsf{pd}(A, B) = cl(A) - max\{i \in \{0, \ldots, cl(A)\} : A\Delta^{-\inf(A)} \wedge \Delta^i = AB\Delta^{-\inf(AB)} \wedge \Delta^i\}$$

where $cl(\cdot)$ denotes the canonical length and $\inf(\cdot)$ the infimum of a braid.

Based on their experiments, Gebhardt and Tawn conjectured the following.

Conjecture 25. [27] *Let $A \in B_N$ be a braid which is randomly chosen from either the uniformly generated random words or from the uniformly generated random braids of length k and let b_i be a randomly chosen Artin generator of B_N. Then the expected penetration distance is bounded independently of the length k of the braid, i.e. there exists some C such that for all k*

$$\mathbb{E}(\mathsf{pd}(A, b_i)) < C.$$

The conjecture raises the question whether there still exists an upper bound for the expected penetration distance of the product AB of two randomly chosen braids or braid words independently of their lengths. That is when B is an arbitrary randomly chosen braid or braid word as well instead of a single randomly chosen Artin generator.

For the purpose of investigating this question, we conducted an experiment in MAGMA [14]. We generated 2.000 instances of pairs of braid words $A, B \in B_N$ for different given lengths using the

built-in random function of the braid package in MAGMA. To obtain a "random" braid of given length k, this function chooses uniformly at random a_i from $X \cup X^{-1} \backslash a_{i-1}^{-1}$ for $k = 1, \ldots, k$, where X and X^{-1} is the set of Artin generators and their inverses respectively. In other words, the built-in random function chooses uniformly at random a braid word from the set of all freely reduced braid words of a given length k.

Given such pairs of randomly generated braid words A, B, we computed the product AB and the penetration distance for each particular instance. This was done by comparing the permutation braids in the left normal forms of A and AB directly. The diagram in Figure 3 shows the average penetration distance with respect to the lengths of A and B for different values of N.

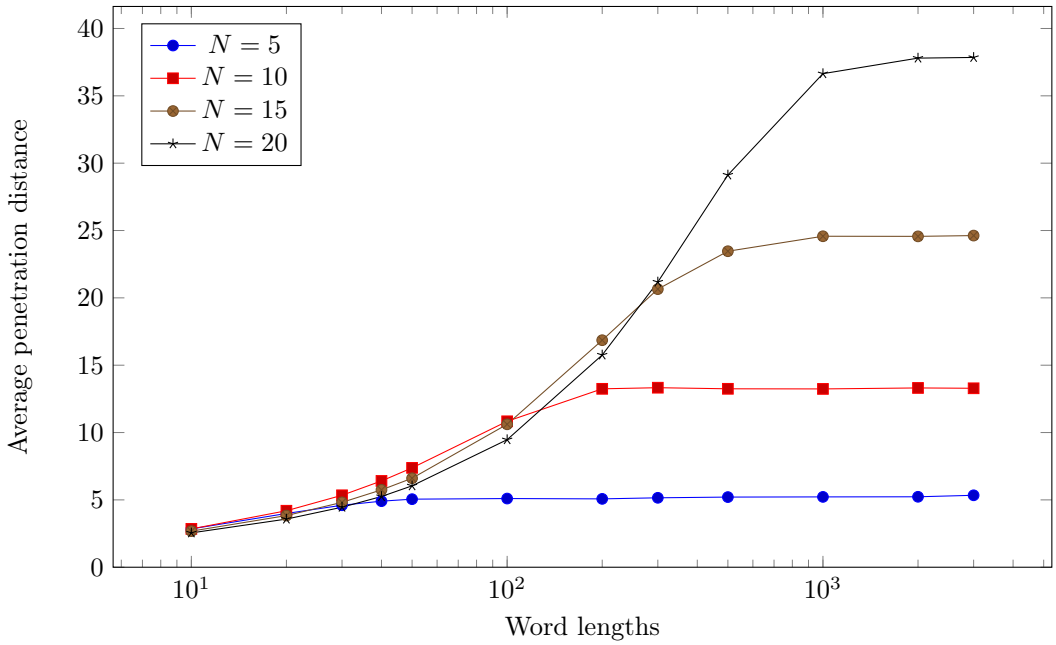

Fig. 3: Average penetration distance after multiplication with braid of given length on the right hand side

We observe that for each N the average penetration distance increases with the word lengths of the random braids and eventually converges to some bound. Furthermore, these bounds increase with the number of strands N of the braid group. Note that for our attack on WalnutDSA$^{\text{TM}}$ we will be mainly interested in estimates for $N = 10$ because this is the parameter used.

The convergence suggests that for every N there exists an upper bound for the expected penetration distance of the product of randomly generated freely reduced braid words independently of their lengths.

Conjecture 26. Let $A, B \in B_N$ 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

$$\mathbb{E}(\mathsf{pd}(A, B)) < C_N.$$

Plotting the distribution of penetration distances for products of randomly chosen freely reduced braid words for different lengths we noted that most data points are distributed closely around the mean.

Now, Conjeture 26 has significant importance for Proposition 20. Let A and B be two randomly chosen braids of canonical length n and m respectively. Assuming Conjecture 26, i.e. assuming that the expected penetration distance is bounded by some C_N independently of the lengths of A and B, Proposition 20 implies that we expect at least the leftmost $n - C_N$ permutation braids of A and AB to coincide up to reflection whenever $n \geq C_N$.

Looking at the proof of Proposition 23 we see that C_N is a bound for the expected size of the parameter c too. This is because the inverse of freely reduced braid words of a given length is a freely reduced braid word of the same length. Thus, drawing freely reduced braid words of a given length from the braid group B_N has the same probability distribution as drawing their inverses. Hence, if A and B are two randomly chosen braids of canonical length n and m, we expect at least the $m - C_N$ rightmost permutation braids of B and AB to coincide whenever $m \geq C_N$.

4.3 The Algorithm

We use the last part of this section to describe how our observation can be utilised to decompose products ABC of braids $A, B, C \in B_N$, when B is known. More precisely, we discuss how to recover $A' \equiv A \pmod{\Delta^2}$, $C' \equiv C \pmod{\Delta^2}$ such that $AC = A'C'$. Here, by $\pmod{\Delta^2}$ we mean up to multiplication with powers of Δ^2. Later, we can apply this algorithm to break WalnutDSA$^{\text{TM}}$ and solve instances of the conjugacy and decomposition search problems.

Let $A = \Delta^a \cdot A_1 \dots A_n$, $B = \Delta^b \cdot B_1 \dots B_m$, and $C = \Delta^c \cdot C_1 \dots C_r$ be the left normal forms of randomly chosen freely reduced braid words $A, B, C \in B_N$. Assume that m is greater than the C_N given by Conjecture 26. We have discussed in the previous section that we can expect the left normal form of BC to be of the form

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

for some permutation braids $Y_1, \dots, Y_l$ such that $\Delta^k \cdot Y_1 \dots Y_l = \tau^c(B_{j+1}) \dots \tau^c(B_m) \cdot \Delta^{-c} C$ and $k \in \mathbb{Z}$ is the number of fundamental braids Δ that are being created when computing the left-weighted form of $\tau^c(B_{j+1}) \dots \tau^c(B_m) C_1 \dots C_r$.

Now, if the part of the normal form of B that was preserved into BC is of canonical length greater than $C_N + 1$, which we expect to happen for $m \geq 2C_N + 1$, the left normal form of $A(BC)$ is expected to be of the form

$$\Delta^{a+b+c+k+k'} \cdot X_1 \dots X_r \cdot \tau^{c+k}(B_{C_N+1}) \dots \tau^{c+k}(B_{m-C_N}) \cdot Y_1 \dots Y_l \tag{11}$$

by Proposition 23 and the previous section, where $\Delta^{k'} \cdot X_1 \dots X_r$ is a left-weighted product of permutation braids equal to $\tau^{b+c+k}(A_1) \dots \tau^{b+c+k}(A_n) \cdot \tau^{c+k}(B_1) \dots \tau^{c+k}(B_i)$ if the centre of B equals Δ^2 which we expect for sufficiently long B.

We will keep this notation for the remainder of this section. Let the left normal form of a given ABC be

$$\Delta^u \cdot X_1 \dots X_r \cdot \tau^{c+k}(B_i) \dots \tau^{c+k}(B_j) \cdot Y_1 \dots Y_l,$$

where $u = a+b+c+k+k'$. By the previous discussion, we know that $i-j>0$ can be expected for randomly chosen freely reduced braid words $A,\ B$ and $C \in B_N$ with B of canonical length greater than $2C_N+1$.

It is now a straightforward procedure to recover $A' \equiv A \pmod{\Delta^2}$ and $C' \equiv C \pmod{\Delta^2}$ such that $AC = A'C'$ knowing only B:

1. Compute the left normal forms of B and ABC.
2. Check, whether there is a contiguous subsequence $B_{i_1} \dots B_{i_2}$ of permutation braids of the left normal form of B for some $1 \le i \le i_1 < i_2 \le j \le m$ in the left normal form of ABC using a string-matching algorithm. If such a subsequence is found, save the location in the left normal form of B and ABC and go to 3. Otherwise, do the same search for contiguous subsequences $\tau(B_{i_1}) \dots \tau(B_{i_2})$ of $\tau(B)$ in the left normal form of ABC.
   If no common subsequence of permutation braids can be found either, we terminate the process and cannot recover the factors. If multiple common subsequences are found, we run the following steps for every of the finitely many possible solution. Notice, the latter is not very likely to happen for randomly chosen braid words and sufficiently long subsequences.
3. Split the braid B or $\tau(B) = \tau(B_1) \dots \tau(B_m)$ at B_{i_1} resp. $\tau(B_{i_1})$ into two parts. Then, do the same for ABC. Denote the parts $B_I,\ B_{II},\ ABC_I$, and ABC_{II}.
   Note that we find the subsequence $\tau^{c+k}(B_i) \dots \tau^{c+k}(B_j)$ in B or $\tau(B)$ depending on whether $c+k$ leaves residue 0 or 1 modulo 2, since τ^2 is the identity. Thus, even though we know neither c nor k we can determine the residue of $c+k \pmod 2$ which we denote by $(c+k)'$.
   Using the notation of previous paragraphs, we compute

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

4. Compute

$$\begin{aligned} A' :&= ABC_I \cdot B_I^{-1} \cdot \Delta^{-(c+k)'} \\ &= \Delta^{a+b+c+k+k'} \cdot X_1 \dots X_r \cdot \tau^{c+k}(B_{i-1})^{-1} \dots \tau^{c+k}(B_1)^{-1} \cdot \Delta^{-b-(c+k)'} \\ &= \Delta^{a+c+k-(c+k)'} \cdot A_1 \dots A_n \end{aligned}$$

   and

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

Since $a + c + k - (c + k)' \equiv a \pmod 2$ and $-k + (c + k)' \equiv c \pmod 2$, we have recovered $A' \equiv A \pmod{\Delta^2}$ and $C' \equiv C \pmod{\Delta^2}$. Using $c \equiv -k + (c + k)' \pmod 2$, we have furthermore

$$A'C' = \Delta^{a+c} \cdot \tau^c(A_1) \dots \tau^c(A_n) \cdot C_1 \dots C_l = AC.$$

The success rate of this decomposition algorithm will be discussed in Section 6.

5 New Attack on WalnutDSA™

In this section we want to present our attack on the group-based signature scheme WalnutDSA™ which is an application of the decomposition algorithm we have developed in Section 4.

In Section 5.1 we present the idea behind our attack on WalnutDSA™, before providing experimental results on the success of our attack in Section 5.2. In Section 5.3 we discuss how different parameters influence the running time and success rate of our attack and we suggest one potential countermeasure.

5.1 Universal Forgery Attack

Let m be a message with the legitimately produced signature $\mathsf{Sig} \in B_N$. Recall that the braids corresponding to signatures of WalnutDSA™ have a representative braid word of the form

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

where $E(H(m))$ is the encoded message and $W_1, W_2 \in B_N$ are braids of the form $v_1 \cdot w^{-1} \cdot v$ and $w' \cdot v_2$ with additional concealed cloaking elements inserted. Here, $w, w' \in B_N$ are the private braids of the signer and v, v_1, v_2 are braids cloaking the identity of S_N and the permutations induced by w and w', respectively.

It is easy to see that the braid $\mathsf{Sig}' := W_1 \cdot E(H(m')) \cdot W_2$ is a valid signature for the message m'. Hence, the ability to locate $E(H(m))$ in a legitimate signature and replacing it by $E(H(m'))$ for an arbitrarily chosen message m' gives rise to a universal forgery attack.

To prevent attackers from finding the encoded message by just parsing through the signature, the designers of WalnutDSA™ suggested an *obfuscation procedure*. That is, the application of a rewriting algorithm such as the Garside normal form, BKL normal form [11], stochastic rewriting [2] or Dehornoy's handle reduction [17] to the braid before appending the signature to a message.

Note that rewriting changes only the representative of the same braid. Consequently, normal forms are the strongest way to obfuscate signatures because every attacker can compute them given another representative of the same braid.

Our experimental results in the next section will show that most legitimately produced signatures of WalnutDSA™ are susceptible to the decomposition algorithm described in Section 4.3. Since anybody can compute the encoding of a message m, this allows us to recover $W_1' \equiv W_1 \pmod{\Delta^2}$ and $W_2' \equiv W_2 \pmod{\Delta^2}$ such that $W_1' \cdot W_2' = W_1 \cdot W_2$ given only one valid signature $W_1 \cdot E(H(m)) \cdot W_2$ of any m. As $W_1' \cdot E(H(m')) \cdot W_2' = W_1 \cdot E(H(m')) \cdot W_2$, this is enough to obtain forged signatures for any other message m'.

Proposition 27. *Let $W_1 \cdot E(H(m)) \cdot W_2 \in B_N$ be a valid signature for some message m and let $W_1', W_2' \in B_N$ such that $W_1' \equiv W_1 \pmod{\Delta^2}$, $W_2' \equiv W_2 \pmod{\Delta^2}$ and $W_1 \cdot W_2 = W_1' \cdot W_2'$. Then,*

$$W_1' \cdot E\big(H(m')\big) \cdot W_2'$$

is a valid signature for any message m'.

Computation of universal forgeries: Given a signature $\mathsf{Sig} = W_1 \cdot E\big(H(m)\big) \cdot W_2$ and the corresponding message m, an adversary computes the encoded message $E\big(H(m)\big)$ and uses the procedure described in Section 4.3 to recover two braids W_1', W_2' such that $W_1' \equiv W_1 \pmod{\Delta^2}$, $W_2' \equiv W_2 \pmod{\Delta^2}$ and $W_1' \cdot W_2' = W_1 \cdot W_2$. By Proposition 27, this suffices to compute a valid signature for *any* message m':

$$\mathsf{Sig}' = W_1' \cdot E\big(H(m')\big) \cdot W_2'$$

Comparison to legitimately produced signatures: Since W_1 and W_2 are legitimately produced and do not depend on $E\big(H(m)\big)$, it is impossible to distinguish a forged signature of the form $W_1' \cdot E\big(H(m')\big) \cdot W_2'$ from a legitimately produced signature for m'. In particular, the length of our forgeries is the same as the one of legitimately produced signatures.

However, given two signatures one could recognize that at least one was likely forged. Note an attacker can solve this issue by adding an additional concealed cloaking element to W_1 and W_2.

Complexity: In our decomposition algorithm of Section 4.3, we need to compute the Garside normal form of Sig and $E\big(H(m)\big)$ in the first step. Using Thurston's method, this takes time in $\mathcal{O}(|\mathsf{Sig}|^2 N \log N)$ and $\mathcal{O}(|E(H(m))|^2 N \log N)$ respectively. Here $|\cdot|$ means the number of permutation braids of the given positive braid word, not necessarily in left normal form.

The second step of the algorithm requires to find a common contiguous subsequence of permutation braids in the normal forms. Fixing a length Len for the common subsequence that we want to find, the naive algorithm compares $\mathcal{O}(rl)$ products of Len permutation braids, where r and l denote the canonical length of $E(H(m))$ and Sig respectively. We implemented this naive approach in our attack on WalnutDSA$^{\text{TM}}$ (see [37]). A more efficient solution is to use the Knuth–Morris–Pratt string-searching algorithm [32]. Running this algorithm on all contiguous subsequence of permutation braids of length Len from the (reflected) encoding and the signature takes $\mathcal{O}(r(l + \mathsf{Len}))$ comparisons of permutation braids.

For WalnutDSA$^{\text{TM}}$, we have $|E(H(m))| \leq |\mathsf{Sig}|$. As the number of permutation braids in the Garside normal is minimal compared to other positive braid words we have moreover $r \leq |E(H(m))|$ and $l \leq |\mathsf{Sig}|$ and thus recovering the positions and whether the subsequence of the encoding in the signature is reflected takes $\mathcal{O}(|\mathsf{Sig}|^2)$ comparisons of permutation braids. Since the rest of the decomposition algorithm runs in linear time, the algorithm to forge signatures is dominated by the time it takes to compute the Garside normal form, i.e $\mathcal{O}(|\mathsf{Sig}|^2 N \log N)$.

Note that generating legitimate signatures is quadratic in N too. Moreover, the Garside normal form of a signature might need to be computed as well, depending on the rewriting algorithm used in the generation of WalnutDSA$^{\text{TM}}$ signatures.

Improvements: As the encoded message is located in between of two braids W_1 and W_2 of roughly the same size in the signature, we anticipate to find the subsequence of the permutation braids of $\tau^k\big(E(H(m))\big)$ roughly in the middle of the signature. Therefore, it is faster on average to start the search for common permutation braids in the middle part of the signature and encoding.

5.2 Experimental Results

We have implemented the relevant parts of WalnutDSA^{TM} and our attack in MAGMA [14]. The source code of our implementations can be found on GitHub [37]. For our experiments we used the recommended parameters as listed in Section 3.2 for the two security levels. In particular, the number of strands N was set to 10.

By Section 4 we know that the crucial part for our decomposition algorithm to work is finding a (potentially reflected) contiguous subsequence of permutation braids of the normal form of $E(H(m))$ in the normal form of the signature of m. We generated 1.000 instances of signatures for randomly chosen messages m and both security levels. In our experiment, we were able to locate such a common subsequence of permutation braids in the normal forms of $\tau^k\big(E(H(m))\big)$ and $W_1 \cdot E(H(m)) \cdot W_2$ for either $k = 0$ or 1 in all instances. The following table, Figure 4, shows the canonical lengths of the common subsequences we found for the 128- and 256-bit parameters respectively.

length of common subsequence	128-bit security level	256-bit security level
mean	100	238
minimum	19	153
maximum	142	288

Fig. 4: Lengths of common subsequences of permutation braids of encodings and signatures

To put this into context, we measured the canonical length of encoded messages. For the 128-bit parameters, encoded messages had canonical lengths ranging from 112 to 165 with a mean of 140. The range for 256-bit parameters was 248 to 310 with a mean of 280 permutation braids.

To determine the position of a common subsequence of permutation braids in (reflected) encoded message $\tau^k\big(E(H(m))\big)$ and signature Sig, we compared a specified number Len of permutation braids of $\tau^k\big(E(H(m))\big)$ and Sig for $k = 0, 1$ at a time. Note that finding common subsequences of a given length is faster than finding all common subsequences of arbitrary lenghts.

The larger the number Len it becomes less likely that a common subsequence appears in the signature just by coincidence. However, we want it to be small enough to actually find a common subsequence in most cases. Fixing $\mathsf{Len} = 15$ turned out to be a good choice in our implementation but taking $\mathsf{Len} = 10$ or 20 leads to almost the same results.
Later, we will see that increasing the number of strands N in the Walnut digital signature algorithm would lead to shorter common subsequences of permutation braids. In this situation we can improve our algorithm to find the common position and whether $k = 0$ or 1 by reducing Len inductively whenever we can not find a common subsequence of permutation braids for $k = 0$ and 1 until we find one or $\mathsf{Len} = 0$.

Testing our entire attack on randomly generated instances, 99.8% of legitimately produced signatures for the 128-bit parameters turned out to allow our universal forgery attack. For the 256-bit security level all 100% of signatures were susceptible.

The algorithm to recover the braids W_1' and W_2' and thus to produce universal forgeries takes time less than a second for the 128-bit and only a couple seconds for the 256-bit parameters.

The higher success rate for the 256-bit parameters can be explained with the output of the hash function being twice as long. This results in the normal form of the encoding containing roughly twice

as many permutation braids. Therefore, it is more likely to find a common contiguous subsequence of permutation braids in the left normal forms of the signature and the (reflected) encoded message.

5.3 Countermeasures

Finally, we want to discuss how different parameters of WalnutDSA$^{\text{TM}}$ influence the running time and success rate of our attack and we suggest one potential countermeasure. Here, the success rate means the proportion of signatures that allows a universal forgery attack.

Independence from q: Unlike the attacks [10, 29], our attack works on the braids only and thus independently of the colored Burau representation. In particular, it is independent of the size q of the underlying finite field $\mathbb{F}_q$.

Increasing the length of the private braids: Increasing the number of concealed cloaking elements or the length of private braids makes both W_1 and W_2 and consequently the signature larger. We see that the running time of our attack is quadratic in the length of the signature and thus it slows down our attack a little bit, while simultaneously enlarging the size of signatures.

We have seen in Section 4.3 that the expected number of permutation braids that change non-trivially when multiplying with randomly chosen braid words on the left and right is bounded independently of their length. Therefore, we do not expect enlarging W_1 and W_2 to have a great influence on the success of our attack. Indeed, we generated random instances of Walnut signatures using different lengths for private keys. This did not seem to have any influence on the number of permutation braids found as a common subsequence in the signature and the (reflected) encoding. The success rate of the attack did not change even for very long private braids either.

For private braids randomly chosen from freely reduced words of length 15.000 Artin generators (instead of 287), our attack is still successful within a few minutes while legitimate signatures reach the imposed upper limit for the length of signatures that are being accepted as valid in the current implementation of WalnutDSA$^{\text{TM}}$. Consequently, increasing the length of private braids is not useful to thwart our attack.

Increasing N: Looking at the formula for the running time, increasing N is another way to slow down the attack slightly.

More interesting, however, is that increasing N decreases the success rate. We conducted an experiment generating WalnutDSA$^{\text{TM}}$ instances for different values of N. Figure 5 shows the percentage of signatures allowing our universal forgery attack out of 1.500 randomly generated Walnut instances depending on N.

We have seen in Figure 3 that raising N influences the number of permutation braids that are expected to change when multiplying with braids on the right. For multiplication on the left, we obtained the same result. At the same time the canonical length of the encoding remains constant when scaling up N since it only depends on the length of the output of the hash function used in WalnutDSA$^{\text{TM}}$. Combined, this implies that the expected length of the common subsequence of permutation braids of signature and encoding shrinks when raising N. Note that we cannot just reduce the length of the output of the hash function as the signature scheme would become vulnerable to collision search attacks [41].

As our attack does not work anymore once there is no common subsequence of permutation braids left, this explains the decreasing success rate when increasing N. In the 256-bit version the

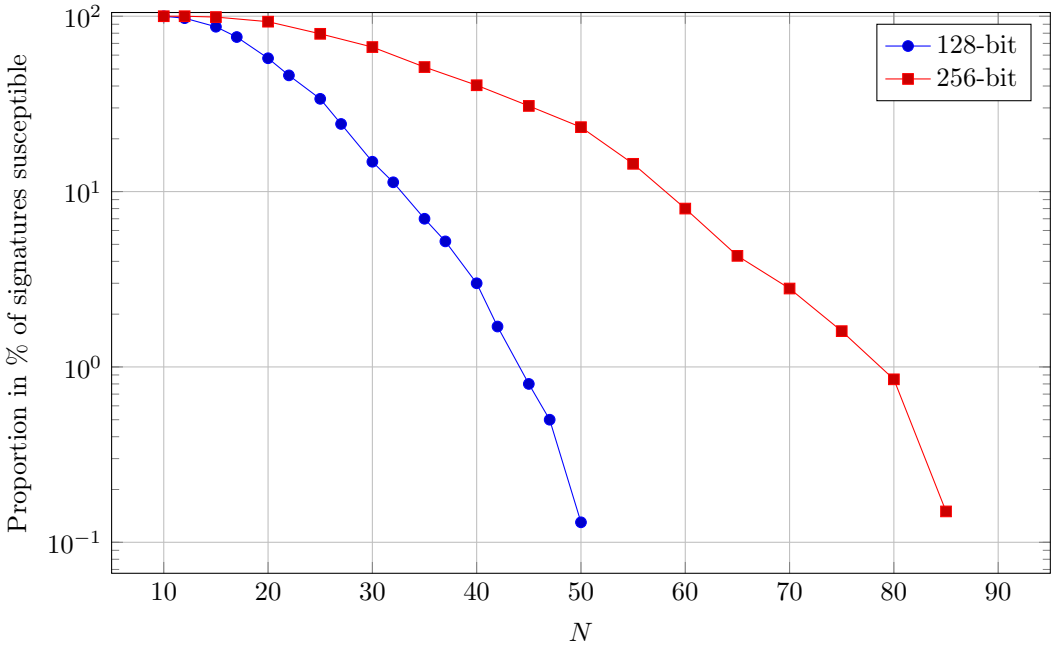

Fig. 5: Success rate of universal forgery attack depending on N

hash function has a longer output and therefore the common subsequence of permutation braids of encoding and signature is larger than in the 128-bit setting. This justifies, why the success probability decreases slower when increasing N for the 256-bit security level. Moreover, we measured the success of our attack by checking whether we recovered the braids W_1 and W_2 modulo their centre successfully. For large N it is more likely that the centre of the encoding $E(H(m))$ does not equal Δ^2 and as we recover braids W_1 and W_2 modulo the centre of $E(H(m))$ this might not be accepted as valid.

Considering our experiment shown in Figure 5, the success of our attack seems to decrease exponentially when increasing N. However, this would increase the size of the public keys and slow down the signature verification quadratically in N. Moreover, one could fear that with N increasing and the hash output length constant, the encoding will not have good mixing properties. It might be possible to isolate the encoding in the signature just parsing through the braid, therefore leading to other weaknesses.

Adding additional cloaking elements to the encoded message: Finally, one could add some randomness to the encoder altering the permutation braids in the signature corresponding to the encoding which can be done by adding concealed cloaking elements (see Section 3.1) to the encoding. This countermeasure was independently found and suggested by the WalnutDSA$^{\text{TM}}$ team in a private correspondence.
Clearly, the previously described attack to recover W_1 and W_2 modulo Δ^2 does not necessarily work anymore after adding cloaking elements to the encoding. However, forging signatures is possible as long as we can find at least one permutation braid in the signature corresponding to a permutation braid in the encoding and the encoding separates the permutation braids of W_1 and W_2. This is,

because we have $\mathcal{P}\Big(E(H(m))_I^* \cdot E(H(m))_I^{-1}\Big) = (\mathrm{Id}, \mathrm{id})$ and $\mathcal{P}\Big(E(H(m))_{II}^{-1} \cdot E(H(m))_{II}^*\Big) = (\mathrm{Id}, \mathrm{id})$, where $E(H(m))_i^*$ are the parts of the encoding $E(H(m))$ containing additionally concealed cloaking elements. Together with the fact that all encodings are pure braids, we have therefore for $k = 0$ or 1

$$\begin{aligned}
&\mathcal{P}\Big(\mathsf{Sig}_I \cdot \tau^k\big(E(H(m))_I^{-1} \cdot E(H(m')) \cdot E(H(m))_{II}^{-1}\big) \cdot \mathsf{Sig}_{II}\Big) \\
&\qquad = \mathcal{P}\Big(W_1 \cdot E(H(m))_I^* E(H(m))_I^{-1} \cdot E(H(m')) \cdot E(H(m))_{II}^{-1} E(H(m))_{II}^* \cdot W_2\Big) \\
&\qquad = \mathcal{P}\big(W_1 \cdot E(H(m')) \cdot W_2\big).
\end{aligned}$$

We know that this still satisfies (5) and thus it is a valid signature for m'. Hence, even though an attacker can not recover W_1 and W_2 up to the centre they can still compute a forged signature for any message m' as long as they find a single permutation braid from the encoding in the signature at the correct position.

Consequently, to counter the attack one needs to make sure that all permutation braids originating from the encoding in the signature are changed. Our experiments show that introducing one cloaking element changes sometimes only 5 permutation braids in their surrounding for $N = 10$. Considering the canonical length of common subsequences measured in Section 5, we would therefore expect that at least 30 and 60 additional concealed cloaking elements need to be added for the two security levels. However, it might be necessary to add even more cloaking elements to prevent being susceptible to our attack after applying an uncloaking procedure such as the one due to Kotov, Menshov, and Ushakov [35] to critical positions in the middle of the signature eventually removing concealed cloaking elements.

Altogether, adding additional concealed cloaking elements to the encoding is the best way we found to thwart our attack. Yet, it would slow down the signature generation as all additional concealed cloaking elements need to be generated separately and it would enlarge the signatures of WalnutDSA$^{\mathrm{TM}}$.

6 Application to the Conjugacy and Decomposition Search Problem

Another problem that can be solved using our decomposition algorithm from Section 4 is the *decomposition search problem* which can be formulated for the braid group as follows.

Definition 28. Given two elements X, Y of the braid group B_N and two subsets $A, B \in B_N$. The *decomposition search problem* (DSP) is to find elements $a \in A$ and $b \in B$ such that $Y = aXb$.

It is straightforward to construct key exchange protocols based on this problem, assuming that elements of A and B commute with each other [34, 43]. Here, our decomposition algorithm of the previous subsection can be used to recover a and b for some instances up to elements of the centre of X, given $Y = aXb$.

Recall that our algorithm to solve DSP by decomposing the braid Y is not only fast but also requires almost no memory. Given B and a product of braids ABC in B_N, the decomposition algorithm of Section 4 is dominated by the time it takes to compute the Garside normal form of ABC, i.e. $\mathcal{O}(|ABC|^2 N \log N)$ using Thurston's approach where $|ABC|$ denotes the number of

permutation braids a given positive braid word of ABC is written in. Note, that the Garside normal form can be computed even faster in practice [27].

We analysed the success of our decomposition algorithm for randomly chosen braid words A, B and C. To this end we generated uniformly at random freely reduced braid words $A, B, C \in B_N$ of given lengths using MAGMA [14]. Given the product ABC and B, we applied the decomposition algorithm and considered a run successful whenever we were able to recover A and C up to the centre of B_N, i.e. up to multiplication by powers of Δ^2.

Figure 6 shows the percentage of successful recoveries depending on the word lengths of A, B and C for different numbers of strands N. We see that the attack is very successful for sufficiently long randomly chosen braid words reaching 100% success rate. Moreover, we see that this "sufficient" length increases with N. This is no surprise since the bound of Conjecture 26 increases with N as previously noticed. Thus, for words that are shorter it is less likely to find a contiguous subsequence of (reflected) permutation braids of B in ABC.
Moreover, for randomly chosen words B of short length it is more likely that the centre of the braid associated to B does not equal the centre of the braid group generated by Δ^2. Therefore, braids recovered for short B using our decomposition algorithm might not be accepted as valid in our experiments.

Clearly, the conjugacy search problem (Definition 1) is a special case of the decomposition search problem and our decomposition algorithm can be used to solve instances of the conjugacy search problem too. Indeed, a successful run of the decomposition algorithm provides us with a braid $\tilde{C}$ equal to C up to the centre of X, given X and $Y = C \cdot X \cdot C^{-1}$. Consequently $\tilde{C}$ is a solution to the conjugacy search problem, as

$$Y = C \cdot X \cdot C^{-1} = \tilde{C} \cdot X \cdot \tilde{C}^{-1}.$$

Recall that our decomposition algorithm needs a common subsequence of permutation braids of $\tau^k(X)$ and $Y = C \cdot X \cdot C^{-1}$, for $k = 0$ or 1, to work. By Section 4.2, we can expect this for braid words X and C that are chosen uniformly at random whenever X has sufficiently large canonical length depending on N. However, in the case of the conjugacy search problem we can apply our decomposition algorithm for some short X as well, exploiting that X and Y are conjugate.
This is because C can be recovered by applying the decomposition algorithm to the braids X^n and $Y^n = (C \cdot X \cdot C^{-1})^n = C \cdot X^n \cdot C^{-1}$ with larger canonical length instead of X and Y, where n is a positive integer. We tested this procedure for randomly generated braid words of a given length X and C. Whenever the decomposition algorithm was not able to find a common subsequence in the permutation braids of $\tau^k(X)$ and $Y = C \cdot X \cdot C^{-1}$ for $k = 0$ or 1, we tried it on X^n and Y^n instead. In our experiments we used $n = 4$ and reran the decomposition algorithm on powers at most 3 times. The result of our experiments can be seen in Figure 7 and shows clearly that the decomposition algorithm works in the case of CSP for shorter words than for the DSP displayed in Figure 6.

However, we want to point out that there is not always an n such that X^n and Y^n share a potentially reflected subsequence of permutation braids. Indeed, the minimal counterexample is $N = 4$, $X = b_1$ and $Y = b_2 b_1 b_2^{-1}$, where b_i are Artin generators. We denote permutation braids by their induced permutation. The left normal forms of X^n and Y^n are the products $(1, 2)^n$ and $(1, 3, 2, 4)(2, 3)^{n-1}(1, 3, 2)$ respectively, which do not share a single permutation braid.

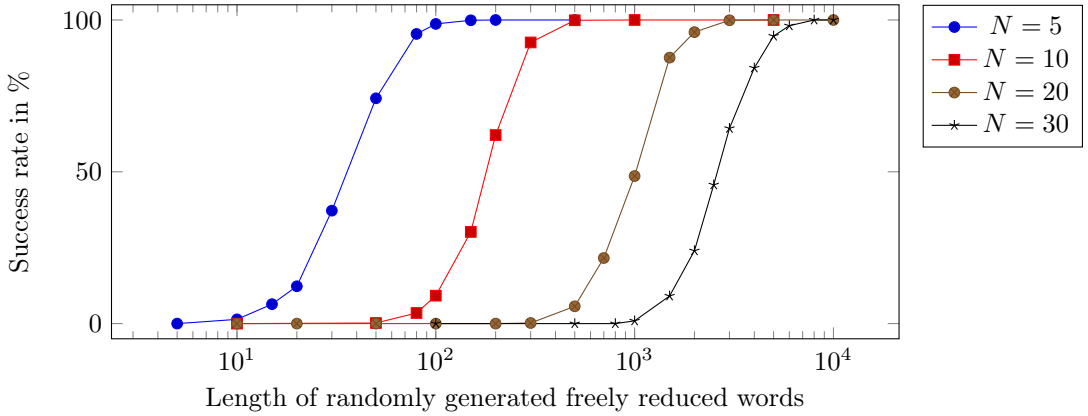

Fig. 6: Success rate of decomposition algorithm for instances of the DSP

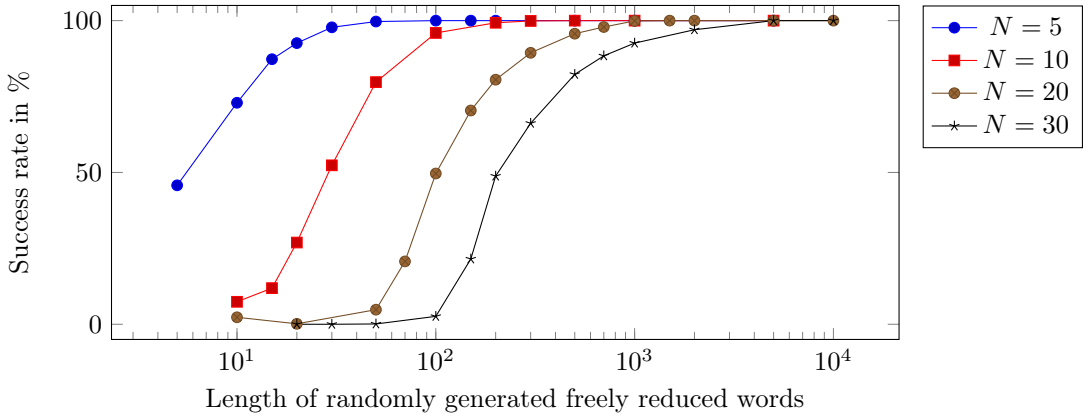

Fig. 7: Success rate of decomposition algorithm for instances of the CSP

Due to the vast use of the CSP, DSP and its variants in the design of cryptographic protocols, studying further applications of our decomposition algorithm and a thorough comparison with other solutions to the conjugacy and decomposition search problem in braid groups will be subject to future work.

7 Conclusion and Further Work

In cryptographic schemes based on braid groups, products of braids are often constructed involving secret braids as factors, and it is hoped that rewriting the product will hide the individual factors. We demonstrated that this is not the case for randomly chosen braid words. We provided an algorithm to compute individual components of products ABC when B is known and ABC is presented in normal form. We expect this decomposition to work for randomly chosen braids A, B and C if B is of canonical length greater than $2C_N + 1$, where C_N is the number given by Conjecture 26. In Section 4.2 we estimated C_N experimentally for some values of N.

As an application of our decomposition algorithm we presented a new universal forgery attack on the previously unbroken instantiation of WalnutDSA$^{\text{TM}}$. Given a single random message-signature pair, our attack allows to forge signatures for arbitrary messages within seconds for the 128-bit and 256-bit security levels. Hereby, the forgeries are indistinguishable from legitimately produced signatures. Our experiments showed that 99.8% and 100% of legitimately produced signatures in WalnutDSA$^{\text{TM}}$ can be used in our new attack for the claimed 128-bit and 256-bit security levels respectively. In contrast to previous attacks, our attack produces signatures that are identically distributed as legitimate signatures and applies to all versions of WalnutDSA$^{\text{TM}}$. Unlike the previous attacks in [10, 29], our attack works on the braids only. Thus, it does not depend on the colored Burau representation of the braid group and is independent of the size q of the underlying finite field $\mathbb{F}_q$. We have further discussed how other parameters influence the success probability and running time of our universal forgery attack. Adding sufficiently many concealed cloaking elements to the encoding may thwart our attack at the cost of increasing the length of signatures and slowing down the signature generation algorithm.

As another application, we provide a new algorithm for solving the conjugacy and decomposition search problems, two problems at the heart of other cryptographic systems based on braid groups [22]. The running time of this algorithm is dominated by the time it takes to compute the Garside normal form of ABC but also requires almost no memory to work.

We leave a full theoretical analysis of our decomposition algorithm for products of braids to further work. In particular, a proof of Conjecture 26 would be very interesting, even from a purely mathematical point of view. Conjecture 25 due to Gebhardt and Tawn [27] which would provide a partial solution is yet to be proven as well.

Improving our attack, finding different countermeasures and studying the efficiency of the one suggested by us might be of interest for further research regarding WalnutDSA$^{\text{TM}}$. More generally, we believe that our decomposition algorithm is applicable to other cryptographic schemes that have been suggested for braid groups. Researching further applications and a thorough comparison of our new solution to the conjugacy and decomposition search problems in braid groups to existing approaches will be subject for future work.

Acknowledgments The authors would like to thank Ward Beullens and the anonymous reviewers for their helpful feedback. This work was produced as part of a master's thesis of the first author at the University of Oxford. He is now supported by the EPSRC as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/P009301/1).

Bibliography

[1] About SecureRF: https://www.securerf.com/about-us/, accessed: 21/11/2018

[2] Anshel, I., Atkins, D., Goldfeld, P., Gunnels, D.: Kayawood, a key agreement protocol. Preprint available at https://eprint.iacr.org/2017/1162 (version: 30-Nov-2017) (2017)

[3] Anshel, I., Anshel, M., Fisher, B., Goldfeld, D.: New key agreement protocols in braid group cryptography. In: Naccache, D. (ed.) Topics in Cryptology — CT-RSA 2001. pp. 13–27. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)

[4] Anshel, I., Anshel, M., Goldfeld, D.: An algebraic method for public-key cryptography. Mathematical Research Letters **6**, 287–292 (1999)

[5] Anshel, I., Anshel, M., Goldfeld, D., Lemieux, S.: Key agreement, the Algebraic Eraser, and lightweight cryptography. Contemporary Mathematics **418**, 1–34 (2007)

[6] Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.: WalnutDSA: A Quantum Resistant Group Theoretic Digital Signature Algorithm. Preprint available at https://eprint.iacr.org/2017/058 (30-Nov-2017) (2017)

[7] Artin, E.: Theorie der Zöpfe. In: Abhandlungen aus dem mathematischen Seminar der Universität Hamburg. vol. 4, pp. 47–72. Springer (1925)

[8] Ben-Zvi, A., Blackburn, S.R., Tsaban, B.: A practical cryptanalysis of the Algebraic Eraser. In: Annual International Cryptology Conference. pp. 179–189. Springer (2016)

[9] Ben-Zvi, A., Kalka, A., Tsaban, B.: Cryptanalysis via algebraic spans. In: Annual International Cryptology Conference. pp. 255–274. Springer (2018)

[10] Beullens, W., Blackburn, S.: Practical attacks against the Walnut digital signature scheme. Accepted to Asiacrypt 2018. Preprint available at https://eprint.iacr.org/2018/318/20180404 (2018)

[11] Birman, J., Ko, K.H., Lee, S.J.: A new approach to the word and conjugacy problems in the braid groups. Advances in Mathematics **139**(2), 322–353 (1998)

[12] Birman, J.S.: Braids, Links, and Mapping Class Groups.(AM-82), vol. 82. Princeton University Press (1975)

[13] Birman, J.S., Gebhardt, V., González-Meneses, J.: Conjugacy in Garside groups I: cyclings, powers and rigidity. Groups, Geometry, and Dynamics **1**(3), 221–279 (2007)

[14] Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system I: The user language. Journal of Symbolic Computation **24**(3-4), 235–265 (1997)

[15] Bressaud, X.: A normal form for braids. Journal of Knot Theory and its Ramifications **17**(06), 697–732 (2008)

[16] Burau, W.: Über Zopfgruppen und gleichsinnig verdrillte Verkettungen. In: Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg. vol. 11, pp. 179–186. Springer (1935)

[17] Dehornoy, P.: A fast method for comparing braids. Advances in Mathematics **125**(2), 200–235 (1997)

[18] Dehornoy, P.: Alternating normal forms for braids and locally Garside monoids. Journal of Pure and Applied Algebra **212**(11), 2413–2439 (2008)

[19] Ding, J., Yang, B.Y.: Multivariate Public Key Cryptography, pp. 193–241. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)

[20] Elrifai, E.A., Morton, H.R.: Algorithms for positive braids. Quarterly Journal of Mathematics **45**(180), 479–498 (1994)

[21] Epstein, D., Cannon, J., Holt, D., Levy, S., Paterson, M., Thurston, W.: Word processing in groups (1992)
[22] Garber, D.: Braid group cryptography. In: Braids: Introductory lectures on braids, configurations and their applications, pp. 329–403. World Scientific (2010)
[23] Garside, F.A.: The braid group and other groups. The Quarterly Journal of Mathematics **20**(1), 235–254 (1969)
[24] Gebhardt, V.: A new approach to the conjugacy problem in Garside groups. Journal of Algebra **292**(1), 282–302 (2005)
[25] Gebhardt, V., González-Meneses, J.: The cyclic sliding operation in Garside groups. Mathematische Zeitschrift **265**(1), 85–114 (2010)
[26] Gebhardt, V., González-Meneses, J.: Generating random braids. Journal of Combinatorial Theory, Series A **120**(1), 111–128 (2013)
[27] Gebhardt, V., Tawn, S.: Normal forms of random braids. Journal of Algebra **408**, 115–137 (2014)
[28] Goldwasser, S., Bellare, M.: Lecture notes on cryptography. Summer course "Cryptography and computer security" at MIT (1996)
[29] Hart, D., Kim, D., Micheli, G., Pascual-Perez, G., Petit, C., Quek, Y.: A Practical Cryptanalysis of WalnutDSA. In: IACR International Workshop on Public Key Cryptography. pp. 381–406. Springer (2018)
[30] Hughes, J., Tannenbaum, A.: Length-based attacks for certain group based encryption rewriting systems. arXiv preprint cs/0306032 (2003)
[31] Kalka, A., Teicher, M., Tsaban, B.: Short expressions of permutations as products and cryptanalysis of the Algebraic Eraser. Advances in Applied Mathematics **49**(1), 57–76 (2012)
[32] Knuth, D.E., Morris, Jr, J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM journal on computing **6**(2), 323–350 (1977)
[33] Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J.s., Park, C.: New public-key cryptosystem using braid groups. In: Bellare, M. (ed.) Advances in Cryptology — CRYPTO 2000. pp. 166–183. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
[34] Ko, K.H., Lee, S.J., Cheon, J.H., Han, J.W., Kang, J.s., Park, C.: New public-key cryptosystem using braid groups. In: Annual International Cryptology Conference. pp. 166–183. Springer (2000)
[35] Kotov, M., Menshov, A., Ushakov, A.: An attack on the Walnut digital signature algorithm (2018)
[36] McEliece, R.: A public-key cryptosystem based on algebraic coding theory. Deep Space Network Progress Report **44**, 114–116 (1978)
[37] Merz, S.P.: Non obfuscating power of Garside normal forms. GitHub repository at https://github.com/SimonMerz/Non-obfuscating-power-of-Garside-normal-forms (2018)
[38] Micciancio, D., Regev, O.: Lattice-based Cryptography, pp. 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
[39] Myasnikov, A.D., Ushakov, A.: Length based attack and braid groups: Cryptanalysis of anshel-anshel-goldfeld key exchange protocol. In: Okamoto, T., Wang, X. (eds.) Public Key Cryptography – PKC 2007. pp. 76–88. Springer Berlin Heidelberg (2007)
[40] National Institute for Standards and Technology (NIST): Post-quantum crypto standardization (2016), https://csrc.nist.gov/projects/post-quantum-cryptography
[41] NIST PQC Forum: Available at https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum, accessed: 21/11/2018

[42] Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on. pp. 124–134. Ieee (1994)
[43] Shpilrain, V., Ushakov, A.: Thompson's group and public key cryptography. In: International Conference on Applied Cryptography and Network Security. pp. 151–163. Springer (2005)
[44] Stolbunov, A.: Constructing public-key cryptographic schemes based on class group action on a set of isogenous elliptic curves. Advances in Mathematics of Communications **4**(2), 215–235 (2010)
[45] Van Oorschot, P.C., Wiener, M.J.: Parallel collision search with cryptanalytic applications. Journal of cryptology **12**(1), 1–28 (1999)