Cryptanalysis via algebraic spans

Adi Ben-Zvi, Arkadius Kalka, and Boaz Tsaban

Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel
adi2lugassy@gmail.com, tschussle@gmail.com, tsaban@math.biu.ac.il

Abstract. We introduce a method for obtaining provable polynomial time solutions of problems in nonabelian algebraic cryptography. This method is widely applicable, easier to apply, and more efficient than earlier methods. After demonstrating its applicability to the major classic nonabelian protocols, we use this method to cryptanalyze the Triple Decomposition key exchange protocol, the only classic group theory based key exchange protocol that could not be cryptanalyzed by earlier methods.

1 Introduction

Since Diffie and Hellman's 1976 key exchange protocol, few alternative protocols withstood cryptanalysis, all based on abelian algebraic structures. In the years 1999 and 2000 [2, 9], two general key exchange protocols based on *nonabelian* algebraic structures were introduced. The security of these protocols is based on variations of the conjugacy problem in nonabelian groups. The *Triple Decomposition* key exchange protocol was introduced in 2006 [10,11], and subsequently included in textbooks on nonabelian cryptography [7,12,13]. Its security is based on a problem that is very different from those of the earlier nonabelian key exchange protocols, and it stood out as the only nonabelian group theoretic protocol resisting known cryptanalyses [24].

All mentioned protocols were implemented over Artin's braid group $\mathbf{B}_N$. For the main part of this paper, it suffices to know that this group has an efficient, faithful representation as a group of matrices, and the computational problems on which the security of the above-mentioned protocols are based reduce to the same problems in groups of matrices over finite fields. The details of the reduction are available in Section 5.

Our main contribution is the introduction of *algebraic span cryptanalysis*, a general approach for provable polynomial time solutions of computational problems in groups of matrices, and thus in all groups with efficient matrix representations. Algebraic span cryptanalysis improves upon earlier ones (such as the Cheon–Jun method and the linear centralizer method [5, 24]), in its wider applicability, simplicity, and efficiency. It solves all problems that were solved by earlier provable methods.

A true challenge for the novelty of a new method is to cryptanalyze a protocol that could not be cryptanalyzed by earlier methods, and the Triple Decomposition key exchange protocol is such. With a novel view at the public information

provided by this protocol, algebraic span cryptanalysis provides its first cryptanalysis.

Previously, provable cryptanalyses of this type were considered theoretical only. Using some algorithmic speed-ups, we also provide the first implementation of a cryptanalysis of this type. All of our experiments, with a wide range of parameters, succeeded in extracting the shared key out of the public information of the protocol.

Related work. The Commutator and the Braid Diffie–Hellman protocols were cryptanalyzed in a number of heuristic ways, but these attacks were foiled by changing the distributions on the group [6, 23, and references therein]. In two breakthrough papers [5,24], provable polynomial time algorithms were found for the precise computational problems on which these, and some related protocols, are based. In a series of works [14–18, and references therein], Roman'kov and others developed a provable polynomial time method that applies to key exchange protocols with certain commuting substructures. None of these methods applies to the Triple Decomposition protocol, that is used here to demonstrate the novelty of our method.

Paper outline. Section 2 introduces the new method, in general terms. The exact application of this method depends on the specific protocol we wish to cryptanalyze. Section 3 demonstrates the applicability of this method to the classic nonabelian key exchange protocols. In addition to demonstrating the flexibility of this method, this section is designed to make the reader comfortable with this method, and make it easier to proceed to Section 4, that addresses a hitherto resistant challenge, where the application of our method is more involved. Section 5 provides the details of the Triple Decomposition key exchange protocol and its reduction to a matrix group over a finite field, together with more detailed complexity estimates and experimental results.

This paper is a composition of a work by the third named author (Sections 1–3) and a joint work of all three authors (Sections 4–5).

2 Algebraic span cryptanalysis in a nutshell

Let $\mathbb{F}$ be a finite field, and $\mathrm{M}_n(\mathbb{F})$ be the set of $n \times n$ matrices with entries in $\mathbb{F}$. An *algebra* of matrices is a family of matrices $A \subseteq \mathrm{M}_n(\mathbb{F})$ that is closed under linear combinations and multiplications. For a set $S \subseteq \mathrm{M}_n(\mathbb{F})$, let $\mathrm{Alg}(S)$ be the algebra generated by S, that is, the smallest Algebra $A \subseteq \mathrm{M}_n(\mathbb{F})$ that contains S as a subset. Every subalgebra of $\mathrm{M}_n(\mathbb{F})$ is also a vector space over the field $\mathbb{F}$. Let $\mathrm{GL}_n(\mathbb{F})$ be the group of invertible matrices in $\mathrm{M}_n(\mathbb{F})$. For a subgroup $G \leq \mathrm{GL}_n(\mathbb{F})$, we have $\mathrm{Alg}(G) = \mathrm{span}(G)$, the vector space spanned by G. For simplicity we assume, throughout, that the dimension of the vector space $\mathrm{Alg}(G)$ is at least a positive constant times n. Notice that even for cyclic groups G, this is typically the case.

Throughout, let ω be the linear algebra constant, the minimal real number such that the complexity of $n \times n$ matrix multiplication is $O(n^\omega)$ field operations.

Proposition 1 *Let $G = \langle g_1, \ldots, g_k \rangle \leq \mathrm{GL}_n(\mathbb{F})$ be a group, and $d \leq n^2$ be the dimension of the vector space $\mathrm{Alg}(G)$. A basis for the vector space $\mathrm{Alg}(G)$ can be computed using $O(kd^2n^2)$ field operations.*

Proof. Initialize a sequence $s = (I)$, the identity matrix, and $i := 1$. Repeat the following as long as there is an element in position i of the sequence s:

1. For $j = 1, \ldots, k$, if $s_i g_j \notin \operatorname{span} S$, append $s_i g_j$ at the end of the sequence s.
2. $i := i + 1$.

The resulting sequence s is a basis for $\operatorname{span} G$. For each i and each j, the complexity of computing the products $s_i g_j$ is n^ω field operations. Assume that the matrices are stored in s in a vector form, and the matrix s is kept in Echelon normal form throughout the process. Since there are at most d vectors in s, each of length n^2, the complexity of checking whether a vector is in $\operatorname{span} s$ is at most $O(dn^2)$. Thus, the overall complexity is $O(kd(n^\omega + dn^2))$ field operations. Since we assume that d is at least a constant multiple of n, the second term dominates the first one. □

Proposition 1 holds, more generally, for semigroups of matrices; but this will not be used here. There are advanced methods, via representation theory, to slightly reduce the complexity of this computation [8] but, for our purposes, Proposition 1 suffices as it is.

2.1 The method

Algebraic span cryptanalysis is applied as follows. Let $G_1, \ldots, G_k$ be given, publicly known subgroups of $\mathrm{GL}_n(\mathbb{F})$. Assume that a secret $f(g_1, \ldots, g_k)$ is computed from unknown matrices $g_1 \in G_1, \ldots, g_k \in G_k$, by means of a prescribed, public function f. Assume that we can extract, from a protocol transaction, a system of linear equations (or constraints) on the entries of the unknown matrices $g_1, \ldots, g_k$, and we wish to find the secret $f(g_1, \ldots, g_k)$.

Instead of solving the given linear equations subject to the restrictions $g_1 \in G_1, \ldots, g_k \in G_k$ (which may be computationally infeasible), we solve these linear equations subject to the *linear* constraints $g_1 \in \mathrm{Alg}(G_1), \ldots, g_k \in \mathrm{Alg}(G_k)$. We then try to prove (or at least verify by experiments) that, for each solution $\tilde{g}_1, \ldots, \tilde{g}_k$, we have $f(\tilde{g}_1, \ldots, \tilde{g}_k) = f(g_1, \ldots, g_k)$.

Strikingly, this simple method applies, provably, in all cases of nonabelian algebraic cryptography where polynomial-time algorithms are known [9, 14–16, 24], and in a case that was not cryptanalyzed thus far. We provide these details in the following sections.

The equations do not have to be given as linear. For example, an equation $g_1 a g_2 = b$ with a and b known can be transformed to the equation $a g_2 = g_1^{-1} b$, which is linear in the entries of the matrices g_1^{-1} and g_2.

2.2 Invertibility

Often, as in the latter example, we need some elements in our solution to be invertible. Since there *is* an invertible solution, namely, $(g_1, \ldots, g_k)$, the following lemma (Invertibility Lemma [24, Lemma 9]) guarantees that random solutions are invertible with probability bounded away from zero, provided that the field is not too small. This will be the situation in all of our applications. Thus, we may pick random solutions until they are invertible.

Lemma 1. *For a finite field $\mathbb{F}$, let $a_1, \ldots, a_m \in \mathrm{M}_n(\mathbb{F})$, such that some linear combination of these matrices is invertible. If $\alpha_1, \ldots, \alpha_m$ are chosen uniformly and independently from $\mathbb{F}$, then the probability that the linear combination $\alpha_1 a_1 + \cdots + \alpha_m a_m$ is invertible is at least $1 - \frac{n}{|\mathbb{F}|}$.*

2.3 Complexity

The next section provides concrete applications of this approach to several problems in the field of nonabelian algebraic cryptography. Enough examples are provided so that the reader can apply this method to additional problems in the field, including essentially all known key exchange protocols based on groups with efficient representations as matrix groups. In these examples, the proposed platform group is Artin's braid group $\mathbf{B}_N$. However, these problems are known to transform into a matrix group $G \leq \mathrm{GL}_n(\mathbb{F})$ [9,24]. The reduction uses the the Lawrence–Krammer representation, and thus the matrices are of rank $n = \binom{N}{2}$. In this reduction, the cardinality of the field $\mathbb{F}$ is, roughly, $2^{M^3N^2}$, for some length parameter M. We may assume that $M \approx N$. Then the cost of field multiplication is about N^5, ignoring a logarithmic factor. Tighter scrutiny of this reduction is likely to lead to substantially smaller field sizes; the extra factor of N^5 should not be considered definite. Additional details are provided in Section 5.

3 Sample applications

In this section, we apply the algebraic span method to the major classic nonabelian key exchange protocols. The application in these cases is not difficult. This demonstrates the applicability of the method, and also serves as a good preparation for the next section, where a more involved application is made. The protocols are presented for general groups, but they were all proposed to use groups that have efficient representations as matrix groups. We thus assume here that the groups are matrix groups. The exact parameters originally suggested for these protocols are not important: The cryptanalyses we provide are provable, and their complexity is unaffected by the exact settings or distributions used by the protocol. For the main application, we will provide details in Section 5.

The protocols to which we apply our method are described succinctly by diagrams. In these diagrams, green letters indicate publicly known elements,

and red ones indicate secret elements, known only to their holders. Results of computations involving elements of both colors may be either publicly known, or secret, depending on the context.

Most of these protocols, and their analyzes, use the following notation. For a nonabelian group G and group elements $g, x \in G$, we define $g^x := x^{-1}gx$. Useful identities involving this notation include $g^{xy} = (g^x)^y$, and $g^c = g$ for every element $c \in G$ that commutes with g, that is, when $cg = gc$.

3.1 The Commutator key exchange protocol

A free group word in the variables $x_1, \ldots, x_k$ is a product of the form

$$x_{i_1}^{\epsilon_1} x_{i_2}^{\epsilon_2} \cdots x_{i_m}^{\epsilon_m},$$

with $i_1, \ldots, i_m \in \{1, \ldots, k\}$ and $\epsilon_1, \ldots, \epsilon_m \in \{1, -1\}$, and with no subproduct of the form $x_i x_i^{-1}$ or $x_i^{-1} x_i$. The *Commutator key exchange protocol* [2] is described in Figure 1 below. In some detail:

1. A nonabelian group G and elements $a_1, \ldots, a_k, b_1, \ldots, b_k \in G$ are publicly given.
2. Alice and Bob choose free group words in the variables $x_1, \ldots, x_k$, $v(x_1, \ldots, x_k)$ and $w(x_1, \ldots, x_k)$, respectively.
3. Alice substitutes $a_1, \ldots, a_k$ for $x_1, \ldots, x_k$, to obtain a secret element $a = v(a_1, \ldots, a_k) \in G$. Similarly, Bob computes a secret element $b = w(b_1, \ldots, b_k)$ in G.
4. Alice sends the conjugated elements ${b_1}^a, \ldots, {b_k}^a$ to Bob, and Bob sends the conjugated elements ${a_1}^b, \ldots, {a_k}^b$ to Alice.
5. The shared key is the *commutator* $a^{-1}b^{-1}ab$.

As conjugation is a group isomorphism, we have

$$v({a_1}^b, \ldots, {a_k}^b) = v(a_1, \ldots, a_k)^b = a^b = b^{-1}ab.$$

Thus, Alice can compute the shared key $a^{-1}b^{-1}ab$ as $a^{-1}v({a_1}^b, \ldots, {a_k}^b)$, using her secret $a, v(x_1, \ldots, x_k)$ and the public elements ${a_1}^b, \ldots, {a_k}^b$. Similarly, Bob computes $a^{-1}b^{-1}ab$ as $w({b_1}^a, \ldots, {b_k}^a)^{-1}b$.

The security of the Commutator key exchange protocol is determined by the difficulty of the following problem. As usual, for a group G and elements $g_1, \ldots, g_k \in G$, the subgroup of G generated by the elements $g_1, \ldots, g_k$ is denoted $\langle g_1, \ldots, g_k \rangle$.

Problem 2 *Let G be a group, $a_1, \ldots, a_k, b_1, \ldots, b_k \in G$, $a \in \langle a_1, \ldots, a_k \rangle$, and $b \in \langle b_1, \ldots, b_k \rangle$.*
Given the group elements $a_1, \ldots, a_k, b_1, \ldots, b_k, a_1^b, \ldots, a_k^b, b_1^a, \ldots, b_k^a$, compute the commutator $a^{-1}b^{-1}ab$.

The Commutator key exchange protocol uses Artin's braid group as its platform group, but it is known that the problem reduces, polynomially, to the same problem in matrix groups over finite fields [24]. Thus, we need to solve Problem 2 in matrix groups.

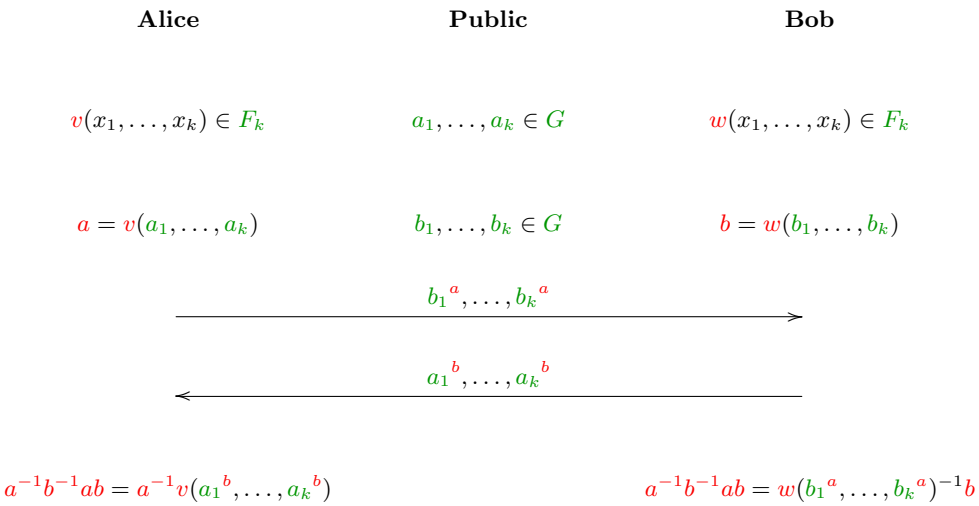

Fig. 1. The Commutator key exchange protocol

Lemma 3 *Let $x, \tilde{x} \in \mathrm{GL}_n(\mathbb{F})$ and $G = \langle g_1,\ldots,g_k\rangle \leq \mathrm{GL}_n(\mathbb{F})$. If ${g_i}^x = {g_i}^{\tilde{x}}$ for all $i = 1,\ldots,k$, then $g^x = g^{\tilde{x}}$ for all $g \in \mathrm{Alg}(G)$.*

Proof. Conjugation is an automorphism of the matrix algebra. □

We apply the algebraic span method to Problem 2, as follows:

1. Compute bases for the vector spaces $\mathrm{Alg}(A)$ and $\mathrm{Alg}(B)$. Let d be the maximum of the sizes of these bases.
2. Solve the following homogeneous system of linear equations in the unknown matrix $x \in \mathrm{Alg}(A)$:
$$b_1 \cdot x = x \cdot {b_1}^a$$
$$\vdots$$
$$b_k \cdot x = x \cdot {b_k}^a,$$
a system of linear equations on the d coefficients determining the matrix x, as a linear combination of the basis of the space $\mathrm{Alg}(A)$.
3. Fix a basis for the solution space, and pick random solutions until the picked solution $\tilde{a}$ is invertible.
4. Solve the following homogeneous system of linear equations in the unknown matrix $y \in \mathrm{Alg}(B)$:
$$a_1 \cdot y = y \cdot {a_1}^b$$
$$\vdots$$
$$a_k \cdot y = y \cdot {a_k}^b,$$
a system of linear equations on the d coefficients determining y.

5. Fix a basis for the solution space, and pick random solutions until the picked solution $\tilde{b}$ is invertible.
6. *Output:* $\tilde{a}^{-1}\tilde{b}^{-1}\tilde{a}\tilde{b}$.

That step (3) terminates quickly follows from the Invertibility Lemma [24]. We prove that the output is correct. As $\tilde{b} \in \mathrm{Alg}(B)$, we have by Lemma 3 that $\tilde{b}^{\tilde{a}} = \tilde{b}^{a}$, and therefore

$$(\tilde{b}^{-1})^{\tilde{a}} = (\tilde{b}^{\tilde{a}})^{-1} = (\tilde{b}^{a})^{-1} = (\tilde{b}^{-1})^{a}.$$

It follows that

$$\tilde{a}^{-1}\tilde{b}^{-1}\tilde{a}\tilde{b} = (\tilde{b}^{-1})^{\tilde{a}}\tilde{b} = (\tilde{b}^{-1})^{a}\tilde{b} = a^{-1}\tilde{b}^{-1}a\tilde{b} = a^{-1}a^{\tilde{b}}.$$

As $a \in \mathrm{Alg}(A)$, we have by Lemma 3 that $a^{\tilde{b}} = a^{b}$, and thus

$$\tilde{a}^{-1}\tilde{b}^{-1}\tilde{a}\tilde{b} = a^{-1}a^{b} = a^{-1}b^{-1}ab.$$

Complexity. The step with linear equations computes the nullspace of a $kn^2 \times d$ matrix. Thus, its complexity is $O(\frac{kn^2}{d}d^{\omega}) = O(kn^2d^{\omega-1})$, which is dominated by the complexity $O(kd^2n^2)$ of computing the algebraic spans. In the actual proposal [2], the dimension d is $O(n^2)$, and the complexity becomes $O(kn^6)$. The parameter k is typically $\sqrt{n}$ (the number of Artin generators in the braid group $\mathbf{B}_N$).

Strikingly, the previous cryptanalysis, by the linear centralizer method, has a much larger complexity: $O(n^8 + kn^6)$, that is typically $O(n^8)$ [24]. Thus, the new cryptanalysis is not only simpler and more general, but also more efficient.

3.2 The Centralizer key exchange protocol

For a group G and an element $g \in G$, the *centralizer of g in G* is the set

$$C_G(g) := \{h \in G : gh = hg\}.$$

The *Centralizer key exchange protocol*, introduced by Shpilrain and Ushakov in 2006 [21], is described in Figure 2. In this protocol, a_1 commutes with b_1 and a_2 commutes with b_2. Consequently, the keys computed by Alice and Bob are identical, and they are equal to the group element $a_1b_1ga_2b_2$.

The security of the Centralizer key exchange protocol is determined by the difficulty of the following problem.

Problem 4 *Let $G \leq \mathrm{GL}_n(\mathbb{F})$, $g, a_1, b_2 \in G$, $g_1, \ldots, g_k \in C_G(a_1)$, $h_1, \ldots, h_k \in C_G(b_2)$, $a_2 \in \langle h_1, \ldots, h_k \rangle$, and $b_1 \in \langle g_1, \ldots, g_k \rangle$.*
Given the group elements $g, g_1, \ldots, g_k, h_1, \ldots, h_k, a_1ga_2, b_1gb_2$, compute the product $a_1b_1ga_2b_2$.

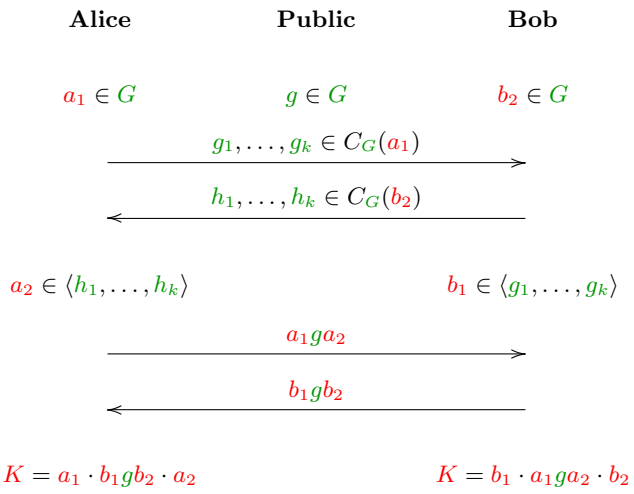

Fig. 2. The Centralizer key exchange protocol

The algebraic span method applies, provably, to this problem: We note that $a_1^{-1}(a_1 g a_2) = g a_2$. Find a solution to the system

$$\begin{aligned} x(a_1 g a_2) &= gy \\ x g_1 &= g_1 x \\ &\vdots \\ x g_k &= g_k x \end{aligned}$$

with x invertible and $y \in \mathrm{Alg}(\{h_1, \ldots, h_k\})$. In practice, we may start with y which has d variables, and this determines x and then we solve for x.

Let $(\tilde{a}_1, \tilde{a}_2) = (x^{-1}, y)$. Then $\tilde{a}_1 g \tilde{a}_2 = x^{-1} g y = a_1 g a_2$. As the element $\tilde{a}_1 = x^{-1}$ commutes with all elements $g_1, \ldots, g_k$, it also commutes with b_1. As b_2 commutes with $h_1, \ldots, h_k$ and $\tilde{a}_2 \in \mathrm{Alg}(\{h_1, \ldots, h_k\})$, we have $b_2 \tilde{a}_2 = \tilde{a}_2 b_2$. Thus,

$$\tilde{a}_1 b_1 g b_2 \tilde{a}_2 = b_1 \tilde{a}_1 g \tilde{a}_2 b_2 = b_1 a_1 g a_2 b_2.$$

Here, too, the complexity is $O(kd^2n^2)$.

3.3 The Braid Diffie–Hellman key exchange protocol and the Double Coset key exchange protocol

The *Braid Diffie–Hellman key exchange protocol*, introduced by Ko, Lee, Cheon, Han, Kang and Park [9], is illustrated in Figure 3. For subsets A, B of a group G, that notation $[A, B] = 1$ means that the sets A and B commute elementwise, that is, a and b commute ($ab = ba$) for all elements $a \in A$ and $b \in B$. Since, in the Braid Diffie–Hellman key exchange protocol, the subgroups A and B of G commute element-wise, the keys computed by Alice and Bob are identical.

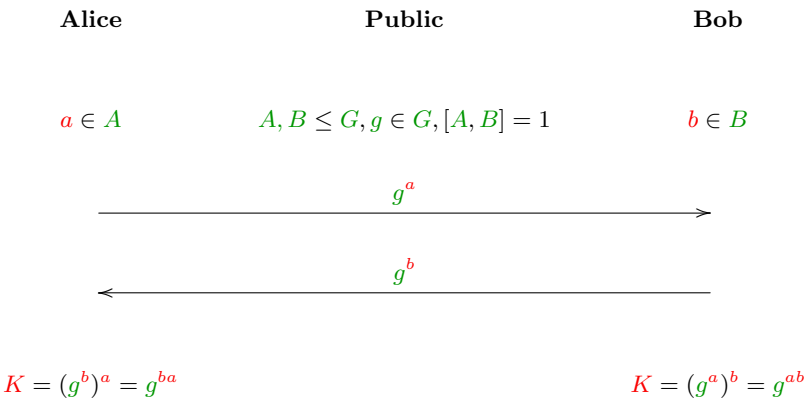

Fig. 3. The Braid Diffie–Hellman key exchange protocol

The security of the Braid Diffie–Hellman key exchange protocol for a platform group G (Figure 3) is captured by the following problem.

Problem 5 *Let A and B be subgroups of $\mathrm{GL}_n(\mathbb{F})$ with $[A, B] = 1$, and an element $g \in \mathrm{GL}_n(\mathbb{F})$ be given.*
Given a pair (g^a, g^b) where $a \in A$ and $b \in B$, find g^{ab}.

As with all problems in this paper, the original problem is stated for Artin's Braid group, and it is known that it reduces to the same problem in matrix groups over finite fields [5]. We solve it for matrix groups.

To apply the algebraic span method to this problem, solve the equation $g^x = g^a$ subject to the linear constraint $x \in \mathrm{Alg}(A)$, and pick an invertible solution $\tilde{a}$. Then

$$(g^b)^{\tilde{a}} = g^{b\tilde{a}} = g^{\tilde{a}b} = (g^{\tilde{a}})^b = (g^a)^b = g^{ab}.$$

Again, the complexity of the solution is dominated by the computation of $\mathrm{Alg}(A)$.

A generalization of the Braid Diffie–Hellman key exchange protocol was proposed by Cha, Ko, Lee, Han and Cheon [4]. A variation of this protocol was proposed in 2005, by Shpilrain and Ushakov [20]. These protocols are both special cases of the *Double Coset key exchange protocol*, illustrated in Figure 4.

One may state the underlying problem as before. Here is how to solve it: Solve the equation $x_1(a_1 g a_2) = g x_2$ subject to $x_1 \in \mathrm{Alg}(A_1)$ and $x_2 \in \mathrm{Alg}(A_2)$, with x_1 invertible. Let $(\tilde{a}_1, a_2) = (x_1^{-1}, x_2)$. Then

$$\tilde{a}_1(b_1 g b_2)\tilde{a}_2 = b_1\tilde{a}_1 g \tilde{a}_2 b_2 = b_1 a_1 g a_2 b_2.$$

The complexity is the same as in the previous solutions.

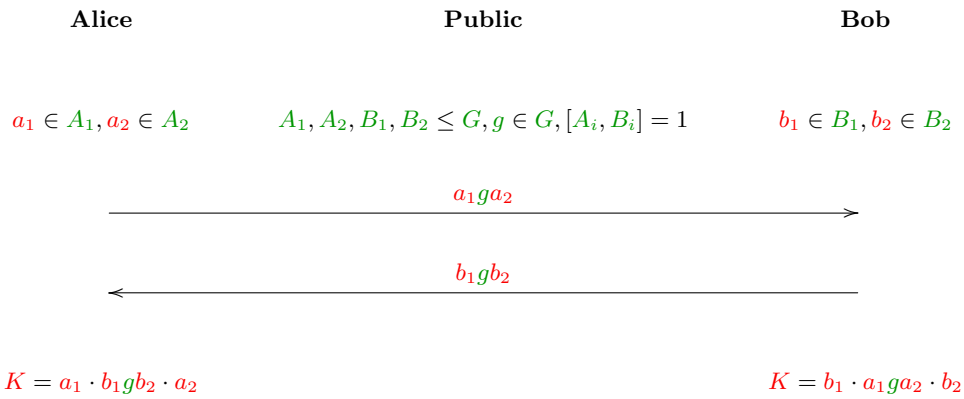

Fig. 4. The Double Coset key exchange protocol

3.4 Stickel's key exchange protocol

We conclude with an example where the complexity of the cryptanalysis is surprisingly small. The key exchange protocol described in Figure 5 was introduced by Stickel in 2005 [22].

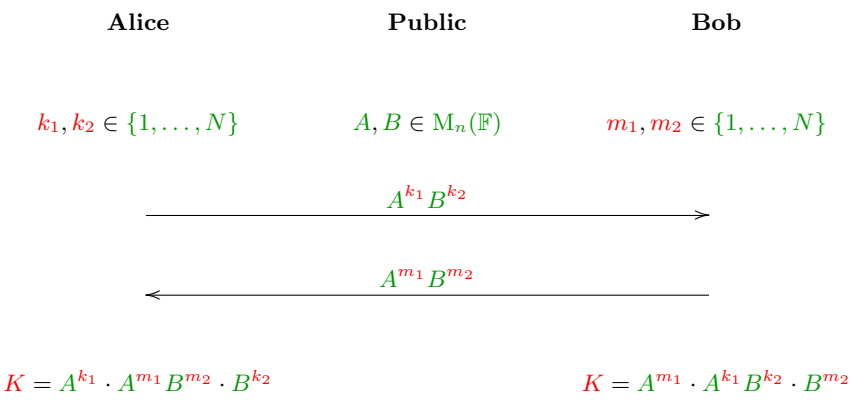

Fig. 5. Stickel's key exchange protocol

A successful heuristic cryptanalysis of complexity roughly $n^{2\omega}$ was presented by Shpilrain [19]. Shpilrain's cryptanalysis turned out provable [24]. The algebraic span method provides a simple alternative, of smaller complexity.

The dimension of the *algebras* spanned by the matrices A and B is, by the Cayley–Hamilton Theorem, at most n. Find a matrix $\tilde{A} \in \mathrm{Alg}(\{A\})$ and an invertible matrix $D \in \mathrm{Alg}(\{B\})$ satisfying the linear equation $\tilde{A} = A^{k_1} B^{k_2} D$. Since the dimension is $O(n)$, the complexity is $O(n^4)$. Let $\tilde{B} = D^{-1}$. A cyclic

algebra is abelian. Moreover, the matrix $\tilde{B}$ is a finite power of D, and is thus in $\mathrm{Alg}(\{B\})$. Thus,

$$\tilde{A} \cdot A^{m_1} B^{m_2} \cdot \tilde{B} = A^{m_1} \tilde{A} \tilde{B} B^{m_2} = A^{m_1} A^{k_1} B^{k_2} B^{m_2} = K.$$

The overall complexity is just $O(n^4)$.

Variations of this key exchange protocol are proposed every now and then (for example, [1]), and are all subject to the cryptanalysis presented here.

4 Cryptanalysis of the Triple Decomposition key exchange protocol

Kurt's *Triple Decomposition* key exchange protocol [11, 13] is described in Figure 6. In this figure, uppercase letters denote subgroups. An edge between two subgroups means that these subgroups commute elementwise. This ensures that the keys computed by Alice and Bob are both equal to $ab_1a_1b_2a_2b$.

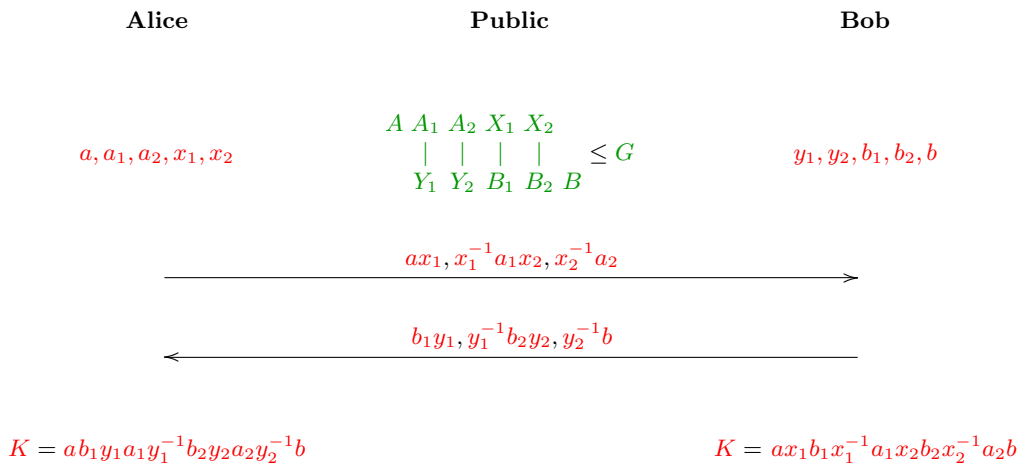

Fig. 6. The Triple Decomposition key exchange protocol

Let $c := x_1^{-1}a_1x_2$. By moving the matrix x_1 or x_2 to the other side of the equation, the public information $x_1^{-1}a_1x_2$ provides a quadratic equation, and similarly for the public information $y_1^{-1}b_2y_2$. Solving quadratic equations may be very difficult. This prevented the application of earlier methods to this key exchange protocol. The natural approach would be to ignore this part of the pubic information, and solve the linear equations provided by the other public items. This works for generic matrix groups, but fails, according to our experiments, for the actual groups proposed in Kurt's paper [11]. We provide here a

way that takes the triple products into account, in a linear way, which still provably obtains the correct key. In the framework of algebraic spans, this solution is natural.

The following sets can be computed from the public information:

$$\begin{aligned}
\mathrm{Alg}(B_1)y_1 &= \mathrm{Alg}(B_1)\cdot b_1 y_1\\
\mathrm{Alg}(B_2\cup Y_2)y_1 &= \mathrm{Alg}(B_2\cup Y_2)\cdot y_2^{-1}b_2^{-1}y_1 = \mathrm{Alg}(B_2\cup Y_2)\cdot (y_1^{-1}b_2y_2)^{-1}\\
\mathrm{Alg}(A_2)x_2 &= \mathrm{Alg}(A_2)\cdot a_2^{-1}x_2\\
\mathrm{Alg}(A_1\cup X_1)x_2 &= \mathrm{Alg}(A_1\cup X_1)\cdot x_1^{-1}a_1x_2
\end{aligned}$$

The invertible matrices y_1 and x_2 are, respectively, in the following intersections of subspaces of $\mathrm{M}_n(\mathbb{F})$:

$$\begin{aligned}
&\mathrm{Alg}(Y_1)\cap \mathrm{Alg}(B_1)y_1\cap \mathrm{Alg}(B_2\cup Y_2)y_1;\\
&\mathrm{Alg}(X_2)\cap \mathrm{Alg}(A_2)x_2\cap \mathrm{Alg}(A_1\cup X_1)x_2.
\end{aligned}$$

By the Invertibility Lemma [24, Lemma 9], we can pick invertible elements $\tilde{y}_1$ and $\tilde{x}_2$ in these intersections, respectively. Then:

1. Since the elements y_1 and $\tilde{y}_1$ are in $\mathrm{Alg}(Y_1)$, they commute with the elements of the subgroup A_1.
2. Since $\tilde{y}_1\in \mathrm{Alg}(B_1)y_1$, we have $\tilde{y}_1y_1^{-1}\in \mathrm{Alg}(B_1)$, and thus the quotient $\tilde{y}_1y_1^{-1}$ commutes with the elements of the subgroup X_1. By (1), this quotient also commutes with the elements of the subgroup A_1.
3. Since $\tilde{y}_1\in \mathrm{Alg}(B_2\cup Y_2)y_1$, we have $\tilde{y}_1y_1^{-1}\in \mathrm{Alg}(B_2\cup Y_2)$.

Similarly, we have:

1. The elements x_2 and $\tilde{x}_2$ commute with the elements of the subgroup B_2.
2. The quotient $\tilde{x}_2x_2^{-1}$ commutes with the elements of the union $Y_2\cup B_2$.
3. The quotient $\tilde{x}_2x_2^{-1}$ is in $\mathrm{Alg}(A_1\cup X_1)$.

It suffices to use one of the items numbered (3). We will use here the former.

Using the public information, compute

$$\tilde{K} := ax_1\cdot b_1y_1\cdot \tilde{y}_1^{-1}\cdot x_1^{-1}a_1x_2\cdot \tilde{x}_2^{-1}\cdot \tilde{y}_1\cdot y_1^{-1}b_2y_2\cdot \tilde{x}_2\cdot x_2^{-1}a_2\cdot y_2^{-1}b.$$

We claim that $\tilde{K}=K=ab_1a_1b_2a_2b$, the key that Alice and Bob established.

Since the subgroups X_1 and B_1 commute elementwise, and $\tilde{y}_1y_1^{-1}\in \mathrm{Alg}(B_1)$, we have

$$x_1\cdot b_1\cdot y_1\tilde{y}_1^{-1}\cdot x_1^{-1} = b_1y_1\tilde{y}_1^{-1}.$$

Since the quotient $\tilde{x}_2x_2^{-1}$ commutes with the elements of the union $Y_2\cup B_2$ and $\tilde{y}_1y_1^{-1}\in \mathrm{Alg}(B_2\cup Y_2)$, we have

$$x_2\tilde{x}_2^{-1}\cdot \tilde{y}_1y_1^{-1}\cdot b_2\cdot y_2\cdot \tilde{x}_2x_2^{-1} = \tilde{y}_1y_1^{-1}b_2y_2.$$

Thus,

$$\tilde{K} = ab_1y_1\tilde{y}_1^{-1}a_1\tilde{y}_1y_1^{-1}b_2y_2a_2y_2^{-1}b.$$

Since the subgroups Y_2 and A_2 commute elementwise, we have

$$y_2 a_2 y_2^{-1} = a_2.$$

Since the quotient $\tilde{y}_1 y_1^{-1}$ commutes with the elements of the subgroup A_1, we have

$$y_1 \tilde{y}_1^{-1} \cdot a_1 \cdot \tilde{y}_1 y_1^{-1} = a_1.$$

It follows that

$$\tilde{K} = a b_1 a_1 b_2 a_2 b,$$

as required.

As in all of our previous examples, the complexity of this cryptanalysis is dominated by the calculation of the algebraic spans, which is $O(kd^2n^2)$, where k the maximum number of generators of the given subgroups, and d is the maximum dimension of the Algebra generated by them. In particular, it is not greater than $O(kn^6)$.

5 Specifications and implementation

5.1 Artin's braid group $\mathbf{B}_N$

All key exchange protocols addressed in this paper use Artin's *braid group* $\mathbf{B}_N$ as the underlying group. This group is parameterized by a natural number N. Elements of $\mathbf{B}_N$ can be identified with braids on N strands. Braid group multiplication is motivated geometrically, but the details play no role in the present paper. We provide here the necessary details, following the earlier paper [24].

Let S_N be the symmetric group of permutations on N symbols. For our purposes, the braid group $\mathbf{B}_N$ is a group of elements of the form $(i, \mathbf{p})$, where i is an integer, and $\mathbf{p}$ is a finite (possibly, empty) sequence of elements of S_N. In other words, $\mathbf{p} = (p_1, \ldots, p_\ell)$ for some $\ell \geq 0$ and $p_1, \ldots, p_\ell \in S_N$. The sequence $\mathbf{p} = (p_1, \ldots, p_\ell)$ is requested to be *left weighted* (a property whose definition will not be used here), and p_1 must not be the involution $p(k) = N - k + 1$.[1]

For "generic" braids $(i, (p_1, \ldots, p_\ell)) \in \mathbf{B}_N$, i is negative and $|i|$ is $O(\ell)$, but this is not always the case. Note that the bit-length of an element $(i, (p_1, \ldots, p_\ell)) \in \mathbf{B}_N$ is $O(\log |i| + \ell N \log N)$.

Multiplication is defined on $\mathbf{B}_N$ by an algorithm of complexity $O(\ell^2 N \log N + \log |i|)$. Inversion is of linear complexity. Explicit implementations are provided, for example, in [4].

[1] For readers familiar with the braid group, we point out that the sequence $(i, (p_1, \ldots, p_\ell))$ encodes the left normal form $\Delta^i p_1 \cdots p_\ell$ of the braid, in Artin's presentation, with Δ being the fundamental, half twist braid on N strands.

5.2 Infimum reduction

The *infimum* of a braid $b = (i, \mathbf{p})$ is the integer $\inf(b) := i$. As the bit-length of b is $O(\log|i| + \ell N \log N)$, an algorithm polynomial in $|i|$ would be at least *exponential* in the bit-length. This obstacle is eliminated by reducing the infimum [24]. We demonstrate this for the Triple Decomposition key exchange protocol (Section 4).

In cases where $\mathbf{p}$ is the empty sequence, we write (i) instead of $(i, \mathbf{p})$. The properties of the braid group $\mathbf{B}_N$ include, among others, the following ones.

(a) $(i) \cdot (j, \mathbf{p}) = (i + j, \mathbf{p})$ for all integers i and all $(j, \mathbf{p}) \in \mathbf{B}_N$.
In particular, $(i) = (1)^i$ for all i.
(b) $(2) \cdot (i, \mathbf{p}) = (i, \mathbf{p}) \cdot (2)$ for all for all $(i, \mathbf{p}) \in \mathbf{B}_N$.

Thus, $(2j)$ is a central element of $\mathbf{B}_N$ for each integer j. If follows that, for each $(i, \mathbf{p}) \in \mathbf{B}_N$,

$$(i, \mathbf{p}) = (i - (i \bmod 2)) \cdot (i \bmod 2, \mathbf{p}).$$

This way, every braid $x \in \mathbf{B}_N$ decomposes to a unique product $c_x \underline{x}$, where c_b is of the form $(2j)$ (and thus *central*), and $\inf(\underline{b}) \in \{0, 1\}$.

Consider the information in Figure 6. Since

$$K = ab_1y_1a_1y_1^{-1}b_2y_2a_2y_2^{-1}b = ax_1b_1x_1^{-1}a_1x_2b_2x_2^{-1}a_2b,$$

The central elements $c_{y_1}, c_{y_2}, c_{x_1}, c_{x_2}$ get canceled and do not affect the shared key. Thus, we may assume that the infimum of the braids y_1, y_2, x_1, x_2 is 0 or 1. Decompose the central parts out of the public information:

$$\begin{array}{ll} ax_1 = c_1\underline{ax_1}, & y_2^{-1}b = d_1\underline{y_2^{-1}b}, \\ x_1^{-1}a_1x_2 = c_2\underline{x_1^{-1}a_1x_2}, & y_1^{-1}b_2y_2 = d_2\underline{y_1^{-1}b_2y_2}, \\ x_2^{-1}a_2 = c_3\underline{x_2^{-1}a_2}, & b_1y_1 = d_3\underline{b_1y_1}. \end{array}$$

The central elements are known, given the public information. Then

$$\begin{aligned} K &= ax_1 \cdot b_1y_1 \cdot y_1^{-1} \cdot x_1^{-1}a_1x_2 \cdot x_2^{-1}y_1 \cdot y_1^{-1}b_2y_2 \cdot x_2 \cdot x_2^{-1}a_2 \cdot y_2^{-1}b \\ &= c_1\underline{ax_1}\ d_3\underline{b_1y_1}\ y_1^{-1}\ c_2\underline{x_1^{-1}a_1x_2}\ x_2^{-1}\ y_1\ d_2\underline{y_1^{-1}b_2y_2}\ x_2\ c_3\underline{x_2^{-1}a_2}\ d_1\underline{y_2^{-1}b} \\ &= c_1c_2c_3d_1d_2d_3\ \underline{ax_1}\ \underline{b_1y_1}\ y_1^{-1}\ \underline{x_1^{-1}a_1x_2}\ x_2^{-1}\ y_1\ \underline{y_1^{-1}b_2y_2}\ x_2\ \underline{x_2^{-1}a_2}\ \underline{y_2^{-1}b} \\ &=: c_1c_2c_3d_1d_2d_3K'. \end{aligned}$$

Assume that we have an algorithm for computing the shared key out of the public information, that succeeds when the public braids have infimum 0 or 1. Applying this algorithm to the reduced public braids, we obtain the braid K'. Multiplying by the known central braid $c_1c_2c_3d_1d_2d_3$, we obtain the original key K. Thus, we may assume that all public braids, as well as the secret braids y_1, y_2, x_1, x_2 have infimum 0 or 1. Assume that, henceforth.

For a braid $x = (i, \mathbf{p})$, let $\ell(x)$ be the number of permutations in the sequence $\mathbf{p}$. For integers i, s, let

$$[i, s] = \{x \in \mathbf{B}_N \,:\, i \le \inf(x) \le \inf(x) + \ell(x) \le s\}.$$

We use the following basic facts about $\mathbf{B}_N$:

1. If $x_1 \in [i_1, s_1]$ and $x_2 \in [i_2, s_2]$, then $x_1 x_2 \in [i_1 + i_2, s_1 + s_2]$.
2. If $x \in [i, s]$, then $x^{-1} \in [-s, -i]$.

By our assumption, the key K is a product of 10 braids with infimum 0 or 1, and thus

$$0 \leq \inf(K) \leq 10.$$

Let ℓ be the maximum of the lengths of the private braids in Figure 6. In the above reduction, we had $K = c_1 c_2 c_3 d_1 d_2 d_3 K'$, and thus

$$\ell(K') = \ell(K) = \ell(a b_1 a_1 b_2 a_2 b) \leq 6\ell.$$

5.3 Reducing to a matrix group over a finite field

Let n be a natural number. As usual, we denote the algebra of all $n \times n$ matrices over a field $\mathbb{F}$ by $\mathrm{M}_n(\mathbb{F})$, and the group of invertible elements of this algebra by $\mathrm{GL}_n(\mathbb{F})$. A *matrix group* is a subgroup of $\mathrm{GL}_n(\mathbb{F})$. A *faithful representation* of a group G in $\mathrm{GL}_n(\mathbb{F})$ is a group isomorphism from G onto a matrix group $H \leq \mathrm{GL}_n(\mathbb{F})$. A group is *linear* if it has a faithful representation.

Bigelow and Krammer, established in their breakthrough papers [**?**, **?**] that the braid group $\mathbf{B}_N$ is linear, by proving that the so-called *Lawrence–Krammer representation*

$$\mathrm{LK} \colon \mathbf{B}_N \longrightarrow \mathrm{GL}_{\binom{N}{2}}(\mathbb{Z}[t^{\pm 1}, \frac{1}{2}]),$$

whose dimension is

$$n := \binom{N}{2},$$

is injective. The Lawrence–Krammer representation of a braid can be computed in polynomial time. This representation is also invertible in (similar) polynomial time [**?**, 5].

Theorem 6 (Cheon–Jun [5]) *Let $x \in [i, s]$ in $\mathbf{B}_N$. Let $M \geq \max(|i|, |s|)$. Then:*

1. *The degrees of t in $\mathrm{LK}(x) \in \mathrm{GL}_n(\mathbb{Z}[t^{\pm 1}, \frac{1}{2}])$ are in $\{-M, -M+1, \ldots, M\}$.*
2. *The rational coefficients $\frac{c}{2^d}$ in $\mathrm{LK}(x)$ (c integer, d nonnegative integer) satisfy: $|c| \leq 2^{N^2 M}, |d| \leq 2NM$.*

In the notation of Theorem 6, Theorem 2 in Cheon–Jun [5] implies that inversion of $\mathrm{LK}(x)$ is of order $N^6 \log M$ multiplications of entries. Ignoring logarithmic factors and thus assuming that each entry multiplication costs $NM \cdot N^2 M = N^3 M^2$, this accumulates to $N^8 M^2$. We also invert the function LK as part of our cryptanalysis. However, the complexity of the computation of the algebraic spans dominates the complexity of these transformations, that are applied only at the beginning and at the end of the cryptanalysis.

Let us return to the Triple Decomposition key exchange protocol. After infimum reduction (who's complexity is negligible), we have

$$K \in [0, 10 + 6\ell].$$

Let $M := 10 + 6\ell$. By the Cheon–Jun Theorem, we have

$$(2^{2NM} t^M) \cdot \mathrm{LK}(K) \in \mathrm{GL}_n(\mathbb{Z}[t]),$$

the absolute values of the coefficients in this matrix are bounded by $2^{N^2(M+1)}$, and the maximal degree of t in this matrix is bounded by $2M$.

Let p be a prime slightly greater than 2^{N^2M+2NM}, and $f(t)$ be an irreducible polynomial over $\mathbb{Z}_p$, of degree d slightly larger than $2M$. Then

$$(2^{2NM} t^M) \cdot \mathrm{LK}(K) = (2^{2NM} t^M) \cdot \mathrm{LK}(K) \bmod (p, f(t)) \in \mathrm{GL}_n(\mathbb{Z}[t]/\langle p, f(t)\rangle),$$

under the natural identification of $\{-(p-1)/2, \ldots, (p-1)/2\}$ with $\{0, \ldots, p-1\}$.

Let $\mathbb{F} = \mathbb{Z}[t]/\langle p, f(t)\rangle = \mathbb{Z}[t^{\pm 1}, \frac{1}{2}]/\langle p, f(t)\rangle$. $\mathbb{F}$ is a finite field of cardinality p^d, where d is the degree of $f(t)$. It follows that the complexity of field operations in $\mathbb{F}$ is, up to logarithmic factors, of order

$$d^2 \log p = O(M^3 N^2).$$

Thus, the key K can be recovered as follows:

1. Apply the composed function $\mathrm{LK}(x) \bmod (p, f(t))$ to the input braids, to obtain a version of this problem in $\mathrm{GL}_n(\mathbb{F})$.
2. Solve the problem there, to obtain $\mathrm{LK}(K) \bmod (p, f(t))$.
3. Compute $(2^{2NM} t^M) \cdot \mathrm{LK}(K) \bmod (p, f(t)) = (2^{2NM} t^M) \cdot \mathrm{LK}(K)$.[2]
4. Divide by $(2^{2NM} t^M)$ to obtain $\mathrm{LK}(K)$.
5. Compute K using the Cheon–Jun inversion algorithm.

The complexity of this *preliminary* cryptanalysis is $O(kn^6)$ field operations, where k is the maximum number of generators in the given subgroups. Roughly, this is $kn^6 = kN^{12} \cdot M^3 N^2 = kN^{14}(10 + 6\ell)^3$.

5.4 Reducing the complexity

To make this cryptanalysis feasible, at least for mildly large parameters, we can improve upon the field multiplication complexity. We do this by applying the Chinese Remainder Theorem (CRT) on both the integer part and the polynomial part. Let

$$p_1, p_2, \cdots = 2, 3, 5, \ldots,$$

the sequence of prime numbers. Also, consider the relatively prime polynomials, or degree 1,

$$x, x \pm 1, x \pm 2, \ldots$$

[2] The equality here is over the integers.

We take just enough primes so that their products exceeds $2^{N^2(M+1)}$, and we take the first $2M$ polynomials in our list.

For each pair $(p, f(t))$ of a prime and a polynomial in our lists, we reduce modulo the prime and the polynomial, and apply the cryptanalysis, over the resulting p-element field. In the end, we combine all results using the CRT. Since CRT is done only once, its complexity is dominated by the complexity of the linear span calculations.

Since we ignore logarithmic factors, it suffices to estimate the complexity of the same algorithm, but using just a single prime $p \approx 2^{N^2M}$. For each linear polynomial, the obtained field size is p, and thus field multiplication is, up to logarithmic factors, of complexity N^2M. We need to repeat this M times, so the overall complexity is, roughly, of order

$$M \cdot kN^{12} \cdot N^2M^2 = kN^{14}M^3 \approx kN^{14}\ell^3.$$

5.5 Specifications of the Triple Decomposition key exchange protocol

We now describe the groups proposed for the actual specification of the Triple Decomposition key exchange protocol. Let $m \geq 2$ be a natural number, and $N := 3m+1$. The braid group $\mathbf{B}_N$ is generated by $N-1$ generators, $\sigma_1, \ldots, \sigma_{N-1}$. One of their defining relations is that σ_i and σ_j commute whenever $|i-j| > 1$. The groups in Figure 6 are chosen as follows: Fix "generic" braids $g_1, g_2, h_1, h_2 \in \mathbf{B}_N$. Then:

$$\begin{aligned}
A &= B = \mathbf{B}_N \\
A_1 &= \langle {\sigma_1}^{g_1}, \ldots, {\sigma_{m-1}}^{g_1} \rangle; \quad Y_1 = \langle {\sigma_{m+1}}^{g_1}, \ldots, {\sigma_{mk}}^{g_1} \rangle \\
A_2 &= \langle {\sigma_1}^{g_2}, \ldots, {\sigma_{m-1}}^{g_2} \rangle; \quad Y_2 = \langle {\sigma_{m+1}}^{g_2}, \ldots, {\sigma_{3m}}^{g_2} \rangle \\
X_1 &= \langle {\sigma_1}^{h_1}, \ldots, {\sigma_{2m-1}}^{h_1} \rangle; \quad B_1 = \langle {\sigma_{2m+1}}^{h_1}, \ldots, {\sigma_{3m}}^{h_1} \rangle \\
X_2 &= \langle {\sigma_1}^{h_2}, \ldots, {\sigma_{2m-1}}^{h_2} \rangle; \quad B_2 = \langle {\sigma_{2m+1}}^{h_2}, \ldots, {\sigma_{3m}}^{h_2} \rangle
\end{aligned}$$

The conjugations prevent an otherwise trivial cryptanalysis [11]. It follows that the parameter k in the complexity estimation is $2m$. This is the same order as $N = 3m$. Thus, the complexity of the cryptanalysis is, roughly, of order $N^{15}\ell^3$.

The value ℓ depends on the way the secret braids are generated. This is was never specified exactly. Comparing to more detailed proposals, it is fair to estimate that ℓ is of order much smaller than N.

5.6 Implementation

This type of provable cryptanalyses is generally considered of theoretical interest only [5, 24]. The algorithmic shortcuts described above made it possible, for the first time, to launch our attack on concrete instances, including ones where brute force or naive attacks are infeasible. Being provable, the attacks must find the shared key in all tests, and this provides a "sanity check" for our mathematical

reasoning. The attacks were implemented on the computational algebra software MAGMA [3], with no optimizations beyond those specified above. Infimum reduction was not implemented, since for generic braids it has little effect.

For a length parameter l, we chose the braids g_1, g_2, h_1, h_2 as products of l random elements from the set $\{\sigma_1^{\pm 1}, \ldots, \sigma_{N-1}^{\pm 1}\}$. The braids in the subgroup were generated as products of l random generators of that subgroup. Since the running time was long, and we already know that the attacks provable succeed, we conducted only one attack for each set of parameters. Each attack was launched on a single core of a standard desktop CPU. The results are summarized in Tables 1 and 2.

Table 1. Experimental results with MAGMA, single CPU core, $N = 10 = 3 \cdot 3 + 1$

Length	**Time**	**Memory** (MB)	**Key recovered?**
2	114 sec	30	Yes
4	10 min	33	Yes
8	38 min	38	Yes
16	3.5 hrs	108	Yes
32	49 hrs	249	Yes
64	629 hrs	640	Yes

Table 2. Experimental results with MAGMA, single CPU core, $N = 13 = 3 \cdot 4 + 1$

Length	**Time**	**Memory** (MB)	**Key recovered?**
2	9 min	195	Yes
4	12 min	201	Yes
8	5 hrs	215	Yes
16	19 hrs	548	Yes
32	298 hrs	1289	Yes

The attacks are highly parallelable. Larger parameters would necessitate parallel implementations over large grids.

6 Conclusions

We have introduced algebraic span cryptanalysis, a provable method for cryptanalysing nonabelian cryptographic protocols and, more generally, solving computational problems in groups. This method applies to all groups with efficient, faithful representations as matrix groups. The examples provided demonstrate the power, generality, and simplicity of this method.

The novelty of this method is demonstrated by showing that it applies to a protocol that was not approachable by earlier methods. The new method cleared out much of the difficulty of the computational problem behind the Triple Decomposition key exchange protocol, and made it possible for us to find the extra idea to make it work.

Initially considered of theoretical interest only, provable cryptanalysis is now a feasible threat to nonabelian cryptographic protocols. It seems very challenging to devise a nonabelian key exchange protocol that cannot be cryptanalyzed by the algebraic span method.

Acknowledgments

We thank Avraham (Rami) Eizenbud and Craig Gentry for intriguing discussions. A part of this work was carried out while the third named author was on Sabbatical at the Weizmann Institute of Science. This author thanks his hosts for their kind hospitality.

References

1. M. Andrecut, *A matrix public key cryptosystem*, arXiv eprint 1506.00277, 2015.
2. I. Anshel, M. Anshel, D. Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters **6** (1999), 287–291.
3. W. Bosma, J. Cannon, C. Playoust, *The Magma algebra system. I. The user language*, Journal of Symbolic Computation **24** (1997), 235–265.
4. J. Cha, K. Ko, S. Lee, J. Han, J. Cheon, *An efficient implementation of braid groups*, ASIACRYPT 2001, LNCS **2248** (2001), 144–156.
5. J. Cheon, B. Jun, *A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem*, CRYPTO 2003, LNCS **2729** (2003), 212–224.
6. R. Gilman, A. Myasnikov, A. Myasnikov, A. Ushakov, *New developments in Commutator Key Exchange*, Proceedings of the First International Conference on Symbolic Computation and Cryptography, Beijing, 2008, 146–150. `http://www-calfor.lip6.fr/~jcf/Papers/scc08.pdf`
7. M. González–Vasco, R. Steinwandt, **Group Theoretic Cryptography**, Cryptography and Network Security Series, Chapman and Hall/CRC Press, 2015.
8. D. Holt, answer to MathOverflow question `http://mathoverflow.net/questions/154761`
9. K. Ko, S. Lee, J. Cheon, J. Han, J. Kang, C. Park, *New public-key cryptosystem using braid groups*, CRYPTO 2000, LNCS **1880** (2000), 166–183.
10. Y. Kurt, *A new key exchange primitive based on the triple decomposition problem*, IACR eprint 2006/378.
11. Y. Kurt Peker, *A new key agreement scheme based on the triple decomposition problem*, International Journal of Network Security **16** (2014), 340–350.
12. A. Myasnikov, V. Shpilrain, A. Ushakov, **Group-based cryptography**, Birkhäuser, 2008.
13. A. Myasnikov, V. Shpilrain, A. Ushakov, **Non-commutative Cryptography and Complexity of Group-theoretic Problems**, American Mathematical Society Surveys and Monographs **177**, 2011.

14. A. Myasnikov, V. Romankov, *A linear decomposition attack*, Groups Complexity Cryptology **7** (2015), 81–94.
15. V. Roman'kov, **Algebraic cryptography**, Omsk State Dostoevsky University, 2013. (In Russian)
16. V. Roman'kov, *Cryptanalysis of some schemes applying automorphisms*, Prikladnaya Discretnaya Matematika **3** (2013), 35–51. (In Russian)
17. V. Roman'kov, *A nonlinear decomposition attack*, Groups Complexity Cryptology **8** (2016), 197–207.
18. V. Roman'kov, A. Obzor, *A general encryption scheme using multiplica- tions with cryptanalysis*, Prikladnaya Discretnaya Matematika 37 (2017), 52–61. (In Russian)
19. V. Shpilrain, *Cryptanalysis of Stickel's key exchange scheme*, in: **Computer Science in Russia**, LNCS 5010 (2008), 283–288.
20. V. Shpilrain, A. Ushakov, *Thompson's group and public key cryptography*, ACNS 2005, LNCS **3531** (2005), 151–164.
21. V. Shpilrain, A. Ushakov, *A new key exchange protocol besed on the decomposition problem*, in: L. Gerritzen, D. Goldfeld, M. Kreuzer, G. Rosenberger and V. Shpilrain, eds., **Algebraic Methods in Cryptography**, Contemporary Mathematics **418** (2006), 161–167.
22. E. Stickel, *A new method for exchanging secret keys*, Proceedings of the Third International Conference on Information Technology and Applications (ICITA05), 2005, 426–430.
23. B. Tsaban, *The Conjugacy Problem: cryptoanalytic approaches to a problem of Dehn*, minicourse, Düsseldorf University, Germany, July–August 2012. `http://reh.math.uni-duesseldorf.de/~gcgta/slides/Tsaban_minicourses.pdf`
24. B. Tsaban, *Polynomial-time solutions of computational problems in noncommutative-algebraic cryptography* Journal of Cryptology **28** (2015), 601–622.