Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

IRONWOOD META KEY AGREEMENT AND AUTHENTICATION PROTOCOL

Iris Anshel*

100 Beard Sawmill Rd
Suite 350
Shelton, CT 06484, USA

Derek Atkins, Dorian Goldfeld and Paul E. Gunnells

100 Beard Sawmill Rd
Suite 350
Shelton, CT 06484, USA

(Communicated by the associate editor name)

Abstract. Number theoretic public-key solutions, currently used in many applications worldwide, 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 "internet of things" (IoT devices), measure its performance, and show how it performs significantly better than ECC using fewer device resources.

1. **Introduction.** Group theoretic cryptography is a relatively new discipline that seeks to bring the core algorithmically difficult problems in combinatorial group theory into the cryptographic landscape. Overviews can be found in the two recent monographs [13], [23]. Among the first generation of group theoretic based key agreement protocols to be introduced, including [4] and [18], were those based of the conjugacy search problem. The attacks on the conjugacy search problem, such as those appearing in [10], [11], [16] 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*

2010 *Mathematics Subject Classification.* Primary: 94A60, 20F36, 68P25; Secondary: 81P94, 20G40, 20E36.

Key words and phrases. Public-key cryptography, Braids, Diffie-Hellman, Post-Quantum cryptography, Internet of Things, IoT..

Protocol (MKAAP) (§4). This protocol has many of the properties and advantages of a public-key method and requires very limited distribution of certain private keys.

In this paper we present an MKAAP based on a conjectured quantum-resistant one-way function based in braid group theory. To date, this MKAAP is immune to all known attacks introduced in group theoretic cryptography and delivers linear time performance on low-footprint processors. As the IoT becomes ubiquitous, the need to secure low-footprint processors can be achieved using group theoretic cryptography via the MKAAP introduced below.

Previous Work. In 2006 [1], I. Anshel (IA) and D. Goldfeld (DG) with M. Anshel and S. Lemieux introduced a key agreement protocol based in group theory (specifically the braid group) that has withstood several attacks over the past decade. First, Myasnikov–Ushakov [22] determined that if braids are too short then one can find the conjugating factor and use that to break the system. However, it was pointed out by P. Gunnells (PG)[15] that in practice the braids are long enough that this attack can never succeed: the method in [22] is analogous to using Fermat's technique to factor short RSA keys, which becomes impractical at secure sizes. Second, Kalka–Teicher–Tsaban [17] described a linear algebra attack (KTT) that would allow an attacker to determine part of the private key data. DG and PG[12] showed, however, that this attack succeeds only on a class of weak keys, and that choosing the private key data more carefully defeats this attack. Subsequent to the KTT attack, Ben-Zvi–Blackburn–Tsaban [7], using all of the available public information of the protocol, were able to reconstruct the shared secret, but only after a large precomputation and several hours of runtime. We later showed [2] that the work necessary to carry out the attack increases as the size of the permutation order grows as well as the size of the braid group.

Some have (incorrectly) questioned the security of Ironwood based on the analysis of other group theoretic cryptographic methods. We remark that the current review of WalnutDSA$^{\text{TM}}$ [3], a group theoretic based digital signature, does not apply to the Ironwood protocol. In particular the (exponential) attack on 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* (Ironwood); its 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, it can be argued that Ironwood is resistant to the current generation of quantum attacks. Specifically, Shor's quantum algorithm[24] – which has been shown to break RSA, ECC, and several other public-key crypto systems – does not seem applicable for attacking Ironwood because the underlying group theoretic foundation of Ironwood is an infinite non-commutative group. Further, Grover's quantum search algorithm[14] is not as impactful on Ironwood as it is on many protocols because the running time of Ironwood is linear in the key length and security strength, and, hence, the need to double the security level amounts to doubling the execution time. This stands in stark contrast to the rapid increase in execution time for standard protocols when the lengths of the private keys are doubled.

This paper begins with a review of the braid group, the colored Burau representation, and E-Multiplication. With these tools in place, we introduce the concept of a meta key agreement and authentication protocol (MKAAP), and present Ironwood. A discussion of security and our implementation experience follows.

2. **Colored Burau representation of the braid group.** Let B_N denote the braid group on N strands, with Artin presentation

$$B_N = \Big\langle b_1, b_2, \ldots, b_{N-1} \;\Big|\; b_i b_j b_i = b_j b_i b_j \quad \text{for } |i-j| = 1, \quad b_i b_j = b_j b_i \quad \text{for } |i-j| \geq 2 \Big\rangle.$$

Let S_N be the permutation group on N letters. Every element $\beta \in B_N$ determines a permutation $\sigma_\beta \in S_N$ as follows. For $1 \leq i < N$, let $\sigma_i \in S_N$ be simple transposition that exchanges i and $i+1$ and leaves the remaining elements $\{1, \ldots, i-1, i+2, \ldots, N\}$ fixed. We write $\sigma_{b_i} = \sigma_i$. Then if $\beta = b_{i_1}^{\epsilon_1}\, b_{i_2}^{\epsilon_2} \cdots b_{i_k}^{\epsilon_k}$, (with $\epsilon_i = \pm 1$), we have $\sigma_\beta = \sigma_{i_1} \cdots \sigma_{i_k}$.

The colored Burau representation of the braid group was introduced by Morton in [20] 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

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

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

$\big(\text{for } 1 < i < N\big)$.

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

$$CB(b_1^{-1}) = \begin{pmatrix} -\frac{1}{t_2} & \frac{1}{t_2} & & & \\ & \ddots & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix},$$

$$CB(b_i^{-1}) = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & 1 & -\frac{1}{t_{i+1}} & \frac{1}{t_{i+1}} & & \\ & & & \ddots & & \\ & & & & 1 & \end{pmatrix} \tag{2}$$

$\big(\text{for } 1 < i < N\big)$.

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 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 $f \mapsto {}^{\sigma}f$:

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

Let $\mathcal{M}$ be the $N \times N$ matrices over $\mathbb{Z}[t_1^{\pm 1}, t_2^{\pm 1}, \ldots, t_N^{\pm 1}]$. We extend this action to $\mathcal{M}$ by acting on each entry in a matrix, and use the same notation for the action. 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

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

The colored Burau representation is then defined by

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

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

3. **E-Multiplication.** E-Multiplication was first introduced in [1] as a core building block for a range of cryptographic constructions. We recall its definition here.

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

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

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 $\mathbb{F}_q$:

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

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 $M \in GL(N, \mathbb{F}_q)$, 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 $(M', \sigma') \in GL(N, \mathbb{F}_q) \times S_N$.

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

$$(M, \sigma_0) \star \left(CB(b_i^\pm),\ \sigma_{b_i^\pm}\right) =$$

$$\left(M \cdot {}^{\sigma_0}\left(CB(b_i^\pm)\right) \downarrow_{t\text{-values}},\ \ \sigma_0 \cdot \sigma_{b_i^\pm}\right).$$

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

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

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

4. **Meta Key Agreement and Authentication Protocol (MKAAP) .** We now introduce the notion of a meta key agreement and authentication protocol, which has many of the features of a public-key cryptosystem. While the MKAAP has many public-key cryptosystem features, it is not a true public-key cryptosystem. Specifically, while it does initially require secure provisioning of each device by a Trusted Third Party (TTP), after they are provisioned devices can authenticate to each other offline without further support. By *device*, we mean a Probabilistic Polynomial-Time Turing Machine (PPTM) that can execute a cryptographic protocol and is 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, $i = 1, 2, 3, \ldots$ 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:

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

• *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 Probabilistic Polynomial-Time Turing Machine (PPTM) and/or a machine capable of running a quantum method like Shor*[24] *or Grover*[14]*, 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.*

There are many benefits of an MKAAP system is over a pure symmetric solution. The MKAAP requires the endpoints to be provisioned similarly to a symmetric solution. However, in a symmetric solution, the HD must be provisioned with the keys for *every* D_i, or at least must have real-time access to the TTP to obtain such keys. In MKAAP, however, the HD can be provisioned *before* keys for the D_i are even generated, and the HD has no need to contact the TTP. This allows for additional devices to be created and provisioned after the HD is already deployed without requiring any changes to the HD. This would not be possible with a symmetric solution.

5. **Description of 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 $\mathbb{F}_q$ of $q \geq 7$ elements.[2]
- A non-singular matrix $m_0 \in GL(N, \mathbb{F}_q)$.
- The operation of E-multiplication based on B_N and $\mathbb{F}_q$.

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

[2]The parameters N and q are chosen to meet the desired security level given the best-known attack (which is currently brute force). See Section 7.

TTP Data Generation and Distribution:

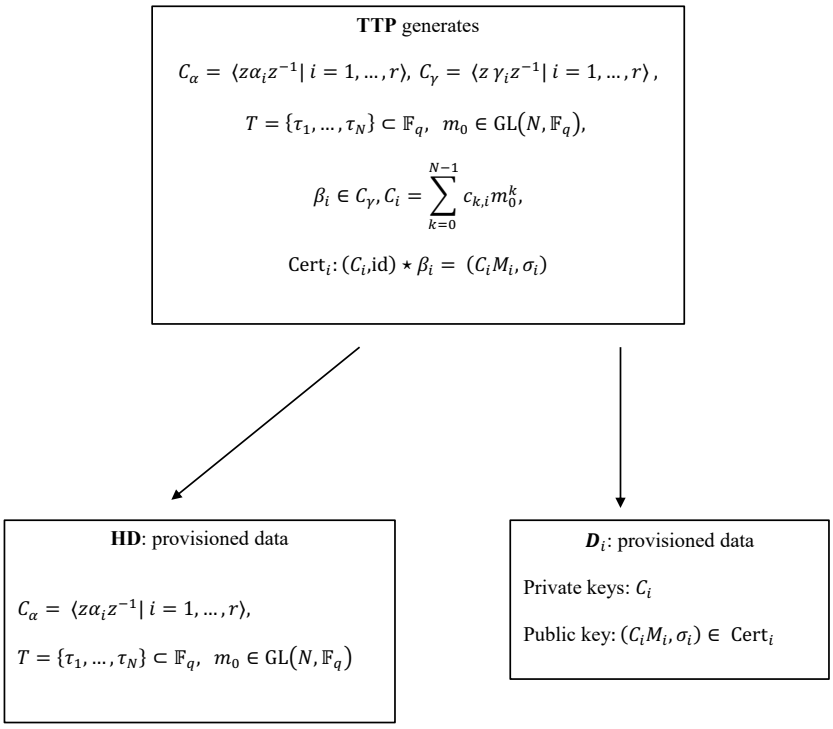

FIGURE 1. Ironwood Data Flows

The TTP begins by generating two sets of braid elements $\{\alpha_1, \dots, \alpha_r\}$, $\{\gamma_1, \dots, \gamma_r\}$ such that $\alpha_i\gamma_j = \gamma_j\alpha_i$. By conjugating these sets of elements by a fixed braid element z, the TTP creates two sets of commuting conjugates $\mathcal{C}_\alpha, \mathcal{C}_\gamma \subset B_N$:[3]

$$\mathcal{C}_\alpha = \{z\alpha_1 z^{-1},\ z\alpha_2 z^{-1},\ \ \dots,\ \ z\alpha_r z^{-1}\},$$
$$\mathcal{C}_\gamma = \{z\gamma_1 z^{-1},\ z\gamma_2 z^{-1},\ \ \dots,\ \ z\gamma_r z^{-1}\}.$$

In addition to the sets $\mathcal{C}_\alpha, \mathcal{C}_\gamma$ conjugates above, the TTP produces a fixed set of T-values:

$$T = \{\tau_1, \tau_2, \dots, \tau_N\} \subset \mathbb{F}_q^\times, \qquad (\tau_i \neq 1).$$

The TTP writes the first set of conjugates $\mathcal{C}_\alpha$ and the set of T-values into the memory of the HD. After being provisioned with $\mathcal{C}_\alpha$ and T the HD functions independently of the TTP, and is capable of producing both its own private and public keys. No further interaction is necessary between the TTP and the HD.

Next, the TTP creates braid words $\beta_i \in B_N$ (for $i = 1, 2, \dots$), which are random products of conjugates from the second set $\mathcal{C}_\gamma$ chosen according to the uniform

[3]The number r of braids in each set is chosen for a time/space/keysize tradeoff, but must satisfy $r \geq 2$. A larger r requires more time to generate the conjugates and more space to store them, but on the other hand requires shorter random words in them to generate keys.

distribution. The TTP then creates colored Burau pairs (β_i, σ_i), where σ_i is the permutation associated to β_i. For each such (β_i, σ_i), the TTP chooses a random non-singular matrix

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

where the $c_{k,i} \in \mathbb{F}_q$ are chosen according to the uniform distribution. The TTP then uses the T-values to perform the E-multiplication

$$\text{Pub}_i := (C_i, \text{Id}) \star (\beta_i, \sigma_i) = (C_i M_i, \sigma_i). \tag{4}$$

In (4), Id denotes the identity permutation and $M_i \in GL(N, \mathbb{F}_q)$. We remark that the one-way nature of E-multiplication makes it impossible for the TTP to choose the public key for D_i prior to specifying the private key for D_i. Thus, the protocol is entirely distinct from any identity-based protocol. Finally, the TTP creates a certificate Cert_i that contains a digitally signed copy of Pub_i and writes Cert_i and C_i into the memory of D_i, the i^{th} device in the network. By using a digital signature to sign the public key of D_i and to generate Cert_i, we eliminate concerns about the man-in-the-middle attack.

After all the data is created, the TTP must securely provision the Home Device and Other Devices with their respective data. Figure 1 summarizes the data created by the TTP, and how it is distributed to the devices.

Upon completion of the TTP distribution, 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 Cert_i, which contains a copy of Pub_i that has been digitally signed by the TTP. Here Pub_i is the public key of D_i and the matrix C_i is the private key of D_i.

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

$$C = \sum_{k=0}^{N-1} c_k m_0^k, \qquad C' = \sum_{k=0}^{N-1} c'_k m_0^k.$$

Here $c_k, c'_k \in \mathbb{F}_q$ are chosen just as the coefficients in the creation of the C_i were.

Step 3: The HD generates an ephemeral permutation σ and two ephemeral braids β, β'; the latter are random words in $\mathcal{C}_\alpha$ as before, and we require that β and β' 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 α_i are purebraids. Alternatively, if none of the α_i are pure, one can simply take a word in the HD's conjugates to a sufficiently high power to make it pure.

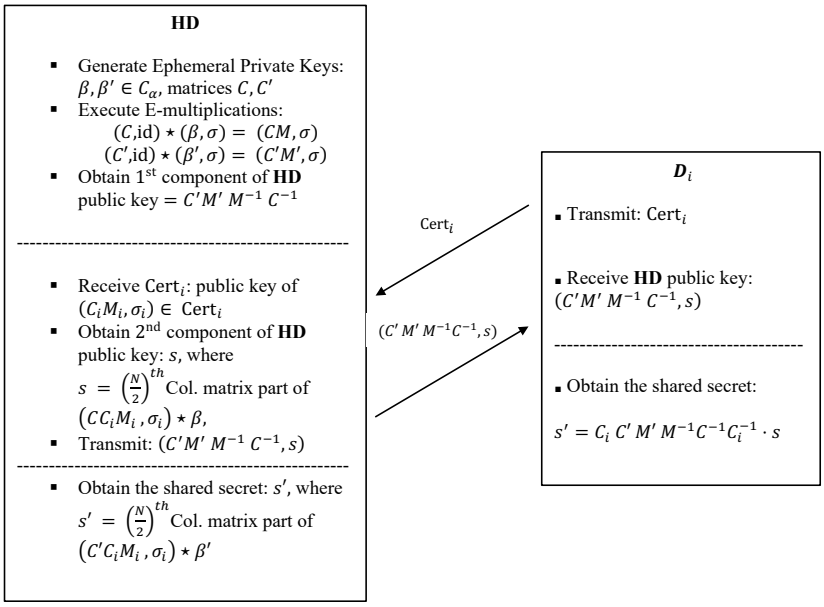

FIGURE 2. Ironwood Protocol Flow

Remark: *After Step 3 the construction of the ephemeral part of the private key of the* HD*, which consists of $C, C', \beta, \beta', \sigma$, is complete. 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:

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

Step 5: The HD has received $\mathrm{Pub}_i = (C_iM_i, \sigma_i)$ 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

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

Step 7: The HD sends D_i the pair

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

Step 8: The device D_i receives the matrix $C'M'M^{-1}C^{-1}$ and the vector s and verifies that at least one half of the entries of s and the matrix are not zero (which can be done in N steps and N^2 steps, respectively). These checks are to prevent a class of invalid public-key attacks. If at least $1/2$ of the entries of either s or the matrix are zero then the protocol halts. Otherwise, the device D_i computes the matrix and vector multiplications:

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

The device D_i can do this since it knows its private key C_i and has received $C'M'M^{-1}C^{-1}$ and s from the HD. Further, if $s = s'$ the protocol should also halt.

Shared Secret: The shared secret is the column vector s', which is now 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.

Figure 2 gives a diagram that compactly summarizes how HD and D_i arrive at the shared secret $s = s'$.

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 $C'M'M^{-1}C^{-1} \neq C_i \left(C'M'M^{-1}C^{-1}\right) C_i^{-1}$ which is equivalent to checking that $s \neq s'$ in step 8. 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. Further, since the protocol halts in step 8 if the vector s has more than $N/2$ entries which are zero, it is not possible for an attacker to act as the HD with an invalid vector s such as $s = 0$.. See Section 6 for further analysis.

It is not at all obvious that the column vectors s, s' produced by the HD and D_i have to be equal. We now provide a proof of this. To begin, the braids β and β' commute with β_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

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

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

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

which shows that the computed secrets agree.

6. **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], [15], [16],[22]) 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 conjugates that 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. Namely, if an attacker can break into one of the devices D_i and obtain its private key, then only the security of D_i is breached; all other devices remain secure 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 D_j with $j \neq i$.

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[21], which studies the security of *Zémor's hash function* [25]. This is a hash function $H\colon \{0,1\}^* \to SL(2, \mathbb{F}_q)$ constructed by fixing two matrices $h_0, h_1 \in SL(2, \mathbb{F}_q)$. Then, if B is the bitstring $b_1b_2\cdots b_n$, one puts

$$H(B) = \prod_{i=1}^{n} h_{b_i}.$$

For example, the bitstring 01101 hashes to the product $h_0h_1h_1h_0h_1 \in SL(2,\mathbb{F}_q)$. Zemor's hash function has not been broken since its inception in 1991. In [21], 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) because 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 is thwarted in Step 8 of the protocol if a rogue HD (i) sends an invalid vector s to D_i, (ii) chooses $\beta, \beta' = 1$ or very short, or (iii) sends a matrix that is mostly 0. In all cases D_i can look at the matrix and ensure there are a sufficient number of non-zero entries; in other words that the matrix is "far" from the identity.

We note that this attack does reduce the minimum possible security level because an attacker needs to only search through q^X possible states (where X is the number of non-zero entries in the s vector). By requiring at least half the entries to be non-zero, we reduce the minimum possible security level by half. If we require more of s to be non-zero, we can minimize the impact at the expense of possibly considering a real transaction to be bogus (because each entry in s has a 1 in q chance of randomly being 0). Moreover, by requiring the HD matrix to be more than half non-zero, it forces the result of a valid D_i computation to mix the results sufficiently such that a reduced-space s vector will still incur the q^X search space, even if an attacker chooses short β, β' and a non-zero matrix.

Further, if 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 (see [6]), then an invalid key attack would not directly reveal any information to a rogue HD.

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 of D_i it may lead to potential leakage. The best approach to protect against an invalid public-key attack against the HD is to have the device D_i's public key signed by a trusted CA/TTP. This allows 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. Recall that this is the method we propose in Step 1 of the Ironwood Protocol.

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 [15]) of the type presented in [10], [11], [16], [22], these attacks completely fail for Ironwood. This is because it is assumed that the two sets of conjugates, $\mathcal{C}_\alpha, \mathcal{C}_\gamma$, 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 $\mathcal{C}_\alpha$ 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 $GL(N, \mathbb{F}_q)$ can be determined as follows. It is well known that there are

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

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

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

for $N, q \geq 8$. For two matrices $X, Y \in GL(N, \mathbb{F}_q)$ 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 $GL(N, \mathbb{F}_q)$ 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 3.815×10^{-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[24] 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[19].

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[14] 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 6.1. (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 $\mathbb{F}_q$. Note that there are q^N polynomials of degree $N-1$ over $\mathbb{F}_q$. So a brute force search for a particular polynomial of degree $N-1$ over $\mathbb{F}_q$ 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 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 β, β' 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 the lower bound

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

An active attacker who attempts to run a weak-key attack can force a reduction in security level. Specifically, we would expect the search-space of s' to be q^X where X is the number of non-zero entres in the s vector sent by the HD to D_i, or more accurately, the number of non-zero entries required by D_i. An attacker who sends an s vector with just under half of the entries 0 would reduce the security level by half. Therefore, to properly defeat this kind of attack requires choices for N and q such that $q^N \geq 2^{2SL}$, or more accurately, $q^X \geq 2^{SL}$ where $X < N$.

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', β, or β'. 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.

7. **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 β and another with β'), 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 $\mathbb{F}_q$. These operations, in turn, gets multiplied by the number of Artin generators in each braid.

As an example, we generated key material using $B_{16}\mathbb{F}_{256}$ for a proposed 2^{64} 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 β and 4302.4 for β'.

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. 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).

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)

TABLE 1. Performance on MSP430, LPC1768 (in Cycles)

Artin Length		**MSP430**	**LPC1768**
$\|\beta\|$	$\|\beta'\|$		
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

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 could not 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 2. Performance on MSP430, LPC1768, CC2650 (in ms)

Artin Length		**MSP430**	**LPC1768**	**CC2650 48Mhz**	
$\|\beta\|$	$\|\beta'\|$	**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.

8. **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.

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] 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.

[15] 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] .

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

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

[22] 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.

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

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

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

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: ianshel@veridify.com
E-mail address: datkins@veridify.com
E-mail address: dgoldfeld@veridify.com
E-mail address: pgunnells@veridify.com