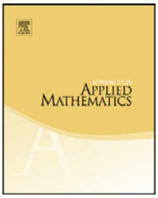
Advances in Applied Mathematics 49 (2012) 57–76

Contents lists available at SciVerse ScienceDirect

Advances in Applied Mathematics

www.elsevier.com/locate/yaama



Short expressions of permutations as products and cryptanalysis of the Algebraic Eraser

Arkadius Kalka [1], Mina Teicher, Boaz Tsaban *

Department of Mathematics, Bar-Ilan University, Ramat Gan 52900, Israel

ARTICLE INFO

Article history:
Received 10 August 2009
Revised 18 May 2011
Accepted 5 March 2012
Available online 22 March 2012

MSC:
20B40
94A60
14G50

Keywords:
Expressions of permutations
Algebraic Eraser
Colored Burau Key Agreement Protocol (CBKAP)
Symmetric group
Expander
Minimal Cycle Conjecture

ABSTRACT

On March 2004, Anshel, Anshel, Goldfeld, and Lemieux introduced the *Algebraic Eraser* scheme for key agreement over an insecure channel, using a novel hybrid of infinite and finite noncommutative groups. They also introduced the *Colored Burau Key Agreement Protocol* (*CBKAP*), a concrete realization of this scheme.

We present general, efficient heuristic algorithms, which extract the shared key out of the public information provided by CBKAP. These algorithms are, according to heuristic reasoning and according to massive experiments, successful for all sizes of the security parameters, assuming that the keys are chosen with standard distributions.

Our methods come from probabilistic group theory (permutation group actions and expander graphs). In particular, we provide a simple algorithm for finding short expressions of permutations in S_n, as products of given random permutations. Heuristically, our algorithm gives expressions of length $O(n^2 \log n)$, in time and space $O(n^3)$. Moreover, this is provable from *the Minimal Cycle Conjecture*, a simply stated hypothesis concerning the uniform distribution on S_n. Experiments show that the constants in these estimations are small. This is the first practical algorithm for this problem for $n \geqslant 256$.

Remark. *Algebraic Eraser* is a trademark of SecureRF. The variant of CBKAP actually implemented by SecureRF uses proprietary distributions, and thus our results do not imply its vulnerability.

* Corresponding author.
E-mail addresses: Arkadius.Kalka@rub.de (A. Kalka), teicher@math.biu.ac.il (M. Teicher), tsaban@math.biu.ac.il (B. Tsaban).
URL: http://www.cs.biu.ac.il/~tsaban (B. Tsaban).

[1] Current address: Max Planck Institute for Mathematics, Bonn 53111, Germany.

0196-8858/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.aam.2012.03.001

1. Introduction and overview

Starting with the seminal papers [1,16], several attempts have been made to construct and analyze public key schemes based on noncommutative groups and combinatorial, or computational, group theory. One motivation is that such systems may provide longer term security than existing schemes. Another motivation, at present theoretical, is that unlike the main present day public key schemes, these new schemes may be resistant to attacks by quantum computers. Already in the short run, these connections between combinatorial group theory and cryptography lead to mathematical questions not asked before, and consequently to new mathematical and algorithmic results.

In this paper, we study a scheme falling in the above category, whose cryptanalysis leads to an algorithm with potential applicability beyond the studied scheme.

The *Algebraic Eraser* key agreement scheme was introduced by Anshel, Anshel, Goldfeld, and Lemieux in the workshop *Algebraic Methods in Cryptography* held in Dortmund, Germany, on March 2004, and in the special session on *Algebraic Cryptography*, at the Joint International Meeting of the AMS, DMV, and ÖMG, held in Mainz, Germany, on June 2005. It was subsequently published as [2].

Apart from its mathematical novelty, the Algebraic Eraser has a surprisingly simple concrete realization, the *Colored Burau Key Agreement Protocol* (*CBKAP*), which consists of an efficient combination of matrix multiplications, applications of permutations, and evaluations of polynomials at elements of a finite field.

For four years since its introduction, no weakness in CBKAP was reported. On January 13, 2008, Kalka and Tsaban have described the attack presented here in Bar-Ilan University's *CGC Seminar* [9]. At about the same time (on January 30, 2008), Myasnikov and Ushakov uploaded to the ArXiv eprint server an independent attack, and subsequently published it [20]. Myasnikov and Ushakov use a length based algorithm to break the Third Trusted Party (TTP) component of CBKAP, with excellent success rates for the parameters proposed in [2]. They indicate that the success rates of their attack drop if the parameters are increased. In his recent paper [15], Gunnells reproduces Myasnikov and Ushakov's attack, and concludes that as the key lengths increase, the attack quickly loses power, and soon fails in all instances. Furthermore, he provides experiments suggesting that their attack is not robust against several easily implemented defenses. He concludes that "the success of the attack seems mainly to be due to it being applied to short words" [15].

The security of the main ingredient of CBKAP is not addressed in [20]. Would fixing the TTP component make the protocol secure? Moreover, in the recent work [3] it is shown that in many scenarios, there is no need to make both groups A and B in the protocol public (details below). For this variant, the attack presented in [20] does not seem applicable [15].

We present an efficient attack, which recovers the shared key out of the public information, even if one of the involved groups mentioned above remains hidden, without attacking the TTP's private key. According to heuristic reasoning as well as massive experiments, the attack is efficient, and has 100% success rates for all feasible sizes of the security parameters, assuming standard distributions on the key spaces.[2]

The methods, which make the attack applicable to large security parameters, come from probabilistic group theory, and deal with permutation groups. About half of the paper is dedicated to a new heuristic algorithm for finding short expressions of permutations as words in a given set of randomly chosen permutations. This algorithm solves efficiently instances which are intractable using previously known, provable or heuristic, techniques.

We conclude this introduction with several general comments.

1.1. Construction versus analysis

Few schemes, not counting minor variations, have been proposed thus far in the context of combinatorial group theory: Mainly, those in [1,16], and the one from [2], which is studied here. Most

[2] See the remark at the abstract, which applies to this paper as well as to [20,15].

of the attempts thus far are on the side of analysis of the proposed schemes, rather than proposals of substantially new ones. Indeed, each of the mentioned schemes is in fact an infinite family of possible schemes, with at least two degrees of freedom: Choosing the platform groups, and choosing the distributions on the chosen platform group. There are at present no known attacks which are guaranteed to succeed against all candidate groups (with standard distributions on them), or against all distributions on certain groups (like the braid group) which can be sampled efficiently. Thorough analyzes may give indication which choices may lead to secure schemes.

1.2. Key generation in infinite groups or monoids

There is a canonical distribution on infinite groups or monoids generated by finitely many generators $g_1, \ldots, g_k$. This is defined by fixing a length parameter L, and then taking a product of L elements g_i, each chosen with uniform distribution from the set $\{g_1, \ldots, g_k\}$.[3] This is not a uniform distribution on the group,[4] but the distribution induced from the uniform distribution on the words of length L in the free monoid. Since the distributions are not specified in [2], we (as well as Myasnikov and Ushakov [20] and Gunnells [15]) assume these natural distributions when finitely generated groups or monoids are considered, and the uniform distribution when finite sets are considered. By Gunnells result [15], the results of the present paper form the first cryptanalysis of CBKAP for these distributions, which works for all key sizes.

1.3. Provable security

Given a cryptographic scheme, it is desirable to have a simply stated (and apparently hard) algorithmic problem such that, if the given scheme can be broken, then there is an efficient algorithm for the problem. From a cryptographic point of view, there is no point in doing so when a scheme can be cryptanalyzed, as is the case here. We believe, however, that cryptanalyses will improve our understanding of schemes based on combinatorial group theory, and this may lead, eventually, to introduction of schemes which look promising (i.e., resist known cryptanalyses). Then, establishing a provable link between the security of the scheme and the difficulty of a simply stated and apparently hard algorithmic problem would be an important task, which would help understanding better the (potential) security of the scheme.

2. The Algebraic Eraser scheme

We describe here the general framework. The concrete realization will be described later.

2.1. Notation, terminology, and conventions

A *monoid* is a set M with a distinguished element $1 \in M$, equipped with an associative multiplication operation for which 1 acts as an identity. Readers not familiar with this notion may replace “monoid” with “group” everywhere, since this is the main case considered here.

Let G be a group acting on a monoid M on the left, that is, to each $g \in G$ and each $a \in M$, a unique element denoted ${}^{g}a \in M$ is assigned, such that:

(1) ${}^{1}a = a$;
(2) ${}^{gh}a = {}^{g}({}^{h}a)$; and
(3) ${}^{g}(ab) = {}^{g}a \cdot {}^{g}b$

for all $a, b \in M$, $g, h \in G$.

[3] In the case of a group, we first extend the list of generators, if necessary, so that for each g in the list, g^{-1} is also in the list.

[4] Since the group or monoid is countably infinite, there is no uniform distribution on it.

$M \times G$, with the operation

$$(a, g) \circ (b, h) = \left(a \cdot {}^{g}b, gh\right),$$

is a monoid denoted $M \rtimes G$, and called the *semi-direct product* of M and G.

Let N be a monoid, and $\varphi : M \to N$ a homomorphism. The *Algebraic Eraser* operation is the function $\star : (N \times G) \times (M \rtimes G) \to (N \times G)$ defined by

$$(a, g) \star (b, h) = \left(a\varphi\left({}^{g}b\right), gh\right). \tag{1}$$

$M \rtimes G$ *acts on the right* on $N \times G$, that is, the following identity holds

$$\big((a, g) \star (b, h)\big) \star (c, r) = (a, g) \star \big((b, h) \circ (c, r)\big) \tag{2}$$

for all $(a, g) \in N \times G$, $(b, h), (c, r) \in M \times G$.

Submonoids A, B of $M \rtimes G$ are $\star$-*commuting* if

$$\big(\varphi(a), g\big) \star (b, h) = \big(\varphi(b), h\big) \star (a, g) \tag{3}$$

for all $(a, g) \in A$, $(b, h) \in B$. In particular, if A, B $\star$-commute, then

$$\varphi(a)\varphi\left({}^{g}b\right) = \varphi(b)\varphi\left({}^{h}a\right)$$

for all $(a, g) \in A$, $(b, h) \in B$.

2.1.1. Didactic convention

Since the actions are superscripted, we try to minimize the use of subscripts. As a rule, whenever two parties, Alice and Bob, are involved, we try to use for Bob letters which are subsequent to the letters used for Alice (as is suggested by their names).

2.2. The Algebraic Eraser key agreement scheme

2.2.1. Public information

(1) The group G and the monoids M, N.
(2) A positive integer m.
(3) $\star$-Commuting submonoids A, B of $M \rtimes G$, each given in terms of a generating set of size k.
(4) Element-wise commuting submonoids C, D of N.

Remark 1. For clarity of exposition, we assume that m, k are public, and identical for Alice's and Bob's parts. However, this assumption is not required for the scheme to work, nor for its cryptanalysis described below.

2.2.2. The protocol

(1) Alice chooses $c \in C$ and $(a_1, g_1), \ldots, (a_m, g_m) \in A$, and sends

$$(p, g) = (c, 1) \star (a_1, g_1) \star \cdots \star (a_m, g_m) \in N \times G$$

(the $\star$-multiplication is carried out from left to right) to Bob.

(2) Bob chooses $d \in D$, $(b_1, h_1), \ldots, (b_m, h_m) \in B$, and sends

$$(q, h) = (d, 1) \star (b_1, h_1) \star \cdots \star (b_m, h_m) \in N \times G$$

to Alice.

(3) Alice and Bob compute the shared key:

$$(cq, h) \star (a_1, g_1) \star \cdots \star (a_m, g_m) = (dp, g) \star (b_1, h_1) \star \cdots \star (b_m, h_m).$$

We will soon explain why this equality holds.

For the sake of mathematical analysis, it is more convenient to reformulate this protocol as follows. The public information remains the same. In the notation of Section 2.2.2, define

$$(a, g) = (a_1, g_1) \circ \cdots \circ (a_m, g_m) \in A;$$
$$(b, h) = (b_1, h_1) \circ \cdots \circ (b_m, h_m) \in B.$$

By Eqs. (2) and (1), Alice and Bob transmit the information

$$(p, g) = (c, 1) \star (a_1, g_1) \star \cdots \star (a_m, g_m) = (c, 1) \star (a, g) = \bigl(c\varphi(a), g\bigr);$$
$$(q, h) = (d, 1) \star (b_1, h_1) \star \cdots \star (b_m, h_m) = (d, 1) \star (b, h) = \bigl(d\varphi(b), h\bigr).$$

Using this and Eq. (3), we see in the same manner that the shared key is

$$\begin{aligned}(cq, h) \star (a, g) &= \bigl(cq\varphi\bigl({}^{h}a\bigr), hg\bigr)\\ &= \bigl(cd\varphi(b)\varphi\bigl({}^{h}a\bigr), hg\bigr) = \bigl(dc\varphi(a)\varphi\bigl({}^{g}b\bigr), gh\bigr)\\ &= \bigl(dp\varphi\bigl({}^{g}b\bigr), gh\bigr) = (dp, g) \star (b, h). \qquad (4)\end{aligned}$$

2.3. When M is a group

In the concrete examples for the Algebraic Eraser scheme, M is a group [2]. Consequently, $M \rtimes G$ is also a group, with inversion

$$(a, g)^{-1} = \bigl({}^{g^{-1}}a^{-1}, g^{-1}\bigr)$$

for all $(a, g) \in M \rtimes G$.

3. A general attack on the scheme

We will attack a stronger scheme, where only one of the groups A or B is made public. Without loss of generality, we may assume that A is known. A is generated by a given k-element subset. Let $(a_1, s_1), \ldots, (a_k, s_k) \in M \rtimes G$ be the given generators of A. Let $S = \{s_1, \ldots, s_k\}$. $S^{\pm 1}$ denotes the symmetrized generating set $\{s_1, \ldots, s_k, s_1^{-1}, \ldots, s_k^{-1}\}$.

3.1. Assumptions

3.1.1. Distributions and complexity

Alice and Bob make their choices according to certain distributions. Whenever we mention a *probability*, it is meant with respect to the relevant distribution. All assertions made here are meant to hold "with significant probability" and the generation of elements must be possible within the available computational power. We will quantify our statements later.

Assumption 2. It is possible to generate an element $(\alpha, 1) \in A$ with $\alpha \neq 1$.

Assumption 2 is equivalent to the possibility of generating $(\alpha, g) \in A$ such that the order o of g in G is smaller than the order of (α, g) in $M \rtimes G$. Indeed, in this case $(\alpha, g)^o$ is as required.

Assumption 3. N is a subgroup of $\mathrm{GL}_n(\mathbb{F})$ for some field $\mathbb{F}$ and some n.

We do not make any assumption on the field $\mathbb{F}$.

Alice generates an element $(a, g) \in A$, and in particular she generates g in the subgroup of G generated by S.

Assumption 4. Given $g \in \langle S \rangle$, g can be explicitly expressed as a product of elements of $S^{\pm 1}$.

3.2. The attack

3.2.1. First phase: Finding d and $\varphi(b)$ up to a scalar

By $\star$-commutativity of A and B, and since $(b, h) \in B$, we have that for each $(\alpha, 1) \in A$,

$$\varphi(\alpha)\varphi(b) = \varphi(b)\varphi\left({}^{h}\alpha\right). \tag{5}$$

By Assumption 2, we can generate such equations with α known, so that only $\varphi(b)$ is unknown.

Now, $q = d\varphi(b)$ is a part of the transmitted information. Substituting $\varphi(b) = d^{-1}q$ in Eq. (5), we obtain

$$d\varphi(\alpha) = \left(q\varphi\left({}^{h}\alpha\right)q^{-1}\right)d,$$

where only d is unknown. Moreover, as C, D commute element-wise, we have that

$$d\gamma = \gamma d \tag{6}$$

for all $\gamma \in C$.

Even for just one nontrivial α and one $\gamma \in C$, we obtain $2n^2$ equations on the n^2 entries of d. Thus, if standard distributions were used to generate the keys, we expect, heuristically, that the solution space will be one-dimensional. (As this is a homogeneous equation and the matrices are invertible, the solution space cannot be zero-dimensional.) If it is accidentally not, we can generate more equations in the same manner.[5]

More formally, let $d, \tilde{d}$ be solutions to Eqs. (5) and (6), say for $(\alpha_1, 1), \ldots, (\alpha_r, 1) \in A$, and $\gamma_1, \ldots, \gamma_s \in C$. Then

[5] In the case of CBKAP (the concrete realization described below), one equation of each type was enough in all experiments we have conducted. Except for few exceptions in tiny parameter settings, where exhaustive search of the key can be carried out easily.

$$\begin{aligned}\tilde{d}d^{-1} &= \tilde{d}\varphi(\alpha_i)\big(d\varphi(\alpha_i)\big)^{-1} = \big(q\varphi\big({}^h\alpha_i\big)q^{-1}\big)\tilde{d}\big(\big(q\varphi\big({}^h\alpha_i\big)q^{-1}\big)d\big)^{-1}\\ &= \big(q\varphi\big({}^h\alpha_i\big)q^{-1}\big)\tilde{d}d^{-1}\big(q\varphi\big({}^h\alpha_i\big)q^{-1}\big)^{-1},\end{aligned}$$

and thus $\tilde{d}d^{-1}$ commutes with $q\varphi({}^h\alpha_i)q^{-1}$, for each $i \in \{1,\ldots,r\}$.

Moreover, $\tilde{d}d^{-1}$ commutes with all elements of C, and thus is in the centralizer of

$$\big\{q\varphi\big({}^h\alpha_1\big)q^{-1},\ldots,q\varphi\big({}^h\alpha_r\big)q^{-1}\big\} \cup \{\gamma_1,\ldots,\gamma_s\}.$$

Similarly, we have that $d^{-1}\tilde{d}$ is in the centralizer of

$$\big\{\varphi(\alpha_1),\ldots,\varphi(\alpha_r)\big\} \cup \{\gamma_1,\ldots,\gamma_s\}.$$

Our precise assumption is that for some, not large, numbers r, s, we have that with high probability, (at least) one of these centralizers is one-dimensional. Since C is, by assumption, a group of matrices, this means that this centralizer is not larger than the centralizer of the full matrix group $\mathrm{GL}_n(\mathbb{F})$, i.e. the scalar matrices. (Observe that $d^{-1}\tilde{d}$ is scalar if and only if $\tilde{d}d^{-1}$ is.)

If one can find a small generating set $\{\gamma_1,\ldots,\gamma_s\}$ for C,[6] then the assumption tells that the centralizer of

$$\big\{\varphi\big({}^h\alpha_1\big),\ldots,\varphi\big({}^h\alpha_r\big)\big\} \cup q^{-1}Cq$$

or of

$$\big\{\varphi(\alpha_1),\ldots,\varphi(\alpha_r)\big\} \cup C$$

is one-dimensional.

Thus, heuristically, we assume that we have found xd for some unknown scalar $x \in \mathbb{F}$. Now use our knowledge of $q = d\varphi(b)$ to compute

$$(xd)^{-1}q = \frac{1}{x}d^{-1}q = \frac{1}{x}\varphi(b).$$

In summary: We know xd and $x^{-1}\varphi(b)$, for some unknown scalar $x \in \mathbb{F}$.

3.2.2. *Second phase: Generating elements with a prescribed G-coordinate and extracting the key*

Using Assumption 4, find $i_1,\ldots,i_\ell \in \{1,\ldots,k\}$ and $\epsilon_1,\ldots,\epsilon_\ell \in \{1,-1\}$ such that

$$g = s_{i_1}^{\epsilon_1}\cdots s_{i_\ell}^{\epsilon_\ell}.$$

Compute

$$(\delta, g) = (a_{i_1}, s_{i_1})^{\epsilon_1} \circ \cdots \circ (a_{i_d}, s_{i_\ell})^{\epsilon_\ell} \in A.$$

δ may or may not be equal to a.

[6] Indeed, in CBKAP, described below, C is cyclic.

Remark 5. If M is finitely generated as a monoid, the expression in Assumption 4 should be as a product of elements of S. In the cases discussed later in this paper, $G = S_n$ and the methods of Section 5 can be adjusted to obtain positive expressions (Remark 10).

By $\star$-commutativity of (δ, g) and (b, h), $\varphi(b)\varphi({}^h\delta) = \varphi(\delta)\varphi({}^g b)$, and thus we can compute

$$x^{-1}\varphi\big({}^g b\big) = \varphi(\delta)^{-1}\big(x^{-1}\varphi(b)\big)\varphi\big({}^h\delta\big).$$

We are now in a position to compute the secret part of the shared key, using Eq. (4):

$$(xd)p\big(x^{-1}\varphi\big({}^g b\big)\big) = dp\varphi\big({}^g b\big).$$

The attack is complete.

4. Cryptanalysis of CBKAP

Anshel, Anshel, Goldfeld, and Lemieux propose in [2] an efficient concrete realization which they name *Colored Burau Key Agreement Protocol* (*CBKAP*). We give the details, and then describe how our cryptanalysis applies in this case.

4.1. CBKAP

CBKAP is the Eraser Key Agreement scheme in the following particular case. Fix positive integers n and r, and a prime number p.

(1) $G = S_n$, the symmetric group on the n symbols $\{1, \ldots, n\}$. S_n acts on $M = \mathrm{GL}_n(\mathbb{F}_p(t_1, \ldots, t_n))$ by permuting the variables $\{t_1, \ldots, t_n\}$.

(2) $N = \mathrm{GL}_n(\mathbb{F}_p)$.

(3) $M \rtimes S_n$ is the subgroup of $\mathrm{GL}_n(\mathbb{F}_p(t_1, \ldots, t_n)) \rtimes S_n$, generated by $(x_1, s_1), \ldots, (x_{n-1}, s_{n-1})$, where s_i is the transposition $(i, i+1)$, and

$$x_1 = \begin{pmatrix} -t_1 & 1 & & \\ 0 & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}; \qquad x_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & 0 & 0 & & \\ & & t_i & -t_i & 1 & & \\ & & 0 & 0 & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

for $i = 2, \ldots, n-1$. Only the ith row of x_i differs from the corresponding row of the identity matrix. A direct calculation shows that (x_i, s_i) commutes with (x_j, s_j) when $|i - j| > 1$, and that

$$(x_i, s_i)(x_{i+1}, s_{i+1})(x_i, s_i) = (x_{i+1}, s_{i+1})(x_i, s_i)(x_{i+1}, s_{i+1}).$$

Thus, the *colored Burau group* $M \rtimes S_n$ is a representation of Artin's braid group B_n, determined by mapping each Artin generator σ_i to (x_i, s_i), $i = 1, \ldots, n-1$.[7]

(4) $\varphi : M \to \mathrm{GL}_n(\mathbb{F}_p)$ is the evaluation map obtained by replacing each variable t_i by a fixed element $\tau_i \in \mathbb{F}_p$.

[7] Additional details on the colored Burau group can be found in, e.g., [18].

(5) $C = D = \mathbb{F}_p(\kappa)$ is the group of nonzero matrices of the form

$$\ell_1\kappa^{j_1} + \cdots + \ell_r\kappa^{j_r},$$

with $\kappa \in \mathrm{GL}_n(\mathbb{F}_p)$ a matrix of order $p^n - 1$, $\ell_1, \ldots, \ell_r \in \mathbb{F}_p$, and $j_1, \ldots, j_r \in \mathbb{Z}$.

Remark 6. Let $f(x)$ be the characteristic polynomial of κ. Then $x \mapsto \kappa$ induces an isomorphism from $\mathbb{F}_{p^n} = \mathbb{F}[x]/\langle f(x)\rangle$ onto $C = D$, that is, C and D are the image of $\mathbb{F}^*_{p^n}$ in $\mathrm{GL}_n(\mathbb{F}_p)$, or in other words, the nonsplit torus in $\mathrm{GL}_n(\mathbb{F}_p)$.

Commuting subgroups of $M \rtimes G$ are chosen once, by a trusted party, as follows:

(1) Fix $I_1, I_2 \subseteq \{1, \ldots, n-1\}$ such that for all $i \in I_1$ and $j \in I_2$, $|i - j| \geqslant 2$. $|I_1|$ and $|I_2|$ are both $\leqslant n/2$.
(2) Define $L = \langle\sigma_i\colon\ i \in I_1\rangle$ and $U = \langle\sigma_j\colon\ j \in I_2\rangle$, subgroups of B_n generated by Artin generators.
(3) L and U commute element-wise. Add to both groups the central element Δ^2 of B_n.
(4) Choose a random $z \in B_n$.
(5) Choose $w_1, \ldots, w_k \in zLz^{-1}, v_1, \ldots, v_k \in zUz^{-1}$, each a product of t generators (t is a parameter of the scheme). Transform them into Garside left normal form, and remove all even powers of Δ. Reuse the names $w_1, \ldots, w_k, v_1, \ldots, v_k$ for the resulting braids.
(6) Let $\rho : B_n \to M \rtimes S_n$ be the colored Burau representation function. A, B are the subgroups of $\rho(zLz^{-1}), \rho(zUz^{-1})$ generated by $\rho(w_1), \ldots, \rho(w_k)$, and by $\rho(v_1), \ldots, \rho(v_k)$, respectively.
(7) $w_1, \ldots, w_k, v_1, \ldots, v_k$ are made public.

To carry out our attack, it suffices to assume that the image in the colored Burau group of one of the sets $\{w_1, \ldots, w_k\}$ or $\{v_1, \ldots, v_k\}$, is given.

4.1.1. Parameter settings and efficiency

These issues are discussed in detail in [2]. For the parameters proposed there, it is shown that CBKAP can be implemented efficiently, even on small devices as RFID tags. However, we are interested in the more general question, whether *some* parameters may make CBKAP secure. E.g., CBKAP can be implemented on standard PC-s with parameters much larger than those proposed in [2], and still be more efficient than the ordinary schemes based on RSA, Diffie–Hellman in $\mathbb{Z}_p^*$, or elliptic curves. We will show that even for such large parameters, CBKAP can be broken.

4.2. The attack

Assumption 3, that N is a subgroup of $\mathrm{GL}_n(\mathbb{F})$ for some field $\mathbb{F}$, is a part of the definition of CBKAP. We consider the remaining ones.

4.2.1. Regarding Assumption 2

This assumption amounted to: It is possible to generate, efficiently, an element $(\alpha, \sigma) \in A$ such that the order o of σ is smaller than that of (α, σ).

In the notation of Section 4.1, $\{i, i+1\colon\ i \in I_1\}$ decomposes to a family $\mathcal{I}$ of maximal intervals $[i, \ell] = \{i, i+1, \ldots, \ell\}$, and $\sum_{[i,\ell]\in\mathcal{I}} \ell - i + 1 \leqslant n/2$. Now

$$U = \langle\Delta^2\rangle \oplus \bigoplus_{[i,\ell]\in I} B_{\ell-i+1}.$$

Each considered s is a permutation induced by the braid $\Delta^{2m}zwz^{-1}$ with $w \in L$. Let $\pi : B_n \to S_n$ be the canonical homomorphism. Then

$$s = \pi\left(\Delta^{2m}zwz^{-1}\right) = \pi\left(\Delta^2\right)^m \pi(z)\pi(w)\pi(z)^{-1} = \pi(z)\pi(w)\pi(z)^{-1},$$

is conjugate to $\pi(w)$. On each component, this is a product of many random transpositions, and is therefore an almost uniformly-random permutation on that component. We therefore have the following:

(1) $U/\langle\Delta^2\rangle$ decomposes into a direct product of braid groups, whose indices do not sum up to more than $n/2$.
(2) $\pi(U)$ decomposes into a direct product of symmetric groups, whose indices do not sum up to more than $n/2$.
(3) For generic[8] $(a,s) \in A$, $\pi(z)^{-1}s\pi(z)$ is generic on each part of the mentioned decomposition.

The probability that the order of a random permutation in S_n is $\leqslant n$ is $O(1/\sqrt[4]{n})$ [6]. Thus, we can find an element $(a,s) \in A$ with s of order $\leqslant n$ by generating (roughly $\sqrt[4]{n}$) elements $(a,s) \in A$, until the order of s is as required.

On the other hand, the element (a,s) is a representation of an element of the braid group, which is known to be torsion-free [19]. While it may be that the representation used here is not faithful,[9] it is very unlikely that (a,s) could have finite order.

The remainder of this paper is dedicated to Assumption 4.

5. Membership search in generic permutation groups

For the second phase of our attack, we need to find a short expression of a given permutation from G in terms of given "random" permutations. In CBKAP, the group G typically has the form $\pi^{-1}H\pi \leqslant S_n$, where $\pi \in S_n$, H is $S_{n/2}$ or $A_{n/2}$, and H is embedded in S_n in a natural way (supported by the $n/2$ higher indices). The conjugation is just relabeling of the indices $1,\dots,n$. Thus, we may reduce the problem to the case $G = S_n$. Modifications of the algorithm can be made, that will make it applicable to any (conjugation of) direct product of groups of the form A_n or S_n.

For *concrete* generators, the problem of finding short expressions for given permutations is well known, and in similar form occurs in the analysis of the Rubik's cube and other puzzles. The best known heuristics for solving it in these cases are based on Minkwitz's algorithms [17], and are incapable of managing Problem 7 for random $s_1,\dots,s_k \in S_n$ where n is large (say, $n \geqslant 128$), as our experiments below show.

Problem 7. Given random $s_1,\dots,s_k \in S_n$ and $s \in \langle s_1,\dots,s_k\rangle$, express s as a short product of elements from $\{s_1,\dots,s_k\}^{\pm 1}$.

In Problem 7, *short* could mean of polynomial length, or of length manageable by the given computational power as explained above. In any case, the length is the number of letters in the expression, and not the length of a compressed version of the expression. This limitation comes from the intended application, where elements of the infinite monoid require storage space which grows with multiplication, and circumventing this problem by performing one $\star$ multiplication for each letter in the word makes it impossible to square in a single operation. If the word is too long (e.g., of the form $a^{(2^{64})}$ for a single generator a), the second phase of the attack becomes infeasible.

Much work was carried out on this problem, by Babai, Beals, Hetyei, Hayes, Kantor, Lubotzky, Seress, and others (see [7,5,6] and references therein). The works of Babai, Beals, Hayes, and Seress [6,5] imply that there is a Las Vegas algorithm for Problem 7, producing expressions of length $n^7(\log n)^{O(1)}$. This remarkable result solves our problem for moderately small values of n. However, for $n \geqslant 128$, the resulting expression is too long to be practical. Our algorithm may be viewed as a (substantial) heuristic simplification of the algorithms induced by their works.

A classical result of Dixon [10] tells that two random elements of S_n, almost always generate A_n (if all generators are even permutations) or S_n (otherwise). Babai proved that getting A_n or S_n happens

[8] By "generic" we mean a typical element with respect to the relevant distribution.

[9] It is open whether the colored Burau representation is faithful, even without reduction of the integers modulo p.

in probability $1 - 1/n + O(1/n^2)$ [4]. Moreover, experiments show that this probability is very close to $1 - 1/n$ even for small n, i.e., the $O(1/n^2)$ is negligible also for small n. This generalizes to arbitrary k, as follows.

Theorem 8 *(Dixon). The asymptotic probability that k random elements of S_n generate A_n or S_n is roughly $1 - n^{-k+1}$.*

Since we do not know of a reference for a proof, we include a proof, suggested to us by Dixon.

Proof of Theorem 8. In Section 4 of [11] it is shown that the probability that k random permutations generate a transitive group is roughly $1 - n^{-k+1}$.

The proof of Lemma 2 in [10] can be modified to show that the proportion of pairs (x, y) with $x, y \in S_n$ which are contained in an imprimitive group (not necessarily generating the imprimitive group) is at most $2^{-n/4}$. Hence, the proportion of k-tuples contained in an imprimitive group is bounded by $2^{-n/4}$ for all $k > 1$.

Theorem 2.8 of Babai's paper [4] states that the probability that k random permutations generate a primitive group different from A_n or S_n is smaller than $(n^{\sqrt{n}}/n!)^{k-1}$ (generalizing his theorem when $k = 2$), which is exponentially small. □

Given that we obtain A_n or S_n, the probability of the former case is 2^{-k}. However, since $k = 2$ is of classical interest, we do not neglect this case. Thus, for randomly chosen permutations Problem 7 reduces (with a small loss in probability) to the following one.

Problem 9.

(1) Given random $s, s_1, \dots, s_k \in A_n$, express s as a short product of elements from $\{s_1, \dots, s_k\}^{\pm 1}$.
(2) Given random $s, s_1, \dots, s_k \in S_n$ with some $s_i \notin A_n$, express s as a short product of elements from $\{s_1, \dots, s_k\}^{\pm 1}$.

A solution of Problem 9(1) implies a solution of Problem 9(2): Let $I = \{i\colon\ s_i \notin A_n\}$. $I \neq \emptyset$. Fix $i_0 \in I$, and for each $i \in I$, replace the generator s_i with the generator $s_{i_0} s_i \in A_n$. Then $\{s_{i_0} s_i\colon\ i \in I\} \cup \{s_i\colon\ i \notin I\}$ is a set of k nearly random elements of A_n (cf. [6]). If $s \in A_n$, use (1) to obtain a short expression of s in terms of the new generators. This gives an expression in the original generators of at most double length. Otherwise, $s_{i_0} s \in A_n$ and its expression gives an expression of s in terms of the original generators.

Thus, in principle one may restrict attention to Problem 9(1). However, we do not take this approach, since we want to make use of transpositions when we can.

5.1. The algorithm

5.1.1. Conventions

(1) During the algorithm's execution, the expressions of some of the computed permutations in terms of the original generators should be stored. We do not write this explicitly.
(2) The statement *for each* $\tau \in \langle S\rangle$ means that the elements of $\langle S\rangle$ are considered one at a time, by first considering the elements of $S^{\pm 1}$, then all (free-reduced) products of two elements from $S^{\pm 1}$, etc. (a breadth-first search), until an *end* statement is encountered.
(3) For $s \in (S^{\pm 1})^*$, $\operatorname{len}(s)$ denotes the length of s as a free-reduced word. s is identified in the usual way with the permutation which is the product of the letters in s.

We are now ready to describe the steps of our algorithm. We do not consider the question of optimal values for the parameters and other optimizations. This is left for future investigation.

Input: $G = S_n$ or A_n; generators $s_1, \dots, s_k$ of G; $s \in G$.

Initialization:

$$c = \begin{cases} 2, & G = S_n, \\ 3, & G = A_n, \end{cases}$$

C is the set of c-cycles in a canonical expression of s as a product of c-cycles.

Step 1: *Find a short c-cycle in $\langle s_1, \dots, s_k \rangle$.*

```
For each $\tau \in \langle s_1, \dots, s_k \rangle$:
  If there is $m \in \{1, \dots, n\}$ such that $\tau^m$ is a $c$-cycle:
    $\mu \leftarrow \tau^m$;
    End Step 1.
```

The result μ of Step 1 is forwarded to the next step.

Step 2: *Find short expressions for additional c-cycles.*

```
$A_0 \leftarrow \{\mu\}$;
For $l = 1, 2, \dots$ :
  $A_l \leftarrow \emptyset$;
  For each $i \in \{1, \dots, k\}$, each $\epsilon \in \{-1, 1\}$, and each $a \in A_{l-1}$:
    If $s_i^{-\epsilon} a s_i^{\epsilon} \notin A_0 \cup \cdots \cup A_l$, add $s_i^{-\epsilon} a s_i^{\epsilon}$ to $A_l$;
  When $C \subseteq A_0 \cup \cdots \cup A_l$:
      End Step 2.
```

Final step: *Find a short expression for s.*

Use the expressions of the c-cycles in C to get an expression of s in terms of the original generators.

Remark 10 *(Positive expressions).* If one seeks for a *positive* expression for s in terms of $\{s_1, \dots, s_k\}$, we can repeatedly activate Step 1, consider only words $\tau \in S^*$, to generate enough c-cycles to present s. This algorithm is more time consuming this way.

6. Analysis of the generic membership search algorithm

6.1. Asymptotic analysis

We provide an asymptotic (in n) analysis of the time complexity and the final expression length, for the generic membership search algorithm (Section 5.1), modulo a probabilistic conjecture, which we later support by heuristic reasoning as well as extensive experiments.

6.1.1. Step 1

Conjecture 11 *(Minimal Cycle Conjecture). Let S be a set of k elements of S_n, each chosen independently, according to the uniform distribution on S_n. Let $c = 3$ if all elements of S are even, and 2 otherwise.*

Consider the following list: The elements of $S^{\pm 1}$, followed by all products of two of elements of $S^{\pm 1}$, followed by all products of three elements of $S^{\pm 1}$, etc.

Then, almost always,[10] *there is among the first n^2 elements of the list an element τ such that τ^m is a c-cycle, for some $m \leqslant n$.*

[10] That is, with probability approaching 1 as $n \to \infty$.

In short, the Minimal Cycle Conjecture asserts that in Step 1, almost always, at most n^2 permutations are considered.

Lemma 12. *Given an element $\tau \in S_n$, deciding whether there is $m \leqslant n$ such that τ^m is a c-cycle can be done in time $O(n)$. Finding the minimal such m, if it exists, can be done in time $O(n \log n)$.*

Proof. Let $\ell_1, \ell_2, \dots, \ell_k$ be the cycle lengths in the cycle decomposition of τ, which can be computed in linear time. There is m as required if, and only if, there is a unique $i \leqslant k$ such that $\ell_i = c$, and for each $j \leqslant k$ different from i, ℓ_j is relatively prime to c, which can be verified in time $O(\log \ell_j)$.[11] As $\sum_{j \neq i} \log \ell_j \leqslant \sum_j \ell_j \leqslant n$, the whole procedure is $O(n)$.

The minimal power m, if it exists, is $\operatorname{lcm}(\ell_j\colon\ j \neq i)$. This can be computed by partitioning the list into pairs, computing the lcm of each pair to obtain a half length list, and doing the same for this list. After $O(\log k)$ steps, we obtain the lcm. Each step requires roughly $\sum \log \ell_i$ operations, and overall we have $(\log k) \sum \log \ell_i$ which is $O(n \log n)$. □

Corollary 13. *Assume the Minimal Cycle Conjecture. Then, almost always, Step 1 produces a c-cycle μ, expressed as a product of $O(n \log n)$ elements of $S^{\pm 1}$, in time and space $O(n^3)$ (or, alternatively, in time $O(n^3 \log n)$ and space $O(n)$).*

Proof. To produce the list of n^2 permutations, we have to compute n^2 products of permutations (the product of an already computed permutation and an element of $S^{\pm 1}$), each requiring n operations. After each such multiplication, we check whether the result τ has the property that τ^m is a c-cycle for some $m \leqslant n$. This is done by Lemma 12.

In summary, we have n^2 steps, each consisting one multiplication of permutations, and one linear time decision. Thus, the overall time complexity is $O(n^3)$.

If τ is among the first n^2 elements of the sequence, then τ is a product of at most $\log_{2k}(n^2) = O(\log n)$ elements of $S^{\pm 1}$, and therefore τ^m is expressed as a product of at most $O(m \log n)$, which is $O(n \log n)$.

Alternatively, one can compute, for each new word in the generators, the whole product. This increases the time complexity to $O(n^3 \log n)$, but reduces the space complexity to $O(n)$. □

6.1.2. Step 2

Let S be a set of k elements of S_n, each chosen independently, according to the uniform distribution on S_n. Consider the graph $G_{n,k,c}$ with vertices all c-cycles, such that there is an edge between u, v if and only if there is $r \in S^{\pm 1}$ with $r^{-1}ur = v$. This graph has $n!/(n-c)!c$ vertices and is $2k$-regular. In the worst case we have to compute in Step 2 all vertices of this graph before this procedure terminates. For every $a \in A_l$, $l \geqslant 1$, keeping track of its predecessor in A_{l-1}, Step 2 computes a spanning tree of this graph, rooted at μ. Let ℓ be the value of l at the termination of Step 2. ℓ is the height of our tree, and the diameter d of this graph satisfies $\ell \leqslant d \leqslant 2\ell$.

For each $a \in A_l$, $1 \leqslant l \leqslant \ell - 1$, $2k-1$ conjugations are performed. Indeed for each $a \in A_l$, by considering which conjugator led to it, the inverse conjugator will not lead to anything new and is thus not performed. Only for the root we perform $2k$ conjugations, but no one for all $a \in A_\ell$. Thus, the overall number of conjugations in this step is bounded above by

$$2k|A_0| + (2k-1)\sum_{l=1}^{\ell-1} |A_l| = 1 + (2k-1)\sum_{l=0}^{\ell} |A_l| - (2k-1)|A_\ell| \leqslant \frac{(2k-1)n!}{(n-c)!c} - 2k + 2.$$

Remark 14. In fact, as S generates G, it also generates it as a monoid, and thus it suffices to consider conjugations by positive generators only, so that the overall number of conjugations is less than

[11] Considering standard CPU operations as requiring constant time.

$(k-1)n!/(n-c)!c-k+2$. Moreover, with high probability one can restrict attention to just two generators generating G, so the number of conjugations becomes less than $n!/(n-c)!c < n^c/c$. Here, we have to consider a digraph rather than a graph. The vertices are again all c-cycles and there is a directed edge from u to v iff $r^{-1}ur = v$ for some $r \in S$. However, the diameter of this digraph is greater or equal than that of $G_{n,k,c}$.

Corollary 15. *The running time of Step* 2 *is* $O(n^2)$ *if* $G = S_n$ *and* $O(n^3)$ *if* $G = A_n$.

Proof. Let $c = 2$ if $G = S_n$ and 3 if $G = A_n$. Each conjugation of a c-cycle requires $c \leqslant 3$ operations:

$$p(i\ j)p^{-1} = \big(p(i)\ p(j)\big),$$
$$p(i\ j\ k)p^{-1} = \big(p(i)\ p(j)\ p(k)\big).$$

Thus, the running time is a small constant times the number of conjugations, which is bounded by $(2k-1)n^c/c$ (or less if we work according to Remark 14). □

For each permutation σ encountered during our algorithm, let $\operatorname{len}(\sigma)$ be the length of its expression as a product of the given permutations $s_1, \dots, s_k$ and their inverses. Recall that μ is the output of Step 1. Then for each c-cycle $\sigma \in A_0 \cup \cdots \cup A_\ell$,

$$\operatorname{len}(\sigma) \leqslant \operatorname{len}(\mu) + 2\ell.$$

Corollary 16. *Using the above notation, the length of the obtained expression for s is smaller than* $n/(c-1) \cdot (\operatorname{len}(\mu) + 2\ell)$.

Proof. s is a product of at most $n/(c-1)$ c-cycles. □

The following theorem consists of Theorems 2.2 and 3.3 of [12].

Theorem 17.

(1) *Fix* $k \geqslant 2$, $c \geqslant 1$ *and a real* $\epsilon > 0$.
Let S *be a set of* k *elements of* S_n, *each chosen independently, according to the uniform distribution on* S_n. *Let* $D_{n,k,c}$ *be the digraph with whose vertices are the* c*-tuples of distinct elements of* $\{1, \dots, n\}$, *and where there is an arrow from* $(a_1, \dots, a_c)$ *to* $(b_1, \dots, b_c)$ *if and only if* $(b_1, \dots, b_c) = (s(a_1), \dots, s(a_c))$ *for some* $s \in S$.
Then, almost always, $D_{n,k,c}$ *is an* α*-expander, for*

$$\alpha = \big((1-\epsilon)/2\big)\big(1 - (\sqrt{2k-1}/k)^{1/(1+c)}\big).$$

(2) *The diameter of an* α*-expander with* v *vertices is smaller than* $2(1 + \log_{1+\alpha} v)$.

Corollary 18. *For* $k \geqslant 2$, $c \geqslant 1$, *the diameter of the graph* $G_{n,k,c}$ *is almost always bounded by* $2(1 + c\log_{1+\alpha}(n))$ *with* $\alpha = (1 - (\sqrt{2k-1}/k)^{1/(1+c)})/2$.

Proof. Consider the equivalence relation $\sim$ on the set of c-tuples of distinct elements of $\{1, \dots, n\}$, which identifies tuples if each is a cyclic rotation of the other. The quotient digraph $D_{n,k,c}/\sim$ is exactly the digraph mentioned in Remark 14. Thus, ℓ is smaller than its diameter, which is smaller than the diameter of $D_{n,k,c}$. By Theorem 17, the latter is smaller than

$$2\big(1 + \log_{1+\alpha}\big(n^c\big)\big) = 2(1 + c\log_{1+\alpha} n). \qquad □$$

Corollary 19. *Assume the Minimal Cycle Conjecture. The length of the obtained expression for s is $O(n^2 \log n)$.*

Proof. By Corollary 16, the length of the obtained expression for s is $O(n(\mathrm{len}(\mu) + 2\ell))$. By Corollary 13, $\mathrm{len}(\mu)$ is $O(n \log n)$. By Corollary 18, the diameter of our graph $G_{n,k,c}$ is $O(\log n)$, and in particular so is ℓ. □

6.2. Heuristic evidence and estimation

6.2.1. Step 1

The following terminology and lemma will make the proof of the subsequent theorem shorter. The *cycle structure* of a permutation $s \in S_n$ is the sequence $(n_1, n_2, \ldots)$ of lengths of cycles of s which are not fixed points. Let $\sigma^n_{(n_1,\ldots,n_k)}$ denote the number of elements of S_n with cycle structure $(n_1, \ldots, n_k)$.

Lemma 20. *For distinct $n_1, \ldots, n_k$: $\sigma^n_{(n_1,\ldots,n_k)} = \frac{n!}{(n-(n_1+\cdots+n_k))! \cdot n_1 \cdots n_k}$.*

Proof. First choose the $n_1 + \cdots + n_k$ elements which will occupy the cycles and consider all their permutations, and then divide out cyclic rotation equivalence, to get

$$\binom{n}{n_1 + \cdots + n_k} \cdot (n_1 + \cdots + n_k)! \cdot \frac{1}{n_1 \cdots n_k}.$$

This is equal to $\sigma^n_{(n_1,\ldots,n_k)}$. □

Proposition 21. *Let c be 2 if $G = S_n$, and 3 if $G = A_n$. For random $\tau \in G$, the probability that there is $d \in \{1, \ldots, n\}$ such that τ^d is a c-cycle is greater than $1/cn$.*

Proof. In fact, we give better bounds for most values of n. We consider the probabilities to have cycle structures $(n-d, c)$ or $(n-d, e, c)$ for appropriate d, such that if τ has such a cycle structure, then τ^{n-d} is a c-cycle. The restrictions on the cycle structures are as follows.

(1) c does not divide $n - d$; and
(2) e divides $n - d$ (in the case $(n-d, e, c)$).

In the case $G = A_n$, we also must have that the cycle structure is possible in A_n:

(3) $n - d$ is odd (in the case $(n-d, 3)$);
(4) $n - d + e$ is even (in the case $(n-d, e, 3)$).

Assuming these restrictions, we compute the probabilities of these cycle structures using Lemma 20. In S_n, the probability for $(n-d, 2)$ is

$$\frac{1}{|S_n|} \cdot \sigma^n_{(n-d,2)} = \frac{1}{(d-2)! \cdot (n-d) \cdot 2} > \frac{1}{(d-2)! \cdot 2n}.$$

In A_n, the probabilities for $(n-d, 3)$ and $(n-d, e, 3)$ are

$$\frac{1}{|A_n|} \cdot \sigma^n_{(n-d,3)} = \frac{2}{(d-3)! \cdot (n-d) \cdot 3} > \frac{2}{(d-3)! \cdot 3n},$$
$$\frac{1}{|A_n|} \cdot \sigma^n_{(n-d,e,3)} = \frac{2}{(d-e-3)! \cdot (n-d) \cdot e \cdot 3} > \frac{2}{(d-e-3)! \cdot 3en},$$

respectively. We now consider some possible cycle structure with at most one cycle of each length, and describe the restrictions they pose on n and their probabilities.

For $G = S_n$, we have the following.

n mod 2	Cycle structure	Prob.	Accumulated probability
0	$(n-3, 2)$	$1/2n$	
	$(n-5, 2)$	$1/12n$	$7/12n$
1	$(n-2, 2)$	$1/2n$	
	$(n-4, 2)$	$1/4n$	$3/4n$

For $G = A_7$, we can compute directly that the cycle structure $(2, 2, 3)$ has probability $1/12$, which is greater than $1/3 \cdot 7$, as required. For all other n, we have the following.

n mod 6	Cycle structure	Prob.	Accumulated probability
0	$(n-5, 3)$	$1/3n$	$1/3n$
1	$(n-5, 2, 3)$	$1/3n$	
	$(n-6, 3)$	$1/9n$	$4/9n$
2	$(n-3, 3)$	$2/3n$	$2/3n$
3	$(n-4, 3)$	$2/3n$	
	$(n-5, 2, 3)$	$1/3n$	$1/n$
4	$(n-3, 3)$	$2/3n$	
	$(n-5, 3)$	$1/3n$	
	$(n-6, 2, 3)$	$1/3n$	$4/3n$
5	$(n-4, 3)$	$2/3n$	
	$(n-6, 3)$	$1/9n$	
	$(n-7, 2, 3)$	$1/6n$	$17/18n$

This completes the proof. □

Corollary 22. *Let c be 2 if $G = S_n$, and 3 if $G = A_n$. Execute Step 1 with random elements $\tau \in G$ instead of the enumerated ones. The probability that it does not end before considering λn permutations is smaller than $e^{-\lambda/c}$.*

Proof. By Corollary 21, the probability of not obtaining a c-cycle for λn randomly chosen $\tau \in G$ is at most

$$\left(1 - \frac{1}{cn}\right)^{\lambda n} = \left(\left(1 - \frac{1}{cn}\right)^{cn}\right)^{\frac{\lambda}{c}} < \left(e^{-1}\right)^{\frac{\lambda}{c}} = e^{-\frac{\lambda}{c}}. \qquad \square$$

Example 23. Let c be 2 if $G = S_n$, and 3 if $G = A_n$, and $\lambda = c\lambda_0 \log n$ for some constant λ_0. Then the probability in Corollary 22 is smaller than

$$e^{-\frac{c\lambda_0 \log n}{c}} = n^{-\lambda_0}.$$

This shows that if we use, in Step 1 of our algorithm, random elements instead of the enumerated ones, then, almost always, this step halts after the consideration of at most $3n \log n$ permutations. This is much smaller than the n^2 in the Minimal Cycle Conjecture 11. We conjecture that this increase from $n \log n$ to n^2 remedies for the fact that in the conjecture, the considered elements are *not* independent. Below, we provide experimental evidence for that.

6.2.2. The expression's length

Using Corollary 18, we can derive a rough upper bound on the *average* length of the expression provided by the generic membership search algorithm, assuming the Minimal Cycle Conjecture 11. According to this conjecture, Step 1 uses on average less than n^2 permutations until finding a good one τ. If τ is the n^2th permutation in our breadth-first enumeration of $\langle s_1, \dots, s_k \rangle$, then its length d as a word in the generators satisfies

$$(2k-1)^{d-1} \leqslant 2k(2k-1)^{d-2} \leqslant n^2.$$

Thus

$$\operatorname{len}(\tau) \lesssim \frac{2}{\log(2k-1)} \cdot \log n.$$

Then, μ is at most an nth power of τ. Thus on average,

$$\operatorname{len}(\mu) \lesssim \frac{2}{\log(2k-1)} \cdot n \log n.$$

By Corollary 18, ℓ is on average much smaller than $\operatorname{len}(\mu)$, and thus by Corollary 16, the average length of the resulting expression is roughly bounded by

$$\frac{2}{(c-1)\log(2k-1)} \cdot n^2 \log n = \begin{cases} \frac{2}{\log(2k-1)} \cdot n^2 \log n, & G = S_n, \\ \frac{1}{\log(2k-1)} \cdot n^2 \log n, & G = A_n. \end{cases}$$

7. Experimental results

7.1. The full attack

We have implemented our full attack on CBKAP, and tested it against a large number of parameter settings, including the suggested ones, smaller ones, much larger ones, and mixed settings (some parameters are small and some are large). The full attack succeeded to extract the shared key out of the public information correctly, in *all* tested cases, including those in which the generated subgroup of $S_{n/2}$ was neither $S_{n/2}$ nor $A_{n/2}$.

7.2. The generic membership search algorithm

We then moved to a systematic examination of the generic membership search algorithm. This algorithm worked efficiently and successfully in all experiments, and its time, space and length of output were all surprisingly close to the estimations computed in the previous sections. The most difficult case for this algorithm is where there are only $k = 2$ random generators s_1, s_2. Thus, we have made a large battery of experiments for $k = 2$.

We make the following conventions. The constant c is 2 if $G = S_n$, and 3 if $G = A_n$. For each $n = 8, 16, 32, 64, 128, 256$, we have conducted at least 1000 independent experiments altogether. As $k = 2$, in about 750 of these experiments $\langle s_1, s_2 \rangle = S_n$, and in about 250, $\langle s_1, s_2 \rangle = A_n$. The few cases where neither S_n nor A_n were generated were ignored.

Each of these many samples suggests a value for the considered parameter. We thus present the minimum, average, and maximum observed values (with the average boldfaced).

Table 1
Ratios for the number of permutations in Step 1.

n	8	16	32	64	128	256
S_n	0.06	0.03	0.02	0.01	0	0
	2.26	**2.53**	**3.47**	**5.05**	**5.4**	**8.55**
	112.13	45.88	25.22	102.81	52.62	77.31
A_n	0.04	0.02	0.01	0.01	0.01	0
	0.51	**0.51**	**1.35**	**1.28**	**2.56**	**1.9**
	7.63	4.15	15.5	7.73	12.65	17.5

Table 2
Ratios for the length of the final expression.

n	8	16	32	64	128	256
S_n	0.07	0.11	0.10	0.08	0.1	0.1
	0.31	**0.45**	**0.52**	**0.62**	**0.63**	**0.68**
	1.14	0.97	0.98	1.06	0.99	0.95
A_n	0.11	0.08	0.08	0.03	0.02	0.03
	0.4	**0.4**	**0.53**	**0.54**	**0.62**	**0.59**
	0.78	0.87	1.07	0.86	0.9	0.87

Table 3
Expression lengths using the generic membership search algorithm.

n	8	16	32	64	128	256
S_n	16	148	674	2603	14 357	65 063
	76	**580**	**3331**	**19 078**	**91 120**	**450 450**
	275	1258	6344	33 015	143 344	631 306
A_n	13	54	248	504	1640	9258
	48	**261**	**1698**	**8328**	**44 739**	**195 534**
	94	564	3454	13 328	65 354	286 628

7.2.1. Step 1

The upper bound n^2 in the Minimal Cycle Conjecture 11 turns out to be an over-estimation for the number of permutations considered in Step 1. Indeed, except for few cases in $n = 8$, *none of our experiments exceeded this bound*. Thus, we present in Table 1 the ratio between the number of permutations actually considered in Step 1 and the estimation cn, which is what one would obtained if the permutations were independent.

7.2.2. Length of the final expression

For $k = 2$, the average length of the final expression of the given permutation is estimated in Section 6.2.2 to be, roughly, below

$$\frac{2}{(c-1)\log(2k-1)} \cdot n^2 \log n = \begin{cases} \frac{2}{\log 3} \cdot n^2 \log n, & G = S_n, \\ \frac{1}{\log 3} \cdot n^2 \log n, & G = A_n. \end{cases}$$

($\log(3) \approx 1.1$). Table 2 shows that this estimation is surprisingly good, and that in fact, the true resulting length is on average better than this bound.

The actual lengths of the expressions produced for the given permutations are given in Table 3. For clarity, the average lengths are rounded to the nearest integer.

For comparison with earlier methods, we looked for expressions of permutations as short products, using GAP's [13] Schreier–Sims based algorithm (division off stabilizer chains), which uses optimizations similar to Minkwitz's [17]. Here, we have 100 experiments for S_n and 100 experiments for A_n.

Table 4
Expression lengths using previous heuristics (Schreier–Sims–Minkwitz).

n	8	16	24	28	32
S_n	5	102	432	1047	∞
	22	**255**	**8039**	**345 272**	∞
	42	418	350 846	32 729 135	∞
A_n	0	95	549	913	∞
	18	**238**	**4101**	**59 721**	∞
	29	413	35 447	4 012 292	∞

Already for $n = 32$, the routines went out of memory in about $1/3$ of the cases for A_n, and in about $2/3$ of the cases for S_n. Thus, we also checked $n = 24$ and $n = 28$ ($n = 28$ seems to be the largest index which the routines handle well). The resulting lengths are shown in Table 4, where ∞ means "out of memory in too many cases".

We can see that Schreier–Sims methods are better than ours only for small values of n, and that they are not applicable for large n, where our algorithm is easily applicable. Also, note the large difference between the minimal and the maximal obtained lengths. Contrast this with the results in Table 3.

8. Possible fixes of the Algebraic Eraser and challenges

As we have demonstrated, no choice of the security parameters makes the Algebraic Eraser immune to the attack presented here, as long as the keys are generated by standard distributions.

A possible fix may be to change the group S into one whose elements do not have short expressions in terms of its generators. This may force the attacker to attack the original matrices (whose entries are Laurent polynomials in the variables t_i) directly, using linear algebraic methods similar to the ones presented here. It is not clear to what extent this can be done.

The most promising way to foil our attacks, at least on a small fraction of keys, may be to use very carefully designed distributions, which are far from standard ones. Following our attack, Dorian Goldfeld and Paul Gunnels devised a distribution for which the equations in phase 1 of the attack have a huge number of solutions, most of which not leading to the correct shared key [14].

Another option would be to work in semigroups, and use noninvertible matrices. This may foil the first phase of our attack.

The generic membership search algorithm is of interest beyond its applicability to the Algebraic Eraser. We have demonstrated, based on our Minimal Cycle Conjecture 11, that this algorithm easily solves instances with random permutations, in groups of index which is intractable when using previously known techniques like those in [17]. Our extensive experiments, reported above, support this assertion.

The most interesting direction of extending the present work is proving the Minimal Cycle Conjecture, even with $O(n^2)$ instead of our n^2. In fact, proving any polynomial bound would imply that the diameter of S_n is almost always $O(n^2 \log n)$, which would improve considerably the presently known bound $n^7(\log n)^{O(1)}$ on the diameter. Alexander Hulpke has informed us that our methods are similar to ones used for constructive recognition of S_n or A_n. This connection may be useful for the proposed analysis.

Finally, we point out that even without changes, our algorithm applies in many cases not treated here, as the experiments of the full attack reported above show.

Acknowledgments

This research was partially supported by the Oswald Veblen Fund, the Emmy Noether Research Institute for Mathematics, and the Minerva Foundation of Germany. We thank John Dixon for his proof of Theorem 8. We thank László Babai, Dorian Goldfeld, Stephen Miller, Ákos Seress, and Adi Shamir, for fruitful discussions. A special thanks is owed to Martin Kassabov, for comprehensive suggestions which improved the presentation of this paper, and for pointing out to us the present version of Step 2 of our algorithm. This version seems to be folklore, but we have initially used a less efficient variant. We also thank Alexander Hulpke and Stefan Kohl for useful information about GAP. Our full attack on the Algebraic Eraser was implemented using MAGMA [8].

References

[1] I. Anshel, M. Anshel, D. Goldfeld, An algebraic method for public-key cryptography, Math. Res. Lett. 6 (1999) 287–291.
[2] I. Anshel, M. Anshel, D. Goldfeld, S. Lemieux, Key agreement, the Algebraic Eraser$^{\mathrm{TM}}$, and lightweight cryptography, Contemp. Math. 418 (2006) 1–34.
[3] I. Anshel, D. Goldfeld, P. Gunnells, Fast asymmetric encryption using the Algebraic Eraser, in preparation.
[4] L. Babai, The probability of generating the symmetric group, J. Combin. Theory Ser. A 52 (1989) 148–153.
[5] L. Babai, R. Beals, Á. Seress, On the diameter of the symmetric group: polynomial bounds, in: 15th ACM-SIAM SODA, 2004, pp. 1108–1112.
[6] L. Babai, T. Hayes, Near-independence of permutations and an almost sure polynomial bound on the diameter of the symmetric group, in: 16th ACM-SIAM SODA, 2005, pp. 1057–1066.
[7] L. Babai, G. Hetyei, W. Kantor, A. Lubotsky, Á. Seress, On the diameter of finite groups, in: 31st IEEE FOCS, 1990, pp. 857–865.
[8] W. Bosma, J. Cannon, C. Playoust, The Magma algebra system, I: The user language, J. Symbolic Comput. 24 (1997) 235–265.
[9] Colloquium on combinatorial group theory and cryptology webpage, available at http://u.cs.biu.ac.il/~tsaban/CGCColloq.html.
[10] J. Dixon, The probability of generating the symmetric group, Math. Z. 110 (1969) 199–205.
[11] J. Dixon, Asymptotics of generating the symmetric and alternating groups, Electron. J. Combin. 12 (2005) R56.
[12] J. Friedman, A. Joux, Y. Roichman, J. Stern, J.-P. Tillich, The action of a few permutations on r-tuples is quickly transitive, Random Structures Algorithms 12 (1998) 335–350.
[13] The GAP Group, GAP – Groups, Algorithms and Programming, Version 4.4.10, http://www.gap-system.org, 2007.
[14] D. Goldfeld, P. Gunnells, Defeating the Kalka–Teicher–Tsaban linear algebra attack on the Algebraic Eraser, arXiv:1202.0598.
[15] P. Gunnells, On the cryptanalysis of the generalized simultaneous conjugacy search problem and the security of the Algebraic Eraser, arXiv:1105.1141.
[16] K. Ko, S. Lee, J. Cheon, J. Han, S. Kang, C. Park, New public-key cryptosystem using braid groups, in: CRYPTO 2000, in: Lecture Notes in Comput. Sci., vol. 1880, 2000, pp. 166–183.
[17] T. Minkwitz, An algorithm for solving the factorization problem in permutation groups, J. Symbolic Comput. 26 (1998) 89–95.
[18] H. Morton, The multivariate Alexandrov polynomial for a closed braid, Contemp. Math. 233 (1999) 167–172.
[19] K. Murasugi, Seifert fibre spaces and braid groups, Proc. Lond. Math. Soc. 44 (1982) 71–84.
[20] A. Myasnikov, A. Ushakov, Cryptanalysis of the Anshel–Anshel–Goldfeld–Lemieux Key Agreement Protocol, Groups Complex. Cryptol. 1 (2009) 63–75, arXiv:0801.4786.