A Note on Automata-based Dynamic Convolutional Cryptosystems

Renji Tao
Institute of Software, Chinese Academy of Sciences,
Beijing 100080, China
E-mail: trj@ios.ac.cn

Abstract

In [1], the automata-based dynamic convolutional cryptosystem is proposed and analyzed; the author claims that "finding partial information about the cipher is quite easy, and the main idea of such an attack, described in detail in Section 4.1, is based on Gaussian elimination." But the deduction supporting this claim in Section 4.1 of [1] cannot work. It seems that this cipher is not so weak so far.

1 Definition of ADCC

Recall some definitions. [1] A *finite automaton,* say M, is a quintuple $\langle X, Y, S, \delta, \lambda\rangle$, where X is a nonempty finite set (the *input alphabet* of M), Y a nonempty finite set (the *output alphabet* of M), S a nonempty finite set (the *state alphabet* of M), $\delta : S \times X \to S$ a single-valued mapping (the *next state function* of M), and $\lambda : S \times X \to Y$ a single-valued mapping (the *output function* of M).

For any set A, we use A^* to denote the set of all words (finite sequences) over A including the *empty word* ε. Expand the domains of δ and λ to $S \times X^*$ as follows.

$$\begin{aligned} &\delta(s,\varepsilon) = s, \qquad && \delta(s,\alpha x) = \delta(\delta(s,\alpha), x),\\ &\lambda(s,\varepsilon) = \varepsilon, \qquad && \lambda(s,\alpha x) = \lambda(s,\alpha)\lambda(\delta(s,\alpha), x),\\ & && s \in S,\ x \in X,\ \alpha \in X^*. \end{aligned}$$

In [1], Trincă proposed a symmetric cryptosystem, named *automata-based dynamic convolutional cryptosystem* (ADCC for short) with q states. The encoder of an ADCC is a finite automaton, say $M = \langle X, Y, S, \delta, \lambda\rangle$, where X and Y are the k-dimensional row vector space over $GF(2)$ (the field with 2 elements), $S = X^m \times \{1, \ldots, q\} = \{\langle x_{-1}, \ldots, x_{-m}, w\rangle \mid x_i \in X, i = -1, \ldots, -m, w = 1, \ldots, q\}$, q and m being two positive integers,

$$\begin{aligned} \delta(\langle x_{-1}, \ldots, x_{-m}, w\rangle, x_0) &= \langle x_0, \ldots, x_{-m+1}, f(w, x_0)\rangle,\\ \lambda(\langle x_{-1}, \ldots, x_{-m}, w\rangle, x_0) &= \sum_{j=0}^{m} x_{-j} G_{j,w},\\ x_0, x_{-1}, \ldots, x_{-m} &\in X, w \in \{1, \ldots, q\}, \end{aligned}$$

f being a single-valued mapping from $\{1, \ldots, q\} \times X$ to $\{1, \ldots, q\}$, for each $w \in \{1, \ldots, q\}$, $G_{0,w}$ being a $k \times k$ nonsingular matrix over $GF(2)$, and $G_{j,w}$ being a $k \times k$ matrix over $GF(2)$, $j = 1, \ldots, m$.

[1] See [2] for example.

For any initial state $s_0 = \langle x_{-1}, \ldots, x_{-m}, w_0 \rangle$ in S and any input sequence $\alpha = x_0 \ldots x_{p-1}$ over X, the output of the encoder is $\beta = y_0 \ldots y_{p-1}$, where $\beta = \lambda(s_0, \alpha)$, that is,[2]

$$\begin{aligned}
& y_i = \lambda(s_i, x_i) = \lambda(\langle x_{i-1}, \ldots, x_{i-m}, w_i \rangle, x_i) = \sum_{j=0}^{m} x_{i-j} G_{j,w_i}, \\
& s_{i+1} = \delta(s_i, x_i) = \delta(\langle x_{i-1}, \ldots, x_{i-m}, w_i \rangle, x_i) = \langle x_i, \ldots, x_{i-m+1}, w_{i+1} \rangle, \\
& w_{i+1} = f(w_i, x_i), \\
& \quad i = 0, 1, \ldots, p-1.
\end{aligned} \tag{1}$$

By notation in automata theory the encoder mentioned above is a weakly invertible finite automaton with delay 0. Thus the decoder can be implemented by a finite automaton. It is easy to construct an decoder, say the finite automaton $M' = \langle Y, X, S, \delta', \lambda' \rangle$, where

$$\begin{aligned}
& \delta'(\langle x_{-1}, \ldots, x_{-m}, w \rangle, y_0) = \langle x_0, \ldots, x_{-m+1}, f(w, x_0) \rangle, \\
& \lambda'(\langle x_{-1}, \ldots, x_{-m}, w \rangle, y_0) = x_0, \\
& x_0 = y_0 G_{0,w}^{-1} - \sum_{j=1}^{m} x_{-j} G_{j,w} G_{0,w}^{-1}, \\
& \quad x_{-1}, \ldots, x_{-m} \in X, w \in \{1, \ldots, q\}, y_0 \in Y.
\end{aligned}$$

For any initial state $s_0 = \langle x_{-1}, \ldots, x_{-m}, w_0 \rangle$ in S and any input sequence $\beta = y_0 \ldots y_{p-1}$ over Y, the output of the decoder is $\lambda'(s_0, \beta) = x_0 \ldots x_{p-1}$, where

$$\begin{aligned}
& x_i = \lambda'(s_i, y_i) = \lambda'(\langle x_{i-1}, \ldots, x_{i-m}, w_i \rangle, y_i) = y_i G_{0,w_i}^{-1} - \sum_{j=1}^{m} x_{i-j} G_{j,w_i} G_{0,w_i}^{-1}, \\
& s_{i+1} = \delta'(s_i, y_i) = \delta'(\langle x_{i-1}, \ldots, x_{i-m}, w_i \rangle, y_i) = \langle x_i, \ldots, x_{i-m+1}, w_{i+1} \rangle, \\
& w_{i+1} = f(w_i, x_i), \\
& \quad i = 0, 1, \ldots, p-1.
\end{aligned} \tag{2}$$

From (1) and (2), it is easy to verify that for any state $s \in S$ and any sequence α over X we have

$$\lambda'(s, \lambda(s, \alpha)) = \alpha.$$

This means that M' is an decoder of the encoder M indeed.

The key of the ADCC, according to [1], consists of f and $G_{j,w}$, $j = 0, 1, \ldots, m$, $w = 1, \ldots, q$.[3]

2 ADCC is not so weak

We have restated Trincă's definition of ADCC with automata-theoretic and linear algebraic notation in the preceding section. In [1, p.8], the author wrote: "An automata-based dynamic convolutional cryptosystem is considered broken as soon as the attacker finds the bits of the matrices[4] $G_{t,0}^1$, …, $G_{t,m}^1$, …, $G_{t,0}^q$, …, $G_{t,m}^q$, and the values of the function f, i.e. $f(i, u)$ for all (i, u). Breaking the cipher completely is a very difficult task, as we will see in Section 4.2. However, finding partial information about the cipher is quite easy, and the main idea of such an attack, described in detail in Section 4.1, is based on Gaussian elimination." In [1,

[2] In [1], the initial state $s_0 = \langle x_{-1}, \ldots, x_{-m}, w_0 \rangle$ always takes the value $\langle 0, \ldots, 0, 1 \rangle$. It seems not necessary.

[3] The key may also include the initial state $\langle x_{-1}, \ldots, x_{-m}, w_0 \rangle$.

[4] $G_{t,j}^w$ is rewritten as $G_{j,w}$ in this paper.

p.10], the author wrote: "We have seen in Section 4.1 that a (256, 256, 32) automata-based dynamic convolutional cryptosystem can be partially broken quite easily." But the deduction supporting this claim in Section 4.1 of [1] cannot work. This observation is implicit in the following discussion in this section.

Similar to the discussion in in Section 4.1 of [1], consider a known-plaintext attack. We still confine to the case where $x_{-1} = \ldots = x_{-m} = 0$ and $w_0 = 1$.

Assume that an adversary had gotten n plaintext-ciphertext pairs of length p, say $(\alpha_i, \beta_i) = (x_{0,i}x_{1,i}\ldots x_{p-1,i}, y_{0,i}y_{1,i}\ldots y_{p-1,i})$, $i = 1, 2, \ldots, n$, satisfying the following conditions:

$$\beta_i = \lambda(\langle 0, \ldots, 0, 1\rangle, \alpha_i),\ \ i = 1, 2, \ldots, n, \tag{3}$$

and[5]

$$\exists w_0 \ldots \exists w_{p-1}[w_0 = 1 \& \forall i \forall j (1 \le i \le n \& 0 \le j < p-1 \to f(w_j, x_{j,i}) = w_{j+1})]. \tag{4}$$

Denote $x_{j,i} = 0$, for $j < 0, 1 \le i \le n$. For any $j, 0 \le j \le p-1$, from (3) and (4), we have

$$A_j \begin{bmatrix} G_{m,w_j} \\ G_{m-1,w_j} \\ \vdots \\ G_{0,w_j} \end{bmatrix} = \begin{bmatrix} y_{j,1} \\ y_{j,2} \\ \vdots \\ y_{j,n} \end{bmatrix}, \tag{5}$$

where

$$A_j = \begin{bmatrix} x_{j-m,1} & x_{j-m+1,1} & \cdots & x_{j,1} \\ x_{j-m,2} & x_{j-m+1,2} & \cdots & x_{j,2} \\ \vdots & \vdots & \vdots & \vdots \\ x_{j-m,n} & x_{j-m+1,n} & \cdots & x_{j,n} \end{bmatrix}. \tag{6}$$

We discuss the number of the solutions $G_{h,w_j}, h = 0, 1, \ldots, m$ of (5). Let

$$A_j = [X_{j-m}, X_{j-m+1}, \ldots, X_j],$$

where X_h has k columns, $h = j-m, j-m+1, \ldots, j$. Let r_j be the rank of A_j, and r'_h the rank of X_h. Clearly, $r'_h \le k$ and $r_j \le \sum_{h=0}^{m} r'_{j-h}$.

Consider the case of $m \le j < p-1$. Since (4) holds, we have

$$f(w_h, x_{h,i}) = w_{h+1},\ \ i = 1, \ldots, n,\ \ h = j-m, j-m+1, \ldots, j. \tag{7}$$

For any $w', w'' \in \{1, \ldots, q-1\}$, define a set

$$X_{w',w''} = \{x \in X \mid f(w', x) = w''\}.$$

We use $r_{w',w''}$ to denote the dimension of the vector space generated by $X_{w',w''}$. Then we have

$$r_j \le \sum_{h=0}^{m} r'_{j-h} \le \sum_{h=0}^{m} r_{w_{j-h}, w_{j-h+1}},\ \ j = m, m+1, \ldots, p-2.$$

It follows that the number of the solutions $G_{h,w_j}, h = 0, 1, \ldots, m$ of the system of equations (5) and (7) is at least[6]

$$2^{k[(m+1)k - \sum_{h=0}^{m} r_{w_{j-h}, w_{j-h+1}}]},\ \ j = m, m+1, \ldots, p-2. \tag{8}$$

[5]The condition (4) has not been considered in [1] !

[6]The condition that $G_{0,w}$ is nonsingular is neglected here.

This yields that if the solution $G_{h,w_j}, h = 0,1,\ldots,m$ is unique, then $r_{w_{j-h},w_{j-h+1}} = k$, for $h = 0,1,\ldots,m$.

For the case of $j = p-1$, similar to (8), noticing $r'_{p-1} \le k$, the number of the solutions $G_{h,w_j}, h = 0,1,\ldots,m$ of the system of equations (5) and (7) is at least

$$2^{k[mk-\sum_{h=1}^{m} r_{w_{p-h-1},w_{p-h}}]}. \tag{9}$$

For the case of $0 \le j < m$ and $x_{-1} = \ldots = x_{-m} = 0$, letting $r_{h,h+1} = 0$ for $h < 0$, then (8) gives a lower bound also for $j = 0,1,\ldots,m-1$.

As shown in (8) and (9), the lower bounds heavily depends on $r_{w',w''}$. We turn back to discuss $X_{w',w''}$.

For any $w \in \{1,\ldots,q\}$, f is called *almost uniform* with respect to w, if for any $w' \in f(w,X)$, $\lfloor 2^k/q \rfloor \le |\{x \in X \mid f(w,x) = w'\}| \le \lceil 2^k/q \rceil$ holds.[7] f is called *almost uniform*, if for any $w \in \{1,\ldots,q\}$, f is almost uniform with respect to w.

Suppose that f is almost uniform. It is easy to see that $|X_{w,w'}| \le \lceil 2^k/q \rceil$ holds for any $w, w' \in \{1,\ldots,q\}$. It follows that $r_{w,w'} \le \lceil 2^k/q \rceil$ holds for any $w, w' \in \{1,\ldots,q\}$. From (8) and (9), the number of the solutions $G_{h,w_j}, h = 0,1,\ldots,m$ of the system of equations (5) and (7) is at least

$$2^{k(m+1)(k-\lceil 2^k/q \rceil)}, \quad \text{for } j = 0,1,\ldots,p-2, \tag{10}$$

and

$$2^{km(k-\lceil 2^k/q \rceil)}, \quad \text{for } j = p-1. \tag{11}$$

In case of $q \ge 2^k$, we have $\lceil 2^k/q \rceil = 1$. Whenever f in the key is almost uniform, from (10) and (11), the number of the solutions $G_{h,w_j}, h = 0,1,\ldots,m$ of the system of equations (5) and (7) is at least

$$2^{k(m+1)(k-1)}, \quad \text{for } j = 0,1,\ldots,p-2,$$

and

$$2^{km(k-1)}, \quad \text{for } j = p-1.$$

In case of $2^{k-d+1} > q \ge 2^{k-d}$, $0 \le d \le k$, we have $\lceil 2^k/q \rceil \le 2^d$. Whenever f in the key is almost uniform, from (10) and (11), the number of the solutions $G_{h,w_j}, h = 0,1,\ldots,m$ of the system of equations (5) and (7) is at least

$$2^{k(m+1)(k-2^d)}, \quad \text{for } j = 0,1,\ldots,p-2,$$

and

$$2^{km(k-2^d)}, \quad \text{for } j = p-1.$$

For example, for $d = 1$, the lower bounds are $2^{(m+1)k(k-2)}$ and $2^{mk(k-2)}$, respectively.

For any small q with $k \le \lceil 2^k/q \rceil$, the lower bounds given by (10) and (11) are trivial.

In an extreme case $q = 1$, $G_{j,w}$ is independent of w. The adversary can learn $G_{h,1}, h = 0,1,\ldots,m$ from $n = (m+1)k$ plaintext-ciphertext pairs of length $p = m+1$, say $(x_{0,i}x_{1,i}\ldots x_{m,i}, y_{0,i}y_{1,i}\ldots y_{m,i})$, $i = 1,2,\ldots,(m+1)k$, so that the $(m+1)k \times (m+1)k$ matrix A_m defined by (6) is nonsingular. This observation is trivial, since the encoder degenerates to a linear finite automaton. In general, contrary to error correcting codes, the linear encoder for a cipher is avoided.

[7] $\lceil a \rceil$ is the least integer $\ge a$, and $\lfloor a \rfloor$ is the greatest integer $\le a$.

In case of $q > 1$, since f in the key is unknown, the probability of $\forall i(1 \le i \le n \& f(w_j, x_{j,i}) = f(w_j, x_{j,1}))$ may be assumed to be $q^{-(n-1)}$. Thus the probability of that (4) holds may be regarded as $q^{-(n-1)(p-1)}$; it equals $q^{-((m+1)k-1)m}$ if $n = (m+1)k$ and $p = m+1$. We have $q^{-((m+1)k-1)m} = 10^{-(33 \cdot 256-1)32} = 10^{-270304}$, if $k = 256$, $m = 32$ and $q = 10$ (parameters in [1]), and $q^{-((m+1)k-1)m} = 2^{-(33 \cdot 8-1)32} = 2^{-8416} < 10^{-2533}$, if $k = 8$, $m = 32$ and $q = 2$. It seems that for moderate n and p, the probability of that (4) holds is too small.

3 Key space of ADCC

In the definition of ADCC with q states, it is not necessary that for any $1 \le i < w \le q$, $[G_{0,i}, \ldots, G_{m,i}] \neq [G_{0,w}, \ldots, G_{m,w}]$ holds. This restriction will reduce the size of the key space of ADCC. Since the number of all $k \times k$ nonsingular matrices over $GF(2)$ is $\prod_{h=0}^{k-1}(2^k - 2^h)$ and the number of all $k \times k$ matrices over $GF(2)$ is 2^{k^2}, the number of all possible values of $[G_{0,w}, G_{1,w}, \ldots, G_{m,w}]$ is $2^{k^2 m} \prod_{h=0}^{k-1}(2^k - 2^h)$, denoted by q_m. Notice that $\log_2 q_m = k^2 m + \sum_{h=0}^{k-1} \log_2(2^k - 2^h) = k^2 m + \sum_{h=0}^{k-1} \log_2(2^h(2^{k-h} - 1)) = k^2 m + k(k-1)/2 + \sum_{h=0}^{k-1} \log_2(2^{k-h} - 1) = k^2 m + k(k-1)/2 + \sum_{h=1}^{k} \log_2(2^h - 1) \ge k^2 m + k(k-1)/2 + \sum_{h=2}^{k} \log_2(2^{h-1}) = k^2 m + k(k-1) = k^2(m+1) - k$.

The key of ADCC with q states is the structure of the finite automaton M. Clearly, the number of all single-valued mapping from $\{1, \ldots, q\} \times X$ to $\{1, \ldots, q\}$ is $q^{2^k q}$ and the number of all possible choices of $[G_{0,w}, \ldots, G_{m,w}]$, $w = 1, \ldots, q$ is $(q_m)^q$.[8] Thus the size of the key space of ADCC with q states is $q^{2^k q}(q_m)^q$, where $q_m = 2^{k^2 m} \prod_{h=0}^{k-1}(2^k - 2^h) \ge 2^{k^2(m+1)-k}$.

The cipher ADCC may be slightly expanded. For any positive integers k and m, we use $\mathcal{G}_{k,m}$ to denote the set of all $(m+1)k \times k$ matrices over $GF(2)$ of which the submatrices of the last k rows are nonsingular. As shown above, the number of elements in $\mathcal{G}_{k,m}$ is q_m. The definition of the automata-based dynamic convolutional cryptosystem with q states may be changed as follow. The encoder is a finite automaton $M = \langle X, Y, S, \delta, \lambda \rangle$, where X and Y are the k-dimensional row vector space over $GF(2)$, $S = X^m \times \{1, \ldots, q\}$, q and m being two positive integers,

$$\begin{aligned}
&\delta(\langle x_{-1}, \ldots, x_{-m}, w \rangle, x_0) = \langle x_0, \ldots, x_{-m+1}, f(w, x_0) \rangle, \\
&\lambda(\langle x_{-1}, \ldots, x_{-m}, w \rangle, x_0) = [x_{-m}, \ldots, x_0] g(x_{-1}, \ldots, x_{-m}, w), \\
&\quad x_0, x_{-1}, \ldots, x_{-m} \in X, w \in \{1, \ldots, q\},
\end{aligned}$$

f being a single-valued mapping from $\{1, \ldots, q\} \times X$ to $\{1, \ldots, q\}$, and g being a single-valued mapping from S to $\mathcal{G}_{k,m}$. The key is the structure of M (i.e., f and g) and the initial state.

A corresponding decoder is the finite automaton $M' = \langle Y, X, S, \delta', \lambda' \rangle$, where

$$\begin{aligned}
&\delta'(\langle x_{-1}, \ldots, x_{-m}, w \rangle, y_0) = \langle x_0, \ldots, x_{-m+1}, f(w, x_0) \rangle, \\
&\lambda'(\langle x_{-1}, \ldots, x_{-m}, w \rangle, y_0) = x_0, \\
&x_0 = y_0 G_0^{-1} - \sum_{j=1}^{m} [x_{-m}, \ldots, x_{-1}] G' G_0^{-1}, \\
&\quad x_0, x_{-1}, \ldots, x_{-m} \in X, w \in \{1, \ldots, q\},
\end{aligned}$$

G' and G_0 being the first mk rows and the last k rows of $g(x_{-1}, \ldots, x_{-m}, w)$, respectively. It is easy to verify that for any $s \in S$ and any $\alpha \in X^*$, $\lambda'(s, \lambda(s, \alpha)) = \alpha$ holds.

[8] $\prod_{h=0}^{q-1}(q_m - h)$ in case where the matrices are different from each other.

The original definition of ADCC in [1] is a special case where $g(x_{-1},\ldots,x_{-m},w)$ does not depend on $x_{-1},\ldots,x_{-m}$, denoted by $g(w)$. In this case,

$$\begin{bmatrix} G_{m,w} \\ \vdots \\ G_{1,w} \\ G_{0,w} \end{bmatrix} = g(w), \qquad w = 1,\ldots,q$$

in original definition of ADCC.

Contrary to the original ADCC, even if $q = 1$, the chosen-plaintext attack is not a trivial task, which can be reduced to solving nonlinear Boolean equations.

From automata-theoretic point of view, any ADCC can be implemented by a "static" cipher. The encoder of the static cipher is a finite automaton $\bar{M}_{kmq} = \langle X, Y, \bar{S}, \bar{\delta}, \bar{\lambda}\rangle$, where X and Y are the k-dimensional row vector space over $GF(2)$, $\bar{S} = X^m \times \{1,\ldots,q\} \times F \times G$, k, m and q being positive integers, F being the set of all single-valued mappings from $\{1,\ldots,q\} \times X$ to $\{1,\ldots,q\}$, G being the set of all single-valued mappings from $X^m \times \{1,\ldots,q\}$ to $\mathcal{G}_{k,m}$,

$$\begin{aligned} &\bar{\delta}(\langle x_{-1},\ldots,x_{-m},w,f,g\rangle, x_0) = \langle x_0,\ldots,x_{-m+1}, f(w,x_0), f, g\rangle, \\ &\bar{\lambda}(\langle x_{-1},\ldots,x_{-m},w,f,g\rangle, x_0) = [x_{-m},\ldots,x_0]g(x_{-1},\ldots,x_{-m},w), \\ &\quad x_0, x_{-1},\ldots,x_{-m} \in X, w \in \{1,\ldots,q\}, f \in F, g \in G. \end{aligned}$$

The key is its initial state. Since $|F| = q^{2^k q}$ and $|G| = (q_m)^{2^{mk} q}$, the size of the key space is $|\bar{S}| = 2^{mk} q^{2^k q+1} (q_m)^{2^{mk} q}$, where $q_m = 2^{k^2 m} \prod_{h=0}^{k-1}(2^k - 2^h) \geq 2^{k^2(m+1)-k}$.

Clearly, for any $\langle x_{-1},\ldots,x_{-m},w\rangle \in S$ and any $\alpha \in X^*$, we have $\lambda(\langle x_{-1},\ldots,x_{-m},w\rangle,\alpha) = \bar{\lambda}(\langle x_{-1},\ldots,x_{-m},w,f,g\rangle,\alpha)$, where f and g are given in the definition of M.

4 Conclusion

We have given a lower bound of the number of the solutions $G_{h,w_j}, h = 0,1,\ldots,m$ of the system of equations (5) and (7) for almost uniform f. In case of large state number, the lower bound shows that the solution is not unique in general. In case of small state number other than 1, the probability of that (4) holds is too small. Therefore, the deduction in Section 4.1 of [1] cannot work.

In addition, by means of expanding the state alphabet, the automata-based dynamic convolutional cryptosystem can be implemented by an automata-based static cryptosystem.

References

[1] D. Trincă, *Efficient FPGA Implementations and Cryptanalysis of Automata-based Dynamic Convolutional Cryptosystems*, available as Cryptology ePrint Archive Report 2006/263.

[2] R. J. Tao, On finite automaton one key cryptosystems, in *Fast Software Encryption*, Lecture Notes in Computer Science 809, Springer-Verlag, Berlin, 1994, 135–148.