CRYPTANALYSIS OF MATRIX CONJUGATION SCHEMES

A. D. MYASNIKOV AND A. USHAKOV

Abstract. In this paper we cryptanalyze two protocols: Grigoriev-Shpilrain authentication protocol and Wang et al. public key encryption protocols that use computational hardness of some variations of the conjugacy search problem in noncommutative monoids. We devise a practical heuristic algorithm solving those problems. As a conclusion we claim that these protocols are insecure for the proposed parameter values.

Keywords and phrases: Group-based cryptography, conjugacy search problem, matrix monoids, truncated polynomials.
AMS Classification: 94A60, 68W30.

1. Introduction

The conjugacy search problem plays a special role in group-based cryptography ([11, 12]). Most of the cryptosystems based on groups use one or another variation of that problem. For instance:

- [9] employs conjugacy search problem in braid groups;
- [1, 21] employ simultaneous conjugacy search problem in braid groups;
- [19] employs twisted conjugacy problem;
- [20, 16, 17] employ decomposition problem in different (semi-)groups;
- [15] employs conjugation and exponentiation in matrix groups.

Yet, another protocol based on the conjugacy search problem was recently proposed in [5] by Grigoriev and Shpilrain, who introduced two authentication protocols: one is called the beta protocol and the other is called the full protocol. The security of both protocols relies on the hardness of the conjugacy search problem in a semigroup of of matrices over truncated polynomials. In this paper we test the following claim ([5, page 199]):

> *Our platform semigroup might be the first serious candidate for having generically hard conjugacy search problem.*

Informally, generic case complexity is the complexity on "almost all" of the inputs, see [7, 8]. We prove that the claim is not true and, in fact, the conjugacy search problem is trivial in most of the cases (see Theorem 3). Trying to find difficult instances of the problem we modify the method of key generation. Our modification allows to generate nontrivial instances of the conjugacy search problem, but even that does not make the protocol

Date: December 10, 2012.
The work of the second author was partially supported by NSF grant DMS-0914773.

secure, we show that it can be attacked by a certain heuristic algorithm (see Section 4).

We also discuss a related work by L. Wang et al. (see [22]) which proposes a non-commutative version of Diffie-Hellman type encryption protocol which uses conjugation in noncommutative monoids and demonstrate their inadequacy.

1.1. **The Grigoriev-Shpilrain authentication scheme.** Here we introduce basic notions and formulate the conjugacy search problem together with the main security assumptions used in [5]. Fix $N, m, n, k \in \mathbb{N}$. Grigoriev-Shpilrain authentication protocol uses a semigroup of $n \times n$ matrices over *N-truncated polynomials* with coefficients in $\mathbb{Z}_m$. An N-truncated polynomial over a ring $\mathbb{Z}_m$ is an element of the factor algebra

$$R = \mathbb{Z}_m[x_1, \dots, x_k]/\langle x_{i_1} \cdots x_{i_s} \mid s = N \rangle.$$

Let G be the ring of all $n \times n$ matrices over truncated k-variable polynomials over $\mathbb{Z}_m$. We say that $X \in G$ is *invertible* if there exists $Y \in G$ satisfying $XY = YX = I$, where I is the identity matrix.

Conjugacy search problem (CSP) in a semigroup G: Given elements $A, C \in G$ find an invertible element X satisfying $C = X^{-1}AX$, provided that such an element exists.

Public/private keys in Grigoriev-Shpilrain protocol:

- The *private key* is an invertible element $X \in G$.
- The *public key* is a pair $(A, X^{-1}AX)$, where $A \in G$.

The key generation procedure is discussed in detail in Section 2. To check Alice's identity, Bob runs the following protocol exactly once (cf. Feige-Fiat-Shamir type protocols, [3]).

Algorithm 1. The Grigoriev-Shpilrain authentication protocol (beta version).

1: Bob sends a random $B \in G$ to Alice.
2: Alice responds with $X^{-1}BX$.
3: Bob chooses a random positive word $w(x, y)$ and checks the equality:

$$\mathrm{trace}(w(A, B)) = \mathrm{trace}(w(X^{-1}AX, X^{-1}BX)).$$

4: Authentication is successful if the equality holds.

Obviously this protocol has the completeness property (Alice is able to prove her identity) of proof-systems. The soundness property (the probability of a wrong person proving that (s)he is Alice is negligible) is much less obvious and probably is not satisfied. The zero knowledge property simply does not hold, although the authors are aware of that and do not claim it.

It is stated in [5, page 199] that forgery seems infeasible without finding the provers private key. This is not completely correct because to break this protocol it is sufficient to find any conjugator for the pair $(A, X^{-1}AX)$, i.e., to solve the conjugacy search problem for G. The security of the full version of the Grigoriev-Shpilrain protocol relies on the difficulty of the same conjugacy problem, as it has exactly the same public key pair $(A, X^{-1}AX)$. We omit the description of the full protocol here.

Another claim made in [5, page 199] is:

> *Our platform semigroup might be the first serious candidate for having generically hard conjugacy search problem.*

This claim is the primary target of this paper. We investigate computational hardness of the conjugacy search problem for G and prove that it is easy on most of the inputs under the proposed key distribution.

1.2. **Wang-Wang-Cao-Okamoto-Shao cryptosystem.** Wang et al. in [22] propose several public key encryption schemes based on the CSP over a noncommutative monoid G. The most basic of the protocols works as follows. Fix $A, X \in G$, with X being invertible, and a number $K \in \mathbb{N}$. Public/private keys in Wang et al. protocol are:

- Alice's *private key* is a number $s \in \mathbb{N}$.
- Alice's *public key* is an element $X^{-s}AX^s$.
- Bob's *private key* is a number $t \in \mathbb{N}$.
- Bob's *public key* is an element $X^{-t}AX^t$.

After exchanging public information Alice and Bob can immediately compute the *shared key* $X^{-s-t}AX^{s+t}$. Based on this simple protocol Wang et al. develop two ElGamal-type public key encryption protocols. Security of this scheme is based on the difficulty of the following computational problem:

CSP-based computational Diffie-Hellman problem in G: Given elements A, X, $X^{-s}AX^s$, $X^{-t}AX^t$ in G, for some random $1 \leq s, t \leq K$, compute $X^{-s-t}AX^{s+t}$.

The value K is the main security parameter here and, hence, should be sufficiently large to provide a necessary level of security. That implies that the order of the cyclic subgroup generated by X must be at least K. Similarly we define the associated decision problem:

CSP-based decision Diffie-Hellman problem in G: Given elements A, X, $X^{-s}AX^s$, $X^{-t}AX^t$ in G, for some $1 \leq s, t \leq K$, distinguish elements $X^{-s-t}AX^{s+t}$ and $X^{-l}AX^l$ for some random $l \in [1, K]$ with probability cryptographically-significantly better than $1/2$.

We discuss Wang's et al. protocol in Section 3.

1.3. **Parameter values.** Both papers, [5] and [22], use almost the same set of parameters. They use the semigroup G of 3×3 matrices of 1000-truncated polynomials in 10 variables over $\mathbb{Z}_{11}$ (in [5]) and over $\mathbb{Z}_{12}$ (in [22]). To be

consistent with the notation above, we have $n = 3$, $N = 1000$, $k = 10$ and $m = 11$ or $m = 12$. Precisely, [22, page 11] states:

> *According to [15], we can choose the parameters as follows:* $\mu = 3$, $\lambda = 1000$ *and* $k = 10$, *while the ring* R *is instantiated with* $\mathbb{Z}_{12}$.

The citation "[15]" references the Grigoriev-Shpilrain protocol [5] discussed above. The choice of $\mathbb{Z}_{11}$ or $\mathbb{Z}_{12}$ gives different algebraic structures with, perhaps, very different algorithmic properties. To have a more complete picture, we analyze the protocol from [22] for both cases: $\mathbb{Z}_{11}$ and $\mathbb{Z}_{12}$ in Section 3.

1.4. **Our contribution.** The main contribution of this paper is a heuristic algorithm for solving the conjugacy search problem in matrices over a semigroup of N-truncated polynomials. The heuristics used in the algorithm allow us to solve the conjugacy search problem on a majority of inputs generated using the parameters proposed in [5, 22] which invalidates the security claims. We also argue that the proposed sampling procedure with high probability generates trivial instances of the conjugacy problem.

In the following sections we argue the validity of the assumptions above with the main focus on solving the conjugacy search problem in G. In section 2 we discuss in more detail the sampling procedures proposed in [5] and emphasize the flaws in the construction of the problem instances. In Section 3 we show that the protocol of Wang et al. is neither secure nor feasible for the proposed parameter values. In Section 4 we describe our heuristic algorithm for the conjugacy search problem in the semigroup of matrices over truncated polynomials and provide experimental results.

2. Key generation

In this section we discuss the key generation procedures for protocols mentioned in the introduction. Both use two matrices A and X, where X is invertible.

The matrix A is somewhat similar to the base element in the Diffie-Hellman protocol and, in particular, the simpler A the faster the computations. To make computations more efficient it is suggested in [5] to use 5-sparse polynomials, i.e., polynomials with 5 monomials. A random 5-sparse polynomial is generated by selecting 5 random monomials of degree at most $N - 1$.

The matrix X is generated as a product of matrices:

$$X = E_{i_1 j_1}(m_1) E_{i_2 j_2}(m_2) \cdots E_{i_l j_l}(m_l),$$

where $E_{ij}(m)$ is an $n \times n$ matrix that differs from the identity matrix I by one nontrivial off-diagonal polynomial $m \in R$ in the position (i, j) (with $i \neq j$). We can assume that m_i's used in the generation of X are monomials because for any $1 \leq i \neq j \leq 3$ and polynomials $m_1, m_2 \in R$ we have:

$$E_{ij}(m_1 + m_2) = E_{ij}(m_1) \cdot E_{ij}(m_2).$$

The monomials m_l and indices i_l, j_l are chosen randomly. It is easy to check that the matrices $E_{ij}(m)$ are invertible and:

$$E_{ij}(m)^{-1} = E_{ij}(-m).$$

Therefore,

$$X^{-1} = E_{i_l j_l}(-m_l) \cdots E_{i_2 j_2}(-m_2) E_{i_1 j_1}(-m_1).$$

In the next subsection we discuss some crucial observations and flaws of the proposed generation procedure with respect to the assumptions above.

2.1. **Sparse truncated polynomials chosen uniformly randomly make CSP trivial.** The original paper [5] does not state explicitly how to generate random monomials used in the generation of entries for A and $E_{i,j}(m)$. On the other hand mentioning the generic case hardness in one of the claims implies that they are generated uniformly. Hence, each m is a sum of 5 monomials $ax_{i_1} \dots x_{i_s}$ where:

- a is a uniformly chosen from $\mathbb{Z}_{11} \setminus \{0\}$;
- $x_{i_1} \dots x_{i_s}$ is uniformly chosen from the (finite) set of power product over $\{x_1, \dots, x_{10}\}$ of power up to $N = 1000$.

Proposition 1. *The following hold:*

(a) *The probability that a uniformly randomly chosen power product has degree smaller than* 500 *is* 0.00102107974.
(b) *The probability that a uniformly randomly chosen* 5*-sparse polynomial involves a monomial of degree less than* 500 *is approximately* 0.0051.
(c) *The probability that a uniformly randomly chosen matrix A involves a monomial of degree less than* 500 *is approximately* 0.0450.

Proof. The number of power products of degree at most d in k variables is $\binom{d+k}{d}$. Therefore, the number of power products of degree less than N is $\binom{N+k-1}{N-1}$ and of degree less then $N/2$ is $\binom{N/2+k-1}{N/2-1}$ and

$$\Pr(\deg(m) < N/2) = \frac{\binom{N/2+k-1}{N/2-1}}{\binom{N+k-1}{N-1}} = \frac{\binom{509}{499}}{\binom{1009}{999}} = 0.00102107974.$$

A randomly generated polynomial p involves 5 uniformly randomly chosen monomials and hence:

$$\Pr(\text{all monomials in } p \text{ are of degree} \geq 500) = (1-0.00102107974)^5 \approx 0.9949.$$

Similarly, a randomly generated matrix A involves 45 uniformly randomly chosen monomials and hence:

$$\Pr(\text{all monomials in } A \text{ are of degree} \geq 500) = (1-0.00102107974)^{45} \approx 0.9550.$$

□

Next observe how conjugation by an elementary matrix $E_{12}(m)$ transforms A:

$$\begin{aligned} E_{12}^{-1}(m)\cdot A\cdot E_{12}(m) &= \begin{bmatrix} 1 & -m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ (1) &= \begin{bmatrix} a_{11}-ma_{21} & a_{12}+m(a_{11}-a_{22})-m^2a_{21} & a_{13}-ma_{23} \\ a_{21} & a_{22}+ma_{21} & a_{23} \\ a_{31} & a_{32}+ma_{31} & a_{33} \end{bmatrix}. \end{aligned}$$

Conjugation by other elementary matrices give similar formulae (see Appendix A). These formulae imply the following.

Lemma 2. *If the degree of the smallest monomial involved in A is at least* 500 *and* $\deg(m) \geq 500$*, then* $E_{12}^{-1}(m)\cdot A\cdot E_{12}(m) = A$. □

Theorem 3. *The probability that $X^{-1}AX = A$ for randomly chosen matrices A and X is not smaller than* 0.90750*.*

Proof. The matrix A is randomly chosen and involves 45 monomials; then it is conjugated by 50 random elementary matrices. The probability that at all steps no monomial of degree less that 500 will appears is:

$$(1 - 0.00102107974)^{95} \approx 0.90750.$$

By Lemma 2 in this event we have $X^{-1}AX = A$. Since we considered only a subset of all events that result in equality $X^{-1}AX = A$ the actual probability of obtaining a trivial conjugacy equation is even higher. □

Therefore, for the suggested generation method the conjugacy search problem is trivial on most of the inputs. That does not mean that there are no difficult instances of CSP, but the key generation procedure is bad. Trying to find hard instances we modified the procedure for generating random monomials. Instead of choosing the monomials uniformly, it randomly uniformly chooses the degree of the monomial and then the required numbers of the involved variables are chosen. As above, $N-1$ is the maximal degree of a monomial and k is the number of variables. The procedure is the following:

Algorithm 2. Monomial generation.

1: Choose the degree s from $\{1, \ldots, N-1\}$ uniformly randomly.
2: Choose numbers $i_1, \ldots, i_s$ from $\{1, \ldots, k\}$ uniformly randomly.
3: Choose a coefficient a from $\mathbb{Z}_{11} \setminus \{0\}$ uniformly randomly.
4: Output the monomial $ax_{i_1}x_{i_2}\cdots x_{i_s}$.

3. Analysis of Wang et al. primitive

In this section we discuss properties of Wang et al. protocols. The first observation is that the CSP-based Diffie-Hellman problem in G and the CSP in G are not much related. Indeed, the CSP-based Diffie-Hellman problem in G is very restrictive because the conjugators come from the cyclic subgroup generated by X. The problem is only to recognize the power of X used in conjugation. The DH problem can be reduced to the conjugacy search problem relative to a subgroup generated by X (cf. [18]), but we do not see how it can be reduced to the general CSP in G (a general solution will not work – the centralizer will spoil the answer). Similarly, there is no way to reduce the general CSP to CSP-based DH problem. Therefore, computational hardness of any of these problems does not imply hardness of the other.

The main problem however is with the CSP-based Diffie-Hellman assumption in the semigroup G.

Recall that the order $ord(X)$ of a group element X is the smallest positive integer m such that $X^m = I$. All Wang et al. protocols require computation of powers X^i for large values of i so that i cannot be recovered from the triple $(A, X, X^{-i}AX^i)$ by simple enumeration. The requirement above implies that the order of the conjugating element X must be large. Indeed, let $ord(X) = m$ and $i > m$. We can write $i = m \cdot k + r$ where $r < m$. Then

$$X^i = X^{m \cdot k + r} = (X^m)^k \cdot X^r = X^r.$$

Therefore, recovering a power X^i by brute-force enumeration is as hard as computing the power X^m, where $m = ord(X)$.

Next we present empirical evidence that $ord(X)$ is small most of the time. First of all if we use the uniform random approach of Grigoriev-Shpilrain, then using the argument similar to Theorem 3 one can show that most of the instances of X have small (in fact trivial) orders.

More surprisingly similar behaviour is observed when X is generated using Algorithm 2. We performed a series of experiments and were able to compute the order of every instance of the matrix X. Moreover, in the case when the polynomial coefficients were taken from $\mathbb{Z}_{11}$, all instances had orders 11, 121 or 1331 with 121 being the most frequent. Notice that these are all powers of 11 which is the order of the polynomial coefficients. In the case of $\mathbb{Z}_{12}$ the orders were more diverse with 31104, 62208 and 124416 being the greatest but each occurring only once. Orders 72 and 144 were the most frequent and all values were multiples of 12. One explanation for the relatively small orders of X is that the terms of the polynomials are eliminated due to the small finite order of the coefficients.

We have to mention here that, since the monomials are chosen randomly with the uniform distribution on the degrees, the number of monomials in X^i grows very fast in terms of i (we can not say exponentially fast, because we

work with finite, though huge semigroup). So the scheme is neither secure nor practical regardless of the generation procedure.

One can try to increase the order by introducing different sampling strategies. We performed experiments where monomials from R were chosen randomly with the degree $N/5 = 200$ at most. The idea is that introducing smaller degree polynomials will make truncation less common. Indeed, the size of power matrices became times larger then in the previous experiments. However, the orders of such matrices were again some small powers of 11.

4. Algorithm for solving the conjugacy problem

In this section we present a heuristic procedure for solving the conjugacy search problem in G. Recall that our goal is: given two matrices A and $C = X^{-1}AX$ in G compute a conjugator X. Essentially, our heuristic is a variation of a length-based attack that has been successfully used several times in group-based cryptography ([6, 4, 14, 10, 13]).

4.1. **General idea of the attack.** Since X is generated as a product of elementary matrices $E_1E_2\cdots E_n$, the matrix C can be viewed as a result of a sequence of conjugations by the matrices E_i:

$$
\begin{array}{rrcl}
C_0 = & & A & \\
 & & \downarrow & \\
C_1 = & E_1^{-1} & A & E_1 \\
 & & \downarrow & \\
C_2 = & E_2^{-1}E_1^{-1} & A & E_1E_2 \\
 & & \downarrow & \\
 & & \vdots & \\
C = C_n = & E_n^{-1}\dots E_2^{-1}E_1^{-1} & A & E_1E_2\dots E_n
\end{array}
\tag{2}
$$

The goal is to reverse the sequence (2) and recover each single elementary matrix E_i one by one. Each E_i is a solution to the CSP for the pair of matrices (C_{i-1}, C_i), with C_{i-1} unknown. So, we need to solve the following problem several times:

Recovering E_i: Given a matrix of the form $E_i^{-1}C_{i-1}E_i$, find E_i.

Clearly, the problem above is not very well defined because any elementary matrix E could have been used to obtain C_i from the unknown C_{i-1}. We find the matrix E which fits the best for the role of the conjugator. This is done using the concept of a size function.

4.2. **Size function attack.** The following function will guide our process of recovery of elementary conjugators.

Definition 4. The *size* of a polynomial $f \in R$ is the total number of monomials in f, denoted size(f). The *size* of an $n \times n$ matrix M is the total

number of monomials in all entries of M:

$$\operatorname{size}(M) = \sum_{1 \le i,j \le n} \operatorname{size}(M_{ij}).$$

For polynomials $p, q \in R$ define a number $\sigma_p(q)$ to be the number of power products in p that are not present in q. Similarly, for matrices $A, B \in G$ define a number:

$$\sigma_A(B) = \sum_{i,j} \sigma_{a_{ij}}(b_{ij}).$$

Intuitively, this function indicates how much one matrix is contained in the other. If two matrices are the same, then $\sigma_A(B) = \sigma_B(A) = 0$. The opposite direction is not necessarily true because we disregard the coefficients of the polynomials.

Proposition 5. *Let $A \in G$ and $B = E_{ij}(m)^{-1} A E_{ij}(m)$, where m is a monomial. Then*

$$Pr(\sigma_A(B) = 0) \ge 1 - \frac{7 \operatorname{size}(A)^2}{10 |B_{\mathbb{Z}^{10}}(1000)|},$$

where $B_{\mathbb{Z}^{10}}(1000)$ is the ball of radius 1000 in the Cayley graph of the free abelian group $\mathbb{Z}^{10}$.

Proof. Without loss of generality we may assume that $i = 1$ and $j = 2$. By formula (1) on page 6, the matrix B gets new monomials from the products ma_{21}, $m(a_{11} - a_{22}) - m^2 a_{21}$, etc. Since, obviously,

$$\operatorname{size}(a_{ij}) \le \operatorname{size}(A),$$

each of those products involves not more than $3 \operatorname{size}(A)$ monomials. The number $\sigma_A(B)$ is positive only if a new monomial cancels out some old monomial. Therefore, the number of choices for m that induce cancelation of some old monomial can be bounded above by $7 \operatorname{size}(A)^2$. Since the monomial m is chosen as a product of a random nonzero coefficient and a random power product and the total number of random power products is $|B_{\mathbb{Z}^{10}}(1000)|$, it is easy to get the claimed bound on $Pr(\sigma_A(B) = 0)$. □

In fact, in Proposition 5 we proved a stronger result. We proved that with high probability every new monomial in a conjugate matrix was not involved in the original matrix. Next, a trivial estimate:

$$|B_{\mathbb{Z}^{10}}(1000)| = \binom{1000}{11} \ge 2 \cdot 10^{25},$$

implies the following result.

Corollary 6. *Let $A \in G$ and $B = E_{ij}(m)^{-1} A E_{ij}(m)$. If $\operatorname{size}(A) \le 10^5$, then $Pr(\sigma_A(B) = 0) \ge 1 - 10^{-15}$.* □

Corollary 7. *Let $A \in G$ and $B = E_{ij}(m)^{-1} A E_{ij}(m)$. Suppose that $\operatorname{size}(A) \le 10^5$. Then*

$$Pr(A = B \text{ or } \operatorname{size}(A) < \operatorname{size}(B)) \ge 1 - 10^{-15}$$

Proof. Excluding all events when all the new monomials have power greater than 999 and, hence, being truncated (the trivial case for conjugacy) we obtain the same estimate as in Corollary 6. □

We consider size 10^5 because we think that bigger matrices are not practical. By Corollary 7 the main assumption of the length based attacks:

$$\text{size}(E_{ij}(m)^{-1}AE_{ij}(m)) > \text{size}(A)$$

holds for the majority of elements $A, E_{ij}(m) \in G$. Hence, given a matrix C_k we find a conjugator as the matrix $E_{ij}(m)$ which gives the least value $\text{size}(E_{ij}(m)C_kE_{ij}(m)^{-1})$.

Here we encounter another challenge. The number of the elementary matrices $E_{ij}(m)$ can be bounded below as $6 \cdot 10 \cdot |B_{\mathbb{Z}^{10}}(1000)| \geq 10^{27}$; we cannot simply enumerate them and test which ones reduce the size of the current matrix C_i. In the next subsection we discuss a method reducing the number of the elementary matrices that need to be tested.

4.3. **Reducing the number of the elementary matrices for tests.** In this section we show how given a matrix of the form:

$$C_i = E_{ij}(m)^{-1}C_{i-1}E_{ij}(m)$$

to find a small number of candidates for (i, j, m). We perform our procedure for all different values of i, j separately, so, here we only consider the values $i = 1$, $j = 2$. Other cases are similar.

For a polynomial $f \in R$ by $\text{Mon}(f)$ denote the set of all monomials involved in f and for a pair of polynomials $f, g \in R$ define $\text{Frac}(f, g)$ to be the set of monomials:

$$\{x_1^{s_1-t_1} \dots x_{10}^{s_{10}-t_{10}} \mid x_1^{s_1} \dots x_{10}^{s_{10}} \in \text{Mon}(f),\ x_1^{t_1} \dots x_{10}^{t_{10}} \in \text{Mon}(g),\ s_i \geq t_i\}.$$

Let $\min(f)$ denote the minimal short-lex degree monomial in $f \in R$. The next theorem immediately follows from the discussion in Section 4.2.

Proposition 8. *If $C_i = E_{12}(m)^{-1}C_{i-1}E_{12}(m)$, $C_{i-1} \neq C_i$ and $\text{size}(C_{i-1}) \leq 10^5$, then with probability at least $1 - 10^{-15}$ the monomial m belongs to the following union:*

$$\begin{aligned}\mathcal{M}_{12}(C_i) \;=\; & \text{Frac}(-c_{11}, \min(c_{21})) \cup \text{Frac}(c_{12}, \min(c_{11})) \cup \\ & \text{Frac}(-c_{12}, \min(c_{22})) \cup \text{Frac}(-c_{13}, \min(c_{23})) \cup \\ & \text{Frac}(c_{22}, \min(c_{21})) \cup \text{Frac}(c_{32}, \min(c_{31})),\end{aligned}$$

where c_{st} is an element of C_i. □

The size of the set $\mathcal{M}_{12}(C_i)$ can be bounded above by

$$\text{size}(c_{11}) + 2\,\text{size}(c_{12}) + \text{size}(c_{13}) + \text{size}(c_{22}) + \text{size}(c_{32})$$

and, therefore, is just linear in terms of $\text{size}(C_i)$.

The sets of monomial candidates $\mathcal{M}_{st}(C_i)$ can be obtained in the similar way for all the other cases. We define these sets explicitly in Appendix B.

Using the set $\mathcal{M}_{st}(C_i)$ we can define the corresponding set of elementary matrices

$$\mathcal{E}_{st}(C_i) = \{E_{st}(u) \mid u \in \mathcal{M}_{st}(C_i)\}.$$

Using Proposition 8 it is easy to see that with probability at least $1 - 10^{-15}$ the true conjugator $E_{st}(m)$ is in the set $\mathcal{M}_{st}(C_i)$.

Some of the matrices in the set $\mathcal{E}_{st}(C_i)$ may be eliminated even further if we impose the condition of Proposition 5

$$\mathcal{E}'_{st}(C_i) = \left\{E \in \mathcal{E}_{st}(C_i) \mid \sigma_{E^{-1}C_iE}(C_i) = 0\right\}.$$

Finally we combine the sets corresponding to all possible locations of the monomial in an elementary matrix:

$$\mathcal{E}(C_i) = \cup_{s,t}\mathcal{E}'_{st}(C_i).$$

$\mathcal{E}(C_i)$ is the reduced set of candidate elementary matrices to be considered in the tests.

4.4. **The results.** Algorithms 3 and 4 are the two components describing the heuristic procedure for solving conjugacy search problem in G. This procedure is similar to the best descent variation of the length-based attack proposed in [14].

Algorithm 3. Reduction

Input: A matrix B.
Output: Matrix Y with minimal $size(Y^{-1}BY)$.

```
1: Let Y = I.
2: loop
3:     Compute E(B);
4:     if E(B) = ∅ then return Y.
5:     Find Ê ∈ E(B) minimizing size(Ê^{-1}BÊ).
6:     if size(B) − size(Ê^{-1}BÊ) ≤ 0 then return Y.
7:     Set Y = Y · Ê;
8:     Set B = Ê^{-1}BÊ;
9: end loop
```

Algorithm 3 attempts to find the minimal matrix in the conjugacy class of a given matrix B. If we give the matrix $B = X^{-1}AX$ as the input to Algorithm 3 then we expect matrix Y to be the best candidate for the solution to the corresponding conjugacy search problem. However, the matrix Y may fail to be the solution in the case when A is not minimal, i.e. there is an elementary matrix that may reduce the size of A itself. One extra step of reducing the size of A is needed and it is implemented in Algorithm 4 which solves the conjugacy search problem in G.

Algorithm 4. Conjugacy search.

Input: A pair of matrices A and $B = X^{-1}AX$ for some X.
Output: A matrix Y such that $Y^{-1}BY = A$ or FAIL.
1: Set $Y_A = Reduce(A)$;
2: Set $Y_B = Reduce(B)$;
3: **if** $Y_B^{-1}BY_B = Y_A^{-1}AY_A$ **then**
4: **return** $Y = Y_B \cdot Y_A^{-1}$;
5: **else**
6: **return** FAIL.
7: **end if**

Algorithm 4 receives a pair A, $B = X^{-1}AX$ as the input. If it successfully finds a conjugator Y (not necessarily equal to X) then it outputs Y, otherwise a failure is reported. We tested the effectiveness of the procedure described above by executing Algorithm 4 on sets of instances of the conjugacy search problem generated using Algorithm 2.

Algorithms were implemented in C++ using CRyptography And Groups (CRAG) [2] library. Since the size of input significantly depends on the number of elementary matrices used to generate the conjugator, experiments were performed for different lengths of the product. There were 100 instances generated using each value. Experiments were performed on a desktop PC with eight core 2.67GHz Intel i7 processor and 6 gigabytes memory.

The results of our experiments are shown in Table 1. From the table we can see that the attack can successfully break a significant portion of instances. Almost 30% of instances generated using products of 50 elementary matrices were broken. This is the most hard case because of the size of the matrices in the conjugacy search problem. We want to point out that though we have considered matrices with 10^5 monomials in our proofs above, matrices of size 10^4 are already not very practical. It takes about 103 bits to store a single 1000-truncated monomial in 10 variables. Hence, public keys obtained using products of 50 elementary matrices will require several megabits of storage on average.

References

[1] I. Anshel, M. Anshel, D. Goldfeld, and S. Lemieux. Key agreement, the algebraic eraser^{TM}, and lightweight cryptography. In *Algebraic Methods in Cryptography*, volume 418 of *Contemporary Mathematics*, pages 1–34. American Mathematical Society, 2006.

[2] CRyptography And Groups (CRAG) C++ Library. available at `http://www.acc.stevens.edu/downloads.php`.

[3] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *STOC '87: Proceedings of the nineteenth annual ACM Conference on Theory of Computing*, pages 210–217, 1987.

Number of elementary matrices in the product generating X	10	20	30	40	50
Success rate %	94	82.7	67.1	44.2	28.5
Average Time,s	0.3	17.7	556.4	2346.6	11279.3
Number of monomials in $X^{-1}AX$					
Average	283	1410	5186	19407	45048
Minimal	63	148	366	1256	3895
Maximal	1574	13989	47122	236308	406332

TABLE 1. The success rate of solving the conjugacy search problem using Algorithm 4.

[4] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne. Length-based conjugacy search in the braid group. In *Algebraic Methods in Cryptography*, volume 418 of *Contemp. Math.*, pages 75–88. Amer. Math. Soc., 2006.

[5] D. Grigoriev and V. Shpilrain. Authentication from matrix conjugation. *Groups Complex. Cryptol.*, 1:199–206, 2009.

[6] J. Hughes and A. Tannenbaum. Length-based attacks for certain group based encryption rewriting systems. preprint. Available at `http://front.math.ucdavis.edu/0306.6032`.

[7] I. Kapovich, A. G. Miasnikov, P. Schupp, and V. Shpilrain. Generic-case complexity, decision problems in group theory and random walks. *J. Algebra*, 264:665–694, 2003.

[8] I. Kapovich, A. Myasnikov, P. Schupp, and V. Shpilrain. Average-case complexity and decision problems in group theory. *Adv. Math.*, 190:343–359, 2005.

[9] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, and C. Park. New public-key cryptosystem using braid groups. In *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes Comp. Sc.*, pages 166–183, Berlin, 2000. Springer.

[10] J. Longrigg and A. Ushakov. Cryptanalysis of the shifted conjugacy authentication protocol. *J. Math. Crypt.*, 2:107–114, 2008.

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

[12] A. G. Miasnikov, V. Shpilrain, and A. Ushakov. *Non-Commutative Cryptography and Complexity of Group-Theoretic Problems*. Mathematical Surveys and Monographs. AMS, 2011.

[13] A. G. Miasnikov and A. Ushakov. Random subgroups and analysis of the length-based and quotient attacks. *J. Math. Crypt.*, 2:29–61, 2008.

[14] A. D. Myasnikov and A. Ushakov. Length based attack and braid groups: Cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol. In *Advances in Cryptology – PKC 2007*, volume 4450 of *Lecture Notes Comp. Sc.*, pages 76–88. Springer, 2007.

[15] L. Sakalauskas, P. Tvarijonas, and A. Raulynaitis. Key agreement protocol (kap) using conjugacy and discrete logarithm problems in group representation level. *Informatica*, 18:115–124, 2007.

[16] V. Shpilrain and A. Ushakov. Thompson's group and public key cryptography. In *Applied Cryptography and Network Security – ACNS 2005*, volume 3531 of *Lecture Notes Comp. Sc.*, pages 151–164. Springer, 2005.

[17] V. Shpilrain and A. Ushakov. A new key exchange protocol based on the decomposition problem. In *Algebraic Methods in Cryptography*, volume 418 of *Contemporary Mathematics*, pages 161–167. American Mathematical Society, 2006.

[18] V. Shpilrain and A. Ushakov. The conjugacy search problem in public key cryptography: unnecessary and insufficient. *Appl. Algebra Engrg. Comm. Comput.*, 17:285–289, 2006.

[19] V. Shpilrain and A. Ushakov. An authentication scheme based on the twisted conjugacy problem. In *ACNS 2008*, volume 5037 of *Lecture Notes Comp. Sc.*, pages 366–372. Springer, 2008.

[20] V. M. Sidelnikov, M. A. Cherepnev, and V. Y. Yaschenko. Systems of open distribution of keys on the basis of noncommutative semigroups. *Russian Acad. Sci. Dokl. Math.*, 48:384–386, 1994.

[21] A. Ushakov. Authenticated commutator key-agreement protocol. submitted.

[22] L. Wang, L. Wang, Z. Cao, E. Okamoto, and J. Shao. New constructions of public-key encryption schemes from conjugacy search problems. In *Information security and cryptology*, volume 6584 of *Lecture Notes Comp. Sc.*, pages 1–17. Springer, 2010.

Appendix A. Elementary conjugation formulas

$$
\begin{aligned}
E_{12}^{-1}(m)\cdot A\cdot E_{12}(m) &= \begin{bmatrix} 1 & -m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} a_{11}-ma_{21} & a_{12}+m(a_{11}-a_{22})-m^2a_{21} & a_{13}-ma_{23} \\ a_{21} & a_{22}+ma_{21} & a_{23} \\ a_{31} & a_{32}+ma_{31} & a_{33} \end{bmatrix} \\
E_{13}^{-1}(m)\cdot A\cdot E_{13}(m) &= \begin{bmatrix} 1 & 0 & -m \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & 0 & m \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} a_{11}-ma_{31} & a_{12}-ma_{32} & a_{13}+m(a_{11}-a_{33})-m^2a_{31} \\ a_{21} & a_{22} & a_{23}+ma_{21} \\ a_{31} & a_{32} & a_{33}+ma_{31} \end{bmatrix} \\
E_{31}^{-1}(m)\cdot A\cdot E_{31}(m) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} a_{11}+ma_{13} & a_{12} & a_{13} \\ a_{21}+ma_{23} & a_{22} & a_{23} \\ a_{31}+m(a_{33}-a_{11})-m^2a_{13} & a_{32}-ma_{12} & a_{33}-ma_{13} \end{bmatrix} \\
E_{23}^{-1}(m)\cdot A\cdot E_{23}(m) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -m \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & m \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} a_{11} & a_{12} & a_{13}+ma_{12} \\ a_{21}-ma_{31} & a_{22}-ma_{32} & a_{23}+m(a_{22}-m_{33})-m^2a_{32} \\ a_{31} & a_{32} & a_{33}+ma_{32} \end{bmatrix} \\
E_{21}^{-1}(m)\cdot A\cdot E_{21}(m) &= \begin{bmatrix} 1 & 0 & 0 \\ -m & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ m & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} a_{11}+ma_{12} & a_{12} & a_{13} \\ a_{21}+m(a_{22}-a_{11})-m^2a_{12} & a_{22}-ma_{12} & a_{23}-ma_{13} \\ a_{31}+ma_{32} & a_{32} & a_{33} \end{bmatrix} \\
E_{32}^{-1}(m)\cdot A\cdot E_{32}(m) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -m & 1 \end{bmatrix}\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & m & 1 \end{bmatrix} \\
&= \begin{bmatrix} a_{11} & a_{12}+ma_{13} & a_{13} \\ a_{21} & a_{22}+ma_{23} & a_{23} \\ a_{31}-ma_{21} & a_{32}+m(a_{33}-a_{22})-m^2a_{23} & a_{33}-ma_{23} \end{bmatrix}
\end{aligned}
$$

Appendix B. Reduced sets of monomial candidates.

$$\begin{aligned}
\mathcal{M}_{12}(C_i) &= \mathrm{Frac}(-c_{11}, \min(c_{21})) \cup \mathrm{Frac}(c_{12}, \min(c_{11})) \cup \mathrm{Frac}(-c_{12}, \min(c_{22})) \cup \\
&\quad \mathrm{Frac}(-c_{13}, \min(c_{23})) \cup \mathrm{Frac}(c_{22}, \min(c_{21})) \cup \mathrm{Frac}(c_{32}, \min(c_{31})).
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_{13}(C_i) &= \mathrm{Frac}(-c_{11}, \min(c_{31})) \cup \mathrm{Frac}(-c_{12}, \min(c_{32})) \cup \mathrm{Frac}(c_{13}, \min(c_{11})) \cup \\
&\quad \mathrm{Frac}(-c_{13}, \min(c_{33})) \cup \mathrm{Frac}(c_{23}, \min(c_{21})) \cup \mathrm{Frac}(c_{33}, \min(c_{31})).
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_{31}(C_i) &= \mathrm{Frac}(c_{11}, \min(c_{13})) \cup \mathrm{Frac}(c_{21}, \min(c_{23})) \cup \mathrm{Frac}(c_{31}, \min(c_{33})) \cup \\
&\quad \mathrm{Frac}(-c_{31}, \min(c_{11})) \cup \mathrm{Frac}(-c_{32}, \min(c_{12})) \cup \mathrm{Frac}(-c_{33}, \min(c_{13})).
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_{23}(C_i) &= \mathrm{Frac}(-c_{21}, \min(c_{31})) \cup \mathrm{Frac}(-c_{22}, \min(c_{32})) \cup \mathrm{Frac}(c_{13}, \min(c_{12})) \cup \\
&\quad \mathrm{Frac}(c_{23}, \min(c_{22})) \cup \mathrm{Frac}(-c_{23}, \min(c_{33})) \cup \mathrm{Frac}(c_{33}, \min(c_{32})).
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_{21}(C_i) &= \mathrm{Frac}(c_{11}, \min(c_{12})) \cup \mathrm{Frac}(c_{21}, \min(c_{22})) \cup \mathrm{Frac}(-c_{21}, \min(c_{11})) \cup \\
&\quad \mathrm{Frac}(c_{31}, \min(c_{32})) \cup \mathrm{Frac}(-c_{22}, \min(c_{12})) \cup \mathrm{Frac}(-c_{23}, \min(c_{13})).
\end{aligned}$$

$$\begin{aligned}
\mathcal{M}_{32}(C_i) &= \mathrm{Frac}(-c_{31}, \min(c_{21})) \cup \mathrm{Frac}(c_{12}, \min(c_{13})) \cup \mathrm{Frac}(c_{22}, \min(c_{23})) \cup \\
&\quad \mathrm{Frac}(c_{32}, \min(c_{33})) \cup \mathrm{Frac}(-c_{32}, \min(c_{22})) \cup \mathrm{Frac}(-c_{33}, \min(c_{23})).
\end{aligned}$$

Department of Mathematics, Stevens Institute of Technology, Hoboken, NJ, USA

E-mail address: amyasnik,aushakov@stevens.edu