Comprehensive Survey on the Constructional Design of Existing Stream Ciphers

Khaled M. Suwais
School of Computer Sciences, Universiti Sains Malaysia, Malaysia
khaled@cs.usm.my

Azman Samsudin
School of Computer Sciences, Universiti Sains Malaysia, Malaysia
azman@cs.usm.my

ABSTRACT

The purpose of this study is to present a comprehensive survey and extensive analysis of existing stream ciphers. The survey is carried out to satisfy several goals. First, to provide comprehensive literature review that summarizes the constructional design of current stream ciphers. Second, to present a consistent classification of stream ciphers in order to facilitate the understanding of existing stream ciphers, and the development of new stream ciphers. Third, to provide extensive security analysis for existing stream cipher categories.

Key Words: Stream Cipher, Cryptography, Encryption, *NP*-hard Problem.

1. Introduction

Cryptography is fundamental to most computer security applications and it is used to help cryptographic services in securing communication over unsecured channels. Cryptography focuses on issues of securing messages so that only the relevant parties can read the messages. Transforming a message (plaintext) to an incomprehensive form (ciphertext) is accomplished by a process known as encryption. In contrast, transforming an encrypted message to its original form is accomplished by a process known as decryption.

Those transformations (encryption and decryption) are achieved by two classes of cryptographic algorithms: symmetric key and asymmetric key algorithms. In this paper we are focusing on one type of the symmetric key algorithms known as stream ciphers.

In stream ciphers, based on an input key, a sequence of random bits is generated and used as keystream that will never be used again during the run of the cipher. The general structure of stream ciphers is portrayed in Figure 1.

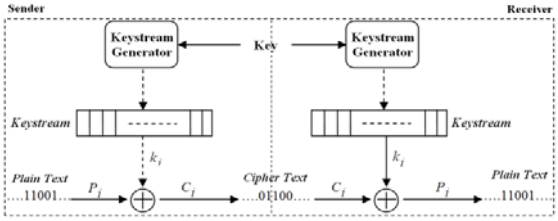

Figure 1: The general design of stream ciphers

As shown in Figure 1, the core of any stream cipher is located in its keystream generator. Therefore in this paper we classify and analyze various stream ciphers from the design perspectives of the keystream generator.

This study is conducted due to the importance of stream ciphers in securing information, which is considered as the most strategic resources. Furthermore, the study aims to fulfill several objectives, including: to provide comprehensive literature review that summarize the constructional designs of existing stream ciphers, to shape a clear vision of stream ciphers designs through a consistent classification that can assist the development process of new stream cipher, and finally to provide an extensive security analysis for several existing stream ciphers categories.

The rest of the paper is organized as follows: Section 2 presents a comprehensive classification and categorization of existing stream ciphers, supported by security analysis for each category. In Section 3, we discuss the general properties of each category from the design and security perspectives. Lastly, the conclusion of our survey paper is presented in Section 4.

2. Stream Ciphers Classification

Stream ciphers are classified into three fundamental categories: hardware-based, software-based and hybrid design stream ciphers. In this

section we explore and analyze the important features and properties of each category of the stream ciphers.

2.1 Hardware Based Stream Ciphers

The use of hardware implementations was significant in providing the security for various cryptographic applications. The majority of stream ciphers designs rest on the use of different types of shift-registers in their implementation. The majority of these ciphers are either rely on Linear Feed-back Shift Register $(LFSR)$, Non-Linear Feed-back Shift Register $(NLFSR)$, Feedback Carry Shift Register $(FCSR)$, or on a combination of two types of shift registers. The second level of our classification shows that the hardware-based stream ciphers are divided into three categories: $LFSR$, $NLFSR/FCSR$ and Clock Control stream ciphers, as shown in Figure 2.

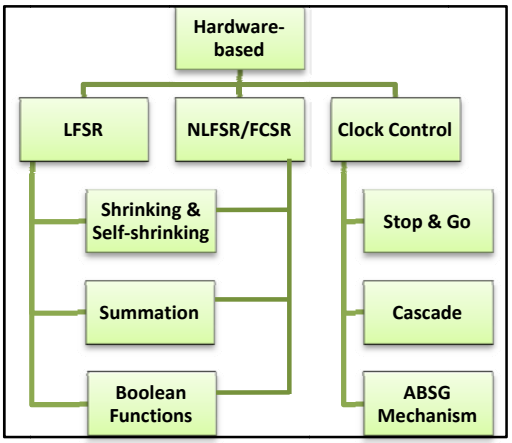

Figure 2: The classification of hardware-based stream ciphers

2.1.1 LFSR Based Stream Ciphers

An $LFSR$ is a shift register which is able to hold one symbol at a time and its input is a linear combination of the previous state. $LFSRs$ rely on system clocks for their operations, in which system clock is responsible for the timing of all events. With every clocking of the $LFSR$, the registers read a new symbol from the input, and the symbols move forward from register $\ell - 1$ to register 0.

One important feature of $LFSR$ is its ability to produce an extremely long pseudorandom sequence equal to 2^{n-1}, where n is the number of register elements in the $LFSR$. $LFSR$ was believed to be able to deliver stream cipher with uniformed distribution of the values generated by the keystream generator. The immediate output of $LFSR$ is not acceptable to be used as keystream since the output is produced in a linear fashion [1]. If an attacker knows the feedback coefficients $(\boldsymbol{C}_0, \boldsymbol{C}_1, \dots, \boldsymbol{C}_{\ell-1})$, the attacker can use ℓ random keystream bits to reconstruct a system of linear equations. This attack is feasible with complexity of $\boldsymbol{O}(\ell^3)$ for any parameter ℓ. On the other hand, if the feedback coefficients are unknown, the inner seed can be reconstructed with 2ℓ consecutive keystream bits. This attack with complexity of $\boldsymbol{O}(\ell^3)$, is also feasible since it is based on solving a system of 2ℓ linear equations.

In order to use $LFSRs$ in generating keystreams with minimum level of security, non-linear functions have to be merged with $LFSRs$ to make the bit production process after each clocking work in non-linear fashion. To achieve this purpose, different techniques had been introduced such as adding non-linear filters, non-linear updates, irregular clocking to remove the linearity found in $LFSRs$.

The non-linearity function can be provided by the following generators:

- ***Shrinking and Self-Shrinking Generator***: Coppersmith, Krawczy and Mansour proposed in [2] a new generator which consists of two $LFSRs$ which was named, shrinking generator. The shrinking generator is designed as pseudorandom keystream generator and it is preferred due to the simplicity of its design. Each one of the $LFSR$ produces a bit stream represented by a_i and b_i produced by $LFSR$-A and $LFSR$-B respectively, to form the keystream K_s. However, shrinking generators are subjected to known-plaintext distinguishing attack which is first introduced in [3]. The attack had detected some non-randomness in the distribution of the keystream bits.

  Self-Shrinking generator [4] is another variant of the shrinking generator concept. The generator rests on single $LFSR$ instead of using two different $LFSRs$ as in the shrinking generator. The procedure of clocking self-shrinking generators is done by firstly clocking two bits from the $LFSR$, resulting in a pair of bits (a_1, a_2). If (a_1, a_2) equals to the value (1,0) or (1,1), then it is taken as a pseudorandom bit 0 or 1 respectively. If the pair equals the value (0,0) or (0,1), the pair will be discarded [4].

  Let $a = (a_0, a_1, a_2, \dots)$ be the output bits of a non-trivial initialized self-shrinking $LFSRs$ of length N. Therefore, a is a sequence with period $2^N - 1$. With respect to the period of a, cryptanalysis attack in [4] showed that if the period of a is at least $2^{N/2}$ and the linear complexity of the construction is $2^{\frac{N}{2-1}}$, an attacker can attack the construction in $2^{0.7N}$ steps.

- ***Summation Generator***: Rainer Rueppel introduced a new generator based on the use of $LFSRs$ called summation generator [5]. The idea behind this generator relies on the non-linearity provided by the carry-in integer addition. Rueppel uses the output of several $LFSRs$ through an adder with carry, which in turn can provide a combination function with good non-linearity and high-order correlation properties [6].

In term of the security Rueppl's generator is subjected to correlation attacks since the probability of input-output correlation is of 0.5 [7]. One example of the summation generator is the E0 stream cipher which is used in the Bluetooth protocol [8]. However, various cryptanalysis and statistical attacks on E0 had been presented in [9], making E0 stream cipher not secure for cryptographic applications.

Another example where summation generation is used, is a parallelized stream cipher presented in [10]. Few years later, an algebraic attack against the generator was presented in [11], making the parallelized stream cipher subjected to security vulnerability.

- ***Boolean Functions***: In mathematics, a Boolean function is defined as a mapping of one or more binary input variables $\boldsymbol{L_k}$ to one binary output variable $\boldsymbol{L}$. Formally, we write the mapping function as follows:

$$\boldsymbol{\beta}: \boldsymbol{L_k} \rightarrow \boldsymbol{L}$$

An interesting property of Boolean functions which attract several cryptographic applications is the balancing of the digits zero and one in the generated sequence. Generally, a Boolean function is said to be balanced if the probability of that function is 0.5 for all input variables chosen uniformly over a binary field.

Examples of stream ciphers based on the combination of $LFSRs$ and Boolean functions are found in A5/1 and LILI-128 stream ciphers. A5/1 was developed in 1987 and later became the most popular stream cipher in most European countries and United States to provide over-the-air communication privacy in GSM cellular telephone standard [12]. The cipher is working in conjunction with three $LFSRs$ (L-A, L-B, L-C) with irregular clocking. The main idea of A5/1 is to mix the cycled bits generated by the three $LFSRs$ with respect to the irregularity in the clocking process. However, A5/1 seems to be vulnerable to cryptanalysis attacks presented in [13] and [14].

LILI-128 is another stream cipher which was introduced in [15]. It uses two binary $LFSRs$ and two functions to generate a pseudorandom binary keystream. Nevertheless, two attacks presented in [16] and [17] make LILI-128 not secure.

Finally, there are many other examples of stream ciphers using different techniques (functions, filters, etc) in conjunction with $LFSRs$ to achieve higher security. One example is the stream cipher SNOW [18]. However, SNOW was also attacked in [19] and makes it infeasible to be used in secure applications.

2.1.2 NLFSR/FCSR Based Stream Cipher

Non-Linear Feedback Shift Register ($NLFSR$) and Feedback with Carry Shift Register ($FCSR$) are another two types of shift registers used in stream ciphers. The main purpose of these registers is to eliminate and destroy the linearity found in $LFSRs$. The design of $NLFSR$ applies a non-linear function in the shift register to ensure the non-linearity in the output values from the corresponding shift register.

$NLFSRs$ are used in several stream cipher designs such as the Grain stream cipher. Grain was developed in 2004 and submitted to eSTREAM project for evaluation in 2005 [20]. However, Grain was attacked in 2006 by two cryptanalysts as found in [21] and [22].

$FCSRs$ are similar to $LFSR$ with the difference that the elementary addition in $FCSR$ is with propagation of carries instead of addition modulo 2 as in $LFSR$. An example of $FCSR$-based stream cipher is the $F\text{-}FCSR$ stream cipher, which was developed recently and submitted for eSTREAM project evaluation [23]. However, $F\text{-}FCSR$ was attacked in [24] due to the weaknesses found in the initialization mechanisms as well as lacking entropy on the internal state.

2.1.3 Clock Control Based Stream Cipher

One way of introducing the non-linearity in the generated keystream is by having a shift register clocked irregularly. Figure 3 shows an example on clock controlled generator called the Altering Step generator, where the output of one $LFSR$ is controlling the other $LFSRs$.

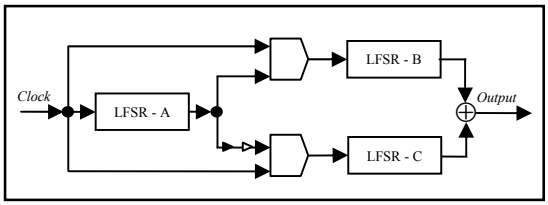

Figure 3: Clock controlled Altering-Step generator

There are various generators that are based on the idea of clock-controlling in shift registers for cryptographic purposes (refer to Figure 2). Some of these generators are:

- ***Stop-and-Go Generator***: Stop-and-Go generator was first introduced by Beth and Piper [25]. The idea of this generator is to let a control register R-A control the stepping of another register R-B. Beth and Piper believe that the stop-and-go generator is secure and immune against cryptanalysis attacks. However, the generator was subjected to efficient cryptanalysis attacks found in [26] and [27].
- ***Cascades Generator***: Cascade generator is basically an extension of the stop-and-go generator, such that it is still relying on the idea that $LFSRs$ are controlling each other. There are two types of

cascades [6]: The first type allows each register to generate *l*-sequence and the second type restricts the length of each register to a prime length *N* with no feedback from any intermediate stage of the register. One example of the cascade stream cipher is the Pomaranch stream cipher which is based on a Jump Controlled Sequence Generator (cascade) and was submitted to eSTREAM project for evaluation [28]. Unfortunately, Pomaranch was vulnerable to several cryptanalysis attacks found in [29] and [30].

- ***ABSG Mechanism***: ABSG is inspired by the shrinking and self-shrinking generator. Its main purpose was to provide irregularity for the generated keystream bits. Unlike shrinking generators, ABSG operates on a single input variable instead of two. ABSG also differs from the self-shrinking generator in that the production of *n*-bits of output sequence requires approximate 3*n*-bits of input, while in self-shrinking, the production requires 4*n*-bits of input sequence [31]. The stream cipher DECIM-128 presented in [32] is based on the use of $LFSRs$ and ABSG decimation mechanism. The process of generating keystreams rests on the non-linearity filtered $LFSR$ and the irregular decimation mechanism of ABSG. However, the attack presented in [33] showed that DECIM-128 is suffering from serious flaws in the initialization stage and the keystream generation algorithm stage.

2.2 Software-Based Stream Ciphers

In contrast to hardware-based stream ciphers, there are various designs of stream ciphers which are based on bits manipulation (substitution, permutations, etc), Boolean functions, and other simple logical and mathematical operations. In this section we introduce a variety of stream cipher designs associated to different sub-category of software-based stream ciphers as shown in Figure 4.

2.2.1 T-Function Based Stream Ciphers

In 2003, Klimov and Shamir introduced a new type of invertible round function (called T-Function) by mixing short arithmetic and Boolean operations on full machine words [34]. The name T-function refers to the triangular dependence between the columns of the operands. The function works as a mapping function formulated as follows:

$$\mathcal{F}: \boldsymbol{B}^{m\times n} \rightarrow \boldsymbol{B}^{k\times n}$$

where $\boldsymbol{B} = \{0,1\}$ is represented by a matrix and there is a dependency between the *k-th* column of the output with the first *k* set of input columns.

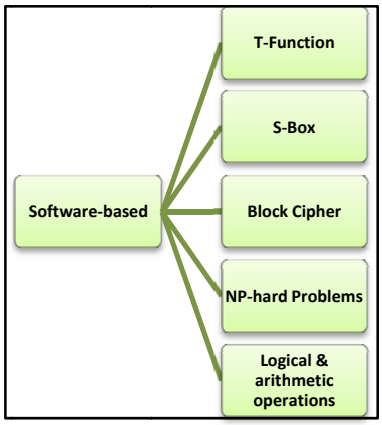

Figure 4: The classification of software-based stream ciphers

T-functions are recent development in stream cipher; therefore only few stream ciphers appear in the literatures that are based on T-function. One example is the stream cipher TSC-1 proposed in [35]. The proposed cipher is based on a single cycle T-function. TSC-1 works in conjunction with a filter function and 4✕4 S-Box.

In general, T-function was subjected to several attacks such as the correlation attack based on the linear approximation of the T-function. The attack was successfully applied on TSC-128 with a complexity of 2^{24} known keystream bits to distinguish it from random [36]. The other attack presented in [37] describes a distinguish attack on single-word and multi-word T-functions based on the deviation found in the integer differences of consecutive outputs with a complexity of 2^{32}.

However, it seems that researchers need to put more efforts on developing and enhancing the security aspects of T-function. The importance of T-function comes from the efficiency of its hardware and software implementations.

2.2.2 S-Box Based Stream Ciphers

A substitution box or also known as S-box is an important component for different cryptographic primitives. S-box basically works as a mapping of *m* input bits into *n* output bits, resulting in an $m \times n$ S-box.

The design of S-boxes comes in two types: fixed and dynamic S-boxes. Fixed S-boxes rest on pre-computed values calculated in several ways based on the cryptographic component being used. Dynamic S-boxes are more interesting since the values in the S-box change during the execution.

In this category of stream ciphers, we found few ciphers whose designs are based on S-boxes. Two examples are discussed here: MUGI and WAKE stream ciphers. MUGI stream cipher was introduced as an efficient stream cipher in hardware and software

implementations [38]. The internal state of MUGI consists of two internal states (state *a* and buffer *b*) updated by two identical functions (called F-function). The F-function uses three main techniques: key addition, non-linear S-box and MDS matrix for linear transformation.

MUGI is not broken yet. However a weakness found in the linear part of MUGI was presented in [39], proved that the real response of the buffer without the feedback from the S-box consists of binary linear recurring sequences with linear complexity and very small period of 48 cycles. This theoretical analysis showed that by using the weakness above, cryptanalysts can use linear cryptanalysis to attack MUGI.

Another example of stream cipher in this category is the WAKE (Word Auto Key Encryption) stream cipher [40]. WAKE has a simple structure and performs fast. The generation of new key depends on the ciphertext produced in the previous round. WAKE uses S-box of 256 32-bit values with special property where some bytes are obtained from a permutation of all possible bytes, and some other bytes are generated randomly.

WAKE was subjected to a chosen plaintext or chosen ciphertext attack, which was fully analyzed in [41]. The analysis includes implementing two chosen plaintext attacks on WAKE with a complexity of $10^{19.2}$ and $10^{14.4}$ for the first and second attacks respectively.

However, it seems that S-boxes are efficient in providing non-linearity with efficient performance in the internal states of the keystream generators. Designing a strong and cryptographic S-box is not easy. Therefore, any misuse of S-boxes in stream cipher leads to serious security vulnerabilities.

2.2.3 Block Cipher Based Stream Ciphers

This is another approach used in the design of stream ciphers. The block cipher is used as a core of the keystream generator of the corresponding stream cipher, such as using AES in LEX [42].

Another design philosophy of stream ciphers in this category is based on the Substitution-Permutation Network (SPN) of block ciphers instead of using the components of block ciphers, as appeared in Hermes8 stream cipher [43]. The security of such a design relies on the underlying block cipher that resides at the core of the stream cipher. Up to this day, among the submitted stream ciphers based on block ciphers, LEX [42] and Sosemanuk are the only two ciphers which have moved to the third phase of eSTREAM evaluation.

2.2.4 NP-hard Problem Based Stream Ciphers

A Non-deterministic Polynomial-time hard problem (*NP*-hard) is a mathematical problem that is considered as intractable and cannot be solved in polynomial time. *NP*-hard problems are widely used in several cryptographic primitives, and they proved to provide high level of security.

A problem is *NP*-hard if every problem that is *NP* can be translated into an *NP*-hard problem in polynomial time. An *NP*-hard problem therefore is as hard as any other *NP*-problem. In fact, it is harder. No one knows a polynomial time solution to any *NP*-hard problem yet; the best known solutions are exponentially explosive. Hence, *NP*-hard problems are generally referred to as computationally intractable.

Examples on popular *NP*-hard problems include: Discrete Logarithm Problem (DLP), Elliptic Curve Discrete Logarithm Problem (ECDLP) and Traveling Salesman Problem (TSP). These *NP*-hard problems and many others have been used in several security applications. Recently, new stream ciphers were proposed based on the intractability of these *NP*-hard problems, such as: ECSC-128 and DSP-128 stream ciphers.

ECSC-128 is a stream cipher that is based on the intractability of ECDLP [44]. ECDLP is formally defined as follows:

Definition 1: Given the points $\boldsymbol{P}$ and $\boldsymbol{Q}$ on elliptic curve $\boldsymbol{E}$ defined over a finite field with q (large prime number) elements of $\boldsymbol{F}_q$, find the integer k such that $\boldsymbol{Q} = k\boldsymbol{P}$.

ECSC-128 deploys the concept of point multiplication over elliptic curve (ECDLP) for generating keystream. Therefore, ECDLP is considered the essential element in providing ECSC-128 with higher level of security. Evaluating the security of ECSC-128 is correlated to the evaluation of ECDLP. Therefore, ECSC-128 is a secure stream cipher since it is based on an intractable *NP*-hard problem.

The other stream cipher in this category is the DSP-128 stream cipher, which is based on DLP [45]. DLP is formally defined as follows:

Definition 2: Given $z, y \in G$, find the integer x such that $z = y^x$, where G is a finite field on n elements.

DSP-128 deploys DLP in its keystream generator to generate keystream for plaintext encryption. The input key of DSP-128 is used as x that is appeared in Definition 2. Since the generation of new keystream is continuous through the encryption process, the value of x is incremented accordingly. DSP-128 is yet another secure stream cipher in this category, due to the intractability of the DLP that is being used for generating keystream.

2.2.5 Logical and Arithmetic Operation Based Stream Ciphers

There are stream ciphers which do not fit into the mentioned categories above. Some of these ciphers are based on bitwise addition and bits rotation

operations as in Phelix, SEAL and RC4, while others based on mixing various functions in conjunction with some addition and rotation operations as in Rabbit. In this section we briefly describe some important stream ciphers that fit in this category.

- ***Phelix Stream Cipher***: Phelix stream cipher [46] is a high speed stream cipher selected for the software and hardware profiles of eSTREAM project. The main operations of Phelix are: addition modulo 2^{32}, bitwise XOR and rotation operations. The state of Phelix is broken into two groups: five state words called *active* states which are always participating in updating the internal function and four states called *old* state which is only used in the process of keystream generation.

One can notice that the main operations in one block of Phelix are low-cost operations, in which they are fast in software and hardware implementations. However, Phelix has not moved to the third phase of the eSTREAM project evaluation due to some security vulnerability. Differential-linear attacks presented in [47], showed that with the assumption of reusing the nonce, the key of Phelix can be recovered with complexity 2^{37} chosen plaintext words and $2^{41.5}$ operations.

- ***Rabbit Stream Cipher***: Rabbit is another design of stream ciphers based on iterating a set of coupled non-linear functions – or as authors called them discretized chaotic maps [48]. The inner state of Rabbit consists of 513 bits. The first 512 bits represent 8-state variables $(x_0, \dots, x_7)$ of 32-bit length each and 8-counter variables $(c_0, \dots, c_7)$. The remainder bit is used as a counter carry bit, b.

It seems that Rabbit stream cipher is strong against cryptanalysis attacks. It is selected among few other ciphers for further evaluation by eSTREAM project.

- ***RC4 Stream Cipher***: This is yet another important example of stream cipher design. The well known stream cipher is widely used in many security protocols and software applications. Generating keystream in RC4 comprises two algorithms: The Key-Scheduling Algorithm (KSA) and the Pseudo-Random Generation Algorithm (PRGA). The KSA algorithm uses a permutation array $\boldsymbol{S}$ of all 256 possible bytes.

At the present time, RC4 is not recommended for use in new applications. Several weaknesses of the KSA algorithm of RC4 [49] can be summarized in two points. First, weakness is the existence of massive classes of weak keys. These classes of weak keys enable the attackers to determine a large number of bits from the KSA output by using a small part of the secret key. Thus, the initial outputs of the weak keys are disproportionately affected by a small portion of key bits. The second weakness rests on a related key vulnerability.

Other kinds of attacks on RC4 have been presented recently. Results in [50] showed a statistical bias of the digraphs distribution of the generated stream of RC4. Furthermore, a distinguishing attack had been developed based on the statistical bias found in the output sequences [51]. This bias is used along with the first two words of a keystream associated with around 2^{30} secret keys.

2.3 Hybrid Design

In this category we discuss other designs of stream ciphers based on a combination of hardware devices and software techniques to achieve their security. Most of the ciphers in this category depend on $LFSRs$ as the main component of the stream cipher. The software techniques vary from using T-function as in ABC stream cipher [52], look-up tables as in ORYX [55], and other techniques.

- ***ABC Stream Cipher***: ABC is a stream cipher algorithm that is submitted for eSTREAM project for evaluation [52]. ABC consists of 38, 32-bit registers. The registers are divided into two groups: 3 registers (z^0, z^1, x) are representing the state of ABC, and 35 registers $(d_0, d_1, e, e_0, \dots, e_{31})$ represent the constant parameters fed to the cipher. In conjunction with the $LFSRs$, ABC uses three main functions (***A***: linear function, ***B***: T-function and ***C***: non-linear mapping function).

In terms of the security, several attacks on ABC make it fail moving to the third phase of eSTREAM project. Based on the weakness of function ***C*** as illustrated in [53], a correlation based divide-and-conquer attack was able to find 63-bit of the state by searching 2^{63} possible choices. Furthermore, a fast correlation attack on ABC was presented in [54]. The attack depends on some weak keys to recover the internal state.

- ***ORYX Stream Cipher***: ORYX is a stream cipher algorithm, proposed for use in North American digital cellular systems [55]. The structure of ORYX is based on binary $LFSRs$, S-box (look-up table) and permutation. The keystream generation is performed by clocking the three $LFSRs$ along with some fixed permutations in order to obtain the high bytes of the current state of each $LFSR$ using a combining function.

ORYX is not a secure stream cipher due to the efficient attack presented in [56]. The attack can recover the full 96 bits internal state using only 25-27 bytes of known plaintext with time complexity of 2^{16}.

3. Discussion of Stream Ciphers Categories

We have seen that researchers have made great efforts to improve the security of stream ciphers. In this paper we explored different categories of stream cipher designs. Each category has important features which make it different from the other categories.

In term of efficiency, stream ciphers are generally implemented to be efficient on both hardware and software platforms. Throughout the survey we found that some stream ciphers tends to be more efficient on hardware as discussed in the category of hardware-based stream ciphers. From the other perspective, the rest of stream ciphers rely on simple bit manipulation and mapping functions that make them more efficient from software perspective.

From the security perspective, the analysis of each category shows that several stream ciphers were subjected to cryptanalysis attacks. Those attacks generally targeted at the keystream generator or as in some other stream ciphers it is called the next-state function.

Based on our classification, we classify the attacks on stream ciphers into two types: hardware-based attacks and software-based attacks. In both types of the attacks, attacker tries to extract useful information from the keystream generator of the corresponding stream cipher. For instance, hardware-based attacks are generally focused on utilizing the linearity characteristic found in *LFSRs* and extracting pattern from the resulted keystream, to attack the overall process of keystream generation. On the other hand, software-based attacks rely on the simplicity of software-based stream ciphers to attack the keystream generator. Therefore, stream ciphers which are based on *NP*-hard problems form the new direction for alternative software-based stream ciphers due to their irresistibility against cryptanalysis attacks.

4. Conclusion

This paper had classified stream ciphers into categories, where each category includes stream ciphers of similar properties. We found that researchers had put a considerable effort in developing new and fast secure stream ciphers. Unfortunately, wide range of stream ciphers are found vulnerable to cryptanalysis attacks. However, this study showed that several cryptographic primitives including stream ciphers, tends to make use of *NP*-hard problems in their implementations. The utilization of *NP*-hard problem had granted these primitives a higher level of security. Future stream ciphers are required to offer an optimum level of security, and at the same time, utilize the existing of multicore processors for optimum performance.

References

[1] E. Zenner, *Cryptanalysis of LFSR-based Pseudorandom Generators - a Survey,* Reihe Informatik, 2004, Survey, http://th.informatik.uni-mannheim.de/pub/zenner04b.pdf.

[2] D. Coppersmith, H. Krawczyk and Y. Mansour, The Shrinking Generator, *Advances in Cryptology-CRYPTO'93,* Springer - Verlag, 1993, Vol. 773, pp. 22-39.

[3] P. Ekdahl, W. Meier and T. Johansson, Predicting the Shriking Generator with Fixed Connections, E. Biham, *Advances in Cryptology - EUROCRYPT2003,* Springer, 2003, Vol. 2656 of LNCS, pp. 330-344.

[4] W. Meier and O. Staffelbach, The Self-Shrinking Generator, *Eurocrypt 94,* Springer, 1994, Vol. 950 of LNCS, pp. 205-214.

[5] R. Rueppel, Correlation immunity and the Summation Generator, *Advances in Cryptography-EUROCRYPT '85,* Berlin : Springer, 1986, pp. 260-272.

[6] M. Robshaw, *Stream Ciphers,* CA : RSA Laboratories, 1995, Technical Report.

[7] M. Park and D. Park, A New Stream Cipher Using Two Nonlinear Functions, *Computational Science and Its Applications – ICCSA 2005,* Berlin : Springer, 2005, Vol. 3481 of LNCS, pp. 235-244.

[8] M. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos and C. Goutis, *Comparison of the Hardware Implementation of Stream Ciphers,* 2005, The International Arab Journal of Information Technology, Vol. 2, pp. 267-274.

[9] Y. Lu and S. Vaudenay, Faster Correlation Attack on Bluetooth Keystream Generator E0, *Advances in Cryptology – CRYPTO 2004.* Berlin : Springer, 2004, Vol. 3152 of LNCS, pp. 407-425.

[10] H. Lee and S. Moon, *Parallel stream cipher for secure high-speed communications,*2002, Signal Processing, pp. 259-265.

[11] D. Han and M. Lee, *An algebraic attack on the improved summation generator with 2-bit*

memory, 2005, Information Processing Letters, Vol. 93, pp. 43 - 46.

[12] E. Biham and O. Dunkelman, Cryptanalysis of the A5/1 GSM Stream Cipher. *Progress in Cryptology —INDOCRYPT 2000.* Berlin : Springer, 2000, Vol. 1977, pp. 43-51.

[13] A. Biryukov, A. Shamir and D. Wagner. *Real Time Cryptanalysis of A5/1 on a PC,* New York : 2000, In Proc. Fast Software Encryption. pp. 1-18.

[14] E. Barkan, E. Biham and N. Keller, Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, *Advances in Cryptology - CRYPTO 2003,* Berlin : Springer, 2003, Vol. 2729 of LNCS, pp. 600-616.

[15] E. Dawson, A. Clark, J. Golic, W. Millan, L. Penna and L. Simpson. Heverlee. *The LILI-128 Keystream Generator,* Belgium 2000, In Proc. of First NESSIE Workshop.

[16] F. Jönsson and Johansson. *A fast correlation attack on LILI-128,* 2002, Information Processing Letters, Vol. 81, pp. 127 - 132.

[17] Y. Tsunoo, T. Saito, M. Shigeri, H. Kubo and K. Minematsu, *Shorter Bit Sequence Is Enough to Break Stream Cipher LILI-128,* 12, 2005, Vol. 51, pp. 4312-4319.

[18] P. Ekdahl and T. Johansson, A New Version of the Stream Cipher SNOW, *Selected Areas in Cryptography,* Berlin : Springer, 2003, Vol. 2595 of LNCS, pp. 47-61.

[19] D. Coppersmith, S. Halevi and C. Jutla, Cryptanalysis of stream ciphers with linear masking, *Advances in Cryptology - CRYPTO'02,* Springer, 2002, Vol. 2442 of LNCS, pp. 515-532.

[20] H. Hell, T. Johansson and W. Meier, Grain - A Stream Cipher for Constrained Environments, *The eSTREAM Project,* [Online] April 29, 2005. [Cited: May 26, 2008.] http://www.ecrypt.eu.org/stream/ciphers/grain/grain.pdf.

[21] A. Maximov, *Cryptanalysis of the "Grain" family of stream ciphers,* Taipei, Taiwan : ACM, 2006, In Proc. of the 2006 ACM Symposium on Information, computer and communications security . pp. 283 - 288.

[22] O. Kucuk, Slide Resynchronization Attack on the Initialization of Grain 1.0, *The eSTREAM Project,* [Online] July 16, 2006. [Cited: May 25, 2008.] http://www.ecrypt.eu.org/stream/papersdir/2006/044.ps.

[23] F. Arnault, T. Berger and C. Lauradoux, Update on F-FCSR Stream Cipher, *The eSTREAM Project,* [Online] January 2, 2006. [Cited: May 26, 2008.] http://www.ecrypt.eu.org/stream/papersdir/2006/025.pdf.

[24] É Jaulmes and F. Muller, Cryptanalysis of the F-FCSR Stream Cipher Family, *Selected Areas in Cryptography,* Berlin : Springer, 2006, Vol. 3897 of LNCS, pp. 20-35.

[25] T. Beth and F. Piper, *The stop-and-go generator,* Paris, France : Springer-Verlag, 1985, In Proc. of the EUROCRYPT 84 workshop on Advances in cryptology: theory and application of cryptographic techniques , pp. 88 - 92.

[26] A. Menezes, P. Oorschot and S. Vanstone, *Handbook of Applied Cryptography,*. Boca Raton, FL : CRC Press, 1997.

[27] D. Golic and R. Menicocci, *Edit probability correlation attacks on stop/go clocked keystream generators,* 2003, Journal of cryptology, Vol. 16, pp. 41-68.

[28] T. Helleseth, C. Jansen and A. Kholosha, Pomaranch - Design and Analysis of a Family of Stream Ciphers, *The eSTREAM Project.* [Online] January 2, 2006, [Cited: May 27, 2008.] http://www.ecrypt.eu.org/stream/papersdir/2006/008.pdf.

[29] H. Englund, M. Hell and T. Johansson, Two General Attacks on Pomaranch-Like Keystream Generators, *Fast Software Encryption,* Berlin : Springer, 2007, Vol. 4593 of LNCS, pp. 274-289.

[30] C. Cid, H. Gilbert and T. Johansson, *Cryptanalysis of Pomaranch.* 2006, In Proc. of IEE Information Security, Vol. 153, pp. 51-53.

[31] M. Afzal, F. Kausar and A. Masood, *Comparative Analysis of the Structures of eSTREAM Submitted Stream Ciphers,* Peshawar, Pakistan : IEEE-ICET, 2006, In Proc. The Second International Conference on Emerging Technologies, pp. 245-250.

[32] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin and H. Sibert, DECIM-128, *The eSTREAM Project,* [Online] April 29, 2005, [Cited: May 26, 2008.] http://www.ecrypt.eu.org/stream/p3ciphers/decim/decim128_p3.pdf.

[33] H. Wu and P. Preneel, Cryptanalysis of the Stream Cipher DECIM, *Fast Software Encryption,* Berlin : Springer, 2006, Vol. 4047 of LNCS, pp. 30-40.

[34] A. Klimov and A. Shamir, A New Class of Invertible Mappings, *Cryptographic Hardware and Embedded Systems - CHES 2002,* London, UK : Springer, 2003, Vol. 2523 of LNCS, pp. 470-483.

[35] J. Hong, D. Lee, Y. Yeom and D. Han, A New Class of Single Cycle T-Functions, *Fast Software Encryption,* Berlin : Springer, 2005, Vol. 3557 of LNCS, pp. 68-82.

[36] F. Muller and T. Peyrin, Linear Cryptanalysis of the TSC Family of Stream Ciphers, *Advances in Cryptology - ASIACRYPT 2005,* Berlin : Springer, 2005, Vol. 3788 of LNCS, pp. 373-394.

[37] S. Künzli, P. Junod and W. Meier, Distinguishing Attacks on T-Functions, *Progress in Cryptology – Mycrypt 2005,* Berlin : Springer, 2005, Vol. 3715 of LNCS, pp. 2-15.

[38] D. Watanabe, S. Furuya, H. Yoshida, K. Takaragi and B. Preneel, A New Keystream Generator MUGI, *Fast Software Encryption,* Berlin : Springer, 2002, Vol. 2365 of LNCS, pp. 179-194.

[39] J. Golic, A Weakness of the Linear Part of Stream Cipher MUGI, *Fast Software Encryption,* Berlin : Springer, 2004, Vol. 3017 of LNCS, pp. 178-192.

[40] D. Wheeler, A Bulk Data Encription Algorithm, *Fast Software Encryption, Cambridge Security Workshop,* London, UK : Springer, 1993, Vol. 809 of LNCS, pp. 127 - 134.

[41] M. Pudovkina, Analysis of chosen plaintext attacks on the WAKE Stream Cipher, *eprint,* [Online] 2001. [Cited: May 29, 2008.] http://eprint.iacr.org/2001/065.pdf.

[42] A. Biryukov, A new 128 bit key stream cipher : LEX, *The eSTREAM Project,* [Online] April 29, 2005. [Cited: June 2, 2008.] http://www.ecrypt.eu.org/stream/ciphers/lex/lex.pdf.

[43] U. Kaiser, Hermes Stream Cipher, *eSTREAM PHASE 2,* [Online] April 29, 2005, [Cited: May 20, 2008.] http://www.ecrypt.eu.org/stream/ciphers/hermes8/hermes8.pdf.

[44] K. Suwais and A. Samsudin, *ECSC-128: New Stream Cipher Based on Elliptic Curve Discrete Logarithm Problem.* Famagusta: Trafford, 2007, First International Conference on Security of Information and Networks (SIN 2007), pp. 13-23.

[45] K. Suwais and A. Samsudin, *DSP-128:New Stream Cipher Based on Discrete Log Problem And Polynomial Arithmetic,* 2008, American Journal of Applied Sciences, vol 5(7), pp. 896-904.

[46] D. Whiting, B. Schneier, S. Lucks and F. Muller, Phelix: Fast Encryption and Authentication in a Single Cryptographic Primitive, *The eSTREAM Project,* [Online] April 29, 2005, [Cited: May 12, 2008.] http://www.ecrypt.eu.org/stream/ciphers/phelix/phelix.pdf.

[47] H. Wu and B. Preneel, Differential-Linear Attacks Against the Stream Cipher Phelix, *Fast Software Encryption,* Berlin : Springer, 2007, Vol. 4593 of LNCS, pp. 87-100.

[48] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen and O. Scavenius, Rabbit: A New High-Performance Stream Cipher, *Fast Software Encryption,* Springer, 2003, Vol. 2887 of LNCS, pp. 307-329.

[49] S. Fluhrer, I. Mantin and A. Shamir, Weaknesses in the Key Scheduling Algorithm of RC4, *Selected Areas in Cryptography,* Berlin : Springer, 2001, Vol. 2259 of LNCS, pp. 1-24.

[50] I. Mantin, Predicting and Distinguishing Attacks on RC4 Keystream Generator, *Advances in Cryptology,* Berlin : Springer, 2007, Vol. 3494 of LNCS, pp. 491-506.

[51] Y., Kubo and T. Suzaki, *A Distinguishing Attack on a Fast Software-Implemented RC4-Like Stream Cipher,* Tsunoo, IEEE Computer Society, 2007, IEEE Trans. on Information Theory. Vol. 53, pp. 3250-3255.

[52] V. Anashin, A. Bogdanovy, I. Kizhvatov and A. Kumar, ABC: A New Fast Flexible Stream Cipher, *The eSTREAM Project,* [Online] April 29, 2005, [Cited: May 20, 2008.] http://www.ecrypt.eu.org/stream/ciphers/abc/abc.pdf.

[53] S. Khazaei, Divide and conquer attack on ABC stream cipher, *eSTREAM, ECRYPT Stream Cipher Project,* [Online] 2005, [Cited: May 15, 2008.] http://www.ecrypt.eu.org/stream.

[54] H. Zhang, L. Li and X. Wang, Fast Correlation Attack on Stream Cipher ABC v3, [Online] 2006, [Cited: May 18, 2008.] http://www.ecrypt.eu.org/stream/papersdir/2006/049.pdf.

[55] ORYX, Wiki, [Online] September 29, 2007, [Cited: April 25, 2008.] http://wapedia.mobi/en/ORYX.

[56] D. Wagner, L. Simpson, E. Dawson, J. Kelsey, W. Millan and B. Schneier, Cryptanalysis of ORYX, *Selected Areas in Cryptography,* Springer, 1998, Vol. 1556 of LNCS, pp. 296-305.