Implementation of the Finite Automaton Public Key Cryptosystem on FPGA

Dina Satybaldina, Altynbek Sharipbayev and Aigul Adamova

L. Gumilyov Eurasian National University, Munaitpasov str. 5
010000, Astana, Kazakhstan

Abstract. Hardware implementation aspects of the finite automaton public key cryptosystem are discussed in this paper. A general architecture of the multiplication of a square matrix on a vector over GF(q) is presented in the paper. Our design was implemented on Altera EP3C5E144C8N of the Cyclone III FPGA family. The performance of finite automaton public key cryptosystems is mainly appointed by the efficiency of the underlying finite field arithmetic. The results are compared with reported reconfigurable hardware implementations of RSA. Proposed hardware realization of cryptographic system allows organizing pipeline calculations.

1 Introduction

Today there are many distributed systems, which use communication resources that must be safeguarded against eavesdropping or unauthorized data alteration. Thus cryptographic protocols are applied to these systems in order to prevent information extraction or to detect data manipulation by unauthorized users. Besides the widely-used RSA method [1], other public-key schemes have gained more and more importance in this context.

The public key cryptosystem on the basis of finite state machines (FSM) has been offered by Tao Renji [2] and was named FAPKC (Finite Automaton Public Key Cryptosystem). The private key consists of two FSM that are constructed so that their inverses are easily calculated. The public key is the automaton which we get by combining these two automata. It is believed to be hard to invert this combined automaton without knowledge of the private key automata [1].

FAPKC cryptosystem is a stream ciphe. There are some modifications: FAPKC0 [2], FAPKC1 and FAPKC2 [3], FAPKC3 [4,5] and FAPKC4 [6]. The cryptosystem can be used as for enciphering (with using public key) and for decryption (with using FSM, which is reversed to public key), and for signing (by this FSM) and for check of the signature (by public key) [6, 7].

Cryptographic transformations can be implemented in both software and hardware [8]. Software implementations are designed and coded in programming languages, such as C, C++, Java, and assembly language, to be executed, among others, on general purpose microprocessors, digital signal processors, and smart cards. Hardware implementations are designed and coded in hardware description languages, such as VHDL and Verilog HDL, and are intended to be realized using two major

Satybaldina D., Sharipbayev A. and Adamova A..
Implementation of the Finite Automaton Public Key Cryptosystem on FPGA.
DOI: 10.5220/0003594301670173
In *Proceedings of the 8th International Workshop on Security in Information Systems* (WOSIS-2011), pages 167-173
ISBN: 978-989-8425-61-4

implementation approaches: application-specific integrated circuits (ASICs) and field programmable gate arrays (FPGAs).

In this paper we describe an implementation method of FAPKC. The method is based on realization of computational structure in GF(q). A project was implemented on the programmed matrix FPGA EP3C5E144C8N of the device family Cyclone III offered by Altera.

2 FAPKC Principles

2.1 Definitions and Denotations

Let us introduce the basic terminology [5]:

Definition 1. *Finite automaton* M – a quintuple $<X, Y, S, \delta, \lambda>$, where we have the input alphabet X, the output alphabet Y and the state alphabet S which all are nonempty finite sets and the transition function $\delta: S\times X\to S$ and the output function $\lambda: S\times X\to Y$ which are single valued functions.

If we denote by A^N the set of all finite words of alphabet A, by A^ω the set of all infinite words of alphabet A and by ϵ the empty word, we can expand the domains of δ and λ to $S\times X^N$ and $S\times (X^N \cup X^\omega)$, respectively, as follows:

$$\delta(s, \epsilon) = s, \qquad \delta(s, \alpha x) = \delta(\delta(s, \alpha), x), \tag{1}$$

$$\lambda(s, \epsilon) = \epsilon, \qquad \lambda(s, x\alpha') = \lambda(s, x)\,\lambda(\delta(s, x), \alpha'), \tag{2}$$

where $s \in S$, $x \in X$, $\alpha \in X^N$ and $\alpha' \in X^N \cup X^\omega$.

Definition 2. Let $M = <X,Y,S,K,\delta,\lambda>$ be a finite automaton, X and Y are column vector spaces over GF(q) of dimension l and m, respectively, and τ be a nonnegative integer. M is *a weakly invertible with delay* τ if, for any $x_i \in$ X, i =0, 1, …, τ and $s \in S$, x_0 can be uniquely determined by the state s and the output $\lambda(s, x_0 \ldots x_\tau)$.

For any states s ∈ S and s′ ∈ S, if $\forall\alpha \in X^\omega$, $\exists\alpha_0 \in X^N$:

$$\lambda'(s', \lambda(s, \alpha)) = \alpha_0\alpha \quad \text{and} \quad |\alpha_0| = \tau, \tag{3}$$

then (s', s) is *a matching pair* with delay τ *or* s' *matches* s *with delay* τ .

Definition 3. Let $M = <X, Y, S, \delta, \lambda>$ and $M' = <Y, X, S', \delta', \lambda'>$ be two finite automata and τ be a nonnegative integer. M' is *a weak inverse with delay* τ *of* M if, for any $s \in S$, there exists s' in S' that (s', s) is a matching pair with delay τ.

Definition 3. Let $M_1 = <X, Y, S_1, \delta_1, \lambda_1>$ and $M_2 = <X, Y, S_2, \delta_2, \lambda_2>$ be two finite automata. *Superposition* of M_1 and M_2 is a finite automaton $C(M_1,M_2) = <X_1, Y_2, S_1 \times S_2, \delta, \lambda >$, where

$$\delta(<s_1, s_2>, x) = <\delta_1(s_1, x), \delta_2(s_2, \lambda_1(s_1, x))> \tag{4}$$

$$\lambda(< s_1, s_2 >, x) = \lambda_2(s_2, \lambda_1(s_1, x)), \tag{5}$$

for any $x \in X_1$ and $s_1 \in S_1$, $s_2 \in S_2$.

Definition 4. Let φ be a mapping from $Y^k \times X^{h+1}$ to Y. This mapping defines $M_\varphi = <X,Y,Y^k \times X^h,\delta,\lambda>$ to be *an (h, k)-order memory finite automaton* if

$$y(i) = \varphi(y(i-1), \ldots, y(i-k), x(i), \ldots, x(i-h)),\ i = 0, 1, \ldots \tag{6}$$

If the mapping φ is from X^{h+1} to Y then M_φ is *an h-order input memory finite automaton*.

2.2 Brief Overview of FAPKC Cryptosystems

In the paper [5] it was shown how to break the FAPKC cryptosystem if not a suitable nonlinear automaton is used. From the mathematical point of view of rules and conditions finding of invertible nonlinear automata of the suitable form and superposition of its are in detail considered in [5,9]. Therefore here we will consider only the cryptographic algorithm steps for encryption/decryption without mathematical calculations.

Let X and Y be column vector spaces over GF(q), with dimension l.

A FAPKC3 user A chooses his own public key and secret key as follows:

(1) Construct two automata: an (h_0, k_0)-order memory finite automaton $M_0 =<X, Y, S_0, \delta_0, \lambda_0>$ and a (τ_0 + k_0, h_0)-order memory finite automaton $M_0^* =<Y, X, S_0^*, \delta_0^*, \lambda_0^*>$.

(2) Construct an h_1-order input memory finite automaton $M_1 ==<X, X, S_1, \delta_1, \lambda_1>$ and a (τ_1, h_1)-order memory finite automaton $M_1^* ==<X, X, S_1^*, \delta_1^*, \lambda_1^*>$.

(3) Construct the finite automaton $C'(M_1,M_0) =< X, Y, S, \delta, \lambda >$ from M_1 and M_0.

(4) Denote $\tau = \max(\tau_0, \tau_1, h_0)$.

Choose an arbitrary state $s_e = \langle y_{-1,e} \dots y_{-k_0,e}, x_{-1,e} \dots x_{-h_0-h_1,e} \rangle$ of C' (M_1,M_0) to be the starting state for encryption. Compute

$$x'_{-h_0,e} \dots x'_{-1,e} = \lambda_1(\langle x_{-h_0-1,e}, \dots x_{-h_0-h_1,e} \rangle, x_{-h_0,e} \dots x_{-1,e} \tag{7}$$

Define parts needed in decryption

$$s_{1,d}^{out} = \langle x_{-1,e}, \dots, x_{-h_1,e} \rangle \tag{8}$$

and

$$s_{0,d}^{out} = \langle x'_{-1,e}, \dots, x'_{-h_0,e} \rangle \tag{9}$$

(5) The public key of the user A is $\{C'(M_1,M_0), s_e, \tau_0 + \tau_1\}$.

The secret key of the user A is $\{M_0^*, M_1^*, s_{0,d}^{out}, s_{1,d}^{out}, \tau_0, \tau_1\}$.

Encryption. When user B wants to send to user A a plaintext $x_0 \dots x_n$ in secret, he first adds some $\tau_0 + \tau_1$ letters $x_{n+1} \dots x_{n+\tau 0+\tau 1}$ to the end of the plaintext. Then he can compute the ciphertext using A's public key:

$$y_0 \dots y_{n+\tau_0+\tau_1} = \lambda(s_e, x_0 \dots x_{n+\tau_0+\tau_1}). \tag{10}$$

Decryption. After receiving the ciphertext $y_0 \dots y_{n+\tau_0+\tau_1}$, user A first computes using parts M_0^* and $s_{0,d}^{out}$ of his secret key and part $y_{-1,e} \dots y_{-k_0,e}$ of his public key

$$x'_0 \dots x'_{n+\tau_1} = \lambda_0^*(\langle x'_{-1,e}, \dots x_{-h_0,e}, y_{\tau_0-1}, \dots, y_0, y_{-1,e} \dots y_{-k_0,e} \rangle, y_{\tau_0} \dots y_{n+\tau_0+\tau_1}). \tag{11}$$

Then using parts M_1^* and $s_{1,d}^{out}$ of his secret key user A retrieves the plaintext

$$x_0 \dots x_n = \lambda_1^*(\langle x_{-1,e}, \dots x_{-h_1,e}, x'_{\tau_1-1}, \dots, x_0 \rangle, x'_{\tau_1} \dots x'_{n+\tau_1}). \tag{12}$$

As present in [5] automaton M_0 is in the form of

$$y(i) = \sum_{j=1}^{k_0} A_j y(i-j) + \sum_{j=0}^{h_0} B_j x(i-j), \quad i = 0,1, \dots \tag{13}$$

and M_0^* in the form of

$$x(i) = \sum_{j=1}^{h_0} A_j^* x(i-j) + \sum_{j=0}^{\tau_0+k_0} B_j^* y(i-j), \quad i = 0,1,\ldots \quad (14)$$

The nonlinear automaton M_1 is in the form of

$$x'(i) = \sum_{j=0}^{h_1} F_j \; x(i-j) + \sum_{j=0}^{h_1+\epsilon} F_j^* s\big(x(i-j),\ldots,x(i-j-\epsilon)\big), \quad i = 0,1,\ldots \quad (15)$$

where $s\big(x(i-j),\ldots,x(i-j-\epsilon)\big)$ is a nonlinear function from $X^{\epsilon+1}$ to X, ϵ is a small positive integer.

Then $C'(M_1,M_0)$ can be expressed by

$$y(i) = \sum_{j=1}^{k_0} A_j y(i-j) + \sum_{j=0}^{h_0+h_1} C_j x(i-j) + \sum_{j=0}^{h_1+h_0-\epsilon} C_j' s\big(x(i-j),\ldots,x(i-j-\epsilon)\big), \quad i = 0,1,\ldots \quad (16)$$

where

$$C_j = \sum_{\substack{k+n=j,\\ 0\le k\le h_0,\\ 0\le n\le h_1}} B_k F_n \quad (17)$$

and

$$C'_j = \sum_{\substack{k+n=j,\\ 0\le k\le h_0,\\ 0\le n\le h_1}} B_k F'_n \quad (18)$$

Parameters h_0, h_1, k_0, τ_0 and τ_1 may be chosen by users.

3 FPGA Implementation of FAPKC

3.1 Hardware Implementation Environment

In this paper FAPKC3 was implemented on the programmed matrix Altera FPGA of the device family Cyclone III (EP3C5E144C8N). This Cyclone III device has a total of 5,136 logic elements, 23 multipliers, 2 phase-locked loops (PLLs) [10]. Quartus II Web Edition software includes an integrated development environment to accelerate system-level design and seamless integration with leading third-party software tools and flows [11].

3.2 Implementations Details

The equations show that multiplication of a square matrix on a vector is the basic operation in algorithm FAPKC3. Matrix $Ar[n, n]$, a vector $V[n]$, dimension of a matrix are input parameters for the given procedure. $P[n]$ is a resultant vector. Each element of a resultant vector calculated by search of elements of lines of matrix Ar and their multiplication to corresponding elements of a vector V. The result of multiplication increases to current value of an element of vector P. Each element of a resultant vector is calculated by the following formula:

$$P_i = \sum_{j=1}^{n} Ar_{ij} \cdot V_j \quad (19)$$

The information graph of calculation realization according to the formula (19) is presented in Figure 1.

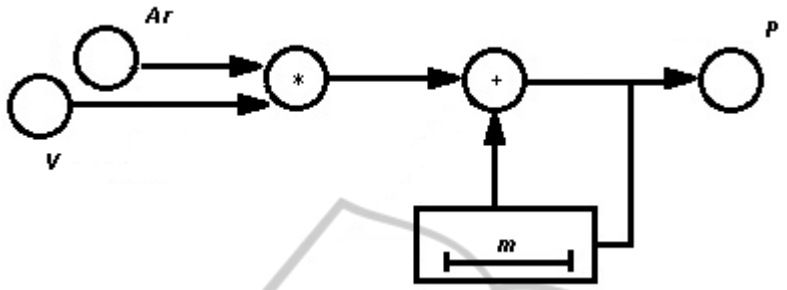

Fig. 1. The graph of multiplication of a matrix on a vector-column. *Ar* and *V* – input information tops, *P* – output information top, *m* – depth of a delay line.

Depth of a delay line is equal to m, size of a vectors stream. Thus for the stream of vectors *V* we have:

$$V = \langle v_1^1\ v_2^1 v_3^1 \dots v_m^1 v_1^2\ v_2^2 v_3^2 \dots v_m^2 \dots v_1^n\ v_2^n v_3^n \dots v_m^n \rangle, \qquad (20)$$

where the top index means a vector's element, the bottom index means number of vector in the stream.

Thus, elements streams of matrix and vectors from information tops *Ar* and *V* accordingly are put on multiplier inputs. Further, the result of multiplication is added with earlier keeped value of an element of vector *P* from the delay line.

The structure of block P, which is used for multiplication of a matrix on vectors stream, is presented on Figure 2. CR blocks provide communication of the computing scheme with memory cells from which commands and the data are read out and written. CR1 is designed for storage and getting of commands and lines elements of a matrix, and also for writing of calculation results from CalB. CR2 it is designed for storage and getting of vectors stream.

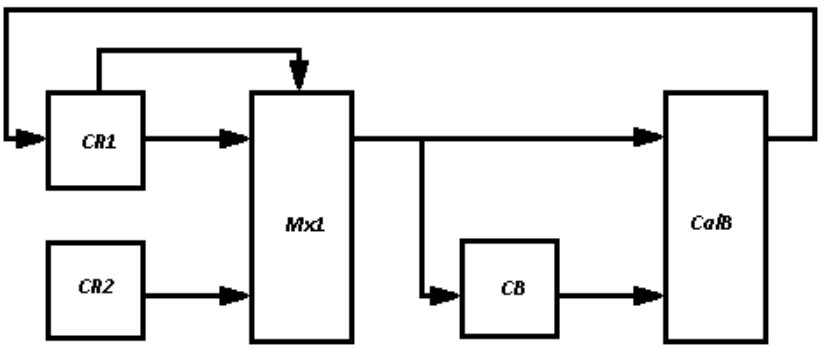

Fig. 2. Structure of calculator P: CR1, CR2 – controllers of the memory; Mx1 – the multiplexer of choice CR1 or CR2; CB – the control block; CalB – the calculation block.

Multiplexer Mx1 is used for switching of the input data flows to CalB. At each moment of time on CalB arrives:

- elements of a matrix line for their preservation in the internal buffer (loading of elements of a matrix),
- or elements of stream vectors for computing structure (performance of calculations). Commands to CB are arrive from CR1.

CB is designed for decoding of commands and formation of control signals for management of calculations by CalB. The structure of CalB block is presented on Figure 3.

Buffer BUF1 is used for storage of a matrix line. Multiplier MUL, adder ADD and a delay line BUF2 realize calculations according to an information graph as shown on Figure 1. Multiplexer Mx2 is intended for switching of the input data flows to CalB. Data flows represent streams of matrix lines, streams of vectors and streams of results. Multiplexer Mx1 is intended for switching of streams of intermediate results and results from previous CalB, which come into buffer BUF2. Thus, hardware realization of set CalB allows organizing pipeline calculations.

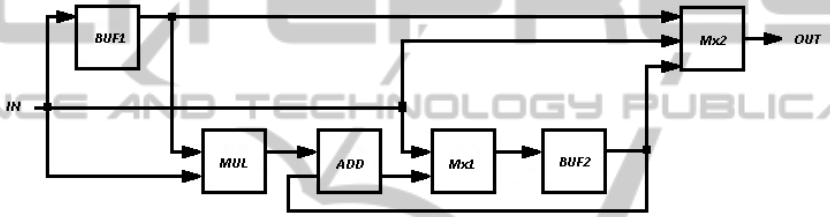

Fig. 3. The structure of calculatoin block CalB.

3.3 FPGA Implementations Results

The proposed architecture was captured by using VHDL. All the system components were described with structural architecture. The system tested using confirmed test vectors [5] in order to examine its correctness. The whole design was synthesized, placed and routed by using Altera FPGA device (Cyclone III, EP3C5E144C8N) [10]. Synthesis results for the proposed implementation are obtained as follows: 4,314 logic elements used (84% from available area) at a maximum frequency of 45 MHz and data rate of 24,3 Kb/s.

The maximum operating frequency F as well as the number of logic elements required for our implementation was obtained from the Quartus II Web Edition software. Comparison between the proposed FAPKC implementation and a 512-bit RSA implementations are presented in Table 1. Area resource comparisons are not given because they are not provided by the other implementations.

Table 1. Public key cryptosystem implementations comparison.

Public key cryptosystem	*F* (MHz)	Data rate (Kb/s)
RSA in [12]	45.6	140-460
RSA in [13]	100	100
RSA in [14]	50	43
Proposed	45 MHz	24,3

4 Concluding Remarks

FAPKC implementation oriented on multiplication of a square matrix on a vector over GF(q) on the reprogrammable matrix is offered in this paper. FAPKC3 was implemented on Altera FPGA of the device family Cyclone III (EP3C5E144C8N).

The proposed system achieves a data throughput up to 24,3 Kb/s in a clock frequency of 45 MHz. The most part of an integrated circuit area (84%) was used in our FPGA implementation of FAPKC3 with small parameters ($q = 2$, $l = 3$, $h_0 = 1$, $k_0 = 2$ and $\tau_0 = 1$).

Thus, a future work will to concern of the implementation of FAPKC with the large parameters. The most obvious extension is to design a fully pipelined architecture in order to obtain a higher throughput at the price of area.

References

1. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. John Wiley & Sons, New York (1996).
2. Tao R. C. and Chen S. H.: A Finite Automaton Public Key Cryptosystem and Digital Signatures. Chinese J. of Computer 8 (1985) 401-409.
3. Tao R. J. and Chen S. H.: Two Varieties of Finite Automaton Public Key Cryptosystem and Digital Signatures. J. of Compt. Sci. and Tech.1 (1986) 9-18.
4. Tao R. J. and Chen S. H. and Chen X. M. FAPKC3: a new finite automaton public key cryptosystem. ISCAS–LCS–95–07. Chinese Academy of Sciences, Beijing (1995).
5. Meskanen, M.: On Finite Automaton Public Key Cryptosystems. TUCS Technical Report, 408 (2001).
6. Tao, R. J. and Chen S. H.: The generalization of public key cryptosystem FAPKC4. Chinese Science Bulletin, Vol. 44 (1999) 784-790.
7. Xiang, G.: Finite automaton public key cryptosystems and digital signatures - analysis, design and implementation. Dissertation (in Chinese). Institute of Software. Chinese Academy of Sciences, Beijing (1994).
8. Çetin Kaya Koç (ed.): Cryptographic Engineering. Springer Science+Business Media, New York (2009).
9. Tao, R. J.: Finite Automata and Application to Cryptography. Jointly Published with Tsinghua University Press (2009).
10. Cyclone III device handbook. Available at the web site “http:// www.altera.com /literature/lit-cyc3.jsp”.
11. Quartus II handbook v10.1.0. Available at the web site “http:// www.altera.com /literature/lit-qts.jsp”
12. Blum, T. and Paar, C.: High–Radix Montgomery Modular Exponentiation on Reconfigurable Hardware. IEEE Transactions on Computers, Vol. 50, No. 7 (2001) 759 - 764.
13. Chih-Yuang Su, Shih-Arn Hwang, Po-Song Chen, and Cheng-Wen Wu: An Improved Montgomery’s Algorithm for High-Speed RSA Public-Key Cryptosystem. IEEE Transaction on Very Large Scale Integration (VLSI) Systems, Vol. 7, No. 2 (1999) 280 - 284.
14. Chen, P. S., Hwang, S. A. and Wu, C. W.: A systolic RSA public key cryptosystem. In Proceedings of International Symposium of Circuit and System (ISCAS’96), Vol. 4 (1996) 408-411.