ON THE CRYPTANALYSIS OF THE GENERALIZED SIMULTANEOUS CONJUGACY SEARCH PROBLEM AND THE SECURITY OF THE ALGEBRAIC ERASER

PAUL E. GUNNELLS

Abstract. The *Algebraic Eraser* (AE) is a cryptographic primitive that can be used to obscure information in certain algebraic cryptosystems. The *Colored Burau Key Agreement Protocol* (CBKAP), which is built on the AE, was introduced by I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux [1] in 2006 as a protocol suitable for use on platforms with constrained computational resources, such as RFID and wireless sensors. In 2009 A. Myasnikov and A. Ushnakov proposed an attack on CBKAP [7] that attempts to defeat the *generalized simultaneous conjugacy search problem*, which is the public-key computational problem underlying CBKAP. In this paper we investigate the effectiveness of this attack. Our findings are that success of the attack only comes from applying it to short keys, and that with appropriate keys the attack fails in 100% of cases and does not pose a threat against CBKAP. Moreover, the attack in [7] makes assumptions about CBKAP that do not hold in practical implementations, and thus does not represent a threat to the use of CBKAP in applications.

1. Introduction

1.1. In [1] I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux propose a key agreement protocol intended for use on low-cost platforms with constrained computational resources. Such platforms typically arise in radio frequency identification (RFID) networks and wireless sensor networks. The protocol is built on the *Algebraic Eraser* (AE), a cryptographic primitive that disguises information in many algebraic cryptosystems, such as those built on conjugation problems in braid groups. For more details, including a formal description of the AE, we refer to [1, §2].

The security of the AE is based on the hardness of the *generalized simultaneous conjugacy search problem* (GSCSP) which can be described as follows. Suppose G is a group and (X) is a property potentially satisfied by elements of G (i.e., elements satisfy property (X) if and only if they satisfy certain identities in G). Then given $y_1, \ldots, y_n \in G$, the associated GSCSP is to find elements $z, a_1, \ldots, a_n$, such that $y_i = z a_i z^{-1}$ for all i and the a_i satisfy (X). Note that GSCSP is a broader problem

Date: 28 February 2011.

Key words and phrases. Algebraic eraser, colored Burau key agreement protocol, braid group cryptography, cryptography for RFID systems.

This research was partially supported by SecureRF Corporation.

than the simultaneous conjugacy search problem where the elements $a_1, \dots, a_n$ are known and there is no specified property (X).

As an example implementation of a protocol based on the AE, the authors of [1] present the *Colored Burau Key Agreement Protocol* (CBKAP). The algebraic structure underlying this protocol is the braid group B_n, and an essential part of CBKAP is a trusted third party (TTP) algorithm that chooses secret data in B_n. We give this data in detail in §2, and for now only mention that the TTP chooses a secret element $z \in B_n$—the *conjugator*—and uses it to produce finite lists of conjugates $\{V_i\}, \{W_i\} \subset B_n$. These sets are made available for the protocol's users. As shown in [1, §6], knowledge of z allows one to break CBKAP. If both sets $\{V_i\}, \{W_i\}$ are published, the security of CBKAP relies on the assumed difficulty of recovering z from these sets. This is an instance of the GSCSP.

1.2. In [7] A. Myasnikov and A. Ushnakov present an attack on CBKAP that relies on both sets of conjugates $\{V_i\}, \{W_i\}$ being known. Instead of trying to determine z, they try to find an alternative element ζ that can play the role of z in the attack in [1, §6]. They also heuristically analyze the difficulty of finding ζ, and make the claim that they can recover the secret conjugator in all instantiations of the TTP at the security levels proposed in [1].

1.3. In this paper, we report on tests we performed with the attack in [7]. We tested the attack on randomly generated TTP data at a variety of security parameters. We also tested some of the heuristic assumptions in [7] that form the core of the attack.

We found that for suitable choices of the parameters, the attack fails in 100% of cases. More precisely, for low TTP data length, the attack in [7] is indeed successful in recovering z, and thus in breaking CBKAP. However, as lengths increase, the attack becomes far less successful and eventually fails in 100% of all cases. We also found that some of the heuristics underlying their attack are too optimistic when word lengths become long, as one would find in a typical deployment of CBKAP in a constrained computational setting.

Our tests suggest that the apparent power of the attack in [7] comes from using it against poorly chosen TTP data, in particular against braid words that are short. Moreover, with appropriate TTP data the attack poses no threat against CBKAP, even for data leading to small public/private key sizes that may be successfully deployed in low cost platforms with constrained computational resources.

1.4. Finally, we also remark that [7] uses heavily the assumption that both sets $\{V_i\}, \{W_i\}$ are known to the attacker. Indeed, this assumption can be found in [1]. However, in most practical implementations of CBKAP this will not be the case. For example, see [2] where it is shown that the AE version of the El Gamal public key encryption algorithm [4] requires only one of the sets $\{V_i\}, \{W_i\}$ to be made public. In this case, the attack in [7] cannot even be applied, and thus fails completely.

2. The TTP algorithm and the attack

2.1. Let B_n be the braid group on n strands. We denote the Artin generators by $s_1, \dots, s_{n-1}$; they satisfy the defining relations $s_i s_{i+1} s_i = s_{i+1} s_i s_{i+1}$ and $s_i s_j = s_j s_i$ if $|i-j| > 1$. Let Δ be the half-twist $(s_1 \dots s_{n-1})(s_1 \dots s_{n-2}) \dots (s_1 s_2) s_1$, whose square generates the center of B_n.

To set up an instance of CBKAP, the TTP performs the following algorithm:

(1) Choose a freely reduced word z in the generators s_i and their inverses.
(2) Choose two subgroups $B_A, B_B \subset B_n$ that are mutually commuting: $ab = ba$ for all $a \in B_A, b \in B_B$.
(3) Choose $2N$ words $v_1, \dots, v_N \in B_A$ and $w_1, \dots, w_N \in B_B$, and form the conjugates $zv_1z^{-1}, \dots, zv_Nz^{-1}$, $zw_1z^{-1}, \dots, zw_Nz^{-1}$.
(4) For $i = 1, \dots, N$:
    (a) compute the left normal form [5] of v_i and reduce the result modulo Δ^2;
    (b) let V_i be a braid word corresponding to the element obtained in (4a);
    (c) compute the left normal form of w_i and reduce the result modulo Δ^2;
    (d) let W_i be a braid word corresponding to the element obtained in (4c).

The lists $\{V_i\}$ and $\{W_i\}$ are made available to the implementers of CBKAP, and the element z is kept secret. The fundamental computational problem to break the protocol is the following: *Given the lists $\{V_i\}, \{W_i\}$ of disguised (rewritten using the braid relations) conjugates, find z and the original words $\{v_i\}, \{w_i\}$.* If one knows z, then an attack on CBKAP was already described in [1, §6].

2.2. Now we turn to the attack in [7], which begins with the following observation. To break CBKAP using the strategy in [1, §6], it is not necessary to know the original words $v_1, \dots, w_N$, which were chosen from a specific pair of mutually commuting subgroups of B_n. In fact, to apply [1, §6] one only needs to find *some* way to produce conjugates of the published lists that lie in mutually commuting subgroups. This leads to the following computational problem, which the authors of [7] call the *simultaneous conjugacy separation search problem* (SCSSP): *Given the published lists $\{V_i\}, \{W_i\}$, find an element ζ and integers $p_1, \dots, p_N, q_1, \dots, q_N$ such that the two sets $\{w_i'\} = \{\Delta^{2p_i}\zeta^{-1}W_i\zeta \mid i = 1, \dots, N\}$ and $\{v_i'\} = \{\Delta^{2q_i}\zeta^{-1}V_i\zeta \mid i = 1, \dots, N\}$ are subsets of mutually commuting subgroups of B_n.* The element ζ is then applied in the linear attack described in [1, §6], in which it plays the role of the conjugator z. Of course the original z and exponents of Δ^2 used in the normal form reduction in steps (4a), (4c) will solve the SCSSP, but there could be other choices that work as well.

Thus the attack falls naturally into two steps:

(1) Determine the exponents p_i, q_i.
(2) Determine the conjugator ζ.

Both steps rely heavily on a function $|\cdot|_a \colon B_n \to \mathbb{Z}$, the *approximate length function.* This function, originally defined in [8], serves as a replacement for the geodesic length

$l\colon B_n \to \mathbb{Z}$ in the Cayley graph of B_n. We discuss this function more below, and for now explain how it is used in attack.

2.3. We begin with step (1). Let X be any element from the published lists of disguised conjugates. We want to find the associated exponent p of Δ^2 that should be applied with X to solve the SCSSP. Consider the set of integers

$$\{|\Delta^{2j}X|_a \mid j \in \mathbb{Z}\}. \tag{1}$$

We assume that the set (1) attains a minimum at some integer p. This is our desired exponent for X. We repeat the procedure for all V_i and W_i.

2.4. After finding all the exponents $p_1, \dots, q_N$, the next step (2) is finding ζ. To explain how this is done, we need more notation. Let $\mathbf{x} = (x_1, \dots, x_N)$ be a tuple of words in B_n. Let $|\mathbf{x}|_a = \sum |x_i|_a$ be the total approximate length of $\mathbf{x}$. For any $w \in B_n$ let $\mathbf{x}^w = (w^{-1}x_1w, \dots, w^{-1}x_Nw)$.

Now suppose we have two tuples $\mathbf{x}, \mathbf{y}$ that we know a priori can be conjugated into two commuting subgroups. Put $\zeta = 1$. We consider simultaneous conjugation of $\mathbf{x}, \mathbf{y}$ by generators, and how the total approximate length of the tuples $\mathbf{x}, \mathbf{y}$ change. In other words, for each $\sigma \in \{s_1^{\pm 1}, \dots, s_{n-1}^{\pm 1}\}$, let δ_σ be defined by

$$\delta_\sigma = |\mathbf{x}^\sigma|_a + |\mathbf{y}^\sigma|_a - (|\mathbf{x}|_a + |\mathbf{y}|_a).$$

If $\delta_\sigma > 0$, then conjugation by σ makes the tuples $\mathbf{x}, \mathbf{y}$ longer overall, and so σ should not appear on the end of a reduced expression for ζ. On the other hand, if $\delta_\sigma < 0$, then conjugation by σ represents progress towards constructing ζ. We replace ζ with $\zeta\sigma$, replace $\mathbf{x}, \mathbf{y}$ with $\mathbf{x}^\sigma, \mathbf{y}^\sigma$, and repeat the process if $\mathbf{x}^\sigma, \mathbf{y}^\sigma$ are not supported on mutually commuting subgroups. A variation of this procedure keeps track of the sequence $\sigma_1, \sigma_2, \dots$ and uses backtracking to try more possibilities for ζ.

3. The approximate length function

3.1. A key role in the attack is played by the approximate length function $|\cdot|_a$, originally defined in [8]. To explain it we need more notation.

Let $w \in B_n$ be represented by a reduced expression $s_{i_1} \cdots s_{i_r}$. The *main generator* of w in this expression is the generator s_j such that j is the minimal subscript i_k appearing in the expression. A word is *Dehornoy reduced* if its main generator does not appear simultaneously with its inverse [3]. Typically there are many Dehornoy reduced expressions representing w, but one can write a deterministic program to produce a unique one. Following [3], one can further use the reduction procedure to produce a *fully reduced word.* Such a word is also Dehornoy reduced, but satisfies additional properties that tend to make it substantially shorter than the original word. We assume this has been done, and let $D(w)$ be the full reduction of w.

3.2. The idea behind computing $|w|_a$ is to produce a word w' equivalent to w using a combination of full reduction and right conjugation by Δ. The latter affects a reduced expression $w = s_{i_1}^{\varepsilon_{i_1}} \cdots s_{i_k}^{\varepsilon_{i_k}}$ by replacing each generator s_j by its "complement" s_{n-j}:

$$w^{\Delta} := \Delta^{-1} w \Delta = s_{n-i_1}^{\varepsilon_{i_1}} \cdots s_{n-i_k}^{\varepsilon_{i_k}}.$$

The algorithm to compute $|w|_a$ works as follows. We begin by putting $w_0 = w$ and $i = 0$. Let the word length of z be denoted $|z|$. Then we apply the sequence

```
(1) Increment $i$ and put $w_i = D(w_{i-1})$.
(2) If $|w_i| < |w_{i-1}|$, then
    (a) put $w_i = w_i^{\Delta}$ and
    (b) goto Step 1.
(3) Otherwise,
    (a) if $i$ is even output $w' = w_{i+1}^{\Delta}$;
    (b) if $i$ is odd output $w' = w_{i+1}$.
```

The output is a word w' equivalent to w with $|w'| \leq |w|$. Finally we define $|w|_a$ to be $|w'|$. In practice, for instance as implemented in [6], one does not repeat (1)–(2) until $|w_i| \geq |w_{i-1}|$, but instead iterates a fixed number of times.

The authors of [7] claim that $|\cdot|_a$ approximates the geodesic length l well enough so that two key properties hold. First, they claim that for generic tuples $\mathbf{x}$ and words w, we have $|\mathbf{x}^w|_a > |\mathbf{x}|_a$. Next, they claim that $|\cdot|_a$ approximately satisfies the triangle inequality. Namely, we have $|w|_a + |u|_a \geq |wu|_a$ for generic words w, u. Both properties play a key role in the heuristic justifying the attack on the TTP algorithm.

4. Tests and findings

4.1. Our tests naturally split into two topics. First we tested features of the approximate length function, in particular how well the computation of $|\cdot|_a$ shortens words compared to full reduction, as well as how well the approximate length function satisfies the triangle inequality. Next we tested the attack against the TTP algorithm for a variety of randomly generated TTP data.

All algebraic computations with braid groups—including randomly generating braid words, the implementation of the attack in [7], and the computation of the approximate length function $|\cdot|_a$—were performed using the C++ library `crag` written by the authors of [7], and distributed through the Algebraic Cryptography Group at the Stevens Institute of Technology. The code is freely available on the internet [6].

4.2. **Approximate length function: reduction.** In these tests we fixed a braid group B_n, then generated many freely reduced words w of various lengths and computed $|w|_a/|D(w)|$. The results for the groups $B_8, B_{16}, \ldots, B_{48}$ are plotted in Figure 1; each data point represents 100 trials.

We found that when lengths of random generated initial words w are short relative to the rank n, the function $|w|_a$ is essentially the length $|D(w)|$ of the full reduction $D(w)$ of w. On the other hand, when the length of the initial word increases, the ratio

$|w|_a/|D(w)|$ drops quickly, even more rapidly as the number of strands is increased. As the length increases even more, it appears that the ratio $|w|_a/|D(w)|$ stabilizes to a constant. The data for B_8 may suggest that this constant is asymptotically 1. Thus it appears that full reduction combined with conjugation by Δ can be used to produce rather short words, at least for a certain range of initial lengths depending on the index.

We also checked how well $|\cdot|_a$ performed before and after applying Thurston left normal form Tlnf to a long freely reduced word. This normal form, described in [5], can be used to prove automaticity of the braid group. For a randomly chosen freely reduced word w representing an element of the braid group, the word $\mathrm{Tlnf}(w)$ is generally much longer than w. We found that the approximate length function is relatively insensitive to passing through Tlnf, and in particular $|w|_a$ is very close to $|\,\mathrm{Tlnf}(w)|_a$ in most cases.

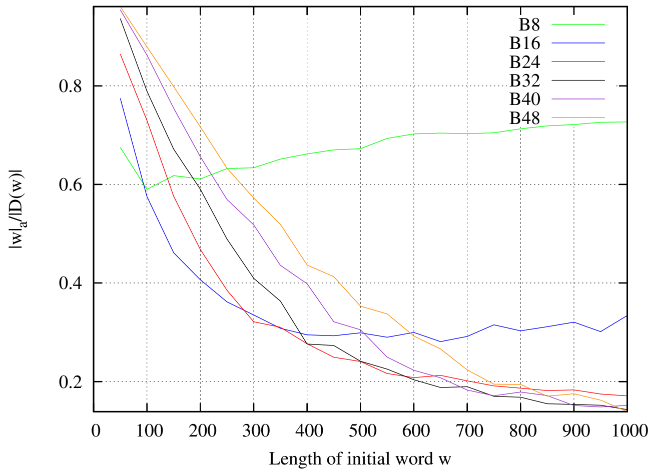

FIGURE 1. Approximate length compared with length of full reduction in various braid groups.

4.3. **Approximate length function: triangle inequality.** Next we checked how well the approximate length function satisfies the triangle inequality $|x|_a+|y|_a \geq |xy|_a$. We considered the same sequence of braid groups as in §4.2. After fixing B_n, we generated many freely reduced words x, y of the same length, and then computed the relative error $100\cdot(|xy|_a-(|x|_a+|y|_a))/|xy|_a$. The results are shown in Figure 2. Again each data point shows the average over 100 trials; the horizontal axis represents the length of the randomly chosen x, y.

Thus Figure 2 shows the average relative error between $|xy|_a$ and $|x|_a+|y|_a$. If this quantity is negative, then the inequality holds on average, and if positive, then $|x|_a+|y|_a < |xy|_a$ on average. The data indicates that for short words, if the rank n is increased then the triangle inequality seems to hold, with $|xy|_a$ considerably smaller than $|x|_a + |y|_a$ on average. But if the lengths of x, y are increased, then for all ranks ultimately the triangle inequality fails to hold for $|\cdot|_a$ on average. The asymptotic behavior is not clear. We remark that the of course the triangle inequality holds in all cases for the geodesic metric in the Cayley graph of B_n.

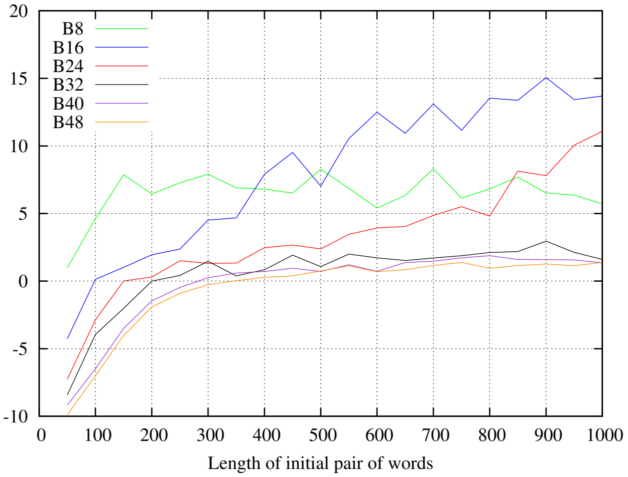

FIGURE 2. Failure of the triangle inequality for $|\cdot|_a$ in various braid groups.

4.4. **The attack I: sensitivity to overall length of TTP data.** Next we tested the attack against randomly generated TTP data. We fixed the braid group B_{16}. As above each data point represents 100 trials with given parameter choices.

First we ran a series of tests in which N varied from 2 to 10; recall that the TTP data consists of $2N$ conjugates divided into two sets of N. In these tests the elements z and v_i, w_j were chosen to have approximately the same word length. The results are plotted in Figure 3. The data clearly shows that for small elements, i.e. when $|z|, |v_i|, |w_j| \approx 67$ and thus $|zv_iz^{-1}|, |zw^jz^{-1}| \approx 200$, the attack is successful in almost all cases, regardless of the number of conjugates. But as the lengths are increased, the success rate drops off quickly until the attack fails in all cases. Moreover, the success rate drops off more quickly as the number of conjugates is increased.

4.5. **The attack II: dependence on relative sizes of z and v_i, w_j.** Next we ran a series of tests to investigate the performance of the attack when the lengths of $\{zv_iz^{-1}, zw_jz^{-1}\}$ are fixed and approximately equal, but $|z|$ is very different from $|v_i|, |w_j|$. We fixed N to be 8 and $|zv_iz^{-1}| \approx |zw^jz^{-1}| \approx 350$, and considered $|z| = 25, 50, \ldots, 150$. These parameters were chosen because Figure 3 shows that the attack is successful about 15% of the time when the lengths of z, v_i, and w_j are roughly equal and the length of the conjugates is about 350. Hence at these lengths one can evaluate the performance of the attack when the relative lengths are varied.

The results, shown in Figure 4, indicate that increasing the length of z significantly hampers the success of the attack. Comparison of Figure 4 with the relevant data point in the plot for $N = 8$ in Figure 3 is also instructive. In the former, we have $|z| \approx 125, |v_i|, |w_j| \approx 100$ with the total length of each conjugate about 350. In the latter when $|z| \approx 125$ we have $|v_i|, |w_j| \approx 125$, so that the total length is 375. Thus the conjugators have the same length and conjugates are slightly longer, yet the success rate of the attack in the latter case is substantially lower.

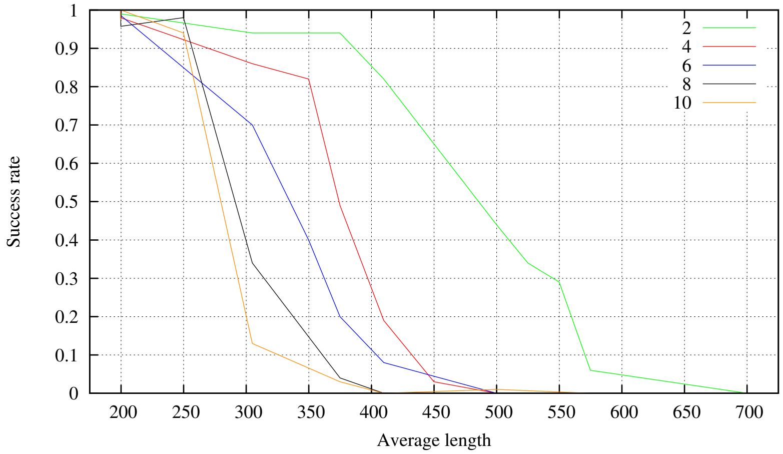

Figure 3. Performance of the attack against randomly generated TTP data. The ambient braid group is B_{16} and we show data for different N. *Average length* refers to word lengths of $\{zv_iz^{-1}, zw^jz^{-1}\}$. In this data the length of z is roughly equal to that of v_i, w_j. In all cases the attack is successful for short lengths and experiences a phase transition to failure as lengths are increased. The rapidity of the transition depends on how many conjugates are used in the two sets.

5. Conclusions and discussion

5.1. First, the approximate length function described in [8] uses a combination of full reduction and conjugation by Δ to produce short expressions for words. It does appear to offer an improvement over full reduction, in that in almost all cases we tested it appears to produce rather short words. We conclude that this reduction technique can be used to produce shorter words than those from full reduction, at least for a set of lengths depending on the index.

5.2. Next, the assumption that the triangle inequality holds for the approximate length function $|\cdot|_a$ appears to be too optimistic. As the lengths of words increase, the triangle inequality apparently holds less and less often. The asymptotic behavior is

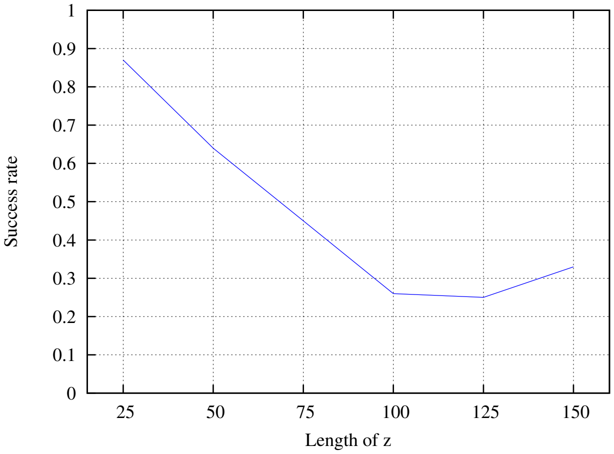

FIGURE 4. Investigating attack performance on short words when the relative lengths of z and v_i, w_j are varied. For this data we work in in B_{16} with $N = 8$ and take $2|z| + |v_i|, 2|z| + |w_j| \approx 350$, in B_{16} with $N = 8$. This graph refines one data point in Figure 3 at which the attack is successful about 15% of the time.

not clear from our experiments, but nevertheless some of the data (B_{16}, B_{24}) suggests that the inequality might fail quite badly in the long run.

5.3. Regarding the attack on CBKAP described in §4.4, we find that it is successful if the words $\{V_i\}, \{W_i\}$ are short, and that the claims in [7] about the data they tested appear valid. However, as the lengths of $\{V_i\}, \{W_i\}$ increase, the attack quickly loses power, and soon fails in all instances. Furthermore, the attack does not seem robust against easily implemented defenses. Increasing the number of conjugates, for instance, causes the attack to fail at much shorter word lengths. Modifying key selection by varying the length of the conjugator also adversely affects the attack's success. Experiments also show that selecting keys more carefully—for instance, but applying criteria to randomly generated TTP data that go beyond length considerations alone—also quickly hampers the performance of the attack. We conclude that the success of the attack seems mainly to be due to it being applied to short words.

REFERENCES

[1] I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux, *Key agreement, the Algebraic Eraser*$^{\mathrm{TM}}$*, and lightweight cryptography*, Algebraic methods in cryptography, Contemp. Math., vol. 418, Amer. Math. Soc., Providence, RI, 2006, pp. 1–34.

[2] I. Anshel, D. Goldfeld, and P. E. Gunnells, *Fast asymmetric encryption using the Algebraic Eraser*, in preparation.

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

[4] T. El Gamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions of Information Theory **31** (1985), no. 4, 469–472.

[5] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. V. F. Levy, M. S. Paterson, and W. P. Thurston, *Word processing in groups*, Jones and Bartlett Publishers, Boston, MA, 1992.

[6] A. D. Myasnikov and A. Ushakov, *CRyptography And Groups (CRAG) C++ and Python Library*, available from `www.stevens.edu/algebraic/index.php`.

[7] 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.

[8] A. Myasnikov, V. Shpilrain, and A. Ushakov, *A practical attack on a braid group based cryptographic protocol*, Advances in cryptology—CRYPTO 2005, Lecture Notes in Comput. Sci., vol. 3621, Springer, Berlin, 2005, pp. 86–96.

Dept. of Mathematics and Statistics, UMass Amherst, Amherst, MA 01003
E-mail address: `gunnells@math.umass.edu`