ATTACK ON KAYAWOOD PROTOCOL: UNCLOAKING PRIVATE KEYS

MATVEI KOTOV, ANTON MENSHOV, AND ALEXANDER USHAKOV

ABSTRACT. We analyze security properties of a two-party key-agreement protocol recently proposed by I. Anshel, D. Atkins, D. Goldfeld, and P. Gunnels, called Kayawood protocol. At the core of the protocol is an action (called *E-multiplication*) of a braid group on some finite set. The protocol assigns a secret element of a braid group to each party (private key). To disguise those elements, the protocol uses a so-called cloaking method that multiplies private keys on the left and on the right by specially designed elements (stabilizers for E-multiplication).

We present a heuristic algorithm that allows a passive eavesdropper to recover Alice's private key by removing cloaking elements. Our attack has 100% success rate on randomly generated instances of the protocol for the originally proposed parameter values and for recent proposals that suggest to insert many cloaking elements at random positions of the private key. Implementation of the attack is available on GitHub.

Keywords. Kayawood protocol, group-based cryptography, key agreement, algebraic eraser, braid group, colored Burau presentation, E-multiplication, cloaking problem.
2010 Mathematics Subject Classification. 94A60, 68W30.

1. Introduction

Braid group cryptography received significant attention since invention of the first braid-based key-agreement protocols in 1999: Ko-Lee protocol [17] and Anshel-Anshel-Goldefeld protocol [2]. Both protocols use conjugation as main operation, and both were found vulnerable to linear attacks (such as [10] and [26]) and heuristic length-based attacks (such as [16, 14, 20, 21, 23]).

Kayawood protocol (and other protocols from its family: Algebraic Eraser proposed in [3], WalnutDSA proposed in [7], and Ironwood proposed in [5]) uses a different type of action, called E-multiplication, and utilizes commuting actions of **non-commuting** braids. This is what in our opinion distinguishes Kayawood from "classic" braid-based schemes such as Ko-Lee.

1.1. **Kayawood protocol.** The Kayawood protocol is a two-party (Alice and Bob) key-agreement protocol recently proposed by I. Anshel, D. Atkins, D. Goldfeld, and P. Gunnells in [6]. The design of the protocol is very similar to the design of the digital signature algorithm WalnutDSA [7] that has been accepted by the National Institute of Standards and Technology

Date: April 26, 2019.

for evaluation as a standard for post-quantum, public-key cryptography. At the core of the protocol is an action (called *E-multiplication*) of the group B_n of braids on n strands on some finite set which is claimed to be a suitable primitive for use within lightweight cryptography.

By design, Alice and Bob's private keys are braids from two commuting subgroups of B_n. To disguise the private keys, the (original version of the) protocol uses a so-called cloaking method that multiplies the keys on the left and on the right by specially designed elements (stabilizers for E-multiplication) and applies a certain rewriting procedure to obfuscate the result. Recently, after a series of attacks on WalnutDSA ([15, 8, 19]), the authors proposed several changes to the protocol including changes to the cloaking procedure (see [1, D. Atkins on May 23, 2018]). Namely, they suggested to use several cloaking element inserted into the private keys at random positions.

In this paper, we show that the cloaking elements can be efficiently identified and removed from public keys, and private keys can be reconstructed. Furthermore, following the suggestion from [1, D. Atkins on May 23, 2018], we show that private keys can be reconstructed even when many cloaking elements are inserted at random positions. Our attack has 100% success rate on randomly generated instances of the protocol, and its implementation is available on GitHub [11].

2. Action of colored Burau group on some finite set

Here we review one non-faithful representation of a braid group called the *colored Burau group.*

2.1. **Braid group.** In this section we follow the exposition of [22, Section 5.1]. A braid is obtained by laying down a number of parallel pieces of strands and intertwining them, without loosing track of the fact that they run essentially in the same direction. In our pictures the direction is horizontal. We number strands at each horizontal position from the top down. See Figure 1 for example.

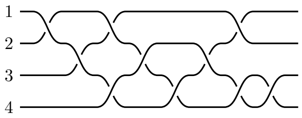

Figure 1. A 4-strand braid.

If we put down two braids u and v in a row, so that the end of u matches the beginning of v, we get another braid denoted by uv, i.e., concatenation of n-strand braids is a product. We consider two braids equivalent if there exists an isotopy between them, i.e., it is possible to move the strands of one of the braids in space (without moving the endpoints of strands and moving strands through each other) to get the other braid. We distinguish a special n-strand braid which contains no crossings and call it a trivial braid. Clearly, the trivial braid behaves as left and right identity relative to the defined multiplication. The set B_n of isotopy classes of n-strand braids has a group structure, because if we concatenate a braid with its mirror image in a vertical plane, the result is isotopic to the trivial braid.

Each braid is uniquely defined by a sequence of strand crossings. A crossing is called positive if the front strand has a positive slope, otherwise it is called negative. There are exactly $n-1$ crossing types for n-strand braids, we denote them by $x_1, \ldots, x_{n-1}$, where x_i is a positive crossing of ith and $(i+1)$st strands. As we mentioned above, any braid is a sequence of crossings, and the set $\{x_1, \ldots, x_{n-1}\}$ generates B_n. It is easy to see that crossings $x_1, \ldots, x_{n-1}$ are subject to the relations

$$\begin{array}{ll} x_i x_j = x_j x_i & \text{for } i,\ j \text{ such that } |i-j|>1 \\ x_i x_{i+1} x_i = x_{i+1} x_i x_{i+1} & \text{for } 1 \le i \le n-2. \end{array}$$

In fact, it can be shown that the relations above define equivalence relation on braids and, hence, B_n has the following combinatorial presentation:

$$B_n \simeq \left\langle x_1, \ldots, x_{n-1} \;\middle|\; \begin{array}{l} x_i x_j = x_j x_i \text{ for } |i-j|>1, \\ x_i x_{i+1} x_i = x_{i+1} x_i x_{i+1} \end{array} \right\rangle.$$

It easily follows from the presentation above that elements in the subgroups $L_n = \langle x_1, \ldots, x_{m-1} \rangle$ and $U_n = \langle x_{m+1}, \ldots, x_{n-1} \rangle$ pairwise commute, where $m = \lfloor \frac{n}{2} \rfloor$.

A *braid word* is a word $w = w(x_1, \ldots, x_{n-1})$ in the generators of B_n and their inverses:

$$w = x_{i_1}^{\varepsilon_1} \ldots x_{i_k}^{\varepsilon_k}, \tag{1}$$

where $1 \le i_j \le n-1$ and $\varepsilon_j = \pm 1$. The length of the braid word (1) is k, denoted by $|w|$. If $\overline{u} = (u_1, \ldots, u_k)$ is a k-tuple of braid words, then the total length $|\overline{u}|$ of $\overline{u}$ is defined as $\sum_{i=1}^{k} |u_i|$.

Every braid w naturally defines a permutation σ_w, which is a permutation of the endpoints of the involved strands. The corresponding map $w \mapsto \sigma_w$ is an epimorphism. If σ_w is trivial, then w is called a *pure* braid.

Recall that the *commutator* of braid words u and v is the braid word $[u,v] = u^{-1}v^{-1}uv$. For a set of braids $u_1, \ldots, u_k$ define a set

$$C(u_1, \ldots, u_k) = \{c \in B_n \mid [c, u_i] = 1 \text{ for every } 1 \le i \le k\},$$

called the *centralizer* of $u_1, \ldots, u_k$. It is easy to check that a centralizer is a subgroup of B_n.

The group B_n has a cyclic center generated by the element Δ^2, where Δ is the element, called the *half twist*, defined as follows:

$$\Delta = (x_1 \ldots x_{n-1}) \cdot (x_1 \ldots x_{n-2}) \cdot \ldots \cdot (x_1).$$

2.2. **Geodesic braid approximation.** Let w be a word in generators of B_n. The algorithmic problem to find a shortest braid word representing the same element as w, called *geodesic*, is known to be computationally hard (see [25]). In this paper, following [20, 21], we use a geodesic-braid approximation method to estimate the geodesic length of a braid. The algorithm attempts to minimize the given braid word exploiting the property of Dehornoy's form $D(w)$ (introduced in [12]) that for a "generic" braid word w one has $|D(w)| < |w|$.

2.3. **Colored matrices.** Fix a finite field $\mathbb{F}_q$ and denote by R_n the ring of Laurent polynomials in variables $\{t_1, \dots, t_n\}$ with coefficients in $\mathbb{F}_q$. Let $\mathbf{GL}_n(R_n)$ be the group of invertible matrices over R_n. The symmetric group S_n naturally acts on $\mathbf{GL}_n(R_n)$ by permuting the variables $\{t_1, \dots, t_n\}$. The result of action of $\sigma \in S_n$ on $M \in \mathbf{GL}_n(R_n)$ is denoted by M^σ. Recall that the semidirect product of $\mathbf{GL}_n(R_n)$ and S_n is a group

$$\mathbf{GL}_n(R_n) \rtimes S_n = \{(M, \pi) \mid M \in \mathbf{GL}_n(R_n) \text{ and } \pi \in S_n\},$$

equipped with the operation

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

Define $n-1$ $n \times n$-matrices over polynomials in variables $\{t_1, \dots, t_n\}$:

$$C_1(t_1) = \left(\begin{array}{cc|c} -t_1 & 1 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & I_{n-2} \end{array}\right) \quad \text{and} \quad C_i(t_i) = \left(\begin{array}{c|ccc|c} I_{i-2} & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & t_i & -t_i & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & I_{n-i-1} \end{array}\right)$$

for $2 \le i \le n-1$.

Lemma. A map φ on the generators $x_1, \dots, x_{n-1}$ of B_n:

$$x_i \overset{\varphi}{\mapsto} (C_i(t_i), \pi_i),$$

where $\pi_i = (i, i+1) \in S_n$, extends into a group homomorphism.

The group $\langle (C_1(t_1), \pi_1), \dots, (C_{n-1}(t_{n-1}), \pi_{n-1}) \rangle$ is called the *colored Burau representation* of B_n and is denoted by CB_n.

2.4. **Action of CB_n on a certain finite set.** Fix n nontrivial elements $\tau_1, \dots, \tau_n \in \mathbb{F}_q$, termed t-values, and define an evaluation map

$$\epsilon : \mathbf{GL}_n(R_n) \to \mathbf{GL}_n(\mathbb{F}_q),$$

that for each i replaces t_i with the value τ_i. For $(M, \sigma) \in \mathbf{GL}_n(\mathbb{F}_q) \times S_n$ and $(C, \rho) \in \mathrm{CB}_n$ define the following element:

$$(M, \sigma) \star (C, \rho) = (M \cdot \epsilon(C^\sigma), \sigma\rho).$$

It is straightforward to check that the map $\star$ defines an action of CB_n on $\mathbf{GL}_n(\mathbb{F}_q) \times S_n$. By E-multiplication we understand the induced action of B_n on $\mathbf{GL}_n(\mathbb{F}_q) \times S_n$.

2.5. **Cloaking elements.** Let G be a group acting on a set X, $x \in X$, and $x^g \in X$ denotes the result of the action of $g \in G$ on x. The *stabilizer* of x is the set

$$\mathbf{Stab}(x) = \{g \in G \mid x^g = x\}.$$

It is easy to check that $\mathbf{Stab}(x)$ is a subgroup of G. The protocol [6] requires braids stabilizing some $(M, \sigma) \in \mathbf{GL}_n(\mathbb{F}_q) \times S_n$ through the right action of the braid group via E-multiplication. Such braids are called *cloaking elements* in [6, Definition 2.1]. Observe that these elements depend on t-values that are used to define E-multiplication. The following way of constructing cloaking elements was proposed in [6].

Proposition 2.1 ([6, Proposition 2.2])**.** *Fix $(M, \sigma) \in \mathbf{GL}_n(\mathbb{F}_q) \times S_n$ and assume that $a, b, i \in \mathbb{N}$ and $w \in B_n$ satisfy the following conditions:*

$$1 \le a < b \le n \text{ and } \tau_a = \tau_b = 1,$$
$$1 \le i < n \text{ and } \sigma_w(i) = \sigma^{-1}(a), \sigma_w(i+1) = \sigma^{-1}(b).$$

Then $wx_i^{\pm 2}w^{-1} \in \mathbf{Stab}((M, \sigma))$.

The main purpose of a cloaking element is to "cloak" a braid A that acts on a given pair (M, σ), multiplication of A on the left by a cloaking element hides some structure of A without changing the way it acts. Observe that the property of a braid to cloak (M, σ) depends on σ only. Hence, we can denote the subgroup of $\mathbf{Stab}((M, \sigma))$ generated by cloaking elements from Proposition 2.1 by C_σ.

The following naturally follows from Proposition 2.1.

Corollary 2.2. *If $\sigma, \rho \in S_n$ are such that $\sigma^{-1}(a) = \rho^{-1}(a)$ and $\sigma^{-1}(b) = \rho^{-1}(b)$, then $C_\sigma = C_\rho$.*

Remark 2.3. Geometrically, conditions of Proposition 2.1 define a braid that:

- intertwists strands getting strands a and b next to each other using w,
- double twists a and b using x_i^2,
- intertwists strands backwards using w^{-1}.

The obtained braid has the structure as shown in Figure 2.

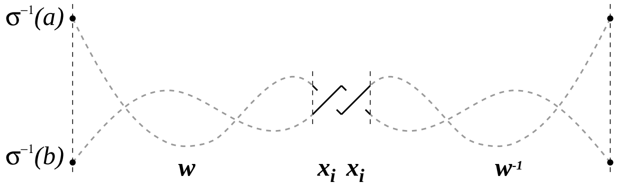

FIGURE 2. Cloaking element.

Another way to generate cloaking elements was suggested in [1, D. Atkins on April 4, 2018], see [19, Proposition 2.3]. We do not consider elements of this type here since they are similar to elements of Proposition 2.1, and cryptanalysis [19] showed they are less secure.

2.6. **Braid word obfuscation.** An *obfuscation procedure* $\mathcal{R}$ for braids is an algorithm that rewrites a braid word w into a braid word $\mathcal{R}(w)$ satisfying $w =_{B_n} \mathcal{R}(w)$. The main goal of an obfuscation procedure is to modify and hide information in the public keys generated by Alice and Bob. There are several methods suggested in [6].

- Stochastic rewriting process described in [6, Section 7].
- Dehornoy form [12].
- BKL normal forms [9].
- Garside normal forms [13, Chapter 9].

BKL normal forms and Garside normal forms provide a unique form for a given braid (i.e., if $u = v$ in B_n, then their normal forms are the same) and, hence, are the strongest possible obfuscation algorithms for braids.

3. Kayawood protocol

Kayawood protocol is a two-party, Alice and Bob, key-agreement protocol that uses E-multiplication defined in Section 2.4. The following *initial public information* is generated by one of the parties or by another entity (and distributed to each party):

- The braid group B_n, where $n \geq 16$ is even.
- Obfuscation procedures $\mathcal{R}$.
- A finite field $\mathbb{F}_q$.
- Integers a and b satisfying $1 \leq a < b \leq n$.
- Non-zero elements $\tau_1, \ldots, \tau_n \in \mathbb{F}_q$ such that $\tau_a = \tau_b = 1$.

Then Alice generates the following private data:

- $\beta_1, \ldots, \beta_r \in U_n$ such that $\sigma_{\beta_1}, \ldots, \sigma_{\beta_r}$ have high order.
- $z \in B_n$ such that $|\sigma_z(\{1, \ldots, m\}) \cap \{1, \ldots, m\}| \approx m/2$. Recall that $m = \lfloor \frac{n}{2} \rfloor$.
- Her private key is $A = z\alpha z^{-1}$, where $\alpha \in L_n$.

Key establishment:

(1) Alice sends to Bob $\{\mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1})\}$ and σ_A.

(2) Bob performs the following:
   - Generates his private key B as a random product of elements $\mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1})$ and their inverses.
   - Generates random $v_1, v_2 \in B_n$ cloaking σ_A and $\sigma_A\sigma_B$ respectively.
   - Sends his public key $P_B = \mathcal{R}(v_1 B v_2)$ to Alice.

(3) Alice performs the following:
   - Computes $\sigma_B = \sigma_{P_B}$.

- Generates random $u_1, u_2 \in B_n$ cloaking σ_B and $\sigma_B\sigma_A$ respectively.
- Sends her public key $P_A = \mathcal{R}(u_1 A u_2)$ to Bob.

(4) Finally, the shared key is

$$(I, 1) \star A \star B$$

computed by Alice as $(I, 1) \star A \star P_B$ and by Bob as $(I, 1) \star B \star P_A$.

We say that a protocol is secure against a passive eavesdropper if there is no probabilistic polynomial time algorithm that can compute the shared key $(I, 1) \star A \star B$ based on the public information exchanged by the parties, namely:

$$n, q, a, b, \tau_1, \ldots, \tau_n, \{\mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1})\}, P_A, P_B$$

The corresponding computational problem can be approached on two different levels: matrices and braids.

4. Passive attack: finding Alice's private key

In [6, Section 5] the authors show that the problem of computing the shared key based on public data is polynomial-time equivalent to the *cloaking problem* formulated as follows.

Cloaking problem. Given a braid $\beta = \mathcal{R}(v_1 \beta_0 v_2)$, where v_1, v_2 are cloaking elements for known permutations and β_0 is a braid in an unknown subgroup of B_n, find the element $(I, \mathrm{id}) \star \beta_0$.

The authors of [6] claim that there is no known approach to the problem and even brute-force enumeration will result in a collection of possible pairs $(I, \mathrm{id}) \star \beta_0$ and there is no a priori way to decide which β_0 is correct. In this section we show that the last statement is incorrect and reduce security of Kayawood protocol to some clearly stated problem of computational group theory.

Proposition 4.1. *Consider $P_A, P_B \in B_n$ and $\sigma_A, \sigma_B \in S_n$ as defined in Section 3. Suppose that $u_1' \in C_{\sigma_B}$ and $u_2' \in C_{\sigma_B\sigma_A}$ satisfy the system:*

$$\begin{cases} [u_1' P_A u_2', z\beta_1 z^{-1}] = 1 \\ \ldots \\ [u_1' P_A u_2', z\beta_r z^{-1}] = 1 \end{cases}$$

Then the shared key is equal to $(I, \mathrm{id}) \star (u_1' P_A u_2') \star P_B$.

Proof. Straightforward check:

$$\begin{aligned} (I,1) \star u_1' P_A u_2' \star P_B &= (I,1) \star u_1' P_A u_2' \star v_1^{-1} P_B v_2^{-1} && \text{(by definition of } v_1, v_2\text{)} \\ &= (I,1) \star v_1^{-1} P_B v_2^{-1} \star u_1' P_A u_2' && \text{(since } [v_1^{-1} P_B v_2^{-1}, u_1' P_A u_2'] = 1\text{)} \\ &= (I,1) \star B \star u_1' P_A u_2' && \text{(since } v_1^{-1} P_B v_2^{-1} = B\text{)} \\ &= (I,1) \star B \star A && \text{(by definition of } u_1', u_2'\text{).} \end{aligned}$$

Observe that $[v_1^{-1}P_Bv_2^{-1}, u_1'P_Au_2'] = 1$ since $v_1^{-1}P_Bv_2^{-1} = B$ is a product of $z\beta_iz^{-1}$ and their inverses. □

Proposition 4.1 implies that the element $u_1'P_Au_2'$ can be used instead of Alice's private key in communication with Bob. By definition of P_A such u_1' and u_2' exist, namely u_1^{-1} and u_2^{-1}. Furthermore, the next proposition claims that we may assume $u_2' = 1$.

Proposition 4.2. *If $u_1 \in C_{\sigma_B}$ and $u_2 \in C_{\sigma_B\sigma_A}$, then*

$$u_1Au_2 = u_1Au_2A^{-1} \cdot A,$$

where $u_1Au_2A^{-1} \in C_{\sigma_B}$.

Proof. Clearly, $(I, \mathrm{id}) \star B \star A \star u_2 = (I, \mathrm{id}) \star B \star A$. Hence, $Au_2A^{-1} \in C_{\sigma_B}$. □

In other words, multiplying A on the right by an element cloaking $\sigma_B\sigma_A$ is the same as multiplying A on the left by an element cloaking σ_B. The same is true if we insert a cloaking element in the middle of $A = A_1 \circ A_2$:

$$A_1 \circ A_2 \to A_1 \circ u \circ A_2,$$

where u cloaks $\sigma_B\sigma_{A_1}$. Insertion of u can be viewed as multiplication of A on the left by an element cloaking σ_B.

Corollary 4.3. *The intersection*

$$C_{\sigma_B}P_A \cap C(z\beta_1z^{-1}, \ldots, z\beta_rz^{-1}) \tag{2}$$

is not empty. Each of its elements plays the role of the Alice's private key.

Proof. Follows from Propositions 4.1 and 4.2. □

Corollary 4.3 allows us to reformulate the cloaking problem as the following algorithmic question.

Cloaking problem for Alice (CPA). Given braids $b_1, \ldots, b_r$ commuting with an unknown braid A, a permutation σ_B associated with an unknown braid $B \in \langle b_1, \ldots, b_r\rangle$, and a braid $P_A \in C_{\sigma_B}A$, find any element in the intersection $C_{\sigma_B}P_A \cap C(b_1, \ldots, b_r)$.

An instance of CPA can be viewed as a tuple

$$(b_1, \ldots, b_r, \sigma_B, P_A) \tag{3}$$

satisfying the conditions mentioned in the statement of the problem. A solution for that instance is any element from the intersection

$$C_{\sigma_B}P_A \cap C(b_1, \ldots, b_r).$$

This defines the basic idea of our attack: we uncloak Alice's public key (solving the CPA problem) and obtain a substitute for her private key.

5. Conjugating instances of the cloaking problem: search for the secret conjugator z

In this section we show that design of the Kayawood protocol leaves us some freedom to manipulate with the secret conjugator z efficiently reducing its length to much smaller values (see Tables 1 and 2). We would like to stress out from the beginning that the problem of finding the exact element z based on the available public data is futile (see Section 5.2), and we never approach that problem. Instead, we are looking for a "sufficiently good" substitute for z.

Remark 5.1. The original proposal [6] does not address the importance of the element z, it simply prescribes to use some randomly generated z of length $[150, 400]$. It is not explained why it is not secure to use z of length, say, 50 (or even 0). The only possible explanation is that the attack [23] (recently improved in [18] for conjugators of length 1000) does not work for elements of length greater than 150. But [23] attempts to solve a different problem, namely, conjugacy separation of braid-tuples modulo Δ^2, where Δ^2-recovery is the hardest part.

5.1. **Conjugating an instance of CPA.**

Given a CPA instance (3), and conjugating $C_{\sigma_B}P_A \cap C(b_1,\dots,b_r)$ by some element $c \in B_n$, we get

$$\begin{aligned} c^{-1}(C_{\sigma_B}P_A \cap C(b_1,\dots,b_r))c &= c^{-1}(C_{\sigma_B}P_A)c \cap c^{-1}C(b_1,\dots,b_r)c \\ &= C_{\sigma_B\sigma_c}c^{-1}P_Ac \cap C(c^{-1}b_1c,\dots,c^{-1}b_rc), \end{aligned}$$

which proves the following proposition.

Proposition 5.2. *$A \in B_n$ is a solution of the instance $(b_1,\dots,b_r,\sigma_B,P_A)$ if and only if $c^{-1}Ac$ is a solution of the instance $(c^{-1}b_1c,\dots,c^{-1}b_rc,\sigma_B\sigma_c,c^{-1}P_Ac)$.*

Notice that conjugating the instance

$$(z\beta_1z^{-1},\dots,z\beta_rz^{-1},\sigma_B,u_1z\alpha z^{-1}u_2) \tag{4}$$

by $c = z$ produces the instance

$$(\beta_1,\dots,\beta_r,\sigma_B\sigma_z,(z^{-1}u_1z)\alpha(z^{-1}u_2z)) \tag{5}$$

with (unknown) cloaking elements $z^{-1}u_1z$ and $z^{-1}u_2z$. In particular, knowledge of z allows to "drop" it from consideration. Also, observe that α is a solution to the latter instance.

In our experiments we were never able to find the exact z, but we were able to find an element c such that $\delta = z^{-1}c$ is a relatively short braid (relative to $|z|$) in the standard word metric on B_n. Conjugating (4) by $c = z\delta$ produces an instance

$$(\delta^{-1}\beta_1\delta,\dots,\delta^{-1}\beta_r\delta,\sigma_B\sigma_c,\delta^{-1}z^{-1}u_1z\alpha z^{-1}u_2z\delta) \tag{6}$$

instead of (5). Observe that

- the instance (4) has a solution $z\alpha z^{-1}$.

- the instance (6) has a solution $\delta\alpha\delta^{-1}$, which is much shorter than $z\alpha z^{-1}$.

Hence, the new instance (6) is more advantageous for a heuristic solver described in Section 6 than the instance (4). Below we describe how we find an appropriate element c such that $\delta = z^{-1}c$ is short.

5.2. **Heuristic search for z.** A part of the public data available to the eavesdropper Eve includes the elements $\beta_1', \ldots, \beta_r'$:

$$\begin{cases} \beta_1' = z\beta_1 z^{-1}, \\ \ldots \\ \beta_r' = z\beta_r z^{-1}, \end{cases}$$

where $\beta_1, \ldots, \beta_r \in L_n$ and z are unknown. Based on this data it is impossible to recover the original element z. Even if the elements $\beta_1, \ldots, \beta_r$ are given, we can only find z modulo the centralizer of $\beta_1, \ldots, \beta_r$. But, as we mentioned before, it is not our goal to find the exact z. Instead, we attempt to find an element c such that $|z^{-1}c|$ is relatively small. This task is approached heuristically using ideas described in [24] and its advanced version [18].

Below we outline a procedure that for a given tuple $(\beta_1', \ldots, \beta_r')$ searches for y that minimizes the total length of the conjugate tuple:

$$\sum_{i=1}^{r} |y^{-1}\beta_i' y| \to \min .$$

The procedure constructs a set of conjugates of the tuple $(\beta_1', \ldots, \beta_r')$. Initially, the set contains $(\beta_1', \ldots, \beta_r')$ only. On each iteration it chooses an unchecked tuple, let us call it $(\gamma_1, \ldots, \gamma_r)$, of the least total length and conjugates the tuple by each generator and its inverse $x_1^{\pm 1}, \ldots, x_{n-1}^{\pm 1}$:

$$(x_i^{\pm 1}\gamma_1 x_i^{\mp 1}, \ldots, x_i^{\pm 1}\gamma_r x_i^{\mp 1}),$$

All words are minimized using the braid minimization procedure and new tuples are saved as unchecked conjugates of $(\beta_1', \ldots, \beta_r')$. We say that an iteration is *successful* if the total length of one of the new tuples is less than the total length of any checked tuple. We *terminate* the procedure after 20 unsuccessful iterations. The output is a checked tuple of the least total length. The described procedure does not fail, but, in principle, it can produce a poor result.

To *accelerate convergence* to a (local) minimum, we perform the following trick (cf. [18, Section 4.4]). On each iteration, if $|\gamma_1| > 50$, then we take the initial segment c of γ_1 of length 50 and add the tuple $(c^{-1}\gamma_1 c, \ldots, c^{-1}\gamma_r c)$, with braid-minimized entries, to the set of unchecked tuples. This trick dramatically improves running time of the procedure.

6. C_σ-coset enumeration

One way to find a solution for the instance (3) is to enumerate elements in the coset $C_{\sigma_B} P_A$ until an element commuting with $b_1, \ldots, b_r$ is found. A straightforward approach to coset enumeration requires to find a generating set for the subgroup C_{σ_B} and enumerate its elements. That subgroup is finitely generated, its size is proportional to the size of the Schreier's graph of the action which is finite but, usually, huge. Hence, we developed a different way to solve (3) that attempts to directly identify and remove cloaking elements from P_A. Our algorithm is based on the following rather informally stated observations.

- As mentioned in Remark 2.3, the letters x_i in the word $wx_i^{\pm 2}w^{-1}$ from Proposition 2.1 twist two particular strands $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$.
- Replacement $wx_i^{\varepsilon}x_i^{\varepsilon}w^{-1} \to wx_i^{-\varepsilon}x_i^{\varepsilon}w^{-1}$, where $\varepsilon = \pm 1$, produces the trivial braid.
- Replacement of a single letter $x_i^{\pm 1}$ that twists strands $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$ with $x_i^{\mp 1}$ in a braid word $w_1 x_i^{\pm 1} w_2$ corresponds to multiplication of the word on the left by cloaking element $w_1 x_i^{\mp 2} w_1^{-1}$.
- Multiplying a braid word with cloaking elements on the left or on the right (or inserting a cloaking element into a random position) usually increases the length of the braid.
- Even though obfuscation of a cloaked braid word changes the way the word looks, it preserves the isotopy type of the braid and the result of obfuscation typically twists strands $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$ at the crossing corresponding to the middle of $wx_i^{\pm 2}w^{-1}$.
- By tracing strands in a given braid, we can algorithmically find all letters that twist strands $\sigma^{-1}(a)$ and $\sigma^{-1}(b)$. We call those letters *critical letters* for the corresponding strands.

Recall that we switch from the instance (4) to an instance (6), so we need to enumerate the coset $C_{\sigma_B \sigma_c} c^{-1} P_A c$. Instead of total coset enumeration, our algorithm attempts to decrease the length of the element $c^{-1}P_A c$ by flipping powers of the critical letters and applying braid-minimization to the result expecting the length to decrease.

In more detail, the algorithm iteratively constructs a subset of $C_{\sigma_B \sigma_c} c^{-1} P_A c$ starting from the set $\{c^{-1}P_A c\}$. On each iteration it picks a shortest unprocessed word w and performs the following manipulations.

- For each critical letter $x_i^{\pm 1}$ in w that twists strands $(\sigma_B \sigma_c)^{-1}(a)$ and $(\sigma_B \sigma_c)^{-1}(b)$
  - compute a new word by replacing $x_i^{\pm 1}$ with its inverse $x_i^{\mp 1}$;
  - shorten the obtained braid word using geodesic-braid minimization algorithm;
  - add the result to the current set.

We say that the algorithm is *successful* if it finds a word that commutes with $\delta^{-1}\beta_1\delta, \ldots, \delta^{-1}\beta_r\delta$. We admit *failure* if the algorithm is unable to find such a word, and there is no length decrease on the last 100 iterations.

In case of a failure, we randomly *reset* the instance. To do that, we choose a shortest word in the set of checked words, cloak it by 3 cloaking elements on the left and on the right respectively, apply the normal form and the braid minimization procedure, and start $C_{\sigma_B \sigma_c} c^{-1} P_A c$ enumeration from the resulting braid word. Each instance gets at most 3 attempts.

7. Tested parameter values and the results

The paper [6] does not describe the precise procedure to generate cloaking elements from Proposition 2.1, but such a description can be found in [4], see also [19, Section 2]. In particular, for security reasons, the conjugator w in a cloaking element is augmented with L random pure braid generators. Since values of L for 128- and 256-bit security levels are not mentioned in [6], we choose the corresponding values from [4].

For 128-bit security level we use the following parameters:

- $n = 16$.
- $q = 32$.
- $r = 32$.
- $L = 15$.
- $|B| = 22$ (in terms of generators $\mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1})$).
- $|z| \in [180, 250]$, $|\alpha| \in [300, 400]$, $|\beta_i| \in [50, 100]$.

For 256-bit security level the parameters are:

- $n = 16$.
- $q = 256$.
- $r = 32$.
- $L = 30$.
- $|B| = 43$.
- $|z| \in [300, 400]$, $|\alpha| \in [300, 400]$, $|\beta_i| \in [100, 200]$.

As mentioned in Introduction, after a series of attacks on WalnutDSA the authors proposed several changes to the protocol including changes to the cloaking procedure (see [1, D. Atkins on May 23, 2018]). It was suggested to use several cloaking element inserted into the private keys at random positions. We implemented and tested this idea as well. For cloaking elements we use conjugators w of lengths in the range $[30, 50]$ and insert 30 (for 128-bit level) and 60 (for 256-bit level) such cloaking elements into random positions inside private keys. These insertions are made iteratively, so randomly chosen positions may also be inside previously inserted cloaking elements.

Overall, we tested four versions of the Kayawood protocol, two original versions for 128- and 256-bit security levels and two versions using multiple cloaking elements to mask private keys (with the other parameters corresponding to 128- and 256-bit levels). We used Garside normal form followed by the braid minimization reduction to obfuscate $z\beta_i z^{-1}$ and stochastic rewriting to obfuscate public keys. For each version of the protocol we performed 100 experiments consisting of the following steps:

1. Generate a random protocol instance.
2. Generate random Alice's private data.
3. Run key establishement protocol.

(4) Run heuristic search for z as described in Section 5.2.
(5) Run coset enumeration to find a substitute for Alice's private key as described in Section 6.

Our algorithm solved all randomly generated instances, i.e., our attack had 100% success rate. Moreover, in most cases we found the original private key. Also, we investigated the behavior of heuristic search for z and, for the recovered conjugator c, collected the lengths $|z^{-1}c|$. For words $c^{-1}z\beta_1 z^{-1}c, \ldots, c^{-1}z\beta_r z^{-1}c$ we checked whether they are actually written in the generators of U_n, and for $c^{-1}z\alpha z^{-1}c$ we checked whether it is written in the generators of L_n.

All experiments were performed on a machine with two 8-core 3.1 GHz Intel Xeon CPU E5-2687W and 64GB RAM. The results are provided in Tables 1 and 2.

	128-bit	256-bit
Average running time	12 s	40 s
Average $\|z^{-1}c\|$	25	34
Original private key recovered	83%	75%
All $c^{-1}z\beta_i z^{-1}c$ are in generators of U_n	95%	91%
$c^{-1}z\alpha z^{-1}c$ is in generators of L_n	71%	81%

Table 1. Results for the original versions of the protocol

	128-bit	256-bit
Average running time	14 s	56 s
Average $\|z^{-1}c\|$	25	34
Original private key recovered	75%	76%
All $c^{-1}z\beta_i z^{-1}c$ are in generators of U_n	95%	91%
$c^{-1}z\alpha z^{-1}c$ is in generators of L_n	71%	81%

Table 2. Results for versions using multiple cloaking elements

8. Conclusion

Kayawood protocol, described in [6], does not provide the claimed level of security. It suffers from poor choice of cloaking elements. By design, cloaking elements have very specific geometric type defined by a fixed pair of strands that can be algorithmically recognized and removed. Thus, the definition of cloaking elements seems to be the weak part of the protocol. We doubt that security can be improved simply by increasing parameter values. Nevertheless, we believe that stabilizers for E-multiplication have very rich and algebraically interesting structure and using better cloaking elements (not simply conjugates of squares) can make the protocol more secure.

Appendix A. Finding the shared key using public keys

Testing out our generating procedures, we discovered a very surprising property of random keys. In about 60% of the cases one of the following equalities was satisfied:

$$(I,1)\star P_A\star P_B=(I,1)\star A\star P_B, \tag{7}$$

$$(I,1)\star P_B\star P_A=(I,1)\star B\star P_A, \tag{8}$$

i.e., the shared key could be obtained using public keys. After a thorough check of our implementation we realized that our observation is not a result of an error, but a feature of the design of the protocol. We suspect that the authors are unaware of this problem, otherwise it would be mentioned in the description of Kayawood.

Proposition A.1. *Let $n\in\mathbb{N}$ be even. In the notation of Kayawood protocol:*

(1) If $\sigma_A(a)=a$ and $\sigma_A(b)=b$, then (8) holds.
(2) If $\sigma_B(a)=a$ and $\sigma_B(b)=b$, then (7) holds.
(3) If $\sigma_z^{-1}(a),\sigma_z^{-1}(b)>n/2$, then $\sigma_A(a)=a$ and $\sigma_A(b)=b$.
(4) If $\sigma_z^{-1}(a),\sigma_z^{-1}(b)\le n/2$, then $\sigma_B(a)=a$ and $\sigma_B(b)=b$.
(5) Assuming that $\sigma_z\in S_n$ has uniform distribution:

$$\Pr[(7)\ \textit{or}\ (8)]\ge\frac{n}{2(n-1)}>\frac{1}{2}.$$

Proof. Observe that (1) and (2) are particular cases of Corollary 2.2. Indeed, if $\sigma_A(a)=a$ and $\sigma_A(b)=b$, then $C_1=C_{\sigma_A}$, so $P_B=v_1Bv_2B^{-1}B$, where $v_1,Bv_2B^{-1}\in C_1$. Similarly, (2) holds. Item (3) holds since $A=z\alpha z^{-1}$ and $\alpha\in L_n$, so σ_α acts trivially on $\frac{n}{2}+1,\dots,n$. Similarly, (4) holds since $\beta_1,\dots,\beta_r\in U_n$. Finally, notice that $\frac{n}{2(n-1)}$ is the chance that $a,b\in\{1,\dots,\frac{n}{2}\}$ or $a,b\in\{\frac{n}{2}+1,\dots,n\}$. Hence (5) holds. □

The lower bound in item (5) of Proposition A.1 is not very precise as it takes into account only two particular cases for (7) or (8) to be true. Yet, for $n=16$, it estimates the chance of (7) or (8) as 53.3% which is relatively close to our observations.

We note that we did not filter cases (1) and (2) of Proposition A.1 when generating random protocol instances, since they do not affect the behavior of our attack in any way.

References

[1] NIST PQC forum. Available at `https://groups.google.com/a/list.nist.gov/forum/#!forum/pqc-forum`, accessed: June 10, 2018.

[2] I. Anshel, M. Anshel, and D. Goldfeld. An algebraic method for public-key cryptography. *Math. Res. Lett.*, 6(3-4):287–291, 1999.

[3] I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux. Key agreement, the algebraic eraser^{TM}, and lightweight cryptography. In *Algebraic Methods in Cryptography*, volume 418 of *Contemporary Mathematics*, pages 1–34. American Mathematical Society, 2006.

[4] I. Anshel, D. Atkins, and P. Goldfeld, D. Gunnels. The Walnut digital signature algorithm(TM) specification. Submitted to NIST PQC project (2017). Available at https://csrc.nist.gov/projects/post-quantum-cryptography/round-1-submissions, accessed: June 10, 2018.

[5] I. Anshel, D. Atkins, and P. Goldfeld, D. Gunnels. Ironwood Meta Key Agreement and Authentication Protocol. Preprint. Available at https://arxiv.org/abs/1702.02450, 2017.

[6] I. Anshel, D. Atkins, and P. Goldfeld, D. Gunnels. Kayawood, a Key Agreement Protocol. Preprint. Available at https://eprint.iacr.org/2017/1162, 2017.

[7] I. Anshel, D. Atkins, and P. Goldfeld, D. Gunnels. WalnutDSA(TM): A Quantum-Resistant Digital Signature Algorithm. Preprint. Available at https://eprint.iacr.org/2017/058, 2017.

[8] W. Beullens and S. Blackburn. Practical attacks against the Walnut digital signature scheme. Preprint. Available at https://eprint.iacr.org/2018/318/20180404:153741, 2018.

[9] J. S. Birman, K. H. Ko, and S. J. Lee. A new approach to the word and conjugacy problems in the braid groups. *Adv. Math.*, 139:322–353, 1998.

[10] J. H. Cheon and B. Jun. A polynomial time algorithm for the braid diffie-hellman conjugacy problem. In *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes Comp. Sc.*, pages 212–225, Berlin, 2003. Springer.

[11] CRyptography And Groups (CRAG) C++ Library. Available at https://github.com/stevens-crag/crag.

[12] P. Dehornoy. A fast method for comparing braids. *Adv. Math.*, 125:200–235, 1997.

[13] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston. *Word processing in groups*. Jones and Bartlett Publishers, 1992.

[14] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne. Length-based conjugacy search in the braid group. In *Algebraic Methods in Cryptography*, volume 418 of *Contemp. Math.*, pages 75–88. Amer. Math. Soc., 2006.

[15] D. Hart, K. DoHoon, G. Micheli, G. Perez, C. Petit, and Y. Quek. A Practical Cryptanalysis of WalnutDSA. Preprint. Available at https://eprint.iacr.org/2017/1160 (version: 30-Nov-2017), 2017.

[16] D. Hofheinz and R. Steinwandt. A practical attack on some braid group based cryptographic primitives. In *Advances in Cryptology – PKC 2003*, volume 2567 of *Lecture Notes Comp. Sc.*, pages 187–198, Berlin, 2003. Springer.

[17] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, and C. Park. New public-key cryptosystem using braid groups. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes Comp. Sc.*, pages 166–183, Berlin, 2000. Springer.

[18] M. Kotov, A. Menshov, A. Myasnikov, Panteleev. D., and A. Ushakov. Conjugacy separation problem in braids: an attack on the original Colored Burau key agreement protocol. Available at https://eprint.iacr.org/2018/491.

[19] M. Kotov, A. Menshov, and A. Ushakov. An attack on the Walnut digital signature algorithm. Available at https://eprint.iacr.org/2018/393.

[20] A. G. Miasnikov, V. Shpilrain, and A. Ushakov. A practical attack on some braid group based cryptographic protocols. In *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes Comp. Sc.*, pages 86–96, Berlin, 2005. Springer.

[21] A. G. Miasnikov, V. Shpilrain, and A. Ushakov. Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol. In *Advances in Cryptology – PKC 2006*, volume 3958 of *Lecture Notes Comp. Sc.*, pages 302–314, Berlin, 2006. Springer.

[22] A. G. Miasnikov, V. Shpilrain, and A. Ushakov. *Non-Commutative Cryptography and Complexity of Group-Theoretic Problems*. Mathematical Surveys and Monographs. AMS, 2011.

[23] A. D. Myasnikov and A. Ushakov. Length based attack and braid groups: Cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol. In *Advances in Cryptology – PKC 2007*, volume 4450 of *Lecture Notes Comp. Sc.*, pages 76–88. Springer, 2007.

[24] A. D. Myasnikov and A. Ushakov. Cryptanalysis of Anshel-Anshel-Goldfeld-Lemieux key agreement protocol. *Groups Complex. Cryptol.*, 1:263–275, 2009.

[25] M. Paterson and A. Razborov. The set of minimal braids is co-NP-complete. *J. Algorithms*, 12:393–408, 1991.

[26] B. Tsaban. Polynomial-Time Solutions of Computational Problems in Noncommutative-Algebraic Cryptography. *J. Cryptology*, 28:601–622, 2012.

Department of Mathematics, Stevens Institute of Technology, Hoboken, NJ, USA

E-mail address: `mkotov,manton,aushakov@stevens.edu`