

# A Practical Cryptanalysis of the Algebraic Eraser

Adi Ben-Zvi<sup>1</sup>, Simon R. Blackburn<sup>2(✉)</sup>, and Boaz Tsaban<sup>1</sup>

<sup>1</sup> Department of Mathematics, Bar-Ilan University, Ramat Gan 5290002, Israel

<sup>2</sup> Department of Mathematics, Royal Holloway University of London,  
Egham TW20 0EX, Surrey, UK  
s.blackburn@rhul.ac.uk

**Abstract.** We present a novel cryptanalysis of the Algebraic Eraser primitive. This key agreement scheme, based on techniques from permutation groups, matrix groups and braid groups, is proposed as an underlying technology for ISO/IEC 29167-20, which is intended for authentication of RFID tags. SecureRF, the company owning the trademark Algebraic Eraser, markets it as suitable in general for lightweight environments such as RFID tags and other IoT applications. Our attack is practical on standard hardware: for parameter sizes corresponding to claimed 128-bit security, our implementation recovers the shared key using less than 8 CPU hours, and less than 64 MB of memory.

## 1 Introduction

The Algebraic Eraser<sup>TM</sup> is a key agreement scheme using techniques from non-commutative group theory. It was announced by Anshel, Anshel, Goldfeld and Lemieaux in 2004; the corresponding paper [1] appeared in 2006. The Algebraic Eraser is defined in a very general fashion: various algebraic structures (monoids, groups and actions) need to be specified in order to be suitable for implementation. Anshel *et al.* provide most of this extra information, and name this concrete realisation of the Algebraic Eraser the *Colored Braid Key Agreement Protocol (CBKAP)*. This concrete representation involves a novel blend of finite matrix groups and permutation groups with infinite braid groups. A company, SecureRF, owns the trademark to the Algebraic Eraser, and is marketing this primitive as suitable for low resource environments such as RFID tags and Internet of Things (IoT) applications. The primitive is proposed as an underlying technology for ISO/IEC 29167-20, and work on this standard is taking place in ISO/IEC JTC 1/SC 31/WG 7. The company has also presented the primitive to the Internet Research Task Force's Crypto Forum Research Group (IRTF CFRG), with a view towards standardisation. IoT is a growth area, where current widely-accepted public key techniques struggle to operate due to tight efficiency constraints. It is likely that solutions which are efficient enough for these applications will become widely deployed, and the nature of these applications make system changes after deployment difficult. Thus, it is vital to scrutinise the security of primitives such as the Algebraic Eraser

early in the standardisation process, to ensure only secure primitives underpin standardised protocols.

In a presentation to the NIST Workshop in Lightweight Cryptography in 2015, SecureRF claims a security level of  $2^{128}$  for their preferred parameter sizes, and compares the speed of their system favourably with an implementation of a key agreement protocol based on the NIST recommended [14] elliptic curve K-283. The company reports [3] a speed-up by a factor of 45–150, compared to elliptic curve key agreement at 128-bit security levels. It claims that the computational requirements of the Algebraic Eraser scales linearly with the security parameter, in contrast to the quadratic scaling of elliptic-curve-based key agreement.

**Related Work.** The criteria for choosing some global parameters of the scheme (namely certain subgroups  $C$  and  $D$  of matrices over a finite field, and certain subgroups  $A$  and  $B$  of a certain infinite semidirect product of groups) are not given in [1], and have not been made available by SecureRF. In the absence of this information, it is reasonable to proceed initially with a cryptanalysis on the basis that these parameters are chosen in a generic fashion. All previous cryptanalyses have taken this approach.

Myasnikov and Ushakov [13] provide a heuristic length-based attack on the CBKAP that works for the parameter sizes originally suggested [1]. However, Gunnells [10] reports that this attack quickly becomes unsuccessful as parameter sizes grow; the parameter sizes proposed by SecureRF make this attack impractical<sup>1</sup>. Kalka et al. [11] provide an efficient cryptanalysis of the CBKAP for arbitrary parameter sizes. The attack uses the public key material of Alice and the messages exchanged between Alice and Bob to derive an equivalent to the secret random information generated by Bob, which then compromises the shared key, and so renders the scheme insecure. In particular, the techniques of [11] will succeed when the global parameters are chosen generically.

SecureRF uses proprietary distributions for global parameters, so the cryptanalysis of [11] attack does not imply that the CBKAP as implemented is insecure<sup>2</sup>. Indeed, Goldfeld and Gunnells [9] show that by choosing the subgroup  $C$  carefully one step of the attack of [11] does not recover the information required to proceed, and so this attack does not succeed when parameters are generated in this manner.

**Our Contribution.** There are no previously known attacks on the CBKAP for the proposed parameter sizes, provided the parameters are chosen to resist the attack of [11]. The present paper describes a new attack on the CBKAP that

<sup>1</sup> There is an analogy with the development of RSA here: the size of primes (200 digits) proposed in the original article [15] was made obsolete by improvements in integer factorisation algorithms [4].

<sup>2</sup> The analogy with RSA continues: factorisation of a randomly chosen integer  $n$  is much easier than when  $n$  is a product of two primes of equal size, which is why the latter is used in RSA.

does not assume any structure on the subgroup  $C$ . Thus, a careful choice of the subgroup  $C$  will have no effect on the applicability of our attack, and so the proposed security measure offered by Goldfeld and Gunnells [9] to the attack of [11] is bypassed.

The earlier cryptanalyses of CBKAP ([11, 13]) attempt to recover parts of Alice's or Bob's secret information. The attack presented here recovers the shared key directly from Alice's public key and the messages exchanged between Alice and Bob.

SecureRF have kindly provided us with sets of challenge parameters of the full recommended size, and our implementation is successful in recovering the shared key in all cases. Our (non-optimised) implementation recovers the common key in under 8 h of computation, and thus the security of the system is *much* less than the  $2^{128}$  level claimed for these parameter sizes. The attack scales well with size, so increasing parameter sizes will not provide a solution to the security problem for the CBKAP.

**Conclusion and Recommendation.** Because our attack efficiently recovers the shared key of the CBKAP for recommended parameter sizes, using parameters provided by SecureRF, we believe the results presented here cast serious doubt on the suitability of the Algebraic Eraser for the applications proposed. We recommend that the primitive in its current form should not be used in practice, and that full details of any revised version of the primitive should be made available for public scrutiny in order to ensure a rigorous security analysis.

**Recent Developments.** Since the first version of this paper was posted, there have been two recent developments. Firstly, authors from SecureRF have posted [2] a response to our attack, concentrating in the main on the implications for the related ISO standard and providing some preliminary thoughts on how they might redesign the primitive. Until the details are finalised, it is too soon to draw any conclusions on the security of any redesigned scheme, though there have already been some discussions on Cryptography Stack Exchange [8]. Secondly, Blackburn and Robshaw [6] have posted a paper that cryptanalyses the ISO standard itself, rather than the more general underlying Algebraic Eraser primitive.

**Structure of the Paper.** The remainder of the paper is organised as follows. Sections 2 and 3 establish notation, and describe the CBKAP. We describe a slightly more general protocol than the CBKAP, as our attack naturally generalises to a larger setting. We describe our attack in Sect. 4. In Sect. 5 we describe the results of our implementations and provide a short conclusion.

## 2 Notation

This section establishes notation for the remainder of the paper. We closely follow the notation from [11], which is in turn mainly derived from the notation in [1], though we do introduce some new terms.

Let  $\mathbb{F}$  be a finite field of small order (e.g.,  $|\mathbb{F}| = 256$ ) and let  $n$  be a positive integer (e.g.,  $n = 16$ ). Let  $S_n$  be the symmetric group on the set  $\{1, 2, \dots, n\}$ , and let  $\text{GL}_n(\mathbb{F})$  be the group of invertible  $n \times n$  matrices with entries in  $\mathbb{F}$ .

Let  $M$  be a subgroup of  $\text{GL}_n(\mathbb{F}(t_1, \dots, t_n))$ , where the elements  $t_i$  are algebraically independent commuting indeterminates. Indeed, we assume that the group  $M$  is contained in the subgroup of  $\text{GL}_n(\mathbb{F}(t_1, \dots, t_n))$  of matrices whose determinant can be written as  $at$  for some non-zero element  $a \in \mathbb{F}$  and some, possibly empty, word  $\mathbf{t}$  in the elements  $t_i$  and their inverses. Let  $\overline{M}$  be the subgroup of  $\text{GL}_n(\mathbb{F}(t_1, \dots, t_n))$  generated by permuting the indeterminates of elements of  $M$  in all possible ways.

Fix non-zero elements  $\tau_1, \dots, \tau_n \in \mathbb{F}$ . Define the homomorphism  $\varphi: \overline{M} \rightarrow \text{GL}_n(\mathbb{F})$  to be the evaluation map, computed by replacing each indeterminate  $t_i$  by the corresponding element  $\tau_i$ . Our assumption on the group  $M$  means that  $\varphi$  is well defined.

The group  $S_n$  acts on  $\overline{M}$  by permuting the indeterminates  $t_i$ . Let  $\overline{M} \rtimes S_n$  be the semidirect product of  $\overline{M}$  and  $S_n$  induced by this action. More concretely, if we write  ${}^g a$  for the action of an element  $g \in S_n$  on an element  $a \in \overline{M}$ , then the elements of  $\overline{M} \rtimes S_n$  are pairs  $(a, g)$  with  $a \in \overline{M}$  and  $g \in S_n$ , and group multiplication is given by

$$(a, g)(b, h) = (a{}^g b, gh)$$

for all  $(a, g), (b, h) \in \overline{M} \rtimes S_n$ .

Let  $C$  and  $D$  be subgroups of  $\text{GL}_n(\mathbb{F})$  that *commute elementwise*:  $cd = dc$  for all  $c \in C$  and  $d \in D$ . The CBKAP specifies that  $C$  is a subgroup consisting of all invertible matrices of the form  $\ell_0 + \ell_1 \kappa + \dots + \ell_r \kappa^r$  where  $\kappa$  is a fixed matrix,  $\ell_i \in \mathbb{F}$  and  $r \geq 0$ . So  $C$  is the group of units in the  $\mathbb{F}$ -algebra generated by  $\kappa$ . Moreover, the CBKAP specifies that  $D = C$ . But we do not assume anything about the forms of  $C$  and  $D$  in this paper, other than the fact that they commute.

Let  $\Omega = \text{GL}_n(\mathbb{F}) \times S_n$  and let  $\widehat{S}_n = \overline{M} \rtimes S_n$ . We have two actions on  $\Omega$ . Firstly, there is the right action of the group  $\widehat{S}_n$  on  $\Omega$  via a map

$$*: \Omega \times \widehat{S}_n \rightarrow \Omega,$$

as defined in [1, 11]. So

$$(s, g) * (b, h) = (s\varphi({}^g b), gh)$$

for all  $(s, g) \in \Omega$  and all  $(b, h) \in \widehat{S}_n$ . Secondly, there is a left action of the group  $\text{GL}_n(\mathbb{F})$  on  $\Omega$  via the map

$$\bullet: \text{GL}_n(\mathbb{F}) \times \Omega \rightarrow \Omega$$

given by matrix multiplication:

$$x \bullet (s, g) = (xs, g)$$

for all  $x \in \text{GL}_n(\mathbb{F})$  and all  $(s, g) \in \Omega$ . Note that for all  $x \in \text{GL}_n(\mathbb{F})$ , all  $\omega \in \Omega$  and all  $\widehat{g} \in \widehat{S}_n$  we have that

$$(x \bullet \omega) * \widehat{g} = x \bullet (\omega * \widehat{g}).$$

Also note that the left action is  $\mathbb{F}$ -linear, in the sense that if  $x \in \text{GL}_n(\mathbb{F})$  can be written in the form

$$x = \sum_{i=1}^r \ell_i c_i$$

for some  $c_i \in \text{GL}_n(\mathbb{F})$  and  $\ell_i \in \mathbb{F}$ , then for all  $(s, g) \in \Omega$  we have

$$x \bullet (s, g) = \sum_{i=1}^r \ell_i (c_i \bullet (s, g)).$$

To interpret the right hand side of the equality above: the subset of  $\Omega$  whose second component is a fixed element of  $S_n$  is naturally an  $\mathbb{F}$ -vector space, where addition and scalar multiplication takes place in the first component only.

Finally, let  $A$  and  $B$  be subgroups of  $\widehat{S}_n$  that *\*-commute*: for all  $(a, g) \in A$ ,  $(b, h) \in B$  and  $\omega \in \Omega$ ,

$$(\omega * (a, g)) * (b, h) = (\omega * (b, h)) * (a, g).$$

### 3 The CBKAP Protocol

#### 3.1 Overview

The CBKAP is unusual in that the parties executing it, Alice and Bob, use different parts of the public key in their computations: neither party needs to know all of the public key. The security model assumes that one party’s public key material is known to the adversary: say Alice’s public key material is known, but Bob’s ‘public’ key (which is better thought of as part of his private key material) is not revealed. The adversary, Eve, receives just Alice’s public information, and the messages sent over the insecure channel. Security means that Eve cannot feasibly compute any significant information about  $K$ . The attack in [11] works in this model. The same is true for the attack we describe below.

In a typical proposed application, the protocol might be used to enable a low-power device, such as an RFID tag, to communicate with a central server. Data on an RFID tag is inherently insecure, as is system-wide data. So the above security model is realistic (and conservative) for these application settings.

### 3.2 The Protocol

Public parameters (for Alice) include the parameters  $n, \mathbb{F}, M, \tau_1, \dots, \tau_n, C$  and  $A$ . The groups  $M, C$  and  $A$  are specified by their generating sets. For efficiency reasons, the generators of the group  $A$  are written as words in a certain standard generating set for the group  $\widehat{S}_n$ . We discuss this further in Sect. 5, but see the TTP algorithm in [1] for full information. It is assumed that Eve knows the parameters  $n, \mathbb{F}, M, \tau_1, \dots, \tau_n, C$  and  $A$ . Bob needs to know the groups  $B$  and  $D$ , rather than the groups  $A$  and  $C$ . Eve does not need to know the subgroups  $B$  and  $D$  for our attack to work.

We write  $e$  for the identity element of  $S_n$ . We write  $I_n$  for the identity matrix in  $\text{GL}_n(\mathbb{F})$ , and write  $1 = (I_n, e) \in \Omega$ .

Alice chooses elements  $c \in C$  and  $\widehat{g} = (a, g) \in A$ . She computes the product

$$c \bullet 1 * \widehat{g} = (c\varphi(a), g) \in \Omega$$

and sends it to Bob over an insecure channel.

Bob, who knows the groups  $B$  and  $D$ , chooses elements  $d \in D$  and  $\widehat{h} = (b, h) \in B$ . He computes the product

$$d \bullet 1 * \widehat{h} = (d\varphi(b), h) \in \Omega$$

and sends it to Alice over the insecure channel.

Note that  $cd = dc$  because  $c \in C$  and  $d \in D$ , and the groups  $C$  and  $D$  commute elementwise. Thus,

$$\begin{aligned} d \bullet (c \bullet 1 * \widehat{g}) * \widehat{h} &= (dc) \bullet (1 * \widehat{g}) * \widehat{h} \\ &= (cd) \bullet (1 * \widehat{g}) * \widehat{h} \\ &= (cd) \bullet (1 * \widehat{h}) * \widehat{g} \\ &\quad (\text{as } \widehat{g} \in A, \widehat{h} \in B, \text{ and } A \text{ and } B \text{ * -commute}) \\ &= c \bullet (d \bullet 1 * \widehat{h}) * \widehat{g}. \end{aligned}$$

The common key  $K$  is defined by

$$K = d \bullet (c \bullet 1 * \widehat{g}) * \widehat{h} = c \bullet (d \bullet 1 * \widehat{h}) * \widehat{g}.$$

Alice can compute the key  $K$  using the right hand expression in the equation above; Bob can compute  $K$  by computing the middle expression.

## 4 The Proposed Attack

Eve, the adversary, sees all public information, and also sees the elements  $(p, g) := c \bullet 1 * \widehat{g} \in \Omega$  and  $(q, h) := d \bullet 1 * \widehat{h} \in \Omega$  that are transmitted between Alice and Bob. Eve's goal is to compute the shared key. Rather than attempting to compute Alice's private key material  $c$  and  $\widehat{g}$ , or Bob's private key material  $d$  and  $\widehat{h}$ , our attack will recover the shared key directly.

An overview of our attack is as follows. We first argue that the group  $C$  can be replaced by a ‘linearised’ version of  $C$ : this makes it easier to test membership in  $C$ . We then show that Eve does not need to compute Alice’s or Bob’s secret information in order to derive the shared key: more limited information suffices. (This information is specified in Eqs. (1) and (2) below.) Finally, we show how Eve can compute this information.

For a group  $H$  of  $n \times n$  matrices over a field  $\mathbb{F}$ , we write  $\text{Alg}(H)$  for the  $\mathbb{F}$ -algebra generated by  $H$  [5]. So  $\text{Alg}(H)$  is the set of all  $\mathbb{F}$ -linear combinations of matrices in  $H$ . We write  $\text{Alg}^*(H)$  for the set of all invertible matrices in  $\text{Alg}(H)$ .

The groups  $C$  proposed in the CBKAP satisfy  $C = \text{Alg}^*(C)$ . More generally, we may assume that this is always the case. To see this, first note  $\text{Alg}^*(C)$  and  $D$  commute elementwise since every element of  $\text{Alg}^*(C)$  is a linear combination of elements in  $C$ . Thus,  $C$  may be replaced by  $\text{Alg}^*(C)$  to obtain a valid new instance of the protocol. Moreover, since  $C \subseteq \text{Alg}^*(C)$  the new instance of the protocol is more general than the original protocol: Alice can choose her matrix  $c$  from the larger group  $\text{Alg}^*(C)$ . So if we successfully recover the common key in *every* new instance of the protocol, we can successfully recover the common key in the original instance.

Thus, from now on, we assume that  $C = \text{Alg}^*(C)$ . Let  $\kappa_1, \kappa_2, \dots, \kappa_r \in C$  be a basis for  $\text{Alg}(C)$ . Such a basis is not difficult to compute, using standard techniques. Our assumption means that any invertible linear combination of the matrices  $\kappa_i$  lies in  $C$ .

Let  $P \trianglelefteq A$  be the *pure subgroup of  $A$* , defined by

$$P = \{ (\alpha, g) \in A : g = e \}.$$

Then  $\varphi(P)$  is a subgroup of  $\text{GL}_n(\mathbb{F})$ . Consider the subgroup  $\text{Alg}^*(\varphi(P))$  of  $\text{GL}_n(\mathbb{F})$ . Concretely, an element  $\alpha' \in \text{Alg}^*(\varphi(P))$  is an invertible matrix of the form

$$\alpha' = \sum_{i=1}^k \ell_i \varphi(\alpha_i)$$

where  $k \geq 0$ ,  $\ell_i \in \mathbb{F}$  and  $(\alpha_i, e) \in P$ .

Suppose that Eve finds elements  $\tilde{c} \in C$ ,  $\alpha' \in \text{Alg}^*(\varphi(P))$  and  $(\tilde{a}, g) \in \widehat{S}_n$  such that

$$(p, g) = \tilde{c} \bullet (\alpha', e) * (\tilde{a}, g). \tag{1}$$

Moreover, suppose that Eve can find an element  $(\alpha_i, e) \in P$  and  $\ell_i \in \mathbb{F}$  such that

$$\sum_{i=1}^k \ell_i \varphi(\alpha_i) = \alpha'. \tag{2}$$

Then Eve can compute the common key, as follows. Firstly, she computes the matrix

$$\beta' = \sum_{i=1}^k \ell_i \varphi({}^h \alpha_i).$$

This computation is possible for Eve, since  $h$  is part of the message  $(q, h) = (d\varphi(b), h) \in \Omega$  transmitted from Bob to Alice. Now,  $(\alpha_i, e) \in P \leq A$ , and so  $(\alpha_i, e)$   $*$ -commutes with all elements in  $B$ . Thus,

$$(q\varphi({}^h\alpha_i), h) = d \bullet 1 * (b, h) * (\alpha_i, e) = d \bullet 1 * (\alpha_i, e) * (b, h).$$

Eve then computes  $\tilde{c} \bullet (q\beta', h) * (\tilde{a}, g)$ . We claim that this is equal to the common key  $K$ . To see this, first note that

$$\begin{aligned} (q\beta', h) &= \sum_{i=1}^k \ell_i(q\varphi({}^h\alpha_i), h) \\ &= \sum_{i=1}^k \ell_i(d \bullet 1 * (\alpha_i, e) * (b, h)) \\ &= \sum_{i=1}^k \ell_i(d\varphi(\alpha_i)\varphi(b), h) \\ &= (d \sum_{i=1}^k \ell_i\varphi(\alpha_i)\varphi(b), h) \\ &= (d\alpha'\varphi(b), h) \\ &= d \bullet (\alpha', e) * (b, h). \end{aligned}$$

Hence

$$\begin{aligned} \tilde{c} \bullet (q\beta', h) * (\tilde{a}, g) &= \tilde{c} \bullet d \bullet (\alpha', e) * (b, h) * (\tilde{a}, g) \\ &= d \bullet \tilde{c} \bullet (\alpha', e) * (b, h) * (\tilde{a}, g) \\ &\quad (\text{since } \tilde{c} \in C \text{ and } d \text{ centralises } C) \\ &= d \bullet \tilde{c} \bullet (\alpha', e) * (\tilde{a}, g) * (b, h) \\ &\quad (\text{as } (\tilde{a}, g) \in A \text{ and } (b, h) \in B \text{ } * \text{-commute}) \\ &= d \bullet (p, g) * \hat{h} \\ &= d \bullet (c \bullet 1 * \hat{g}) * \hat{h} \\ &= K. \end{aligned}$$

So it suffices to show that Eve can find elements  $\alpha_i, \ell_i, \tilde{a}, \tilde{c}$  and  $\alpha'$  so that Eqs. (1) and (2) are satisfied.

*Precomputation stage:* Find the  $\alpha_i$ . Eve computes a collection of elements  $(\alpha_i, e)$  such that the matrices  $\varphi(\alpha_i)$  form a basis of  $\text{Alg}(\varphi(P))$ . Once this is done, any  $\alpha \in \text{Alg}^*(\varphi(P))$  can easily be written in the form (2). Eve does not need to know the messages  $(p, g)$  and  $(q, h)$  in this stage, so this stage can be carried out as a precomputation. Eve proceeds as follows.

Eve generates, as in [11], short products  $(a', g')$  of generators of  $A$  such that the order  $r$  of the permutation  $g'$  is small ( $n$  or less), and computes



$\alpha_1 = (a', g')^r = (a'', e)$ . She repeats this procedure to generate  $\alpha_2, \alpha_3 \dots$  (Eve may also take products of some of the previously generated elements  $(\alpha_1, e)$ ,  $(\alpha_2, e), \dots, (\alpha_{i-1}, e)$  to define  $(\alpha_i, e)$ .) Eve stops when the dimension of the  $\mathbb{F}$ -linear span of the matrices  $\varphi(\alpha_i)$  stops growing, and fixes a linearly independent subset of these matrices.

At the end of this process (relabelling after throwing linearly dependent elements  $\varphi(\alpha_i)$  away), Eve has  $\alpha_1, \alpha_2, \dots, \alpha_r$  such that  $\varphi(\alpha_1), \varphi(\alpha_2), \dots, \varphi(\alpha_r)$  are a basis for a subspace  $V$  of  $\text{Alg}(\varphi(P))$ . Indeed, we expect (with high probability) that  $V = \text{Alg}(\varphi(P))$ . We assume that this is true from now on.

*Stage 1: Find  $\tilde{a}$ .* Find a product of generators in  $A$  whose second component is equal to  $g$ , using the method in [11]. Let  $(\tilde{a}, g)$  be this product. Define  $\gamma \in \text{GL}_n(\mathbb{F})$  by

$$(\gamma, e) = (p, g) * (\tilde{a}, g)^{-1}.$$

*Stage 2: Find  $\tilde{c}$ .* Recall that Eve knows  $\kappa_1, \kappa_2, \kappa_3, \dots, \kappa_r \in C$  that form a basis of  $\text{Alg}(C)$ . She finds (see below) field elements  $x_1, x_2, \dots, x_r \in \mathbb{F}$  such that

$$\gamma^{-1}(x_1\kappa_1 + x_2\kappa_2 + \dots + x_r\kappa_r) \in V, \text{ and} \tag{3}$$

$$x_1\kappa_1 + x_2\kappa_2 + \dots + x_r\kappa_r \text{ is invertible.} \tag{4}$$

Set  $\tilde{c} = x_1\kappa_1 + x_2\kappa_2 + \dots + x_r\kappa_r$ . Since  $\tilde{c}$  is an invertible element of  $\text{Alg}(C)$ , we see that  $\tilde{c} \in C$ .

To find a solution to Eqs. (3) and (4), Eve randomly generates solutions  $x_i$  that satisfy (3), which is easy, as the conditions are linear. She stops when (4) is also satisfied. We claim that the proportion of solutions to (3) that satisfy (4) is bounded below by  $1 - n/|\mathbb{F}|$ , which is a non-trivial proportion for the parameters that are proposed. The claim follows by applying the Invertibility Lemma [16, Lemma 9], which states that the proportion of invertible matrices in any  $\mathbb{F}$ -subspace of matrices over  $\mathbb{F}$  is at least  $1 - (n/|\mathbb{F}|)$ , provided that the subspace contains at least one invertible matrix. We note that the elements of the form  $x_1\kappa_1 + x_2\kappa_2 + \dots + x_r\kappa_r$  that satisfy (3) are a subspace of matrices. So it remains to show that there exists an invertible element of this form. But let  $x_1, x_2, \dots, x_r \in \mathbb{F}$  be such that  $x_1\kappa_1 + x_2\kappa_2 + \dots + x_r\kappa_r = c$ . The elements  $x_i$  exist since  $c \in C \subseteq \text{Alg}(C)$ . Clearly  $\tilde{c} = c$  is invertible. Moreover (3) holds, because we may show that  $\gamma^{-1}c \in \varphi(P) \subseteq V$  as follows. Firstly,

$$(\gamma, e) = (p, g) * (\tilde{a}, g)^{-1} = (c\varphi(a), g) * (g^{-1}(\tilde{a}^{-1}), g^{-1}) = (c\varphi(a)\varphi(\tilde{a}^{-1}), e),$$

so  $\gamma = c\varphi(a)\varphi(\tilde{a}^{-1})$  and therefore  $\gamma^{-1}c = \varphi(\tilde{a})\varphi(a)^{-1}$ . And secondly, we see that  $\varphi(\tilde{a})\varphi(a)^{-1} = \varphi(\tilde{a}a^{-1}) \in \varphi(P)$ , since

$$(\tilde{a}, g)(a, g)^{-1} = (\tilde{a}, g)(g^{-1}a^{-1}, g^{-1}) = (\tilde{a}a^{-1}, e)$$

and  $(\tilde{a}, g), (a, g) \in A$ .

*Stage 3: The remaining parameters.* Eve sets  $\alpha' = \tilde{c}^{-1}\gamma$ . Since  $(\alpha')^{-1} \in V$ , we see that  $\alpha'$  (being a power of  $(\alpha')^{-1}$ ) also lies in  $V$ . So Eve can easily calculate coefficients  $\ell_i$  such that

$$\sum_{i=1}^k \ell_i \varphi(\alpha_i) = \alpha'.$$

Hence, Eq. (2) holds. We may also verify that Eq. (1) holds:

$$\begin{aligned} \tilde{c} \bullet (\alpha', e) * (\tilde{a}, g) &= \tilde{c} \bullet (\tilde{c}^{-1}\gamma, e) * (\tilde{a}, g) = (\gamma, e) * (\tilde{a}, g) \\ &= ((p, g) * (\tilde{a}, g)^{-1}) * (\tilde{a}, g) = (p, g). \end{aligned}$$

## 5 Experiments and Conclusion

We have implemented our attack in Magma [7], running on one 2 GHz core of a multi-core server. We used 5 sets of actual challenge parameters kindly provided by SecureRF. These parameters all used the values  $|\mathbb{F}| = 256$  and  $n = 16$ . The subgroup  $A$  is specified by a generating set; each generator for  $A$  is given as a word of length approximately 650 (notice the large parameter setting!) in the generating set  $\mathcal{X} = \{ (x_i(t), s_i) : 1 \leq i \leq n - 1 \}$  for  $\overline{M} \times S_n$  defined in [1]. In all 5 cases, our attack terminated successfully, producing the exact shared key. Our attack used less than 64 MB of memory, and terminated in less than 8 h. We would like to emphasise that our code is far from being optimised; we estimate an improvement in CPU time by a significant factor in an optimised version of the attack.

Let  $B_n$  be a braid group on  $n$  strands, and let  $\sigma_1, \sigma_2, \dots, \sigma_{n-1}$  be the Artin generators for  $B_n$ . (See, for example, [12] for an introduction to braid groups.) There is a homomorphism  $\psi: B_n \rightarrow \overline{M} \times S_n$  such that  $\psi(\sigma_i) = (x_i, s_i)$  for  $1 \leq i \leq n - 1$ , which gives rise to the coloured Burau representation. Thus we could (and did) use standard routines for computing with braids in  $B_n$ , rather than dealing with words in  $\mathcal{X}$  directly.

The most computationally intensive part of the attack is the computing of  $\varphi(a)$  where  $(a, g) \in \overline{M} \times S_n$  is given as a word in the generators of  $A$ . The long length of the generators in  $A$  as words in  $\mathcal{X}$  is the cause of difficulty here; we were computing with words of length approximately 20,000 in Stage 1 of our attack.

To decide when the precomputation stage should terminate, we use the criterion that the  $\mathbb{F}$ -dimension of the algebra generated by the matrices  $\varphi(\alpha_i)$  should not grow when 4 generators  $(\alpha_i, e)$  in a row are considered.

Not surprisingly, this attack is highly parallelisable. We did not exploit this fact since for the actual parameters a single CPU core sufficed.

It remains open how to immunise the Algebraic Eraser against the presented cryptanalysis. The only hope seems to be to make the problem of expressing a permutation as a short product of given permutations difficult, by working with very carefully chosen distributions. However, for the intended applications, the computational constraints necessitate small values of  $n$ . In this case, Schreier–Sims methods solve this problem efficiently, no matter how the permutations are used. See the discussion around [11, Table 4].

**Acknowledgement.** The authors would like to thank Arkadius Kalka for providing code from the earlier attack [11] on the Algebraic Eraser, and for explaining how to use this code. The authors would also like to thank Martin Albrecht for various excellent editorial suggestions.

## References

1. Anshel, I., Anshel, M., Goldfeld, D., Lemieux, S.: Key agreement, the algebraic eraser<sup>TM</sup> and lightweight cryptography. *Contemp. Math.* **418**, 1–34 (2006)
2. Anshel, I., Atkins, D., Goldfeld, D., Gunnells, P.: Defeating the Ben-Zvi, Blackburn, and Tsaban attack on the algebraic eraser. *IACR ePrint* 2016/044
3. Atkins, D., Gunnells, P.: Algebraic Eraser<sup>TM</sup>: a lightweight, efficient asymmetric key agreement protocol for use in no-power, low-power, and IoT devices. In: *NIST Lightweight Cryptography Workshop*, 20 July 2015. [http://www.nist.gov/itl/csd/ct/lwc\\_workshop2015.cfm](http://www.nist.gov/itl/csd/ct/lwc_workshop2015.cfm)
4. Bahr, F., Boehm, M., Franke, J., Kleinjung, T.: RSA200. <http://www.crypto-world.com/announcements/rsa200.txt>
5. Ben-Zvi, A., Kalka, A., Tsaban, B.: Cryptanalysis via algebraic spans. *IACR ePrint* 2014/041
6. Blackburn, S.R., Robshaw, M.J.B.: On the security of the algebraic eraser tag authentication protocol. In: Manulis, M., Sadeghi, A.-R., Schneider, S. (eds.) *ACNS 2016*. LNCS, vol. 9696, pp. 3–17. Springer, Heidelberg (2016). doi:10.1007/978-3-319-39555-5\_1
7. Bosma, W., Cannon, J., Playoust, C.: The Magma algebra system. I. The user language. *J. Symb.Comput.* **24**, 235–265 (1997)
8. Cryptography Stack Exchange: Status of Algebraic Eraser Key Exchange? <http://crypto.stackexchange.com/questions/30644/status-of-algebraic-eraser-key-exchange>. Accessed 5 Feb 2016
9. Goldfeld, D., Gunnells, P.: Defeating the Kalka-Teicher-Tsaban linear algebra attack on the algebraic eraser. [arXiv:1202.0598](https://arxiv.org/abs/1202.0598), February 2012
10. Gunnells, P.: On the cryptanalysis of the generalized simultaneous conjugacy search problem and the security of the algebraic eraser. [arXiv:1105.1141](https://arxiv.org/abs/1105.1141), May 2011
11. Kalka, A., Teicher, M., Tsaban, B.: Short expressions of permutations as products and cryptanalysis of the algebraic eraser. *Adv. Appl. Math.* **49**, 57–76 (2012)
12. Kassel, C., Turaev, V.: *Braid Groups*. Graduate Texts in Mathematics, vol. 247. Springer, New York (2008)
13. Myasnikov, A., Ushakov, A.: Cryptanalysis of the Anshel-Anshel-Goldfeld-Lemieux key agreement protocol. *Groups Complex. Cryptol.* **1**, 63–75 (2009)
14. National Institute of Standards and Technology: Digital Signature Standard (DSS). Federal Information Processing Standards Publication FIPS PUB **186–4**, July 2013. <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf>
15. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* **21**, 120–126 (1978)
16. Tsaban, B.: Polynomial-time solutions of computational problems in noncommutative algebraic cryptography. *J. Cryptol.* **28**, 601–622 (2015)