On Finite Automaton Public Key Cryptosystems

Tommi Meskanen

Department of Mathematics, University of Turku
FIN-20014, Turku, Finland
and
Turku Centre for Computer Science TUCS
e-mail: tommes@utu.fi

Turku Centre for Computer Science
TUCS Technical Report No 408
August 2001

ISBN 952-12-0837-6
ISSN 1239-1891

Abstract

A public key cryptosystem FAPKC3 by Tao and Chen is presented in details. It is based on automata theory, in particular the automaton inversion problem. Two methods for breaking certain instances of this crytosystem are introduced and also ways of preventing this are discussed.

Keywords: cryptography, finite automata

TUCS Research Group
Theory Group: Mathematical Structures in Computer Science

1 Introduction

Most of the public key cryptosystems created are based on number theoretic problems. There is a widespread interest in cryptosystems based on problems of the other areas of mathematics. The cryptosystem FAPKC3 discussed in this paper is based on automata theory.

The private key consists of two finite automata that are constructed so that their inverses are easily calculated. The private key is the automaton 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. On the other hand, using the inverses of the private key automata we can effectively calculate the inverse.

The inverse of a finite automaton is the automaton that given the initial state and the output of the original automaton outputs the input of the original automaton.

When they present the FAPKC3 cryptosystem, writers do not reveal all the details of how to generate the nonlinear automaton needed [TCC].

First we discuss the generation of the automata according to Tao et al. and then describe the encryption scheme and the signature scheme. Later we present an another way to generate an invertible nonlinear automaton of the suitable form.

We also show how to break the cryptosystem if not a suitable nonlinear automaton is used. This attack may work in some extent to many correctly generated nonlinear automata also.

This cryptosystem is vulnerable to an attack where the attacker knows or guesses right the ending of the plaintext. Using this information the attacker can in most cases easily guess the rest of the plaintext letter at the time. A method for preventing this is presented.

For a more detailed presentation of the subject see [Mes].

2 Invertibility of a Finite Automaton

We start by defining the finite automaton: by *finite automaton* M we mean 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:

$$\begin{aligned} \delta(s, \epsilon) &= s, & \delta(s, \alpha x) &= \delta(\delta(s, \alpha), x), \\ \lambda(s, \epsilon) &= \epsilon, & \lambda(s, x\alpha') &= \lambda(s, x)\lambda(\delta(s, x), \alpha'), \end{aligned}$$

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

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 said to be *weakly invertible with delay* τ if, for any $x_i \in X$, $i = 0, 1, \ldots, \tau$ and $s \in S$, x_0 can be uniquely determined by the state s and the output $\lambda(s, x_0 \cdots x_\tau)$.

For any states $s \in S$ and $s' \in S'$, if

$$\forall \alpha \in X^\omega,\ \exists \alpha_0 \in X^N : \lambda'(s', \lambda(s, \alpha)) = \alpha_0 \alpha \text{ and } |\alpha_0| = \tau,$$

then (s', s) is said to be a *matching pair with delay* τ or we say that s' *matches* s *with delay* τ.

M' is said to be 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.

When automaton M has a weak inverse M' with some delay τ we can find the input x_0 with the help of M' and the outputs $\lambda(s, x_0 \cdots x_\tau)$ and therefore M is weakly invertible with delay τ.

Let φ be a mapping from $Y^k \times X^{h+1}$ to Y. This mapping defines a finite automaton $M =< X, Y, Y^k \times X^h, \delta, \lambda >$ when we have

$$y(i) = \varphi(y(i-1), \ldots, y(i-k), x(i), \ldots, x(i-h)), \qquad i = 0, 1, \ldots,$$

i.e.

$$\begin{aligned} &\delta(< y_{-1}, \ldots, y_{-k}, x_{-1}, \ldots, x_{-h} >, x_0) =< y_0, \ldots, y_{-k+1}, x_0, \ldots, x_{-h+1} >, \\ &\lambda(< y_{-1}, \ldots, y_{-k}, x_{-1}, \ldots, x_{-h} >, x_0) = y_0, \\ &\qquad\qquad\qquad\qquad\qquad\qquad y_0 = \varphi(y_{-1}, \ldots, y_{-k}, x_0, \ldots, x_{-h}). \end{aligned}$$

Now M is said to be an (h, k)*-order memory finite automaton*, denoted by M_φ. If the mapping φ is from X^{h+1} to Y M_φ is said to be an *h-order input memory finite automaton.*

We will define two different combinations of finite automata, $C(M_1, M_2)$ and $C'(M_1, M_2)$. For any two finite automata $M_i =< X_i, Y_i, S_i, \delta_i, \lambda_i >$, $i = 1, 2$, with $Y_1 = X_2$ we denote by $C(M_1, M_2)$ the superposition of M_1 and M_2, i.e. the finite automaton $< X_1, Y_2, S_1 \times S_2, \delta, \lambda >$, where

$$\begin{gathered}\delta(< s_1, s_2 >, x) =< \delta_1(s_1, x), \delta_2(s_2, \lambda_1(s_1, x)) >,\\ \lambda(< s_1, s_2 >, x) = \lambda_2(s_2, \lambda_1(s_1, x)),\\ s_1 \in S_1, \quad s_2 \in S_2, \quad x \in X_1.\end{gathered}$$

Let g be a mapping from $U^r \times V^{p+1}$ to U and f a mapping from W^{t+1} to V. $C'(M_f, M_g)$ is the $(p + t, r)$-order memory automaton defined by

$$\begin{aligned}u(i) =& g(u(i-1), \ldots, u(i-r), f(w(i)), \ldots, w(i-t)), \ldots,\\ & f(w(i-p), \ldots, w(i-p-t)))\\ =& g'(u(i-1), \ldots, u(i-r), w(i), \ldots, w(i-p-t)),\\ & i = 0, 1, \ldots.\end{aligned} \tag{1}$$

Let $M_i =< X_i, Y_i, S_i, \delta_i, \lambda_i >$, $i = 1, 2$, be two finite automata with $X_1 = X_2$. States $s_1 \in S_1$ are $s_2 \in S_2$ are said to be *equivalent*, if $\lambda_1(s_1, \alpha) = \lambda_2(s_2, \alpha)$ holds for any $\alpha \in X_1^N$.

Theorem 1 *Let $s =< u_{-1}, \ldots, u_{-r}, w_{-1}, \ldots, w_{-p-t} >$ be some state of $C'(M_f, M_g)$. If $s_f =< w_{-1}, \ldots, w_{-t} >$, $s_g =< u_{-1}, \ldots, u_{-r}, v_{-1}, \ldots, v_{-p} >$, where*

$$v_i = f(w_i, \ldots, w_{i-t}), \qquad i = -1, \ldots, -p,$$

then the state $< s_f, s_g >$ of an automaton $C(M_f, M_g)$ and the state s of $C'(M_f, M_g)$ are equivalent.

Proof: The proof is very straightforward and is omitted here. ■

In the following we assume that X and Y are column vector spaces over $GF(q)$ of dimension l and m, respectively. Let $M =< X, Y, Y^t \times X^r, \delta, \lambda >$ be a (r, t)-order memory finite automaton over $GF(q)$ defined by

$$y(i) = f(y(i-1), \ldots, y(i-t), x(i), \ldots, x(i-r)), \qquad i = 0, 1, \ldots. \tag{2}$$

Let $x(i)$ and $y(i)$ be column vectors over $GF(q)$ of dimensions l and m, respectively, and τ be a nonnegative integer. The value of the function f is a column vector and later we sometimes understand f as m different functions which all have values in field $GF(q)$.

In the following we define the rules to find automata with invertibility. Parameters r and t define the size of the resulting automaton.

Rule R_a: Let $eq_k(i)$ be an equation in the form of

$$f_k(x(i), \ldots, x(i-r), y(i+k), \ldots, y(i-t)) = \bar{0}. \tag{3}$$

Let φ_k be a transformation on $eq_k(i)$ and the equation in the form

$$eq'_k(i): \quad f'_k(x(i), \ldots, x(i-r), y(i+k), \ldots, y(i-t)) = \bar{0}$$

be the result of the transformation. If $eq_k(i)$ and $eq'_k(i)$ are equivalent, then $eq'_k(i)$ *is said to be obtained from* $eq_k(i)$ *by Rule* R_a *using* φ_k, denoted by

$$eq_k(i) \xrightarrow{R_a[\varphi_k]} eq'_k(i).$$

Rule R_b: Assume that $eq'_k(i)$ is an equation in the form

$$f'_k(x(i), \ldots, x(i-r), y(i+k), \ldots, y(i-t)) = \bar{0}, \tag{4}$$

and that the last $m - r_{k+1}$ components of the left side of $eq'_k(i)$ don't depend on $x(i)$. Let $eq_{k+1}(i)$ be the equation

$$\begin{pmatrix} I'_k f'_k(x(i), \ldots, x(i-r), y(i+k), \ldots, y(i-t)) \\ I''_k f'_k(x(i+1), \ldots, x(i+1-r), y(i+1+k), \ldots, y(i+1-t)) \end{pmatrix} = \bar{0},$$

where I'_k and I''_k are submatrices of the first r_{k+1} and of the last $m - r_{k+1}$ rows of the $(m \times m)$-identity matrix, respectively. Now $eq_{k+1}(i)$ is said to be *obtained from* $eq'_k(i)$ *by Rule* R_b, denoted by

$$eq'_k(i) \xrightarrow{R_b[r_{k+1}]} eq_{k+1}(i).$$

The equation $eq_{k+1}(i)$, result of applying Rule R_b to $eq'_k(i)$, is still in the form of

$$f_{k+1}(x(i), \ldots, x(i-r), y(i+k+1), \ldots, y(i-t)) = \bar{0}$$

on which Rule R_a can be applied.

Intuitively the rule R_b makes the delay longer and and the rule R_a makes sure we can use the rule R_b again.

Denote the chain of transformations

$$eq_j(i) \xrightarrow{R_a[\varphi_j]} eq'_j(i), \qquad eq'_j(i) \xrightarrow{R_b[r_{j+1}]} eq_{j+1}(i), \qquad j = 0, 1, \ldots, k$$

by

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq'_0(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_k]} eq'_k(i) \xrightarrow{R_b[r_{k+1}]} eq_{k+1}(i).$$

Let $eq_0(i)$ be the equation

$$f(y(i-1), \ldots, y(i-t), x(i), \ldots, x(i-r)) - y(i) = \bar{0}, \tag{5}$$

where f is defined as in equations (2) and assume that

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq'_0(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_{\tau-1}]} eq'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i).$$

It is easy to prove the following properties:

Property A. For any k, $0 \le k < \tau$, equations $eq_k(i)$ have a solution if and only if equations $eq'_k(i)$ have a solution.

Property B. For any k, $0 \le k < \tau$, if equations $eq'_k(i),\ i = b, b+1, \ldots,$ have a solution then equations $eq_{k+1}(i),\ i = b, b+1, \ldots$ have a solution.

Property C. For any k, $0 \le k < \tau$, if equations $eq_{k+1}(i),\ i = b, b+1, \ldots,$ have a solution then equations $eq'_k(i),\ i = b+1, b+2, \ldots$ have a solution.

From property A and property B we have

Property D. If equations $eq_0(i),\ i = b, b+1, \ldots,$ have a solution then equations $eq_\tau(i),\ i = b, b+1, \ldots$ have a solution.

From property A and property C we have

Property E. If equations $eq_\tau(i),\ i = b, b+1, \ldots,$ have a solution then equations $eq_0(i),\ i = b+\tau, b+\tau+1, \ldots$ have a solution.

When C is a matrix with m columns, by $Ceq'_k(i)$ we denote the equation we get by multiplying both sides of $eq'_k(i)$ on the left by C. It is easy to show the following property:

Property F. For any $k, 0 \le k < \tau$, and $b < e$, we have that equations $eq_k(i), i = b, b+1, \ldots, e$ have a solution if and only if equations

$$\begin{array}{ll} eq_{k+1}(i), & i = b, b+1, \ldots, e-1, \\ & I'_k eq'_k(e), \\ & I''_k eq'_k(b) \end{array}$$

have a solution.

By applying Property F repeatedly, we get

Property G. Equations $eq_0(i), i = 0, 1, \ldots, \tau$ have a solution if and only if equations

$$\begin{array}{cccc} & eq_\tau(0), & & \\ I'_0 eq'_0(\tau), & I'_1 eq'_1(\tau-1), & \ldots, & I'_{\tau-1} eq'_{\tau-1}(1), \\ I''_0 eq'_0(0), & I''_1 eq'_1(0), & \ldots, & I''_{\tau-1} eq'_{\tau-1}(0) \end{array}$$

have a solution.

Theorem 2 *Let M be defined by (2). Let $eq_0(i)$ be defined by (5) and*

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq'_0(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_{\tau-1}]} eq'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i).$$

If for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, the equation $eq_\tau(i)$ has a solution $x(i)$, then M is a weak inverse with delay τ.

Proof: Assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $f_\tau(x(i), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)) = \bar{0}$ has a solution $x(i)$. We choose an arbitrary solution and denote it by $f^*_\tau(x(i-1), \ldots, x(i-$

$r), y(i+\tau), \ldots, y(i-t))$. Below we construct a finite automaton M^{**} and prove that M is a weak inverse with delay τ of M^{**}. Let $M^* =< Y, X, X^r \times Y^{\tau+t}, \delta^*, \lambda^* >$ be a $(\tau+t, r)$-order memory finite automaton defined by

$$x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y'(i), \ldots, y'(i-\tau-t)), \qquad i = 0, 1, \ldots. \quad (6)$$

We take a subautomaton M^{**} of M^* for which the input alphabet is Y, the output alphabet is X and the state alphabet is

$$\{\delta^*(s^*, y_0 \cdots y_{\tau-1}) \mid s^* \in X^r \times Y^{\tau+t},\ y_0, \ldots, y_{\tau-1} \in Y\},$$

i.e. the states that the automaton M^* can reach from some state in τ steps. Let

$$s_0^{**} =< x(-1), \ldots, x(-r), y'(-1), \ldots, y'(-\tau-t) >$$

be an arbitrary state of M^{**} and $y'(0), y'(1), \cdots \in Y$, $x'(0), x'(1), \cdots \in X$ such that

$$x(0)x(1)\cdots = \lambda^*(s_0^{**}, y'(0)y'(1)\cdots).$$

From the construction of M^{**} there are $y'(-\tau-t-1), \ldots, y'(-2\tau-t) \in Y$ and $x(-r-1), \ldots, x(-r-\tau) \in X$ such that

$$s_{-\tau}^* =< x(-\tau-1), \ldots, x(-\tau-r), y'(-\tau-1), \ldots, y'(-2\tau-t) >$$

is a state of M^* and

$$\delta^*(s_{-\tau}^*, y'(-\tau)\cdots y'(-1)) = s_0^{**},$$

implying

$$\lambda^*(s_{-\tau}^*, y'(-\tau)\cdots y'(-1)) = x(-\tau)\cdots x(-1).$$

It follows that

$$x(-\tau)\cdots x(-1)x(0)x(1)\cdots = \lambda^*(s_{-\tau}^*, y'(-\tau)\cdots y'(-1)y'(0)y(1)\cdots). \quad (7)$$

From equations (6) we have

$$x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y'(i), \ldots, y'(i-\tau-t)),$$
$$i = -\tau, \ldots, -1, 0, 1, \ldots.$$

By denoting $y(i+\tau) = y'(i)$ for any integer i, we have

$$x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)),$$

that is, $x(i)$ is a solution of $eq_\tau(i)$, i.e.

$$f_\tau(x(i), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)) = 0,$$

for $i = -\tau, \ldots, -1, 0, 1 \ldots$.

Because the set of simultaneous equations $eq_\tau(i),\ i = -\tau, \ldots, -1, 0, 1 \ldots$, has a solution, from Property E we have that the system of equations $eq_0(i),\ i = 0, 1, \ldots$ has a solution. So the equation (2) holds and we have

$$y'(-\tau) \cdots y'(-1)y'(0)y'(1) \cdots = y(0)y(1) \cdots = \lambda(s_0, x(0)x(1) \cdots),$$

where

$$\begin{aligned} s_0 &=< y(-1), \ldots, y(-t), x(-1), \ldots, x(-r) > \\ &=< y'(-\tau-1), \ldots, y'(-\tau-t), x(-1), \ldots, x(-r) > . \end{aligned}$$

It follows from (7) that the state s_0 of M matches the state s_0^{**} of M^{**} with delay τ. Thus M is a weak inverse with delay τ of M^{**}. Therefore, M is a weak inverse with delay τ. ■

From the proof of the Theorem above we have

Theorem 3 *Let M be defined by (2). Let $eq_0(i)$ be defined by (5) and*

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq_0'(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_{\tau-1}]} eq_{\tau-1}'(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i).$$

Assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has a solution $x(i)$. Let M^{**} be defined as in the proof of Theorem 2. Now M is a weak inverse with delay τ of M^{**}. Furthermore, for any state $s_0^{**} =< x(-1), \ldots, x(-r), y'(-1), \ldots, y'(-\tau-t) >$ of M^{**} the state $s_0 =< y'(-\tau-1), \ldots, y'(-\tau-t), x(-1), \ldots, x(-r) >$ of M matches the state s_0^{**} with delay τ, and for any $y'(0), y'(1), \cdots \in Y$, if

$$x(0)x(1) \cdots = \lambda^*(s_0^{**}, y'(0)y'(1) \cdots),$$

then

$$y'(-\tau) \cdots y'(-1)y'(0)y'(1) \cdots = \lambda(s_0, x(0)x(1) \cdots).$$

We can prove also a result to the other direction.

Theorem 4 *Let M be defined by (2). Let $eq_0(i)$ be defined by (5) and*

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq_0'(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_{\tau-1}]} eq_{\tau-1}'(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i).$$

If for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has at most one solution $x(i)$, then M is weakly invertible with delay τ.

Proof: Assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has at most one solution $x(i)$. For any initial state

$$s_0 =< y(-1), \ldots, y(-t), x(-1), \ldots, x(-r) >$$

of M and any infinite input sequence $x(0), x(1), \ldots$ of M, denote

$$y(0)y(1)\cdots = \lambda(s_0, x(0)x(1)\cdots).$$

From the equations (2) and (5), equations $eq_0(i),\ i = 0, 1, \ldots$ have a solution. Using Property D, $eq_\tau(i)$ holds for $i = 0, 1, \ldots$. It immediately follows that for such values of $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has a unique solution $x(i),\ i = 0, 1, \ldots$. Thus $x(0)$ is uniquely determined by the initial state s_0 and the output sequence $y(0)\cdots y(\tau)$. Therefore, M is weakly invertible with delay τ. ■

Let $eq_0(i)$ be defined by (5). Let

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq_0'(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_{\tau-1}]} eq_{\tau-1}'(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i).$$

Assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has at most one solution $x(i)$. Therefore, there exists a mapping f_τ^* from $X^r \times Y^{\tau+t+1}$ to X, such that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, if $eq_\tau(i)$ has a solution $x(i)$, then $x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t))$.

For any initial state $s_0 =< y(-1), \ldots, y(-t), x(-1), \ldots, x(-r) >$ and for any infinite input sequence $x(0)x(1)\cdots$ of M, denote

$$y(0)y(1)\cdots = \lambda(s_0, x(0)x(1)\cdots).$$

From the proof of the previous Theorem, we have that for such values of $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, the set of simultaneous equations $eq_\tau(i)$ has a unique solution $x(i),\ i = 0, 1, \ldots$. It follows immediately that

$$x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)), \qquad i = 0, 1, \ldots. \tag{8}$$

We denote $x'(j+\tau) = x(j)$ for any j. Now (8) can be rewritten as the following

$$x(i) = f_\tau^*(x'(i-1), \ldots, x'(i-r), y(i), \ldots, y(i-\tau-t)), \qquad i = \tau, \tau+1, \ldots. \tag{9}$$

We construct a finite automaton $M_{eq_\tau}^\tau =< Y, X, S^*, \delta^*, \lambda^* >$ according to (9), where

$$\begin{gathered}
S^* = X^r \times Y^{\tau+t} \times \{0, 1, \ldots, \tau\}, \\
\delta^*(< x'_{-1}, \ldots, x'_{-r}, y_{-1}, \ldots, y_{-\tau-t}, c >, y_0) \\
= \begin{cases} < x'_{-1}, \ldots, x'_{-r}, y_0, \ldots, y_{-\tau-t+1}, c+1 >, & \text{if } c < \tau, \\ < x'_0, \ldots, x'_{-r+1}, y_0, \ldots, y_{-\tau-t+1}, c >, & \text{if } c = \tau, \end{cases} \qquad (10) \\
\lambda^*(< x'_{-1}, \ldots, x'_{-r}, y_{-1}, \ldots, y_{-\tau-t}, c >, y_0) = \begin{cases} g, & \text{if } c < \tau, \\ x'_0 & \text{if } c = \tau,, \end{cases} \\
x'_0 = f_\tau^*(< x'_{-1}, \ldots, x'_{-r}, y_0, \ldots, y_{-\tau-t}).
\end{gathered}$$

Here g means garbage. Let $s_0^* =< x(-1), \ldots, x(-r), y(-1), \ldots, y(-\tau - t), 0 >$ be a state of $M^\tau_{eq_\tau}$, where $y(-t-1), \ldots, y(-\tau - t)$ are arbitrary elements of Y. From the construction of $M^\tau_{eq_\tau}$, using (8) it is easy to verify that

$$\lambda^*(s_0^*, y(0)y(1)\cdots) = x''(0)\ldots x''(\tau-1)x(0)x(1)\cdots$$

for some $x''(0), \ldots, x''(\tau-1)$ in X. Thus s_0^* matches s_0 with delay τ. Therefore $M^\tau_{eq_\tau}$ is a weak inverse with delay τ of M.

We have proven the following result:

Theorem 5 *Let M be a (r,t)-order memory finite automaton defined by (2). Let $eq_0(i)$ be defined by (5) and*

$$eq_0(i) \stackrel{R_a[\varphi_0]}{\longrightarrow} eq_0'(i) \stackrel{R_b[r_1]}{\longrightarrow} eq_1(i) \stackrel{R_a[\varphi_1]}{\longrightarrow} \cdots \stackrel{R_a[\varphi_{\tau-1}]}{\longrightarrow} eq_{\tau-1}'(i) \stackrel{R_b[r_\tau]}{\longrightarrow} eq_\tau(i).$$

Let f_τ^* be a mapping from $X^r \times Y^{\tau+t+1}$ to X. Assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has at most one solution $x(i)$ and that $x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t))$ is such a solution when the solution exists. Let $M^\tau_{eq_\tau}$ be defined by (10). Now $M^\tau_{eq_\tau}$ is a weak inverse with delay τ of M. Furthermore, for any state $s_0 =< y(-1), \ldots, y(-t), x(-1), \ldots, x(-r) >$ of M, the state $s_0^* =< x(-1), \ldots, x(-r), y(-1), \ldots, y(-\tau - t), 0 >$ of $M^\tau_{eq_\tau}$ matches s_0 with delay τ, for all $y(-t-1), \ldots, y(-\tau - t)$ in Y.

Corollary: *Let M be a (r,t)-order memory finite automaton defined by (2). Let $eq_0(i)$ be defined by (5) and*

$$eq_0(i) \stackrel{R_a[\varphi_0]}{\longrightarrow} eq_0'(i) \stackrel{R_b[r_1]}{\longrightarrow} eq_1(i) \stackrel{R_a[\varphi_1]}{\longrightarrow} \cdots \stackrel{R_a[\varphi_{\tau-1}]}{\longrightarrow} eq_{\tau-1}'(i) \stackrel{R_b[r_\tau]}{\longrightarrow} eq_\tau(i).$$

Let f_τ^* be a mapping from $X^r \times Y^{\tau+t+1}$ to X. Assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t)$, $eq_\tau(i)$ has at most one solution $x(i)$ and that $x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i-t))$ is such a solution when the solution exists. Let M^* be a $(\tau+t, r)$-order memory finite automaton defined by

$$x(i) = f_\tau^*(x(i-1), \ldots, x(i-r), y'(i), \ldots, y'(i-\tau-t)), \qquad i = \tau, \tau+1, \ldots.$$

For any state $s_0 =< y(-1), \ldots, y(-t), x(-1), \ldots, x(-r) >$ and any input $x(0)x(1)\cdots$ of M, if

$$y(0)y(1)\cdots = \lambda(s_0, x(0)x(1)\cdots)$$

then

$$\begin{aligned} \lambda^*(< x(-1), \ldots, x(-r), y(\tau-1), \ldots, y(-t) >, y(\tau)y(\tau+1)\cdots) \\ = x(0)x(1)\cdots, \end{aligned} \tag{11}$$

where λ^* is the output function of M^*.

Below we turn on the special case $m = l$.

Let $M =< X, Y, X^r, \delta, \lambda >$ be a r-order input memory finite automaton over $GF(q)$, defined by

$$y(i) = f(x(i), \ldots, x(i-r)), \qquad i = 0, 1, \ldots \tag{12}$$

and $\tau \le r$. Let $eq_0(i)$ be the equation

$$f(x(i), \ldots, x(i-r)) - y(i) = \bar{0}. \tag{13}$$

Theorem 6 *Assume that $m = l$. Let M be defined by (12) and $eq_0(i)$ by (13). Assume that*

$$eq_0(i) \xrightarrow{R_a[\varphi_0]} eq'_0(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[\varphi_1]} \cdots \xrightarrow{R_a[\varphi_{\tau-1}]} eq'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i).$$

Let f^*_τ be a mapping from $X^r \times Y^{\tau+1}$ to X. Furthermore assume that for any parameters $x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i)$, $eq_\tau(i)$ has a unique solution $x(i) = f^*_\tau(x(i-1), \ldots, x(i-r), y(i+\tau), \ldots, y(i))$. Let M^* be a (τ, r)-order memory finite automaton defined by

$$x(i) = f^*_\tau(x(i-1), \ldots, x(i-r), y'(i), \ldots, y'(i-\tau)), \qquad i = 0, 1, \ldots. \tag{14}$$

Then we have:

(a). M^* is a weak inverse with delay τ of M. For any $x(-1), \ldots, x(-\tau-r) \in X$, the state $< x(-\tau-1), \ldots, x(-\tau-r), y(-1), \ldots, y(-\tau) >$ of M^* matches the state $< x(-1), \ldots, x(-r) >$ of M with delay τ, when

$$y(-\tau) \cdots y(-1) = \lambda(< x(-\tau-1), \cdots, x(-\tau-r) >, x(-\tau) \cdots x(-1)).$$

(b). M is a weak inverse with delay τ of M^*. Furthermore, for any $x(-1), \ldots, x(-r), x'(-r+\tau-1), \ldots, x'(-r) \in X$ and any $y'(-1), \ldots, y'(-\tau) \in Y$ the state $< x(-1), \ldots, x(-r+\tau), x'(-r+\tau-1), \ldots, x'(-r) >$ of M matches the state $< x(-1), \ldots, x(-r), y'(-1), \ldots, y'(-\tau) >$ of M^* with delay τ.

Proof: The proof of this Theorem uses the same ideas as previous Theorems and is omitted here. ■

Corollary: *Under the hypothesis of Theorem 6, for any initial state $s_0 =< x(-1), \ldots, x(-r) >$ and any infinite input sequence $x(0), x(1) \cdots \in X$ of M, if*

$$y(0)y(1) \cdots = \lambda(s_0, x(0)x(1) \cdots),$$

then

$$\lambda^*(< x(-1), \ldots, x(-r), y(\tau-1), \ldots, y(0) >, y(\tau)y(\tau+1) \cdots) = x(0)x(1) \cdots .$$

3 Generating an invertible automaton

We still consider (r,t)-memory automata with input alphabet X a vector space of dimension l and output alphabet Y of dimension m over the field $GF(q)$. Let τ be a non-negative integer and $\tau \le r$.

In the previous chapter we considered the existence of invertible automata in general. In the following we consider more specific equations $eq_k(i)$ and transformations R_a and R_b.

We denote the equation (3) by $eq_k(i)$ and consider its left side $f_k(x(i),\ldots, x(i-r), y(i+k),\ldots,y(i-t))$. In the following we only consider cases where the function f_k can be written as a polynomial of components of $x(i),\ldots,x(i-r)$. Let s be some polynomial of components of $x(i),\ldots,x(i-r)$ and $p \ge 0$. If s has components from parameters $x(i)$ and $x(i-p)$, but not from parameters $x(i-p-1),\ldots,x(i-r)$, we say that s has *span* p. We denote $l_0 = (q^l-1)$ and $l_p = q^{l(p-1)}(q^l-1)^2$ when $p \ge 1$. We denote the l_p-vector with as its elements all the products of components of $x(i),\ldots,x(i-p)$ with span p by $\pi(x(i),\ldots,x(i-p))$. Using these notations we can write the equation $eq_k(i)$ as

$$f_k(x(i),\ldots,x(i-r),y(i+k),\ldots,y(i-t)) = F_{0k}(y(i+k),\ldots,y(i-t)) \tag{15}$$

$$+\sum_{p=0}^{r}\sum_{j=0}^{r-p} F_{pjk}(y(i+k),\ldots,y(i-t))\pi(x(i-j),\ldots,x(i-j-p)),$$

where $F_{0k}(y(i+k),\ldots,y(i-t))$ is an m-column vector, $F_{pjk}(y(i+k),\ldots,y(i-t))$ is an $(m\times l_p)$-matrix and $0 \le p \le r,\ 0 \le j \le r-p$. The elements of these matrices are polynomials of parameters $y(i+k),\ldots,y(i-t)$. Let

$$\begin{gathered} G_{pk}(y(i+k),\ldots,y(i-t)) \\ = \left(F_{p0k}(y(i+k),\ldots,y(i-t)),\quad\cdots,\quad F_{p,r-p,k}(y(i+k),\ldots,y(i-t))\right), \\ p=0,1,\ldots,r, \qquad k=0,1,\ldots,\tau. \end{gathered}$$

and

$$\pi_p(x(i),\ldots,x(i-r)) = \begin{pmatrix} \pi(x(i),\ldots,x(i-p)) \\ \vdots \\ \pi(x(i-j),\ldots,x(i-p-j)) \\ \vdots \\ \pi(x(i-r+p),\ldots,x(i-r)) \end{pmatrix}, \quad p=0,1,\ldots,r.$$

Now (15) can be written as

$$f_k(x(i),\dots,x(i-r),y(i+k),\dots,y(i-t)) = F_{0k}(y(i+k),\dots,y(i-t)) \tag{16}$$

$$+\sum_{p=0}^{r} G_{pk}(y(i+k),\dots,y(i-t))\pi_p(x(i),\dots,x(i-r)).$$

The elements of the ordered set

$$G_k(i) = \{F_{0k}(y(i+k),\dots,y(i-t)),\\ G_{pk}(y(i+k),\dots,y(i-t)) \mid p=0,1,\dots,r\},$$

are said to be the *coefficient matrices* of $eq_k(i)$.

In the same way we denote the equation (4) by $eq'_k(i)$ and now can write its left side as

$$f'_k(x(i),\dots,x(i-r),y(i+k),\dots,y(i-t)) = F'_{0k}(y(i+k),\dots,y(i-t)) \tag{17}$$

$$+\sum_{p=0}^{r}\sum_{j=0}^{r-p} F'_{pjk}(y(i+k),\dots,y(i-t))\pi(x(i-j),\dots,x(i-j-p))$$

or as

$$f'_k(x(i),\dots,x(i-r),y(i+k),\dots,y(i-t)) = F'_{0k}(y(i+k),\dots,y(i-t)) \tag{18}$$

$$+\sum_{p=0}^{r} G'_{pk}(y(i+k),\dots,y(i-t))\pi_p(x(i),\dots,x(i-r)),$$

where $F'_{0k}(y(i+k),\dots,y(i-t))$ is an m-column vector, $F'_{pjk}(y(i+k),\dots,y(i-t))$ is an $(m\times l_p)$-matrix, $0\le p\le r,\ 0\le j\le r-p$, and

$$G'_{pk}(y(i+k),\dots,y(i-t))\\ = \left(F'_{p0k}(y(i+k),\dots,y(i-t)) \quad \cdots \quad F'_{p,r-p,k}(y(i+k),\dots,y(i-t))\right),\\ p=0,1,\dots,r, \qquad k=0,1,\dots,\tau.$$

The elements of the ordered set

$$G'_k(i) = \{F'_{0k}(y(i+k),\dots,y(i-t)),\\ G'_{pk}(y(i+k),\dots,y(i-t)) \mid p=0,1,\dots,r\},$$

are said to be the *coefficient matrices* of $eq'_k(i)$.

If we by a transformation φ_k mean multiplying both sides of an equation by a matrix $P_k(y(i+k),\dots,y(i-t))$, we can write $eq_k(i) \overset{R_a[P_k]}{\longrightarrow} eq'_k(i)$ instead of $eq_k(i) \overset{R_a[\varphi_k]}{\longrightarrow} eq'_k(i)$. In this case the rule R_a is said to be *linear*.

In the following the rule R_a is always linear. We denote the group of all matrices with their elements polynomials over $GF(q)$ by $MP(q)$.

Using notations above we can write the linear rule R_a as

Rule R_a**:** Let $G_{pk}(y(i+k),\dots,y(i-t)) \in MP(q)$ be $(m \times l_p(r-p+1))$-matrices, $0 \le p \le r$ and $F_{0k}(y(i+k),\dots,y(i-t)) \in MP(q)$ be an m-column vector. We write

$$\begin{aligned}G_k(i) =& \{F_{0k}(y(i+k),\dots,y(i-t)),\\ & G_{pk}(y(i+k),\dots,y(i-t)) \mid p=0,1,\dots,r\},\\ G'_k(i) =& \{F'_{0k}(y(i+k),\dots,y(i-t)),\\ & G'_{pk}(y(i+k),\dots,y(i-t)) \mid p=0,1,\dots,r\}.\end{aligned}$$

Let $P_k(y(i+k),\dots,y(i-t)) \in MP(q)$ be a nonsingular $(m \times m)$-matrix for all $y(i+k),\dots,y(i-\tau)$,

$$\begin{aligned}& G'_{pk}(y(i+k),\dots,y(i-t))\\ =& P_k(y(i+k),\dots,y(i-t))G_{pk}(y(i+k),\dots,y(i-t)),\end{aligned}$$

when $0 \le p \le r$ and

$$\begin{aligned}& F'_{0k}(y(i+k),\dots,y(i-t))\\ =& P_k(y(i+k),\dots,y(i-t))F_{0k}(y(i+k),\dots,y(i-t)).\end{aligned}$$

Now we say that $G'_k(i)$ is *obtained from* $G_k(i)$ *by rule* R_a *using* P_k denoted by

$$G_k(i) \overset{R_a[P_k]}{\longrightarrow} G'_k(i).$$

If we denote by $G_k(i)$ the coefficient matrices of equation $eq_k(i)$ and by $G'_k(i)$ the coefficient matrices of equation $eq'_k(i)$ the transformations $eq_k(i) \overset{R_a[P_k]}{\longrightarrow} eq'_k(i)$ and $G_k(i) \overset{R_a[P_k]}{\longrightarrow} G'_k(i)$ are the same transformation.

Because the matrix P_k used in the transformation was nonsingular the equations have the same solutions.

Also the rule R_b can now be rewritten as

Rule R_b**:** Let $G'_{pk}(y(i+k),\dots,y(i-t)) \in MP(q)$ be $(m \times l_p(r-p+1))$-matrices, $0 \le p \le r$ and $F'_{0k}(y(i+k),\dots,y(i-t)) \in MP(q)$ be an m-column vector. Let the first l_p columns in the last $m-r_{k+1}$ rows of matrices $G'_{pk}(y(i+k),\dots,y(i-t))$ be zeroes if $r_{k+1} < m$. We denote by $G_{p,k+1}(y(i+k+1),\dots,y(i-t))$ for all $0 \le p \le r$ the matrices we get by moving last $m-r_{k+1}$ rows of matrix $G'_{pk}(y(i+k),\dots,y(i-t))$ l_p columns to the left, adding zeroes to right and replacing all variables $y(j)$ in the last $m-r_{k+1}$ rows by $y(j+1)$. We also denote by $F_{0,k+1}(y(i+k+1),\dots,y(i-t))$ the matrix we get by replacing the variable $y(j)$ in the last $m-r_{k+1}$ rows of $F'_{0k}(y(i+k),\dots,y(i-t))$ by $y(j+1)$. Now we say that $G_{k+1}(i)$ is *obtained from* $G'_k(i)$ *by rule* R_a and denote

$$G'_k(i) \overset{R_b[r_{k+1}]}{\longrightarrow} G_{k+1}(i),$$

$$\left(\begin{array}{c|c} \multicolumn{2}{c}{G'_{pk1}(y(i+k),\dots,y(i-t))} \\ \hline \underbrace{\quad 0 \quad}_{l_p} & G'_{pk2}(y(i+k),\dots,y(i-t)) \end{array}\right) \begin{array}{l} \}r_{k+1} \\ \\ \end{array}$$

$$\xrightarrow{R_b[r_k+1]} \left(\begin{array}{c|c} \multicolumn{2}{c}{G'_{pk1}(y(i+k),\dots,y(i-t))} \\ \hline G'_{pk2}(y(i+k+1),\dots,y(i-t+1)) & \underbrace{\quad 0 \quad}_{l_p} \end{array}\right) \begin{array}{l} \}r_{k+1} \\ \\ \end{array}$$

Figure 1: Rule R_b.

where

$$\begin{aligned} G_{k+1}(i) = \{&F_{0,k+1}(y(i+k+1),\dots,y(i-t)), \\ &G_{p,k+1}(y(i+k+1),\dots,y(i-t)) \mid p = 0,1,\dots,r\}. \end{aligned}$$

The picture 1 shows how rule R_b works. This rule corresponds to the rule R_b in the previous chapter.

If we denote by $G_{k+1}(i)$ the coefficient matrices of equation $eq_{k+1}(i)$ and by $G'_k(i)$ the coefficient matrices of equation $eq'_k(i)$ the transformations $eq'_k(i) \xrightarrow{R_b[r_{k+1}]} eq_{k+1}(i)$ and $G'_k(i) \xrightarrow{R_b[r_{k+1}]} G_{k+1}(i)$ are the same transformation.

We notice that if the function

$$\sum_{p=0}^{r} F_{p0\tau}(y(i+\tau),\dots,y(i-t))\pi(x(i),\dots,x(i-p))$$

of variable $x(i)$ is a surjection for all values of $x(i-1),\dots,x(i-r),y(i+\tau),\dots,y(i-t)$, the condition of theorem 2, "for all values of $x(i-1),\dots,x(i-r),y(i+\tau),\dots,y(i-t)$ equation $eq_\tau(i)$ has a solution $x(i)$", is true. In a similar fashion, if for all values of variables $x(i-1),\dots,x(i-r),y(i+\tau),\dots,y(i-t)$ function

$$\sum_{p=0}^{r} F_{p0\tau}(y(i+\tau),\dots,y(i-t))\pi(x(i),\dots,x(i-p))$$

of variable $x(i)$ is an injection, the condition of theorem 4, "for all values of variables $x(i-1),\dots,x(i-r),y(i+\tau),\dots,y(i-t)$ equation $eq_\tau(i)$ has at most one solution $x(i)$", is true.

Let G be an incomplete specified $(m \times n(\tau+1))$-matrix and $0 \leq k \leq \tau$. If there exists $0 = r_0 \leq r_1 \leq \cdots \leq r_k \leq m$, so that when $r_i < r_{i+1}$, at rows from $r_i + 1$ to r_{i+1} of matrix G, elements at first $n(\tau + 1 - i)$ columns are defined and the last ni elements are not defined then we say that G is an (n, k)-*echelon matrix*. Clearly the $r_0, r_1, \ldots, r_k$ above are unique. The number r_i, $0 \leq i \leq k$, is called the i-*height* of matrix G. If G is (n, k)-echelon matrix with i-height r_i when $0 \leq i \leq k$ then G on also a $(n, k+1)$-echelon matrix with i-height r_i when $0 \leq i \leq k$ and with $(k+1)$-height m.

Example: The incomplete specified $(6 \times 8n)$-matrix

$$\begin{pmatrix} G_{11} & G_{12} & G_{13} & G_{14} & G_{15} & G_{16} & G_{17} & G_{18} \\ G_{21} & G_{22} & G_{23} & G_{24} & G_{25} & G_{26} & G_{27} & * \\ G_{31} & G_{32} & G_{33} & G_{34} & G_{35} & G_{36} & G_{37} & * \\ G_{41} & G_{42} & G_{43} & G_{44} & G_{45} & G_{46} & * & * \\ G_{51} & G_{52} & G_{53} & * & * & * & * & * \\ G_{61} & G_{62} & * & * & * & * & * & * \end{pmatrix},$$

where every G_{ij} is an $(1 \times n)$-matrix and "$*$" stands for an undefined matrix, is an $(n, 7)$-echelon matrix. It has i-heights $r_0 = 0$, $r_1 = 1$, $r_2 = 3$, $r_3 = 4$, $r_4 = 4$, $r_5 = 4$, $r_6 = 5$ and $r_7 = 6$. Even if previous matrix would have been totally defined ($r_0 = \cdots = r_8 = 0$, $r_9 = 6$), or totally undefined ($r_0 = 0$, $r_1 = 6$), it would still be an echelon matrix by the definition.

To use the echelon matrices we can modify rules R_a and R_b into the following:

Rule R_a: (for echelon matrices) Let $G_{pk}(y(i+k), \ldots, y(i-t)) \in MP(q)$ be an (l_p, k)-echelon-$(m \times l_p(\tau+1))$-matrix for all $0 \leq p \leq r - \tau$ with the same j-heights and $0 \leq k \leq \tau$. Let the matrix $G_{0k}(y(i+k), \ldots, y(i-t))$ have k-height r_k and $P_k(y(i+k), \ldots, y(i-t)) \in MP(q)$ be an $(m \times m)$-matrix

$$\begin{pmatrix} I_{r_k} & 0 \\ P_{k1}(y(i+k), \ldots, y(i-t)) & P_{k2}(y(i+k), \ldots, y(i-t)) \end{pmatrix},$$

that is nonsingular for all values of variables $y(i+k), \ldots, y(i-t)$ and where I_{r_k} is the $(r_k \times r_k)$-identity matrix. Let $G'_{pk}(y(i+k), \ldots, y(i-t))$ be the matrix you get from the matrix $G_{pk}(y(i+k), \ldots, y(i-t))$ when you multiply it from the left by matrix $P_k(y(i+k), \ldots, y(i-t))$, that is

$$\begin{aligned} &G'_{pk}(y(i+k), \ldots, y(i-t)) \\ &= P_k(y(i+k), \ldots, y(i-t)) G_{pk}(y(i+k), \ldots, y(i-t)), \end{aligned}$$

when $0 \leq p \leq r - \tau$. Now we say that $G'_k(i)$ is *obtained from $G_k(i)$ by rule R_a using matrix P_k* and denote

$$G_k(i) \xrightarrow{R_a[P_k]} G'_k(i),$$

where

$$G_k(i) = \{G_{pk}(y(i+k), \ldots, y(i-t)) \mid p = 0, 1, \ldots, r-\tau\},$$
$$G'_k(i) = \{G'_{pk}(y(i+k), \ldots, y(i-t)) \mid p = 0, 1, \ldots, r-\tau\}.$$

When we calculate the items of matrix $P_k G_{pk}$ we define $x \cdot u = u \cdot x = u$, $0 \cdot u = u \cdot 0 = 0$ and $y + u = u + y = u$, where u stands for undefined, $x, y \in GF(q)$ and $x \neq 0$.

Let $G_{pk}(y(i+k), \ldots, y(i-t)) \in MP(q)$ be an (l_p, k)-echelon-$(m \times l_p(\tau+1))$-matrix for all $0 \le p \le r-\tau$ and $0 \le k \le \tau$. If $G_k(i) \xrightarrow{R_a[P_k]} G'_k(i)$ then for all p, $0 \le p \le r-\tau$ matrix $G'_{pk}(y(i+k), \ldots, y(i-t)) \in MP(q)$ is also an (l_p, k)-echelon-$(m \times l_p(\tau+1))$-matrix and its j-height is the same as of $G_{pk}(y(i+k), \ldots, y(i-t))$, for all $0 \le j \le k$ and the first r_k rows in both matrices $G'_{pk}(y(i+k), \ldots, y(i-t))$ and $G_{pk}(y(i+k), \ldots, y(i-t))$ are the same.

The difference between this rule and the previous rule R_a is that here the matrix P_k is more specified. Also some of the coefficient matrices are missing and some are truncated. This is also true for the following rule:

Rule R_b: (for echelon matrices) Let $G'_{pk}(y(i+k), \ldots, y(i-t)) \in MP(q)$ be a (l_p, k)-echelon-$(m \times l_p(\tau+1))$-matrix for all $0 \le p < r-\tau$ with the same j-height and $0 \le k < \tau$. We assume that for some r_{k+1}, $r_k \le r_{k+1} \le m$, in the last $m - r_{k+1}$ rows of matrices $G'_{pk}(y(i+k), \ldots, y(i-t))$ the first l_p items are zeroes for all $0 \le p \le r-\tau$, if $r_{k+1} < m$. For all p, $0 \le p \le r-\tau$, we denote by $G_{p,k+1}(y(i+k+1), \ldots, y(i-t))$ the matrix we get from matrix $G'_{pk}(y(i+k), \ldots, y(i-t))$ in the following way: we move the last $m - r_{k+1}$ rows l_p positions left, the new items at the left are set undefined and replace all variables $y(j)$ in the last $m - r_{k+1}$ rows by variable $y(j+1)$. Now we say that $G_{k+1}(i)$ is *obtained from $G'_k(i)$ by rule R_b* and denote

$$G'_k(i) \xrightarrow{R_b[r_{k+1}]} G_{k+1}(i),$$

where

$$G'_k(i) = \{G'_{pk}(y(i+k-1), \ldots, y(i-t)) \mid p = 0, 1, \ldots, r-\tau\},$$
$$G_{k+1}(i) = \{G_{p,k+1}(y(i+k-1), \ldots, y(i-t)) \mid p = 0, 1, \ldots, r-\tau\}.$$

If we obtained $G_{k+1}(i)$ from $G'_k(i)$ by rule R_b then for all p, $0 \le p \le r-\tau$, the matrix $G_{p,k+1}(y(i+k+1), \ldots, y(i-t)) \in MP(q)$ is an $(l_p, k+1)$-echelon-$(m \times l_p(\tau+1))$-matrix with j-heights, $0 \le j \le k$, the same as of $G'_{pk}(y(i+k), \ldots, y(i-t))$ and with $(k+1)$-height r_{k+1}.

We denote the chaining of the rules

$$G_j(i) \xrightarrow{R_a[P_j]} G'_j(i), \qquad G'_j(i) \xrightarrow{R_b[r_{j+1}]} G_{j+1}(i), \qquad j = 0, 1, \ldots, k$$

by

$$G_0(i) \xrightarrow{R_a[P_0]} G'_0(i) \xrightarrow{R_b[r_1]} G_1(i) \xrightarrow{R_a[P_1]} \cdots \xrightarrow{R_a[P_k]} G'_k(i) \xrightarrow{R_b[r_{k+1}]} G_{k+1}(i).$$

Lemma 1 *Let* $eq_0(i)$

$$F_{00}(y(i+k),\ldots,y(i-t))+$$
$$\sum_{p=0}^{r-\tau}\sum_{j=0}^{r-p} F_{pj0}(y(i),\ldots,y(i-t))\pi(x(i-j),\ldots,x(i-j-p)) = \bar{0}$$

and $G_{p0}(y(i),\ldots,y(i-t))$ *be a* $(l_p,0)$*-echelon-*$(m\times l_p(\tau+1))$*-matrix*

$$\begin{pmatrix} F_{p00}(y(i),\ldots,y(i-t)) & \cdots & F_{p\tau 0}(y(i),\ldots,y(i-t)) \end{pmatrix}.$$

If $G_0(i) = \{G_{p0}(y(i),\ldots,y(i-t)) \mid p = 0,\ldots,r-\tau\}$ *and if*

$$G_0(i) \xrightarrow{R_a[P_0]} G'_0(i) \xrightarrow{R_b[r_1]} G_1(i) \xrightarrow{R_a[P_1]} \cdots \xrightarrow{R_a[P_{\tau-1}]} G'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} G_\tau(i),$$

for the echelon versions of rules R_a *and* R_b *then*

$$eq_0(i) \xrightarrow{R_a[P_0]} eq'_0(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[P_1]} \cdots \xrightarrow{R_a[P_{\tau-1}]} eq'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i)$$

is a transformation chain and for all p, $0 \le p \le r-\tau$, *the first* l_p *columns in matrices* $G_{p\tau}(y(i+\tau),\ldots,y(i-t))$ *create the matrix* $F_{p0\tau}(y(i+\tau),\ldots,y(i-t))$ *on the left side of the equation* $eq_\tau(i)$.

Proof: The proof of this Lemma is straight forward and can be found from [TC95]. ■

In the following there are two rules to move to the opposite direction of a transformation chain.

Rule R_a^{-1}: Let $G'_{pk}(y(i+k-1),\ldots,y(i-t)) \in MP(q)$ be an (l_p,k)-echelon-$(m\times l_p(\tau+1))$-matrix, for all $0 \le p \le r-\tau$, with the same j-heights r_j, $0 \le j \le k$, and $0 \le k \le \tau$. We assume that the rows from $r_{j-1}+1$ to r_j of matrix $G'_{pk}(y(i+k-1),\ldots,y(i-t))$ only depend on variables $y(i+j-2),\ldots,y(i+j-1-t)$ when $1 \le j \le k$ and that rows from r_k+1 to m only depend on variables $y(i+k-1),\ldots,y(i+k-t)$. Let $P'_k(y(i+k-1),\ldots,y(i+k-t)) \in MP(q)$ be an $(m\times m)$-matrix,

$$P'_k(y(i+k-1),\ldots,y(i+k-t))$$
$$= \begin{pmatrix} I_{r_k} & 0 \\ P'_{k1}(y(i+k-1),\ldots,y(i+k-t)) & P'_{k2}(y(i+k-1),\ldots,y(i+k-t)) \end{pmatrix},$$

that is nonsingular for all values of variables $y(i+k-1),\ldots,y(i+k-t)$ and where I_{r_k} is the $(r_k\times r_k)$-identity matrix. Let $G_{pk}(y(i+k-1),\ldots,y(i-t))$

be the matrix we get from $G'_{pk}(y(i+k-1),\ldots,y(i-t))$ by multiplying it from the left by $P'_k(y(i+k-1),\ldots,y(i+k-t))$, that is

$$\begin{gathered} G_{pk}(y(i+k-1),\ldots,y(i-t)) = \\ P'_k(y(i+k-1),\ldots,y(i+k-t))G'_{pk}(y(i+k-1),\ldots,y(i-t)), \end{gathered}$$

for all $0 \le p \le r-\tau$. If the columns from r_k+1 to m of $G_{pk}(y(i+k-1),\ldots,y(i-t))$ only depend on variables $y(i+k-1),\ldots,y(i+k-t)$ we say that $G_k(i)$ is *obtained from $G'_k(i)$ with rule R_a^{-1} using P'_k*, and denote

$$G'_k(i) \xrightarrow{R_a^{-1}[P'_k]} G_k(i),$$

where

$$\begin{aligned} G'_k(i) &= \{G'_{pk}(y(i+k-1),\ldots,y(i-t)) \mid p=0,1,\ldots,r-\tau\}, \\ G_k(i) &= \{G_{pk}(y(i+k-1),\ldots,y(i-t)) \mid p=0,1,\ldots,r-\tau\}. \end{aligned}$$

Remark 1. Let $G'_{pk}(y(i+k-1),\ldots,y(i-t)) \in MP(q)$ be an (l_p,k)-echelon-$(m\times l_p(\tau+1))$-matrix for all $0\le p\le r-\tau$ and $0\le k\le\tau$. If $G'_k(i) \xrightarrow{R_a^{-1}[P'_k]} G_k(i)$ then also matrix $G_{pk}(y(i+k-1),\ldots,y(i-t)) \in MP(q)$ is an (l_p,k)-echelon-$(m\times l_p(\tau+1))$-matrix and its j-height is the same as of $G'_{pk}(y(i+k-1),\ldots,y(i-t))$ when $0\le j\le k$. Also the first r_k rows of matrices $G_{pk}(y(i+k-1),\ldots,y(i-t))$ and $G'_{pk}(y(i+k-1),\ldots,y(i-t))$ are the same, rows from $r_{j-1}+1$ to r_j of $G_{pk}(y(i+k-1),\ldots,y(i-t))$ only depend on variables $y(i+j-2),\ldots,y(i+j-1-t)$, when $1\le j\le k$ and rows from r_k+1 to m only depend on variables $y(i+k-1),\ldots,y(i+k-t)$.

Rule R_b^{-1}: Let $G_{p,k+1}(y(i+k),\ldots,y(i-t)) \in MP(q)$ be an $(l_p,k+1)$-echelon-$(m\times l_p(\tau+1))$-matrix for all $0\le p\le r-\tau$, with the same j-height r_j, $0\le j\le k+1$ and $0\le k<\tau$. We assume that rows from $r_{j-1}+1$ to r_j of $G_{p,k+1}(y(i+t),\ldots,y(i-t))$ only depend on variables $y(i+j-2),\ldots,y(i+j-1-t)$, when $1\le j\le k+1$, and rows from $r_{k+1}+1$ to m only depend on variables $y(i+k),\ldots,y(i+k+1-t)$. We denote by $G'_{p,k-1}(y(i+k-1),\ldots,y(i-t))$ the matrix we get from the matrix $G_{pk}(y(i+k),\ldots,y(i-t))$ in the following way: We move the last $m-r_{k+1}$ rows l_p positions to the right and pad from the right by zeroes and also replace the variable $y(j)$ in the last $m-r_{k+1}$ rows by variable $y(j-1)$. Now we say that $G'_k(i)$ is *obtained from $G_{k+1}(i)$ by rule R_b^{-1}* and denote

$$G_{k+1}(i) \xrightarrow{R_b^{-1}[r_{k+1}]} G'_k(i),$$

where

$$\begin{aligned} G_{k+1}(i) &= \{G_{p,k+1}(y(i+k),\ldots,y(i-t)) \mid p=0,1,\ldots,r-\tau\}, \\ G'_k(i) &= \{G'_{pk}(y(i+k-1),\ldots,y(i-t)) \mid p=0,1,\ldots,r-\tau\}. \end{aligned}$$

The effect of rule R_b^{-1} is clear from picture 2 .

$$\left(\begin{array}{c|c|c}
\multicolumn{2}{c}{G_{p,k+1,1}(y(i+k),\ldots,y(i-t))} & u \\
\hline
G_{p,k+1,2}(y(i+k),\ldots,y(i-t)) & \underbrace{u}_{l_p} & u
\end{array}\right)\begin{array}{l}\}r_{k+1}\\ \end{array}$$

$$\xrightarrow{R_b^{-1}[r_{k+1}]}\left(\begin{array}{c|c|c}
\multicolumn{2}{c}{G_{p,k+1,1}(y(i+k),\ldots,y(i-t))} & u \\
\hline
\underbrace{0}_{l_p} & G_{p,k+1,2}(y(i+k-1),\ldots,y(i-t-1)) & u
\end{array}\right)\begin{array}{l}\}r_{k+1}\\ \end{array}$$

Figure 2: Rule R_b^{-1}

Remark 2. Let $G_{p,k+1}(y(i+k),\ldots,y(i-t)) \in MP(q)$ be an $(l_p,k+1)$-echelon-$(m\times l_p(\tau+1))$-matrix for all $0\le p\le r-\tau$ and $0\le k<\tau$. If $G_{k+1}(i) \xrightarrow{R_b^{-1}[r_{k+1}]} G'_k(i)$ then for all p, $0\le p\le r-\tau$, matrix $G'_{pk}(y(i+k-1),\ldots,y(i-t)) \in MP(q)$ is also an (l_p,k)-echelon-$(m\times l_p(\tau+1))$-matrix and its j-height is the same as of $G_{p,k+1}(y(i+k-1),\ldots,y(i-t))$, when $0\le j\le k$, the first r_{k+1} rows of matrices $G'_{pk}(y(i+k-1),\ldots,y(i-t))$ and $G_{p,k+1}(y(i+k),\ldots,y(i-t))$ are the same and, if $r_{k+1}<m$, the first l_p items in the last $m-r_{k+1}$ rows of $G'_{pk}(y(i+k+1),\ldots,y(i-t))$ are zeroes. Also the rows from $r_{j-1}+1$ to r_j of $G'_{pk}(y(i+k-1),\ldots,y(i-t))$ only depend on variables $y(i+j-2),\ldots,y(i+j-1-t)$ when $1\le j\le k$, and rows from r_k+1 to m only depend on variables $y(i+k-1),\ldots,y(i+k-t)$.

The chain of transformations

$$G_{j+1}(i) \xrightarrow{R_b^{-1}[r_{j+1}]} G'_j(i), \qquad G'_j(i) \xrightarrow{R_a^{-1}[P'_j]} G_j(i), \qquad j=k,k-1,\ldots,0$$

is denoted

$$G_{k+1}(i) \xrightarrow{R_b^{-1}[r_{k+1}]} G'_k(i) \xrightarrow{R_a^{-1}[P'_k]} G_k(i) \xrightarrow{R_b^{-1}[r_k]} \cdots \xrightarrow{R_b^{-1}[r_1]} G'_0(i) \xrightarrow{R_a^{-1}[P'_0]} G_0(i).$$

In the next lemma we see that these two transformation chains do correspond to each other.

Lemma 2 *Let $G_{p\tau}(y(i+\tau-1),\ldots,y(i-t)) \in MP(q)$ be an (l_p,τ)-echelon matrix $0\le p\le r-\tau$ with the same j-heights $r_1,\ldots,r_\tau$. If*

$$G_\tau(i) \xrightarrow{R_b^{-1}[r_\tau]} G'_{\tau-1}(i) \xrightarrow{R_a^{-1}[P_{\tau-1}^{-1}]} G_{\tau-1}(i) \xrightarrow{R_b^{-1}[r_{\tau-1}]} \cdots \xrightarrow{R_b^{-1}[r_1]} G'_0(i) \xrightarrow{R_a^{-1}[P_0^{-1}]} G_0(i), \tag{19}$$

then

$$G_0(i) \xrightarrow{R_a[P_0]} G'_0(i) \xrightarrow{R_b[r_1]} G_1(i) \xrightarrow{R_a[P_1]} \cdots \xrightarrow{R_a[P_{\tau-1}]} G'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} G_\tau(i), \quad (20)$$

Proof: The proof of this Lemma is straight forward and can be found from [TC95]. ■

With these two lemmata we can prove:

Theorem 7 *Let us suppose that (2) defines an automaton M and*

$$f(y(i),\dots,y(i-t),x(i),\dots,x(i-r)) = F_0(y(i),\dots,y(i-t))$$
$$+\sum_{p=0}^{r-\tau}\sum_{j=0}^{r-p} F_{pj}(y(i-1),\dots,y(i-t))\pi(x(i-j),\dots,x(i-j-p))$$

and $G_0(i) = \{G_{p0}(y(i-1),\dots,y(i-t)) \mid p = 0,1,\dots,r-\tau\}$, where

$$G_{p0}(y(i-1),\dots,y(i-t)) =$$
$$\left(F_{p0}(y(i-1),\dots,y(i-t)) \quad \cdots \quad F_{p\tau}(y(i-1),\dots,y(i-t))\right),$$

$p = 0,1,\dots,r-\tau$. Now we have:

(a) If there exists an (l_p,τ)-echelon-$(m \times l_p(\tau+1))$-matrix $G_{p\tau}(y(i+\tau-1),\dots,y(i-t))$, $0 \le p \le r-\tau$, with the same j-heights, $0 \le j \le \tau$ and such a chain of transformations

$$G_\tau(i) \xrightarrow{R_b^{-1}[r_\tau]} G'_{\tau-1}(i) \xrightarrow{R_a^{-1}[P'_{\tau-1}]} G_{\tau-1}(i) \xrightarrow{R_b^{-1}[r_{\tau-1}]} \cdots \xrightarrow{R_b^{-1}[r_1]} G'_0(i) \xrightarrow{R_a^{-1}[P'_0]} G_0(i), \quad (21)$$

that for all values of variables $x(i-1),\dots,x(i-r),y(i+\tau-1),\dots,y(i+\tau-t)$ function

$$\sum_{p=0}^{r-\tau} G_{p\tau}(y(i+\tau-1),\dots,y(i+\tau-t))\pi(x(i),\dots,x(i-p)) \quad (22)$$

of $x(i)$ is a surjection then there exists a chain of transformations

$$eq_0(i) \xrightarrow{R_a[P_0]} eq'_0(i) \xrightarrow{R_b[r_1]} eq_1(i) \xrightarrow{R_a[P_1]} \cdots \xrightarrow{R_a[P_{\tau-1}]} eq'_{\tau-1}(i) \xrightarrow{R_b[r_\tau]} eq_\tau(i), \quad (23)$$

where (5) defines an equation $eq_0(i)$ and

$$P_k(y(i+k-1),\dots,y(i+k-t)) = P'_k(y(i+k-1),\dots,y(i+k-t))^{-1}, \quad (24)$$

$$0 \le k < \tau,$$

and for all values of the variables $x(i-1),\ldots,x(i-r),y(i+\tau),\ldots,y(i-t)$ *the equation* $eq_\tau(i)$ *has a solution* $x(i)$*. Thus* M *is a weak inverse with delay* τ*.*

(b) If there exists an (l_p,τ)*-echelon-*$(m \times l_p(\tau+1))$*-matrix* $G_{p\tau}(y(i+\tau-1),\ldots,y(i-t))$*,* $0 \le p \le r-\tau$*, with same* j*-heights,* $0 \le j \le \tau$ *and there exists a transformation chain (21), such that for all values of variables* $x(i-1),\ldots,x(i-r),y(i+\tau-1),\ldots,y(i+\tau-t)$ *function (22) of* $x(i)$ *is an injection then there is a chain (23), where (5) defines an equation* $eq_0(i)$ *and (24) holds true, and for all values of variables* $x(i-1),\ldots,x(i-r),y(i+\tau),\ldots,y(i-t)$ *equation* $eq_\tau(i)$ *has at most one solution* $x(i)$*. Thus* M *is weakly invertible with delay* τ*.*

Proof: The proof of this Theorem can be found from [TC95]. ∎

Remark: The transformation φ_k used in rule R_a does not need to be necessarily linear. All results in this chapter hold true even for nonlinear transformations φ_k and φ_k^{-1} as long as the heights of the echelon matrices do not change.

4 Finite Automaton Public Key Cryptosystem

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) First construct two automata: an (h_0, k_0)-order memory finite automaton $M_0 =< X, Y, S_0, \delta_0, \lambda_0 >$ and a $(\tau_0 + k_0, h_0)$-order memory finite automaton $M_0^* =< Y, X, S_0^*, \delta_0^*, \lambda_0^* >$ satisfying the following two conditions: (a) For any state $s_0 =< y_{-1}, \ldots, y_{-k_0}, x'_{-1}, \ldots, x'_{-h_0} >$ of M_0 and any $x'_0, x'_1, \ldots$ in X, if

$$y_0 y_1 \cdots = \lambda_0(s_0, x'_0 x'_1 \cdots),$$

then

$$\lambda_0^*(s_{\tau_0}^*, y_{\tau_0} y_{\tau_0+1} \cdots) = x'_0 x'_1 \cdots ,$$

where $s_{\tau_0}^* =< x'_{-1}, \ldots, x'_{-h_0}, y_{\tau_0-1}, \ldots, y_{-k_0} >$. (b) For any state $s_0^{**} =< x'_{-1}, \ldots, x'_{-h_0}, y_{-1}, \ldots, y_{-\tau_0-k_0} >$ in

$$\{\delta_0^*(s^*, y_0 \cdots y_{\tau_0-1}) \mid s^* \in S_0^*, y_0, \ldots, y_{\tau_0-1} \in Y\}$$

and state $s_0 =< y_{-1-\tau_0}, \ldots, y_{-k_0-\tau_0}, x'_{-1}, \ldots, x'_{-h_0} >$ of M_0, for any $y_0, y_1, \ldots$ in Y, if

$$x'_0 x'_1 \cdots = \lambda_0^*(s_0^{**}, y_0 y_1 \cdots),$$

then

$$\lambda_0(s_0, x'_0 x'_1 \cdots) = y_{-\tau_0} y_{-\tau_0+1} \cdots .$$

(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 $M_1^* =< X, X, S_1^*, \delta_1^*, \lambda_1^* >$ satisfying the conditions: (a) For any state $s_1 =< x_{-1}, \ldots, x_{-h_1} >$ of M_1 and any $x_0, x_1, \ldots$ in X, if

$$x'_0 x'_1 \cdots = \lambda_1(s_1, x_0 x_1 \cdots),$$

then

$$\lambda_1^*(s_{\tau_1}^*, x'_{\tau_1} x'_{\tau_1+1} \cdots) = x_0 x_1 \cdots ,$$

where $s_{\tau_1}^* =< x_{-1}, \ldots, x_{-h_1}, x'_{\tau_1-1}, \ldots, x'_0 >$. (b) For any state $s_0^* =< x_{-1}, \ldots, x_{-h_1}, x'_{-1}, \ldots, x'_{-\tau_1} >$ of M_1^* and for any $x'_0, x'_1, \ldots$ of X, if

$$x_0 x_1 \cdots = \lambda_1^*(s_0^*, x'_0 x'_1 \cdots),$$

then

$$\lambda_1(< x_{\tau_1-1}, \ldots, x_{\tau_1-h_1} >, x_{\tau_1} x_{\tau_1+1} \cdots) = x'_0 x'_1 \cdots .$$

(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 arbitrary $y_{-1,s}, \ldots, y_{-\tau-\tau_1,s} \in Y$, and an arbitrary state $s_{-\tau}^0 =< x'_{-\tau-\tau_1-1,s}, \ldots, x'_{-\tau-\tau_1-h_0,s} y_{-\tau-\tau_1-1,s}, \ldots,$

$y_{-\tau-\tau_1-k_0-\tau_0,s}$ > of M_0^* and a state $s^1_{-\tau}$ =< $x_{-\tau-\tau_1-1,s}, \ldots, x_{-\tau-\tau_1-h_1,s}$ $x''_{-\tau-\tau_1-1,s}, \ldots, x''_{-\tau-2\tau_1,s}$ > of M_1^*. Compute the states needed in signing

$$x'_{-\tau-\tau_1,s} \cdots x'_{-1,s} = \lambda_0^*(s^0_{-\tau}, y_{-\tau-\tau_1,s} \cdots y_{-1,s})$$
$$s_{0,s} = \delta_0^*(s^0_{-\tau}, y_{-\tau-\tau_1,s} \cdots y_{-1,s})$$

and

$$x_{-\tau-\tau_1,s} \cdots x_{-1,s} = \lambda_1^*(s^1_{-\tau}, x'_{-\tau-\tau_1,s} \cdots x'_{-1,s})$$
$$s_{1,s} = \delta_1^*(s^1_{-\tau}, x'_{-\tau-\tau_1,s} \cdots x'_{-1,s}).$$

Define parts needed in verification

$$s_v^{out} =< y_{-1,s}, \ldots, y_{-k_0,s} >, \qquad s_v^{in} =< x_{-1,s}, \ldots, x_{-h_0-h_1+\tau_0+\tau_1,s} > .$$

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

$$x'_{-h_0,e} \cdots x'_{-1,e} = \lambda_1(< x_{-h_0-1,e}, \ldots, x_{-h_0-h_1,e} >, x_{-h_0,e} \cdots x_{-1,e}).$$

Define parts needed in decryption

$$s_{1,d}^{out} =< x_{-1,e}, \ldots, x_{-h_1,e} >, \qquad s_{0,d}^{out} =< x'_{-1,e}, \ldots, x'_{-h_0,e} > .$$

(5) The public key of the user A is

$$C'(M_1, M_0), \quad s_v^{out}, \quad s_v^{in}, \quad s_e, \quad \tau_0 + \tau_1.$$

The secret key of the user A is

$$M_0^*, \quad M_1^*, \quad s_{0,s}, \quad s_{1,s}, \quad s_{0,d}^{out}, \quad s_{1,d}^{out}, \quad \tau_0, \quad \tau_1.$$

Encryption. When user B wants to send to user A a plaintext $x_0 \cdots x_n$ in secret, he first adds some $\tau_0 + \tau_1$ letters $x_{n+1} \cdots x_{n+\tau_0+\tau_1}$ to the end of the plaintext. Then he can compute the ciphertext $y_0, \ldots, y_{n+\tau_0+\tau_1}$,

$$y_0 \cdots y_{n+\tau_0+\tau_1} = \lambda(s_e, x_0 \cdots x_{n+\tau_0+\tau_1}),$$

using A's public key.

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

$$x'_0 \cdots x'_{n+\tau_1} =$$
$$\lambda_0^*(< x'_{-1,e}, \ldots, x'_{-h_0,e}, y_{\tau_0-1}, \ldots, y_0, y_{-1,e}, \ldots, y_{-k_0,e} >, y_{\tau_0} \cdots y_{n+\tau_0+\tau_1}).$$

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

$$x_0 \cdots x_n = \lambda_1^*(< x_{-1,e}, \dots, x_{-h_1,e}, x'_{\tau_1-1}, \dots, x'_0 >, x'_{\tau_1} \cdots x'_{n+\tau_1}).$$

Signing. The user A wanting to sign a message $y_0 \cdots y_n$ first adds any $\tau_0 + \tau_1$ letters $y_{n+1} \cdots y_{n+\tau_0+\tau_1}$ to the end of the message. Then he computes the signature $x_0, \dots, x_{n+\tau_0+\tau_1}$,

$$x_0 \cdots x_{n+\tau_0+\tau_1} = \lambda_1^*(s_{1,s}, \lambda_0^*(s_{0,s}, y_0 \cdots y_{n+\tau_0+\tau_1})),$$

using parts M_0^*, M_1^*, $s_{0,s}$ and $s_{1,s}$ of his secret key.

Validation. Any user, say B, can verify the validity of the signature $x_0 \cdots x_{n+\tau_0+\tau_1}$ as follows. Using parts $C'(M_1, M_0)$, s_v^{out} and s_v^{in} of A's public key B computes

$$\lambda(< y_{-1,s}, \dots, y_{-k_0,s}, x_{\tau_0+\tau_1-1}, \dots, x_0, x_{-1,s}, \dots, x_{-h_0-h_1+\tau_0+\tau_1,s} >, \\ x_{\tau_0+\tau_1} \cdots x_{n+\tau_0+\tau_1}) \tag{25}$$

which should coincide with the message $y_0 \dots y_n$.

The correctness of above encryption/decryption is due to the following Theorem:

Theorem 8 *Let the automata be defined as above and* $s =< y_{-1}, \dots, y_{-k_0}, x_{-1}, \dots, x_{-h_0-h_1} >$ *be a state of* $C'(M_1, M_0)$. *Let*

$$x'_{-h_0} \cdots x'_{-1} = \lambda_1(< x_{-h_0-1}, \dots, x_{-h_0-h_1} >, x_{-h_0} \cdots x_{-1}). \tag{26}$$

For any $x_0, x_1, \dots \in X$ *and* $y_0, y_1, \dots \in Y$, *if*

$$y_0 y_1 \cdots = \lambda(s, x_0 x_1 \cdots),$$

then

$$\lambda_1^*(< x_{-1}, \dots, x_{-h_1}, x'_{\tau_1-1}, \dots, x'_0, >, x'_{\tau_1} x'_{\tau_1+1} \cdots) = x_0 x_1 \cdots ,$$

where

$$x'_0 x'_1 \cdots = \lambda_0^*(< x'_{-1}, \dots, x'_{-h_0}, y_{\tau_0-1}, \dots, y_{-k_0} >, y_{\tau_0} y_{\tau_0+1} \cdots).$$

Proof: From Theorem 1 and equation (26) we have that state s of automaton $C'(M_1, M_2)$ is equivalent to state $< s_1, s_0 >$ of $C(M_1, M_2)$, when $s_1 =< x_{-1}, \dots, x_{-h_1} >$ and $s_0 =< y_{-1}, \dots, y_{-k_0}, x'_{-1}, \dots, x'_{-h_0} >$. Denote

$$x'_0 x'_1 \cdots = \lambda_1(s_1, x_0 x_1 \cdots). \tag{27}$$

Then we have

$$y_0 y_1 \cdots = \lambda_0(s_0, x'_0 x'_1 \cdots). \tag{28}$$

From the definition of M_0 and M_0^*, (28) yields

$$\lambda_0^*(< x'_{-1}, \ldots, x'_{-h_0}, y_{\tau_0-1}, \ldots, y_{-k_0} >, y_{\tau_0} y_{\tau_0+1} \cdots) = x'_0 x'_1 \cdots ,$$

and from the definition of M_1 and M_1^*, using (27) we now have

$$\lambda_1^*(< x_{-1}, \ldots, x_{-h_1}, x'_{\tau_1-1}, \ldots, x'_0 >, x'_{\tau_1} x'_{\tau_1+1} \cdots) = x_0 x_1 \cdots .$$

■

The correctness of signing/verification could also be easily proven.

In the following automaton M_0 is in the form of

$$y(i) = \sum_{j=1}^{k_0} A_j y(i-j) + \sum_{j=0}^{h_0} B_j x(i-j), \qquad i = 0, 1, \ldots \tag{29}$$

and M_0^* in the form of

$$x(i) = \sum_{j=1}^{h_0} A_j^* x(i-j) + \sum_{j=0}^{\tau_0+k_0} B_j^* y(i-j), \qquad i = 0, 1, \ldots . \tag{30}$$

The nonlinear automaton M_1 is in the form of

$$x'(i) = \sum_{j=0}^{h_1} F_j x(i-j) + \sum_{j=0}^{h_1-\epsilon} F'_j s(x(i-j), \ldots, x(i-j-\epsilon)), \qquad i = 0, 1, \ldots \tag{31}$$

where $s(x_0, \ldots, x_\epsilon)$ 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

$$y(i) = \sum_{j=1}^{k_0} A_j y(i-j) +$$

$$\sum_{j=0}^{h_0+h_1} C_j x(i-j) + \sum_{j=0}^{h_0+h_1-\epsilon} C'_j s(x(i-j), \ldots, x(i-j-\epsilon)), \qquad i = 0, 1, \ldots ,$$

where

$$C_j = \sum_{\substack{k+n=j \\ 0 \le k \le h_0 \\ 0 \le n \le h_1}} B_k F_n \qquad \text{and} \qquad C'_j = \sum_{\substack{k+n=j \\ 0 \le k \le h_0 \\ 0 \le n \le h_1-\epsilon}} B_k F'_n .$$

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

5 FAPKC3 example

We choose $q = 2$, $l = 3$, $h_0 = 1$, $k_0 = 2$ and $\tau_0 = 1$ and start from the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & & * & \\ 0 & 0 & 1 & & * & \end{pmatrix}.$$

By choosing $r_1 = 1$, we get

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and choosing

$$P_0 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

we get by multiplying by matrix P_0^{-1}

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The automaton M_0 is now

$$\begin{aligned} y(i) = & \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} y(i-1) + \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} y(i-2) \\ & + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i) + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} x(i-1), \end{aligned}$$

When the other matrix is chosen as

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

By matrix P_0 we get

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and by height r_1

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

The automaton M_0^* is now

$$
\begin{aligned}
x(i) = {} & \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i-1) + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} y(i) \\
& + \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} y(i-1) + \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} y(i-2) + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} y(i-3).
\end{aligned}
$$

When generating the nonlinear automata we choose $h_1 = 2$, $\tau_1 = 2$, heights $r_1 = 1$ and $r_2 = 2$, coefficient matrices

$$
P_0 = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \text{ and } P_1 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}
$$

and start from the matrices

$$
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & & * & \\ 0 & 0 & 1 & & * & & & * & \end{pmatrix} \text{ and } \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & & * & \\ 0 & 0 & 0 & & * & & & * & \end{pmatrix}.
$$

From these we get the following pairs of matrices

$$
\begin{matrix}
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & & * & \\ 0 & 0 & 0 & 0 & 0 & 1 & & * & \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & & * & \\ 0 & 0 & 0 & 0 & 0 & 0 & & * & \end{pmatrix} \\
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & & * & \\ 1 & 0 & 0 & 0 & 0 & 0 & & * & \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & & * & \\ 0 & 1 & 0 & 0 & 0 & 0 & & * & \end{pmatrix} \\
\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.
\end{matrix}
$$

As the automaton M_1 we now have

$$
\begin{aligned}
y(i) = {} & \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} x(i) + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} x(i-1) + \begin{pmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} x(i-2) \\
& + \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} s(x(i), x(i-1)) + \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} s(x(i-1), xi-2)),
\end{aligned}
$$

where function s means the componentwise multiplication. Going back the transformation sequence we have the matrices

$$\begin{pmatrix}1&0&0&0&1&1&0&1&1\\0&0&0&0&1&0&1&1&1\\1&0&0&1&0&1&1&0&0\end{pmatrix}\begin{pmatrix}0&1&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0&0\\0&1&0&0&1&0&0&0&0\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&0&0&1&1&0&0\\0&0&0&0&1&0&1&1&1\\0&0&0&1&0&0&0&0&0\end{pmatrix}\begin{pmatrix}0&1&0&0&0&0&0&0&0\\0&0&0&0&0&1&0&0&0\\0&0&0&0&1&0&0&0&0\end{pmatrix}\begin{pmatrix}1&1&0\\0&1&0\\1&1&1\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&0&0&1&1&0&0\\0&1&0&1&1&1&0&0&0\\1&0&0&0&0&0&0&0&0\end{pmatrix}\qquad\begin{pmatrix}0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\end{pmatrix}$$

$$\begin{pmatrix}0&0&0&1&1&0\\0&1&0&0&0&0\\1&1&1&0&0&0\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&0&0&1&1&0&0\\0&1&0&1&1&0&1&0&0\\0&0&0&0&0&1&1&0&0\end{pmatrix}\qquad\begin{pmatrix}0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{pmatrix}$$

$$\begin{pmatrix}0&0&0&1&1&0\\1&0&1&1&1&0\\1&1&1&1&1&0\end{pmatrix}$$

$$\begin{pmatrix}1&0&0&0&0&1&1&0&0\\0&1&0&1&1&0&1&0&0\\0&0&1&1&0&0&0&0&0\end{pmatrix}\qquad\begin{pmatrix}0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\0&0&0&0&0&0&0&0&0\end{pmatrix}$$

$$\begin{pmatrix}0&0&0&0&0&0&1&1&0\\0&0&0&1&0&1&1&1&0\\1&1&1&1&1&0&0&0&0\end{pmatrix}.$$

From the last matrices we get the automaton M_1^*

$$\begin{aligned}x(i) = {} & \begin{pmatrix}0&0&1\\1&1&0\\1&0&0\end{pmatrix}x(i-1)+\begin{pmatrix}1&0&0\\1&0&0\\0&0&0\end{pmatrix}x(i-2)\\ & +\begin{pmatrix}0&1&0\\0&0&1\\0&0&0\end{pmatrix}t(x(i),x(i-1))\\ & +\begin{pmatrix}0&0&0\\0&0&0\\1&1&1\end{pmatrix}y(i)+\begin{pmatrix}0&0&0\\1&0&1\\1&1&0\end{pmatrix}y(i-1)+\begin{pmatrix}1&1&0\\1&1&0\\0&0&0\end{pmatrix}y(i-2).\end{aligned}$$

As the combined automaton $C'(M_1, M_0)$ we now have

$$
\begin{aligned}
y(i) = &\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} y(i-1) + \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} y(i-2) \\
&+ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i) + \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} x(i-1) + \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} x(i-2) \\
&+ \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} x(i-3) + \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} t(x(i), x(i-1)) \\
&+ \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} t(x(i-1), x(i-2)) + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} t(x(i-2), x(i-3)).
\end{aligned}
$$

For the starting state of encryption we choose $s_e =< 010, 100; 011, 100, 101 >$, and so

$$
M_1 : \quad \begin{array}{cc|c}
1 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 1 \\
\hline
 & & 1 \\
 & & 0 \\
 & & 0
\end{array} \quad .
$$

Now we have $s^{out}_{0,d} =< 100 >$ and $s^{out}_{1,d} =< 011, 100 >$.

When we want to encrypt the message $101, 110, 010$, we choose arbitrary letters $111, 100, 001$ and compute

$$
M : \quad \begin{array}{ccc|cccccc}
1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{array} \quad ,
$$

and the cryptotext is $011, 000, 000, 001, 100, 101$.

The decrypting is in two parts:

$$
M_0^* : \quad \begin{array}{cc|c|ccccc}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\hline
 & & 1 & 1 & 1 & 1 & 0 & 0 \\
 & & 0 & 0 & 1 & 1 & 1 & 0 \\
 & & 0 & 1 & 1 & 1 & 0 & 0
\end{array}
$$

and

$$M_1^*: \quad \begin{array}{cc|ccc} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{array} \; .$$

This system is not secure because the automaton M_1 was created using only linear transformations. More about the breaking techniques in the following chapters.

6 Breaking the key by R_a, R_b-method

Because we generated the automata M_0 and M_1 by R_a, R_b-method it is natural to try to break the key automaton $C'(M_1, M_0)$ by the transformations R_a and R_b. It is easy indeed to find an inverse for the automaton $C'(M_1, M_0)$ this way if both the automata M_0, M_1 are generated using linear rules R_a. In the following we present a way to avoid this situation.

Let the automata M_0, M_1 and $C'(M_1, M_0)$ be defined as before. We consider the matrix C_0 of the automaton $C'(M_1, M_0)$. Because $C_0 = B_0F_0$ the rows of C_0 are sums of rows of matrix F_0. Also the rank of C_0 is smaller than the number of rows l due to the way the matrix is generated.

Let us multiply the both sides of the equation defining the automaton $C'(M_1, M_0)$ by such a matrix that as many of the lowest rows as possible in both matrices P_0C_0 and P_0C_0' are zero rows. Because in our case these are (8×8)-matrices this is easily done. Even if we now fail to reach the maximum number of zero rows we can fix it in the next round. We now choose r_1^c so that the number of zero rows generated to both matrices is $l - r_1^c$. Now we have employed the rule $R_a[P_0]$ and to the result we can employ the rule $R_b[r_1^c]$. This means we move $l - r_1^c$ lowest rows l steps to the left in every matrix on both sides of the equation. The matrices C_j in automaton $C'(M_1, M_0)$ are substituted by matrices where r_1^c upper rows are from matrices P_0C_j and the rest from P_0C_{j+1}. In the same way we change the matrices C_j' and A_j. In addition we have a new term $A_{-1}y(i+1)$, where r_1^c upper rows of A_{-1} are zero rows and the rest are from P_0A_0. Like always using the rule R_b we have now an extra variable, $y(i-1)$. Luckily this is known from the cryptotext. We can use the rules back to back until after some round τ_c, $r_{\tau_c}^c = l$.

Since the automata M_1 and M_1^* are generated using only linear transformations rows in matrices F_0 and F_0' are sums of the same rows of matrices F_0^* and $F_0'^*$, respectfully. This is due to the fact that the first are generated from the latter only by multiplying these from the left by matrices and replacing rows by zero rows. Also rows in matrices C_0 and C_0' are sums of the same rows in matrices F_0^* and $F_0'^*$ because $C_0 = B_0F_0$ and $C_0' = B_0F_0'$.

Since the matrix F_0^* is nonsingular, if $l - r_1^c$ lowest rows in P_0C_0 are zero rows also lowest $l - r_1^c$ rows in P_0C_0' are zero rows. This is also true in general: if there are zero rows in the matrices generated from the matrix C_0 by the rules R_a and R_b the same rows in the matrices generated from the matrix C_0' by the same rules are zero rows. Because we used linear rules R_a the inversion is now possible.

When we have achieved the point where $r_{\tau_c}^c = l$ we have an equation we can use to solve $x(i)$. Using this information reversing the encryption is easy. If we used other than linear rules in the generation of the automaton M_1 we still probably can use this technique for some rounds before $r_j^c = l$, for some j. This would make the exhaustive search method a little faster.

Example: Let us consider the equation

$$
\begin{aligned}
\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} x(i) &= \begin{pmatrix}1&1&1\\1&0&0\\1&0&0\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\1&1&1\\1&0&1\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&1&1\\1&0&0\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} s(x(i), x(i-1))\\
&+ \begin{pmatrix}0&1&1\\0&1&0\\0&1&0\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&1&1\\0&1&0\end{pmatrix} s(x(i-2), x(i-3))\\
&+ \begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} y(i) + \begin{pmatrix}1&1&0\\0&0&1\\0&1&0\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix} y(i-2).
\end{aligned}
$$

We can apply the rule $R_b[1]$ resulting

$$
\begin{aligned}
\begin{pmatrix}1&0&0\\1&0&0\\1&0&0\end{pmatrix} x(i) &= \begin{pmatrix}1&1&1\\1&1&1\\1&0&1\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\0&1&1\\1&0&0\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix} s(x(i), x(i-1))\\
&+ \begin{pmatrix}0&1&1\\0&1&1\\0&1&0\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} s(x(i-2), x(i-3))\\
&+ \begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix} y(i+1) + \begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix} y(i)\\
&+ \begin{pmatrix}1&1&0\\1&0&0\\0&0&1\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} y(i-2).
\end{aligned}
$$

We choose the matrix P_1 to be

$$
\begin{pmatrix}1&0&0\\1&1&0\\1&0&1\end{pmatrix}
$$

and with the rule $R_a[P_1]$ we have

$$
\begin{aligned}
&\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} x(i) = \begin{pmatrix}1&1&1\\0&0&0\\0&1&0\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\1&1&1\\0&0&0\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&1&1\\0&1&1\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} s(x(i), x(i-1))\\
&+ \begin{pmatrix}0&1&1\\0&0&0\\0&0&1\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&1&1\\0&1&1\end{pmatrix} s(x(i-2), x(i-3))\\
&+ \begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix} y(i+1) + \begin{pmatrix}1&0&0\\1&0&1\\1&1&0\end{pmatrix} y(i)\\
&+ \begin{pmatrix}1&1&0\\0&1&0\\1&1&1\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\0&1&0\\0&1&0\end{pmatrix} y(i-2).
\end{aligned}
$$

We use again the rule $R_b[1]$ resulting

$$
\begin{aligned}
&\begin{pmatrix}1&0&0\\0&0&0\\0&1&0\end{pmatrix} x(i) = \begin{pmatrix}1&1&1\\1&1&1\\0&0&0\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\0&1&1\\0&1&1\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&1\end{pmatrix} s(x(i), x(i-1))\\
&+ \begin{pmatrix}0&1&1\\0&1&1\\0&1&1\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} s(x(i-2), x(i-3))\\
&+ \begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix} y(i+2) + \begin{pmatrix}0&0&0\\1&0&1\\1&1&0\end{pmatrix} y(i+1) + \begin{pmatrix}1&0&0\\0&1&0\\1&1&1\end{pmatrix} y(i)\\
&+ \begin{pmatrix}1&1&0\\0&1&0\\0&1&0\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} y(i-2).
\end{aligned}
$$

We apply the rule $R_a[P_2]$, where

$$
P_2 = \begin{pmatrix}1&0&0\\0&0&1\\0&1&0\end{pmatrix},
$$

and get

$$
\begin{aligned}
\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix} x(i) &= \begin{pmatrix}1&1&1\\0&0&0\\1&1&1\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\0&1&1\\0&1&1\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&0&1\\0&0&0\end{pmatrix} s(x(i), x(i-1))\\
&+ \begin{pmatrix}0&1&1\\0&1&1\\0&1&1\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} s(x(i-2), x(i-3))\\
&+ \begin{pmatrix}0&0&0\\0&0&1\\0&1&0\end{pmatrix} y(i+2) + \begin{pmatrix}0&0&0\\1&1&0\\1&0&1\end{pmatrix} y(i+1) + \begin{pmatrix}1&0&0\\1&1&1\\0&1&0\end{pmatrix} y(i)\\
&+ \begin{pmatrix}1&1&0\\0&1&0\\0&1&0\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} y(i-2).
\end{aligned}
$$

Now we use the rule $R_b[2]$ resulting

$$
\begin{aligned}
\begin{pmatrix}1&0&0\\0&1&0\\1&1&1\end{pmatrix} x(i) &= \begin{pmatrix}1&1&1\\0&0&0\\0&1&1\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\0&1&1\\0&0&0\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&0&1\\0&1&1\end{pmatrix} s(x(i), x(i-1))\\
&+ \begin{pmatrix}0&1&1\\0&1&1\\0&0&0\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&0&0\\0&0&0\end{pmatrix} s(x(i-2), x(i-3))\\
&+ \begin{pmatrix}0&0&0\\0&0&0\\0&1&0\end{pmatrix} y(i+3) + \begin{pmatrix}0&0&0\\0&0&1\\1&0&1\end{pmatrix} y(i+2) + \begin{pmatrix}0&0&0\\1&1&0\\0&1&0\end{pmatrix} y(i+1)\\
&+ \begin{pmatrix}1&0&0\\1&1&1\\0&1&0\end{pmatrix} y(i) + \begin{pmatrix}1&1&0\\0&1&0\\0&0&0\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} y(i-2).
\end{aligned}
$$

We can make this look more simple by choosing

$$
P_3 = \begin{pmatrix}1&0&0\\0&1&0\\1&1&1\end{pmatrix},
$$

and using the rule $R_a[P_3]$:

$$\begin{pmatrix}1&0&0\\0&1&0\\0&0&1\end{pmatrix} x(i) = \begin{pmatrix}1&1&1\\0&0&0\\1&0&0\end{pmatrix} x(i-1) + \begin{pmatrix}1&0&0\\0&1&1\\1&1&1\end{pmatrix} x(i-2)$$
$$+ \begin{pmatrix}0&1&1\\0&0&0\\0&1&1\end{pmatrix} x(i-3) + \begin{pmatrix}0&1&0\\0&0&1\\0&0&0\end{pmatrix} s(x(i), x(i-1))$$
$$+ \begin{pmatrix}0&1&1\\0&1&1\\0&0&0\end{pmatrix} s(x(i-1), x(i-2)) + \begin{pmatrix}0&1&1\\0&0&0\\0&1&1\end{pmatrix} s(x(i-2), x(i-3))$$
$$+ \begin{pmatrix}0&0&0\\0&0&0\\0&1&0\end{pmatrix} y(i+3) + \begin{pmatrix}0&0&0\\0&0&1\\1&0&0\end{pmatrix} y(i+2) + \begin{pmatrix}0&0&0\\1&1&0\\1&0&0\end{pmatrix} y(i+1)$$
$$+ \begin{pmatrix}1&0&0\\1&1&1\\0&0&1\end{pmatrix} y(i) + \begin{pmatrix}1&1&0\\0&1&0\\1&0&0\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\0&0&0\\0&1&0\end{pmatrix} y(i-2).$$

If the function s is for example componentwise multiplication the decrypting is easy because we can solve $x(i)$ easily.

If we want to generate the automata by R_a, R_b-method we must use the nonlinear transformations. In the article [TC95] there are no details how to do this.

In the article [BI] Feng Bao and Yoshihide Igarashi suggest that the nonlinear automaton M_1 could be the automaton defined by

$$y(i) = \sum_{j=0}^{\tau} F_j x(i-j) + \sum_{j=1}^{\tau-1} F'_j s(x(i-\tau-j), \ldots, x(i-\tau-j-\epsilon))$$
$$i = 0, 1, \ldots.$$

Matrices F_j are the coefficient matrices of some linear automaton and matrices F'_j are arbitrary. Also the combined automaton $C'(M_1, M_0)$ we get using these automaton can be inverted using the method above.

Let the function s now be the componentwise multiplication. We consider the automaton M_1 defined by equation

$$y(i) = \sum_{j=0}^{h_1} F_j x(i-j) + \sum_{j=0}^{h_1-\epsilon} F'_j s(x(i-j), \ldots, x(i-j-\epsilon)),$$

where

$$F_j = \begin{pmatrix} a^j_{00} & & & & & & \\ & \ddots & & & & a^j_{km} & \\ & & a^j_{jj} & & & & \\ & & & 1 & & & \\ & & & & 0 & & \\ & 0 & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}, F'_j = \begin{pmatrix} {a'}^j_{00} & & & & & \\ & \ddots & & & {a'}^j_{km} & \\ & & {a'}^j_{jj} & & & \\ & & & 0 & & \\ & 0 & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

for all $j = 0, \ldots, l-1$ and

$$F_j = \begin{pmatrix} a^j_{00} & & a^j_{km} \\ & \ddots & \\ 0 & & a^j_{l-1,l-1} \end{pmatrix}, \qquad F'_j = \begin{pmatrix} {a'}^j_{00} & & {a'}^j_{km} \\ & \ddots & \\ 0 & & {a'}^j_{l-1,l-1} \end{pmatrix}$$

for all $j = l, \ldots, h_1$.

Example:

$$F_0 = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad F'_0 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$F_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad F'_1 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$F_2 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad F'_2 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$F_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad F'_3 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$F_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad F'_4 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

We can easily show that this automaton is invertible. We start from the last row. The last row of the equation defining the automaton M_1 defines the last components of the inputs. After that the second to last row defines the second to last components of the inputs. At the end we know the whole input.

This automaton M is clearly weakly invertible with delay at most $\frac{(l-1)l}{2}$. We can easily construct an automaton M_1^* that is a weak inverse with delay $\frac{(l-1)l}{2}$ of M. If we need to have a longer delay we can add between any two matrices a new matrix of the same type than the previous matrix.

These automata are an example of automaton that make the R_a, R_b-method inefficient against the $C'(M_1, M_0)$ matrix. These automata are not generated by R_a, R_b-method and thus the zero rows in F_j and F_j' are not dependable of each other. The R_a, R_b-method always makes the exhaustive search method a little faster and therefore it might be wise to choose the parameter τ_1 bigger than originally recommended.

7 Exhaustive search from the end to the beginning

In the way it is presented FAPKC3 is vulnerable against an attack where the attacker knows a short part from the end of the plaintext. In the following we present a version of FAPKC3 where this is not possible.

Let the automata M_0, M_1 and $C'(M_1, M_0)$ be as defined in FAPKC3.

If the attacker knows or succeeds in guessing $h_0 + h_1$ inputs $x(n - h_0 - h_1 + 1), \ldots, x(n)$ he knows a state

$$s(n+1) =< y(n), \ldots, y(n-k_0+1), x(n), \ldots, x(n-h_0-h_1+1) >$$

of $C'(M_1, M_0)$. Now he can try to find a state $s(n)$ from which, with an input $x(n)$, an automaton $C'(M_1, M_0)$ outputs $y(n)$ and moves to the state $s(n+1)$. The only unknown in the state

$$s(n) =< y(n-1), \ldots, y(n-k_0), x(n-1), \ldots, x(n-h_0-h_1) >$$

is the input $x(n - h_0 - h_1)$ and it solves the equation

$$\begin{aligned} y(n) = &\sum_{j=1}^{k_0} A_j y(n-j) + \sum_{j=0}^{h_0+h_1} C_j x(n-j) \\ &+ \sum_{j=0}^{h_0+h_1-\epsilon} C'_j s(x(n-j), \ldots, x(n-j-\epsilon)) \end{aligned} \tag{32}$$

Using the public key and the cryptotext we can generate such a system of equations that $x(n - h_0 - h_1)$ has a unique solution.

When we generated the key automata we did not consider how hard it is to do the exhaustive search backwards. So the complexity of this depends only from our luck. Usually the complexity of the exhaustive search is much lower backwards than forwards that is designed to be hard. In the worst case the solution of $x(n - h_0 - h_1)$ is unique even in equation (32).

We can make things harder for the attacker by changing the equation

$$y(i) = \sum_{j=1}^{k_0} A_j y(i-j) + \sum_{j=0}^{h_0} B_j x(i-j) + \sum_{j=h_0+1}^{2h_0} B_{2h_0-j} x(i-j)$$

defining the automaton M_0 to equation

$$\begin{aligned} y(i) = &\sum_{j=0}^{h_1} F_j x(i-j) + \sum_{j=h_1+1}^{2h_1} F_{2h_1-j} x(i-j) \\ &+ \sum_{j=0}^{h_1-\epsilon} F'_j s(x(i-j), \ldots, x(i-j-\epsilon)) \\ &+ \sum_{j=h_1}^{2h_1-\epsilon} F'_{2h_1-j} s(x(i-j), \ldots, x(i-j-\epsilon)). \end{aligned}$$

We get the new automata M_0^* and M_1^* using the same transformations as before.

The equation defining the automaton $C'(M_1, M_0)$ is now

$$\begin{aligned} y(i) = &\sum_{j=1}^{k_0} A_j y(i-j) + \sum_{j=0}^{2(h_0+h_1)} C_j x(i-j) \\ &+ \sum_{j=0}^{2(h_0+h_1)-\epsilon} C'_j s(x(i-j), \dots, x(i-j-\epsilon)) \end{aligned}$$

where

$$C_j = \sum_{\substack{k+n=j \\ 0 \le k \le 2h_0 \\ 0 \le n \le 2h_1}} B_k F_n \quad \text{and} \quad C'_j = \sum_{\substack{k+n=j \\ 0 \le k \le 2h_0 \\ 0 \le n \le 2h_1-\epsilon}} B_k F'_n.$$

We write

$$B_j = B_{2h_0-j}, \quad F_j = F_{2h_1-j} \quad \text{and} \quad F'_j = F'_{2h_1-\epsilon-j},$$

where $h_1 \le j \le 2h_1 - \epsilon$ and $F'_j = 0$, where $h_1 - \epsilon < j < h_1$.

Using this equation it is as hard to do the exhaustive search backwards as it is to forwards, provided the function s is symmetric enough. The size of the keys only grows by the difference in the sizes of states because

$$C_j = C_{h_0+h_1-j} \quad \text{and} \quad C'_j = C_{h_0+h_1-\epsilon-j}$$

for all $j = 1, \dots, h_0 + h_1$.

Example: We can make our previous example more resistant in the following way: for the automaton M_0 we get

$$\begin{aligned} y(i) = &\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} y(i-1) + \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} y(i-2) \\ &+ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i) + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} x(i-1) + \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i-2), \end{aligned}$$

and for the automaton M_1 we get

$$
\begin{aligned}
y(i) = &\begin{pmatrix}1&0&0\\0&0&0\\1&0&0\end{pmatrix} x(i) + \begin{pmatrix}0&1&1\\0&1&0\\1&0&1\end{pmatrix} x(i-1) + \begin{pmatrix}0&1&1\\1&1&1\\1&0&0\end{pmatrix} x(i-2)\\
&+ \begin{pmatrix}0&1&1\\0&1&0\\1&0&1\end{pmatrix} x(i-3) + \begin{pmatrix}1&0&0\\0&0&0\\1&0&0\end{pmatrix} x(i-4)\\
&+ \begin{pmatrix}0&1&0\\0&0&0\\0&1&0\end{pmatrix} s(x(i), x(i-1)) + \begin{pmatrix}0&0&1\\0&0&1\\0&1&0\end{pmatrix} s(x(i-1), xi-2))\\
&+ \begin{pmatrix}0&0&1\\0&0&1\\0&1&0\end{pmatrix} s(x(i-2), xi-3)) + \begin{pmatrix}0&1&0\\0&0&0\\0&1&0\end{pmatrix} s(x(i-3), x(i-4)).
\end{aligned}
$$

From these we get the automaton $C'(M_1, M_0)$

$$
\begin{aligned}
y(i) = &\begin{pmatrix}1&1&0\\0&0&1\\0&1&0\end{pmatrix} y(i-1) + \begin{pmatrix}0&1&0\\1&0&0\\0&0&1\end{pmatrix} y(i-2) + \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} x(i)\\
&+ \begin{pmatrix}1&1&1\\1&0&0\\1&0&0\end{pmatrix} x(i-1) + \begin{pmatrix}0&0&0\\1&1&1\\1&0&1\end{pmatrix} x(i-2) + \begin{pmatrix}0&1&1\\0&1&1\\1&0&0\end{pmatrix} x(i-3)\\
&+ \begin{pmatrix}0&0&0\\1&1&1\\1&0&1\end{pmatrix} x(i-4) + \begin{pmatrix}1&1&1\\1&0&0\\1&0&0\end{pmatrix} x(i-5) + \begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix} x(i-6)\\
&+ \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} s(x(i), x(i-1)) + \begin{pmatrix}0&1&1\\0&1&0\\0&1&0\end{pmatrix} s(x(i-1), x(i-2))\\
&+ \begin{pmatrix}0&0&0\\0&1&1\\0&1&0\end{pmatrix} s(x(i-2), x(i-3)) + \begin{pmatrix}0&0&0\\0&1&1\\0&1&0\end{pmatrix} s(x(i-3), x(i-4))\\
&+ \begin{pmatrix}0&1&1\\0&1&0\\0&1&0\end{pmatrix} s(x(i-4), x(i-5)) + \begin{pmatrix}0&1&0\\0&0&0\\0&0&0\end{pmatrix} s(x(i-5), x(i-6)).
\end{aligned}
$$

For example for the private key automaton M_0^* we get by matrices

$$
\begin{pmatrix}1&0&0&0&1&0&1&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&0&0&1&0&0&0\end{pmatrix}
\begin{pmatrix}1&0&1&1&0&0&0&1&1\\0&1&1&0&1&1&1&0&1\\0&0&1&0&1&0&0&0&1\end{pmatrix}
$$

$$
\begin{pmatrix}1&0&0&0&1&0&1&0&0\\0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\end{pmatrix}
\begin{pmatrix}0&0&0&1&0&1&1&0&0&0&1&1\\0&1&1&0&1&1&1&0&1&0&0&0\\0&0&1&0&1&0&0&0&1&0&0&0\end{pmatrix}.
$$

the automaton

$$x(i) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i-1) \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x(i-1) + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} y(i)$$
$$+ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} y(i-1) + \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} y(i-2) + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} y(i-3).$$

References

[BI] Feng Bao and Yoshihide Igarashi, Break Finite Automata Public Key Cryptosystem. ICALP 1995, sivut 147-158.

[Mes] Tommi Meskanen, Äärellisiin automaatteihin perustuvista julkisen avaimen kryptosysteemeistä. Licentiate Thesis, University of Turku, April 2000 (in Finnish).

[TC95] Renji Tao and Shihua Chen, Generating a kind of nonlinear finite automata with invertibility by transformation method. Technical Report No. ISCAS-LCS-95-05, Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, June 1995.

[TCC] Renji Tao, Shihua Chen and Xuemei Chen, FAPKC3: a new automaton public key cryptosystem. Technical Report No. ISCAS-LCS-95-07, Laboratory for Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, June 1995.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**