# Ironwood Meta Key Agreement and Authentication Protocol

Iris Anshel*, Derek Atkins†, Dorian Goldfeld‡ and Paul E. Gunnells§
SecureRF Corporation, 100 Beard Sawmill Rd #350, Shelton, CT 06484
Email: *ianshel@securerf.com, †datkins@securerf.com, ‡dgoldfeld@securerf.com,
§pgunnells@securerf.com

***Abstract*—Number theoretic public-key solutions, used in many applications worldwilde, will be subject to various quantum attacks, making them less attractive for longer-term use. Certain group theoretic constructs are now showing promise in providing quantum-resistant cryptographic primitives, and may provide suitable alternatives for those looking to address known quantum attacks. In this paper, we introduce a new protocol called a Meta Key Agreement and Authentication Protocol (MKAAP) that has some characteristics of a public-key solution and some of a shared-key solution. Specifically it has the deployment benefits of a public-key system, allowing two entities that have never met before to authenticate without requiring real-time access to a third-party, but does require secure provisioning of key material from a trusted key distribution system (similar to a symmetric system) prior to deployment. We then describe a specific MKAAP instance, the Ironwood MKAAP, discuss its security, and show how it resists certain quantum attacks such as Shor's algorithm or Grover's quantum search algorithm. We also show Ironwood implemented on several IoT devices, measure its performance, and show how it performs significantly better than ECC using fewer device resources.**

***Index Terms*—Group Theoretic Cryptography, E-Multiplication, Braids**

## I. Introduction

Group theoretic cryptography is a relatively new discipline and overviews can be found in the two recent monographs [13], [21]. A number of group theoretic key agreement protocols have been introduced in the last two decades, including [4] and [17], but attacks on the conjugacy search problem such as those appearing in [10], [11], [15] suggest that these types of schemes may not be practical over braid groups in low-resource environments. To overcome these concerns, we introduce the notion of a Meta Key Agreement and Authentication Protocol (MKAAP) (see §IV) which has many of the properties and advantages of a public-key method and requires very limited distribution of certain private keys.

Starting with a conjectured quantum-resistant one-way function based in braid group theory we've constructed an MKAAP which we present in this paper. To date, this MKAAP is immune to all known attacks introduced in group theoretic cryptography and delivers linear time performance on low-footprint processors.

### *Previous Work*

In 2006 [1] introduced a key agreement protocol based in group theory (specifically the braid group) that withstood several attacks over the past decade. First [20] determined that if braids are too short then it's possible to find the conjugating factor and use that to break the system. However it was pointed out in [14] that in practice the braids are long enough that this attack can never actually succeed. It's akin to using Fermat to factor short RSA keys, which becomes impractical at "secure" sizes. Second, [16] showed a linear algebra attack (KTT) that would allow an attacker to determine part of the private key data. However, [12] showed that this is just a class of weak keys and by choosing the private key data in a specific way this attack is defeated.

Subsequent to the KTT attack, [7], using all of the available public information, were able to, after a large precomputation and spending several hours, able to reconstruct the shared secret. This attack not only required access to the public parameters but also both public keys (including their permutations). It was shown in [2] that the attack work grows as the size of the permutation order grows as well as the size of the braid group.

The current review of WalnutDSA, see [3], does not apply to the Ironwood protocol because the attempt at reversing E-multiplication requires data not available to an attacker, and hence the underlying hard problems considered in these approaches do not impact the Ironwood security (see §VI).

### *Our Contribution*

This paper introduces the Ironwood MKAAP whose security is based on hard problems in group theory. Ironwood leverages the conjectured one-way function, E-Multiplication, but creates a different construction that removes some of the public information required to mount any of the previous attacks. In addition to being immune from previous attacks, Ironwood also appears to be quantum resistant. Specifically, Shor's quantum algorithm [22] – which has been shown to break RSA, ECC, and several other public-key crypto systems – does not seem applicable for attacking Ironwood. Further, Grover's quantum search algorithm [23] is not as impactful on Ironwood due to the fact that the running time of Ironwood is linear in the key length and security strength.

This paper first reviews the braid group and colored Burau representation. Next it reviews E-Multiplication, and then introduces the meta key agreement and authentication protocol. Following that it introduces Ironwood, discusses its security, and then presents implementation experience.

## II. Colored Burau representation of the braid group

Let $B_N$ denote the braid group on $N$ strands with Artin presentation

$$B_N = \left\langle b_1, b_2, \ldots, b_{N-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{for } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{for } |i-j| \geq 2 \end{array} \right\rangle.$$

Every $\beta \in B_N$ determines a permutation $\sigma_\beta \in S_N$, the group of permutations of $N$ letters, as follows. For $1 \leq i < N$, define $\sigma_i \in S_N$ as the $i^{\text{th}}$ simple transposition, which maps $i \to i+1, i+1 \to i$, and leaves the elements $\{1, \ldots, i-1, i+2, \ldots, N\}$ fixed. We may take $\sigma_{b_i} = \sigma_i$. Then if $\beta = b_{i_1}^{\epsilon_1}\, b_{i_2}^{\epsilon_2}\, \cdots\, b_{i_k}^{\epsilon_k}$, (with $\epsilon_i = \pm 1$), it is easy to see that $\sigma_\beta = \sigma_{i_1} \cdots \sigma_{i_k}$.

The colored Burau representation of the braid group was introduced by Morton in [18] in 1998, but we shall make use of a variation of Morton's original representation. Associate to each Artin generator $b_i$, with $1 \leq i < N$, a colored Burau matrix $CB(b_i)$ where

$$CB(b_1) = \begin{pmatrix} -t_1 & 1 & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$
$$CB(b_i) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & t_i & -t_i & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \tag{1}$$
$\big($for $1 < i < N\big)$.

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

$$CB(b_1^{-1}) = \begin{pmatrix} -\frac{1}{t_2} & \frac{1}{t_2} & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$
$$CB(b_i^{-1}) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & 1 & -\frac{1}{t_{i+1}} & \frac{1}{t_{i+1}} & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \tag{2}$$
$\big($for $1 < i < N\big)$.

Thus each braid generator $b_i$ (respectively, inverse generator $b_i^{-1}$) determines 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 colored Burau pairs such that the natural mapping from the braid group to the group of matrices with entries in the ring of Laurent polynomials in the $t_i$ is a homomorphism.

Given a Laurent polynomial

$$f(t_1, \ldots, t_N) \;\in\; \mathbb{Z}[t_1^{\pm 1}, t_2^{\pm 1}, \ldots, t_N^{\pm 1}],$$

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 $N \times N$ matrices over $\mathbb{Z}[t_1^{\pm 1}, t_2^{\pm 1}, \ldots, t_N^{\pm 1}]$ denoted, $\mathcal{M}$, by acting on each entry in the matrix, and denote the action in the same way. The general definition for multiplying two colored Burau pairs is now defined as follows from the definition of the semidirect product $\mathcal{M} \rtimes S_N$. Given $b_i^\pm, b_j^\pm$, the colored Burau/permutation pair associated with the product $b_i^\pm \cdot b_j^\pm$ is

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

Given any braid

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

with $\epsilon_i = \pm 1$ for $1 \leq i \leq k$, the colored Burau pair $(CB(\beta), \sigma_\beta)$ is given by

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

The colored Burau representation is then defined by

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

One checks that $\Pi_{CB}$ satisfies the braid relations and, hence, defines a representation of $B_N$.

## III. E-MULTIPLICATION

E-Multiplication was first introduced in [1] as a possible one-way function used as a building block to create multiple cryptographic constructions. We recall its definition here.

Let $\mathbf{F}_q$ denote the finite field of $q$ elements. A set of $T$-values is defined to be a collection of non-zero field elements:

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

Given a set of $T$-values, we can evaluate any Laurent polynomial $f(t_1, t_2, \ldots, t_N)$ to obtain an element of $\mathbf{F}_q$:

$$f(t_1, t_2, \ldots, t_N) \downarrow_{t\text{-values}} := f(\tau_1, \tau_2, \ldots, \tau_N).$$

We extend this notation to matrices over Laurent polynomials in the obvious way.

With all these components in place, we can now define E-Multiplication. By definition, E-Multiplication is an operation that takes as input two ordered pairs,

$$(M, \sigma_0), \quad (CB(\beta), \sigma_\beta),$$

where $\beta \in B_N$ and $\sigma_\beta \in S_N$ as before, and where $M \in GL(N, \mathbf{F}_q)$, and $\sigma_0 \in S_N$. We denote E-Multiplication with a star: $\star$. The result of E-Multiplication, denoted

$$(M', \sigma') = (M, \sigma_0) \star (CB(\beta), \sigma_\beta),$$

will be another ordered pair $(M', \sigma') \in GL(N, \mathbf{F}_q) \times S_N$.

We define E-Multiplication inductively. When the braid $\beta = b_i^{\pm}$ is a single generator or its inverse, we put

$$(M, \sigma_0) \star \big(CB(b_i^{\pm}),\ \sigma_{b_i^{\pm}}\big) =$$
$$\Big(M \cdot {}^{\sigma_0}\big(CB(b_i^{\pm})\big) \downarrow_{t\text{-values}},\ \ \sigma_0 \cdot \sigma_{b_i^{\pm}}\Big).$$

In the general case, when $\beta = b_{i_1}^{\epsilon_1} b_{i_2}^{\epsilon_2} \cdots b_{i_k}^{\epsilon_k}$, we put

$$\begin{aligned}&(M, \sigma_0) \star (CB(\beta), \sigma_\beta) = \\ &\quad (M, \sigma_0) \star (CB(b_{i_1}^{\epsilon_1}), \sigma_{b_{i_1}}) \star (CB(b_{i_2}^{\epsilon_2}), \sigma_{b_{i_2}}) \star \\ &\quad \cdots \star (CB(b_{i_k}^{\epsilon_k}), \sigma_{b_{i_k}}),\end{aligned} \tag{3}$$

where we interpret the right of (3) by associating left-to-right. One can check that this is independent of the expression of $\beta$ in the Artin generators.

## IV. META KEY AGREEMENT AND AUTHENTICATION PROTOCOL (MKAAP)

We now introduce the notion of a meta key agreement and authentication protocol which is not a true public-key crypto system but has many of the features of a public-key cryptosystem. Specifically, while it does require secure provisioning of each device by a Trusted Third Party (TTP), once provisioned, devices can authenticate to each other offline without further support. By a *device*, we mean a Probabilistic Polynomial-Time Turing Machine (PPTM) that can execute a cryptographic protocol and be capable of transmitting and receiving messages.

**Definition (MKAAP)** Assume there is a network consisting of a Home Device (HD) and a set of other devices ($D_i$, i=1,2,3,...) that communicate with the HD over an open channel[1]. Assume that there is a TTP which has distributed secret information to the HD and the other devices. An MKAAP is an algorithm with the following properties:

- *The MKAAP allows the HD to authenticate (and/or be authenticated by) and obtain a shared secret with any $D_i$ over an open channel.*

- *It is infeasible for an attacker, eavesdropping on the open communication channel between the HD and a device $D_i$, to obtain the shared secret assuming the attacker does not know the secret information distributed by the TTP.*

- *The private keys of the $D_i$ are provided by the TTP, fixed, and are not known to the HD. The TTP may update the keys over time.*

- *The private key of the HD may be ephemeral and is not known to any of the $D_i$'s, or it may be provided by the TTP.*

- *If an attacker can break into one of the devices $D_i$ and obtain its private key, then only the security of that particular device is breached, all other devices remain secure.*

- *An attacker is assumed to be a PPTM and/or a machine capable of running a quantum method like Shor [22] or Grover [23], capable of passive eavesdropping on all communications between the HD and $D_i$, and can actively attempt to impersonate an HD or $D_i$, using the other side as an oracle.*

## V. IRONWOOD MKAAP

We now describe the Ironwood MKAAP. It may be assumed that the following information is publicly known.

**Public Information:**

- The braid group $B_N$ for a fixed even integer $N \geq 10$.
- A finite field $\mathbf{F}_q$ of $q \geq 7$ elements[2].
- A non-singular matrix $m_0 \in GL(N, \mathbf{F}_q)$.

[1]While the open channel may be a unicast or multicast medium, authentication between HD and $D_i$ is 1:1.

[2]$N$ and $q$ are chosen to meet the desired security level given the best-known attack (which is currently brute force). See Section VII.

• The operation of E-multiplication based on $B_N$ and $\mathbf{F}_q$.

Next, we discuss the initial distribution of secret information by the TTP.

**TTP Data Generation and Distribution:**

The TTP creates two sets of commuting conjugates[3]:

$$\mathcal{C}_\alpha = \{z\alpha_1 z^{-1},\ z\alpha_2 z^{-1},\ \ldots,\ z\alpha_r z^{-1}\},$$

$$\mathcal{C}_\gamma = \{z\gamma_1 z^{-1},\ z\gamma_2 z^{-1},\ \ldots,\ z\gamma_r z^{-1}\} \subseteq B_N,$$

where some portion of the $\alpha_i$ are purebraids (i.e., have a trivial permutation), and one set of $T$-values:

$$T = \{\tau_1, \tau_2, \ldots, \tau_N\} \subset \mathbf{F}_q, \qquad (\tau_i \neq 0, 1).$$

The TTP writes the first set of conjugates $\mathcal{C}_\alpha$ and the set of $T$-values into the memory of the HD.

Next, the TTP creates braid words $\beta_i \in B_N$ (for $i = 1, 2, \ldots$) which are random products of conjugates from the second set $\mathcal{C}_\gamma$ (chosen according to the standard normal distribution) and creates the colored Burau pairs $(\beta_i, \sigma_i)$ where $\sigma_i$ is the permutation associated to $\beta_i$. For each such $(\beta_i, \sigma_i)$, the TTP chooses a random non-singular matrix

$$C_i = \sum_{k=0}^{N-1} c_{k,i} m_0^k, \qquad \big(\text{with } c_{k,i} \in \mathbf{F}_q\big),$$

(where the $c_{k,i} \in \mathbf{F}_q$ are chosen according to the standard normal distribution) and using the $T$-values performs the E-multiplication

$$\text{Pub}_i := (C_i, \text{Id}) \star (\beta_i, \sigma_i) = (C_i M_i, \sigma_i).$$

Here Id is the identity permutation and $M_i \in GL(N, \mathbf{F}_q)$. Finally, the TTP creates a certificate $\text{Cert}_i$ which contains a digitally signed copy of $\text{Pub}_i$ and writes $\text{Cert}_i$ and $C_i$ into the memory of $D_i$, the $i^{th}$ device in the network.

Once the TTP distribution is completed authentication and key agreement between the HD and the other devices in the network may begin. A key assumption is that there is only one HD and that the secret information on the HD is secure and cannot be obtained by any adversary. The protocol proceeds as follows.

**Ironwood Authentication and Key Agreement Protocol**

**Step 1:** The device $D_i$ sends HD the certificate $\text{Cert}_i$ which contains a copy of $\text{Pub}_i$ which has been digitally signed by the TTP. Here $\text{Pub}_i$ is the public key of $D_i$ and the matrix $C_i$ is the private key of $D_i$.

[3] $r$ is chosen for a time/space/keysize tradeoff, but must be $\geq 2$. The larger $r$ implies a longer time to generate the conjugates and more to store, but shorter random words.

**Step 2:** The HD generates two ephemeral non-singular matrices

$$C = \sum_{k=0}^{N-1} c_k m_0^k, \qquad C' = \sum_{k=0}^{N-1} c'_k m_0^k,$$
$$\big(\text{with } c_k, c'_k \in \mathbf{F}_q\big).$$

**Step 3:** The HD generates an ephemeral permutation $\sigma$ and two ephemeral braids $\beta, \beta'$ which are random words in $\mathcal{C}_\alpha$ and which have the same permutation $\sigma = \sigma_\beta = \sigma_{\beta'}$. This can be accomplished efficiently by first generating a braid using the first half of conjugates, and then create the second braid by using the same set of conjugates and adding choices from the set of conjugates where $\alpha_i$ are purebraids.

**Remark:** *This completes the construction of the ephemeral part of the private key of the* HD *which consists of $C, C', \beta, \beta', \sigma$. The $T$-values and the set of conjugates $\mathcal{C}_\alpha$ are also part of the private key of the* HD *and must be treated as confidential information.*

**Step 4:** Using the $T$-values, the HD computes the following two E-multiplications:

$$(C, \text{Id}) \star (\beta, \sigma) := (CM, \sigma),$$
$$(C', \text{Id}) \star (\beta', \sigma) := (C'M', \sigma).$$

**Step 5:** The HD has received $\text{Pub}_i = (C_i M_i, \sigma_i)$ in the signed digital signature sent by $D_i$. Next, using the $T$-values, the HD computes the following two E-multiplications:

$$(CC_iM_i, \sigma_i) \star (\beta, \sigma) := (Y,\ \sigma_i\sigma),$$
$$(C'C_iM_i, \sigma_i) \star (\beta', \sigma) := (Y',\ \sigma_i\sigma).$$

**Step 6:** The HD computes:

$$s = (N/2)^{\text{th}} \text{ column of the matrix } Y,$$
$$s' = (N/2)^{\text{th}} \text{ column of the matrix } Y'.$$

**Step 7:** The HD sends $D_i$ the pair:

$$\big(C'M'M^{-1}C^{-1},\ s\big).$$

**Step 8:** The device $D_i$ computes the matrix and vector multiplications:

$$s' = C_i\left(C'M'M^{-1}C^{-1}\right)C_i^{-1} \cdot s$$

which it can do since it knows its private key $C_i$ and has received $C'M'M^{-1}C^{-1}$ and $s$ from the HD.

**Shared Secret:** The shared secret is the column vector $s'$ known to both HD and $D_i$.

**Step 9:** The final step is to authenticate the device $D_i$. Mutual authentication can be established by checking that the HD and $D_i$ have obtained the same shared secret. This is because the device $D_i$ has sent the HD the signed certificate containing a copy of its public key and the unique HD is the only entity with access to the secret conjugate material and $T$-values enabling it to produce the correct response. Methods for doing this, such as using a hash to create a validation value or using a nonce and Message Authentication Code (MAC) in a challenge/response protocol, are well known, so we do not reproduce them here.

At this point the devices can mutually authenticate. The HD can authenticate the device $D_i$ by verifying its certificate and then having $D_i$ prove knowledge of the private matrix associated with the public matrix in the certificate. The device $D_i$ proves this knowledge by showing that it can generate the same shared secret as the HD.

In the other direction, the $D_i$ device can authenticate the HD if the HD proves that it has created the same shared secret and $D_i$ verifies that $C'M'M^{-1}C^{-1} \neq C_i\left(C'M'M^{-1}C^{-1}\right)C_i^{-1}$. The latter verification thwarts a trivial spoof of the HD (where $\beta, \beta' = 1$). The former condition is sufficient because only one HD contains the conjugate data to generate the HD keypair, so only that HD could generate a public key that would create the same shared secret.

It is not at all obvious that the column vector $s'$ produced by the HD and $D_i$ have to be the same. We now provide a proof of this.

First of all, the braids $\beta$ and $\beta'$ commute with $\beta_i$, since they are formed from the sets of conjugates $\mathcal{C}_\alpha, \mathcal{C}_\gamma$, respectively, and these sets of conjugates commute. It follows from step 5 that

$$\begin{aligned}(CC_iM_i, \sigma_i) \star (\beta, \sigma) &= \left(C_iCM, \sigma\right) \star (\beta_i, \sigma_i)\\ &= (Y, \sigma_i\sigma),\\ (C'C_iM_i, \sigma_i) \star (\beta', \sigma) &= \left(C_iC'M', \sigma\right) \star (\beta_i, \sigma_i)\\ &= (Y', \sigma_i\sigma).\end{aligned}$$

Now, define an unknown matrix $X$ by the formula

$$(1, \sigma) \star (\beta_i, \sigma_i) = (X, \sigma_i)\,.$$

It follows that

$$Y = C_iCMX, \qquad Y' = C_iC'M'X.$$

Next, define a column vector $x$ where

$$x = (C_iCM)^{-1} \cdot s.$$

The column vector $x$ is just the $(N/2)^{\text{th}}$ column of the matrix $X$. Hence

$$s' = C_iC'M' \cdot x = C_iC'M'M^{-1}C^{-1}C_i^{-1} \cdot s.$$

## VI. Security Analysis of Ironwood

The Ironwood protocol is an outgrowth of the Algebraic Eraser$^{\text{TM}}$ key agreement protocol (AEKAP) first published in [1] in 2006. The security of the AEKAP was based on the difficulty of inverting E-multiplication and the hard problem of solving the simultaneous conjugacy search problem for subgroups of the braid group. The AEKAP had withstood numerous attacks (see [10], [11], [12], [14], [15], [20]) in the last 10 years. However, the recent successful attack of Ben-Zvi, Blackburn, Tsaban (BBT) [7], for small parameter sizes, requires an increase in key size (see [2]) to make AEKAP secure against the BBT attack.

The Ironwood protocol was designed to be totally immune to the BBT attack [7] without compromising on key size, speed or power consumption. A necessary requirement for the security of Ironwood is that the $T$-values and/or conjugates which are distributed to the HD cannot be obtained by an adversary. The $T$-values and conjugates are not on any of the other devices $D_i$ in the network. Without knowing the $T$-values and conjugates the BBT attack [7] cannot proceed at all.

It is also clear that the Ironwood protocol satisfies the last requirement of an MKAAP that if an attacker can break into one of the devices $D_i$ and obtain its private key, then only the security of that particular device is breached, all other devices remain secure. This is because the only secret information on the device $D_i$ is the private key $C_i$. Knowledge of $C_i$ has no effect on the key agreement and authentication protocol between the HD and other devices $C_j$ with $j \neq i$.

Recent discussions about WalnutDSA (see [3]) are not relevant to the Ironwood protocol. The first of these approaches is linear algebraic in nature and focuses on finding a braid $\beta$ where the $(1,1) \star \beta$ is known. The Ironwood protocol does not depend on this process being hard and, in fact, the more general question of finding $\beta$ when $(M, \sigma) \star \beta$ is known remains unaddressed. The more recent approach focuses on the use of cloaking elements in WalnutDSA, which do not appear in the Ironwood protocol.

We now present a preliminary informal security analysis of Ironwood.

*Reversing E-multiplication is Algorithmically Hard*

Strong support for the hardness of reversing E-Multiplication can be found in [19] which studies the security of Zémor's [25] hash function $h : \{0,1\}^* \to SL_2(\mathbb{F}_q)$, with the property that $h(uv) = h(u)h(v)$, where $h(0), h(1)$ are fixed matrices in $SL_2(\mathbb{F}_q)$ and $uv$ denotes concatenation of the bits $u$ and $v$. For example a bit string $\{0,1,1,0,1\}$ will hash to $h(0)h(1)h(1)h(0)h(1)$. Zemor's hash function has not been broken since its inception in 1991. In [19] it is shown that feasible cryptanalysis for bit strings of length 256 can only be applied for very special instances of $h$. Now E-Multiplication, though much more complex, is structurally similar to a Zémor-type scheme involving a large finite number of fixed matrices in $SL_2(\mathbb{F}_q)$ instead of just two matrices $h(0), h(1)$. Further, E-multiplication is highly non-linear (in contrast to ordinary matrix multiplication) since it involves permutation of variables of Laurent polynomials. This serves as an additional basis for the assertion that E-Multiplication is difficult to reverse.

*Invalid Public-Key Attack*

We now consider an invalid public-key attack of the type presented in [8]. Such an attack assumes that an adversary can impersonate the HD and run the Ironwood authentication protocol (using invalid public keys) with a device $D_i$.

This type of attack can be easily defeated (see [6]) provided the $D_i$ uses a hash to create a validation value that does not reveal the shared secret in any way or the $D_i$ uses a nonce and Message Authentication Code (MAC) in a challenge/response protocol.

Consider now the reverse case where a rogue device $D_i$ is trying to attack the HD by sending an invalid public key to the HD. If the HD reveals $s$ to a rogue $D_i$ using an invalid public-key attack it may lead to potential leakage. There are two approaches to protect against an invalid public-key attack against the HD. In the first case, the device $D_i$ would have its public key signed by a trusted CA/TTP. This would allow the HD to check that the public key of the device $D_i$ is valid by validating the certificate. If the certificate is not valid the protocol terminates. This is the method we propose in Step 1 of the Ironwood Protocol. Next, assume certificates are not being used in the protocol. Then the HD must check that the public key of $D_i$ is not invalid before releasing $s$. To do this it must ensure that a sufficient number of elements in the public key matrix are not zero. With a sufficient number of non-zero entries the E-Multiplication process will ensure a sufficient mixing in the resulting computations, eliminating the possibility of using linear algebra to obtain information about the private key of the HD.

In both cases the use of single-use ephemeral keys prevent an attack. If an attacker works against an HD (or a $D_i$) which uses a single-use ephemeral key, then multiple invalid-key attacks would always return unique responses.

*Length Attacks and Simultaneous Conjugacy Search Attacks*

Although AEKAP has withstood length attacks and simultaneous conjugacy search attacks (see [14]) of the type presented in [10], [11], [15], [20], these attacks completely fail for Ironwood. This is because it is assumed that the two sets of conjugates, $\mathcal{S}_\alpha, \mathcal{S}_\gamma$, are not known to an adversary. These two sets of conjugates are not in memory on any of the devices $D_i$, and only one of the sets $\mathcal{S}_\alpha$ is in memory on the HD. An assumption of Ironwood is that an adversary cannot obtain secret information stored on the HD.

*A Class of Weak Keys*

It is crucial that $C_i$ does not commute with $M'M^{-1}$. Otherwise an adversary can compute

$$s' = (C'M') \cdot (CM)^{-1} \cdot s.$$

Similarly, it is also crucial that $M_i$ does not commute with $(C'M') \cdot (CM)^{-1}$. Otherwise an attacker can compute

$$s' = (C_iM_i) \cdot (C'M')^{-1} \cdot (CM) \cdot (C_iM_i)^{-1} \cdot s.$$

The probability that one of the above commuting occurs is very small. An upper bound for the probability that two matrices commute in $GL(N, F_q)$ can be determined as follows. It is well known that there are

$$\prod_{k=0}^{N-1} (q^N - q^k)$$

elements in $GL(N, F_q)$, denoted $\#(GL(N, F_q)$. On taking logarithms, summing over $k$, and exponentiating back, it may be shown that

$$\#(GL(N, F_q) \geq q^{N(N-1) - \frac{N}{q \log q}}$$

for $N, q \geq 8$. For two matrices $X, Y \in GL(N, F_q)$ to commute, $X$ must be in the centralizer of $Y$, and for a generic matrix $X$, its centralizer consists of polynomials in $X$. The number of such polynomials is at most $q^N$. So an upper bound for the probability that two matrices in $GL(N, F_q)$ commute is given by

$$\frac{q^N}{q^{N(N-1) - \frac{N}{q \log q}}}.$$

For example, when $N = 16$ and $q = 256$, the upper bound for the probability is $3.815 \times 10^{-540}$.

*Quantum Resistance of Ironwood*

The Ironwood MKAAP and underlying E-Multiplication appear resistant to known quantum attacks. The following sections provide an overview and analysis.

*Resistance to Shor's Quantum Algorithm*

Shor's quantum algorithm [22] enables a sufficiently large quantum computer to factor numbers or compute discrete logs in polynomial time, effectively breaking RSA, ECC, and DH. It relies on the existence of a fast quantum algorithm to solve the Hidden Subgroup Problem (HSP) when the hidden subgroup is a finite cyclic group. It is known that HSP can be solved on a quantum computer when the hidden subgroup is abelian [24].

Ironwood, but more specifically E-Multiplication, are constructions based on the infinite non-abelian braid group. In fact, the braid group is torsion free and, hence, has no finite subgroups. As a result, there seems to be no way to apply Shor's algorithm to attack Ironwood.

*Resistance to Grover's Quantum Search Algorithm*

Grover's quantum search algorithm [23] allows a Quantum computer to search for a particular element in an unordered $n$-element set in a constant times $\sqrt{n}$ steps as opposed to a constant times $n$ steps required on a classical computer. Resistance to Grover's search algorithm requires increasing the search space. Since E-Multiplication scales linearly, this means that if an attacker has access to a quantum computer running Grover's algorithm, it is only necessary to double the running time of Ironwood to maintain the same security level that currently exists for attacks by classical computers. In comparison, the running time of ECC would have to increase by a factor of 4 since ECC is a based on a quadratic algorithm.

*Brute Force Attacks on the Ironwood Key Agreement Protocol*

We now discuss the security level of the individual secret components in the Ironwood protocol. For accuracy we give the following definition of *security level*.

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

We assume that Ironwood is running on the braid group $B_N$ over the finite field $\mathbf{F}_q$. Note that there are $q^N$ polynomials of degree $N-1$ over $\mathbf{F}_q$. So a brute force search for a particular polynomial of degree $N-1$ over $\mathbf{F}_q$ has security level $q^N$.

• The brute force security level of the matrix $C_i$ is $q^N$.

• The brute force security levels of the matrices $C, C'$ are $q^N$.

The $T$-values is a set of field elements $\{\tau_1, \tau_2, \ldots, \tau_N\}$ where none of the $\tau_i$ = 0 or 1.

• The brute force security level of the $T$-values is $(q-2)^N$.

Note that the size of the public keys $\text{Pub}_i$ of the devices $D_i$ is $N^2 \cdot \log_2(q) + N \log_2(N)$ and the size of the public key of the HD is $(N^2+N) \cdot \log_2(q)$. We can thus assert

• The brute force security level of the exchanged key is $2^{N \log_2(q)} = q^N$.

• The brute force security level of either of the private braids $\beta, \beta'$ is

$$SL \ > \ (2r)^L$$

where $L$ is the length of the braid as a word in the conjugates assigned to the HD, and hence we have lower bound

$$SL \ > \ \min((2r)^L, (q-2)^N).$$

A passive eavesdropper only gains access to the public keys and s-column. That does not provide enough information to reproduce the shared secret. In that E-Multiplication is conjectured to be a one-way function, knowledge of $(C_i M_i, \sigma_i)$ does not enable an attacker to learn $C_i$, which would be required to compute the shared secret as $D_i$. Similarly, knowledge of $(C'M'M^{-1}C^{-1})$ does not provide enough information to deduce $C, C', \beta$, or $\beta'$. This prevents computing the shared secret as the HD using $D_i$'s public key.

If an attacker breaks into one $D_i$ device and reads out its key material, they cannot use that against another device $D_j$ ($i \neq j$). Each device's matrices are independently generated, so knowledge of one provides no information about any other device keys.

It has become standard in the art to give security proofs for both asymmetric key exchange protocols and digital signatures. The structure of such proofs do not lend themselves to the Ironwood protocol (or any other MKAAP), and as of this writing no other security proof which would has been introduced to the field.

## VII. Implementation Experience

For testing purposes Ironwood was implemented on multiple platforms. Because the Other Devices only need to perform a single matrix multiplication and a single vector multiplication, we focused our effort on the requirements of the HD, as those operations are more consuming and therefore more interesting to explore.

Operationally the HD needs to perform two sets of E-Multiplication operations (one with $\beta$ and another with $\beta'$), which take the majority of the execution time. A single E-Multiplication operation in $B_N$ requires $N$ multiplies and $2N$ additions over the finite field $F_q$. These operations, in turn, gets multiplied by the number of Artin generators in each braid.

We generated key material using $B_{16}F_{256}$ for a proposed $2^{128}$ security level. We generated 32 conjugates for each set and from there generated key material for testing. For this testing we generated 10 sets of HD keys which averaged a braid length of 2659.2 Artin Generators for $\beta$ and 4302.4 for $\beta'$.

The first platform tested was a Texas Instruments (TI) MSP430 16-bit (model) microcontroller. This platform runs at various speeds from 8Mhz to 30Mhz (or faster). On this platform we used the IAR (2011) compiler, version 5.40.1 with Optimizations set to High and all transformations and unrolling options checked. With this setting the Ironwood HD implementation built into 3126 bytes of ROM and ran with 354 bytes of RAM. Running over the 10 keys, the MSP430 required anywhere from 4,532,480 to 6,002,668 cycles with an average of 5,309,182. At 25MHz this equates to an average runtime of 212ms. Compare this to Curve25519 in [9] which takes 9,119,840 cycles in 13,112 bytes of ROM and 384 bytes of RAM. Ironwood does not require a hardware multiplier.

The second platform was an NXP LPC1768 running at 48MHz, which contains an embedded ARM Cortex M3. We compiled our code using GCC (arm-none-eabi-gcc) version 4.9.3 using optimization level -O3. This built down into 2578 bytes of ROM and the runtime required 1192 bytes of RAM. Running the Ironwood shared secret calculation over the 10 keys, this ARM platform required anywhere from 1,538,472 to 2,026,216 cycles to compute a shared secret, resulting in a runtime of 32.1 to 42.2ms (averaging 37.4ms). Compare this to Curve25519 in [9] which required 3,589,850 cycles (on a Cortex-M0) in 7,900 bytes of ROM and 584 bytes of RAM.

The third platform was a TI CC2650, an embedded ARM Cortex M3 running at 48MHz on TI-RTOS. On this platform we used TI's arm compiler (listed as TI v5.2.0). It was configured at optimization level 4 (Whole Program optimizations) with a size-speed tradeoff (SST) of 5 (ranging from 0 to 5, 0 being fully size optimized, 5 being fully speed optimized). At this level the code used 3568 bytes of ROM and 1192 bytes of RAM. With this setting Ironwood computed a shared secret in an average of 37.4ms.

We also performed tests using the size-speed tradeoff of 2, which resulted in a smaller code size of only 1954 bytes of ROM and resulted in a very minor speed penalty, reducing the average computation time to 37.6ms. Note that on this platform we couldn't get a cycle count, only a timer, and the timer API only has a $2^{16}$ cycle resolution timer, which means the timer increments every $2^{16}/48^{6} = 1.37$ms. This implies the timer results are +/- 0.7ms. However the times are still on par with the timing on the LPC1768.

TABLE I
PERFORMANCE ON MSP430, LPC1768 (IN CYCLES)

| **Artin Length** $\lvert\beta\rvert$ | $\lvert\beta'\rvert$ | **MSP430** | **LPC1768** |
|---|---|---|---|
| 2626 | 5272 | 6002668 | 2026216 |
| 2332 | 3580 | 4532480 | 1538472 |
| 2414 | 3944 | 4862464 | 1648742 |
| 3172 | 4266 | 5661952 | 1914009 |
| 2168 | 4514 | 5101824 | 1728545 |
| 3092 | 4698 | 5922048 | 2000312 |
| 2978 | 3968 | 5297664 | 1792959 |
| 2744 | 4420 | 5459456 | 1845502 |
| 2430 | 4762 | 5479424 | 1854446 |
| 2636 | 3600 | 4771840 | 1617670 |
| 2659.2 | 4302.4 | 5309182 | 1796687 |

TABLE II
PERFORMANCE ON MSP430, LPC1768, CC2650 (IN MS)

| **Artin Length** | | **MSP430** | **LPC1768** | **CC2650 48Mhz** | |
|---|---|---|---|---|---|
| $\lvert\beta\rvert$ | $\lvert\beta'\rvert$ | **25MHz** | **48MHz** | **(SST 5)** | **(SST 2)** |
| 2626 | 5272 | 240.1 | 42.2 | 42 | 42 |
| 2332 | 3580 | 181.3 | 32.2 | 32 | 32 |
| 2414 | 3944 | 194.5 | 34.3 | 34 | 35 |
| 3172 | 4266 | 226.5 | 39.9 | 40 | 40 |
| 2168 | 4514 | 204.1 | 36.0 | 36 | 36 |
| 3092 | 4698 | 236.9 | 41.2 | 42 | 42 |
| 2978 | 3968 | 211.9 | 37.4 | 37 | 37 |
| 2744 | 4420 | 218.4 | 38.4 | 38 | 39 |
| 2430 | 4762 | 219.2 | 38.6 | 39 | 39 |
| 2636 | 3600 | 190.9 | 33.7 | 34 | 34 |
| 2659.2 | 4302.4 | 212.4 | 37.4 | 37.4 | 37.6 |

We should note that implementations of the Ironwood Other Device are approximately 50-times faster than the HD computations.

## VIII. CONCLUSION

In this paper we have introduced a new concept called a Meta Key Agreement and Authentication Protocol and defined an instance of this protocol called the Ironwood MKAAP. We show how it resists the range of known attacks against E-Multiplication based protocols and how, in addition, it is quantum resistant in that it resists both the Shor and Grover algorithms.

Implementations of Ironwood have been built and tested on multiple platforms, and we have shown the performance numbers achieved on three different platforms leveraging two different architectures. Specifically, we show that we can achieve a key agreement on an MSP430 in 212ms and 37ms on an ARM Cortex M3 acting as the HD, which is twice as fast as Curve25519 using less code.

# REFERENCES


[1] 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.

[2] 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].

[3] I. Anshel, D. Atkins, D. Goldfeld, P. E. Gunnells, *WalnutDSA$^{TM}$: A Lightweight Quantum Resistant Digital Signature Algorithm,* https://eprint.iacr.org/2017/058.pdf.

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

[5] E. Artin, The theory of braids, Annals of Math. 48 (1947) 101–126.

[6] D. Atkins; D. Goldfeld, Addressing the algebraic eraser over the air protocol, https://eprint.iacr.org/2016/205.pdf.

[7] A. Ben-Zvi, S. R. Blackburn and B. Tsaban, *A practical cryptanalysis of the Algebraic Eraser*, Advances in Cryptology – CRYPTO 2016, to appear. See http://eprint.iacr.org/2015/1102.

[8] S. R. Blackburn and M.J.B. Robshaw, *On the security of the Algebraic Eraser tag authentication protocol*, 14th International Conference on Applied Cryptography and Network Security (ACNS 2016), to appear. See http://eprint.iacr.org/2016/091.

[9] M. Düll; B. Haase; G. Hinterwälder; M. Hutter; C. Paar; A. H. Sánchez; P. Schwabe, *High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers*, https://eprint.iacr.org/2015/343.pdf

[10] 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.

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

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

[13] M. I. González Vasco, S. Magliveras and R. Steinwandt, *Group-theoretic cryptography*, Chapman & Hall / CRC Press, (2015).

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

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] C. Mulland, B. Tsaban; *SL2 homomorphic hash functions: Worst case to average case reduction and short collision search,* arXiv:1306.5646v3 [cs.CR] (2015).

[20] 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.

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

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

[23] 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.

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

[25] G. Zémor; *Hash functions and graphs with large girths,* Eurocrypt '91, Lecture Notes in Computer Science 547 (1991), 508–511.