SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2020, Vol. 63 212206:1–212206:13
https://doi.org/10.1007/s11432-019-2813-7

Boolean-network-based approach for construction of filter generators

Bowen LI[1,2] & Jianquan LU[2*]

[1]*School of Information Science and Engineering, Southeast University, Nanjing* 210096*, China;*
[2]*School of Mathematics, Southeast University, Nanjing* 210096*, China*

Received 24 November 2019/Revised 14 January 2020/Accepted 5 February 2020/Published online 9 October 2020

Abstract In this paper, we view filter generators as Boolean networks (BNs), and discuss their power-analysis-based side-channel analysis. An incompletely specified binary sequence always contains some bits called unnecessary bits comprising 1 or 0. Our motivation for considering this type of sequence is to reduce direct dependencies between side-channel information and key sequences. An algorithm is proposed to determine the unnecessary bits to increase the key search time required for adversaries rather than simply turning all unnecessary bits to 0 (or 1). Then, to reduce area dissipation, under the framework of semi-tensor product (STP) of matrices, the problem of constructing filter generators with minimum number of stages is converted into the one of determining the corresponding transition matrices. Compared with the existing results, the lower bound of the minimum number of stages is provided, which can reduce the exhaustive search time required to find it. Finally, one example is used to illustrate the efficacy of the proposed algorithm.

Keywords filter generator, Boolean network, semi-tensor product

Citation Li B W, Lu J Q. Boolean-network-based approach for construction of filter generators. Sci China Inf Sci, 2020, 63(11): 212206, https://doi.org/10.1007/s11432-019-2813-7

1 Introduction

With the growth of Internet-of-Things applications, an increasing number of every-day-life applications have become security-critical, demanding high levels of assurance. Stream ciphers having good correlation properties have been widely applied to protect confidential information via Internet communications, error-correcting codes, spread-spectrum communications, etc. These ciphers are generally faster and less complex for hardware devices than block ciphers [1]. Therefore, in 2004, the ECRYPT (European Network of Excellence for Cryptology) launched a program called eSTREAM to identify stream ciphers that could be widely used. Seven algorithms having good applicability were selected. Note that feedback shift registers (FSRs) were the main building blocks in three types of selected stream ciphers: Grain, Trivium and Michkey. An FSR comprises combinational logical circuits, feedback function, and the storage cells (i.e., stages). For the past few decades, the use of FSRs has resulted in a relatively mature theory, and some interesting results have been obtained, such as [2, 3].

Recently, a new linear representation for FSRs was proposed in [4], where FSRs were viewed as Boolean networks (BNs). A BN is a class of logical networks, and was firstly used to model genetic regulatory systems in [5]. BNs are quite different from common complex networks [6]. A BN comprises n nodes and n Boolean functions, where each node $i \in [1, n]$ has an associated state variable given as x_i taking a value of either 0 or 1. Then, x_i is updated with its corresponding Boolean function denoted by

* Corresponding author (email: jqluma@seu.edu.cn)

$f_i : \{0,1\}^n \to \{0,1\}$. The semi-tensor product (STP) was proposed by Cheng et al. [7]. It has become a commonly used tool to analyze BNs. It is a generalization of the conventional matrix product, and breaks the traditional dimension-matching condition required for matrix products. By utilizing the method of STP, logical systems can be converted into their corresponding algebraic forms. Based on these, many interesting results have been found, including controllability [8,9], stability and stabilization [10–15], function perturbation [16], observability [17], disturbance decoupling [18], output tracking [19] and optimal control [20]. Additionally, the STP can be also applied in the game [21] and fault detection for logical circuits [22].

Because FSRs are considered as BNs, many matured BN theories can be applied. Compared with their traditional analysis methods, an FSR can now be converted into a conventional discrete-time linear system with the help of STP. In [23], the transformation between two configurations of FSRs was analyzed using STP, where a uniform condition was not required. In [24], Dubrova analyzed non-singularity of FSRs with nonlinear configurations by converting them into their corresponding algebraic normal forms. One sufficient condition was derived. In [25, 26], some necessary and sufficient conditions for non-singularity were obtained by utilizing STP. Apart from non-singularity, a few studies including ones of driven stability [2], decomposition [3] and minimum period of FSRs in Grain-like configurations [27], were obtained under the STP.

Obviously, if there exist some algorithms that can break one stream cipher, then the encryption mechanism is not very secure. Note that the binary sequence generated by an FSR having a linear configuration can be easily determined if $2n$ bits are known. Therefore, in order to increase the security of the cipher, the outputs from FSRs with n stages become the input n-vector to a Boolean function $g(x_1, x_2, \ldots, x_n)$. Then, the output sequences comprise the values of g. This structure is a filter generator that can increase the complexity of key sequences. Based on the different requirements of key sequences, corresponding filter generators has been constructed.

There exist two kinds of key sequences, that are: completely specified binary sequences and incompletely specified binary sequences. In detail, each bit in the completely specified binary sequences is certain, whereas, in the incompletely specified binary sequences, there exist some bits whose values can be either 1 or 0. We call these bits unnecessary bits or do-not-care bits [28]. For example, there is an 8-bit incompletely specified binary sequence: 0, 0, z_1, 1, 0, 1, 0, z_2 with z_1 and z_2 being unnecessary bits. This implies that the information of the corresponding positions of z_1 and z_2 is trivial. Thus, bits z_1 and z_2 can be selected arbitrarily from $\{0,1\}$, and the values of other positions are fixed. Specifically, when the values of z_1 and z_2 are determined, then the incompletely specified binary sequence will become the completely specified binary sequence. Thus, we have four possible sequences, that are: (1) 0, 0, 1, 1, 0, 1, 0, 1; (2) 0, 0, 1, 1, 0, 1, 0, 0; (3) 0, 0, 0, 1, 0, 1, 0, 1; (4) 0, 0, 0, 1, 0, 1, 0, 0. For a given completely specified binary sequence, filter generators were constructed to generate the sequence in [29], and the number of stages of constructed filter generators was guaranteed to be minimal. However, the search space was too large in order to find the minimum number of stages. In this paper, we continue the search for an effective algorithm to reduce the search space of the minimum number of stages.

Side channel analysis is a class of cryptanalysis, and is used to conjecture keystreams by observing information obtained from the physical implementation of a cipher, such as timing information and power consumption. In other words, by effectively exploiting the unintentional leakage of information from applications, a system breakdown can be achieved, and the targeted internal state or the secret key can be deduced. For an FSR having a nonlinear configuration, a power-analysis-based side-channel analysis can be practically applied to complementary metal oxide semiconductor hardware to determine the bit values of an FSR having a nonlinear configuration, as was accomplished in [30]. However, a few results utilized side-channel analysis to investigate filter generators. In this paper, the power-analysis-based side-channel analysis is used to improve the security of constructed filter generators.

We stress the following interesting issues in this paper. First, for an incompletely specified binary sequence, a power-analysis-based side-channel analysis is used to determine the unnecessary bits. As introduced in [28], there are usually two approaches to ascertaining the unnecessary bits: (1) setting all unnecessary bits to 0 (or 1); (2) setting them to random values. In this paper, from a security perspective, an algorithm is proposed to determine unnecessary bits to increase the key search time for adversaries. Then, the incompletely specified binary sequence becomes one completely specified binary sequence. Furthermore, in order to reduce the area and power dissipation, Dubrova in [31, 32] proposed some algorithms to find the minimum number of stages of binary machines for completely specified binary sequences. However, these algorithms could not be applied to filter generators because of the chain connections between registers in FSRs [29]. Thus we utilize BNs to construct filter generators to generate completely specified binary sequences. Hence, we not only find the minimum number of stages but we also find the corresponding transition matrix. Additionally, to improve the security of the filter generators, balanced functions are discussed, and corresponding structure matrices are constructed.

The rest of this paper is organized as follows. Section 2 introduces some preliminaries on BNs and STP, and then explains how to convert a logical system into its corresponding algebraic form. Section 3 proposes some interesting algorithms for constructing filter generators to improve security and reduce area and power dissipation. Section 4 provides conclusion to this paper.

2 Preliminaries

In the following, we present some necessary notations, which will be used in the rest of this paper.

- $\mathcal{D} = \{0, 1\}$ and $\mathcal{D}^n = \underbrace{\{0, 1\} \times \{0, 1\} \times \cdots \times \{0, 1\}}_{n}$.
- Let $[a, b]$ be the set of $\{a, a + 1, \dots, b\}$ with a and b being integers and $b \geqslant a$.
- $\Delta_n := \{\delta_n^i : 1 \leqslant i \leqslant n\}$, where δ_n^i represents the ith column of identity matrix of size $n \times n$.
- A matrix B whose elements are all integers is called a logical matrix if all the columns of B are in the set Δ_n.
- Let $\mathcal{L}_{m\times n}$ be the set of $m \times n$ logical matrices.
- $B = [\delta_m^{i_1}, \delta_m^{i_2}, \dots, \delta_m^{i_n}] \in \mathcal{L}_{m\times n}$, which can also be represented by $B = \delta_m[i_1, i_2, \dots, i_n]$ for brevity.
- Notation $*$ represents the Khatri-Rao product.
- Let $\mathrm{col}_i(B)$ be the i-th column of the matrix B.

A BN consists of n nodes and n Boolean functions [7]. Each node $i \in [1, n]$ has an associated state variable denoted by x_i that represents the current value of the node i, and $x_i(t)$ represents the value of node i at time t. The value of node i is updated by its corresponding Boolean function denoted by $f_i : \{0, 1\}^n \to \{0, 1\}$. A BN having n nodes can be described by the following form:

$$\begin{cases} x_1(t+1) = f_1(x_1(t), x_2(t), \dots, x_n(t)), \\ x_2(t+1) = f_2(x_1(t), x_2(t), \dots, x_n(t)), \\ \qquad \vdots \\ x_n(t+1) = f_n(x_1(t), x_2(t), \dots, x_n(t)). \end{cases} \tag{1}$$

Clearly, BNs are logical systems, and every state variable of a BN takes a value from the set $\mathcal{D}$. BNs can be used to model many systems such as gene regulatory networks [5]. Recently, BNs have also been widely used to study logical circuits and to analyze and synthesize FSRs with the help of the STP method.

Definition 1 ([7]). The STP of two matrices $A \in M_{m\times n}$ and $B \in M_{p\times q}$, is defined as

$$A \ltimes B = \left(A \otimes I_{\frac{l}{n}}\right)\left(B \otimes I_{\frac{l}{p}}\right), \tag{2}$$

where $\otimes$ is the Kronecker product and $l = \mathrm{lcm}(n, p)$ is the least common multiple of n and p.

Note that in the BNs, $\delta_2^1 \sim 1$ and $\delta_2^2 \sim 0$. Therefore, $\mathcal{D}$ ($\mathcal{D}^n$) and Δ_2 (Δ_{2^n}) represent two different forms of the same object, and can be used interchangeably. Generally, we say X is a scalar if $X \in \mathcal{D}^n$. Inversely, X is in a vector form if $X \in \Delta_{2^n}$. To distinguish these two different forms, X represents the scalar form and x represents the vector form. The equivalent relation $\delta_{2^n}^i \sim (i_1, i_2, \dots, i_n)$ satisfies that $i = 2^n - (i_1 2^{n-1} + i_2 2^{n-2} + \cdots + i_n)$.

Lemma 1 ([7]). Consider a logical function $f(x_1, x_2, \ldots, x_n)$, with logical variables $x_1, x_2, \ldots, x_n$. There exists a unique matrix $M_f \in \mathcal{L}_{2\times 2^n}$, called the structure matrix of f, such that in vector form, we have

$$f(x_1, x_2, \ldots, x_n) = M_f \ltimes_{i=1}^{n} x_i,$$

where $\ltimes_{i=1}^{n} x_i = x_1 \ltimes x_2 \cdots \ltimes x_n \in \Delta_{2^n}$.

In fact the structure matrix can be obtained from the corresponding truth table of the logical function. For example, $x_1(t+1) = x_1(t) \vee x_2(t)$ with $x_i \in \mathcal{D}, i \in [1, 2]$. Then, we consider four cases: (1) $x_1(t) = x_2(t) = 1$; (2) $x_1(t) = 1, x_2(t) = 0$; (3) $x_1(t) = 0, x_2(t) = 1$; and (4) $x_1(t) = 0, x_2(t) = 0$. Thus, we have $x_1(t+1) = 1, 1, 1$, and 0 successively. Because $\delta_2^1 \sim 1$ and $\delta_2^2 \sim 0$, the corresponding structure matrix $M_g = \delta_2[1\ 1\ 1\ 2]$, i.e., $x_1(t+1) = M_g x_1(t) x_2(t), x_i \in \Delta_2, i \in [1, 2]$. Therefore, for each logical function, there is a logical matrix such that the logical expression can be equivalently to an algebraic expression. We assume that $M_{f_i}, i \in [1, n]$ is the unique structure matrix of logical function f_i. Thus, via STP, the corresponding algebraic expression of system (1) can be obtained:

$$x(t+1) = Lx(t), \tag{3}$$

where $\mathrm{col}_i(L) = \mathrm{col}_i(M_{f_1}) * \mathrm{col}_i(M_{f_2}) * \cdots * \mathrm{col}_i(M_{f_n}), i \in [1, 2^n]$ is called the transition matrix of system (1), and $x(t) = \ltimes_{i=1}^{n} x_i$. The state of a BN is in the form of $(x_1, x_2, \ldots, x_n) \in \mathcal{D}^n$. The next state of BNs can be determined from the current state via a transition matrix. Then, the trajectory of system (1) can also be determined by L after the initial state is given.

The state-transition graph is a directed graph and consists of 2^n nodes and 2^n directed edges. The 2^n nodes represent the 2^n states of a BN. If the state vector $(i_1, i_2, \ldots, i_n)$ is changed into $(i_1', i_2', \ldots, i_n')$ next time, then there exists a directed edge from the node representing $(i_1, i_2, \ldots, i_n)$ to the node representing $(i_1', i_2', \ldots, i_n')$. Obviously, the state-transition graph plays the same role as the transition matrix L.

Example 1. The following BN was constructed in [33] to simulate the dynamics of the reduced model for Th-lymphocyte differentiation:

$$\begin{cases} x_1(t+1) = f_1(x_1(t), x_2(t)) = x_1(t) \wedge \neg x_2(t), \\ x_2(t+1) = f_2(x_1(t), x_2(t)) = \neg x_1(t) \wedge x_2(t), \end{cases} \tag{4}$$

where x_1 and x_2 represent T-bet and GATA-3, respectively. The structure matrices can be calculated as $M_{f_1} = \delta_2[2\ 1\ 2\ 2]$ and $M_{f_2} = \delta_2[2\ 2\ 1\ 2]$. Let $x(t) = x_1(t)x_2(t) \in \Delta_{2^2}$, and then the corresponding algebraic expression can be obtained:

$$x(t+1) = Lx(t) = \delta_4[4\ 2\ 3\ 4]x(t). \tag{5}$$

According to the definition of the state-transition graph, there are four nodes labeled by $(1,\ 1), (1,\ 0), (0,\ 1), (0,\ 0)$. According to (5), the state-transition graph is shown in Figure 1, where $(1,\ 1) \sim \delta_4^1, (1,\ 0) \sim \delta_4^2, (0,\ 1) \sim \delta_4^3$ and $(0,\ 0) \sim \delta_4^4$.

Based on the algebraic expression and the state-transition graph, many interesting results and applications about BNs can be obtained as mentioned in Section 1. Moreover, in analysis and synthesis logical circuits, the Boolean-network-based method shows some advantages which are presented latter.

3 Main results

A filter generator is composed of an FSR, denoted by c_f and external circuit c_e (see Figure 2). The c_f comprises of n binary storage elements that are usually called stages, and the associated state variable of each stage $i \in [1, n]$ is denoted by x_i. The state of an FSR is the ordered set of values of its state variables $(x_1, x_2, \ldots, x_n)$, denoted by X. As usual, the first stage is called the lowest stage, and the n-th stage is called the highest stage. In c_f, every updated function $f_i, i \in [1, n-1]$, except for f_n takes the form of $f_i = x_{i+1}(t)$, indicating that the value of function f_i only depends on $x_{i+1}(t)$, and the value of

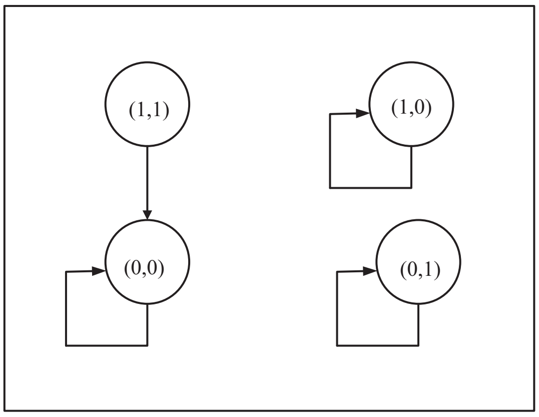

Figure 1 The state-transition graph of system (4).

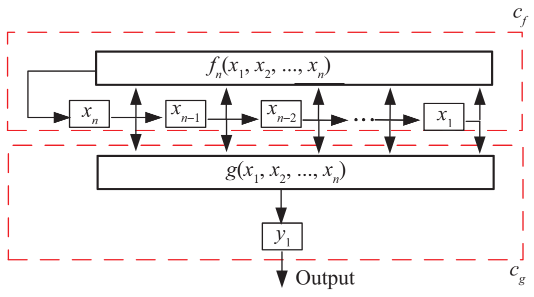

Figure 2 (Color online) A filter generator, where the arrow represents the passing directions of states.

stage $i+1$ is transmitted to that of stage i at every time. However, f_n potentially depends on all of the values of n stages, and is of type:

$$f_n = f_n(x_1, x_2, \dots, x_n).$$

Clearly, the updated process of finding the value for the stage n differs from other stages, and it is potentially dependent on values of some stages via the updated function f_n rather than simply transmitting the value of the upper stage to that of the next stage (as shown in Figure 2). Therefore, the next state overlaps the current state at $n-1$ positions in every time except for the stage n. Generally, the updated function f_n is called the feedback function of c_f. Thus, once the updated function is determined, c_f is therewith determined. The FSR can be described as

$$\begin{cases} x_i(t+1) = x_{i+1}(t), \quad i \in [1, n-1], \\ x_n(t+1) = f_n(x_1(t), x_2(t), \dots, x_n(t)). \end{cases} \tag{6}$$

The external circuit c_e comprises a logical function whose inputs are taken from certain stages to produce the output, denoted by g. Then, a filter generator can be described as

$$\begin{cases} x_i(t+1) = x_{i+1}(t), \quad i \in [1, n-1], \\ x_n(t+1) = f_n(x_1(t), x_2(t), \dots, x_n(t)), \\ y_1(t) = g(x_1(t), x_2(t), \dots, x_n(t)). \end{cases} \tag{7}$$

Note that functions f_n and g are not minimally represented, implying that there potentially exist $i_1, i_2 \in [1, 2^n]$, such that

$$\begin{aligned} & f_n(x_1(t), \dots, x_{i_1-1}(t), 1, x_{i_1+1}(t), \dots, x_n(t)) \\ & \quad = f_n(x_1(t), \dots, x_{i_1-1}(t), 0, x_{i_1+1}(t), \dots, x_n(t)), \end{aligned}$$

and

$$\begin{aligned} & g(x_1(t), \dots, x_{i_2-1}(t), 1, x_{i_2+1}(t), \dots, x_n(t)) \\ & \quad = g(x_1(t), \dots, x_{i_2-1}(t), 0, x_{i_2+1}(t), \dots, x_n(t)). \end{aligned}$$

Assume that the structure matrices of feedback functions f_n and g are M'_{f_n} and M_g by Lemma 1. For any $i \in [1, n-1]$, the corresponding Boolean function of x_i denoted by $f_i(x_1(t), x_2(t), \dots, x_n(t))$ can be regarded as $f_i(x_1(t), x_2(t), \dots, x_n(t)) = x_{i+1}(t)$ in (6). Then, via Lemma 1, we can also obtain the corresponding structure matrix denoted by M'_{f_i}. Therefore, system (6) can be rewritten as

$$\begin{cases} x_i(t+1) = M'_{f_i} x_1(t) x_2(t) \cdots x_n(t), \qquad i \in [1, n], \\ y_1(t) = M_g x_1(t) x_2(t) \cdots x_n(t). \end{cases} \tag{8}$$

By multiplying all equations together in (8), we get

$$\begin{cases} x(t+1) = L'x(t), \\ y_1(t) = M_g x(t), \end{cases} \tag{9}$$

where $x(t) = \ltimes_{i=1}^{n} x_i(t),\ \mathrm{col}_i(L') = \mathrm{col}_i(M'_{f_1}) * \mathrm{col}_i(M'_{f_2}) * \cdots * \mathrm{col}_i(M'_{f_n}), i \in [1, 2^n]$.

Sometimes, not all information is important and needs to be kept secret. For example, if in a given plaintext transmission, there exist some characters which are trivial, then it does not matter that the corresponding bits are either 1 or 0 in the key sequence. We call these bits unnecessary bits [28]. In general, there are two common strategies used to find these unnecessary bits: (1) specifying all unnecessary bits as 0 or all as 1; (2) specifying them to random values (0 or 1). Herein, we consider the power-analysis-based side-channel analysis, where the unnecessary bits are determined as a simple countermeasure of the attack. In order to increase the key search time for adversaries, an algorithm is proposed to determine the unnecessary bits. Thus, the importance of low-power optimization techniques for our daily life and social development is self-evident. Therefore, filter generators using the minimum number of stages should be designed.

Definition 2. A completely specified binary sequence $\mathcal{S} : s_0, s_1, \ldots$ is called periodic if there exists a positive integer r such that $s_{i+jr} = s_i$ for $i = 0, 1, \ldots$ and $j = 0, 1, \ldots$. The smallest positive integer r having this property is called the smallest period of $\mathcal{S}$. Note that the multiple of r, i.e., jr is also the period of sequence $\mathcal{S}$.

Definition 3. An incompletely specified binary sequence W: $w_1, w_2, \ldots,$ is called pseudo-periodic if there exists a positive integer l, assuming that the subscripts of all the specified bits in the first l bits are $\{\alpha_1, \ldots, \alpha_\lambda\}$, such that $w_{i+jl} = w_i, i = \alpha_1, \ldots, \alpha_\lambda, j = 1, 2, \ldots$, where both w_{i+jl} and w_i are determined bits. Moreover, the positive integer l, represents the smallest period of incompletely specified binary sequence W. Furthermore, jl also is the period of sequence W.

We assume that, for any given pseudo-periodic incompletely specified binary sequence, the value of every unnecessary bit corresponding to the same position is the same between any two periods. For example, there is an incompletely specified binary sequence W: 1, 0, z_1, 0, 1, 0, z_2, 0, 1, 0, z_3, 0. It can be found from Definition 3 that the incompletely specified binary sequence is pseudo-periodic and its minimum period is 4. Although, if we let $z_1 = z_2 = z_3 = 1$, then the incompletely specified binary sequence W, becomes a completely specified binary sequence denoted by S. Thus, we have that the sequence S is periodic and its minimum period is 2 by Definition 2.

The above assumption is reasonable because the number of state variables is finite, i.e., 2^n, where n is the number of stages in an FSR. This implies that the sequence generated by FSRs will be ultimately periodic. Hereafter, the period usually indicates the minimum period. The following lemma reveals the relation between the periods of sequences W and S.

Lemma 2. Assume that there is a pseudo-periodic incompletely specified binary sequence W with period l. Then, the incompletely specified binary sequence becomes a completely specified binary sequence denoted by S when the values of all unnecessary bits are determined. The period of completely specified binary sequence S, denoted by l' is a divisor of l.

Next, we consider the following two sequences having period l: $\mathbb{a} :\ a_0,\ a_1,\ a_2,\ a_3,\ a_4, \ldots, a_{l-1}, \ldots,$ and $\mathbb{b} :\ b_0,\ b_1,\ b_2,\ b_3,\ b_4, \ldots, b_{l-1}, \ldots$, where $a_i, i \in [r_1, r_2, \ldots, r_s], s \leqslant l$ are unnecessary bits. Note that the periods of these two sequences are not required to be the same. In fact, the period of $\mathbb{b}$ is a divisor of that of $\mathbb{a}$. For simplicity, we assume that the periods of these two sequences are the same. We then must design a filter generator to generate the above two sequences. In detail, sequences $\mathbb{a}$ and $\mathbb{b}$ are generated using one feedback shift register and an external circuit, respectively. We must solve the following two problems.

- Problem 1: for an incompletely specified binary sequence $\mathbb{a}$, determining unnecessary bits to increase key search time for adversaries.

- Problem 2: once the sequence $\mathbb{a}$ is specified, designing filter generators with the minimum number of stages to generate $\mathbb{b}$.

Table 1 All relations between PD_t and $\{\mathrm{HW}(x_1(t)\oplus x_1(t+1)),\ \mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}}),\ \mathrm{HW}(x_n(t+1)\oplus x_n(t+2)),\ \mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})\}$

PD_t	$\{\mathrm{HW}(x_1(t)\oplus x_1(t+1)),\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}}),\mathrm{HW}(x_n(t+1)\oplus x_n(t+2)),\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})\}$
-2	{0, 0, 1, 1}
-1	{1, 0, 1, 1}, {0, 1, 1, 1}, {0, 0, 1, 0}, {0, 0, 0, 1}
0	{1, 1, 1, 1}, {1, 0, 1, 0}, {0, 1, 1, 0}, {0, 1, 0, 1}, {0, 0, 0, 0}, {1, 0, 0, 1}
1	{1, 1, 1, 0}, {1, 1, 0, 1}, {1, 0, 0, 0}, {0, 1, 0, 0}
2	{1, 1, 0, 0}

We first solve problem 1. The overall power consumption of filter generator (7) at time t represented by P_t is given by

$$P_t=\sum_{i=1}^{n}P_{F_i}(t)+P_{y_1}(t), \tag{10}$$

where $\sum_{i=1}^{n}P_{F_i}(t)$ and $P_{y_1}(t)$ represent the power consumptions of the FSR stages and the output y_1 at time t, respectively. Note that the number of stages n is not determined here.

Let S_t denote the values of the filter generator (7) at time t, where $S_t=(X(t),y_1(t))$. Let HD_t be the Hamming distance between S_t and S_{t+1}. Let $\mathrm{HW}(S_t)$ denote the Hamming weight (i.e., the number of ones) of S_t. Thus, we have

$$\begin{aligned}\mathrm{HD}_t&=\mathrm{HW}(S_t\oplus S_{t+1})\\&=\mathrm{HW}((X(t),y_1(t))\oplus(X(t+1),y_1(t+1)))\\&=\mathrm{HW}(x_1(t)\oplus x_1(t+1),x_2(t)\oplus x_2(t+1),\ldots,x_n(t)\oplus x_n(t+1),y_1(t)\oplus y_1(t+1)),\\ \mathrm{HD}_{t+1}&=\mathrm{HW}(S_{t+1}\oplus S_{t+2})\\&=\mathrm{HW}((X(t+1),y_1(t+1))\oplus(X(t+2),y_1(t+2)))\\&=\mathrm{HW}(x_1(t+1)\oplus x_1(t+2),x_2(t+1)\oplus x_2(t+2),\ldots,\\&\quad x_n(t+1)\oplus x_n(t+2),y_1(t+1)\oplus y_1(t+2)).\end{aligned}$$

Based on the property of FSRs, the state transition relation is $(x_1,x_2,\ldots,x_n)\rightarrow(x_2,x_3,\ldots,f_n(x_1,x_2,\ldots,x_n))$. Then we have that $x_i(t)=x_{i-1}(t+1), i\in[2,n]$. Assume that $M_g=\delta_2[c_1,c_2,\ldots,c_{2^n}]$ and $X(t+i)\sim\delta_{2^n}^{\beta_{t+i}}$. Let PD_t be the theoretical power difference given by

$$\begin{aligned}\mathrm{PD}_t&=\mathrm{HD}_t-\mathrm{HD}_{t+1}\\&=\mathrm{HW}(x_1(t)\oplus x_1(t+1))-\mathrm{HW}(x_n(t+1)\oplus x_n(t+2))\\&\quad+\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}})-\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}}).\end{aligned}$$

Clearly, $\mathrm{HW}(x_1(t)\oplus x_1(t+1))$, $\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}})$, $\mathrm{HW}(x_n(t+1)\oplus x_n(t+2))$ and $\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})$ can take values from set $\mathcal{D}$. Let $\mathrm{PD}_t\sim\{\mathrm{HW}(x_1(t)\oplus x_1(t+1)),\ \mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}}),\ \mathrm{HW}(x_n(t+1)\oplus x_n(t+2)),\ \mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})\}$, and all possible combinations of $\mathrm{HW}(x_1(t)\oplus x_1(t+1))$, $\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}})$, $\mathrm{HW}(x_n(t+1)\oplus x_n(t+2))$ and $\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})$ are shown in Table 1.

From Table 1, PD_t can take any value from the set $\{-2,-1,0,1,2\}$. As shown by [34], power consumption P_t is proportional to HD_t, and PD_t is proportional to the difference of the measured power consumption at times t and $t+1$. As usual, the difference of the measured power consumption can be known, so the theoretical PD values can also be known. It can be found that if $\mathrm{PD}_t=-2$, then we immediately obtain that $\mathrm{HW}(x_1(t)\oplus x_1(t+1))=0$, $\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}})=0$, $\mathrm{HW}(x_n(t+1)\oplus x_n(t+2))=1$ and $\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})=1$. Similarly, when $\mathrm{PD}_t=2$, we have $\mathrm{HW}(x_1(t)\oplus x_1(t+1))=1$, $\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}})=1$, $\mathrm{HW}(x_n(t+1)\oplus x_n(t+2))=0$ and $\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})=0$. Therefore, when PD_t is either -2 or 2, the above four values can be uniquely determined. This implies that the attacker is susceptible to finding the initial state. Based on these two cases, the unnecessary bits in the incompletely specified binary sequence should specify some certain values to avoid these two cases.

Given $-2\sim\{0,\ 0,\ 1,\ 1\}$ and $2\sim\{1,\ 1,\ 0,\ 0\}$, we begin with $t=0$. Then, we find that $c_{\beta_t}=b_t$, implying that $\mathrm{HW}(c_{\beta_t}\oplus c_{\beta_{t+1}})$ and $\mathrm{HW}(c_{\beta_{t+1}}\oplus c_{\beta_{t+2}})$ can be determined. Because the number of stages cannot be determined, the value of $\mathrm{HW}(x_n(t+1) \oplus x_n(t+2))$ is still unknown until the problem 2 is solved. Let $\star$ represent the value of $\mathrm{HW}(x_n(t+1) \oplus x_n(t+2))$, that can be ignored temporarily. In order to reduce the times of P_t being equal to -2 or 2, i.e., the combinations $\{0, 0, \star, 1\}$ and $\{1, 1, \star, 0\}$, we should carefully choose the values of the unnecessary bits to reduce direct dependencies between the side channel information and key sequences. Thus, we increase the difficulty of adversaries' attack. To achieve this objective, Algorithm 1 is provided to determine the values of unnecessary bits in sequence $\mathbb{a}$.

Algorithm 1 Determine the values of unnecessary bits in sequence $\mathbb{a}$

```
 1: for $t = 0$ to $l - 1$ do
 2:     if both of $a_t$ and $a_{t+1}$ are unnecessary bits or one of them is unnecessary bit then
 3:         if $(b_t \oplus b_{t+1}) = 0$ and $(b_{t+1} \oplus b_{t+2}) = 1$ then
 4:             Let $a_t$ and $a_{t+1}$ satisfy $a_t \oplus a_{t+1} = 0$;
 5:         else
 6:             if $(b_t \oplus b_{t+1}) = 1$ and $(b_{t+1} \oplus b_{t+2}) = 0$ then
 7:                 Let $a_t$ and $a_{t+1}$ satisfy $a_t \oplus a_{t+1} = 1$;
 8:             end if
 9:         end if
10:     else
11:         $t = t + 1$.
12:     end if
13: end for
```

By using Algorithm 1, the values of all unnecessary bits in $\mathbb{a}$ can be determined. We assume that the incompletely specified binary sequence $\mathbb{a}$ becomes a completely specified binary sequence $\mathbb{c}$ given by $\mathbb{c} : c_0, c_1, c_2, \ldots$ with period l'. Next, we analyze problem 2. To obtain the minimum number of stages for generating sequence $\mathbb{c}$, we must carefully analyze the properties of FSRs. For a given state $(x_1(t), x_2(t), \ldots, x_n(t))$, the next state can be represented by $(x_2(t), x_3(t), \ldots, f_n(x_1(t), x_2(t), \ldots, x_n(t)))$, which is also called the successor to the state $(x_1(t), x_2(t), \ldots, x_n(t))$. The output of FSRs is always the value of the first stage, i.e., $x_1(t)$. Moreover, it can be found that the period of output sequences is equivalent to that of state sequences [35].

Based on the property of FSRs, the following result can be obtained.

Lemma 3. For a completely specified binary sequence $\mathbb{c}$ having period l': $c_0, c_1, \ldots, c_{l'-1}, \ldots$, if the number of stages for an FSR is k, then the trajectory with an initial state $(a_0,\ a_1, \ldots, a_{k-1})$ of the FSR can be determined.

For sequence $\mathbb{c}$, if we know that the number of stages for the FSR generating the sequence $\mathbb{c}$ is r, $r \leqslant l'$, then one of the trajectories for the FSR is $(c_0, c_1, \ldots, c_{r-1}) \to (c_1, c_2, \ldots, c_r) \to \cdots \to (c_{l'-1}, c_{l'}, \ldots, c_{l'+r-2}) \to (c_0, c_1, \ldots, c_{r-1}) \to \cdots$, and others cannot be ensured. For an FSR having r stages, there totally exist 2^r nodes in the state-transition graph. However, we can only ensure the successor of $(c_i, c_{1+i}, \ldots, c_{r-1+i}), i \in [0, l'-1]$. Therefore, the successors of the remaining $2^r - l'$ in the state-transition graph are not determined. Let $\mathcal{G}(\mathbb{c}, r)$ denote this state-transition graph. Moreover, we can also conclude from $\mathcal{G}(\mathbb{c}, r)$ that there exists more than one FSR with r stages generating the output sequence $\mathbb{c}$, because the successors of the rest $2^r - l'$ can be random.

For example, consider the following binary sequence having period 11: $1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, \ldots$. If we know that the above sequence was generated by an FSR having 4 stages, then based on Lemma 3, one of the state trajectories for the feedback shift register must be $(1, 1, 0, 1) \to (1, 0, 1, 1) \to (0, 1, 1, 1) \to (1, 1, 1, 0) \to (1, 1, 0, 0) \to (1, 0, 0, 1) \to (0, 0, 1, 0) \to (0, 1, 0, 0) \to (1, 0, 0, 1) \to (0, 0, 1, 1) \to (0, 1, 1, 1) \to (1, 1, 1, 0) \to (1, 1, 0, 1)$. However, there still exist four states which are not included in the above trajectory. If we have no idea about the number of stages in the FSR, then it is hard to confirm any state trajectory that may comprise triples or quintuples. Thus, it is a challenge for us to confirm the number of stages for generating the given periodic sequence, and the number must be minimized.

If the FSR (7) is regarded as a BN having n nodes, then by [36], the generating maximum period sequence is 2^n. Thus, for given periodic sequence $\mathbb{c}$ with period l', the minimum number of stages denoted by $N_{\min}$ satisfies the following inequality:

$$N_{\min} \geqslant \log_2 l'. \tag{11}$$

According to (11), for the sequence $\mathbb{c}$, we begin to consider from $n = \lceil \log_2 l' \rceil$. If there exists a node whose out-degree is greater than 1 in $\mathcal{G}(\mathbb{c}, \lceil \log_2 l' \rceil)$, then the number of stages must be greater than $\lceil \log_2 l' \rceil$. For any given state, the next state must be uniquely determined. This fact implies that it is impossible to have a node whose out-degree is greater than 1 in $\mathcal{G}(\mathbb{c}, \lceil \log_2 l' \rceil)$. Therefore, an FSR having $\lceil \log_2 l' \rceil$ stages cannot generate sequence $\mathbb{c}$.

Considering the case $n = (\lceil \log_2 l' \rceil) + r, r \in \mathcal{N}$, the trajectory is $(c_0, c_1, \ldots, c_{(\lceil \log_2 l' \rceil)+r-1}) \rightarrow (c_1, c_2, \ldots, c_{(\lceil \log_2 l' \rceil)+r}) \rightarrow \cdots \rightarrow (c_{l'-1}, c_{l'}, \ldots, c_{l'+(\lceil \log_2 l' \rceil)+r-2})$. Because the sequence $\mathbb{c}$ is periodic and its period is l', then it indicates that $c_{jl'+i} = c_i, i \in [0, l'-1], j \geqslant 1$. If the out-degree of every node in $\mathcal{G}(\mathbb{c}, \lceil \log_2 l' \rceil + 1)$ is 1, then it implies that the given sequence $\mathbb{a}$ can be generated using an FSR with $(\lceil \log_2 l' \rceil) + r$ stages.

Based on the above analysis, Algorithm 2 shows the process how to find the minimum number of stages for the FSR generating the given sequence $\mathbb{c}$.

Algorithm 2 Determine the minimum number of stages for the FSR generating completely specified binary sequence $\mathbb{c}$

```
1. Consider a given sequence $\mathbb{c}$ having period $l'$: $c_0, c_1, \ldots, c_{l'-1} \ldots$, and set $n = \lceil \log_2 l' \rceil$;
2. Obtain the trajectory with initial state $(c_0, \ldots, c_{n-1})$;
3. If there exists one node in the state-transition graph $\mathcal{G}(\mathbb{c}, n)$ such that its out-degree is greater than 1, go to the next
   step. However, if the output degree of every node is equal to 1, then the minimum number of stages generating $\mathbb{c}$ is
   $n$;
4. Let $n = n + 1$. Then, repeat steps 2 and 3.
```

Meanwhile, we can obtain the following theorem about the minimum number of stages.

Theorem 1. For a completely specified binary sequence $\mathbb{c}$ having a period l': $c_0, c_1, \ldots, c_{l'-1} \ldots$, if there exists a minimum integer k ($k \geqslant \lceil \log_2 l' \rceil$) such that the out-degree of all the determined l' nodes is 1 in $\mathcal{G}(\mathbb{c}, k)$, then the minimum number of stages for FSRs generating the sequence $\mathbb{c}$ is k.

Proof. Because the out-degree of all the determined l' nodes is 1, the successors to these l' nodes are uniquely determined. Therefore, such an FSR having k stages generating the output sequence $\mathbb{c}$ exists. Inversely, if there exists one node owning two out-degrees denoted by $(c_i, c_{i+1}, \ldots, c_{i+k-1}) \rightarrow (c_{i+1}, c_{i+2}, \ldots, c_{i+k-1}, 1)$ and $(c_i, c_{i+1}, \ldots, c_{i+k-1}) \rightarrow (c_{i+1}, c_{i+2}, \ldots, c_{i+k-1}, 0)$, then it implies that the feedback function should satisfy $f_n(c_i, c_{i+1}, \ldots, c_{i+k-1}) = 1$ and $f_n(c_i, c_{i+1}, \ldots, c_{i+k-1}) = 0$, which is obviously contradictory.

If the number of stages is determined to be k, then the trajectory beginning with the initial state $(c_0, c_1, \ldots, c_{k-1})$ is $(c_0, c_1, \ldots, c_{k-1}) \rightarrow (c_1, c_2, \ldots, c_k) \rightarrow \cdots \rightarrow (c_{l'-1}, c_{l'}, \ldots, c_{l'+k-1})$, indicating that the sequence of the FSR having the initial state $(c_0, c_1, \ldots, c_{k-1})$ must be $\mathbb{c}$. Therefore, the minimum number of stages for FSRs generating sequence $\mathbb{c}$ is k.

Remark 1. Compared with [29], by resorting to the method of BNs, the lower bound of the minimum number of stages can be obtained, which reduces the exhaustive search time. Moreover, the trajectory of a given initial state is reflected on the state-transition graph, for which, the minimum number of stages is found, and the relationship between some columns of L' is subsequently revealed.

Suppose that we have obtained the minimum number of k stages for FSRs generating the completely specified binary sequence $\mathbb{c}$ by Theorem 1. Then, we must construct these FSRs. We further assume that $(c_0, c_1, \ldots, c_{k-1}) \sim \delta_{2^k}^{i_0}, (c_1, c_2, \ldots, c_k) \sim \delta_{2^k}^{i_1}, (c_2, c_3, \ldots, c_{k+1}) \sim \delta_{2^k}^{i_2}, \ldots$, and $(c_{l'-1}, c_{l'}, \ldots, c_{l'+k-2}) \sim \delta_{2^k}^{i_{l'-1}}$. Consider system (9) and let $n = k$. Then, we have that $\mathrm{col}_{i_0}(L') = \delta_{2^k}^{i_1}, \mathrm{col}_{i_1}(L') = \delta_{2^k}^{i_2}, \ldots, \mathrm{col}_{i_{l'-2}}(L') = \delta_{2^k}^{i_{l'-1}}$, and $\mathrm{col}_{i_{l'-1}}(L') = \delta_{2^k}^{i_0}$. Thus, L' has the following form:

$$L' = \delta_{2^k}\left[\diamond \cdots \diamond\ \underset{i_0}{i_1}\ \diamond \cdots \diamond\ \underset{i_1}{i_2}\ \diamond \cdots \diamond\ \underset{i_{l'-2}}{i_{l'-1}}\ \diamond \cdots \diamond\ \underset{i_{l'-1}}{i_0}\right],$$

where the notation $\diamond$ represents undetermined elements. It can be seen that there exist a total of $2^{(2^k-l')k}$ matrices L'. Next, we determine matrices satisfying the corresponding updated functions of the forms of $f_i = x_{i+1}(t), i \in [1, n-1]$ and $f_n = f_n(x_1, x_2, \ldots, x_n)$. Fortunately, Zhao et al. [4] has solved this problem and reached the following result:

$$\mathrm{col}_i(L') = \begin{cases} \delta_{2^k}^{2i-1} \text{ or } \delta_{2^k}^{2i}, \quad 1 \leqslant i \leqslant 2^{k-1}, \\ \delta_{2^k}^{2(i-2^{k-1})-1} \text{ or } \delta_{2^k}^{2(i-2^{k-1})}, \quad 2^{k-1}+1 \leqslant i \leqslant 2^k. \end{cases} \tag{12}$$

Therefore, based on (12), we can conclude that each column of L' has two possible values except for the known l' columns, implying that there exist $2^{2^k-l'}$ FSRs generating the given output sequence $\mathbb{c}$.

In the following, we must construct the external circuit c_e using a logical function $g(x_1, x_2, \ldots, x_k)$ to generate the given binary sequence $\mathbb{b}$ having period l', where l' is a divisor of l. Assuming that the structure matrix of $g(x_1, x_2, \ldots, x_k)$ is denoted by M_g, then it implies that we only need to construct matrix M_g.

As discussed above, when the minimum number of stages is k, the trajectory of the FSRs is $(c_0, c_1, \ldots, c_{k-1}) \rightarrow (c_1, c_2, \ldots, c_k) \rightarrow \cdots \rightarrow (c_{l'-1}, c_{l'}, \ldots, c_{l'+k-2}) \rightarrow (c_0, c_1, \ldots, c_{k-1}) \rightarrow \cdots$. Then, consider system (7), and $\mathrm{col}_{i_0}(M_g) = \delta_2^{(2-b_0)}, \mathrm{col}_{i_1}(M_g) = \delta_2^{(2-b_1)}, \ldots,$ and $\mathrm{col}_{i_{l'-1}}(M_g) = \delta_2^{(2-b_{l'-1})}$. Therefore, M_g takes the form of

$$M_g = \delta_2 \left[\star \cdots \star \underset{i_0}{(2-b_0)} \star \cdots \star \underset{i_1}{(2-b_1)} \star \cdots \star \underset{i_{l'-1}}{(2-b_{l'-1})} \right], \tag{13}$$

where the notation $\star$ represents undetermined elements. It follows from (12) that the columns of L' have some limitations because the FSRs should satisfy (6). However, there is no limitation on the columns of M_g except the determined l' columns. Therefore, the notation $\star$ can take value randomly from $\{1, 2\}$, implying that there exist a total of $2^{2^k-l'}$ matrices M_g, generating the binary sequence $\mathbb{b}$ having a period l'.

As usual, the objective of cryptanalytic attacks is to ensure the unknown initial state by providing a long enough segment of its keystream sequence. Therefore, to further enhance the exhaustive search time, the choice of function g is particularly important such that the generated output gets independent random bits.

Definition 4. The function is said to be balanced if the probability of the function output being 0 or 1 is 1/2.

For any given state $\mathcal{S} = (s_1, s_2, \ldots, s_k)$, its conjugate state is defined as $\mathcal{S}^* = (\bar{s}_1, s_2, \ldots, s_k)$ with $\bar{s}_1 = 1 \oplus s_1$, where $\oplus$ means XOR. Clearly, we have $|\mathcal{D}(\mathcal{S}) - \mathcal{D}(\mathcal{S}^*)| = 2^{k-1}$.

Lemma 4. Consider system (9). The function g is balanced if for any given state $\mathcal{S}$ and its conjugate state $\mathcal{S}^*$, the $\mathcal{D}(\mathcal{S})$-th column and $\mathcal{D}(\mathcal{S}^*)$-th column of M_g are different.

It can be found from (13) that some columns of M_g have been restricted, implying that it is tough to guarantee M_g while satisfying Lemma 4. To increase the key search time for adversaries, we make some modifications to M_g based on (13) and Lemma 4, shown in Algorithm 3.

Algorithm 3 Determine M_g to increase the key search time

```
1: for $j = 1$ to $2^{k-1}$ do
2:     if both of $\mathrm{col}_j(M_g)$ and $\mathrm{col}_{j+2^{k-1}}(M_g)$ are not determined or one of them is not determined then
3:         Let $\mathrm{col}_j(M_g)$ and $\mathrm{col}_{j+2^{k-1}}(M_g)$ take different values;
4:     end if
5:     $j = j + 1$.
6: end for
```

The aim of Algorithm 3 is to make the probability of the output being 0 or 1 closer to $\frac{1}{2}$. It can be found that both Algorithms 1 and 3 play important roles of increasing the key search time for adversaries. Actually, most of algorithms generating key sequences are not be guaranteed to be absolutely secure.

Clearly, from the security point of view, it is not preferable that the adversaries cannot easily find the initial state. Therefore, in order to improve the probability of regulators finding vulnerabilities, we try to increase the key search time for adversaries by utilizing Algorithms 1 and 3. Based on these two algorithms, the transition matrix L' and structure matrix M_g can be determined. Cheng et al. [7] showed how to convert the algebraic form (9) back to logical form (7). Thus, referring to [7], the corresponding filter generator can be constructed.

Example 2. We construct a filter generator which generates the following sequences with period 11:

$$\begin{aligned}&\mathbb{a}: 1,\ z_1,\ z_2, 0,\ 0,\ z_3,\ 1,\ 1,\ z_4,\ 0,\ 1, \ldots\\ &\mathbb{b}: 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 0, \ldots.\end{aligned}$$

First, we confirm the values of these unnecessary bits in sequence $\mathbb{a}$ using Algorithm 1. Because a_1 is one unnecessary bit and $b_0 \oplus b_1 = 0$ as well as $b_1 \oplus b_2 = 1$, we can conclude that $a_0 \oplus a_1 = 0$, implying that $a_1 = 1$. Similarly, we have that $a_2,\ a_5$ and a_8 are unnecessary bits. Moreover,

$$\begin{aligned}b_1 \oplus b_2 = 1 \quad &\text{and} \quad b_2 \oplus b_3 = 0,\\ b_4 \oplus b_5 = 0 \quad &\text{and} \quad b_5 \oplus b_6 = 1,\\ b_7 \oplus b_8 = 1 \quad &\text{and} \quad b_8 \oplus b_9 = 0.\end{aligned}$$

Using Algorithm 1, $a_2,\ a_5$ and a_8 can be determined and are all equal to 0. Therefore, the sequence $\mathbb{a}$ becomes a completely specified binary sequence denoted by $\mathbb{c}$, that is, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, ...

Then, we must determine the minimum number of stages to generate the sequence $\mathbb{c}$ by Algorithm 2. We consider the case that $n = \lceil \log_2 11 \rceil = 4$, and then it can be observed from Figure 3 that the out-degrees of nodes $(1, 1, 0, 0)$ and $(0, 0, 1, 1)$ are 2. Therefore, the sequence $\mathbb{c}$ can also not be generated by FSRs with stages being 4. When $n = 5$, it can be learned from Figure 4 that the out-degree of every node is equal to 1. Therefore, the minimum number of stages for FSRs generating the sequence $\mathbb{c}$ is 5 by Theorem 1.

Obviously, $(1,\ 1,\ 0,\ 0,\ 0) \sim \delta_{32}^{8}$, $(1,\ 0,\ 0,\ 0,\ 0) \sim \delta_{32}^{16}$, $(0,\ 0,\ 0,\ 0,\ 1) \sim \delta_{32}^{31}$, $(0,\ 0,\ 0,\ 1,\ 1) \sim \delta_{32}^{29}$, $(0,\ 0,\ 1,\ 1,\ 0) \sim \delta_{32}^{26}$, $(0,\ 1,\ 1,\ 0,\ 0) \sim \delta_{32}^{20}$, $(1,\ 1,\ 0,\ 0,\ 1) \sim \delta_{32}^{7}$, $(1,\ 0,\ 0,\ 1,\ 1) \sim \delta_{32}^{13}$, $(0,\ 0,\ 1,\ 1,\ 1) \sim \delta_{32}^{25}$, $(0,\ 1,\ 1,\ 1,\ 0) \sim \delta_{32}^{18}$ and $(1,\ 1,\ 1,\ 0,\ 0) \sim \delta_{32}^{4}$. Thus, L' takes the following form:

$$L' = \delta_{32}\left[\diamond\diamond\diamond\underset{4}{8}\diamond\diamond\underset{7}{13}\,\underset{8}{16}\diamond\diamond\diamond\diamond\underset{13}{25}\diamond\diamond\underset{16}{31}\diamond\underset{18}{4}\diamond\underset{20}{7}\diamond\diamond\diamond\diamond\underset{25}{18}\,\underset{26}{20}\diamond\diamond\underset{29}{26}\diamond\underset{31}{29}\diamond\right], \tag{14}$$

and the rest columns (represented by $\diamond$) of L' satisfy (12). According to (13), M_g is in the form of

$$M_g = \delta_{32}\left[\star\star\star\underset{4}{2}\star\star\underset{7}{2}\,\underset{8}{1}\star\star\star\star\underset{13}{2}\star\star\underset{16}{1}\star\underset{18}{1}\star\underset{20}{1}\star\star\star\star\underset{25}{1}\,\underset{26}{1}\star\star\underset{29}{2}\star\underset{31}{2}\star\right]. \tag{15}$$

The remaining columns (represented by $\star$) of M_g can randomly take values from $\{1, 2\}$.

Considering the property of balanced functions and in order to increase the key search time, then by Algorithm 3, structure matrix M_g can take the following form:

$$M_g = \delta_{32}[1\ 2\ 2\ 2\ 1\ 2\ 2\ 1\ 2\ 2\ 2\ 1\ 2\ 2\ 1\ 1\ 2\ 1\ 1\ 1\ 2\ 1\ 1\ 2\ 1\ 1\ 1\ 2\ 2\ 1\ 2\ 2].$$

According to (12), the transition matrix L' can be

$$L' = \delta_{32}[1\ 4\ 5\ 8\ 9\ 12\ 13\ 16\ 18\ 20\ 22\ 24\ 25\ 27\ 29\ 31\ 1\ 4\ 5\ 7\ 9\ 11\ 14\ 16\ 18\ 20\ 22\ 24\ 26\ 27\ 29\ 32].$$

Therefore, referring to [7], the corresponding logical forms can be obtained; i.e., the filter generator,

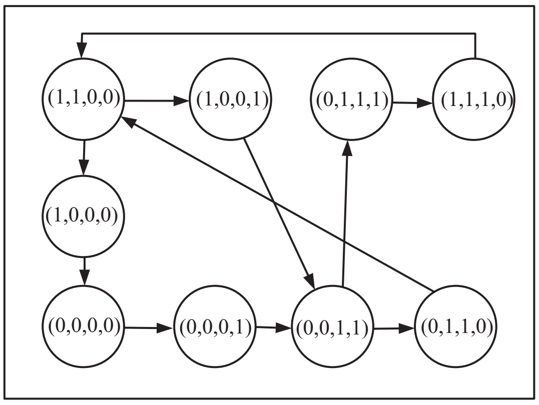

Figure 3 The state-transition graph $\mathcal{G}(\mathbb{c}, 4)$.

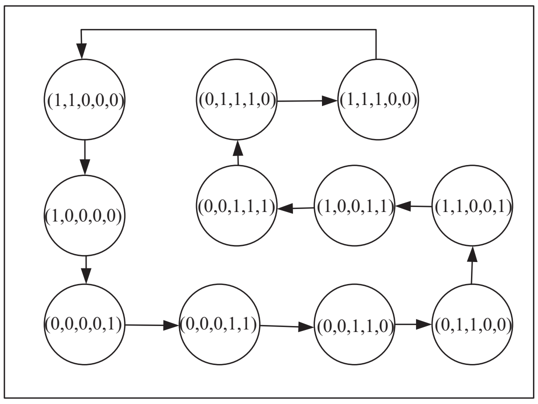

Figure 4 The state-transition graph $\mathcal{G}(\mathbb{c}, 5)$.

generating sequences $\mathbb{a}$ and $\mathbb{b}$, can be constructed as follows:

$$\left\{\begin{aligned}
x_1(t+1) =& x_2(t),\\
x_2(t+1) =& x_3(t),\\
x_3(t+1) =& x_4(t),\\
x_4(t+1) =& x_5(t),\\
x_5(t+1) =& (x_1(t) \wedge ((x_2(t) \wedge x_5(t)) \vee \neg(x_2(t) \vee x_3(t))))\\
& \vee (\neg x_1(t) \wedge ((x_2(t) \wedge ((x_3(t) \wedge (x_4 \rightarrow x_5))\\
& \vee (\neg x_3 \wedge x_5))) \vee (\neg x_2(t) \wedge (\neg x_3(t) \wedge (x_4(t) \bar{\vee} x_5(t)))))),\\
y_1(t) =& (x_1(t) \wedge ((x_2(t) \wedge ((x_3(t) \wedge x_4(t) \wedge x_5(t))\\
& \vee (\neg x_3(t) \wedge (x_4(t) \leftrightarrow x_5(t))))) \vee (\neg x_2(t) \wedge ((x_3(t)\\
& \wedge \neg(x_4(t) \vee x_5(t))) \vee \neg(x_3(t) \wedge x_4(t)))))) \vee (\neg x_1(t)\\
& \wedge ((x_2(t) \wedge ((x_3(t) \wedge \neg(x_4(t) \wedge x_5(t))) \vee (\neg x_3(t)\\
& \wedge (x_4(t) \bar{\vee} x_5(t))))) \vee (\neg x_2(t) \wedge ((x_3(t) \wedge x_4(t)\\
& \wedge x_5(t)) \vee \neg(x_3(t) \vee (x_4(t) \rightarrow x_5(t))))))).
\end{aligned}\right.$$

4 Conclusion

In this paper, we regarded a filter generator as a BN. By utilizing the STP, the corresponding algebraic form was obtained. Based on the algebraic form, algorithms were presented to construct filter generators to improve security and reduce area and power dissipation for a given incompletely specified binary sequence. Finally, one example was given to illustrate the effectiveness of our results.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61973078), Natural Science Foundation of Jiangsu Province (Grant No. BK20170019), Jiangsu Provincial Key Laboratory of Networked Collective Intelligence (Grant No. BM2017002), Jiangsu Province Six Talent Peaks Project (Grant No. 2015-ZNDW-002), Fundamental Research Funds for the Central Universities (Grant No. 2242019k1G013), and Postgraduate Research & Practice Innovation Program of Jiangsu Province (Grant No. KYCX19_0111).

References

1 Wang J, Mu J Q, Wei S Q, et al. Statistical characterization of decryption errors in block-ciphered systems. IEEE Trans Commun, 2015, 63: 4363–4376

2 Zhong J H, Lin D D. Driven stability of nonlinear feedback shift registers with inputs. IEEE Trans Commun, 2016, 64: 2274–2284

3 Zhong J H, Lin D D. Decomposition of nonlinear feedback shift registers based on Boolean networks. Sci China Inf Sci, 2019, 62: 039110

4 Zhao D W, Peng H P, Li L X, et al. Novel way to research nonlinear feedback shift register. Sci China Inf Sci, 2014, 57: 092114

5 Kauffman S A. Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol, 1969, 22: 437–467

6 Zhang Y, Liu Y. Nonlinear second-order multi-agent systems subject to antagonistic interactions without velocity constraints. Appl Math Comput, 2020, 364: 124667

7 Cheng D Z, Qi H S, Li Z Q. Analysis and Control of Boolean Networks: A Semi-tensor Product Approach. London: Springer, 2011

8 Zhong J, Liu Y, Kou K I, et al. On the ensemble controllability of Boolean control networks using STP method. Appl Math Comput, 2019, 358: 51–62

9 Lin L, Cao J D, Rutkowski L. Robust event-triggered control invariance of probabilistic Boolean control networks. IEEE Trans Neural Netw Learn Syst, 2020, 31: 1060–1065

10 Huang C, Lu J Q, Ho D W C, et al. Stabilization of probabilistic Boolean networks via pinning control strategy. Inf Sci, 2020, 510: 205–217

11 Zhu S Y, Lu J Q, Liu Y. Asymptotical stability of probabilistic Boolean networks with state delays. IEEE Trans Automat Contr, 2020, 65: 1779–1784

12 Zhong J, Li B W, Liu Y, et al. Output feedback stabilizer design of Boolean networks based on network structure. Front Inform Technol Electron Eng, 2020, 21: 247–259

13 Xu M X, Liu Y, Lou J G, et al. Set stabilization of probabilistic Boolean control networks: a sampled-data control approach. IEEE Trans Cybern, 2019. doi: 10.1109/TCYB.2019.2940654

14 Zhu S Y, Liu Y, Lou Y J, et al. Stabilization of logical control networks: an event-triggered control approach. Sci China Inf Sci, 2020, 63: 112203

15 Lu J Q, Sun L J, Liu Y, et al. Stabilization of Boolean control networks under aperiodic sampled-data control. SIAM J Control Opt, 2018, 56: 4385–4404

16 Liu Y, Li B W, Chen H W, et al. Function perturbations on singular Boolean networks. Automatica, 2017, 84: 36–42

17 Liu H C, Liu Y, Li Y Y, et al. Observability of Boolean networks via STP and graph methods. IET Control Theor Appl, 2018, 13: 1031–1037

18 Liu Y, Li B W, Lu J Q, et al. Pinning control for the disturbance decoupling problem of Boolean networks. IEEE Trans Automat Contr, 2017, 62: 6595–6601

19 Li Y Y, Liu R J, Lou J G, et al. Output tracking of Boolean control networks driven by constant reference signal. IEEE Access, 2019, 7: 112572

20 Wu Y H, Shen T L. A finite convergence criterion for the discounted optimal control of stochastic logical networks. IEEE Trans Automat Contr, 2018, 63: 262–268

21 Li Y L, Li H T, Xu X J, et al. Semi-tensor product approach to minimal-agent consensus control of networked evolutionary games. IET Control Theor Appl, 2018, 246: 2269–2275

22 Li H T, Wang Y Z. Boolean derivative calculation with application to fault detection of combinational circuits via the semi-tensor product method. Automatica, 2012, 48: 688–693

23 Lu J Q, Li M L, Huang T W, et al. The transformation between the Galois NLFSRs and the Fibonacci NLFSRs via semi-tensor product of matrices. Automatica, 2018, 96: 393–397

24 Dubrova E. On constructing secure and hardware-efficient invertible mappings. In: Proceedings of International Symposium on Multiple-Valued Logic, Sapporo, 2016. 211–216

25 Liu Z B, Wang Y Z, Cheng D Z. Nonsingularity of feedback shift registers. Automatica, 2015, 55: 247–253

26 Lu J Q, Li M L, Liu Y, et al. Nonsingularity of Grain-like cascade FSRs via semi-tensor product. Sci China Inf Sci, 2018, 61: 010204

27 Zhong J H, Lin D D. On minimum period of nonlinear feedback shift registers in Grain-like structure. IEEE Trans Inform Theor, 2018, 64: 6429–6442

28 Li N, Dubrova E. Synthesis of power- and area-efficient binary machines for incompletely specified sequences. In: Proceedings of Asia and South Pacific Design Automation Conference, Singapore, 2014. 634–639

29 Wan Z, Dai Z, Liu M, et al. Nonlinear Feedback Shift Registers (in Chinese). Beijing: Science Press, 1978

30 Zadeh A A, Heys H M. Simple power analysis applied to nonlinear feedback shift registers. IET Inform Secur, 2014, 3: 188–198

31 Dubrova E. Synthesis of binary machines. IEEE Trans Inform Theor, 2011, 57: 6890–6893

32 Dubrova E. Synthesis of parallel binary machines. In: Proceedings of the International Conference on Computer-Aided Design, San Jose, 2011. 200–206

33 Veliz-Cuba A. Reduction of Boolean network models. J Theor Biol, 2011, 289: 167–172

34 Burman S, Mukhopadhyay D, Veezhinathan K. LFSR based stream ciphers are vulnerable to power attacks. In: Proceedings of International Conference on Cryptology, 2007. 384–392

35 Goresky M, Klapper A. Algebraic Shift Register Sequences. Cambridge: Cambridge University Press, 2012

36 Li R, Yang M, Chu T G. State feedback stabilization for Boolean control networks. IEEE Trans Automat Contr, 2013, 58: 1853–1857