Kayawood, a Key Agreement Protocol

Iris Anshel, Derek Atkins, Dorian Goldfeld, and Paul E. Gunnells

SecureRF Corporation
100 Beard Sawmill Rd #350, Shelton, CT 06484
ianshel@securerf.com, datkins@securerf.com, dgoldfeld@securerf.com, pgunnells@securerf.com

Abstract. Public-key solutions based on number theory, including RSA, ECC, and Diffie-Hellman, are subject to various quantum attacks, which makes such solutions less attractive long term. Certain group theoretic constructs, however, show promise in providing quantum-resistant cryptographic primitives because of the infinite, non-cyclic, non-abelian nature of the underlying mathematics. This paper introduces Kayawood Key Agreement protocol (Kayawood, or Kayawood KAP), a new group-theoretic key agreement protocol, that leverages the known NP-Hard shortest word problem (among others) to provide an Elgamal-style, Diffie-Hellman-like method. This paper also (i) discusses the implementation of and behavioral aspects of Kayawood, (ii) introduces new methods to obfuscate braids using Stochastic Rewriting, and (iii) analyzes and demonstrates Kayawood's security and resistance to known quantum attacks.

Keywords: Group Theoretic Cryptography, Diffie–Hellman, Key Agreement, E-Multiplication, Braids

1 Introduction

Methods to apply group theory to cryptography have been studied for decades. In the last two decades, a number of group theoretic key agreement protocols were introduced, including [1] and [28]. Attacks on the conjugacy search problem such as those appearing in [17], [19], [25] suggested that these types of schemes may not be practical in low-resource environments. More recently, there is a renewed interest in Group Theoretic Cryptography (GTC) as reflected in two recent monographs [24], [33]. Instead of focusing on the conjugacy search problem, other problems in the Braid group have been explored for purposes of creating a quantum-resistant key agreement protocol.

Previous Work

The Artin Braid group B_N, which has generators

$$b_1, b_2, \ldots, b_{N-1},$$

and defining relations

$$\begin{aligned} b_i\, b_{i+1}\, b_i &= b_{i+1}\, b_i\, b_{i+1} \qquad (1 \le i \le n-2), \\ b_i\, b_j &= b_j\, b_i \qquad (|i-j| \ge 2), \end{aligned}$$

serves as the underlying structure for the quantum-resistant one-way function, E-Multiplication. A range of applications of E-Multiplication have been introduced: a cryptographic hash function [3], [4], the Walnut digital signature algorithm [5], and a Meta Key Agreement protocol [6]. Methods based on E-Multiplication are not subject to Shor's quantum algorithm [35] because

the underlying group is infinite, non-cyclic, and non-abelian. Moreover, methods based on E-Multiplication also scale linearly and, therefore, scale better against Grover's quantum search algorithm [21] versus methods that scale quadratically (or worse).

In 2006, [2] introduced a key agreement protocol based on GTC (specifically the Braid group) that withstood several attacks over the past decade. First, [32] determined that if braids are too short then it is possible to find the conjugating factor and use that to break the system. It was pointed out, however, in [23] that, in practice, the braids are long enough to thwart this attack. Such an attack can never actually succeed. It is akin to using Fermat to factor short RSA keys, which becomes impractical at "secure" sizes. Second, [26] showed a linear algebra attack (KTT) that would allow an attacker to determine part of the private key data. Similarly, in practice, [22] showed that this is just a class of weak keys, that would not be used. That is, by choosing the private key data in a specific way, this attack is defeated.

More recently, [11] built upon the defeated KTT attack; [11] reconstructed the shared secret by using all of the public information and leveraging a large precomputation before spending several hours to reconstruct it. In other words, this attack not only required access to the public parameters but also both public keys (including their permutations). It was shown in [7] that the attack work grows as the size of the permutation order grows as well as the size of the Braid group.

Still, none of these attacks targeted the underlying hard problems in the Braid group, or attempted to attack the one-way E-Multiplication function introduced in [2].

A meta key agreement protocol was designed to defeat all these attacks; this protocol, however, requires secure provisioning of both sides [6]. A true public key method based on hard problems in the Braid group obviates this need.

Our Contribution

This paper introduces an asymmetric key agreement protocol, Kayawood, which is structured as an Elgamal-type. The security of Kayawood relies on the difficulty in solving a specific set of equations over B_N. If the shortest representation of a braid word could be found, it would make solving equations in B_N tenable. However, the shortest word problem is known to be NP-hard [34]. Inherently, Kayawood possesses some features that suit cryptographic applications and environments where not all devices can be preloaded with data in advance of being utilized in the field. In addition, Kayawood possesses the features that other E-Multiplication-based protocols do: quantum resistance; rapidly computable output; and the security level of the system scales linearly with the size of the user ephemeral private keys.

The paper proceeds as follows. First, it reviews E-Multiplication and Cloaking Elements. Next, it introduces the Kayawood protocol. Following the protocol is a security discussion and then a security reduction. Next, the paper introduces Stochastic Rewriting and then a new braid collecting method. Finally, the papers finishes with some notes on implementation experience and conclusions.

2 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 $(M, \sigma) \in GL_N(\mathbb{F}_q) \times S_N$:

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

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 2.1 (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 $(m, \sigma) \star v = (m, \sigma)$.*

Thus a cloaking element will essentially disappear when E-Multiplication is evaluated. It is not immediately obvious how to construct cloaking elements. The following proposition addresses this issue when certain assumptions about t-values are made.

Proposition 2.2 *Fix integers $N \geq 2$, and $1 \leq a < b \leq N$, and assume that the t-values τ_a, τ_b are both equal to 1. Let $m \in GL(N, \mathbb{F}_q)$and let $\sigma \in S_N$.Then a cloaking element v of (m, σ) is given by $v = wb_i^2w^{-1}$ where b_i is any Artin generator $(1 \leq i < N)$ and $w \in B_N$ has associated permutation which move $i \to \sigma^{-1}(a),\ i+1 \to \sigma^{-1}(b)$.*

Since stabilizing elements of a group action form a subgroup, the following proposition is immediate:

Proposition 2.3 *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.

3 The Kayawood Protocol

The Kayawood protocol allows a user, Alice, to generate sets of data for both herself and for a second user, Bob. Alice will use her data to choose an ephemeral private key, and Bob, upon receiving his data from Alice, will likewise generate an ephemeral private key. A novel feature of the Kayawood protocol is that each user's ephemeral public key is based not only on their respective private keys, but also on some received public data from the other user.

The public Information below may be generated by Alice, or by another entity and relayed to Alice.

Public Information:

- The Braid group B_N, where $N \geq 16$ is an even integer.
- One or more rewriting algorithms, $\mathcal{R}\colon B_N \to B_N$, such as [13] or [15], or the Stochastic rewriting method of Section 7.
- A finite field $\mathbb{F}_q$ of $q \geq 32$ elements.
- Two integers $1 \leq a < b \leq N$.
- T-values $= \{\tau_1, \tau_2, \ldots, \tau_N\}$, where each τ_i is an invertible element in $\mathbb{F}_q$, and $\tau_a = \tau_b = 1$.

With these choices in place, Alice generates the following private data.

Private Data Generated by Alice:

- Braid elements $\beta_1, \ldots, \beta_r$ from the subgroup $\langle b_{\frac{N}{2}+1}, \ldots, b_{N-1} \rangle \leq B_N$ whose associated permutations have high order. Such elements can be obtained via a search algorithm.
- A braid element $z \in B_N$ whose associated permutation has the following mixing property: if $\Lambda_A = \{1, 2, \ldots, \frac{N}{2}\}$ and $\Lambda_B = \{\frac{N}{2}+1, \ldots, N\}$, then half of Λ_A moves to Λ_B and half of Λ_B moves to Λ_A. The element z may be obtained constructively or via a search algorithm.
- Alice's Private key, $\text{Priv(A)} = z\alpha z^{-1}$, where α is an element in the subgroup of B_N generated $b_1, \ldots, b_{\frac{N}{2}-1}$.

With the above data in place, we can proceed as follows.

3.1 The Protocol

1. Alice sends Bob the rewritten conjugates $= \{\mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1})\}$, together with permutation σ_A associated to her private key Priv(A) .

2. Bob chooses for his private key Priv(B) a word in the conjugates $\left\langle \mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1}) \right\rangle$ and their inverses:

$$\text{Priv}(B) \;=\; R(z\beta_{i_1} z^{-1})^{\epsilon_{i_1}}\, R(z\beta_{i_2} z^{-1})^{\epsilon_{i_2}} \cdots R(z\beta_{i_\ell} z^{-1})^{\epsilon_{i_\ell}}.$$

   Letting σ_B denote the permutation associated to Priv(B), Bob constructs two cloaking elements v_{B_1}, v_{B_2} which cloak for the permutations σ_A and $\sigma_A \cdot \sigma_B$, respectively, and evaluates his public key:

$$\text{Pub(B)} \;=\; \mathcal{R}\left(v_{B_1} \cdot \text{Priv(B)} \cdot v_{B_2}\right) \in B_N. \tag{1}$$

   Bob sends his public key to Alice.

3. Alice evaluates the permutation associated to Pub(B), σ_B, and constructs two cloaking elements v_{A_1}, v_{A_2} which cloak for the permutations σ_B and $\sigma_B \cdot \sigma_A$, respectively. She then evaluates her public key:

$$\text{Pub(A)} \;=\; \mathcal{R}\left(v_{A_1} \cdot \text{Priv(A)} \cdot v_{A_2}\right) \in B_N. \tag{2}$$

   Alice sends her public key to Bob.

4. Alice evaluates

$$(\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Pub(B)} = (\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Priv(B)},$$

and, likewise, Bob evaluates

$$(\mathrm{Id}, 1) \star \mathrm{Priv(B)} \star \mathrm{Pub(A)} = (\mathrm{Id}, 1) \star \mathrm{Priv(B)} \star \mathrm{Priv(A)}.$$

These equations hold because Pub(A) evaluates the same as Priv(A) under E-Multiplication due to the way cloaking elements are built, and the outputs of these evaluations are the same because, by construction, Priv(A) and Priv(B) commute in the Braid group.

3.2 Protocol Analysis

An astute reader will notice that the Kayawood protocol requires one extra message beyond a pure Diffie-Hellman / Elgamal protocol. Specifically, Alice cannot generate her public key until Bob has sent enough information for Alice to obtain his key permutation. This has several implications:

- Both Alice's and Bob's public keys are dependent on both Alice's and Bob's private data (specifically, the permutation thereof)
- Alice's public key will necessarily be different when communicating with different "Bob" entities (or the same "Bob" entity with different ephemeral private keys)
- Alice's public key cannot be a priori generated and put into a certificate, however
- Alice's permutation, conjugate material, and even t-values can be signed by a CA; only the real Alice would know the z and be able to present a $z\alpha z^{-1}$ that would create a valid shared secret with the conjugates

4 Security Discussion

The security of Kayawood rests upon the following assertions:

(i) The search space for each users private key is sufficiently large;

(ii) The simultaneous conjugacy search problem cannot yield a solution;

(iii) The methods of Ben-Zvi–Blackburn–Tsaban [11] do not apply;

(iv) Users private keys are obscured by the cloaking elements and the rewriting operation.

Assessing each of the points above in turn, we begin with the search space for Alice's private key $z\alpha z^{-1}$, which is tantamount to estimating how many braids α Alice could choose from. Assuming α is a product of L_A Artin generators $b_1, \ldots, b_{\frac{N}{2}-1}$, it is shown in [5] that the security level is bounded below by

$$\log_2 \left(\frac{2^{L_A}}{L_A} \cdot \left(\frac{N}{2} - 2 \right) \binom{L_A - 3 + \frac{N}{2}}{\frac{N}{2} - 2} \right).$$

The search space for Bob's private key is somewhat simpler to analyze and again amounts to estimating the number of private keys Bob could create. Assuming Bob's private key is a word of length L_B in the subgroup generated by the conjugates,

$$\langle \mathcal{R}(z\beta_1 z^{-1}), \ldots, \mathcal{R}(z\beta_r z^{-1}) \rangle,$$

the security level will be

$$\log_2 \left((2r)^{L_B} \right) = L_B \left(1 + \log_2(r) \right),$$

because nontrivial relations for the subgroup will generally not be short.

Next, the simultaneous conjugacy search problem (SCSP) is the following generalization of the conjugacy problem. Let $z \in B_N$, fix an integer ℓ, ℓ', and let $\{\alpha_1, \ldots, \alpha_{\ell'}\} \subset \langle b_1, \ldots, b_{\frac{N}{2}-1} \rangle$ and $\{\beta_1, \ldots, \beta_\ell\} \subset \langle b_{\frac{N}{2}+1}, \ldots, b_{N-1} \rangle$. Then SCSP is the following: *Given the lists*

$$\{\mathcal{R}(z\alpha_i z^{-1})\}, \{\mathcal{R}(z\beta_i z^{-1})\}$$

of rewritten/disguised conjugates, find z and the original words $\{\alpha_i\}, \{\beta_i\}$. An attack on the SCSP was described by Myasnikov–Ushakov [32] and was analyzed in [23], where it was shown to be ineffective once the conjugating element z is chosen to be even moderately long. However, regardless of the efficacy of this attack, it cannot be used to attack Kayawood, since the attack needs both sets of conjugates to proceed. In Kayawood, only the set $\{\mathcal{R}(z\beta_i z^{-1})\}$ is known; Alice's private key $z\alpha z^{-1}$ is not made public.

In contrast to Myasnikov–Ushakov, the Ben-Zvi–Blackburn–Tsaban approach makes no attempt to solve SCSP. Instead it proceeds by using the public data exchanged by Alice and Bob and one set of conjugates from the SCSP to build surrogate private keys that can be used to recover the shared secret. The presence of the nontrivial cloaking elements in Kayawood, which can involve any generator of B_N, prevents this linear algebraic method from succeeding.

The process of obscuring of the private keys by surrounding them by cloaking elements is a novel feature of the Kayawood protocol. The Walnut digital signature algorithm uses cloaking elements to obscure an encoded message which has been conjugated by the users private key. In Kayawood, the users private and public keys are ephemeral and different cloaking elements are generated each time. An observer cannot, a priori, tease the ephemeral private key out of the public key, because a rewriting method has been applied to the ephemeral private key. Further, the shared secret is obtained by the E-Multiplications

$$(\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Pub(B)},$$

respectively

$$(\mathrm{Id}, 1) \star \mathrm{Priv(B)} \star \mathrm{Pub(A)}.$$

The set of all possible matrices which appear in the first component of the E-Multiplications

$$(\mathrm{Id}, 1) \star \mathrm{Priv(A)}$$

and

$$(\mathrm{Id}, 1) \star \mathrm{Priv(B)}$$

has order

$$q^{N(N-1)},$$

and, thus, any attempt at a brute force attack[1] on the possible users matrix will be ineffective.

In the Kayawood protocol, increasing the security level does require a proportional increase in key size and linear increase in computation. This stands in contrast to the situation that arises in other protocols that are standard in the art, where the increase is quadratic. Given the emergence of quantum attacks on cryptographic protocols, this is a significant and noteworthy feature.

5 A Security Reduction for the Kayawood Protocol

Following the model presented in Kudla and Paterson [29], and earlier work of Bellare, Pointcheval, and Rogaway [9], we posit the following algorithmically hard problem. Assume we are given a braid word β which is known to the output of a rewriting operation $\mathcal{R}$,

$$\beta \;=\; \mathcal{R}\left(v_1 \cdot \beta_0 \cdot v_2\right) \in B_N,$$

where v_1, v_2 are cloaking elements for known permutations, and the braid β_0 may be in an unknown subgroup of B_N. The *cloaking problem*, which is the underlying hard problem on which the Kayawood protocol is based, asks for the ordered pair

$$(\mathrm{Id}, 1) \star \beta_0.$$

To date, there is no known approach to solving the cloaking problem: even a brute-force enumeration of all possible braids b_0 and cloaking elements v_1, v_2 will at best result in a collection of possible $(\mathrm{Id}, 1) \star \beta_0$. There is no a priori way to decide which β_0 is correct.

In order to demonstrate that the Kayawood protocol is as robust as the cloaking problem, let us assume that there is a random oracle $\mathcal{O}$ that outputs the shared secret of the Kayawood protocol, i.e., given the inputs

$$\mathrm{Pub(A)} = \mathcal{R}\left(v_{A_1} \cdot \mathrm{Priv(A)} \cdot v_{A_2}\right),$$

and

$$\mathrm{Pub(B)} = \mathcal{R}\left(v_{B_1} \cdot \mathrm{Priv(B)} \cdot v_{B_2}\right),$$

$\mathcal{O}$ produces the shared secret

$$(M, \sigma) \;=\; (\mathrm{Id}, 1) \star \mathrm{Priv(B)} \star \mathrm{Priv(A)} = (\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Priv(B)}.$$

Consider the braid Pub(B). By definition, we know that

$$\mathrm{Pub(B)} = \mathcal{R}\left(v_{B_1} \cdot \mathrm{Priv(B)} \cdot v_{B_2}\right) = v_{B_1} \cdot \mathrm{Priv(B)} \cdot v_{B_2}.$$

Thus, taking the inverse of Pub(B), we have

$$\mathrm{Pub(B)}^{-1} = v_{B_2}^{-1} \cdot \mathrm{Priv(B)}^{-1} \cdot v_{B_1}^{-1}.$$

[1] We shall note later there is a more conservative estimate due to the fact that a t-value of 1 generally results in a row in the matrix that substantially duplicates the row before. Conservatively, there are only $q^{N(N-3)}$ matrices.

Recalling that v_{B_2} is a cloaking element for the permutation $\sigma_B \cdot \sigma_A$, we have that $v_{B_2}^{-1}$ is likewise a cloaking element for the permutation $\sigma_B \cdot \sigma_A$. Similarly, $v_{B_1}^{-1}$ is a cloaking element for σ_A. Observe further that the permutation in output of the Kayawood protocol (M, σ) is precisely $\sigma = \sigma_B \cdot \sigma_A$. By E-multiplying the output of the random oracle with the in $\mathrm{Pub(B)}^{-1}$ and using the properties of the above cloaking elements we obtain

$$\begin{aligned}(M,\sigma) \star \mathrm{Pub(B)}^{-1} &= (\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Priv(B)} \star \mathrm{Pub(B)}^{-1} \\ &= (\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Priv(B)} \star v_{B_2}^{-1} \cdot \mathrm{Priv(B)}^{-1} \cdot v_{B_1}^{-1} \\ &= (\mathrm{Id}, 1) \star \mathrm{Priv(A)} \star \mathrm{Priv(B)} \star \mathrm{Priv(B)}^{-1} \cdot v_{B_1}^{-1} \\ &= (\mathrm{Id}, 1) \star \mathrm{Priv(A)}.\end{aligned}$$

Thus, we obtain a solution to the cloaking problem and the argument is complete.

6 Resistance to Quantum Attacks

A cryptographic protocol is said to be *quantum resistant* if it remains secure even when an attacker has access to a quantum computer and can perform polynomial time quantum computations. In this section and the next, we present evidence that Kayawood resists two major quantum attack methods, Grover's quantum search algorithm and Shor's quantum factoring algorithm.

6.1 Resistance to Grover's Quantum Search Algorithm

Grover's quantum search algorithm [21] allows a quantum computer of sufficient size to search for a particular item in an unordered n-element search space in $\mathcal{O}(\sqrt{n})$ steps. By comparison, searching on a classical computer takes $\mathcal{O}(n)$ steps. It follows that if the security level of a cryptosystem is based on a brute force search of a keyspace, then Grover's quantum search algorithm will reduce a security level of 2^k to $2^{k/2}$. In order to yield a net desired security level of 2^k, if the crypto system running time is quadratic in the security level, then Grover's algorithm generally requires a fourfold increase in running time.

Given that Kayawood's runtime is linear in its security level, to double the security strength, one must double the complexity. Thus, in order to thwart an attacker's use of a quantum computer, the running time will only have to double to maintain the same security level because doubling the security level to counteract Grover's algorithm only requires double the runtime.

6.2 Resistance to Shor's Factoring Algorithm

Shor's quantum method [35] enables a sufficiently large quantum computer to break RSA, ECC, and Diffie-Hellman by factoring or solving the discrete log problem. Kitaev [27] noted that Shor's method can be vastly generalized to solve the *Hidden Subgroup Problem* (HSP):

Hidden Subgroup Problem: Let G be a finite group and let H be a subgroup of G. For a finite set X, let $f\colon G \to X$ be a function that is constant and distinct on left cosets of H. In other words, $f(gh) = f(gh')$ for any fixed $g \in G$ and all $h, h' \in H$ and $f(gh) \neq f(g'h)$ for

$g \neq g'$ $(g, g' \in G)$ and any $h \in H$. We say f *hides* the subgroup $H \subset G$. The *Hidden Subgroup Problem* is then the following: *given an f as above, find the subgroup H that it hides.*

Shor's algorithm [35] reduces the hard problem of factoring to finding the order of an element in a cyclic group (discrete logarithm problem). Shor then solves the order-finding problem with a quantum period-finding subroutine that reduces to the HSP for cyclic groups. It is known that the HSP can be solved on a quantum computer when the hidden subgroup H is abelian [30].

By contrast, at present there is no known polynomial-time solution to the HSP when the hidden subgroup is non-abelian, although some special cases have been solved [8,12,16,18,20].

The Kayawood protocol takes place on the Braid group B_N, which is an infinite, non-cyclic, non-abelian group. The security of Kayawood is based on the hardness of reversing E-Multiplication and CCSP. There seems to be no way to connect these hard problems with HSP.

The core operation in Kayawood is E-Multiplication, as described in Section 2. Given an element

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

where $i_j \in \{1, \ldots, N-1\}$, and $\epsilon_j \in \{\pm 1\}$, we can define a function $f : B_N \to GL(N, F_q)$ where $f(\beta)$ is given by the E-Multiplication $(1, 1) * (\beta, \sigma_\beta)$ and σ_β is the permutation associated to β. Now, E-Multiplication is a highly nonlinear operation. As the length k of the word β increases, the complexity of the Laurent polynomials occurring in the E-Multiplication defining $f(\beta)$ increases exponentially. It does not seem to be possible that the function f exhibits any type of simple periodicity, so it is very unlikely that inverting f can be achieved with a polynomial quantum algorithm. Note that this does reduce the Kayawood into a finite, yet still highly non-cyclic, non-abelian group; however, this does not affect the analysis.

The class NP (nondeterministic polynomial time) is the set of all decision problems with the property that if someone reveals the answer to the decision problem, then it is possible to vary the answer in polynomial time. In [10] it is shown that relative to an oracle chosen uniformly at random with probability 1, the class NP cannot be solved on a quantum Turing machine in time $o(2^{n/2})$. In this sense, the quantum search algorithms, which are purely based on the blackbox property of the formulae, cannot solve the search problem in time $o(2^{n/2})$. This implies that the search problem remains resistant in the quantum computational model.

Given a group presentation G and a word w in G, consider the class of all words in G that are equivalent to w, under the defining relations of G. Can we find the element in this class of words which has the shortest word length in the generators and their inverses? This is called the *shortest word problem* in the group G. Under the assumption that the shortest word problem in the Braid group can be efficiently solved, [32] developed an heuristic length attack on AEDH which can be refuted for large key lengths [23]. In 1991, Paterson–Razborov [34] showed that the shortest word problem in the Braid group on infinitely many strands is at least as hard as an NP-complete problem. This suggests that the shortest word problem in the Braid group on infinitely many strands may be quantum resistant. Tatsuoka [37] has shown that the shortest word problem in the Braid group on a fixed finite number of strands can be solved in quadratic time. But the constants in the estimate of the running time must be growing exponentially large as the number of strands increase for the Paterson–Razborov result to hold. This lends further credibility for the quantum resistance of Kayawood.

7 A New Braid Obfuscation Method: Stochastic Rewriting

There are three standard methods that can be used to rewrite braids: the Artin generator based Garside normal form (see [14]); the band generator Birman, Ko, Lee normal form (see [13]); and the Artin generator based Dehornoy handle reduction method (see [15]). While combining handle reduction with a normal form provides effective means of braid obfuscation, in certain applications the running time of these options becomes too great. Here we introduce a new set of generators for B_N and a randomized rewriting method that takes advantage of the shorter relations between the new generators.

Let $\mathcal{P} = \{p_1, p_2, \ldots, p_\ell\}$ denote a partition of $N-1$, where $3 \leq p_i$,

$$N - 1 \;=\; p_1 + p_2 + \cdots + p_\ell.$$

Defining the sequence of values

$$\begin{aligned}
r_1 &= 1 \\
r_2 &= r_1 + p_1 = 1 + p_1 \\
&\vdots \\
r_i &= r_i + p_i = 1 + p_1 + p_2 + \cdots + p_{i-1} \\
&\vdots \\
r_{\ell+1} &= r_\ell + p_\ell = 1 + p_1 + p_2 + \cdots + p_\ell = N,
\end{aligned}$$

we then form the new set of generators, the y-generators $yGen(\mathcal{P})$, for the Braid group as follows:

$$\begin{aligned}
&y_1 = b_1\, b_2 \cdots b_{r_2-1} \\
&y_2 = b_2 \cdots b_{r_2-1} \\
&\quad\vdots \\
&y_{r_2-1} = b_{r_2-1} \\
&y_{r_2} = b_{r_2}\, b_{r_2+1} \cdots b_{r_3-1} \\
&y_{r_2+1} = b_{r_2+1} \cdots b_{r_3-1} \\
&\quad\vdots \\
&y_{r_3-1} = b_{r_3-1} \\
&y_{r_3} = b_{r_3}\, b_{r_3+1} \cdots b_{r_4-1} \\
&\quad\vdots \\
&y_{r_{\ell+1}-1} = y_{N-1} = b_{r_{\ell+1}-1}.
\end{aligned}$$

Observe that the Artin generators can be expressed as words in the y-generators $yGen(\mathcal{P})$ as follows:

$$\begin{aligned} b_1 &= y_1\, y_2^{-1} \\ b_2 &= y_2\, y_3^{-1} \\ &\vdots \\ b_{r_2-1} &= y_{r_2-1} \\ b_{r_2} &= y_{r_2}\, y_{r_2+1}^{-1} \\ &\vdots \\ y_{N-1} &= y_{r_{\ell+1}-1}. \end{aligned}$$

Hence the braid relations, expressed as words in the Artin generators, can likewise be expressed as words in $yGen(\mathcal{P})$: for example, the relation

$$b_1\, b_2\, b_1 = b_2\, b_1\, b_2$$

becomes

$$\begin{aligned} y_1\, y_2^{-1}\, y_2\, y_3^{-1}\; y_1\, y_2^{-1} \;&=\; y_2\, y_3^{-1}\, y_1\, y_2^{-1}\, y_2\, y_3^{-1}, \\ \Longrightarrow\; y_1\, y_3^{-1}\, y_1\, y_2^{-1} &= y_2\, y_3^{-1}\, y_1\, y_3^{-1}. \end{aligned}$$

The relation $b_1\, b_3 = b_3\, b_1$ is handled similarly. The collection of Artin relations, expressed as words in the y-generators, is termed the y-Relations, and is denoted by $yRel(\mathcal{P})$. That any cyclic permutation of a y-relation is itself a relation is clear, but there are other less obvious relations: for example, if $r_\mu \le i < j \le r_{\mu-1}$, then

$$y_j\, y_i \;=\; y_i\, y_{j-1}\, y_{r_\mu-1}^{-1}.$$

Amassing these additional relations, together with the y-Relations and cyclic permutations of all of the above, yields the full set of relations that will enable the Stochastic rewriting. This complete set is denoted $SRel(\mathcal{P})$.

Given a word w in the Artin generators, and a fixed partition $\mathcal{P}$ of $N-1$, the Stochastic rewriting will obfuscate a braid and proceeds as follows:

(i) Express w as a word in the y-generators, denoted $w_{yGen(\mathcal{P})}$.

(ii) Choose a specified interval $[a, b]$, and break $w_{yGen(\mathcal{P})}$ into a string of blocks $\text{Block}_1, \text{Block}_2, \ldots$ where each block has a length, $\text{length}(\text{Block}_j)$, in the specified interval. In most cases $5 \le \text{length}(\text{Block}_j) \le 10$.

(iii) Choose a subword, u, of length 2 in each block.

(iv) For each of the above subwords u, search for a relation r in the list $SRel(\mathcal{P})$ which contains the subword u: $r = L\, u\, R = 1$. If no such relation exists, the block remains untouched. If r can be found, replace the u within the block by $L^{-1}\, R^{-1}$.

(v) Concatenate the modified blocks and freely reduce this new word in the y-generators.

(vi) Repeat (ii)-(v) as needed. The final word in the y-generators should be expressed as a word in the Artin generators.

We tested Stochastic Rewriting to determine how many rounds were needed to successfully obfuscate a braid. Testing consisted of taking a braid and running it through several iterations of rewriting and using the Smith-Waterman algorithm [36] to find the maximal run-length of similarity between both outputs.

When we look at Stochastically Rewritten braids, we can clearly see the average of the maximum run length between two rewrites of the same signature goes down as one increases the number of iterations of the rewriting method. For instance, we take 100 braids of appoximate length of 1012 generators, and then take each input and run Stochastic Rewriting twice with k iterations to produce two rewritten outputs. Obviously, the length increases, but surprisingly, the largest jump in length is from $k = 0$ to $k = 1$. After that, the length seems to increase linearly. As k increases, the maximal run size decreases (See Table 1).

k	Avg Length	Avg Max Run
0	1012	
1	1785	28
2	2155	20
3	2454	18
4	2743	17
5	3010	17

Table 1. Comparing the average output length and average run length with number of rounds of Stochastic Rewriting

Testing shows that $k = 3$ represents a "sweet spot", at least as far as this quantity is concerned. Increasing k beyond this point does not need to reduce the average maximal length.

8 A New Braid Collecting Method

In the case of the trivial partition of $N-1$, $\mathcal{P} = \{0, N-1\}$, the y–generators take the form

$$\begin{aligned} y_1 &= b_1\, b_2 \cdots b_{N-1} \\ y_2 &= b_2 \cdots b_{N-1} \\ &\vdots \\ y_{N-1} &= b_{N-1}. \end{aligned}$$

Observe that the following relations hold: for $i = 2, \ldots, N-1$,

$$y_i\, y_1 \;=\; y_1\, y_{i-1}\, y_{N-1}^{-1}.$$

Some simple manipulations with the above yields a second set of relations:

$$y_i^{-1}\, y_1 \;=\; y_1\, y_{N-1}\, y_{i-1}^{-1},$$

and we see that given an arbitrary word in the y-generators, w, it is possible to express w in the form

$$w = y_1^{a_1}\, w_1,$$

where w_1 is a word in the generators $\{y_2, \ldots, y_{N-1}\}$. This method of moving all the occurrences of y_1 in w to the left is a collecting method akin to commutator collection methods (see [31]). This process can be continued because occurrences of y_i can be collected on the left once occurrences of y_{i-1} have been collected via the relations:

$$y_j\, y_i \;=\; y_i\, y_{j-1}\, y_{N-1}^{-1} \qquad \text{if } i < j,$$

and

$$y_j^{-1}\, y_i \;=\; \begin{cases} y_i\, y_{N-1}\, y_{j-1}^{-1}, & \text{if } j > i \\ y_{i-1}\, y_{N-1}^{-1}\, y_i^{-1}, & \text{if } j < i. \end{cases}$$

The above formulae can be extended to moving powers of y_i past powers of $y_j^{\pm 1}$. For example, when $i < j$, we have

$$y_j\; y_i^k = \begin{cases} y_i^k\; y_{j-k}\; y_{N-k}^{-1}, & k < j-i, \\ y_i^{k+1}\; y_{N-k}^{-1}, & k = j-i, \\ y_i^{k+1}\; y_{N-(k-j+i))}\; y_{N-k}^{-1}, & j-i < k < N-i, \\ y_i^{k+1}\; y_j\; y_i^{-1}, & k = N-i, \\ y_i^{N-i+1}\; y_j, & k = N-i+1. \end{cases}$$

The output of this collection method is a sequence of decompositions of the original word w:

$$\begin{aligned} w \;&= = y_1^{a_1}\, w_1 \\ w \;&=\; y_1^{a_1}\, y_2^{a_2}\, w_2 \\ &\;\;\vdots \\ w \;&=\; y_1^{a_1}\, y_2^{a_2}\, \cdots\, y_{N-1}^{a_{N-1}}\, w_{N-1} \end{aligned}$$

where w_i is an expression in the generators

$$\{y_1^{-1}, y_2^{-1}, \ldots, y_{N-1}^{-1}, y_{i+1}, \ldots, y_{N-1}\}.$$

Note that in the final output of the collection method, $y_1^{a_1}\, y_2^{a_2}\, \cdots\, y_{N-1}^{a_{N-1}}\, w_{N-1}$ the word w_{N-1} only involves inverses of y-generators, i.e.,

$$w \;=\; y_1^{a_1}\, y_2^{a_2}\, \cdots\, y_{N-1}^{a_{N-1}} \cdot \text{negative word}.$$

Thus, the output is an expression of significantly different form when compared to the original expression for w. While it is beyond the scope of this paper, this collecting method has potential to be another means of obscuring a braid.

9 Implementation Experience

We have implemented Kayawood in C and run it on several platforms. The summary of our experience is that generating Bob's conjugates takes the greatest amount of time. Generating

Bob's public key is the second-largest operation. Third largest is generating Alice's public key. Generating the shared secret, because it uses E-Multiplication, is extremely fast.

For testing purposes, we chose two configurations, $B_{16},\ F_{32},\ r = 32,\ L = 22$ for a 128-bit security level, and $B_{16},\ F_{256},\ r = 32,\ L = 43$ for a 256-bit security level. Then, we generated 250 keysets for Alice which include z, T-values, and Bob's conjugates. For each keyset, we then created 20 Alice keys and 20 Bob keys and ran the key exchange.

At the 128-bit security level, the lengths of z varied from 141 to 332 generators with an average length of 212.76 and standard deviation of 35.21. Using these private braids, the conjugates (after using BKL Normal Form to obfuscate them and Dehornoy Reduction to reduce them) ranged in length from 422 to 1,868 generators with an average length of 884.45 and a standard deviation of 196.7.

Alice's private key, generated directly from z, ranged in length from 328 to 752 with an average length of 489.15 and standard deviation of 70.78. Bob's private key, however, is generated from the conjugates and is expected to be much longer. Specifically, because it is generated by 22 words in the conjugates (and inverses), they ranged from 6,876 to 22,554 generators with an average length of 12,378.83 and a standard deviation of 2,554.93.

Alice and Bob's public keys are generated by creating cloaking elements around their private keys and then running the result through Stochastic Rewriting. This results in a public key for Alice ranging from 2,018 to 4,218 generators with an average length of 3,025.33 and a standard deviation of 288.57. Bob's public key ranged from 12,900 to 57,972 with an average of 30,141.29 and a standard deviation of 5,974.7. This is an average increase of 2.4x to 6.2x over the private key. Alice's public key increased more over the private key because the cloaking elements were a larger percentage of the raw public key before processing with Stochastic Rewriting.

For 256-bit security, the lengths of z varied from 241 to 496 generators with an average length of 359.05 and standard deviation of 48.11. The conjugates created by these braids (after BKL and Dehornoy) resulted in a length from 740 to 2,698 generators with an average length of 1,463.84 and a standard deviation of 269.

Using these parameters, Alice's private key ranged in length from 568 to 1,114 generators with an average length of 825.05 and a standard deviation of 97.33. Generating Bob's private key in 43 words in the conjugates results in a length ranging from 17,530 to 54,030 with an average length of 32,205.33 and a standard deviation of 5,208.67.

An astute reader will notice that Alice's private key is on average twice as long when doubling the security level, whereas Bob's private key is, on average, 2.6 times longer (just a bit over double in length). This implies, due to the linear nature of E-Multiplication, that the operation will take twice as long to process.

The private keys at 256-bit are also approximately double in size from their 128-bit siblings. After Stochastic Rewriting, Alice's public keys ranged from 3,086 to 5,694 generators with an average of 4,329.93 and standard deviation of 366.35. Bob's public keys ranged from 37,356 to 132,200 generators with an average length of 76,458.29 and a standard deviation of 11,919.39.

Note that it is possible to run Dehornoy Reduction on Bob's public key prior to Stochastic Rewriting. This will condense the private-key portion (removing the intermediary, adjacent z and z^{-1} terms) and shorten the result prior to Stochastic Rewriting. When this is applied the results change significantly.

Specifically, if we run Dehornoy first, then this reduces Bob's public key to the range of 6,500 to 12,856 generators, with an average of 9,737.46 and a standard deviation of 1,446.99. Running Stochastic rewriting on these reduced public keys results in an output ranging from 15,768 to 32,170 generators, with an average of 24,512.09 and a standard deviation of 3,621.32. This is a reduction of 3x from the original non-reduced keys.

It should be noted that Dehornoy could also be run after Stochastic Rewriting, but the resulting shortening of the public key may not offset the time required to run Dehornoy at that point in the process. Moreover, whether or not to run Dehornoy prior to Stochastic Rewriting is more affected by the length of Bob's raw private key. Specifically, as the security level is increased or r is decreased, increasing the resulting L, the length of the raw private key increases.

If we run Dehornoy on the SL256 public keys once more, this reduces them to a range of 6,478 to 12,988 generators, with an average of 9,753.63 and a standard deviation of 1,430.71. These lengths are similar to the lengths obtained by the first Dehornoy operation, however, these versions are better-mixed than the original.

Studying the exact points of when it is beneficial to apply Dehornoy is left as an exercise for the reader. For example, we did not test running Dehornoy after Stochastic Rewriting on the raw public key. Most likely the end length would again result in similar output, but this would be future work.

10 Conclusions

This paper introduced the Kayawood Key Agreement Protocol, a quantum-resistant, Diffie-Hellman-like, Elgamal-style method based on infinite group theory. It analyzed its security and provided a security reduction. The paper also introduced Stochastic Rewriting and a new braid collecting method as alternative methods to obscure braids. Due to the non-cyclic, non-abelian nature of the underlying group, Kayawood is not subject to Shor's quantum algorithm, and because the data grows linearly, defeating Grover's quantum search algorithm only requires doubling the computation time. Consequently, Kayawood is an interesting, quantum-resistant method.

References

1. I. Anshel; M. Anshel; D. Goldfeld, *An algebraic method for public-key cryptography,* Math. Res. Lett. 6 (1999), 287-291.
2. I. Anshel; M. Anshel; D. Goldfeld; S. Lemieux, *Key agreement, the Algebraic Eraser^{TM}, and Lightweight Cryptography,* Algebraic methods in cryptography, Contemp. Math., vol. 418, Amer. Math. Soc., Providence, RI, 2006, pp. 1–34.
3. I. Anshel; D. Atkins; D. Goldfeld; P. E. Gunnells, *A Class of Hash Functions Based on the Algebraic Eraser,* 2015.
4. I. Anshel; D. Atkins; D. Goldfeld; P. E. Gunnells, *Hickory Hash^{TM}: Implementing an Instance of an Algebraic Eraser^{TM} Hash Function on an MSP430 Microcontroller,* 2016, https://eprint.iacr.org/2016/1052.
5. I. Anshel; D. Atkins; D. Goldfeld; P. E. Gunnells, *WalnutDSA$^{\text{TM}}$: A Quantum-Resistant Digital Signature Algorithm*, to appear.
6. I. Anshel; D. Atkins; D. Goldfeld; P. E. Gunnells, *Ironwood Meta Key Agreement and Authentication Protocol*, to appear.

7. I. Anshel; D. Atkins; D. Goldfeld; P. E. Gunnells, *Defeating the Ben-Zvi, Blackburn, and Tsaban Attack on the Algebraic Eraser*, arXiv:1601.04780v1 [cs.CR].

8. D. Bacon; A. Childs; W. van Dam, *From optimal measurements to efficient quantum algorithms for the hidden subgroup problem over semidirect product groups,* 46th Annual IEEE Symposium on Foundations of Computer Science, pages 469–478, 2005.

9. M. Bellare; D. Pointcheval; P. Rogaway, *Authenticated Key Exchange Secure Against Dictionary Attacks*, Advances in Cryptology – Eurocrypt 2000, Lecture Notes in Computer Science.

10. C. H. Bennett; E. Bernstein; G. Brassard; U. Vazirani, *Strengths and weaknesses of quantum computing,* SIAM J. Comput., 26(5):1510–1523, October 1997.

11. A. Ben-Zvi; S. R. Blackburn; B. Tsaban, *A practical cryptanalysis of the Algebraic Eraser*, CRYPTO 2016, Lecture Notes in Computer Science 9814 (2016), 179–189.

12. T. Beth; M. Rötteler, *Polynomial-time solution to the hidden subgroup problem for a class of non-abelian groups,* 1998, arXiv:9812070.

13. 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.

14. J. Birman, *Braids, Links and Mapping Class Groups*, Annals of Mathematics Studies, Princeton University Press, 1974.

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

16. K. Friedl; G. Ivanyos; F. Magniez; M. Santha; P. Sen, *Hidden translation and orbit coset in quantum computing.* In Proc. 35th Annual ACM Symposium on Theory of Computing, pages 1–9, 2001.

17. D. Garber; S. Kaplan; M. Teicher; B. Tsaban; U. Vishne, *Length-based conjugacy search in the braid group,* Algebraic methods in cryptography, 75-87, Contemp. Math., 418, Amer. Math. Soc., Providence, RI, 2006.

18. D. Gavinsky, *Quantum solution to the hidden subgroup problem for poly-near- hamiltonian groups,* Quantum Information and Computing, 4:229–235, 2004.

19. V. Gebhardt, *A new approach to the conjugacy problem in Garside groups*, J. Algebra 292(1) (2005), 282–302.

20. M. Grigni; L. Schulman; M. Vazirani; U. Vazirani, *Quantum mechanical algorithms for the nonabelian hidden subgroup problem,* in Proc. 33rd Annual ACM Symposium on Theory of Computing, page 6874, 2001.

21. L. K. Grover, *A fast quantum mechanical algorithm for database search,* Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212.

22. D. Goldfeld and P. Gunnells, *Defeating the Kalka-Teicher-Tsaban linear algebra attack on the Algebraic Eraser*, Arxiv eprint 1202.0598, February 2012.

23. P. Gunnells, *On the cryptanalysis of the generalized simultaneous conjugacy search problem and the security of the Algebraic Eraser,* arXiv:1105.1141v1 [cs.CR].

24. M. I. G. Vasco; S. Magliveras and R. Steinwandt, *Group-theoretic cryptography*, Chapman & Hall / CRC Press, (2015).

25. D. Hofheinz; R. Steinwandt, *A practical attack on some braid group based cryptographic primitives,* Public Key Cryptography, Proceedings of PKC 2003 (Yvo Desmedt, ed.), Lecture Notes in Computer Science, no. 2567, Springer-Verlag, 2002, pp. 187-198.

26. A. Kalka, M. Teicher and B. Tsaban, *Short expressions of permutations as products and cryptanalysis of the Algebraic Eraser*, Advances in Applied Mathematics 49 (2012), 57-76.

27. A. Y. Kitaev, *Quantum measurements and the Abelian stabilizer problem,* 1995, quant-ph/9511026.

28. K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, and C. Park, *New public-key cryptosystem using braid group,* in Advances in Cryptology–CRYPTO 2000 (M. Bellare, ed.), Lecture Notes in Computer Science 1880 (Springer, Berlin, 2000), 166–183.

29. C. Kudla and K. Paterson, *Modular Security Proofs for Key Agreement Protocols*, in Advances in Cryptology–ASIACRYPT 2005, Lecture Notes in Computer Science 3788.

30. C. Lomont, *The hidden subgroup problem - review and open problems,* 2004, arXiv:0411037.

31. W. Magnus; A. Karrass; D. Solitar, *Combinatorial group theory: Presentations of groups in terms of generators and relations,* Interscience Publishers (John Wiley & Sons, Inc.), New York-London-Sydney (1966).

32. A. D. Myasnikov and A. Ushakov, *Cryptanalysis of the Anshel-Anshel-Goldfeld-Lemieux key agreement protocol*, Groups Complex. Cryptol. 1 (2009), no. 1, 63-75.

33. A. G. Myasnikov, V. Shpilrain, and A. Ushakov, *Group-based Cryptography,* Advanced Courses in Mathematics CRM Barcelona (Birkhäuser, Basel, 2008).

34. M. S. Paterson; A. A. Razborov, *The Set of Minimal Braids is co-NP-Complete,* J. Algorithms,12, (1991), 393–408.

35. P. Shor, *Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,* SIAM J. on Computing, (1997) 1484–1509.

36. T. F. Smith; M. S. Waterman (1981), *Identification of Common Molecular Subsequences*, Journal of Molecular Biology, 147: 195–197. doi:10.1016/0022-2836(81)90087-5. PMID 7265238.

37. K. Tatsuoka, *An isoperimetric inequality for the Artin groups of finite type,* Trans. of the Amer. MBest,ath. Soc., Volume 339, Number 2 (1993), 537–551.