Groups-Complexity-Cryptology
Volume 1 (2009), No. 1, 63–75

Cryptanalysis of the Anshel-Anshel-Goldfeld-Lemieux Key Agreement Protocol

Alex D. Myasnikov
Department of Mathematics, Stevens Institute of Technology,
Hoboken, NJ 07030, USA
amyasnik@stevens.edu

Alexander Ushakov
Department of Mathematics, Stevens Institute of Technology,
Hoboken, NJ 07030, USA
aushakov@stevens.edu

Received: February 20, 2008

The Anshel-Anshel-Goldfeld-Lemieux (abbreviated AAGL) key agreement protocol [1] is proposed to be used on low-cost platforms which constraint the use of computational resources. The core of the protocol is the concept of an Algebraic Eraser^{TM} (abbreviated AE) which is claimed to be a suitable primitive for use within lightweight cryptography. The AE primitive is based on a new and ingenious idea of using an action of a semidirect product on a (semi)group to obscure involved algebraic structures. The underlying motivation for AAGL protocol is the need to secure networks which deploy Radio Frequency Identification (RFID) tags used for identification, authentication, tracing and point-of-sale applications.

In this paper we revisit the computational problem on which AE relies and heuristically analyze its hardness. We show that for proposed parameter values it is impossible to instantiate a secure protocol. To be more precise, in 100% of randomly generated instances of the protocol we were able to find a secret conjugator z generated by the TTP algorithm (part of AAGL protocol).

1. The Colored Burau Key Agreement Protocol

A general mathematical framework of the AAGL protocol is quite complicated. In this paper we try to omit unnecessary details and simplify the notation of [1] as much as possible. We refer an interested reader to [1, Sections 2 and 3] for a complete description. Here we start out by giving a particular implementation of the primitive called the Colored Burau Key Agreement Protocol (CBKAP).

ISSN 1867-1144 / $ 2.50

1.1. A platform group

Fix an integer $n \geq 7$ and a prime p. Let $\mathbf{t} = (t_1, \ldots, t_n)$ be a tuple of formal variables. Define matrices

$$x_1(\mathbf{t}) = \begin{pmatrix} -t_1 & 1 & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}$$

and for $i = 2, \ldots, n-1$

$$x_i(\mathbf{t}) = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & t_i & -t_i & 1 & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

which is the identity matrix except for the ith row where it has successive entries t_i, $-t_i$, 1 with $-t_i$ on the diagonal. We look at the matrices $x_1(\mathbf{t}), \ldots, x_{n-1}(\mathbf{t})$ as elements of the group $GL(n, \mathbb{F}_p(\mathbf{t}))$ of $n \times n$ matrices whose entries are Laurent polynomials over the finite field $\mathbb{F}_p$. The symmetric group S_n on n symbols acts on $GL(n, \mathbb{F}_p(\mathbf{t}))$ by permuting the variables $t_1, \ldots t_n$. We denote the result of the action of $s \in S_n$ on $x \in GL(n, \mathbb{F}_p(\mathbf{t}))$ by ${}^s x$.

The semidirect product $GL(n, \mathbb{F}_p(\mathbf{t})) \rtimes S_n$ of the groups $GL(n, \mathbb{F}_p(\mathbf{t}))$ and S_n relative to the defined action of S_n on matrices $GL(n, \mathbb{F}_p(\mathbf{t}))$ is a set of pairs

$$\{(m, s) \mid m \in GL(n, \mathbb{F}_p(\mathbf{t})),\ s \in S_n\}$$

with multiplication given by

$$(m_1, s_1) \cdot (m_2, s_2) := (m_1 \cdot {}^{s_1} m_2, s_1 \cdot s_2).$$

Denote by $s_i = (i, i+1) \in S_n$ the transposition which interchanges i and $i+1$ and by g_i the element of the semidirect product $GL(n, \mathbb{F}_p(\mathbf{t})) \rtimes S_n$

$$g_i = (x_i(\mathbf{t}), s_i).$$

A subgroup

$$G = \langle g_1, \ldots, g_{n-1} \rangle$$

of $GL(n, \mathbb{F}_p(\mathbf{t})) \rtimes S_n$ is called the *colored Burau group*. The group G is a platform group for the AAGL key agrement protocol.

Recall that the group B_n of n-strand braids has the classical Artin's presentation:

$$B_n = \left\langle \sigma_1, \ldots, \sigma_{n-1} \;\middle|\; \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{if } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{if } |i-j| > 1 \end{array} \right\rangle.$$

A word over the group alphabet $\{\sigma_1, \ldots, \sigma_{n-1}\}$ is called a *braid word*. Any n-strand braid can be represented by a braid word. The length of a shortest braid word representing an element $g \in B_n$ is called the *geodesic length* of g relative to Artin's set of generators and is denoted by $|g|$. The function $|\cdot| : B_n \to \mathbb{N}$ is called the *geodesic length function* on B_n.

Lemma 1.1. *The elements $g_i = (x_i(\mathbf{t}), s_i)$, for $i = 1, 2, \ldots, n-1$, satisfy the braid relations and hence determine a representation of the braid group B_n, i.e., the mapping $\sigma_i \stackrel{\varphi}{\mapsto} g_i$ defines a group epimorphism*

$$\varphi : B_n \to G.$$

Proof. Straightforward check. □

1.2. Action of the platform group on $GL(n, \mathbb{F}_p)$

Fix elements $\tau_1, \ldots, \tau_n \in \mathbb{F}_p$ and define a homomorphism π which maps $GL(n, \mathbb{F}_p(\mathbf{t}))$ into $GL(n, \mathbb{F}_p)$ by assigning the value τ_i to the variable t_i, i.e., by evaluating a matrix at $\tau_1, \ldots, \tau_n$. We call π the *evaluation function*.

> **Assumption on $\tau_1, \ldots, \tau_n$.** We assume that π defines a correct group homomorphism.

Relative to the chosen tuple $\tau_1, \ldots, \tau_n \in \mathbb{F}_p$ and the corresponding function π one can define an action of $GL(n, \mathbb{F}_p(\mathbf{t})) \rtimes S_n$ on $GL(n, \mathbb{F}_p) \times S_n$ by putting

$$(m_1, s_1) \star (m_2, s_2) = (m_1 \cdot \pi({}^{s_1}m_2), s_1 s_2)$$

where $\star$ denotes the action. Indeed, it is not difficult to check that $\star$ is an action and satisfies the property

$$((m_1, s_1) \star (m_2, s_2)) \star (m_3, s_3) = (m_1, s_1) \star ((m_2, s_2) \cdot (m_3, s_3)).$$

We say that (m_1, t_1) and (m_2, t_2) *$\star$-commute* if the equality

$$(\pi(m_1), s_1) \star (m_2, s_2) = (\pi(m_2), s_2) \star (m_1, s_1)$$

holds. The next lemma is obvious.

Lemma 1.2. *Let $w = \prod_{k=1}^{m} (x_{i_k}(\mathbf{t}), s_{i_k})$ and $v = \prod_{p=1}^{l} (x_{j_p}(\mathbf{t}), s_{j_p})$ be such that $|i_k - j_p| > 1$ for every $k = 1, \ldots, m$ and $p = 1, \ldots, l$. Then the elements w and v $\star$-commute.*

1.3. The protocol

Before the parties perform actual transmissions the following data is being prepared by the Third Trusted Party (TTP).

- A matrix $m_0 \in GL(n, \mathbb{F}_p)$ which has an irreducible characteristic polynomial over $\mathbb{F}_p$. The choice of m_0 is not relevant for the purposes of this paper, we refer the reader to [1] for more information on how m_0 can be generated randomly.

- $\star$-commuting subgroups $A = \langle w_1, \ldots, w_\gamma \rangle$ and $B = \langle u_1, \ldots, u_\gamma \rangle$ of the group G. We want to point out that the elements w_i and v_j are given to us as products of generators of G and there inverses, i.e., as formal words in group alphabet $\{g_1, \ldots, g_{n-1}\}$. We prefer this form because it allows us to avoid time consuming matrix multiplication in $GL(n, \mathbb{F}_p(\mathbf{t}))$.

Both the matrix m_0 and subgroups A and B can be chosen only once. Now, the public and private keys are chosen as follows:

Alice's Private Key: is a pair which consists of a matrix of the form

$$n_a = l_1 m_0^{\alpha_1} + l_2 m_0^{\alpha_2} + \ldots + l_r m_0^{\alpha_r} \in GL(n, \mathbb{F}_p)$$

(where $l_1, \ldots, l_r \in \mathbb{F}_p$ and $r, \alpha_1, \ldots, \alpha_r \in \mathbb{Z}^+$) and a random sequence $w_{i_1}^{\varepsilon_1}, \ldots, w_{i_m}^{\varepsilon_m}$ of generators of A and their inverses.

Alice's Public Key: is an element

$$A_{public} = (n_a, id) \star w_{i_1}^{\varepsilon_1} \star \ldots \star w_{i_m}^{\varepsilon_m} \in GL(n, \mathbb{F}_p) \times S_n.$$

Recall that each w_{i_k} is given as a formal product of the generators of G. To perform the $\star$-operation efficiently one should not directly compute w_{i_k}, but consequently apply the factors of w_{i_k} to the argument.

Bob's Private Key: is a pair which consists of a matrix of the form

$$n_b = l'_1 m_0^{\beta_1} + l'_2 m_0^{\beta_2} + \ldots + l'_{r'} m_0^{\beta_{r'}} \in GL(n, \mathbb{F}_p)$$

(where $l'_1, \ldots, l'_{r'} \in \mathbb{F}_p$ and $r', \beta_1, \ldots, \beta_{r'} \in \mathbb{Z}^+$) and a random sequence $v_{j_1}^{\delta_1}, \ldots, v_{j_l}^{\delta_l}$ of generators of B and their inverses.

Bob's Public Key: is a pair

$$B_{public} = (n_b, id) \star v_{j_1}^{\delta_i} \star \ldots \star v_{j_l}^{\delta_l} \in GL(n, \mathbb{F}_p) \times S_n.$$

Again, each v_{j_k} is given as a formal product of the generators of G. To perform the $\star$-operation efficiently one should not directly compute v_{j_k}, but consequently apply the factors of v_{j_k} to the argument.

The shared key: is an element of $GL(n, \mathbb{F}_p) \times S_n$ obtained by Alice in the form

$$[(n_a, id) \cdot B_{public}] \star w_{i_1}^{\varepsilon_1} \star \ldots \star w_{i_m}^{\varepsilon_m}$$

and by Bob in the form

$$[(n_b, id) \cdot A_{public}] \star v_{j_1}^{\delta_i} \star \ldots \star v_{j_l}^{\delta_l}$$

It requires a little work to prove that the obtained elements are indeed equal in $GL(n, \mathbb{F}_p)$. We omit the proof.

1.4. TTP algorithm

The cornerstone part of the proposed key exchange is the choice of $\star$-commuting subgroups of the group G. The basic idea is to use Lemma 1.1 and choose commuting subgroups A and B in B_n and then pull them into G using the epimorphism φ. The resulting subgroups $\varphi(A)$ and $\varphi(B)$ of G commute. Moreover, for any choice of π the subgroups $\varphi(A)$ and $\varphi(B)$ $\star$-commute.

Before we present the algorithm we need to give some details about the braid group B_n. The group B_n has a cyclic center generated by the element Δ^2, where Δ is the element called the half twist and can be expressed in the generators of B_n as follows:

$$\Delta = (\sigma_1 \ldots \sigma_{n-1}) \cdot (\sigma_1 \ldots \sigma_{n-2}) \cdot \ldots \cdot (\sigma_1).$$

Any element $g \in B_n$ can be uniquely represented in the form

$$\Delta^p \xi_1 \ldots \xi_p$$

satisfying certain conditions and called the left *Garside normal form*.

Now, since Δ^2 is a central element, it follows that two elements u, w commute in B_n if and only $u\Delta^{2p}$ and $w\Delta^{2r}$ do (for any choice of $p, r \in \mathbb{Z}$). Hence we may always assume that the normal forms of the generators $\{w_1, \ldots, w_\gamma\}$ and $\{v_1, \ldots, v_\gamma\}$ have the exponent on Δ equal to 0 or -1. When we say that we reduce a braid modulo Δ^2 we mean changing the Δ-power of its normal form to -1 or 0 depending on the parity.

The algorithm below (originally proposed in [1]) generates two $\star$-commuting subgroups.

Algorithm 1.3. (TTP algorithm)

(*1*) *Choose two secret subsets* $BL = \{\sigma_{l_1}, \ldots, \sigma_{l_\alpha}\}$, $BR = \{\sigma_{r_1}, \ldots, \sigma_{r_\beta}\}$ *of the set of generators of* B_n, *where* $|l_i - r_j| \geq 2$ *for all* $1 \leq i \leq l_\alpha$ *and* $1 \leq j \leq r_\beta$.

(*2*) *Choose a secret element* $z \in B_n$.

(*3*) *Choose words* $\{w_1, \ldots, w_\gamma\}$ *of bounded length over the generators* BL.

(*4*) *Choose words* $\{v_1, \ldots, v_\gamma\}$ *of bounded length over the generators* BR.

(*5*) *For each* $i = 1, \ldots, \gamma$:

   (*a*) *calculate the left normal form of* zw_iz^{-1} *and reduce the result modulo* Δ^2;

   (*b*) *put* w_i' *to be a braid word corresponding to the element calculated in* (*a*);

   (*c*) *calculate the left normal form of* zv_iz^{-1} *and reduce the result modulo* Δ^2;

   (*d*) *put* v_i' *to be a braid word corresponding to the element calculated in* (*c*).

(*6*) *Publish the sets* $\{v_1', \ldots, v_\gamma'\}$ *and* $\{w_1', \ldots, w_\gamma'\}$.

We want to point out that the TTP algorithm produces generators of two commuting subgroups in B_n. Alice and Bob need to compute their images in $GL(n, \mathbb{F}_p(\mathbf{t}))$ to obtain $\star$-commuting subgroups.

1.5. Security assumptions

It was noticed in [1] that if the conjugator z generated randomly by the TTP algorithm is known, then there exists an efficient linear attack on the scheme which is able to recover the shared key of the parties. The problem of recovering the exact z seems like a very difficult mathematical problem because it reduces to solving the system of equations

$$\begin{cases} w_1' = \Delta^{2p_1} z w_1 z^{-1} \\ \dots \\ w_\gamma' = \Delta^{2p_\gamma} z w_\gamma z^{-1} \\ v_1' = \Delta^{2r_1} z v_1 z^{-1} \\ \dots \\ v_\gamma' = \Delta^{2r_\gamma} z v_\gamma z^{-1} \end{cases} \tag{1}$$

which has too many unknowns, since only left hand sides (i.e., elements $w_1', \dots, w_\gamma'$, $v_1', \dots, v_\gamma'$) are known. Hence, it might be difficult to find the original z.

Now observe that the AAGL key exchange protocol uses only the output of the TTP algorithm, namely the tuples $\{v_1', \dots, v_\gamma'\}$ and $\{w_1', \dots, w_\gamma'\}$ since all internal values in the TTP algorithm are not available to the parties. In other words it is irrelevant for the protocol how two particular commuting generating sets were constructed. This observation leads us to the following problem

> For tuples $\{v_1', \dots, v_\gamma'\}$ and $\{w_1', \dots, w_\gamma'\}$ find any z' and any numbers $p_1, \dots, p_\gamma, r_1, \dots, r_\gamma \in \mathbb{Z}$ such that the words $\{\Delta^{2p_1} z'^{-1} v_1' z', \dots, \Delta^{2p_\gamma} z'^{-1} v_\gamma' z'\}$ and $\{\Delta^{2r_1} z'^{-1} w_1' z', \dots, \Delta^{2r_\gamma} z'^{-1} w_\gamma' z'\}$ can be expressed as words over two disjoint commuting subsets of generators of B_n.

This is a new problem for computational group theory. Let us refer to it as *simultaneous conjugacy separation search problem* (abbreviated SCSSP). We want to emphasize that SCSSP has little in common with the *simultaneous conjugacy search problem* often referenced in the papers on the braid group cryptanalysis. The main difference is that in the conjugacy search problem both conjugate elements are available and the goal is to recover the secret conjugator. In case of SCSSP, only the left hand side of the equation is known. It is not clear if one of the problems can be reduced to the other.

It follows from the observation above that *any solution z' to a problem stated above plays a role of a conjugator z and can be used in a linear attack outlined in [1].* The main goal of this paper is to present an algorithm which for proposed parameter values solves the SCSSP. Experimental results convince us that our attack is a serious threat to the AAGL as the success rate is 100%. Furthermore, a slight modification of the algorithm produces the exact z generated by TTP in 40% of randomly generated instances.

While this paper was undergoing the reviewing process, it was brought to our attention by B. Tsaban (private communication) that the security of this protocol can be studied using a different approach [6].

1.6. Proposed parameter values

To provide 80 bits of security against the exhaustive search for z for the scheme the authors propose two slightly different sets of parameters:

- **Parameter set # 1.**
  - Let $n = 14$, $p = 13$, and $r = 3$.
  - Choose the conjugator z randomly of length 17.
  - Choose the words w_i and v_j randomly of length approximately 10.
  - The number γ of the words w_i and v_j is 27.
- **Parameter set # 2.**
  - Let $n = 12$, $p = 13$, and $r = 3$.
  - Choose the conjugator z randomly of length 18.
  - Choose the words w_i and v_j randomly of length approximately 10.
  - The number γ of the words w_i and v_j is 27.

2. TTP attack

In this section we describe a heuristic attack which finds a solution to a given instance of the SCSSP. The main ingredient in our attack is a length function on the group B_n. As it is explained in [9] there are no known efficiently computable and “sharp" length functions for braid groups. Therefore, for our attack we adopt the method of approximation of the geodesic length function originally proposed in [7]. In all our algorithms by $|\cdot|$ we denote approximation of the geodesic length function.

We present results of experiments which show that a fast heuristic procedure based on the length-based reduction is extremely successful for the suggested parameters. In fact, *every* instance of TTP algorithm generated in our experiments has been broken.

2.1. Generation

The original paper [1] lacks any details on how to randomly generate the secret element z and the words $\{w_1, \ldots, w_\gamma\}$, $\{v_1, \ldots, v_\gamma\}$ in TTP algorithm. Hence, in all our experiments:

- The word z is taken uniformly randomly as a word of a particular length from the ambient free group $F(\sigma_1, \ldots, \sigma_{n-1})$.
- The words $w_1, \ldots, w_\gamma$ and $v_1, \ldots, v_\gamma$ are taken uniformly randomly as words of particular lengths from the ambient free groups $F(BL)$ and $F(BR)$.

Also, the authors suggest to take the sets BL and BR randomly on step (*1*) of TTP algorithm. Observe that in general this might result in a choice of BL such that for some $1 \leq i < j < k \leq n-1$

$$\sigma_i, \sigma_k \in BL, \quad \text{but } \sigma_j \in BR.$$

We think that this situation is not desirable as it excludes the use of at least two braid generators in the words w_i and v_j. We think that the choice of the following sets

$$BL = \{\sigma_1, \ldots, \sigma_l\} \quad \text{and} \quad BR = \{\sigma_{l+2}, \ldots, \sigma_{n-1}\}$$

(where n is an even number and $l = (n-2)/2$) is optimal as it excludes only σ_{l+1} which maximizes the size of a space for the words $w_1, \ldots, w_\gamma$ and $v_1, \ldots, v_\gamma$.

2.2. Recovering Δ-powers

The first stage in our attack is recovering Δ powers in the system (1), i.e., computing numbers $p_1, \ldots, p_\gamma$ and $r_1, \ldots, r_\gamma$. The main tool in our computations below is the triangular inequality for the Cayley graph of the braid group B_n. Observe that the following inequalities hold.

(Parameter set #1) For each $i = 1, \ldots, \gamma$

$$|z^{-1}u_i z| \le 2|z| + |u_i| = 44 \quad \text{and} \quad |z^{-1}w_j z| \le 2|z| + |w_j| = 44$$

and

$$|\Delta^{2p}| = pn(n-1) = 182p.$$

Hence, $|\Delta^{2p} z^{-1}u_i z|, |\Delta^{2p} z^{-1}w_j z| \in [182p - 44, 182p + 44]$ and

$$|\Delta^{2p} z^{-1}u_i z| - |\Delta^{2(p-1)} z^{-1}u_i z| \ge 182 - 2 \cdot 44 = 94,$$
$$|\Delta^{2p} z^{-1}w_j z| - |\Delta^{2(p-1)} z^{-1}w_j z| \ge 182 - 2 \cdot 44 = 94.$$

(Parameter set #2) For each $i = 1, \ldots, \gamma$

$$|z^{-1}u_i z| \le 2|z| + |u_i| = 46 \quad \text{and} \quad |z^{-1}w_j z| \le 2|z| + |w_j| = 46$$

and

$$|\Delta^{2p}| = pn(n-1) = 132p.$$

Hence $|\Delta^{2p} z^{-1}u_i z|, |\Delta^{2p} z^{-1}w_j z| \in [132p - 46, 132p + 46]$ and

$$|\Delta^{2p} z^{-1}u_i z| - |\Delta^{2(p-1)} z^{-1}u_i z| \ge 132 - 2 \cdot 46 = 40,$$
$$|\Delta^{2p} z^{-1}w_j z| - |\Delta^{2(p-1)} z^{-1}w_j z| \ge 132 - 2 \cdot 46 = 40.$$

This observation implies that for both parameter sets the sequences $\{|\Delta^{2p} z^{-1}u_i z|\}_{p=0}^{\infty}$ and $\{|\Delta^{2p} z^{-1}w_j z|\}_{p=0}^{\infty}$ are strictly increasing. Thus, to recover the original power of Δ one can repeatedly multiply u_i' (and w_j') on the left by Δ^2 until the length cannot be reduced anymore (see Algorithm 2.1). Therefore, the task of recovering of Δ-powers reduces to computation of the length function, which, according to [10], might be hard. Nevertheless, we showed above that for both parameter sets the values $|\Delta^{2p} z^{-1}w_j z| - |\Delta^{2(p-1)} z^{-1}w_j z|$ and $|\Delta^{2p} z^{-1}u_i z| - |\Delta^{2(p-1)} z^{-1}u_i z|$ are at least 40 and hence even crude approximation of the length function can detect such a change of the length.

In this paper we use approximation of the length function proposed in [7] which employs Dehornoy handle free form of braid words (see [3]). The approximation algorithm in [7] for a braid word w finds a braid word w' representing the same element of the braid group as w does, with $|w'| \le |w|$. The obtained braid word w' in general is not geodesic, but numerous experiments and successful applications of that technique in [7], [8], [9] prove that w' is sufficiently close to being a geodesic. There is no known polynomial upper bound on the complexity of the approximation algorithm as there is no known polynomial upper bound on the complexity of the Dehornoy algorithm, but series of computations suggest that it has linear time complexity in terms of the length of the input word w.

Algorithm 2.1 (Δ-power recovery).
Input: *An element $w \in B_n$.*
Output: *An element u minimal in the left coset $\langle \Delta^{-2} \rangle w$.*
Computations:

A. *Set $u = w$.*
B. *If $|u| > |\Delta^{-2}u|$ then set $u = \Delta^{-2}u$ and goto B.*
C. *If $|u| > |\Delta^{2}u|$ then set $u = \Delta^{2}u$ and goto B.*
D. *Otherwise output u.*

Clearly Algorithm 2.1 always terminates. Moreover, under the assumption that the approximation algorithm has linear time complexity, it is easy to see that the power-recovery algorithm can be executed in at most $O((|w| + n^2)|w|/n^2) = O(|w|^2/n^2 + |w|)$ steps as the algorithm performs up to $|w|/n^2$ iterations and on each iteration for a word u of length at most $|w|$ the length of a word $\Delta^2 u$ is estimated.

2.3. Recovering conjugator

The second part of the attack computes a secret conjugator. At this point we assume that all Δ-powers from the system (1) are successfully found and we have a system of equations of the form

$$\left\{\begin{array}{l} w_1'' = z w_1 z^{-1} \\ \dots \\ w_\gamma'' = z w_\gamma z^{-1} \\ v_1'' = z v_1 z^{-1} \\ \dots \\ v_\gamma'' = z v_\gamma z^{-1} \end{array}\right. \quad \text{or} \quad \left\{\begin{array}{l} z^{-1} w_1'' z = w_1 \\ \dots \\ z^{-1} w_\gamma'' z = w_\gamma \\ z^{-1} v_1'' z = v_1 \\ \dots \\ z^{-1} v_\gamma'' z = v_\gamma \end{array}\right. \tag{2}$$

where only elements $u_i'' = \Delta^{-2p_i} u_i'$ and $w_j'' = \Delta^{-2r_i} w_j'$ are known. Let us call two sets of braids *separated* if they can be expressed as words over disjoint commuting sets of generators of B_n. As mentioned in Section 1.5, to break the protocol it is sufficient to find any conjugator z' which conjugates two tuples of elements $(u_1'', \dots, u_\gamma'')$ and $(w_1'', \dots, w_\gamma'')$ into two separated tuples of elements $(u_1, \dots, u_\gamma)$ and $(w_1, \dots, w_\gamma)$. This is the main goal of our attack.

Let $\bar{u} = (u_1, \dots, u_m)$ be a tuple of elements in B_n and x an element of B_n. Denote by $|\bar{u}|$ the total length of elements in $\bar{u}$, i.e., put

$$|\bar{u}| = \sum_{i=1}^{m} |u_i|.$$

Denote by $\bar{u}^x$ the tuple obtained from $\bar{u}$ by conjugation of each its element by x. It is intuitively clear that conjugation of a tuple of braids by a random element x almost always increases the length of the tuple. In other words, for a random element x the inequality

$$|\bar{u}^x| > |\bar{u}| \tag{3}$$

is *almost always* true. We do not have a proof of this fact, but numerous experiments convince us that this is true. Moreover, conjugation by longer elements almost always results in longer tuples.

The idea that conjugation consequently increases the length of tuples is not new. It was used in papers [5], [4] for different length functions with different success rates. But the most successful is a recent attack [9] which uses approximation of the geodesic length. In this paper we use the idea of separating two tuples of braids. To find z' we repeatedly conjugate the tuple $(u_1'', \ldots, u_\gamma'', w_1'', \ldots, w_\gamma'')$ by generators of B_n and their inverses and if for some generator $\sigma_k^{\pm 1}$ the decrease of the total length of the tuple is observed, then it is reasonable to guess that $\sigma_k^{\pm 1}$ is involved in z'.

Algorithm 2.2 (Recovering conjugator - I).
Input: *Tuples $\bar{a} = \{a_1, \ldots, a_\gamma\}$ and $\bar{b} = \{b_1, \ldots, b_\gamma\}$.*
Output: *An element z' separating tuples $\bar{a}$ and $\bar{b}$.*
Initialization: *Set $z' = 1$.*
Computations:

A. *For each $i = 1, \ldots, n-1$ and $\varepsilon = \pm 1$ conjugate tuples $\bar{a}$ and $\bar{b}$ by a generator σ_i^ε and compute*
$$\delta_{i,\varepsilon} = |\bar{a}^{\sigma_i^\varepsilon}| + |\bar{b}^{\sigma_i^\varepsilon}| - (|\bar{a}| + |\bar{b}|).$$

B. *If for some σ_i^ε the sets $\bar{a}^{\sigma_i^\varepsilon}$ and $\bar{b}^{\sigma_i^\varepsilon}$ are separated, then output $z' = \sigma_i^\varepsilon z'$.*

C. *Otherwise, if all $\delta_{i,\varepsilon}$ are positive (i.e., conjugation by σ_i^ε cannot further decrease the total length), then output FAILURE.*

D. *Otherwise, choose i and ε for which $\delta_{i,\varepsilon}$ is minimal. Set $z' = \sigma_i^\varepsilon z'$, $\bar{a} = \bar{a}^{\sigma_i^\varepsilon}$, and $\bar{b} = \bar{b}^{\sigma_i^\varepsilon}$. Goto step A.*

The described attack is similar to the one described in [9]. Recall that the main problem in [9] was the existence of so-called *peaks* (see [9, Definition 2.5]). This phenomenon is a consequence of difficult structure of finitely generated subgroups of braid groups. In this paper, we do not have this problem as z is chosen in the whole group B_n.

Note that Algorithm 2.2 is a greedy descend procedure. It may fail due to the fact that there exists a small fraction of words for which the inequality (3) does not hold. It is also prone to the length approximation errors. One can significantly reduce the failure rate of a descent procedure by introducing a backtracking algorithm which allows exploration of more than one search path. Algorithm 2.3 gives an implementation of the attack with backtracking.

Algorithm 2.3 (Recovering conjugator with Backtracking).
Input: *Tuples* $\bar{a} = \{a_1, \ldots, a_\gamma\}$ *and* $\bar{b} = \{b_1, \ldots, b_\gamma\}$.
Output: *An element* z' *separating tuples* $\bar{a}$ *and* $\bar{b}$.
Initialization: *Set* $S = \{(\bar{a}, \bar{b}, 1)\}$.
Computations:

A. *If* $S = \emptyset$ *then output FAILURE.*
B. *Choose* $(\bar{x}, \bar{y}, c) \in S$ *such that* $|\bar{x}| + |\bar{y}|$ *is the minimal.*
C. *For each* $i = 1, \ldots, n-1$ *and* $\varepsilon = \pm 1$ *conjugate tuples* $\bar{x}$ *and* $\bar{y}$ *by a generator* σ_i^ε *and compute*
$$\delta_{i,\varepsilon} = |\bar{x}^{\sigma_i^\varepsilon}| + |\bar{y}^{\sigma_i^\varepsilon}| - (|\bar{x}| + |\bar{y}|).$$
D. *If for some* σ_i^ε *the sets* $\bar{x}^{\sigma_i^\varepsilon}$ *and* $\bar{y}^{\sigma_i^\varepsilon}$ *are separated then output* $z' = \sigma_i^\varepsilon c$.
E. *Otherwise, for each* $i = 1, \ldots, n-1$ *and* $\varepsilon = \pm 1$ *add the tuple* $(\bar{x}^{\sigma_i^\varepsilon}, \bar{x}^{\sigma_i^\varepsilon}, \sigma_i^\varepsilon c)$ *to the set* S. *Goto step A.*

We must mention here that, although there is a possibility that Algorithm 2.3 outputs FAILURE or does not terminate on some inputs, this situation has never occurred in our experiments.

Finally, we present another modification of Algorithm 2.2.

Algorithm 2.4 (Recovering conjugator - II).
Input: *Tuples* $\bar{a} = \{a_1, \ldots, a_\gamma\}$ *and* $\bar{b} = \{b_1, \ldots, b_\gamma\}$.
Output: *An element* z' *separating tuples* $\bar{a}$ *and* $\bar{b}$.
Initialization: *Set* $z' = 1$.
Computations:

A. *For each* $i = 1, \ldots, n-1$ *and* $\varepsilon = \pm 1$ *conjugate tuples* $\bar{a}$ *and* $\bar{b}$ *by a generator* σ_i^ε *and compute*
$$\delta_{i,\varepsilon} = |\bar{a}^{\sigma_i^\varepsilon}| + |\bar{b}^{\sigma_i^\varepsilon}| - (|\bar{a}| + |\bar{b}|).$$
B. *If all* $\delta_{i,\varepsilon}$ *are positive (i.e., conjugation by* σ_i^ε *cannot further decrease the total length) and the sets* $\bar{a}$ *and* $\bar{b}$ *are separated then output* z'.
C. *If all* $\delta_{i,\varepsilon}$ *are positive (i.e., conjugation by* σ_i^ε *cannot further decrease the total length), but the sets* $\bar{a}$ *and* $\bar{b}$ *are not separated then output FAILURE.*
D. *Otherwise, choose* i *and* ε *for which* $\delta_{i,\varepsilon}$ *is minimal. Set* $z' = \sigma_i^\varepsilon z'$, $\bar{a} = \bar{a}^{\sigma_i^\varepsilon}$, *and* $\bar{b} = \bar{b}^{\sigma_i^\varepsilon}$. *Goto step A.*

Algorithms 2.2 and 2.4 are almost the same except that they have different termination conditions. Algorithm 2.2 stops as soon as the tuples are separated, while Algorithm 2.4 tries to minimize the total length of the tuple, and when the minimal value is reached, it checks if the current tuples are separated.

The complexity of step A in Algorithms 2.2 and 2.4 is $O(\gamma n(|\overline{a}_i| + |\overline{b}_i|))$. The maximal number of iterations can be bounded by the total length of the input $|\overline{a}_i| + |\overline{b}_i|$. A very rough upper bound on the complexity of the two algorithms is $O(\gamma n(|\overline{a}_i| + |\overline{b}_i|)^2)$.

The complexity of Algorithm 2.3 is harder to estimate. Potentially, the backtracking mechanism may cause the algorithm to explore exponentially many potential solutions. However, our experiments show that a very few backtracking steps are required to find a solution.

2.4. Results of experiments

The attack was implemented using routines of "CRyptography And Groups" package [2]. It was tested on different sets of instances of the protocol. In particular, we generated the sets BL and BR randomly and used fixed sets $BL = \{\sigma_1, \ldots, \sigma_l\}$ and $BR = \{\sigma_{l+2}, \ldots, \sigma_{n-1}\}$. We used the proposed values of the parameters (see Section 1.6). In addition the attack was tested on instances generated with the increased length of the secret conjugator z.

In all the experiments Algorithm 2.3 had 100% success (1000 successful recovery out of total 1000 experiments) of producing a separating conjugator z'. The average time of a run of the algorithm was 4.5 seconds when executed on a Dual Core Opteron 2.2 GHz machine with 4GB of ram. The algorithm without backtracking had slightly smaller but still respectable success rate of 90%. It is very interesting to notice that Algorithm 2.4 actually recovered the original secret conjugator z in about 40% of the cases. This is the reason why we mention this algorithm in our paper.

Experiments with instances of the TTP protocol generated using $|z| = 50$ (which is almost three times greater than the suggested value) again showed 100% success rate. However, we need to point out that the attack may fail when the length of z is large relative to the length of Δ^2. For instance, when in the second parameter set the length of z is increased to 100, the algorithm recovering Δ-powers sometimes output wrong values. Nevertheless, the success rate of Algorithm 2.3 is still about 90% in this case. We think it is possible to modify our algorithms to work with increased parameter values. But the biggest concern here is that the protocol with increased parameter values may not be suitable for purposes of lightweight cryptography.

References

[1] I. Anshel, M. Anshel, D. Goldfeld, S. Lemieux: Key agreement, the algebraic eraser^{TM}, and lightweight cryptography, in: Algebraic Methods in Cryptography, L. Gerritzen et al. (ed.), Contemp. Math. 418, AMS, Providence (2006) 1–34.

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

[3] P. Dehornoy: A fast method for comparing braids, Adv. Math. 125 (1997) 200–235.

[4] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, U. Vishne: Length-based conjugacy search in the braid group, in: Algebraic Methods in Cryptography, L. Gerritzen et al. (ed.), Contemp. Math. 418, AMS, Providence (2006) 1–34.

[5] J. Hughes, A. Tannenbaum: Length-based attacks for certain group based encryption rewriting systems, preprint.

[6] A. G. Kalka, M. Teicher, B. Tsaban: Cryptanalysis of the algebraic eraser, in preparation.

[7] A. G. Miasnikov, V. Shpilrain, A. Ushakov: A practical attack on some braid group based cryptographic protocols, in: Advances in Cryptology (CRYPTO, Santa Barbara, 2005), V. Shoup (ed.), Lect. Notes Comput. Sci. 3621, Springer, Berlin (2005) 86–96.

[8] A. G. Miasnikov, V. Shpilrain, A. Ushakov: Random subgroups of braid groups: an approach to cryptanalysis of a braid group based cryptographic protocol in: Public Key Cryptography (PKC, New York, 2006), M. Yung, et al. (ed.), Lect. Notes Comput. Sci. 3958, Springer, Berlin (2006) 302–314.

[9] A. D. Myasnikov, A. Ushakov: Length based attack and braid groups: cryptanalysis of Ahsel-Anshel-Goldfeld key exchange protocol, in: Public Key Cryptography (PKC, Beijing, 2007), T. Okamoto et al. (ed.), Lect. Notes Comput. Sci. 4450, Springer, Berlin (2007) 76–88.

[10] M. Paterson, A. Razborov: The set of minimal braids is co-NP-complete, J. Algorithms 12 (1991) 393–408.