On the Security of the Algebraic Eraser Tag Authentication Protocol

Simon R. Blackburn[1] and M.J.B. Robshaw[2(✉)]

[1] Information Security Group, Royal Holloway University of London, Egham TW20 0EX, UK
[2] Impinj, 400 Fairview Ave. N., Suite 1200, Seattle, WA 98109, USA
matt.robshaw@impinj.com

Abstract. The Algebraic Eraser has been gaining prominence as SecureRF, the company commercializing the algorithm, increases its marketing reach. The scheme is claimed to be well-suited to IoT applications but a lack of detail in available documentation has hampered peer-review. Recently more details of the system have emerged after a tag authentication protocol built using the Algebraic Eraser was proposed for standardization in ISO/IEC SC31 and SecureRF provided an open public description of the protocol. In this paper we describe a range of attacks on this protocol that include very efficient and practical tag impersonation as well as partial, and total, tag secret key recovery. Most of these results have been practically verified, they contrast with the 80-bit security that is claimed for the protocol, and they emphasize the importance of independent public review for any cryptographic proposal.

Keywords: Algebraic Eraser · Cryptanalysis · Tag authentication · IoT

1 Introduction

Extending security features to RAIN RFID tags[1] and other severely constrained devices in the *Internet of Things* is not easy. However the different pieces of the deployment puzzle are falling into place. Over-the-air (OTA) commands supporting security features have now been standardized [11] and both tag and reader manufacturers can build to these specifications knowing that interoperability will follow. The OTA commands themselves are crypto-agnostic so parallel work on a range of cryptographic interfaces, so-called *cryptographic suites*, is ongoing within ISO/IEC SC31. These cryptographic suites provide the detailed specifications that allow algorithms such as the AES [14,23], PRESENT-80 [8,15], and Grain-128a [1,16] to be used on even the most basic of RFID devices.

[1] Following the creation of the RAIN Industry Alliance, UHF RFID tags are increasingly branded as RAIN RFID tags. These RFID tags operate at 860–960 MHz and are far more constrained than the HF RFID tags that are familiar from public transport and NFC applications.

M. Manulis et al. (Eds.): ACNS 2016, LNCS 9696, pp. 3–17, 2016.
DOI: 10.1007/978-3-319-39555-5_1

For symmetric cryptography a range of lighter alternatives to the Advanced Encryption Standard (AES) [23] have received a high level of cryptanalytic attention over several years. While the AES will always be an important implementation option, some of these alternative algorithms may be appropriate for certain use-cases. To those not in the field the cost and performance advantages provided by these new algorithms might appear slight. But the requirements of the RAIN RFID market are such that even a minor degradation in the read range or a small percentage increase in silicon price can eliminate the business case for adding security to many use-cases.

Turning to asymmetric cryptography there are several work items in ISO/IEC 29167 that describe public-key solutions. Parts 29167-12 [17] and 29167-16 [18] describe tag authentication based on elliptic curve cryptography, though they carry significant implementation challenges for RAIN RFID. 29167-17 [19] provides another elliptic-curve tag authentication solution with the additional property that compact pre-computed coupons can be used to provide implementation advantages. In 29167-20 [20], however, we encounter an alternative to elliptic curves: 29167-20 proposes a method for asymmetric tag authentication that is based on *braid groups*. This proposal is based on the *Algebraic Eraser* (AE) key agreement protocol [3,25]. SecureRF, the company commercializing (and owning the trademark to) the Algebraic Eraser, claims significant implementation advantages for the Algebraic Eraser over solutions that use elliptic curves. In particular the Algebraic Eraser is claimed to be well-suited to deployments as part of the Internet of Things.

Note. The Algebraic Eraser has been proposed for use in many environments. However the commentary and descriptions in this paper will use the typical RFID setting of an Interrogator (or reader) interacting with a Tag. This provides the closest match with the terms used in the protocol [25].

Related Work

Until recently, crucial details about the Algebraic Eraser and any associated cryptographic protocol were not available. This made independent security analysis and performance evaluation difficult. (See [12,13,21,22,24] for what little exists in the published literature.) However, in October 2015 SecureRF provided a detailed public description of the Algebraic Eraser tag authentication protocol [9,25]. This means that the protocol can now be publicly reviewed and discussed. The published description includes a specific set of system parameters, a set of test vectors, and a description of the tag authentication protocol. However SecureRF do not disclose how the system parameters were generated, an aspect of the technology that is known to be of crucial importance. Indeed, some of the attacks in this paper are able to exploit structure in the system parameters that have been proposed for standardization.

While general documentation [3] describes the Algebraic Eraser in terms of braid groups, company presentations [4,6] distance the technology from previous cryptographic proposals that use braid groups. Instead the security of the Algebraic Eraser is said to depend on a problem called the *simultaneous conjugacy*

separation search problem [4] and sample parameter sizes have been published for different security levels. In [25] the parameters are claimed to correspond to an 80-bit security level, though a precise security model is not provided. Most likely the intention is that the work effort to recover a private key from the corresponding public key should be roughly equivalent to 2^{80} operations.

The tag authentication protocol in [25] is based upon a Diffie–Hellman-like key agreement scheme. Very recently Ben-Zvi, Blackburn, and Tsaban [7] presented an innovative cryptanalysis of the underlying key agreement protocol.[2] Using only information that is exchanged over the air, and avoiding the hard problem upon which the security of the Algebraic Eraser key agreement protocol is claimed to be based, Ben-Zvi *et al.* provide a method for deriving the shared secret key. Using non-optimized implementations they successfully recovered—in under eight hours—the shared secrets generated using the Algebraic Eraser key agreement protocol with parameters provided by SecureRF that were intended to provide 128-bit security [6].

Since then Anshel, Atkins, Goldfeld and Gunnells (researchers associated with SecureRF) have posted a technical response [2] to the Ben-Zvi–Blackburn–Tsaban (BBT) attack. This is not the place to comment on that document, except to highlight one feature that is relevant for our work here.

In [2] Anshel *et al.* consider the implications of the BBT attack and state that the attack would not apply to one of two profiles proposed for standardization [20]. Section 4.2 of Anshel *et al.* [2] reveals that the profile claimed to be secure is one where "... an attacker never has access to one of the public keys ..." [2]. However the idea that it is reasonable for the security of a public key scheme to depend on the public key being hidden is very strange. While it is true Tag public keys could be delivered to interrogators out-of-band, the security of the scheme should not depend on the Interrogator keeping those keys secret. Indeed, if we trust an Interrogator not to reveal the Tag public key then we can trust the Interrogator with a symmetric key and there would be no need to use the Algebraic Eraser at all! So while two of the five attacks described in this paper use the Tag public key for the required calculations, we see no limitation in assuming that the tag public key is, as the name implies, public.

Finally, we should point out a recent posting of Atkins and Goldfeld [5] that suggests modifications to the tag authentication protocol in the light of the results of this paper.

Our Contribution

In this work we derive a range of new and very efficient attacks on the tag authentication protocol [25]. We side-step the bulk of the mathematical machinery behind the Algebraic Eraser, but observe some curious features of the Algebraic Eraser that cause significant failures in this protocol. In particular we provide the following attacks against the variant that is currently proposed for standardization:

[2] The results in this paper are entirely independent of the work in Ben-Zvi *et al* [7].

1. Tag impersonation of a target tag with success probability $\approx 2^{-7}$ after 273 queries against the target tag and storage $\approx 2^{16}$ bits.
2. Tag impersonation of a target tag with 100 % success rate after $\approx 2^{15}$ queries against the target tag and using $\approx 2^{23}$ bits of storage.
3. Full recovery of a tag private key matrix (see Sect. 3.3) with negligible work after running the tag authentication protocol 33 times against the target tag.
4. Tag impersonation of a target tag with 100 % success rate, using Attack 3 and a small pre-computed look-up table of around 128 64-bit words. The on-line work in the attack is negligible while the off-line pre-computation for current parameter sizes is also negligible. This attack uses (a non-heuristic part of) an attack due to Kalka, Teicher and Tsaban [21] together with a novel application of a certain permutation group algorithm.
5. Complete tag private key (or equivalent key) recovery—recovering both the tag private matrix and the secret tag conjugate set (see Sect. 3.3)—building on Attack 3 and requiring a work effort of 2^{49} operations and storage $\approx 2^{48}$ 64-bit words for one of the parameter choices proposed for standardization that is claimed to provide 80-bit security.

Our attacks avoid using any heuristic methods, and apply for all parameter sets of the size proposed in the standard (not just the specific given parameters). These failures in the tag authentication protocol severely undermine claims for an 80-bit security level. We conclude that the protocol is unsuitable for both deployment and standardization in its current form.

Our paper is structured as follows. In Sect. 2 we provide an overview of the Algebraic Eraser tag authentication protocol with the mathematical formalities following in Sect. 3. The attacks are described in Sects. 4, 5, 6, and 7 respectively and we close the paper with our conclusions.

2 Algebraic Eraser and Tag Authentication

The Algebraic Eraser does not use familiar mathematics and a description can be, at first sight, somewhat complicated. However, for our attacks we will only need the basic tools that we provide in Sect. 3. For a more complete view the reader is referred to both the general description of the Algebraic Eraser [3] and the specific protocol details in [25].

As mentioned in the Introduction, at the core of the Algebraic Eraser is a key agreement protocol. Using the familiar protocol flow that dates back to Diffie–Hellman [10], an Interrogator and Tag exchange public keys. Then, by each applying their own secret component to the other public key, both Interrogator and Tag can arrive at a shared common secret key value. To turn this key agreement protocol into a tag authentication protocol, the Interrogator specifies a portion of the shared secret that should be returned by the Tag. The correctness of this response can be verified by the Interrogator. This is illustrated in Table 1 and described more technically in Sect. 3.4.

Table 1. An outline of the Algebraic Eraser tag authentication scheme [25]. The underlying key agreement protocol is used to derive a shared secret. The Interrogator instructs the Tag, using byte index s and bit-length l, to extract an authentication token t of length l from this shared secret.

Interrogator secret key $\texttt{Int}_{\text{priv}}$ public key $\texttt{Int}_{\text{pub}}$		**Tag** secret key $\texttt{Tag}_{\text{priv}}$ public key $\texttt{Tag}_{\text{pub}}$
Request Tag public key/ certificate	$\xrightarrow{start}$	
	$\xleftarrow{\texttt{Tag}_{\text{pub}}}$	Send Tag public key
Send Interrogator public key, index, and token bit length	$\xrightarrow{\texttt{Int}_{\text{pub}}, s, l}$	
Compute secret using $\texttt{Int}_{\text{priv}}$		Compute secret using $\texttt{Tag}_{\text{priv}}$
Check correctness of t	$\xleftarrow{t}$	Using index s and length l extract and return token t

We will refer to the portion of the secret key returned by the Tag as an authentication token t. In [25] the Interrogator indicates to the Tag how to construct t by sending a starting index s and length l during the message exchange between Interrogator and Tag. The protocol description neither specifies nor gives guidelines on s and l. Clearly a fake tag will always be able to fool an Interrogator with probability 2^{-l} but the field specifying the length l in [25] is eight bits long so we have $0 \leq l \leq 255$. This certainly covers all the natural choices. Note that generating an authentication token t by revealing parts of a shared secret means that the Interrogator will need to generate and use different public keys at each tag authentication. While this is alluded to in Section B.1.2 of [25] it is unclear whether the ensuing performance penalty in storage and transaction time is always reflected in published performance figures.

3 Some Technical Details

This section reviews some of the technical details of the protocol. We describe only as much of the detail as we need to describe our attacks.

3.1 System Parameters

The protocol specifies some system parameters, the *key space*, as follows.

Let N be a small positive integer; [25] mandates that $N = 10$. Let $B = \{b_1, b_2, \ldots, b_{N-1}\}$ be an alphabet of size $N-1$ (the b_i are known as *Artin generators*). Let F be the set of all formal strings in the disjoint union $B \cup B^{-1}$. So, for example, $b_2 b_1^{-1} b_1 b_4 b_2^{-1}$ is a length 5 element of F.

The *set of Tag conjugates* is a set $C = \{c_1, c_2, \ldots, c_{32}\}$ of size 32, where each $c_i \in F$. The *set of Interrogator conjugates* is a set $D = \{d_1, d_2, \ldots, d_{32}\}$ of size 32, where each $d_i \in F$. While C and D are specified in [25] SecureRF does not describe how they have been generated. In fact, in Sect. 4 we will exploit an important structural property of the sets C and D that have been proposed for standardization. Here, however, we restrict ourselves to noting that each of C and D require around 90 Kbits to store and that while a tag might not need to store C the Interrogator needs D to generate ephemeral keys.

We write $\mathrm{Sym}(N)$ for the set of all permutations of N objects $\{1, 2, \ldots, N\}$. Let $s_i = (i, i+1) \in \mathrm{Sym}(N)$ be the permutation that swaps i and $i+1$ and leaves the remaining elements fixed. Let

$$w = b_{i_1}^{\epsilon_1} b_{i_2}^{\epsilon_2} \cdots b_{i_r}^{\epsilon_r}$$

be an element in F of length r, where $i_j \in \{1, 2, \ldots, N-1\}$ and $\epsilon_j \in \{-1, 1\}$. The *permutation* $\pi(w) \in \mathrm{Sym}(N)$ *corresponding to* $w \in F$ is the permutation

$$\pi(w) = s_{i_1}^{\epsilon_1} s_{i_2}^{\epsilon_2} \cdots s_{i_r}^{\epsilon_r} = s_{i_1} s_{i_2} \cdots s_{i_r}$$

where product means composition of permutations.

Finally, the protocol [25] specifies using arithmetic in the finite field $\mathbb{F}_{256}$ and defines a specific sequence of $N = 10$ non-zero elements in $\mathbb{F}_{256}$, called *T-values*, and a specific $N \times N$ matrix M_* with entries in $\mathbb{F}_{256}$ called a *seed matrix*. This choice of parameter sizes is denoted B10F256 and, according to Section B.3, is intended to provide 80-bit security.

Another set of parameters, denoted B16F256, has been independently provided by SecureRF to the first author. The same underlying field is used for both parameter sets but the matrices, the set of T-values, and the permutations are defined for $N = 16$ rather than $N = 10$. The parameters B16F256 are intended to provide 128-bit security.

3.2 E-Multiplication

E-multiplication is the public key operation, analogous to finite field exponentiation in Diffie–Hellmann, that lies at the heart of the Algebraic Eraser. It takes two parameters as input. The first parameter is a pair (M, σ) where M is an $N \times N$ matrix over $\mathbb{F}_{256}$ and $\sigma \in \mathrm{Sym}(N)$ is a permutation. The second parameter is a string $w \in F$. The output is a pair (M', σ') where M' is an $N \times N$ matrix over $\mathbb{F}_{256}$ and $\sigma' \in \mathrm{Sym}(N)$. We write

$$(M, \sigma) * w = (M', \sigma').$$

The permutation σ' is easy to define: $\sigma' = \sigma\, \pi(w)$. The matrix M' is computed by first finding a certain $N \times N$ matrix $\phi(\sigma, w)$ with entries in $\mathbb{F}_{256}$, and then

setting $M' = M\phi(\sigma, w)$. We do not specify the details of how $\phi(\sigma, w)$ is defined, but just give the following details. To compute $\phi(\sigma, w)$, we replace the symbols b_i and b_i^{-1} in w by certain fixed matrices and their inverses. These matrices have entries in a polynomial ring in N variables, and the last row of all these matrices is all zero apart from the final entry which is 1. We multiply our matrices together (obtaining a matrix whose last row is all zero apart from the final entry which is 1). We evaluate each entry of this product (which is a ratio of two polynomials in N variables) by replacing each variable by one of the T-values to form the matrix $\phi(\sigma, w)$ with entries in $\mathbb{F}_{256}$. We use σ to decide which T-value replaces each variable in this process.

We note four properties that follow from the way E-multiplication is defined:

1. If w is the concatenation of strings w' and w'' then
$$(M, \sigma) * w = ((M, \sigma) * w') * w''. \tag{1}$$
In fact E-multiplication has other nice properties related to the fact that E-multiplication is derived from the action of a braid group. However we do not need these properties here.
2. The matrix $\phi(\sigma, w)$ only depends on σ and w (and on the T-values, which are fixed).
3. The entries of the last row of $\phi(\sigma, w)$ are all zero, except the final entry which is 1.
4. The following linearity property follows from our partial description of E-multiplication:
$$\begin{array}{c}\text{If } (M_1, \sigma) * w = (M_1', \sigma') \text{ and } (M_2, \sigma) * w = (M_2', \sigma') \\ \text{then } (a_1 M_1 \oplus a_2 M_2, \sigma) * w = (a_1 M_1' \oplus a_2 M_2', \sigma')\end{array} \tag{2}$$
for any $a_1, a_2 \in \mathbb{F}_{256}$.

3.3 Private and Public Keys

The Tag private key has two components.

1. The first component is an $N \times N$ matrix K_T over $\mathbb{F}_{256}$ that is generated from the seed matrix M_*. During the key generation process a random degree 9 polynomial $p(x)$ over $\mathbb{F}_{256}$ is selected and we set $K_T = p(M_*)$. See Section B.1.2 of [25]. The parameters are chosen so that the probability of recovering K_T by guessing the polynomial $p(x)$ is $(2^{-8})^{10} = 2^{-80}$.
2. The second component of the private key is a string $c \in F$ that is obtained by concatenating at least 16 of the Tag conjugates and their inverses. (The inverse of a word $b_{i_1}^{\epsilon_1} b_{i_2}^{\epsilon_2} \cdots b_{i_r}^{\epsilon_r}$ is the word $b_{i_r}^{-\epsilon_r} b_{i_{r-1}}^{-\epsilon_{r-1}} \cdots b_{i_1}^{-\epsilon_1}$.)

The matrix K_T and the string c form the *private key* of the Tag. The Tag's *public key* is defined to be
$$(M_T, \sigma_T) = (K_T, 1) * c$$
where 1 is the identity permutation.

When interacting with the Tag, the Interrogator generates an ephemeral private and public key, using the set of Interrogator conjugates rather than Tag conjugates. This means that the Interrogator's private key is an $N \times N$ matrix K_I over $\mathbb{F}_{256}$ and a concatenation d of at least 16 of the Interrogator conjugates and their inverses. The Interrogator's public key is

$$(M_I, \sigma_I) = (K_I, 1) * d.$$

3.4 Authenticating a Tag

The Tag authentication protocol runs as follows. The Interrogator requests the Tag's public key (M_T, σ_T). The Interrogator also generates an ephemeral private key and sends the corresponding public key (M_I, σ_I) to the Tag. The Tag computes the shared key

$$(K_T M_I, \sigma_I) * c$$

and the Interrogator computes the shared key

$$(K_I M_T, \sigma_T) * d.$$

The function ϕ and the parameters of the scheme are designed so that these values are equal. The Interrogator requests that part of the shared key be returned to the Interrogator and authenticates the Tag if the Tag replies correctly. Though the shared key is a matrix-permutation pair, the permutation is easy to compute from public material (it is just a product of two public permutations: $\sigma_I \sigma_T = \sigma_T \sigma_I$). So the matrix is the only non-public part of the shared key.

We note that all the attacks in this paper use knowledge of the shared secret key generated during the tag authentication protocol. It is a minor detail, but since [25] restricts the length of the authentication token ($l \leq 255$) an attacker might need to repeat tag authentication using three different choices for s and $l = 255$ before recovering the entire shared secret (as the shared matrix is represented by a sequence of $8 \times N(N-1) = 720$ bits). This three-fold increase in the work effort is included in our estimates.

4 Basic Tag Impersonation

In a tag authentication protocol, an attacker can always run the tag authentication protocol against a target tag at will. The goal would be to derive enough information so that the attacker can impersonate the target tag to a genuine Interrogator in a future run of the tag authentication protocol. We now describe a simple impersonation attack of this type.

Suppose an attacker chooses a permutation σ and a set of matrices M_i, for $0 \leq i \leq N(N-1) = 90$. The matrices are chosen so that they form a basis for the space of all $N \times N$ $\mathbb{F}_{256}$ matrices for which the last row begins with $N-1$ zero values. Taken together, the matrices and the single permutation σ provide $N(N-1)+1 = 91$ spoof Interrogator public keys that are used in 91 runs of

the tag authentication protocol against the target Tag. This yields 91 shared secrets S_i, for $0 \leq i \leq N(N-1)$, remembering from Sect. 3.4 that we will need to include a further factor of three in any work effort computation.

Now suppose the attacker attempts to impersonate the target Tag to a genuine Interrogator and receives a random public key (M_I, σ_I), where $\sigma_I = \sigma$. Emulating the target Tag, the attacker computes a_i for $0 \leq i \leq N(N-1)$, so that

$$M_I = \bigoplus_{i=0}^{N(N-1)} a_i M_i.$$

The linearity observed in Eq. 2 of Sect. 3.2 guarantees that the secret S that would be computed by a genuine tag can also be computed as

$$S = \bigoplus_{i=0}^{N(N-1)} a_i S_i.$$

The attacker will be able to extract the correct authentication token from S and fool the Interrogator with 100 % certainty.

As described, the attack requires that the Interrogator choose a public key with $\sigma_I = \sigma$. At first sight, for the parameters in [25], it appears that since $N! \approx 2^{21.8}$ the probability a genuine Interrogator chooses the hoped-for σ_I is around $2^{-21.8}$. However closer analysis reveals additional structure in the conjugate sets C and D. In particular, all the permutations generated by C have five fixed points, as do all the permutations generated by D. This means that the space of possible permutations that might be encountered from a genuine Interrogator is reduced from $N!$ to $(N/2)! \approx 2^7$. The probability a genuine Interrogator chooses the hoped-for σ_I is therefore greater than 2^{-7}.

For those that prefer certainty, it is obvious an attacker can increase his success probability by performing more off-line interrogation of the target Tag using different σ. This gives a variety of trade-offs, with the extreme being an attacker who will be able to emulate the target tag with 100 % certainty after interrogating that tag around $91 \times 3 \times 5! < 2^{15}$ times.

5 Tag Private Matrix Recovery

The security of the Algebraic Eraser tag authentication protocol depends on the secrecy of two components: the $N \times N$ private $\mathbb{F}_{256}$-matrix K_T and the tag string $c \in F$. In fact, both of these need to be kept secret: in the section below we provide details of a very efficient tag impersonation attack if K_T is known; moreover, K_T can be recovered from the public key if c is known. In Section B.3 of [25] parameters are chosen so that the work effort to recover K_T by guessing the polynomial $p(x)$ used to construct it is equal to the claimed security level of 2^{80} operations.

Exploiting the linearity observed in Eq. 2 of Sect. 3.2 we show how a differential cryptanalytic attack can recover the entirety of the secret matrix K_T after

11 tag authentications. Taking into account protocol constraints and parameters specified in [25] we will need 33 tag authentications in practice, but in the following description we will set aside the factor of three for clarity.

To begin, the attacker authenticates a target Tag using any Interrogator public key (A, σ) and stores the shared secret S that results. The attacker then authenticates the same tag with N related public keys that use the same permutation σ and matrices $P_1, \ldots, P_N$ constructed as follows.

Let $E_{i,j}$ be the $N \times N$ matrix that is all zero, except its (i, j) entry which is 1. Set $P_t = A \oplus E_{t,N}$ for $1 \leq t \leq N$. The attacker challenges the target tag with the ten public keys (P_t, σ), for $1 \leq t \leq N$, and stores the secret matrices S_t that result.

One can observe that $S = K_T AV$ and $S_t = K_T P_t V$, for $1 \leq t \leq N$, where the matrix $V = \phi(\sigma, c)$ will depend on σ and the Tag's secret product c in a complicated way; the last row of V is all zero, except its last entry which is 1, by a property of E-multiplication stated above. However neither P_t nor V depend on K_T and we observe that

$$S \oplus S_t = (K_T AV) \oplus (K_T P_t V) = K_T(A \oplus P_t)V = K_T E_{t,N} V.$$

Since the last row of V has a special form, $S \oplus S_t$ will be zero everywhere except in the last column, for $1 \leq t \leq N$. Further, the values in this last column will correspond to the t^{th} column of the tag secret matrix K_T. Taken together, the entirety of the tag secret matrix K_T can be recovered column-by-column and something that is intended to require 2^{80} operations can be accomplished with negligible work after $N + 1 = 11$ interactions with the target Tag, or 33 tag authentications if we take into account the protocol constraints in [25].

This attack has been confirmed using the parameters and examples given in [25]. It has also been confirmed on parameter sets of the form B16F256 that have been supplied by SecureRF. In this latter case, with N=16, we are required to perform 17 interactions with the target Tag, or 136 tag authentications if we respect protocol considerations and only recover at most 255 bits in each interaction. Recall that parameter sets of the form B16F256 are intended to provide 128-bit security.

The linearity property that facilitates this attack appears intrinsic to the definition of the Algebraic Eraser and thus hard to avoid; increasing the size of parameters will not provide any significant additional security.

6 Efficient Tag Impersonation

Even though the tag impersonation attack of Sect. 4 is already very effective, a more efficient attack can be designed using the result of Sect. 5. This new attack is more efficient in terms of all three measures of tag queries, computation, and storage.

Recall that $d_1, d_2, \ldots, d_{32} \in F$ are the interrogator conjugates. Define their corresponding permutations $g_i \in \text{Sym}(N)$ by $g_i = \pi(d_i)$. We already observed in Sect. 4 that these permutations are highly structured and have five fixed points.

Algorithm 1. Constructing a lookup table

```
Construct a table indexed by the N! permutations in Sym(N), with all entries empty.
Add 'terminate' to the entry corresponding to the identity permutation.
Let L be a list that contains just the identity permutation.
while L non-empty do
    Let g be the first element in L.
    for i ∈ {1, 2, ..., 32} and e ∈ {−1, 1} do
        Compute g g_i^e.
        if the table entry indexed by g g_i^e is still empty then
            Change this entry to (i, e).
            Add g g_i^e to L.
    Remove g from L.
```

Stage 0: A pre-computation stage. Build an oracle which, when given a permutation $\sigma \in \mathrm{Sym}(N)$ that lies in the subgroup of $\mathrm{Sym}(N)$ generated by the g_i, returns r (a small integer), $i_1, i_2, \ldots, i_r \in \{1, 2, \ldots, 32\}$ and $\epsilon_1, \ldots, \epsilon_r \in \{-1, 1\}$ such that

$$\sigma = g_{i_1}^{\epsilon_1} g_{i_2}^{\epsilon_2} \cdots g_{i_r}^{\epsilon_r}.$$

Since $N! = 10! \leq 2^{22}$, we can build a very efficient oracle by constructing a lookup table of size $N!$ which contains the pair i_r and ϵ_r for each permutation σ that can be written as a product of the g_i (and a termination string for the identity permutation). The table may be constructed by using Algorithm 1.

Since each permutation g is added to the list L at most once, constructing the table takes at most about $N! \times 32 \times 2 \approx 2^{28}$ operations. Once the table is constructed, the oracle works on input σ by using the table to find the last element in a product of the permutations g_i and their inverses that is equal to σ. It then multiplies σ by the inverse of this last element, and iterates until it reaches the identity permutation. The oracle returns the shortest expression of the form we want (though we do not need this). The oracle is very efficient: just a few table lookups and permutation compositions are needed.

The subgroup generated by the permutations g_i in [25] is extremely small (as these permutations all fix the same five points). So building the table for the oracle above is extremely fast. We have implemented Algorithm 1 in C. It takes just 0.014 s to generate the table, and resulting oracle takes an average of under 0.00005 seconds to answer typical query, running on a 2.7 GHz i7 MacBook Pro. So the pre-computation stage takes a negligible time to complete, and the resulting oracle is extremely fast in practice.

Note that Algorithm 1 and the resulting oracle are very efficient even if the permutations d_i generate the whole of the symmetric group (the worst case for the pre-computation). Experiments with our implementation show that the table is constructed in 66 s, and the resulting oracle answers a typical query in an average of under 0.0015 s. So the pre-computation is always efficient.

For situations where it becomes impossible to store (in RAM) a table of length equal to the order of the subgroup generated by the permutations g_i, for

example if N is much larger, we would suggest first using standard Schreier–Sims techniques (see Seress [26, Chap. 4], for example) and then the powerful heuristic approach of Kalka, Teicher and Tsaban [21], to construct the oracle. Note that the pre-computed oracle can be used whenever the same set of reader conjugates are used. Since the reader conjugate set D is a public system parameter [25] an oracle can be collaboratively computed and shared over the Internet.

Stage 1: Interact with the Tag as in Sect. 5 to obtain the Tag's public key (M_T, σ_T) and then its secret key K_T.

Stage 2: Impersonate the Tag using the techniques of Phase 2 of the attack of Kalka, Teicher and Tsaban [21, Section 3.2.2]. The details are as follows.

When a legitimate interrogator queries (M_I, σ_I), query the oracle to obtain $i_1, i_2, \ldots, i_r \in \{1, 2, \ldots, 32\}$ and $\epsilon_1, \ldots, \epsilon_r \in \{-1, 1\}$ such that

$$\sigma_I = g_{i_1}^{\epsilon_1} g_{i_2}^{\epsilon_2} \cdots g_{i_r}^{\epsilon_r}.$$

Define

$$w = d_{i_1}^{\epsilon_1} d_{i_2}^{\epsilon_2} \cdots d_{i_r}^{\epsilon_r}.$$

Compute the matrix L_1 that is the result of the following E-multiplication:

$$(K_T M_I, \sigma_I) * w^{-1}.$$

Compute the matrix L_2 that is the result of the following E-multiplication:

$$(K_T^{-1} M_T, \sigma_T) * w.$$

The shared key is $(L_1 L_2, \sigma_T \sigma_I)$. This derivation has been implemented and confirmed.

7 Full Private Key Recovery

Given the extreme effectiveness of the tag impersonation attack of Sect. 6 the need for a full key recovery attack on the Algebraic Eraser tag authentication protocol is questionable. Under normal circumstances one might prefer a key-recovery attack so that recovered keys could be inserted into a cloned device, thereby exploiting the storage and performance advantages of the original algorithm. However, in our attacks, the pre-computed look-up table is small and impersonation is exceptionally fast; in fact it would be interesting to compare the performance of the impersonation attack to the computation required by the legitimate tag.

Nevertheless, to illustrate that a complete key recovery attack does exist we outline a basic attack using a *meet-in-the-middle* technique. While the attack in this section is already very effective (2^{48} storage and 2^{49} time for one of the parameter choices proposed for standardization) we believe that more analysis could reveal more practical variants.

To start, we will say that Tag conjugate products $c, c' \in F$ are *equivalent*, which we write as $c \equiv c'$, if

$$(I, 1) * c = (I, 1) * c'$$

where I is the $N \times N$ identity matrix and where 1 is the identity permutation. The definition of E-multiplication shows that when V is any fixed invertible matrix $c \equiv c'$ if and only if

$$(V, 1) * c = (V, 1) * c'.$$

In particular, when $V = K_T$ and c are the two components of the Tag private key, a private key consisting of K_T and c' will produce the same Tag public key if, and only if, $c \equiv c'$ (because K_T is invertible). Since all shared keys can be derived from the public key and the interrogator's secret information, replacing c by c' in the Tag makes no difference to any of the shared keys computed by the Tag in the protocol. So to recover the full secret key of the Tag we need only find $c' \in F$ that is equivalent to c.

Assume that the Tag's secret product c of conjugates has length 16, as allowed by [25]. There are $2 \times 32 = 2^6$ possibilities for each term in the product, and so there are $2^{6\times 16} = 2^{96}$ possibilities for c. We now describe a simple meet-in-the-middle technique that will recover an equivalent product c' using a look-up table with $\sqrt{2^{96}} = 2^{48}$ entries. The attack extends in a natural way to longer products of conjugates.

Suppose now that an attacker has recovered the Tag private matrix K_T by the attack of Sect. 5. Clearly the attacker has the Tag's public key (M_T, σ_T). The attacker then searches for products $c' \in F$ of Tag conjugates that are equivalent to c by finding c' such that $(K_T, 1) * c' = (M_T, \sigma_T)$. We write $c' = w_1'(w_2')^{-1}$ where the w_i are length eight products of Tag conjugates and their inverses. Note that

$$(K_T, 1) * w_1' = (M_T, \sigma_T) * w_2'.$$

For each of the 2^{48} possibilities for w_1', we compute $(K_T, 1) * w_1'$. We store the results in such a way that it is easy to find w_1' if we are given $(K_T, 1) * w_1'$. For example, we could use an array of pairs $((K_T, 1) * w_1', w_1')$, sorted by its first component.

For each of the 2^{48} possibilities for w_2', we compute $(M_T, \sigma_T) * w_2$ and check whether this value occurs as the first of a pair in our array. Once we find such a value w_2, we set $c' = w_1'(w_2')^{-1}$ where w_1' is the second element of the pair we have found in the array. Note that

$$(K_T, 1) * c' = ((K_T, 1) * w_1') * (w_2')^{-1} = ((M_T, \sigma_T) * w_2') * (w_2')^{-1} = (M_T, \sigma_T),$$

and so c' and K_T form a private key that produces the Tag's public key. Hence $c \equiv c'$, and we have found an equivalent private key for the Tag.

Small-scale variants of this attack—using a reduced Tag conjugate set C and shorter products—have been successfully implemented for the parameter sets B10F256 given in [25].

8 Conclusion

The Algebraic Eraser has been on the periphery of the cryptographic literature for nearly ten years. However the designers have not made it easy for independent

researchers to analyze the scheme. The reason for this approach is unclear, but the consequence has been a lack of independent peer-review.

It is too soon to determine whether or not secure schemes can be built around the mechanisms seen in the Algebraic Eraser. Certainly it is always interesting to see new techniques based on different hard problems. But any performance claims for the Algebraic Eraser are premature without a more complete understanding of the security that is delivered. The work of Ben-Zvi *et al.* [7] and that presented in this paper suggest that a lack of independent analysis has hindered the algorithm proponents from seeking out alternative viewpoints and, critically, from recognizing some very effective attacks. These have only become apparent as the profile of the algorithm has been raised and details about the algorithm have been made public.

It is hard to avoid the conclusion that the Algebraic Eraser should not be used or standardized in its current form. If future versions are proposed, and [5] provides hints that this may be the case, then it is important that a full and detailed specification be made publicly available. Just as for the parent algorithm, we believe any variants should not be used until there has been sufficient independent public cryptanalysis.

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of grain-128 with optional authentication. Int. J. Wirel. Mob. Comput. **5**(1), 48–59 (2011). Inderscience
2. Anshel, I., Atkins, D., Goldfeld, D., Gunnels, P.: Defeating the Ben-Zvi, Blackburn, and Tsaban Attack on the Algebraic Eraser. http://arxiv.org/pdf/1601.04780v1.pdf, http://eprint.iacr.org/2016/044.pdf
3. Anshel, I., Anshel, M., Goldfeld, D., Lemieux, S.: Key agreement, the Algebraic Eraser and Lightweight Cryptography. Contemporary Mathematics 418, pp. 1–34 (2006). www.securerf.com/wp-content/uploads/2014/03/SecureRF-Technical-White-Paper-06-with-Appendix-A-B.pdf
4. Atkins, D.: Algebraic Eraser: A lightweight, efficient asymmetric key agreement protocol for use in no-power, low-power, and IoT devices. www.csrc.nist.gov/groups/ST/lwc-workshop2015/papers/session8-atkins-paper.pdf
5. Atkins, D., Goldfeld, D.: Addressing the Algebraic Eraser Diffie–Hellman over-the-Air Protocol. http://eprint.iacr.org/2016/205.pdf (Pre-print)
6. Atkins, D., Gunnells, P.E.: Algebraic Eraser: A lightweight, efficient asymmetric key agreement protocol for use in no-power, low-power, and IoT devices. www.csrc.nist.gov/groups/ST/lwc-workshop2015/presentations/session8-atkins-gunnell.pdf
7. Ben-Zvi, A., Blackburn, S.R., Tsaban, B.: A Practical Cryptanalysis of the Algebraic Eraser. 7 October 2015. http://eprint.iacr.org/2015/1102 (Pre-print)
8. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
9. Cryptography Stack Exchange. Posting, November 18 2015. http://crypto.stackexchange.com/questions/30644/status-of-algebraic-eraser-key-exchange

10. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Trans. Inf. Theor. **IT–22**(6), 644–654 (1976)
11. EPCglobal. EPC Radio Frequency Identity Protocols, Generation 2 UHF RFID. Specification for RFID Air Interface Protocol for Communications at 860 MHz-960 MHz Version 2.0.1. www.gs1.org/gsmp/kc/epcglobal/uhfc1g2
12. Goldfeld, D., Gunnells, P.: Defeating the Kalka-Teicher-Tsaban linear algebra attack on the Algebraic Eraser, arXiv:1202.0598, February 2012
13. Gunnells, P.: On the cryptanalysis of the generalized simultaneous conjugacy search problem and the security of the Algebraic Eraser, arXiv:1105.1141, May 2011
14. ISO/IEC 29167-10:2015 - Information technology - Automatic identification and data capture techniques - Part 10: Crypto suite AES-128 security services for air interface communications
15. ISO/IEC 29167-11:2014 - Information technology - Automatic identification and data capture techniques - Part 11: Crypto suite PRESENT-80 security services for air interface communications
16. ISO/IEC 29167-13:2015 - Information technology - Automatic identification and data capture techniques - Part 13: Crypto suite Grain-128a security services for air interface communications
17. ISO/IEC 29167-12:2015 - Information technology - Automatic identification and data capture techniques - Part 12: Crypto suite ECC-DH security services for air interface communications
18. ISO/IEC 29167-16 - Information technology - Automatic identification, data capture techniques - Part 16: Crypto suite ECDSA-ECDH security services for air interface communications
19. ISO/IEC 29167-17:2015 - Information technology - Automatic identification and data capture techniques - Part 17: Crypto suite cryptoGPS security services for air interface communications
20. ISO/IEC 29167-20 - Information technology - Automatic identification, data capture techniques - Part 20: Crypto suite Algebraic Eraser security services for air interface communications. Working Draft
21. 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)
22. Myasnikov, A., Ushakov, A.: Cryptanalysis of the Anshel-Anshel-Goldfeld-Lemieux key agreement protocol. Groups Complex. Crypt. **1**, 63–75 (2009)
23. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001
24. SecureRF Corporation. Corporate materials. www.securerf.com
25. SecureRF Corporation. Algebraic Eraser OTA Authentication. 5 October 2015. www.securerf.com/wp-content/uploads/2015/10/Algebraic_Eraser_Over-the-Air_Authentication.pdf. Also posted at [9]
26. Seress, Á.: Permutation Group Algorithms. Cambridge University Press, Cambridge (2003)