Security and performance analysis of SCDSP

[1]Fardous Mohamed Eljadi, [2]Imad Fakhri Al-Shaikhli
Department of Computer Science, IIUM, Malaysia
IIUM Cyber Security Malaysia Center for Cyber Space Security
[1]fardous.eljadi@live.iium.edu.my
[2]imadf@iium.edu.my

Abstract—There are few approaches that attempt to add dynamicity to the structure of stream ciphers in order to improve their security level. SCDSP is a dynamic stream cipher that based on these approaches. It uses dynamic structure and parameters to increase the complexity of the cipher to improve its security level. The dynamic parameters are specified using bits from the secret key. In this paper, SCDSP is evaluated by conducting a performance and security analysis. Furthermore, a comparison between SCDSP and the seven winners of eSTREAM competition is performed. The results show that SCDSP is very promising for practical use.

Keywords— Stream Cipher, Dynamic Structure and Parameters (DSP), Linear Feedback Shift Registers.

I. Introduction

Stream ciphers are more suitable than block ciphers in time-critical applications or processing-constrained devices because of their real-time operation and adaptability to hardware implementations [1]. Multimedia systems, hand held communication devices, and wireless sensor networks provide some examples in which a stream cipher is preferred for an encryption operation. Currently there is no standard model for stream ciphers, regardless that this type of stream ciphers are needed for a lot of applications. To deal with the lack of standards for secure stream ciphers that can be utilised by industry, a number of standardization efforts which included stream ciphers were made by the cryptographic community. The first one was the New European Schemes for Signature, Integrity and Encryption (NESSIE) project that began in 2000 and ended in 2004. All the stream cipher proposals sent to NESSIE were discarded mainly because of the discovery of cryptanalytic attacks [2]. After that, Japan initiated another standardization effort named the Cryptographic Research and Evaluation Committee (Cryptrec) [3]. Cryptrec highly recommended a number of stream ciphers among them: 128 bit RC4, MUGI and MULTI-S01[4]. But, afterwards these ciphers were also found to be vulnerable to the cryptanalytic attacks [5-7]. Another project targeting stream ciphers was launched as a part of the European Network of Excellence for Cryptology (ECRYPT) in 2004 [1]. This was named eSTREAM - the ECRYPT Stream Cipher Project. It was running from 2004 to 2008. It had the objective of activating the research area of analysis and design of stream ciphers. Researchers were invited to submit stream cipher proposals in two categories: high performance software applications and hardware applications with restricted resources. These submissions have been subjected to rigorous cryptanalysis and have resulted in the enhancement of overall understanding of stream cipher design. In this competition, sixteen stream ciphers reached the final phase. Seven of them were selected to be winners [1]. These winners are HC-128 [8], Salsa20/12 [9], Rabbit[10] and SOSEMANUK [11] in profile 1 (software-oriented Ciphers) and Grain v1 [12], MICKEY 2.0[13] and Trivium [14] in profile 2 (hardware oriented cipher). Even after these standardisation efforts, several weaknesses were found in these ciphers [15-18]. Therefore, it is necessary to make a large amount of effort toward the invention of new replacement schemes.

Few approaches, which attempt to add dynamicity to the structure of stream ciphers to improve their security level, are proposed. The idea behind these approaches is that the structure of these ciphers is unknown to the attackers, and it makes them more resistant to attacks. These ciphers are not widely discussed among researchers. Moreover, the research about using dynamic structure in stream ciphers has mostly focused on dynamic polynomial switching in the Linear Feedback Shift Registers [19]. In this paper, a dynamic stream cipher algorithm called SCDSP [20] is analysed. It is based on using dynamic structure and parameters to increase the complexity of the cipher and consequently improve its security level. SCDSP is evaluated by conducting a performance and security analysis. Moreover, a comparison between SCDSP and the seven winners of eSTREAM competition is performed.

This paper is organized as follows: Section 2 describes SCDSP. Section 3 gives the details of the security analysis conducted on the cipher and the performance analysis of the cipher is discussed in section 4. Section 5 and 6 report a comparison between SCDSP and the seven winners of eSTREAM competition. Finally, section 7 concludes this paper.

II. SCDSP Architecture

In SCDSP, a number of pools that contain several options for constructing the algorithm are used. In every run, the algorithm is built based on the secret key. The variable parameters are the number of registers, the length of registers, the clocking system, the initialisation procedure, the confusion and diffusion method, and the output function. These parameters are specified using bits from the secret key.

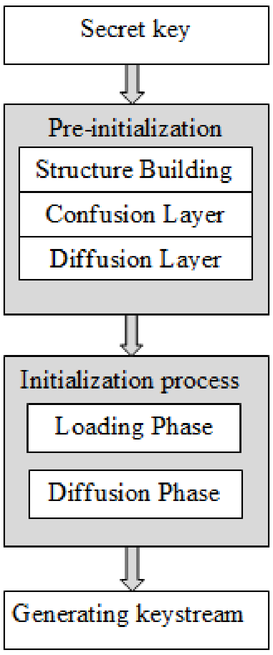
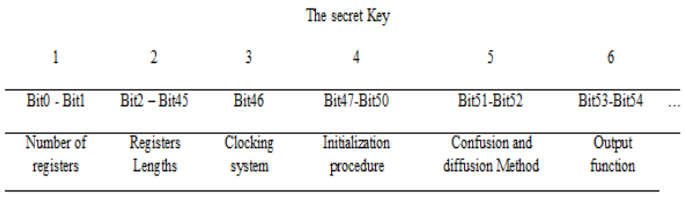


Figure 1 The basic steps of SCDSP.

Figure 1 shows the basic steps of our algorithm. Initially, the secret key is exchanged. Then, the first stage "pre-initialization" starts, which includes selecting the algorithm's structure based on the secret key and applying the confusion and diffusion method on the key before the initialisation stage. Afterward, the selected initialization method is executed, and the generator starts to produce the keystream.

In our initial implementation, the secret key and the IV lengths are chosen to be 128 bits in order to limit the scope of brute force attacks [21]. For connecting registers, the output of each register is connected to the input of another register. Figure 2 illustrates the bits that determine the dynamic parameters in the secret key.



The secret Key

1	2	3	4	5	6	
Bit0 - Bit1	Bit2 – Bit45	Bit46	Bit47-Bit50	Bit51-Bit52	Bit53-Bit54	…
Number of registers	Registers Lengths	Clocking system	Initialization procedure	Confusion and diffusion Method	Output function	

Figure 2 The bits that determine the dynamic parameters in the secret key.

We make every pool consist of several components. The following section describes the options that are used to construct the proposed design.

A. *Number of registers*

The number of registers will be determined by the initial bits of the secret key. In order to use the majority clocking function of the A5 stream cipher [22], an odd number of registers is used. Therefore, the number of registers will be chosen from {7, 9, 11}. Based on a choice of 7, 9 or 11 registers, the first two bits (Bit0Bit1) of the secret key are used to determine the number of registers.

- *If Bit0Bit1=00, then the number of registers =7.*
- *If Bit0Bit1=01, then the number of registers =9.*
- *If Bit0=1, then the number of registers =11.*

B. *Registers' lengths*

The following points are considered to determine the ranges of the lengths of the registers.

- *The sum of the lengths of the registers used in any session should be equal to or greater than the key size, in order to avoid the known Time-Memory Trade Off attacks for stream ciphers [23]. In addition, the lengths of the registers should not present any difficulty in the implementation for either the software or hardware [24].*
- *The registers' sizes must be limited to use the proper primitive feedback polynomials.*
- *The registers' lengths should be pairwise coprime in order to reduce the correlation attacks against the cipher and ensure that the cycle time is maximised [24, 25]. The registers' lengths will be chosen from {49,53,59,61,65,67,71,73,79,83,89,97,101}. This set has 13 elements. To determine the registers' lengths, 44 bits from the key are reserved.*
  - *If the number of registers =7, then 28 bits (7x4) are required to represent the registers' lengths, which allows 50,388 choices of register lengths.*
  - *If the number of registers =9, then 36 bits (9x4) are needed to represent the registers' lengths, which allows for 293,930 choices of register lengths.*
  - *If the number of registers =11, then 44 bits (11x4) are required to represent the registers' lengths, which allows 1,352,078 choices of register lengths.*

Table 1 illustrates the chosen primitive feedback polynomials. These polynomials were taken from [21].

TABLE I
PRIMITIVE POLYNOMIALS

Register size	Primitive polynomial
49	X49+ X9 + 1
53	X53+ X6 + X2+ X + 1
59	X59+ X7 + X4+ X2 + 1
61	X61+ X5 + X2+ X + 1
65	X65+ X18 + 1
67	X67+ X5 + X2+ X + 1
71	X71+ X6 + 1
73	X73+ X25 + 1
79	X79+ X9 + 1
83	X83+ X7 + X4+ X2 + 1
89	X89+ X38 + 1
97	X97+ X6 + 1
101	X101+X7+X6+X+1

C. *Clocking mechanism*

According to Courtois *[26]*, the complex clocking mechanisms provide significant immunity to algebraic attacks. For this reason, two different clocking functions are used, which are the majority clocking function as in A5.1[22] and a keystream-dependent clocking function. In our implementation, Bit46 is used to determine the clocking system. At the beginning, in the initialisation, the selected registers are clocked regularly 128 times without producing a keystream. Then, at each clock cycle, the registers are clocked according to the selected clocking system.

In the voting clocking function, each register has one clocking tap in the middle. At each clock cycle, the selected registers are clocked according to their clocking taps. The registers make a clocking vote using their taps to find the majority of the current clocking taps. Afterward, each register compares the voting result with its own clocking tap. If it is equal to the voting result, then the register is clocked [22]. In a keystream-dependent clocking function, three registers are chosen to be clocked regularly, and the other registers are clocked based on the previous keystream.

If Bit46 = 0, then the majority clocking function of the A5 stream cipher is used; otherwise, a keystream-dependent clocking function is used.

D. *Confusion and Diffusion Method*

Muller [27] stated that the initialisation process can provide protection against differential attacks by mixing nonlinear and linear functions. Therefore, using S-box and P-box in the initialisation process prevents a high probability of differential attack. In the SCDSP, a confusion and diffusion method is applied on the key in the pre-initialisation stage by using three types of S-box and P-box.

If Bit51Bit52 =00, then Present [28] S-box and P-box are used ; else if Bit51Bit52 =01, then HISEC [29] S-box and P-box are used. Otherwise, LBlock [30] S-box and P-box are used.

E. *Initialisation procedure*

In the initialisation process, the loading phase and the diffusion phase are dynamic. The next sections provide the details.

- Loading phase: This phase has 9 options. The loading process that is used is inspired by Dragon stream cipher [31]. Based on the Bit47, Bit48, Bit49, and Bit50 from the secret key, the loading option will be chosen from the following options:

1) $k \| k' \oplus IV' \| IV \| k \oplus IV' \| k' \| k \oplus IV \| IV' \| k' \oplus IV \| \overline{k \oplus IV}$
2) $k' \oplus IV' \| IV \| k \oplus IV' \| k' \| k \oplus IV \| IV' \| k' \oplus IV \| \overline{k \oplus IV} \| K$
3) $IV \| k \oplus IV' \| k' \| k \oplus IV \| IV' \| k' \oplus IV \| \overline{k \oplus IV} \| K \| k' \oplus IV'$
4) $k \oplus IV' \| k' \| k \oplus IV \| IV' \| k' \oplus IV \| \overline{k \oplus IV} \| K \| k' \oplus IV' \| IV$
5) $k' \| k \oplus IV \| IV' \| k' \oplus IV \| \overline{k \oplus IV} \| K \| k' \oplus IV' \| IV \| k \oplus IV'$
6) $k \oplus IV \| IV' \| k' \oplus IV \| \overline{k \oplus IV} \| K \| k' \oplus IV' \| IV \| k \oplus IV' \| k'$
7) $IV' \| k' \oplus IV \| \overline{k \oplus IV} \| K \| k' \oplus IV' \| IV \| k \oplus IV' \| k' \| k \oplus IV$
8) $k' \oplus IV \| \overline{k \oplus IV} \| K \| k' \oplus IV' \| IV \| k \oplus IV' \| k' \| k \oplus IV \| IV'$
9) $\overline{k \oplus IV} \| K \| k' \oplus IV' \| IV \| k \oplus IV' \| k' \| k \oplus IV \| IV' \| k' \oplus IV$

where x' denotes the swapping of the upper and lower 64-bit halves of x, and X¯ denotes the complement of x.

According to [32], the padding pattern should not be identical or cyclic (identical means that the padding is all-zeros or all-ones, and cyclic means that the padding consists of a repeated specific pattern) in order to reduce the chance of finding slid pairs and also reduce the probability of obtaining shifted keystreams. In our initialization process, we do not use an identical or cyclic padding pattern.

- Diffusion phase: The shift registers are updated 128 times before starting to produce the keystream. This approach makes any cryptanalysis of the keystream generated from this cipher difficult.

F. *Output functions*

Three types of output functions are used, which are XORing the output bits of the selected registers, an output function based on the Grain stream cipher [33], and an output function based on the BEAN stream cipher [34].

In the output function based on the Grain stream cipher, two registers are used. The first register is an NLFR, which is defined and filled with a part of the keystream that is produced in the initialisation phase. The second register is an LFSR, and it is chosen from the pool of selected registers in each clock cycle.

In the output function based on the BEAN stream cipher, two registers are used, and both are chosen from the pool of selected registers in each clock cycle.

If Bit53Bit54=00, then the output function is XORing the output bits of the chosen registers; else if Bit53Bit54=01, then the output function based on the Grain stream cipher is chosen. Otherwise, the output function based on the BEAN stream cipher is selected.

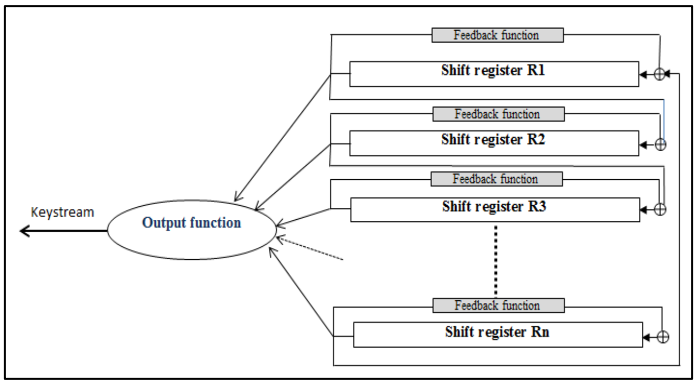


Figure 3 The general construction of SCDSP

III. Security Analysis of the proposed algorithm

Security analysis plays an essential role in the evaluation process of the new ciphers. In this section, the period and the linear complexity of the proposed algorithm are analysed. Then, the results of the NIST test suit are presented. Finally, a discussion about some possible attacks is given based on the specific design choices behind the different functions and the parameters used in SCDSP.

1) *Period and Linear Complexity*

Due to the irregular clock control, the method of connecting the registers, and the used output function in the SCDSP, it is difficult to establish mathematical results about the period and linear complexity of the cipher. However, we can predict a lower bound for the period (considering only the internal state) and measure the linear complexity of the algorithm using the NIST test suit, which uses the Berlekamp-Massey algorithm to measure the linear complexity.

According to [21], the period of LFSR that has a primitive feedback polynomial is equal to 2^L-1, where L is the length of the register. Assume that our cipher consists of n registers ($7 \leq n \leq 11$). Each register cycles periodically with the period $P = 2^L-1$. The output of each register is connected to the input of another register. Therefore, the period of n LFSRs is at most the LCM of the periods of the sequences output by the n LFSRs [35]. Hence, the minimum period of the keystream is estimated to be much more than LCM ($P_1P_2 \ldots P_n$), where P_i is the period of register i.

In order to measure the linear complexity of the SCDSP, the NIST test suit is used. 10^8 bits are generated by the proposed algorithm and examined by the NIST statistical suit. The best p-value for this test is equal to 0.924076, which is a very high value because 0.01 is the passing rate, while the value 1 represents perfect randomness.

2) *Randomness of the keystream*

The statistical tests on SCDSP were performed using the NIST Test Suite for different keys. The results were obtained for 10^8 bits generated by the proposed algorithm. The average is calculated for the non-overlapping-templates, random-excursions and random-excursions variant test results, which are decomposable into a variety of subtests. Table 2 shows the results of the NIST test suite for one of the keys. It can be observed from the results that the proposed algorithm passes all the tests because the P-value of all the tests is below 0.01.

TABLE 2
NIST TEST RESULTS OF SCDSP

ID	Statistical test	P-value	Result
1	Frequency	0.994250	Pass
2	Block-frequency	0.637119	Pass
3	Cumulative-sums(forward)	0.319084	Pass
	Cumulative-sums(reverse)	0.798139	Pass
4	Runs	0.739918	Pass
5	Longest-runs of ones	0.145326	Pass
6	Rank	0.162606	Pass
7	DFT (Spectral)	0.051942	Pass
8	Non-overlapping-template	0.463171	Pass
9	Overlapping-templates	0.028817	Pass
10	Universal	0.964295	Pass
11	Approximate entropy	0.883171	Pass
12	Random-excursions	0.463186	Pass
13	Random-excursions variant	0.483266	Pass
14	Serial1	0.137282	Pass
	Serial2	0.145326	Pass
15	Linear complexity	0.924076	Pass

There are some statistical tests that showed excellent results. For example, the p-value for the frequency test is 0.994250, which indicates an excellent distribution of zeroes and ones for the entire sequence. Moreover, the linear complexity test's p-value is equal to 0.924076, which indicates that the generated keystream is sufficiently complex to be considered random [36].

3) *Autocorrelation Test Results*

The autocorrelation test [37] is performed 5 times using 5 different keys, and the number of occurrences in each time is compared to the expected value using a chi-square table with one degree of freedom and a significance level of α=0.05. The algorithm fails if the value of the test exceeds 3.84. It is notable from the results of Table 3 that SCDSP successfully passes all five tests because the autocorrelation value of all tests is less than 3.84.

TABLE 3
AUTOCORRELATION TEST RESULTS OF SCDSP

key	Autocorrelation value	Result
Key1	1.5943	Pass
Key2	0.9396	Pass
Key3	1.698	Pass
Key4	1.5943	Pass
Key5	0.7866	Pass

4) *Resistance against Known Attacks*

In this section, a discussion about resistance of SCDSP against known attacks is presented.

Brute force attack. In this attack, if the key is n bits, then the attacker must try 2^n keys in the worst case and 2^{n-1} keys on average. According to [21], the minimum size for the secret key is set to be 128 bits in order to limit the scope of brute force attacks on these systems. In SCDSP, the Key size is 128 bits, and it can be increased to be 256 bits.

Algebraic Attacks. Algebraic attacks on stream ciphers have received a large amount of attention lately because they are very efficient if the designer is not careful. A filter generator that uses only a nonlinear Boolean output function and an LFSR can be very vulnerable in algebraic attacks [38]. Babbage and Dodd [1] stated that algebraic attacks usually become possible when the keystream is correlated to one or more linearly clocking registers, whose clocking is either entirely predictable or can be guessed. In SCDSP, an S box, a P-box, and an irregular clocking function are used to introduce nonlinearity together with a nonlinear Boolean output function. Solving equations for the initial state that has a minimum size of 343 bit (7 is the minimum number of registers and 49 bit is the size of the smallest register) is not possible due to the existence of nonlinearity. The algebraic degree of the output bit expressed in the initial state bits will be large in general and will also vary over time. This property will defeat any algebraic attack on SCDSP.

Time-Memory-Data Trade-off Attack. It is well known that the state of a stream cipher must be at least twice the key size, and the IV size should be at least as large as the key size in order to prevent time-memory trade-off attacks [23, 39]. In SCDSP, there are at least seven shift registers, and their minimum size is 49 bits each; therefore, the total number of state sizes is at least 343 bit, and thus, the number of states is greater than twice the key size. Moreover, the IV size is equal to the key size. According to Biryukov and Shamir [39], a generic time-memory-data trade off attack on stream ciphers costs $O(2^{n/2})$, where n is the number of inner state variables in the stream cipher. In SCDSP, there are at least seven shift registers, and their minimum size is 49 bits each. Thus, the expected complexity of a time-memory-data trade-off attack is not lower than $O(2^{343/2})$,which is greater than the complexity of brute force attacks.

Guess-and-determine attacks. The strategy for this type of attack is to guess a few of the unknown variables of the cipher and, from those, deduce the remaining unknowns. The system is then iterated a few times, producing output that can be compared with the actual cipher output, to verify the guess [1]. In order to make guessing attacks infeasible, the key length must be increased. Moreover, irregular clocking can increase the resistance against these attacks [40]. In SCDSP, the initial key length has 128 bits. Moreover, two types of irregular clocking are used.

Distinguishing Attacks. A distinguishing attack on a cipher works with a formal model of security, in which an attacker can distinguish between the output of a specific cipher and the output of a truly random process, with a non-negligible probability [1]. If attackers cannot make this distinction, then an algorithmically derived stream cipher will view them as a One Time Pad and will be information-theoretically secure. In fact, the reality is that there is always a distinguishing attack against any algorithmic cipher because it should have a finite key, and thus, brute-force key enumeration will yield a distinguishing attack of complexity 2^{k-1}, where k is the key length. An easy way to get away from such attacks is to state that the cipher must be rekeyed after a certain amount of keystream [41]. In SCDSP, the key and IV are used to determine the structure of the cipher in the next 2^{64} bytes. This approach adds to the non-predictability of future set-ups and consequently provides security against attacks.

Correlation attacks. Key-stream generators based on regularly clocked LFSR's are susceptible to basic and fast correlation attacks. However, there are two major obstacles to the adaptation of this attack on SCDSP:

a) The register lengths are pairwise coprime, which reduces the correlation attacks against the cipher and ensures that the cycle time is maximised [24, 25].

b) The use of irregular clocking limits the possibilities for mounting classical correlation attacks [42].

Differential attacks. A differential attack on stream ciphers examines the behaviour of the initialisation and keystream generation processes for a differential in the inputs. It analyses how differences in the inputs (either the key or IV, or the internal state) affect the output (either the internal state or keystream) [43]. Muller [27] stated that the initialisation process can provide protection against differential attacks by mixing nonlinear and linear functions. Therefore, using S-box and P Box in the initialisation process prevents a high probability of differentials [31]. In SCDSP, the use of S-box and P-box is an essential part of the initialisation process.

IV. Performance of the proposed algorithm

The performance of the proposed algorithm is measured as the number of keystream bits produced over a given time period. Specifically, it was measured in megabits per second (Mbps). Our own reasonably efficient implementation of SCDSP generated 1.23764 Megabits of keystream per second using a PC with a 2.1GHz Intel® Core™ i7-3612QM processor. It is worthwhile to mention that SCDSP has the flexibility through its dynamic design to implement it efficiently. This goal can be accomplished by using light weight and low cost components with a medium level of security.

V. Comparisons of NIST Test Results

Statistical analysis in terms of the keystream has been conducted for the seven stream ciphers that were selected to be winners in the eSTREAM competition and the proposed algorithm. In this experiment, 100 keystreams of length 10^6 bits are generated using randomly chosen key and IV pairs. The outputs were tested using the 15 tests with standard parameters. Table 4 shows the results of these tests for the compared algorithms.

All the ciphers pass the 15 tests because they have p-values greater than 0.01. However, some of the algorithms have higher p-values than others, which means that they have better statistical properties than the others [36]. In addition, there are several observations about the results with regard to the range of p-values for some of the tests and the p-values of each algorithm.

The first observation is that the range of p-values (the difference between the highest p-value and the lowest p-value in a test) is high in some tests. For example, in the linear complexity test, Trivium has the lowest p-value, which is 0.01455, while SCDSP has the highest p-value, which is 0.924076. Therefore, the range of p-values is 0.909526. This finding indicates that there is a significant gap in the linear complexity between the compared ciphers. On the other hand, the p-values in some of the tests of the compared stream ciphers are convergent. For example, in the non-overlapping-template test, the range of p-values is 0.075629.

The second observation is that some of the algorithms have high p-values in several tests, which make them outperform the other algorithms. In contrast, some of the algorithms have low p-values in several of the tests, which indicates that the statistical properties of their generated sequences are low compared to the others. In addition, the p-values of some of the algorithms are convergent.

It is worthwhile to mention that Trivium has a very low linear complexity. That finding could be due to its simple structure [44]. On the other hand, HC-128 has a high p-value in a linear complexity test. This finding could result from using two large dynamic s boxes in its design. However, the proposed algorithm has the highest value in the test of linear complexity, which is a good sign and indicates the good security level of this algorithm.

TABLE 4
THE RESULTS OF NIST TESTS FOR THE COMPARED ALGORITHMS.

ID	Statistical test	Grain	MICKY	Rabbit	Sosemanuk	HC-128	Trivium	Salsa20	SCDSP
1	Frequency	0.455937	0.455937	0.971699	0.883171	0.262249	0.401199	0.955835	0.99425
2	Block-frequency	0.494392	0.595549	0.657933	0.419021	0.834308	0.637119	0.262249	0.637119
3	Cumulative-sums(forward)	0.883171	0.304126	0.897763	0.494392	0.851383	0.978072	0.474986	0.319084
	Cumulative-sums(reverse)	0.058984	0.816537	0.494392	0.181557	0.122325	0.595549	0.514124	0.798139
4	Runs	0.719747	0.304126	0.911413	0.897763	0.955835	0.383827	0.678686	0.739918
5	Longest-runs of ones	0.042808	0.162606	0.455937	0.719747	0.883171	0.122325	0.574903	0.145326
6	Rank	0.911413	0.719747	0.851383	0.162606	0.202268	0.191687	0.419021	0.162606
7	DFT	0.275709	0.202268	0.383827	0.779188	0.494392	0.739918	0.025193	0.051942
8	Non-overlapping-template	0.514027	0.488552	0.489705	0.473059	0.478228	0.5388	0.484063	0.463171
9	Overlapping-templates	0.971699	0.595549	0.595549	0.867692	0.678686	0.181557	0.437274	0.028817
10	Universal	0.678686	0.383827	0.798139	0.021999	0.514124	0.816537	0.719747	0.964295
11	Approximate entropy	0.595549	0.699313	0.319084	0.935716	0.289667	0.262249	0.137282	0.883171
12	Random-excursions	0.239402	0.550557	0.538622	0.369533	0.561113	0.390931	0.418249	0.463186
13	Random-excursions variant	0.364894	0.394504	0.490274	0.474825	0.41484	0.546774	0.516361	0.483266
14	Serial1	0.319084	0.162606	0.798139	0.12962	0.153763	0.911413	0.137282	0.137282
	Serial2	0.514124	0.037566	0.401199	0.23681	0.066882	0.834308	0.153763	0.145326
15	Linear complexity	0.719747	0.045675	0.419021	0.616305	0.851383	0.01455	0.699313	0.924076

VI. PERFORMANCE COMPARISON BETWEEN SCDSP AND MICKEY-128

Because the Mickey-128 stream cipher is the only cipher that has a dynamic clocking system (irregular clocking system), it is compared with SCDSP in terms of the speed. The compared algorithms were implemented on the same PC in the same environment. Table 5 shows the comparison between SCDSP and Mickey-128 from the perspective of performance.

TABLE 5
PERFORMANCE COMPARISON BETWEEN SCDSP AND MICKEY-128

Algorithm	Throughput(Mbps)
SCDSP	1.23764
MICKEY-128	0.518624

The result of the performance analysis shows that there is a significant gap in the performance between our proposed algorithm and Mickey-128. Despite the fact that our proposed algorithm has more dynamic parameters than Mickey-128, the proposed algorithm is faster than Mickey-128.

VII. CONCLUSION

In this paper, a dynamic stream cipher algorithm called SCDSP was evaluated by conducting a performance and security analysis. Furthermore, a comparison between this algorithm and the seven winners of eSTREAM competition was performed. The results were encouraging. It is worth mentioning that adding the dynamicity to the design of stream cipher enhances the security level and increases the immunity of this cipher to several attacks. Moreover, the design of SCDSP can extend the opportunity for customizing several designs according to the need of the industries. In future work, several algorithm designs will be developed to suit different levels of security and several types of usage. For example, developing a software oriented algorithm design and developing a hardware oriented algorithm design. The components of the dynamic structure can be chosen to fit any type of usage with any level of security. This flexibility is the main advantage of the dynamic design.

REFERENCES

[1] M. Robshaw and O. Billet, *New stream cipher designs: the eSTREAM finalists* vol. 4986: Springer, 2008.

[2] F. M. Eljadi and I. F. Al-Shaikhli, "Statistical Analysis of the eSTREAM competition winners," in *Advanced Computer Science Applications and Technologies (ACSAT), 2015 International Conference*, 2015.

[3] H. Imai and A. Yamagishi, "CRYPTREC Project Cryptographic Evaluation Project for the Japanese Electronic Government," in *Advances in Cryptology—ASIACRYPT 2000*, ed: Springer, 2000, pp. 399-400.

[4] J. Y. Cho, *New Results on Cryptanalysis of Stream Ciphers*: Macquarie University, 2007.

[5] J. D. Golić, "A weakness of the linear part of stream cipher MUGI," in *Fast Software Encryption*, 2004, pp. 178-192.

[6] M. Henricksen and E. Dawson, "Rekeying issues in the MUGI stream cipher," in *Selected Areas in Cryptography*, 2006, pp. 175-188.

[7] A. Nagao, T. Ohigashi, T. Isobe, and M. Morii, "Expanding Weak-key Space of RC4," *Journal of Information Processing,* vol. 22, pp. 357-365, 2014.

[8] H. Wu, "The stream cipher HC-128," in *New Stream Cipher Designs*, ed: Springer, 2008, pp. 39-47.

[9] D. J. Bernstein, "The Salsa20 family of stream ciphers," in *New stream cipher designs*, ed: Springer, 2008, pp. 84-97.

[10] M. Boesgaard, M. Vesterager, and E. Zenner, "The Rabbit stream cipher," in *New Stream Cipher Designs*, ed: Springer, 2008, pp. 69-83.

[11] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, *et al.*, "Sosemanuk, a fast software-oriented stream cipher," in *New Stream Cipher Designs*, ed: Springer, 2008, pp. 98-118.

[12] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing,* vol. 2, pp. 86-93, 2007.

[13] S. Babbage and M. Dodd, "The stream cipher MICKEY 2.0," *ECRYPT Stream Cipher, available at http://www. ecrypt. eu. org/stream/p3ciphers/mickey/mickey_p3. pdf,* 2006.

[14] C. De Cannière, "Trivium: A stream cipher construction inspired by block cipher design principles," in *Information Security*, ed: Springer, 2006, pp. 171-186.

[15] L. Ding and J. Guan, "Related Key Chosen IV Attack on Grain-128a Stream Cipher," *Information Forensics and Security, IEEE Transactions on,* vol. 8, pp. 803-809, 2013.

[16] Z. Ma and D. Gu, "Improved Differential Fault Analysis of SOSEMANUK," in *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*, 2012, pp. 487-491.

[17] M. J. Mihaljevic, S. Gangopadhyay, G. Paul, and H. Imai, "Internal state recovery of Grain-v1 employing normality order of the filter function," *Information Security, IET,* vol. 6, pp. 55-64, 2012.

[18] Z.-y. Shao and L. Ding, "Related-cipher attack on Salsa20," in *Computational and Information Sciences (ICCIS), 2012 Fourth International Conference on*, 2012, pp. 1182-1185.

[19] F. M. Eljadi and I. F. Al-Shaikhli, "Dynamic linear feedback shift registers: A review," in *Information and Communication Technology for The Muslim World (ICT4M), 2014 The 5th International Conference on*, 2014, pp. 1-5.

[20] F. M. Eljadi and I. F. Al-Shaikhli, "SCDSP: A Novel Stream Cipher with Dynamic Structure and Parameters," *International Journal of Advancements in Computing Technology,* vol. 7, p. 49, 2015.

[21] B. Schneier, "Applied cryptography: protocols, algorithms, and source code in C," *John Wiley & Sons, Inc,* 1996.

[22] N. Bajaj, "Enhancement of A5/1: Using variable feedback polynomials of LFSR," in *Emerging Trends in Networks and Computer Communications (ETNCC), 2011 International Conference on*, 2011, pp. 55-60.

[23] M. A. B. Shemaili, C. Y. Yeun, M. J. Zemerly, and K. Mubarak, "A novel hybrid cellular automata based cipher system for internet of things," in *Future Information Technology*, ed: Springer, 2014, pp. 269-276.

[24] B. Colbert, A. H. Dekker, and L. M. Batten, "Heraclitus: A LFSR-based stream cipher with key dependent structure," in *Communications and Signal Processing (ICCSP), 2011 International Conference on*, 2011, pp. 141-145.

[25] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*, 1996.

[26] N. T. Courtois, "Fast algebraic attacks on stream ciphers with linear feedback," in *Advances in Cryptology-CRYPTO 2003*, ed: Springer, 2003, pp. 176-194.

[27] F. Muller, "Differential attacks and stream ciphers," in *The State of the Art of Stream Ciphers, Workshop Record, ECRYPT Network of Excellence in Cryptology*, 2004, pp. 133-146.

[28] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, *et al.*, *PRESENT: An ultra-lightweight block cipher*: Springer, 2007.

[29] S. S. M. AlDabbagh, I. F. T. Al Shaikhli, and M. A. Alahmad, "HISEC: A New Lightweight Block Cipher Algorithm," in *Proceedings of the 7th International Conference on Security of Information and Networks*, 2014, p. 151.

[30] W. Wu and L. Zhang, "LBlock: a lightweight block cipher," in *Applied Cryptography and Network Security*, 2011, pp. 327-344.

[31] E. Dawson, M. Henricksen, and L. Simpson, "The Dragon Stream Cipher: Design, Analysis, and Implementation Issues," in *New Stream Cipher Designs*, ed: Springer, 2008, pp. 20-38.

[32] A. A. Alhamdan, "Secure stream cipher initialisation processes," Doctor of Philosophy, Queensland University of Technology, 2014.

[33] M. Hell, T. Johansson, A. Maximov, and W. Meier, "The Grain family of stream ciphers," in *New Stream Cipher Designs*, ed: Springer, 2008, pp. 179-190.

[34] N. Kumar, S. Ojha, K. Jain, and S. Lal, "BEAN: a lightweight stream cipher," in *Proceedings of the 2nd international conference on Security of information and networks*, 2009, pp. 168-171.

[35] Y. Crama and P. L. Hammer, *Boolean models and methods in mathematics, computer science, and engineering* vol. 2: Cambridge University Press, 2010.

[36] A. Rukhin, J. Soto, J. Nechvatal, E. Barker, S. Leigh, M. Levenson, *et al.*, "Statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST special publication," 2010.

[37] N. Kolokotronis, "Cryptographic properties of nonlinear pseudorandom number generators," *Designs, Codes and Cryptography,* vol. 46, pp. 353-363, 2008.

[38] N. T. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback," in *Advances in Cryptology—EUROCRYPT 2003*, ed: Springer, 2003, pp. 345-359.

[39] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Advances in Cryptology—ASIACRYPT 2000*, ed: Springer, 2000, pp. 1-13.

[40] V. A. Ghaffari and A. Vardasbi, "On the period of GSM's A5/1 stream cipher and its internal state transition structure," in *Information Security and Cryptology (ISCISC), 2011 8th International ISC Conference on*, 2011, pp. 37-40.

[41] G. G. Rose and P. Hawkes, "On the Applicability of Distinguishing Attacks Against Stream Ciphers," *IACR Cryptology ePrint Archive,* vol. 2002, p. 142, 2002.

[42] C. J. Jansen, T. Helleseth, and A. Kholosha, "Cascade jump controlled sequence generator and Pomaranch stream cipher," in *New Stream Cipher Designs*, ed: Springer, 2008, pp. 224-243.

[43] E. Biham and O. Dunkelman, "Differential Cryptanalysis in Stream Ciphers," *IACR Cryptology ePrint Archive,* vol. 2007, p. 218, 2007.

[44] T. E. Schilling and H. Raddum, "Analysis of trivium using compressed right hand side equations," in *Information Security and Cryptology-ICISC 2011*, ed: Springer, 2012, pp. 18-32.