Hybrid FCSR Based Stream Cipher for Secure Communications in IoT

Shyi-Tsong Wu[*]

Department of Electronic Engineering, National Ilan University, Taiwan
stwu@niu.edu.tw

Abstract

Linear Feedback Shift Register (LFSR) is the basic hardware of stream cipher, and Feedback with Carry Shift Register (FCSR) is the nonlinear analogues of LFSR. FCSR is a feedback architecture to generate long pseudorandom sequence. In this paper, we study the characteristics of FCSRs combined with nonlinear circuits such as Dawson's Summation Generator (DSG), l_p-Geffe generator and etc. Then we proposed a hybrid FCSR applying DSG and l_p-Geffe generator as nonlinear combining elements to increase the period and the linear complexity of the output sequence. In addition, we further investigate the period, linear complexity, randomness, and use known attacks to verify the security strength of the proposed keystream generator. The pass rates of the proposed scheme are 100% for FIPS PUB 140-1 random tests, and at least 98% for SP800-22 random test, respectively.

Keywords: Hybrid FCSR, Hardware security, IoT, Stream cipher

1 Introduction

The seurity of Internet of Things (IoT) is an important issue. However, under the resource-constrained IoT, a complex system becomes a heavy load and leads to a reduction in communications efficiency [1-3]. Because the communication of the IoT is in real time, a stream cipher with the characteristics of simplicity and high speed is suitable to the real-time communications of IoT for its efficiency [4-8].

The security of a stream cipher depends on the pseudorandom sequences of the keystream generator. How to generate secure pseudorandom sequences efficiently is a hot topic in cryptology. The good pseudorandom sequences should have good statistical distributions, large period, high linear complexity, and good randomness [9-12].

Klapper and Goresky proposed Feedback with Carry Shift Registers (FCSR) [13]. The FCSR is a new feedback architecture to generate long pseudorandom sequence efficiently [14]. It is very fast and easy to implement in both software and hardware and with the property of nonlinear [15-16]. There are few papers describe or analyze the properties of FCSR application circuits related with their period, linear complexity and etc.

In this paper, we study the characteristics of FCSRs merged by nonlinear circuits at first. Then we apply FCSRs and cascaded nonlinear circuits as a keystream generator. To increase the period and linear complexity of sequences, we use Dawson's summation generator (DSG) and l_p-Geffe generator as nonlinear combining elements to produce keystream. From the experimental results, we investigate their period, linear complexity, randomness. The proposed scheme can resist known attacks. The statistical tests of Federal Information Processing Standards Publication 140-1 (FIPS PUB 140-1) and the Special Publication 800-22 (SP800-22) are performed on the proposed scheme. The pass rates are 100% for FIPS PUB 140-1 random tests, and at least 98% for SP800-22 random test, respectively. The main contributions of this study can be outlined as follows:

- The proposed stream cipher employs hybrid FCSR is suitable to resource-contrained environment of IoT.
- The non-linear selection and output combining functions in our proposed scheme ensure that the correlation probability is well-balanced, making it difficult for attackers to exploit any weaknesses and compromise the security of the stream cipher.
- The proposed scheme based on a hybrid FCSR has excellent statistical distribution, a long period, high linear complexity, and produces highly random outputs.
- The proposed scheme is a hardware-based security solution that can be easily implemented using hardware components.

The organization of this paper is as follows. Section 2 introduces the FCSR. In section 3, we describe our proposed scheme and detail of the design circuit. In section 4, we introduce statistical properties and some attacks with respect to our design. We present the period and linear complexity of the proposed generator with large parameters. Section 5 describes the experimental results for the proposed stream cipher. Finally, we give the conclusions of proposed scheme in Section 6.

2 Preliminaries

In this section, we introduce the properties of FCSR, two basic FCSRs combiners XOR function and the l_p-Geffe generator based on FCSRs. Besides we will present the properties with their period and linear complexity.

2.1 Feedback with Carry Shift Register (FCSR)

FCSR is similar to the LFSR, such as structure, characteristics, and it is a new feedback architecture to

*Corresponding Author: Shyi-Tsong Wu; E-mail: stwu@niu.edu.tw
DOI: 10.53106/160792642023112406010

efficiently generate long pseudorandom sequence [14, 17]. FCSR has a small amount of memory and the analysis of FCSR is based on the arithmetic of 2-adic numbers. It is very fast and easy to implement in both software and hardware. There are two basic structure of FCSR, i.e., Fibonacci architecture of FCSR and Galois architecture of FCSR [18-19].

The Fibonacci architecture of an r-stage FCSR is depicted in Figure 1, where $s_{n-1}, s_{n-2}, \ldots, s_{n-r} \in \{0,1\}$ denoted the cell contents, m_{n-1} denoted the current memory contents, and Σ denoted integer addition. The feedback coefficient $(q_1, q_2, \ldots, q_r) \in \{0,1\}$ represent the existence or absence of a feedback tap [14-15].

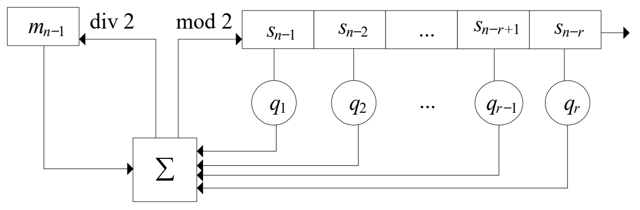


Figure 1. Fibonacci architecture of FCSR

The register operations are as follows [14-15]:

1. Calculate ${}_{n} = \sum_{i=1}^{r} q_i s_{n-i} + m_{n-1}$ as an integer sum.
2. Shift the cell contents to the right, the output bit is rightmost bit s_{n-r}.
3. Return $s_n = \sigma_n$ (mod 2) into the leftmost cell s_{n-1} of the shift register.
4. Substitute m_{n-1} by $m_n = \lfloor \sigma_n/2 \rfloor$.

The Galois architecture of FCSR is illustrated in Figure 2. The feedback coefficient $(q_1, q_2, \ldots, q_r)$ of an r-stage FCSR correspond to the binary expansion of q.

$$q = q_r 2^r + q_{r-1} 2^{r-1} + \ldots + q_1 2 - 1, \quad \textbf{(1)}$$

the $c_1, c_2, \ldots, c_{r-1}$ are the memory bits (or carry), and the Σ represents a full adder. Where $q_i \in \{0, 1\}$, $q_r = 1$. The integer q is the connection integer and it is analog the connection polynomial of LFSR [14].

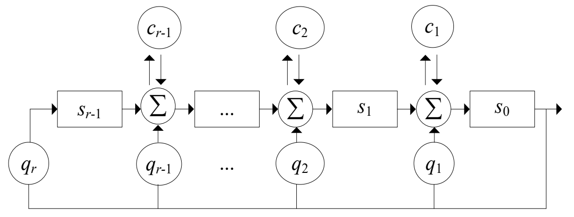


Figure 2. Galois architecture of FCSR

At the j-th adder, the following input bits are received:

- s_j from the preceding cell
- $s_0 q_j$ from the feedback line
- c_j from the memory cell

which are added to form a sum σ_j (with $1 \le j \le r - 1$). At the next clock cycle, this sum σ_j modulo 2 is passed on to the next cell in the register,

$s_{j-1}' = \sigma_j \bmod 2,$

and the sum σ_j div 2 is passed on to the memory,

$c_j' = \sigma_j$ div 2.

The connection integer q determines the period of the sequences generated by an FCSR. We usually choose the connection integer according to the following criterions [24-25]:

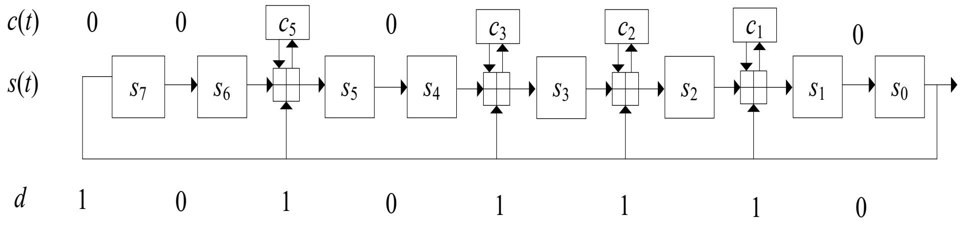
1. q is a (negative) prime and the bit length of q is $n + 1$ (n is the size of the main register).
2. Period $T = (|q| - 1)/2$ is prime.
3. $d = (1 + |q|)/2$.
4. The carry register c has l cells, and the number of nonzero d_i is $l + 1$, where $d = \sum_{i=0}^{n-1} d_i 2^i$.

To obtain maximum period of FCSR sequence, connection integer q should be prime number, and 2 is a primitive root modulo q. The FCSR produced sequence with maximum-period is called l-sequence [14, 17]. The period of l-sequences is $q - 1$ [20-21], and its linear complexity is $(q + 1)/2$ [14].

The properties of FCSR are described as follows [15, 17, 22-23]:

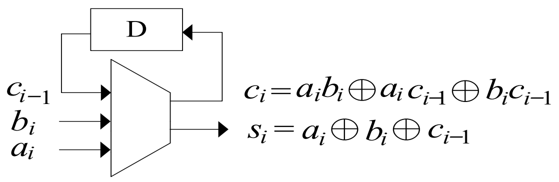
1. Every binary l-sequence of period $2t$, where t is a positive integer, has the property that the second half of any segment of length $2t$ is the bit-wise complement of the first half. This property is known as the symmetrical complementary property. The converse is not true. Not every symmetrically complementary sequence is an l-sequence. For example, when $q = 17$, the sequence is symmetrically complementary, but it is not an l-sequence because 2 is not primitive modulo 17.
2. Any strictly periodic sequence generated by a 2-adic FCSR with connection integer q is symmetrically complementary if and only if q divides $2^{T/2} + 1$, where T is the period of the sequence.
3. The linear complexity of an l-sequence of period $2t$ is at most $t + 1$
4. If q is a prime number and 2 is primitive root modulo q, then q is 2-prime. If $q = 2p + 1$, both p and q are 2-prime, then q is called strong 2-prime.
5. If the connection integer q of an FCSR is 2-prime, then the linear span (linear complexity) of the FCSR is less than or equals to $(q + 1)/2$. If $q = 2p + 1$ is a strong 2-prime, then the linear span (linear complexity) is $p + 1$.

Figure 3 shows the hardware circuit of FCSR Galois architecture. A Galois FCSR for $q = -347$, $d = 174 = (10101110)2$, $n = 8$ and $l = 4$ [24-25]. The symbol ⊞ represents addition with carry, as represented in Figure 4, where D is the D-type Flip-Flop.



$c(t)$
$s(t)$
d
0 0 c_5 0 c_3 c_2 c_1 0
s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0
1 0 1 0 1 1 1 0

Figure 3. Hardware description of FCSR Galois architecture



D
c_{i-1}
b_i
a_i
$c_i = a_i b_i \oplus a_i c_{i-1} \oplus b_i c_{i-1}$
$s_i = a_i \oplus b_i \oplus c_{i-1}$

Figure 4. Addition with Carry

2.2 Two FCSRs Combiners with XOR Function

The literature [15] proposed two FCSRs combiners with bit-wise XOR operation as the combining function. A schematic diagram depicting the generator is shown in Figure 5.

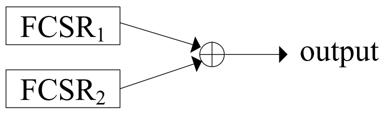


Figure 5. Two FCSRs combiners with XOR function

According to [15], the period T of output is lcm(T_1, T_2), the linear complexity LC of output is $(T_1 + T_2)/2 + 2$, where the lcm( ) is least common multiple, and the T_1, T_2 are the period of FCSR$_1$, FCSR$_2$, respectively.

2.3 The l_p-Geffe Generator Based on FCSRs

The Figure 6 shows the l_p-Geffe generator, which is composed of FCSR$_1$, FCSR$_2$, FCSR$_3$ and nonlinear function f [14]. Table 1 is the correlation probability of l_p-Geffe generator. From this table, we find the correlation probability of both the inputs bits w_j, y_j and the output bit z_j at clock j are 3/4. This is the main drawback of this circuit. And the correlation probability of input bit x_j and output bit z_j at clock j is 1/2.

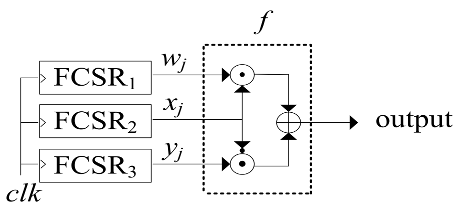


Figure 6. The l_p-Geffe generator

According to [14], the period T of output is lcm(T_1, T_2, T_3), the linear complexity L_{UB} of output is

$$L_{UB} = \min(\overline{L}_{UB}, T) . \tag{2}$$

with

$$\overline{L}_{UB} = \frac{(q_1+1)(q_2+1)}{4} + \frac{(q_2+1)(q_3+1)}{4} + \frac{(q_3+1)}{2} . \tag{3}$$

where T_1, T_2, T_3 are the period of FCSR$_1$, FCSR$_2$, and FCSR$_3$, and q_1, q_2, q_3 are the connection integer of FCSR$_1$, FCSR$_2$, and FCSR$_3$, respectively.

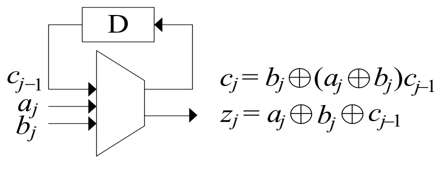
Table 1. Correlation probability of l_p-Geffe generator

w_j	x_j	y_j	z_j	Correlation probability
0	0	0	0	
0	0	1	1	
0	1	0	0	Output - Inputs:
0	1	1	0	prob($w_j = z_j$) = 3/4
1	0	0	0	prob($x_j = z_j$) = 1/2
1	0	1	1	prob($y_j = z_j$) = 3/4
1	1	0	1	
1	1	1	1	

2.4 Dawson's Summation Generator

Dawson's Summation Generator (DSG) is proposed by Dawson [26]. The Dawson's Summation Generator (DSG) is shown in Figure 7, and the symbols are defined as follows [27]:

– a_j: the input bit at clock j,
– b_j: the input bit at clock j,
– c_j: the carry bit at clock j with carry initial value c_{-1} = 0,
– z_j: the sum output at clock j, $z_j = a_j \oplus b_j \oplus c_{j-1}$.



D
c_{j-1}
a_j
b_j
$c_j = b_j \oplus (a_j \oplus b_j) c_{j-1}$
$z_j = a_j \oplus b_j \oplus c_{j-1}$

Figure 7. Dawson's Summation Generator (DSG)

Table 2 shows the input-output correlation probability and the carry-output correlation probability are both 1/2. The good features can prevent correlation attack. Therefore, [28] concluded that DSG is secure.

Table 2. Correlation probability of DSG

a_j	b_j	c_{j-1}	c_j	z_j	Correlation probability
0	0	0	0	0	
0	0	1	0	1	Output - Inputs:
0	1	0	1	1	prob($a_j = z_j$) = 1/2
0	1	1	0	0	prob($b_j = z_j$) = 1/2
1	0	0	0	1	prob($c_{j-1} = z_j$) = 1/2
1	0	1	1	0	Carry - Output:
1	1	0	1	0	prob($c_j = z_j$) = 1/2
1	1	1	1	1	

3 The Proposed Scheme

In this section, we introduce the proposed FCSR-based keystream generator for real-time communications in IoT. The basic proposed structure is combined the l_p-Geffe generator with two level DSG. We use two level DSG to increase the linear complexity, and apply good features of correlation probability of DSG to prevent attack. By the experiment results, we study the properties of the sequences in terms of their period, linear complexity and randomness, and the characteristics of the proposed keystream generator.

3.1 The Proposed of FCSR Based on DSG

We describe the properties of FCSR based DSG with four different inputs, that are divided into: (1) *l*-sequences as inputs, (2) non *l*-sequences and *l*-sequences as inputs, (3) *l*-sequences and non *l*-sequences as inputs, and (4) non *l*-sequences as inputs. And we will study the properties with their period and linear complexity.

(1) *l*-sequences as inputs

As shown in Figure 8, we use two FCSR *l*-sequences as the inputs of DSG, and we use different five experimental *l*-sequences as DSG inputs. In our experiment, there are five combinations of the *l*-sequences as shown in Table 3. From Table 3, we can find period and linear complexity of output *l*-sequences as DSG inputs, where q_1, T_1, LC_1, q_2, T_2, LC_2, T and LC are the connection integer, period and linear complexity of FCSR$_1$, FCSR2 and output, respectively.

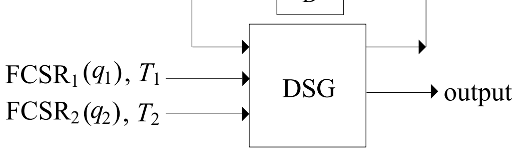


Figure 8. *l*-sequences as inputs

Table 3 lists the experimental result of *l*-sequences as inputs, where q_i, T_i , LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, and i = 1, 2. According to the above experimental result of *l*-sequences as inputs, we find the period T of output is

$$T \cong T_1 * LC_2. \tag{4}$$

And the linear complexity LC of output is

$$LC \cong T. \tag{5}$$

Table 3. The experimental result of *l*-sequences as inputs

q_1	q_2	T_1	LC_1	T_2	LC_2	T	LC
107	131	106	54	130	66	6890	6889
19	131	18	10	130	66	1170	1166
37	83	36	19	82	42	1476	1476
83	149	82	41	148	75	6068	6064
19	83	18	10	82	42	738	734

(2) Non *l*-sequences and *l*-sequences as inputs

We use non *l*-sequence and *l*-sequence as the inputs of DSG, shown in Figure 9. In our experiment, there are five combinations of the non *l*-sequence and *l*-sequences as shown in Table 4. From Table 4, we can find period and linear complexity of output non *l*-sequences and *l*-sequences as DSG inputs, where q_1, T_1, LC_1, q_2, T_2, LC_2, T and LC are the connection integer, period and linear complexity of non *l*-sequence, *l*-sequence, and output respectively.

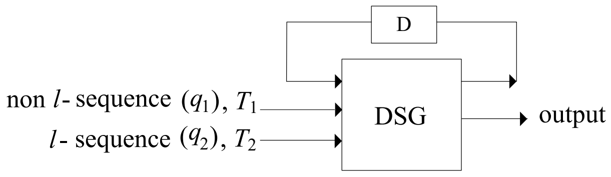

D
non *l*- sequence (q_1), T_1
l- sequence (q_2), T_2
DSG
output

Figure 9. Non *l*-sequence and *l*-sequence as inputs

Table 4 shows the experimental result of non *l*-sequences and *l*-sequences as DSG inputs, where q_i, T_i , LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, and i = 1, 2. According to the above experimental result, we find the period T of output is

$$T = \mathrm{lcm}(T_1, T_2). \tag{6}$$

and the linear complexity LC of output is

$$LC \cong T. \tag{7}$$

Table 4. The experimental result of non *l*-sequences and *l*-sequences as the inputs of DSG

q_1	q_2	T_1	LC_1	T_2	LC_2	T	LC
15	19	4	4	18	10	36	36
31	19	5	5	18	10	90	89
35	19	12	12	18	10	36	35
39	37	12	9	36	19	36	34
17	37	4	4	36	19	36	34

(3) *l*-sequences and non *l*-sequences as inputs

The DSG inputs are *l*-sequence and non *l*-sequence shown in Figure 10. We use five experimental comnibations of *l*-sequences and non *l*-sequences as inputs of DSG shown in Table 5. From Table 5, we can find period and linear complexity of output *l*-sequences and non *l*-sequences as DSG inputs, where q_1, T_1, LC_1, q_2, T_2, LC_2, T and LC are the connection integer, period and linear complexity of l-sequence, non l-sequence, and output respectively.

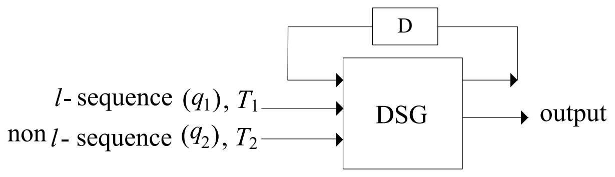

D
DSG
l- sequence (q_1), T_1
non l- sequence (q_2), T_2
output

Figure 10. l-sequences and non l-sequences as inputs

Table 5 lists the experimental result of l-sequences and non l-sequences as DSG inputs, where q_i, T_i, LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, and $i = 1, 2$. According to the above experimental result, we find the period T of output is

$$T = \mathrm{lcm}(T_1, T_2). \tag{8}$$

and the linear complexity LC of output is

$$LC \cong T. \tag{9}$$

Table 5. The experimental result of l-sequences and non l-sequences as the inputs of DSG

q_1	q_2	T_1	LC_1	T_2	LC_2	T	LC
19	15	18	10	4	4	36	35
19	31	18	10	5	5	90	89
19	35	18	10	12	12	36	35
37	39	36	19	12	9	36	34
37	17	36	19	4	4	36	34

(4) Non l-sequences as inputs

As shown in Figure 11, we use two non l-sequences are as the inputs of DSG, and we use five experimental combinations of non l-sequences as DSG inputs as shown in Table 6. From Table 6, we can find period and linear complexity of output non l-sequences as DSG inputs, where q_1, T_1, LC_1, q_2, T_2, LC_2, T and LC are the connection integer, period and linear complexity of non l-sequence 1, non l-sequence 2, and output respectively.

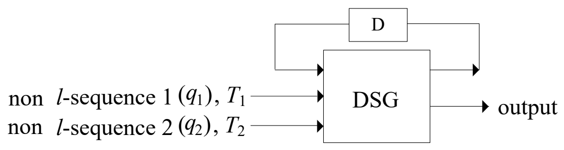

D
DSG
non l-sequence 1 (q_1), T_1
non l-sequence 2 (q_2), T_2
output

Figure 11. Non l-sequences as inputs

Table 6. The experimental result of non l-sequences as the inputs of DSG

q_1	q_2	T_1	LC_1	T_2	LC_2	T	LC
15	31	4	4	5	5	20	19
15	39	4	4	12	9	12	12
31	39	5	5	12	9	60	54
31	17	5	5	4	4	12	18
17	35	4	4	12	12	12	12

Table 6 shows the experimental result of non l-sequences as DSG inputs, where q_i, T_i, LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, and $i = 1, 2$. According to the above experimental result, we find the period T of output is

$$T = \mathrm{lcm}(T_1, T_2). \tag{10}$$

and the linear complexity LC of output is

$$LC \cong T. \tag{11}$$

Finally, we find output of l-sequences as DSG inputs has large period than other period of output, and all the linear complexity (LC) of output is close to period of output.

3.2 FCSR Based on Two Level DSG

In this section, we describe FCSR based two level DSG. As shown in Figure 12, the two level DSG is composed of three FCSRs and two DSGs. Table 7 shows the experimental result of FCSR based two level DSG, where q_i, T_i, LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, $i = 1, 2, 3$ and T and LC are the period and linear complexity of output.

According to the above experimental result, we find the period T of output is

$$T = \mathrm{lcm}(T_1, T_2, T_3). \tag{12}$$

and the linear complexity LC of output is

$$LC \cong T. \tag{13}$$

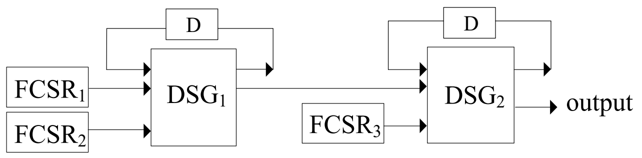


Figure 12. FCSR based on two level DSG

Table 7. The experimental result of FCSR based on two level DSG

q_1	q_2	q_3	T_1	LC_1	T_2	LC_2	T_3	LC_3	T	LC
19	37	53	18	10	36	19	52	27	468	466
5	11	19	4	3	10	6	18	10	90	88
17	29	53	36	19	28	15	52	27	3276	3026
11	61	29	10	6	60	31	28	15	420	409
53	61	107	52	27	60	31	106	54	41340	41338

3.3 FCSR Based on Hierarchical DSG

In this section, we describe FCSR based hierarchical DSG. As shown in Figure 13, the FCSR based hierarchical DSG is composed of four FCSRs and three DSGs. Table 8 shows the experimental result of FCSR based hierarchical DSG of FCSR, where q_i, T_i, LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, and i = 1, 2, 3, 4 and T and LC are the period and linear complexity of output.

According to the above experimental result, we find the period T of output is

$$T = \mathrm{lcm}(T_1, T_2, T_3, T_4). \tag{14}$$

and the linear complexity LC of output is

$$LC \cong T. \tag{15}$$

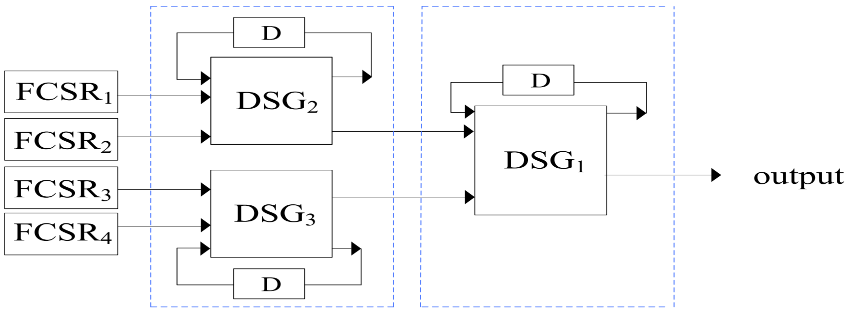


Figure 13. FCSR based on hierarchical DSG

Table 8. The experimental result of FCSR based on hierarchical DSG

q_1	q_2	q_3	q_4	T_1	LC_1	T_2	LC_2	T_3	LC_3	T_4	LC_4	T	LC
19	37	83	131	18	10	36	19	82	42	130	66	95940	95940
19	37	373	1019	18	10	36	19	372	182	1018	510	568044	568036
11	29	61	227	10	6	28	15	60	31	226	114	47460	47460
19	29	37	53	18	10	28	15	36	19	52	27	3276	3276
5	11	29	53	4	3	10	6	28	114	52	27	1820	1820

3.4 FCSR Based on *lp*-Geffe Generator Cascading DSG

In this section, we will investigate the properties of FCSR based on l_p-Geffe generator cascading DSG. FCSR based Geffe generator is also known as l_p-Geffe generator [14]. As shown in Figure 14, the l_p-Geffe generator cascading DSG is composed of four FCSRs and a DSG. Table 9 shows the experimental result of FCSR based l_p-Geffe generator cascading DSG, where q_i, T_i, LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, i = 1, 2, 3, 4 and T and LC are the period and linear complexity of output.

According to the above experimental result, we find the period T of output is

$$T = \mathrm{lcm}(T_1, T_2, T_3, T_4). \tag{16}$$

and the linear complexity LC of output is

$$LC \cong T. \tag{17}$$

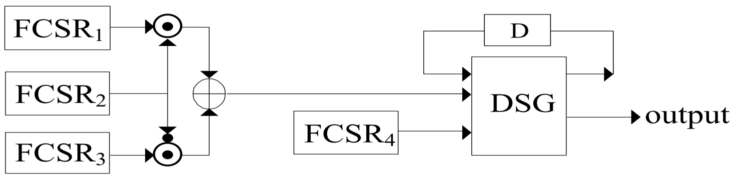


Figure 14. FCSR based on *lp*-Geffe generator cascading DSG

Table 9. The experimental result of FCSR based on l_p-Geffe generator cascading DSG

q_1	q_2	q_3	q_4	T_1	LC_1	T_2	LC_2	T_3	LC_3	T_4	LC_4	T	LC
19	37	107	131	18	10	36	19	106	54	130	66	124020	124017
19	107	131	11	18	10	106	54	130	66	10	6	62010	61994
19	83	131	11	18	10	82	66	130	66	10	6	47970	47970
19	53	107	131	18	10	52	27	106	54	130	66	124017	124017
19	107	173	37	18	10	106	54	172	87	36	19	124017	82044

3.5 FCSR Based on l_p-Geffe Generator and DSG Hybrid Keystream Generator

In this section, we describe the proposed FCSR based l_p-Geffe generator and DSG hybrid keystream generator. The circuit architecture is composed of FCSRs, nonlinear function f, and DSG. The design of circuit architecture is shown in Figure 15.

The FCSR$_1$, FCSR$_2$, FCSR$_3$, combining with nonlinear function f are called l_p-Geffe generator [14]. The nonlinear function f is a Boolean function that can be expressed algebraic normal form. The algebraic normal form is a sum of products of binary variables, addition and multiplication are defined in the binary field $GF(2)$ [15]. The Boolean function is

$$f(x_1, x_2, x_3) = x_1x_2 + x_2x_3 + x_3. \quad \textbf{(18)}$$

where x_1, x_2, and x_3 are the outputs of FCSR$_1$, FCSR$_2$, and FCSR$_3$, respectively. The inputs of DSG are the output sequence of l_p-Geffe generator y and output of FCSR$_4$. Finally, DSG produces the output bits of keystream z.

Assume clock j from 0, 1, 2, …, and the variables define of DSG are described as follows:

- x_4 : output bit of FCSR$_4$ at clock j
- y : output bit of l_p-Geffe generator at clock j
- $w_{(j)}$: carry bit of DSG at clock j with initial value $w_{(-1)} = 0$
- z : the final output bit at clock j

The module of DSG in our circuit can be defined as follows:

DSG:

$$z = y \oplus x_4 \oplus w_{(j-1)}. \quad \textbf{(19)}$$

$$w_{(j)} = x_4 \oplus (y \oplus x_4)w_{(j-1)}. \quad \textbf{(20)}$$

The input-output correlation probability and the carry-output correlation probability of DSG are both 1/2 as demonstrated in Table10.

Table 10. Correlation probability of DSG based FCSR

y	x_4	$w_{(j-1)}$	$w_{(j)}$	z	Correlation probability
0	0	0	0	0	
0	0	1	0	1	Output - Inputs:
0	1	0	1	1	prob($y = z$) = 1/2
0	1	1	0	0	prob($x_4 = z$) = 1/2
1	0	0	0	1	prob($w_{(j-1)} = z$) = 1/2
1	0	1	1	0	Carry - Output:
1	1	0	1	0	prob($w_{(j)} = z$) = 1/2
1	1	1	1	1	

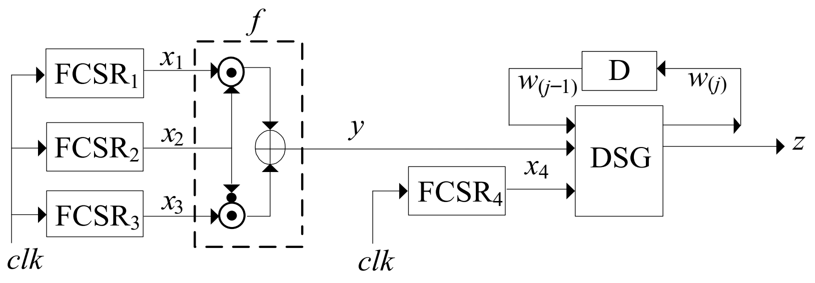


Figure 15. The block diagram of FCSR based on l_p-Geffe generator and DSG hybrid keystream generator

Table 11. The experimental of FCSR based on l_p-Geffe generator and DSG hybrid keystream generator

q_1	q_2	q_3	T_1	LC_1	T_2	LC_2	T_3	LC_3	T_y	LCy
19	53	107	18	10	52	27	106	54	24804	1647
19	107	131	18	10	106	54	130	66	62010	3890
19	83	131	18	10	82	42	130	66	47970	3026
19	107	173	18	10	106	54	172	87	82044	5044
19	11	11	18	10	10	6	148	75	6660	424

3.6 The Properties of FCSR Based on l_p-Geffe Generator and DSG Hybrid Keystream Generator

In this section, we describe the properties FCSR based l_p-Geffe generator and DSG hybrid keystream generator. As shown in Figure 15, the y denotes the output sequences of l_p-Geffe generator, the z denotes the output sequences of DSG. The period of y and z are expressed as T_y and T_z, respectively. The q_i, T_i, LC_i are the connection integer, period, linear complexity of FCSR$_i$, respectively, and i = 1, 2, 3, 4.

Table 11 shows the experimental of l_p-Geffe generator. According to [14], the output period T_y of l_p-Geffe generator is

$$T_y = \text{lcm}(T_1, T_2, T_3). \quad \textbf{(21)}$$

the linear complexity LC_y of l_p-Geffe generator output is

$$LC_y = \min(LC, T_y). \quad \textbf{(22)}$$

and

$$LC = \frac{(q_1+1)(q_2+1)}{4} + \frac{(q_2+1)(q_3+1)}{4} + \frac{(q_3+1)}{2}. \quad \textbf{(23)}$$

where lcm is mean the Least Common Multiple.

Table 12. The experimental result of DSG

Q_4	T_4	LC_4	T_z	LC_z
131	130	124020	124020	124020
11	10	6	62010	61994
11	10	6	47970	47970
37	36	19	82044	82044
53	52	27	86580	86575

Table 12 lists the experimental result of DSG. We use output of l_p-Geffe generator and output of FCSR$_4$ as the inputs of DSG. According to the above experimental result, we find the period of z, T_z is:

$$T_z = \text{lcm}(T_y, T_4). \quad \textbf{(24)}$$

and the linear complexity LC_z is:

$$LC_z \cong T_z. \quad \textbf{(25)}$$

According to subsection 3.1, because the output sequence of l_p-Geffe generator is not l-sequences, and the finial overall period is the Least Common Multiple (lcm) of the period of non l-sequences T_y and the period of l-sequences T_4.

3.7 The Connection Integers of FCSRs

For the aim of security, we expand the parameters of the circuit proposed in section 3.6. The size of internal state of the proposed keystream generator is 416 bits, and it composed of FCSR$_1$, FCSR$_2$, FCSR$_3$, and FCSR$_4$, respectively. The corresponding maximum length of FCSR$_1$, FCSR$_2$, FCSR$_3$, and FCSR$_4$ are 96, 128, 64, 128, and the connection integer q_i, i = 1, 2, 3, 4, for maximal-period FCSRs are described as follows [29]:

$$\text{FCSR}_1\text{: } q_1 = 2^{96} + 2^{58} + 2^{35} + 2^2 - 1. \quad \textbf{(26)}$$

$$\text{FCSR}_2\text{: } q_2 = 2^{128} + 2^{5} + 2^{4} + 2^2 - 1. \quad \textbf{(27)}$$

$$\text{FCSR}_3\text{: } q_3 = 2^{64} + 2^{59} + 2^{8} + 2^2 - 1. \quad \textbf{(28)}$$

$$\text{FCSR}_4\text{: } q_4 = 2^{128} + 2^{21} + 2^{19} + 2^2 - 1. \quad \textbf{(29)}$$

Basicly, the architectures of FCSR are divided into Galois architecture and Fibonacci architecture. In the proposed keystream generator, we choose Galois architecture of FCSR to speed up the operation of initialization.

3.8 Key/IV Setup Procedure

To meet the security strength of Advanced Encryption Standard (AES) cipher, the secret *key* (k_i) and initialization vector (iv_i) both are 128 ($0 \leq i \leq 127$) bits as inputs of keystream generator. We introduce Key/IV setup procedure, and it can be divided initial filling procedure and key initialization procedure. First, we load part bits of the secret key into initial state of the FCSRs. Then we load remaining bits of secret key and *iv* into the kystream generator in the key initialization procedure. It works as follows:

Initial Filling Procedure:

We denote secret *key* 128 bits ($k = k_0, \ldots, k_{127}$) and initialization vector 128 bits ($iv = iv_0, \ldots, iv_{127}$). The initial states of FCSR$_1$, FCSR$_2$, FCSR$_3$, and FCSR$_4$ are expressed as a_i, b_i, c_i, d_i, respectively:

$$a_i,\ 0 \leq i \leq 95. \quad \textbf{(30)}$$

$$b_i,\ 0 \leq i \leq 127. \quad \textbf{(31)}$$

$$c_i,\ 0 \leq i \leq 63. \quad \textbf{(32)}$$

$$d_i,\ 0 \leq i \leq 127. \quad \textbf{(33)}$$

The initial fillings of initial states in FCSRs are described as follows:

$$(a_{95}, a_{94}, \ldots, a_1, a_0) \leftarrow (k_{127}, k_{126}, \ldots, k_{33}, k_{32}). \quad \textbf{(34)}$$

$$(b_{127}, b_{126}, \ldots, b_1, b_0) \leftarrow (k_{127}, k_{126}, \ldots, k_1, k_0). \quad \textbf{(35)}$$

$$(c_{63}, c_{62}, \ldots, c_1, c_0) \leftarrow (k_{127}, k_{126}, \ldots, k_{65}, k_{64}). \quad \textbf{(36)}$$

$$(d_{127}, d_{126}, \ldots, d_1, d_0) \leftarrow (k_{127}, k_{126}, \ldots, k_1, k_0). \quad \textbf{(37)}$$

Notice that, only part bits of the secret *key* are loaded into the initial state of FCSRs.

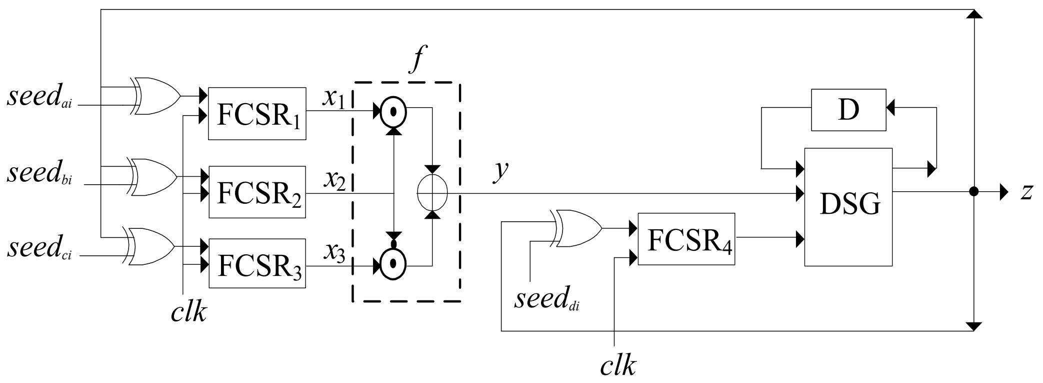


Figure 16. Key initialization procedure

Key Initialization Procedure:

We load the remaining of secret *key* and *iv* are loaded into the keystream generator in key initialization procedure, as shown in Figure 16. We use 192 clocks to implement this procedure and there is no bit of keystream output during this initialization procedure.

We apply parameter $seed_{ai}$, $seed_{bi}$, $seed_{ci}$, and $seed_{di}$ denoted remaining *key*, and *iv* for $0 \le t \le 191$. The parameters $seed_{ai}$, $seed_{bi}$, $seed_{ci}$, and $seed_{di}$ are given as follows:

$$(seed_{a191}, \ldots, seed_{a0}) \leftarrow (0, \ldots, 0, iv_{127}, \ldots, iv_0, k_{31}, \ldots, k_0). \quad \textbf{(38)}$$

$$(seed_{b191}, \ldots, seed_{b0}) \leftarrow (0, \ldots, 0, iv_{127}, \ldots, iv_0). \quad \textbf{(39)}$$

$$(seed_{c191}, \ldots, seed_{c0}) \leftarrow (iv_{127}, \ldots, iv_0, k_{63}, \ldots, k_0). \quad \textbf{(40)}$$

$$(seed_{d191}, \ldots, seed_{d0}) \leftarrow (0, \ldots, 0, iv_{127}, \ldots, iv_0). \quad \textbf{(41)}$$

4 Security Properties

A good pseudorandom sequence should have good statistical properties, large period and high linear complexity. In this section, we expand the parameters of the proposed l_p-Geffe generator and DSG hybrid keystream generator, and the period, linear complexity and some attacks of the proposed circuit will be evaluated.

4.1 Period and Linear Complexity of the Proposed Generator with Large Parameters

In this subsection, we describe period of large system. As shown in Figure 15, the period of *y*, and *z* are expressed as T_y, T_z, respectively. The T_i, LC_i are the period and linear complexity of FCSR$_i$, respectively, and $i = 1, 2, 3, 4$. The periods are described as follows [29]:

$$\text{FCSR}_1\text{: } T_1 = 2^{96} + 2^{58} + 2^{35} + 2. \quad \textbf{(42)}$$

$$\text{FCSR}_2\text{: } T_2 = 2^{128} + 2^{5} + 2^{4} + 2. \quad \textbf{(43)}$$

$$\text{FCSR}_3\text{: } T_3 = 2^{64} + 2^{59} + 2^{8} + 2. \quad \textbf{(44)}$$

$$\text{FCSR}_4\text{: } T_4 = 2^{128} + 2^{21} + 2^{19} + 2. \quad \textbf{(45)}$$

According to [14], the T_y can be written as

$$T_y = \text{lcm}(T_1, T_2, T_3). \quad \textbf{(46)}$$

$$T_y = \text{lcm}(2^{96} + 2^{58} + 2^{35} + 2, 2^{128} + 2^{5} + 2^{4} + 2, 2^{64} + 2^{59} + 2^{8} + 2). \quad \textbf{(47)}$$

$$T_y \cong 2^{281}. \quad \textbf{(48)}$$

Finally, the T_z can be written as

$$T_z = \text{lcm}(T_y, T_4). \quad \textbf{(49)}$$

$$T_z = \text{lcm}(2^{281}, 2^{128} + 2^{21} + 2^{19} + 2). \quad \textbf{(50)}$$

$$T_z \cong 2^{408}. \quad \textbf{(51)}$$

where lcm( ) denotes the function of Least Common Multiple.

According to above result, the proposed circuit has large period and it is required for security consideration.

The linear complexity is described as follows [29]:

$$\text{FCSR}_1\text{: } LC_1 = 2^{95} + 2^{57} + 2^{34} + 2. \quad \textbf{(52)}$$

$$\text{FCSR}_2\text{: } LC_2 = 2^{127} + 2^{4} + 2^{3} + 2. \quad \textbf{(53)}$$

$$\text{FCSR}_3\text{: } LC_3 = 2^{63} + 2^{58} + 2^{7} + 2. \quad \textbf{(54)}$$

$$\text{FCSR}_4\text{: } LC_4 = 2^{127} + 2^{20} + 2^{18} + 2. \quad \textbf{(55)}$$

From [14], the LC_y can be written as

$$LC_y = \min(LC, T_y), \quad \textbf{(56)}$$

and

$$LC = \frac{(q_1+1)(q_2+1)}{4} + \frac{(q_2+1)(q_3+1)}{4} + \frac{(q_3+1)}{2}. \quad \textbf{(57)}$$

$$LC \cong 2^{222}. \quad \textbf{(58)}$$

$$LC_y \cong 2^{222}. \quad \textbf{(59)}$$

Finally, the LC_z can be written as

$$LC_z \cong T_z. \quad \textbf{(60)}$$

$$LC_z \cong 2^{408}. \quad \textbf{(61)}$$

We use large parameters to increase period, linear complexity, and security of proposed scheme. When the computation cost of a stream cipher is required equal to the security strength of AES, and it is approximately equal to $O(2^{128})$. For our scheme, the linear complexity LC of the proposed stream cipher is approximately equal to 2^{408}. The computation cost of the proposed cipher can be written as follows:

$$O(LC^2) = O((2^{408})^2) = O(2^{816}) > O(2^{128}). \quad \textbf{(62)}$$

It can be seen that, the computation cost of our proposed cipher satisfied the security strength of AES, even stronger than AES.

4.2 Algebraic Attack

In this subsection, algebraic attack will be discussed. The basic principle of algebraic attacks goes back to Shannon's work: these techniques consist in expressing the whole cipher as a large system of algebraic equations, which can be solved to recover the secret key [30].

First, we call L is connection function and assume L is public, and only the state is secret. We also assume the function f computes the output bit from the state is public and does not depend on the secret of the cipher. The function f is called nonlinear filter. Let $k_0, \ldots, k_{n-1}$ is the initial state, then the output of the cipher is given by:

$$\begin{cases} b_0 = f(k_0,\ldots,k_{n-1}) \\ b_1 = f(L(k_0,\ldots,k_{n-1})) \\ b_2 = f(L^2(k_0,\ldots,k_{n-1})) \\ \qquad \vdots \end{cases}$$

Our problem is to recover the key $k = (k_0, \ldots, k_{n-1})$ from some subset of keystream bits b_i [31]. The numbers of equation are more than variables of equation. It can be easily solved by Gaussian elimination [31].

For our scheme, the algebraic equation at clock t for FCSR$_i$, $i = 1, 2, 3, 4$ and output z are expressed as follows.

FCSR$_1$:

clock t	Algebraic equation
0	$x_1(t=0) = k_{32}$
1	$x_1(t=1) = k_{33}$
2	$x_1(t=2) = k_{34} \oplus k_{32}$
3	$x_1(t=3) = k_{35} \oplus k_{33} \oplus k_{34}k_{32}$
4	$x_1(t=4) = k_{36} \oplus (k_{34} \oplus k_{32}) \oplus \{k_{35}k_{33} \oplus k_{33}(k_{34}k_{32}) \oplus k_{35}(k_{34}k_{32})\}$
$\vdots$	$\vdots$

FCSR$_2$:

clock t	Algebraic equation
0	$x_2(t=0) = k_0$
1	$x_2(t=1) = k_1$
2	$x_2(t=2) = k_2 \oplus k_0$
3	$x_2(t=3) = k_3 \oplus k_1 \oplus k_2k_0$
4	$x_2(t=4) = (k_4 \oplus k_0) \oplus (k_2 \oplus k_0) \oplus \{k_3k_1 \oplus k_1(k_2 \oplus k_0) \oplus (k_2 \oplus k_0)k_3\}$
$\vdots$	$\vdots$

FCSR$_3$:

clock t	Algebraic equation
0	$x_3(t=0) = k_{64}$
1	$x_3(t=1) = k_{65}$
2	$x_3(t=2) = k_{66} \oplus k_{64}$
3	$x_3(t=3) = k_{67} \oplus k_{65} \oplus k_{66}k_{64}$
4	$x_3(t=4) = k_{68} \oplus (k_{66} \oplus k_{64}) \oplus \{k_{67}k_{65} \oplus k_{65}(k_{66} \oplus k_{64}) \oplus (k_{66} \oplus k_{64})k_{67}\}$
$\vdots$	$\vdots$

FCSR$_4$:

clock t	Algebraic equation
0	$x_4(t=0) = k_0$
1	$x_4(t=1) = k_1$
2	$x_4(t=2) = k_2 \oplus k_0$
3	$x_4(t=3) = k_3 \oplus k_1 \oplus k_2k_0$
4	$x_4(t=4) = k_4 \oplus (k_2 \oplus k_0) \oplus \{k_3k_1 \oplus k_1(k_2 \oplus k_0) \oplus (k_2 \oplus k_0)k_3\}$
$\vdots$	$\vdots$

FCSR$_5$:

clock t	Algebraic equation
0	$z\,(t=0) = (k_{32}k_0 \oplus k_0k_{64} \oplus k_{64}) \oplus k_0$
1	$z\,(t=1) = (k_{33}k_1 \oplus k_1k_{65} \oplus k_{65}) \oplus k_1 \oplus k_0$
2	$z\,(t=2) = [(k_{34} \oplus k_{32})(k_2 \oplus k_0) \oplus (k_2 \oplus k_0)(k_{66} \oplus k_{64}) \oplus (k_{66} \oplus k_{64})] \oplus (k_2 \oplus k_0) \oplus (k_1 \oplus [(k_{33}k_1 \oplus k_1k_{65} \oplus k_{65}) \oplus k_1] \oplus k_0)$
3	$z\,(t=3) = [(k_{35} \oplus k_{33} \oplus k_{34}k_{32})(k_3 \oplus k_1 \oplus k_2k_0) \oplus (k_3 \oplus k_1 \oplus k_2k_0)(k_{67} \oplus k_{65} \oplus k_{66}k_{64}) \oplus (k_{67} \oplus k_{65} \oplus k_{66}k_{64})] \oplus (k_3 \oplus k_1 \oplus k_2k_0) \oplus \{[(k_{34} \oplus k_{32})(k_2 \oplus k_0) \oplus (k_2 \oplus k_0)(k_{66} \oplus k_{64}) \oplus (k_{66} \oplus k_{64})] \oplus (k_2 \oplus k_0)\}(k_1 \oplus [(k_{33}k_1 \oplus k_1k_{65} \oplus k_{65}) \oplus k_1]k_0)\}$
4	$x_4(t=4) = k_4 \oplus (k_2 \oplus k_0) \oplus \{k_3k_1 \oplus k_1(k_2 \oplus k_0) \oplus (k_2 \oplus k_0)k_3\}$
$\vdots$	$\vdots$

The monomials of output z are listed as follows:

$z\,(t=0)$: $k_{32}k_0, k_0k_{64}, k_{64}, k_0$

$z\,(t=1)$: $k_{33}k_1, k_1k_{65}, k_{65}, k_1, k_0$

$z\,(t=2)$: $k_{34}k_2, k_{34}k_0, k_{32}k_2, k_{32}k_0, k_2k_{66}, k_2k_{64}, k_0k_{66}, k_0k_{64}, k_{66}, k_{64}, k_2, k_0, k_1, k_{33}k_1, k_1k_{65}, k_{65}$

$z\,(t=3)$: $k_{35}k_3, k_{35}k_1, k_{35}k_2k_0, k_{33}k_3, k_{33}k_1, k_{33}k_2k_0, k_{34}k_{32}k_3, k_{34}k_{32}k_1, k_{34}k_{32}k_2k_0, k_3k_{67}, k_3k_{65}, k_3k_{66}k_{64}, k_1k_{67}, k_1k_{65}, k_1k_{66}k_{64}, k_2k_0k_{67}, k_2k_0k_{65}, k_2k_0k_{66}k_{64}, k_{34}k_2, k_{34}k_0, k_{32}k_2, k_{32}k_0, k_2k_{66}, k_2k_{64}, k_0k_{66}, k_0k_{64}, k_{33}k_1k_0, k_1k_{65}k_0, k_{65}k_0, k_1k_0$

$\vdots$

From the above analysis, we find the numbers of variable for output $z\,(t)$ at $t = 0, 1, 2, 3, 4\ldots$ are 4, 5, 16, 30…, respectively, and they are much than the algebraic equations can be listed. When the clock increase, the variables of equation is also gradually increase. However, the numbers of variable are more than the numbers of equation. The results show that the proposed stream cipher can against algebraic attack.

4.3 Time-Memory-Data Tradeoff Attack

In 1980, Hellman introduced a general technique for breaking arbitrary block cipher called time-memory tradeoff attack. Furthermore, Babbage-Golić and Biryukov-Shamir proposed that a different time-memory-data tradeoff attack that is applicable to stream ciphers [32].

The time-memory-data tradeoff attack has two phases: During the preprocessing phase (which can take a very long time) the attacker explores the general structure of the cryptosystem, and summarizes his findings in large tables. During the realtime phase, the attacker is given actual data produced from a particular unknown key, and his goal is to use the precomputed tables in order to find the key as quickly as possible.

There are five key parameters for time-memory-data tradeoff attack [32]:

- N represents the size of the search space.
- P represents the time required by the preprocessing phase of the attack.
- M represents the amount of random access memory available to the attacker.
- T represents the time required by the realtime phase of the attack.
- D represents the amount of realtime data available to the attacker.

Babbage and Golić suppose internal state of the streamcipher has N different states and D different keystreams of length log$N$. The attacker aims to recover one of the internal states corresponding to any one of the keystreams. We have memory requirement for this attack is $M = N/D$. It suffices to look for table D times and the time complexity is $T = D$. If ignoring some of the data, the time-memory-data tradeoff attack is $TM = N$ with $P = M$ for $1 \leq T \leq D$.

For example, $T = M = D = N^{1/2}$ constitutes an attack. Babbage suggests if a secret key length of k bits, then a state size of at least $2k$ bits, and it is a design principle for stream ciphers [33].

Biryukov and Shamir combined the works of Hellman and Babbage-Golić to bring a new time-memory-data tradeoff attack on stream ciphers. As with the work of Babbage and Golić, objective of the attacker is to recover any one of the internal states of the stream cipher, given D different keystreams. Biryukov and Shamir proposed the (t/D) tables, each of size m and taking time t, require a storage space of size $M = tm/D$, and the time complexity $T = t^2$. When $1 \leq D^2 \leq T$, then time-memory-data tradeoff attack is $TM^2D^2 = N^2$ and $P = N/D$. For example, $T = M = N^{1/2}$ with $D = N^{1/4}$ constitutes an attack. Biryukov and Shamir suggest the internal state of a stream cipher should at least twice the number of key bits [33].

Under the situation of stream cipher with *IV*, we only consider *key* and *IV*, and don't care the other states of circuit. The search space is $N = 2^{k+v}$, time and memory complexity $T = M = 2^{\frac{1}{2}(k+v)}$, where k is *key* bits and v is *IV* bits. If $v < k$, this complexity is smaller than key exhaustive search complexity 2^k. That is, if bit-length of *IV* is shorter than *key*'s, then the stream cipher is vulnerable to time-memory-data tradeoff attack [33].

For our scheme, the proposed stream cipher has *key* 128 bits and *iv* 128 bits. In order to avoid the time-memory-data tradeoff attacks, the size of the internal state must at least 256 bits. However, the size of the internal state of the proposed cipher is 416 bits, it means that the size of the search space $N = 2^{416} > 2^{256}$. For the Babbage-Golić proposed attack $T = M = D = N^{1/2} = 2^{208}$, for the Biryukov-Shamir proposed attack $T = M = N^{1/2} = 2^{208}$, $D = N^{1/4} = 2^{104}$, and under the situation of stream cipher with *IV*, we only consider *key* and *IV*, and don't care the other states of circuit the search space is $N = 2^{256}$, time and memory complexity $T = M = 2^{128}$.

In the proposed scheme, the length of *iv* is not small than the length of *key*. So it is not vulnerable to time-memory-data tradeoff attack. The results show that the proposed stream cipher can resist against these attacks.

4.4 Correlation Immunity Properties

In this section, we describe correlation immunity properties of the proposed scheme. In 1983, Siegenthaler showed that if the combining function is carelessly chosen then cryptosystems can be broken by the correlation attack, so the combining function must have good correlation-immunity [34].

For our scheme, we use l_p-Geffe generator and DSG to produce the output keystream and x_1, x_2, x_3 are as the inputs of l_p-Geffe generator, the y, x_4, $w_{(j\text{-}1)}$ are as the inputs of DSG.

Table 13. Correlation probabilities for l_p-Geffe generator

x_1	x_2	x_3	Output of l_p-Geffe generator (y)	Correlation probability
0	0	0	0	
0	0	1	1	Input Output:
0	1	0	0	$P[x_1 = y] = 3/4$
0	1	1	0	$P[x_2 = y] = 1/2$
1	0	0	0	$P[x_3 = y] = 3/4$
1	0	1	1	
1	1	0	1	
1	1	1	1	

Table 14. Correlation probabilities for output z

y	x_4	$w_{(j\text{-}1)}$	z	$w_{(j)}$	Correlation probability
0	0	0	0	0	
0	0	1	1	0	Input Output:
0	1	0	1	1	$P[y = z] = 1/2$
0	1	1	0	0	$P[x_4 = z] = 1/2$
1	0	0	1	0	$P[w_{(j\text{-}1)} = z] = 1/2$
1	0	1	0	1	Carry Output:
1	1	0	0	1	$P[w_{(j)} = z] = 1/2$
1	1	1	1	1	

Table 15. Correlation probabilities for output z

x_1	x_2	x_3	y	x_4	$w_{(j\text{-}1)}$	z
0	0	0	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	1	1
0	1	0	0	0	1	1
0	1	1	0	0	1	1
1	0	0	0	0	1	1
0	0	0	0	1	0	1
0	1	0	0	1	0	1
0	1	1	0	1	0	1
1	0	0	0	1	0	1
0	0	0	0	1	1	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	1	1	0
0	0	1	1	0	0	1
1	0	1	1	0	0	1
1	1	0	1	0	0	1
1	1	1	1	0	0	1
0	0	1	1	0	1	0
1	0	1	1	0	1	0
1	1	0	1	0	1	0
1	1	1	1	0	1	0
0	0	1	1	1	0	0
1	0	1	1	1	0	0
1	1	0	1	1	0	0
1	1	1	1	1	0	0
0	0	1	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Table 13 shows the correlation probabilities for l_p-Geffe generator, the correlation probabilities of l_p-Geffe generator are

$$P[x_1 = y] = P[x_3 = y] = 3/4,\ P[x_2 = y] = 1/2. \quad \textbf{(63)}$$

Table 14 shows the correlation probabilities for output y, the correlation probabilities of output z are

$$P[y = z] = P[x_4 = z] = P[w_{(j-1)} = z] = P[w_{(j)} = z] = 1/2. \quad \textbf{(64)}$$

Table 15 shows the correlation probabilities for l_p-Geffe generator and output z, the correlation probabilities of l_p-Geffe generator and output z are

$$P[x_1 = z] = 1/2,\ P[x_2 = z] = 1/2,\ P[x_3 = z] = 1/2. \quad \textbf{(65)}$$

5 Experimental Results

In this section, we use Verilog hardware description language to design proposed circuit, and introduce randomness simulation result of our design circuit. In the simulation, 100 random *key*s and 100 random *iv*s are used to produce 100 output keystreams, and the randomness tests are performed under the FIPS PUB 140-1 [35] and SP800-22 [36].

5.1 Statistical Random Number Tests

We use 100 random *key*s and 100 random *iv*s to produce 100 output keystream, and we test the randomness properties of the keystream by FIPS PUB 140-1 and SP800-22.

We put the output keystream of the proposed circuit under FIPS PUB 140-1 to test it randomness. We select 100 random *key*s and 100 random *iv*s to produce 100 output keystreams. Each keystream length is 20,000 bits. Finally, in Table 16, we present the FIPS PUB 140-1 randomness testing results in percentage. From Table 16, we can find the pass rate of each test result is 100.00%.

Table 16. Random Test Results under FIPS PUB 140-1

FIPS PUB 140-1 Tests	Pass rate under 20,000 bits/sample
Monobit Test	100 %
Poker Test	100 %
Runs Test	100 %
Long Run Test	100 %

5.2 Random Test Result under SP800-22

We select 100 random *key*s and 100 random *iv*s to produce 100 different keystreams. Each keystream output length is 10,000,000 bits. We list the testing results in Table 17. From the Table 17, we can find the pass rate of each test result is at least about 98%.

Table 17. Statistical test results of the proposed keystream generator under NIST SP800-22

Statistical tests	p-value	Pass rate under 10^6 bits/sample
Frequency	0.564465	99 %
Block frequency	0.824463	99 %
Runs	0.739989	100 %
Longest Runs of Ones	0.521068	99 %
Rank	0.624465	100 %
Discrete fourier transform	0.574463	100 %
Non-overlapping Templates Matching	0.649989	99 %
Overlapping templates matching	0.621068	98 %
Universal statistical	0.788428	98 %
Linear complexity	0.819180	100 %
Serial	0.708286	100 %
Approximate entropy	0.689007	99 %
Cumulative sums	0.645081	100 %
Random excursions	0.744741	98 %
Random excursions variant	0.755939	98 %

6 Conclusion

In this paper, we have studied the characteristics of FCSR based nonlinear circuits firstly and proposed FCSR based hybrid keystream generator. We apply DSG and lp-Geffe generator to increase the period and the linear complexity of sequence. We analyze the period and linear complexity and use some attacks to verity its security.

The experimental results show that the proposed stream cipher has large period, high linear complexity, good randomness, and the linear complexity is very close to its overall period. The linear complexity for the proposed stream cipher satisfies the security strength of AES. In addition, the pass rates of the proposed scheme are 100% for FIPS PUB 140-1 random test, and at least about 98% for NIST SP800-22 random test. The proposed scheme can resist known attack, such as algebraic attack and time-memory-data tradeoff attack and etc.

Acknowledgments

The author would like to express gratitude to Ms. Yu-Jing Du for her valuable assistance and to the anonymous referees whose insightful comments have greatly improved the quality of this paper.

References

[1] D. Ma, Y. Shi, A Lightweight Encryption Algorithm for Edge Networks in Software-Defined Industrial Internet of Things, *2019 IEEE 5th International Conference on Computer and* Communications *(ICCC)*, Chengdu, China, 2019, pp. 1489-1493.

[2] S. Roy, U. Rawat, J. Karjee, A Lightweight Cellular Automata Based Encryption Technique for IoT Applications, *IEEE Access*, Vol. 7, pp. 39782-39793, March, 2019.

[3] S.-H. Lee, P.-H. Shih, An Improved Gated System that Combines the Techniques of the Internet of Things for Community Security, *Journal of Internet Technology*, Vol. 23, No. 2, pp. 345-362, March, 2022.

[4] S.-T. Wu, A Secure Real-Time IoT Data Stream based on Improved Compound Coupled Map Lattices, *Applied Sciences*, Vol. 12, No. 17, Article No. 8489, pp. 1-19, September, 2022.

[5] M. Rana, Q. Mamun, R. Islam, Lightweight cryptography in IoT networks: A survey, *Future Generation Computer Systems*, Vol. 129, pp. 77-89, April, 2022.

[6] S. A. Jassim, A. K. Farhan, A Survey on Stream Ciphers for Constrained Environments, *2021 1st Babylon International Conference on Information Technology and Science (BICITS)*, Babil, Iraq, 2021, pp. 309-312.

[7] S. Y. Moon, J. H. Park, J. H. Park, Authentications for Internet of Things Security: Threats, Challenges and Studies, *Journal of Internet Technology*, Vol. 19, No. 2, pp. 349-358, March, 2018.

[8] H. Park, J. Kim, S. Lee, D. G. Duguma, I. You, lwEPSep: A Lightweight End-to-end Privacy-preserving Security Protocol for CTI Sharing in IoT Environments, *Journal of Internet Technology*, Vol. 22, No. 5, pp. 1069-1081, September, 2021.

[9] K. Zeng, C.-H. Yang, D.-Y. Wei, T. R. N. Rao, Pseudorandom Bit Generators in Stream-Cipher Cryptography, *Computer*, Vol. 24, No. 2, pp. 8-17, February, 1991.

[10] R. P. Prajapat, R. Bhadada, G. Sharma, Implementation of Enhanced A5/1 Stream Cipher and its Randomness Analysis by NIST Test Suite, *2021 IEEE International Symposium on Smart Electronic System*s (*iSES*), Jaipur, India, 2021, pp. 426-431.

[11] N. A. Mohandas, A. Swathi, A. R., A. Nazar, G. Sharath, A4: A Lightweight Stream Cipher, 2*020 5th International Conference on Communication and Electronics Systems* (*ICCES*), Coimbatore, India, 2020, pp. 573-577.

[12] S.-T. Wu, An Application of Keystream Using Cellular Automata for Image Encryption in IoT, *Journal of Internet Technology*, Vol. 24, No. 1, pp. 149-162, January, 2023.

[13] A. Klapper, M. Goresky, 2-Adic shift registers, in: R. A. Anderson (Eds.), *Fast Software Encryption*, *Cambridge SecurityWorkshop*. *Lecture Notes in Computer Science*, Vol. 809, Springer, New York, 1993, pp. 174-178.

[14] M. Mittelbach, A. Finger, Investigation of FCSR-based Pseudorandom Sequence Generators for Stream Ciphers, pp. 1-7, 2004. https://www.researchgate.net/publication/228853356_Investigation_of_FCSR-based_pseudorandom_sequence_generators_for_stream_ciphers

[15] S. Anand, G. V. Ramanan, Periodicity, Complementarity and Complexity of 2-adic FCSR Combiner Generators, *ASIACCS* '06: *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, Taipei, Taiwan, 2006, pp. 275-282.

[16] L. Zhang, C. Wang, T. Pei, Y. Zeng, Another Analysis of a Synchronizing Stream Cipher Combining LFSR and FCSR, *2019 International Conference on Networking and Network Applications* (*NaNA*), Daegu, Korea, 2019, pp. 309-312.

[17] Y. Zheng, X. Tang, D. He, L. Xu, Investigation on Pseudorandom Properties of FCSR Sequence, *Proceedings. 2005 International Conference on Communications, Circuits and Systems*, Hong Kong, China, 2005, pp. 66-70.

[18] M. Hell, T. Johansson, Breaking the F-FCSR-H Stream Cipher in Real Time, in: J. Pieprzyk (Ed.), *ASIACRYPT 2008*, *LNCS 5350*, Springer, Berlin, Heidelberg, 2008, pp. 557-569.

[19] M. Goresky, A. Klapper, Fibonacci and Galois Representations of Feedback-with-Carry Shift Registers, *IEEE Transactions on Information Theory*, Vol. 48, No. 11, pp. 2826-2836, November, 2002.

[20] F. Arnault, T. P. Berger, M. Minier, Some Results on FCSR Automata with Applications to the Security of FCSR-Based Pseudorandom Generators, *IEEE Transactions on Information Theory*, Vol. 54, No. 2, pp. 836-840, February, 2008.

[21] V. P. Shyrochin, I. V. Vasyltsov, B. Z. Karpinskij, Investigations of the Basic Component of FCSR-Generator, *IEEE International Workshop on Intelligent DATA Acquisition and Advanced Computing Systems*: *Technology and Applications*, Lviv, Ukraine, 2003, pp. 132-135.

[22] S.-L. Su, K.-M. Chiu, L.-C. Wuu, The Cryptanalysis of LFSR/FCSR Based Alternating Step Generator, *2006 International Conference on Computer Engineering and Systems*, Cairo, Egypt, 2006, pp. 228-231.

[23] T. Tian, W.-F. Qi, Linearity properties of binary FCSR sequences, *Designs, Codes and Cryptography*, Vol. 52, No. 3, pp. 249-262, September, 2009.

[24] F. Arnault, T. P. Berger, M. Minier, On the security of FCSR-based pseudorandom generators, *ECRYPT Network of Excellence – SASC Workshop*, 2007, Bochum, Germany, pp. 1-12.

[25] F. Arnault, T. P. Berger, C. Lauradoux, F-FCSR Stream ciphers, in: M. Robshaw, O. Billet (Eds.), *New Stream*

cipher Designs, *Lecture Notes in Computer Science,* Vol. 4986, Springer, Berlin, Heidelberg, 2008, pp. 170-178.

[26] E. Dawson, Cryptanalysis of Summation Generator, in: J. Seberry, Y. Zheng (Eds.), *Advances in Cryptology-AUSCRYPT'92*, *Lecture Notes in Computer Science*, Springer, Berlin, 1993, pp. 209-215.

[27] M.-H. Lim, B.-M. Goi, S. Lee, H. Lee, Hierarchical Dawson's Summation Generator, *International Conference on Convergence Information Technology*, Gwangju, Korea, 2007, pp. 1395-1401.

[28] W. Meier, O. Staffelbach, Correlation Properties of Combiners with Memory in Stream Ciphers, (Extended Abstract), *Advances in Cryptology – EUROCRYPT 90*, Aarhus, Denmark, 1990, pp. 204-213.

[29] B. Schneier, Applied Cryptography, *John Wiley and Sons*, Inc., 1995, pp. 402-413.

[30] A. Canteaut, Open problems related to algebraic attacks on stream ciphers, *International Workshop on Coding and Cryptography, WCC2005*, Bergen, Norway, 2005, pp. 120-134.

[31] N. T. Courtois, W. Meier, Algebraic Attacks on Stream Ciphers with Linear Feedback, in: E. Biham (Eds.), *Advances in Cryptology-EUROCRYPT 2003*. *Lecture Notes in Computer Science*, Vol. 2656, Springer, Berlin, Heidelberg, 2003, pp. 345-359.

[32] A. Biryukov, A. Shamir, Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers, in: T. Okamoto (Ed.): *Advances in Cryptology-ASIACRYPT 2000, Lecture Notes in Computer Science*, Vol. 1976, Springer, Berlin, Heidelberg, 2000, pp. 1-13.

[33] J. Hong, P. Sarkar, *Rediscovery of Time Memory Tradeoffs*, International Association for Cryptologic Research Eprint archive, 2005. http://eprint.iacr.org/2005/090

[34] Y. X. Yang, Correlation-Immunity of boolean Functions, *Electronics Letters*, Vol. 23 No. 25, pp. 1335-1336, December, 1987.

[35] National Institute of Standards and Technology, *Security Requirements for Cryptographic Modules*, Federal Information Processing Standards Publication 140-1, January, 1994.

[36] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, National Institute of Standards and Technology, Special Publication 800-22 Revision 1, August, 2008.

Biography

Shyi-Tsong Wu was born in Jiaoxi, Yilan, Taiwan. He received his Ph.D. in Electronic Engineering from National Taiwan University of Science and Technology, Taipei, Taiwan, in 2005. He is now a Professor at the Department of Electronic Engineering, National Ilan University, Yilan City, Taiwan. His research interests include IoT security and applications, cryptography, and electronic circuits.