A Novel Stream Cipher on Feed-forward Model

Yuhang Xing, Min Li and Litao Wang

ABSTRACT

Feed-forward model is a basic pseudo random sequence generator (PRSG) using in stream cipher. A novel feed-forward model is constructed. Inversion attack on feed-forward model is analysed and detailed inversion attack algorithms and complexity of these algorithms are given, design advice on being resistant to inversion attack is presented on the basis of complexity analysis. The random attributions of the generated number are test with regards to period, balance, correlation and run property. And the result shows that the random number generator generates numbers with very good randomness.

INTRODUCTION

Nonlinear combination generator is a kind of pseudo random sequence generator that is often used in stream cipher [1-2]. Pseudo random sequence generator is comprised of a linear feedback shift register and a feed-forward logic is most basic one. The model that consists of single register and feed-forward function is called feed-forward model in this paper. When feed-forward model of the synchronous stream cipher generator is designed in cryptography, it is always divided into two parts, the driving part and the nonlinear combination part [3]. The driving part uses the LFSR sequences and the nonlinear combination part do some confusion and diffusion with the sequences generated by linear part.

Actually, if the state transition matrix is well designed, the state serial and its fixed component can be obtained when any initial state is given, and thus pseudo random sequence of a large period will be generated. That is to say, linear shift register is implemented from a special state transition matrix. So even with the state transition matrix that is a general one, sequences meet the requirements of the
driving part will be obtained if it is designed well enough. And there are many attack methods with different feed-forward models, such as guess and decision attack, fast correlation attack and algebraic attacks [4-7]. J. Dj. Golic proposed the extended inversion attack specified to the feed-forward model in 2000. The attack is a known plaintext attack without exhaustive search of all possible initial state, and it shows its fast and simple features in some special situations. We propose a novel feed-forward model based on special state transition matrix by designing the state transition matrix. The model is analysed with the inversion attack, and the specific attack algorithm as well as the complexity analysis are also given in this paper. On the basis of complexity analysis, the design strategy of anti-inversion attack feed-forward model based on special state transition matrix is put forward.

In this paper, the symbols agree as follows:

\[ F_2 = \{0, 1\} \oplus \] indicates the addition modulo 2, \( F_2^n = \{0, 1\}^n \) means m-dimensional vector space on \( F_2 \); \( x \in F_2^m \), denoted as \( x = (x_1, x_2, \ldots, x_m) \); \( \mathbb{Z} \) indicates the set of integers and \( \mathbb{Z}^+ \) the set of positive integers. The finite field with q elements is noted as \( GF(q) \).

**SPECIAL STATE TRANSITION MATRIX AND FEED-FORWARD MODEL**

In order to build a new class of feed-forward model, special state transition matrix is defined as follows,

**Definition 1** We call such matrix in the form of \( T = (c_{ij}) \) a special state transition matrix, if

\[
  c_{ij} = \begin{cases} 
    a_i, & \text{if } i = j; \\
    b_j, & \text{if } i = j + 1; \\
    d_i, & \text{if } i = j - 1; \\
    0, & \text{else.}
  \end{cases}
\]

where \( a_i, b_j, d_i \in F_2 \).

**Theorem 1** Let \( T \) be a special state transition matrix on \( F_2 \) and primitive, there is

\[ b_i = d_i = 1, \quad \forall 1 \leq i \leq m - 1. \]

Just as the feed-forward model of shift register transformation based on multiplication circuits, the following feed-forward model is defined. In this section we will discuss the situation when the state transition matrix is primitive [8].

**Definition 2** Let \( T \) be special state transition matrix, then we call

\[
  (x_1, x_2, \ldots, x_m)^T = (0, b_1 x_1, \ldots, b_m x_m) \oplus (a_1 x_1, a_2 x_2, \ldots, a_m x_m) \oplus (d_1 x_2, d_2 x_3, \ldots, 0),
\]

the sequence generated by state transition transformation, which is a linear recursive sequence denoted as \( \{S_i\}_{i=1}^\infty \). This transformation is also call state transition transformation based on multiplication circuits, where \( S_i = (S_{i-1})^T \). And \( S_i = (x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)}) \) is the state of the linear recursive sequence at time \( t \). Again, we
let $f : F_2^n \rightarrow F_2^k$ be a multi-output Boolean function, and the positive integers $\lambda_1, \lambda_2, \ldots, \lambda_n$ satisfy $1 \leq \lambda_1 < \lambda_2 < \cdots < \lambda_n \leq m$. For $t \geq 1$, let $y_t = f(x_t^{(i)}, x_t^{(j)}, \ldots, x_t^{(l)})$, then we say the sequence $\{y_t\}_{t=1}^\infty$ is the feed-forward sequence of the feed-forward model designed by multiplication circuits based on special state transition matrix.

The main problem of the feed-forward model is how to design the state transition matrix, tap positions, feed-forward function $f$. Special transition matrix is designed; Feed-forward function $f$ is nonlinear usually, we use a nonlinear Boolean function existing and don’t design feed-forward function in the paper. We design tap positions of feed-forward model according to the following analysis. The basic problem of the feed-forward model is how to find the initial state when the state transition matrix, tap positions $\lambda_1, \lambda_2, \ldots, \lambda_n$, feed-forward function $f$ and feed-forward sequence of $\{S_t\}_{t=1}^\infty$ are all given. All the feed-forward functions discussed in the paper are Boolean functions, and so it is with the multi-output feed-forward function. Obviously, special state transition matrix discussed is primitive in the paper, so there is $b_i = d_i = 1$, $\forall 1 \leq i \leq m-1$.

### INVERSION ATTACK ON THE FEED-FORWARD MODEL BASED ON SPECIAL STATE TRANSITION MATRIX

In this section, we first analyze the feed-forward model based on the multiplication circuits of special state transition matrix. There are two methods on inversion attack according to two conditions.

**Condition I:** If $\forall 1 \leq i \leq n-1$, there is $d(\lambda_{i+1}, \lambda_i) < d(\lambda_{i}, \lambda_i)$, $d(\lambda, \lambda)$ is denoted distance between $\lambda_i$-th tap and $\lambda_j$-th tap.

In general, we denote the input of $f$ at time $t$ (i.e. a segment obtained from the $t$-th state $S_t$) as $X^{(t)} = (x^{(t)}_1, x^{(t)}_{i+1}, x^{(t)}_{i+2}, \ldots, x^{(t)}_{\lambda_i-1}, x^{(t)}_{\lambda_i})$, where $\lambda_i$ represents the starting time.

Since $X^{(t+1)} = (x^{(t+1)}_1, x^{(t+1)}_{\lambda_i+1}, x^{(t+1)}_{\lambda_i+2}, \ldots, x^{(t+1)}_{\lambda_i-1}, x^{(t+1)}_{\lambda_i})$, when $\lambda_i = 1$, then

$$X^{(t+1)} = (x^{(t+1)}_{\lambda_i}, x^{(t+1)}_{\lambda_i+1}, \ldots, x^{(t+1)}_{\lambda_i}) = (0, x^{(t)}_{\lambda_i}, \ldots, x^{(t)}_{\lambda_i}) \oplus (a_{\lambda_i} x^{(t)}_1, a_{\lambda_i} x^{(t)}_{\lambda_i+1}, \ldots, a_{\lambda_i} x^{(t)}_{\lambda_i-1}) \oplus (x^{(t)}_{\lambda_i+1}, x^{(t)}_{\lambda_i+2}, \ldots, x^{(t)}_{\lambda_i}) \quad (1)$$

When $\lambda_i > 1$, then

$$X^{(t+1)} = (x^{(t+1)}_{\lambda_i}, x^{(t+1)}_{\lambda_i+1}, \ldots, x^{(t+1)}_{\lambda_i}) = (x^{(t)}_{\lambda_i}, x^{(t)}_{\lambda_i+1}, \ldots, x^{(t)}_{\lambda_i}) \oplus (a_{\lambda_i} x^{(t)}_1, a_{\lambda_i} x^{(t)}_{\lambda_i+1}, \ldots, a_{\lambda_i} x^{(t)}_{\lambda_i-1}) \oplus (x^{(t)}_{\lambda_i+1}, x^{(t)}_{\lambda_i+2}, \ldots, x^{(t)}_{\lambda_i}) \quad (2)$$
(1) If $\lambda = 1$, with given $X^{(t+1)}$, can be uniquely confirmed if $x_{s_{t+1}}^{(t)}$ has been already known.
(2) If $\lambda \neq 1$, with given $X^{(t+3)}$, can be uniquely confirmed if $x_{s_{t-1}}^{(t)}, x_{s_{t+1}}^{(t)}$, has been already known.

So we can obtain all the possible solutions of $X^{(t+1)}$. That is all the possible solutions of $\{X^{(i)}\}_{i=t}^{t+n}$ can be easily calculated with the given $X^{(t)}$, and then we can filter the correct solution and find the initial state $S_i$ of the recursive sequence.

Here we find all the possible solutions of $X^{(t+3)}$ by determining whether the equation

$$y_{t+1} = f(x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, \ldots, x_{s_{t}}^{(t)})$$

(3)

is true: we exhaust all the possible $x_{s_{t+1}}^{(t)}$, calculate all the possible $(x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, \ldots, x_{s_{t}}^{(t)})$ and retain a set of $X^{(t+1)}$ which make equation (3) true.

Condition II: If $\exists 1 \leq i \leq n-1$, there is $d(\lambda_{i+1}, \lambda_i) \geq d(\lambda_n, \lambda_i)$.

In general, we denote the input of $f$ at time $t$ (i.e. a segment obtained from the $t$-th state $S_t$) as $X^{(t)} = (x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, \ldots, x_{s_{t}}^{(t)})$, where $\lambda_{t+1}$ represents the starting time. So

$$X^{(t+4)} = (0, x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, x_{s_{t+4}}^{(t)}, \ldots, x_{s_{t}}^{(t)})$$

$$\oplus (a_{s_{t+1}} x_{s_{t+1}}^{(t)}, a_{s_{t+2}} x_{s_{t+2}}^{(t)}, a_{s_{t+3}} x_{s_{t+3}}^{(t)}, a_{s_{t+4}} x_{s_{t+4}}^{(t)}, \ldots, a_{s_{t+4}} x_{s_{t+4}}^{(t)}) \oplus (x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, x_{s_{t+4}}^{(t)}, x_{s_{t+5}}^{(t)}, \ldots, 0)$$

(4)

Then given $X^{(t)}$, $X^{(t+1)}$ can be uniquely confirmed if $x_{s_{t+1}}^{(t)}, x_{s_{t+1}}^{(t)}$ has been already known.

So we can obtain all the possible solutions of $X^{(t+1)}$. That is all the possible solutions of $\{X^{(i)}\}_{i=t}^{t+n}$ can be easily calculated with the given $X^{(t)}$, and then we can filter the correct solution and find the initial state $S_i$ of the recursive sequence.

Here we find all the possible solutions of $X^{(t+1)}$ by determining whether the equation

$$y_{t+1} = f(x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, \ldots, x_{s_{t}}^{(t)})$$

(5)

is true: we exhaust all the possible $x_{s_{t+1}}^{(t)}$, calculate all the possible $(x_{s_{t+1}}^{(t)}, x_{s_{t+2}}^{(t)}, x_{s_{t+3}}^{(t)}, \ldots, x_{s_{t}}^{(t)})$ and retain a set of $X^{(t+1)}$ which make equation (5) true.

The algorithm of inversion attack in condition I is detailed as follows:

Algorithm 1 ($\lambda = 1$):
**Inputs** of algorithm 1 are listed as follows,  
(1) The level of the linear sequence designed by multiplication circuits based on special state transition matrix is $m$;  
(2) The structure parameters of the linear recurrence relation $a_1, a_2, a_3, \ldots, a_{n-1}$;  
(3) Feed-forward function $f$ and its tap positions $\lambda_1, \lambda_2, \ldots, \lambda_n$;  
(4) Segment with length of $T$ of the feed-forward sequence $\{y_t\}_{t=1}^{\infty}$.

**Output**: The initial state $S_i$ of the linear recursive sequence generated by the multiplication circuits based on special state transition matrix.

**Pre-processing**:
(1) For $t \geq 1$, compute the $m$-dimension vector $\alpha_i$ on $F_2$ that makes $x_{i+1}^{(i)} = \alpha_i \cdot S_i$ true. And denote $\alpha_i$ a $m$-dimension vector on $F_2$ with the $i$-th component of it equals 1 and all the others are 0.  
(2) For $t \geq 1$, let $c_i = 1$ if $x_{i+1}^{(i)}$ has the linear constraint relation given by definition 1, or $c_i = 0$. Then we find the numbers of the non-zero coefficient of the linear constraint relation equation when $c_i = 1$, and denote the minimum time of $x_{i+1}^{(i)}$ with the linear constraint relation as $t_i$.  
(3) For $t \geq m - \lambda_n + 1$, examine whether the rank of the coefficient matrix of the equations set with $x_{i+1}^{(i)}, x_{i+2}^{(i)}, \ldots, x_{i+1}^{(i)}, x_{i+2}^{(i)}, \ldots, x_{i+1}^{(i)}$ (concerned with component $S_i$) is $m$ and denote the minimum time as $T_m$ when the rank is $m$.

**Main Algorithms As Follows.**

**Step1** Let $T = T_m + \varepsilon$.

**Step2** For any possible value of $X^{(i)} = (x_{i+1}^{(i)}, x_{i+2}^{(i)}, \ldots, x_{i+1}^{(i)})$, Select the input of $f$, $x^{(i)} = (x_{i+1}^{(i)}, x_{i+2}^{(i)}, \ldots, x_{i+1}^{(i)})$. Then calculate $z = f(x^{(i)})$. Go to Step2 if $z \neq y_i$ and select the next possible value of $X^{(i)} = (x_{i+1}^{(i)}, x_{i+2}^{(i)}, \ldots, x_{i+1}^{(i)})$. Set $t = 2$ and $\text{deep} = 1$ when $z = y_i$ and go to Step3.  

**Step3** Go to Step6 if $t \geq T$. Exhaust $x_{i+1}^{(i-1)}$ if $t < t_i$, save the set $\Omega_{\text{deep}}$ consisting of all the possible values of $x_{i+1}^{(i-1)}$ and go to Step4. Else find the value of $x_{i+1}^{(i-1)} a$ by using linear constraint relation, then go to Step5.

**Step4** Pop the first value of set $\Omega_{\text{deep}}$;  
**Step5** According to (1) and the saved $X^{(i-1)}$, calculate $X^{(i)} = (0, x_{i+1}^{(i-1)}, \ldots, x_{i+1}^{(i-1)}) \oplus (a_{i+1} x_{i+1}^{(i-1)}, a_{i+2} x_{i+2}^{(i-1)}, \ldots, a_{i+1} x_{i+1}^{(i-1)}) \oplus (x_{i+1}^{(i-1)}, x_{i+2}^{(i-1)}, \ldots, a)$.

\[230\]
and save $X^{(i)}$. Then get the input of $f$, $x^{(i)} = (x_{A_1}^{(i)}, x_{A_2}^{(i)}, \ldots, x_{A_k}^{(i)})$, so as to obtain $z = f(x^{(i)})$. If the set $\Omega_{deep}$ is empty and $z \neq y_i$, minus deep by 1. Go back to Step2 to check the next possible values of $X^{(i)}$ if deep $= 0$, else go to Step4. If set $\Omega_{deep}$ is not empty, pop the next value of $\Omega_{deep}$ and redo Step5. Add $t_{deep}$ by 1 when $z = y_i$ and return to Step3.

Step6 The algorithm ends until all the possible solutions of the initial state are found according to $X^{(i)}$ and $x_{A_1}^{(i)}, x_{A_2}^{(i)}, \ldots, x_{A_k}^{(i)}$.

We can determine the time of the initial state by $T_w$. According to the above analysis of algorithm 1, if $T_w < m$, it is better to select a bigger $\varepsilon$ to make the expectation approximation $2^{m-T}$ of incorrect initial state smaller. It can be respected that the obtained initial state is unique with $\varepsilon = 10$ when $T_w \geq m$.

The complexity of algorithm 1 is analysed as follows.

**Theorem 2** Given $T_w, \varepsilon, t_i$, then the maximum computational complexity of algorithm 1 approximates to $(t_i + 1)2^k$ times execution of $f$, its storage complexity is $O(\lambda_n T_w)$, and data complexity is $T_w + \varepsilon$ random numbers. The probability of finding correct $S_i$ closes to $2^{2\varepsilon - m + \varepsilon + 1}$.

With the help of verification method of algorithm 1, its maximum computational complexity is $(t_i + 1)2^k$ times $f$ executions. Which means it’s far smaller than the half of maximum computational complexity.

Similarly, we can give results and algorithm of $\lambda_i = 1$ and condition II, which is not given because of length of the paper is limited.

**DESIGN STRATEGY OF ANTI-INVERSION ATTACK FEED-FORWARD MODEL**

So as to resist the inversion attack, there are some issues we should pay attention to with the tap positions choosing in our method. According to theorem 1, assume that $T_w, \lambda_i = 1$, $t_i$ are given, the maximum computation complexity of algorithm 1 is closed to $(t_i + 1)2^k$ times executions of function $f$. Obviously, assume that $\lambda_n = m$ are given, the maximum computation complexity of algorithm 1 is closed to $(t_i + 1)2^m$ the inversion attack is least effective when $\lambda_n = m, \lambda_i = 1, \lambda_{i+1} = 2$, when $i$ satisfies $d(x_{A_i}^{(i)}, x_{A_i}^{(i)}) \forall 1 \leq i \leq n - 1$, which is the best design strategy of anti-inversion attack schemes. On the other hand, if the number
of input taps is \( n \),

\[
d(x^{(i)}_h, x^{(i)}_k) = d(x^{(i)}_h, x^{(i)}_k) = \cdots = d(x^{(i)}_h, x^{(i)}_k)
\]

is the best design strategy of anti-inversion attack schemes.

**EXPERIMENT AND TEST**

A random number generator according to the feed-forward model based on special state transition matrix is implemented in C. Then we test the generated random number on a computer with Pentium IV 1.3GHz processor, 1GB memory.

The random attributions of the generated number are test with regards to period, balance, correlation and run property. And the result shows that the random number generator based on special state transition matrix generates numbers with very good randomness.

**REFERENCES**