University of Tripoli
Faculty of Science
Department of Computer science

Enhancement of Grain Stream Cipher Using Dynamic Linear Feedback Shift Register

Submitted by
Fayrouz Mohamed Aljadi

Supervised by
Dr. Ibrahim Almerhag

A Thesis Submitted to the Department of Computer Science

In Partial Fulfillment of the Requirements for the Degree

Master of Science in Computer Science

Fall 2017

DECLARATION

I Fayrouz Mohamed Aljadi the undersigned hereby confirm that the work contained in this thesis / dissertation, unless otherwise referenced is the researcher's own work, and has not been previously submitted to meet requirements of an award at this University or any other higher education or research institution, I furthermore, cede copyright of this thesis / dissertation in favour of University of Tripoli.

Name: Fayrouz Mohamed Aljadi

Signature: ....

Date: .... / ..../ 20

ABSTRACT

Stream ciphers are commonly used to provide confidentiality for a wide range of frame based applications such as mobile devices or embedded systems. For these applications, stream ciphers are preferred for encryption due to the simplicity of their implementation, efficiency and high throughput. However, practical attacks have been discovered on well-known stream ciphers. Many stream ciphers are designed to resist these attacks. The majority of these ciphers have a fixed structure, which is an advantage that their security against the known attacks can be proved. However, the fixed building structure of these ciphers also provides opportunities for potential new attacks. There have been a few approaches that have tried to add dynamicity to the structure of these ciphers to improve their security level. The idea behind this is that when the structures of ciphers are unknown to attackers, they are more resistant to attacks. However, these ciphers are not widely discussed among researchers. Moreover, the existing research concerning stream ciphers with dynamic structures has focused on dynamic polynomial switching in the Linear Feedback Shift Registers.

This study proposes a modification to Grain128 stream cipher based on a dynamic feedback approach to increase the complexity of the cipher, consequently, improving its security level. In the proposed cipher, the dynamic parameters are the feedback polynomial and the polynomial switching method. A determined set of polynomials was used to change the feedback function, and two ways were used for switching the feedback polynomials of the LFSR: regular and irregular way. The randomness of the proposed cipher, which is called Dynamic Grain Stream Cipher (DGSC), was evaluated using the National Institute of Standards and Technology (NIST) suite; and those results were encouraging. Furthermore, the proposed algorithm was compared with the original algorithm and the results indicated that the modified algorithm outperforms the original cipher in several features. A performance analysis of the proposed algorithm was carried out. Furthermore, a comparison between the proposed algorithm and the original Grain was performed. The results showed that there is no significant difference in speed between the original Grain and the modified algorithm with regular changing of taps. However, the speed decreases in the proposed algorithm with irregular changing of taps. Using regular or irregular ways to change the taps may depend on the need of specific industries.

المستخلص

يستخدم التشفير التدفقي عادة لتوفير السرية لمجموعة واسعة من التطبيقات مثل الأجهزة النقالة أو النظم المضمنة. في هذا النوع من التطبيقات يفضل استخدام التشفير التدفقي نظراً لبساطة تنفيذه وكفاءته وإنتاجيته العالية. وبالرغم من أهمية هذا النوع من التشفير فقد تم اكتشاف هجمات عملية على خوارزميات تشفير معروفة. بعد ذلك تمّ تصميم العديد من خوارزميات التشفير لمقاومة هذه الهجمات. اغلب هذه الخوارزميات لديها بنية ثابتة، وذلك يعتبر ميزة من حيث إمكانية إثبات مقاومتها للهجمات المعروفة. ومع ذلك، فإن استخدام بنية ثابتة لخوارزمية التشفير يتيح المجال لهجمات جديدة محتملة. هناك بعض المحاولات لإضافة خواص ديناميكية لبنية هذه الخوارزميات لغرض تحسين مستوى أمنها. الفكرة من وراء ذلك هي، عندما تكون بنية خوارزميات التشفير غير معروفة للمهاجمين، تكون أكثر قدرة على مقاومة الهجمات. وهذا النوع من الخوارزميات لم يناقش على نطاق واسع بين الباحثين. وعلاوة على ذلك، فإن معظم الأبحاث الحالية المتعلقة بخوارزميات التشفير التدفقي ركزت على استخدام البُنى الديناميكية في تبديل متعددة الحدود في مسجلات إزاحة التغذية العكسية الخطية.

في هذه الدراسة أجري تعديل على خوارزمية التشفير التدفقي (Grain128) استناداً على منهج التغذية العكسية الديناميكية لزيادة تعقيد الخوارزمية وبالتالي تحسين مستوى الأمان. وهذه المعاملات الديناميكية في الخوارزمية المقترحة هي متعددة الحدود وطريقة تبديل متعددة الحدود. حيث تم استخدام مجموعة محددة مسبقاً من الدوال متعددة الحدود، أما بالنسبة لطريقة تبديل الدوال فقد تم استخدام طريقتين: منتظمة وغير منتظمة. وتم تقييم عشوائية الخوارزمية المقترحة والتي سميت تشفير قرين التدفقي الديناميكي (DGSC) باستخدام اختبار نيست (NIST) وكانت النتائج مشجعة. علاوة على ذلك، تمت مقارنة الخوارزمية المقترحة مع الخوارزمية الأصلية وأظهرت النتائج أن الخوارزمية المعدلة تفوق أداء الخوارزمية الأصلية في عدة نواحي. كذلك تم إجراء تحليل أداء للخوارزمية المقترحة. و أظهرت نتائج المقارنة أنه لا يوجد فرق معنوي في السرعة بين الخوارزمية الأصلية والخوارزمية المعدلة مع التغيير المنتظم للبتات الداخلة في حساب معادلة التغذية العكسية (Taps). ومع ذلك، فإن السرعة تنخفض في الخوارزمية المقترحة مع التغيير غير المنتظم لهذه البتات. استخدام طرق منتظمة أو غير منتظمة لتغيير البتات الداخلة في حساب معادلة التغذية العكسية يعتمد على الحاجة حسب الصناعات (التطبيقات) المحددة.

This thesis is dedicated to my beloved parents

ACKNOWLEDGEMENTS

First, Alhamdulillah and thanks to almighty Allah for guiding and giving me strength and patience to finish this thesis. Then, I would like to thank the people who helped me in my study and supported me**.**

A special thanks to my family. Words cannot express how grateful I am to my mother, father, sisters and my husband for all the sacrifices that they have made for me. I was sustained in this study due to their excessive prayers.

I would like to express my sincere thanks and gratitude to my sister Fardous for her moral and scientific support and her continuous encouragement for me, this accomplishment would not have been possible without her.

I would like to express my sincere appreciation to my supervisor, Dr. Ibrahim Almerhag for his support, guidance, advice, and feedback throughout the course of my master study.

At the end, I think it is impossible to thank all those who deserve to be thanked, but I would like to say many thanks to everyone who has helped through advices, comments, and complements or even by a smile. Thank you all from the bottom of my heart.

Fayrouz Aljadi
Tripoli, 28/11/2017

TABLE OF CONTENT

List of TABLEs

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	**Meaning**
IV	Initial Value / Initialization Vector
K	The secret key
LFSR	Linear Feedback Shift Register
NFSR	Nonlinear Feedback Shift Register
DLFSR	Dynamic Linear Feedback Shift Register
ECRYPT	The European Network of Excellence for Cryptology
eSTREAM	The ECRYPT Stream Cipher Project
NIST	National Institute of Standards and Technology Statistical Test Suite

CHAPTER ONE: INTRODUCTION

1.1 INTRODUCTION

In recent years, the Increasing use of the Internet and the growing exchange of digital information have led to the necessity of reinforcing security. Cryptography is one of the most commonly used techniques for maintaining security. It produces the methods for building most of the modern security protocols used to transmit information [1].

Cryptography is the art and science of keeping messages secure by hiding their meanings from unauthorized users [2]. It is one of two main branches of the term cryptology; the other branch is cryptanalysis, which is the science of recovering information without knowledge of the key. The key is secret information used to configure a cryptosystem for encryption and decryption. A cipher or a cryptosystem is used to encrypt data. The original data is known as plaintext, and the result of encryption is called ciphertext. And by decrypting the ciphertext, the original plaintext could be restored [3].

In cryptography, there are two basic techniques used for encrypting information: symmetric encryption (also called secret key encryption) and asymmetric encryption (also called public key encryption). In asymmetric encryption the encryption key and the decryption key are different. Since different keys are used, It is possible to make the encryption key public [2]. In symmetric encryption, the same key is used to encrypt and to decrypt data. This secret key is known to both ends before the transmission starts and it must be securely kept. In this research, the focuses on one type of the symmetric key cryptography known as stream ciphers.

1.2 STREAM CIPHERS

Symmetric algorithms can be divided into two categories: stream algorithms or stream ciphers that operate on the plaintext a single bit (or sometimes byte) at a time, and block algorithms or block ciphers that operate the plaintext in groups of bits. The groups of bits are called blocks [2].

Stream ciphers are very popular due to their many attractive features: they are generally fast, can typically be efficiently implemented in hardware, have no (or limited) error propagation, and are particularly suitable for use in environments where no buffering is available and/or plaintext elements need to be processed individually. These are particularly important features in the telecommunications sector, and stream ciphers are ubiquitous in the field [4].

Stream ciphers encrypt bits individually. This is achieved by adding a bit from a key stream to a plaintext bit to get a ciphertext bit. It can be transformed back into its original form using the same key stream [5]. A key stream is a pseudo-random sequence; it can be generated using a Linear Feedback Shift Register (LFSR). LFSRs are simple, fast, and easy to implement for both software and hardware. They are capable of generating pseudo-random sequences with the same uniform statistical distribution of 0's and 1's in a truly random sequence. However, they are not cryptographically secure, because the construction of an LFSR of length n-bits can be easily deduced by observing 2n consecutive bits of its sequence using Berlekamp-Massey algorithm. Due to its inherent linearity, LFSR-based stream ciphers are vulnerable to several forms of attacks, such as fast algebraic attack and correlation attack [1, 2].

1.3 LINEAR FEEDBACK SHIFT REGISTERS

A feedback shift register consists of two parts: a shift register and a feedback function (see Fig. 1.1). The shift register is a sequence of bits. Its length is determined in bits; if it is n bits long, it is called n-bit shift register. All of the bits in the shift register are shifted one bit to the right each time a bit is needed. The new left-most bit is computed as a function of the other bits in the register. The feedback function is normally the XOR of selected bits in the register; the list of these bits is called a tap sequence. The output of the shift register is one bit, usually the least significant bit [2].

LFSRs are commonly used as part of key stream generators in stream ciphers. Certain criteria are considered for the parts of keystream generators. These criteria include

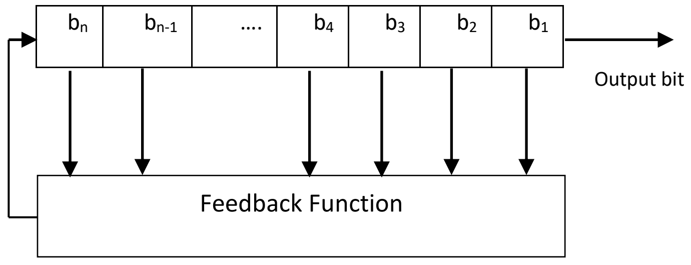


Figure 1.1: The general constructions of a linear feedback shift register.

period, linear complexity, and statistical measures of the keystreams.

- ***Period***

The period of a shift register is the length of the output sequence before it starts repeating. If the feedback polynomial of the n-bit LFSR is primitive and its initial state is at a non-zero state, then the output sequence generated by this LFSR has the maximum period of 2^n -1. This sequence is called the maximum-length sequence, or m-sequence. The m-sequences possess excellent randomness properties [1].

- ***Linear complexity***

One essential metric used to evaluate LFSR-based generators is linear complexity, or linear span. This is described as the length (n) of the shortest LFSR that can imitate the generator output. Linear complexity is very important, because a simple algorithm, called the Berlekamp-Massey algorithm, can generate this LFSR after investigating only 2n bits of the keystream. Once this LFSR is generated, the stream cipher is broken. It is worth noting that a big linear complexity does not always indicate a secure generator. However, a small linear complexity does indicate an insecure one [2].

- ***Statistical measures***

Suitable metrics are required to examine the degree of randomness for binary sequences generated by random number generators. A number of statistical tests exist to determine the statistical behavior of the sequence. These tests usually check for random distribution, distribution of ones and zeroes in a sequence, linear dependence among fixed length substrings, the level of compression that can be carried out on tested sequence, and whether a sequence is complex enough to be considered random. Three well-known tests are used for this purpose, which are: the Federal Information Processing Standard tests (FIPS), Diehard suite, and National Institute of Standards and Technology Statistical Test Suite (NIST) [6].

1.4 INTRODUCING NONLINEARITY

The output sequences of LFSR have a linear structure. Therefore, the immediate output of LFSR is unsuitable to be used as a keystream. In order to use LFSRs in the design of keystream generators, their linearity must be destroyed. To achieve that, different methods have been introduced [1], which include:

1.4.1 Combination Generators

In this generator, the output of several LFSRs is combined by a Boolean function *f* to produce the keystream. To generate a secure and random keystream, the Boolean function has to satisfy certain criteria. Figure 1.2 illustrates the general construction of this generator.

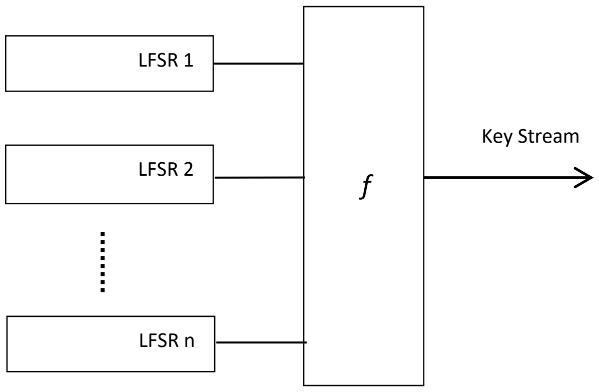


Figure 1.2: The Combination generator

1.4.2 Filter Generators

Filter generators only use a single LFSR. A Boolean function generates the keystream by filtering the contents of the LFSR. Figure 1.3 illustrates the general construction of this generator.

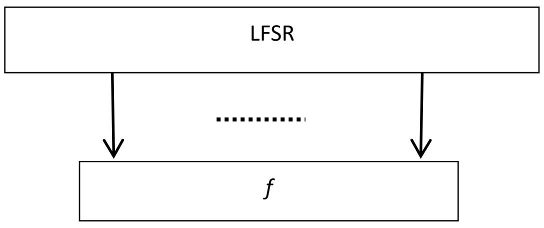


Figure 1.3: The Filter generator

1.4.3 Clock-Controlled Generators

A Clock-controlled generator has at least one LFSR, which is clocked in an irregular manner by some other part of the cipher.

In addition to the aforementioned methods, there is a method [7] based on dynamic polynomial switching in the Linear Feedback Shift Registers.

1.5 DYNAMIC LINEAR FEEDBACK SHIFT REGISTERS

The Dynamic Linear Feedback shift register (DLFSR) is a LFSR where the feedback taps are changed in run time [7]. As shown in Fig. 1.4, the conceptual design of a DLFSR is constructed of a main LFSR and an additional unit that controls the moment of time when the feedback taps are modified. The purpose of this design is to produce longer sequences with higher linear complexity than those produced by the LFSR. For doing that, the control unit modifies several feedback parameters. Therefore, the main DLFSR component is the algorithm of switching the polynomial [8].

The dynamic feedback control mechanism converts the deterministic linear recurrence

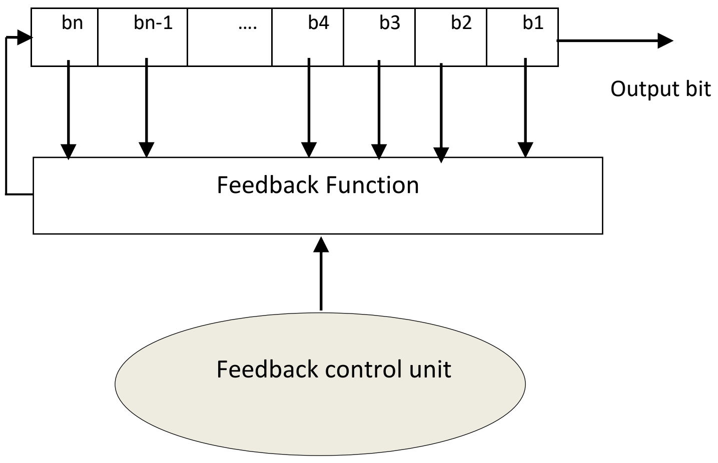


Figure 1.4: The general construction of a DLFSR.

of some registers into a probabilistic recurrence. This effectively protects against several attacks. The attacker has to guess the inputs to the dynamic feedback control unit first to perform an attack. This guessing is very difficult due to the irregular modification. Therefore, irregular modification of the feedback function improves the security of the stream cipher [9, 10].

1.6 PROBLEM STATEMENT

An improvement in the security of stream cipher is achieved by introducing dynamic polynomial switching in the Linear Feedback Shift Registers.

Several researches declared that this addition enhances the stream cipher's immunity to cryptanalysts. However, good inviolability and statistical properties of the Dynamic LFSR generator can be achieved when the parameters of the switching algorithm are correctly chosen.

1.7 RESEARCH QUESTIONS

- What are the recent approaches for applying the dynamic Linear Feedback Shift Register in stream cipher designs?
- Why using the dynamic Linear Feedback Shift Register in stream cipher enhances the security level of these ciphers?
- How could we find a suitable adjustment for the parameters of switching algorithm of the dynamic Linear Feedback Shift Register?

1.8 RESEARCH OBJECTIVES

- To investigate the recent approaches for applying the dynamic Linear Feedback Shift Register in stream cipher designs.
- To analyze the investigated approaches signs in order to study the relationship between the security level and using dynamic polynomial switching.
- To find a suitable adjustment for the parameters of switching algorithm of the dynamic Linear Feedback Shift Register.

1.9 RESEARCH SIGNIFICANCE

This research aims to provide comprehensive information about recent approaches for applying dynamic polynomial switching in the Linear Feedback Shift Registers. Besides that, it provides a study of the relationship between the dynamic polynomial switching and the security level of the stream ciphers. That will assist in the enhancement process of stream ciphers.

1.10 RESEARCH SCOPE

In this research, the main focus is on one type of the symmetric key algorithms known as stream ciphers.

1.11 RESEARCH METHODOLOGY

- Reviewing and analysing several approaches that use the dynamic polynomial switching in the Linear Feedback Shift Registers in their designs.
- Modifying Grain stream cipher in order to apply the idea of dynamic polynomial switching to it.
- Evaluating modified Grain stream cipher using NIST test suit.

- Adjusting parameters of the modified Grain stream cipher until getting satisfying results.
- Comparing the performance of modified Grain stream cipher with the original one.

1.12 TERMINOLOGY

The following terminology and notation will be used throughout this thesis:

- **Ciphertext:** the result of an encryption algorithm on a plaintext. For most stream ciphers, it is the combination of a plaintext and a keystream using the bitwise operation “XOR”.
- **Decryption algorithm:** the inverse of the encryption algorithm. It is the procedure used to convert a ciphertext to a plaintext using a secret key and (optionally) an initialization vector.
- **Encryption algorithm**: a mathematical procedure for performing encryption on a plaintext to convert it to a ciphertext using a secret key and (optionally) an initialization vector.
- **Initialization vector (IV)**: or frame number, publicly known information which is used with the secret key (master key) to generate the session key that is used in turn to generate the keystream. The IV serves as a randomizer and should take a new value for every encryption session. The stream cipher will produce different sequences of key stream material for each IV.
- **Keystream:** the sequence of random or pseudorandom bits that are generated by a keystream generator using a secret key and (optionally) an IV.
- **Output function**: the function that is used to generate the keystream.
- **Plaintext:** also called clear text, is the massage to be encrypted, it can be a stream of bits, a text file, a bitmap, a stream of digitized voice, a digital video image, etc. [2]. Plaintext is the input to an encryption process, and the output of a decryption process.
- **Polynomial degree:** is the length of the shift register [2].
- **Primitive polynomials:** a polynomial is primitive (irreducible) if it cannot be expressed as the product of two other polynomials [2].
- **Randomness:** the randomness concept is defined as an independent sequence of numbers which have a specific distribution and probability [11].

- **Secret key**: the piece of information or parameter that is used to encrypt and decrypt messages in a symmetric cryptosystem. It is combined with a known IV to produce a session key. It is assumed that this input is known only to the sender and receiver. All the security of these algorithms depends primarily on the key and none is based on the details of the algorithm; this means that the algorithm must be published and analysed to ensure its perfection [2]. Another important issue, usually referred to as “key distribution problem”, since the system requires a different secure channel for distribution of the key between sender and receiver.
- **Seed**: the initial value of the Linear Feedback Shift Register (LFSR).

1.13 THESIS ORGANIZATION

The organization of this thesis will be as follows:

Chapter 1: This chapter has provided an introduction and background of the problem under investigation, it has outlined the problem statement, the research objectives, the research questions, and has stated the significance of the study.

Chapter 2: This chapter is a literature review of recent approaches for applying the dynamic Linear Feedback Shift Register in stream cipher designs.

Chapter 3: This chapter will represent the proposed algorithm for enhancement and applying the idea of dynamic polynomial switching on it.

Chapter 4: This chapter includes the performance analysis and the security analysis of the modified stream cipher and comparing it with the original one.

Chapter 5: This chapter will contain the conclusions and suggestions for future research.

1.14 SUMMARY

This chapter has presented and discussed the background of the study. It defined the linear feedback shift registers and explains why they are not cryptographically secure. In addition, the dynamic linear feedback shift register and its featureswere discussed. Additionally, the statement of the problem was discussed, as this study set to introduce a dynamic stream cipher that meets the standards of efficiency in terms of security, implementation, and speed. This chapter also presents the research questions, hypotheses and objectives. Finally, the research methodology of this study is mentioned.

CHAPTER TWO: LITERATURE REVIEW

2.1 INTRODUCTION

This chapter provides an overview of previous studies related to this research, focusing mainly on several currently used stream cipher algorithms.That use a dynamic polynomial switching in the Linear Feedback Shift Registers of their designs.

2.2 CONSTRUCTION OF DLFSR

In 2002, Mita et al [12] presented a pseudorandom sequence generator based on a DLFSR whose feedback taps are updated based on ***m*** bits decoder driven by an ***m*** order LFSR as shown in Fig. 2.1, Obviously when increasing ***m*** the period of the overall system increases, but the complexity of the circuit also increases.

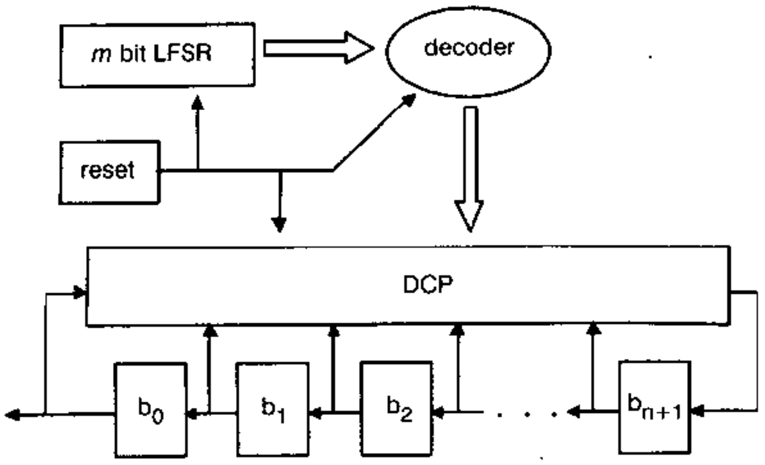


Figure 2.1: Block diagram of proposed DLFSR.

A simple example is presented, where the length of the shift register was equal to 16 bits (n=16) use a fifth-order LFSR (m=5) to implement the selector blocks. And to reduce the number of XOR gates inside the DCP (Dynamic Characteristic Polynomials) block, only five terms including x^{I6} and x^{o} were chosen among the overall characteristic polynomials. To implement the DCP block a predefined set of primitive polynomials which represent the largest set of 16-order characteristic polynomials having four equal terms was used.

The properties of the output sequence are studied and compared with the output of a conventional LFSR of similar size by using the Federal Information Processing Standard (FIPS) tests published by the National Institute of Standards and Technology

(NIST). The results of the analysis indicate a big improvement in terms of security level for the introduced stream cipher.

In [13], a stream cipher called Mickey was introduced by Babbage and Dodd which stands for Mutual Irregular Clocking keystream generator. As showing in fig 2.2 it consists of two LFSRs of the same length (80 bit), one linear (R) and one non-linear (S), each of which is irregularly clocked under control of the other. Mickey takes two input parameters, an 80-bit secret key K and an Initialization Variable IV, anywhere between 0 and 80 bits in length. The cipher specification states that each key can be used with up to 2^{40} different IVs of the same length, and that 2^{40} keystream bits can be generated from each key/IV pair.

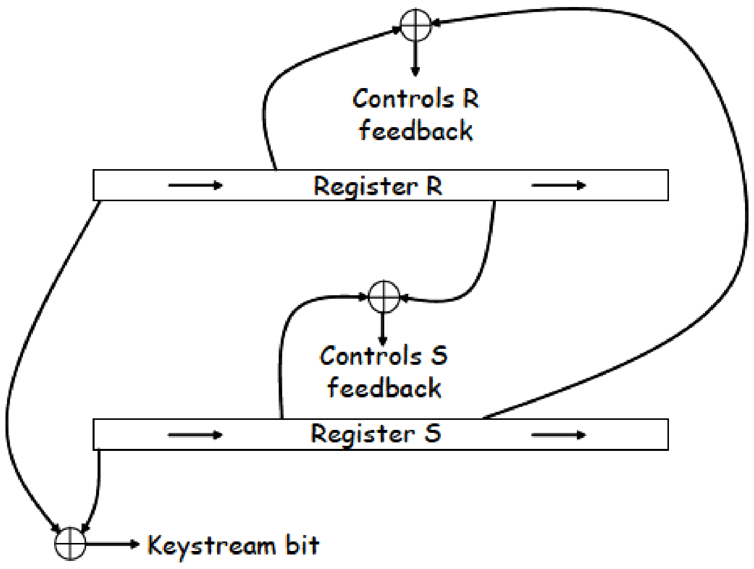


Figure 2.2: the Structure of MICKEY.

MICKEY uses a very simple output function ($s_0 \oplus r_0$) to compute keystream bits from the register states. Ciphertext is produced from plaintext by bitwise XOR with keystream bits, as in most stream ciphers.

The clock control bits chosen for each register to be derived from both registers, in such a way that knowledge of either register state is not sufficient to tell the attacker how either register will subsequently be clocked. This helps to guard against “guess and determine” or “divide and conquer” attacks.

In 2006, a pseudo-random bit generator was proposed based on dynamically changing the primitive polynomial of a LFSR to convert it to DLFSR, The architecture of the dynamic linear feedback shift register (DLFSR) stream cipher is shown in Fig. 2.3 where the block called dynamic characteristic polynomials (DCP) includes the logical circuit that dynamically switches among several feedback networks. The primitive polynomial of LFSR controlled by decoder a circuit and a counter that divides the operation time of each polynomial. A specific set of taps are used to change the feedback polynomials.

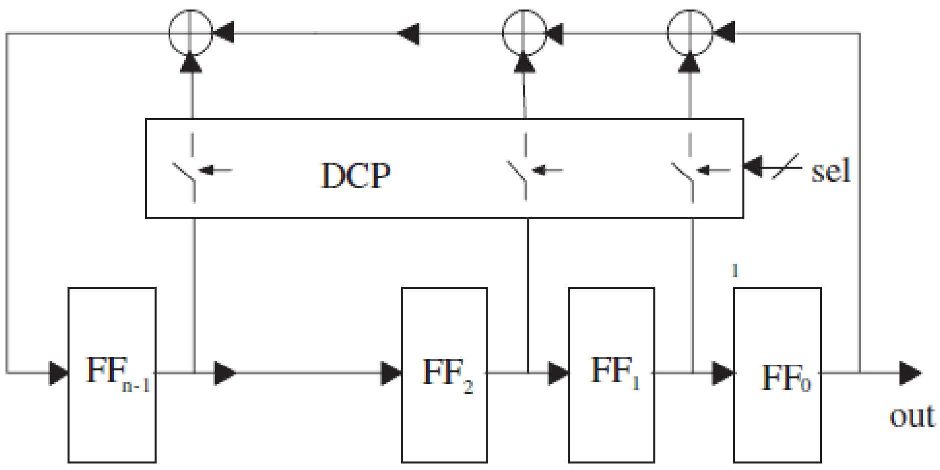


Figure 2.3: Block diagram of a generic DLFSR cipher.

The statistical properties of the proposed and the classical LFSRs are tested using ‘Statistical Random Number Generator Tests’ of FIPS. The results show that both generators have similar randomness and statistical properties. In addition, a methodology, based on a multi perception neural network is used. The results show that the proposed generator has an excellent inviolability property (the attribute of being secured against violation) [11].

In the same year, Horant and Guinee [14] introduced a stream cipher construction that is based on the A5/1 cipher which employs clock-control with majority voting. The proposed generator structure consists of five LFSRs that are connected to a Dynamic Feedback Polynomial (DFP) switching block. The DFP block contains the logical switching circuit required for implementing different feedback polynomial networks. Each register is configured with any one of the set of five feedback polynomials at any given time. Each LFSR has a clocking tap that controls its clock and polynomial switching time. At each step the five clocking taps are put to a logic majority vote and three, four or five registers are simultaneously clocked. When a LFSR is unclocked, its taps are changed. Each register has a set of five feedback polynomials that are used to change its taps. The resultant keystream sequence is given by the exclusive or (XOR) combination of the final stage of each LFSR.

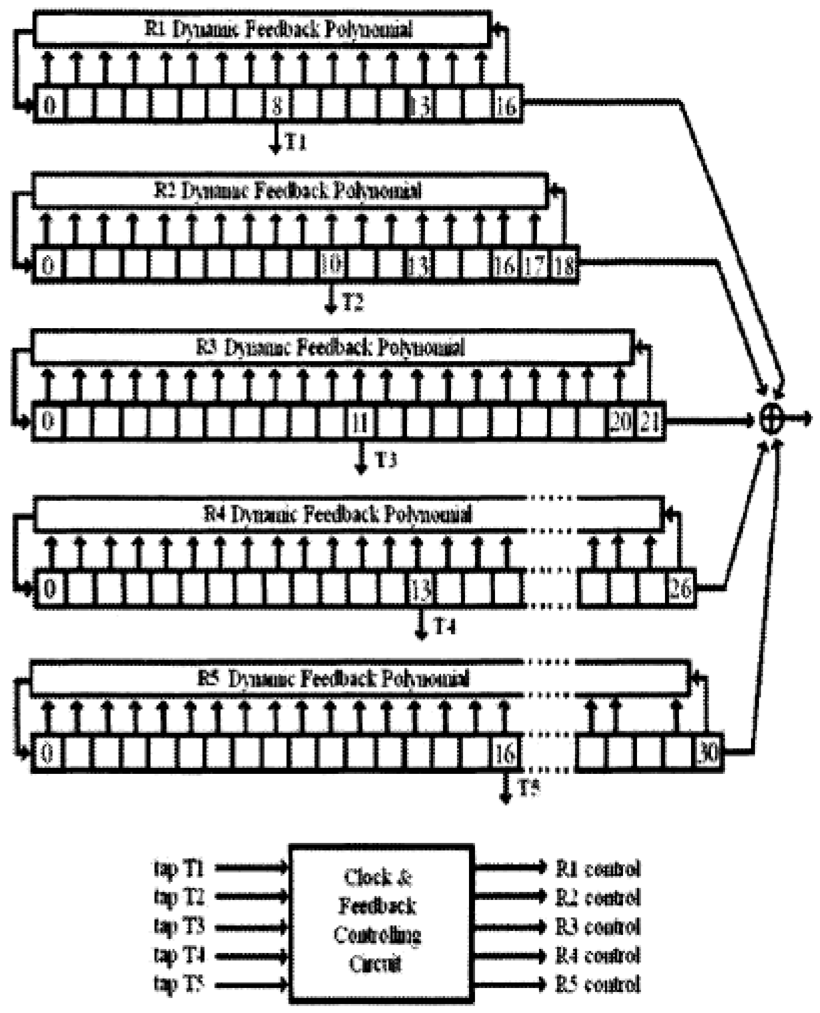

R1 Dynamic Feedback Polynomial
0
8
13
16
T1
R2 Dynamic Feedback Polynomial
0
10
13
16
17
18
T2
R3 Dynamic Feedback Polynomial
0
11
20
21
T3
R4 Dynamic Feedback Polynomial
0
13
26
T4
R5 Dynamic Feedback Polynomial
0
16
30
T5
tap T1
tap T2
tap T3
tap T4
tap T5
Clock & Feedback Controlling Circuit
R1 control
R2 control
R3 control
R4 control
R5 control

Figure 2.4: Generator Construction.

The results indicate that the proposed generator has excellent statistical properties via both the NIST and Diehard test suites.

In 2007, Kiyomoto et al [9] introduced the stream cipher K2v2.0 which is word oriented stream cipher using feedback control. It relies on two feedback shift registers (FSRs) FSR-A and FSR-B, a non-linear function and a dynamic feedback controller. FSR-B is a dynamic feedback shift registers. The feedback polynomial of the main LFSR is controlled by two bits of the secondary LFSR state. Therefore, four polynomials are selected to change the taps.

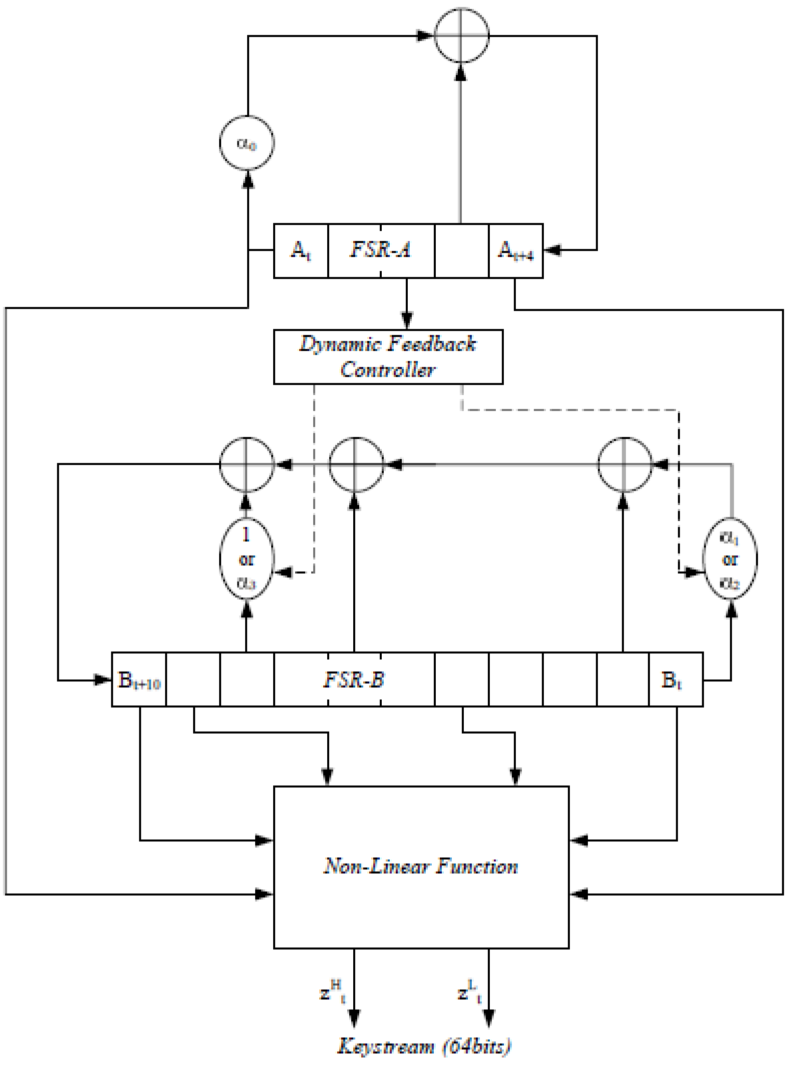


Figure 2.5: K v2.0 stream cipher.

The NIST test suite is used to evaluate the statistical properties of the generated sequence and the results confirmed that these properties are good.

In 2007, Hell et al. [15] introduced Grain stream cipher. The first version used an 80-bit key and a 64-bit initialization vector; but analysis during the early stages of the eSTREAM effort compromised its security [16]. After that, the Grain v1 was presented. It described two stream ciphers that supported 80-bit keys with 64-bit initialization vector, and 128-bit keys with 80-bit initialization vector. Due to the cryptanalysis of the 128-bit version of Grain v1, a new cipher called Grain-128a is presented [17].

Grain stream cipher is static in nature, i.e. it does not have any dynamic properties. So, it has been targeted by this study for modification and improvement. The modification is based on a dynamic feedback shift register. This approach will be explained in more details in Chapter Three.

In 2008, a DLFSR construction, which had an algorithm to generate irreducible polynomials, was introduced by Molina-Rueda et.al.[18]. A number of irreducible polynomials are generated in the initialization stage by Blum Blum Shub generator [19]. Then, the generated polynomials are scrambled in a pseudorandom way in order to increase the unpredictability, so even if the current tap is broken the attacker will not know which will be the next tap. After this, a 127 bit LFSR is used, with an initialization vector based on the key provided by the user. When the $2^{127}-1$ bits of the output sequence are depleted the connection polynomial is reset with another one of those generated in the initialization stage, and a different sequence is obtained. Each time the polynomials are depleted, they will be chosen again but in a different order, this will increase the period and is still statistically secure.

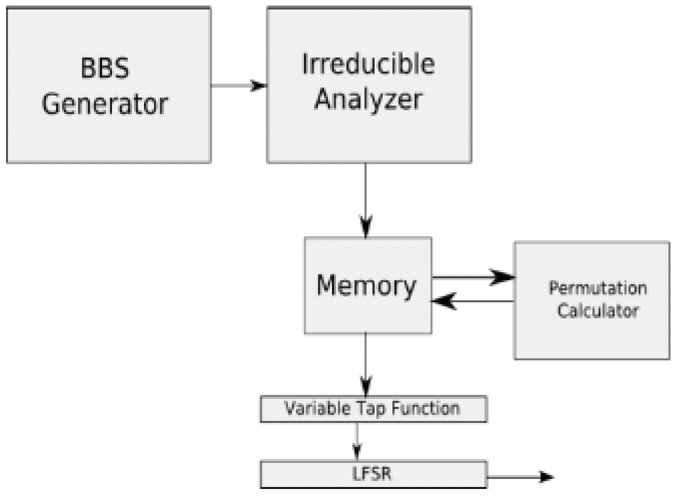


Figure 2.6: Simplified Architecture Diagram.

This LFSR was created with the objective of using it as a replacement of the ordinary LFSR in secure generators, increasing the global security of a stream cipher. The proposed generator is implemented in software as a test of the viability. The average speed of this generator after the setup phase is 100 bit/sec.

The Rakaposhi stream cipher was presented in 2009 by Cid et al [4]. Its main component is the bit-oriented dynamic linear feedback shift register. It consists of a 128-bit Non-Linear Feedback Shift Register and a 192-bit Linear Feedback Shift Register, denoted as registers A and B, respectively. The cipher uses two bits from the state of the NLFSR to select, and dynamically modify four (linear) feedback function of the LFSR. The cipher keystream is produced by combining the output of both registers with the output of a non-linear Boolean function. This function takes six bits from the state of register B and two bits from the state of register A as input.

In the initialization process, the secret key and IV are loaded into the registers and mixed. The secret key and IV are loaded into the NLFSR and DLFSR, respectively. The cipher then clocks 448 times with the output of the filter function being fed back into the cipher state. The cipher must be re-initialized (potentially by only modifying the IV) after at most 264 cycles.

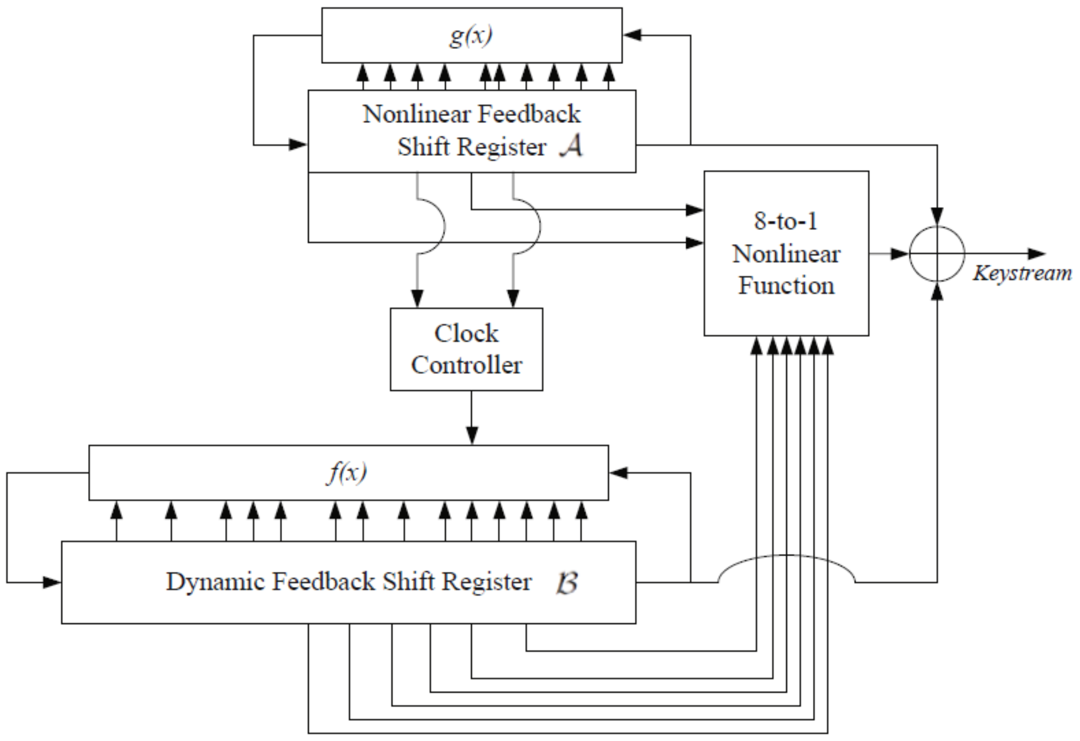

g(x)
Nonlinear Feedback Shift Register A
8-to-1 Nonlinear Function
Keystream
Clock Controller
f(x)
Dynamic Feedback Shift Register B

Figure 2.7: Rakaposhi Stream Cipher.

The NIST test Suite is used to evaluate this cipher. The results indicate that the statistical properties of the Rakaposhi output sequence are good.

In 2010, a new version of stream cipher modified SNOW 2.0 based on dynamic feedback was introduced [10]. Dynamic feedback is determined using dynamic number generator function. The linearity in this version is converted into non linearity. In other words LFSR property is converted into NLFSR property. The design of modified SNOW 2.0 is divided into three steps; Main Operation of dynamic feedback based modified version of SNOW 2.0, Updating of Linear Feedback Shift Register (LFSR), and Updating of Finite State Machine (FSM).

The analysis and experimental results show that the suggested technique has more resistance against Guess and Determine attacks and more affective for the encryption of plaintext. It is more secure and reliable for a secure communication as compared to static feedback based modified SNOW 2.0.

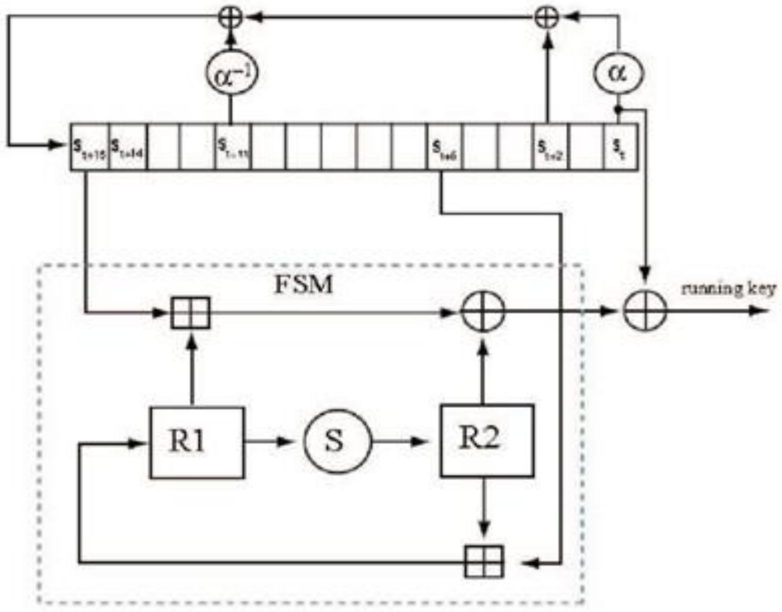


Figure 2.8: Model of modified SNOW 2.0.

In [20], Bajaj suggested using DLFSR instead of LFSR in the A5/1 stream cipher . Two modifications have been proposed in this cipher; one is in feedback tapping unit and other modification is in the clocking rule. The feedback unit is modified in two different ways; the first is Shuffling LFSRs by extending all LFSRs to 23 bit to shuffle these LFSR. This shuffling is done periodically but the state of LFSR will change randomly. The second way is Feedback polynomial unit*;* in this proposal four different feedback polynomials for each LFSR are selected. The feedback polynomials are chosen such that there would be only one tap that is different in all tap configurations of an LFSR. A LFSR changes its feedback polynomial after generating bits more than twice its length. The other modification is clocking unit, the clock controlled unit of conventional A5/1 works on majority rule.

The proposed stream cipher passed all the NIST’s random tests. And the proposed scheme is robust to the cryptographic attacks compared to the conventional A5/1 stream cipher.

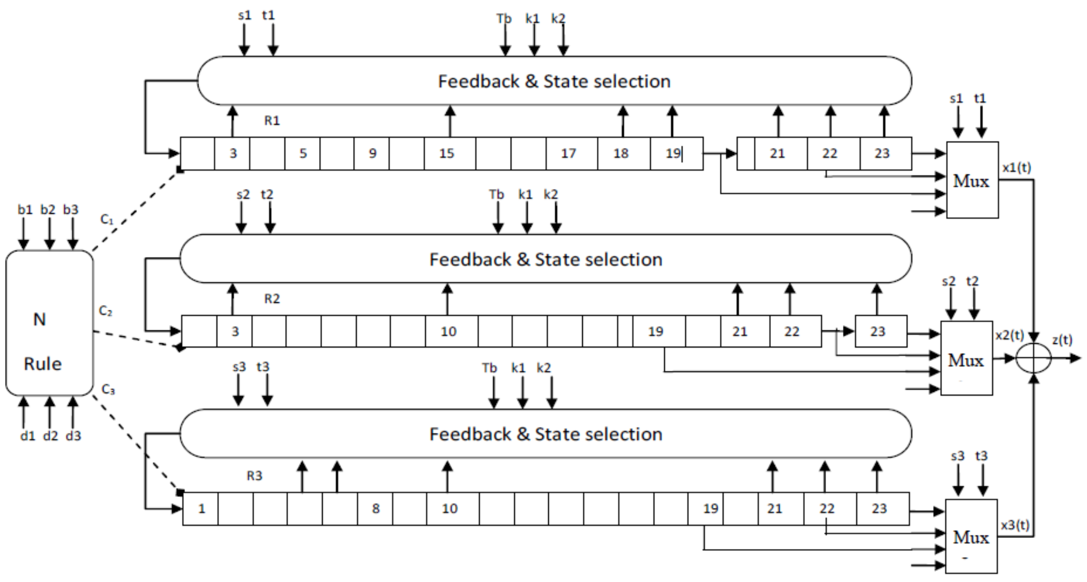


Figure 2.9: Proposed modified A5/1.

The Heraclitus stream cipher was proposed in 2011 by Colbert et al [21]. The authors used a key dependent structure, whose variable parameters are the number of registers, the length of registers, and the feedback polynomials of the registers, these variable parameters are selected using an index. A fixed set of irreducible polynomials (one for each register) and the hash function SHA512 [22] are used to generate the feedback polynomials. The variable parameters are changed every 2^{64} frames of a session. The clocking mechanism used majority clocking based on a fixed bit positions in the LFSRs.

The design of Heraclitus exploited the choices available in cipher design, such as the choice of irreducible polynomials or the choice a function which satisfies certain conditions. They also represent an increase in strength of the ciphers because: (a) each cipher generated is designed to satisfy particular criteria to ensure the strength of the cipher and (b) each cipher is expected to be only used once — therefore it is infeasible to determine any weakness, even if the cipher is known.

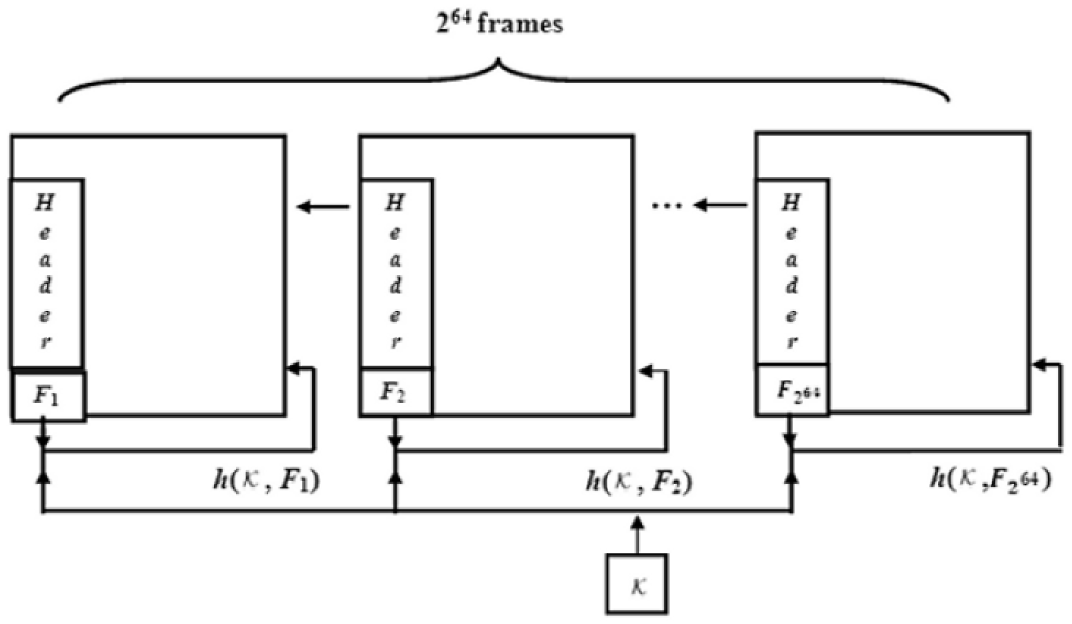


Figure 2.10: Each session key K is used for 2^{64}frames.

In 2013, the J3Gen generator was presented by Melià-Seguí et al [23]. Its construction is based on a DLFSR, with a number of feedback polynomials selected by a round robin scheme. The feedback polynomial is changed after a given number of DLFSR cycles. The Polynomial Selector Module shifts its position towards a new configuration. The number of shifts, i.e., the corresponding selection of each primitive polynomial at a certain LFSR cycle, is determined by a true random bit obtained from a physical source of randomness provided by the TRNG module. The feedback polynomials are implemented as a wheel, which rotates depending on this bit value. If the truly random bit is a logical 0, the wheel rotates one position, that is, it selects the next feedback polynomial. Instead, if the truly random bit is a logical 1, then the wheel rotates two positions, that is, the polynomial selector jumps one feedback polynomial and selects the next one.

The Decoding Logic is responsible for managing the internal PRNG clock of J3Gen. It activates and deactivates the PRNG modules for its proper performance. The authors introduced a hardware implementation of J3Gen, and evaluated it regarding nonlinearity of the design, different design parameters, and defining the key-equivalence security.

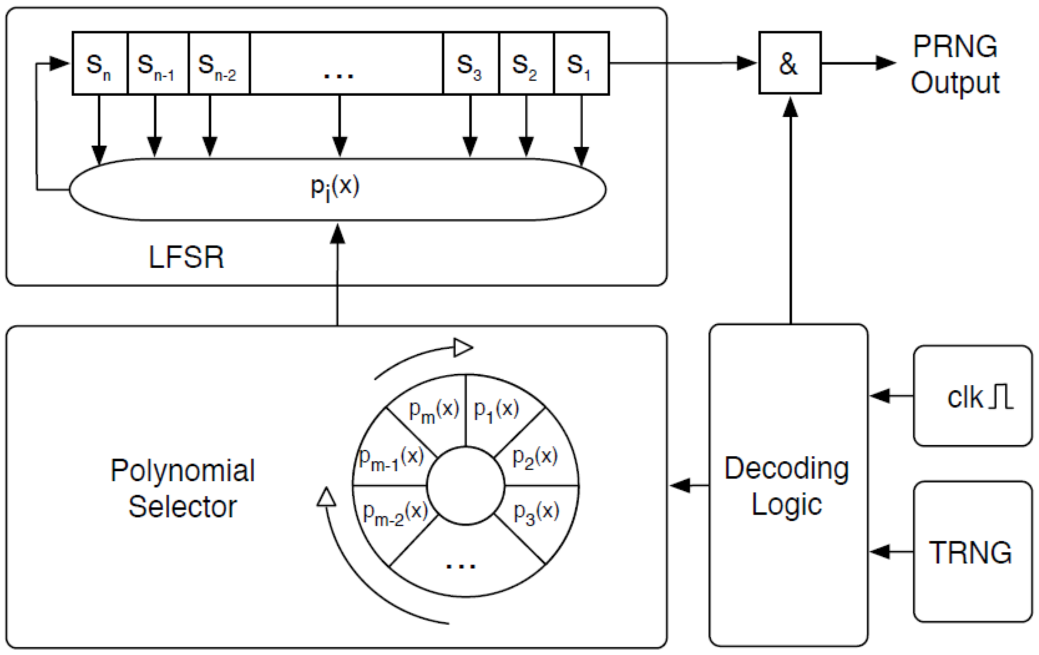


Figure 2.11: Block diagram of J3Gen.

In [8] the authors presented the LFSR and DLFSR structures and their differences. LFSR generator is built from the shift register and the feedback loop, most often implemented as a multiple input XOR gate. The LFSR generator is a synchronous circuit and it requires clocking signal to work properly. The feedback loop is described by a polynomial, the general structure of the LFSR generator is shown in figure 2.12. The feedback loop is described by the polynomial, which exponents of the variable x are numbers of each shift register bits that are connected to the feedback loop. The LFSR structure does not change while operating.

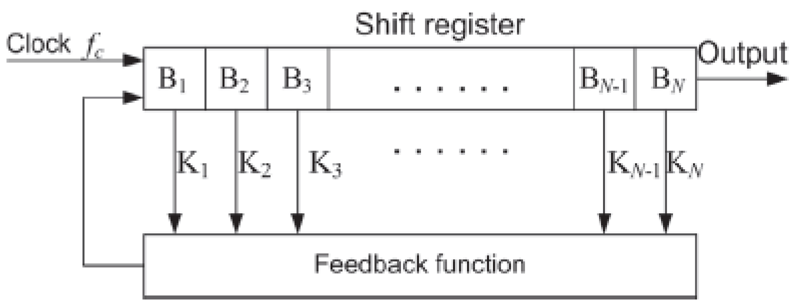
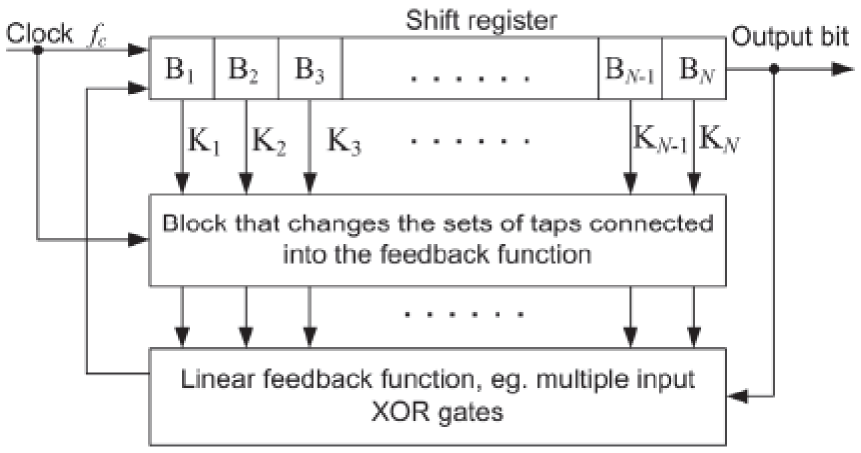


Figure 2.12: the general LFSR generator.

Another type of the pseudo random signal generator, based on a shift register, is the Dynamic Linear Feedback Shift Register generator. The basic structure of this generator is shown in figure 2.13.This generator is made from three basic functional

Elements; the N bit shift register, the block that changes the feedback polynomial, and the feedback loop function made from multiple XOR gate. Algorithm of the polynomial change depends on its designer. This algorithm requires having some parameters to be strictly determined; moment of time, related to the clock signal that determines the feedback polynomial change, and set of the feedback polynomials that will be used by the feedback block.



Clock f_c
Shift register
Output bit
B_1 B_2 B_3 . . . . . . B_{N-1} B_N
K_1 K_2 K_3 . . . . K_{N-1} K_N
Block that changes the sets of taps connected into the feedback function
Linear feedback function, eg. multiple input XOR gates

Figure 2.13: The DLFSR generator structure.

The authors used experimental methods to choose the parameters of DLFSR switching algorithm. They compared between the Diehard statistical tests results of the LFSR and DLFSR generators. This comparison confirmed that DLFSR pseudo random sequences have better statistical properties than the conventional ones and it passed all DIEHARD tests.

In 2014, Peninado et al [7] presented a DLFSR model that consisted of two LFSR and a counter. The main LFSR is a regular LFSR of n cells with primitive feedback polynomials which are applied in a round robin scheme. The secondary LFSR is a clock-controlled primitive LFSR of $m = \log_2 n$ cells to control the feedback of the main LFSR. The state of this LFSR sets the initial value of a counter. When the counter downs to zero, then the secondary LFSR generates a new bit and the new state resets again the counter to a different value. The counter synchronizes the secondary LFSR with the feedback polynomial of the main LFSR. Each time the secondary LFSR generates a new bit, the feedback polynomial of the main LFSR is updated, in such a way that the number of consecutive bits generated by the corresponding polynomial is the decimal value of the state of the secondary LFSR.

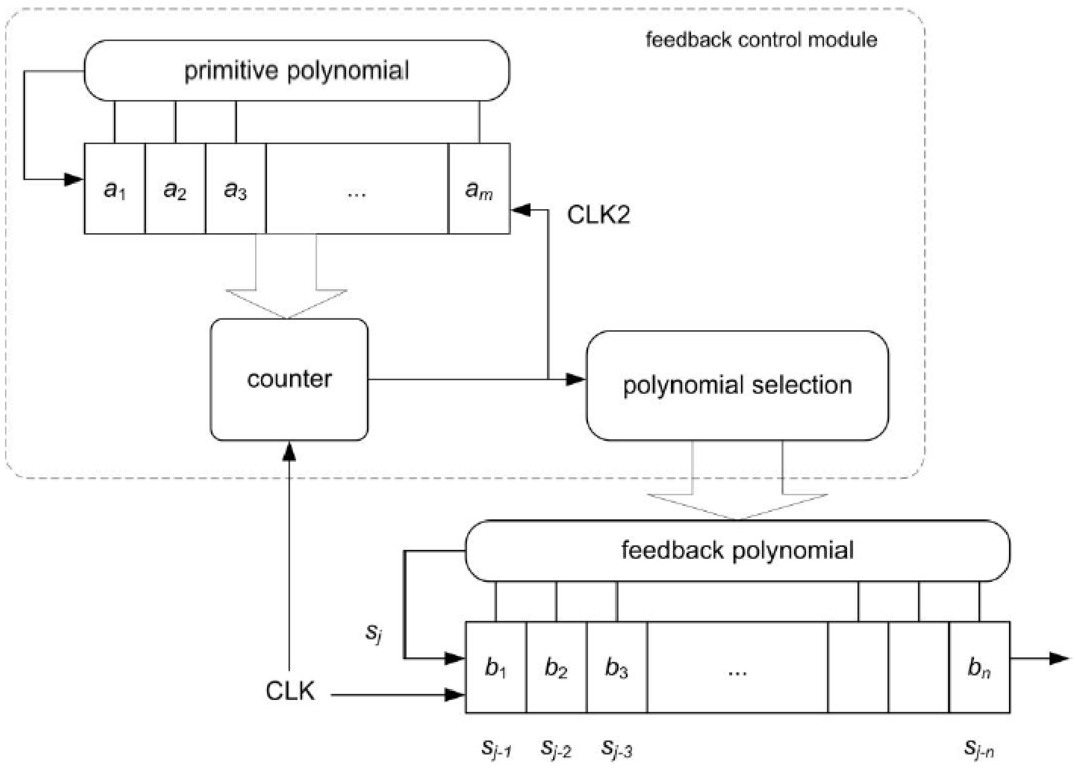


Figure 2.14: Diagram of the generic DLFSR-with-counter module.

A comparative analysis of the proposed DLFSR design with other DLFSR designs is performed. The results indicate that the introduced design is better than others in certain aspects.

2.3 A BRIEF SUMMARY OF THE PREVIOUS STUDIES ON DLFSRS

This section shows a brief summary of the researcher's contributions on dynamic linear feedback shift registers as well as the names of authors and year, the title of papers, and the main idea of each work which are presented below in Table (2.1).

Table 2.1: A brief description of the previous papers of DLFSR.

#	Authors	Title	Main idea
1	R. Mita, G. Palumbo, S. Pennisi, and M. Poli (2002)	Pseudorandom bit generator based on dynamic linear feedback topology	A pseudorandom sequence generator based on a DLFSR whose feedback taps are updated based on the state of a secondary LFSR.
2	Babbage and M. Dodd (2005)	The stream cipher MICKEY (version 1)	A stream cipher consists of two LFSRs of the same length connected in such a manner that both registers mutually control their corresponding feedbacks.
3	R. Mita, G. Palumbo, and M. Poli (2006)	Pseudo-random sequence generators with improved inviolability performance	A generator was proposed based on DLFSR that is controlled by a decoder circuit and a counter that divides the operation time of each polynomial
4	D. Horan and R. Guinee (2006)	A novel keystream generator using pseudo random binary sequences for cryptographic applications	A stream cipher whose DLFSRs' construction is based on the A5/1 cipher majority voting function.
5	S. Kiyomoto, T. Tanaka, and K. Sakurai (2007)	K2: A Stream Cipher Algorithm using Dynamic Feedback Control	A Stream Cipher that relies on two LFSRs and a non-linear function. The feedback polynomial of the main LFSR is controlled by two bits of the secondary LFSR state.
6	A. Molina-Rueda, F. Uceda-Ponga, and C. F. Uribe (2008)	Extended period LFSR using variable TAP function	A DLFSR construction, which had an algorithm to generate irreducible polynomials.
7	C. Cid, S. Kiyomoto, and J. Kurihara (2009)	The rakaposhi stream cipher	A DLFSR construction that consists of a LFSR whose feedback polynomial is chosen among four different options controlled by two bits of Non-LFSR state.

#	Authors	Title	Main idea
8	S. Khan, A. Khan, S. Khayal, T. Naz, S. Bashir, and F. Khan (2010)	Dynamic feedback based modified SNOW 2.0	A new version of stream cipher modified SNOW 2.0 based on dynamic feedback was introduced.
9	N. Bajaj (2011)	Enhancement of A5/1: Using variable feedback polynomials of LFSR	Using DLFSR instead of LFSR in the A5/1 stream cipher. Each DLFSR has four different feedback polynomials, and it changes its feedback polynomial after it generates twice more bits than its length.
10	B. Colbert, A. H. Dekker, and L. M. Batten (2011)	Heraclitus: A LFSR-based stream cipher with key dependent structure	A stream cipher with key dependent structure, whose variable parameters are the number of registers, the length of registers, and the feedback polynomials of the registers.
11	J. Melià-Seguí, J. Garcia-Alfaro, and J. Herrera-Joancomartí (2013)	J3Gen: A PRNG for low-cost passive RFID	A generator construction that is based on a DLFSR, with a number of feedback polynomials selected by a round robin scheme.
12	R. Stepien and J. Walczak (2013)	Comparative analysis of pseudo random signals of the LFSR and DLFSR generators	The authors compared between the statistical tests results of the LFSR and DLFSR generators. This comparison confirmed that DLFSR pseudo random sequences have better statistical properties than the conventional ones.
13	A. Peinado, J. Munilla, and A. Fúster-Sabater (2014)	Improving the Period and Linear Span of the Sequences Generated by DLFSRs	A DLFSR model that consists of two LFSRs and a counter. The main LFSR polynomial is controlled by a counter, whose value depends on the internal state of the secondary LFSR.

2.4 SUMMARY

This chapter represented a helpful insight to the previous work that have been done till now on DLFSR constructions and their polynomial switching algorithms to provide a clear understanding to the researchers who want to search in this area and participate in enhancing the security issues of communication.

CHAPTER THREE: THE PROPOSED ALGORITHM

3.1 INTRODUCTION

The eSTREAM project is a multi-year effort to create a portfolio of promising new stream ciphers, Funded by the ECRYPT Network of Excellence. The project started in 2004 with a call for proposals from industry and academia. These proposals were designed to satisfy either a software-oriented or hardware-oriented profile. In total 34 submissions are generated during the original call for proposal [17].

The project was divided into three phases, the third phase completed in April 2008 with the announcement of the candidates that had been selected for the final eSTREAM portfolio. The algorithms in Profile 1 (software-oriented algorithms) are suitable for software applications with high throughput requirements. The length of their keys is either 128 or 256 bits, and the initialization vector (IV) is required to be 64 or 128 bits. This profile contains the following ciphers: HC-128, Rabbit, Salsa20/12, and SOSEMANUK.

The algorithms in Profile 2 (hardware-oriented algorithms) are supposed to be efficient with regards to the physical resources required when implemented in hardware. These algorithms were required to support 80-bit keys, and can also support 128-bit keys. The initialization vector can be 32 or 64 bits. This profile contains the following ciphers: Grain, Mickey, and Trivium.

Grain stream cipher was chosen to test the idea of dynamic polynomial switching, In order to examine the relationship between the dynamic polynomial switching and the security level. In this chapter, the original Grain algorithm and the proposed algorithm are explained in details.

3.2 GRAIN STREAM CIPHER

In 2007, Hell et al. [15] introduced Grain stream cipher. The first version used an 80-bit key and a 64-bit initialization vector; but analysis during the early stages of the eSTREAM effort compromised its security [16]. After that, the Grain v1 was presented. It described two stream ciphers that supported 80-bit keys with 64-bit initialization vector, and 128-bit keys with 80-bit initialization vector. The main building blocks of this cipher are two shift registers as shown in Figure (3.1), one with linear feedback

(LFSR) and the second with non-linear feedback (NFSR). The contents of the two shift registers represent the state of the cipher. From this state, 5 variables are chosen as input to a Boolean function that is selected to be balanced, correlation immune of the first order and have algebraic degree of 3. Due to the cryptanalysis of the 128-bit version of Grain v1, a new cipher called Grain-128a is presented [17].

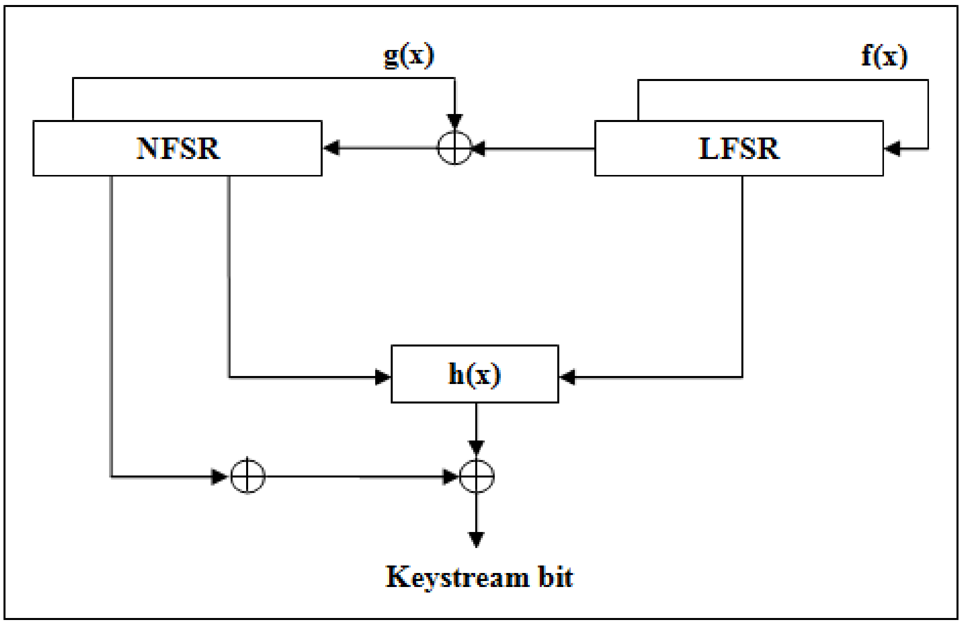


Figure 3.1: The Structure of Grain [15].

Grain has high speed, low gate count and low power consumption [24]. However, there is some cost incurred during initialization and the impact of this will be determined by the intended application and the likely size of the messages being encrypted [17]. Another weakness of Grain-v1 is that certain 18 bits of the internal state can be efficiently recovered based on the corresponding keystream segment and the assumption on certain 133 bits of the considered internal state [25]. Regarding security, there are some recently introduced attacks on Grain.

In 2008, De Canniere et al. [26] observed the existence of a sliding property in the initialization algorithm of the Grain family, and show that it can be used to decrease by half the cost of exhaustive key search. In 2012, Banik et al. [27] introduced a differential fault attack on the Grain 128a authenticated encryption scheme using certain properties of the Boolean function h used in the cipher design. In 2013, Ding and Guan proposed a related key chosen IV attack on Grain-128a based on some observations.

Their result showed that their attack is much better than an exhaustive key search in the related key setting [28].

3.2.1 GRAIN-128'S OUTPUT AND STATE UPDATE FUNCTIONS

Grain and Grain-128 ciphers follow the same design principle [29]. They include three essential building blocks, which are an NFSR, an LFSR and an output function. The contents of the two shift registers represent the state of the cipher and their sizes are |K| bits each, where K is the key. In the following, the content of the NFSR is denoted by$B_t = b_t + b_{t+1}, \dots, b_{t+|K|-1}$ and the content of the LFSR is denoted by $S_t = s_t + s_{t+1}, \dots, s_{t+|K|-1}$. The output function, denoted by $H(B_t, S_t)$ contains two parts. A nonlinear Boolean function h(x) and a set of linear terms combined with h(x). The output of $H(B_t, S_t)$ is the keystream bit z_t. Figure 3.2 shows a general overview of the design.

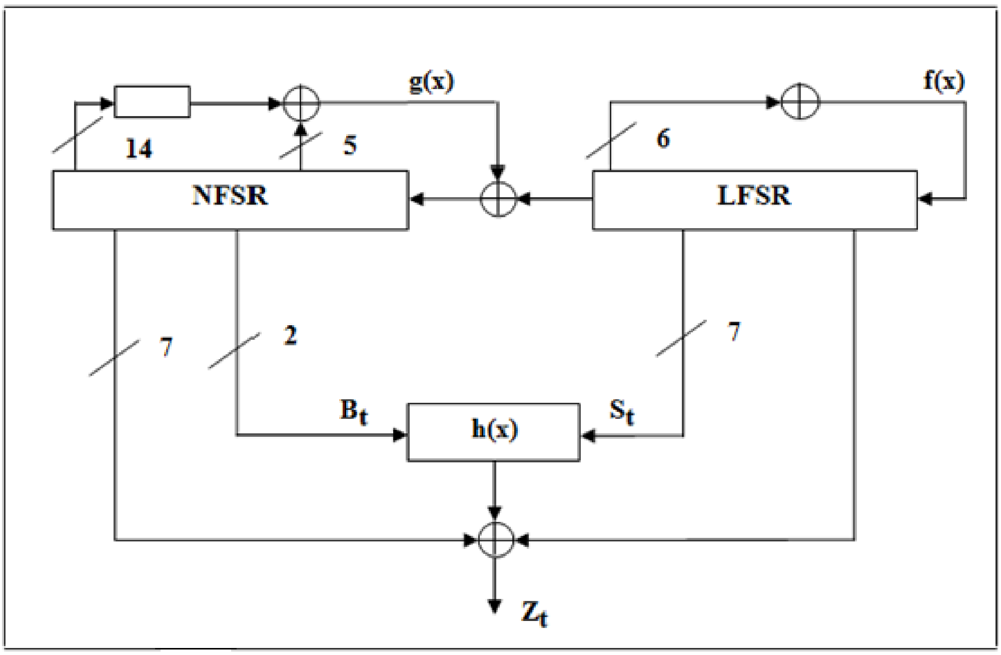


Figure 3.2: Grain128 stream cipher [10].

Grain-128 supports a key size of |K| = 128 bits. The size of the IV is stated to be |IV| = 96 bits. The feedback polynomial of the LFSR, f(x) is a primitive polynomial of degree 128. It is defined as in Equ (3.1)

$f(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$. .... (3.1)

The corresponding update function of the LFSR defined as in Equ (3.2)

$S_{t+128} = s_t \oplus s_{t+7} \oplus s_{t+38} \oplus s_{t+70} \oplus s_{t+81} \oplus s_{t+96}$ ………………………… (3.2)

g(x) is the nonlinear feedback polynomial of the NFSR. It is defined as in Equ (3.3)

$g(x) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} + x^{80}x^{88} + x^{110}x^{111} + x^{115}x^{117}$ ….…………………………………….... (3.3)

In the corresponding update function of the NFSR below in Equ (3.4) observe that the bit S_t which is masked with the input to the NFSR is included, while ignored in the feedback polynomial.

$b_{t+128} = s_t \oplus b_t \oplus b_{t+26} \oplus b_{t+56} \oplus b_{t+91} \oplus b_{t+96} \oplus b_{t+3}b_{t+67} \oplus b_{t+11}b_{t+13} \oplus b_{t+17}b_{t+18} \oplus b_{t+27}b_{t+59} \oplus b_{t+40}b_{t+48} \oplus b_{t+61}b_{t+65} \oplus b_{t+68}b_{t+84}$ …………… (3.4)

In the cipher state, 9 variables are used as input to the Boolean function, h(x). 2 inputs to h(x) are obtained from the NFSR and 7 are obtained from the LFSR. This function is of degree 3 and quite simple. It can be defined as in Equ (3.5)

$h(x) = h(x_0, x_1, \dots, x_8) = x_0x_1 \oplus x_2x_3 \oplus x_4x_5 \oplus x_6x_7 \oplus x_0x_4x_8$...................... (3.5)

where the variables x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7 and x_8 match the tap positions b_{t+12}, S_{t+8}, S_{t+13}, S_{t+20}, b_{t+95}, S_{t+42}, S_{t+60}, S_{t+79} and S_{t+95} respectively.

The output function H(Bt, St) is described as in Equ (3.6)

$z_t = H(B_t, S_t) = \bigoplus_{j \in A} b_{t+j} \oplus h(x) \oplus s_{t+93}$, ………………………………….…… (3.6)

$where\ A = \{2, 15, 36, 45, 64, 73, 89\}$.

3.2.2 GRAIN-128 INITIALIZATION PROCESS

The initialisation process is an essential process which is required to be carried out prior to keystream production gets started. Next, the keystream generators could be used to generate keystream sequences. With regard to security, the particular initialisation process should not expose any kind of information regarding the secret key and also is required to be protected to prevent, as a minimum, all the widely known common attacks.

In Grain-128's initialization process [29], the loading phase is performed by loading the 128-bit secret key into the nonlinear feedback shift register. The 96 bits of IV are loaded into 96 bits of the linear feedback shift register (LFSR). The remaining bits of the LFSR are filled by ones. In the diffusion phase, the LFSR and NFSR registers are clocked 256 times before producing any keystream, and the output bit is XORed and fed back to the input of both the LFSR and NFSR. The general concept of Grain is illustrated in Figure 3.3.

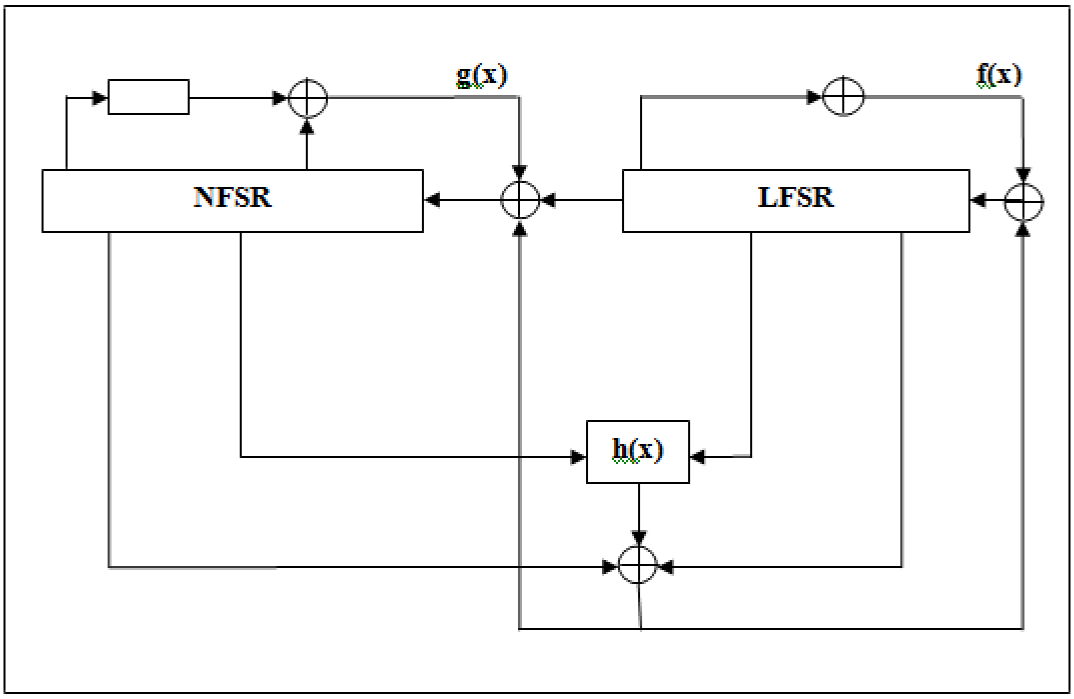


Figure 3.3: Overview of Grain128 key initialization [1]

3.3 PROPOSED ALGORITHM ARCHITECTURE

In the original algorithm the output sequences of LFSR have a linear structure. So, in order to overcome the linearity of the bits generated using LFSR, a dynamic Linear Feedback Shift Register is used instead of LFSR. The proposed algorithm is called Dynamic Grain Stream Cipher (DGSC).

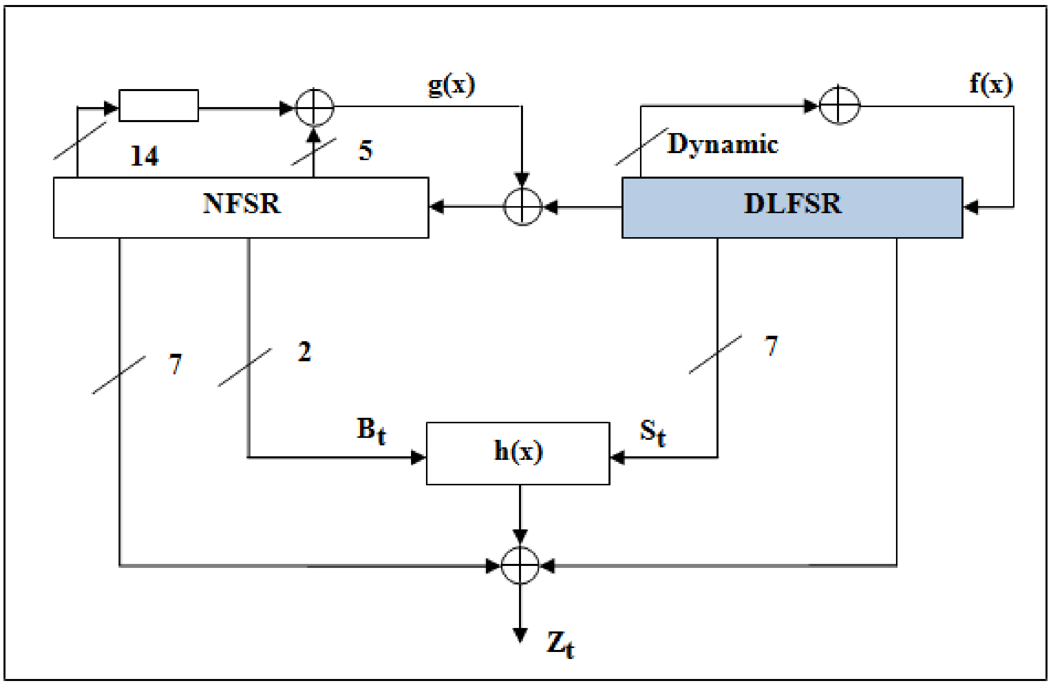


Figure 3.4: The proposed algorithm.

The conceptual design of a DLFSR is constructed of a main LFSR and an additional unit that controls the moment of time where the feedback taps are modified. The purpose of this design is to produce longer sequences than those produced by the LFSR with higher linear complexity. To do that, the control unit modifies several feedback parameters. Therefore, the main DLFSR component is the algorithm of switching the polynomial [8], and the proposed modification is centered in this part.

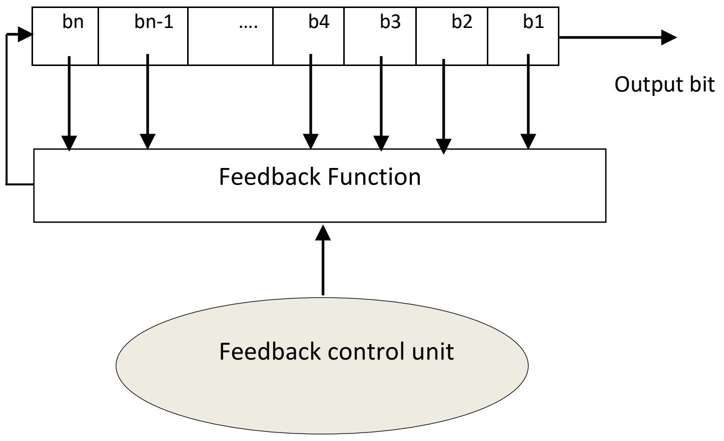


Figure 3.5: The general construction of a DLFSR.

3.3.1 THE PROPOSED POLYNOMIAL SWITCHING ALGORITHM

The main DLFSR component is the algorithm of switching the polynomial [8]. It has two parameters: the set of feedback taps which will be used to change the feedback polynomials, and the method of changing these polynomials.

3.3.1.1 Feedback polynomials

In DLFSRs, determining the set of polynomials that will be used to change the feedback function is very important. While some approaches used a predefined set of primitive polynomials, others used algorithms to generate these polynomials.
Predefined set of irreducible polynomials will be used in the proposed algorithm to switch the primitive polynomial of the LFSR. Five irreducible polynomials of degree 128 are used with the original polynomial:

$F_1(X) = X^{128} + X^9 + X^8 + X^7 + X^6 + X^5 + X^4 + X^3$…………………………………. (1)

$F_2(X) = X^{128} + X^8 + X^6 + X^5 + X^4 + X$ …………………………………………….. (2)

$F_3(X) = X^{128} + X^8 + X^6 + X^5 + X^4 + X^2$ ……………………………………………. (3)

$F_4(X) = X^{128} + X^{10} + X^9 + X^7 + X^3 + X^2$ ………………………………...……… (4)

$F_5(X) = X^{128} + X^7 + X^2 + X$ ……………………………………………………... (5)

3.3.1.2 Polynomial changing method

Another important consideration is the way of switching the feedback polynomials. There are two methods that can be used to switch the taps; regular and irregular methods. If taps are changed in a regular manner, then polynomials are changed after a specific amount of time. But when irregular ways are used, the moment of change is not fixed and usually depends on the value of certain bits within the operated registers.

Both methods (regular and irregular) will be used for switching the DGSC feedback polynomials. In the regular way, two options will be used to change the taps based on number of output bits: changing the taps every 128 bits or changing the taps every 256 bits. In the irregular way, changing the taps will happen if the output bits of LFSR and NLFSR match for 50, 100, 128, 256 times. Later a comparison will be done between these methods.

Table 3.1 illustrates the shortcuts that are used to represent different versions of the proposed algorithm (DGSC).

Table 3.1: Abbreviation for the names of algorithms.

No	**Algorithm**	**Abbreviation**
1	Original Grain-128	Grain
2	Grain-128 with regular changing of taps (n=128)	DGSCR128
3	Grain-128with regular changing of taps(n=256)	DGSCR256
4	Grain-128 with irregular changing of taps (m=128)	DGSCI128
5	Grain-128 with irregular changing of taps (m=256)	DGSCI256
6	Grain-128with irregular changing of taps (m=50)	DGSCI50
7	Grain-128with irregular changing of taps(m=100)	DGSCI100

3.4 SUMMARY

This chapter presented the details of the original algorithm (Grain-128) and the proposed algorithm DGSC with detailed explanation of modification have made. The modification has been concentrated on replacing the LFSR with DLFSR. In addition, a new switching method for changing polynomial is introduced in order to find a suitable adjustment for the parameters of switching algorithm of the dynamic Linear Feedback Shift Register.

CHAPTER FOUR: EXPERIMENTAL RESULTS

4.1 INTRODUCTION

This chapter presents an evaluation of the proposed algorithm (DGSC). A performance and security analysis of the DGSC were conducted. Moreover, a comparison is performed between the proposed cipher and the original cipher in terms of statistical properties and performance. The NIST test suite is used to examine the statistical properties of the compared algorithms, and C code is used to measure their performance.

4.2 NIST TEST SUITE

The NIST Test Suite is a statistical package that includes 15 tests designed to evaluate the randomness of binary sequences generated by either hardware or software-based cryptographic random or pseudorandom number generators. These tests concentrate on a number of different types of non-randomness which could exist in a sequence. One of these tests is linear complexity test. This test is an essential metric that is used to evaluate the randomness of binary sequences. Some tests are split into a number of subtests.

In the NIST tests, when a P-value for a test is determined to be equal to 1, then the sequence seems to have perfect randomness. A P-value of zero indicates that the sequence seems to be completely non-random. A significance level (a=0.01) is selected for the tests. If p-value is greater than or equal to 0.01, then the sequence seems to be random with a confidence of 99%. A P-value less than 0.01 indicates that the sequence appears to be non-random with a confidence of 99% [6]. Each test is explained in more details below:

- **Frequency (Monobit)**

This test computes the proportion of zeroes and ones for the whole sequence. The intent behind this test is to check if the number of ones and zeros in a sequence are almost the same as would be expected for a random sequence.

- **Frequency Test within a Block**

This test computes the proportion of zeroes and ones within M-bit blocks. The intent behind this test is to check if the number of ones and zeros in an M-bit block is approximately M/2, as would be estimated for a random sequence.

- **Runs Test**

This test concentrates on determining the total number of runs in the sequence, where a run is a continuous sequence of identical bits. A run of length K contains K identical bits and is bounded before and after with a bit of the opposite value. The intent behind this test is to check if the number of runs of ones and zeros of several lengths is approximately the same as would be expected for a truly random sequence.

- **Test for the Longest Run of Ones in a Block**

This test computes the longest run of ones within M-bit blocks. It checks if the length of the longest run of ones within the tested sequence is consistent with the length of the longest run of ones which would be estimated for a truly random sequence.

- **Binary Matrix Rank**

This test computes the rank of disjoint sub-matrices of the entire sequence. It checks for linear dependence among fixed length substrings of the original sequence.

- **Discrete Fourier Transform (DFT)**

This test detects periodic features. It focuses on the peak heights in the discrete Fourier transform of the sequence.

- **Non-overlapping Template Matching**

This test detects generators that generate a lot of occurrences of a specified non-periodic pattern. It focuses on the number of occurrences of pre-specified target strings.

- **Overlapping Template Matching**

This test focuses on the number of occurrences of pre-specified target strings.

- **Maurer’s “Universal Statistical”**

This test detects whether or not the sequence can be significantly compressed without losing information. Therefore, this test computes the number of bits between matching patterns. The non-random sequence is a very compressible sequence.

- **Linear Complexity**

This test checks if the sequence is complex enough to be considered random. It is based on the Berlekamp-Massey algorithm that provides a way for measuring linear complexity.

- **Serial Test**

This test checks if the number of occurrences of the 2mm-bit overlapping patterns is roughly the same as would be estimated for a random sequence. Its focus is on the frequency of all possible overlapping m-bit patterns across the whole sequence.

- **Approximate Entropy**

This test compares the frequency of overlapping blocks of two adjacent lengths (m and m+1) with estimated result for a truly random sequence. It focuses on the frequency of all possible overlapping m-bit patterns across the whole sequence.

- **Cumulative Sums**

This test concentrates on determining the maximal excursion (from zero) of the random walk described by the cumulative sum of adjusted (-1, +1) digits in the sequence.

- **Random Excursions**

This test concentrates on determining the number of cycles having exactly K visits in a cumulative sum random walk. After the (0, 1) sequence is transferred to the appropriate (-1, +1) sequence, the cumulative sum random walk is produced from partial sums.

- **Random Excursions Variant**

This test discovers deviations from the estimated number of visits to various states in the random walk. It focuses on the total number of times a particular state is visited in a cumulative sum random walk [6].

4.3 SECURITY ANALYSIS OF THE PROPOSED ALGORITHM (DGSC)

Security analysis plays an essential role in the evaluation process of new ciphers. This section reports the results of the NIST test suite of the proposed algorithm. The results are obtained for 10^8 bits generated by the DGSC. The output logs of empirical results will be stored in two files, stats and results that correspond respectively to the computational information e.g., test statistics, intermediate parameters, and P-values for each statistical test applied to a data set.

A file final Analysis Report is generated when statistical testing is complete. The results are represented via a table with p rows and q columns. The number of rows, p, corresponds to the number of statistical tests applied. The number of columns, q = 13, are distributed as follows: columns 1-10 correspond to the frequency of P-values10, column 11 is the P-value that arises via the application of a chi-square test11, column 12 is the proportion of binary sequences that passed, and the 13th column is the corresponding statistical test [6]. As shown in table 4.1 below.

Table 4.1: Output NIST Test of the DGSCR128.

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
generator is <data/DynamicTaps_Grain_regular128.txt>
------------------------------------------------------------------------------
C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
11   8  13   9  15   4   7  12  11 10   0.437274   100/100    BlockFrequency
 8  17   9   6  10   9   9  11  10 11   0.595549    99/100    CumulativeSums
15   6   9  12   8   7  10  12   8 13   0.574903   100/100    CumulativeSums
11   8   8  16   4  11  11   9  10 12   0.455937    98/100    Runs
12   8  10  13  10  10  11   9   5 12   0.851383   100/100    LongestRun
 7  12   4   7  17  10   7  14   8 14   0.085587    98/100    Rank
17   8  12  17   8   5   9   9   6  9   0.080519    97/100    FFT
12   6   8   9   9   7  10  14  15 10   0.574903    98/100    NonOverlappingTemplate
10  14  10  10   9  11  12  11   6  7   0.851383   100/100    NonOverlappingTemplate
12  11  11  12   3  10  12   7  12 10   0.574903    98/100    NonOverlappingTemplate
10  13   6  14  10   8  12   7   6 14   0.437274    98/100    OverlappingTemplate
 8   7  16   9  11  11   7   10 12  9   0.678686    98/100    Universal
 5  10   7  13  11  18  10    2  9 15   0.019188    99/100    ApproximateEntropy
 3   2   6   7   8   5   7   6   2  8   0.350485    54/54     RandomExcursions
 5   4   9   5   4   7   8   3   4  5   0.616305    54/54     RandomExcursions
 6   4   6   4   4   7   3   8   7  5   0.816537    53/54     RandomExcursions
 6  11   5  10  10  11  12  12  13 10   0.739918    99/100    Serial
10   5  20   5  10   8  10  13  11  8   0.051942    99/100    Serial
 9  14   5  11  13   8  10  10  11  9   0.759756    98/100    LinearComplexity
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
The minimum pass rate for each statistical test with the exception of the
random excursion (variant) test is approximately = 96 for a
sample size = 100 binary sequences.

The minimum pass rate for the random excursion (variant) test
is approximately = 51 for a sample size = 54 binary sequences.

For further guidelines construct a probability table using the MAPLE program
provided in the addendum section of the documentation.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

The p-values reported in tables, and the average is calculated for test results of the non-overlapping-templates, random-excursions and random-excursions variant, which are decomposable into a variety of subtests. Table 4.2 shows the results of the NIST test suite for the DGSC with regular changing of taps every 128 bits of keystream (n=128). It can be observed from the results that the DGSCR passes all the tests as the p-value for all tests are greater than 0.01.

Table 4.2: NIST Test Results of the DGSCR128.

#	Statistical test	P-value	Result
1	Frequency	0.678686	Pass
2	Block-frequency	0.437274	Pass
3	Cumulative-sums(forward)	0.595549	Pass
	Cumulative-sums(reverse)	0.574903	Pass
4	Runs	0.455937	Pass
5	Longest-runs of ones	0.851383	Pass
6	Rank	0.085587	Pass
7	DFT (Spectral)	0.080519	Pass
8	Non-overlapping-template	0.526587	Pass
9	Overlapping-templates	0.437274	Pass
10	Universal	0.678686	Pass
11	Approximate entropy	0.019188	Pass
12	Random-excursions	0.589688	Pass
13	Random-excursions variant	0.363008	Pass
14	Serial1	0.739918	Pass
	Serial2	0.051942	Pass
15	Linear complexity	0.759756	Pass

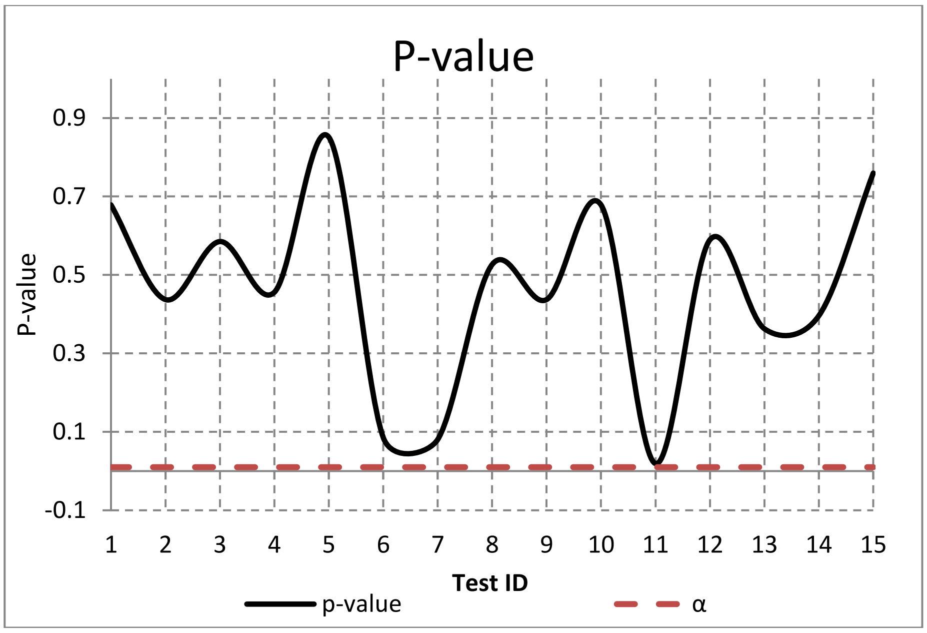


Figure 4.1: NIST Test Results of the DGSCR128.

Figure 4.1 shows the NIST test analysis of the randomness of the proposed cipher. The horizontal line represents the passing rate, which is 0.01; the value 1 represents perfect randomness. In Figure 4.1, it is worth noting that all 15 statistical tests exceeded the passing rate.

Table 4.3: NIST Test Results of the DGSCR256.

#	Statistical test	P-value	Result
1	Frequency	0.145326	Pass
2	Block-frequency	0.616305	Pass
3	Cumulative-sums(forward)	0.048716	Pass
	Cumulative-sums(reverse)	0.224821	Pass
4	Runs	0.657933	Pass
5	Longest-runs of ones	0.494392	Pass
6	Rank	0.181557	Pass
7	DFT (Spectral)	0.637119	Pass
8	Non-overlapping-template	0.454613	Pass
9	Overlapping-templates	0.075719	Pass
10	Universal	0.366918	Pass
11	Approximate entropy	0.834308	Pass
12	Random-excursions	0.4089375	Pass
13	Random-excursions variant	0.517047	Pass
14	Serial1	0.911413	Pass
	Serial2	0.383827	Pass
15	Linear complexity	0.366918	Pass

Another experiment is conducted in order to find a suitable adjustment for the number of bits after which the taps are changed. In this experiment, the taps are changed every 256 bits. Table 4.3 and figure 4.2 shows the results of the NIST test suite for the DGSC with regular changing of taps after 256 bits of keystream (n=256) are generated.

It can be observed from the results that the DGSC with regular changing of taps every 256 bits of keystream passes all the tests because the p-values for all tests are greater than 0.01.

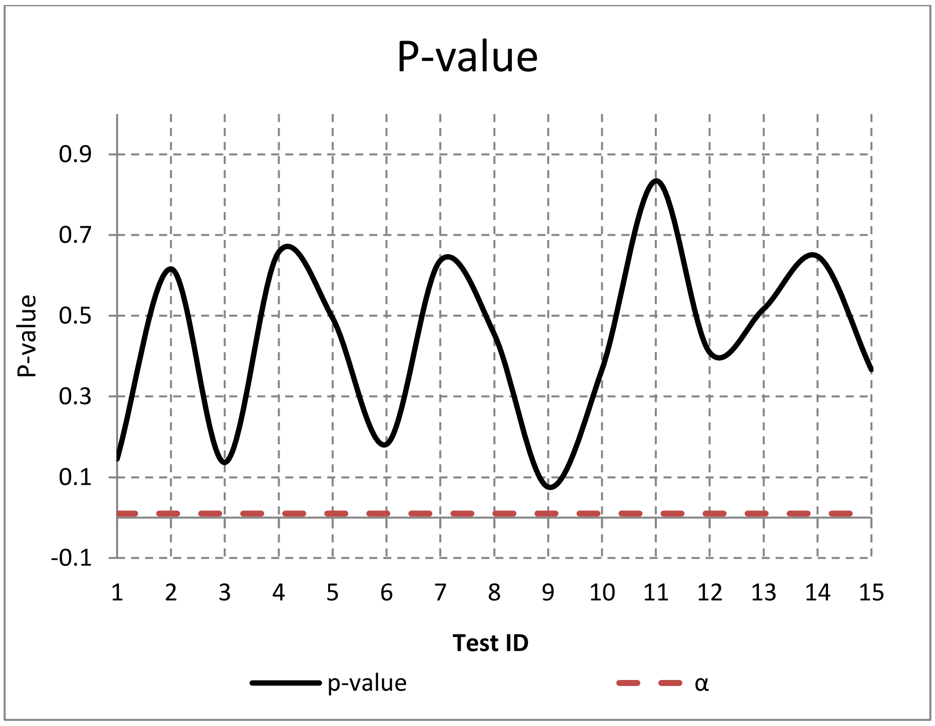


Figure 4.2: NIST Test Results of the DGSCR256.

In the second experiment, the taps are changed irregularly when there is matching between the outputs bit of LFSR and NLFSR for 50, 100, 128, and 256 times. Table 4.4 shows the results of the NIST test suite for 50 matching (m=50).

Table 4.4 and Figure 4.3 show the results of the NIST test suite for 50 matching (m=50).

Table 4.4: NIST Test Results of the DGSCI50.

#	Statistical test	P-value	Result
1	Frequency	0.798139	Pass
2	Block-frequency	0.350485	Pass
3	Cumulative-sums(forward)	0.534146	Pass
	Cumulative-sums(reverse)	0.955835	Pass
4	Runs	0.719747	Pass
5	Longest-runs of ones	0.657933	Pass
6	Rank	0.987896	Pass
7	DFT (Spectral)	0.55442	Pass
8	Non-overlapping-template	0.552449	Pass
9	Overlapping-templates	0.779188	Pass
10	Universal	0.035174	Pass
11	Approximate entropy	0.23681	Pass
12	Random-excursions	0.33767225	Pass
13	Random-excursions variant	0.365456	Pass
14	Serial1	0.637119	Pass
	Serial2	0.224821	Pass
15	Linear complexity	0.437274	Pass

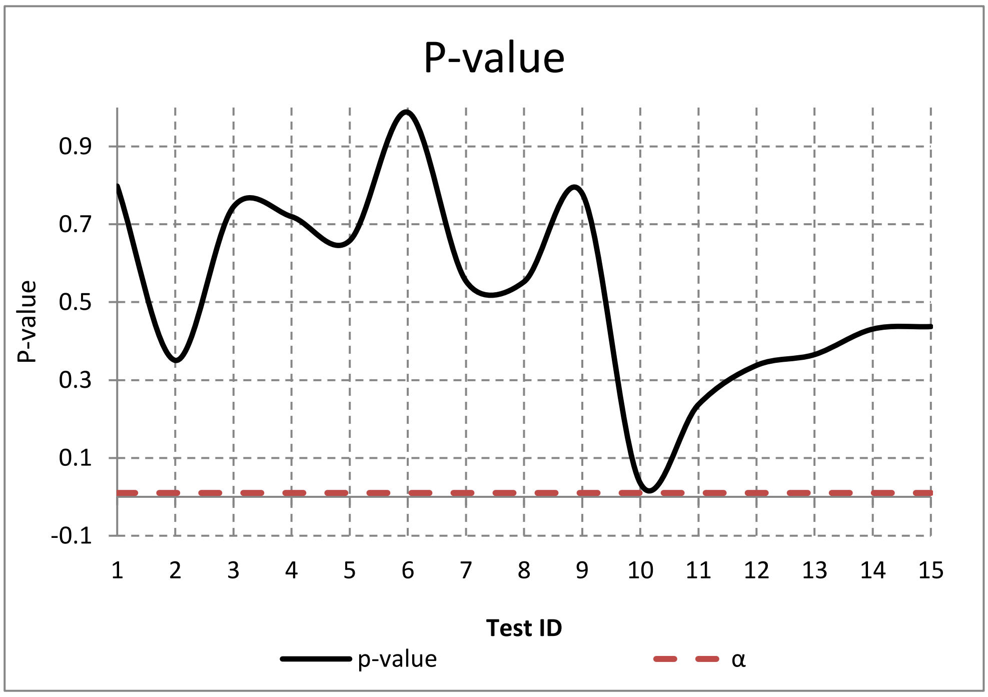


Figure 4.3: NIST Test Results of the DGSCI50.

Table 4.5 and Figure 4.4 show the results of the NIST test suite for 100 matching (m=100).

Table 4.5: NIST Test Results of DGSCI100.

#	Statistical test	P-value	Result
1	Frequency	0.455937	Pass
2	Block-frequency	0.191687	Pass
3	Cumulative-sums(forward)	0.213309	Pass
	Cumulative-sums(reverse)	0.181557	Pass
4	Runs	0.191687	Pass
5	Longest-runs of ones	0.383827	Pass
6	Rank	0.224821	Pass
7	DFT (Spectral)	0.798139	Pass
8	Non-overlapping-template	0.493655	Pass
9	Overlapping-templates	0.595549	Pass
10	Universal	0.304126	Pass
11	Approximate entropy	0.678686	Pass
12	Random-excursions	0.427572	Pass
13	Random-excursions variant	0.435763	Pass
14	Serial1	0.015598	Pass
	Serial2	0.181557	Pass
15	Linear complexity	0.202268	Pass

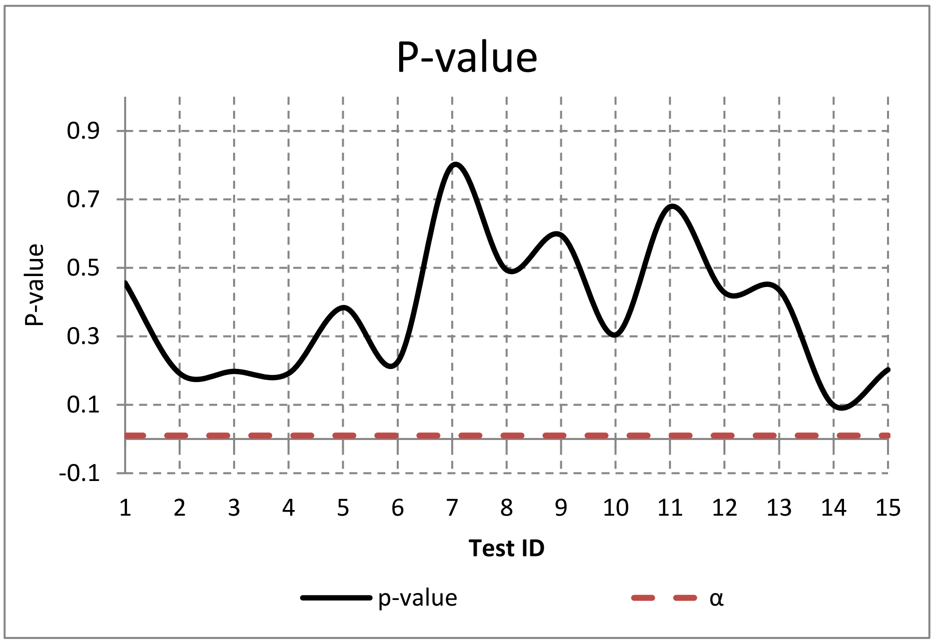


Figure 4.4: NIST Test Results of the DGSCI100.

Table 4.6 and Figure 4.5 show the results of the NIST test suite for 128 matching (m=128).

Table 4.6: NIST Test Results of the DGSCI128.

#	Statistical test	P-value	Result
1	Frequency	0.637119	Pass
2	Block-frequency	0.181557	Pass
3	Cumulative-sums(forward)	0.897763	Pass
	Cumulative-sums(reverse)	0.719747	Pass
4	Runs	0.171867	Pass
5	Longest-runs of ones	0.455937	Pass
6	Rank	0.798139	Pass
7	DFT (Spectral)	0.224821	Pass
8	Non-overlapping-template	0.488985	Pass
9	Overlapping-templates	0.897763	Pass
10	Universal	0.971699	Pass
11	Approximate entropy	0.574903	Pass
12	Random-excursions	0.293469	Pass
13	Random-excursions variant	0.322444	Pass
14	Serial1	0.911413	Pass
	Serial2	0.334538	Pass
15	Linear complexity	0.851383	Pass

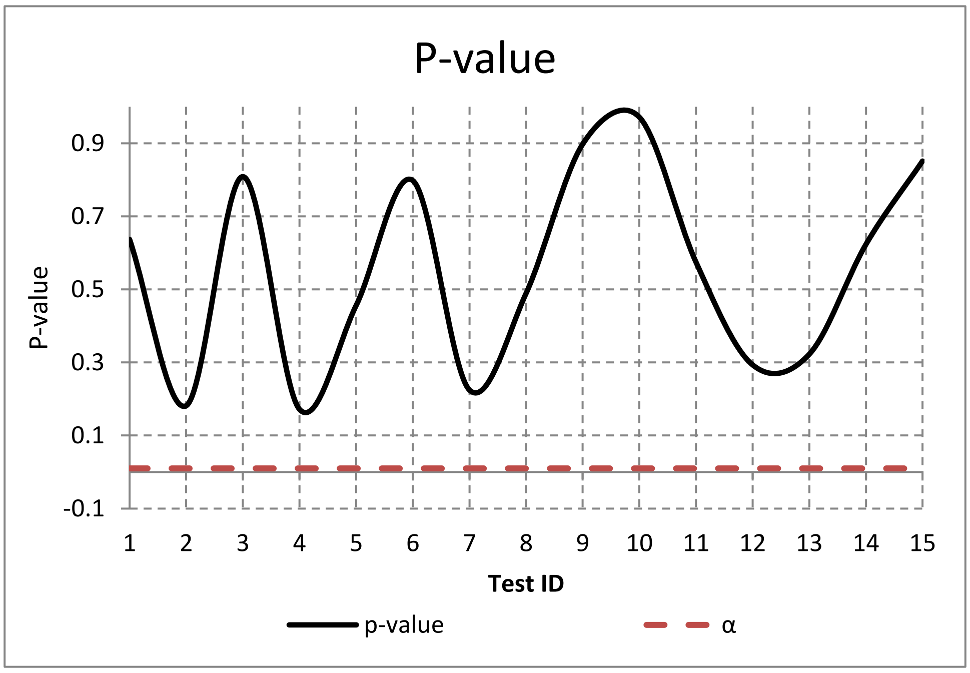


Figure 4.5: NIST Test Results of DGSCI128.

Table 4.7 and Figure 4.6 show the results of the NIST test suite for 256 matching (m=256).

Table 4.7: NIST Test Results of the DGSCI256.

#	Statistical test	P-value	Result
1	Frequency	0.213309	Pass
2	Block-frequency	0.719747	Pass
3	Cumulative-sums(forward)	0.350485	Pass
	Cumulative-sums(reverse)	0.657933	Pass
4	Runs	0.350485	Pass
5	Longest-runs of ones	0.102526	Pass
6	Rank	0.085587	Pass
7	DFT (Spectral)	0.213309	Pass
8	Non-overlapping-template	0.487295	Pass
9	Overlapping-templates	0.834308	Pass
10	Universal	0.816537	Pass
11	Approximate entropy	0.759756	Pass
12	Random-excursions	0.375971	Pass
13	Random-excursions variant	0.242827	Pass
14	Serial1	0.514124	Pass
	Serial2	0.455937	Pass
15	Linear complexity	0.678686	Pass

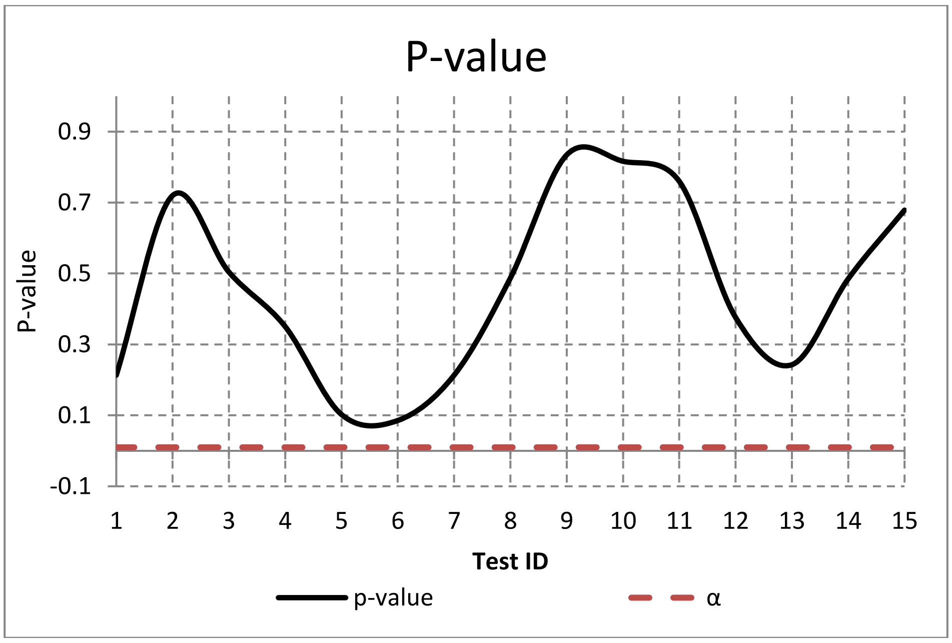


Figure 4.6: NIST Test Results of the DGSC256.

All the NIST Test Results show that the proposed algorithms with dynamic switching taps pass all the tests because the p-values for all tests are greater than 0.01.

4.4 PERFORMANCE OF THE PROPOSED ALGORITHM (DGSC)

The second important factor is the Performance. Grain and the proposed modified Grain have been coded into C language to test their performance of them. Table 4.8 shows the comparison between Grain and the modified algorithms in the perspective of performance. The under consideration algorithms were tested on the same PC and same environment. In this test, the performance is done by measuring the number of generated keystream bits versus time.

Table 4.8: Speed comparison between Grain and the DGSC.

No	**Algorithm**	**Performance in Megabyte per second(Mbps)**	**Taps changing condition**
1	Grain	0.269459	---
2	DGSCR128	0.269318	Every 128 bits.
3	DGSCR256	0.269418	Every 256 bits.
4	DGSCI128	0.266941	The output bit of the LFSR matches the output bit of the NFSR 128 times.
5	DGSCI256	0.267322	The output bit of the LFSR matches the output bit of the NFSR 256 times.
6	DGSCI50	0.264457	The output bit of the LFSR matches the output bit of the NFSR 50 times.
7	DGSCI100	0.265927	The output bit of the LFSR matches the output bit of the NFSR 100 times.

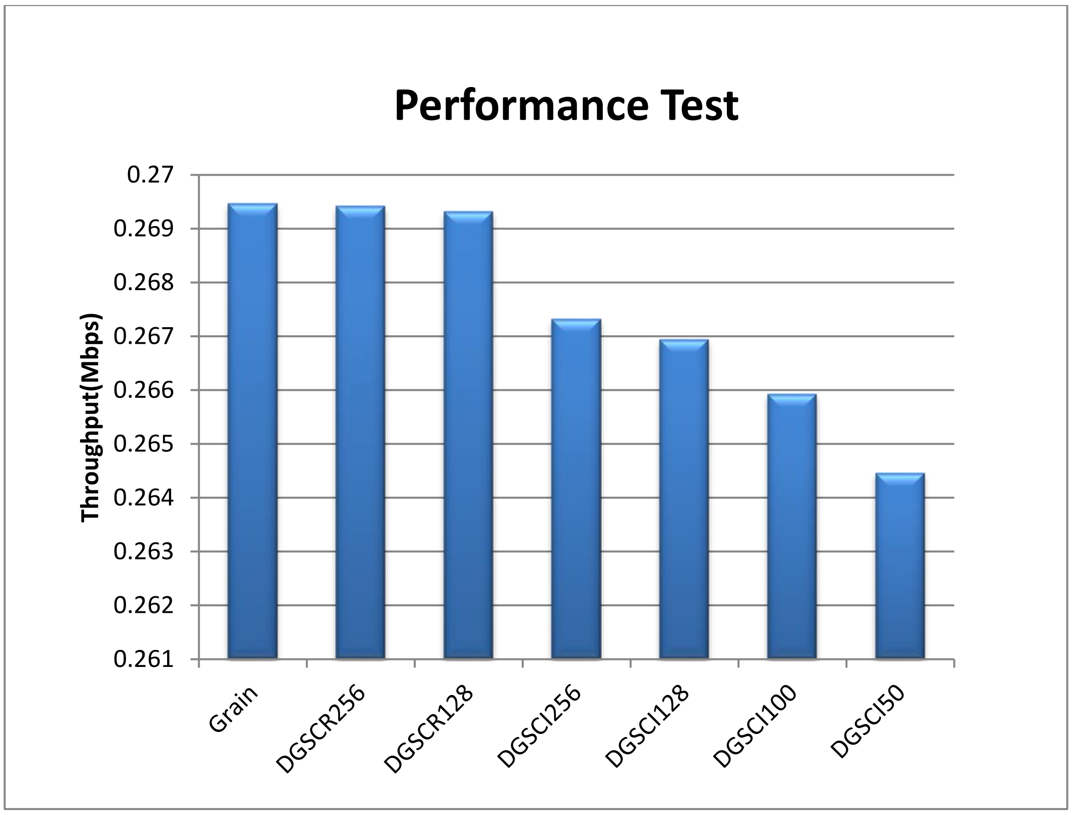


Figure 4.7: Speed comparisons between Grain and the DGSC.

From Figure 4.7, it is clear that there is no significant difference in speed between the original Grain and the modified algorithms with regular changing of taps. However, the speed decreases in the proposed algorithms with irregular changing of taps. This could be due to the comparisions that are needed to find match between the output of the LFSR and NLFSR. When the number of required maching is small, the times of changing taps increases, therefore, the speed decreases. on the other hand, when the number of required maching increases, the times of changing of taps decreases. Thus, the speed increases.

Despite the fact that using irregular ways to change the feedback bits slows down the algorithm, it is more secure than using regular ones, because these ways increase the nonlinearity of the produced sequence.

4.5 COMPARISONS OF THE NIST TEST RESULTS

In the following tables, the best result in each row is highlighted. In these tables, results that are close to 1 indicate better statistical properties. It is clear that the use of regular and irregular tap switching increases the security level of Grain algorithm.

Table 4.9 shows the NIST results for DGSCR128 and DGSCR256.

Table 4.9: The NIST results of DGSCR128 and DGSCR256.

#	Statistical test	DGSCR128	DGSCR256
1	Frequency	0.678686	0.145326
2	Block-frequency	0.437274	0.616305
3	Cumulative-sums(forward)	0.595549	0.048716
	Cumulative-sums(reverse)	0.574903	0.224821
4	Runs	0.455937	0.657933
5	Longest-runs of ones	0.851383	0.494392
6	Rank	0.085587	0.181557
7	DFT (Spectral)	0.080519	0.637119
8	Non-overlapping-template	0.526587	0.45461
9	Overlapping-templates	0.437274	0.075719
10	Universal	0.678686	0.366918
11	Approximate entropy	0.019188	0.834308
12	Random-excursions	0.589688	0.408937
13	Random-excursions variant	0.363008	0.517047
14	Serial1	0.739918	0.911413
	Serial2	0.051942	0.383827
15	Linear complexity	0.759756	0.366918

It is obvious that DGSCR128 has the highest number of highlighted cells. Therefore, DGSCR128 has better statistical properties than DGSCR256.

Table 4.10 shows the NIST results of DGSCI50, DGSCI100, DGSCI128, and DGSCI256.

Table 4.10: The NIST results of DGSCI50, DGSCI100, DGSCI128, and DGSCI256.

#	Statistical test	DGSCI50	DGSCI100	DGSCI128	DGSCI256
1	Frequency	0.798139	0.455937	0.637119	0.213309
2	Block-frequency	0.350485	0.191687	0.181557	0.719747
3	Cumulative-sums(forward)	0.534146	0.213309	0.897763	0.350485
	Cumulative-sums(reverse)	0.955835	0.181557	0.719747	0.657933
4	Runs	0.719747	0.191687	0.171867	0.350485
5	Longest-runs of ones	0.657933	0.383827	0.455937	0.102526
6	Rank	0.987896	0.224821	0.798139	0.085587
7	DFT (Spectral)	0.55442	0.798139	0.224821	0.213309
8	Non-overlapping-template	0.552449	0.493655	0.488985	0.487295
9	Overlapping-templates	0.779188	0.595549	0.897763	0.834308
10	Universal	0.035174	0.304126	0.971699	0.816537
11	Approximate entropy	0.23681	0.678686	0.574903	0.759756
12	Random-excursions	0.33767	0.427572	0.293469	0.375971
13	Random-excursions variant	0.365456	0.435763	0.322444	0.242827
14	Serial1	0.637119	0.015598	0.911413	0.514124
	Serial2	0.224821	0.181557	0.334538	0.455937
15	Linear complexity	0.437274	0.202268	0.851383	0.678686

From Table 4.10, it can be seen that DGSCI50 and DGSCI128 have the highest number of highlighted cells. Furthermore, the DGSCI128 is faster than DGSCI50. Therefore, it can be concluded that the DGSCI128 attained the best result.

DGSCR128 and DGSCI128 are chosen to be compared with the original Grain in order to conform that the modification improves the original algorithm. Table 4.11 shows NIST results for Grain, DGSCR128, and DGSCI128.

Table 4.11: NIST results of Grain, DGSCR128, and DGSCI128.

#	Statistical test	Grain	DGSCR128	DGSCI128
1	Frequency	0.026948	0.678686	0.637119
2	Block-frequency	0.494392	0.437274	0.181557
3	Cumulative-sums(forward)	0.55442	0.595549	0.897763
	Cumulative-sums(reverse)	0.12962	0.574903	0.719747
4	Runs	0.334538	0.455937	0.171867
5	Longest-runs of ones	0.779188	0.851383	0.455937
6	Rank	0.595549	0.085587	0.798139
7	DFT (Spectral)	0.037566	0.080519	0.224821
8	Non-overlapping-template	0.517810	0.526587	0.488985
9	Overlapping-templates	0.153763	0.437274	0.897763
10	Universal	0.350485	0.678686	0.971699
11	Approximate entropy	0.066882	0.019188	0.574903
12	Random-excursions	0.256454	0.589688	0.293469
13	Random-excursions variant	0.414677	0.363008	0.322444
14	Serial1	0.595549	0.739918	0.911413
	Serial2	0.171867	0.051942	0.334538
15	Linear complexity	0.699313	0.759756	0.851383

In Table 4.11, 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 Overlapping-templates, original algorithm has the lowest p-value (0.153763), while DGSCI128 has the highest p-value (0.897763). Therefore, the range of p-values is .0744. This indicates that there is a significant gap in Overlapping-templates between the compared algorithms. On the other hand, the p-values in some tests of the compared algorithms are convergent.

The second observation is that using dynamic polynomial switching improves the statistical properties of the keystream generated by Grain; and that can be seen from the number of highlighted cells. In addition, the best results are obtained when DGSCI128 is used to generate the keystream. Graphical comparisons of the results are displayed in Figures 4.8–4.22.

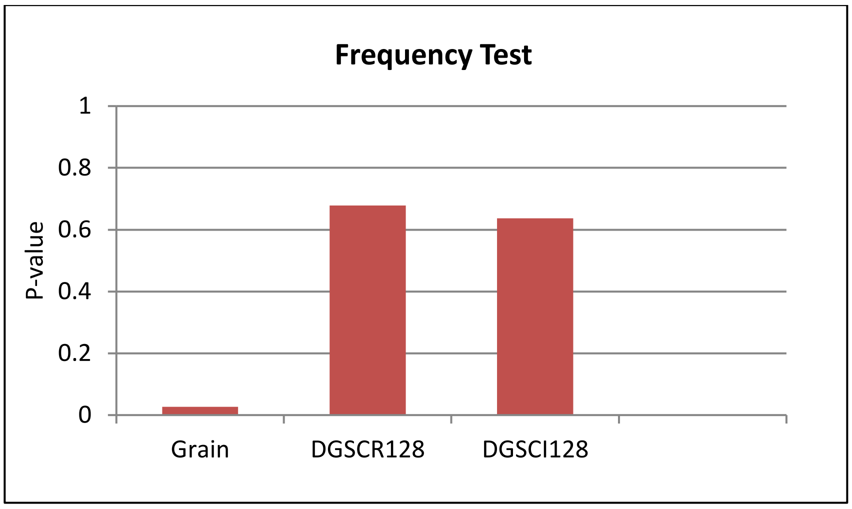


Figure 4.8: Frequency Test.

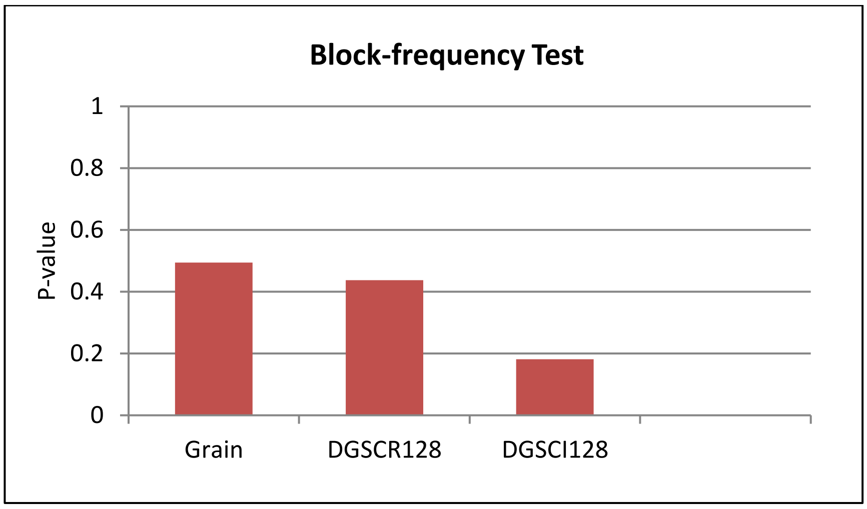


Figure 4.9: Block-Frequency Test.

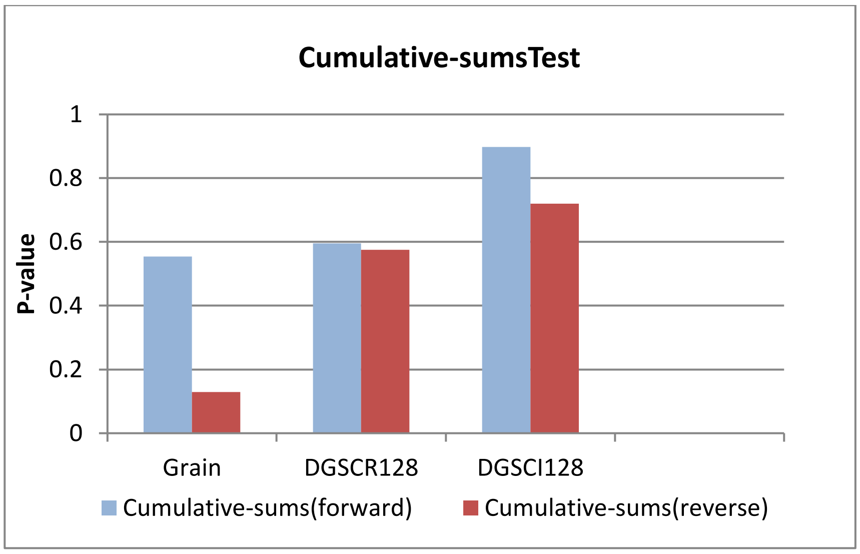


Figure 4.10: Cumulative-sums Test

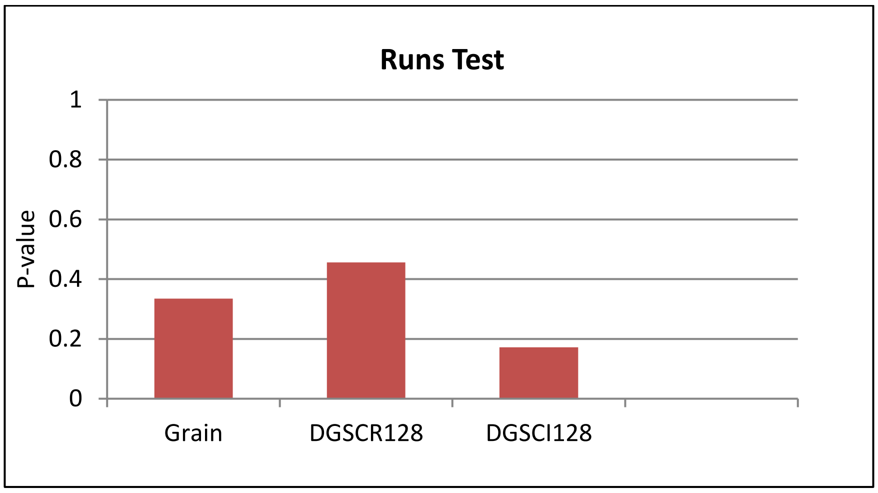


Figure 4.11: Runs Test.

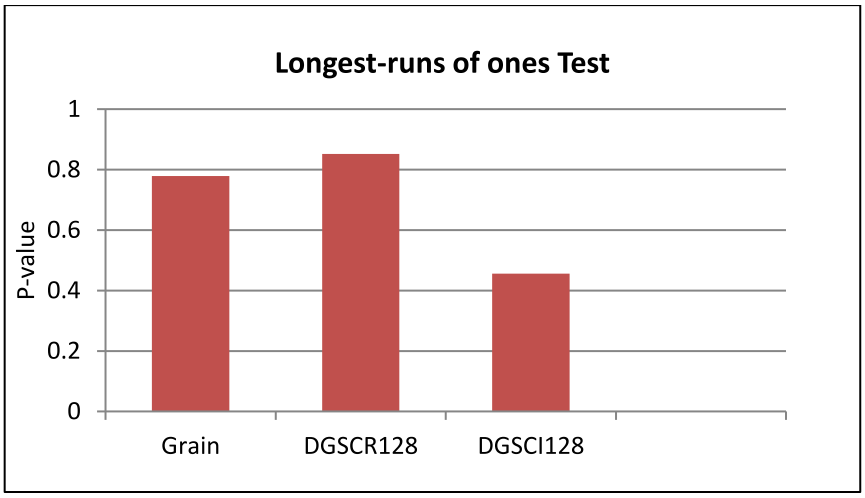


Figure 4.12: Longest-runs of ones Test.

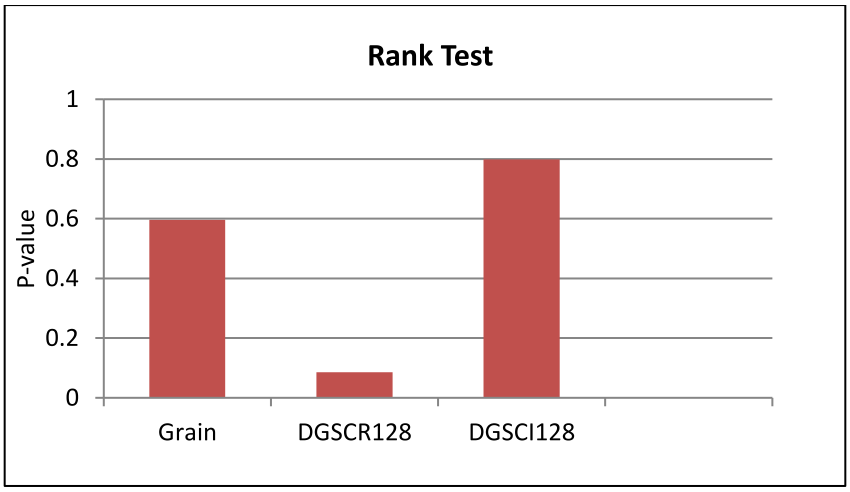


Figure 4.13: Rank Test.

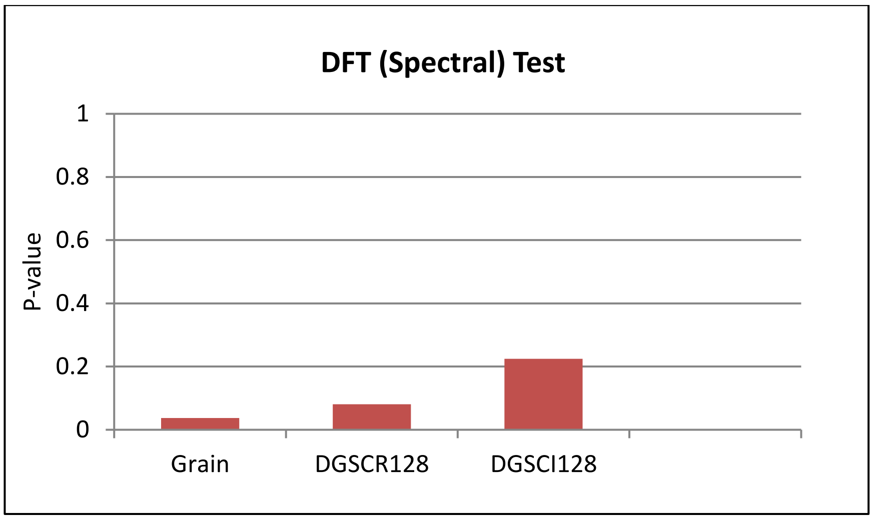


Figure 4.14: Discrete Fourier Transform test.

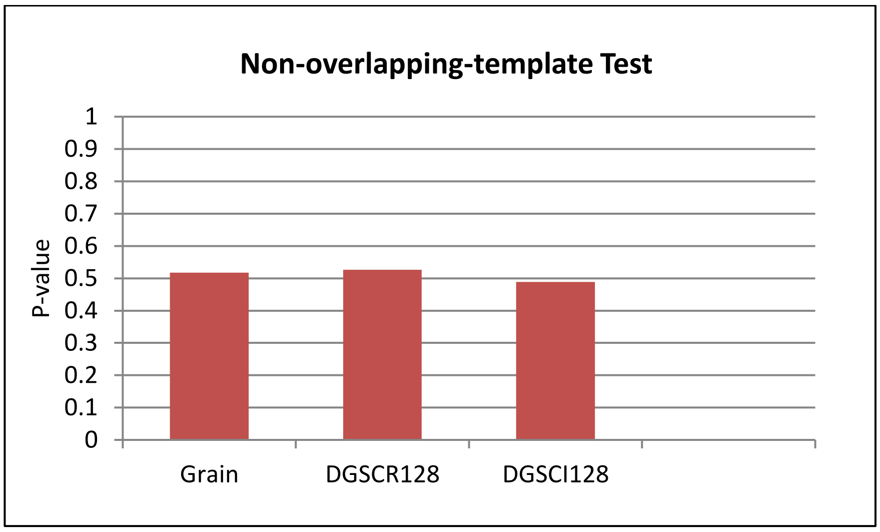


Figure 4.15: Non-overlapping-template Test.

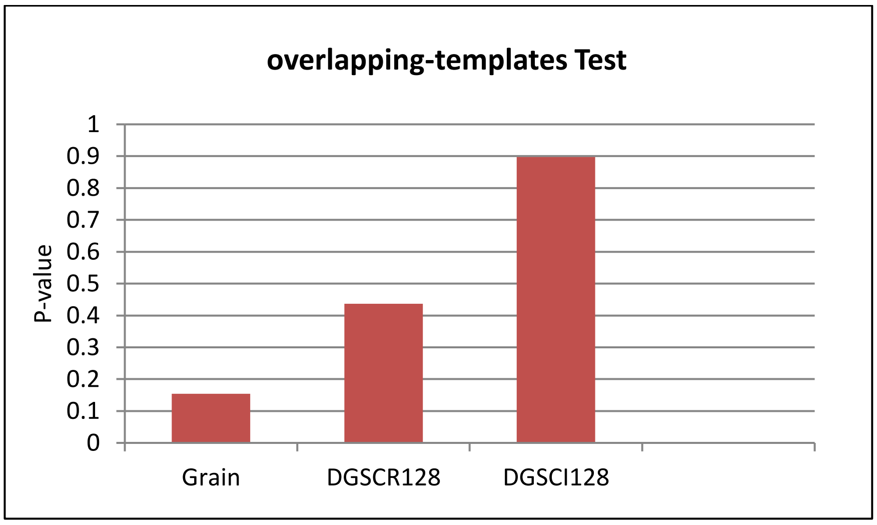


Figure 4.16: Overlapping-templates Test.

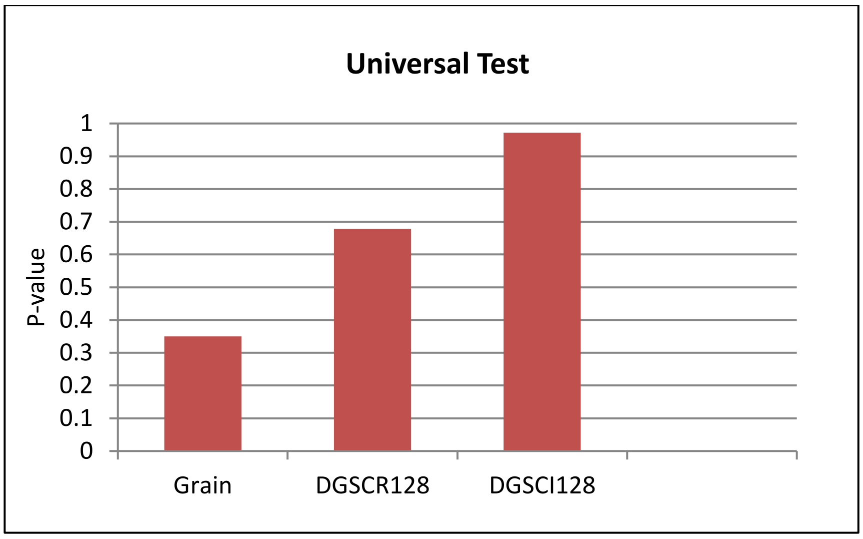


Figure 4.17: Universal Test.

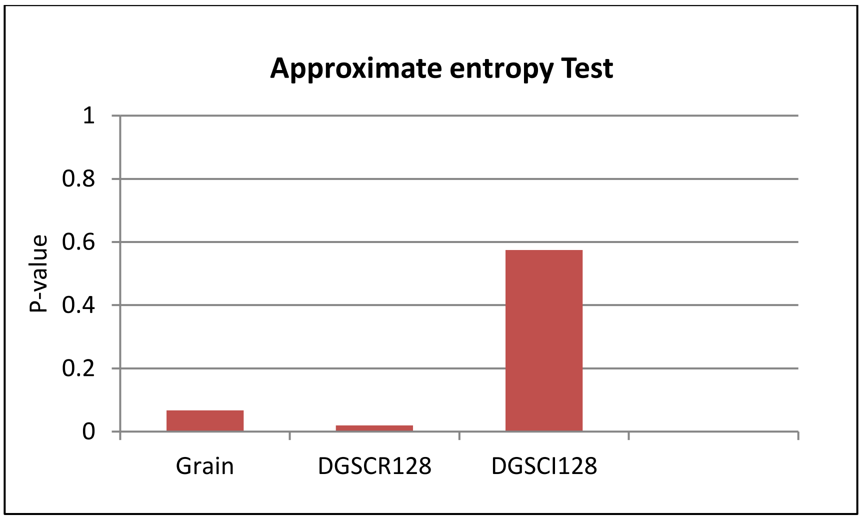


Figure 4.18: Approximate Entropy Test.

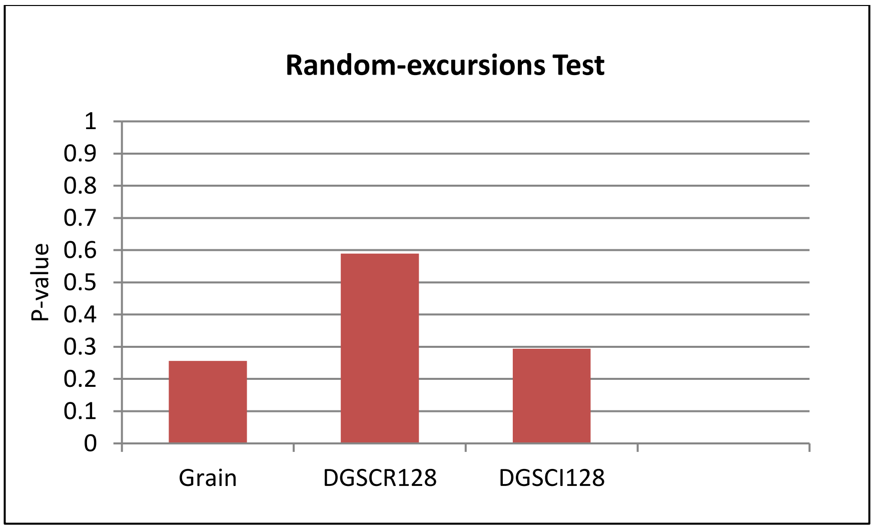


Figure 4.19: Random-excursions Test.

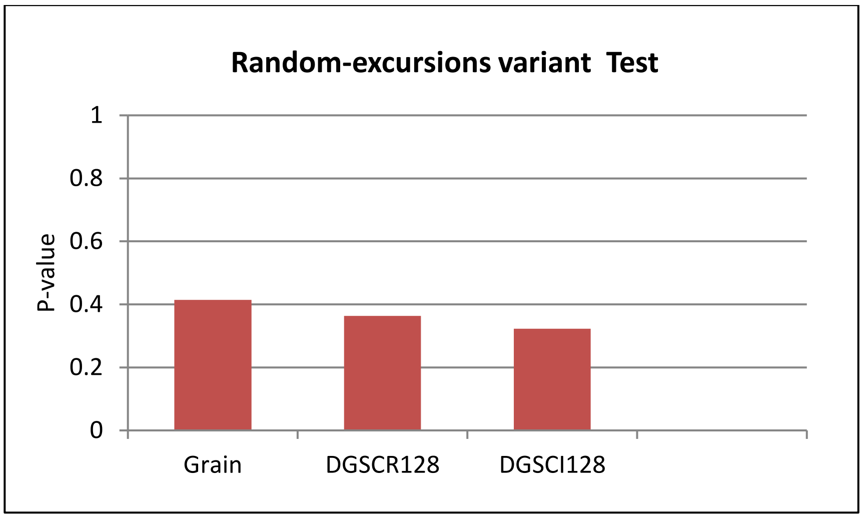


Figure 4.20: Random-excursions Variant Test.

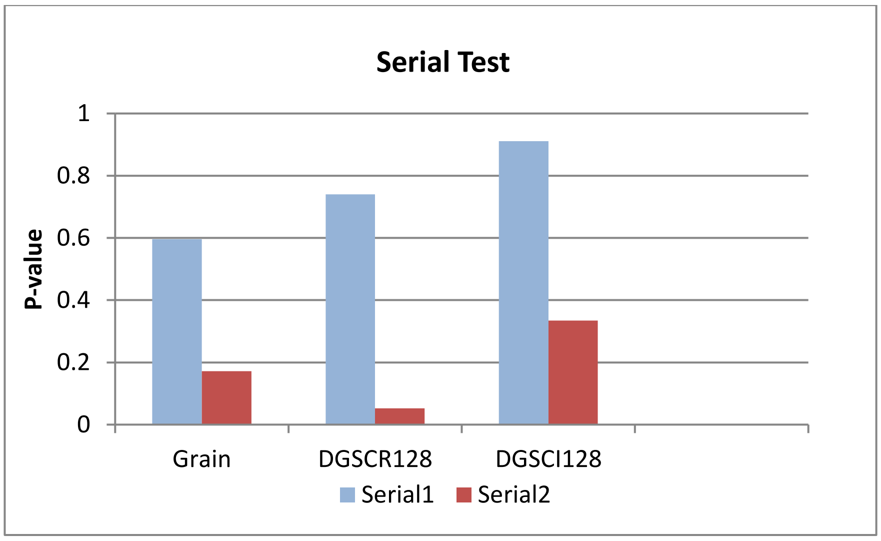


Figure 4.21: Serial Test.

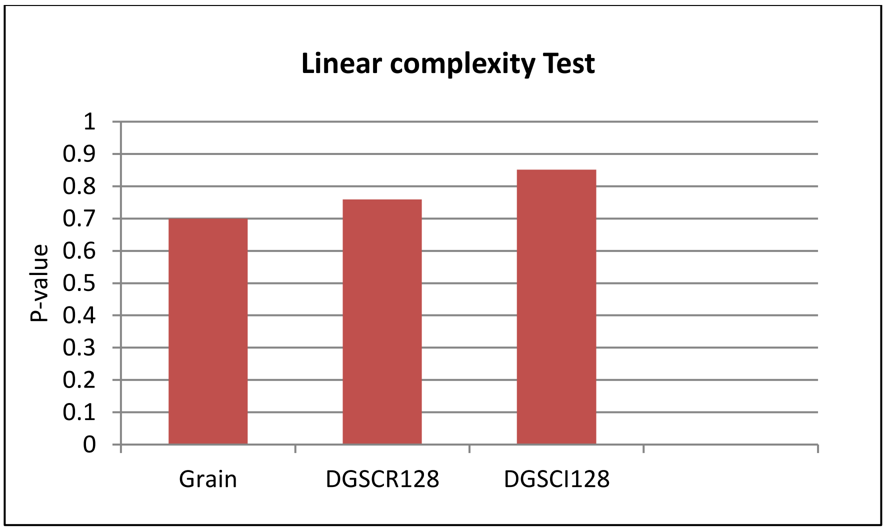


Figure 4.22: Linear Complexity Test.

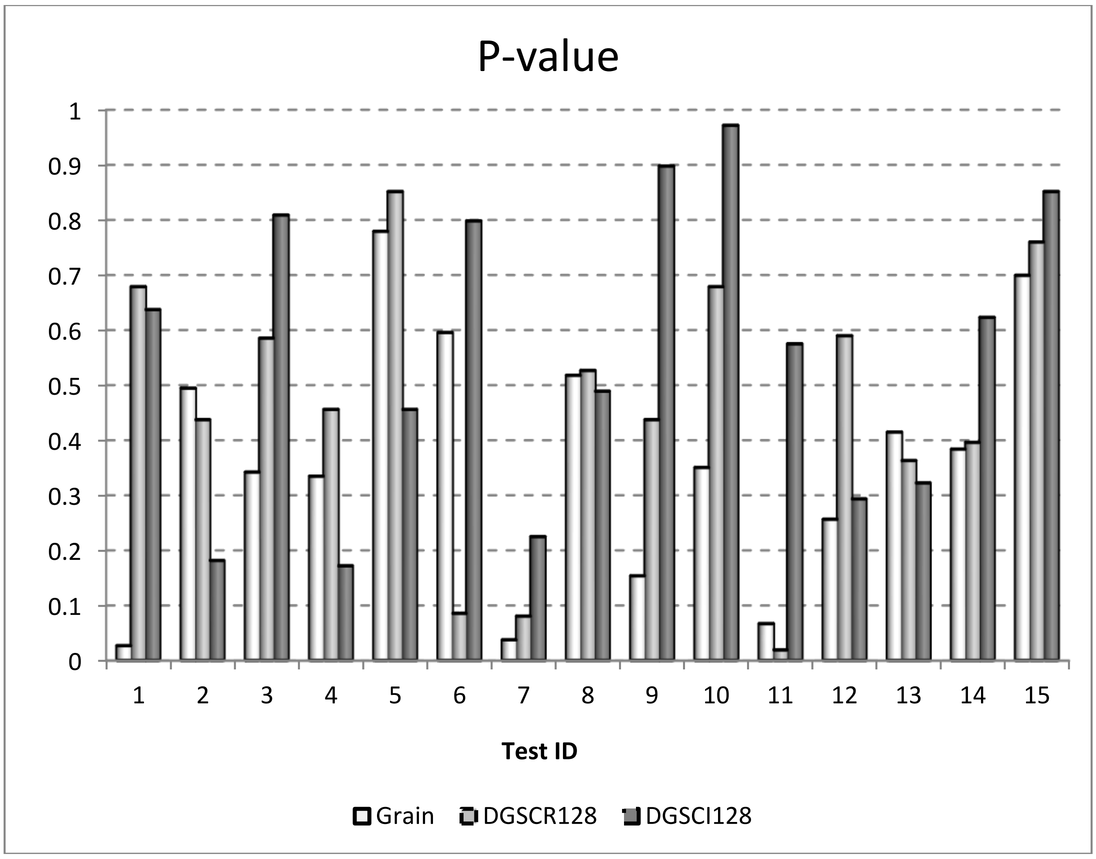


Figure 4.23: comparison of NIST Test results

Figure 4.23 displayed the comparison of NIST Test results between Grain, DGCR128 and DGCI128. It is clear that the proposed algorithm have best results in the most tests (thirteen from fifteen).

4.6 SUMMARY

This chapter presented the NIST Test Suite and its tests. NIST Test suite was used to evaluate the randomness of several versions of the proposed algorithm. Security and performance analysis were conducted in order to prove the level of security and the efficiency of the proposed design with different strategies to change the taps. In addition, the proposed algorithm was compared with the original algorithm; the results indicated that the proposed algorithm outperforms the original cipher in several tests.

CHAPTER FIVE: CONCLUSIONS AND FUTURE WORK

5.1 INTRODUCTION

To promote the design of efficient and secure stream ciphers suitable for widespread adoption, an effort must be made to investigate the security issues of stream ciphers in current use. One important issue in stream cipher design is to investigate the relationship between using dynamic polynomial switching and the security level. To address this issue, extensive research on previous works was conducted. Then, a new algorithm was proposed. In this chapter, the contribution of this research is presented and discussed. Finally, possible future work is suggested at the end of this chapter.

5.2 CONCLUSIONS

The main goal of this chapter is to summarize the conclusions and highlight the contributions of this research, which are detailed as follows:

1. This study proposed a new algorithm called DGSC which is based on Grain stream cipher. It has been shown that the produced keystreams possess a high linear complexity, and good statistical properties. These characteristics and properties make DGSC suitable encryption system for stream cipher applications.
2. In the proposed algorithm, two ways (regular and irregular) are used for switching the feedback polynomials of the LFSR. In the regular way, the taps are changed every specific amount of time. Two options are used, which are changing the taps every 128 bits and changing the taps every 256 bits. In the irregular way, changing the taps happens when there is matching between the output of LFSR and NLFSR for 50, 100, 128, and 256 times.
3. NIST test suite was used to evaluate the statistical properties of the keystream that are generated by the proposed algorithm with several ways of changing the taps. In addition, the proposed algorithm was compared with Grain stream cipher. The results showed that the proposed algorithm outperforms the original Grain in many tests.
4. A performance analysis of the proposed algorithm was carried out. Furthermore, a comparison between the proposed algorithm and the original Grain was performed. The results showed that there is no significant difference in speed between the original Grain and the modified algorithm with regular changing of taps. However, the speed of the proposed algorithm with irregular changing of taps decreases.

5. Despite the fact that using irregular ways to change the feedback bits slows down the algorithm, it is more secure than using regular ones, because these ways increase the nonlinearity of the produced sequence.
6. Overall, the DLFSR could be used instead of LFSR in stream ciphers design to enhance the security of these ciphers. Using regular or irregular ways to change the taps may depend on the need of specific industries. For instance, the mobile industry can customize its stream ciphers to be light weight by using regular ways for changing the taps; while banking and military industries can use irregular tap switching to enhance the security level.

5.3 FUTURE WORK

As a future work, the proposed algorithm can be exposed to several attacks in order to prove the security of this cipher. Moreover, a comparison between the proposed stream cipher and several currently used stream ciphers might be conducted.

REFERENCES

[1] A. Eljadi, F. Mohamed, T. Al-Shaikhli, and I. Fakhri, "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.

[2] B. Schneier, *Applied cryptography: protocols, algorithms, and source code in C*: john wiley & sons, 2007.

[3] M. Stamp, *Information security: principles and practice*: John Wiley & Sons, 2011.

[4] C. Cid, S. Kiyomoto, and J. Kurihara, "The rakaposhi stream cipher," in *Information and Communications Security*, ed: Springer, 2009, pp. 32-46.

[5] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*: Springer Science & Business Media, 2009.

[6] 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.

[7] A. Peinado, J. Munilla, and A. Fúster-Sabater, "Improving the Period and Linear Span of the Sequences Generated by DLFSRs," in *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, 2014, pp. 397-406.

[8] R. Stepien and J. Walczak, "Comparative analysis of pseudo random signals of the LFSR and DLFSR generators," in *Mixed Design of Integrated Circuits and Systems (MIXDES), 2013 Proceedings of the 20th International Conference*, 2013, pp. 598-602.

[9] S. Kiyomoto, T. Tanaka, and K. Sakurai, "K2: A Stream Cipher Algorithm using Dynamic Feedback Control," in *SECRYPT*, 2007, pp. 204-213.

[10] S. Khan, A. Khan, S. Khayal, T. Naz, S. Bashir, and F. Khan, "Dynamic feedback based modified SNOW 2.0," in *Emerging Technologies (ICET), 2010 6th International Conference on*, 2010, pp. 250-255.

[11] R. Mita, G. Palumbo, and M. Poli, "Pseudo-random sequence generators with improved inviolability performance," *IEE Proceedings-Circuits, Devices and Systems,* vol. 153, pp. 375-382, 2006.

[12] R. Mita, G. Palumbo, S. Pennisi, and M. Poli, "Pseudorandom bit generator based on dynamic linear feedback topology," *Electronics Letters,* vol. 38, pp. 1097-1098, 2002.

[13] S. Babbage and M. Dodd, "The stream cipher MICKEY (version 1)," *ECRYPT Stream Cipher Project Report,* vol. 15, p. 2005, 2005.

[14] D. Horan and R. Guinee, "A novel keystream generator using pseudo random binary sequences for cryptographic applications," 2006.

[15] 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.

[16] C. Berbain, H. Gilbert, and A. Maximov, "Cryptanalysis of grain," in *Fast Software Encryption*, 2006, pp. 15-29.

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

[18] A. Molina-Rueda, F. Uceda-Ponga, and C. F. Uribe, "Extended period LFSR using variable TAP function," in *Electronics, Communications and Computers, 2008. CONIELECOMP 2008, 18th International Conference on*, 2008, pp. 129-132.

[19] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on computing,* vol. 15, pp. 364-383, 1986.

[20] 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.

[21] 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.

[22] I. F. Al-shaikhli, M. A. Alahmad, and K. Munthir, "Hash Function of Finalist SHA-3: Analysis Study," *Information Technology (IJACSIT),* vol. 2, 2013.

[23] J. Melià-Seguí, J. Garcia-Alfaro, and J. Herrera-Joancomartí, "J3Gen: A PRNG for low-cost passive RFID," *Sensors,* vol. 13, pp. 3816-3830, 2013.

[24] T. Good and M. Benaissa, "Hardware performance of eStream phase-III stream cipher candidates," in *Proc. of Workshop on the State of the Art of Stream Ciphers (SACS'08)*, 2008.

[25] 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.

[26] C. De Cannière, Ö. Kücük, and B. Preneel, "Analysis of Grain's initialization algorithm," in *Progress in Cryptology–AFRICACRYPT 2008*, ed: Springer, 2008, pp. 276-289.

[27] S. Banik, S. Maitra, and S. Sarkar, "A differential fault attack on grain-128a using MACs," in *Security, Privacy, and Applied Cryptography Engineering*, ed: Springer, 2012, pp. 111-125.

[28] 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.

[29] 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.