Modification of Finite Automata Public Key Cryptosystem

Siranush Chopuryan, Gevorg Margarov
State Engineering University of Armenia
siranush.ch@gmail.com, gmargarov@gmail.com

ABSTRACT: *Existing finite automata public key cryptosystems (FAPKC) are analyzed in this paper. General types of cryptanalytic attacks on FAPKC are discussed and methods of breaking FAPKC by some known types of attacks are introduced. The times needed to perform the usual cryptanalytic attacks on FAPKC are calculated, in order to estimate processing complexity of those attacks. The contrastive analysis of performed attacks' processing complexities is made to compare the vulnerabilities of FAPKC against discussed attacks. As a result, an improved FAPKC is designed in order to resist the presented types of attacks. Strong FAPKC is obtained by generating invertible nonlinear and linear automata of the suitable form.*

Keywords: Automata theory, Linear finite automata, Public key cryptosystem, Cryptanalysis, Cryptanalytic attacks, Time complexity

Received: 1 March 2010, Revised 28 March 2010, Accepted 3 April 2010

1. Introduction

Public key cryptosystems, discussed in this paper, are based on the automata theory. Public key in FAPKC is the composition of nonlinear and linear finite automata, whose inverses are easily calculated. Private key is a specific combination of those inverses.

It is known that the general inversion of public key automaton is a hard problem [1,2]. On the other hand, the public key automaton components and their inverses can be efficiently discovered using the algebraic theory of automata [3].

Weakness of the cryptosystem against the chosen plaintext attack, in case of non suitable nonlinear and linear automata usage, is investigated in this article. A method to generate suitable linear and nonlinear automata is introduced to increase the stability of the FAPKC.

It is shown that FAPKC is vulnerable to the exhaustive search attack as well, if the ending of the plaintext is known to the attacker. A method to prevent the exhaustive search attack is suggested in this article.

2. Finite automata public key cryptosystem

To design a FAPKC a pair of finite automata is offered. Encryption principle in FAPKC is shown in Figure 1, where M_{nl} is a nonlinear weakly invertible finite automaton (WIFA for short) with delay 0 and M_l is a linear WIFA with delay τ. The encryption automaton is the composition of M_{nl} and M_l, denoted by $M = M_{nl} \circ M_l$ that is also a nonlinear WIFA with delay $\tau = 0 + \tau$ [4].

Figure 2 shows the decryption principle, where M_{nl}^{-1} is the weak inverse of M_{nl} with delay 0 and M_l^{-1} is the weak inverse of M_l with delay τ. The composite finite automaton M is the *public key* in FAPKC, and the *private key* is M_l^{-1} and M_{nl}^{-1} finite automata and their connection order.

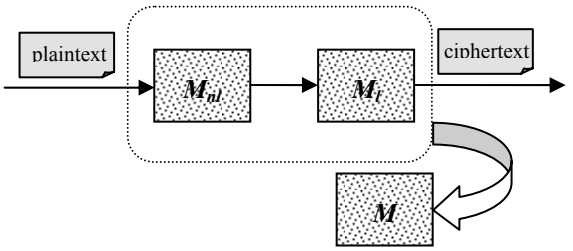


Figure 1. The encryption scheme

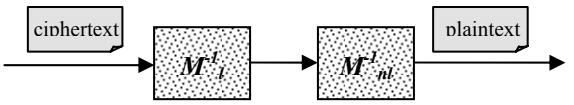


Figure 2. The decryption scheme

2.1 Linear WIFA

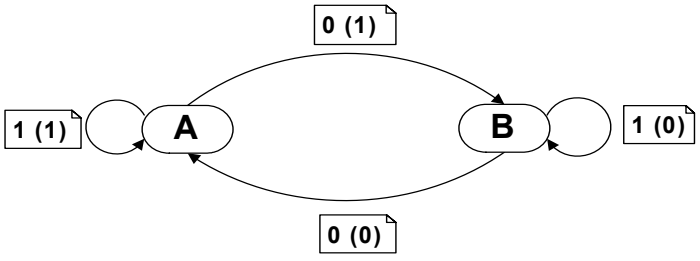
Linear automaton is of the $M_l = < X,Y,S_l,\delta_l,\lambda_l >$ form, where X is the input alphabet, Y is the output alphabet, S_l is the state alphabet, $\delta_l: S_l \times X \rightarrow S_l$ is the transition function and $\lambda_l: S_l \times X \rightarrow Y$ is the output function. X and Y are l -dimensional linear spaces over $GF(2)=\{0,1\}$. The input sequence $\{x(i)\}$ of the elements of the input alphabet X gives as an input of the finite automaton at time i. Here and after the input sequence of the automaton at time i will be denoted as a $x(i)$ (similarly $s(i)$ and $y(i)$). If $x(i) \in X$ and $y(i) \in Y$ present the input and the output at time i , respectively and both are l -dimensional column vectors, then the automaton M_l can be defined as follows:

$$y(i) = \sum_{j=0}^{\tau} A_j x(i-j) + \sum_{j=1}^{t} B_j y(i-j), \qquad i = 0,1,2,... \tag{1}$$

Finite automaton M_l is said to be an $\langle \tau,t \rangle$ -order memory finite automaton, which means that its initial state can be determined uniquely from the information at least about the last τ inputs and corresponding t outputs. Therefore, the initial state of the automaton M_l is determined the set of τ inputs and t outputs $\langle x(-1),x(-2),\ldots,x(-\tau),y(-1),y(-2),\ldots,y(-t) \rangle$.

In equation (1) A_j ($j = 0,1,2,\ldots,\tau$) and B_j ($j = 0,1,2,\ldots,t$) are l -dimensional linear coefficient matrices, which uniquely determine the finite automaton M_l . Typically l is equal to the block size, which is usually matched with the key size of the cryptosystem. The value of l defined by $\langle \tau,t \rangle$ -order of the encryption automaton. It can be noticed, that operations in (1) are usual addition and multiplication over $GF(2)$.

The finite automaton M_l described above, defined by (1), is called a linear finite automaton [3]. The graphical and tabular representations of the automaton M_l , as well as its realization scheme are presented in Figure 3 and 4 respectively.



x=0	**A**	**B**
A	**0**	**1**
B	**1**	**0**

x=1	**A**	**B**
A	**1**	**0**
B	**0**	**1**

Figure 3. Graphical and tabular representation of M_l

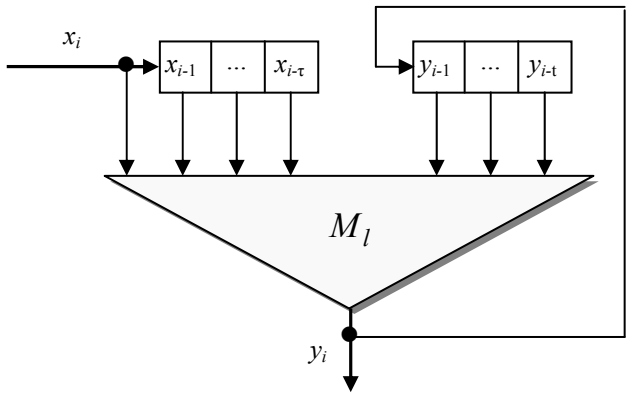


Figure 4. Realization scheme of M_l

The finite automaton M_l can be also considered as a linear finite automaton, only depending on τ inputs. The initial state of such automaton is determined the set of τ inputs $x(-1),x(-2),\ldots,x(-\tau)$. The finite automaton M_l defined by (2):

$$y(i) = A_0 x(i) + A_1 x(i-1) + \cdots + A_\tau x(i-\tau), \qquad i = 0,1,2,... \tag{2}$$

It is known, that for any finite automaton its inverse automaton can be constructed, which reproduces the inputs of the initial finite automaton. Thus, the following statement is true.

Statement: The finite automaton M_l is a weakly invertible finite automaton with delay τ if and only if the collection of matrices, $A_0^{-1}, A_1^{-1},\ldots,A_\tau^{-1},\tilde{A}_0^{-1},\tilde{A}_1^{-1},\ldots,\tilde{A}_\tau^{-1}$, can be derived from the collection of A_j ($j = 0,1,2,\ldots,\tau$)such that [3,5]:

$$x(i) = \sum_{j=0}^{\tau} A_j^{-1} y(i+j) + \sum_{j=1}^{t} \tilde{A}_j\, x(i-j), \qquad i = 0,1,2,... \tag{3}$$

For any initial state $s = \langle x(-1),x(-2),\ldots,x(-\tau)\rangle$ of M_l and for any input sequence $x(0)x(1)\ldots\ x(n+\tau)$, if $y(0)y(1)\cdots\ y(n+\tau) = \lambda(x(0)x(1)\cdots\ x(n+\tau))$, then inputs $x(0),x(1),\ldots,x(n+\tau)\in X$ can be calculated one by one from (3). Hence, the finite automaton defined by the equation (3) specifies the weak inverse with delay τ of M_l.

2.2 Nonlinear WIFA

Nonlinear automaton is of the $M_{nl} = <X,Y,S_{nl},\delta_{nl},\lambda_{nl}>$ form, where X is the input alphabet, Y is the output alphabet, S_{nl} is the state alphabet, $\delta_{nl}:S_{nl}\times F(X) \to S_{nl}$ is the transition function and $\lambda_{nl}:S_{nl}\times F(X) \to Y$ is the output function. X and Y are l -dimensional linear spaces over $GF(2) = \{0,1\}$ and $F(X)$ is a function introducing a nonlinear operation $\circ$ defined over $GF(2)$ [3]. The definition formula of M_{nl} is

$$y(i) = \sum_{j=0}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} \tilde{B}_j\, x(i-j) \circ x(i-j-1), \qquad i = 0,1,2,... \tag{4}$$

where B_j ($j = 0,1,2...,r$) and $\tilde{B}_j$ ($j = 1,2...,r-1$) are l -dimensional linear coefficient matrices over $GF(2)$, and B_0 is an invertible matrix.

The M_{nl} defined by equation (4) is an r -input memory finite automaton. As B_0^{-1} exists, then the definition formula of M_{nl}^{-1} will be

$$x(i) = B_0^{-1}\Big(y(i) + \sum_{j=0}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} \tilde{B}_j\, x(i-j) \circ x(i-j-1)\Big), \qquad i = 0,1,2,... \tag{5}$$

The r -output memory automaton $M_{nl}^{-1}$is a weak inverse with delay 0 of M_{nl}. For any initial state $\langle x(-1),x(-2),\ldots,x(-r)\rangle$ of M_{nl} there exists s' of M_{nl}^{-1} such that $\lambda'_{nl}(s',\lambda_{nl}(s,x)) = x$, where s' is the match state of s and is also defined by $\langle x(-1),x(-2),\ldots,x(-r)\rangle$.

3. FAPKC design principles

Finite automata public key cryptosystem works in the following way:

1. First construct two finite automata M_{nl} and M_l as defined above.
2. Construct the encryption automaton from the composition of M_{nl} and M_l, denoted by $M = M_{nl} \circ M_l$. The definition formula of M is obtained by substituting equation (1) into equation (3).

$$z(i) = \sum_{t=0}^{\tau} A_t \left(\sum_{j=0}^{r} B_j x(i-j-t) + \sum_{j=1}^{r-1} \tilde{B}_j \, x(i-j-t) \circ x(i-j-t-1) \right) \qquad i = 0,1,2,... \tag{6}$$

$M = M_{nl} \circ M_l$ is not exactly the same as $M_{nl} \cdot M_l$. Each state $s = < x(-1), x(-2), \ldots, x(-r-\tau) >$ of $M = M_{nl} \circ M_l$, is equivalent to the state $< s_{nl}, s_l >$, where $s_{nl} = < x(-1), x(-2), \ldots, x(-r) >$ is a state of M_{nl} and $s_l = < y(-1), y(-2), \ldots, y(-\tau) >$ is a state of M_l.

The equation (6) can be simplified as follows:

$$z(i) = \sum_{j=0}^{r+\tau} C_j x(i-j) + \sum_{j=1}^{r+\tau-1} \tilde{C}_j \, x(i-j) \circ x(i-j-1), \qquad i = 0,1,2,... \tag{7}$$

where

$$C_j = \sum_{\substack{t=0 \\ j=0}}^{\substack{t=\tau \\ j+r}} A_t B_j, \qquad \tilde{C}_j = \sum_{\substack{t=0 \\ j=0}}^{\substack{t=\tau \\ j=r-1}} A_t \tilde{B}_j, \tag{8}$$

are l-dimensional matrix polynomials over $GF(2)$, uniquely determining the finite automaton M.

The automaton M is made public.

3. Construct the inverse automata M_{nl}^{-1}, M_l^{-1} as defined above and keep them secret.
4. First chose a sequence $x(m+1)x(m+2)\ldots x(m+\tau)$ arbitrarily to encrypt the plaintext $x(0)x(1)\ldots x(m)$. Then input the plaintext $x(0)x(1)\ldots x(m+\tau)$ into $M = M_{nl} \circ M_l$ with initial state s.

The output $z(0)z(1)\ldots z(m+\tau)$ is the ciphertext.

5. To decrypt $z(0)z(1)\ldots z(m+\tau)$, first M_l^{-1} and the initial state s_l are used to obtain $y(0)y(1)\ldots y(m)$. Then the temporary sequence $y(0)y(1)\ldots y(m)$ is supplied into M_{nl}^{-1} with initial state s_{nl} to obtain $x(-\tau)\ldots x(-1)x(0)x(1)\ldots x(m)$as the output. The first τ is thrown out to restore the plaintext $x(0)x(1)\ldots x(m)$.

Take a notice, that the public key generated in the 2nd step certainly is transmitted by some method, including unsecured information channels for guaranteeing secured information exchange. Meanwhile, for the secret key, generated in the 3rd step, during the whole process of information exchange it doesn't appear any necessity for its transition.

4 Cryptanalysis of FAPKC

The aim of the cryptanalysis of public key cryptosystem is to find out a method of private key or plaintext recovering. Private key recovering performs with the assumption that appropriate private key is available for the attacker. While recovering plaintext it's assumed that private key is missing. Successful cryptanalysis of the cryptosystem make possible to elicit its vulnerabilities, which leads to private key or plaintext recovery. It is obvious, that estimation and elimination of vulnerabilities of cryptosystem allows to make it more secure. Therefore, while designing the cryptosystem it is necessary to analyze it via simulating attacker possible actions and taking into account the following worst assumptions about information awareness of the attacker [6]:

- attacker knows the whole information about the cryptosystem except for the private key;
- attacker has enough amount of ciphertexts;
- attacker has some ciphertexts and its corresponding plaintexts;
- attacker knows all public keys of current cryptosystem.

During cryptanalysis of cryptosystem it is necessary to examine various scenarios of attacker behavior, depend on his information awareness. Based on above mentioned worst assumptions the following five types of attacks on public key cryptosystem are formed [6]:

1. Private key direct calculation from public key
2. Ciphertext-only attack
3. Chosen-ciphertext attack
4. Chosen-plaintext attack
5. Privat key searching via exhaustive method

The first type of attack gives an opportunity to attacker to recover the private key from the appropriate public key of public key cryptosystem. The next three types of attacks encourage recovering of the plaintext from the ciphertext. The fifth type of attack like the first type is a method of private key recovery. The difference is the possibility to perform it without the knowledge of public key.

The unified mathematical theory of cryptanalysis doesn't exist to define whether the public key cryptosystem is secured or not. The only way to analyze the public key cryptosystem is to evaluate it against the probable types of attacks. In order to construct strong public key cryptosystem based on finite automata it is necessary to perform the cryptanalysis of existing FAPKC's. During FAPKC cryptanalysis it isn't reasonable examine chosen-ciphertext attack and private key searching via exhaustive method. It is conditioned by the following facts: the first one is just an insignificant modification of ciphertext-only attack, and the second one is an enough trivial, but unsolvable problem for a reasonable period of time. That is why during FAPKC cryptanalysis is examined only the following three types of attacks:

1. Private key direct calculation from public key
2. Ciphertext-only attack
3. Chosen-plaintext attack

As a result, becomes reasonable to find out the vulnerabilities of existing FAPKC' against abovementioned three types of attack. For this purpose, the methods are examined that allow attacker to recover the private key from the public key of the cryptosystem, and methods to recover the plaintext from the ciphertext.

4.1 Private Key Direct Calculation from Public Key

Private key recovery is one of the problem of attacker, who's main goal is to break public key cryptosystem. Hence, it is reasonable to estimate stability of the FAPKC against attack of private key direct calculation from public key. Public key of the cryptosystem FAPKC includes some information about encryption automaton, which is a composition of linear and nonlinear automata. Attacker, who has an access to public key, in other words to composite automaton, is able to calculate private key via composite automaton inversion. Inversion is possible to perform by two ways:

1. composite automaton direct inversion;
2. automaton componentwise inversion.

4.1.1 Composite Automaton Direct Inversion

The main idea of automaton direct inversion is to construct inverse automaton of the composite one without any knowledge about component automata. Direct inversion schematic diagram presented in Figure 5 for composite finite automaton M consisting of component automata $M_0, M_1,\ldots, M_n$ with delay $\tau_0,\tau_1,\ldots,\tau_n$ correspondingly.

It is known, that if finite automaton M is a weak invertible automaton with delay τ, then exists its inverse finite automaton M^{-1} [10]. In case of inverse finite automaton M^{-1} calculation success from the composite automaton $M = \langle M_0,\ldots, M_n\rangle$ with delay $\tau = \tau_0+\cdots+\tau_n$ private key calculation will be possible. Direct inversion of finite automaton M is a way to M^{-1} determine the finite automaton . The objective of automaton direct inversion algorithm is the calculation of its input sequence for $\tau = \tau_0+\cdots+\tau_n$ arbitrary output elements.

Further it is necessary to estimate attacker time resources for different types of attack. Time complexity is the time needed to perform attack [7].

Time complexity of finite automaton direct inversion is calculated in consequence of analysis of current finite automaton tree. Fig. 6 shows the input tree of the automaton M_l, where A is its initial state. The finite automaton tree is analyzed by value τ. The value τ defines the depth of analysis in the automaton tree. This means that the more is the depth of finite automaton tree analysis the more is time complexity of its direct inversion.

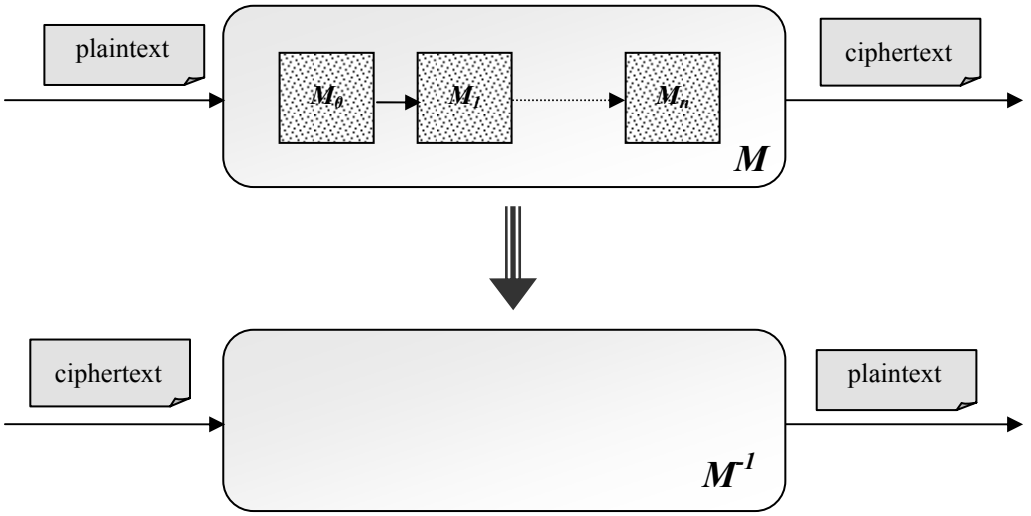


Figure 5. Direct inversion schematic diagram for finite automaton M

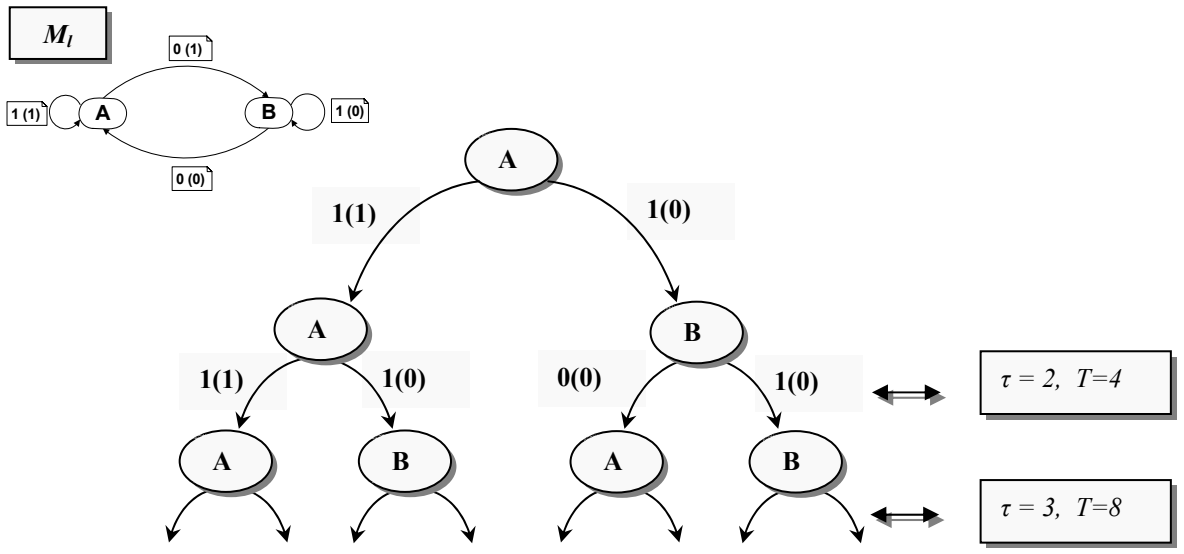


Figure 6. Finite automaton M_l input tree

The composite finite automaton states quantity and its delay τ are the input data necessary to solve the problem of composite finite automaton direct inversion. Time complexity of the composite automaton direct inversion problem for the input word of length τ is equal to:

$$T(\tau) = \sum_{j=0}^{\tau-1} |S|^{\tau-j}, \tag{9}$$

where τ is the composite finite automaton delay;

S is the finite set of states of the composite finite automaton M_l. As it is noticed, time complexity of finite automaton direct inversion depends on depth finite automaton input tree analysis.

For instance, the finite automaton M_l direct inversion time complexity by three-level analysis exceeds time complexity by two-level analysis approximately two times.

Let us assume that delays τ_i, $i \in [0,n]$ of all component automata M_i are equal to each other and are $\tau*$. Time complexity of the composite finite automaton M direct inversion will be equal to:

$$T(\tau) = \sum_{j=0}^{n\tau^*} |S|^{n\tau^* - j}. \quad (10)$$

From the equation (9) one can notice, that direct inversion method is impractical for the composite automaton with sufficiently long delay $\tau = \tau_0 + \cdots + \tau_n$. Figure 7 shows direct inversion time complexity changing with increasing the delay τ of the composite automaton for the definite automaton M with the set of the states $S = \{s_0, s_1, \ldots, s_9\}$.

Attack of the private key direct calculation from public key against FAPKC0 is reduced to the problem of composite automaton direct inversion; therefore, time complexity of current attack has the same dependence on the value of τ as it is shown in Figure 7.

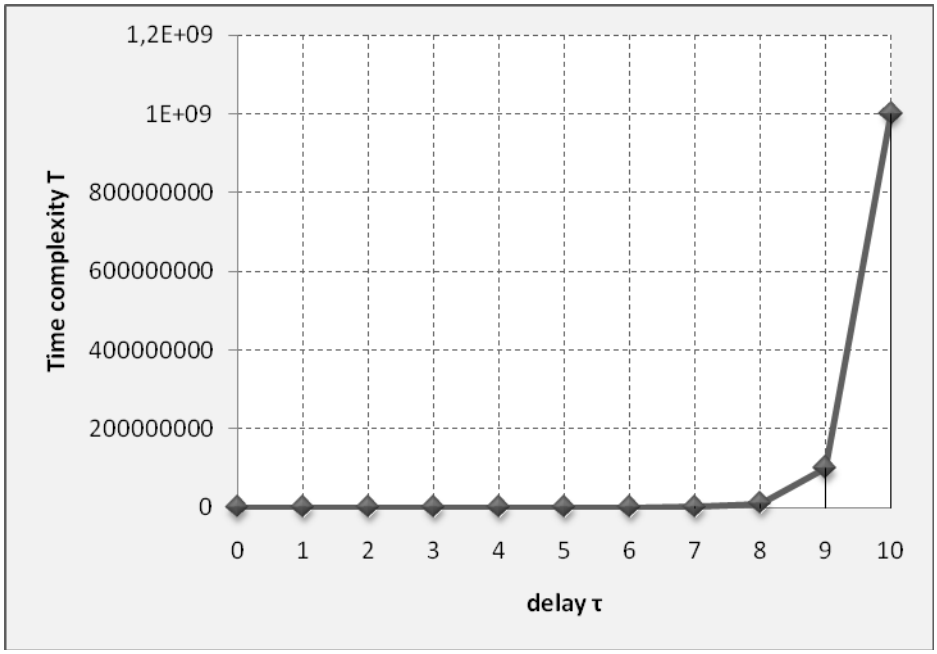


Figure 7. The composite automaton direct inversion time complexity

Thereby, as it is shown, the FAPKC stability against attack of private key direct calculation from public key depends on the composite automaton's delay τ, which is the public key of the cryptosystem.

4.1.2 Composite Automaton Componentwise Inversion

Unlike to the previous method, where attacker tries to figure out the inverse automaton of the composite one directly, without the knowledge about component automata, chief matter of this method is the decomposition of the composite automaton for construction of its inverse automaton. Current method also might favor the realization of private key direct calculation from public key attack. Attacker using information about composite finite automaton of encryption $\langle M = \mathrm{M}_0, \ldots, M_n \rangle$, which is the public key of the FAPKC cryptosystem, might detect component finite automata $M_0, \ldots, M_n$ and inverse finite automaton M_i^{-1} for any i, where $i \in [0,n]$. As a result of cascade connection of detected component automata inverses $M_n^{-1}, \ldots, M_0^{-1}$ constructed inverse finite automaton M^{-1}. The last one is the inverse automaton of the composite automaton $M = \langle M_0, \ldots, M_n \rangle$. The main object is decomposition of the composite automaton M. Up to now exact method of finite automaton decomposition doesn't exist [8].

The schematic representation of composite automaton M inversion via its decomposition is shown Figure 8.

In certain cases, for instance, in the public key cryptosystem FAPKC presented in Chapter 2, where the composite automaton is consist of two linear automata which are connected sequentially, finite automaton $M = \langle M_1, M_2 \rangle$ decomposition problem is reduced to the factorization problem of the matrix polynomial C_j [9].

$$C_j = \sum_{t=0}^{t=\tau_1} \sum_{k=0}^{k=\tau_2} A_t B_k, \quad (11)$$

where C_j are coefficient matrices that are uniquely determine composite finite automaton M , where the definition formula of M is the following:

$$z(i) = \sum_{j=0}^{\tau_1+\tau 2} C_j x(i-j), \qquad i = 0,1,2... \tag{12}$$

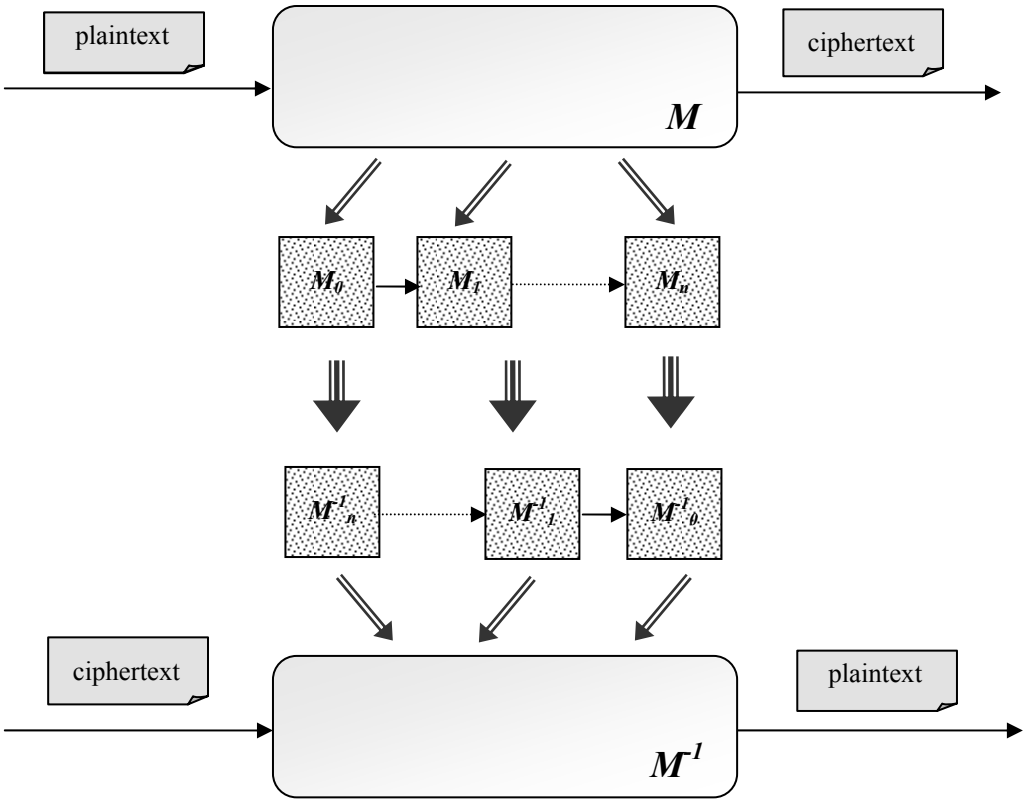


Figure 8. Componentwise inversion schematic diagram for finite automaton M

After figuring out the component matrices $A_t (t = 0,1,2..., \tau_1)$ and $B_k (k = 0,1,2..., \tau_2)$ it is possible to construct the finite automaton M_1 and M_2, where the finite automata M_1 and M_2 are defined by the following equations correspondingly:

$$y(i) = \sum_{t=0}^{\tau_1} A_t x(i-t), \qquad i = 0,1,2... \tag{13}$$

and

$$y(i) = \sum_{k=0}^{\tau_2} B_k x(i-k), \qquad i = 0,1,2... \tag{14}$$

where $x(i) \in X$ and $y(i) \in Y$ are input and output elements at the moment i respectively.

Thereby, the inverse automaton of the composite automaton $M = \langle M_1, M_2 \rangle$ can be detected when it is possible to construct the inverse automata for the component finite automata M_1 and M_2.

In spite of the fact, that exists the polynomial algorithm of polynomials factorization over the field $GF(q)$, there is no probabilistic algorithm of matrix polynomials factorization over the field $GF(q)$. Heretofore it is known some specific algorithms of matrix polynomials factorization, e.g: factorization over linear (R_a, R_b) transformation [9] and factorization by reducing canonical diagonal form for matrix polynomials. But those specific factorizations simplify private key detection problem only if the public key is constructed from weak invertible finite automata, like in public key cryptosystem FAPKC.

For identification of the FAPKC stableness against private key direct calculation from public key attack it is necessary to estimate time complexity of the composite automaton compositewise inversion algorithm, via its decomposition.

If one successfully decomposed the composite automaton and figured out the component automata, then time complexity of componentwise inversion of n-component automata will be equal to:

$$T(\tau) = \sum_{i=1}^{n} T(\tau_i), \tag{15}$$

where $T(\tau_i)$ is time complexity of i^{th} component automaton inversion and equal to:

$$T(\tau_i) = \sum_{j=0}^{n} |S_i|^{\tau_i - j}. \tag{16}$$

The input data, necessary for component-wise inversion problem solving for n-component automata, are number of states of each component finite automaton and their delays τ_i, $i \in [0,n]$.

If assume, that number of states of all component automata are equal to and delays τ_i are the same, then time complexity of componentwise inversion will be equal to:

$$T(\tau) = \sum_{i=0}^{n} T(\tau_i) = \sum_{i=0}^{n} \sum_{j=0}^{\tau_i - j} |S_i|^{\tau_i - j} = n \times \sum_{j=0}^{\tau/n - j} \left(|S|/n\right)^{\tau/n - j} \tag{17}$$

This implyies that time complexity of private key detection problem for attacker who has an access to component automata that are the part of the public key, is estimated via equation (17).

Figure 9 shows time complexity behavior for composite automaton componentwise inversion depending on increase in value of τ of current composite automaton. Results are displayed for specific automaton M, which is the composition of two finite automata with the set of states $S_{1,2} = \{s_0, s_1, \ldots, s_4\}$.

Confront Figure 7 and Figure 9 one can notice, that time complexity of component automaton componentwise inversion excels time complexity of component automaton direct inversion, presented in the previous chapter. Hence, in the public key cryptosystem FAPKC private key direct calculation from public key via composite automaton direct inversion needs more time then private key direct calculation from public key via composite automaton componentwise inversion.

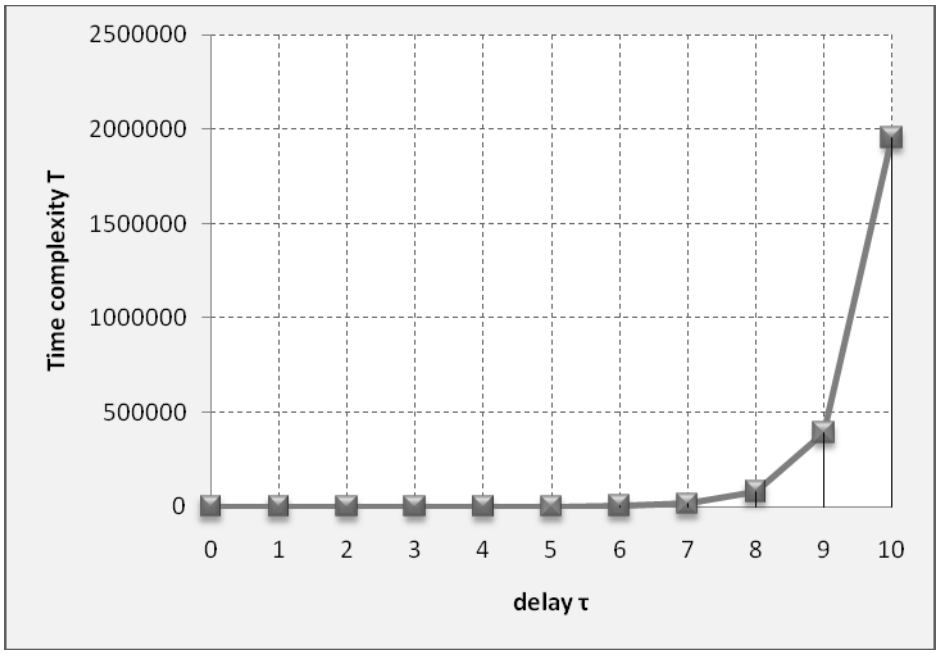


Figure 9. The composite automaton componentwise inversion time complexity

4.2 Ciphertext-only Attack

When attacker has access to the ciphertexts of several plaintexts, which are encrypted with the same encryption algorithm, then he has a possibility to perform ciphertext-only attack. In the public key cryptosystem FAPKC attacker's main goal is to obtain appropriate plaintext $x(0)x(1)...x(n)$ from the possessed ciphertext $z(0)z(1)...z(n+\tau)$, where $x(i)$ and $z(i)$ are input and output symbols of the encryption automaton respectively. Ciphertext-only attack for FAPKC is reduced to the problem of solving a system of linear equations over GF(2). This system of linear equations is formed from definition formula of encryption automaton with arguments $x(0), x(1),\cdots, x(\tau)$ [11].

$$\begin{pmatrix} z(0)=C_0x(0)+C_{-1}x(-1)+\cdots+C_{-\tau}x(-\tau) \\ z(1)=C_1x(1)+C_0x(0)+\cdots+C_{1-\tau}x(1-\tau) \\ \vdots \\ z(\tau)=C_\tau x(\tau)+C_{\tau-1}x(\tau-1)+\cdots+C_0x(0) \end{pmatrix} \qquad i=0,1,2,\ldots \tag{18}$$

The current system of linear equations is formed from the equation (12) where the arguments are τ, $\tau < n$ input elements x(0), x(1),...,x(τ) of the encryption finite automaton. Solving of the system of linear equations (18) is simplified when it is made out only for the first τ input elements. For the complete detection of the plaintext $x(0)x(1)...x(n)$, is formed the system of linear equations (18) not only for the first but also for next τ input elements.

$C_j(j=0,1,2,\ldots,\tau)$are $l\times l$ -dimensional coefficient matrices over GF(2) in the system of linear equations (18), that are uniquely determine encryption finite automaton. Input elements x(0),x(1),…,x(τ)are only unknowns in this system of linear equations, as the encryption finite automaton and so the coefficient matrices C_j are public knowledge. Thereby, plaintext detection via ciphertext-only attack is reduced to the problem of solving the system of τ linear equations with τ unknowns.

Time necessary to attacker for ciphertext-only attack against FAPKC is estimated via time complexity of solving the system of τ linear equations with τ unknowns [4]. Hence, time complexity of ciphertext-only attack for FAPKC is equal:

$$\mathrm{T}=\Theta\left(\tau^3\right) \tag{19}$$

Solving the system of linear equations over the field $GF(2)$ is not an intractable problem for the moderate amount of arguments [8]. The arguments of the system of linear equations (18) are input elements x(0), x(1),...,x(τ) of the encryption finite automaton, which is defined by the equation (12). The amount of arguments in the equation (12) is depends on delay τ of the encryption finite automaton. Delay τ of the encryption finite automaton is equal to $\tau=\tau_1+\tau_2$, where τ_1 is delay of the component automaton M_1, and τ_2 is delay of the component automaton M_2, where M_1 and M_2 compose the encryption automaton.

From the equation (19) one can notice, that current FAPKC shortcoming against ciphertext-only attack is conditioned by usage of linear finite automata with short delay τ. Finite automata short delay τ cause the small amount of arguments in the system of linear equations over $GF(2)$, that simplifies its solving. The above mentioned shortcoming of public key cryptosystem FAPKC is possible to eliminate by increasing the encryption automaton delay τ, that in turn leads to increasing of the arguments of the equation (12). As a result, it makes difficult the solving of the system of linear equations (18).

Figure 10 shows the relation of ciphertext-only attack time complexity against the cryptosystem FAPKC and the encryption composite automaton delay τ of current cryptosystem.

Thereby, as it is shown, the public key cryptosystem FAPKC stability against ciphertext-only attack depends on time complexity of system of linear equation solving for τ unknowns and defines by the equation (19).

4.2 Chosen-plaintext Attack

Since the encryption algorithm of FAPKC is known for everyone, one may guess the possible plaintexts and encrypt them easily. When the result of encrypting some guessed plaintext coincides with the ciphertext, the guessed plaintext is the virtual plaintext. FAPKC is sequential and its block length is small in order to provide a small key size. But small block length causes to the weakness of the cryptosystem for the divide and conquer attack. In fact, the guess process can be reduced to guessing a piece of plaintext of length τ and deciding its first digit. That is, guess a value of the first τ digits of the plaintext and encrypt it using the public key and compare the result with the first τ digits of the virtual ciphertext. If they coincide, the process repeated for guessing next digit of the plaintext.

Different texts random selection, their encryption and comparison with available ciphertext is an intractable problem and practically unsolvable problem. This complicates plaintext detection for the attacker. The mentioned problem is simplified in the public key cryptosystem FAPKC, presented in Chapter 2. It is conditioned by the usage of finite automata which makes

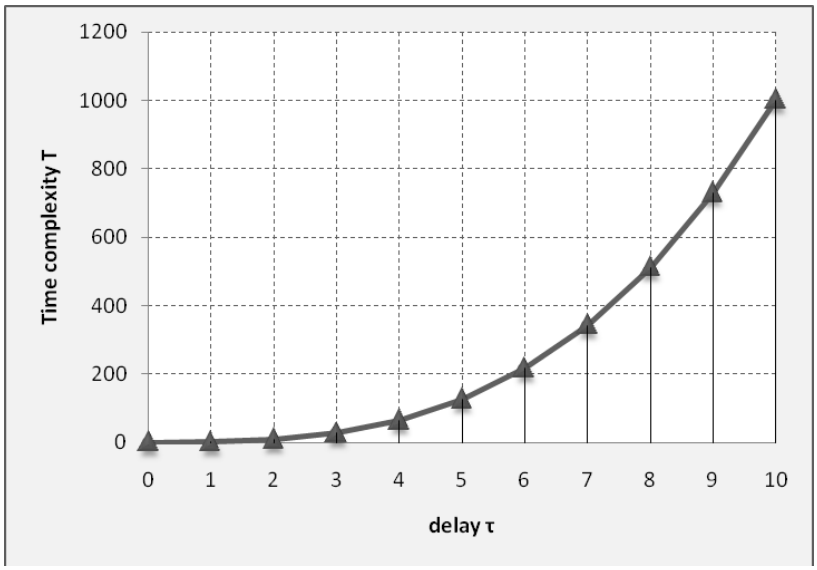


Figure 10. Time Complexity of the Ciphertext-only attack

public key cryptosystem sequential. It means that each symbol encrypted via such cryptosystem includes information about previously encrypted symbols. Finite automata public key cryptosystem is sequential because of finite automata behavior. Chosen-plaintext attack against FAPKC is simplified for attacker also by the usage of small block size during block encryption that leads to the increasing of the key size in the current cryptosystem. As a rule, the block size is defined by delay τ of the encryption finite automaton. But the small key size causes the cryptosystem FAPKC vulnerability against chosen-plaintext attack because of the possibility to apply divide and conquer principle. As it is noticed, plaintext detection in public key cryptosystem FAPKC can be reduced to the detection of the part of the plaintext. In fact, the guess process can be reduced to guessing a piece of plaintext of length $\tau = \tau_0 + \cdots + \tau_n$ and deciding its first digit. That is, guess a value of the first $\tau = \tau_0 + \cdots + \tau_n$ digits of the plaintext first, and then encrypt it using the public key and compare the result with the first $\tau = \tau_0 + \cdots + \tau_n$ digits of the virtual ciphertext. If they coincide, then the first digit of the guessed plaintext is indeed the first digit of the virtual plaintext. Repeat this process for guessing next digit of the plaintext, and so on [12].

Schematic representation of chosen-plaintext attack against FAPKC is introduced in Figure 11. The length of guessing piece of the plaintext is defined by the encryption finite automaton delay.

Since in finite automata public key cryptosystem the plaintext is encrypted by the usage of finite automaton, then the plaintext sequential detection is reduced to the sequential search problem, allowing to find output sequence of finite automata from its

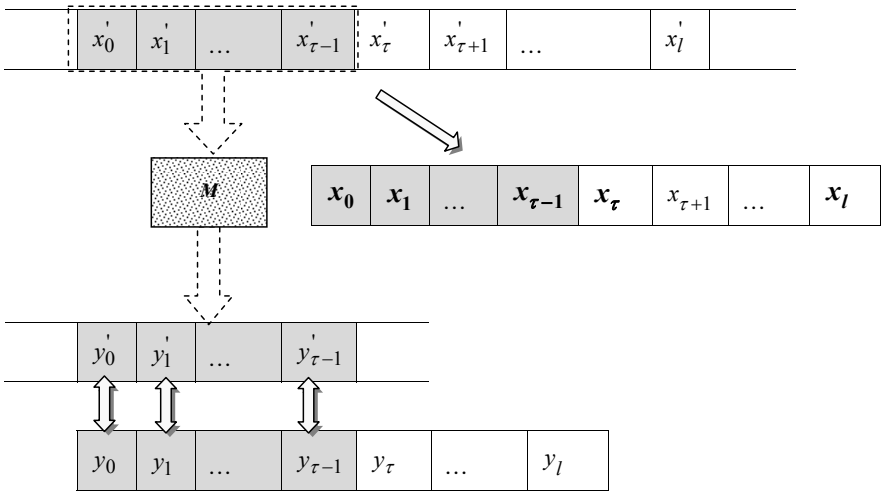


Figure 11. Schematic representation of chosen-plaintext attack against FAPKC

input sequence of length τ . For sequential search algorithm description is examined the finite automaton $M = \langle X,Y,S,\delta,\lambda \rangle$. It is assumed, that attacker knows the initial state s_0 of the finite automaton M and its output sequence $y_0 y_1 ... y_\tau$. The main goal of attacker is the finding of finite automaton M input sequence $x_0 x_1 ... x_\tau$. The sequential search algorithm to find an input sequence from an output sequence is the like of the following, where the input data is the stat s of automaton M and an output sequence $y_0 y_1 ... y_\tau$, and the output data is an input sequence $x_0 x_1 ... x_\tau$:

1. Assume, that $i = 0$.
2. Identify with $X_{s,x'_0,x'_1,...,x'_{i-1}} = \{x | x \in X, y_i = \lambda(\delta(s_0,x'_0,x'_1,...,x'_{i-1}),x)\}$ in the case of $i > 0$, or with $\{x | x \in X, y_i = \lambda(s_0,x)\}$ otherwise.
3. If $X_{s,x'_0,x'_1,...,x'_{i-1}} \neq \varnothing$, then choose an element in it as x'_i, delete this element from the set. Increase i by 1 and go to the Step 4, otherwise, decrease i by 1 and go to the Step 5.
4. If $i > \tau$, then the output sequence $x'_0,x'_1,...,x'_{i-1}$ is the desired sequence $x_0 x_1 \cdots x_\tau$, and the further operations is stopped. Otherwise, go to the Step 2.
5. If $i \geq 0$, go to the Step 3. Otherwise, prompt failure information and the further operations are stopped.

The presented algorithm implements backtracking, that is during each unsuccessful outcome make the jump to the previous state.

It is convenient to understand the execution of this algorithm, by means of the tree $T_{M,s_0,y_0,...,y_\tau}$, presented in the Figure 12.

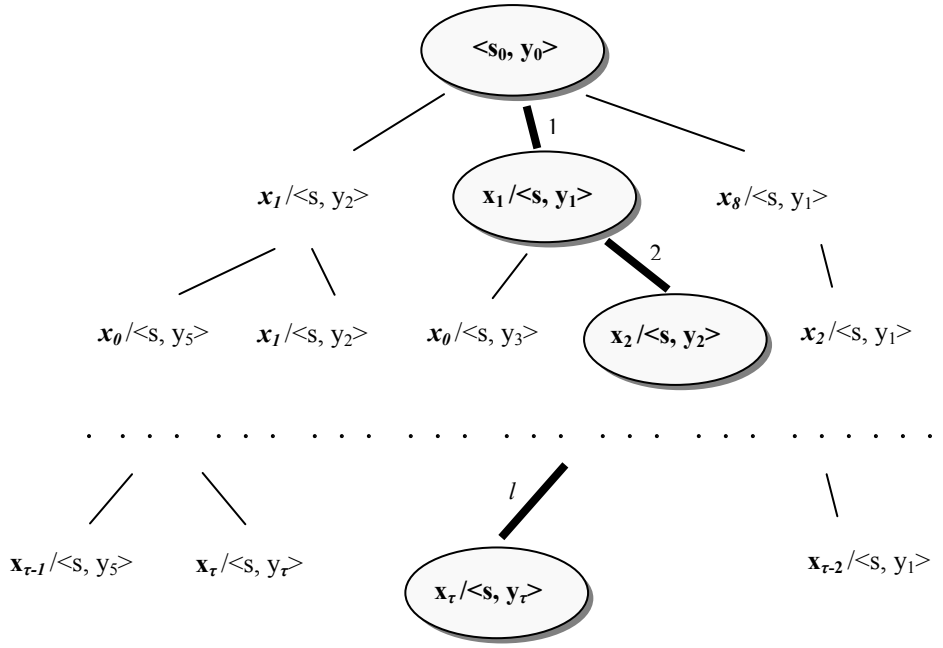


Figure 12. Work tree $T_{M,s_0,y_0,...,y_\tau}$ of the finite automaton M

The input data of the algorithm is the initial state s and the output sequence $y_0 y_1 ... y_\tau$. The purpose of the algorithm is to figure out the arc label sequence of input sequanece $x_0 x_1 ... x_\tau$ from the paths with longth τ+1 in the tree $T_{M,s_0,y_0,...,y_\tau}$. To find one of the paths of length τ+1 in $T_{M,s_0,y_0,...,y_\tau}$ the algorithm attempts to exhaust all possible paths from the root to leaves. Whenever the level of a searched leaf is less than τ+1, i.e., the path is not one of the longest, the search process comes back until an arc which is not searched yet is met, then the search process goes forward again. Whenever the level of a searched leaf is τ+1, i.e., the path from the root to this leaf is the longest, the search process finishes and the arc label sequence of this path is the output. As a result, the detected sequence of arcs is the desired sequence, that leads from the input element x_0 to the input element x_τ. The amount of passed arcs in during sequential search algorithm $T_{M,s_0,y_0,...,y_\tau}$ shows the search amount in the tree.

Since chosen-plaintext attack against FAPKC reduced to finite automaton tree sequential search problem where search depth is τ , time complexity of chosen-plaintext attack against FAPKC is estimated via tree search time complexity. The last one is equal to:

$$T = \Theta(\tau) \tag{20}$$

Time complexity behavior of the chosen-plaintext attack against the public key cryptosystem FAPKC due to increasing the value τ of encryption finite automaton is presented in Figure 13.

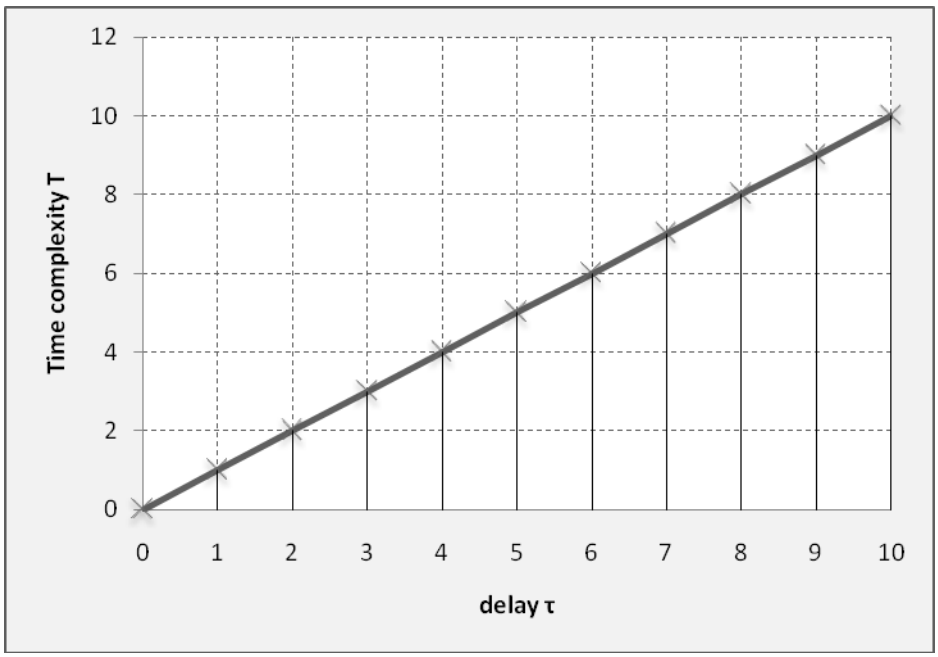


Figure 13. Time Complexity of the Chosen-plaintext attack

As it is shown in this chapter, the public key cryptosystem FAPKC stability against chosen-plaintext attack depends on time complexity of sequential search in the input tree of the encryption finite automaton and the stability is defined by the equation (2.12).

5. Contrastive Analysis of Cryptanalytic Attacks on FAPKC

Cryptanalysis of the FAPKC cited above allows to estimate the cryptosystem and to detect its vulnerabilities against examined types of attack. Consequently, taking into account discovered vulnerabilities, it is possible to define basic objectives of strong public key cryptosystem construction based on finite automata.

In order to analyze findings of the performed attacks on the FAPKC it is necessary to compare their time complexities. On the assumption of the large spread of time complexities of performed attacks, they are presented in the logarithmic scale on the Figure 14. Logarithmic scale allows to show the evident differences between time complexities of attacks. As one can see from the Figure 14 time complexity of the private key direct calculation from public key attack in both direct and componentwise inversion of the encryption automaton excels time complexities of ciphertext-only attack and chosen-plaintext attack.

From the equations (9) and (16) it is obvious that the private key direct calculation from the public key attack via both methods of the encryption automaton inversion can be implemented with the exponential time algorithms. Implementation algorithms of ciphertext-only attack and chosen-plaintext attack are polynomial-time algorithms, i. e., they have the polynomial time complexities, as it is shown in the equations (19) and (20).

As it is visible from the Figure 14, the spread of the attacks implemented via polynomial algorithm and attacks implemented via exponential algorithms, changes its behavior for small amount of input data. In the cryptosystem FAPKC input data depends on the value of τ. For instance, time complexity function of the private key direct calculation from the public key attack via encryption automaton componentwise inversion has a better behavior then time complexity function of the ciphertext-only attack for $\tau \leq 3$. For mentioned values these attacks can be performed in reasonable time without any special computation resources. Hence, here and after attack on FAPKC will be examined only for definite value of τ.

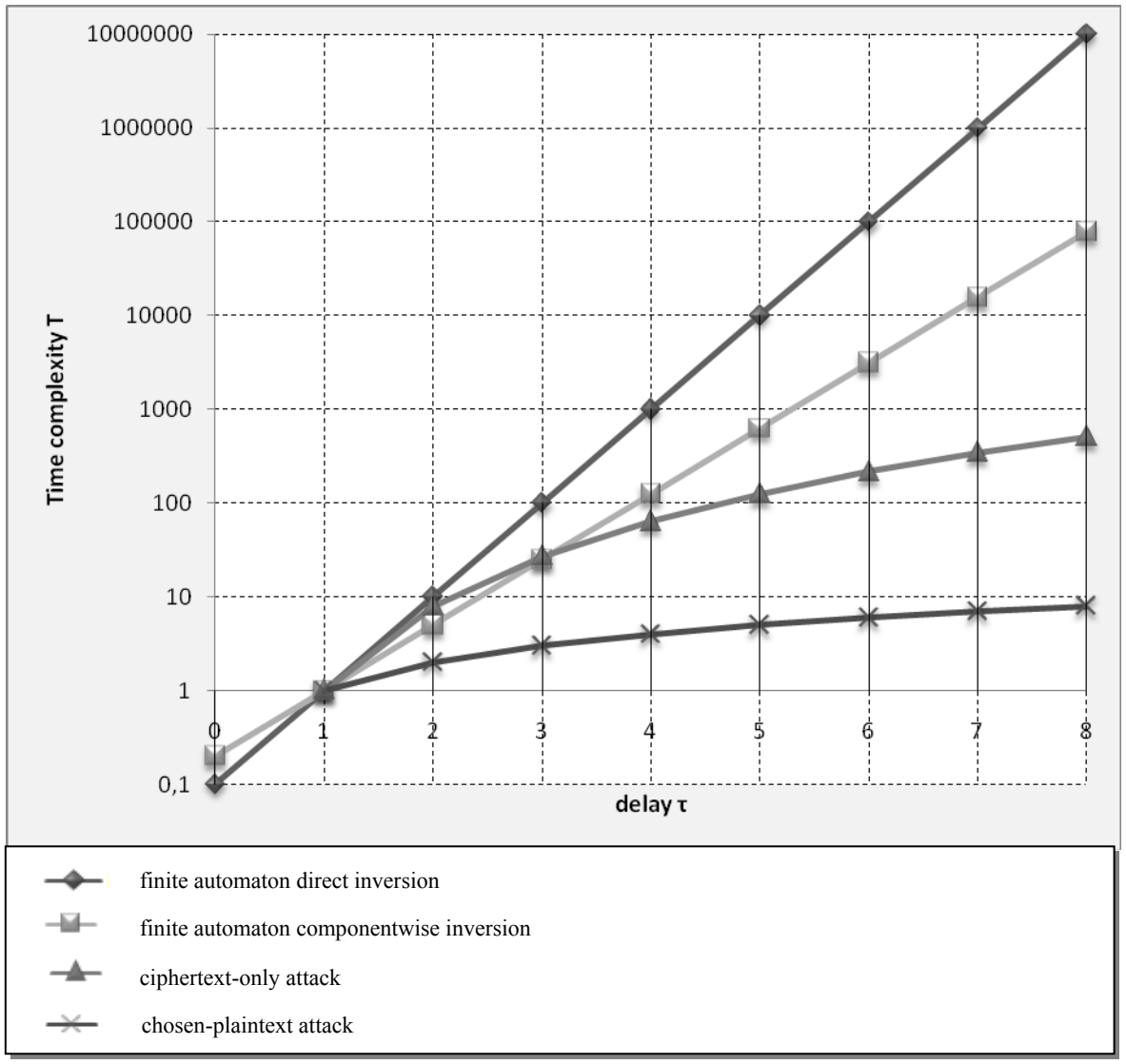


Figure 14. Time complexities contention of different attack against FAPKC

The private key direct calculation from the public key with both methods of encryption automaton inversion is implemented via exponential time algorithms. Therefore, it is not vital to increase FAPKC resistance against attack of private key's direct calculation from public key. FAPKC resistance is reasonable to increase against ciphertext-only attack and chosen-plaintext attack, as their algorithms are polynomial-time.

6. Modified FAPKC

The weakness of FAPKC against chosen plaintext attack is conditioned by usage of nonlinear WIFA with delay 0 and linear WIFA with delay τ.

Chosen plaintext attack for FAPKC is reduced to the problem of solving a system of nonlinear equations (18) over GF(2), that is known to be very hard if the number of its arguments is large. To increase the number of arguments in (12), the delay τ of the encryption automaton is increased. The delay τ of the encryption automaton is: $\tau = 0 + \tau$, where 0 is the delay value of the nonlinear component automaton, and τ is the delay value of the linear component automaton.

First way to increase the encryption automaton delay τ is to change the nonlinear component automaton into a WIFA with delay τ_1. Second way is to make the component linear WIFA automaton's delay longer by adding new states to its state alphabet.

6.1 Nonlinear WIFA modification

The definition formula of the automaton M_{nl} is $y(i)=\sum_{j=0}^{r} B_j x(i-j)+\sum_{j=1}^{r-1} \tilde{B}_j\, x(i-j)\circ x(i-j-1)$, where B_j (j = 0,1,2...,r) j = 0 j =1 and B_j (j = 1,2...,r-1) are $l \times l$ coefficient matrices over GF(2), and B_0 is an invertible matrix. The operation $\circ$ is defined to be a nonlinear operation over GF(2).

To modify the nonlinear WIFA with delay 0 to a nonlinear WIFA with delay τ, we redefine the operation $\circ$ such a way to get nonlinear WIFA with delay τ. Definition formula will be

$$y(i)=\sum_{j=0}^{r} B_j x(i-j)+\sum_{j=1}^{r+\tau} \tilde{B}_j\, x(i-j)\circ\cdots\circ x(i-j-\tau) \qquad i=0,1,2..., \tag{21}$$

Apparently, M_{nl} now is a nonlinear WIFA with delay τ. The equation (21) now can be rewritten as follows:

$$z(i)=\sum_{j=0}^{r+\tau} C_j x(i-j)+\sum_{j=1}^{r+2\tau} \tilde{C}_j\, x(i-j)\circ\cdots\circ x(i-j-\tau) \qquad i=0,1,2..., \tag{22}$$

6.2 Linear WIFA modification

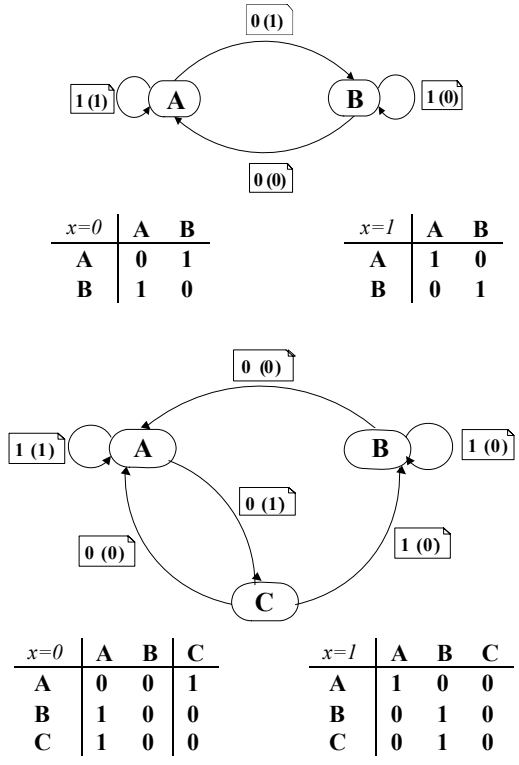
New states can be added between any two states to have a longer delay for the linear automaton M_l. The resultant automaton has to be equivalent to the source automaton according to the ***Definition 1.***

Definition 1. Let $M_1 =< X,Y,S_1,\delta_1,\lambda_1 >$ and $M_2 =< X,Y,S_2,\delta_2,\lambda_2 >$ be a pair of automata. States $s_1 \in S_1$ and $s_2 \in S_2$ are said to be *equivalent* if for any x(0)x(1)··· x(m), such that x(0)x(1)··· x(m) $\in X$,

$$\lambda_1(s_1,x(0),x(1),...,x(m)) = \lambda_2(s_2,x(0),x(1),...,x(m))$$

Finite automata M_1 and M_2 are said to be equivalent if for any state $s_1 \in S_1$, there exists a state $s_2 \in S_2$ equivalent to s_1 and for any $s_2 \in S_2$, there exists $s_1 \in S_1$ equivalent to s_2.

Figure 15 shows a pair of equivalent automata with different number of states.



x=0	A	B
A	0	1
B	1	0

x=1	A	B
A	1	0
B	0	1

x=0	A	B	C
A	0	0	1
B	1	0	0
C	1	0	0

x=1	A	B	C
A	1	0	0
B	0	1	0
C	0	1	0

Figure 15. Graphical and tabular representation of two equivalent automata

One can make sure that the same input supplied to both automata produces the same output.

It is known that linear automaton is weak invertible with delay at most

$$\tau = \frac{|S|(|S|-1)}{2}, \tag{23}$$

where S is the automaton state alphabet [3].

Equation (23) shows that with increasing S the value of τ is being increased quadratically.

Thus, replacing the linear automaton M_l by an equivalent automaton with larger state alphabet, we can obtain longer delay τ. The above mentioned modification increases the resultant delay τ of the encryption automaton.

7. Conclusion

The presented finite automata public key cryptosystem is secure against the chosen plaintext attack and the exhaustive search attack. Security of FAPKC is mainly based on the growth of public key size due to increasing delays of component automata. The proposed modifications of both component nonlinear and linear automata complicate the process of breaking the cryptosystem allowing to design stronger FAPKC.

References

[1] Margarov, G. I., Chopuryan, S. H., Alaverdyan, Y. (2007). Fast Public Key Algorithm Based on Finite Automata, *In:* Proc. of the Int' Conf. on Computer Science and Information Technologies (CSIT'07), Yerevan, p. 112-115.

[2] Chopuryan, Y. (2009). The Stability of Trapdoor One Way Function Based on Finite Automata, *In:* Proc. of the 2009 Int' Conf. on Security & Management (WORLDCOMP'09), V.2, CSREA Press, Las Vegas Nevada, USA, p. 429-433.

[3] Arbib, M. A. (1969). *Theories of Abstract Automat,* Prentice-Hall, Englewood Cliffs, NJ, p.414.

[4] Tao, R. (2009). Finite Automata and Application to Cryptography, Springer Berlin Heidelberg, 2009

[5] Salomaa, A. (1985). Computation and Automata, Cambridge University Press, 1985, p.284.

[6] Schneier B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Second Edition. N.Y.: John Wiley & Sons, 1996, p.760.

[7] Garey, M.R., Johnson, D.S. (1979). Computer and Intractability (A Guide to the Theory of NP-completeness), W. H. Freeman and Co., San Francisco.

[8] Kohavi, Z. (1978). *Switching and Finite Automata Theory*. McGraw-Hill.

[9] Tao, R. J., Chen, S. H. (1997). A necessary condition on invertibility of finite automata, *Science in China, Ser. E,* p. 637-643.

[10] Levenshtein, S. H. (1962).On the Inversion of Finite Automata, *Doklady Akad. Nauk SSSR*, 147 (6) 1300-1303.

[11] Margarov, G., Chopuryan, S. (2009). Strong Finite Automata Public Key Cryptosystem, *In:* Proc. of the Second International Conference on the Applications of Digital Information and Web Technologies (ICADIWT'09), London, United Kingdom August 2009.

[12] Bao, F., Igarashi, Y (1995). Break Finite Automata Public Key Cryptosystem, *ICALP*, p. 147-158.