Proceedings of the 8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010. Vol. 1. pp. 311–317.

An Attack on Dömösi's Cryptosystem

Zita Kovács, Andor Pénzes

University of Debrecen, Faculty of Informatics
e-mail: kovacs.zitu@gmail.com, andor.penzes@gmail.com

Abstract

In this paper we introduce an attack on a practical stream cipher based on a finite automata without outputs. For encryption and decryption the apparatus uses the same secret keys, which have the transition matrix of a key-automaton without outputs and with an initial state and final states. This cryptosystem called Dömösi cryptosystem from its maker. First, we introduce the system without its restrictions and we introduce the attack, which is based on probability theorem and create equivalent classes. The result of the attack an automaton which is state-equivalent to the key-automaton of the cryptosystem. Then we show the restrictions which help us against the attack.

Keywords: cryptography, cryptosystem, Dömösi system, finite automata, attack, security

MSC: 94A60, 68P25

1. Introduction

The Dömösi's cryptosystem [1, 2, 3] is a cryptographic apparatus based on a Rabin-Scott automaton as key for encoding and decoding information. Unlike other cryptosystem based on abstract automata, the key is neither a Mealy automaton nor a cellular automaton. It is similar to the Mealy machine based cryptosystems in that encoding and decoding is performed using a key automaton. Unlike Mealy machine or generalized sequential machine based cryptosystems, whether certain mappings preserve length and prefix or other mappings preserve prefix or not has no significance here, so these properties cannot exploined for breaking. The discussed system is similar to cellular automata based cryptosystems in that the key-automaton is an automaton without outputs, like the cells in a cellular automaton. Furthermore, the encoded message here is created in a somewhat similar manner, via manipulation of states, to the functioning of cellular automata, i.e. using the changes in the state of the key automaton. It is also similar to the cellular automata based cryptosystems in the sense that its random number generator is independent from the key.

Because of these properties, the Dömösi's cryptosystem is not a standard system, therefore we need own developed attacks which are different from standard attacks to proof the security of the system [7]. Although it uses a random number generator, it can take random number generators which are proved to be random indeed, or it can use any radioactive or other physical random number sources. The system is completely immune against reused key attack and substitution attack. It can not be attacked by exhaustive attack because of the sidereal number of possible keys. Statistical attacks and algebraic attacks are also hopeless. Last but not least, adaptive (or non-adaptive) chosen-ciphertext attack and chosen-plaintext attacks are ineffectual.

The aim of this paper is to introduce new attacks which may be more succesful comparing the above mentioned methods. On the basis of our results, we can improve the system such that it is going to become immune against our discussed new attacks too.

2. Preliminaries

We start with fundamental concepts. For all notions and notations not defined here we refer to [4, 5, 6].

By an alphabet we mean a finite nonempty set. The elements of an alphabet are called letters. A word over an alphabet Σ is a finite string consisting of letters of Σ. The string consisting of zero letters is called the empty word, written by λ. The length of a word w, in symbols $|w|$, means the number of letters in w when each letter is counted as many times it occurs. By definition $|\lambda| = 0$. At the same time, for any set H, $|H|$ denotes the cardinality of H. In addition, for every nonempty word w, denote by $\overrightarrow{w}$ the last letter of w. ($\overrightarrow{\lambda}$ is not defined.) Let Σ^* be the set of all words over Σ, moreover let $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$.

By an automaton we mean a finite Rabin-Scott automaton, i.e. a deterministic finite initial automaton without outputs supplied by a set of final states which is a subset of the state set. In more details, an automaton is an algebraic structure $\mathbf{A} = \{A, a_0, A_F, \Sigma, \delta\}$ consisting of the nonempty and finite *state set* A, the nonempty and finite *input set* Σ which is the *ciphertext alphabet*, a *transition function* $\delta : A \times \Sigma \to A$ which is the *key* used for encryption (and for the decryption), the *initial state* $a_0 \in A$ and (not necessarily nonempty) *set* $A_F \subseteq A$ *of final states.* The elements of the state set are the *states*, the elements of A_F are the *final states*, and the elements of the input set are the *input symbols.*

It may happens that the initial state is a final state as well (this is not excluded). An element of A^+ is called a *state word* and an element of Σ^* is called an *input word* or *plaintext.* State and input words are also called *state strings* and *input strings*, respectively.

If a state string $a_1a_2 \dots a_s (a_1, \dots, a_s \in A)$ has at least three elements, the states $a_2, \dots, a_{s-1}$ are also called *intermediate states.* It is understood that δ is extended to $\delta^* : A \times \Sigma^* \to A$ with $\delta^*(a,\lambda) = a, \delta^*(a,xq) = \delta(a,x)\delta^*(\delta(a,x),q), a \in A, x \in \Sigma, q \in \Sigma^*$.

In other words, $\delta^*(a,\lambda) = a$ and for every nonempty input word $x_1x_2\dots x_s$ $(x_1,x_2,\dots,x_s \in \Sigma)$ there are $a_1,\dots,a_s \in A$ with $\delta(a,x_1) = a_1, \delta(a_1,x_2) = a_2,\dots,\delta(a_{s-1},x_s) = a_s$ such that $\delta^*(a,x_1\dots x_s) = a_1\dots a_s$.

In the sequel, we will consider the transition of an automaton in this extended form and thus we will denote it by the same greek letter δ. If $\overrightarrow{\delta(a,w)} = b$ holds for some $a,b \in A, w \in \Sigma^*$ then we say that w *takes* the automaton from its state a into the state b, and we also say that the automaton *goes* from the state a into the state b under the effect of w. We say that $z \in \Sigma^*$ is a *dummy* string with respect to the input word $u \in \Sigma^*$ if for every nonempty prefix w of z, $\overrightarrow{\delta(a_0,uw)} \notin A_F$.

Finally for every pair $a,b \in A$ of states define the language $L_{a,b} \subseteq \Sigma^*$ of input words which take the automaton from the state a into the state b without intermediate final states. In formula, let

$$L_{a,b} = \{w \in \Sigma^* \mid \overrightarrow{\delta^*(a,w)} = b,$$

$$\forall u,w \in \Sigma^* : w = (uv \text{ and } u,v \neq \lambda) \Rightarrow \overrightarrow{\delta(a,u)} \notin A_F\}.$$

Furthermore we give a φ mapping ($\varphi : A_F \to P$), which orders each characters from the plaintext alphabet (P) to the final states. The encryption function orders a ciphertext to the plaintext, using the **A** automaton as a key: $C_A : P^* \to C^*$. The decryption function orders plaintext to the ciphertext, using the **A** automaton as a key: $D_A : C^* \to P^*$.

The encryption is the following: Each plaintext character p_i is mapped through a bijection to each final state by the function φ. Let p be the plaintext consisting of $p_1, p_2, \dots, p_k$. The plaintext is read character by character starting from the initial state of the key automaton. To each plaintext character is assigned a random input string of adjustable length in a given length interval. By linking these input strings together we get the encrypted message. Many ciphertext belongs to a plaintext. We have to choose one from these.

The decryption is the following: Let c be the ciphertext consists of $c_1, c_2, \dots, c_k$ strings. We starts the key-automaton from its initial state and we give it the word c. When we get a state that the function φ maps a character to, we mark the given character, and so on. Linking together these characters gives us the decrypted ciphertext (that means this is the plaintext).

3. Restrictions

3.1. The Minimum and the Maximum

The ciphertext increasable to arbitrary size because of the properties of the system. To control this we introduce the minimum and the maximum. These values are out of increase the security. The higher the minimum, the higher security level of the encryption, however this increases the size of the ciphertext.

We get different blocks which length is between the minimum and the maximum during the encryption of a character of the plaintext, which is also increases the security level of the encryption.

Of course, every block has equal length by chosen minimum equals to maximum.

So that, based on giving the minimum and maximum it can be controlled the block length of ciphertext generated by encryption of a character. We always considerate these values, therefore we choose not from an $L_{i,j}$ set but from an $L_{i,j}^{min,max}$ set. If the ciphertext $c = c_1 c_2 \dots c_k$, and the minimum is the min, and the maximum is the max, then $min < |c_i| < max, i = 1, \dots, k$ and $c_i \in L_{i,j}^{min,max}$ where $L_{i,j}^{min,max} = \{p \in P \mid p \in L_{i,j}$ and $min \leq |p| \leq max\}$. In the case of denominate automaton these values have lower bounds so there exist $k_1, k_2 : k_1 \leq min$ and $k_2 \leq max$, ergo the minimum and the maximum can not be chosen arbitrarily. There are many different ways to determine these lower bounds , we developed a method which based on making sets. We discuss this method later.

3.2. Initial conditions

Our minimum initial conditions needed for the encryption are the following: (a) to be able to get from the initial state to every final state ($\forall a \in A_F$ there exists a word $w \in L_{a_0,a}^{min,max}$), (b) to be able to get from every final state to every final state ($\forall a, b \in A_F$ there exists a word $w \in L_{a,b}^{min,max}$).

If an automaton completes these requirements it is called an *encryption automaton.* Since we could go from every state to every final state in one step ($\forall a \in A, \forall b \in A_F$ there exists a word $w \in L_{a,b}^{k,l}$ and $k = l = 1$), we have a *fast encryption automaton.*

Further, for simplicity, we suppose that the automaton used for encryption is minimal.

There are many methods to check if an automaton is an encryption automaton or not. We developed a method which based on make sets. The method is the following: for every final state and for the initial state (denote these states b) we making the following sets:

$H_0 := \{a \in A \mid$ there exists $w \in L_{b,a}^{min,max}\}$

$H_i := H_{i-1} \cup \{a \in A \mid$ there exists $w \in L_{c,a}^{min,max}, c \in H_{i-1}\}$

If there exists an index j where the set A_F is a subset of the set H_j then we say that every final state is *available* from the b state.

If this index exists for every considered b state, then the automaton is an encryption automaton.

Remark 3.1. These H sets are used to determine the lower bound of the minimum and the maximum.

3.3. Uploading method

From the automata uploaded with random numbers comply to the initial conditions only $14,7 - 15,8\%$ (where $|A| = 16$). The encryption could significantly slow down, if such a way completing a key-automaton, it is therefore necessary an uploading method which immediately generates an encryption automaton. This automaton satisfies that every final state were included exactly ones in each rows and each columns (fast encryption automaton) and in the other positions the non-final states have standard deviation. This standard deviation becomes an important factor against the attack.

4. The Attack

The goal of the attack is to make an automaton $\mathbf{A}'$ which has equivalent functions to the $\mathbf{A}$ automaton. That is if we encrypt any plaintext with the original key-automaton and decrypted with the new generated automaton we must get the original plaintext: $\forall p \in P : D_{A'}(C_A(p)) = p$.

If we whenever encrypt an arbitrary plaintext with the original $\mathbf{A}$ key-automaton, we do not receive the same ciphertexts, they are always different, so we do not expect from the new generated key-automaton $\mathbf{A}'$ that satisfying this: $C_A(p) = C_{A'}(p)$. So we can use the new key-automaton for decryption.

4.1. Statistical analysis

The first step of the attack is a statistical analysis, which is a sufficiently large sample is needed. The sample contains ciphertexts, each of them has been obtained to that we started the key-automaton $\mathbf{A}$ from its initial state for arbitrary plaintexts. According to the following statement every state is available a p word which length at most $|A| - 1$, so during the statistical analysis it is enough to analyse these p words.

Theorem 4.1. *Let $\mathbb{A} = (A, \Sigma, \delta)$ be a finite automaton. Moreover, let $a, b \in A$ be an arbitrary pair of states such that there exists an input word $p \in \Sigma^*$ with $\overrightarrow{\delta(a,p)} = b$. Then there exists an input word $q \in \Sigma^*$ with $\overrightarrow{\delta(a,q)} = b$ and $|q| \leq |A| - 1$.*

Proof of Theorem 4.1. Let $p \in \Sigma^*$ be a shortest input word with $\overrightarrow{\delta(a,p)} = b$. If $p = \lambda$ then the statement is trivial. Otherwise let $p = x_1x_2...x_n$ with $x_1, x_2, ..., x_n \in \Sigma$. If $n \leq |A| - 1$ then we are ready. Therefore, we may suppose $n \geq |A|$. But then the sequence $a, \delta(a, x_1), \overrightarrow{\delta(a, x_1x_2)}, ..., \overrightarrow{\delta(a, x_1x_2...x_n)}$ has at least one repetition. If $a = \overrightarrow{\delta(a, x_1x_2...x_n)}(= b)$ then we are ready because $a = \delta(a, \lambda)(= b, |\lambda| \leq |A| - 1))$. Therefore, if there exists an $1 \leq i \leq n$ with $a = \overrightarrow{\delta(a, x_1x_2...x_i)}$ then we may assume $i < n$. But then $a = \overrightarrow{\delta(a, x_{i+1}...x_n)}(= b) = \overrightarrow{\delta(a, x_i...x_n)}(= b)$ also holds contrary of the minimality of $|p|$. Therefore, we may suppose that for every $1 \leq i \leq n$, $a \neq \overrightarrow{\delta(a, x_1...x_i)}$. Hence there are $1 \leq i < j \leq n$ with $\overrightarrow{\delta(a, x_1...x_i)} = \overrightarrow{\delta(a, x_1...x_j)}$. Thus $\overrightarrow{\delta(a, x_1...x_{i-1}x_ix_{j+1}...x_n)} = \overrightarrow{\delta(a, x_1...x_n)}(= b)$ contrary of the minimality of n. $\square$

During the analysis we make all possible p words where $p \in \Sigma^*$ and $|p| \leq |A| - 1$. After we analyse the sample. We collecting those ciphertexts which begins with p and between these ciphertexts we calculating the number of the occurrance of ciphertexts which begins with px_i $(i = 1, \ldots, |\Sigma|)$ Thus each of every analysed p text generated a vector, denote this v_p. The v_p vector has $|\Sigma|$ elements and its components are the number of occurrances.

4.2. Classification

After the statistical analysis, we are creating equivalent classes using the obtained vectors. These classes containing the analysed p texts and each class will be given by those p texts whose vectors suchlike the euclidian distance of the corresponding components are insignificant ($p \equiv q$, if $|v_{p_i} - v_{q_i}| < \varepsilon, \varepsilon > 0, (i = 1, \ldots, |\Sigma|)$). The equivalent classes representing each of the states of the key-automaton **A**. Namely that state the **A** key-automaton get to from its initial state effect to the texts from the corresponding equivalent class. Since we could get infinitely many texts to a state, it is possible that to the equivalent classes are contains infinite number of elements.

4.3. Completing the automaton

The obtained classes will be the states of the key-automaton $\mathbf{A}'$, which cardinality is greater than or equal to the cardinality of the state-set of the **A** key-automaton. The input symbols and the original input symbols are the same. The initial state of the key-automaton $\mathbf{A}'$ is the class which corresponding the initial state of the key-automaton **A**. This is the class which contains the λ word. The final states of the key-automaton $\mathbf{A}'$ that classes which are corresponding the final states of the key-automaton **A**.

The uploading of the transition matrix is the following. We are writing the C_{px_i} class which corresponding the px_i word to the intersection cell of column of the C_p class and of the row of the input symbol x_i, for every texts from the given class (C_p).

Therefore, each cell may be as many entries as many triples in the C_p class. These entries are not necessarily the same class of. This means that the completed automaton might not be deterministic. There are further work with a non-deterministic automaton, first we should do the determinisation and then do the minimalisation. There are many algorithms for these tasks.

It is also possible that a cell does not contain anything (for that sample), i.e., does not occur any transition from C_p class effects to input symbols x_i. This means that the completed automaton might be partial. It surely happens when the original key-automaton is not fast encryption automaton.

5. Conclusions

If the automaton used for encryption is non-fast encryption automaton, then the obtained automaton from the attack will be partial (by the analysed sample), hence it is not be able to use for decryption. But the automata used by the system must be fast because using these automata will increase the speed of encryption. From an attack we get a deterministic automaton whose number of states is less than the number of the key-automaton. This is due to the developed construction of the automaton. Therefore the attack can not be used against the system. A further challenge is to verify this result, therefore we are going to develop a computer program which is implementing the attack.

Acknowledgements. We thank professor Pál Dömösi for his help and helpful comments. We also thank PhD School in Faculty of Informatics in University of Debrecen for supporting our research.

References

[1] Pál Dömösi, A novel stream cipher based on finite automata, *IntelliSec - The 1st International Workshop on Intelligent Security Systems,* 11-14th November 2009, Bucharest, Romania.

[2] Géza Horváth, The φ factoring algorithm (in Hungarian), *Alkalmazott Mat. Lapok*, Vol. 21 (2004), 355-364.

[3] Zoltán Mecsei, Andor Pénzes, A comparison of the DES and 3DES with Dömösi cryptosystems, *Proc. Pali'65, International Conference on Automata, Languages, and Related Topics, Debrecen, submitted for publication.*

[4] J.E. Hopcroft, R. Motwani, J.D.Ullman, Introduction to Automata Theory, Languages, and Computation, *Pearson Education, Addison Wesley*, Second Edition (2001).

[5] Renji Tao, Finite Automata and Application to Cryptography, *Tsinghua University Press*, (2008).

[6] A. J. Menezes, P. C. Oorschot, S. A. Vanstone (1996, 2001, 2008) Handbook of Applied Cryptography, *CRC Press.*

[7] Zita Kovács, Andor Pénzes, Analysis of the security of the Domosi's cryptosystem (in Hungarian), *II. Nyíregyházi Doktorandusz (PHD/DLA) Konferencia*, (2009), 261–265.

Zita Kovács, Andor Pénzes
University of Debrecen, Faculty of Informatics, 4032 Debrecen, Egyetem tér 1.