On Oleshchuk's Public Key Cryptosystem

Heiko Stamer **Friedrich Otto**
Universität Kassel, Fachbereich Mathematik/Informatik
Heinrich-Plett-Straße 40, D-34132 Kassel, Germany
{stamer,otto}@theory.informatik.uni-kassel.de

September 1, 2004

Abstract

This paper revisits a public key cryptosystem which is based on finite Church-Rosser string-rewriting systems. We consider some ideas for cryptanalysis and discuss issues concerning practical usage. It turns out that without changing crucial details of key generation this cryptosystem does not offer acceptable cryptographic security. We also provide the source code of our rudimentary implementation, if someone would like to use it for further cryptanalysis.

1 Introduction

The security of almost every public key cryptosystem relies on the intractability of only a few number-theoretic problems, e.g., factoring large integers or computing discrete logarithms in finite groups. Unfortunately, no strict proof of hardness (from a complexity theoretic point of view) is known for these assumptions. Therefore it sounds reasonable to look for other possible trapdoor functions in different areas of theoretical computer science. Further there is hope that such proposals [10, 11, 12, 9] will provide some kind of 'provable security', because their underlying questions (e.g. the word problem for finitely presented groups) are undecidable in general. Beside other difficulties a primary problem in the design of secure cryptosystems remains: The gap between the average and the worst case hardness of instances, and hence the possibility of weak keys.

Vladimir A. Oleshchuk [1] proposed a public key cryptosystem that relies on the undecidability of the word problem in semigroups. A basic ingredient of his approach are string-rewriting systems. Each system is represented by a rule set containing ordered pairs of strings over a finite alphabet. Bidirectional rewriting on a string is performed through replacing (non-deterministically chosen) occurrences of the left-hand side by the right-hand side of a rule or vice versa. This operation induces an equivalence relation and we say that two strings are congruent, if they can be rewritten to each other in finitely many bidirectional steps. The word problem is the question of whether two given strings are congruent modulo a given rewriting system. This problem is undecidable in general, i.e. there exists no algorithm which terminates for all instances with the right answer. However, if the rule set is finite and the string rewriting system has the Church-Rosser property, then the word problem can be solved in linear time. This fact has been used recently by Oleshchuk to construct a trapdoor function and Church-Rosser codes [1, 2].

2 Preliminaries

Let Σ be a finite alphabet. Σ^* denotes the set of all strings over this alphabet including the empty word ϵ. The concatenation of two strings x and y is simply written as xy. Further, $|x|$ denotes the *length* of string x, where $|\epsilon| = 0$, $|a| = 1$ for $a \in \Sigma$, and $|xa| = |x| + 1$ for $x \in \Sigma^*, a \in \Sigma$. If $A, B \subseteq \Sigma^*$, then the product of these languages is defined to be $AB = \{xy \mid x \in A, y \in B\}$.

The *string-rewriting system* R on Σ is a subset of $\Sigma^* \times \Sigma^*$. An element $(u, v) \in R$ is called *rewrite rule.* The word $u \in \Sigma^*$ is the left-hand side (LHS) and word $v \in \Sigma^*$ the right-hand side (RHS) of such a rule. Here we will only be dealing with finite string-rewriting systems, i.e. R is finite. Each string-rewriting system R induces a *reduction relation* $\rightarrow_R^*$ on Σ^*, which is the reflexive transitive closure of the single-step reduction relation $\rightarrow_R = \{(xuy, xvy) \mid x, y \in \Sigma^* \text{ and } (u, v) \in R\}$. If $u \rightarrow_R^* v$, then u is an *ancestor* of v, and v is a *descendant* of u. If there is no $v \in \Sigma^*$ such that $u \rightarrow_R v$ holds, then the string u is called *irreducible modulo* R. We denote the set of all irreducible words modulo R by $\mathrm{IRR}(R)$. For finite string-rewriting systems this is a regular language, i.e. a finite-state acceptor recognizing $\mathrm{IRR}(R)$ can be effectively constructed from the rules of R. The reflexive, symmetric, and transitive closure of $\rightarrow_R$ is the *Thue congruence* $\leftrightarrow_R^*$. We define the *congruence class* of a string $u \in \Sigma^*$ as $[u]_R = \{v \in \Sigma^* \mid v \leftrightarrow_R^* u\}$. This notation is extendable to $A \subseteq \Sigma^*$ by $[A]_R = \{v \in \Sigma^* \mid \exists u \in A : v \leftrightarrow_R^* u\}$.

A string-rewriting system R is called

- *noetherian* if there exists no infinite sequence of reductions,
- *confluent* if, for all $u, v, w \in \Sigma^*$, $u \rightarrow_R^* v$ and $u \rightarrow_R^* w$ imply that v and w have a common descendant (i.e. $\exists z \in \Sigma^* : v \rightarrow_R^* z$ and $w \rightarrow_R^* z$),
- *convergent* if it is noetherian and confluent,
- *length-reducing* if $|u| > |v|$ holds for each rule $(u, v) \in R$,
- *normalized* if $u \in \mathrm{IRR}(R \setminus \{(u, v)\})$ and $v \in \mathrm{IRR}(R)$ for each rule $(u, v) \in R$.

A string-rewriting system R is *Church-Rosser* (i.e. has the Church-Rosser property) if, for all $x, y \in \Sigma^*$ with $x \leftrightarrow_R^* y$, there exists a word $z \in \Sigma^*$ such that $x \rightarrow_R^* z$ and $y \rightarrow_R^* z$. Hence, it is Church-Rosser if and only if it is confluent. For finite length-reducing systems this property is decidable in polynomial time [7]. If the string-rewriting system R is Church-Rosser, then each congruence class has a unique irreducible element (modulo R) and the corresponding word problem[1] can be solved in linear time [4].

Let R_1 and R_2 be string-rewriting systems on Σ. R_1 *refines* R_2, if for all $x, y \in \Sigma^*$ the congruence $x \leftrightarrow_{R_1}^* y$ implies $x \leftrightarrow_{R_2}^* y$. If R_1 refines R_2 and R_2 refines R_1, then they generate the same Thue congruence and are called *equivalent*. R_1 refines R_2, if and only if the congruence $u \leftrightarrow_{R_2}^* v$ holds for each rewrite rule $(u, v) \in R_1$.

A Language $L \subseteq \Sigma^*$ is a *Church-Rosser Congruential Language* [8] if there exists a finite, length-reducing, and confluent string-rewriting system R on Σ and finitely many strings $w_1, \ldots, w_n \in \mathrm{IRR}(R)$ such that

$$L = \bigcup_{i=1}^{n} [w_i]_R.$$

A nonempty set $C \subseteq \Sigma^*$ is called a *code*, if for all words $u_{i_1}, \ldots, u_{i_n} \in C$, $u_{j_1}, \ldots, u_{j_m} \in C$, the equality of $u_{i_1} \cdots u_{i_n} = u_{j_1} \cdots u_{j_m}$ implies $u_{i_1} = u_{j_1}$. By induction we get $n = m$ and $u_{i_k} = u_{j_k}$ for all $1 \leq k \leq n$. If C is a code then any word from C^* has a unique factorization over C.

3 Oleshchuk's Public Key Cryptosystem

First we briefly repeat the original definition [1]. Subsection 3.1 appends a necessary requirement for unique decryption, which was established later in Oleshchuk's second paper [2] on this topic.

Let Σ be the plaintext alphabet of possible messages $\mathcal{M} = \{w \mid w \in \Sigma^*\}$. Without loss of generality, we consider only the binary case $\Sigma = \{x_0, x_1\}$. The ciphertext alphabet Δ is supposed to be bigger than Σ, i.e. $|\Delta| > |\Sigma|$.

Key Generation Let T be a finite Church-Rosser string-rewriting system on Δ. We choose $u_1, u_2, \ldots, u_t \in \mathrm{IRR}(T)$ such that, for all $i, j = 1, \ldots, t$, the word $u_i u_j$ is irreducible modulo T and the set $\{u_1, u_2, \ldots, u_t\}$ is a code. Now let $R_0, R_1 \subset \{u_1, u_2, \ldots, u_t\}$ be two nonempty disjoint sets, i.e. $R_0 \cap R_1 = \emptyset$. Further let $L_0 \subseteq [R_0]_T$ and $L_1 \subseteq [R_1]_T$ be two regular languages which may be constructed effectively by applying reverse rules of T arbitrarily and non-deterministically to R_0 resp. R_1. Note that also $L_0 \cap L_1 = \emptyset$ because T is confluent and R_0, R_1 are disjoint. The next step of key generation picks a finite string-rewriting system S on Δ such that $u \leftrightarrow_T^+ v$, for all $(u, v) \in S$, i.e.

[1] Instance: $x, y \in \Sigma^*$, Question: Are x and y congruent modulo R? (i.e. $x \leftrightarrow_R^* y$)

S refines T. This property can be tested easily because T is Church-Rosser and thus the corresponding word problem is decidable in linear time [4].

The string-rewriting system S and the languages L_0, L_1 form the public key. The finite Church-Rosser system T and R_0, R_1 should be kept secret, because they represent the private part of the key.

Encryption The encryption of a letter x_i is a random word $y_i \in [L_i]_S$. Therefore the non-deterministic encryption function $\mathsf{Enc} : \mathcal{M} \to \mathcal{C}$ maps a possible message $m = x_{i_1}x_{i_2}\ldots x_{i_n}$, where $x_{i_k} \in \Sigma,\ k = 1,\ldots,n$, to a random ciphertext $c \in [L_{i_1}L_{i_2}\ldots L_{i_n}]_S$. In practice we will do as follows:

1. Encode the plaintext $m = x_{i_1}x_{i_2}\ldots x_{i_n}$ into $\hat{m} = \hat{x}_{i_1}\hat{x}_{i_2}\ldots\hat{x}_{i_n}$, where each string $\hat{x}_{i_k}$ is randomly chosen from the corresponding set L_{i_k}.

2. Rewrite $\hat{m}$ arbitrarily into c according to the rules of S.

Decryption For decryption of a secret message $c \in \mathcal{C}$ we have to find a word $\hat{m} \in \{L_0 \cup L_1\}^*$ such that $c \leftrightarrow_S^* \hat{m}$. In general the finite string-rewriting system S may have an undecidable word problem [4] and even decidability does not guarantee that it is computationally feasible [6].

With the secret trapdoor (T, R_0, R_1) decryption becomes easy, because T has the Church-Rosser property and thus there exists a uniquely defined word $\tilde{m} \in \mathrm{IRR}(T)$ such that $c \to_T^* \tilde{m}$. This irreducible string can be found in linear time [4] and its factorization $\tilde{m} = u_{i_1}u_{i_2}\ldots u_{i_n}$ (where $u_{i_k} \in R_{i_k}$) reveals obviously the plaintext $m = x_{i_1}x_{i_2}\ldots x_{i_n}$ of the message.

3.1 Necessary requirement for unique decryption

It was observed [2] that the condition $u_iu_j \in \mathrm{IRR}(T)$ for all $i, j = 1,\ldots,t$ is not sufficient for unique decoding. Consider the following simple counterexample within the original definition of key generation.

Example 1 (Non-uniqueness of decryption)

$$T = \{(acb, b), (cab, c)\},\ R_0 = \{b\},\ R_1 = \{a, c\}$$

Of course, T is a finite, length-reducing and confluent string-rewriting system on $\Delta = \{a, b, c\}$. Hence it has the desired Church-Rosser property. Further the strings $b \in R_0$, $a, c \in R_1$ and their possible concatenations aa, ab, ac, ba, $\ldots$, cc are irreducible modulo T. Public languages $L_i \subseteq [R_i]_T$ and the string-rewriting system S (which refines T) are arbitrary.

Nevertheless, the intermediate encoding $\hat{m}_1 = acabacbcab$ of a message $m_1 = 1101$ leads to a decryption failure, because one possible encoding of the second message $m_2 = 01$ belongs to the same congruence class modulo T, i.e. $\hat{m}_1 \to_T^* acbc = \hat{m}_2 \to_T bc = \tilde{m}_1 = \tilde{m}_2 \in [R_0R_1]_T$.

Therefore we have to ensure (during the process of key generation) that all words in $(R_0 \cup R_1)^*$ are irreducible modulo T. This reformulated condition $(R_0 \cup R_1)^* \subseteq \mathrm{IRR}(T)$ can be effectively tested, since both sides are regular languages and thus the inclusion property is decidable in polynomial time.

Let $\ell_T = \max\{|u_1|, \ldots, |u_n|\}$ for all rewrite rules $(u_i, v_i) \in T$ and let $\ell_R = \max\{|u_1|, \ldots, |u_m|\}$ for all words $u_i \in (R_0 \cup R_1)$. Then it is sufficient to check, whether the inclusion $(R_0 \cup R_1)^{\leq 2\max\{\ell_T, \ell_R\}} \subseteq \mathrm{IRR}(T)$ holds. This can easily be done by generating all concatenations of length lower or equal than $2 \cdot \max\{\ell_T, \ell_R\}$ and verifying irreducibility for each of them.

4 Cryptanalysis

Like other cryptosystems based on rewriting techniques [14, 15] this approach is vulnerable to particular cryptanalytic attacks, if weak keys (here string-rewriting systems) are chosen during key generation.

4.1 Completion of string-rewriting system S

Oleshchuk noticed [1] that it is not necessary to find the exact secret string-rewriting system T generated by the owner. Any Church-Rosser system T' where all of the conditions

1. S refines T'
2. $[L_0]_{T'} \cap [L_1]_{T'} = \emptyset$
3. $([L_0]_{T'} \cup [L_1]_{T'}) \cap \mathrm{IRR}(T')$ is a code
4. $([L_0]_{T'} \cup [L_1]_{T'})^* \subseteq \mathrm{IRR}(T')$ (reformulated according to [2])

apply, can be used to decrypt messages. In general, there is no algorithm to decide whether a finite string-rewriting system S is equivalent to any finite Church-Rosser system T' [5]. But a cryptanalyst can try techniques known as *completion procedures* to construct such a convergent system T', and with some luck he will succeed.

Let $\geq$ be a partial ordering on Δ^*. This ordering is called *admissible* if $u \geq v$ implies that $xuy \geq xvy$ holds for all $x, y \in \Delta^*$, and it is called *well-founded* if there is no infinite strictly descending sequence $u_0 > u_1 > \cdots > u_i > u_{i+1} > \cdots$. A string-rewriting system S on Δ is *compatible* with an ordering $\geq$ if $u > v$ holds for each rule $(u, v) \in S$.

The Knuth-Bendix [16] completion procedure takes as input a finite string-rewriting system S on Δ and an admissible well-founded partial ordering $\geq$ on Δ^*. Based on this ordering the system S is turned into an equivalent system T' that is compatible with $\geq$ by orienting each rule with respect to this ordering. Thus, T' will be noetherian. Then the critical pairs of T' are computed, and for each critical pair that does not resolve a new rule is introduced. Unfortunately, each new rule can lead to new unresolvable critical pairs, and hence, this process may not terminate. A detailed description of the Knuth-Bendix completion is omitted here, due to lack of space. Interested readers are referred to the existing literature [3, 16].

Example 2 (Knuth-Bendix completion of S)

$$T = \{(bb, b), (ca, c), (bc, c)\}, \; S = \{(ab, abb), (abb, ab), (bbc, bc)\}$$

A malicious cryptanalyst can easily obtain the convergent string-rewriting system $T' = \{(abb, ab), (bbc, bc)\}$ by using Knuth-Bendix completion with the length-lexicographical ordering. Obviously, S refines T' because all rewrite rules $(u, v) \in S$ satisfy $u \leftrightarrow_{T'} v$. Moreover S and T' are equivalent.

Now assume that the public regular languages L_0 and L_1 are finite. Then based on the above observation we can establish the following result:

Theorem 1 *If, on input of S (finite string-rewriting system) and $\geq$ (admissible well-founded partial ordering), the Knuth-Bendix completion procedure terminates with a convergent string-rewriting system T', then a passive adversary can retrieve the plaintext of an encrypted message with non negligible probability in linear time.*

Proof. By using Knuth-Bendix completion one obtains a convergent string-rewriting system T' which is equivalent to S, i.e. $\leftrightarrow_S^* = \leftrightarrow_{T'}^*$. With it the cryptanalyst can reduce an arbitrary ciphertext $c \in \mathcal{C}$ to a normal form $\tilde{m}' \in \mathrm{IRR}(T')$ in linear time [4], i.e. $c \rightarrow_{T'}^* \tilde{m}'$. Further, assuming the L_i's are finite, he will get the finite sets $R_i' = \{u \in \mathrm{IRR}(T') \mid \exists \hat{x}_i \in L_i : \hat{x}_i \rightarrow_{T'}^* u\}$ by reducing all words of L_i to their unique normal form modulo T'.

First, consider the restricted case of a one-bit message ($n = 1$). We will show that $c \in [L_i]_S$ if and only if $\tilde{m}' \in R_i'$:

- $\Rightarrow$ For each $c \in [L_i]_S$ there exists an encoding string $\hat{x}_i \in L_i$ such that $c \leftrightarrow_S^* \hat{x}_i$ holds. Since S and T' are equivalent, the reduction $c \rightarrow_{T'}^* \tilde{m}'$ implies $\tilde{m}' \leftrightarrow_{T'}^* \hat{x}_i$. Further, T' is Church-Rosser and thus each $\hat{x}_i \in L_i$ has a unique normal form r_i' modulo T'. By construction $r_i' \in R_i'$. Finally $r_i' = \tilde{m}'$ and hence $\tilde{m}' \in R_i'$, because normal forms are unique.

$$\begin{tikzcd}
c \arrow[rrrr, leftrightarrow, "*", "{S,T'}"'] \arrow[dr, "*", "T'"'] \arrow[dd, "T", "*"'] & & & & \hat{x}_i \arrow[dl, "*"', "T'"] \arrow[dd, "*"', "T"] \\
& \tilde{m}' \arrow[dl, "*"', "T"] & = & r_i' \arrow[dr, "*", "T"'] & \\
\tilde{m} & & = & & r_i
\end{tikzcd}$$

$\Leftarrow$ The other direction follows by similar arguments.

Now we turn to the case of longer messages. Here we get the problem that not necessarily $(R'_0 \cup R'_1)^* \subseteq \mathrm{IRR}(T')$ holds and thus the decoding may be ambiguous. Let $\bar{L}'_0 \subseteq L_0$ resp. $\bar{L}'_1 \subseteq L_1$ be the set of all code strings $\hat{x}_{i_j} \in L_{i_j}$ used during the encryption of a fixed ciphertext c, i.e. $c \in [\bar{L}'_{i_1}\bar{L}'_{i_2}\ldots\bar{L}'_{i_n}]_S$. Further let $\bar{R}'_i = \{u \in \mathrm{IRR}(T') \mid \exists \hat{x}_i \in \bar{L}'_i : \hat{x}_i \to^*_{T'} u\}$ be the corresponding sets of normal forms modulo T'. Obviously, if

(i) $[\bar{L}'_0]_{T'} \cap [\bar{L}'_1]_{T'} = \emptyset$,

(ii) $([\bar{L}'_0]_{T'} \cup [\bar{L}'_1]_{T'}) \cap \mathrm{IRR}(T')$ is a code, and

(iii) $([\bar{L}'_0]_{T'} \cup [\bar{L}'_1]_{T'})^* \subseteq \mathrm{IRR}(T')$,

then $(\bar{R}'_0 \cup \bar{R}'_1)$ is a code and the inclusion $(\bar{R}'_0 \cup \bar{R}'_1)^* \subseteq \mathrm{IRR}(T')$ holds. Hence $c \in [\bar{L}'_{i_1}\bar{L}'_{i_2}\ldots\bar{L}'_{i_n}]_S$ if and only if $\tilde{m}' \in \bar{R}'_{i_1}\bar{R}'_{i_2}\ldots\bar{R}'_{i_n}$. As $(\bar{R}'_0 \cup \bar{R}'_1)$ is a code the corresponding factorization of $\tilde{m}'$ can be determined easily.

The conditions (i), (ii), and (iii) are often satisfied for short messages or sparse sets $\bar{L}'_0, \bar{L}'_1$. Further, if one of these conditions does not hold, a passive adversary may probably still be able to retrieve some partial information about the corresponding plaintext by looking at $\tilde{m}'$ and R'_0 resp. R'_1. $\square$

Example 3 (Knuth-Bendix completion attack)

$$T = \{(cb, c), (aa, a), (ab, a)\},\ S = \{(ab, aab), (cba, ca), (baa, ba)\}$$

$$R_0 = \{cacac\},\ R_1 = \{aca\},\ L_0 = \{caacbabc\},\ L_1 = \{abacbaab\}$$

On input of S and $\leq_{\mathrm{llex}}$ (length-lexicographical ordering) the Knuth-Bendix completion procedure terminates with the Church-Rosser system

$$T' = \{(aab, ab), (cba, ca), (baa, ba), (caa, ca)\}.$$

By reducing L_0, L_1 modulo T' we get $R'_0 = \{cacabc\}$ and $R'_1 = \{abacab\}$.

$$caacbabc \overset{(4)}{\to}_{T'} cacbabc \overset{(2)}{\to}_{T'} cacabc$$
$$abacbaab \overset{(2)}{\to}_{T'} abacaab \overset{(4)}{\to}_{T'} abacab$$

Now suppose that the cryptanalyst observes the ciphertext $c = cbaacabc$ which can be reduced in two steps to $\tilde{m}' = cacabc \in [R'_0]_{T'}$.

$$c = \quad cbaacabc \overset{(2)}{\to}_{T'} caacabc \overset{(4)}{\to}_{T'} cacabc \quad = \tilde{m}' \in [R'_0]_{T'}$$

Thus the corresponding plaintext is the single letter x_0.

As a consequence of this attack we have to ensure that the string-rewriting system S cannot easily be completed. Since such a property depends on the ordering used, this seems to be a hard task.

4.2 Guessing T by prefix and suffix properties of S

Further properties of weak keys can be exploited: A pitfall stems from the fact that S refines T. Of course, if $S = \{(xuy, xvy) \mid (u,v) \in T,\, x,y \in \Delta^*\}$ the congruence $xuy \leftrightarrow_T^* xvy$ holds for each rewrite rule of S.

Example 4 (Guessing T by common prefix)

$$T = \{(ba,b),(ab,b),(aa,a)\},\;\; S = \{(baa,ba),(bb,bab),(aba,ab)\}$$

A cryptanalyst can guess the Church-Rosser system T, if S was improper chosen, i.e. for some $(u,v) \in T$ a prefix z exists such that $(zu,zv) \in S$ or $(zv,zu) \in S$. In our example this is the case for all rewrite rules of T: $(\cancel{b}aa,\, \cancel{b}a)$ leads to (aa,a), $(\cancel{b}b,\, \cancel{b}ab)$ to (ab,b), and $(\cancel{a}ba,\, \cancel{a}b)$ to (ba,b).

4.3 Ciphertext-only attack

If the string-rewriting system S and the sets L_0, L_1 are not carefully chosen, then (analogously to [15]) information about the corresponding plaintext $m \in \mathcal{M}$ may be retrieved just by observing a given ciphertext $c \in \mathcal{C}$.

That might be the case if a letter of the ciphertext alphabet Δ appears only in words either from L_0 or from L_1 and this relation is preserved by the rules of S applied during encryption. A cryptanalyst can count the occurrences of such a letter in c and thus obtain information about the minimum number of x_0's or x_1's in the plaintext m. This attack can be generalized to other measures, e.g. if S preserves some unique subword or the characteristic lengths of strings either from L_0 or from L_1.

4.4 Structure of cryptogram space $\mathcal{C}$

Obviously, for finite message space $\mathcal{M}$ and Church-Rosser system T the set of all cryptograms is a *Church-Rosser Congruential Language* (CRCL) [8], because each ciphertext $c \in \mathcal{C}$ belongs to a congruence class $[R_{i_1} R_{i_2} \dots R_{i_n}]_T$ represented by the corresponding plaintext $x_{i_1} x_{i_2} \dots x_{i_n} = m \in \mathcal{M}$.

5 Practical issues

This section describes some questions that arose during our implementation of Oleshchuk's cryptosystem. The programming was done as *proof of concept* in approximately 1 200 lines of C++ code. Thus features and documentation are very limited: First the program constructs a random key pair ($K_{\text{pub}} = (S, L_0, L_1), K_{\text{sec}} = (T, R_0, R_1)$) according to the described procedure of key generation. Then some simple encryption/ decryption operations on one-bit and longer messages are performed. Finally, if possible, the completion attack (see section 4.1) is mounted. We provide the source code [18] under the *GNU General Public License*, if the reader would like to investigate or use parts of our work for further cryptanalysis.

Choosing "cryptographically good" parameter settings This seems to be a serious question since many possible parameters may have influence on the security of the entire cryptosystem, e.g. the size of the ciphertext alphabet Δ, the sizes of the string-rewriting systems T and S, the sizes of the sets R_i, and, if finite sets L_i are used, the number of applied rules during their generation. Concerning the first measure we can find that in the unary case ($|\Delta| = 1$) the word problem becomes decidable for finite string-rewriting systems [3].

On the other hand, if we consider a bounded number of rewrite rules over an arbitrary finite alphabet, then there exists a string-rewriting system with only three rules and undecidable word problem [17]. It is an open question of whether or not this problem becomes decidable if we consider only a one-rule rewriting system. Hence S should have at least three rules.

Generating a string-rewriting system S that refines T Up to now we don't have any other method than to randomly guess S and check if $u \leftrightarrow_T^+ v$ holds for all rules $(u, v) \in S$. Each obvious strategy to perform this in a more efficient way will probably introduce new vulnerabilities.

Encrypting messages Here we have to ensure that a cryptanalyst cannot handle the word problem by a brute-force search in the Thue congruence. Therefore the number of nodes in the derivation tree of $c \rightarrow_S^* \tilde{m}$ should grow exponentially in the number of performed rewrite steps during encryption.

6 Conclusion

Our contribution shows that without changing crucial details of key generation Oleshchuk's cryptosystem is vulnerable to the described attacks and thus does not offer acceptable cryptographic security. It is a open question whether this system can be repaired to withstand the proposed attacks.

Acknowledgement The authors want to thank Tomasz Jurdziński, Hartmut Messerschmidt, and Andreas Conz for fruitful discussions regarding cryptanalysis and implementation of Oleshchuk's cryptosystem.

References

[1] Vladimir A. Oleshchuk: *On Public-Key Cryptosystem Based on Church-Rosser String-Rewriting Systems*, Proceedings of COCOON'95, Lecture Notes in Computer Science **959**, pp. 264–269, 1995

[2] Vladimir A. Oleshchuk: *Church-Rosser Codes*, Proceedings of 5^{th} IMA Conference, Lecture Notes in Computer Science **1025**, pp. 199 – 204, 1996

[3] Ronald V. Book, Friedrich Otto: *String-Rewriting Systems*, Texts and Monographs in Computer Science, Springer, New-York, 1993

[4] Ronald V. Book: *Confluent and other types of Thue systems*, Journal of the ACM **29**, pp. 171–183, 1982

[5] Colm O'Dunlaing: *Undecidable questions related to Church-Rosser Thue systems*, Theoretical Computer Science **23**, pp. 339–345, 1983

[6] Günther Bauer, Friedrich Otto: *Finite Complete Rewriting Systems and the Complexity of the Word Problem*, Acta Informatica **21**, pp. 521–540, 1984

[7] Deepak Kapur, Mukkai S. Krishnamoorthy, Robert McNaughton, Paliath Narendran: *An $O(|T|^3)$ algorithm for testing the Church-Rosser property of Thue systems*, Theoretical Computer Science **35**, pp. 109–114, 1985

[8] Robert McNaughton, Paliath Narendran, Friedrich Otto: *Church-Rosser Thue systems and formal languages*, Journal of the ACM **35**, pp. 324–344, 1988

[9] Valtteri Niemi: *Cryptology: Language-Theoretic Aspects*, G. Rozenberg, A. Salomaa (eds.): Handbook of Formal Languages, Springer, Berlin, 1997

[10] Neal R. Wagner, Marianne R. Magyarik: *A Public-Key Cryptosystem Based on the Word Problem*, Advances in Cryptology: Proceedings of CRYPTO'84, Lecture Notes in Computer Science **196**, pp. 19–36, 1985

[11] Rani Siromoney, Lisa Mathew: *A Public Key Cryptosystem Based on Lyndon Words*, Information Processing Letters **35**, pp. 33–36, 1990

[12] Akihiro Yamamura: *Public-Key Cryptosystems Using the Modular Group*, 1^{st} International Workshop on Practice and Theory in Public Key Cryptography (PKC'98), Lecture Notes in Computer Science **1431**, pp. 203–216, 1998

[13] S.C. Samuel, D.G. Thomas, P.J. Abisha, K.G. Subramanian: *Tree Replacement and Public Key Cryptosystem*, Progress in Cryptology — INDOCRYPT 2002: Third International Conference on Cryptology in India, Lecture Notes in Computer Science **2551**, pp. 71–78, 2003

[14] Maria I.G. Vasco, Rainer Steinwandt: *Pitfalls in public key cryptosystems based on free partially commutative monoids and groups*, Cryptology ePrint Archive: Report 2004/012, 2004

[15] David P. Garcia, Maria G. Vasco: *Attacking a Public Key Cryptosystem Based on Tree Replacement*, Cryptology ePrint Archive: Report 2004/098, 2004

[16] Donald E. Knuth, Peter B. Bendix: *Simple word problems in universal algebras*, J. Leech (ed.): Computational Problems in Abstract Algebra, pp. 263–297, Pergamon Press, New-York, 1970

[17] Yuri Matiyasevich, Geraud Sénizergues: *Decision problems for semi-Thue systems with a few rules*, Proceedings of the 11^{th} IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, pp. 523–531, 1996

[18] `http://www.theory.informatik.uni-kassel.de/~stamer/OlkPK.tar.gz`