DEFEATING THE HART ET AL, BEULLENS-BLACKBURN, KOTOV-MENSHOV-USHAKOV, AND MERZ-PETIT ATTACKS ON WALNUTDSA™

IRIS ANSHEL, DEREK ATKINS, DORIAN GOLDFELD, AND PAUL E. GUNNELLS
SECURERF CORPORATION
100 BEARD SAWMILL RD #350, SHELTON, CT 06484

Abstract. The Walnut Digital Signature Algorithm (WalnutDSA) brings together methods in group theory, representation theory, and number theory, to yield a public-key method that provides a means for messages to be signed and signatures to be verified, on platforms where traditional approaches cannot be executed. After briefly reviewing the various heuristic/practical attacks that have be posited by Hart et al, Beullens-Blackburn, Kotov-Menshov-Ushakov, and Merz-Petit, we detail the parameter choices that defeat each attack, ensure the security of the of the method, and demonstrate its continued utility.

1. Introduction

The Walnut Digital Signature Algorithm (WalnutDSA) is a group-theoretic, public-key, method that enables messages to be signed and signatures of said messages to be verified on platforms where legacy protocols cannot be efficiently executed. Recently, various cryptography and cryptology researchers have proposed a range of attacks on WalnutDSA using methods that involve linear algebra, combinatorics, and canonical forms, all of which have been successfully blocked. These approaches rely on algorithms that are exponential in run–time and can be addressed with suitable choices of parameters. It is a feature of WalnutDSA that there are, in fact, many parameters which can be adjusted: this is a benefit in contemporary implementations where there may be constraints on energy, memory, or processing which make parameters for common cryptographic choices unusable.

This paper proceeds as follows. First, it provides an introduction to WalnutDSA along with a survey of the requisite mathematics. It, then, reviews WalnutDSA signature generation and verification. The foregoing serves as the foundation and reference for understanding how the attacks were defeated. Next, it examines each of the attacks as of the date of publication and explains the techniques to defeat each specific attack. In each instance, this paper specifies and details the appropriate parameter choices to defeat the various heuristic and practical attacks and explains why such choices do not alter the utility of the method. In addition, the paper identifies the parameter choices that ensure the security of the method.

2. Brief Introduction to WalnutDSA$^{\text{TM}}$

For, $N \geq 2$, let B_N denote the N-strand braid group with Artin generators $\{b_1, b_2, \ldots, b_{N-1}\}$, subject to the following relations:

$$b_i b_{i+1} b_i = b_{i+1} b_i b_{i+1}, \qquad (i = 1, \ldots, N-2), \tag{1}$$

$$b_i b_j = b_j b_i, \qquad (|i-j| \geq 2). \tag{2}$$

Thus any $\beta \in B_N$ can be expressed as a product of the form

$$\beta = b_{i_1}^{\epsilon_1} \, b_{i_2}^{\epsilon_2} \, \cdots \, b_{i_k}^{\epsilon_k}, \tag{3}$$

where $i_j \in \{1, \ldots, N-1\}$, and $\epsilon_j \in \{\pm 1\}$. Note that β is not uniquely represented by (3); any braid has infinitely many different expressions in terms of the Artin generators, thanks to the relations (1) and (2).

Each braid $\beta \in B_N$ determines a permutation in S_N (group of permutations of N letters) as follows: For $1 \leq i \leq N-1$, let $\sigma_i \in S_N$ be the i^{th} simple transposition, which maps $i \to i+1$, $i+1 \to i$, and leaves $\{1, \ldots, i-1, i+2, \ldots, N\}$ fixed. Then σ_i is associated to the Artin generator b_i. Further, if $\beta \in B_N$ is written as in (3), we take β to be associated to the permutation $\sigma_\beta = \sigma_{i_1} \cdots \sigma_{i_k}$. A braid is called pure if its underlying permutation is trivial (i.e., the identity permutation).

Let $\mathbb{F}_q$ denote the finite field of q elements, and for variables $t_1, t_2, \ldots, t_N$, let

$$\mathbb{F}_q[t_1, t_1^{-1}, \ldots, t_N, t_N^{-1}]$$

denote the ring of Laurent polynomials in $t_1, t_2, \ldots, t_N$ with coefficients in $\mathbb{F}_q$. Next, we introduce the colored Burau representation

$$\Pi_{CB} \colon B_N \to GL\Big(N, \mathbb{F}_q[t_1, t_1^{-1}, \ldots, t_N, t_N^{-1}]\Big) \times S_N.$$

First, we define the $N \times N$ colored Burau matrix (denoted CB) of each Artin generator as follows[**?**].

$$CB(b_1) = \begin{pmatrix} -t_1 & 1 & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & & 1 \end{pmatrix}, \tag{4}$$

For $2 \leq i \leq N-1$, the matrix $CB(b_i)$ is defined by

$$CB(b_i) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & t_i & -t_i & 1 & & \\ & & & \ddots & & \\ & & & & & 1 \end{pmatrix}, \tag{5}$$

where the indicated variables appear in row i, and if $i = 1$ the leftmost t_1 is omitted.

We similarly define $CB(b_i^{-1})$ by modifying (5) slightly:

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

where again the indicated variables appear in row i, and if $i = 1$ the leftmost 1 is omitted.

Recall that each b_i has an associated permutation σ_i. We may then associate to each braid generator b_i (respectively, inverse generator b_i^{-1}) a colored Burau/permutation pair $(CB(b_i), \sigma_i)$ (resp., $(CB(b_i^{-1}), \sigma_i)$). We now wish to define a multiplication of such colored Burau pairs. To accomplish this, we require the following observation. Given a Laurent polynomial $f(t_1, \ldots, t_N)$ in N variables, a permutation in $\sigma \in S_N$ can act (on the left) by permuting the indices of the variables. We denote this action by $f \mapsto {}^{\sigma}f$:

$${}^{\sigma}f(t_1, t_2, \ldots, t_N) = f(t_{\sigma(1)}, t_{\sigma(2)}, \ldots, t_{\sigma(N)}).$$

We extend this action to matrices over the ring of Laurent polynomials in the t_i by acting on each entry in the matrix, and denote the action by $M \mapsto {}^{\sigma}M$. The general definition for multiplying two colored Burau pairs is now defined as follows: given $b_i^{\pm}, b_j^{\pm}$, the colored Burau/permutation pair associated with the product $b_i^{\pm} \cdot b_j^{\pm}$ is

$$(CB(b_i^{\pm}), \sigma_i) \cdot (CB(b_j^{\pm}),\ \sigma_j) = \Big(CB(b_i^{\pm}) \cdot ({}^{\sigma_i}CB(b_j^{\pm})),\ \sigma_i \cdot \sigma_j\Big).$$

We extend this definition to the braid group inductively: given any braid

$$\beta = b_{i_1}^{\epsilon_1} b_{i_2}^{\epsilon_2} \cdots b_{i_k}^{\epsilon_k},$$

we can define a colored Burau pair $(CB(\beta), \sigma_\beta)$ by

$$(CB(\beta), \sigma_\beta) = (CB(b_{i_1}^{\epsilon_1}) \cdot {}^{\sigma_{i_1}}CB(b_{i_2}^{\epsilon_2}) \cdot {}^{\sigma_{i_1}\sigma_{i_2}}CB(b_{i_3}^{\epsilon_3}))\ \ \cdots\ \ {}^{\sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_{k-1}}}CB(b_{i_k}^{\epsilon_k}),\ \ \sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_k}).$$

The colored Burau representation is then defined by

$$\Pi_{CB}(\beta) := (CB(\beta), \sigma_\beta).$$

One checks that Π_{CB} satisfies the braid relations and hence defines a representation of B_N.

3. E-Multiplication and Cloaking Elements

In brief, E-Multiplication is an action of a group of ordered pairs associated with B_N on a direct product of two groups. Given an element $\beta \in B_N$, we can associate with β both the colored Burau matrix $CB(\beta)$ (whose entries are Laurent polynomials in N variables) and the natural permutation σ_β of the braid which is an element in S_N. Since permutations themselves act on the colored Burau matrices, the ordered pairs $(CB(\beta), \sigma_\beta)$ form a group under the semi-direct product operation. By fixing a field $\mathbb{F}_q$, and a collection of N invertible elements in $\mathbb{F}_q$, $\{\tau_1, \ldots, \tau_N\}$, termed t-values, we can define the right action of $(CB(\beta), \sigma_\beta)$ on the ordered pair $(M, \sigma) \in GL_N(\mathbb{F}_q) \times S_N$:

$$(M, \sigma) \star (CB(\beta), \sigma_\beta)\ =\ \big(M \cdot {}^{\sigma}\big(CB(\beta)\big) \downarrow_{t\text{-values}}, \sigma \circ \sigma_\beta\big)\,,$$

where the $\downarrow_{t\text{-values}}$ indicates the polynomials are evaluated at the t-values. While the Laurent polynomials which would naturally occur as entries of the colored Burau matrices would become computationally unmanageable, the generators b_i of B_N have sparse colored Burau matrices, and, hence, E-Multiplication can be evaluated very efficiently and rapidly.

The above discussion of an infinite group acting on a finite group necessitates the existence of stabilizing elements in the group B_N. With this in mind, we have the following:

Definition (Cloaking element) *Let $m \in GL(N, \mathbb{F}_q)$ and $\sigma \in S_N$. An element v in the pure braid subgroup of B_N (i.e., the permutation associated to v is the identity) is termed a cloaking element of (m, σ) if it satisfies $(m, \sigma) \star v = (m, \sigma)$.*

Thus a cloaking element will essentially disappear when E-Multiplication is evaluated. Since stabilizing elements of a group action form a subgroup, the following proposition is immediate:

Proposition 3.1. *The set of braids that cloak a specific ordered pair (m, σ) forms a subgroup of B_N.*

It should be remarked that when cloaking elements are constructed in the manner above, such elements only depend on the permutation σ. Thus, with a small abuse of language, we can say the element v cloaks for the permutation σ without any ambiguity.

When we fix a braid β, say

$$\beta = b_{i_1}^{\epsilon_1} \cdots b_{i_\ell}^{\epsilon_\ell},$$

and choose some point $1 \leq k \leq \ell$. Clearly, $\beta = x_1 \cdot x_2$ where $x_1 = b_{i_1}^{\epsilon_1} \cdots b_{i_{k-1}}^{\epsilon_{k-1}}$ and $x_2 = b_{i_k}^{\epsilon_k} \cdots b_{i_\ell}^{\epsilon_\ell}$, and, hence, for any matrix/permutation pair (m_0, σ_0), we have that $(m_0, \sigma_0) \star \beta = ((m_0, \sigma_0) \star x_1) \star x_2$.

Assume we have a method to generate a cloaking element v for the product of $\sigma_0 \cdot \sigma_{x_1}$ where σ_{x_1} deotes the permutation associated with x_1. By construction, given any matrix M we have that $(M, \sigma_0 \cdot \sigma_{x_1}) \star v = (M, \sigma_0 \cdot \sigma_{x_1})$. Since $(m_0, \sigma_0) \star x_1$ takes the form $(m_0, \sigma_0) \star x_1 = (M, , \sigma_0 \cdot \sigma_{x_1})$. It follows that

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

Hence we have generated a new braid β' which contains v,

$$\beta' = x_1 \cdot v \cdot x_2,$$

which has the property that $(m_0, \sigma_0) \star \beta = (m_0, \sigma_0) \star \beta'$. We shall refer to this inserted cloaking element as a *concealed* cloaking element.

Definition (κ cloaking) *Given an element $\beta \in B_N$, the output of randomly inserting κ concealed cloaking elements into the braid β is defined to be a κ–cloaking of β and is denoted by $\kappa(\beta)$.*

4. WalnutDSA$^{\text{TM}}$ Signature Generation and Verification

Let $\mathcal{R}\colon B_N \to B_N$ denote a braid group rewriting algorithm. Well known examples are the Garside canonical form [5], Birman-Ko-Lee canonical form [3], and the Dehornoy handle reduction algorithm [4]. For $\beta \in B_N$ let $\mathcal{P}(\beta)$ denote the E-multiplication of β against the identity element, i.e.,

$$\mathcal{P}(\beta) = (\mathrm{Id}_N, \mathrm{Id}_{S_N}) \star \beta$$

where Id_N is the $N \times N$ identity matrix and Id_{S_N} is the identity element in the symmtric group S_N. The Signer's private key consists of two random freely reduced braids $w, w' \in B_N$. The Signer's public key is $\big(\mathcal{P}(w),\, \mathcal{P}(w')\big)$.

Fix a hash function H. To sign a message $m \in \{0,1\}^*$ the Signer performs the following steps:

Digital Signature Generation:

1. Compute $H(m)$.

2. Generate cloaking elements v, v_1, and v_2 such that

- v cloaks $(\mathrm{Id}_N, \mathrm{Id}_{S_N})$,
- v_1 cloaks $\mathcal{P}(w)$.
- v_2 cloaks $\mathcal{P}(w')$.

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

4. Compute $\mathrm{Sig} = \mathcal{R}\big(\kappa\left(v_1 \cdot w^{-1} \cdot v \cdot E(H(m)) \cdot w' \cdot v_2\right)\big)$, which is a rewritten braid.

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

Signature Verification: The signature (m, Sig) is verified as follows:

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

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

3. Evaluate the E-Multiplication $\mathcal{P}(w) \star \mathrm{Sig}$.

4. Test the equality

$$\mathrm{Matrix}\Big(\mathcal{P}(w) \star \mathrm{Sig}\Big) \stackrel{?}{=} \mathrm{Matrix}\Big(\mathcal{P}\big(E(H(m))\big)\Big) \cdot \mathrm{Matrix}\Big(\mathcal{P}\big(w'\big)\Big), \tag{6}$$

where Matrix denotes the matrix part of the ordered pair in question, and the multiplication on the right is the usual matrix multiplication. The signature is valid if and only if (6) holds and the signature has length $\leq 2L$ where L is a certain positive integer such that all valid WalnutDSA$^{\text{TM}}$ signatures have length in the range $[L, 2L]$.

The security of WalnutDSA$^{\text{TM}}$ is based on the hard problems known as Reversing E-multiplication (REM) together with the problem of removing the κ randomly inserted concealed cloaking elements.

5. The Hart, Kim, Micheli, Pascuel-Perez, Petit, Quek Attack

Hart et al [13] proposed a practical forgery attack on WalnutDSA$^{\text{TM}}$. As pointed out by the authors, the attack can be defeated by increasing the parameter sizes, and that even in the range where the attack is successful, it produces forgeries that are many orders of magnitude larger than the signatures allowed in the protocol, i.e., the attack is blocked because the WalnutDSA$^{\text{TM}}$ protocol specifies a length limit on the signatures. In fact, the run time of the attack is exponential and can be easily defeated while still retaining the high efficiency and low power consumption advantages of WalnutDSA$^{\text{TM}}$ for constrained devices. For example, the attack can be completely (see §6) thwarted and a 2^{128} (respectively 2^{256}) security level can be maintained by running WalnutDSA$^{\text{TM}}$ on the braid group B_{10} and the finite field $\mathbb{F}_{M_{31}}$, where M_{31} is the Mersenne prime $2^{31}-1$ (respectively B_{10}, M_{61}).

The Hart et al attack [13] is a universal forgery attack that works in the special case when the two private keys w, w' are equal. The attack is based on a solution of the following group factorization problem in $GL(N, \mathbb{F}_q)$.

Definition (Group Factorization Problem) *Let G be a finitely generated group with generators $\{g_1, \ldots, g_r\}$. Given $h \in G$ find a small integer L and sequences $(m_1, \ldots, m_L) \in \{1, 2, \ldots, r\}^L$ and $(\epsilon_1, \ldots \epsilon_L) \in \{\pm 1\}^L$ such that*

$$h = \prod_{i=1}^{L} g_{m_i}^{\epsilon_i}.$$

We now explain how a solution to the group factorization problem can be used to forge signatures. Assume an attacker is in possession of many messages m_i and WalnutDSA$^{\text{TM}}$ signatures s_i with $i \in I$ in a finite indexing set. Let $E(H(m_i))$ denote the encoding of the hash of the message m_i into the braid group B_N and define $g_i := \text{Matrix}\big(\mathcal{P}(E(H(m_i))\big) \in GL(N, \mathbb{F}_q)$.

Assume that the attacker wants to forge a signature for a message m. Let $h = \text{Matrix}\big(\mathcal{P}(E(H(m))\big)$. Suppose the attacker can find $\epsilon_{i_j} \in \{\pm 1\}$ and a small positive integer L such that

$$h = \prod_{j=1}^{L} g_{i_j}^{\epsilon_{i_j}}$$

where $i_j \in I$ for $j = 1, 2, \ldots, L$. Then as shown in [13] a valid signature for m is given by $s = \prod_{j=1}^{L} s_{i_j}^{\epsilon_{i_j}}$.

The basic strategy for the attack is to build forgeries iteratively using a nested sequence of subgroups. In particular, there is a chain of subgroups $A_1 \subset A_2 \subset \cdots \subset A_{N-1}$ in $GL(N, \mathbb{F}_q)$, and a corresponding sequence of subgroups $P_1 \subset P_2 \subset \cdots \subset P_{N-1}$ of the braid group B_N. The two are related in that the matrix part under E-multiplication of any braid in P_i lands in A_i. The main step of the attack attempts to improve a partial solution of the problem in A_i, P_i to a more complete one in the smaller subgroups A_{i-1}, P_{i-1}. An essential role in building and improving solutions is played by the *distinguished point method*, which is a general collision attack on all one-way functions that has nothing to do with E-multiplication in particular.

6. Defeating the The Hart, Kim, Micheli, Pascuel-Perez, Petit, Quek Attack

In the Hart et al paper [13], the time complexity, memory complexity, and signature length are carefully estimated. Assume we are running WalnutDSA$^{\text{TM}}$ on the braid group B_n and finite field $\mathbb{F}_q$. They show that the running time complexity of the algorithm is

$$\approx 2 \cdot \gamma \cdot q^{\frac{N-1}{2}},$$

the memory complexity is

$$\log_2(q) \cdot N^2 q^{\frac{N-1}{2}},$$

while the forged signature length is

$$\ell \cdot q\big(\log_\gamma(q)\big)^{2N-3} N!\,(N-2)!,$$

where ℓ is the length of the original signature. Here the constant $\gamma \geq 1$ can be chosen by the attacker. They point out that if the parameters N, q are chosen as $q = 2^{16}$ and $N = 14$ then their attack is defeated with time complexity 2^{100}. It is clear that if we choose $q = M_{31} = 2^{31} - 1$, $N = 10$ then the attack is completely defeated with security level $> 2^{128}$ while if we choose $q = M_{61} = 2^{61} - 1$, $N = 10$, then we achieve security level at least 2^{256}. Even with much smaller choices of q, N the attack is still defeated because the forged signatures produced are significantly longer than the actual signatures.

Increasing N and q does affect the performance of WalnutDSA. In a software implementation, each E-Multiplication step requires N multiplications and $2N$ additions within $\mathbb{F}_q$. This means that increasing N from 8 to 10 changes the number of basic operations from 8 to 10 multiplications and 16 to 20 additions, a 25% increase in the number of operations per E-Multiplication.

Increasing N also affects the length of the signature. The length increase can be obtained heuristically through testing. Using $N = 8$ the average length of a signature was 1399 Artin generators whereas increasing to $N = 10$ increased the length to 1909, a 36% increase in signature length (and an equivalent increase in signature verification time due to the 36% increase in the number of E-Multiplications required).

It should be noted that the increase of N also affects the signature storage size, because with $N = 8$ each generator only needs 4 bits, whereas 5 bits are required for $N = 10$. This increases the storage requirements by an additional 25%, for a total storage increase of 70%.

Increasing N and q affect the public key size, because the matrix is an $N \times N$ matrix over $\mathbb{F}_q$, which requires $N^2\log_2(\text{q})$ bits for each matrix. Increasing from $N = 8,\ q = 32$ to $N = 10,\ q = M_{31}$ results in an increase in public key matrices from 320 to 3100 bits each (a 10x increase). However, this 10x increase still results in public keys significantly shorter than the majority of NIST signature candidates.

Finally, increasing q from 32 to M_{31} does change the implementation of operations in $\mathbb{F}_q$. Whereas on F_{32} the operations could be implemented as a table lookup, using M_{31} no longer provides for that option. The primary consideration for performance of $\mathbb{F}_q$ is the state of the multiplier. Specifically, if the platform has a $32 \times 32 \to 64$ bit multiplier then the operation can be performed in only two instructions (multiplication and reduction). Some platforms don't provide this, but do provide a $32 \times 32 \to 32$(high) and $32 \times 32 \to 32$(low) operation. Other platforms truncate the result. And finally, some very small platforms don't provide for a 32-bit multiplier at all. The resulting

performance degredation is determined by the available multiplier. We note that even on small platforms like an ARM Cortex M4, the multiplier is sufficient to compute the result in the single multiply instruction. The use of Mersenne primes like M_{31} affords a simple reduction methodology, which is simply a shift, addition, and possibly overflow subtraction.

All in, the signature verification times of WalnutDSA on the NIST test platform increased from 160,000 to 230,000 cycles due to these changes, a performance degredation of only 43%.

7. The Beullens-Blackburn Forgery Attacks

Following in the footsteps of Hart et al [13], Blackburn and Beullens [10] modified the Hart et al attack to the case $w \neq w'$. The forgeries produces satisfy the length constraints specified in WalnutDSA, but the attacks are exponential in running time and can be completely thwarted by running the protocol on $B_{10}, \mathbb{F}_{M_{31}}$, where M_{31} is the Mersenne prime $2^{31} - 1$ for 128-bit security and $B_{10}, \mathbb{F}_{M_{61}}$ for 256-bit security. Furthermore, even with these increased parameter sizes, the high efficiency and low power consumption advantages of WalnutDSA$^{\text{TM}}$ for constrained devices are still retained.

In addition to the modified factorization attack, Blackburn and Beullens [10] present a birthday attack on reversing E-Multiplication (which is a hard problem underlying the WalnutDSA), together with a collision search forgery method. Each of these attacks is again exponential in running time and is thwarted by running the WalnutDSA$^{\text{TM}}$ protocol on $B_{10}, \mathbb{F}_{M_{31}}$, for 128-bit security and $B_{10}, \mathbb{F}_{M_{61}}$ for 256-bit security.

Factorization Attack: In order to remove the assumption that the private keys (w, w') may not be equal which is required for the Hart et al attack [13] Blackburn and Beullens point out that if one has 3 messages m, m_1, m_2 with $h = \text{Matrix}\big(\mathcal{P}\left(E(H(m))\right)\big)$, $h_1 = \text{Matrix}\big(\mathcal{P}\left(E(H(m_1))\right)\big)$, $h_2 = \text{Matrix}\big(\mathcal{P}\left(E(H(m_2))\right)\big)$, and three private keys w_1, w_2, w_3 then the following holds:

- *If $h = h_1^{-1}$ and s_1 is a valid signature for m_1 under the public key $(\mathcal{P}(v_1), \mathcal{P}(v_2))$ then s_1^{-1} is a valid signature for h under the public key $(\mathcal{P}(v_2), \mathcal{P}(v_1))$;*

- If $h = h_1 h_2$ and s_1, s_2 are valid signatures for m_1, m_2 under the public keys $(\mathcal{P}(v_1), \mathcal{P}(v_2))$, $(\mathcal{P}(v_2), \mathcal{P}(v_1))$, respectively, then $s_1 \cdot s_2$ is a valid signature for m under the public key $(\mathcal{P}(v_1), \mathcal{P}(v_3))$.

Assume the attacker knows message signature pairs (m_i, s_i) and associated matrices $h_i = \text{Matrix}\big(\mathcal{P}\left(E(H(m_i))\right)\big)$, $(i = 1, 2, \ldots, r)$ that are valid under the same public key $(\mathcal{P}(v_1), \mathcal{P}(v_2))$. Then it easily follows that for an odd number of factors

$$s_{i_1} \cdot s_{i_2}^{-1} \quad \cdots \quad s_{i_{L-1}}^{-1} \cdot s_{i_L}$$

is a valid signature under the public key $(\mathcal{P}(v_1), \mathcal{P}(v_2))$ for any message m satisfying

$$\text{Matrix}\big(\mathcal{P}\left(E(H(m))\right)\big) = h_{i_1} \cdot h_{i_2}^{-1} \quad \cdots \quad h_{i_{L-1}}^{-1} \cdot h_{i_L}.$$

At this point the attacker may implement the Hart et al factorization attack [13], but the forged signatures will be way too long and the running time will still be exponential so the attack will be completely thwarted on $B_{10}, \mathbb{F}_{M_{31}}$ and $B_{10}, \mathbb{F}_{M_{61}}$ with security levels at least $2^{128}, 2^{256}$, respectively.

Collision Search Attack:

The second forgery attack introduced in [10] seeks to find two messages m_1, m_2 such that

$$\mathcal{P}\left(E(H(m_1)\right) = \mathcal{P}\left(E(H(m_2)\right).$$

Finding such a collision breaks EUF-CMA. The method used in [10] to find a collision is based on the Oorschot and Wiener algorithm [12] which is a parallelizing collision search algorithm built upon the Pollard rho-method. Clearly, to determine the probability of a collision it is enough to find the size of $\mathcal{P}\Big(E\left(\{0,1\}^*\right)\Big)$. Let P_N denote the pure braid subgroup of B_N whose index in B_N is $N!$. Since the encoding function E takes values in P_N it is enough to obtain the size of $\mathcal{P}(P_N)$ which they estimate as $\approx q^{(N-2)^2+1}$ in [10].

As also pointed out in [10] any braid output by the encoding mechanism E is a product of the image (under $\mathcal{P}$) of the encoding braids used and, thus, it is essential that the subspace spanned by said images is sufficiently large. An example of an encoding that yields sufficient security, and hence, defeats this avenue of attack is given as follows. Let $N = 12$ and let S be the periodic sequence of tuples

$$\{(5,7,9,11),(4,6,8,10),(3,5,7,9),(2,4,6,8),(1,3,5,7),(2,4,6,8),(3,5,7,9),(4,6,8,10),...\}.$$

One can check that this dimension is 122, so using $q = 32$ or 256 results in sufficiently large spaces. For the case of $N = 10$, S can be the sequence $\{(3,5,7,9),(2,4,6,8),(1,3,5,7),(2,4,6,8),...\}$ which results in a dimension of 82.

8. Defeating the Beullens-Blackburn Forgery Attacks

The forgery attacks presented by Blackburn and Beullens in [10] are all exponential in running time as explained above. They can be completely thwarted by running the protocol on $B_{10}, \mathbb{F}_{M_{31}}$ for 128-bit security and $B_{10}, \mathbb{F}_{M_{61}}$ for 256-bit security.

9. Defeating the Beullens-Blackburn REM Attack

The third attack discussed in [10] is an exponential attack to reverse E-Multiplication (REM) which is a hard problem underlying the security of WalnutDSA^{TM}. The running time of this attack is estimated as $q^{N/2-1}$ in [10]. Here is a direct quote from the Blackburn-Beullens paper [10].

"There does not seem to be a better way to block the attack other than just increasing the parameters to ensure that $q^{N/2-1}$ is higher than the desired security level. One way to do this is to take $N = 10, q = 2^{32}$ to achieve 128 bits of security, and $N = 10, q = 2^{64}$ for 256 bits of security."

Recall that the T-values for E-Multiplication are just a subset of N invertible elements in $\mathbb{F}_q$ denoted $\{\tau_1, \tau_2, \ldots, \tau_N\}$. The Beullens-Blackburn REM attack assumed that for two integers a, b with $1 \leq a < b \leq N$ we specify that $\tau_a = \tau_b = 1$. If we instead we choose τ_a, τ_b so that $\tau_a \cdot \tau_b = -1$ then the running time of the REM attack is much higher. In fact, an additional factor of $\sqrt{q} \cdot \sqrt{x}$ is added to the runtime, where x is a parameter in their attack (they set $x = 60$ for $N = 8$ and we expect $x = 96$ for $N = 10$). This results in an (unverified) search time of at least $\sqrt{x} \cdot q^{(N-1)/2}$.

Here again, the parameters $B_{10}, \mathbb{F}_{M_{31}}$ for 128-bit security and $B_{10}, \mathbb{F}_{M_{61}}$ for 256-bit security effectively defeat the REM attack.

10. The Kotov, Menshov, Ushakov Attack

Kotov et al [8] proposed a heuristic algorithm to search for and remove cloaking elements from a WalnutDSA signature to produce a surrogate signer private key that would enable an attacker to forge signatures of any message. We show that by using appropriately chosen cloaking elements with the WalnutDSA signature, we can defeat this attack without any significant degredation to size or performance. The use of these cloaking elements renders WalnutDSA completely secure against this attack.

The attack proceeds by collecting a number of messages, together with their associated WalnutDSA signatures, which have all been generated by a single user whose private key is denoted by (w, w'). Next, a heuristic method is used on each of the signatures to search and remove the specified cloaking elements v, v_1, v_2 from each of the signatures. The search relies on the attacker knowing the permutations that each of the three cloaking elements v, v_1, v_2 are cloaking for. Letting σ denote one of these permutations, the attacker searches for locations in a signature where $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$ are switched (see [1] for the discussion of τ_a, τ_b). This can be explained as follows. Since a braid in B_N is a configuration of strands connecting N equally spaced points on a line with another N equally spaced points on a parallel line, one can search for subwords of the braid with the property that the strand starting at the point $\sigma^{-1}(a)$ crosses the strand starting at the point $\sigma^{-1}(b)$. The attack further assumes that cloaking elements are of the form $ub_i^{\pm 2}u^{-1}$ (i.e., a conjugate) where the permutation associated to u maps i to $\sigma^{-1}(a)$ and $i+1$ to $\sigma^{-1}(b)$. Writing $ub_i^{\pm 2}u^{-1} = ub_i^{\epsilon}b_i^{\epsilon}u^{-1}$ with $\epsilon = \pm 1$ the attack attempts to find the location of b_i^{ϵ} and replaces it with its inverse $b_i^{-\epsilon}$ resulting in the cloaking element turning into $ub_i^{-\epsilon}b_i^{\epsilon}u^{-1} = \mathrm{Id}$ where Id is the identity element in the braid group. If successful, this procedure effectively deletes the cloaking element. These manipulations do not always work. To make the attack more effective Kotov et al [8] perform the above procedure many times on a pair of signatures S_1, S_2 generating two lists of altered signatures $\{S_1^{(1)}, \ldots, S_1^{(k)}\}$, $\{S_2^{(1)}, \ldots, S_2^{(\ell)}\}$. The attack attempts to braid minimize

$$\left(S_1^{(i)}\right) \cdot \left(S_2^{(j)}\right)^{-1}, \qquad (\text{for } 1 \le i \le k,\ 1 \le j \le \ell),$$

which may remove the cloaking element. Since it is assumed that there are only three cloaking elements which cloak for known permutations the heuristic attack proceeds as above to systematically remove the three cloaking elements.

If the three cloaking elements are successfully removed it is then possible to construct a surrogate for the private key (w, w') of the signer as follows. With the cloaking elements removed, the signature of a message m_i takes the form

$$\mathrm{Sig}(m_i) = w^{-1} \cdot E(H(m_i)) \cdot w'.$$

Assuming that the attacker has signatures for k messages, $m_1, \ldots, m_k$, the sequence of products

$$\mathrm{Sig}(m_i) \cdot \mathrm{Sig}(m_{i+1})^{-1} = w^{-1} \cdot E(H(m_i)) \cdot E(H(m_{i+1}))^{-1} \cdot w,$$

yield a set of simultaneous conjugacy equations whose solution will be a surrogate of the signer's private key. This surrogate private key can then be used to forge signatures of further messages.

11. Defeating the Kotov, Menshov, Ushakov attack

The heuristic attack of Kotov et al [8] can be easily defeated by introducing concealed cloaking elements into the WalnutDSA signature. Following [1], we fix a braid β, say

$$\beta = b_{i_1}^{\epsilon_1} \cdots b_{i_\ell}^{\epsilon_\ell},$$

and choose some point $1 \leq k \leq \ell$. Clearly, $\beta = x_1 \cdot x_2$ where $x_1 = b_{i_1}^{\epsilon_1} \cdots b_{i_{k-1}}^{\epsilon_{k-1}}$ and $x_2 = b_{i_k}^{\epsilon_k} \cdots b_{i_\ell}^{\epsilon_\ell}$, and, hence, for any matrix/permutation pair (m_0, σ_0), we have that $(m_0, \sigma_0) \star \beta = ((m_0, \sigma_0) \star x_1) \star x_2$.

We can generate a cloaking element v for the product of $\sigma_0 \cdot \sigma_{x_1}$ where σ_{x_1} deotes the permutation associated with x_1. By construction, given any matrix M we have that $(M, \sigma_0 \cdot \sigma_{x_1}) \star v = (M, \sigma_0 \cdot \sigma_{x_1})$. Since $(m_0, \sigma_0) \star x_1$ takes the form $(m_0, \sigma_0) \star x_1 = (M, , \sigma_0 \cdot \sigma_{x_1})$. It follows that

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

Hence we have generated a new braid β' which contains v,

$$\beta' = x_1 \cdot v \cdot x_2,$$

which has the property that $(m_0, \sigma_0) \star \beta = (m_0, \sigma_0) \star \beta'$. We shall refer to this inserted cloaking element as a *concealed* cloaking element.

It is the presence of κ concealed cloaking elements (for sufficiently large κ) that effectively blocks this attack. The key point is that for concealed cloaking elements we do not know the permutation that is being cloaked. In general, knowing that κ concealed cloaking elements have been placed in a nested fashion in a known braid, it would require $(N!)^\kappa$ searches to find them. To insure κ-bit security we would require

$$(N!)^\kappa > 2^\kappa,$$

and hence

$$\kappa > \text{Security Level} / \log_2(N!).$$

We have explored possible birthday attacks and have ruled out obvious ways to use a birthday attack to discover all the concealed cloaking elements. Indeed, multiple cloaking elements could use the same permutation but each would still need to individually be discovered. Without access to a birthday attack, in the case of $N = 10$, and a security level of 128 we can comfortably take $\kappa = 6$ (which results in a work factor of $2^{130.74}$). Likewise, when $N = 10$ and the security level is 256, taking $\kappa = 12$ is sufficient (resulting in a work factor of $2^{261.49}$).

We also note that concealed cloaking elements have a secondary purpose in blocking this attack. Recall that the attack not only relies on knowing the permutation being cloaked, but it also relies on a cloaking element being in the form of a conjugate. By placing a concealed cloaking element inside one side, e.g. converting $v = u b_i^{\pm 2} u^{-1}$ to $v = \kappa(u) b_i^{\pm 2} u^{-1}$, we block the attack in both ways. Specifically, while the permutation v is cloaking for is known, it is no longer a conjugate, and while the inner-most concealed cloaking element is a conjugate, the permutation it is cloaking for is not known.

With $N = 10$, a cloaking element using a random permutation for u averages 87.16 Artin generators (with a standard deviation of 22.03). This can be shortened by choosing the permutation of u carefully (note that this is different than the permutation being cloaked). If we add 6 concealed cloaking elements (necessary for 128-bit security), this implies an average signature-size increase of approximately 523 generators. However, after running BKL and Dehornoy, additional size reductions can be made. This results in an average signature increase from 1909 to 2037 Artin generators, or an increase in only 6.7%.

Because signature validation performance is linearly correlated with the length of the signature, this 6.7% average length increase results in a 6.7% increase in the average time required to validate signatures.

12. The Merz–Petit Attack and Mitigation

Recently, Merz and Petit [6] proposed a practical forgery attack on WalnutDSA$^{\text{TM}}$. They found that using the Garside Normal Form of the signature allowed them to find commonalities with the Garside form of the encoded message, and using those commonalities they could create a forgery. As pointed out by the authors, the attack can be defeated by adding cloaking elements into the encoded message. Specifically, they conjecture that each additional cloaking element effectively mutates approximately five (5) permutation braids in the Garside Normal Form, but, when mutated, their attack no longer succeeds. We will confirm that sufficient insertions of cloaking elements will prevent the attack from producing forgeries.

Before describing the Garside based approach proposed by Merz–Petit [6] we review some of the basic components Garside introduced to the field which date back to 1965. Recalling that the Artin presentation of the N strand braid group has generators $\{b_1, b_2, \ldots, b_{N-1}\}$, subject to the following relations:

$$\begin{aligned} b_i b_{i+1} b_i &= b_{i+1} b_i b_{i+1}, \qquad (i = 1, \ldots, N-2), \\ b_i b_j &= b_j b_i, \qquad (|i-j| \geq 2). \end{aligned}$$

A brief summary of Garside's approach [5] proceeds as follows. The fundamental braid Δ_N, which is defined to be

$$\Delta_N = (b_1 \cdots b_{N-1})(b_1 \cdots b_{N-2}) \cdots (b_1 b_2)\, b_1,$$

satisfies the properties: for $i = 1, \ldots, N-1$,

$$b_i \Delta = \Delta b_{N-i} \quad b_i^{-1} = x_i \Delta^{-1},$$

where x_i is a positive word in the generators (i.e., a word without negative exponents). Focussing on positive words in the braid generators, denoted B_N^+ enabled Garside to introduce an ordering of positive words: given two positive words $a, b \in B_N^+$ then $a \leq b$ if there exists a $c \in B_N^+$ such that $ac = b$. Further, given said $a, b \in B_N^+$ we can look for the smallest positive braid d such that $d \leq a$ and $d \leq b$. Garside proved such a smallest d exists and is unique (it is often denoted $a \wedge b$). Garside's seminal theorem states that every braid β can be uniquely expressed in the form

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

where $r \in \mathbb{Z}$, $1 < A_i < \Delta$, and $A_i\, A_{i+1} \wedge \Delta = A_i$.

The underlying mathematical structure supporting the WalnutDSA protocol is the action of the braid group on a direct product of a large finite matrix group and a symmetric group. The action

is inherently algorithmically difficult to reverse, and finding stabilizers (termed cloaking elements) is likewise a difficult problem. However, specialized classes of cloaking elements can be explicitly generated and it is, hence, possible to use them as a means of obscuring a braid: by inserting sufficiently many cloaking elements the structure of the original braid cannot be recovered in a tractable way.

The Merz and Petit universal forgery attack is a heuristic method that, using knowledge of a valid signature of a message M, aims to generate a signature of a second message M' that will be validated by a receiver. The decomposition algorithm introduced in their paper (which uses the Garside canonical for as its basis) can be applied because a Walnut signature has the form

$$W_1 E(H(M)) W_2$$

and, critically, the braid element $E(H(M))$ is known to everyone. Knowledge of $E(H(M))$ allows the algorithm to try to derive braids W_1', W_2' which satisfy the conditions $W_i \equiv W_i'$ (Mod Δ^2), and $W_1 \cdot W_2 = W_1' \cdot W_2'$. Once a forger has said elements in place, the braid $W_1' \cdot E\big(H(M')\big) \cdot W_2'$ will verify as a signature of a message M'.

In fact, knowledge of the entire $E(H(M))$ is not actually requisite. Were one to insert a single concealed cloaking element into the encoding $E(H(M))$ it is still possible that the A_i's in the Garside normal form (see above) of said encoding still appear in the Garside normal form for the signature. While the forgery in this case would be longer than the average signature, it might be within the acceptable length range. Thus, in order to completely thwart the heuristic attack, the signer must insert sufficiently many concealed cloaking elements into the braid $E(H(M))$ to completely alter the Garside normal form. We have done significant testing and have concluded that inserting cloaking elements every 7-12 generators will successfully block this attack. It should be noted that the approaches to removing cloaking elements required the attacker to be able to reduce the problem to a conjugacy search problem, Finding concealed cloaking elements in the encoded message does not fit into that effort.

13. Conclusion

The Walnut Digital Signature Algorithm (WalnutDSA) is a group-theoretic, public-key method which has been introduced as a means for signing messages and verifying signatures when legacy methods are not viable due to the nature of the platform in question. This paper demonstrated that WalnutDSA is particularly effective on platforms where legacy methods are not viable because of platform constraints. By making appropriate parameter choices, i.e., $N = 10$, $p = 2^{31} - 1$, and by ensuring that the cardinality of the set of concealed cloaks are sufficiently large, WalnutDSA is secure and immune to the range of attacks reviewed in this paper. Moreover, the protocol maintains its novel feature of being viable on embedded and low resource devices.

References

[1] Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.E.:WalnutDSA$^{\text{TM}}$: a quantum-resistant digital signature algorithm. Cryptology ePrint Archive, Report 2017/058 (2017).

[2] E. Artin, Theory of braids, *Ann. of Math.* (2) 48 (1947), 101–126.

[3] J. Birman; K. H. Ko; S. J. Lee, A new approach to the word and conjugacy problems in the braid groups, Adv. Math. 139 (1998), no. 2, 322–353.

[4] P. Dehornoy, A fast method for comparing braids, Adv. Math. 125 (1997), no. 2, 200–235.

[5] Garside, F.A., The braid group and other groups. The Quarterly Journal of Mathematics bf 20(1), 235–254 (1969).

[6] Merz S.P., Petit C. (2018) *Factoring Products of Braids via Garside Normal Form,* https://eprint.iacr.org/2018/1142.

[7] H.R. Morton, The multivariable Alexander polynomial for a closed braid, Low-dimensional topology, (Funchal, 1998), 167–172, Contemp. Math., 233, Amer. Math. Soc., Providence, RI, 1999.

[8] M. Kotov; A. Menshov; A. Ushakov, An attack on the Walnut digital signature algorithm, Cryptology ePrint Archive: Report 2018/393 (2018).

[9] I. Anshel, D. Atkins, D. Goldfeld, P.E. Gunnells, Defeating the Hart, Kim, Micheli, Pascuel-Perez, Petit, Quek Attack on WalnutDSA$^{\mathrm{TM}}$, To appear.

[10] S. Blackburn, W. Beullens, Practical attacks against the Walnut digital signature scheme, Cryptology ePrint Archive, Report 2018/318 (2018).

[11] D. Hart, D. Kim, G. Micheli, G. Pascual-Perez, C. Petit, Y. Quek. A Practical Cryptanalysis of WalnutDSA TM. In: Abdalla M., Dahab R. (eds) *Public-Key Cryptography – PKC 2018. PKC 2018.* Lecture Notes in Computer Science, vol 10769. Springer, Cham. (2018)

[12] P.C. Van Oorschot, M.J. Wiener, Parallel collision search with cryptanalytic applications. Journal of cryptology 12(1), 1–28 (1999).

[13] Hart D., Kim D., Micheli G., Pascual-Perez G., Petit C., Quek Y. (2018) A Practical Cryptanalysis of WalnutDSA TM. In: Abdalla M., Dahab R. (eds) *Public-Key Cryptography – PKC 2018. PKC 2018.* Lecture Notes in Computer Science, vol 10769. Springer, Cham.

Email address: IANSHEL@SECURERF.COM, DATKINS@SECURERF.COM, DGOLDFELD@SECURERF.COM, PGUNNELLS@SECURERF.COM