Cryptanalysis of Group-based Key Agreement Protocols Using Subgroup Distance Functions

Dima Ruinskiy, Adi Shamir, and Boaz Tsaban

The Weizmann Institute of Science, Rehovot, Israel

Abstract. We introduce a new approach for cryptanalysis of key agreement protocols based on noncommutative groups. Our approach uses functions that estimate the distance of a group element to a given subgroup. We test it against the Shpilrain-Ushakov protocol, which is based on Thompson's group F, and show that it can break about half the keys within a few seconds on a single PC.

Key words: Key agreement, Cryptanalysis, Thompson's group, Shpilrain-Ushakov, Subgroup distance function

1 Introduction

Key agreement protocols have been the subject of extensive studies in the past 30 years. Their main task is to allow two parties (in the sequel, Alice and Bob) to agree on a common secret key over an insecure communication channel. The best known example of such a protocol is the Diffie-Hellman protocol, which uses a (commutative) cyclic group. Over the last few years, there was a lot of interest in key agreement protocols based on noncommutative groups, and much research was dedicated to analyzing these proposals and suggesting alternative ones (see, e.g., [1, 4–8, 10–12], and references therein).

A possible approach for attacking such systems is the *length-based cryptanalysis*, which was outlined in [6]. This approach relies on the existence of a good length function on the underlying group, i.e., a function $\ell(g)$ that tends to grow as the number of generators multiplied to obtain g grows. Examples of groups known to have such length functions are the *braid group* B_N [2] and *Thompson's group* F [3]. For these groups, several practical realizations of length-based attacks were demonstrated [4, 5, 9]. These attacks can achieve good success rates, but usually only when we allow the algorithm to explore many suboptimal partial solutions, which greatly increases both the time and space complexities (see [5] for more details).

We introduce a novel approach to cryptanalysis of such key agreement protocols, which relies on the notion of *subgroup distance functions*, i.e., functions that estimate, for an element $g \in G$ and a subgroup $H \leq G$, the distance from g to H. The motivation for these distance-based attacks is the fact that several families of public key agreement protocols suggest predefined pairs of subgroups of the main group to be used for key generation, and their security depends on the ability of the adversary to generate any elements in these subgroups, which are in some way equivalent to the originals (see [9, 11]). We construct the theoretical framework for *distance-based attacks* and demonstrate its applicability using the Shpilrain-Ushakov protocol in Thompson's group F [12] as an example. Although it has recently been shown by Matucci [8] that the implementation of the proposed protocol in F can be broken deterministically using a specialized attack based on the structural properties of the group, it is still an interesting test case for more generic attacks, such as the one proposed here.

The paper is organized as follows: in Section 2 we present the protocol in its general form. We then introduce in Section 3 the notion of subgroup distance function and a general attack scheme based on it. Section 4 describes the setting for the protocol in Thompson's group F. In Section 5 we introduce several subgroup distance functions in F. Section 6 describes our experimental cryptanalytic results.

2 The Shpilrain-Ushakov Key agreement Protocol

The protocol below was suggested by Shpilrain and Ushakov in [12]. The authors suggested to use Thompson's group F for its implementation. Before we focus on that example, we'll discuss the general case.

(0) Alice and Bob agree (publicly) on a group G and subgroups $A, B \leq G$, such that $ab = ba$ for each $a \in A$ and each $b \in B$.
1. A public word $z \in G$ is selected.
2. Alice selects privately at random elements $a_1 \in A$ and $b_1 \in B$, computes $u_1 = a_1 z b_1$, and sends u_1 to Bob.
3. Bob selects privately at random elements $a_2 \in A$ and $b_2 \in B$, computes $u_2 = b_2 z a_2$, and sends u_2 to Alice.
4. Alice computes $K_A = a_1 u_2 b_1 = a_1 b_2 z a_2 b_1$, whereas Bob computes $K_B = b_2 u_1 a_2 = b_2 a_1 z b_1 a_2$.

As $a_1 b_2 = b_2 a_1$ and $a_2 b_1 = b_1 a_2$, $K_A = K_B = K$ and so the parties share the same group element, from which a secret key can be derived.

2.1 Breaking the protocol

The goal of the adversary is to obtain the secret group element K from the publicly known elements u_1, u_2 and z. For this it suffices to solve the following problem:

Definition 1 (Decomposition problem) *Given $z \in G$ and $u = azb$ where $a \in A$ and $b \in B$, find some elements $\tilde{a} \in A$ and $\tilde{b} \in B$, such that $\tilde{a}z\tilde{b} = azb$.*

Indeed, assume that the attacker, given $u_1 = a_1 z b_1$, finds $\tilde{a}_1 \in A$ and $\tilde{b}_1 \in B$, such that $\tilde{a}_1 z \tilde{b}_1 = a_1 z b_1$. Then, because $u_2 = b_2 z a_2$ is known, the attacker can compute

$$\tilde{a}_1 u_2 \tilde{b}_1 = \tilde{a}_1 b_2 z a_2 \tilde{b}_1 = b_2 \tilde{a}_1 z \tilde{b}_1 a_2 = b_2 u_1 a_2 = K_B \ .$$

Alternatively, the attacker can break the protocol by finding a valid decomposition of $u_2 = b_2 z a_2$.

For any given $\tilde{a} \in A$ we can compute its *complement* $\tilde{b} = z^{-1}\tilde{a}^{-1}u = z^{-1}\tilde{a}^{-1}(azb)$, which guarantees that $\tilde{a}z\tilde{b} = azb$. The pair $\tilde{a}, \tilde{b}$ is a solution to this problem if, and only if, $\tilde{b} \in B$. A similar comment applies if we start with $\tilde{b} \in B$. This involves being able to solve the *group membership problem*, i.e., to determine whether $\tilde{b} \in B$ (or $\tilde{a} \in A$ in the second case).

It should be stressed that solving the decomposition problem is sufficient, but not necessary in order to cryptanalyze the system. All that is required in practice is finding some pair $\tilde{a}, \tilde{b}$ that succeeds in decrypting the information passed between Alice and Bob. Any pair $\tilde{a} \in A$ and $\tilde{b} \in B$ will work, but there can be other pairs, which are just as good. This observation can be useful in cases where the group membership problem is difficult or in groups where the centralizers of individual elements are considerably larger than the centralizers of the subgroups (which is not the case in F, see [9]). For simplicity, in the sequel we will restrict ourselves to solutions where $\tilde{a} \in A$ and $\tilde{b} \in B$.

3 Subgroup distance functions

Definition 2 (Subgroup distance function) *Let G be a group, $H \leq G$ a subgroup. A function $d_H : G \to \mathbb{R}^+$ is a* subgroup distance function *if it satisfies the following two axioms:*

1. *Validity: $d_H(h) = 0$ for all $h \in H$.*
2. *Non-triviality: $d_H(g) > 0$ for all $g \notin H$.*

It is an invariant *subgroup distance function if it also satisfies:*

(3) Invariance: $d_H(gh) = d_H(hg) = d_H(g)$ *for all* $g \in G$ *and* $h \in H$.

Clearly, if it is possible to evaluate a subgroup distance function d_H on all elements of G, then the membership decision problem for H is solvable: $g \in H \iff d_H(g) = 0$. Conversely, if one can solve the membership decision problem, a trivial distance function can be derived from it, e.g., $d_H(g) = 1 - \chi_H(g)$, where χ_H is the characteristic function of H.

Obviously, this trivial distance function is not a good example. For the subgroup distance function to be useful, it has to somehow measure how close a given element g is to H, that is, if $d_H(g_1) < d_H(g_2)$, then g_1 is closer to H than g_2. This concept of "closeness" can be hard to define, and even harder to evaluate. The notion of what's considered a good distance function may vary, depending on the subgroups and on the presentation. In the sequel we will discuss concrete examples of subgroup distance function in Thompson's group F.

Assuming the existence of such functions, consider the following algorithm for solving the decomposition problem:

Algorithm 1 (Subgroup distance attack)

We are given words $z, xzy \in G$, *where* $x \in X$ *and* $y \in Y$, X, Y *are commuting subgroups of* G *and* S_X, S_Y *are their respective (finite) generating sets. The goal it to find some* $\tilde{x} \in X$ *and* $\tilde{y} \in Y$, *such that* $xzy = \tilde{x}z\tilde{y}$. *The algorithm runs at most a predefined number of iterations* N.

1. *Let* $\tilde{x} \leftarrow 1$.
2. *For each* $g_i \in S_X^{\pm 1}$ *compute* $x_i = \tilde{x}g_i$, *its complement* $y_i = z^{-1}x_i^{-1}xzy$ *and evaluate* $d_Y(y_i)$. *If* $d_Y(y_i) = 0$, *let* $\tilde{x} = x_i$, $\tilde{y} = y_i$ *and halt.*
3. *Let* j *be the index of the minimum* $d_Y(y_i)$ *(if several such* j *are possible, choose one arbitrarily).*
4. *If the maximal number of iterations* N *has been reached, terminate. Otherwise, let* $\tilde{x} \leftarrow x_j$ *and return to step 2.*

Observe that if the algorithm halts in step 2, then the pair $\tilde{x}, \tilde{y}$ is a solution of the decomposition problem.

Algorithm 1 is very similar to the length-based attacks described in [4, 9]. The difference is that it uses the subgroup distance function, instead of the length function to evaluate the quality of candidates. As such, any extensions applicable to the length-based algorithms (such as memory, lookahead, etc.) can be used with the distance-based attack as well. Refer to [5, 9] for more information.

3.1 Attacking the Shpilrain-Ushakov protocol

The adversary is given the common word z and the public elements u_1, u_2. These can be translated into four equations in the group:

$$\begin{aligned} u_1 &= a_1 z b_1 \\ u_2 &= b_2 z a_2 \\ u_1^{-1} &= b_1^{-1} z^{-1} a_1^{-1} \\ u_2^{-1} &= a_2^{-1} z^{-1} b_2^{-1} \end{aligned} \tag{1}$$

Algorithm 1 (with or without possible extensions) can be applied to each of the four equations separately, thus attacking each of the four private elements $a_1, a_2, b_1^{-1}, b_2^{-1}$. A single success out of the four attempts is sufficient to break the cryptosystem (see Section 2.1).

4 Thompson's group

Thompson's group F is the infinite noncommutative group defined by the following generators and relations:

$$F = \langle\ x_0, x_1, x_2, \dots \mid x_i^{-1} x_k x_i = x_{k+1}\ (k > i)\ \rangle \tag{2}$$

Remark 1 *From Equation* (2) *it's evident that the elements* x_0, x_1 *and their inverses generate the entire group, because* $x_k^{\pm 1} = x_0^{1-k} x_1^{\pm 1} x_0^{k-1}$ *for every* $k \geq 2$.

Definition 3 *A basic generator* $x_i^{\pm 1}$ *of* F *is called a* letter. *A generator* x_i *is a* positive letter. *An inverse* x_i^{-1} *is a* negative letter. *A* word *in* F *is a sequence of letters. We define* $|w|$ *as the length of the word* w *, i.e., the number of letters in it.*

Definition 4 *A word* $w \in F$ *is said to be in* normal form, *if*

$$w = x_{i_1} \cdots x_{i_r} x_{j_t}^{-1} \cdots x_{j_1}^{-1} \tag{3}$$

and the following two conditions hold:

(NF1) $i_1 \leq \cdots \leq i_r$ *and* $j_1 \leq \cdots \leq j_t$
(NF2) *If both* x_i, x_i^{-1} *occur in* w*, then at least one of* x_{i+1}, x_{i+1}^{-1} *occurs too.*

A word is said to be in seminormal form *if only* ***(NF1)*** *holds.*

While a seminormal form is not necessarily unique, a normal form is, i.e., two words represent the same group element if and only if they have the same normal form [3]. The following rewriting rules can be used to convert any word to its seminormal form [12]:

For all non-negative integers $i < k$:

$$\begin{aligned}
&(R1)\ x_k x_i &&\to x_i x_{k+1}\\
&(R2)\ x_k^{-1} x_i &&\to x_i x_{k+1}^{-1}\\
&(R3)\ x_i^{-1} x_k &&\to x_{k+1} x_i^{-1}\\
&(R4)\ x_i^{-1} x_k^{-1} &&\to x_{k+1}^{-1} x_i^{-1}
\end{aligned}$$

For all non-negative integers i:

$$(R5)\ x_i^{-1} x_i \quad \to 1$$

The seminormal form can be subsequently converted to a normal form by searching for pairs of indices violating (NF2), starting from the boundary between the positive and negative parts, and applying the inverses of rewriting rules (R1) and (R4) to eliminate these pairs [12]:

Suppose that $(x_{i_a}, x_{j_b}^{-1})$ is a pair of letters violating (NF2) and that a and b are maximal with this property (i.e., there exists no violating pair $(x_{i_k}, x_{j_l}^{-1})$ with $k > a$ and $l > b$). Then $i_a = j_b$ and all indices in $x_{i_{a+1}} \cdots x_{i_r} x_{j_t}^{-1} \cdots x_{j_{b+1}}^{-1}$ are higher than $i_a + 1$ (by definition of (NF2)). Applying the inverse of (R1) to x_{i_a} and the inverse of (R4) to $x_{j_b}^{-1}$ we get:

$$\begin{aligned}
w &= x_{i_1} \cdots x_{i_a} \underbrace{\left(x_{i_{a+1}} \cdots x_{i_r} x_{j_t}^{-1} \cdots x_{j_{b+1}}^{-1}\right)}_{c} x_{j_b}^{-1} \cdots x_{j_1}\\
&\to x_{i_1} \cdots x_{i_{a+1}-1} \cdots x_{i_r-1} \underbrace{\left(x_{i_a} x_{j_b}^{-1}\right)}_{cancel} x_{j_t-1}^{-1} \cdots x_{j_{b+1}-1}^{-1} \cdots x_{j_1}\\
&\to x_{i_1} \cdots x_{i_{a-1}} \underbrace{\left(x_{i_{a+1}-1} \cdots x_{i_r-1} x_{j_t-1}^{-1} \cdots x_{j_{b+1}-1}^{-1}\right)}_{c'} x_{j_{b-1}} \cdots x_{j_1}
\end{aligned}$$

The violating pair $(x_{i_a}, x_{j_b}^{-1})$ is cancelled and the subword c' obtained from c by index shifting contains no violating pairs (by the assumption of maximality on (a, b)). Thus, we can continue searching for bad pairs, starting from $a - 1$ and $b - 1$ down. Thus we are guaranteed to find and remove all the violating pairs and reach the normal form.

Definition 5 (Normal form length) *For $w \in F$, whose normal form is $\hat{w}$, define the* normal form length *as $\ell_{NF}(w) = |\hat{w}|$.*

The following lemma shows the effect multiplication by a single letter has on the normal form of the word. This result will be useful in the following sections.

Lemma 1 *Let $w \in F$ and $x = x_t^{\pm 1}$ be a basic generator of F in the presentation (2). Then $\ell_{NF}(xw) = \ell_{NF}(w) \pm 1$ (and due to symmetry, $\ell_{NF}(wx) = \ell_{NF}(w) \pm 1$).*

Proof. We'll concentrate on the product xw (obviously, the case of wx is similar) and observe what happens to the normal form of w when it's multiplied on the left by the letter x. Without loss of generality, $w = x_{i_1} \cdots x_{i_k} x_{j_l}^{-1} \cdots x_{j_1}^{-1}$ is in normal form. Denote the positive and negative parts of w by w_p and w_n respectively.

Assume that $x = x_t$ is a positive letter. Then bw is converted to a seminormal form by moving x into its proper location, while updating its index, using repeated applications of (R1). Assuming m applications of (R1) are necessary, the result is of the form:

$$\overline{bw} = x_{i_1} \cdots x_{i_m} \boldsymbol{x_{t+m}} x_{i_{m+1}} \cdots x_{i_k} x_{j_l}^{-1} \cdots x_{j_1}^{-1} \ ,$$

where $i_m < t + m - 1$ and $i_{m+1} \geq t + m$.

Remark 2 *Observe that it is not possible that $i_m = t+m-1$, because in order to apply (R1): $x_{t+m-1} x_{i_m} \to x_{i_m} x_{t+m}$, one must have $i_m < t+m-1$.*

Example 1 *$w = x_3 x_7 x_{11} x_9^{-1} x_4^{-1}$, $b = x_8$. $bw = x_8 \cdot x_3 x_7 x_{11} x_9^{-1} x_4^{-1}$ is converted to $\overline{bw} = x_3 x_7 \boldsymbol{x_{10}} x_{11} x_9^{-1} x_4^{-1}$, by 2 applications of (R1).*

*Obviously, $\overline{bw}$ is a seminormal form and $|\overline{bw}| = |w| + 1$. If $\overline{bw}$ is in normal form (as in the above example), we're done. The only situation where it's not in normal form, is if it contains pairs violating **(NF2)**. Since x_{t+m} is the only letter introduced, the only violating pair can be (x_{t+m}, x_{t+m}^{-1}). This may occur, if w contained x_{t+m}^{-1}, but neither x_{t+m}, nor $x_{t+m+1}^{\pm 1}$.*

Example 2 *$w = x_3 x_7 x_{11} x_9^{-1} x_4^{-1}$, $b = x_7$. $bw = x_7 \cdot x_3 x_7 x_{11} x_9^{-1} x_4^{-1}$ is converted to $\overline{bw} = x_3 x_7 \boldsymbol{x_9} x_{11} x_9^{-1} x_4^{-1}$. In this case (x_9, x_9^{-1}) violates **(NF2)**. The inverse of (R1) is applied to rewrite $x_9 x_{11} \to x_{10} x_9$, and $x_9 x_9^{-1}$ are canceled out, yielding the (normal) word $\widehat{bw} = x_3 x_7 x_{10} x_4^{-1}$.*

Whenever a situation occurs as described above, the pair (x_{t+m}, x_{t+m}^{-1}) is cancelled, according to the procedure described in Section 4. This causes all indices above $t + m$ to be decreased by 1. The resulting word is

$$\widehat{bw} = x_{i_1} \cdots x_{i_m} x_{i_{m+1}-1} \cdots x_{i_k-1} x_{j_l-1}^{-1} \cdots x_{j_{n+1}-1}^{-1} x_{j_n}^{-1} \cdots x_{j_1}^{-1} \ ,$$

where $i_m < t+m-1$, $i_{m+1} \geq t+m+2$, $j_n \leq t+m$ *and* $j_{n+1} \geq t+m+2$. *We have* $|\widehat{bw}| = |w| - 1$ *and, in fact,* $\widehat{bw}$ *is in normal form. Indeed, once the pair* (x_{t+m}, x_{t+m}^{-1}) *is cancelled, the only new pair violating* ***(NF2)*** *that can be introduced is* $(x_{t+m-1}, x_{t+m-1}^{-1})$, *but this is not possible, because* x_{t+m-1} *does not appear in* $\widehat{bw}$, *due to Remark 2. This completes the proof for positive letters.*

Now, consider the case where $x = x_t^{-1}$, *a negative letter.* bw *is converted to a seminormal form by moving* x_t^{-1} *to the right, while updating its index, using the different rewriting rules. There are two possible outcomes:*

(1) *After* m *applications of (R2) the resulting word is*

$$\overline{bw} = x_{i_1} \cdots x_{i_m} \boldsymbol{x_{t+m}^{-1}} x_{i_{m+1}} \cdots x_{i_k} x_{j_l}^{-1} \cdots x_{j_1}^{-1} \ ,$$

where $i_{m+1} = t + m$, *and so the pair is cancelled by applying (R5). Now, because* $i_m < t + m - 1$, *the elimination of the pair* (x_{t+m}, x_{t+m}^{-1}) *does not introduce pairs that violate* ***(NF2)***, *and so* $\overline{bw}$ *is in normal form and has* $|\overline{bw}| = |w| - 1$.

Example 3 $w = x_3 x_7 x_9^{-1} x_4^{-1}$, $b = x_6^{-1}$. $bw = \boldsymbol{x_6^{-1}} x_3 x_7 x_9^{-1} x_4^{-1}$ *is converted to* $x_3 \boldsymbol{x_7^{-1}} \boldsymbol{x_7} x_9^{-1}$ *and the pair of inverses is cancelled out to obtain* $x_4^{-1} \rightarrow x_3 x_9^{-1} x_4^{-1}$.

(2) x_t^{-1} *is moved to its proper place among the negative letters, updating its index if necessary. This is completed through* m *applications of (R2), followed by* $k-m$ *applications of (R3) and finally,* $l-n$ *applications of (R4), to obtain*

$$\overline{bw} = x_{i_1} \cdots x_{i_m} \boldsymbol{x_{t+m}^{-1}} x_{i_{m+1}+1} \cdots x_{i_k+1} x_{j_l+1}^{-1} \cdots x_{j_{n+1}+1}^{-1} \boldsymbol{x_{t+m}^{-1}} x_{j_n}^{-1} \cdots x_{j_1}^{-1} \ ,$$

where $i_m < t+m-1$, $i_{m+1} > t+m$, $j_{n+1} > t+m$ *and* $j_n \leq t+m$. *Because the letter* x_{t+m} *is not present in* $\overline{bw}$ *(otherwise the previously described situation would occur), the newly introduced letter* x_{t+m}^{-1} *cannot violate* ***(NF2)***, *and therefore* $\overline{bw}$ *is in fact in normal form and* $|\overline{bw}| = |w| + 1$.

Example 4 $w = x_3 x_7 x_9^{-1} x_4^{-1}$, $b = x_5^{-1}$. $bw = \boldsymbol{x_5^{-1}} x_3 x_7 x_9^{-1} x_4^{-1}$ *is rewritten as:* $\boldsymbol{x_6^{-1}} x_7 x_9^{-1} x_4^{-1} \rightarrow x_3 x_8 \boldsymbol{x_6^{-1}} x_9^{-1} x_4^{-1} \rightarrow x_3 x_8 x_{10}^{-1} \boldsymbol{x_6^{-1}} x_4^{-1}$.

This completes the proof for negative letters.

□

4.1 The Shpilrain-Ushakov protocol in Thompson's group

For a natural number $s \geq 2$ let $S_A = \{x_0 x_1^{-1}, \ldots, x_0 x_s^{-1}\}$, $S_B = \{x_{s+1}, \ldots, x_{2s}\}$ and $S_W = \{x_0, \ldots, x_{s+2}\}$. S_W generates F (see Remark 1). Denote by A_s and B_s the subgroups of F generated by S_A and S_B, respectively.

All of the following facts are shown in [12]: A_s is exactly the set of elements whose normal form is

$$x_{i_1} \cdots x_{i_m} x_{j_m}^{-1} \cdots x_{j_1}^{-1} \ ,$$

i.e, has positive and negative parts of the same length m, and additionally satisfies $i_k - k < s$ and $j_k - k < s$ for every $k = 1, \ldots, m$. B_s is the set of all elements of F whose normal form consists only of letters with indices $\geq s + 1$. Additionally, A_s and B_s commute elementwise, which makes them usable for implementing the protocol in Section 2.

Key generation Let $s \geq 2$ and L be positive integers. The words $a_1, a_2 \in A_s$, $b_1, b_2 \in B_s$, and $w \in F$ are all chosen of normal form length L, as follows: Let X be A, B, or W. Start with the empty word, and multiply it on the right by a generator (or inverse) selected uniformly at random from the set S_X. Continue this procedure until the normal form of the word has length L.

For practical and (hopefully) secure implementation of the protocol, it is suggested in [12] to use $s \in \{3, 4, \ldots, 8\}$ and $L \in \{256, 258, \ldots, 320\}$.

5 Subgroup distance functions in Thompson's group

In this section we'll suggest several natural distance functions from the subgroups $A_s, B_s \leq F$ defined in Section 4.1. These distance functions can be used to implement the attack outlined by Algorithm 1.

5.1 Distance functions from B_s

For $w \in F$ define $P_i(w)$ and $N_i(w)$ as the number of occurrences of x_i and x_i^{-1} in the normal form $\hat{w}$ of w.

Definition 6 (Distance from B_s) *Let $s \leq 2$ be an integer. For $w \in F$ the distance from B_s is defined as*

$$d_{B_s}(w) = \sum_{i=0}^{s} \left(P_i(w) + N_i(w)\right)$$

Claim 1 *d_{B_s} is a distance function.*

Proof. This is immediate, since an element is in B_s if and only if its normal form does not contain generators with indices below $s+1$ (see Section 4.1). □

Claim 2 *d_{B_s} is an invariant distance function.*

Proof. It is enough to consider only the generators of B_s. Indeed, if multiplication by a single generator of B_s does not change the distance of a word w, neither does multiplication by a sequence of these generators.

*Let $w \in F$. Let $b = x_{s+\alpha}^{\pm 1}$, where $\alpha > 0$. By Lemma 1, we know that b is either moved to its proper position (and $\ell_{NF}(bw) = \ell_{NF}(w) + 1$) or it is cancelled with its inverse, either by (R5) or as part of a pair violating **(NF2)**, in which case $\ell_{NF}(bw) = \ell_{NF}(w) - 1$. The index of b is initially above s, and may only increase when the rewriting rules are applied. Therefore, if b is cancelled at some point, the index of its inverse is also above s. Furthermore, when pairs of elements are rewritten, the lower-indexed element is not affected, so any letters with indices $\leq s$ will not be affected by moving b. Finally, if b is cancelled out due to violating **(NF2)**, the process again only affects letters with indices higher than b's (see the proof of Lemma 1). In all cases, the generators with indices $\leq s$ are not affected at all, and so $d_{B_s}(bw) = d_{B_s}(w)$.*

□

One can intuitively feel that d_{B_s} is a natural distance function, because it counts the number of "bad" letters in w (letters that do not belong to the subgroup B_s). Indeed, if w is in normal form, $w = w_p w_c w_n$, where w_p and w_n are the "bad" positive and negative subwords, respectively, then $d_{B_s}(w) = |w_p| + |w_n|$ and $w_p^{-1} w w_n^{-1} \in B$.

We now introduce another natural function that measures distance from B_s.

Definition 7 (Weighted distance from B_s) *Let $s \leq 2$ be an integer. For $w \in F$ the weighted distance from B_s is defined as*

$$\overline{d_{B_s}}(w) = \sum_{i=0}^{s} (s+1-i)\,(P_i(\hat{w}) + N_i(\hat{w}))$$

$\overline{d_{B_s}}$ does not only count the "bad" letters, but assigns a score for each letter, depending on how far below $s+1$ it is (in particular, $d_{B_s}(w) \leq \overline{d_{B_s}}(w)$ for all $w \in F$. The following claim is straightforward.

Claim 3 *$\overline{d_{B_s}}$ is an invariant distance function.*

Proof. The proof of Claim 2 shows that multiplication by b does not alter any letters below $s+1$ in w. Therefore, the weight of each such letter is also preserved. □

5.2 Distance functions from A_s

We will now describe a number of natural distance functions from the subgroup A_s. Recall (Section 4.1) that A_s is the set of all elements in F, whose normal form is of the type $x_{i_1} \cdots x_{i_m} x_{j_m}^{-1} \cdots x_{j_1}^{-1}$, i.e, has positive and negative parts of the same length m, and additionally satisfies $i_k - k < s$ and $j_k - k < s$ for every $k = 1, \ldots, m$.

Definition 8 (Distance from A_s) *Let $s \geq 2$ be an integer. Let $w \in F$, such that its normal form is $\hat{w} = x_{i_1} \cdots x_{i_p} x_{j_n}^{-1} \cdots x_{j_1}^{-1}$. The distance from A_s is defined as*

$$d_{A_s}(w) = |\{k : i_k - k \geq s\}| \ + \ |\{l : j_l - l \geq s\}| \ + \ |p - n|$$

$d_{A_s}(w)$ is the number of "bad" letters in $\hat{w}$, i.e., letters that violate the A_s property, plus the difference between the lengths of the positive or negative parts. d_{A_s} is clearly a distance function. However, it is not invariant, as shown by the following example:

Similarly we can define a weighted distance function from A_s, which not only counts the number of bad letters, but gives a score to each such letter, based on the difference $i_k - k$ (or $j_k - k$).

Definition 9 (Weighted distance from A_s) *Let $s \geq 2$ be an integer. Let $w \in F$, such that its normal form is $\hat{w} = x_{i_1} \cdots x_{i_p} x_{j_n}^{-1} \cdots x_{j_1}^{-1}$. The weighted distance from A_s is defined as*

$$\overline{d_{A_s}}(w) = \sum_{k=1\ldots p}^{i_k - k \geq s} (i_k - k - s + 1) \ + \sum_{k=1\ldots n}^{j_k - k \geq s} (j_k - k - s + 1) \ + \ |p - n|$$

For each bad letter x_{i_k} or $x_{j_k}^{-1}$, $\overline{d_{A_s}}$ adds a positive integer. As such, it's a distance function, which is again not invariant (the example above works here too).

A somewhat different approach to defining distance from A_s arises from the observation that the number of bad letters can be less important than the maximum value of the differences $i_k - k$ and $j_k - k$ across the word, which measures the size of the violation. The difference between

the two distance functions roughly corresponds to the difference between the L_1 and L_∞ norms.

Let $\hat{w} = x_{i_1} \cdots x_{i_p} x_{j_n}^{-1} \cdots x_{j_1}^{-1}$. Suppose that for some integer k we have $i_k - k - s + 1 = m_p > 0$ and that m_p is the maximum for all i_k. By multiplying the word by $x_0^{m_p}$ we shift the position for all the original positive letters of w by m_p, and so all of the positive letters, including the first m x_0's have $i_k - k < s$. Similarly, if m_n is the maximum violation in the negative subword, multiplication by $x_0^{-m_n}$ on the right eliminates all violations among negative letters. However, this still does not mean that the word is in A_s, because the positive and negative lengths may differ. Let $\hat{w'}$ be the normal form obtained from $\hat{w}$ through multiplication by $x_0^{m_p}$ and $x_0^{-m_n}$ on the left and right, respectively. Let l_p and l_n be the corresponding lengths of the positive and negative parts of $\hat{w'}$. If $l_p - l_n > 0$, then $\hat{w'} x_0^{l_n - l_p} \in A_s$. If $l_p - l_n < 0$, then $x_0^{l_n - l_p} \hat{w'} \in A_s$. Altogether, any word can be changed to a word in A_s through multiplication by $m_p + m_n + |l_p + l_n|$ indices (when l_p and l_n are evaluated *after* multiplying by $x_0^{m_p}$ and $x_0^{-m_n}$).

This observation suggests the following distance function:

Definition 10 (Maximum-based distance from A_s) *Let $s \geq 2$ be an integer. Let $w \in F$, such that its normal form is $\hat{w} = x_{i_1} \cdots x_{i_p} x_{j_n}^{-1} \cdots x_{j_1}^{-1}$. Let*

$$m_p = \max\left(\{0\} \cup \{i_k - k - s + 1 : k = 1 \ldots p\}\right)$$

and

$$m_n = \max\left(\{0\} \cup \{j_k - k - s + 1 : k = 1 \ldots n\}\right) \ .$$

The maximum-based distance from A_s is defined as

$$d_{A_s}^m(w) = m_p + m_n + |(p + m_p) - (n + m_n)|$$

For every $w \in A_s$ m_p, m_n and $|p - n|$ are 0 by definition, while for every $w \notin A_s$ at least one of them has to be positive, so the $d_{A_s}^m$ is a distance function. It turns out that, unlike the two previously defined distance functions, $d_{A_s}^m$ is also invariant.

Claim 4 *$d_{A_s}^m$ is an invariant distance function.*

Proof. As with Claim 2, it's sufficient to prove that multiplication by a single generator of A_s does not change the distance from any word w to A_s. We will consider multiplications on the left by generators and their inverses. The multiplication on the right follows symmetrically.

Let $w = x_{i_1} \cdots x_{i_p} x_{j_n}^{-1} \cdots x_{j_1}^{-1}$, without loss of generality, in normal form. Consider the generator $x_0 x_t^{-1}$, where $1 \leq t \leq s$. Define w' as the normal form of $x_0 x_t^{-1} w$. For the parameters p, n, m_p, m_n of w, denote by p', n', m'_p, m'_n their corresponding values in w'.

From Lemma 1 it follows that each of the letters x_t^{-1} and x_0 can either be cancelled out with the appropriate inverse, decreasing the length by 1, or placed in its appropriate location, increasing the length by 1. There is a total of 4 possible options:

(1) *x_t^{-1} is cancelled out, but x_0 is not: $w' = x_0 x_{i_1} \cdots x_{i_m} x_{i_{m+2}} \cdots x_{i_p} x_{j_n}^{-1} \cdots x_{j_1}^{-1}$, where x_{t+m}^{-1} is cancelled out with $x_{i_{m+1}}$ after m applications of (R2). It follows that $p' = p$, $n' = n$ and $m'_n = m_n$ (because the negative letters are unaffected). Observe also that there can be no bad letters among the first m: indeed, (R2) is applied m times, for each $k = 1 \ldots m$ rewriting $x_{t+k-1}^{-1} x_{i_k} \to x_{i_k} x_{t+k}^{-1}$, so necessarily $i_k < t+k-1$ for all k, or equivalently, $i_k - k < t - 1 < s$. The multiplication by x_0 on the left only increases their relative positions, thus decreasing $i_k - k$. Now, any possible bad letters above i_m are unchanged, and neither is their relative position, so $m'_p = m_p$ and overall $d_{A_s}^m(w') = d_{A_s}^m(w)$.*

(2) *Both x_t^{-1} and x_0 are cancelled out: $w' = x_{i_1 - 1} \cdots x_{i_m - 1} x_{i_{m+2} - 1} \cdots x_{i_p - 1} x_{j_n - 1}^{-1} \cdots x_{j_{q+1} - 1}^{-1} x_0^{1-q}$. Here $p' = p - 1$, $n' = n - 1$ and $m'_n = m_n$ because all negative letters $x_{j_k}^{-1}$ with $j_k > 0$ had both their indices and their relative positions decreased by 1. The same thing applies to positive letters above i_m, which are the only positive letters that may be bad. So again, $m'_p = m_p$ and $d_{A_s}^m(w') = d_{A_s}^m(w)$.*

(3) *Neither x_t^{-1}, nor x_0 are cancelled out: $w' = x_0 x_{i_1} \cdots x_{i_m} x_{i_{m+1}+1} \cdots x_{i_p+1} x_{j_n+1}^{-1} \cdots x_{j_{q+1}+1}^{-1} \boldsymbol{x_{t+m}^{-1}} x_{j_q}^{-1} \cdots x_{j_1}^{-1}$. Here $p' = p + 1$ and $n' = n + 1$. Due to the former observation, bad positive letters may only exist beyond the first m. All these letters had their indices i_k and their relative positions k increased by 1, so the difference is preserved and $m'_p = m_p$. Among the negative letters, only the letters whose indices increased, also had their relative position increased, so $j_k - k$ is preserved for all the original letters of w. Hence, $m'_n \geq m_n$ and the only situation when it may actually increase is when the new maximum is attained at the new letter, i.e., $m'_n = (t + m) - (q + 1) - s + 1 > m_n$. Because $t \leq s$, $m \leq p$ and $q \leq n$, we have $m'_n \leq p - q$, from which it follows that*

$$(p'+m'_p)-(n'+m'_n) = (p'-n')+(m'_p-m'_n) = (p+1)-(n+1)+m_p-m'_n \geq$$

$$\geq m_p + (p-n) - (p-q) = m_p + q - n \geq 0$$

Assuming $m'_n > m_n$, it's obvious that

$$(p-n)+(m_p-m_n) > (p'-n')+(m'_p-m'_n) \geq 0 \ ,$$

and so if m_n increases, $|(p+m_p)-(n+m_n)|$ decreases by the same amount, and overall $d^m_{A_s}(w') = d^m_{A_s}(w)$.

(4) *x_t^{-1} is not cancelled out, but x_0 is: $w' = x_{i_1-1}\cdots x_{i_m-1}x_{i_{m+1}}\cdots x_{i_p}x_{j_n}^{-1}\cdots x_{j_{q+1}}^{-1}\boldsymbol{x_{t+m-1}^{-1}}x_{j_q-1}^{-1}\cdots x_{j_{r+1}-1}^{-1}x_0^{1-r}$, where $p'=p$, $n'=n$, $m'_p = m_p$ (because the first m positive letters, whose indices have changed, contained no bad letters), and m'_n again may only increase, if it's attained at x_{t+m-1}^{-1}. Repeating the same calculations shows that $d^m_{A_s}(w') = d^m_{A_s}(w)$ in this case too.*

Now consider the inverse $x_t x_0^{-1}$ and denote $w' = x_t x_0^{-1} w$. The four possible outcomes are:

(1) *x_0^{-1} is cancelled out, but x_t is not: x_0^{-1} can only be cancelled out if $i_1 = 0$, and the resulting word is: $w' = x_{i_2}\cdots x_{i_m}\boldsymbol{x_{t+m-1}}x_{i_{m+1}}\cdots x_{i_p}x_{j_n}^{-1}\cdots x_{j_1}^{-1}$. Here $p'=p$, $n'=n$, $m'_n = m_n$ (negative part is not affected) and $m'_p = m_p$ because the letters x_{i_2} to x_{i_m} cannot be bad and the relative position of other positive letters has not changed.*

(2) *Both x_0^{-1} and x_t are cancelled out: Assuming x_t is cancelled out (due to violation of **(NF2)**) with $x_{j_q}^{-1}$, $w' = x_{i_2}\cdots x_{i_m}x_{i_{m+1}-1}\cdots x_{i_p-1}$ $x_{j_n-1}^{-1}\cdots x_{j_{q+1}-1}^{-1}x_{j_{q-1}}^{-1}\cdots x_{j_1}^{-1}$. Here $p' = p-1$, $n' = n-1$, $m'_p = m_p$, because x_{i_2} to x_{i_m} cannot be bad and the relative position of other positive letters has not changed, and $m'_n = m_n$, because the letters whose positions shifted also had their indices decreased.*

(3) *Neither x_0^{-1}, nor x_t are cancelled out. $w' = x_{i_1+2}\cdots x_{i_m+2}\boldsymbol{x_{t+m}}$ $x_{i_{m+1}+1}\cdots x_{i_p+1}x_{j_n+1}^{-1}\cdots x_{j_q+1}^{-1}\boldsymbol{x_0^{-q}}$. Here $p'=p+1$, $n'=n+1$, $m'_p = m_p$, because indices above i_m grew by 1, as did their positions, and indices $i_1,\ldots,i_m$ cannot be bad, and also $m'_n = m_n$, because all letters whose indices increased (j_q and above) shifted in position accordingly.*

(4) *x_0^{-1} is not cancelled out, but x_t is: $w' = x_{i_1+2}\cdots x_{i_m+2}x_{i_{m+1}}\cdots x_{i_p}$ $x_{j_n}^{-1}\cdots x_{j_{q+1}}^{-1}x_{j_{q-1}+1}^{-1}\cdots x_{j_r+1}^{-1}\boldsymbol{x_0^{-r}}$, the cancelled pair being $(x_{t+m}, x_{j_q}^{-1})$, where $j_q = t+m$. In this case, any positive letters that can be bad kept their indices and positions, the negative letters $j_{r+1},\ldots,j_{q-1}$ had their indices and positions shifted, while the letters $j_{q+1},\ldots,j_n$ kept their indices and positions. So $m'_p = m_p$ and $m'_n = m_n$ and obviously $p'=p$ and $n'=n$.*

We see that in all the possible cases, $d^m_{A_s}(w') = d^m_{A_s}(w)$. This completes the proof. □

6 Experimental results

To test the applicability of the subgroup distance functions to cryptanalysis, we tested Algorithm 1 against the Shpilrain-Ushakov protocol in the settings of Thompson's group. Initially, each of the five distance functions presented in the previous section was tested separately: we generated a public element azb and tried to recover a single private element a or b from it. For the recovery of a, the functions d_{B_s} and $\overline{d_{B_s}}$ were used to assess the quality of the complements. Similarly, for the recovery of b, we tried d_{A_s}, $\overline{d_{A_s}}$ and $d^m_{A_s}$.

For each distance function, the experiment was run at least 1000 times, each time with new, randomly generated keys, with the minimum recommended parameters of $s = 3, L = 256$. The bound $N = 2L$ was chosen on the number of iterations, since preliminary experiments have shown that the success rates do not increase beyond that. The results are summarized in Table 1. It can be seen that the distance functions d_{B_s} and $d^m_{A_s}$ noticeably outperform the other distance functions, in recovering a and b, respectively. The fact that $d^m_{A_s}$ clearly outperforms its counterparts suggests that the notion of invariance may be useful for assessing the suitability of a given distance function.

Table 1. Success rates for the different subgroup distance functions

	d_{B_s}	$\overline{d_{B_s}}$	d_{A_s}	$\overline{d_{A_s}}$	$d^m_{A_s}$
Recovery probability	11.7%	3.4%	3.7%	3.4%	23.3%

Preliminary experiments have shown that, regardless of the settings, the success probability of finding a_1 given $a_1 z b_1$ is similar to that of finding a_2^{-1} given $a_2^{-1} z^{-1} b_2^{-1}$. A similar assertion holds for b_2 and b_1^{-1}. Therefore, in order to estimate the overall success rate against an actual instance of the cryptosystem, it's sufficient to try to recover one of the four a's and b's. If we denote by p_a and p_b the probability of successfully recovering a and b, respectively, and assume that all probabilities are independent, then, the expected total success rate is roughly $1 - (1 - p_a)^2(1 - p_b)^2$ (because each instance of the protocol contains two elements of type a and two of type b).

When the success rates of the two best distance functions, d_{B_s} for a and $d^m_{A_s}$ for b, are combined, the expected overall success probability, ac-

cording to the above, is between 50% and 54%, which was experimentally verified. Note that this attack is very efficient, since it involves no backtracking, no lookahead, and no analysis of suboptimal partial results: it tries to peel off the generators by a greedy algorithm, which considers only locally optimal steps. Attacking each key required only a few seconds on a single PC, and it is very surprising that such a simple attack succeeds about half the time. These results are much better than those achieved by length-based attacks of similar complexity on this cryptosystem (see [9]).

It is interesting to note that possible extensions of the attack, such as memorizing many suboptimal partial solutions or using significant lookahead (which require much higher time and space complexities) have different effects on length-based and distance-based attacks. While it was shown in [9] that these extensions greatly improve the success rates of the length-based attack, experiments with the distance-based attack, with similar values of the memory and lookahead parameters, showed almost no improvement. However, the situation may be very different for other cryptosystems and other subgroup distance functions.

To further test the performance of the distance functions, several experiments were run with different values of the parameters (s, L). We used the combination of d_{B_s} and $d^m_{A_s}$, which was established as the best in the former experiment. Table 2 shows the overall success probability, for $L \in \{128, 256, 320, 512, 640, 960\}$ and $s \in \{3, 5, 8\}$. The success rates stay remarkably consistent across different lengths for a given s, and even increasing s does not cause a significant drop. The time complexity of the attack grows linearly with s and roughly quadratically with L, with most of the time being spent on computing normal forms of elements in the group. For the largest parameters presented here, the attack still required under a minute in most cases. This suggests that for the Shpilrain-Ushakov cryptosystem the distance-based attack remains a viable threat, even when the security parameters s and L are increased beyond the original recommendations.

Table 2. Success rates for different combinations of (s, L)

	$L = 128$	$L = 256$	$L = 320$	$L = 512$	$L = 640$	$L = 960$
$s = 3$	51.7%	47.9%	55.5%	51.2%	50.4%	52.6%
$s = 5$	46.0%	47.1%	48.4%	51.1%	48.2%	48.3%
$s = 8$	36.2%	42.8%	41.3%	46.5%	42.4%	50.3%

7 Conclusion

We introduced a novel form of heuristic attacks on public key cryptosystems that are based on combinatorial group theory, using functions that estimate the distance of group elements to a given subgroup. Our results demonstrate that these distance-based attacks can achieve significantly better success rates than previously suggested length-based attacks of similar complexity, and thus they are a potential threat to any cryptosystem based on equations in a noncommutative group, which takes its elements from specific subgroups. It will be interesting to test this approach for other groups and other protocols.

References

1. I. Anshel, M. Anshel and D. Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters **6** (1999), 287–291.
2. E. Artin, *Theory of Braids*, Annals of Mathematics **48** (1947), 127–136.
3. J.W. Cannon, W.J. Floyd and W.R. Parry, *Introductory notes on Richard Thompson's groups*, L'Enseignement Mathematique (2) **42** (1996), 215–256.
4. D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, *Length-based conjugacy search in the Braid group*, Contemporary Mathematics **418** (2006), 75–87.
5. D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, *Probabilistic solutions of equations in the braid group*, Advances in Applied Mathematics **35** (2005), 323–334.
6. J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Workshop SECI02 Sécurité de la Communication sur Internet (2002).
7. K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J. Kang and C. Park, *New Public-Key Cryptosystem Using Braid Groups*, Lecture Notes in Computer Science **1880** (2000), 166–183.
8. F. Matucci, *The Shpilrain-Ushakov Protocol for Thompson's Group F is always breakable*, e-print `arxiv.org/math/0607184` (2006).
9. D. Ruinskiy, A. Shamir and B. Tsaban, *Length-based cryptanalysis: The case of Thompson's group*, e-print `arxiv.org/cs/0607079` (2006).
10. V. Shpilrain, *Assessing security of some group based cryptosystems*, Contemporary Mathematics **360** (2004), 167–177.
11. V. Shpilrain and A. Ushakov, *The conjugacy search problem in public key cryptography: unnecessary and insufficient*, Applicable Algebra in Engineering, Communication and Computing **17** (2006), 291–302.
12. V. Shpilrain and A. Ushakov, *Thompson's group and public key cryptography*, ACNS 2005, Lecture Notes in Computer Science **3531** (2005), 151–164.