Break Finite Automata Public Key Cryptosystem

Feng Bao and Yoshihide Igarashi

Department of Computer Science,
Gunma University, Kiryu, 376 Japan

Abstract. In this paper we break a 10-year's standing public key cryptosystem, Finite Automata Public Key Cryptosystem(FAPKC for short). The security of FAPKC was mainly based on the difficulty of finding a special common left factor of two given matrix polynomials. We prove a simple but previously unknown property of the input-memory finite automata. By this property, we reduce the basis of the FAPKC's security to the same problem in **module** matrix polynomial rings. The problem turns out to be easily solved. Hence, we can break FAPKC by constructing decryption automata from the encryption automaton(public key). We describe a modification of FAPKC which can resist above attack.

1 Introduction

1.1 About FAPKC

Finite Automaton Public Key Cryptosystem, denoted by FAPKC, is a public key cryptosystem based on the invertibility theory of finite automata. For the invertibility theory of finite automata, the reader may be referred to [1,3,5-9,11,17-19,22]. FAPKC is a 10-year's standing public key cryptosystem that first appeared in [20], and its modified versions were given in the following year [21]. The identity-based finite automaton public key cryptosystem and digital signature schemes were proposed in CRYPTO-CHINA'92 [23]. Since FAPKC is a stream cipher, it possesses an advantageous property that plaintexts need not be divided into blocks. Its speed is very fast(much faster than RSA), and its key size is comparable with RSA's [12] [24]. FAPKC can be easily implemented. since it includes only logic operations. FAPKC was practically used in some local networks in China, and the product of digital signature based on FAPKC was developed for practical use. Although almost all the relevant references about FAPKC were published in China, the copies of [21] were distributed at ICALP'88 when Tao, the inventor of FAPKC, presented [22] at the conference. FAPKC was also mentioned in Section 5.3 of Salomaa's book [16].

1.2 Weakly Invertible Finite Automata

Since the relevant references are not conveniently accessed by the reader, we try to make this paper self-contained. Invertible finite automata were first proposed by Huffman as "information-lossless finite state logical machines" [11]. Since then, they have been much studied [1,3,8,9,13-15,25]. In [8] and [9], "generalized finite automata" were used instead of "sequential machines" and "logic machines". Reference [17] is a book on this subject, in which both invertible finite automata and weakly invertible finite automata have been intensively studied.

Formally speaking, a finite automaton is a five-tuple $M =< X, Y, S, \delta, \lambda >$, where X is a finite input alphabet, Y is a finite output alphabet, S is a finite set of internal states, δ is a next-state function, $\delta : S \times X \rightarrow S$ and λ is an output function, $\lambda : S \times X \rightarrow Y$.

We can naturally extend δ to a mapping from $S \times X^*$ to S, where X^* denotes the set of all finite strings over X (including the empty string ε): For any $s \in S$, $\alpha \in X^*$ and $x \in X$, define $\delta(s, \varepsilon) = s$ and $\delta(s, \alpha x) = \delta(\delta(s, \alpha), x)$. Similarly, λ can be naturally extended to a mapping from $S \times (X^* \cup X^\omega)$ to $Y^* \cup Y^\omega$, where X^ω denotes the set of all infinite strings over X: For any $s \in S$, $\alpha \in X^* \cup X^\omega$ and $x \in X$, define $\lambda(s, \varepsilon) = \varepsilon$ and $\lambda(s, x\alpha) = \lambda(s, x)\lambda(\delta(s, x), \alpha)$.

Our finite automaton is a transducer converting a string over the input alphabet into a length-preserving string over the output alphabet, rather than just an acceptor.

Definition 1. Let $M =< X, Y, S, \delta, \lambda >$ be a finite automaton. If for any $s \in S$ and any $\alpha, \alpha' \in X^\omega$, $\lambda(s, \alpha) = \lambda(s, \alpha')$ always implies $\alpha = \alpha'$, then M is said to be a *weakly invertible finite automaton* (WIFA for short).

Definition 2. A finite automaton $M =< X, Y, S, \delta, \lambda >$ is said to be a WIFA *with delay* τ if for any $s \in S$, $a, a' \in X$ and $\alpha, \alpha' \in X^\tau$, $\lambda(s, a\alpha) = \lambda(s, a'\alpha')$ always implies $a = a'$, where τ is a non-negative integer and X^τ is the set of all strings of length τ over X.

Proposition 3. *A finite automaton M is a WIFA if and only if M is a WIFA with delay τ for some $\tau \leq n(n-1)/2$, where n is the number of states of M.*

Definition 4. Let $M =< X, Y, S, \delta, \lambda >$ and $M' =< Y, X, S', \delta', \lambda' >$ be a pair of finite automata. If for any $s \in S$ there exists $s' \in S'$ such that for any $\alpha \in X^\omega$, there exists $\alpha_0 \in X^\tau$ satisfying $\lambda'(s', \lambda(s, \alpha)) = \alpha_0\alpha$, then M is called a *weak inverse with delay* τ of M. Such a state s' is called the *match state* of s, and such a string $\alpha_0 = x_{-\tau} \cdots x_{-1}$ the *delay prefix*.

Proposition 5. *A finite automaton M is a WIFA with delay τ if and only if M has a weak inverse with delay τ.*

For the proofs of above two propositions, the reader may be referred to [9] [17].

Definition 6. Let $M_1 =< X, Y, S_1, \delta_1, \lambda_1 >$ and $M_2 =< Y, Z, S_2, \delta_2, \lambda_2 >$ be a pair of finite automata. The *composition* of M_1 and M_2 is a new finite automaton $M_1 \cdot M_2 = < X, Z, S_1 \times S_2, \delta, \lambda >$, 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 $< s_1, s_2 > \in S_1 \times S_2$.

$M_1 \cdot M_2$ is the natural concatenation of M_1 and M_2 (*i.e.*, M_2 takes the output of M_1 as its input). it is not difficult to see that if M_1 is a WIFA with delay τ_1 and M_2 is a WIFA with delay τ_2, then $M_1 \cdot M_2$ is a WIFA with delay $\tau_1 + \tau_2$.

Definition 7. 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 finite automata. States $s_1 \in S_1$ and $s_2 \in S_2$ are said to be *equivalent* if for any $\alpha \in X^*$, $\lambda_1(s_1, \alpha) = \lambda_2(s_2, \alpha)$. 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.

1.3 The Basic Idea of Breaking FAPKC

In FAPKC, the plaintext is encrypted by a nonlinear WIFA with delay τ (typically $\tau > 15$) which is the composition of a nonlinear WIFA with delay 0 and a linear WIFA with delay τ. The security of FAPKC depends on the following facts: (a) It is difficult to construct a weak inverse with delay τ of any given nonlinear WIFA with delay τ. (b) It is easy to construct a weak inverse with delay τ of any given linear WIFA with delay τ. (c) It is easy to construct a weak inverse with delay 0 of any given nonlinear WIFA with delay 0.

The principle of FAPKC is as shown in Fig. 1, where M_0 is a nonlinear WIFA with delay 0, M_1 is a linear WIFA with delay τ, M_0^{-1} is a weak inverse with delay 0 of M_0, M_1^{-1} is a weak inverse with delay τ of M_1, s_0^{-1} is a match state of s_0 and s_1^{-1} is a match state of s_1.

Fig. 1. The principle of FAPKC.

In FAPKC, M_0 and M_1 are kept as the secret key, from which M_0^{-1} and M_1^{-1}, the decryption automata, can be easily derived. The public key is a sub-automaton of the composition of M_0 and M_1 (not exactly the $M_0 \cdot M_1$, see Section 3 for details). We call it the encryption automaton. To separate M_0 and M_1 from the encryption automaton, we needed to find a special left common factor of two given matrix polynomials. However, that is a very difficult problem. So far, no divisibility theory has been established for matrix polynomial rings.

In this paper, we prove a previously unknown property of WIFAs. Then, by this property, we reduce the problem of separating M_0 and M_1 from the encryption automaton to a much easier problem: finding a special left common factor of two given matrix polynomials in module matrix polynomial rings. Hence, we can break FAPKC by successfully finding decryption automata from the public key.

2 Linear WIFAs

Finite automata used in FAPKC are actually sequential machines. In order to keep consistent with the references, we still use the term "finite automaton"(FA for short). Finite automata considered here are of the form $M =< X, Y, S, \delta, \lambda >$, where $X = Y =$ the l-dimensional linear space over $GF(2) = \{0, 1\}$ (typically $l = 8$, so that FAPKC encrypts byte by byte), and δ and λ are specified by the following *definition formula.*

$$M: \quad y(i) = f(x(i), x(i-1), \cdots, x(i-r), y(i-1), \cdots, y(i-t)), \quad i = 0, 1, 2, \cdots \quad (1)$$

In the definition formula (1), $x(i)$ represents the input at time i, $y(i)$ represents the output at time i, and each of $x(i)$ and $y(i)$ is a binary l-dimensional vector. Obviously, S is the $l(r+t)$-dimensional linear space over $GF(2)$. A finite automaton defined by (1) is so-called r-input and t-output memory FA, where $< x(-1), \cdots, x(-r), y(-1), \cdots, y(-t) >$ is the initial state. All finite automata used in FAPKC are of this type.

The M defined by (1) is called a linear FA, if f is a linear function. In this case, there exist $l \times l$ matrices over $GF(2)$, $A_0, A_1, \cdots, A_r$, $B_1, B_2, \cdots, B_t$ such that the definition formula (1) can be written as

$$M: \qquad y(i) = \sum_{j=0}^{r} A_j x(i-j) + \sum_{j=1}^{t} B_j y(i-j), \qquad i = 0, 1, 2, \cdots \quad (2)$$

Here, each $x(i)$ is regarded as a column vector, $A_j x(i-j)$ is the usual multiplication of a matrix and a column vector, and the addition is the usual addition of vectors. Next we consider only r-input linear FAs, defined by (3).

$$M: \qquad y(i) = A_0 x(i) + A_1 x(i-1) + \cdots + A_r x(i-r), \qquad i = 0, 1, 2, \cdots \quad (3)$$

The linear coefficients $A_0, A_1, A_2, \cdots, A_r$ uniquely determine the finite automaton M. We can decide from $A_0, A_1, A_2, \cdots, A_r$ whether M is a WIFA with delay r. Furthermore, if M is a WIFA with delay r, we can construct a weak inverse with delay r of M.

Let $s_0 =< x(-1), x(-2), \cdots, x(-r) >$ be an arbitrarily given initial state of M. Let the *time variate* i take $0, 1, 2, \cdots, r$ in (3). Then we obtain

$$\begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(r) \end{pmatrix} = \begin{pmatrix} A_0 & (0) & \cdots & (0) \\ A_1 & A_0 & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ A_r & A_{r-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} x(0) \\ x(1) \\ \vdots \\ x(r) \end{pmatrix} + \begin{pmatrix} A_1 & \cdots & A_r \\ \vdots & \vdots & \vdots \\ A_r & \cdots & (0) \\ (0) & \cdots & (0) \end{pmatrix} \begin{pmatrix} x(-1) \\ x(-2) \\ \vdots \\ x(-r) \end{pmatrix} \quad (4)$$

where (0) denotes the zero $l \times l$ matrix over $GF(2)$.

Proposition 8. *There exists a fast algorithm for finding an invertible $(r+1)l \times (r+1)l$ matrix $\mathbf{P}$ from $A_0, A_1, A_2, \cdots, A_r$, such that*

$$\mathbf{P}\begin{pmatrix} A_0 & (0) & \cdots & (0) \\ A_1 & A_0 & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ A_r & A_{r-1} & \cdots & A_0 \end{pmatrix} = \begin{pmatrix} A_{0,0} & (0) & \cdots & (0) \\ A_{1,1} & A_{1,0} & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ A_{r,r} & A_{r,r-1} & \cdots & A_{r,0} \end{pmatrix} \qquad (5)$$

where $A_{i,j}$'s satisfy the following condition: for any $0 \leq h \leq r$,

$$\forall x_1, x_2, \cdots, x_h \in X, \exists x_0 \in X, A_{h,1}x_1 + A_{h,2}x_2 + A_{h,h}x_h = A_{h,0}x_0 \qquad (6)$$

We explain how to find $\mathbf{P}$ in Appendix.

Proposition 9. *The finite automaton M is a WIFA with delay r if and only if $A_{0,0}$ is an invertible matrix, i.e., of full rank.*

If M is a WIFA with delay r, we can construct a weak inverse with delay r of M from $\mathbf{P}$ easily. Adding the time variate i into (4), we have

$$\begin{pmatrix} y(i) \\ y(i+1) \\ \vdots \\ y(i+r) \end{pmatrix} = \begin{pmatrix} A_0 & (0) & \cdots & (0) \\ A_1 & A_0 & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ A_r & A_{r-1} & \cdots & A_0 \end{pmatrix} \begin{pmatrix} x(i) \\ x(i+1) \\ \vdots \\ x(i+r) \end{pmatrix} + \begin{pmatrix} A_1 & \cdots & A_r \\ \vdots & \vdots & \vdots \\ A_r & \cdots & (0) \\ (0) & \cdots & (0) \end{pmatrix} \begin{pmatrix} x(i-1) \\ x(i-2) \\ \vdots \\ x(i-r) \end{pmatrix},$$

$$i = 0, 1, 2, \cdots \qquad (7)$$

Multiply the the matrix $\mathbf{P}$ from left to (7), we obtain

$$\mathbf{P}\begin{pmatrix} y(i) \\ y(i+1) \\ \vdots \\ y(i+r) \end{pmatrix} = \begin{pmatrix} A_{0,0} & (0) & \cdots & (0) \\ A_{1,1} & A_{1,0} & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ A_{r,r} & A_{r,r-1} & \cdots & A_{r,0} \end{pmatrix} \begin{pmatrix} x(i) \\ x(i+1) \\ \vdots \\ x(i+r) \end{pmatrix} + \mathbf{P}\begin{pmatrix} A_1 & \cdots & A_r \\ \vdots & \vdots & \vdots \\ A_r & \cdots & (0) \\ (0) & \cdots & (0) \end{pmatrix} \begin{pmatrix} x(i-1) \\ x(i-2) \\ \vdots \\ x(i-r) \end{pmatrix},$$

$$i = 0, 1, 2, \cdots \qquad (8)$$

Denote $\mathbf{P} = \begin{pmatrix} P_{0,0} & P_{0,1} & \cdots & P_{0,r} \\ P_{1,0} & P_{1,1} & \cdots & P_{1,r} \\ \vdots & \vdots & \vdots & \vdots \\ P_{r,0} & P_{r,1} & \cdots & P_{r,r} \end{pmatrix}$ and $\mathbf{P}\begin{pmatrix} A_1 & \cdots & A_r \\ \vdots & \vdots & \vdots \\ A_r & \cdots & (0) \\ (0) & \cdots & (0) \end{pmatrix} = \begin{pmatrix} Q_{0,1} & Q_{0,2} & \cdots & Q_{0,r} \\ Q_{1,1} & Q_{1,2} & \cdots & Q_{1,r} \\ \vdots & \vdots & \vdots & \vdots \\ Q_{r,1} & Q_{r,2} & \cdots & Q_{r,r} \end{pmatrix}$,

where $P_{i,j}$ and $Q_{i,j}$ are $l \times l$ matrices. Then from the first line of (8), we have

$$x(i) = A_{0,0}^{-1}\sum_{j=0}^{r} P_{0,j}y(i+j) + A_{0,0}^{-1}\sum_{j=1}^{r} Q_{0,j}x(i-j), \qquad i = 0, 1, 2, \cdots \qquad (9)$$

Denote $P_j = A_{0,0}^{-1}P_{0,j}$ and $Q_j = A_{0,0}^{-1}Q_{0,j}$ for $j = 0, 1, 2, \cdots, r$. We can rewrite (9) as (10),

$$x(i) = \sum_{j=0}^{r} P_j y(i+j) + \sum_{j=1}^{r} Q_j x(i-j), \qquad i = 0, 1, 2, \cdots \tag{10}$$

For any initial state $s = < x(-1), x(-2), \cdots, x(-r) >$ of M and any $x(0), x(1), \cdots, x(n+r) \in X$, if $y(0)y(1)\cdots y(n+r) = \lambda_M(s, x(0)x(1)\cdots x(n+r))$ (*i.e.*, (2) holds for $i = 0, 1, \cdots, n+r$). Then from (10) we can calculate $x(0), x(1), \cdots, x(n)$ one by one. Hence, (10) specifies a weak inverse with delay r of M, although (10) itself is not a definition formula.

3 FAPKC and Matrix Polynomials

In this section, we explain FAPKC in detail. As introduced in Section 1.3, the secret key of FAPKC consists of a nonlinear WIFA with delay 0, M_0 and a linear WIFA with delay τ, M_1. The definition formula of M_0 is

$$M_0: \quad y(i) = \sum_{j=0}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} \bar{B}_j x(i-j) \cdot x(i-j-1), \qquad i = 0, 1, 2, \cdots \tag{11}$$

where $B_0, B_1, B_2, \cdots, B_r, \bar{B}_1, \bar{B}_2, \cdots, \bar{B}_{r-1}$ are $l \times l$ matrices over $GF(2)$, and B_0 is an invertible matrix. The operation $x \cdot y$ is defined to be $(a_1b_1, a_2b_2, \cdots, a_lb_l)^T$ for $x = (a_1, a_2, \cdots, a_l)^T$ and $y = (b_1, b_2, \cdots, b_n)^T$. (It does not matter what the operation '$\cdot$' is. Actually, we only need '$\cdot$' to be a nonlinear operation from two vectors to one vector. The operation '$\cdot$' defined above is adopted in the practical FAPKC.) M_0 is an r-input memory FA. It is easy to see that the r-output memory M_0^{-1} is a weak inverse with delay 0 of M_0.

$$M_0^{-1} : x(i) = B_0^{-1}\left(y(i) + \sum_{j=1}^{r} B_j x(i-j) + \sum_{j=1}^{r-1} \bar{B}_j x(i-j) \cdot x(i-j-1) \right) \tag{12}$$

For any initial state $s_0 = < x(-1), x(-2), \cdots, x(-r) >$ of M_0, its match state in M_0^{-1} is also $< x(-1), x(-2), \cdots, x(-r) >$.

The definition formula of M_1 is

$$M_1: \qquad z(i) = A_0 y(i) + A_1 y(i-1) + \cdots + A_\tau y(i-\tau) \qquad i = 0, 1, 2, \cdots \tag{13}$$

The encrypt automaton is the composition of M_0 and M_1, denoted by $M = M_0 \circ M_1$. The definition formula of M is obtained by substituting (11) into (13).

$$M : z(i) = \sum_{t=0}^{\tau} A_t \left(\sum_{j=0}^{r} B_j x(i-j-t) + \sum_{j=1}^{r-1} \bar{B}_j x(i-j-t) \cdot x(i-j-t-1) \right) \tag{14}$$

Actually, $M = M_0 \circ M_1$ is not exactly the same as $M_0 \cdot M_1$, but it is equivalent to a sub-automaton of $M_0 \cdot M_1$. That is, for any state $s =< x(-1), x(-2), \cdots, x(-r-\tau) >$ of $M = M_0 \circ M_1$, s is equivalent to the state $< s_0, s_1 >$ of $M_0 \cdot M_1$, where $s_0 =< x(-1), x(-2), \cdots, x(-r) >$ and $s_1 =< y(-1), y(-2), \cdots, y(-\tau) >$, and

$$y(-h) = \sum_{j=0}^{r} B_j x(-h-j) + \sum_{j=1}^{r-1} \bar{B}_j x(-h-j) \cdot x(-h-j-1), h = 1, 2, \cdots, \tau \quad (15)$$

Now let us look back at (14). It can be simplified as (16).

$$M: \quad z(i) = \sum_{j=0}^{r+\tau} C_j x(i-j) + \sum_{j=1}^{r+\tau-1} \bar{C}_j x(i-j) \cdot x(i-j-1), \quad i = 0, 1, 2, \cdots \quad (16)$$

$$C_j = \sum_{h+t=j} A_h B_t, j = 0, 1, \cdots, r+\tau; \bar{C}_j = \sum_{h+t=j} A_h \bar{B}_t, j = 1, 2, \cdots, r+\tau-1 \quad (17)$$

Hence, FAPKC works in the following way:

1. Choose M_0 and M_1 as defined above. All the A_j, B_j and $\bar{B}_j$ are kept as secret key. (There are some criterions about the way how we choose A_j, B_j and $\bar{B}_j$ [2] [12] [24].)
2. Calculate C_j and $\bar{C}_j$ from A_j, B_j and $\bar{B}_j$ as defined by (17), and arbitrarily choose $s =< x(-1), x(-2), \cdots, x(-r-\tau) >$ as the initial state. Then C_j, $\bar{C}_j$ and s are made public.
3. From $s =< x(-1), x(-2), \cdots, x(-r-\tau) >$, get s_0 and s_1 as defined in (15).
4. To encrypt plaintext $x(0)x(1)\cdots x(m)$, first arbitrarily choose $x(m+1)x(m+2)\cdots x(m+\tau) \in X^\tau$. Then, input $x(0)x(1)x(2)\cdots x(m+\tau)$ into $M = M_0 \circ M_1$ with initial state s. The output $z(0)z(1)z(2)\cdots z(m+\tau)$ is the ciphertext.
5. To decrypt $z(0)z(1)z(2)\cdots z(m+\tau)$, first use M_1^{-1} (as defined in Section 2) and s_1 to obtain $y(0)y(1)\cdots y(m)$. Then supply $y(0)y(1)\cdots y(m)$ into M_0^{-1} with initial state s_0, and obtain $x(0)x(1)\cdots x(m)$ as the output.

One of possible attacks to FAPKC is to reduce (16) to simultaneous quadratic equations over $GF(2)$, and solve them. However, it is known that solving nonlinear equations over $GF(2)$ is very difficult if the number of its arguments is large. Another way is the ciphertext attack through randomized searching. FAPKC can resist such a kind of attacks by taking $\tau > 15$ and choosing uniformly *increasing ranks* for M_1 [2].

Now we consider the possibility of separating M_0 and M_1 from $M = M_0 \circ M_1$. We denote the $l \times l$ matrix polynomial ring over $GF(2)$ by $\mathbf{MP}(x)$. More precisely, $\mathbf{MP}(x)$ is the set of all the polynomials like $\mathbf{G}(x) = G_0 + G_1 x + \cdots + G_n x^n$, where $G_0, G_1, \cdots, G_n$ are $l \times l$ matrices over $GF(2)$ and x is the formal variable.

Let $\mathbf{A}(x) = \sum_{j=0}^{\tau} A_j x^j$, $\mathbf{B}(x) = \sum_{j=0}^{r} B_j x^j$, $\bar{\mathbf{B}}(x) = \sum_{j=1}^{r-1} \bar{B}_j x^{j-1}$, $\mathbf{C}(x) = \sum_{j=0}^{r+\tau} C_j x^j$ and $\bar{\mathbf{C}}(x) = \sum_{j=1}^{r+\tau-1} \bar{C}_j x^{j-1}$. Then from (17), we have $\mathbf{C}(x) = \mathbf{A}(x)\mathbf{B}(x)$ and $\bar{\mathbf{C}}(x) = \mathbf{A}(x)\bar{\mathbf{B}}(x)$.

Problem 1: For $\mathbf{C}(x)$ and $\bar{\mathbf{C}}(x)$ given above, find $\mathbf{A}'(x) = \sum_{j=0}^{\tau} A'_j x^j$, $\mathbf{B}'(x) = \sum_{j=0}^{r} B'_j x^j$ (where B'_0 is invertible) and $\bar{\mathbf{B}}'(x) = \sum_{j=1}^{r-1} \bar{B}'_j x^{j-1}$ in $\mathbf{MP}(x)$, such that $\mathbf{C}(x) = \mathbf{A}'(x)\mathbf{B}'(x)$ and $\bar{\mathbf{C}}(x) = \mathbf{A}'(x)\bar{\mathbf{B}}'(x)$.

If we could solve Problem 1 by a fast algorithm, then we could break FAPKC. This is because in such a case we could obtain a nonlinear WIFA with delay 0, M'_0 from $B'_0, B'_1, \cdots, B'_r, \bar{B}_1, \bar{B}_2, \cdots, \bar{B}_{r-1}$, and a linear WIFA with delay τ, M'_1 from $A'_0, A'_1, \cdots, A'_\tau$, while $M'_0 \circ M'_1 = M_0 \circ M_1$. Hence, we could break FAPKC by constructing ${M'_0}^{-1}$ and ${M'_1}^{-1}$. However, it is very difficult to solve Problem 1. Since $\mathbf{MP}(x)$ is noncommunitive and contains divisors of zero, establishing divisibility theory for $\mathbf{MP}(x)$ seems to be difficult [10]. Although polynomial time algorithms for factorization of polynomials over a finite field exist [4], any feasible algorithm for factoring matrix polynomials is not yet known.

In the next section, we prove a simple but previously unknown property for the input memory WIFAs. By this property we can reduce the secure base of FAPKC to Problem 2, which is much easier than Problem 1.

Problem 2: For $\mathbf{C}(x)$ and $\bar{\mathbf{C}}(x)$ given above, find $\mathbf{A}'(x) = \sum_{j=0}^{\tau} A'_j x^j$, $\mathbf{B}'(x) = \sum_{j=0}^{r} B'_j x^j$ (where B'_0 is invertible) and $\bar{\mathbf{B}}'(x) = \sum_{j=1}^{r-1} \bar{B}'_j x^{j-1}$ in $\mathbf{MP}(x)$, such that $\mathbf{C}(x) = \mathbf{A}'(x)\mathbf{B}'(x) \pmod{x^{\tau+1}}$ and $\bar{\mathbf{C}}(x) = \mathbf{A}'(x)\bar{\mathbf{B}}'(x) \pmod{x^{\tau-1}}$.

4 A Property of WIFAs

In this section, we prove that only those terms related with $x(i), x(i-1), \cdots, x(i-\tau)$ determine whether an FA is a WIFA with delay τ. Let M_f and M_{f+g} be a pair of $(r+\tau)$-input memory FAs. Their definition formulae are

$$M_f: \quad y(i) = f(x(i), x(i-1), \cdots, x(i-r-\tau)), \qquad i = 0, 1, 2, \cdots \tag{18}$$

$$M_{f+g}: y(i) = f(x(i), \cdots, x(i-r-\tau)) + g(x(i-\tau-1), \cdots, x(i-r-\tau)), i = 0, 1, 2, \cdots \tag{19}$$

Theorem 10. *The finite automaton M_f is a WIFA with delay τ if and only if M_{f+g} is a WIFA with delay τ.*

Proof. ($\Longrightarrow$) The proof is simply from the definition of WIFA with delay τ. For any state $s =< x(-1), x(-2), \cdots, x(-r-\tau) >$ of M_f and any $x(0), x(1), \ldots, x(\tau)$, $x(0)', x(1)', \ldots, x(\tau)' \in X$, $\lambda_{M_f}(s, x(0)x(1)\cdots x(\tau))$=$\lambda_{M_f}(s, x(0)'x(1)'\cdots x(\tau)')$ implies $x(0) = x(0)'$, *i.e.*,

$$\begin{cases} f(x(0), x(-1), x(-2), \cdots, x(-r-\tau)) = f(x(0)', x(-1), x(-2), \cdots, x(-r-\tau)) \\ f(x(1), x(0), x(-1), \cdots, x(1-r-\tau)) = f(x(1)'x(0)', x(-1), \cdots, x(1-r-\tau)) \\ \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ f(x(\tau), \cdots, x(0), x(-1), \cdots, x(-r)) \;=\; f(x(\tau)', \cdots, x(0)', x(-1), \cdots, x(-r)) \end{cases}$$

implies $x(0) = x(0)'$. Adding $g(x(-\tau-1), \cdots, x(-\tau-r))$, $g(x(-\tau), \cdots, x(1-\tau-r))$, $\cdots$, $g(x(-1), x(-2), \cdots, x(-r))$ to both sides of equations above respectively, $\lambda_{M_{f+g}}(s, x(0)x(1)\cdots x(\tau))$=$\lambda_{M_{f+g}}(s, x(0)'x(1)' \cdots x(\tau)')$ implies $x(0) = x(0)'$.

From the arbitrariness of $s, x(0), x(1), \ldots, x(\tau)$, $x(0)'$, $x(1)', \ldots$, and $x(\tau)'$, M_{f+G} is a WIFA with delay τ. ($\Longleftarrow$) The opposit direction can be proved similarly. □

Next we show that we can construct a weak inverse with delay τ of M_{f+g} from a weak inverse with delay τ of M_f.

Theorem 11. *If M_f^{-1} is a weak inverse with delay τ of M_f, then a weak inverse with delay τ of M_{f+g} can be constructed from M_f^{-1}.*

Proof. Let $s =< x(-1), x(-2), \cdots, x(-r-\tau) >$ be an initial state of M_f and M_{f+g}, and let s^{-1} be the match state of s of M_f^{-1}. We replace the first τ output of M_f^{-1} by $x(-\tau), x(1-\tau), \cdots, x(-1)$. The principle of constructing M_{f+g}^{-1} is given in Fig. 2. □

Fig. 2. M_{f+g} and its weak inverse with delay τ M_{f+g}^{-1}.

5 Break FAPKC

Let M be the encryption WIFA with delay τ in FAPKC, as defined by (16). Then $M = M_0 \circ M_1$, where M_0 and M_1 are defined by (11) and (13), respectively.

Theorem 12. *If Problem 2 can be solved by a fast algorithm, then we can break FAPKC.*

Proof. Suppose that we can find $\mathbf{A}'(x) = \sum_{j=0}^{\tau} A_j' x^j$, $\mathbf{B}'(x) = \sum_{j=0}^{r} B_j' x^j$ (where B_0' is invertible) and $\bar{\mathbf{B}}'(x) = \sum_{j=1}^{r-1} \bar{B}_j' x^{j-1}$ in $\mathbf{MP}(x)$, such that $\mathbf{C}(x) = \mathbf{A}'(x)\mathbf{B}'(x) \pmod{x^{\tau+1}}$ and $\bar{\mathbf{C}}(x) = \mathbf{A}'(x)\bar{\mathbf{B}}'(x) \pmod{x^{\tau-1}}$. We construct a nonlinear WIFA with delay 0, M_0' from $B_0', B_1', \cdots, B_r', \bar{B}_1, \bar{B}_2, \cdots, \bar{B}_{r-1}$, and a linear WIFA with delay τ, M_1' from $A_0', A_1', \cdots, A_\tau'$. ($M_1'$ is a linear WIFA with delay τ from Proposition 9 and Theorem 10.)

It is easy to construct $M_0'^{-1}$ and $M_1'^{-1}$ from $B_0', B_1', \cdots, B_r', \bar{B}_1, \bar{B}_2, \cdots, \bar{B}_{r-1}$ and $A_0', A_1', \cdots, A_\tau'$ respectively. Hence, we can construct a weak inverse with delay τ of $M' = M'_0 \circ M'_1$, denoted by ${M'}^{-1}$.

$$M': \qquad z(i) = \sum_{j=0}^{r+\tau} C'_j x(i-j) + \sum_{j=1}^{r+\tau-1} \bar{C}'_j x(i-j)\cdot x(i-j-1), \qquad i = 0,1,2,\cdots$$

Let $\mathbf{C}'(x) = \mathbf{A}'(x)\mathbf{B}'(x) = \sum_{j=0}^{r+\tau} C'_j x^j$ and $\bar{\mathbf{C}}'(x) = \mathbf{A}'(x)\bar{\mathbf{B}}'(x) = \sum_{j=0}^{r+\tau} \bar{C}'_j x^j$. Since $\mathbf{C}(x) = \mathbf{C}'(x) \pmod{x^{\tau+1}}$ and $\bar{\mathbf{C}}(x) = \bar{\mathbf{C}}'(x) \pmod{x^{\tau-1}}$, from Theorem 11, we can construct the weak inverse with delay τ of M, M^{-1} from ${M'}^{-1}$. □

Now we illustrate that Problem 2 can be easily solved. Let $\mathbf{A}(x) = \sum_{j=0}^{\tau} A_j x^j$, $\mathbf{B}(x) = \sum_{j=0}^{r} B_j x^j$, $\bar{\mathbf{B}}(x) = \sum_{j=1}^{r-1} \bar{B}_j x^{j-1}$, $\mathbf{C}(x) = \sum_{j=0}^{r+\tau} C_j x^j$, $\bar{\mathbf{C}}(x) = \sum_{j=1}^{r+\tau-1} \bar{C}_j x^{j-1}$, $\mathbf{C}(x) = \mathbf{A}(x)\mathbf{B}(x)$ and $\bar{\mathbf{C}}(x) = \mathbf{A}(x)\bar{\mathbf{B}}(x)$, as defined in Section 3. Our aim is to find $\mathbf{A}'(x) = \sum_{j=0}^{\tau} A'_j x^j$, $\mathbf{B}'(x) = \sum_{j=0}^{r} B'_j x^j$ and $\bar{\mathbf{B}}'(x) = \sum_{j=1}^{r-1} \bar{B}'_j x^{j-1}$ such that $\mathbf{C}(x) = \mathbf{A}'(x)\mathbf{B}'(x) \pmod{x^{\tau+1}}$, $\bar{\mathbf{C}}(x) = \mathbf{A}'(x)\bar{\mathbf{B}}(x) \pmod{x^{\tau-1}}$, and B'_0 is invertible.

Let $\mathbf{B}'(x) = I$ (the $l \times l$ unit matrix), $\mathbf{A}'(x) = \sum_{j=0}^{\tau} C_j x^j$. We can find $\bar{\mathbf{B}}'(x)$ such that $\bar{\mathbf{C}}(x) = \mathbf{A}'(x)\bar{\mathbf{B}}(x) \pmod{x^{\tau-1}}$, by solving a linear equation group

$$\begin{pmatrix} C_0 & (0) & \cdots & (0) \\ C_1 & C_0 & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ C_{\tau-2} & C_{\tau-3} & \cdots & C_0 \end{pmatrix} \begin{pmatrix} \bar{B}'_1 \\ \bar{B}'_2 \\ \vdots \\ \bar{B}'_{\tau-1} \end{pmatrix} = \begin{pmatrix} \bar{C}_1 \\ \bar{C}_2 \\ \vdots \\ \bar{C}_{\tau-1} \end{pmatrix} \tag{20}$$

The linear equation group (20) definitely has solutions. Actually, one simple solution is

$$\begin{pmatrix} \bar{B}'_1 \\ \bar{B}'_2 \\ \vdots \\ \bar{B}'_{\tau-1} \end{pmatrix} = \begin{pmatrix} B_0 & (0) & \cdots & (0) \\ B_1 & B_0 & \cdots & (0) \\ \vdots & \vdots & \ddots & \vdots \\ B_{\tau-2} & B_{\tau-3} & \cdots & B_0 \end{pmatrix}^{-1} \begin{pmatrix} \bar{B}_1 \\ \bar{B}_2 \\ \vdots \\ \bar{B}_{\tau-1} \end{pmatrix} \tag{21}$$

However, we can only obtain $\bar{B}'_1, \bar{B}'_2, \cdots, \bar{B}'_{\tau-1}$ by solving (20) since $\mathbf{A}(x) = \sum_{j=0}^{\tau} A_j x^j$, $\mathbf{B}(x) = \sum_{j=0}^{r} B_j x^j$ and $\bar{\mathbf{B}}(x) = \sum_{j=1}^{r-1} \bar{B}_j x^{j-1}$ are kept secret in FAPKC.

6 A Modification

We have shown that FAPKC can be broken by constructing the decryption automata from the encryption automaton. The key point of insecure FAPKC is that M_0 is a WIFA with delay 0, *i.e.*, B_0 is an invertible matrix. A natural modification to FAPKC is to change M_0 into a WIFA with delay τ, based on Theorem 10 and Theorem 11.

$$M_0: \quad y(i) = \sum_{j=0}^{\tau} B_j x(i-j) + \sum_{j=1}^{\tau-1} \bar{B}_j x(i-\tau-j)\cdot x(i-\tau-j-1), \qquad i = 0,1,2,\cdots$$

Here $B_0, B_1, \cdots, B_\tau$ are taken to be the linear coefficients of a linear WIFA with delay τ. Apparently, M_0 is a nonlinear WIFA with delay τ. M_1 is the same as before. Let the encryption automaton be

$$M : z(i) = \sum_{j=0}^{2\tau} C_j x(i-j) + \sum_{j=1}^{\tau} \bar{C}_j x(i-\tau-j) \cdot x(i-\tau-j-1), i = 0, 1, 2, \cdots$$

Let $\mathbf{A}(x) = \sum_{j=0}^{\tau} A_j x^j$, $\mathbf{B}(x) = \sum_{j=0}^{\tau} B_j x^j$, $\bar{\mathbf{B}}(x) = \sum_{j=1}^{\tau-1} \bar{B}_j x^{j-1}$, $\mathbf{C}(x) = \sum_{j=0}^{2\tau} C_j x^j$ and $\bar{\mathbf{C}}(x) = \sum_{j=1}^{\tau} \bar{C}_j x^{j-1}$. Then $\mathbf{C}(x) = \mathbf{A}(x)\mathbf{B}(x)$ and $\bar{\mathbf{C}}(x) = (\mathbf{A}(x)\bar{\mathbf{B}}(x) \bmod x^\tau)$. This is also a variation of the version in [23].

This modified FAPKC can resist the attack described in this paper. To resist the attack in which $\mathbf{A}'(x)$ is taken to be $\mathbf{C}(x)$, $\mathbf{B}'(x)$ is taken to be I and $\bar{\mathbf{B}}'(x)$ is calculated by solving the linear equation group, we should choose $\mathbf{A}(x)$, $\mathbf{B}(x)$ and $\bar{\mathbf{B}}(x)$ carefully so that they satisfy some tradeoff. However, further analyses are needed before we can give any definite conclusion on its security.

References

1. F. Bao, “Limited error-propagation, self-synchronization and finite input memory FSMs as weak inverses”, in Advances in Chinese Computer Science, Vol. 3, World Scientific, Singapore, 1991, pp. 1-24.
2. F. Bao, Y. Igarashi, “A randomized algorithm to finite automata public key cryptosystem”, in Proc. of ISAAC’94, LNCS 834, Springer-Verlag, 1994, pp. 678-686.
3. F. Bao, Y. Igarashi, X. Yu, “Some results on the decomposability of weakly invertible finite automata”, IPSJ Technical Report, 94-AL, November, 1994, pp. 17-24.
4. E. Berlekamp, “Factoring polynomials over large finite field”, Math. Comp. Vol. 24, 1970, pp. 713-735.
5. S. Chen, “On the structure of finite automata of which M’ is an (weak) inverse with delay τ”, J. of Computer Science and Technology, Vol. 1, No. 2, 1986, pp. 54-59.
6. S. Chen, “On the structures of (weak) inverses of an (weakly) invertible finite automata”, J. of Computer Science and Technology, Vol. 1, No. 3, 1986, pp. 92-100.
7. S. Chen, R. Tao, “The structure of weak inverses of a finite automaton with bounded error propagation”, Kexue Tongbao, Vol. 32, No. 10, 1987, pp. 713-714.
8. S. Even, “Generalized automata and their information losslessness”, in Switching Circuit Theory and Logic Design, 1962, pp. 144-147.
9. S. Even, “On Information lossless automata of finite order”, IEEE Trans. on Electric Computer, Vol. 14, No. 4, 1965, pp. 561-569.
10. I. Gohberg, P. Lancaster, L. Rodman, Matrix Polynomials, Academic Press, New York.
11. D. A. Huffman, “Canonical forms for information-lossless finite-state logic machines”, IRE Trans. on Circuit Theory, Vol. CT-6, Special Supplements, May, 1959, pp. 41-59.
12. J. Li, X. Gao, “Realization of finite automata public key cryptosystem and digital signature”, in Proc. of the Second National Conference on Cryptography, CRYPTO-CHINA’92, pp. 110-115. (in Chinese)
13. J. L. Massey, M. K. Sain, “Inverse of linear sequential circuits”, IEEE Trans. on Computers, Vol. 17, No. 4, 1968, pp. 330-337.

14. J. L. Massey, A. Gubser, A. Fisger, et al., "A selfsynchronizing digital scrambler for cryptographic protection of data", in Proc. of International Zurich Seminar on Digital Communications, March, 1984.
15. R. R. Olson, "A note on feedforward inverses for linear sequential circuits", IEEE Trans. on Computers, Vol. 19, No. 12, 1970, pp. 1216-1221.
16. A. Salomaa, Public-Key Cryptography, EATCS Monographs on Theoretical Computer Science, Vol. 23, Springer-Verlag, 1990.
17. R. Tao, Invertibility of Finite Automata, Science Press, 1979, Beijing. (in Chinese)
18. R. Tao, "On the relation between bounded error propagation and feedforward inverse", Kexue Tongbao, Vol. 27, No. 7, 1982, pp. 406-408. (in Chinese)
19. R. Tao, "On the structure of feedforward inverse", Science in China, A, Vol. 26, No. 12, 1983, pp. 1073-1078. (in Chinese)
20. R. Tao, S. Chen, "Finite automata public key cryptosystem and digital signature", Computer Acta, Vol. 8, No. 6, 1985, pp. 401-409. (in Chinese)
21. R. Tao, S. Chen, "Two varieties of finite automata public key cryptosystem and digital signature", J. of Computer Science and Technology, Vol. 1, No. 1, pp. 9-18.
22. R. Tao, "Invertibility of linear finite automata over a ring", in Proc. of ICALP'88, LNCS 317, Springer-Verlag, 1988, pp. 489-501.
23. R. Tao, S. Chen, "An implementation of identity-based cryptosystems and signature schemes by finite automaton public key cryptosystems", in Proc. of the Second National Conference on Cryptography, CRYPTO-CHINA'92, pp. 87-104. (in Chinese)
24. H. Zhang, Z. Qin, et al., "The software implementation of FA public key cryptosystem", in Proc. of the Second National Conference on Cryptography, CRYPTO-CHINA'92, pp. 105-109. (in Chinese)
25. X. Zhu, "On the structure of binary feedforward inverses with delay 2", J. of Computer Science and Technology, Vol. 4, No. 2, 1989, pp. 163-171.

Appendix

The algorithm for finding **P** was originally described in [17] as "Ra-Rb algorithm". **P** actually is a sequence of linear row transformations. We use the following example with $r = 2$ to show how the algorithm works. First operatea row transformation so that A_0 is changed to a matrix B whose first several rows are linear indepedent and the rest rows are all zero.

$$\begin{pmatrix} A_0 & (0) & (0) \\ A_1 & A_0 & (0) \\ A_2 & A_1 & A_0 \end{pmatrix} \Rightarrow \begin{pmatrix} B & (0) & (0) \\ C & B & (0) \\ D & C & B \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} upperB \\ allzero \end{pmatrix} & \begin{pmatrix} allzero \\ allzero \end{pmatrix} & \begin{pmatrix} allzero \\ allzero \end{pmatrix} \\ \begin{pmatrix} upperC \\ lowerC \end{pmatrix} & \begin{pmatrix} upperB \\ allzero \end{pmatrix} & \begin{pmatrix} allzero \\ allzero \end{pmatrix} \\ \begin{pmatrix} upperD \\ lowerD \end{pmatrix} & \begin{pmatrix} upperC \\ lowerC \end{pmatrix} & \begin{pmatrix} upperB \\ allzero \end{pmatrix} \end{pmatrix} \begin{matrix} (1) \\ \\ (2) \\ \\ (3) \end{matrix}$$

Then we operate the following row transformations to the above matrix: shift (3) to (2), (2) to (1) and (1) to (3). Now we get a new matrix $\begin{pmatrix} A'_0 & (0) & (0) \\ A'_1 & A'_0 & (0) \\ A_{2,2} & A_{2,1} & A_{2,0} \end{pmatrix}$, where $A_{2,2}$, $A_{2,1}$ and $A_{2,0}$ satisfy (6) in Section 2. Next, we continue applying the same procedure to the matrix $\begin{pmatrix} A'_0 & (0) & (0) \\ A'_1 & A'_0 & (0) \end{pmatrix}$. Then we obtain the final matrix we want.

This article was processed using the LaTeX macro package with LLNCS style