LENGTH-BASED CRYPTANALYSIS: THE CASE OF THOMPSON'S GROUP

DIMA RUINSKIY, ADI SHAMIR, AND BOAZ TSABAN

ABSTRACT. The length-based approach is a heuristic for solving randomly generated equations in groups which possess a reasonably behaved length function. We describe several improvements of the previously suggested length-based algorithms, that make them applicable to Thompson's group with significant success rates. In particular, this shows that the Shpilrain-Ushakov public key cryptosystem based on Thompson's group is insecure, and suggests that no practical public key cryptosystem based on this group can be secure.

(*Preliminary version. Comments are welcome.*)

1. Introduction

Noncommutative groups are often suggested as a platform for public key agreement protocols, and much research is dedicated to analyzing existing proposals and suggesting alternative ones (some examples are given in our bibliography and in the references therein).

One possible approach for attacking such systems was outlined by Hughes and Tannenbaum [5]. 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. Such a length function can be used to solve, heuristically, *arbitrary* random equations in the group [3].

In the case of the braid group, a practical realization of this approach was suggested in [3], and the method was extended in [4] to imply high success rates for subgroups of the braid group, which are of the type considered in some previously suggested cryptosystems (e.g., [1]).

This *length-based cryptanalysis* usually has smaller success rates than specialized attacks, but it has the advantage of being *generic* in the sense that, if there is a good length function on a group, then this attack applies with nontrivial success rates to all cryptosystems based on the same group (provided that an equation in the group can be extracted from the public information).

The third author is supported by the Koshland Fellowship.

The main problem with existing length-based algorithms is that they tend to perform well only when the underlying subgroup has few relations, i.e., it is not too far from the free group. In 2004, Shpilrain and Ushakov proposed a key exchange protocol that uses Richard Thompson's group F as its platform [10]. The particular subgroups suggested for the protocol have many relations and indeed, Shpilrain and Ushakov report a complete failure of a length-based attack on their cryptosystem [10].

In the sequel, we introduce several improvements to the length-based algorithm, that yield a tremendous boost in the success rates for full size instances of the cryptosystem. The generalized length-based algorithms presented here would be useful in testing the security of any future cryptosystem based on combinatorial group theoretic problems.

1.1. **History and related works.** The results reported here form the first practical cryptanalysis of the Shpilrain-Ushakov cryptosystem: The first version of our attack was announced in the Bochum Workshop *Algebraic Methods in Cryptography* (November 2005) [7]. An improved attack was announced in the CGC Bulletin on March 2006 [8].

While we were finalizing our paper for publication, a very elegant specialized attack on the same cryptosystem was announced by Matucci [6]. The main contribution of the present paper is thus the generalization of the length-based algorithms to make them applicable to a wider class of groups. Moreover, while our general attack can be easily adapted to other possible cryptosystems based on Thompson's group, Matucci's specialized methods may not.

2. The basic length-based attack

Let G be a finitely generated group with $S_G = \{g_1^{\pm 1}, \dots, g_k^{\pm 1}\}$ being its set of generators. Assume that $x \in G$ was generated as a product, $x = x_1 \cdots x_n$, where each $x_i \in S_G$ is chosen at random according to some nontrivial (e.g., uniform) distribution on S_G. Assume further that $w \in G$ is chosen in a way independent of x, and that x, w are unknown, but $z = xw \in G$ is known. Assume that there is a "length function" $\ell(g)$ on the elements of G, such that with a nontrivial probability,

$$\ell(x_1^{-1}z) < \ell(z) < \ell(x_j z)$$

for each $x_j \neq x_1^{-1}$. To retrieve x, we can try to "peel off" the generators that compose it, one by one, using the following procedure.

Algorithm 1 (Length-based attack)**.**

(1) Let $j \leftarrow 1$ and $y \leftarrow z$.

(2) For each $g \in S_G$ compute $g^{-1}y$.
(3) Consider the $h \in S_G$ that minimizes $\ell(h^{-1}y)$. (If several such h's exist, choose one arbitrarily or randomly).
(4) (a) If $j = n$, terminate.
(b) Otherwise, Let $h_j \leftarrow h$, $j \leftarrow j+1$ and $y \leftarrow h^{-1}y$ and return to step 2.

If ℓ is a good length function, then in step (3), with some nontrivial probability, $h = x_1$ (or at least y can be rewritten as a product of n or fewer generators, where h is the first). It follows that with a nontrivial probability, $x = h_1 h_2 \cdots h_n$ after termination.

Instead of assuming that n is known, we can assume that there is a known, reasonably sized, bound N on n, and then terminate the run after N steps and consider it successful if for some $k \leq N$, $x = h_1 \cdot h_2 \cdots h_k$. This way, we obtain a short list of N candidates for x. In many practical situations each suggestion for a solution can be tested, so this is equally good.

However, in practice the best known length functions in many types of groups are not good enough for Algorithm 1 to succeed with noticeably probability. This is shown in [3], and is demonstrated further by the Shpilrain-Ushakov key agreement protocol.

3. The Shpilrain-Ushakov Key agreement Protocol

This section is entirely based on [10].

3.1. **Thompson's group.**

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

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

Throughout, by *atom* we mean a generator x_i (a *positive* atom) or its inverse (a *negative* atom).

Each $w \in F$ admits a unique *normal form* [2]

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

where $i_1 \leq \cdots \leq i_r$, $j_1 \leq \cdots \leq j_t$, and if x_i and x_i^{-1} both occur in this form, then either x_{i+1} or x_{i+1}^{-1} occurs as well. The transformation of an element of F into its normal form is very efficient.

Definition 1. *The normal form length* of an element $w \in F$, $\ell_{\mathrm{NF}}(w)$, is the number of generators in its normal form: If $w = x_{i_1} \cdots x_{i_r} x_{j_t}^{-1} \cdots x_{j_1}^{-1}$ is in normal form, then $\ell_{\mathrm{NF}}(w) = r + t$.

3.2. **The protocol.**

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

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

3.3. **Settings and parameters.** Let $s \geq 2$ be some positive integer. Let $S_A = \{x_0 x_1^{-1}, \ldots, x_0 x_s^{-1}\}$, $S_B = \{x_{s+1}, \ldots, x_{2s}\}$ and $S_W = \{x_0, \ldots, x_{s+2}\}$. Denote by A, B and W the subgroups of F generated by S_A, S_B and S_W, respectively. A and B commute, as required.

Let L be a positive integer. The words $a_1, a_2 \in A$, $b_1, b_2 \in B$, and $w \in W$ are all chosen of normal form length L: Let X be A, B, or W. The distribution according to which an element of length L in X is chosen is defined by the following. Start with the empty word, and multiply it on the right by a (uniformly) randomly selected generator, inversed with probability $1/2$, from the set S_X. Continue this procedure until the normal form of the word has length L.

For practical implementation of the protocol, it is suggested in [10] to use $s \in \{3, 4, \ldots, 8\}$ and $L \in \{256, 258, \ldots, 320\}$.

4. Success rates for the basic length attack

We are given w, u_1, u_2, where $u_1 = a_1 w b_1$ and $u_2 = b_2 w a_2$. This gives 4 equations:

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

We can apply Algorithm 1 to each equation, hoping that its leftmost unknown element will appear in the resulting list of candidates. Note that even a single success out of the 4 runs suffices to find the shared key.

Here n, the number of generators multiplied to obtain each element, is not known. We took the bound $2L$ on n, as experiments show that the success probability does not increase noticeably when we increase the bound further. This is the case in all experiments described in this paper.

Experiments show that the success probability of finding a_1 given a_1wb_1 is the same as that of finding a_2^{-1} given $a_2^{-1}w^{-1}b_2^{-1}$ (despite using the same w in both cases). A similar assertion holds for b_2 and b_1^{-1}. We may therefore describe the task in a compact manner:

Given awb, try to recover either a or b.

The probabilities p_a, p_b of successfully recovering a and b (respectively) induce the total success rate by $1-(1-p_a)^2(1-p_b)^2$.

The attack was tested for the cut-down parameters $s = 2$ and $L \in \{8, 16, 32, 64, 128\}$. The results, presented in Table 1, are not satisfactory: Already for $L = 128$ (recall that the recommended parameter is $L = 256$), the attack failed in all of our $10,000$ tries.

Table 1. Success rates for the basic length attack

L	a recovery	b recovery	Total
8	63.53%	32.43%	94.51%
16	26.59%	10.84%	57.96%
32	3.54%	1.54%	9.77%
64	0.08%	0.05%	0.24%
128	0%	0%	0%

5. Using memory and avoiding repetitions

To improve the success rates, it was suggested in [4] to keep in memory, after each step, not only the element that yielded the shortest length, but a fixed number $M > 1$ of elements with the shortest lengths. Then, in the next step, all possible extensions of each one of the M elements in memory with each one of the generators are tested and again the best M elements are kept.

The time and space complexities of this attack increase linearly with M. The previous length-based attack is the special case of the memory attack, where $M = 1$. Except for pathological cases, the success rates increase when M is increased. See [4] for more details.

We have implemented this attack with $M = 1,024$, and obtained a very small success rate: 0.7%. The experiments in [4] yielded much higher success rates for other groups. The reason for this seems to be that the length-based approach is more suitable for groups which have few relations (i.e., are close to being free) [3], whereas here the underlying groups have many relations. To partially overcome this problem, we introduce the following.

Improvement 1 (Avoiding repetitions)**.** Keep a hash list. During the run of the algorithm, before checking the length score of an element, check if it is already in the hash list (i.e., it has been considered in the past). If it is, ignore it. Otherwise, add it to the list and proceed as usual.

In the case $M = 1$, Improvement 1 forces the algorithm not to get into loops. Thus, this improvement can be viewed as a generalization of avoiding loops to the case of arbitrary M. The philosophy is that we prefer new candidates over candidates that were considered in the past and lost the battle against other candidates.

5.1. **Results.** The attack was first tested for the cut-down parameters $s = 2$ and $L = 64$, with $M \in \{1, 2, 4, 8, 16, 32, 64\}$. The results are presented in Table 2.

TABLE 2. Success rates for $s = 2, L = 64$

M	a recovery	b recovery	Total
1	0.1%	0.05%	0.27%
2	1.51%	1.01%	4.94%
4	8.58%	8.03%	29.63%
8	19.71%	15.37%	54.96%
16	32.44%	21.37%	74.71%
32	43.38%	25.14%	83.93%
64	52.39%	28.29%	90.53%

In the case of the smallest recommended parameters $s = 3$ and $L = 256$, the probabilities are not as high, so we also tried $M = 256$. The results are summarized in Table 3.

TABLE 3. Success rates for $s = 3, L = 256$

M	a recovery	b recovery	Total
1	0%	0%	0%
4	0%	0%	0%
16	2.3%	1.1%	6.63%
64	10.8%	2.3%	24.05%
256	14.3%	3.8%	32.03%

Comparing the success rate of 32% for $M = 256$ in Table 3 to the success rate of 0.7% for $M = 1,024$ obtained before Improvement 1 was implemented, we see that this improvement is crucial for the current system.

A success probability of 32% should be considered a complete cryptanalysis of the suggested cryptosystem. We will, however, describe additional improvements, for two reasons.

Generality. The Shpilrain-Ushakov cryptosystem is just a test case for our algorithms. Our main aim is to obtain generic algorithms that will also work when other groups are used, or when Thompson's group is used in a different way.

Iterability. As pointed out by Shpilrain [9], there is a very simple fix for key agreement protocols that are broken in probability less than p: Agree on k independent keys in parallel, and xor them all to obtained the shared key. The probability of breaking the shared key is at most p^k. In other words, if a system broken with probability p_0 or higher is considered insecure, and k parallel keys are xored, then the attack on a single key should succeed in probability at least $p_0^{1/k}$. If we consider a parallel agreement on up to 100 keys practical, and require the probability of breaking all of them to be below 2^{-64}, then we must aim at a success rate of at least

$$2^{-64/100} \approx 64\%.$$

For $p_0 = 2^{-32}$, we should aim at 80%.

6. Look-ahead

Another extension of the basic attack is done by testing in each step not just the $2k$ generators in S_G, but all the $(2k)^t$ t-tuples of generators $g_{i_1}^{\pm 1} \cdots g_{i_t}^{\pm 1}$. This is called *look-ahead of depth t* [5, 3]. The complexity of this approach grows exponentially with t.

6.1. **Results.** The attack was tested for $s = 2, L = 64$ for $M = 4$ and $t \in \{1, 2, 3, 4, 5\}$. The results are presented in Table 4.

Taking the complexity into account, and comparing these results to those of the previous section, it follows that increasing M is better than using look-ahead. This was also observed in [3] for other settings.

TABLE 4. Results of attack in toy model for different look-ahead depths

t	a recovery	b recovery	Total
1	8.53%	7.74%	29.17%
2	15.25%	13.08%	46.43%
3	28.55%	19.40%	68.65%
4	38.87%	23.59%	80.55%
5	45.55%	27.20%	86.72%

7. Alternative length functions

The length function is the key measure used to evaluate the quality of candidates in the attack algorithm. In addition to the normal form length (Definition 1), we tried several additional length functions.

Instead of using the infinite presentation of F (Equation 1), we can use finite presentations. An extreme example is presenting F using just the generators x_0 and x_1. This is possible since, for any $k \geq 2$:

$$x_k^{\pm 1} = x_0^{-(k-1)} x_1^{\pm 1} x_0^{k-1} \tag{2}$$

Definition 2. The 2-*atom length* of an element $w \in F$, $\ell_{2\mathrm{A}}(w)$, is the number of generators in the presentation of w obtained from its normal form by replacing each $x_k^{\pm 1}$ appearing in the normal form, $k \geq 2$, by the string $x_0^{-(k-1)} x_1^{\pm 1} x_0^{k-1}$ of length $2k-1$.

The above presentation is not particularly compact, and we may apply to it the trivial (free group) reductions $gg^{-1} = 1$.

Definition 3. The *reduced* 2-*atom length* of an element $w \in F$, $\ell_{2\mathrm{R}}(w)$, is the length of the free-reduced form of the word obtained in Definition 2.

7.1. **Comparison.** Table 5 compares the performance of the three length functions for three sets of parameters. The alternative length functions turn out better than the normal form length.

TABLE 5. Total success rates for different length functions

Length function	ℓ_{NF}	$\ell_{2\mathrm{A}}$	$\ell_{2\mathrm{R}}$
s=2,L=32,M=1	9.8%	53.7%	27.3%
s=2,L=64,M=8	52.5%	71.2%	63.2%
s=2,L=128,M=4	3.3%	12.4%	22.2%

8. Automorphism attacks

Let $g \in F$ be a fixed word. Conjugation by g, $x \mapsto g^{-1}xg$, is an automorphism of F. For a given g, the cryptosystem defined in 3.3 can be constructed using the subgroups $g^{-1}Ag$, $g^{-1}Bg$ and $g^{-1}Wg$. This is equivalent to using the original groups, and conjugating the generated keys.

Any conjugation transforms the original key space to another, equivalent space. The introduced heuristic algorithms may yield different results for different key spaces. It's possible that certain keys that could not be cracked in the original space, can be cracked by transforming them to another space, executing the attack algorithm, then applying the inverse transformation.

An alternative approach to this procedure is executing the original algorithms in the original key space, using an alternative length function, which takes the conjugating element into account.

While the following can be defined for any length function, we used $\ell = \ell_{\mathrm{NF}}$ in our experiments.

Definition 4. For a fixed $g \in F$ and for any $w \in F$, the *g-conjugate* length of w is:

$$\ell^g(w) = \ell(g^{-1}wg)$$

Since it is unlikely that for a single arbitrary g we will get a good length function, we can try an average version of Definition 4.

Definition 5. For a fixed nonempty set $T \subset F$ and any $w \in F$, the *T-conjugate* length of w is:

$$\ell^T(w) = \frac{1}{|T|} \sum_{g \in T} \ell(g^{-1}wg)$$

The idea of using a length function induced by an automorphism is to introduce some randomization into the system, so that equations that cannot be solved using one automorphism may be solvable using another automorphism, if the correlation is not too high. In general it is expected that a length function induced by a particular automorphism would yield worse results than a standard length function, but the hope is that attacking with many different automorphisms would do better. We now show that this is the case.

8.1. Results.

The experiments were run for parameters $s = 2, L = 64$ and with memory $M = 4$.

First experiment: A single alternative length. For each set of keys generated, the attack was first executed using the regular (normal form) length function. Then the same set of keys was attacked using the T-conjugate length function, where $|T|$ was 1, 3, or 9. In all cases, the conjugators were random words of (normal form) length 8. The success rates appear in Table 6. The last column tells the probability that at least one of the attacks succeeded. It is observed that the correlation between the different length functions is rather high. This phenomenon repeated itself in other experiments as well.

TABLE 6. Success rates for conjugate length functions

$\|T\|$	ℓ_{NF}	ℓ^T	Combined
1	28.8%	3.4%	29.4%
3	30.6%	29.8%	41.5%
9	30.8%	10.7%	32.6%

Second experiment: Many alternative lengths. Here the normal form length was used in conjunction with the functions ℓ^g, g being either of length 8 or length 16, selected uniformly at random. The number of different length functions was 10 or 100. Table 7 displays the results. For each experiment we provide the success rate of the normal form length function, the minimum and maximum success rates attained by any conjugate length function, as well as the average and standard deviation of the distributions. The general attack is as follows: Try to crack the cryptosystem with ℓ_{NF}. If this fails, try the first conjugate length function. If this fails, try the second, etc., until you finish all length functions in the list. The total success rates appear in the last column.

TABLE 7. Simultaneous effect of multiple conjugate length functions

$\|g\|$	$\#_{func}$	NF	Max	Min	Avg	Std	Total
8	10	31.20%	13.80%	2.5%	8.85%	4.93%	48.8%
8	100	29.94%	32.60%	0.4%	12.05%	7.16%	81.60%
16	100	26.56%	30.11%	0%	7.76%	6.27%	78.00%

While a big improvement is observed, it is also seen that there remain substantial correlations the probability is not close enough to 100%, which is necessary to deal with many xored keys.

9. Alternative solutions

Thus far, we have concentrated on the problem: Given w and awb, find the *original* a, or rather, a short list containing a. But as Shpilrain and Ushakov point out [11], it suffices to solve the following.

Problem 1 (Decomposition)**.** Given $w \in F$ and $u = awb$ where $a \in A$ and $b \in B$, find some elements $\tilde{a} \in A$ and $\tilde{b} \in B$, such that $\tilde{a}w\tilde{b} = awb$.

Indeed, assume that the attacker, given $u_1 = a_1wb_1$, finds $\tilde{a}_1 \in A$ and $\tilde{b}_1 \in B$, such that $\tilde{a}_1w\tilde{b}_1 = a_1wb_1$. Then, because $u_2 = b_2wa_2$ is known, the attacker can compute

$$\tilde{a}_1u_2\tilde{b}_1 = \tilde{a}_1b_2wa_2\tilde{b}_1 = b_2\tilde{a}_1w\tilde{b}_1a_2 = b_2u_1a_2 = K_B,$$

and similarly for b_2wa_2.

Consider Problem 1. To each $\tilde{a} \in A$ we can compute its *complement* $\tilde{b} = w^{-1}\tilde{a}^{-1}u = w^{-1}\tilde{a}^{-1}(awb)$, such that $\tilde{a}w\tilde{b} = awb$. 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 determine whether $\tilde{b} \in B$ (or $\tilde{a} \in A$ in the second case). This *membership decision problem* turns out to be trivial in our case.

A is exactly the set of all elements in F, whose normal form is of the type

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

i.e., positive and negative parts are of the same length, and in addition $i_k - k < s$ and $j_k - k < s$ for every $k = 1, \dots, m$. B consists of the elements in F, whose normal form does not contain any of the generators $x_0, x_1, \dots, x_s$ (or their inverses) [10]. In both cases, the conditions are straightforward to check.

Following is an algorithm for solving Problem 1.

Algorithm 2 (Alternative solution search)**.**

(1) Execute Algorithm 1 (with any of the introduced extensions), attempting to recover a.
(2) For each candidate (prefix) $\tilde{a}$ encountered during any step of the algorithm, compute the complement $\tilde{b} = w^{-1}\tilde{a}^{-1}u$.
(3) If $\tilde{b} \in B$, halt.

Note that if the algorithm halts in step (3), then $\tilde{a}, \tilde{b}$ is a solution for the decomposition problem.

The above procedure can be executed separately for each of the four given equations. It suffices to recover a single matching pair in any of the four runs to effectively break the cryptosystem.

It should be stressed that solving the group membership is not necessary in order to cryptanalyze the system. Indeed, given $u_1 = a_1 w b_1$ and $u_2 = b_2 w a_2$, we can apply Algorithm 2 to, e.g., $u_1 = a_1 w b_1$, replacing its step (3) by checking whether the suggested key $\tilde{a} u_2 \tilde{b}$ succeeds in decrypting the information encrypted between Alice and Bob. Our experiments showed that for all reasonable parameters, this formally stronger attack has the same success rates. However, this alternative approach is useful in cases where the group membership problem is difficult.

9.1. **Results.** We repeated the experiments of Section 4, but this time counted every equivalent pair as a success. Table 8 shows the results. To emphasize the difference, we repeated the results of Table 1 in the last column. We observe a dramatic improvement.

TABLE 8. Equivalent solution attack for different key lengths

L	a recovery	b recovery	Total	Original
8	71.57%	39.67%	97.00%	94.51%
16	39.52%	20.83%	76.55%	57.96%
32	18.34%	9.24%	46.08%	9.77%
64	11.98%	6.19%	32.93%	0.24%
128	11.24%	5.41%	30.15%	0%

Table 9 shows the results of the equivalent solution attack when $s = 2, L = 64$, for various memory sizes. As above, we repeated the results of Table 2 for reference. Again, a noticeable improvement can be seen.

TABLE 9. Equivalent solution attack for various memory sizes

M	a recovery	b recovery	Total	Original
1	11.98%	6.19%	32.93%	0.24%
2	18.24%	8.38%	44.91%	4.94%
4	27.75%	17.20%	64.27%	29.63%
8	41.58%	26.53%	80.87%	54.96%
16	55.07%	31.57%	89.37%	74.71%

Finally, in Table 10 we compared the behavior of the three primary length functions when equivalent solutions are taken into account. Unlike in Table 5 the performance of the normal form length is not as bad, compared to the others. Again, the reduced 2-atom length seems to get better when L is increased (But see Table 13 below).

Table 10. Equivalent solution attack for different length functions

Length function	ℓ_{NF}	ℓ_{2A}	ℓ_{2R}
s=2,L=32,M=1	43.2%	76.8%	59.0%
s=2,L=64,M=8	80.5%	85.2%	78.2%
s=2,L=128,M=4	38.9%	35.6%	51.9%

10. The real cryptosystem

Recall that Shpilrain and Ushakov suggested using $s \in \{3, 4, \ldots, 8\}$ and $L \in \{256, 258, \ldots, 320\}$ [10]. Consider the case of the minimal recommended parameters $s = 3, L = 256$. Our experiments showed that the best attack out of those suggested in this paper is the one looking for equivalent solutions (Algorithm 2), while avoiding repetitions and using memory. Table 11 shows the results for the normal form length function.

Table 11. Success rates against the real cryptosystem

M	a recovery	b recovery	Total
1	8.2%	6.6%	27.3%
4	11.9%	7.6%	34.4%
16	16.3%	10.2%	44.6%
64	27.6%	16.6%	62.8%
256	34.5%	18.6%	69.4%

While ℓ_{2A} is better when $M = 1$, it is rather poor for larger M, whereas ℓ_{NF} and ℓ_{2R} preform roughly the same.

Table 12. Different length functions against the real cryptosystem

M	ℓ_{NF}	ℓ_{2A}	ℓ_{2R}
1	27.3%	51.7%	28.8%
4	34.4%	22.0%	39.2%
16	44.6%	19.0%	50.0%
64	62.8%	20.1%	59.8%
256	69.4%	19.7%	67.2%

Finally, in Table 13 we tested ℓ_{NF} and ℓ_{2R} for the parameters $s = 8, L = 320$ and M up to 64. Again we see a similar performance, slightly lower than in the case $s = 3, L = 256$.

The Shpilrain-Ushakov cryptosystem is therefore broken, even if iterated a medium number of times.

Table 13. Different length functions against the real cryptosystem

M	ℓ_{NF}	ℓ_{2R}
1	27.58%	30.28%
4	31.44%	37.12%
16	41.63%	41.48%
64	48.59%	50.16%

11. Conclusions

We have described several improvements on the standard length based attack and its memory extensions. They include:

(1) Avoiding repetitions, which is especially important in groups which are far from being free;
(2) Attacking each key multiple times, by applying each time a random automorphism, or equivalently taking the length function induced by the automorphism;
(3) Looking for alternative solutions which are not necessarily the ones used to generate the equations.

We have tested our improvements against the Shpilrain-Ushakov cryptosystem, and in this case each of them increased the success probability substantially.

We did not find any clear advantage of one of the extensions (2),(3) over the other. Combinations of these extensions were also tried, but they did not yield significant improvements. However, there could be other public key cryptosystems where one of these two approaches substantially outperforms the other.

The important advantage of our approach is that it is generic and can be easily adjusted to any cryptosystem based on a group that admits a reasonable length function on its elements. As such, we feel that this approach may become a useful tool in cryptanalysis of additional public key protocols. No cryptosystem leading to equations in a noncommutative group can be considered secure before tested against these attacks.

References

[1] I. Anshel, M. Anshel and D. Goldfeld, *An algebraic method for public-key cryptography*, Mathematical Research Letters **6** (1999), 287–291.

[2] 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.

[3] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, *Length-based conjugacy search in the Braid group*, Contemporary Mathematics, to appear. `arxiv.org/math/0209267` (2002)

[4] 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.

[5] J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Workshop SECI02 Sécurité de la Communication sur Internet (2002).

[6] F. Matucci, *The Shpilrain-Ushakov Protocol for Thompson's Group F is always breakable*, e-print `arxiv.org/math/0607184` (2006).

[7] D. Ruinskiy, A. Shamir, and B. Tsaban, *Cryptanalysis of the Shpilrain-Ushakov Thompson group cryptosystem (preliminary announcement)*, `http://homepage.ruhr-uni-bochum.de/Arkadius.Kalka/workshop05/articles/researchannouncement.pdf` (2005).

[8] D. Ruinskiy, A. Shamir, and B. Tsaban, *A substantial improvement on the decomposition problem in Thompson's group*, CGC Bulletin **5** (March 2006), Item 7.

[9] V. Shpilrain, *Assessing security of some group based cryptosystems*, Contemporary Mathematics **360** (2004), 167–177.

[10] V. Shpilrain and A. Ushakov, *Thompson's group and public key cryptography*, ACNS 2005, Lecture Notes in Computer Science **3531** (2005), 151–164.

[11] V. Shpilrain and A. Ushakov, *The conjugacy search problem in public key cryptography: unnecessary and insufficient*, Applicable Algebra in Engineering, Communication and Computing, to appear.

Faculty of Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel

E-mail address: {`dmitriy.ruinskiy, adi.shamir, boaz.tsaban`}`@weizmann.ac.il`