


Article

A Key-Based Multi-Mode Clock-Controlled Stream Cipher for Real-Time Secure Communications of IoT

Shyi-Tsong Wu 

Department of Electronic Engineering, National Ilan University, Yilan City 26047, Taiwan; stwu@niu.edu.tw

Abstract: With the rapid development of the Internet and wireless communications, as well as the popularization of personal communication systems, the security of real-time communications is demanded. The efficient technology of stream ciphers can satisfy this requirement of security. In this paper, to enhance the security strength of stream ciphers, we design a key-based multi-mode clock-controlled stream cipher for real-time secure communications of the Internet of things (IoT). The proposed stream cipher is equipped with a multi-mode depending on the key. The different working modes are shipped with different encrypting circuits depending on the user's key. We analyze the period, the linear complexity, and use known attacks to verify the security strength of the proposed cipher. Compared with existing dual mode clock-controlled stream ciphers, the merits of our proposed cipher are its long period, high linear complexity, low hardware complex, low initialization clock, and simplicity in mode switching. Furthermore, the proposed cipher passes the FIPS PUB 140-1 and SP800-22 tests, obtaining at least 97.00%.

Keywords: stream cipher; hardware security; multi-mode; clock-controlled; key-based; IoT



Citation: Wu, S.-T. A Key-Based Multi-Mode Clock-Controlled Stream Cipher for Real-Time Secure Communications of IoT. *Electronics* **2023**, *12*, 1076. <https://doi.org/10.3390/electronics12051076>

Academic Editors: Ivan Ganchev, Máirtín O'Droma and Zhanlin Ji

Received: 3 January 2023

Revised: 16 February 2023

Accepted: 18 February 2023

Published: 21 February 2023



Copyright: © 2023 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Today, with the popularization of personal communication systems, such as cellular phones, PDAs, notebook computers, etc., and with the rapid development of the Internet of things (IoT), people can share information or transmit sensitive data by using these communication systems. There are a lot of information and communication services in the surrounding areas of human lives at present. These services are combined with various applications, for example, voice over Internet protocols, electronic commerce, distance learning, video conferencing, etc. These real-time streaming technologies provide convenience in terms of people's instant requirement for information and communication.

The IoT provides convenience in terms of information transmission, but it is unsafe when transmitting unencrypted data via the openness of wireless communications. It is easy to be overheard without protection on such systems. For this reason, the most effective method is to encrypt the transmitted content to prevent the information from being directly known by eavesdropping. Even if the transmitted content is overheard from the channel, they will be nonsensical data. In order to achieve secure communications, cryptography is applied to protect privacy and to avoid fraud in secure communications in the IoT.

The security of the IoT can be achieved by implementing Secure Shell (SSH) and Transport Layer Security (TLS) protocols. However, they have heavy overheads that are not suitable for the resource-constrained environment of the IoT [1]. There are three basic popular communication protocols at the IoT application level; they are the CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport), and the XMPP (Advanced Message Queuing Protocol). The MQTT protocol is the most widely used protocol for the communication of these devices in IoT systems due to its low resource requirements [2]. Some cryptosystems have been proposed in IoT systems that communicate using the MQTT protocol, but they have not been widely accepted because of their performances [3,4]. For confidentiality, the cipher of an asymmetric cryptosystem, e.g., RSA, ECC,

etc., is not suitable for the IoT, due to the computational load. In symmetric cryptosystems, stream ciphers outperform block ciphers because of their simple encrypting operation.

The stream cipher is a class of symmetric encryption algorithms, and it is generally much faster than block ciphers, so stream ciphers are widely used in digital communications and real-time transmissions. For the security demands of real-time communications, stream ciphers are used to meet the necessary requirements [5–8]. For example, the stream cipher A5/1 supports the confidentiality of mobile communications [9]. Similarly, in real-time communications of the IoT, security efficiency can benefit from using stream ciphers.

At the core of stream ciphers is the keystream generator. One of the basic structures of keystream generators is the linear feedback shift register (LFSR) [10]. For the attack of the Berlekamp-Massey algorithm, the output of the sequences of LFSRs is straightforwardly predictable. To resist this attack and spoil the linearity properties of LFSRs, there are three basic schemes that can be achieved, which include: a nonlinear combining function on the outputs of several LFSRs, a nonlinear filtering function on the contents of a single LFSR, and using the output of one (or more) LFSRs to control the clock of one or more other LFSRs, which are the clock-controlled LFSRs. All of these schemes require a nonlinear function to combine the outputs of LFSRs or control the input clock for clock-controlled LFSRs [10–14]. The clock-controlled based stream cipher A5/1 uses the nonlinear majority function as the nonlinear function to promote its security. Erguler and et al. proposed a clock-controlled stream cipher with dual modes, and that has two different clocking mechanisms to provide security enhancements [15].

Regarding the hardware of stream ciphers, a cipher with a multiple working nonlinear circuit is a strategy by which to gain the security strength of the output keystream. To further the security strength of stream ciphers, in this paper, we design a key-based multi-mode clock-controlled stream cipher for real-time secure communications using the IoT. The cipher is equipped with a multi cipher mode, depending on the secret key. The different modes are shipped with different encryption circuits depending on the user's session key. We analyze the period, linear complexity, evaluate the randomness, and use known attacks to verify the security strength of the cipher. From the experimental results, the proposed cipher passes the FIPS PUB 140-1 and SP800-22 tests, attaining at least 97.00%. The contributions of this study can be briefly stated as follows:

- The proposed scheme employs multiple working modes depending on the user's session key.
- The multiple working modes of the different working circuits include different nonlinear selecting functions and different nonlinear output combining functions.
- All of the nonlinear selecting functions and output combining functions provide the balance correlation probability. It prevents weakness for attackers to break through the stream cipher.
- The proposed scheme is one of hardware security, and is easy to implement using hardware.

This paper is organized as follows. Section 2 introduces the related research regarding stream ciphers. Then, we present our proposed scheme, describe each component of the proposed stream cipher, and specify the details of the design in Section 3. In Section 4, we consider statistical properties and some attacks with respect to our design. In addition, we present the results regarding the period and linear complexity of our scheme. Section 5 describes the test criteria and the experimental results for our proposed stream cipher. We use the Federal Information Processing Standards Publication 140-1 (FIPS PUB 140-1) [16] and the Special Publication 800-22 (SP800-22) [17] to perform the statistical tests for our scheme. Finally, we provide the conclusions in Section 6.

2. Preliminaries

The basic design of a stream cipher requires a short key and expands it into a binary pseudorandom number sequence. This sequence is also called the keystream. The keystream is XORed to the plaintext and generates the ciphertext. Similarly, the same

keystream is used to decrypt through XORing with the cipher to recover the original plain-text [18]. Therefore, the keystream generator plays an important role in the research of stream ciphers. Generally speaking, the keystream generator can be composed of the finite state machine (FSM) and the output function. Among the design of many stream ciphers, the LFSR is the most common class of FSM due to its simplicity, speed of implementation in hardware, and the fact that it provides sequences with good statistical properties.

2.1. Linear Feedback Shift Register

The Linear Feedback Shift Register (LFSR) can be implemented in two ways. One is the Fibonacci structure, and the other is the Galois structure. The Fibonacci structure of a LFSR consists of a simple shift register and additive operations. Figure 1 shows the Fibonacci structure of a LFSR. At the time t , the LFSR of length L consists of L stages $S_{0+t}, \dots, S_{L-1+t}$, where $t \geq 0$. Each stage is a D-type flip-flop and stores one bit. The output position of each D-type flip-flop that affects the next state is called the tap. The taps are sequentially XORed and then fed back into the leftmost bit. The Fibonacci structure of a LFSR can use a feedback polynomial to record the structure. We call the polynomial the connection polynomial $f(x)$. It is defined as follows:

$$f(x) = 1 + \sum_{i=1}^L C_i x^i = 1 + C_1 x + C_2 x^2 + \dots + C_L x^L \tag{1}$$

where L is called the degree of the connection polynomial and the C_i is called the feedback coefficient. In general, the additive operations are included in module 2. The feedback coefficient C_i ($1 \leq i \leq L$) that is not zero is the tap of the connection polynomial. For any feedback coefficient, C_i is either 0, meaning “no connection”, or 1, meaning it is sequentially XORed with the other taps and then fed back into the leftmost bit. Furthermore, the connection polynomial is called a characteristic polynomial.

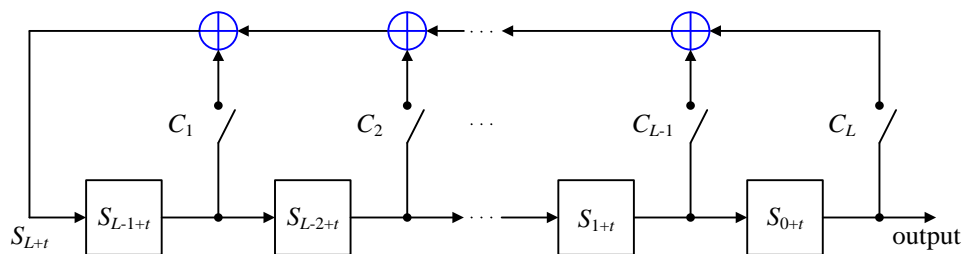


Figure 1. The Fibonacci structure of a LFSR.

The Galois configuration of the LFSR is illustrated in Figure 2. It also consists of a shift register of length L . The Galois structure of a LFSR of length L can also use a feedback polynomial to record the structure of the LFSR. It is defined by the characteristic polynomial $p(x)$:

$$p(x) = 1 + \sum_{i=1}^{L-1} p_i x^i + x^L = 1 + p_1 x + p_2 x^2 + \dots + p_{L-1} x^{L-1} + x^L \tag{2}$$

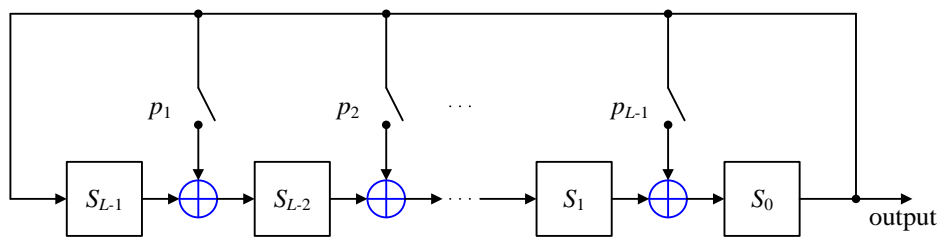


Figure 2. The Galois structure of a LFSR.

The Galois LFSR does not concatenate every tap to produce the new input, but each tap is parallel to compute the new input bits. If the feedback polynomial of the LFSR is a primitive polynomial and the initial state of the LFSR is not all zero, the period of the output sequence of the LFSR is at most equal to $2^L - 1$. Such a LFSR produces a sequence with the longest period, and we call the sequence a maximal sequence or m-sequence.

2.2. LFSR-Based Stream Cipher

In this subsection, we introduce some LFSR-based stream ciphers. The A5/1 is a clock-controlled, LFSR-based stream cipher which is used for encrypting air transmissions in the GSM standard. The diagram of an A5/1 stream cipher is illustrated in Figure 3. It is composed of three LFSRs with different lengths and primitive feedback polynomials. Each LFSR is shifted, using clock cycles that are determined by a majority function. The major issue with A5/1 security is the short period problem. The cipher operation is based on three LFSRs, R_1 , R_2 , and R_3 , of lengths 19, 22, and 23 bits, respectively. The experiment shows that the period of A5/1 is equal to $(4/3)(2^{23} - 1)$ [19]. Another basic issue is the collision problem. It means that A5/1 may result in the same keystream when the LFSR's different seeds are used.

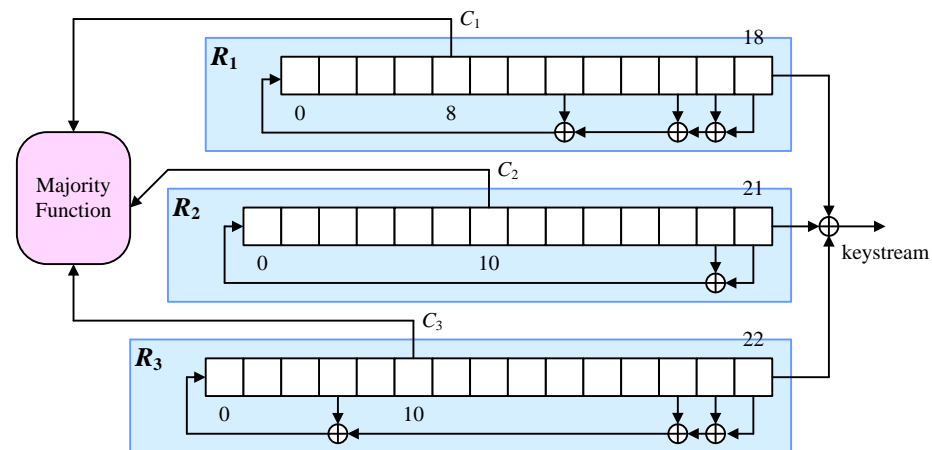


Figure 3. The diagram of A5/1 stream cipher.

To reduce the weaknesses of A5/1, some solutions have been proposed in the literature. Erguler and Anarim, in 2005, proposed an A5/2 algorithm that consists of four registers and modified the clocking control mechanism to promote its security strength [20]. In 2006, Erguler and Erguler proposed a LFSR-based CCDM (Clock-Controlled with Dual Mode) stream cipher with a dual operating mode [15]. It is a novel clock-controlled stream cipher with Dual Mode, which is based on irregular clocking and operates with two different modes. The CCDM-Mode I merges eight 4×16 S-boxes of DES [21] for the keystream generation, and the CCDM-Mode II operates with a mutual clock-control mechanism. Zakaria and et al., in 2011, presented two modifications of A5/1 [22]. One is the changing of the original primitive polynomials of LFSR in A5/1. The second modification added two LFSRs and proposed five LFSRs in total. The scheme passed the randomness tests. Yohana, in 2015, improved A5/1 by means of a unit delay to increase the period of the keystream [23]. The keystream of the proposed methodology succeeded in randomness tests. In 2019, Sadkhan and Hamza added a fourth register and applied a new filtration function to A5/1 on each register to strengthen the original linear combination function and XOR operation [24]. The authors implemented hardware and made the generator more secure.

Sadkhan and Reza, in 2017, proposed a new method to investigate the best structure for the nonlinear combining function [10]. Based on LFSR, to build a nonlinear combination function consisting of n levels, the designers built it within the program and changed part of the chosen nonlinear combination function every time to observe the results. The

scheme required the support of a powerful computer. In 2021, Prajapat and et al. proposed a security enhancement of the A5/1 stream cipher in GSM communications [14]. The scheme reduced the non-linear combinational generator (NLFSRs), reused the 32 bits of SRES generated by the A3 algorithm, and finally, combined the output stream with the remaining 32-bit of CGI (Cell Global Identity). From the results of the NIST Statistical Test, the scheme achieved enhanced security. In 2022, Kopparthi and et al. proposed a pseudorandom number generator based on a digital piecewise linear chaotic map with perturbation [25]. The basic algorithm for the pseudorandom number generator is based on chaos, rather than LFSR. The scheme increases the period of random sequence and succeeds in increasing security; its requirements in terms of hardware costs were also increased.

3. The Proposed Scheme

To promote the randomness and the chaos of the output keystream, we propose a multi-mode keystream generator. The different working modes are dependent on the input key. In this section, we introduce the proposed stream cipher. We apply a clock-controlled stream cipher and propose a key-based multi-mode clock-controlled stream cipher. The structure of the proposed cipher is based on multi-LFSR and is equipped with multiple cipher modes to enhance security. The mode selection is dependent on the secret key bits. For each cipher mode, the output sequences have a large period and a high linear complexity. In the following subsections, we present our proposed scheme and describe each component of the proposed stream cipher. Furthermore, we specify the details of the design.

3.1. Keystream Generator

To match the AES, the proposed key-based multi-mode clock-controlled stream cipher takes a 128-bit secret key denoted K_i , $0 \leq i \leq 127$, and a 128-bit initialization vector denoted IV_i , $0 \leq i \leq 127$, as its inputs. The cipher consists of four main building blocks, namely LFSRs, a clock controller, a mode controller, and an output generator. An overview of the blocks used in the stream cipher is illustrated in Figure 4.

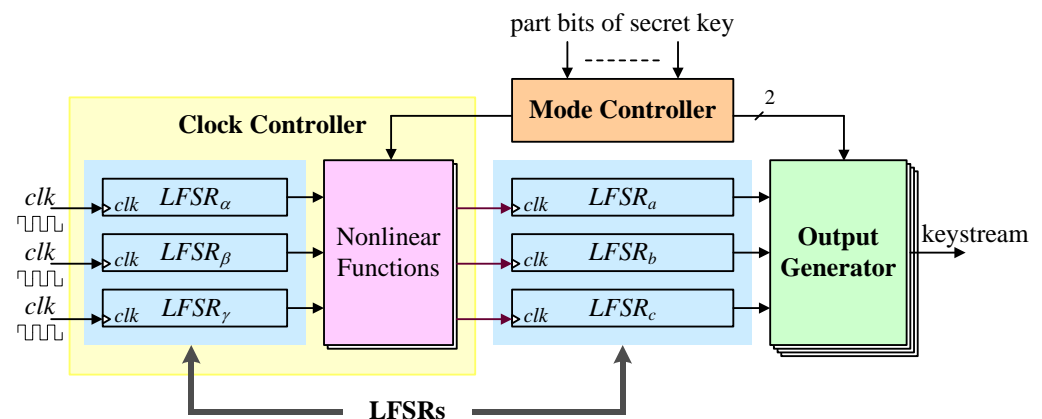


Figure 4. The block diagram of keystream generator.

The size of the internal state of the proposed keystream generator is 318 bits, which consists of six LFSRs: $LFSR_\alpha$, $LFSR_\beta$, $LFSR_\gamma$, $LFSR_a$, $LFSR_b$, and $LFSR_c$, respectively. The main work of the $LFSR_a$, $LFSR_b$, $LFSR_c$, and output generator is to produce the keystream. The main work of the $LFSR_\alpha$, $LFSR_\beta$, $LFSR_\gamma$, and the nonlinear functions is to control when $LFSR_a$, $LFSR_b$ or $LFSR_c$, will be shifted. There are two clock operation modes in the clock controller and four output sequence generators in the output generator. They organize eight cipher operation modes. The responsibility of the mode controller is the selection of which cipher mode will operate; where the input of the mode controller is the part of secret key bits. The rightmost bits of $LFSR_a$, $LFSR_b$, and $LFSR_c$ are inputted to the output generator to produce the keystream.

3.2. The LFSRs

The size of the internal state of the keystream generator is 318 bits, which consists of six LFSRs, namely $LFSR_\alpha$, $LFSR_\beta$ and $LFSR_\gamma$, $LFSR_a$, $LFSR_b$, and $LFSR_c$. The underlying $LFSR_\alpha$, $LFSR_\beta$, $LFSR_\gamma$, $LFSR_a$, $LFSR_b$, and $LFSR_c$ are six maximum-length LFSRs of lengths 31, 17, 13, 61, 89, and 107, respectively. The primitive feedback polynomials of the registers are defined as follows [26]:

$$LFSR_\alpha: P_\alpha(x) = x^{31} + x^{25} + x^{23} + x^8 + 1 \tag{3}$$

$$LFSR_\beta: P_\beta(x) = x^{17} + x^{16} + x^{12} + x^4 + 1 \tag{4}$$

$$LFSR_\gamma: P_\gamma(x) = x^{13} + x^8 + x^5 + x^3 + 1 \tag{5}$$

$$LFSR_a: P_a(x) = x^{61} + x^{59} + x^{52} + x^{47} + x^{38} + x^{33} + 1 \tag{6}$$

$$LFSR_b: P_b(x) = x^{89} + x^{81} + x^{68} + x^{31} + x^{21} + x^{18} + 1 \tag{7}$$

$$LFSR_c: P_c(x) = x^{107} + x^{89} + x^{84} + x^{40} + x^{29} + x^{23} + 1 \tag{8}$$

If the length of a LFSR is L and the LFSR is using the Fibonacci structure, it must be repeated L times to update each content of the register during initialization. In order to increase the efficiency of key initialization, we select different length LFSRs instead of a single LFSR. We use the Galois structure of a LFSR to speed up the operation of initialization.

3.3. Mode Controller

The responsibility of the mode controller is the selection of which cipher mode will operate in the system. We built a mode controller module, as shown in Figure 5. There are five input signals and three output signals in the mode controller, where the inputs of the mode controller are K_0, K_1, K_2, K_3 , and K_4 of the secret key and the outputs are m_0, m_1 , and m_2 .

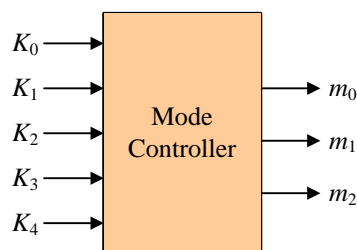


Figure 5. The module of the mode controller.

The signal m_0 is taken to control which clock operation mode will be operated, and it is generated by XORing K_0, K_1, K_2, K_3 , and K_4 . Here, \oplus denotes logic XOR, the Boolean function of m_0 is given by:

$$m_0 = K_0 \oplus K_1 \oplus K_2 \oplus K_3 \oplus K_4 \tag{9}$$

Another two output signals, m_1 and m_2 , of the mode controller are taken to control which output sequence generator will operate. We use one of four quasigroups of Edon80 for a part of the mode controller. Since the four quasigroups are suitable for implementation, no hidden weaknesses can be imposed [27]. The quasigroup is shown in Table 1. We let X be K_1K_0 and let Y be K_3K_2 . The value X defines the row r of the quasigroup, and the value Y defines the column c . For the data on the (r, c) in Table 1, each r and c has a two-bit length, respectively. The result of (r, c) is a two-bit length, too. Then, we take the result of (r, c) to control which output sequence generator will operate.

Table 1. The quasigroup table.

Nr. 61		X				
		0	1	2	3	
Y	0	0	2	1	3	
	1	2	1	3	0	
	2	1	3	0	2	
	3	3	0	2	1	

In the hardware implementation, to reduce the gate numbers in order to store the quasigroup table, we use the Boolean functions and map the quasigroup table to logic [28]. Here, \oplus denotes logic XOR and \cdot denotes logic AND. The output signals m_1 and m_2 are computed by the Boolean functions as follows:

$$m_1 = K_0 \cdot K_2 \oplus K_1 \oplus K_3 \tag{10}$$

$$m_2 = K_0 \oplus K_2 \tag{11}$$

Finally, we obtain the logic circuit to implement the mode controller. The circuit of the mode controller is shown in Figure 6.

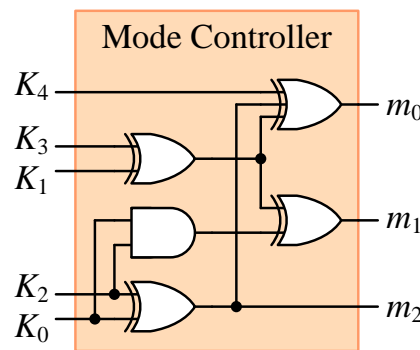


Figure 6. The circuit of the mode controller.

3.4. Clock Controller

The main work of the $LFSR_a$, $LFSR_b$, $LFSR_c$, and output generator is to produce the keystream. The respective rightmost bits of $LFSR_a$, $LFSR_b$, and $LFSR_c$ are inputted into the output generator to produce the keystream, but the respective clock pulses of these three LFSRs are dependent on the clock controller. There are two clock operation modes in the clock controller, which are mode 0 and mode 1. Which outputs of the two modes will be outputted is determined according to the output signal m_0 of the mode controller. If $m_0 = 0$, then the outputs of mode 0 are outputted. On the other hand, if $m_0 = 1$, then the outputs of mode 1 are outputted.

The clock controller consists of three LFSRs and some nonlinear Boolean functions. The three LFSRs are $LFSR_\alpha$, $LFSR_\beta$, and $LFSR_\gamma$, and the respective rightmost bits of these LFSRs are inputted into nonlinear Boolean functions, and then the outputs of the nonlinear Boolean functions are used to control whether $LFSR_a$, $LFSR_b$, or $LFSR_c$ will be shifted. It means the clock controller controls respective clock pulses of the $LFSR_a$, $LFSR_b$, and $LFSR_c$. We describe how the clock controller controls the respective clock pulses of the $LFSR_a$, $LFSR_b$, and $LFSR_c$.

First, the respective rightmost bits of the $LFSR_\alpha$, $LFSR_\beta$, and $LFSR_\gamma$ are denoted by α_{0t} , β_{0t} , and γ_{0t} , respectively, at time t . They are inputted into the f_{maj} function and the f_{and} function, which are given by:

$$f_{maj} = (\alpha_{0t} \cdot \beta_{0t}) + (\alpha_{0t} \cdot \gamma_{0t}) + (\beta_{0t} \cdot \gamma_{0t}) \tag{12}$$

$$f_{and} = \alpha_{0t} \cdots \beta_{0t} \cdots \gamma_{0t} \tag{13}$$

where, +denotes logic OR and \cdots denotes logic AND. For the mode of $m_0 = 0$, then f_{maj} function is selected. If $\alpha_{0t} = f_{maj}$, then $LFSR_a$ is shifted. While $\beta_{0t} = f_{maj}$ and $\gamma_{0t} = f_{maj}$, then $LFSR_b$ and $LFSR_c$ are shifted, respectively. Similarly, for another mode of $m_0 = 1$, the f_{and} function will be selected. If $\alpha_{0t} = f_{and}$, then $LFSR_a$ is shifted. While $\beta_{0t} = f_{and}$ and $\gamma_{0t} = f_{and}$, then $LFSR_b$ and $LFSR_c$ are shifted, respectively. According to the above, we define clk_{ai} , clk_{bi} , and clk_{ci} as the clocking condition of $LFSR_a$, $LFSR_b$, and $LFSR_c$, respectively, where $i \in \{0, 1\}$. For the mode of $m_0 = 0$, then clk_{a0} , clk_{b0} , and clk_{c0} are selected. If $clk_{a0} = 1$, then $LFSR_a$ is shifted. While $clk_{b0} = 1$ and $clk_{c0} = 1$, then $LFSR_b$ and $LFSR_c$ are shifted, respectively. On the other hand, for the mode of $m_0 = 1$, then clk_{a1} , clk_{b1} , and clk_{c1} are selected. If $clk_{a1} = 1$, then $LFSR_a$ is shifted. While $clk_{b1} = 1$ and $clk_{c1} = 1$, then $LFSR_b$ and $LFSR_c$ are shifted, respectively.

Next, we build two truth tables for the clocking conditions of the three LFSRs, $LFSR_a$, $LFSR_b$, and $LFSR_c$, which are shown in Tables 2 and 3. Then, we can simplify clk_{a0} , clk_{b0} , and clk_{c0} into Boolean functions for clock operation mode 0, as well as clk_{a1} , clk_{b1} , and clk_{c1} for clock operation mode 1. The simplified Boolean functions of the two clock operation modes are shown as follows:

Table 2. Clocking condition under $m_0 = 0$.

$LFSR_\alpha$	$LFSR_\beta$	$LFSR_\gamma$	f_{maj}	$LFSR_a$	$LFSR_b$	$LFSR_c$
α_{0t}	β_{0t}	γ_{0t}		clk_{a0}	clk_{b0}	clk_{c0}
0	0	0	0	1	1	1
0	0	1	0	1	1	0
0	1	0	0	1	0	1
0	1	1	1	0	1	1
1	0	0	0	0	1	1
1	0	1	1	1	0	1
1	1	0	1	1	1	0
1	1	1	1	1	1	1

Table 3. Clocking condition under $m_0 = 1$.

$LFSR_\alpha$	$LFSR_\beta$	$LFSR_\gamma$	f_{and}	$LFSR_a$	$LFSR_b$	$LFSR_c$
α_{0t}	β_{0t}	γ_{0t}		clk_{a1}	clk_{b1}	clk_{c1}
0	0	0	0	1	1	1
0	0	1	0	1	1	0
0	1	0	0	1	0	1
0	1	1	0	1	0	0
1	0	0	0	0	1	1
1	0	1	0	0	1	0
1	1	0	0	0	0	1
1	1	1	1	1	1	1

Clock operation mode 0:

$$clk_{a0} = \bar{\alpha}_{0t}\bar{\beta}_{0t} + \alpha_{0t}\gamma_{0t} + \beta_{0t}\bar{\gamma}_{0t} \tag{14}$$

$$clk_{b0} = \bar{\beta}_{0t}\bar{\gamma}_{0t} + \bar{\alpha}_{0t}\gamma_{0t} + \alpha_{0t}\beta_{0t} \tag{15}$$

$$clk_{c0} = \bar{\beta}_{0t}\bar{\gamma}_{0t} + \alpha_{0t}\gamma_{0t} + \bar{\alpha}_{0t}\beta_{0t} \tag{16}$$

Clock operation mode 1:

$$clk_{a_1} = \bar{\alpha}_{0t} + \beta_{0t}\gamma_{0t} \tag{17}$$

$$clk_{b_1} = \bar{\beta}_{0t} + \alpha_{0t}\gamma_{0t} \tag{18}$$

$$clk_{c_1} = \bar{\gamma}_{0t} + \alpha_{0t}\beta_{0t} \tag{19}$$

At each clock cycle, only one of the two clock operation modes will determine whether the $LFSR_a$, $LFSR_b$, or $LFSR_c$ are shifted or not, since the signal m_0 is taken to switch which outputs of clock operation mode will output. Furthermore, each rightmost bit of $LFSR_a$, $LFSR_b$, and $LFSR_c$ will input to the output generator to generate the keystream.

3.5. Output Generator

The output generator takes sequence a_{0t} , b_{0t} , and c_{0t} as its input, which are the respective rightmost bits of $LFSR_a$, $LFSR_b$, and $LFSR_c$ at time t . There are four output sequence generators in the output generator. The sequence outputs are denoted by z_{0t} , z_{1t} , z_{2t} and z_{3t} . According to the selected mode, the output signals m_1 and m_2 of the mode controller are taken to control which output sequence generator will operate. For example, if $m_2 = 0$ and $m_1 = 0$, then output sequence generator 0 (OSG₀) will be operated, and so on. Table 4 lists the operating conditions under m_2 and m_1 .

Table 4. The operating condition under m_2 and m_1 .

m_2	m_1	Operating OSG _{<i>i</i>}
0	0	OSG ₀
0	1	OSG ₁
1	0	OSG ₂
1	1	OSG ₃

The output sequence generators are given as follows:

Output Sequence Generator 0 (OSG₀): In this generator, we use exclusive-or operation with a_{0t} , b_{0t} , and c_{0t} to produce keystream z_{0t} , which is given by:

$$z_{0t} = a_{0t} \oplus b_{0t} \oplus c_{0t} \tag{20}$$

The output-inputs correlation probability of OSG₀ is demonstrated in Table 5. All correlations between inputs and output are both 1/2.

Table 5. Correlation probability of OSG₀.

a_{0t}	b_{0t}	c_{0t}	z_{0t}	Correlation Probability
0	0	0	0	Output-Inputs: prob($z_{0t} = a_{0t}$) = 1/2 prob($z_{0t} = b_{0t}$) = 1/2 prob($z_{0t} = c_{0t}$) = 1/2
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

Output Sequence Generator 1 (OSG₁): In this generator, we use two Dawson’s summation generators to produce the keystream. The diagram of Dawson’s summation generator (DSG) is shown in Figure 7 and the functions are defined as follows [29]:

$$z_j = a_j \oplus b_j \oplus c_{j-1} \tag{21}$$

$$c_j = b_j \oplus (a_j \oplus b_j) \cdot c_{j-1} \tag{22}$$

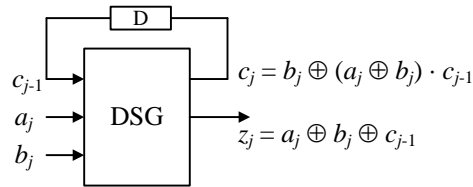


Figure 7. The Dawson’s summation generator.

Here $a_j, b_j,$ and c_{j-1} denote the input sequences of DSG, and c_{j-1} is c_j delayed one clock. The initial state of the bit, c_{j-1} , is defined to be zero. In addition, the DSG has high resistance against correlation attacks [29], due to all its output-inputs correlation probabilities being $1/2$, as demonstrated in Table 6.

Table 6. Correlation probability of DSG.

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

In Figure 1, we use two DSGs to generate keystream z_{1t} . The diagram of OSG₁ is shown in Figure 8. The functions can be written as follows:

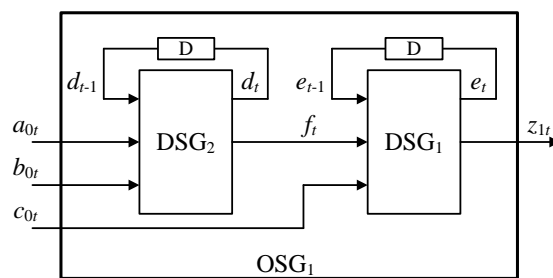


Figure 8. The output sequence generator 1.

DSG₁:

$$z_{1t} = f_t \oplus c_{0t} \oplus e_{t-1} \tag{23}$$

$$e_t = c_{0t} \oplus (f_t \oplus c_{0t}) \cdot \dots \cdot e_{t-1} \tag{24}$$

DSG₂:

$$f_t = a_{0t} \oplus b_{0t} \oplus d_{t-1} \tag{25}$$

$$d_t = b_{0t} \oplus (a_{0t} \oplus b_{0t}) \cdot \dots \cdot d_{t-1} \tag{26}$$

The initial state of the bits, f_{t-1} and e_{t-1} , are defined to be zero and the output function of the OSG₁ can be sorted as follows:

$$z_{1t} = a_{0t} \oplus b_{0t} \oplus c_{0t} \oplus d_{t-1} \oplus e_{t-1} \tag{27}$$

$$d_t = b_{0t} \oplus (a_{0t} \oplus b_{0t}) \cdot d_{t-1} \tag{28}$$

$$e_t = c_{0t} \oplus (a_{0t} \oplus b_{0t} \oplus c_{0t} \oplus d_{t-1}) \cdot \dots \cdot e_{t-1} \tag{29}$$

We also sort the respective correlation probabilities of the output-inputs, output- d_t and output- e_t , of OSG₁. All of them are equal to 1/2, which is demonstrated in Table 7.

Table 7. Correlation probabilities of OSG₁.

a_{0t}	b_{0t}	c_{0t}	d_{t-1}	e_{t-1}	d_t	e_t	z_{1t}	Correlation Probability
0	0	0	0	0	0	0	0	
0	0	0	0	1	0	0	1	
0	0	0	1	0	0	0	1	
0	0	0	1	1	0	1	0	
0	0	1	0	0	0	1	1	
0	0	1	0	1	0	0	0	
0	0	1	1	0	0	1	0	
0	0	1	1	1	0	1	1	
0	1	0	0	0	1	0	1	
0	1	0	0	1	1	1	0	
0	1	0	1	0	0	0	0	
0	1	0	1	1	0	0	1	
0	1	1	0	0	1	1	0	
0	1	1	0	1	1	1	1	
0	1	1	1	0	0	1	1	
0	1	1	1	1	0	0	0	
1	0	0	0	0	0	0	1	
1	0	0	0	1	0	1	0	
1	0	0	1	0	1	0	0	
1	0	0	1	1	1	0	1	
1	0	1	0	0	0	1	0	
1	0	1	0	1	0	1	1	
1	0	1	1	0	1	1	1	
1	0	1	1	1	1	0	0	
1	1	0	0	0	1	0	0	
1	1	0	0	1	1	0	1	
1	1	0	1	0	1	0	1	
1	1	0	1	1	1	1	0	
1	1	1	0	0	1	1	1	
1	1	1	0	1	1	0	0	
1	1	1	1	0	1	1	0	
1	1	1	1	1	1	1	1	

Output-Inputs:
 $\text{prob}(z_{1t} = a_{0t}) = 1/2$
 $\text{prob}(z_{1t} = b_{0t}) = 1/2$
 $\text{prob}(z_{1t} = c_{0t}) = 1/2$
 $\text{prob}(z_{1t} = d_{t-1}) = 1/2$
 $\text{prob}(z_{1t} = e_{t-1}) = 1/2$

Output- d_t :
 $\text{prob}(z_{1t} = d_t) = 1/2$

Output- e_t :
 $\text{prob}(z_{1t} = e_t) = 1/2$

Output Sequence Generator 2 (OSG₂): The OSG₂ is shown in Figure 9. In this generator, we build a hybrid carry bit g_t and merge it with DSG to generate keystream z_{2t} , which can be defined as follows:

$$z_{2t} = a_{0t} \oplus b_{0t} \oplus c_{0t} \oplus g_{t-1} \tag{30}$$

$$g_t = g_{t-1} \oplus (a_{0t} \oplus g_{t-1} \oplus c_{0t}) \cdot \cdot \cdot b_{0t} \tag{31}$$

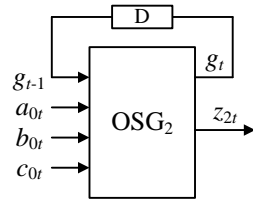


Figure 9. The output sequence generator 2.

The initial state of the bits, g_{t-1} , is defined to be zero. All of the correlation probabilities of the output-inputs and the output- g_t of OSG₂ are equal to 1/2, which are demonstrated in Table 8.

Table 8. Correlation probability of OSG₂.

a_{0t}	b_{0t}	c_{0t}	g_{t-1}	g_t	z_{2t}	Correlation Probability
0	0	0	0	0	0	
0	0	0	1	1	1	
0	0	1	0	0	1	
0	0	1	1	1	0	
0	1	0	0	0	1	
0	1	0	1	0	0	
0	1	1	0	1	0	
0	1	1	1	1	1	
1	0	0	0	0	1	
1	0	0	1	1	0	
1	0	1	0	0	0	
1	0	1	1	1	1	
1	1	0	0	1	0	
1	1	0	1	1	1	
1	1	1	0	0	1	
1	1	1	1	0	0	

Output-Inputs:
 $\text{prob}(z_{2t} = a_{0t}) = 1/2$
 $\text{prob}(z_{2t} = b_{0t}) = 1/2$
 $\text{prob}(z_{2t} = c_{0t}) = 1/2$
 $\text{prob}(z_{2t} = g_{t-1}) = 1/2$

Output- g_t :
 $\text{prob}(z_{2t} = g_t) = 1/2$

Output Sequence Generator 3 (OSG₃): The OSG₃ is shown in Figure 10. In this generator, we establish another hybrid carry bit h_t , and output functions then merge them with DSG to generate keystream z_{3t} ; which can be defined as follows:

$$z_{3t} = a_{0t} \oplus b_{0t} \oplus c_{0t} \oplus h_{t-1} \tag{32}$$

$$h_t = a_{0t} \cdot (b_{0t} \oplus c_{0t} \oplus h_{t-1}) \oplus b_{0t} \oplus c_{0t} \tag{33}$$

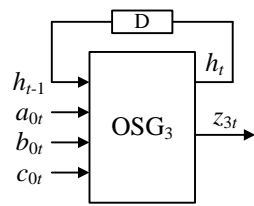


Figure 10. The output sequence generator 3.

The initial state of the bits, h_{t-1} , is defined to be 0. The correlation probabilities of the output-inputs and the output- h_t of OSG_3 are both $1/2$, which are demonstrated in Table 9.

Table 9. Correlation probability of OSG_3 .

a_{0t}	b_{0t}	c_{0t}	h_{t-1}	h_t	z_{3t}	Correlation Probability
0	0	0	0	0	0	
0	0	0	1	0	1	
0	0	1	0	1	1	
0	0	1	1	1	0	
0	1	0	0	1	1	
0	1	0	1	1	0	
0	1	1	0	0	0	
0	1	1	1	0	1	
1	0	0	0	0	1	
1	0	0	1	1	0	
1	0	1	0	0	0	
1	0	1	1	1	1	
1	1	0	0	0	0	
1	1	0	1	1	1	
1	1	1	0	0	1	
1	1	1	1	1	0	

Output-Inputs:
 $\text{prob}(z_{3t} = a_{0t}) = 1/2$
 $\text{prob}(z_{3t} = b_{0t}) = 1/2$
 $\text{prob}(z_{3t} = c_{0t}) = 1/2$
 $\text{prob}(z_{3t} = h_{t-1}) = 1/2$

Output- h_t :
 $\text{prob}(z_{3t} = h_t) = 1/2$

3.6. Key/IV Setup

The inputs of the keystream generator are called seeds. The requirements of these seeds are that they must be random and unpredictable before generating the keystream. For this reason, we must use the key initialization procedure to perform the requirement. In this subsection, we describe the computation of the initial inner state before starting the keystream generation. First, part bits of the secret key are collaterally loaded into the 318-bit initial state of the cipher. Then, the remaining bits of the secret key and the 128-bit initialization vector are fed into the 318-bit initial state of the cipher using the key initialization procedure. We generalize the Key/IV Setup into two phases, the initial filling phase and the key initialization procedure phase. It works as follows.

3.6.1. Initial Filling Phase

The 128-bit secret key K is denoted by $K = K_0, \dots, K_{127}$ and the 128-bit initialization vector IV is denoted by $IV = IV_0, \dots, IV_{127}$. The internal states of $LFSR_\alpha$, $LFSR_\beta$, and $LFSR_\gamma$ are denoted by α_i , β_i , and γ_i , respectively, where:

$$\alpha_i, 0 \leq i \leq 30 \tag{34}$$

$$\beta_i, 0 \leq i \leq 16 \tag{35}$$

$$\gamma_i, 0 \leq i \leq 12 \tag{36}$$

The internal states of $LFSR_a$, $LFSR_b$, and $LFSR_c$ are denoted by a_i , b_i , and c_i , respectively, where:

$$a_i, 0 \leq i \leq 60 \tag{37}$$

$$b_i, 0 \leq i \leq 88 \tag{38}$$

$$c_i, 0 \leq i \leq 106 \tag{39}$$

The initial filling of each LFSR is carried out as follows:

$$(a_{60}, a_{59}, \dots, a_1, a_0) \leftarrow (K_{59}, K_{58}, \dots, K_0, 1) \tag{40}$$

$$(\alpha_{30}, \alpha_{29}, \dots, \alpha_1, \alpha_0) \leftarrow (K_{89}, K_{88}, \dots, K_{60}, 1) \tag{41}$$

$$(b_{88}, b_{87}, \dots, b_1, b_0) \leftarrow (K_{87}, K_{86}, \dots, K_0, 1) \tag{42}$$

$$(\beta_{16}, \beta_{15}, \dots, \beta_1, \beta_0) \leftarrow (K_{105}, K_{104}, \dots, K_{88}, 1) \tag{43}$$

$$(c_{106}, c_{105}, \dots, c_1, c_0) \leftarrow (K_{105}, K_{104}, \dots, K_0, 1) \tag{44}$$

$$(\gamma_{12}, \gamma_{11}, \dots, \gamma_1, \gamma_0) \leftarrow (K_{117}, K_{116}, \dots, K_{106}, 1) \tag{45}$$

Notice that only part bits of the secret key are collaterally loaded into the 318-bit initial state of the cipher. The 38-bit secret key and 128-bit initialization vector have not been used yet.

3.6.2. Key Initialization Procedure Phase

After part bits of the 128-bit secret key are collaterally loaded into the 318-bit initial state of the cipher, the remaining 38-bit secret key and the 128-bit initialization vector are fed into the cipher using the key initialization procedure, which is described in Figure 11. We must use 166 cycles to perform this procedure. Here, we use the parameters $seed_{\alpha t}$, $seed_{\beta t}$, and $seed_{\gamma t}$ to denote the remaining bits of the secret key and the initialization vector, for $0 \leq t \leq 165$. The parameters $seed_{\alpha t}$, $seed_{\beta t}$, and $seed_{\gamma t}$ are given as follows:

$$(seed_{\alpha 165}, \dots, seed_{\alpha 0}) \leftarrow (IV_{127}, \dots, IV_0, K_{127}, \dots, K_{90}) \tag{46}$$

$$(seed_{\beta 165}, \dots, seed_{\beta 0}) \leftarrow (0, \dots, 0, IV_{127}, \dots, IV_0, K_{127}, \dots, K_{106}) \tag{47}$$

$$(seed_{\gamma 165}, \dots, seed_{\gamma 0}) \leftarrow (0, \dots, 0, IV_{127}, \dots, IV_0, K_{127}, \dots, K_{118}) \tag{48}$$

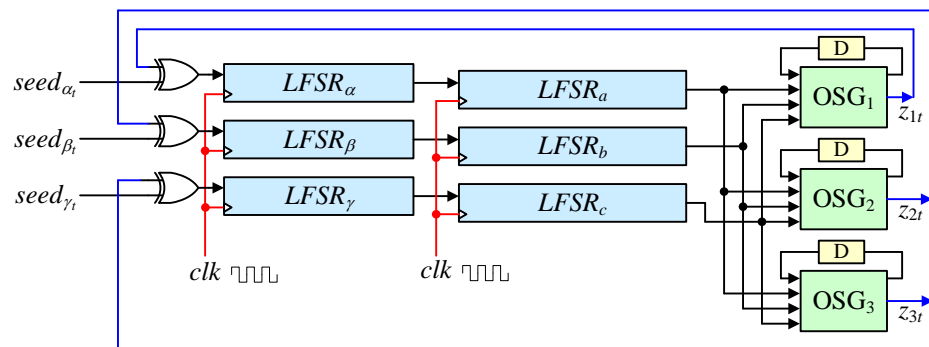


Figure 11. Key initialization.

During the key initialization procedure, the respective rightmost bits of $LFSR_a$, $LFSR_b$, and $LFSR_c$ are still inputted to OSG₁, OSG₂, and OSG₃. The output sequences z_{1t} , z_{2t} , and z_{3t} of OSG₁, OSG₂, and OSG₃, respectively, are fed back and XORed with $seed_{\alpha t}$, $seed_{\beta t}$, and $seed_{\gamma t}$, respectively, into the leftmost bit of $LFSR_\alpha$, $LFSR_\beta$, and $LFSR_\gamma$ in parallel, where $0 \leq t \leq 165$. Note that there is no bit of the keystream output during this initialization

procedure, and each LFSR is clocked 166 times using the *clk* clock. It means that each LFSR is clocked in the normal way by ignoring the clocking controller.

4. Security Properties

The long period, high linear complexity, and good statistical properties are three of the basic requirements for pseudorandom binary sequences in cryptographic applications. In this section, we consider the period, linear complexity, statistical properties, and some attacks with respect to our design. Due to the proposed scheme, the multi-clocking keystream generator and the periods of each LFSR affect each other by current states. We provide the mathematical results regarding the period and linear complexity of our scheme.

4.1. Period

One of the important attributes to be considered for a stream cipher is the period of the keystream. The period of a keystream $s = s_0, s_1, s_2, \dots$ is the smallest positive integer N if $s_i = s_{i+N}$ for all $i \geq 0$. If the period of the keystream is too short, it will result in different parts of the plaintext being encrypted in the identical bits of the keystream. The long period can avoid the keystream being reused when encrypting long plaintexts.

In our scheme, the internal state of the proposed cipher consists of $LFSR_\alpha, LFSR_\beta, LFSR_\gamma, LFSR_a, LFSR_b,$ and $LFSR_c,$ and all of these LFSRs are six maximum-length LFSRs of lengths 31, 17, 13, 61, 89, and 107, respectively. The respective periods of $LFSR_\alpha, LFSR_\beta, LFSR_\gamma, LFSR_a, LFSR_b,$ and $LFSR_c$ are denoted as $P_\alpha, P_\beta, P_\gamma, P_a, P_b,$ and $P_c,$ respectively, and they are equal to $2^{31} - 1, 2^{17} - 1, 2^{13} - 1, 2^{61} - 1, 2^{89} - 1,$ and $2^{107} - 1,$ respectively. Notice that all of these periods are prime numbers. The main work of $LFSR_a, LFSR_b,$ and $LFSR_c$ is to produce the keystream, but their respective clock pulses are dependent on the clock controller. The clock controller consists of $LFSR_\alpha, LFSR_\beta, LFSR_\gamma,$ and some nonlinear Boolean functions, and its output sequences $clk_{ai}, clk_{bi},$ and $clk_{ci},$ where $i \in \{0, 1\},$ are taken to control the respective clock pulses of $LFSR_a, LFSR_b,$ and $LFSR_c.$ It means the periods of $LFSR_a, LFSR_b,$ and $LFSR_c$ will be affected by the periods of $clk_{ai}, clk_{bi},$ and $clk_{ci}.$ Let us define the periods of $clk_{ai}, clk_{bi},$ and clk_{ci} as $P_{clkai}, P_{clkb_i},$ and $P_{clkci},$ respectively. The periods $P_{clkai}, P_{clkb_i},$ and P_{clkci} can be written as [30]:

$$P_{clk_{ai}} = lcm(P_\alpha, P_\beta, P_\gamma) = lcm(2^{31} - 1, 2^{17} - 1, 2^{13} - 1) \tag{49}$$

$$P_{clk_{bi}} = lcm(P_\alpha, P_\beta, P_\gamma) = lcm(2^{31} - 1, 2^{17} - 1, 2^{13} - 1) \tag{50}$$

$$P_{clk_{ci}} = lcm(P_\alpha, P_\beta, P_\gamma) = lcm(2^{31} - 1, 2^{17} - 1, 2^{13} - 1) \tag{51}$$

where $lcm(\cdot)$ denotes the function of the least common multiple. Since all of the periods $P_\alpha, P_\beta,$ and P_γ are prime numbers, $P_{clkai}, P_{clkb_i},$ and P_{clkci} can be simplified to:

$$P_{clkai} = P_{clkb_i} = P_{clkci} = P_\alpha P_\beta P_\gamma = (2^{31} - 1)(2^{17} - 1)(2^{13} - 1) \cong 2^{61} \tag{52}$$

Next, the sequences $clk_{ai}, clk_{bi},$ and clk_{ci} are taken to control the respective clock pulses of the $LFSR_a, LFSR_b,$ and $LFSR_c.$ If $clk_{ai} = 1$ and the *clk* edge trigger simultaneously occurs, then $LFSR_a$ is shifted, and for $LFSR_b$ and $LFSR_c.$ Let us define $S_a, S_b,$ and S_c as the number of 1's in the sequences $clk_{ai}, clk_{bi},$ and $clk_{ci},$ respectively, in every period. The $T_a, T_b,$ and T_c represent affected periods of $LFSR_a, LFSR_b,$ and $LFSR_c,$ respectively. Thus, the periods $T_a, T_b,$ and T_c can be written as [15]:

$$T_a = \frac{P_{clk_{ai}} \times P_a}{gcd(S_a, P_a)} \tag{53}$$

$$T_b = \frac{P_{clk_{bi}} \times P_b}{gcd(S_b, P_b)} \tag{54}$$

$$T_c = \frac{P_{clk_{ci}} \times P_c}{\gcd(S_c, P_c)} \tag{55}$$

Since all of the periods $P_a, P_b,$ and P_c are prime numbers, and they are greater than $S_a, S_b,$ and $S_c,$ respectively, it means $\gcd(S_k; P_k) = 1$ for $k \in \{a; b; c\}$. Thus, $T_a, T_b,$ and T_c can be simplified to:

$$T_a = \frac{P_{clk_{ai}} \times P_a}{\gcd(S_a, P_a)} \cong 2^{61} \times (2^{61} - 1) \tag{56}$$

$$T_b = \frac{P_{clk_{bi}} \times P_b}{\gcd(S_b, P_b)} \cong 2^{61} \times (2^{89} - 1) \tag{57}$$

$$T_c = \frac{P_{clk_{ci}} \times P_c}{\gcd(S_c, P_c)} \cong 2^{61} \times (2^{107} - 1) \tag{58}$$

Finally, the output generator takes the respective rightmost bits of $LFSR_a, LFSR_b,$ and $LFSR_c$ as its input to generate the keystream. The period T_z of the keystream can be written as follows:

$$T_z = lcm(T_a, T_b, T_c) \cong 2^{(61+61+89+107)} \cong 2^{318} \tag{59}$$

It can be seen that, for the period of our proposed method, each cipher is high by considering the security requirements for each cipher mode.

4.2. Linear Complexity

Any periodic sequence can be generated by a Linear Feedback Shift Register (LFSR), since a linear recurrence (or a characteristic polynomial) can be implemented by using a LFSR. Given a periodic sequence, the Berlekamp-Massey algorithm [31] can be used to calculate this recurrence and linear complexity. The length of the shortest recurrence is defined as the linear complexity of a periodic sequence.

The linear complexity is also defined as the size of the shortest LFSR, which can reproduce the same sequence as the given sequence. If the keystream has a linear complexity $LC = n,$ it can be reconstructed by a LFSR after examining only $2n$ bits of the keystream. Once the LFSR is generated by the attacker, they can break the stream cipher. Therefore, the high linear complexity of the keystream is a necessary condition and very important for the design of stream ciphers. The high linear complexity of a keystream means that it possesses higher non-predictability.

According to the Beth-Piper stop-and-go generator and the Gollmann cascade stop-and-go generator in [30], we can provide the lower bound of the linear complexity of our scheme. For our scheme, we use the clock controller to control the respective clock pulses of the $LFSR_a, LFSR_b,$ and $LFSR_c.$ Their respective clock pulses are dependent on the output sequences $clk_{ai}, clk_{bi},$ and clk_{ci} of the clock controller, where $i \in \{0, 1\}.$ We use $P_{clk_{ai}}, P_{clk_{bi}},$ and $P_{clk_{ci}}$ to represent the respective periods of $clk_{ai}, clk_{bi},$ and $clk_{ci}.$ The primitive feedback polynomials of $LFSR_1, LFSR_2,$ and $LFSR_3$ have degrees of $L_a, L_b,$ and $L_c,$ respectively. The lower bound of the linear complexity LC of the proposed cipher can be written as follows:

$$\begin{aligned} LC &= P_{clk_{ai}}L_a + P_{clk_{bi}}L_b + P_{clk_{ci}}L_c \\ &\cong 2^{61} \times 61 + 2^{61} \times 89 + 2^{61} \times 107 \\ &\cong 2^{61} \times 257 \\ &\cong 2^{69} \end{aligned} \tag{60}$$

The linear complexity LC and characteristic polynomial of a keystream can be computed by the Berlekamp-Massey algorithm to obtain the linear complexity LC and characteristic polynomial of the keystream which costs approximately $O(LC^2)$ [32].

The requirement of the linear complexity of the stream cipher is determined by different security levels. For example, when the computation cost of a stream cipher is required equal to the security strength of AES, it must to be approximately equal to $O(2^{128}).$ For our scheme, the linear complexity LC of the proposed stream cipher is approximately equal to

2^{69} . According to the above, the computation cost of the proposed cipher can be written as follows:

$$O(LC^2) = O((2^{69})^2) = O(2^{138}) > O(2^{128}) \quad (61)$$

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

4.3. Statistical Properties

Good statistical properties for randomness are one of the basic important requirements for stream ciphers. The keystream of a good stream cipher not only has the feature of non-predictability, but also has good statistical properties for randomness. So, to implement a stream cipher, the capability to perform statistical tests for randomness must be incorporated. Randomness testing of random and pseudorandom number generators is used in many cryptographic, modeling, and simulation applications. The National Institute of Standards and Technology (NIST) has developed different criteria that may be employed to investigate the randomness of cryptographic applications.

In order to evaluate the randomness of the proposed keystream, we use the Federal Information Processing Standards Publication 140-1 (FIPS PUB 140-1) [16] and the Special Publication 800-22 (SP800-22) [17] to perform the statistical tests for our scheme. They are also issued by the NIST, where the FIPS PUB 140-1 standard is the security requirement for cryptographic modules and the SP800-22 is a statistical test suite for random and pseudorandom number generators for cryptographic applications.

For the FIPS PUB 140-1 standard, there are four test types in the random tests. The specifications of the recommended tests are based on a single bit stream of 20,000 consecutive output bits. To perform the FIPS PUB 140-1 tests, we sampled 100 different keystreams which were generated by 100 random keys and 100 random initialization vectors. Each keystream was a single bit stream of length 20,000 bits. The proposed cipher passed the FIPS PUB 140-1 tests by a proportion of at least 97.00%.

Furthermore, for the SP800-22 statistical test suite, there are fifteen test types in the statistical tests that were developed to test the randomness of binary sequences. These tests focus on a variety of different types of non-randomness that could exist in a sequence. For the SP800-22 statistical test suite, we sampled 100 different keystreams under the 100 secret keys, and initialization vectors were randomly chosen. Each sample was 10,000,000 bits in length. The proposed cipher passed the SP800-22 tests by a proportion of at least 98.00%, with a significance level of 0.01.

4.4. Time-Memory-Data Tradeoff Attack

In 1980, Hellman introduced a general technique for breaking arbitrary block ciphers called a time-memory tradeoff attack. It can also be generalized to the general problem of inverting one-way functions. Babbage and Golić, and later, Biryukov, Shamir, and Wagner, pointed out that a different tradeoff attack called a time-memory-data tradeoff attack is applicable to stream ciphers. Several stream ciphers have been broken by time-memory-data tradeoff attacks, including the famous GSM encryption scheme A5/1 [31].

The time-memory-data tradeoff attack consists of two phases; i.e., the pre-computation phase and the online phase. In the pre-computation phase, the attacker builds large tables relating to the behavior of the system in question. During the online phase, the attacker obtains actual data produced from an unknown key, and his goal is to use the pre-computed tables to find the key as quickly as possible. There are five parameters for any time-memory-data tradeoff attack [33,34]:

- N denotes the size of the search space
- P denotes the time required for the pre-computation phase
- M denotes the size of memory used to store the pre-computed tables
- T denotes the time required for the online phase
- D denotes the amount of output data available to the attacker

The requirement for the attack is that T and M should be smaller than N , since the sum or maximum of T and M is usually signified by the complexity of the time-memory-data tradeoff attack.

For the Babbage-Golić tradeoff attack described in [34], it assumes that the size of the internal state of the stream cipher is N bits and D different keystreams of length $\log N$ are given. The goal of this attack is to recover one of the internal states from any one of the given keystreams. Once a state is found, the corresponding internal states are derived from the rest of the plaintext by running the generator forwards from this known state. For this time-memory-data tradeoff attack, the memory requirement is $M = N/D$. It suffices to search the entries in the table D times and the time complexity is $T = D$ with the pre-computation time $P = M$. By ignoring some of the available data, the T can be reduced from T towards 1, and thus, generalize the tradeoff to $TM = N$ and $P = M$ for any $1 \leq T \leq D$. $T = M$ constitutes an attack of $T = M = D = N^{1/2}$.

Another enhanced tradeoff attack, as described in [34], was presented by Biryukov and Shamir. This attack combined the works of Hellman and Babbage-Golić to launch a new time-memory-data tradeoff attack on stream ciphers. It assumed that the internal state could take N different values. As with the work of Babbage-Golić, the aim of this attack is to recover any one of the many internal states of the stream cipher for D different keystreams that are given. In the tradeoff attack by Biryukov-Shamir, the parameters of which satisfy the relation $P = N/D$ and $TM^2D^2 = N^2$ for $1 \leq T \leq D$. $T = M = N^{1/2}$ with $D = N^{1/4}$ constitutes an attack.

In fact, Babbage suggested that a secret key length of k bits is required and a state size of at least $2k$ bits is required as a design principle of stream ciphers. Similarly, Golić stated that a simple way of increasing security is to make the internal memory size larger.

According to the above, we know the time-memory-data tradeoff attack can be applied when the state size of the stream cipher is too small. A necessary condition on the state size of a stream cipher is that it has to be at least two times the secret key length. For our scheme, the proposed stream cipher takes the 128-bit secret key. In order to avoid time-memory-data tradeoff attacks, the size of the internal state must be at least 256. However, the size of the internal state of the proposed cipher is 318 bits, which means that the size of the search space $N = 2^{318} > 2^{256}$. This gives $T = M = D = N^{1/2} = 2^{159}$ for the Babbage-Golić tradeoff and $T = M = N^{1/2} = 2^{159}$, $D = N^{1/4} = 2^{79.5}$ for the Biryukov-Shamir tradeoff attack, respectively. The results show that they are not better than an exhaustive key search, so the proposed stream cipher can resist against these attacks.

4.5. Correlation Immunity Properties

Correlation immunity as a measure of resistance against ciphertext-only correlation attacks in stream ciphers was defined by Siegenthaler [35]. As shown in Figure 12, there is a nonlinear function f with n input sequences $x = \{x_0, x_1, \dots, x_{n-1}\}$. If the correlation between output sequence z and m input sequences is statistically independent, then the function has correlation immunity of order m , where $m \leq n$. Therefore, the mutual information between the output sequence z and any subset of m input sequences is zero.

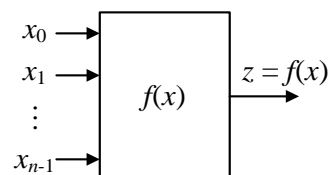


Figure 12. A nonlinear function f with n input sequences.

The m -th order correlation immune function f with n input sequences is defined as follows:

$$\text{prob}[f(x) = 1 \mid x_{i1} = s_1, x_{i2} = s_2, \dots, x_{im} = s_m] = \text{prob}[f(x) = 1] \tag{62}$$

$$\text{prob}[f(x) = 0 \mid x_{i1} = s_1, x_{i2} = s_2, \dots, x_{im} = s_m] = \text{prob}[f(x) = 0] \tag{63}$$

where the $x_{i1}, x_{i2}, \dots, x_{im}$ denote the input variables for $0 \leq i_1 < i_2 < \dots < i_m \leq n - 1$, and $(s_1, s_2, \dots, s_m) \in \{0, 1\}^m$. If f is balanced, $\text{prob}[f(x) = 1] = \text{prob}[f(x) = 0] = 1/2$, then f is also called the m -resilient function.

For our scheme, we proposed four output sequence generators (OSGs) to produce the keystream, which was described in Section 3.5. The four OSGs consist of nonlinear Boolean functions, except OSG₀. All of these OSGs satisfy the properties of being balanced and correlation immune of order one. The four OSGs, namely OSG₀, OSG₁, OSG₂, and OSG₃, take sequence a_{0t}, b_{0t} , and c_{0t} as their inputs, and their output sequences are denoted by z_{0t}, z_{1t}, z_{2t} , and z_{3t} , respectively. All of the output-inputs correlation probabilities of each OSG are listed as follows:

$$\text{prob}[z_{0t} = a_{0t}] = \text{prob}[z_{0t} = b_{0t}] = \text{prob}[z_{0t} = c_{0t}] = 1/2 \tag{64}$$

$$\text{prob}[z_{1t} = a_{0t}] = \text{prob}[z_{1t} = b_{0t}] = \text{prob}[z_{1t} = c_{0t}] = \text{prob}[z_{1t} = d_t] = \text{prob}[z_{1t} = e_t] = 1/2 \tag{65}$$

$$\text{prob}[z_{2t} = a_{0t}] = \text{prob}[z_{2t} = b_{0t}] = \text{prob}[z_{2t} = c_{0t}] = \text{prob}[z_{2t} = g_t] = 1/2 \tag{66}$$

$$\text{prob}[z_{3t} = a_{0t}] = \text{prob}[z_{3t} = b_{0t}] = \text{prob}[z_{3t} = c_{0t}] = \text{prob}[z_{3t} = h_t] = 1/2 \tag{67}$$

where d_t and e_t are carry bits of Dawson’s method in OSG₁, g_t and h_t are respective hybrid carry bits of OSG₂ and OSG₃. After delaying one clock, they are fed back as extra inputs of OSG₁, OSG₂, and OSG₃, respectively.

It is apparent that the output bit is uncorrelated to all the individual input bits for each OSG. The proposed stream cipher is deemed secure and it can resist against some correlation attacks.

5. Experimental Results

In this chapter, the criteria tests and experimental results are described. First, we use the Verilog hardware description language to describe the behavior of the proposed stream cipher. Next, the simulation results of the keystream are taken to evaluate the statistical properties for randomness according to the FIPS PUB 140-1 and SP800-22 packages. Using the 100 secret keys and with initialization vectors randomly chosen, we sampled 100 different keystreams to utilize the statistical test suite. Finally, we provide a performance comparison between the CCDM stream ciphers.

5.1. Statistical Random Number Tests

Good statistical properties for randomness are one of the basic important requirements for stream ciphers. Any cryptographic modules that implement a random or pseudo-random number generator shall incorporate this capability to perform statistical tests for randomness. The NIST has developed different criteria that may be employed to investigate the randomness of cryptographic applications. In order to evaluate the randomness of the proposed keystream, we use the Federal Information Processing Standards Publication 140-1 (FIPS PUB 140-1) and the Special Publication 800-22 (SP800-22) to perform the statistical tests for our scheme.

5.1.1. Random Test Results under FIPS PUB 140-1

As required by FIPS PUB 140-1, the proposed cipher must perform four different test types. These tests include a monbit test, a prker test, a runs test and a long runs test. According to the specifications of FIPS PUB 140-1, these tests were based on a stream of 20,000 consecutive bits. To determine the randomness properties of our scheme, we randomly chose 100 secret keys and initialization vectors, and used them to generate 100

different keystreams for our scheme. Then, we sampled 100 different keystreams to perform the FIPS PUB 140-1 tests, where each keystream was a single bit stream of 20,000 consecutive bits. The test results are shown in Table 10. The proposed cipher passes the FIPS PUB 140-1 tests by a proportion of at least 97.00%.

Table 10. Random Test Results under FIPS PUB 140-1.

FIPS PUB 140-1 Tests	Pass Rate under 20,000 bits/Sample
Monobit Test	100.00%
Poker Test	97.00%
Runs Test	100.00%
Long Run Test	100.00%

5.1.2. Statistical Test Results under NIST SP800-22

We sampled 100 different keystreams to perform the SP800-22 statistical test suite with 100 secret keys and initialization vectors that were randomly chosen. Each sample was 10,000,000 bits in length. Notice, in the cases of the Random Excursions test and the Random Excursions variant test, the requirement for the number J must be greater than 500. As mentioned above, the proposed cipher passed the SP800-22 test suite with a proportion of at least 98.00%. The test results are shown in Table 11. Figure 13 is the comparison chart of p -values between A5/1, Prajapat's scheme [14], and the proposed scheme. The parameter p -value is calculated during testing, which is the strength of the randomness of the dataset. Each p -value corresponds to the probability that the bit sequence of the dataset under testing is random. If it is equal to 1, then the bit sequence has ideal randomness. From the figure, it can be seen that the proposed scheme made progress in the NIST statistical tests.

Table 11. Statistical Test under NIST SP800-22.

No.	Statistical Tests	p -Value	Pass Rate Under 10^7 bits/Sample
1	Frequency	0.874153	100.00%
2	Block Frequency	0.934875	98.00%
3	Runs	0.938110	100.00%
4	Longest Runs of Ones	0.987412	98.00%
5	Rank	0.841558	99.00%
6	Discrete Fourier Transform	0.994201	100.00%
7	Non-Overlapping Templates Matching	0.835479	98.98%
8	Overlapping Templates Matching	0.885029	99.00%
9	Universal Statistical	0.847512	98.00%
10	Linear Complexity	0.928920	99.00%
11	Serial	0.974116	99.00%
12	Approximate Entropy	0.844743	99.00%
13	Cumulative Sums	0.841321	99.00%
14	Random Excursions	0.954667	98.63%
15	Random Excursions Variant	0.985103	99.11%

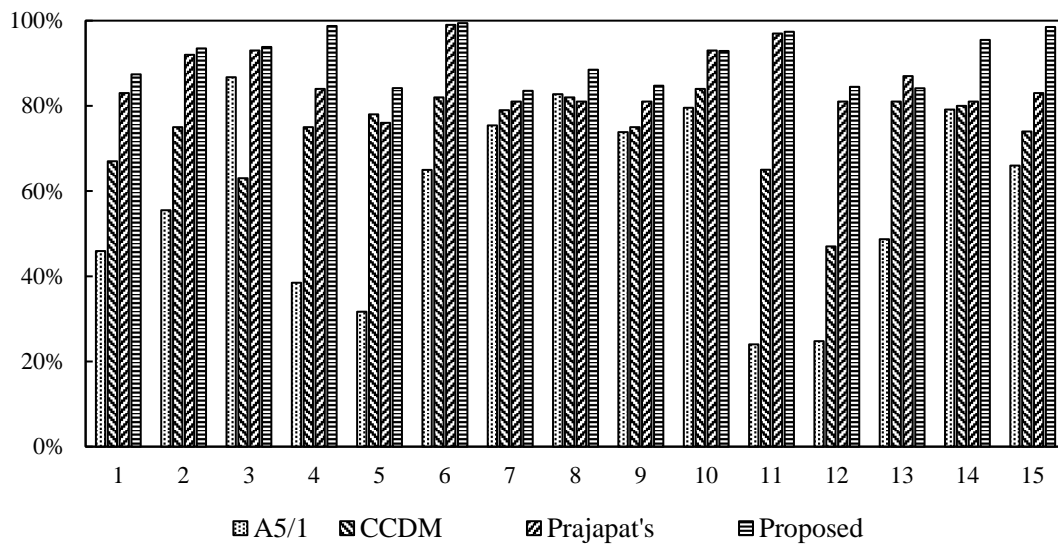


Figure 13. Comparison of p-values [14,15].

5.2. Performance

We compare the performance of the proposed scheme against other stream ciphers in this subsection. The CCDM [15] stream cipher is based on irregular clocking and operating on two different modes, but it does not refer to the design of the key initialization and the operation of mode selection. On the other hand, the proposed scheme has low key initialization cycles and simplicity in terms of mode switching. We used the Verilog hardware description language as our design entry to describe the behavior of the key-based multi-mode clock-controlled stream cipher. The proposed design was synthesized using TSMC 0.18 μm CMOS technology. After synthesis, it showed that the gate level design contained about 5599 gates. Furthermore, our proposed scheme was allowed to run at a working frequency up to 284 MHz. Table 12 shows the comparison results of our design with some other stream ciphers in ASIC [14,15,36]. The results shown are under different CMOS processes. The power cannot be reliably scaled between different processes and libraries, but the gate count can be scaled to a 0.13 μm process for comparison. Furthermore, the initialization cycle of our design is somewhat smaller than other ciphers. In addition, the proposed cipher is able to run on low-end IoT devices, such as ESP32, ATmega328, Arduino, or Raspberry Pi IoT platforms.

Table 12. Performance comparison with some stream ciphers.

Stream Ciphers	Key (bits)	Initialization Cycle	Bits/Cycle	Max. Clock Freq. (MHz)	Gate Count	Process (mm)
Trivium	80	1152	1	327.9	2580	0.13
Grain128	128	256	1	925.5	1857	0.13
F-FCSR-16	128	258	16	317.5	8072	0.13
Mickey128	128	$IV_{\text{length}} + 128 + 160$	1	413.2	5039	0.13
Decim128	128	1152	0.25	309.6	3819	0.13
Edon80	80	160	1	243.3	13,010	0.13
CCDM [15]	128	-	1	237.4	7486	0.19
Prajapat's [14]	64	-	1	297.6	4871	0.16
Proposed scheme	128	166	1	284.2	5599	0.18

6. Conclusions

In this paper, we proposed a key-based multi-mode clock-controlled stream cipher to enhance the security of stream ciphers. The proposed multi-mode depended on the secret key. The different modes were shipped with different encrypting circuits depending on the user's session key. We also analyzed the period, linear complexity, and used known attacks to verify the security strength of the cipher.

We presented the mathematical results regarding the period and linear complexity of the proposed cipher. The results showed that the period of the proposed stream cipher was enough to consider the security requirements for each cipher mode. On the other hand, the linear complexity of the proposed stream cipher satisfied the security strength of AES, and was even stronger than AES. For good statistical properties for randomness, the proposed cipher passed the FIPS PUB 140-1 tests by a proportion of at least 97.00% and passed the SP800-22 test suite by a proportion of at least 98.00%. The experimental results showed that the proposed stream cipher possessed a considerably good randomness property. Regarding security, the proposed cipher could resist against time-memory-data tradeoff attacks and some correlation attacks. In terms of the confidentiality of the IoT, the stream cipher outperformed other symmetric ciphers and asymmetric ciphers. Our proposed stream cipher with a multiple-mode encryption scheme could be applied to the actual IoT environment and promotes the security strength of ciphers.

LFSRs have the advantage of high operation speeds in hardware security. The eight operation modes and the output of 1-bit per clock in the proposed scheme represent its limitations, but these could be ingeniously extended. To elasticize the stream cipher, the designs of an optimized structure for secure hardware circuits will be an interesting research direction. Furthermore, the application of these circuits when merged with cellular automata, chaos, and multi-mode operations represent potential future works.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study.

Acknowledgments: The author wishes to thank Kuo-Hsin Wu for his important assistance and the anonymous referees for their valuable and useful comments that have enriched this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Iqbal, Y.; Amjad, M.; Khan, F.; Abbas, H. The Implementation of Encryption Algorithms in MQTT Protocol for IoT Constrained devices. In Proceedings of the 2022 14th IEEE International Conference on Computational Intelligence and Communication Networks, Al-Khobar, Saudi Arabia, 4–6 December 2022.
2. Atilgan, E.; Ozcelik, I.; Yolacan, E.N. MQTT Security at a Glance. In Proceedings of the 14th International Conference on Information Security and Cryptology, Ankara, Turkey, 2–3 December 2021.
3. Wardana, A.A.; Perdana, R.S. Access control on internet of things based on publish/subscribe using authentication server and secure protocol. In Proceedings of the 2018 10th International Conference on Information Technology and Electrical Engineering: Smart Technology for Better Society, ICITEE 2018, Xiamen China, 7–8 December 2018; pp. 118–123.
4. Polyakov, Y.; Rohloff, K.; Sahu, G.; Vaikuntanathan, V. Fast Proxy Re-Encryption for Publish / Subscribe Systems. *ACM Trans. Priv. Secur.* **2017**, *20*, 14. [[CrossRef](#)]
5. Yao, Y.; Jiang, C.; Wang, X. Enhancing RC4 algorithm for WLAN WEP Protocol. In Proceedings of the IEEE Chinese Control and Decision Conference, Xuzhou, China, 26–28 May 2010.
6. Olakanmi, O. RC4c: A Secured Way to View Data Transmission in Wireless Communication Networks. *Int. J. Comput. Netw. Commun. (IJCNC)* **2012**, *4*, 117.
7. Nagendar, Y.; Prasad, V.K. Allam Appa Rao, G. Padmavathi, Applications of Stream ciphers in wireless communications. *Int. J. Comput. Sci. Eng.* **2018**, *6*, 1121–1126.
8. eSTREAM: The ECRYPT Stream Cipher Project. Available online: <http://www.ecrypt.eu.org/stream/> (accessed on 23 August 2022).

9. Galanis, M.D.; Kitsos, P.; Kostopoulos, G.; Sklavos, N.; Koufopavlou, O.; Goutis, C.E. Comparison of the hardware architectures and FPGA implementations of stream ciphers. In Proceedings of the IEEE International Conference on Electronics, Circuits and Systems, Tel Aviv, Israel, 15 December 2004; pp. 571–574.
10. Sadkhan, S.B.; Reza, D.M. Investigation of the Best Structure for the Nonlinear Combining Function. In Proceedings of the Annual Conference on New Trends in Information & Communications Technology Applications-(NTICT'2017), Baghdad, Iraq, 7–9 March 2017.
11. Tuncer, T.; Avaroğlu, E. Random Number Generation with LFSR Based Stream Cipher Algorithms. In Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2017, Opatija, Croatia, 22–26 May 2017.
12. Yerukala, N.; Padmavathi, G.; Nalla, V.; Prasad, V.K. LFL—A New Stream Cipher for Secure Communications. In Proceedings of the 2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCI2018), Madurai, India, 13–15 December 2018.
13. Sadkhan, S.B. A proposed Development of Clock Control Stream Cipher based on Suitable Attack. In Proceedings of the 1st International Conference of Information Technology to enhance E-learning and other Application, (IT-ELA 2020), Baghdad, Iraq, 12–13 July 2020.
14. Prajapat, R.; Bhadada, R.; Sharma, G. Security Enhancement of A5/1 Stream Cipher in GSM Communication & its Randomness Analysis. In Proceedings of the 2021 IEEE 6th International Forum on Research and Technology for Society and Industry (RTSI), Naples, Italy, 6–9 September 2021.
15. Erguler, I.; Anarim, E. A Clock-Controlled Stream Cipher with Dual Mode. In *Complex Computing-Networks*; Springer Proceedings in Physics Series; Springer: Berlin/Heidelberg, Germany, 2006; Volume 104, pp. 343–352.
16. National Institute of Standards and Technology. *Security Requirements for Cryptographic Modules*; Federal Information Processing Standards Publication 140-1; National Institute of Standards and Technology: Gaithersburg, MD, USA, 1994.
17. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *Special Publication 800-22 Revision 1: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2008.
18. Laih, C.S.; Harn, L.; Chang, C.C. *Contemporary Cryptography and Its Applications*; Flag Publishing Co., Ltd.: Taipei, Taiwan, 2004.
19. Kostopoulos, G.; Sklavos, N.; Galanis, M.; Koufopavlou, O. VLSI Implementation of GSM Security: A5/1 and W7 Ciphers. In Proceedings of the IEEE Work in Wireless Circuits on System (IEEE WoWCAS04), May 2004. Available online: https://www.researchgate.net/publication/229039885_VLSI_Implementation_of_GSM_Security_A51_and_W7_Ciphers (accessed on 2 January 2023).
20. Erguler, I.; Anarim, E. A modified stream generator for the GSM encryption algorithm A5/1 and A5/2. In Proceedings of the 2005 13th European Signal Processing Conference, Istanbul, Turkey, 4–8 September 2005.
21. Stallings, W. *Cryptography and Network Security: Principles and Practice*, 6th ed.; Pearson Education Limited: London, UK, 2014.
22. Zakaria, N.H.; Seman, K.; Abdullah, I. Modified A5/1 Based Stream Cipher for Secured GSM Communication. *IJCSNS Int. J. Comput. Sci. Netw. Secur.* **2011**, *11*, 223.
23. Fauzi, S.; Othman, M.; Shuib, F.M.M.; Seman, K. Modified A5/1 Stream Cipher for Secured Global System for Mobile (GSM) Communication. In Proceedings of the 3rd International Conference on Artificial Intelligence and Computer Science (AICS2015), Penang, Malaysia, 12–13 October 2015.
24. Sadkhan, S.B.; Hamza, Z. Proposed enhancement of A5/1 stream cipher. In Proceedings of the 2nd International Conference on Engineering Technology and their Applications 2019 (IICET2019), Chengdu, China, 10–13 May 2019.
25. Kopparthi, V.R.; Kali, A.; Sabat, S.L.; Anumandla, K.K.; Peesapati, R.; Fouda, J.A.E. Hardware architecture of a digital piecewise linear chaotic map with perturbation for pseudorandom number generation. *AEU Int. J. Electron. Commun.* **2022**, *147*, 154138. [[CrossRef](#)]
26. Živković, M. A table of primitive binary polynomials. *Math. Comput.* **1994**, *62*, 385–386. [[CrossRef](#)]
27. Gligoroski, D.; Markovski, S.; Kocarew, L.; Gusev, M. Algorithm Description of Edon80. eSTREAM Report, ECRYPT Stream Cipher Project. 2005. Available online: http://www.ecrypt.eu.org/stream/p3ciphers/edon80/edon80_p3.zip (accessed on 2 July 2021).
28. Kasper, M.; Kumar, S.; Lemke-Rust, K.; Paar, C. A Compact Implementation of Edon80. eSTREAM Report, ECRYPT Stream Cipher Project. 2006. Available online: <http://www.ecrypt.eu.org/stream/papersdir/2006/057.pdf> (accessed on 15 July 2021).
29. Lim, M.H.; Goi, B.M.; Lee, S.G.; Lee, H.J. Hierarchical Dawson's Summation Generator. In Proceedings of the 2007 International Conference on Convergence Information Technology, Gyeongju, Korea, 21–23 November 2007; pp. 1395–1401.
30. Zeng, K.; Yang, C.-H.; Wei, D.-Y.; Rao, T. Pseudorandom bit generators in stream-cipher cryptography. *Computer* **1991**, *24*, 8–17. [[CrossRef](#)]
31. Biryukov, A.; Shamir, A.; Wagner, D. Real Time Cryptanalysis of A5/1 on a PC. In Proceedings of the 7th International Workshop on Fast Software Encryption, New York, NY, USA, 10–12 April 2000; pp. 1–18.
32. Massey, J. Shift-register synthesis and BCH decoding. *IEEE Trans. Inf. Theory* **1969**, *15*, 122–127. [[CrossRef](#)]
33. Biryukov, A.; Shamir, A. Cryptanalytic Time/Memory/Data Tradeoffs for stream ciphers. In Proceedings of the 6th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, Kyoto, Japan, 3–7 December 2000; pp. 1–13.

34. Hong, J.; Sarkar, P. Rediscovery of Time Memory Tradeoffs. IACR ePrint Archive, Report 2005/090. 2005. Available online: <http://eprint.iacr.org/2005/090> (accessed on 31 October 2022).
35. Siegenthaler, T. Correlation-immunity of nonlinear combining functions for cryptographic applications (Corresp.). *IEEE Trans. Inf. Theory* **1984**, *30*, 776–780. [[CrossRef](#)]
36. Good, T.; Benaissa, M. Hardware Results for Selected Stream Cipher Candidates. Workshop Record State of the Art of Stream Ciphers 2007 (SASC 2007). 2007. Available online: <http://www.ecrypt.eu.org/stream/papersdir/2007/023.pdf> (accessed on 1 November 2022).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.