Attacking a Public Key Cryptosystem Based on Tree Replacement

María Isabel González Vasco* and David Pérez García†

Área de Matemática Aplicada, Universidad Rey Juan Carlos
C/ Tulipán s/n. 28933, Móstoles, Madrid, Spain
{migonzalez, dperezg} @escet.urjc.es

Abstract

We point out several security flaws in the cryptosystem based on tree replacement systems proposed by Samuel, Thomas, Abisha and Subramanian at INDOCRYPT 2002. Due to the success of (among others) very simple ciphertext-only attacks, we evidence that this system does not, in its present form, offer acceptable security guarantees for cryptographic applications.

Keywords: Cryptanalysis, Public Key Cryptosystems Tree Replacement systems

1 Introduction

In the recent years, the unceasing harassment of quantum computing to number-theoretical cryptographic tools has encouraged an intensive search for cryptographic primitives arising from different fields of mathematics and theoretical computer science. A particularly active field of research has been that of constructing public key cryptosystems from word/rewritting problems in different structures (formal languages [9, 10], groups [11, 12, 13], monoids [1] etc.). However, up until now all the proposed schemes have been identified as insecure [3, 4, 5, 6].

At INDOCRYPT 2002, Samuel, Thomas, Abisha and Subramanian proposed a public key encryption scheme based on tree replacement systems.

*Work partially supported by project BFM2001-3239-C03-01.
†Work partially supported by project BFM2001-1284.

The spirit of their proposal is quite on the line of those mentioned above, though their approach presented also the nice feature of representing both plaintext and ciphertex messages in the form of trees, exploiting thus the nice computational features of this discrete mathematical structure. Unfortunately, we have identified several security flaws in their proposal, some of which actually arise from the tree representation.

This contribution is organized as follows; in Section 2 we briefly describe the scheme of Samuel et al. and its building blocks. Thereafter, we detail the identified flaws on the system in Section 3. In Section 3.1 we remark the necessity of detailing a careful key generation procedure in order to ensure uniqueness of decryption. In this direction we study in Section 3.1.1 the case when no variables are involved in the tree replacement system. Section 3.2. is devoted to the description of the attacks we have successfully carried out into the scheme. We conclude by pointing out roughly possible modifications of the original design that could prevent our attacks.

2 The Scheme of Samuel et al.

We give a brief description of the Public Key Cryptosystem proposed in [8] by Samuel et al.

Recall that, given a finite alphabet Υ and a finite set of variables $X = \{x_1, \ldots, x_n\}$, the set of *trees over* Υ *, with variables in* X*,* $T_\Upsilon(X)$*,* can be defined as the set of functions: $t : D \longmapsto \{\Upsilon \bigcup X\}$ such that:

- D is a tree domain (that is, a set of strings over $\mathbb{N}$ s.t., for each $u \in D$ every prefix of u is also in D, and if $ui \in D, i \in \mathbb{N}$, then $uj \in D$, for all $j \in \{1, \ldots, i-1\}$)
- For every $u \in D$, if $n = |\{i \in \mathbb{N} | ui \in D\}|$, then $n = r(t(u))$, where $r : \Upsilon \cup X \longmapsto \mathbb{N}$ is a rank function.

That is, $T_\Upsilon(X)$, is a set of trees which nodes are addressed by strings of natural numbers (in D) and labelled with elements of $\Upsilon \cup X$. Each node has a certain natural number assigned, its rank, which is actually its number of direct descendants. The elements of a tree with no descendants are called *leaves* and the element corresponding to the empty string is called the *root* of the tree (and is, of course, unique).

A composition law of trees can be easily defined. For the case $t, s \in T_\Upsilon(\{x\})$ is carried out as a substitution of each node of t labelled by a variable x by the tree s. We denote the composition by ts. For the general definition in the case $|X| > 1$ see [8].

In this setting, a *tree replacement system* $(\mathcal{S}, \longrightarrow)$, is a subset of $T_\Upsilon(X) \times T_\Upsilon(X)$, together with a relation $\longrightarrow$ defined by:

$$t \longrightarrow s \Longleftrightarrow t \longleftrightarrow s \text{ and } |t| > |s|$$

where $t \longleftrightarrow s$ if and only if t and s can be obtained from the same tree $T \in T_\Upsilon(X)$ by substituting one of its subtrees by two trees $\bar{s}$ and $\bar{t}$ s.t., either

1. $(\bar{s}, \bar{t}) \in \mathcal{S}$

or

2. $\bar{s}$ and $\bar{t}$ are constructed by choosing a function $h : X \to T_\Upsilon(X)$ and then substituting each variable $x \in X$ of two trees $\hat{s}, \hat{t}$ (with $(\hat{s}, \hat{t}) \in S$) by the tree $h(x)$.

Here $|t|$ denotes the number of nodes of t. We say that a tree t is *irreducible* with respect to S if there exists no tree s s.t. $t \longrightarrow s$.
Essentially, a tree replacement system $\mathcal{S}$ defines a congruence relation on the set $T_\Upsilon(X)$, which identifies trees constructed by inserting at some point 'branches' that are equivalent with respect to $\mathcal{S}$. Using the function h, we can insert this equivalent branches in the 'middle' of a tree. However, it is easy to see that, when $\mathcal{S} \subset T_\Upsilon \times T_\Upsilon$, this cannot be done and equivalent trees are constructed only by subtitutions of type 1 as above.

A tree replacement system is called *Church-Rosser* if,

$$\forall\, t_1, t_2 \in T_\Upsilon(X) \text{with } t_1 \longleftrightarrow^* t_2 \text{ there is } t_3 \in T_\Upsilon(X)$$

$$\text{such that } t_1 \longrightarrow^* t_3 \text{ and } t_2 \longrightarrow^* t_3.$$

Here, $\longleftrightarrow^*$ and $\longrightarrow^*$ denote, respectively, the equivalence relation defined as the reflexive transitive closure of $\longleftrightarrow$ and $\longrightarrow$.
As it is proved in [2], the word problem for any Church Rosser tree replacement system is solvable in polynomial time. On the other hand, the problem of deciding whether two trees are congruent w.r.t. an arbitrary tree replacement system can be computationally very hard.

With these ingredients, the scheme of Samuel et al. can be described as follows:

Let Δ, Σ be two finite alphabets, such that the cardinal of Δ, $|\Delta|$, is significantly greater than that of Σ, $|\Sigma|$.
Consider a Church-Rosser tree replacement system $(S, \longrightarrow)$ over Σ, and n trees $t_1, \ldots, t_n$ which are irreducible and not congruent with respect to S.
Let $g : \Delta \longrightarrow \Sigma \cup \{\lambda\}$ be a mapping, which we may extend to map trees in $T_\Delta(X)$ to trees in $T_\Sigma(X)$. Let $(\bar{S}, \longrightarrow)$ be a tree replacement system over Δ fulfilling the following requirements:

- $\bar{S} \subseteq \{(s,t) \mid g(s) \longleftrightarrow^* g(t) \text{ w.r.t. } S\}$
- If a node in t is labelled by a symbol mapped by g to λ, (i.e., a *dummy* symbol), all but one of the subtrees rooted by a children of that node consist only of dummy-labelled nodes.

Public Key: The tree replacement system $(\bar{S}, \longrightarrow)$, together with n trees $s_1, \ldots, s_n \in T_\Delta(X)$, such that

$$\forall i = 1, \ldots, n, \quad g(s_i) \text{ is congruent to } t_i.$$

Secret Key: The Church Rosser replacement system $(S, \longrightarrow)$, together with the trees $t_1, \ldots, t_n \in T_\Sigma(X)$ and the mapping g.

Encryption: Given a plaintext $p = i_1 \cdots i_k$, with $i_r \in \{1, \ldots, n\}$, the corresponding ciphertext is a tree $C = s'_{i_1} \cdots s'_{i_k}$, where for each r, s'_{i_r} is a tree congruent to s_{i_r} w.r.t. S. To be sure that we can encrypt every plaintex using this system (and avoid composition problems), the natural assumption is to consider $X = \{x\}$.

Decryption: Given a valid ciphertext c, $g(c)$ is actually a composition $t'_{i_1} \cdots t'_{i_k}$, where each t'_{i_r} is a tree congruent to t_{i_r} w.r.t. $\bar{S}$. As $\bar{S}$ is Church-Rosser, there exists a polynomial time algorithm for computing t_{i_r} from t'_{i_r}, and thus the authors claim that the plaintext p can be recovered in polynomial time.

3 Cryptanalysis of the scheme

3.1 A remark on the uniqueness of decryption

A first question that arises when reading the description of the above scheme is whether the decryption procedure can actually be done uniquely. As a matter of fact, from the given proposal, it is not clear why a given ciphertext should only correspond to a valid plaintext. In other words, given a tree $t \in T_\Sigma(X)$, could it be the case that t is congruent to two different trees constructed composing the elements of $\{t_1, \ldots, t_n\}$?

It is known (see [7]) that in the above setting, if S is Church Rosser, *normal forms* for trees are unique, that is, if given $t \in T_\Sigma(X)$, there exists s, s', irreducible, so that $t \longrightarrow^* s$ and $t \longrightarrow^* s'$, then $s = s'$. However, from this property it is not clear whether given a set of irreducible trees $\{t_1, \ldots, t_n\}$ there may not be two trees

$$t_{i_1} \cdots t_{i_k}, \text{ and } t_{j_1} \cdots t_{j_m}, (i_r, j_s \in \{1, \ldots, n\} \text{ for } r = 1, \ldots, k,\ s = 1, \ldots, m)$$

both congruent to a fixed $t \in T_\Sigma(X)$.

For a trivial example take $\Sigma = \{a\}$, $X = \{x\}$, $S = \emptyset$ and consider $t_1 := a(x)$, $t_2 := a(a(x))$ (note that, trivially, the system S is Church-Rosser). It is clear that these trees are irreducible and non-congruent. However, $t_1 t_1$ is congruent (in fact equal) to t_2.
As a result, we stress the necessity of imposing further conditions on the set $\{t_1, \ldots, t_n\}$ to assure uniqueness of the decryption process (see Corollary 3.3). To avoid this problem with the current design, it suffices to impose encrypting only one element of $\{1, \ldots, n\}$ at a time (with the corresponding lack of efficiency).

3.1.1 The case $\mathcal{S} \subset T_\Upsilon \times T_\Upsilon$

When no variable is involved in a tree replacement system, that is, $\mathcal{S} \subset T_\Upsilon \times T_\Upsilon$, the location of the variables in equivalent trees is in some sense stable:

Proposition 3.1 *If $\mathcal{S} \in T_\Upsilon \times T_\Upsilon$ and $s, t \in T_\Upsilon(X)$ verify $s \longleftrightarrow t$, then s and t have the same variables, in the same addresses and with the same ancestors. Formally,*

If $u = u_1 \cdots u_n \in \mathrm{dom}(t)$ is s.t. $t(u) = x_i \in X$, then $u \in \mathrm{dom}(s)$, $s(u) = x_i$ and $t(u_1 \cdots u_j) = s(u_1 \cdots u_j)$ for every $1 \leq j \leq n$.

Proof: The idea behind is really simple. If $\mathcal{S} \in T_\Upsilon \times T_\Upsilon$, we know that both s and t are obtained from a tree T by substituting a certain address u by $\bar{s} \in T_\Upsilon$ and $\bar{t} \in T_\Upsilon$ (recall that in this case transformations of type 2 are not carried out). Therefore, as in the resulting tree there is no variable among the children of u, u cannot be an ancestor of any variable of t or s.

In conclusion, t and s have the same variables, in the same addresses and with the same ancestors. $\square$

The result above shows that two equivalent trees have the same 'skeleton', consisting of the set of ancestors of the variables. Using this it is not difficult to obtain the following

Corollary 3.2 *If $\mathcal{S} \subseteq T_\Upsilon \times T_\Upsilon$ and $t_1, \ldots, t_n \in T_\Upsilon(X)$, the following are equivalent:*

1. *$t_1, \ldots, t_n$ are irreducible.*
2. *$t = t_1 \cdots t_n$ is irreducible.*

We can now show that the example given in [8] does have uniqueness in the decryption.

Corollary 3.3 *If $\mathcal{S} \in T_\Upsilon \times T_\Upsilon$, $\mathcal{S}$ is a Church-Rosser, $X = \{x\}$ and $t_1, \ldots, t_n \in T_\Upsilon(X)$ are non-congruent irreducible trees such that x appears just one time in each t_i and such that*

$$|t_i| = |t_j|, \quad \textit{for every } i, j \in \{1, \ldots, n\}, \tag{1}$$

then there is uniqueness in the decryption, that is, if $t_{i_1} \cdots t_{i_k}$ is equivalent (by $\mathcal{S}$) to $t_{j_1} \cdots t_{j_m}$, then $k = m$ and $i_r = j_r$ for each $r \in \{1, \ldots, k\}$.

Proof: By Corollary 3.2, both $t_{i_1} \cdots t_{i_k}$ and $t_{j_1} \cdots t_{j_m}$ are irreducible. As they are equivalent and $\mathcal{S}$ is Church-Rosser, we obtain that they are in fact equal.

By the hypothesis, there is just one x in the tree. So, as we know $|t_{i_k}|$, we can go up in the tree from x to obtain the root of t_{i_k}. But $|t_{j_m}| = |t_{i_k}|$, so, doing the same reasoning, we obtain that the root of t_{j_m} is exactly the same. As $t_{i_1} \cdots t_{i_k} = t_{j_1} \cdots t_{j_m}$ we can conclude that $t_{i_k} = t_{j_m}$. We erase this branch from the tree and go on to obtain that $k = m$ and $i_r = j_r$ for each $r \in \{1, \ldots, k\}$. $\square$

3.2 Attacking the scheme

We expose different types of attacks that may be applied to the proposed scheme. To illustrate its effectiveness, we make use of the example described in [8], specified by:

$$\Sigma = \{a, b, c, d, e, f, g\},$$

$$r(e) = 2, r(a) = r(b) = r(c) = 1, r(d) = r(f) = r(g) = 0$$

$$X = \{x\},\ \Delta = \{c_1, \ldots, c_{17}\}$$

$$t_1 = e(a(d), e(x, c(d))), \quad t_2 = e(e(x, c(d)), b(d))$$

$$s_1 = c_5(c_{12}(c_3(c_7), c_2(x, c_4(c_7))), c_9(c_{16})),$$
$$s_2 = c_5(c_2(c_{12}(x, c_4(c_7)), c_6(c_{13})), c_5(c_{16}, c_9))$$

$$S \approx [a(d) \longleftrightarrow a(a(d));\ b(d) \longleftrightarrow e(f, b(d));\ c(d) \longleftrightarrow e(g, b(d))]$$

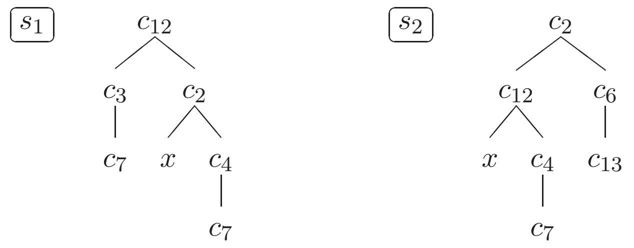
$$\bar{S} \approx \left[\begin{array}{c} c_3(c_7) \longleftrightarrow c_8(c_8(c_7)) \longleftrightarrow c_{11}(c_5(c_3(c_{13}))) \\ \longleftrightarrow c_5(c_3(c_{13})) \longleftrightarrow c_{16}(c_9, c_8(c_8(c_7))); \\ c_6(c_{13}) \longleftrightarrow c_2(c_1, c_{10}(c_7)) \longleftrightarrow c_{12}(c_1, c_5(c_6(c_{13}))); \\ c_4(c_7) \longleftrightarrow c_{12}(c_{17}, c_6(c_{13})) \longleftrightarrow c_5(c_9, c_4(c_7)); \\ c_9 \longleftrightarrow c_5(c_{16}, c_9); c_{19} \longleftrightarrow c_9(c_5) \end{array} \right]$$

$$g(c_1) = f; g(c_{17}) = g; g(c_4) = g(c_{14}) = c; g(c_7) = g(c_{13}) = d;$$
$$g(c_3) = g(c_8) = g(c_{11}) = a; g(c_6) = g(c_{10}) = g(c_{15}) = b;$$
$$g(c_5) = g(c_9) = g(c_{16}) = \lambda; g(c_2) = g(c_{12}) = e.$$

Rank analyzing attack. Note that, if the mapping g is to preserve the tree structure, it should indeed preserve ranks. Thus, knowledge of the ranks in Δ and Σ may reveal valuable information for retrieving g. Rank analyzing can in particular help identifying dummy symbols, reducing subsequently the size of the alphabet Δ.

For instance, in the example above; c_5 appears to have rank two and one (as it appears in the trees $c_5(c_6(c_{13})))$ and $c_5(c_9, c_4(c_7))$ in which it has one and two sons, respectively. Thus, we conclude $g(c_5) = \lambda$. Similarly, just by analyzing the public key one can infer $g(c_{16}) = g(c_9) = \lambda$ (as c_{16} appears as a node with zero and two sons in two different trees, and c_9 as a node with zero and one son).

After erasing these dummy symbols, we reduce the public key to:

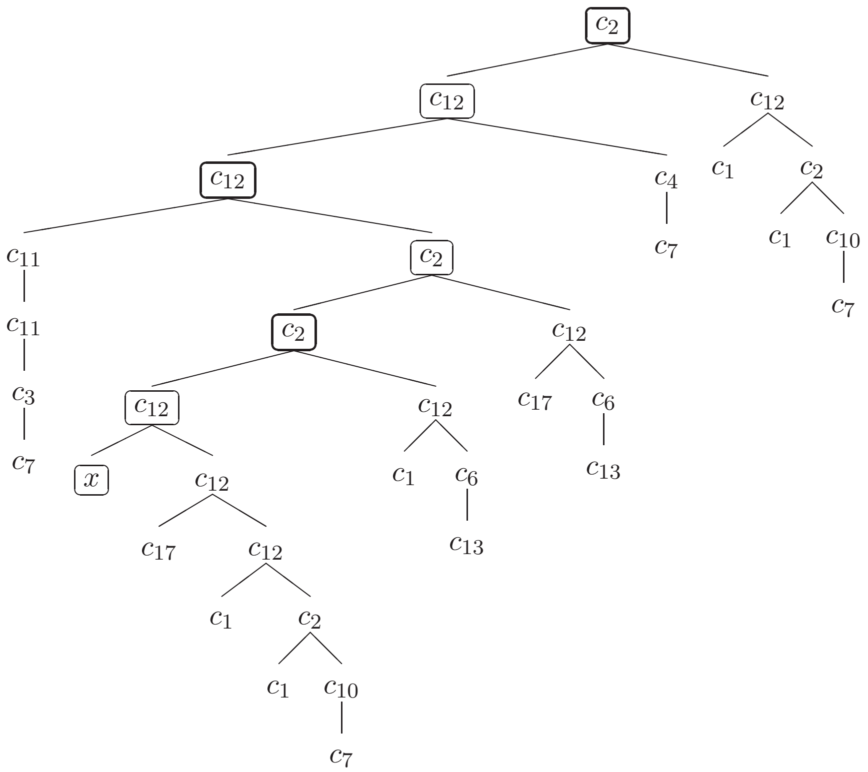


$$\bar{S} \approx \left[\begin{array}{c} c_3(c_7) \longleftrightarrow c_8(c_8(c_7)) \longleftrightarrow c_{11}(c_3(c_{13})) \longleftrightarrow c_3(c_{13}); \\ c_6(c_{13}) \longleftrightarrow c_2(c_1, c_{10}(c_7)) \longleftrightarrow c_{12}(c_1, c_6(c_{13})); \\ c_4(c_7) \longleftrightarrow c_{12}(c_{17}, c_6(c_{13})) \end{array} \right]$$

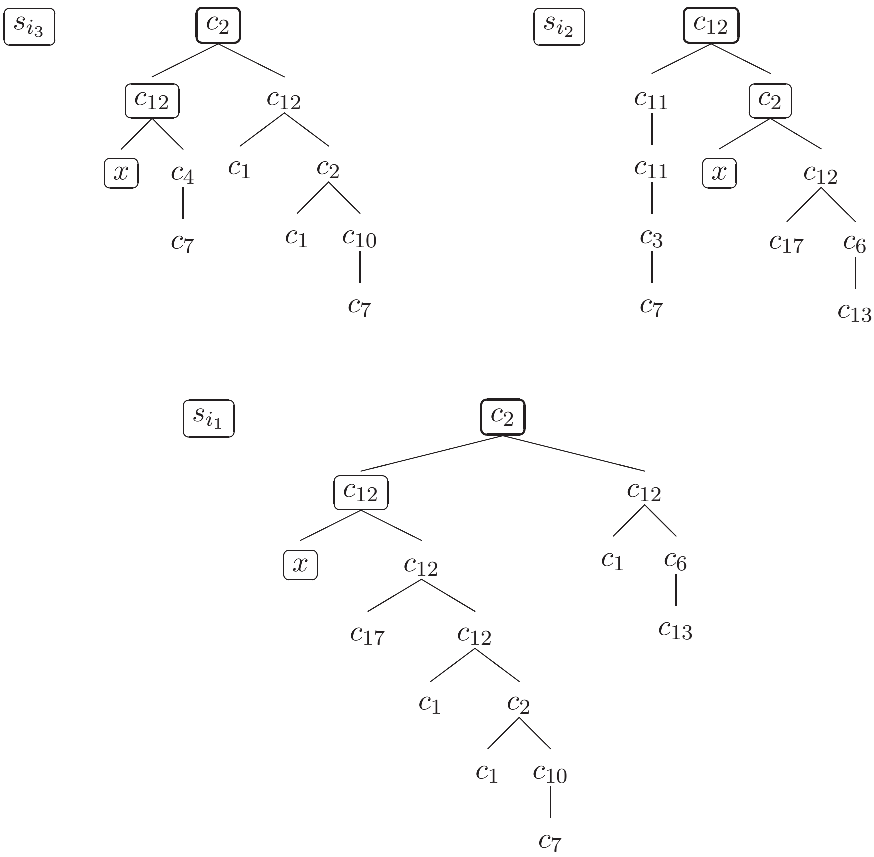
But we can even know more. For example, as $r(c_{12}) = r(c_2) = r(e)$, we know it must be $g(c_{12}), g(c_2) \in \{e, \lambda\}$. But if $r(c_2) = \lambda$, then looking at s_1 one sees that $c_4(c_7)$ must be also equal to λ. But using that, we obtain that c_{12} appears with rank 1 in s_2 and so c_{12} is also equal to λ. But this cannot happen because then, necessarily, $s_1 = s_2 = \{x\}$.
Also, it is clear that g should map the elements of the set $\{c_{13}, c_{19}, c_7, c_1, c_{17}\}$ to elements of rank 0 in Σ or to λ, that is, to elements in $\{d, f, g, \lambda\}$. In a similar way, $g(\{c_3, c_8, c_{11}, c_6, c_{10}, c_4\}) \subseteq \{a, b, c, \lambda\}$.

The case $\mathcal{S} \subset \mathbf{T_\Sigma} \times \mathbf{T_\Sigma}$. By Proposition 3.1, we have a fixed 'skeleton' in the ciphertext from which we can separate the basic trees $s'_{i_1}, \ldots, s'_{i_k}$ to crytanalize them separately.

For example, in [8], they give as ciphertext a tree Q. After erasing the dummy symbols (so we will work with the reduced public key) we obtain the following tree.



As in both s_1 and s_2, x has just two ancestors, we begin in the x of Q and go up to its grandfather. We call s'_{i_1} the subtree generated by it, replace it by an x and apply the same procedure again. Doing this three times we are able to decompose $Q = s'_{i_3}s'_{i_2}s'_{i_1}$ (see below), where (as stated in Proposition 3.1) s'_{i_j} is equivalent to s_{i_j} and the plaintext we are trying to obtain is $i_3i_2i_1$. Now, the father and grandfather of x in s'_{i_j} and s_{i_j} are the same (again by Proposition 3.1). As c_2 is the root of s_2, s'_{i_3}, s'_{i_1} and c_{12} is the root of s_1, s'_{i_2}, we obtain directly that the plaintext has to be 212.



Moreover, there are even more ways to gain information about the plaintext.

Ciphertext-only attacks. It is also worth noticing that if the public key (namely, the rewriting system) is not carefully chosen, a lot of information about the corresponding plaintext may be retrieved just by observing a given ciphertext string. Let us consider the encryption of a one-digit text in the example given in [8] (we consider the reduced form obtained after erasing the dummy symbols). The letter c_3 appears only in s_1 and the only chain of relations in $\bar{S}$ that contains c_3 is the following

$$c_3(c_7) \longleftrightarrow c_8(c_8(c_7)) \longleftrightarrow c_{11}(c_3(c_{13})) \longleftrightarrow c_3(c_{13})$$

Now, if we use this relations we will obtain at least one of the letters c_3, c_8, c_{11}. But neither of them appears in any other relation of $\bar{S}$. Thus, if the ciphertext contains the letters c_8, c_{11} or c_3 it clearly corresponds to the plaintext 1. If not, it will correspond to the plaintext 2.

Reaction attacks. Finally we want to stress that also, if the keys of the above scheme are not chosen carefully, it could also be vulnerable to so-called *Reaction Attacks*, which have been proven useful in similar rewriting-based cryptosystems [5, 6].

Let us now suppose that the attacker has access to a *black box* that allows him to distinguish random trees in $T_\Delta(X)$ from proper ciphertexts. Such a device could indeed prove useful for gaining information about the public key; namely it can be used for identifying dummy symbols, or letters that have the same image by g (see the discussion in [6]).

4 Conclusion

Indeed, there are ways of minimizing the effect of the above mentioned attacks. In order to prevent rank attacks, it would be necessary that knowing the rank of a letter $\delta \in \Delta$ there still were 'many ' possibilities for its image by g. Ideally, most letters in Σ should have the same rank, which in the end means the message spaces should be restricted to homogeneous trees. Preventing ciphertex-only attacks is also a rather subtle task; one should carefully study the distributions of the trees in $T_\Delta(X)$ in all public words as well as in the rewriting rules, so that they are as close to uniform as possible (and thus its appearance on the ciphertext provides no information about the corresponding plaintext). In particular, that implies all letters in Δ should play a similar role (appear in all the trees s_i and in all the rewritting rules). Situations like that of the example in [8], where the letters c_{14}, c_{15} neither appear on the public trees, nor in the rules of $\bar{S}$, should not occur. Finally, in order to avoid the attacks given by Proposition 3.1, one should consider tree replacement systems that involve the variables. The problem is that then it seems to be very involved to prove the necessary uniqueness of the decryption.

Thus, as a summary, we can give no concrete steps towards the design of a key generation process with could guarantee an acceptable security level.

References

[1] P.J. Abisha and D.G. Thomas and K.G. Subramanian, 'Public Key Cryptosystems Based on Free Partially Commutative Monoids and

Groups' *In Proceedings of INDOCRYPT 2003*, Lecture Notes in Computer Science **2904**, 2003, pp. 218–227.

[2] J.H. Gallier and R.V. Book. 'Reductions in tree replacement systems', *Theoretical Computer Science*, **37**, 1985, pp. 123–150.

[3] M.I. González Vasco and R. Steinwandt 'Clouds over a Public Key Cryptosystem Based on Lyndon Words' *Information Processing Letters*, **80**, 2001, pp.239–242.

[4] J-M. Bohli and M.I. González Vasco and C. Martínez and R. Steinwandt 'Weak Keys in MST_1' *Preprint. Available at: Cryptology ePrint Archive: Report 2002/070*, 2002.

[5] M.I. González Vasco and R. Steinwandt. 'A Reaction Attack on a Public Key Cryptosystem Based on the Word Problem', *Applicable Algebra in Engineering, Communication and Computing*, 14: 335–340, 2004.

[6] M.I. González Vasco and R. Steinwandt. 'Pitfalls in public key cryptosystems based on free partially commutative monoids and groups', *Cryptology ePrint Archive: Report 2004/012* .

[7] B.K. Rosen 'Tree-Manipulating Systems and Church-Rosser Theorems', *Journal of the ACM*, **20**, 1, 1973, pp. 160–187.

[8] S.C. Samuel, D.G. Thomas, P.J. Abisha, and K.G. Subramanian 'Tree Replacement and Public Key Cryptosystem', *Proc. of INDOCRYPT 2002, LNCS* , **2551**, 2002, pp. 71–78.

[9] K.G. Subramanian, R. Siromoney and P.J. Abisha 'A DOL-TOL Public Key Cryptosystem', *Information Processing Letters*, **26**, 1987, pp. 95–97.

[10] R. Siromoney and L. Mathew 'A Public Key Cryptosystem Based on Lyndon Words', *Information Processing Letters*, **35**, 1990, pp. 33–36.

[11] N.R. Wagner and M.R. Magyarik 'A Public Key Cryptosystem Based on the Word Problem' *Advances in Cryptology: Proceedings of CRYPTO 84, LNCS*, **196**, 1985, pp. 19–36.

[12] S.S. Magliveras and D.R. Stinson and T. Trung 'New approaches to designing public key cryptosystems using one-way functions and trap-doors in finite groups' *Journal of Cryptology*, **15**, 2002, pp. 285–297.

[13] M. Garzon and Y. Zalcstein 'The Complexity of Grigorchuk groups with application to cryptography' *Theoretical Computer Science*, **88**, 1991, pp. 83–98.