International Journal of Advanced Science and Technology
Vol.67 (2014), pp.33-42
http://dx.doi.org/10.14257/ijast.2014.67.04

Implementation FPGA of Public Key Cryptosystems Based on Finite State Machines Reconfiguration

Nguyen Huu Khanh Nhan

Ton Duc Thang University, Ho Chi Minh City, Viet Nam
nguyenhuukhanhnhan@tdt.edu.vn

Abstract

The method of the finite state machine (FSM) for public key cryptosystem is allows to reduce key's length of the cryptosystem without reducing cryptographic strength. A reconfigurable finite state machine is entered into public key cryptosystem's model. A reduced key is used for adjustment of the reconfigured finite state machine. Each adjustment of the reconfigurable model generates some finite state machines which sets process of the encryption/decryption. Software implementation includes the finite state machines generator and a translator for transfer the table description of the finite automaton to the hardware description language VHDL. This project was implemented on XStend board containing FPGA XC4010XL of Xilinx.

Key words: *FPGA, public key cryptosystem, FAPKC, finite automata*

1. Introduction

Public key cryptosystem based on finite automata has been proposed Chinese cryptographer Renji Tao [1] and was named FAPKC (Fnite Automaton Public Key Cryptosystem). Algorithm is based on the composition of two finite automatas with some initial memories and reversible states. Task degradation of finite automata composition into components is such a difficult task, as well as the product factoring of two large numbers [2]. Cryptosystem is FAPKC stream cipher does not require partitioning the plaintext and the block has a high speed (higher than RSA). There are a few modifications: FAPKC0 [1], FAPKC1 and FAPKC2 [3], FAPKC3 [4, 5] and FAPKC4 [6]. FAPKC can be used for both encryption and digital signature.

The disadvantages include the cryptosystem FAPKC large key size. For example, key length that provides resistance to the algorithm is achieved by using 512 bits - key RSA, 2792 bits for power FAPKC [2]. Furthermore, there is a problem generating random keys and equally as key space algorithm FAPKC given the description of properties of its elements. For practical use requires algorithm with generating the strongly coupled machines, allowing software and/or hardware implementation.

Automatic generation algorithm depends on the initial key of acceptable length. Initialization key is used to adjust tunable machine. Each setting tunable machine affects the encrypting machine that implements a cryptographic transformation.

Modifiability is achieved by combining the advantages of the proposed models of finite automata with means technology FPGA (Field Programmable Gate Array). Software implementation includes a generator and automatic translator, which allows to translate the description to hardware description language VHDL. The logical structure of a tunable machine implements some fixed setting the output function. The transition function is built through the transition function of basic state machines. In this regard, the design redundancy

ISSN: 2005-4238 IJAST

is obtained at the duplication level of elements that implement transition function. The project was implemented on XStend board containing FPGA XC4010XL of Xilinx.

2. Basic Concepts on Automata and Invertible Automata

As usual, for a finite set X, we denote by X^n the set of words of length n, with $n \hat{I}$ ¥ $_0$, and $X^0=\{\varepsilon\}$, where ε denotes the empty word. We will also use $X^* = \dot{E}_{n^3 0} X^n$ the set of all finite words, and X^ω will denote the set of infinite words [7].

Definition 2.1. *A finite automata is a quintuple $(X, Y, S, \delta, \lambda)$, where:*

- *X is a nonempty finite set called the input alphabet of the finite automaton;*
- *Y is a nonempty finite set called the output alphabet of the finite automaton;*
- *S is a nonempty finite set called the set of states of the finite automaton;*
- *δ is a function from $S \times X$ to S called the state transition function of the finite automaton;*
- *λ is a function from $S \times X$ to Y called the output function.*

Let $M = (X, Y, S, \delta, \lambda)$ be a finite automaton. The state transition function δ and the output function λ can be extended to words, i.e. elements of X^*, recursively, as follows:

$$d(s,e) = s$$

$$d(s, x_0x_1...x_n) = d(d(s, x_0)x_1x_2...x_n)$$

$$l(s,e) = e$$

$$l(s, x_0x_1...x_n) = l(d(s, x_0)x_1x_2...x_n)$$

where $s \in S$, $n \hat{I}$ ¥ and $x_0 x_1 \cdots x_n \in X^{n+1}$. In an analogous way, λ may be extended to X^ω. From these definitions it follows that one has, for all $s \in S$, $\alpha \in X^*$, and for all $\beta \in X^* \cup X^\omega$,

$$l(s, ab) = l(s,a)l(d(s,a),b). \quad (1)$$

An important class of finite automata, providing an infinite number of examples, is given by the following:

Definition 2.2. *Let $f : X^{h+1} \times Y^k \to Y$, with $h, k \in N$, and X, Y two nonempty finite sets. The finite automaton with (h, k)-order memory determined by f is the automaton $M_f = (X, Y, X^h \times Y^k, \delta_f, \lambda_f)$ defined by:*

$$l_f(< x_1x_2....x_h, y_1y_2...y_k >, x) = f(x_1x_2....x_hx, y_1y_2...y_k) =: y,$$

$$d_f(< x_1x_2....x_h, y_1y_2...y_k >, x) = < x_2....x_hx, y_2...y_ky),$$

for all $y_1 \ldots y_k \in Y^k$ and $x_0 x_1 \ldots x_hx \in X^{h+1}$. When $k = 0$, M_f is called the finite automaton with h- order input memory determined by f. When $h = 0$, M_f is called the finite automaton with k-order output memory determined by f. And, we will say that a finite automaton M is a finite automaton with (h, k) - order memory if $M = M_f$ for some function $f : X^{h+1} \times Y^k \to Y$.

A central notion, essential for cryptographic purposes, is the notion of invertibility. We start with a concept related to the determination of the inputs by the outputs.

Definition 2.3. *A finite automaton $M = (X, Y, S, \delta, \lambda)$ is said to be invertible with delay τ, where $\tau \in N_0$, if $\forall s, s' \in S$, $\forall x, x' \in X$, $\forall \alpha, \alpha' \in X^{\tau}$,*

$$l(s, x\alpha) = l(s', x'\alpha') \Rightarrow x = x'$$

That is, for any $s \in S$ and $\alpha \in X^{\tau}$, x can be uniquely determined by $\lambda(s, x\alpha)$.

Invertible automata should have inverses of some sort. The following definition introduces the apropriate concept that we will see is closely related to the previous one.

Definition 2.4. *Let $M = (X, Y, S, \delta, \lambda)$, $M' = (X, Y, S', \delta', \lambda')$ be two finite automata. A pair of states $(s', s) \in S' \times S$ is said to be a match pair with delay τ if the following condition holds*

$$\forall \alpha \in X^{w}, \exists \gamma \in X^{t} : l'(s', l(s, \alpha)) = \gamma\alpha .$$

Remark: In the previous definition one may replace X^{ω} by X^{*}, but then one must take into account that on the right one only gets the first $|\alpha| - \tau$ characters of α.

Proposition 2.5. *If (s', s) is a match pair with delay τ and $\beta = \lambda(s, \alpha)$ for some $\alpha \in X^{*}$, then $(\delta'(s', \beta), \delta(s, \alpha))$ is also a match pair with delay τ.*

Proof. Assume that *(s', s)* is a match pair with delay τ, and let *$\beta = \lambda(s, \alpha)$* for some $\alpha \in X^{*}$. Let $\alpha' \in X^{\omega}$. By (1), one has:

$$l'(s', l(s, a\alpha')) = l'(s', bl(d(s, a), \alpha'))$$

$$= l'(s', b)l'(d(s', b), l(d(s, a), \alpha')),$$

Since *(s', s)* is a match pair with delay τ, $\exists \alpha_1 \in X^{\tau}$ such that $\lambda'(s', \lambda(s, \alpha\alpha')) = \alpha_1\alpha\alpha'$. Therefore, $\alpha_1\alpha\alpha' = \gamma\alpha'$, where $\gamma \in X^{\tau+|\alpha|}$.

But, $\lambda'(s', \beta) \in X^{|\alpha|}$. So, $\lambda'(\delta(s', \lambda(s, \alpha), \lambda(\delta(s, \alpha), \alpha')) = \varphi\alpha'$, for some $\varphi \in X^{\tau}$. That is, *$(\delta(s', \beta), \delta(s, \alpha))$* is a match pair with delay τ.

Definition 2.6. *M' is called an inverse with delay τ of M, if $\forall s \in S$ and $\forall s' \in S'$, (s', s) is a match pair with delay τ. M' is called an inverse with delay τ, if M' is an inverse with delay τ of some finite automaton. M' is called an inverse, if M' is an inverse with delay τ, for some τ.*

Part of the important role of the automata determined by a function as defined above, in definition 2.2, is revealed by the following result.

Theorem 2.7. *If M is invertible with delay τ, then there exists a finite automaton with τ -order input memory M_f that is an inverse with delay τ of M.*

Proof. Suppose that *$M = (X, Y, S, \delta, \lambda)$* is invertible automaton with delay τ. Then $\forall s \in S$, $\forall x \in X$, $\forall \alpha \in X^{\tau}$, x can be uniquely determined by the value of *$\lambda(s, x\alpha)$*. Let $f : Y^{\tau+1} \rightarrow X$ be the function defined in the following way: if $\exists s \in S$, $\exists x \in X$, $\exists \alpha \in X^{\tau} : y_0 y_1 \ldots y_{\tau} = \lambda(s, x\alpha)$, then f is defined at $y_0 y_1 \ldots y_{\tau}$ by $f(y_0 y_1 \ldots y_{\tau}) = x$; otherwise one defines f arbitrarily. Let $M_f = (Y, X, Y^{\tau}, \delta_f, \lambda_f)$ be the finite automaton with τ - order input memory determined by f. To prove the claimed result, one must show that, for all $y_1 \ldots y_{\tau} \in Y^{\tau}$, for all $s \in S$ and for all $\alpha = x_0 x_1 x_1 \cdots \in X^{\omega}$, there exists an $\gamma \in X^{\tau}$, such that

$$\lambda_f(y_1 \ldots y_{\tau}, \lambda(s, \alpha)) = \gamma\alpha.$$

Putting:

$$s_0 = s, \; s_{i+1} = d(s_i, x_i),$$

$$z_i = l(s_i, x_i),$$

$$a_i = x_i x_{i+1} x_{i+2} \ldots$$

$$x'_i = f(y_i \ldots y_t z_0 \ldots z_{i-1})$$

$$g = x'_1 x'_2 \ldots x'_t,$$

One has that $\lambda(s, \alpha) = z_0 z_1 z_2 \ldots$, and (1) yields

$$l_f(y_1 \ldots y_t, l(s,a)) = l_f(y_1 \ldots y_t, z_0) l_f(y_2 \ldots y_t z_0, l(s_1, a_1))$$

$$= x'_1 l_f(y_2 \ldots y_t z_0, l(s_1, a_1))$$

$$= x'_1 x'_2 l_f(y_3 \ldots y_t z_0 z_1, l(s_2, a_2))$$

$$= \ldots$$

$$= x'_1 x'_2 \ldots x'_t l_f(z_0 z_1 \ldots z_{t-1}, l(s_t, a_t))$$

$$= g l_f(z_0 z_1 \ldots z_{t-1}, z_t) l_f(z_1 z_1 \ldots z_t, l(s_{t+1}, a_{t+1}))$$

$$= \ldots$$

$$= g f(z_0 z_1 \ldots z_{t-1}, z_t) f(z_1 z_2 \ldots z_t z_{t+1}) \ldots$$

But $z_i z_{i+1} \ldots z_{i+\tau} = \lambda(s_i, x_i x_{i+1} \ldots x_{i+\tau})$, and therefore it follows from the definition of f that $f(z_i z_{i+1} \ldots z_{i+\tau}) = x_i$, which finishes the proof.

It immediately follows that

Corollary 2.8. *M is invertible with delay τ if and only if there exist a finite automaton M′ such that M′ is an inverse with delay τ of M.*

A weaker form of invertibility is described in the following definition.

Definition 2.9. *A finite automaton M = (X, Y, S, δ, λ) is said to be weakly invertible with delay τ, with* $t \in \mathbb{N}_0$, *if*

$$\forall s \in S, \forall x_0 \ldots x_t, x'_0 \ldots x_t \in X^{t+1},$$

$$l(s, x_0 \ldots x_t) = l(s, x'_0 \ldots x'_t) \Rightarrow x_0 = x'_0$$

That is, for any $s \in S$, and any $x_i \in X$, with $i \in \{0, 1, \ldots, \tau\}$, x_0 can be uniquely determined by s and $\lambda(s, x_0 x_1 \ldots x_\tau)$.

Definition 2.10. *Let $M = (X, Y, S, \delta, \lambda)$ and $M' = (X, Y, S', \delta', \lambda')$ be two finite automata. M' is called a weak inverse with delay τ of M, if $\forall s \in S, \exists s' \in S$ such that (s', s) is a match pair with delay τ. M' is called a weak inverse with delay τ, if M' is a weak inverse with delay τ of some finite automaton. M' is called a weak inverse, if M' is a weak inverse with delay τ for some τ.*

Definition 2.11. *Let $M_1 = (X, Y, S_1, \delta_1, \lambda_1)$ and $M_2 = (X, Y, S_2, \delta_2, \lambda_2)$ - the two end automaton. Composition of automata M_1 and M_2 is a finite automaton $M = M_1 \circ M_2 = (X, Y, S_1 \times S_2, \delta, \lambda)$,*

where $\delta((s_1, s_2), x) = (\delta_1(s_1, x), \delta_2(s_2, \lambda_1(s_1, x)))$ *and* $\lambda((s_1, s_2), x) = \lambda_2(s_2, \lambda_1(s_1, x))$ *for any* $x \in X$ *and* $(s_1, s_2) \in S_1 \times S_2$. *Composition* $M_1 \circ M_2$ *is a structure corresponding serial connection machines* M_1 *and* M_2, *ie. the input automaton* M_2 *comes output automaton* M_1. *If* M_1 *is invertible with delay* τ_1, *and* M_2 *is reversible automatic delay* τ_2, *then the automaton* $M_1 \circ M_2$ *will have a delay* $\tau_1 + \tau_2$.

3. Description Cryptographic System FAPKC

Finite state machines, which will be considered in the future, have the form $M = (X, Y, S, \delta, \lambda)$, where $X = Y = Z_2^l - l$ - dimensional linear space over the field $\not\in_2 = \{0,1\}$. In practice, the typical value of $l = 8$ (so that encryption is performed byte), and the functions δ and λ are determined by the mapping $f: Y^t \times X^{r+1} \rightarrow Y$ and can be defined by the following formula:

$$y(i) = f(x_i, x_{i-1}, ..., x_{i-r}, y_{i-1}, ..., y_{i-t}), \quad i = 0,1,2,... \tag{2}$$

Automatic represented by formula (1) is called a finite automaton with the procedure memory (r, t), where $(x_{-1}, ..., x_{-r}, y_{-1}, ..., y_{-t})$ - initial state. If $t = 0$, then this machine is called a finite automaton with input memory of order r. Finite state machine defined by formula (2) is called linear if f is linear. For linear automaton, formula (2) takes on the following form

$$y(i) = \sum_{j=0}^{r} A_j x(i-j) + \sum_{j=1}^{t} B_j y(i-j), \quad i = 0,1,2,... \tag{3}$$

The coefficients $A_0, ..., A_r, B_0, ..., B_t$ are $l \times l$ matrix over the field $\not\in_2$, $x(i)$ - column vectors, $A_j x(i-j)$ - the usual vector matrix multiplication column. As M_1 machine uses a linear reversible automaton with input memory having a delay $\tau = r$, for which the formula (3) takes on the following form

$$M_1 : z(i) = \sum_{j=0}^{t} A_j y(i-j), \qquad i = 0,1,2,... \tag{4}$$

This machine is uniquely determined by the coefficients $A_0, ..., A_r$, Representing an $l \times l$ matrix over the field $\not\in_2$. If M_1 is reversible automaton with delay τ, it can be easily obtained from its inverse delay τ as follows

$$M_1^{-1} : y(i) = \sum_{j=0}^{t} P_j z(i+j) + \sum_{j=1}^{t} Q_j y(i-j) \tag{5}$$

For a nonlinear machine M_0 formula (2) has the following form

$$M_0 : y(i) = \sum_{j=0}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} B'_j x(i-j)x(i-j-1), \quad i = 0,1,2,... \tag{6}$$

Here the coefficients $B_0, ... B_r$ and $B_0', ..., B'_{r-1}$ is a matrix of $l \times l$ over the field $\not\in_2$, wherein the matrix to be invertible B_0 (this ensures that the zero delay). Then, multiplying both sides of equation (6) on the B_0^{-1}, we obtain the inverse automaton in the following form

$$M_0^{-1}: x(i) = B_0^{-1}\left(y(i) + \sum_{j=1}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} B_j' x(i-j)x(i-j-1)\right) \quad (7)$$

For every initial state $s_0 = (x(-1), x(-2), ..., x(-r))$ automaton M_0 consistent state automaton M_0^{-1} are also equal to $(x(-1), x(-2), ..., x(-r))$.

Automatic encryption M, representing the composition of automata M_0 and M_1, may be obtained by substituting (4) into (6) and written in the following form.

$$M: z(i) = \sum_{t=0}^{t} A_t \left(\sum_{j=0}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} B_j' x(i-j)x(i-j-1)\right) \quad (8)$$

Any state $s = (x(-1), x(-2), ..., x(-r-\tau))$ automaton $M = M_0 \circ M_1$ equivalent aggregate (s_0, s_1) state $s_0 = (x(-1), ..., x(-r))$ and $s_1 = (y(-1), ..., y(-\tau))$. Formula (8) can be simplified to (9)

$$M: z(i) = \sum_{j=0}^{r+t} C_j x(i-j) + \sum_{j=1}^{r+t-1} C_j' x(i-j)x(i-j-1), i = 0,1,2... \quad (9)$$

$$C_j = \sum_{h+t=j} A_h B_t, \quad j = 0,1,2,...,r+t$$

$$C_j' = \sum_{h+t=j} A_h B_t', \quad j = 1,2,...,r+t-1$$

FAPKC algorithm consists of the following steps

1. Choose M_0 and M_1. All A_j, B_j and B'_j kept well as a secret key.

2. Compute C_j and C_j' of A_j, B_j and B_j', then randomly choose $s = (x(-1), x(-2), ..., x(-r-\tau))$ as the initial state. Making C_j, C_j' with the public.

3. To encrypt the plaintext $x(0)\, x(1) ... x(m)$, first choose arbitrary $x(m+1) ... x(m+\tau) \in X^{\tau}$. Then give $x(0)\, x(1) ... x(m)\, x(m+1) ... x(m+\tau)$ to the input automaton $M = M_0 \circ M_1$ with initial state s. Exit $z(0)\, z(1) ... z(m)\, z(m+1) ... z(m+\tau)$ would be a ciphertext.

4. To decrypt $z(0)\, z(1) ... z(m)\, z(m+1) ... z(m+\tau)$ must first be use automatic M_1^{-1} and s_1 for $y(0), ..., y(m)$, and then transferred $y(0), ..., y(m)$ at the input automaton M_0^{-1} with the initial state s_0, to get the output plaintext.

4. Model of Tunable Machine

As mentioned above, the private key of the components $A_0, ..., A_r, B_0, ..., B_t$ represents an $l \times l$ matrix over the field $\mathbb{Z}_2$. And the component to the public key C_j, C'_j are matrix polynomials, the size of which also depends on the parameters l and τ.

Computational complexity of the expansion machine for encrypting machines M_0 and M_1 equal to $2^{\frac{l(t+1)}{2}}$ [8]. When $l = 8$, $\tau > 15$ complexity 2^{64} . Therefore cryptosystem FAPKC plaintext encrypted nonlinear reversible automaton with delay $\tau > 15$. Due this increases the length key cryptosystem, as is seen from Table 1, here for some values l, r_1 and r_2 corresponding dimensions are shown in bits N_1 and N_2 of public key FAPKC with $\tau_2 \le r_2 = \mu(f_2)$, $\tau_1 \le r_1 = \mu(f_1)$, respectively , and linear and nonlinear function f_1 [9,11].

Table 1. Dependence of the size of the key parameters of the cryptosystem FAPKC [9]

(l)	7	7	5	5	3	3	3
(r_2, r_1)	$(1, 14)$	$(7, 8)$	$(1, 19)$	$(10, 10)$	$(1, 34)$	$(10, 25)$	$(17, 18)$
(N_1)	8281	32948	4075	20950	1593	8883	13041
(N_2)	105840	414512	29850	181725	5400	34560	51192

Necessary is for practical to use of the cryptosystem FAPKC, on the one hand, to keep the size of the public key within acceptable limits, on the other hand, does not reduce cryptosystem parameters, thereby lowering the Cipher.

To solve this problem, we propose to use the model of a tunable automaton [8].

Automatic is tunable if its transition and output functions are not only depend on the input alphabet and the set of states, but also on a parameter $k \in K$, where K - finite set of settings.

Definition 4.1. Tunable machine is six $M = (X, Y, S, K, \delta, \lambda)$, where the input alphabet X, output alphabet Y, alphabet of the S and K are setting non-empty finite sets , and the transition function $\delta: S \times X \times K \to S$ and output $\lambda: S \times X \times K \to Y$ - valued functions . A set tunable specifies of automatic machines Mealy $\{A_k = (X, Y, S, \delta_k, \lambda_k): k \in K\}$, where $\delta_k(s, x) = \delta(s, x, k)$, $\lambda_k(s, x) = \lambda(s, x, k)$, for all $s \in S$, $x \in X$ and $k \in K$.

In [18] it is shown that a finite automaton with modifiable behavior can be created based on statically or dynamically reconfigurable matrix FPGA using blocks of memory. Cascade model proposed by the reprogrammable finite automaton consists of two blocks of memory, register and programmable multiplexer. To configure the machine using a tunable initialization key, which is a boolean vector.

Let us consider an example. Fig. 1 depicts four state transition graphs for FSMs that permit to perform the following operations with Boolean vectors of size S:

a) Detecting three or more successive ones in the Boolean vector;

b) Counting the number of ones in the Boolean vector;

c) Testing if the vector contains just one position with value "1" and returning an index of this position in the counter (see also fig. 1). If vector does not satisfy this requirement the counter is set to "0";

d) Testing if the vector contains either odd (in this case counter =1) or even (in this case counter =0) number of values "1".

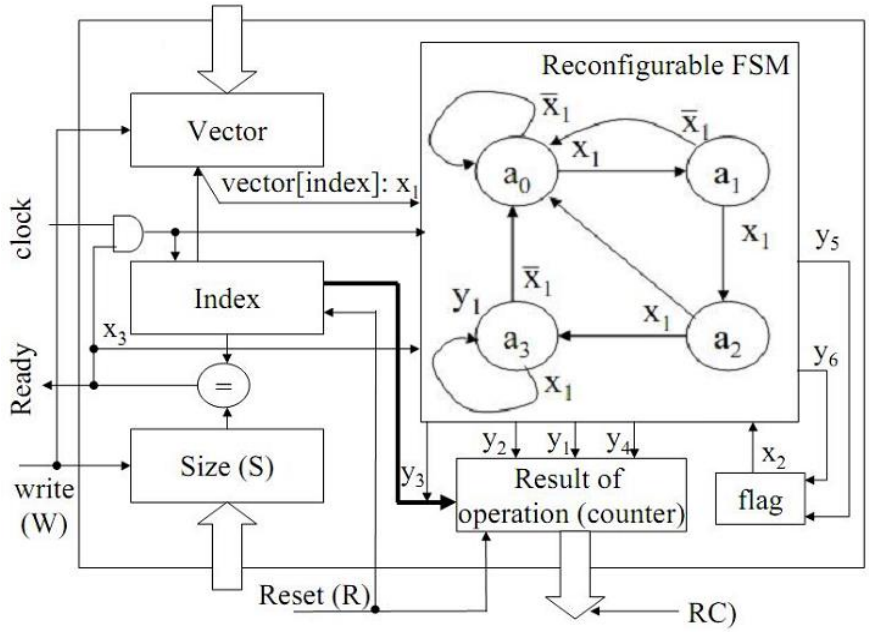

Figure 1. A circuit that detects three or more successive ones in Boolean vector

The structure in Figure 1 can be modeled by the following C++ class, which we call Boolean_vector:

```
class Boolean_vector
{   public:
      unsigned Solve(FSM_template&);
     unsigned Run(unsigned);
     Boolean_vector(unsigned V=0, int S=0) :
     vector(V),size(S), index(0), counter(0), flag(0) {};
     virtual ~Boolean_vector();
     // other functions
   protected:
     unsigned vector;
     int index;
     int size;
     int counter;
     int flag;
```


5. Implementation of Cryptosystems FAPKC

Cryptosystem values also affect the amount of computation involved with the generation of weakly nonlinear reversible automata. Generating technique of nonlinear reversible automaton is suitable and invulnerable species to attack with the chosen plaintext is described in [6, 10].

Using this method, and a class library functions for C++, implementing basic and derivative operations in various groups, rings, fields, designed generator ciphering machines for cryptosystems FAPKC3. The result of the generator are the values of public and private keys, as well as tables of states and transitions ciphering machines.

To move from the abstract machine to a structural part in the program implemented encryption algorithm encrypts the state machine. Each setting tunable machine generates an automaton that specifies the process encryption/ decryption. According to the results of the synthesis system allows to receive output code in VHDL. Subsequent automatic synthesis and final implementation on FPGA implemented by StateCAD ISE Xilinx.

The logical structure of the proposed hardware implementation cryptosystem FAPKC modeled using two machines work together, one (encoded machine) of which is rigidly fixed behavior, and the behavior of other tunable machine specified by the user using an initialization key. Output function system is fixed and the transition function is constructed from two transition functions automata. In this regard, the design and implementation on XStend board containing FPGA XC4010XL of Xilinx level redundancy overlapping elements implementing the transition function.

As already mentioned, the shortcomings can be attributed cryptosystem FAPKC large size of the keys. At the same time a large amount of computation to generate ciphering machine, and the need for frequent rekeying make cryptosystems finitely automata models unsuitable for practical widespread use. Implementation of the proposed model with adjustable machine allows use once generated automatic encryption for a long time with frequent change shorter initialization key.

Configuring each machine $k \in K$ automaton $M = (X, Y, S, K, \delta, \lambda)$, in correspondence one-to-one put vector values of the transition δ_k: $S \times X \times K \rightarrow S$. It is a Boolean vector of length $|S \times X|$, so the key length of the proposed implementation FAPKC cryptosystem does not exceed the number mn, Herein $n = |S|$, $m = |X| = |Y|$. For example, when m = 32, n = 20, this number is 640 bits.

6. Conclusions

The paper presents a novel technique for the design of FSMs with statically and dynamically modifiable behavior and demonstrates the use of such FSMs for finite automaton public key cryptosystem. It is shown that reconfigurable FSM can be constructed in such a way that it might be used for reducing the length of the key cryptosystems preserving stability. The paper examines some models of reconfigurable FSMs and demonstrates their implementation in software and in hardware. The results of hardware implementation based on FPGA XC4010XL of Xilinx have shown that the respective circuits require very limited FPGA resources and they can be reprogrammed much like we are doing this for software development.

References

[1] R. C. Tao and S. H. Chen, “A Finite Automaton Public Key Cryptosystem and Digital Signatures”, Chinese J. of Computer, no. 8, **(1985)**, pp. 401-409.

[2] B. Schneier, "Applied Cryptography: Protocols, Algorithms, and Source Code", John Wiley & Sons, New York, **(1996)**.
[3] R. J. Tao and S. H. Chen, "Two Varieties of Finite Automaton Public Key Cryptosystem and Digital Signatures", J. of Compt. Sci. and Tech., no. 1, **(1986)**, pp.9-18.
[4] R. J. Tao and S. H. Chen and X. M. Chen, "FAPKC3: a new finite automaton public key cryptosystem", Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100080, China, ISCAS-LCS-95-07, **(1995)**.
[5] T. Meskanen, "On Finite Automaton Public Key Cryptosystems", TUCS Technical Report, no. 408, **(2001)** August.
[6] R. J. Tao and S. H. Chen, "The generalization of public key cryptosystem FAPKC4", Chinese Science Bulletin, vol. 44, no. 9, **(1999)**, pp. 784-790.
[7] A. Ivone, M. Antonio, R. Rogerio, "On Linear Finite Automata and Cryptography", Technical Report Series: DCC-2011-11, Version 1.0, August **(2011)**.
[8] G. Xiang, "Finite automaton public key cryptosystems and digital signatures - analysis, design and implementation", Dissertation (in Chinese), Institute of Software, Chinese Academy of Sciences, Beijing, **(1994)**.
[9] V. Sklyarov, "Reconfigurable models of finite state machines and their implementation in FPGAs", Systems Architecture, no. 47, **(2002)**, pp. 1047-1064.
[10] P. Kitsos, N. Sklavos, M. Galanis and O. Koufopavlou, "64-bit ciphers: hardware implementations and comparison analysis", Computer and Electrical Engineering, no. 30, **(2004),** pp. 593-604.
[11] P. Vishwanath, R. Joshi and J. Saxena, "FPGA implementation of DES using pipeling concept with skew core key-scheduling", Journal of Theoretical and Applied Information Technology, no. 3, **(2009)**, pp. 295-300.

Author

Nguyen Huu Khanh Nhan

He received his B. Eng. degrees in Electrical and Electronic Engineering from University of Technical education Ho Chi Minh City, Vietnam in 1991- 1996, and received his M. Eng. degrees in Nano materials and electronic devices from Ho Chi Minh City National University, Vietnam in 2005 – 2007, and Studied PhD. degree at Institute of researchs and experiments for electrical and electronic equipments, Mosscow – Russia in 2012. Now, He is teaching at Department of electrical and electronics engineering, Ton Duc Thang University, Ho Chi Minh city, Vietnam. His research interests include VLSI, MEMS and RF chip.

