Design and Analysis of Symmetric Random Function Generator as a New Cryptographic Primitive

thesis submitted to

LOVELY PROFESSIONAL UNIVERSITY

for the award of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

Rahul Saha
41400102

Under the supervision of

Dr. G. Geetha
Professor, Division of Research and Development

FACULTY OF TECHNOLOGY AND SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

May 28, 2018

DECLARATION

I declare that the thesis entitled ”Design and Analysis of Symmetric Random Function Generator as a New Cryptographic Primitive” has been prepared by me under the guidance of Dr. G. Geetha, Professor, Division of Research and Development, Lovely Professional University, India. No part of this thesis has formed the basis for the award of any degree or fellowship previously.

Rahul Saha
School of Computer Science and Engineering
Lovely Professional University
Jalandhar Delhi G.T.Road (NH-1)
Phagwara, Punjab 144411
India
Date: May 28, 2018

CERTIFICATE

This is to certify that the thesis entitled ”Design and Analysis of Symmetric Random Function Generator as a New Cryptographic Primitive”, which is being submitted by Mr. Rahul Saha for the award of the degree of Doctor of Philosophy in Computer Science and Engineering from the Faculty of Engineering and Technology, Lovely Professional University, Punjab, India, is entirely based on the work carried out by him under my supervision and guidance. The work reported, embodies the original work of the candidate and has not been submitted to any other university or institution for the award of any degree or diploma, according to the best of my knowledge.

Dr. G. Geetha
Professor
Division of Research and Development
Lovely Professional University
Phagwara, Punjab-144411, India
Date: May 28, 2018

DEDICATION

"Cryptography is the ultimate form of non-violent direct action."

Julian Assange

This thesis is dedicated to my parents Mr. Goutam Saha and Mrs. Shikha Saha.

ACKNOWLEDGEMENT

First of all, I thank to God for making me able to use my knowledge and wisdom for this research work and giving me patience in all the hard times during this period of time.

I would like to express my sincere thanks to my supervisor, Dr. G. Geetha, for her supervision, advice, and guidance for each and every stage of this research. I am truly very fortunate to have the opportunity to work with her.

I am grateful to my best friend and fellow researcher, Dr. Gulshan Kumar for his constructive criticism and suggestions. His motivation has worked as a catalyst in the completion of the research work in due time. Besides, I would also like to thank Dr. Mritunjay Kumar Rai and Mr.Vikash Dahiya for helping me in my research work with their valuable suggestions .

I would like to take this opportunity to record our thanks to the entire family of Lovely Professional University, specifically Division of Research and Development,School of Computer Science and Engineering and most obviously School of Computer Applications for all the support and encouragement throughout the research work. I am also very much grateful to my parents, Ms. Shikha Saha and Goutam Saha, my uncle and aunt Prof.Subal Chandra Bera and Ms. Shila Bera and all other family members for being there as moral support and keeping me always active with their belief, love and blessings on me.

Date: May 28, 2018 Rahul Saha

Abbreviations

Abbreviations	Description
IoTs	Internet of Things
HTTPS	Hyper Text Transfer Protocol (Secure)
LA	Level of Assurance
AES	Advanced Encryption Standard
SRFG	Symmetric Random Function Generator
DES	Data Encryption Standard
DFT	Discrete Fourier Transform
FSM	Finite State Machine
LFSR	Linear Feedback Shift Register
NLPFSR	Non-Linear Parallel Feedback Shift Register
GA	Genetic Algorithm
GP	Genetic Programming
CGP	Cartesian Genetic Programming
ES	Evolution Strategies
APC	Aperiodic Propagation Criteria
ANF	Algebraic Normal Form
KSA	Key Scheduling Algorithm

ABSTRACT

Information systems are coherent with applications. With the increasing number of applications, information systems have also been increasing day by day. Therefore, Information theory has been evolved from its beginning form to its present one. The story progressive development of information theory has redefined our world in a fascinating way. It has its major applications in Internet of Things (IoTs), cloud computing, big data analysis, social networking, artificial intelligence and many more. Now, the question is whether we need any safety and security for this information theoretic models. The answer is validated with the categories of information. The data or information can be classified as: unclassified, classified, confidential, secret and top secret. Therefore, different types of security services are required. Researches have been done rigorously on these aspects and it has been found cryptographic processes are the one of the most suitable way to provide multidimensional security provisions.

A number of algorithms, processes and methods have been developed so far in the domain of cryptography as well as cryptology. At the time of introduction of these algorithms, they were working with their efficiency, but with the technological progress and the sophistication of cryptanalysis processes have opened up the weaknesses of these algorithms and some of them are obsolete now. Therefore, the cryptology designers and coders have urged the need of randomness in the algorithms so that the algorithms are able to withstand the cryptanalytic attacks.

In this thesis, we have developed a symmetric and balanced function generator which produces random bit patterns. This function generator operates on random selection of variables given as input and also selects random basic GATEs (among AND, OR, XOR, NOT) for the operation. As a result, this produces a random output which we have utilized in the key expansion of AES algorithm and key scheduling of RC4 algorithm. We have analysed its properties and behaviour to evaluate the performance of the developed function generator. We have also identified some new propositions through our experimentation. Moreover, this can be used as a cryptographic primitive in different other cryptographic processes or algorithms. Besides cryptography, this function generator can be used in other fields such as statistical analysis, mechanical and electronics engineering and many more.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 General

Information systems are coherent with applications. Information systems deal with the systems with a specific reference to information. It considers the complementary networks of hardware and software that people and organizations use to collect, filter, process, create and also distribute data. When we talk about information systems, we must know that all the information systems possess some data that works as information. Moreover, the distribution of such information over the network is also critical process in our present digital age. The urge of the process has emerged another domain of the computing science termed as information theory. The process of information theory is started its journey with the research of Claude Shannon's in 1948 [1]. Information theory has been evolved from its beginning form to its present one. The story progressive development of information theory has redefined our world in a fascinating way. It provides the opportunity to study the social, political, and technological interactions among entities. It has helped to provide a guidance for the development of the information theory constructions and has defined its trajectory of progress. It provides an insight into how information theory can be modelled as a mathematical model and can be conjugated with a new emerging field of technology. Information theory begins with a broad spectrum of fields, ranging from management to biology, all information theory to be a 'magic key' to multidisciplinary understanding. It has its major applications in Internet of Things (IoTs), cloud computing, big data analysis, social networking, artificial intelligence and many more. This field has moved from the initial propositions to the various influences which have narrowed its focus specifically to the computing relations to is distribution in technological domain. Within these established boundaries, external influences such as the constrained space has steered the progress of the field. Further, the expansion of information theory has been constantly controlled by hardware technological limitations for various applications. As we have said, information theory is coined by Claude Shannon, he has not used the phrase of information theory in his research. The main objective is "information" need to be quantified, analyzed, and reduced to a mathematical formula; therefore the research work and now the domain of Information Theory has attracted a number of researchers.

Let us take some of the very influential applications of information theory in the recent times. The present world is converging to the domain of IoT where all the applications are going to used internet based infrastructure for the communication among them. Senor networks, vehicular networks, road-side traffic analysis, smart home systems all are going to generate the huge amount of data as well as the processed information. Along with advantages of the virtualization, cloud computing draws an attention of the information technology domain. The services of cloud are also good sources of information which are led to data science perceptions. The automation systems and the artificial intelligence with robotics need rigorous dealing with data and information. All the information generated or processed by the above systems must follow the information theory so that the applications are modelled with mathematical interpretations. These mathematical models of information theory help us to monitor or control the results of the systems.

Now, the question is whether we need any safety and security for this information theoretic models. The answer is validated with the categories of information. The data or information can be classified as: unclassified, classified, confidential, secret and top secret. Let us take some examples. We often save our personal pictures, videos, credentials of credit-debit cards, bank details in our laptop and tablets. We use passwords, cryptolockers or such applications for the protection of such our personal data. Think of another scenario. We have the transaction data of bank need to be transmitted over network. So, if the network is not secure any third party can modify or intercept the information in between of transmission. Therefore, we use secure connections with HTTPS. The strategic planning of a nations defence domain is at the most top secret data to be considered. So, these data or information need to be secure for sharing among personnel. Three main security aspects are always in consideration for any information sharing process: confidentiality, authenticity and integrity. The data for IoT, data access in clouds, verification of threshold values in artificial intelligence must require such security aspects, otherwise the systems cannot be considered as success. Therefore, we can understand that if information theory is evolving with its more developed methods and processes, the approaches for providing the security aspects to such information processing need to be rigorously researched for the progress of information theory.

1.2 Motivation

In recent times, cryptography has attained lot of importance. As a natural consequence, its hardware effective, both in terms of time and space requirement. Being a part of the computer science, information theory has been considered as one of the integral part of this domain as all the applications of computer science follow the mathematical models of information theory. Further, the security requirements of the information theory have always worked as a motivational point for our research work. The algorithmic features of the cryptography are one of the deterministic means of providing security services. Moreover, all the distributed infrastructures such as internet, data processing use cryptographic security features in the form of encryption, hashing, digital signature, key management etc.

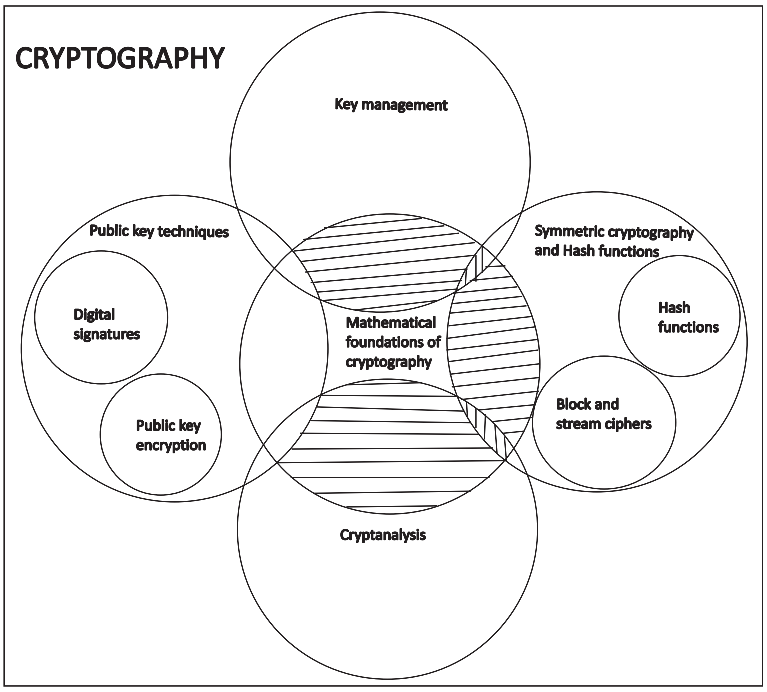


Figure 1.1: *Domain of the present work with ACM taxonomy*

A number of researches have been executed to design robust cryptographic algorithms. Some algorithms work efficiently and some have been dominated by the cryptanalysis attacks. We have analysed the algorithms of the previous works in this domain and have identified some of the prominent research gaps as discussed in the next chapter. We have found that the existing algorithms are lacking in incorporating randomness in their corresponding features. Therefore, we have been motivated towards the generation of a new cryptographic feature as a random function generator.

Our motivation towards the present research work is an addition of an attribute in the field of intersected domain of mathematical foundations of cryptography and block and stream ciphers. The shaded area in Figure 1.1 shows the domain of the present work according to ACM taxonomy.

1.3 Basics of Cryptography

Cryptography from Greek κρυπτός *kryptós*, "hidden, secret"; and $\gamma\rho\acute{\alpha}\varphi\varepsilon\iota\nu$ *graphein*, "writing", or -$\lambda o\gamma\iota\alpha$ -logia, "study", respectively is the practice of executing secret communication in the presence of third parties or adversaries. It plays an important role in the concern of logical barriers to attacks on information systems security.

Cryptography [2] in the previous time was analogous to encryption where the main task was to convert the readable message to an unreadable format. The process was complex as the sender needed to send the decoding technique personally to the receiver. Modern cryptography, in comparison, is easy to manage as it is an intersection of mathematics, computer science and electrical engineering. Different types of transformations are used in modern cryptography. The schemes of modern cryptography are computationally secure because practical breaking down of such cryptosystems is infeasible with any practical means.

The growth of cryptographic technology has emerged a number of legal issues in this digital information era. The usage of cryptography as a tool for espionage and sedition has led many governments to consider it as a potential weapon limiting or even prohibiting its use. In some jurisdictions where the use of cryptography is legal, laws permit investigators to compel the disclosure of encryption keys for documents relevant to an investigation. Cryptography is also having a significant role in digital rights management and piracy of digital media.

In our daily life, we often experience the cryptographic measures and methods. From electronic mail to cellular communications, from secure web browsing to digital cash, everywhere the applications of cryptographic algorithms is observed. Therefore, the information systems and the data communication among such information systems are dependable on such cryptographic schemes to make the process enough secure.

Different applications [3] of cryptography demand for different types of security services. Cryptography provides the following services [2].

Confidentiality: the data is hidden from the third party.

Authentication: the data is handled by only the legitimate person.

Integrity: the data is not changed in the process of communication.

Non-repudiation: the responsibility of sending and receiving the data cannot be avoided by either sender or the receiver.

1.3.1 Types of cryptography

There are two basic categories of cryptography [2][3][4] exist in this domain- Symmetric cryptography and Asymmetric cryptography [5]. Both the categories convert the plaintext (original message) to ciphertext (encrypted message) with the help cipher and the vice versa is done with the help of decipher.

Symmetric key cryptography: Symmetric key cryptography [8] deals with the usage of a single shared key between a sender and a receiver. The schematic diagram of this process has shown below in Figure 1.2

Asymmetric key cryptography: Asymmetric key cryptography [5] deals with two keys for an encryption-decryption process. They key which is public is used for encryption process and another key called as private key, which is mathematically related with the public key, is used to decrypt the encoded message. The schematic diagram of asymmetric key cryptography is shown below in Figure 1.3.

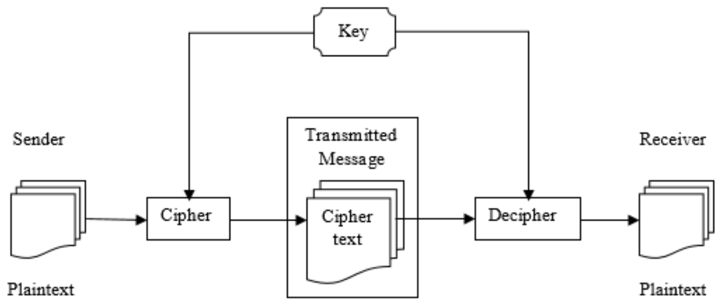


FIGURE 1.2: *Schematic diagram of symmetric cryptography process*

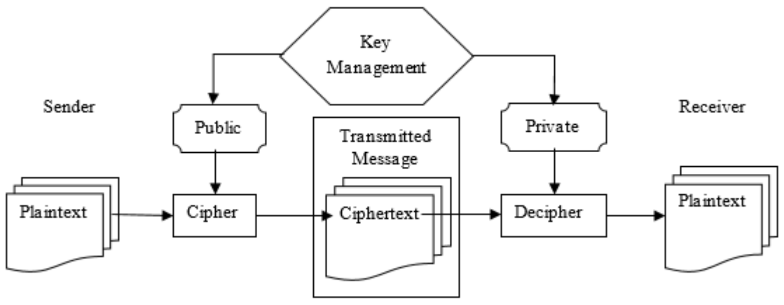


FIGURE 1.3: *Schematic diagram of asymmetric cryptography process*

Both the symmetric key and asymmetric key cryptography have their relative advantages and disadvantages. Symmetric is faster than the asymmetric key process and has more strength with a large size of key. On the other hand, asymmetric key process is slower though it maintains better scalability. It can also provide authentication and non-repudiation which is not provided by the former.

Another way of categorization [2] of the cryptographic algorithms depends upon the data it works upon. One of them is called as block ciphers which deal with the predefined size of the block of data on which the cipher is applied. For example, DES algorithm works with 64 bit data block. On the other hand, ciphers like RSA works on data stream where each bit of data is encrypted with the cipher until the total bits of data are encrypted.

1.3.2 Cryptographic Algorithm Metrics

A number of algorithms are getting developed each day. Some of them are eventually getting approved by different standard organization and some of them are getting rejected in comparison with others. This selection of the cryptographic algorithms is based upon some metrics which helps to provide a utility for Common Criteria Level of Assurance (LA). These metrics assist in developing a framework for specifying the appropriate measures to design a cryptographic algorithm. Although all the characteristics of the algorithms cannot be quantified, the parameters or the metrics of such algorithms can be objective or subjective. Cryptographic algorithms characteristics [3][4] which are considered for the development of metrics have been listed below.

Type: This metric is subjective as the type of algorithm is developed is depending upon the key structure used for the algorithm. If the algorithm uses a shared secret key, it is considered to be the symmetric cryptographic algorithm and if the algorithm uses two keys (public and private), it is considered as asymmetric cryptography. But the type of the algorithm leads to the measurement of complexity and time consumption factors of the algorithms.

Structure of the algorithm: This is another subjective metric which depends upon the plaintext format requirement for the algorithms. The algorithms can use data block as a smallest unit considered as block ciphers and some algorithms use each bit as a smallest unit to be encrypted in each iteration, considered as stream cipher. Moreover, the structure of the algorithm also makes an effect on the strength of the algorithms. For example, most of the block ciphers use Feistel structure [9][10] which emphasizes on permutation of data blocks at the starting of each round and in the completion of all rounds. This attributes the algorithm with security by providing confusion and diffusion.

Functions: Different cryptographic algorithms are used to provide different security services such as confidentiality, integrity, authentication, non-repudiation and so on. Each of the services deals with their corresponding factors to be evaluated. For example, the algorithm for confidentiality must emphasis on the key for better perspective. But, the algorithm for authentication or integrity must consider the hash functions that are used to develop the algorithm.

Key size: Keysize is represented in the terms of bits. The security aspect of all the cryptographic algorithms is a function of length of key. Higher the bits, more is security and less bits, less security as the less bit counts of keys are vulnerable to the brute force attacks, factoring attacks or discrete log attacks.

Rounds: Rounds are specifically not having any threshold as per the metric concern. As one time pad is having only one round with block size of 1 bit, but still this classical cipher is good. More number of rounds is adopted to generate more confusion and diffusion property [7] in the algorithm which is desired characteristics of a good cryptographic algorithm. This round metric is easy to quantify and therefore can be considered as a measurable metric for cryptographic algorithms.

Complexity: The complexity of a cryptographic algorithm depends upon the setup of encryption, decryption and the key. Basically, the round functions consist of different bitwise operations, modular arithmetic. This metric is critical because the cryptographic algorithm must not be too complex as the algorithms are used even for resource constraint environments. The main objectivity of this metric is to execute a parallelism in the operations.

Cryptanalysis: Cryptography does not deal with only developing encryption or decryption algorithms; the algorithms need to be proved to have strength by the cryptanalysis process [6]. Brute force attack, factoring, linear and differential cryptanalysis are some of the best known attacks that are executed on the cryptographic algorithms to identify the strength of the algorithms. The number of steps performed for the attack, the time requirement and the type plaintext or ciphertext used for the attack are also the parts of this metric.

1.4 Research Objectives

Cryptography is a field of intense research where each and every day a new algorithm emerges to make more secure of all the aspects in the information security or network security. The security services such as confidentiality, authentication, non-repudiation, integrity can be achievable by this variety of cryptographic algorithms.

In our present day, all the applications, whether related to network security or data security, use the cryptographic algorithms for the security services. All the cryptographic algorithms use key, functions as their integral component. To provide the security, keys are confidential. But, the functions that are used in the algorithms are public. Therefore, the security of the functions is somehow less if the keys are revealed or guessed in any manner. The prime objectives of our ongoing research work have been summarized below.

1. To design a symmetric random function generator as a cryptographic primitive.
2. To develop a block cipher using the developed cryptographic primitive.
3. To develop a stream cipher using the developed cryptographic primitive.
4. To evaluate and analyse the performance of the cryptographic primitive using the developed block and stream cipher.

1.5 Researcher's Contribution

The researcher's contribution in this area is illustrated by the flow chart given in Figure 1.4 and is summarized as follows:

Symmetric Random Function Generator development: We have introduced a function generator that produces the symmetric balanced output in the sense of the number of 1's and 0's in the output string irrespective of the input string. The concept of the present work must not be confused with the existing symmetric Boolean function concept. Both the concepts are distinguished by a line of difference: Symmetric Boolean functions consider the hamming weight of the input string, whereas our present work considers the hamming distance of output string. Moreover, symmetric boolean function is a specialized function whereas, our present work outputs a combined function comprised of logic GATEs. The function generator is able to randomly select of boolean functions and randomly select variable inputs for the functions for example, suppose we have the basic gates as AND, OR, NOT, XOR and three variables as V_1, V_2, V_3. Then the outputs can be: V_1 XOR V_2 NOT V_3, V_1 AND V_3 XOR V_2 and many more. These combinations are selected dynamically and randomly for both the operators and variables. It confirms all the desirable cryptographic properties and therefore is well suited for cryptographic algorithms.

Modified AES with function generator: We have used our symmetric random function generator in AES algorithm which is popular as a block cipher. The modification has been done in the key scheduling process of AES. The results show that the present function generator has increased the parametric efficiency of AES and therefore more robust against cryptanalysis.

Modified RC4 with function generator: We have used the developed random function generator in RC4 algorithm which is popular as stream cipher. The modification has done with the key expansion module in RC4. The modification enhances the cryptographic strengths the algorithm.

Behaviour comparison of the function generator: The development of the modified AES and modified RC4 lead to the process behaviour comparison of the developed function generator in block cipher domain and stream cipher domain. Therefore, it will further generate some open research problems for future crypto designers.

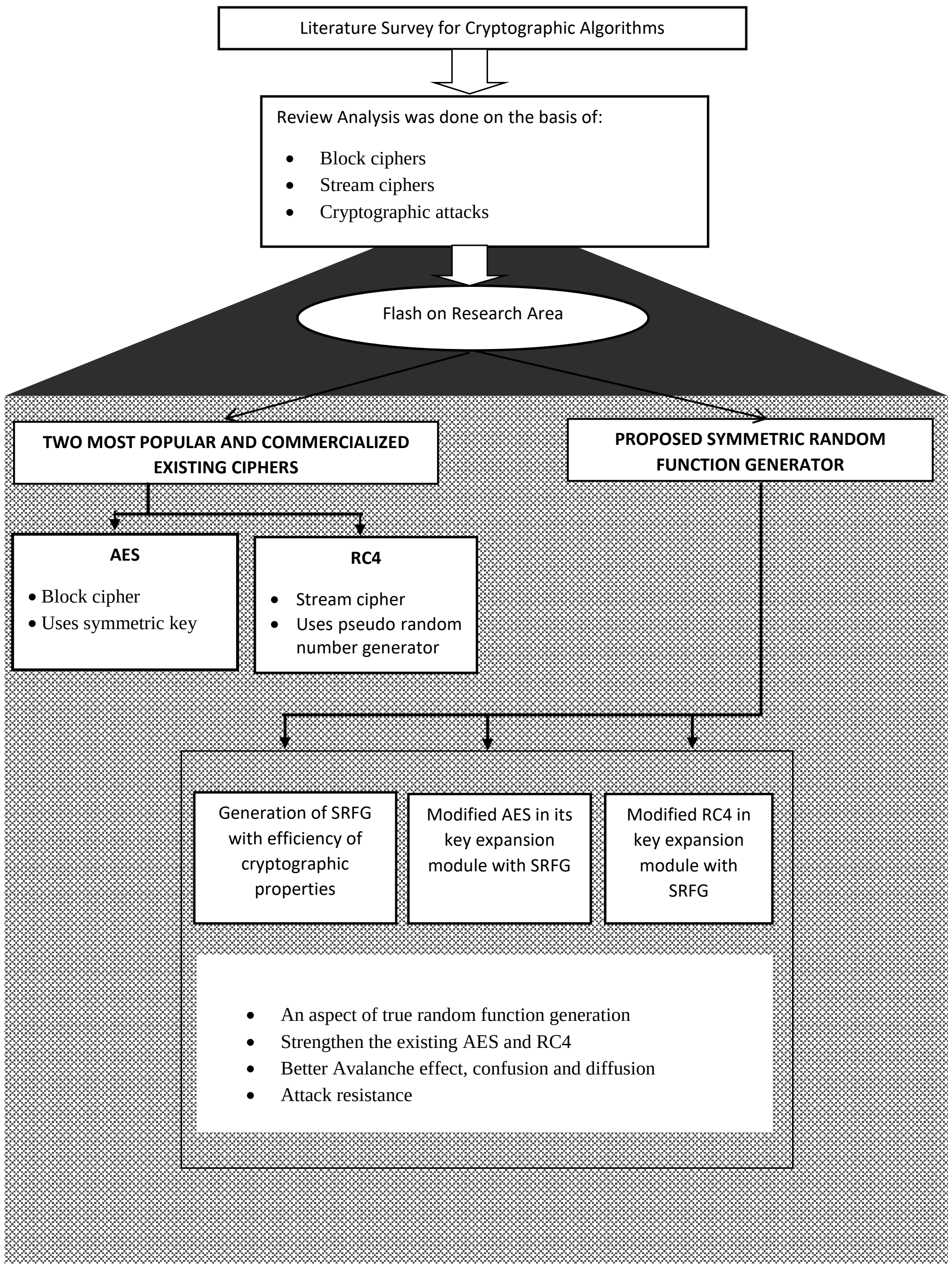

Literature Survey for Cryptographic Algorithms
Review Analysis was done on the basis of:
• Block ciphers
• Stream ciphers
• Cryptographic attacks
Flash on Research Area
TWO MOST POPULAR AND COMMERCIALIZED EXISTING CIPHERS
PROPOSED SYMMETRIC RANDOM FUNCTION GENERATOR
AES
• Block cipher
• Uses symmetric key
RC4
• Stream cipher
• Uses pseudo random number generator
Generation of SRFG with efficiency of cryptographic properties
Modified AES in its key expansion module with SRFG
Modified RC4 in key expansion module with SRFG
• An aspect of true random function generation
• Strengthen the existing AES and RC4
• Better Avalanche effect, confusion and diffusion
• Attack resistance

FIGURE 1.4: *Research Contribution*

1.6 Organization of the thesis

The thesis is organized in seven chapters. A brief outline of the chapters is given below.

Chapter 1 introduces to the domain of cryptography leading to the basic information and working of cryptographic categories. It highlights some of the key points in this domain and also gives brief about the application oriented development of WSNs. Author's contribution has also been highlighted in this chapter.

Chapter 2 shows the related research work by different researchers in this domain. The review work has been categorized in three basic parts. Firstly, the existing algorithms in the domain of block ciphers and stream ciphers and related attacks. Secondly, different weaknesses and modifications of AES have been reviewed and lastly, research works on RC4 have been analysed.

Chapter 3 proposes a modification of AES algorithm. Along with the modification, the behaviour of the modified AES is also discussed. The experimented observations for the modifications are thoroughly analysed.

Chapter 4 proposes a modification of RC4 algorithm. Along with the modification, the behaviour of the modified RC4 is also discussed. The experimented observations for the modifications are thoroughly analysed.

Chapter 5 concludes the thesis highlighting comparison of the behaviour of developed function generator with respect to block and stream cipher. It also emphasizes the prime outcomes of the current research and significant contribution of the thesis and notifies about the scope for future research in this area.

CHAPTER 2

Review of Literature

Cryptography [2] in the previous time was analogous to encryption where the main task was to convert the readable message to an unreadable format. The process was complex as the sender needed to send the decoding technique personally to the receiver. Modern cryptography, in comparison, is easy to manage as it is an intersection of mathematics, computer science and electrical engineering. Different types of transformations are used in modern cryptography. The schemes of modern cryptography are computationally secure because practical breaking down of such cryptosystems is infeasible with any practical means.

The growth of cryptographic technology has emerged a number of legal issues in this digital information era. In our daily life, we often experience the cryptographic measures and methods. From electronic mail to cellular communications, from secure web browsing to digital cash, everywhere the applications of cryptographic algorithms is observed. Therefore, the information systems and the data communication among such information systems are dependable on such cryptographic schemes to make the process enough secure. Different applications [3] of cryptography demand for different types of security services.

Cryptography does not deal with only developing encryption or decryption algorithms; the algorithms need to be proved to have strength by the cryptanalysis process [5]. Brute force attack, factoring, linear and differential cryptanalysis are some of the best known attacks that are executed on the cryptographic algorithms to identify the strength of the algorithms. The number of steps performed for the attack, the time requirement and the type plaintext or ciphertext used for the attack are also the parts of this metric.

2.1 Review of Literature

The review of literature has been categorized in five parts for our research work.

Category 1: Identifying the functions used for the block ciphers

Category 2: Identifying the functions used for the stream ciphers

Category 3: Identifying the cryptanalytic attacks on ciphers that identify the functional relation in the round function.

Category 4: Identifying the cryptographic function properties.

Category 5: Identifying the modifications in Advanced Encryption Standard (AES) Algorithm.

Category 6: Identifying the modifications in Rivest Cipher 4 (RC4) Algorithm.

2.1.1 Literature review on block ciphers

We have observed 60 block ciphers. Bitwise XOR has been used in all the algorithms as it provides permutation objectives. Apart from XOR, the usage of bitwise AND, OR, NOT has also been seen. Some of the specialized functions have also been used such Fast Fourier Transformation [122], Hadamard Transformation [123], Affine Transformation [124], Self inverse Transformation [124], Linear and non linear transformations [125]. The summarized table for the utilized functions and operations of all the block ciphers has been shown in Table 2.1.

Table 2.1: *Literature Review of Block Ciphers*

Algorithm	**Ref**	**Year**	**Block Size**	**Key Size**	**Summary of used function**
Lucifer	[11]	1971	48, 32 or 128 bits	48, 64 or 128 bits	Uses the linear and nonlinear transformation functions, Arithmetic operations like AND, XOR.
DES	[12]	1975	64 bits	56 bits + 8 parity bits)	Bitwise addition modulo, XOR.
DESX	[13]	1984	64 bits	184 bits	Same as DES, but input is XORed with 64 bit key material beforehand and similarly the output is XORed with another 64 bit key part
FEAL	[14]	1987	64 bits	64 bits	XOR operations

RC2	[15]	1987	64 bits	8 - 1024 bits, in steps of 8 bits; default 64 bits	It uses two's complement addition, bitwise AND, bitwise XOR operation, bitwise COMPLEMENT, exponentiation operation and modulo operation.
Khafre	[16]	1989	64 bits	512 bits	OR and XOR operations
Khufu	[16]	1989	64 bits	512 bits	Key whitening with XOR operation.
FEALNX	[17]	1990	64 bits	128 bits	XOR operations as FEAL
LOKI	[18]	1990	64 bits	64 bits	Non linearity used in S-box and key whitening
Redoc II	[19]	1990	80- bits	160 bits	Only XOR operations
IDEA	[20]	1991	64 bits	128 bits	Key generator uses XOR operation, multiplication modulo $2^{16} + 1$ and addition modulo 2^{16}
Blowfish	[21]	1993	64 bits	32-448 bits	Modulo operations and XOR

Safer K-64	[22]	1993	64 bits	64 bits	subkeys are added using either addition modulo 256 or XOR. Pseudo hadamard transform is used here as a diffusion layer.
VINO	[23]	1993	64 bits	128 bits	In round scheme bitwise XOR is used. Addition modulo is used.
GOST	[24]	1994	64 bits	256 bits	Addition modulo and left rotation
MacGuffin	[25]	1994	64 bits	128 bits	XOR operations in different stages.
RC5	[26]	1994	32, 64 or 128 bits (64 suggested)	0 to 2040 bits (128 suggested)	Bitwise XOR is used.
TEA	[27]	1994	64 bits	128 bits	Bitwise XOR is used.
Misty	[28]	1995	64 bits	128 bits	Key Scheduling, input function, output function all of them use the bitwise AND, bitwise inclusive OR, multiplication, quotient, remainder operations

Akelarre	[29]	1996	128 bits	128 bits	Uses the basics of IDEA and RC5.
BEAR	[30]	1996	On the order of 213 to 223 bits or more	160 or 128 bits	Bitwise XOR
CAST128	[31]	1996	64 bits	40 to 128 bits	modular addition and subtraction, and XOR operations, bent function that takes several inputs and gives one output, each of which has two possible values (such as 0 and 1, or true and false)
LION	[30]	1996	On the order of 213 to 223 bits or more	160 or 128 bits	Bitwise XOR

Shark	[32]	1996	64 bits	128 bits	It is a six round Substitution Permutation-network (XOR and rotation) that uses linear and non-linear transformation layers. MDS matrix is used by the linear and the nonlinear layer is composed of eight 8×8-bit S-boxes based on the function $F(x) = x^{-1} over GF(2^8)$.
ICE	[33]	1997	64 bits	64 bits	XOR operations
Square	[34]	1997	128 bits	128 bits	Linear and nonlinear transformations
XMX	[35]	1997	Variable	Variable, equal to block size	the only operations it uses are XORs and modular multiplications
AES	[36]	1998	128 bits	128, 192 or 256 bits	XOR operation in different stages, and multiplication modulo operator.
CAST256	[37]	1998	128 bits	128, 160, 192, 224, 256 bits	Same as CAST128

CS Cipher	[38]	1998	64 bits	128 bits	Round function is based upon Fast Fourier Transformation function
Crypton	[39]	1998	128 bits	128, 192, 256 bits	In linear transformation procedure, bit permutation depends on functions that utilize bitwise AND, XOR, OR and complement too.
DEAL	[40]	1998	128 bits	128, 192, 256 bits	Uses DES as round function.
DFC	[41]	1998	128 bits	128, 192, 256 bits	It uses a round function having a single 6×32-bit S-box, as well as an affine transformation mod $2^{64}+13$.
E2	[42]	1998	128 bits	128, 192, 256 bits	It uses an input transformation and output transformation. Such transformations use modular multiplication. The round function uses XORs and S-box lookups.

Frog	[43]	1998	128 bits	128, 192, 256 bits	All operations are byte-wide and consist of XORs and substitutions.
Hasty Pudding	[44]	1998	variable	Variable	Key expansion uses multiplication, addition and shifting. It also uses bitwise XOR.
LOKI97	[45]	1998	128 bits	128, 192, 256 bits	Round function is dependable on S-boxes which are designed to be highly non-linear and use XORs.
Magenta	[46]	1998	128 bits	128, 192, 256 bits	Different function uses in the whole process of MAGENTA. The functions utilize XOR operation.

Mars	[47]	1998	128 bits	128, 192, 256 bits	Forward core layer and the backward core layer use a combination of S-box lookups, multiplications, data-dependent rotations, additions, and XORs. Addition, subtraction and XOR are used for mixing data and key values.
RC6	[48]	1998	128 bits	128, 192, or 256 bits	Addition, subtraction, XOR, multiplication
Serpent	[49]	1998	128 bits	128, 192, or 256 bits	Mixing operations, S-boxes, linear transformation.
Skipjack	[50]	1998	64 bits	80 bits	Stepping rules use bitwise XOR operation
Twofish	[51]	1998	128 bits	128, 192 or 256 bits	Uses Pseudo Hadamard Transformation
Triple-DES	[52]	1998	64 bits	168, 112 or 56 bits	Same as DES
UES	[53]	1999	128 bits	128, 192 or 256 bits	Two parallel DES.

Khazad	[54]	2000	64 bits	128 bits	Substitution-permutation network with XOR operations for permutations.
Anubis	[55]	2000	128 bits	128 to 320 bits in steps of 32 bits	It uses pseudo-random S-box depends upon XOR operations. The newer version of Anubis is called the tweaked version.
Camellia	[56]	2000	128 bits	128, 192, 256 bits	Bitwise OR, AND, XOR and complement.
DFCv2	[57]	2000	128 bits	128, 192, 256 bits	The round function uses a single 6×32-bit S-box, as well as an affine transformation mod $2^{64}+13$
Grand Cru	[58]	2000	128 bits	128 bits	Based largely on AES
Hierocrypt L1	[59]	2000	64 bits	128 bits	XOR and concatenation used for different functions like whitening, s-box, rounding function
Kasumi	[60]	2000	64 bits	128 bits	Bitwise XOR and Bitwise AND
Nimbus	[61]	2000	64 bits	128 bits	Bitwise XOR

Noekeon	[62]	2000	128 bits	128 bits	Self-inverse transformation
NUSH	[63]	2000	64, 128, or 256 bits	128, 192, or 256 bits	No S-box is used; different bitwise operations such as AND, OR, XOR, modular addition, and bit rotation are used here.
Q	[64]	2000	128 bits	128, 192 or 256 Bits	Uses S boxes that depends on XOR
SC2000	[65]	2000	128 bits	128, 192, or 256 bits	combination Substitution Permutation Network and Feistel network
SHACAL	[66]	2000	160/256 bits	128/ 512 bits	Compression function
PRESENT	[67]	2007	64 bits	80 or 128 bits	XOR operation
KATAN and KTAN-TAN	[68]	2009	32, 48, or 64-bit	80 Bits	two nonlinear Boolean functions that mainly consists of XOR operations
LED	[69]	2012	64 bits	64 /128 bits	XOR operations in different stages
Simon and Speck	[70]	2015	32, 48, 64, 96 or 128 bits	64/72/96/128/ 144/192 or 256 bits	Bitwise XOR, AND, Left Circular Shift, Right Circular Shift and modular addition

We have identified five major categories of the functions and operations used in different block ciphers and how much percentage of our observed block ciphers are using those operations have been shown in the pie-chart shown in Figure 2.1. The categories are as follows.

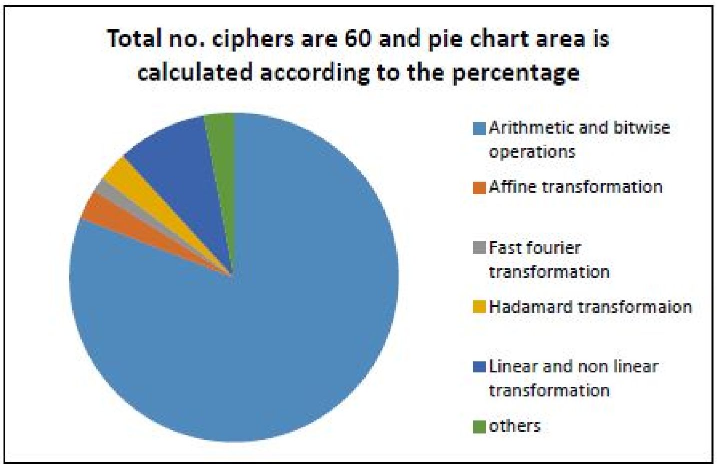


FIGURE 2.1: *Summary of work of operations in block ciphers*

Arithmetic and bitwise operations: This category includes all the bitwise operators such as AND, OR, XOR, NOT, left shift, right shift, modulo operations, arithmetic addition.

Fast Fourier Transformation: It computes the discrete Fourier Transform (DFT) of a sequence or its inverse. It factorizes the DFT matrix into a product of sparse factors which reduces the complexity from $O(N^2)$ *to* $O(NlogN)$.

Hadamard Transformation: Hadamard transformation is an example of Fourier transforms which is linear and works on 2^m real numbers. It is built out of size-2 discrete Fourier transforms, and is in fact equivalent to a multidimensional DFT of size $2 \times 2 \times ... \times 2 \times 2$. It decomposes an arbitrary input vector into a superposition of Walsh functions which take the values of +1 and -1 only with the subintervals of dyadic fraction whose denominator is a power of 2.

Affine Transformation: Affine transformations include translation, scaling, homothety, similarity transformation, reflection, rotation, shear mapping, and compositions of them in any combination and sequence. If X and Y are affine spaces, then

every affine transformation $f : X \to Y$ is of the form $x \to Mx+b_x$, where M is a linear transformation on X and b is a vector in Y. Unlike a purely linear transformation, an affine map need not preserve the zero point in a linear space. Thus, every linear transformation is affine, but not every affine transformation is linear.
Linear and non-linear transformation: Linear transformations deal with linear algebraic maths. The non-linear transformations do not rely on one-to-one linearity, different substitution equations can be made up using such transformations.
Others: In this category we have put such algorithms which do not apply any function or operation related to above categories such self-inverse and compression function.

2.1.2 Literature review on stream ciphers

We have surveyed 32 stream cipher algorithms and also identified the functions in those algorithms as shown in Table 2.2.

Table 2.2: *Literature review of Stream ciphers*

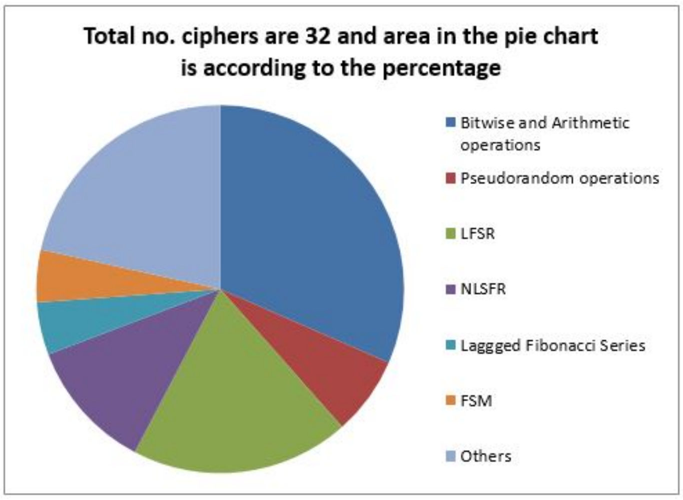
Stream Cipher Name	**Ref**	**Year**	**Functions and operations**
RC4	[71]	1987	Key scheduling and random number generator uses bitwise XOR
A5/1 and A5/2	[72]	1989	linear feedback shift registers with irregular clocking and a non-linear combiner
FISH	[73]	1993	Lagged Fibonacci generators and the shrinking generator
WAKE	[74]	1993	XOR operations for S-boxes
Pike	[75]	1994	Lagged Fibonacci generators
ISAAC	[76]	1996	Uses pseudo random number generator

SEAL	[77]	1997	Pseudo random function family
PANAMA	[78]	1998	Step right up
MUGI	[79]	1998	Linear transformation
E0	[80]	2000	XOR operators
Scream	[81]	2002	The round function is based on the AES-round function, but is narrower, 64 bits instead of 128 bits.
Rabbit	[82]	2003	The mixing function uses a g-function based on arithmetical squaring, and the ARX operations including logical XOR, bit-wise rotation with hard-wired rotation amounts, and addition modulo 232.
Snow	[83]	2003	The cipher consists of a combination of a LFSR and a Finite State Machine (FSM) where the LFSR also feeds the next state function of the FSM.
Sober 128	[84]	2003	32-bit XOR and addition modulo 232
Turing	[85]	2003	Pseudo hadamard function, Linear Feedback Shift Register (LFSR)
Trivium	[86]	2004	Primarily XOR and AND
Sosemanuk	[87]	2004	It uses a combination of a Linear Feedback Shift Register (LFSR) and a Finite State Machine (FSM) where the LFSR also feeds the next state function of the FSM.

Salsa20	[88]	2004	It is built on a pseudorandom function based on add-rotate-xor (ARX) operations comprises of 32-bit addition, bitwise addition (XOR) and rotation operations.
Py	[89]	2004	XOR, addition modulo 2^{32}
Phelix	[90]	2004	The cipher uses only the operations of addition modulo 2^{32}, exclusive or, and rotation by a fixed number of bits.
HC-256	[91]	2004	Bitwise arithmetic operations
Grain	[92]	2004	Linear Feedback Shift Register and Non Linear Feedback Shift Register
CryptMT	[93]	2005	Linear generator and filter
VEST	[94]	2005	It uses a balanced T-function that can also be described as a bijective nonlinear feedback shift register with parallel feedback (NLPFSR) or as a substitution-permutation network, which is assisted by a nonlinear RNS-based counter.
Achterbahn-128/80	[95]	2006	Nonlinear Feedback Shift Registers
Quad	[96]	2006	It works on GF ( ) functions.
WG family	[97]	2008	Primarily XOR operation, besides Linear Feedback Shift Registers and Finite Field operations

Rakaposhi	[98]	2009	Dynamic Linear Feedback Shift Register, Non Linear Feedback Shift Register, Linear Feedback Shift Register
ZUC	[99]	2010	It uses a 16-stage Linear Feedback Shift Register
MICKEY	[100]	2011	Non-linear functions
Spritz	[101]	2014	Arithmetic operations
Espresso	[102]	2015	Non Linear Feedback Shift Register, nonlinear output function

We have identified seven major categories of he functions and operations used in different stream ciphers and how much percentage of our observed stream ciphers are using those operations have been shown in the pie-chart shown in Figure 2.2. The categories that we have identified in this section are listed below.



.

FIGURE 2.2: *Summary of work of operations in stream ciphers*

Bitwise and arithmetic operations: This category includes all the bitwise operators such as AND, OR, XOR, NOT, left shift, right shift, modulo operations, arithmetic addition.
Pseudorandom operations: It uses pseudorandom number generators.

Linear Feedback Shift Registers (LFSR): It is a shift register whose input bit is controlled by the XOR operation of some previous bits' values of the overall shift register. As the input depends on the previous stage values, it works as a feedback.
Non-Linear Feedback Shift Registers (NLFSR): It is an extension of LFSR but its theory is not completed yet to support the general implementation. The period of non-linearity must be high enough.
Lagged Fibonacci functions: It is a type of pseudorandom number generator which generalizes the basic Fibonacci series concept. The operation used here can be any binary operation. The relation is given as:
$S \equiv S_{(n-j)} * S_{(n-k)}\ mod\ (m), 0 < j < k\ , where\ m\ is\ power\ of\ 2\ , generally\ 2^{32}\ or\ 2^{64}$
Finite State Machine (FSM): It is a mathematical model used for digital logic circuits in cryptography. He machine will be in only one state at a time or in a finite number of states.
Others: It includes different functions such as Galois Field functions, Hadamard functions, State-right up functions that do not imply the previous categories mentioned.

2.1.3 Literature review on cryptanalysis attacks

We have reviewed 16 cryptanalytic attacks and also identified how these attacks analyse the round functions and operations in those algorithms as shown in Table 2.3.

Table 2.3: *Literature review of cryptanalytic attacks on block ciphers and stream ciphers*

Name of the attack [Ref No.]	**Year**	**Description**	**Attacks on**
Slide Attack [103]	1977	The slide attack does not consider number of rounds in a cipher. It works by analysing the key schedule and exploiting weaknesses in it to break the cipher. This attack tries to break down a cipher into multiple rounds of an identical F function to identify cyclic key schedule. The F function must be vulnerable to a known-plaintext attack.	New Data Seal (NDS)
Differential Cryptanalysis Attack [104]	1990	It is a chosen plaintext attack. It calculates a constant difference among pairs of plaintexts; Then the differences of the corresponding ciphertexts are calculated to determine statistical patterns in their distribution.	DES
Linear Cryptanalysis Attack [105]	1992	It identifies the affine approximation	FEAL, DES
Davies' attack [106][107]	1993	It is a known-plaintext attack. It detects the non-uniform distribution of the outputs of plaintext-ciphertext pairs of adjacent S-boxes.	DES

Timing Attack [108][112]	1993	It analyses the time taken to execute cryptographic algorithms even for a single logical operation.	Diffie-Hellman, RSA, DSS
Related -key Attack [109]	1994	It can observe the operation of a cipher under several different keys whose values are initially unknown, but where some mathematical relationship connecting the keys is known to the attacker.	Kasumi, WEP
Partitioning Cryptanalysis [110][111]	1995	This attack is an extended version of linear cryptanalysis where affine transformations are replaced by balanced Boolean functions.	DES, CRYPTON
Side Channel Attack [112]	1995	This attack is based on information gained from the physical implementation of a cipher. It depends on the sound, power, time, electromagnetic fields and many more.	RSA
Integral Cryptanalysis [113]	1997	It is a chosen plaintext attack where some bits of the plaintexts are kept constant and other bits are varied. This will generate the value 0 for an XOR sum, and the XOR sums of the all the corresponding sets of cipher texts reveals the information about the cipher's operation.	SQUARE, IDEA, Camellia, Skipjack, Khazad

Interpolation Attack [114]	1997	It uses simple quadratic, or a polynomial or rational function over a Galois field to represent a S-box. Coefficients of the generated equations are determined by standard Lagrange interpolation techniques.	SNAKE, SHARK
Boomerang Attack [115]	1999	It depends upon differential cryptanalysis. The attack generates a ”quartet” structure at a point halfway through the cipher.	COCONUT98, KASUMI
Mod n cryptanalysis [116]	1999	It exploits the differences in how the cipher operates over equivalence classes modulo n. It uses Fermat number concept.	RC5
Amplified boomerang Attack [117]	2001	Same as boomerang but the selection of input output pairs need to be strict to get into the collision to analyse relation among the pairs.	MARS- 11 rounds, SERPENT- 8 rounds
Rectangle Attack [118]	2001	Same base as boomerang but with the modifications of more number of quartets, sorting of piles of wrong beta values to execute the attack to get the quartet values and the gamma dash values instead of gamma for all possible differential characteristics.	SERPENT

XSL Attack [119]	2002	It generates quadratic simultaneous equation system and solves the equations with extended sparse linearization.	SERPENT, AES
Rotational Attack [120][121]	2010	It depends on ARX operations: Addition, rotation, modulo and XOR.	Threefish

We have identified four major categories of the functions and operations where different cryptanalytic attacks have been executed. In the Figure 2.3, we have shown a pie-chart depicting how much percentage of our observed cryptanalytic attacks is executed on a particular category of operations.

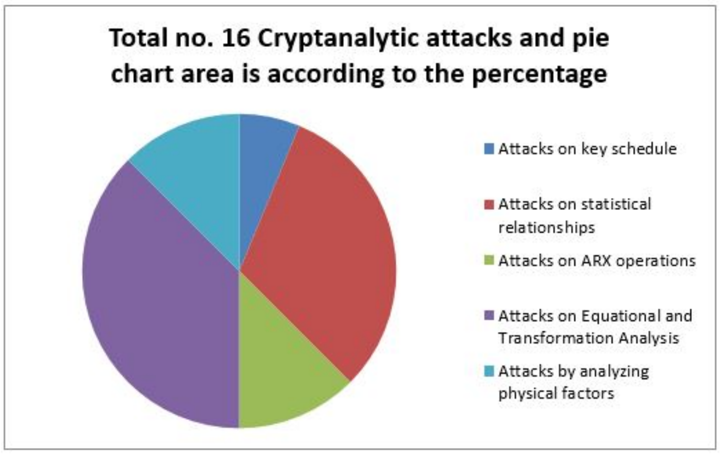
Attacks on key schedule: These attacks identify the key entirely or partially depending upon the internal values of the operations and functions of plaintext ciphertext combinations.

Attacks on statistical relation between plaintext and ciphertext: These attacks identify the statistical relationship between plaintext and ciphertext by utilizing the round functions as a whole or partially.

ARX operations: These attacks work on addition, rotation and XOR operations in the functions of ciphers.

Equation and transformation analysis: The attacks on this category deals with analysing different linear and non-linear equations to solve the seed values to identify the plaintext or keys according to the possibility of occurrences.

Analysing physical factors: In this category, the attacks deal with different physical factors such as sound, electromagnetism, power level evaluation while executing a particular function.

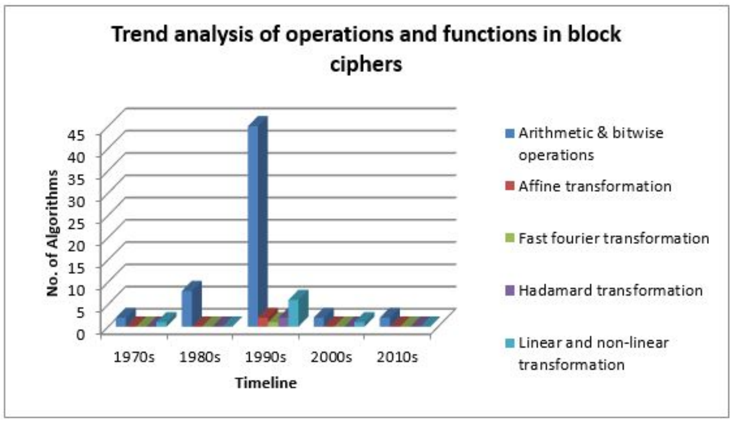


.

Figure 2.3: *Summary of existing attack on different operation categories*

2.1.4 Timeline Analysis

The trend of the operations and functions in the algorithms and the trend of the attacks on the algorithms have also been analysed according to the timeline. We have categorized the timeline into 1970s, 1980s, 1990s, 2000s and 2010s. The trends are shown in Figure 2.4, Figure 2.5 and Figure 2.6 respectively for block ciphers, stream ciphers and attacks.



.

Figure 2.4: *Trend Analysis of Block cipher operations and functions*

The trend in Figure 2.4 shows that use of arithmetic and bitwise operations was increasing exponentially till 2000s but has got a slow down further. But as compared to other operations and function types, these are used more. Moreover, the three transformation functions such as Affine, Hadamard and Fast Fourier are only used

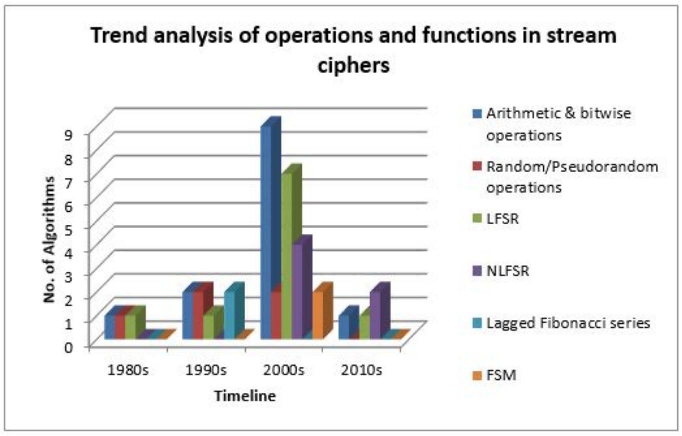


FIGURE 2.5: *Trend Analysis of Stream cipher operations and functions*

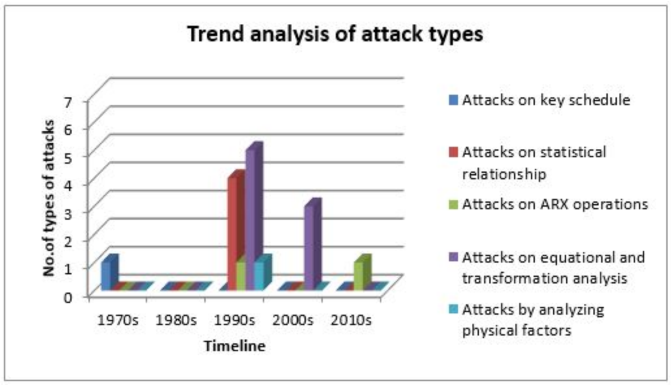


FIGURE 2.6: *Trend Analysis of types of cryptanalytic attacks*

in 1990s. Similarly for stream ciphers, as shown in Figure 2.5, the use of arithmetic and bitwise operations, pseudorandom operations and linear shift back registers are significantly high. Figure 2.6 shows the trend of the attacks. According to the analysed trend, attacks on the bitwise and arithmetic operations are still in process whereas attacks on statistical relationships between plaintext-ciphertext or plaintext-key or ciphertext-key and attacks on equational and transformation analysis is decreasing.

2.1.5 Review of cryptographic function properties

Cryptographic algorithms are dependable on a number of cryptographic functions [162] and transformations. Different types of factors are considered for designing such functions [168].For example, resiliency [169] and non-linearity [170] of the functions receives a major concern. Bruer [171] has suggested considering the same importance for all the inputs so that the properties of the cryptographic functions are evaluated significantly. The highest probable non-linearity factor is achieved for quadratic functions as shown in [163][164]. The existence of correlation-immune and resilient symmetric functions has been investigated in [172][173][174]. Randomness in the bits of the input as well as output is also necessary in the algorithms of cryptography. These algorithms deal with pseudorandom number generators. Recent research dealing with quantum based and automata based pseudorandom generators have been shown in [175][176][177]. Chaotic cryptography is also been improved by random and pseudo-random sequences as shown in [178][179]. Chaos based random number generator is also used for S-box applications and cryptographic operations as shown in [180]. A new linearization method has been observed in the work [183]. The proposed method works for nonlinear feedback shift registers used for stream ciphers. The authors introduce a novel state transition matrix for an NFSR, which is computed from the truth table of its feedback function. Hamming weight is an important metric for the cryptographic functions. This metric is researched in [184] and a number of properties have been identified. These properties are helpful to analyze a novel design for cryptographic function. In this paper, A novel technique for constructing balanced Boolean functions on even numbers of variables has been shown in [185]. The proposed technique uses a set of disjoint spectra functions and a special Boolean permutation to construct a balanced Boolean function with high nonlinearity and optimal algebraic degree. Another method for balanced Boolean functions construction has been shown in [186]. The proposed approach uses even number of variables as previous research work with a bound that the even number is greater than 10. It also satisfies the strict avalanche criterion, and has a high algebraic degree. Following the same line of research, another method is proposed recently in [187] to construct resilient Boolean functions on even number of variables satisfying strict avalanche criterion with nonlinearity. Another two construction methods for balanced boolean functions are provided in [188]. It achieves high nonlinearity and satisfies strict avalanche criterion. Both local and global avalanche characteristics property are followed by the proposed methods. The algebraic immunity of the constructed functions is also considered by the authors. The work described in [189] discusses about the perturbation effect on the symmetric boolean functions. Precisely, the work presented, establishes a relation between exponential sums of these perturbations and Diophantine equations of a particular form. In paper [7], the authors have discussed the propagation properties of boolean functions. New properties are also identified which have the high strict avalanche criterion. The research work in [8] analyses the inverse permutation on the field of order 2n with component functions. The weights of derivatives of the component functions also use Kloosterman sums. The authors in the paper [9] show the aperiodic autocorrelation for a boolean function and also have defined the Aperiodic Propagation Criteria (APC). The authors also make a comparison between their proposed criteria and the extended propagation criteria as defined in [7]. The work also shows the relation between aperiodic autocorrelation and the first derivative of the function. In the paper [6], the authors distinguish among three different objective functions: algebraic constructions, random generation and heuristic constructions. The authors also have analysed important cryptographic properties of Boolean functions, and have examined four evolutionary algorithms: Genetic Algorithms (GAs), Genetic Programming (GP), Cartesian Genetic Programming (CGP) and Evolution Strategies (ESs).

The concept of correlation immune Boolean functions was introduced by Siegenthaler [10]. Further, this feature of boolean functions has been considered as an important characteristics for strong cryptographic algorithms and has been researched with concern as shown in [11][12][13][14]. Some of the contemporary research has been done to provide immunity to the functions both as correlation aspect and algebraic aspect. High algebraic immunity is a desirable factor for cryptographic algorithms to prevent algebraic attacks. In the paper [15], some lower bounds on the algebraic immunity of Boolean functions have been calculated. The results are applied to give lower bounds on the algebraic immunity of symmetric Boolean functions and rotation symmetric Boolean functions. The relation of algebraic immunity and extended algebraic immunity is researched in [16]. The effect of algebraic immunity on even variable rotation symmetric boolean functions has been analysed in [17]. The construction of an infinite class of functions based upon bent functions has been shown in the paper [18]. The authors have considered only odd number of variables. Also an innovative recursive decomposition of the first-order correlation-immune Boolean functions has been shown. Based on this work, the same authors have presented the design of an enumerative encoding of these Boolean functions in [19]. This is the first enumerative encoding of a class of Boolean functions defined by a cryptographic property. In the paper [20] , optimal algebraic immunity construction strategy has been shown using vector spaces and m-sequences. The authors in [21] present two hybrid classes of boolean functions. The functions constructed within these classes possess maximal algebraic degree for balanced functions, optimal algebraic immunity, high non-linearity and good resistance against algebraic attacks. The authors have also proposed hybrid class of 1-resilient functions which have the similar features. Algebraic immunity on balanced functions has been shown in the papers [22][23]. Vector valued functions over finite field has also experimented for maximum algebraic immunity in [24].

2.1.6 Literature review on AES modification

One of the most popular and commercialized algorithm is AES. This algorithm provides the encryption for web security processes as used by different applications such as e-commerce, router applications, wi-fi security. Being so rigorously used in real life applications, AES faces a number of attacks. Some of the recent attacks are mentioned below.
A new kind of fault-based attacks called fault sensitivity analysis (FSA) has been proposed in [126]. The authors have used the zero valued sensitivity model for masked AES. Combining the FSA and zero valued sensitivity, the proposed method of cryptanalysis is able to break code of the S-boxes in masked AES. The attack procedure shows that the zero value input of S-box reveals the key eventually. The authors in the paper [127] have shown a differential faulty approach used in the mix column component of AES. The results show that AES-128 is breakable by such process only using two faulty input of ciphertexts. This attack has been proved better as compared to other differential attacks on AES as shown in [128][129][130]. Another improved version of faulty attack on AES has been executed in the paper [131]. The authors show that a single random byte fault at the input of the eighth round of the AES algorithm is sufficient to deduce the block cipher key. Simulations show that when two faulty ciphertexts pairs are generated, the key can be exactly deduced without any brute-force search. The minimal fault against AES has been used in [132]. The authors show that AES-192 is breakable by using two pairs of correct and fault ciphertexts whereas AES-256 is broken by using three pairs of correct and fault ciphertexts. The work shown previously in [131] was having a key space of 2^{32} which has been reduced by the authors in [133]. Key recovery attacks on AES has been described in [134]. In this paper, the authors have shown practical complexity based attacks against AES-256. The use of two related keys and 2^{39} time complexity has been proved to be sufficient to recover the complete 256-bit key of a 9-round version of AES-256. Another attack works on 10 round version of AES-256 in 2^{45} time complexity. An improved version of

the previous related key attack has been shown in [135] against round transformation and key expansion module in AES. The round has been now minimized from 9^{th} to 7^{th} which means that AES is vulnerable even for the starting rounds. The complexity of the attack has also been reduced from 2^{192} to 2^{104}. Another voltage based fault induction method has been introduced in [136]. The authors show a fault model for a constantly underfed RISC CPU. The faults are described in terms of position, recurring patterns and timing, then the corresponding errors induced in the computation outcomes are specified. The model also support multi-bit patterns. The use of biased faults also provides an efficient way to for fault injection attacks in cryptanalysis. Such a procedure has been shown in [137].

A collision based attack against AES-192/256 has been shown in [138]. The authors have used 4-round distinguisher for 7-round reduced AES. In the paper [139], the authors have used variable key for AES sing pseudorandom number generator for providing better security to the algorithm, but the approach faces the problem of using biased keys against AES rounds. Biased keys are able to reveal the pseudorandomness of the approach and the key is deduced further by applying differential methods or fault injection as shown before. Multiple deductions-based algebraic trace driven cache attacks on AES has been shown in [140]. The behaviour of the cache reveals the input whole or partially. Same input to a particular module and the changes of the cache properties are the key features of this approach. The authors have identified the causes of a bias fault and also have compared different biased fault attacks introduced till. Quantum related key attacks has been shown in [141].

A solution to the fault based injection attacks has been provided in [142]. The proposed scheme is independent of S-box and inverse S-box and achieves more than 95% fault coverage. A recent approach against fault injection or fault analysis has been shown in [143].It combines the principles of redundancy with that of fault space transformation to achieve security against both DFA and DFIA based attacks on AES-like block ciphers.

After surveying the attacks on AES, it is obvious that fault injection attacks are more efficient in revealing the key in AES. Such fault injections are using the biased input too to distinguish the subkeys or other parts of the algorithm. Moreover, as AES is depending upon finite field operations of 8 bit bytes, the attacks are also executable with finite quantified complexities as we have seen above. The biased inputs along with fault bytes creates error in the process and those are denoted for performing differential analysis or linear analysis. Eventually, the key is revealed. Therefore, to overcome such problems, we have introduced the randomness and the balanced symmetric feature in the functional output, specifically in the keys. We have named this modified AES as Random Key AES (RK-AES). As a result, even though attackers are deducing a part of key or injecting a biased fault, the fault will be converted to a symmetric output rather than revealing the original key or plaintext.

2.1.7 Literature review on RC4 modification

One of the most popular and simplest stream cipher is RC4. This algorithm has been used enormously in different network protocols such as SSL, WEP, TLS, WAP and also has been used by a number of top IT companies such as Microsoft, Apple, Nokia and many more. Being so rigorously used in real life applications, RC4 always has been faced cryptanalysis attacks for a long time due to its several drawbacks [144]. Therefore, the popularity has been decreased tremendously for this algorithm. Some of such attacks are mentioned below.
In reference [145], a related key cryptanalysis on RC4 has been shown by the authors. The authors show the existence of a family of related keys for each 2048-bit key. It differs in one of the byte positions. The key-streams generated by RC4 for a key and its related keys are substantially similar. Therefore, it is very easy to use statistical analysis or linear and differential cryptanalysis to achieve the secret key. The statistical analysis of RC4 key stream generator is also shown in [146]. The authors claim that their process uses only $2^{30.6}$ bytes of outputs and also analyses

the distinguisher of 8-bit randomness. A large number of weak keys for RC4 have been identified by the authors in [147]. Weak keys are used to generate the distinguisher for RC4 and to execute related key attack on the cipher. It is also shown that the proposed passive ciphertext attack procedure can break any arbitrary long key in practical time complexities. The authors also analyses the Fortuitous states in RC4. The authors in reference [148] show a statistical bias in the distribution of the first two output bytes of the RC4 keystream generator. The work in the paper shows that only 2^{25} bytes of outputs are required to effectively distinguish RC4 outputs from random strings.

The work presented in reference [149] describes a cryptanalytic attack that uses the tree representation of RC4 cipher and introduces an abstraction in the form of general conditions for managing the information about its internal state. Hill-climbing strategy is followed to find the initial state. The complexity of this attack is lower than that of an exhaustive search. This attack is derived from a general cryptanalytic approach for a class of table-shuffling ciphers, whose next-state function permutes the table entries. In the reference [150], the authors present a general framework for differential cryptanalysis on RC4.

The differences in the key or in the key stream patterns are used to analyse the internal state of the cipher and retrieve it. The authors in the reference [151] analyses the permutation operations which are considered to be non-linear. The theoretical analysis of the work shows that permutation bytes in any stage in key scheduling algorithm of RC4 are biased and reveals the secret key eventually. Colliding key problem in RC4 has been described in the paper [152]. Such keys generate same initial state and hence generate the same pseudo random byte stream. The authors present a new state transition sequence of the key scheduling algorithm for a related key pair of an arbitrary fixed length that can lead to key collisions. Another key collision work on RC4 has been researched in [153]. Attack on a RC4 (n,m) has been shown in [154]. The algorithm is based upon RC4. The authors show two attacks: one is based on non-randomness of internal state. This allows to distinguish it from a truly random cipher. Another attack is depending upon low diffusion of bits in the key scheduling of RSA and PRGA algorithms and recovers all bytes of the secret key. Empirical correlations among the key stream bytes and the secret key have been studied thoroughly in [155]. It shows that the non-randomness behaviour of RC4 works as a bias and exhibits such relations which are used for cryptanalysis attacks.

As we have said earlier, to improve the loopholes of the ciphers, the modification on the ciphers is an evolving process. A number of improvements and modifications have been suggested by different researchers till date. Some of the proposed modifications are cited here. An improved RC4 has been shown in [156]. It uses a new pseudo random bit generator with two secret keys and three pointers. The key size condition for robustness in RC4 has been analysed and shown in [157]. An analysis of modified RC4 following the same usage of two keys has been shown in [158]. Three enhanced variants of RC4 have been proposed in [159]. The authors have concentrated on the pseudorandom number generation rather than the key scheduling. A hybrid chaotic based approach has been used in [160] for improving the strength of RC4. In the reference [161], a new algorithm is proposed by using initial state factorial to solve the correlation issue of RC4. Correlation between public known outputs of the internal state is removed by using an additional state table with the same length as that of the state to contain the factorial of initial state elements.

The analysis of the previous work signifies that RC4 suffers from cryptanalysis attacks due to its key stream biasness primarily. Moreover, the collision of keys and the distinguisher generation also create major drawback in his cipher.

2.2 Open Research Problems

After organizing the literature review and executing the analysis of the trend, as shown in the previous section, some open research problems have been identified. These research problems can help the upcoming researchers in the field of cryptography. The cryptologists can work further on these given aspects to enrich this crypto world with some valuable research work for future. Some of the future research problems are given below.

- New ciphers using different equational or transformations such as Affine, Hadamard, Fast Fourier, Linear-nonlinear need to be explored as attacks are significantly high on such category of operations in ciphers.
- The performance of Fibonacci series or pseudorandom operations in block ciphers can be researched as it is only used in stream ciphers till date.
- New stream ciphers can be designed with more sophisticated pseudorandom or random operations as less work have been executed in this domain.
- The effect of side channel attack on all types of cryptographic algorithms using various factor need to be analysed further.

2.3 Conclusion

We have observed 60 block ciphers, 32 stream ciphers and 16 cryptographic attacks. The survey identifies which domains of the mathematical functions are primarily dominant in cryptographic procedures. The block and stream ciphers are primarily using the arithmetical and bitwise operations. From the literature review, it is also observed that maximum cryptanalytic attacks have been executed on statistical relationship between ciphertext and plaintext. From the attackers view point, these statistical relationships are inferred from the functional operations in the algorithms stepwise or even from implications of the identical operations. The open research problems identified in the above section will be helpful for further research in the domain of cryptography using different functions and operations in the algorithms. Furthermore, we have also analysed some of the modifications done in AES 256 and RC4 algorithm in recent times. This analysis has helped us to develop our proposed system.

CHAPTER 3

Symmetric Random Function Generator (SRFG): A Novel Cryptographic Approach

3.1 Introduction

Cryptographic algorithms [162] depend on the internal structure of the algorithms and their corresponding effect of the boolean functions used in the cryptographic functions [163]. Along with basic gates such as AND, OR, NOT, XOR used in the algorithms, researchers also have shown to have a specialized Boolean function that exhibits the symmetric property. The generic Boolean functions create the basic platform of generating any cryptographic algorithm. However, the technology progress of the attackers has urged a need of introducing randomness in the function generators. This would help against the attacks which evaluate the pattern of Boolean functions in an algorithm to get into the identification process

of the key bits or plaintext bits by reverse engineering [164]. In this chapter, therefore, we have shown such a random function generator that will provide the output as a combination of basic logic gates. The unique feature of this function generator is that it provides the output in symmetric way i.e. the number of 1's and 0's in the output is equal.
Symmetric Boolean functions [168] are distinguished by their outputs that only depend on the Hamming weights of their inputs. This category of functions is represented in a very compact way both for their algebraic normal forms and for their value vectors. The two important cryptographic parameters: algebraic degree and the nonlinearity, cannot be simultaneously optimized for symmetric functions. The researchers have proved in [169] and [170] that the highest possible nonlinearity for a symmetric function is only achieved by quadratic functions. As in comparison, symmetric functions with suboptimal nonlinearity exist and create an interest for designing fast and robust cryptographic primitives. Besides the Hamming distance to linear functions, some other criteria, such as correlation immunity or propagation characteristics, are also required in some applications and need to be addressed in the context of symmetric functions.

3.2 Symmetric Random Function Generator (SRFG)

The aim of this present work is to introduce a function generator that produces the symmetric balanced output in the sense of the number of 1s and 0s in the output string irrespective of the input string. The concept of the present work must not be confused with the existing symmetric Boolean function. Both the concepts are distinguished by a line of difference: Symmetric Boolean functions consider the hamming weight of the input string, whereas our work considers the hamming distance of output string. Moreover, symmetric boolean function is a specialized function whereas, our work outputs a combined function comprised of basic boolean functions as shown in Figure 3.1. This would help the hash algorithms, stream ciphers and round function module of block ciphers to be more robust. The expression for the combined function generator can be given as:

$$f_c = \otimes f_i^L \tag{3.1}$$

where, $i = 1, 2, ...4$ four logic GATES : AND, OR, NOT and XOR; L represents the expression length (Number of terms in the combined function f_c) and $\otimes$ represents the random combination.

The process followed for this has been summarized below.

Step 1: Initializing variables. This step consist conversion of numbers which are greater than B bit into arrays of B bit integers.

Step 2: Generating Random expressions population according to expression length and calculating result by its expression. *Step 3:* Then calculating fitness function of the result R. Fitness value (F) = count(No of Zeroes present in R)

Step 4: If fitness value (F) is equal to $\frac{B}{2}$ then we get our result and we stop our process . For example, if $B = 64$ bit and we get $F = 32$ then we get our result as result has number of zeroes = number of ones = 32

Step 5: If fitness value F belongs to $[\frac{B}{2} - \frac{B}{8}, \frac{B}{2} + \frac{B}{8}]$ interval than we perform some mutation by changing one operation or variable and generate all the expressions from it and repeating step 3 to 5. for example if B = 64 and F lies in [24, 40] then we perform mutation. *Step 6:* After generating all the mutations if we are not getting solution then we repeat step from 1 to 6

To emphasize the randomness [171] in such combined function generator, the equation 3.1 can be further expressed in terms of N input variables' randomness in selection, as shown in equation 3.2.

$$f_c(V_1, V_2...., V_N) = \otimes f_i^L \; [rand(V_1, V_2..., V_N)] \tag{3.2}$$

The structure shown in the Figure 3.1 can be expanded as the function generator can be used for any N variables of n bits each as shown in Figure 3.2. Let F_2 is the

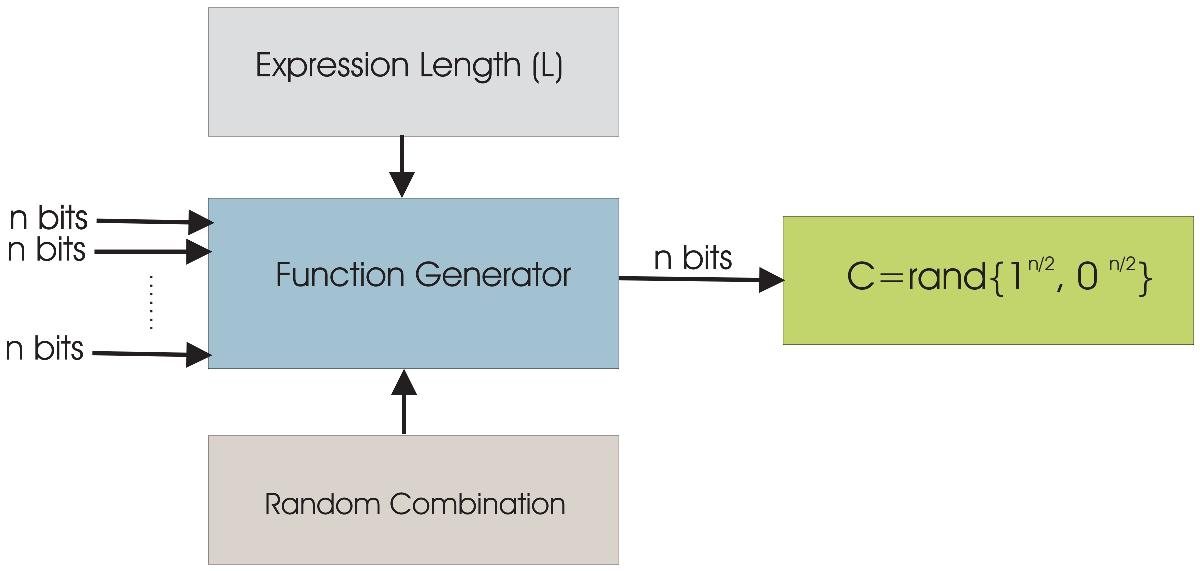


FIGURE 3.1: *Structure of the SRFG*

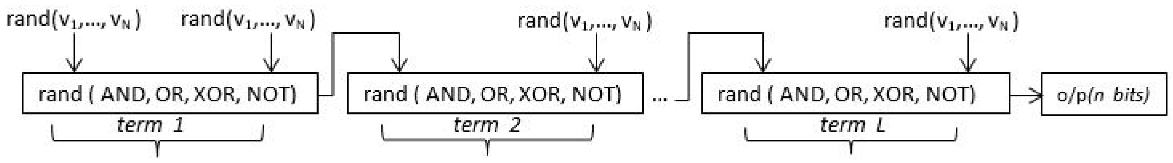


FIGURE 3.2: *Chaining of randomization.*

finite field of two elements 0,1 and $\oplus$ is any operation of the field F_2. N variables are used and each variable is considered to be a n bit vector $v = v_1, v_2, ..., v_n$. In the further discussion each variable V_i is considered as a binary vector v.

The Hamming Weight of such a binary vector v is given by:

$$wt(v) = \sum_{i=1}^{n} v_i \tag{3.3}$$

Proposition 1: A combined function f_c using n bit variables and generating an output v_o of n bits is symmetric and balanced iff $\sum_{i=1}^{n} v_{o_i} = \frac{n}{2}$ where, $v_{o_i} \in 0, 1$.

For any number of variables of n bits, the bit patterns follow a bell-curve of normal distribution and the standard deviation from the mean is very low. Therefore, just analysing $\frac{n}{2}$ combinations of the bit patterns the original bit information can be achieved. Bitsum attack [172] shows this effect. Therefore, if we are able to construct to randomize the operations in the function level, each iteration provides the same bitsums and therefore hard to detect the function patterns.

Proposition 2: Let n be an odd integer and v_o is the output of the combined function f_c, f_c is the random symmetric function iff 0 is added in the least significant bit (LSB) of the variable.

$$\sum_{i=1}^{n} v_{o_i} = \frac{n}{2}, \;\forall\; n \; is \; even \tag{3.4}$$

and,

$$\sum_{i=1}^{n} v_{o_i} + LSB(0) = \frac{n}{2}, \;\forall\; n \; is \; odd \tag{3.5}$$

Let B_N the set of all symmetric random combined boolean functions of N variables of all the functions from F_2^N into F_2 where $F_2^N = \{(V_1, V_2, ..., V_N) | V_i \in F_2\}$.
Any combined function $f_c \in B_N$ of L terms is expressed as a polynomial [173] which is basically termed as Algebraic Normal Form (ANF) of the function and given as:

$$f_c(V_1, V_2, ..., V_N) = \oplus\, \lambda_u \left(\prod_{i=1}^{N} rand(V_i)^{u_i} \right)^{L}, \; \lambda_u \in F_2, \; u \in F_2^N \; and \; L \in \mathbb{Z} \tag{3.6}$$

with,

$$\lambda_u = \oplus\, f_c(v), \; v \preceq u \;, \;\forall\; V_i = \{v_{i_1}, v_{i_2}, ..., v_{i_n}\} \tag{3.7}$$

where,

$$(v_{i_1}, v_{i_2}, ..., v_{i_n}) \preceq (u_1, u_2, ..., u_n) \iff \forall i, j, v_{i_j} \leq u_i \tag{3.8}$$

For any random combined function $f_c \in B_N$ generated by SRFG is expressed by a binary bit vector of length 2^N. This vector is comprised of all the values $f_c(y)$, $y \in F_2^N$.

The degree of function f_c is denoted by $deg(f_c)$ and computed as the maximal value of the weights of the output vector v_{o_i} given as:

$$wt(v_{o_i}) = \frac{n}{2}\ , \ such\ that\ \lambda_u \neq 0 \tag{3.9}$$

Any function $e \in F_2^N$, φ_e denotes the linear function in $B_N : x \longmapsto e.x$, where (.) is the dot operation of two vectors. The sequence of Walsh functions $W_k : [0,1] \rightarrow \{-1,1\}$, $k \in N$ is defined as below[174].

$$W_k(f_c) = (-1)^{\sum_{j=0}^{\infty} k_j v_{j+1}} \tag{3.10}$$

Following the equation 3.10, the Walsh coefficient of $f_c \in B_N$ in point $e \in F_2^N$ is given as

$$W_e(f_c\ +\ \varphi_e) = (-1)^{\sum_{j=0}^{\infty} v_{j+1}\ +\ e.x} \tag{3.11}$$

The series of values for the Walsh coefficients $W_e(f_c + \varphi_e), e \in F_2^N$ forms the Walsh spectrum S of the function f_c. The Walsh spectrum S of an N-variable combinational logic function actually represents the function itself.

$$\forall a \in F_2^N,\ W(f_c\ +\ \varphi_e) = \sum_{w=0}^{N} (-1)^{v_w} P_w(wt(a)) \tag{3.12}$$

Where , P_w is the Krawtchouk polynomial [175] of degree w given as:

$$P_w(i) = \sum_{k=0}^{w} \begin{pmatrix} i \\ k \end{pmatrix} \begin{pmatrix} N-i \\ w-k \end{pmatrix} (-1)^k \tag{3.13}$$

The equation 3.12 and equation 3.13 represent the symmetric feature of the Walsh transformations of f_c. The output of f_c depends on the weight of its input vectors. As a result, f_c corresponds to a function $g_c : 0,1,,n \rightarrow F_2$ such that $\forall x \in F_2^N,\ f_c(x) = g_c(wt(x))$. The sequence $g_c(f_c) = (g_c(0), g_c(1)..., g_c(n))$ for n-bit vector is considered as simplified value vector of f_c. To establish the relation between simplified value vector and arithmetic normal form the equation 3.6 can be rewritten as shown in equation 3.14.

$$f_c(V_1, V_2, ..., V_N) = \oplus\lambda_f\ (j) \oplus \Big(\prod_{i=1}^{N} rand(V_i)^{u_i}\Big)^L \tag{3.14}$$

$$= \oplus\lambda_f\ (j)\mathscr{X}_{j,N} \tag{3.15}$$

where, $\lambda_f(j), u \in F_2^N$ and $L \in \mathbb{Z}$, $j = 1,..,N$. $\mathscr{X}_{j,N}$ is the elementary polynomial [173] of degree j with N variables. The coefficients of arithmetic normal form of f_c is represented by n-bit vector, $\lambda(f_c) = \lambda_f(0), \lambda_f(1), ..., \lambda_f(N)$, called as simplified vector of ANF of f_c.

Following the equation 3.15, for a given x of weight , $\mathscr{X}_{j,N}$ contains $\begin{pmatrix} j \\ k \end{pmatrix}$ nonzero monomials. The expression of binomial coefficients modulo a prime number $\wp$ is given by Lucas Theorem [176]. Given two integers, r and s and their p-adic representations, $r = \sum_{i=0}^{k} r_i \wp^i$ and , $s = \sum_{i=0}^{k} s_i \wp^i$, by Lucas theorem, we have the following.

$$\begin{pmatrix} r \\ s \end{pmatrix} \prod_{i=0}^{k} \begin{pmatrix} r_i \\ s_i \end{pmatrix} mod\ \wp \tag{3.16}$$

For $p = 2$, we get :$\begin{pmatrix} r \\ s \end{pmatrix} \equiv 1\ mod\ 2$. iff $supp(s) \subseteq supp(r)\ s \preceq r$ which means that $\forall i, s_i \le r_i$. We then finally get the following values for g_c and λ_f as below.

$$g_c(i) = \oplus \begin{pmatrix} i \\ k \end{pmatrix} for, k = 0, 1, ..., i \tag{3.17}$$

$$\lambda_f\ (k) = \oplus\ \lambda_f(k),\ k \le i \tag{3.18}$$

3.3 Properties of SRFG

3.3.1 Nonlinearity

Nonlinearity is an important design characteristic for cryptographic functions used in cryptographic algorithms to prevent different types of correlation or linear attacks. In our combined function generator, this feature actually depends on the output vectors in v_{o_i} . v_{o_i} is also be considered as the affine transformations of the functions generated from this SRFG. The nonlinearity is calculated by the hamming distance between two affine transformation. For example, two output vectors are: v_o and u_o of 8 bits each.

According to the above example, the hamming distance between the above two

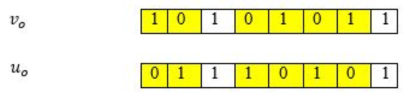


FIGURE 3.3: *Hamming distance of output vectors.*

vectors = 6. Therefore, the non linearity of the function is 6. To generalize the concept of the nonlinearity of such a function f_c is given as:

$$\mathcal{N}l_{f_c} = \sum_{i,j=1}^{n} v_{o_i} \neq u_{o_j},\ v_{o_i}\ and\ u_{o_j}\ are\ output\ vectors \tag{3.19a}$$

Moreover, following equation 3.11, the nonlinearity of SRFG is also related with the Walsh transformation as:

$$\mathcal{N}l_{f_c} = 2^{n-1} - \frac{1}{2} max|W_{f_c}(e)| \tag{3.19b}$$

3.3.2 Balancedness

Balanced property of the SRFG generated combined function f_c exists if its simplified value vector g_c follows the following condition

$$\forall\ 0 \leq i \leq N, g_c(i) = g_c(N-i)\ \boxplus 1, where\ \boxplus\ is\ sum\ over\ F_2 \tag{3.20}$$

Trivial balanced functions exactly correspond to symmetric functions. Therefore, the combined function f_c verifies the condition $D_1 f_c$=1 signifying the all-one vector. Functions with $D_1 f_c$=1 do not really exist for even values of n because for any vector v such that $wt(v) = \frac{n}{2}$, we can calculate the $D_1 f_c$ as:

$$D_1 f_c = f_c(v) \boxplus f_c(v+1) = g_c\left(\frac{n}{2}\right) \boxplus\ g_c\left(\frac{n}{2}\right) = 0 \tag{3.21}$$

The above trivial balanced function property also corresponds to the odd case of trivial partitioning [177] whereas the even cases represent the affine functions [178]. Finding the patterns of the simplified value vector g_c of the combined function f_c leads to the set of the n binomial coefficients for balanced symmetric function.

3.3.3 Resiliency

Resiliency of cryptographic functions is another important characteristic for designing robust and efficient algorithms in cryptography. The correlation between the output of the generated function and a small subset of its input variables leads to the correlation attack [179][180]. A given symmetric balanced combined function f_c of N variables is m-resilient if it remains balanced when any m input variables are fixed and remaining (n-m) bits are altered. There is no upper or lower bound on the order of resiliency of such functions; the function is more resilient if m is higher. The property of resiliency is related to the weights of the restrictions

of the f_c to some subspaces.
$\forall f_c \in B_N$ and any affine subspace $\mathscr{S} \subset F_2^N$, the restriction of f_c to $\mathscr{S}$ is the function given as:

$$f_{\mathscr{S}} : \mathscr{S} \rightarrow F_2 \tag{3.22}$$

$$x \rightarrow f_c(x), \forall x \in \mathscr{S} \tag{3.23}$$

where, $f_{\mathscr{S}}$ can be determined by the with a function of $dim(\mathscr{S})$ variables.
The subspace $\mathscr{S}$ is spanned by k canonical basis vectors and its supplementary subspace is $\mathscr{S}$. The restrictions of f_c to $\mathscr{S}$ and to all its cosets are given by $a + \mathscr{S}$ where, $a \in \bar{\mathscr{S}}$. When f_c is symmetric and balanced, $\mathscr{S}$ is represented as: $\mathscr{S} = s_1, s_2, ..., s_k$ and $f_{a+\mathscr{S}}$ becomes symmetric and balanced too. Moreover, for all $s \in \mathscr{S}$, we can write the following.

$$f_{a+\mathscr{S}}(s) = f(a+s) = g_c\Big(wt(a) + wt(s)\Big) \tag{3.24}$$

which actually depends upon the weight of s when a is fixed. The simplified value vector and the simplified ANF vector of $f_{a+\mathscr{S}}$ can be deduced from f_c as given below.

$$g_{c_{f_{a+\mathscr{S}}}}(i) = g_c(i + wt(a)), \forall\ i, 0 \leq i \leq k \tag{3.25}$$

$$\lambda_{f_{a+\mathscr{S}}}(i) = \oplus\ \lambda_f(i+j),\ \forall i, 0 \leq i \leq k\ and\ j \leq wt(a) \tag{3.26}$$

3.3.4 Propagation criterion

Propagation characteristics are determined by the cryptographic properties of the derivatives of the functions generated by SRFG. All derivatives of the combined functions that output from the SRFG are linearly equivalent when they have a fixed Hamming weight of $\frac{n}{2}$. Our function f_c of N variables generated from the random function generator satisfies the propagation criterion of degree k and order m if any affine function obtained from f_c by keeping m input bits constant satisfies the propagation criterion of degree k.

Let $f_c \in B_N$ and let $x, y \in F_2^N$ be such that $wt(x) = wt(y) = \frac{n}{2}$. Then, $D_x f_c$ and $D_y f_c$ are linearly equivalent. This signifies a linear permutation μ of F_2^N such that $D_x f_c = D_y f_c \circ \mu$, where $\circ$ is composite function. The permutation μ exists on $1, 2, .., N$ such that $y = \mu(x)$. Since, f_c is symmetric and balanced, we can have,

$$\begin{aligned} D_y f_c(\mu(a)) &= f_c(\mu(a)) \boxplus f_c(\mu(a) + y) \\ &= f_c(a) \boxplus f_c(\mu(a) + \mu(x)) \\ &= f_c(a) \boxplus f_c(\mu(a + x)) \\ &= f_c(a) \boxplus f_c(a + x) \\ &= D_x f_c(a) \end{aligned} \tag{3.27}$$

Where $\boxplus$ denotes addition over F_2. Let k be an integer, $1 \leq k \leq n-1$, $\bar{V}_i = (v_{i_1}, v_{i_2}, .., v_{i_{n-k}})$ and $\epsilon_k = v_{n-k+1} + .. + v_n$. Then the restriction of $D_{\epsilon_k} f_c$ to all affine subspaces $y + V, y \in (v_{i_1}, v_{i_2}, .., v_{i_{n-k}})$ is given by:

$$g_y V \to F_2 \tag{3.28}$$

$$x \mapsto D_{\epsilon_k} f_c(a + y) \tag{3.29}$$

Are also symmetric functions of $(n-k)$ variables and weight is $\frac{n}{2}$. Moreover, their simplified value vectors and ANF vectors are given for all $0 \leq i \leq n-k$ by

$$g_{c_{g_y}}(i) = g_c(i + wt(y)) \boxplus g_c(i + k - wt(y)) \tag{3.30}$$

$$\lambda_{g_y}(i) = \otimes_{j \leq k - wt(y)} \otimes_{j \leq wt(y)} \lambda_f(i + j) \boxplus \otimes \lambda_f(i + j) \tag{3.31}$$

Let $y \in \bar{V}_i$, then for any $z = a + y$ with $a \in \bar{V}_i$, then we can have the following.

$$wt(z) = wt(a) + wt(y) \tag{3.32}$$

$$wt(z + \epsilon_k) = wt(a) + wt(y + \epsilon_k) = wt(a) + k - wt(y) \tag{3.33}$$

Thus, $\forall\ a \in V$,

$$\begin{aligned} D_{\epsilon_k} f_c(a+y) &= f_c(a+b) \boxplus f_c(a+\epsilon_k+y) \\ &= wt(a) + wt(y+\epsilon_k) \\ &= wt(a) + k - wt(y) \\ &= g_c(wt(a)+w(y)) \boxplus (wt(a)+k-w(y)) \end{aligned} \tag{3.34}$$

Equation 3.34 signifies $g_{c_{g_y}}$ also follows the symmetric property which means that partial derivatives of function generator outputs are also propagated with the propagation features. To calculate simplified ANF vector of $g_{c_{g_y}}$, we have decomposed the ANF of f_c as given below.

$$f_c(a+b) = \otimes_{u \in F_2^{n-k}} \otimes_{v \in F_2^k} \lambda_{u,v} \prod_{i=1}^{n-k} a_i^{u_i} \prod_{j=n-k+1}^{n-k} b_j^{v_j}, \forall (a,b) \in (V, \bar{V}) \tag{3.35}$$

Then, for any $y \in \bar{V}$ and aV , we have,

$$\begin{aligned} g_{c_{g_y}}(a) &= D_{\epsilon_k} f_c(a+y) \\ = \otimes_{u \in F_2^{n-k}} \otimes_{v \in F_2^k} \lambda_{u,v} \prod_{i=1}^{n-k} a_i^{u_i} &\times \prod_{i=1}^{k} (y_i \boxplus 1)^{v_i} \boxplus \prod_{i=1}^{k} y_i^{v_i} \end{aligned} \tag{3.36}$$

3.3.5 Immunity feature

Now, lets discuss about the correlation immunity characteristics exhibit by our function generator's outputs. Considering each of the N input variable V_i as n bit binary vector $V_i = \{v_1, v_2, ., v_n\}, i = 1, 2, .., N$ for the combined symmetric and balanced function f_c is correlation immune if :

$$Prob(f_c = v_i) = \frac{1}{2}, 1 \leq i \leq n \tag{3.37}$$

The set of all correlation immune functions of N variables of n bits each is denoted by CI_N^n. Further, the weight of correlation immune function f_c of our function generator is given by:

$$a = wt(f_c) = \frac{n}{2}, f_c \in CI_N^n \tag{3.38}$$

Let, f_c^u and f_c^l be the upper and lower halves of equal length $\frac{n}{2}$ for our function generator's outputs as shown in Figure 3.3. As, $f_c \in CI_N^n$, $wt(f_c^u) = wt(f_c^l) = \frac{n}{4}$. If there are k places out of $\frac{n}{4}$ 1s in the f_c^u where the corresponding positions in reverse of f_c^l, denoted as $(f_c^l)^r$, do not match. Therefore, the number of bits match with respect to 1 for the function is given as:

$$M_1(f_c^u, (f_c^l)^r) = \frac{n}{4} - k \tag{3.39}$$

Following the equation 3.39, we can also say that there are k places out of $\frac{n}{4}$ 1s in the f_c^u where the corresponding positions in $(f_c^l)^r$ do not match. Therefore, the number of bits match with respect to 0 for the function is given as:

$$M_0(f_c^u, (f_c^l)^r) = \frac{n}{4} - k \tag{3.40}$$

Hence, we can write:

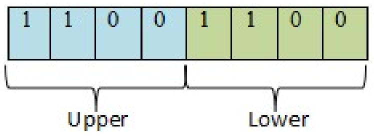


Figure 3.4: *Concept of lower and upper halves of bitstring*

$$\begin{aligned} M(f_c, (f_c)^r) &= 2M(f_c^u, (f_c^l)^r) \\ &= 2\left(\frac{n}{4} - k + \frac{n}{4} - k\right) = n - 4k \end{aligned} \tag{3.41}$$

The correlation immune functions with the same values of $M(f_c, (f_c)^r)$ form an equivalence class.

Proposition 3: Let $f_c \in CI_N^n$ and $M(f_c, (f_c)^r) = m$. Then $M_0(f_c, (f_c)^r)$ and $M_1(f_c, (f_c)^r)$,

$$min[M_0(f_c, (f_c)^r) - M_1(f_c, (f_c)^r)] = min[m] \mapsto 0 \tag{3.42}$$

Apart from the correlation immunity, algebraic immunity is also a requirement for cryptographic algorithms. Algebraic immunity also is related with the annihilator of a function [183]. This property helps the algorithm to prevent algebraic attacks [184][185] against the cryptographic algorithms.
Given, $f_c \in B_N$, any function of the set $A(f_c) = \{g \in B_N | gf = 0\}$ is defined as the annihilator of the function f_c. The algebraic immunity of $f_(c)$ is denoted by $AI(f_c)$ is minimum degree of all nonzero annihilators of f_c or $f_c + 1$. The value of $AI(f_c)$ is given as:

$$AI(f_c) = min[deg(g)|g \neq 0, g \in A(f_c) \cup A(f_c + 1)] \tag{3.43}$$

As the SRFG outputs the symmetric function which are balanced and therefore minimum degree is always $\frac{n}{2}$. Therefore, the algebraic immunity of the outputs from SRFG is always $\frac{n}{2}$ which is always optimal.

3.4 Results

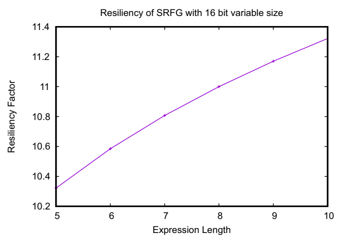
We have executed an experiment for our model based upon the properties as discussed in the section 3.3. To evaluate these properties we have implemented the model with respect to two metrics: number of bits (n), length of expressions (L). The parameters and metrics used for the experiment is summarized in Table 3.1. The symmetric function is balanced as no. of 1s and 0s is equal in every possible

Table 3.1: *Metric set-up*

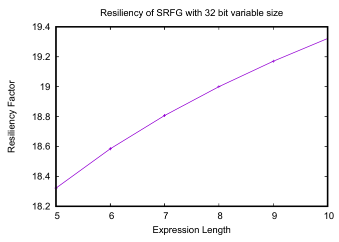
Metrics	Values
No. of variables (N)	2 to 5
No. of bits (n)	16,32, 64, 128, 256, 512, 1024 and 2048
No. of Expressions	5 to 10
Boolean functions	AND, XOR, NOT, OR
Sample size	300
Sample technique	Random

combination of the variable size and expression length. Therefore, the balance factor is always n/2 where n is the size of each variable. The results of other two experiments for resiliency and non-linearity have been shown in Figure 3.4 and Figure 3.5 respectively.

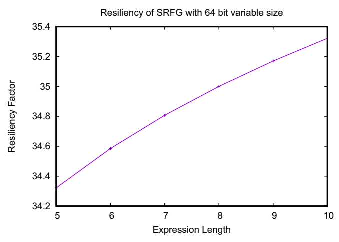
The results describe the fact that, the expression length provides a significant effect



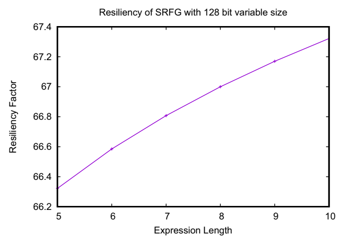
(a)



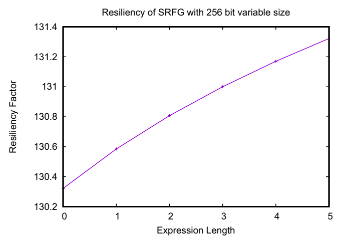
(b)



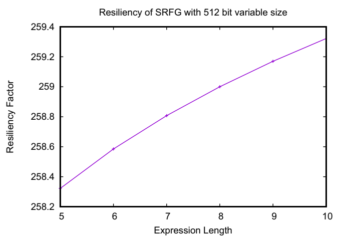
(c)



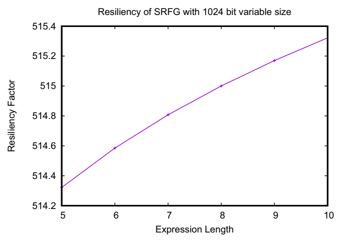
(d)



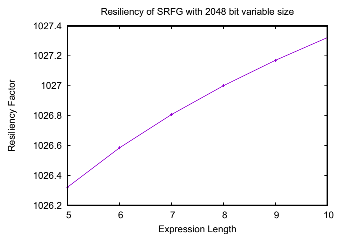
(e)



(f)



(g)

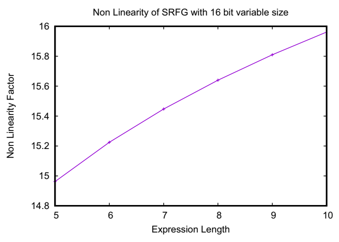


(h)

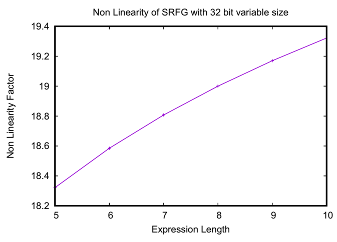
FIGURE 3.5: *Effect of Expression Length on Resiliency.*

on the resiliency as well as non-linearity factor which are the important factors in the cryptographic functions. In Figure 3.5, we have shown the result of resiliency of our function generators outputs varying the expression length from 5 to 10 with variable size from 32, 64, 128, 256, 512, 1024 and 2048 bits. Similarly, in Figure 3.6, we have plotted the result of our experimentation for non-linearity by varying the expression length for each variable size same as considered for resiliency.

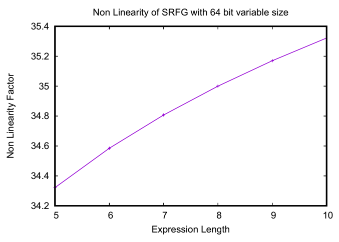
Following the results of our experiments we have calculated the order of resiliency as: $\frac{n}{2} + log_2 L$ where L is expression length of the combined function f_c and n is the size of each variables and output in bits. We have also calculated the order of



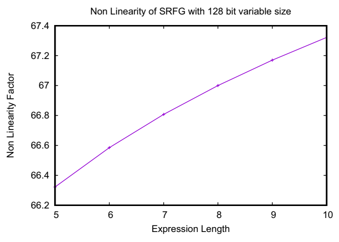
(a)



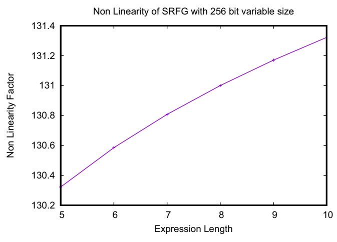
(b)



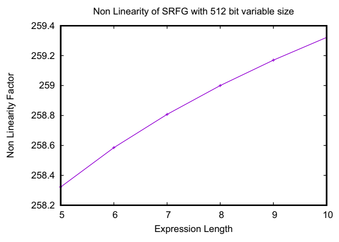
(c)



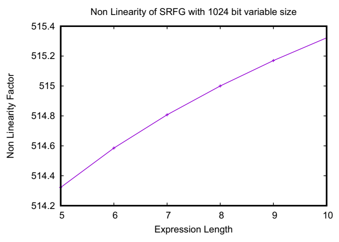
(d)



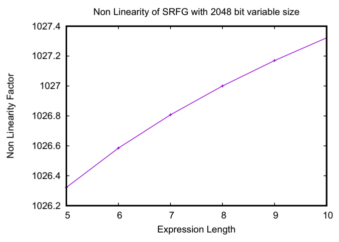
(e)



(f)



(g)



(h)

FIGURE 3.6: *Effect of Expression Length on Non Linearity.*

nonlinearity Nl_{f_c} as:

$$= n * 0.79 + log_2 L \tag{3.44}$$

L is expression length of the combined function f_c and n is the size of each vari-

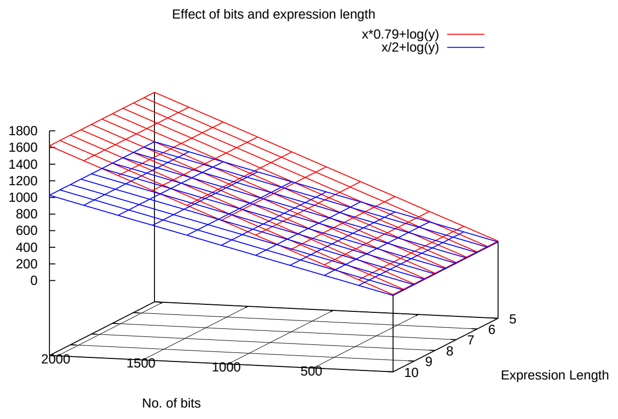
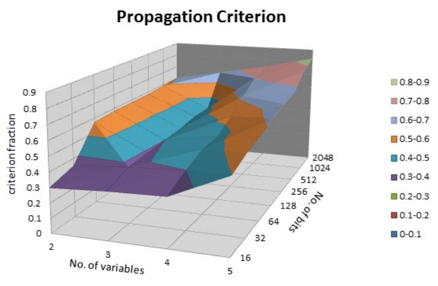
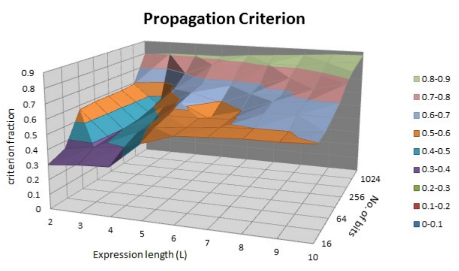


FIGURE 3.7: *Hamming distance of output vectors.*



(a)



(b)

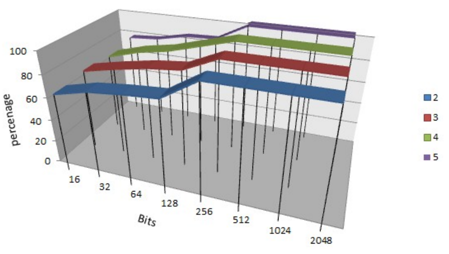
FIGURE 3.8: *Propagation criterion*

ables and output in bits. This result shows that our function generator outputs combined function which is having good nonlinearity and therefore eligible for cryptographic functions. Figure 3.7, shows the overall effect of variable size (in bits) and expression length on non-linearity and resiliency. The expression length is not having any effect on the average outputs of the functions shown in [181] and [182]. The order has been calculated based on the average results of all the outputs. The Table 3.2 shown above compares the parameters of balancedness, nonlinearity and resiliency among our research work, work shown in [181] and work shown in [182]. The results signifies that randomness in the functions gives better results for all the parameters. Moreover, the expression length is not having any effect on the algorithms shown in [181] and [182]. Therefore, our present work is a better candidate for being a cryptographic primitive to be used.

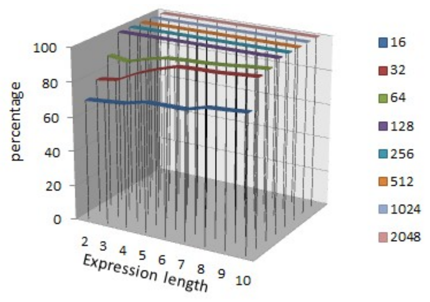
The evaluation for the propagation criterion has been measured with our introduced metric of criterion fraction. Criterion fraction is defined as: $\frac{m}{k}$, where m is the number of bits in the input variable kept constant to get the propagation criterion of degree k. For our developed function generator as the output functions are balanced always, the value of $k = \frac{n}{2}$ and maximum number of bits we can keep constant is also $\frac{n}{2}$. These values give the upper bound of criterion fraction as 1.

TABLE 3.2: *Comparison of Cryptographic Features*

	Balanced outputs (%)	Non-linearity property	Resiliency property	Order of Non-linearity	Order of Resiliency
Proposed Work	100 %	83.72 %	56.5 %	n × 0.79 + log_2 L	$\frac{n}{2}$ + log_2 L
Zhang et. al. [181]	47%	50.5%	27%	$\frac{n}{2} + 0.2$	$\frac{log_2\ n}{2}$
Li et. al. [182]	53%	46.8%	30.4%	$\frac{n}{2}$	$log_2 n$



(a)



(b)

Figure 3.9: *Correlation Immunity*

In Figure 3.8, we have shown the results of the propagation criterion for our developed system and is generated functions. Figure 3.8(a) shows the result by varying the number of variables and number of bits of each variable. Similarly, Figure 3.8(b) shows the result by varying the number of expression length and number of bits of each variable. In both the cases, the results achieves more than 80% propagation criterion effect which is nearly the strictly avalanche effect.

The correlation immunity characteristics have been plotted in Figure 3.9. Figure 3.9(a) shows the result by varying the bits and number of variables from 2 to 5. Figure 3.9(b) shows the result by varying the bits from 16 to 2048 and expression length from 2 to 10. We have noticed a new feature in both the cases that irrespective of the expression length and number of variables, our SRFG is capable of producing 100% correlation immune functions with and after 128 bits. This makes the SRFG efficient and suitable for cryptographic algorithms. The algebraic immunity of the SRFG generated functions are always at $\frac{n}{2}$ irrespective of the number of bits and number of expression length and therefore these parameters is not shown graphically explicitly.
The comparison results shown in Table 3.3 notify the order of the parameters considered for the evaluation.

Table 3.3: *Comparision of Propagation criterion and immunity*

	Propagation criterion	Correlation immunity	Algebraic immunity
Proposed SRFG	$\frac{n}{2}$	All the outputs of SRFG i.e 100%	$\frac{n}{2}$
Wang et. al. [186]	$\frac{n}{2}-1$	57.63%	0

3.5 Statistical tests for validation of randomness

The validation of the randomness has been executed by following the test suite suggested by the NIST in the NIST 800-r1 document [194]. A number of statistical tests have been performed on the outputs of the SRFG. To perform the tests the hypothesis has been set up for all the tests as:

H_0(null hypothesis): The outputs of the SRFG is random.

H_1(alternate hypothesis): The outputs of the SRFG is non-random.

We have summarized the p-values of the tests in the following Table 3.4. We have used 1% significance level in all the statistical tests performed. In all the above tests, the p-values of the test is greater than the 0.01, the significance level of the tests. Therefore, the null hypothesis is accepted. This signifies that the SRFG has passed all the tests and the outputs of SRFG contain randomness in true significance.

Table 3.4: *Tests for validation of Randomness*

	Monobit test	Frequency test within a block	Runs test	Test for longest run in the block	Binary matrix rank test
SRFG	1.00	1.00	0.723	0.1933	0.5320
	Spectral test	Non overlapping template matching test	Overlapping template matching test	Maurers test	Linear complexity test
	0.300	0.300	0.280	0.777	NA
	Serial test	Approximate entropy test	Cumulative sum test	Random excursions test	Random excursions variant test
	Not performed	0.2770	0.433	0.777	0.777

3.6 Conclusion

Research has been expanded in the domain of the cryptographic functions to identify different valuable properties for symmetric balanced functions. But randomness is missing in all those aspects. In our present work, therefore, we have analysed this effect of randomness for our symmetric random function generator outputs. We have discussed about the general properties of a symmetric functions. The experimented results confirms the fact that our developed function generator outputs the combined functions which are balanced always and provide more than 80% of non-linearity in average and more than 50% of resiliency factor. The results also show that the function generator outputs the combined functions which provide more than 85% of propagation in average and near about 100% correlation immunity. Moreover, all the functions generated from the SRFG are algebraic immune under $\frac{n}{2}$ where n is the number of bits in the output variable. Therefore, the function generator discussed in this chapter is well suited for cryptographic functions and can be applied for any cryptographic algorithm for more robustness. Algorithms such as DES, AES and other block ciphers use fixed round functions where we can use this SRFG for better avalanche effect. Moreover, stream ciphers which use pseudo random number generators are now vulnerable to attack. Therefore, use of SRFG in stream ciphers would also make the ciphers more robust. Key generators are another application area of our function generator.

CHAPTER 4

RK-AES: A Modified version of AES with SRFG

4.1 Introduction

Cryptology is an important domain of security measure for providing confidentiality, authentication and other services [162]. It contains two major parts as: cryptography and cryptanalysis. With the progress of technology, where the new cryptographic algorithms are emerging, the cryptanalysis processes are also getting improved; to countermeasure those more secure algorithms are getting developed. So, the cyclic process of cryptography and cryptanalysis goes on. Moreover, the trend of converging to IoT exhibits an urge of improving the cryptographic algorithms for applications to be secure [187][188]. Cryptographic algorithms are broadly categorized in two ways: a) block ciphers and stream ciphers depending upon the format of the message processing; b) symmetric and asymmetric depending upon number of keys used for the algorithms [162]. Designing such algorithms is another concern where a number of principles are needed to be maintained such as: key size, message size, number of rounds, round function and so on. The selection of key and its size is a major concerning factor in cryptography. A weak key can reveal the plaintext message with least time. Though we know that, cryptographic algorithms face brute-force attacks problems, but brute-force is not considered as its complexity is higher than any other process of cryptanalysis. The objective of a third party attacker is to break the ciphertext code or to reveal the key or part of the key to get access of the plaintext. So, the weak keys must be avoided in the algorithms. Further, it may happen that the previously considered strong key is now made weak by the sophisticated technology or large computational abilities. So, the need of strength analysis to withstand with attacks makes the evolving changes in the cryptographic algorithms.

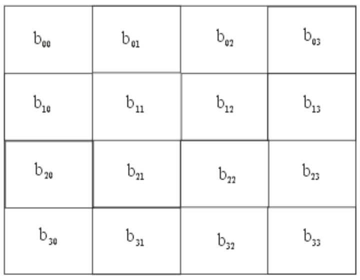
Cryptographic algorithms primarily depend on the structure of the algorithms and their corresponding functions [163]. Apart from using logic gates such as AND, OR, NOT, XOR in the algorithms, researchers also have shown some specialized Boolean functions for the symmetric property. The generic Boolean functions have created the basic functionalities of generating any cryptographic function. However, the technology progress and enhancing computational ability of the attackers has urged a need of introducing new features in the function generators so that they can provide more strength to the ciphers. Balancedness, non-linearity, resiliency, immunity, correlation and propagation characteristics are some of the important parameters to evaluate the strength of the ciphers. In this chapter, we have considered Advanced Encryption Standard (AES) for our experimentation of randomness feature. We have attributed the key generation module of AES undergoing through our Symmetric Random Function generator (SRFG) [189]. We have evaluated the modified AES with the parameters said above.

4.2 AES algorithm

The Advanced Encryption Standard (AES) [36] was published by the National Institute of Standards and Technology (NIST) in 2001. AES is a symmetric block cipher where a single key is used for both encryption and decryption process. The input and output for the AES algorithm each consist of sequences of 128 bits. The key used in this algorithm consists of 128, 192 or 256 bits. AES operates on 8-bit bytes. These bytes are interpreted as the elements of finite field using the following polynomial representation:

$$f(x) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + ... + b_1x + b_0 = \sum_{i=0}^{n-1} b_i x^i \tag{4.1}$$

where each b_i is having the value of 0 or 1.



b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	b_{33}

FIGURE 4.1: *State matrix representation*

The 128 bit input block of AES is arranged in a state matrix of size 4 ×4 as shown in Figure 4.1. The elements of the matrix are represented by the variable b_{ij} where $0 \leq i, j \leq 3$ and i,j denotes the row and column number respectively. Depending upon the size of the bits in keys variables rounds are allowed for AES. For our experimentation we have used key size of 256 bits concept and therefore, the number of rounds used is 14 rounds represented as Nr. Key scheduling algorithm is also used in AES to provide keys to each of the rounds. The design of the key scheduling algorithm is such that the revealing any round key deduces the original input key from which the round keys are derived. The input state matrix is processed by the various round transforms. The state matrix evolves as it passes through the various steps of the cipher and finally produces the ciphertext. Each round in AES follows the following steps.

SubBytes: This is a nonlinear step in the AES. It uses an S-box applied to the bytes of the state matrix. Each byte of the state matrix is replaced by its multiplicative inverse, followed by an affine mapping as follows.

$$\begin{aligned} b_i^{'} = b_i \oplus b_{(i+4)\ mod\ 8} \oplus b_{(i+5)\ mod\ 8} \oplus b_{(i+6)\ mod\ 8} \\ \oplus\, b_{(i+7)\ mod\ 8} \oplus\ c_i\ \ , for\ 0 \le i < 8 \end{aligned} \tag{4.2}$$

where, b_i is the i^{th} bit of the byte, and c_i is the i^{th} bit of a byte c with the value 63 or 01100011. Thus the input byte x is related to the output y of the S-Box by the relation, $y = A.x^{-1} + B$, where A and B are constant matrices [36].
Shift Rows: The last three rows of the state matrix is rotated by a certain number of byte positions. It is executed as follows.

$$s_{r,c}^{'} = s_{(r,(c+shift\ (r+Nb))mod\ Nb)}, for\ 0 < r < 4\ and\ 0 < c < Nb \tag{4.3}$$

where, Nb is the number of words in the state matrix (each column in the state matrix is considered as word). In AES, $Nb = 4$ always as the input size is 128 bits and arranged in state matrix of size 4 ×4. Each cell in the state matrix is denoted as s with the index of row r and column c.
MixColumns: This transformation operates on the state matrix column-by-column, considering each column as a four-term polynomials over GF (2^8) and multiplied modulo x^{4+1} with a fixed polynomial a(x) , given by:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x^1 + \{02\} \tag{4.4}$$

The multiplication process with the columns of state matrix is given by:

$$s'(x) = a(x) \otimes s(x) \tag{4.5}$$

where,⊗ multiplication modulo of polynomials and s(x) is a state in the state matrix.

AddRoundKey: In this process, a round key is added to the state by a simple bitwise XOR operation. Each round key is having the size of Nb words from the key schedule. Those Nb words are each added into the columns of the state matrix to satisfy the following condition.

$$[s'_{0,c}, s'_{1,c}, s'_{2,c}, s'_{3,c}] = [s_{0,c}, s_{1,c}, s_{2,c}, s_{3,c}] \oplus [w_{round \times Nb+c}], \quad for\ 0 \le c < Nb \tag{4.6}$$

Where, ⊕ is the bitwise XOR and round is the round number at which round key is added and $0 \le round \ < \ Nr$.

All these steps are performed for each of the round in the AES excluding the last round. In the last round the MixColumn step is not performed. For a 14-round AES, the round function process is shown in Figure 4.2. One of the important part

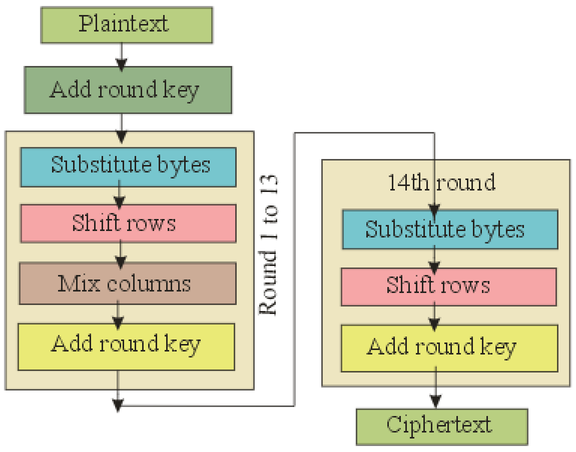


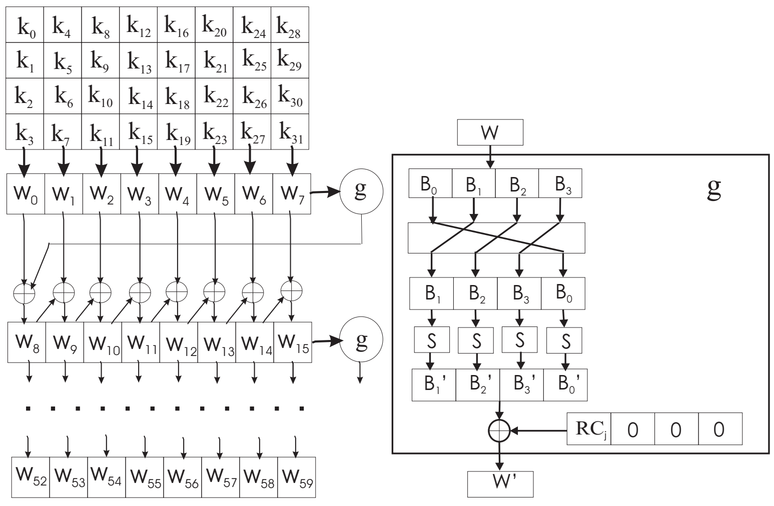
FIGURE 4.2: *Round Function steps in 14-round AES*

of the round function stages is adding of round keys as these keys are generated by the key expansion routine. The Key Expansion generates a total of $Nb(Nr+1)$ words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted by $[w_i\ ], 0 \le i \le Nb(Nr+1)$. It uses a function SubWord() that takes these 4-byte words as input and applies S-box to each of these words.

Another function Rotword() is used to perform a circular permutation. The round constant array Rcon[i] contains the values specified as [x^{i-1},{00},{00},{00}] with x^{i-1} powers of x in the equation 4.7.

$$Rcon[i] = x^{(i-4)/4} \; mod \; (x^8 + x^4 + x^3 + x + 1) \tag{4.7}$$

where, i is the current round.
The key expansion routine for 256-bit keys ($Nk = 8$) is slightly different than for 128- and 192-bit keys. If $Nk = 8$ and i-4 is a multiple of Nk, then SubWord ( ) is applied to w[i-1] prior to the XOR. Nk is the number of 32-bit words of a key.


k0 k4 k8 k12 k16 k20 k24 k28
k1 k5 k9 k13 k17 k21 k25 k29
k2 k6 k10 k14 k18 k22 k26 k30
k3 k7 k11 k15 k19 k23 k27 k31
W0 W1 W2 W3 W4 W5 W6 W7
g
W8 W9 W10 W11 W12 W13 W14 W15
W52 W53 W54 W55 W56 W57 W58 W59
W
B0 B1 B2 B3
B1 B2 B3 B0
S S S S
B1' B2' B3' B0'
RCj 0 0 0
W'

(a) word generation (b) g function

Figure 4.3: *Key Expansion for 14-round AES*

4.3 RK-AES

The main problem in the key expansion of the AES algorithm is that the words w_i generated from the original key are related with each other. If any word is traceable, the overall key is deduced by the differential method or liner methods of cryptanalysis. Though the XOR operation, S-boxes and the shifting in g

function, shown in Figure 4.3, is providing the confusion characteristics to the algorithm, but the reverse engineering process can easily get back to the original key space. Moreover, the biased inputs in the key space reveal the differences between the words to partially gain the key space. To solve this problem in AES, we have modified the key expansion module of AES with Symmetric Random Function Generator (SRFG) [189]. SRFG produces the symmetric balanced output in the sense of the number of 1's and 0's in the output string irrespective of the input string. It outputs a combined function comprised of basic GATEs (AND, OR, NOT and XOR). The expression for the combined function generator is given as:

$$f_c = \otimes\, f_i^L \tag{4.8}$$

where, i=1,2,..4 four logic GATES : AND, OR, NOT and XOR; L represents the expression length (Number of terms in the combined function f_c) and $\otimes$ represents the random combination. In our experiments we have used L=5. To emphasize the randomness in such combined function generator, the equation 4.8 can be further expressed in terms of N input variables' randomness in selection, as shown in equation 4.2.

$$f_c(V_1, V_2 ...., V_N) = \otimes f_i^L\ [rand(V_1, V_2 ..., V_N)] \tag{4.9}$$

For, our experimentation, the equation 4.9 is rewritten as:

$$f_c(V_1, V_2) = \otimes f_i^5\ [rand(V_1, V_2)] \tag{4.10}$$

The main objective of adding SRFG in AES is to enable the key expansion module with some randomness feature. This will help to prevent deducing the words of keys even though partial key is in hand. The modified key expansion module has been shown in Figure 4.4, the changes are highlighted in yellow colour. The randomness of SRFG has been used in three parts. First, in the function of g; secondly, the recursive word generation from key spaces, and thirdly but most prominently, addition of RC_i and SRFG for generating the words from w_0 to w_7. According to the Figure 4.4(a), each column in the key space is considered as w_i word. As the

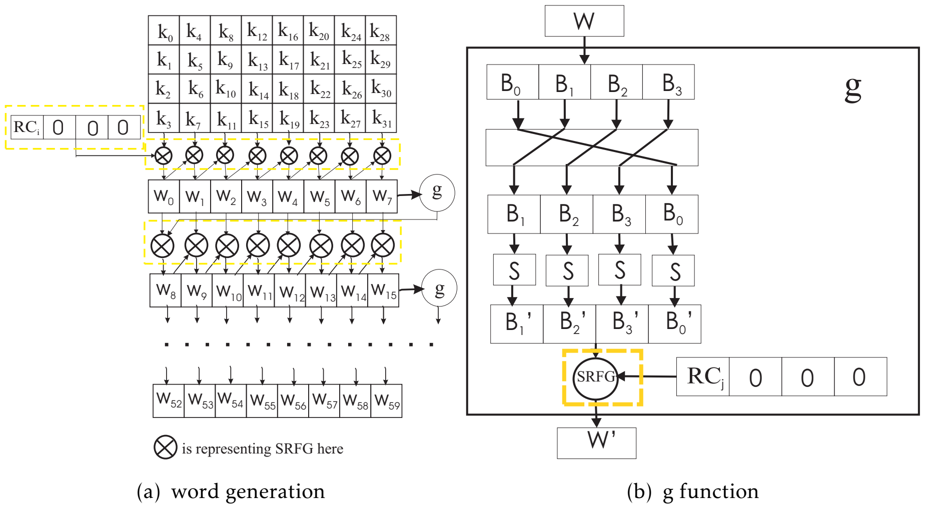


FIGURE 4.4: *Modified key expansion for 14 round AES*

key size is 256 bits, we shall have eight words $w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7$ in the very first step. The 8^{th} word i.e. w_7 is going through a function g. This function is also using SRFG just before the output of the function as in Figure 4.4(b). The output of g is then used to generate the other words processing through a series of SRFGs. The same process is repeated till we get the required number of words for the 14 rounds in AES. The modified key expansion has been shown in Algorithm 1.

For the decryption process, we have saved the generated words and used them reversely with the cipher text to get back to the plaintext. In future, we shall work upon direct transmission of the keys rather than storing them for decryption.

4.4 Feature analysis of RK-AES

We have emphasized on the key generation module of AES-14 round, so that the effect of biased inputs in the key bytes can be removed from deducing the overall key bytes. The keys are deducing if the cryptanalysis process is able to infer a linear or differential equation out of the words generated from the key expansion module. For the cryptanalysis process, it is not always necessary to have the whole

Algorithm 1 Algorithm for RK-AES Key Expansion

```
1: procedure KeyExpansion (byte, key[4*Nk], word w[Nb*(Nr+1)], Nk)
2:     word temp
3:     $w[0] = word(RC[\frac{i}{Nk}], 0, 0, 0) \otimes (word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]))$
4:     $i = 1$
5:     while $i < Nk$ do
6:         $w[i] = w[i-1] \otimes (word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]))$
7:         $i = i+1$
8:     end while
9:     $i = Nk$
10:    while $(i < Nb * (Nr+1))$ do
11:        $temp = w[i-1]$
12:        if $(i \ mod \ Nk = 0)$ then
13:            $temp = SubWord(RotWord(temp) \otimes (RC[\frac{i}{Nk}], 0, 0, 0))$
14:            else if$(Nk > 6 \ and \ i \ mod \ Nk = 4)$
15:            $temp = SubWord(temp)$
16:            end else if
17:        end if
18:        $w[i] = w[i-Nk] \otimes temp$
19:        $i = i+1$
20:    end while
21: end procedure
```

key in hand; rather a single part of key if in the capture, the relationship between different words is sufficient in revealing the overall key space. With the progress of cryptanalysis technologies, generating such relations or deducing keys from sub keys is getting faster with less complexity as we have seen in the literature review. We have identified some of the parameters for modified key-expansion module for RK-AES such as: non-linearity, balancedness, resiliency, propagation criterion and immunity. Each word w_i in the key space comprised of 32 bits (4 bytes) which is considered as 32 bit word vector in our experimentation.

Let $\mathcal{B}_2$ the set of all symmetric random combined functions on two variables of all the functions from F_2^2 into F_2 where $F_2^2 = (w_1, w_2)|w_i \in F_2$. F_2 is the finite field of two elements 0,1 and $\oplus$ is any operation of the field F_2.

Any combined function $f_c \in \mathcal{B}_2$ of five terms is expressed as a polynomial which is basically termed as Algebraic Normal Form (ANF) of the function and given as:

$$f_c(w_1, w_2) = \oplus\ \lambda_u \left(\prod_{i=1}^{2} rand(w_i)^{u_i} \right)^s,\ \lambda_u \in F_2,\ u \in F_2^2\ and\ L \in \mathbb{Z} \tag{4.11}$$

$$\lambda_u = \oplus f_c(v), w \leq u, \forall\ w_i = w_{i_1}, w_{i_2}, ..., w_{i_{32}} \tag{4.12}$$

where,

$$(w_{i_1}, w_{i_2}, .., w_{i_{32}}) \leq (u_1, u_2, .., u_{32})\ iff\ \forall i, j, w_{i_j} \leq u_i\ and\ j = 1, 2, .., 32 \tag{4.13}$$

The output of f_c depends on the weight of its input variables (number of 1's in the variable). As a result, f_c corresponds to a function $g_c : \{0,, 1, ..., 32\} \rightarrow F_2$ such that $\forall x \in F_2^2, f_c(x) = g_c(wt(x))$. The sequence $g_c(f_c) = (g_c(0), g_c(1)..., g_c(32))$ for 32-bit word vector is considered as simplified value vector of f_c. To establish the relation between simplified value vector and arithmetic normal form, the equation 4.11 can be rewritten as shown in equation 4.14.

$$f_c(w_1, w_2) = \oplus\ \lambda_f(j) \oplus \left(\prod_{i=1}^{2} rand(w_i)^{u_i} \right)^L \tag{4.14}$$

$$= \oplus \lambda_f\ (j)\ \mathscr{X}_{j,N} \tag{4.15}$$

where, $\lambda_f(j)$,$u \in F_2^2$ and $L \in \mathbb{Z}, j = \{1, 2\}$. $\mathscr{X}_{j,N}$ is the elementary polynomial of degree j with 2 variables. The coefficients of arithmetic normal form of f_c is represented by 32-bit vector, $\lambda(f_c) = \{\lambda_f(0), \lambda_f(1), ..., \lambda_f(32)\}$, called as simplified vector of ANF of f_c.

4.4.1 Nonlinearity

Nonlinearity is an important design characteristic for cryptographic functions used in cryptographic algorithms to prevent different types of correlation or linear attacks or even related attacks. This feature is depending on the bits of the word vectors w_i. w_i is also considered as the affine transformations of the functions generated from the SRFG used. The nonlinearity is calculated by the hamming distance between two affine transformations. For example, two word are: w_i and w_j of 32 bits each.

$$\mathcal{N}l(w_{i_k}, w_{j_k}) = \sum_{k=1}^{n} w_{i_k} \neq w_{j_k}, where\ n = 32 \tag{4.16}$$

Each of the rounds in AES is using 4 words (128 bits) as sub keys. The nonlinearity between two sub keys used for any two rounds r_i,$r_j$can be measured as:

$$\mathcal{N}l(r_i, r_j) = \sum_{k=1}^{n} r_{ik} \neq r_{j_k}, where\ n = 128 \tag{4.17}$$

4.4.2 Balancedness

Balanced property of the modified key expansion function f_c exists if its simplified value vector g_c follows the following condition.

$$\forall\ i = \{1, 2\}, g_c(i) = g_c(2 - i) \boxplus\ 1, where\ \boxplus\ is\ sum\ over\ F_2 \tag{4.18}$$

The equation 4.18 also provides the feature of trivial balancedness corresponding to symmetric functions. Therefore, f_c verifies the condition $D_1 f_c$=1. The functions having $D_1 f_c$=1 do not exist for even values of n (here n =32 for words and n =128 for rounds) because for any word vector w such that $wt(w) = \frac{n}{2}$ (where, $wt(w)$ is the weight of word vector defined as number of 1s in it), we can calculate the $D_1 f_c$ as:

$$D_1 f_c = f_c(w) \boxplus f_c(w + 1) = g_c(\frac{n}{2}) \boxplus g_c(\frac{n}{2}) = 0 \tag{4.19}$$

4.4.3 Resiliency

The correlation between the output of the key expansion function and a small subset of its input variables leads to the correlation attack [179], linear or differential cryptanalysis [190]. Therefore, it is necessary for the key expansion function to achieve the high resiliency property. A function f_c of N variables each of having n bits is m-resilient if it remains balanced when any m input variables are fixed and remaining $(n-m)$ bits are altered. The function is more resilient if m is higher. The property of resiliency is related to the weights of the restrictions of the f_c to some subspaces.
$\forall f_c \in B_2$ and any affine any subspace $\mathscr{S} \subset F_2^2$, the restriction of f_c to $\mathscr{S}$ is the function given as:

$$f_{\mathscr{S}} : \mathscr{S} \rightarrow F_2 \tag{4.20}$$

$$x \rightarrow f_c(x), \forall x \in \mathscr{S} \tag{4.21}$$

where, $f_{\mathscr{S}}$ can be determined by the with a function of $dim(\mathscr{S})$ variables. The subspace $\mathscr{S}$ is spanned by k canonical basis vectors and its supplementary subspace is $\bar{\mathscr{S}}$. The restrictions of f_c to $\mathscr{S}$ and to all its cosets are given by a+ $\mathscr{S}$ where, $a \in \mathscr{S}$. Being f_c symmetric and balanced, $\mathscr{S}$ is represented as: $\mathscr{S}= (s_1, s_2, .., s_k)$ and $f_{a+\mathscr{S}}$ becomes symmetric and balanced too. Moreover, for all $s \in \mathscr{S}$, we can write the following.

$$f_{a+\mathscr{S}}(s) = f(a+s) = g_c(wt(a) + wt(s)) \tag{4.22}$$

which actually depends upon the weight of s when a is fixed. The simplified value vector and the simplified ANF vector of $f_{a+\mathscr{S}}$ can be deduced from f_c as given below.

$$g_{c_{f_{a+\mathscr{S}}}}(i) = g_c(i + wt(a)), \forall i, 0 \le i \le k \tag{4.23}$$

$$\lambda_{c_{f_{a+\mathscr{S}}}}(i) = \oplus\ \lambda_f(i+j), \forall\ i, 0 \le i \le k\ and\ j \le wt(a) \tag{4.24}$$

4.4.4 Propagation criterion

Propagation criterion is determined by the cryptographic properties of the derivatives of the functions. For the efficiency of a cryptographic function, the function need to propagate its properties to all its derivatives. All derivatives of the key expansion function are linearly equivalent when they have a fixed Hamming weight of $\frac{n}{2}$ [189]. Our approach of key expansion N variables applied from our previous work [189] satisfies the propagation criterion of degree k and order m if any affine function obtained from the outputs by keeping m input bits constant satisfies the propagation criterion of degree k. Considering each round for experimentation,

Let $f_c \in \mathcal{B}_2$ and let $r_i, r_j \in F_2^2, \ \forall i, j = 1, 2, .., 14$ such that $wt(r_i) = wt(r_j) = \frac{n}{2}$. Then, $D_{r_i} f_c$ and $D_{r_j} f_c$ are linearly equivalent.
This signifies that if we change the input variables with a linear permutation μ of F_2^2, such that $D_{r_i} f_c = D_{r_j} f_c \ \circ \ \mu$, where $\circ$ is composite function. The permutation exists on the variable in a way so that that $r_j = \mu(r_i)$. Since, f_c is symmetric and balanced, we can have,

$$D_{r_j} f_c(\mu(a)) = D_{r_i} f_c(a), where a \in \bar{\mathscr{S}} \tag{4.25}$$

Let k be an integer, $1 \leq k \leq n-1$, $z \in \bar{w}_i = (w_{i_1}, w_{i_2}, ..., w_{i_{n-k}})$ and $\epsilon_k = w_{n-k+1} + .. + w_n$. Then for any $z = a + r_j$, with $a \in \bar{\mathscr{S}}$, then we can have the following.

$$wt(z) = wt(a) + wt(r_j) \tag{4.26}$$

$$wt(z + \epsilon_k) = wt(a) + wt(r_j + \epsilon_k) = wt(a) + k - wt(r_j) \tag{4.27}$$

Thus,$\forall\ a \in V$,

$$\begin{aligned}
&D_{\epsilon_k} f_c(a+y) \\
&\qquad = f_c(a+b)\ \boxplus\ f_c(a+\epsilon_k+r_j) \\
&\qquad = wt(a)+wt(r_j+\epsilon_k) = wt(a)+k-wt(r_j) \\
&\qquad = g_c(wt(a)+w(r_j)) \boxplus (wt(a)+k-w(r_j))
\end{aligned} \tag{4.28}$$

Equation 4.28 signifies that g_c follows the symmetric property. This means that partial derivatives of our modified key expansion outputs are also propagated with the propagation features.

4.4.5 Immunity

The modified key expansion module deals with the variables (words) with 32 bits (no modification has been done on bit size). Two types of immunity are in concern: correlation immunity and algebraic immunity. For, correlation immunity, considering each of the two input variables w_i as 32 bit binary vector the outputs are correlation immune if :

$$Prob(f_c = w_i) = \frac{1}{2}\ , 1 \leq i \leq 32 \tag{4.29}$$

The probability distribution must be equal for all the bits and therefore, the output words w_o having the following property.

$$|min[M_0(w_o,(w_o)^r) - M_1(w_o,(w_o)^r)]| = min[m] \rightarrow 0 \tag{4.30}$$

where, $[M_0(w_o,(w_o)^r)]$ is the matching of output words from the key expansion process and its reverse with respect to value 0 and $[M_1(w_o,(w_o)^r)]$ is the matching of output words from the key expansion process and its reverse with respect to value 1. Following the above property, an interesting feature of our modified key expansion module has been identified and the proposition has been given as:

Proposition 4: In AES-256, if $[M_0(w_o,(w_o)^r)] = m_0$ and $[M_1(w_o,(w_o)^r)] = m_1$ *, then* $Nl(w_o,(w_o)^r) = m_0 + m_1$.

Algebraic immunity is related with the annihilator of a function [183]. To evaluate this property for our modified key expansion we can consider the following.

Given, $f_c \in \mathcal{B}_2$, any function of the set $A(f_c) = \{g \in B_2 \mid gf = 0\}$ is defined as the annihilator of the function f_c. The algebraic immunity of f_c is denoted by $AI(f_c)$ is minimum degree of all nonzero annihilators of $f_{(c)}$ or $f_{(c)} + 1$. The value of $AI(f_c)$ is given as:

$$AI(f_c) = min[deg(g)|g \neq 0, g \in A(f_c) \cup A(f_c + 1) \tag{4.31}$$

As we have used SRFG to generate the output words, minimum degree is always $\frac{n}{2}$. Therefore, the algebraic immunity of the outputs from it is always n/2 which is always optimal.

4.5 Security Analysis of RK-AES

In the above section, we have analysed the overall features of the key expansion modification in AES-256 using the SRFG. To justify the features, in this section we have performed the security analysis on our modified AES key expansion module. We have considered two attacks: related attacks and fault analysis attacks.

4.5.1 Related key attack analysis

Related key attacks use the linear relations or differential relations among the keys to deduce the original key.
Let, *nz* is a known non-zero word difference for input and *o* is an output difference of S-Box for the input difference *nz*. To execute the attack with this differences, the difference *o* can be one of $2^{14} - 1$ values, because of the symmetry of the XOR operation as used in generic AES-256 algorithm and the *nz* difference can be one of $2^{15} - 1$ differences including whitening of keys. Once, these differences are in a bounded value region, the probability deducing of the key is also higher. In our modified AES, the non-linearity feature increases this difference and therefore, the key space of searching also increases drastically. For a 32-bit word in key space, the complexity of searching space increases with the following formula.

$$Complexity\ for\ key\ space\ search = 2^{32}.2^{Nl} \tag{4.32}$$

Where, Nl is the value of non-linearity in the modified AES key expansion and the average value of Nl=20.7. Therefore, the complexity becomes $2^{52.7}$ which is more than the differential attacks key searching complexities on AES. This shows that our modified key expansion algorithm is preventive in differential attacks.
Furthermore, the attacker uses four related but unknown keys as $K_{u1}, K_{u2}, K_{u3}, K_{u4}$. The objective of the attacker is to recover K_{u1}. The relation required to establish the attack is:

$$K_{u2} = K_{u1} \oplus \triangle K^{*} \tag{4.33a}$$

$$K_{u3} = K_{u1} \oplus \triangle K^{'} \tag{4.33b}$$

$$K_{u4} = K_{u1} \oplus \triangle K^{*} \oplus \triangle K^{'} \tag{4.33c}$$

Where, $\triangle K^{*}$ is the cipher key difference used for the first related-key differential D^0 for 1 to 7 round and $\triangle K^{'}$ is the cipher key difference used for the second related-key differential D^1 used for 8 to 14 round. Assuming that, the attacker only having the information regarding $\triangle K^{*}$ and $\triangle K^{'}$, the back tracing probability to recover any 32 bit words (any word out of the 60 words) is calculated as:

$$P(w_i) = \frac{1}{(2^n.i.P_L^L.2^V)} \tag{4.34}$$

For, our modified AES-256 key expansion, number of bits in each word is n=32, total number of words including whitening key words is i = 60, total number of expression length L = 5, total number variables used for each operation is V=2. Using the values, the probability becomes as:

$$P(w_i) = \frac{1}{(2^{32} \times 60 \times P_5^5 \times 2^2)} = \frac{1}{2^{39} \times 225} \tag{4.35}$$

The above result show that, the probability is too less to recover a single word of AES-256 using our modified approach of key expansion.
In the Figure 4.4, it is shown that, the words are generated using SRFG rather than using simple XOR operation. Therefore, the equation 4.33 will not be feasible for our modified solution of AES using SRFG. It means, the modified solution is related attack resistant. Moreover, $\triangle K^{*}$ and $\triangle K^{'}$ is a factor in deducing the key. But, as our solution provides a high non-linearity, $\triangle K^{*}$ and $\triangle K^{'}$ is not suitable to recover the words of the key space. From the observation of or experimentation, we have inferred a proposition as follows.

Proposition 5: Considering $\triangle K^{*}$ is the cipher key difference used for the first related-key differential D^0 and $\triangle K^{'}$ is the cipher key difference used for the second related-key differential D^1, non-linearity is inversely proportional to the non-linearity.

$$D^0 \vdash \triangle K^{*} \propto \frac{1}{Nl} \textit{ and } D^1 \vdash \triangle K^{'} \propto \frac{1}{Nl} \tag{4.36}$$

$$\therefore D^0.D^1 \vdash \triangle K^{*}. \triangle K^{'} \propto \frac{1}{Nl^2} \tag{4.37}$$

4.5.2 Fault injection analysis

In this part, we have only considered the fault injection in the key bytes. We assume that the faulty key byte is injected in the key matrix for any random original key byte. The faulty input is inferred from the biased input of all 0 bits byte or all 1 bits byte. In the original AES, using such faulty and biased inputs reveal the relationship among word byte or even words of round. Therefore in original AES, the key recovery space is reduced with less complexity as we have seen in the literature review.
Recollecting the Equation 4.11 and Equation 4.12, we can have the following proposition for AES-256.

Proposition 6: For AES-256, using SRFG with two variables and t expression terms, the complexity of key recovery with any two random faulty byte is calculated as:

$$Prob(FI) = \frac{1}{\sum \oplus \lambda_u \left(\prod_{i=1}^{2} rand(w_i)^{u_i}\right)^t . \ C_2^{60}} \tag{4.38}$$

For any random faulty key byte , the output of the layered SRFGs is always non-linear and balanced. Therefore according to Proposition-2 the differences and or the linear equations become invalid as the fault is not further propagated to other bytes. Therefore, modified key expansion is preventive even in fault injection bytes.

4.6 Results of comparison

We have compared our experimentation results of RK-AES with the original AES algorithm. The comparison is done on the basis of some features: non-linearity, balancedness, resiliency, propagation criterion, correlation immunity and algebraic immunity. As we have modified only the key expansion module, the results are derived only for key expansion only without involving the plaintext processing or transformations in round function. We have compared 215 data samples for each RK-AES and original AES. The comparison results shown in the Table 4.1 by averaging all the results.

TABLE 4.1: *Comparison of cryptographic features of AES and RK-AES*

	Order of Non-linearity	Order of Balancedness	Order of Resiliency	Order of Propagation criterion	Order of Correlation immunity	Order of Algebraic immunity
Original AES-256	$\frac{n}{2} - log_2 n$	0	$\frac{n}{2} - 2$	$log_2 n + \frac{n}{4}$	$\frac{n}{10}$	$\frac{n}{6}$
RK-AES	$\frac{n}{2} \times 0.65 + log_2 5$	$\frac{n}{2}$	$\frac{n}{2} - 2$	$n \times 0.73 + log_2 5$	n	$\frac{n}{2}$
n is the value of bits in a word of key space						

The comparison results in Table 4.1 signifies that the modification of key expansion is working efficiently in AES in terms of the above said features. Moreover, the balancedness and the correlation immunity is 0 in original AES. The modification is providing a higher value for balancedness which is useful for preventing bitsum attacks [172]. The high correlation immunity will also help he modified AES to prevent correlation attacks [179].
Moreover, we have compared the computation time for our experiments with the original AES algorithm. In this comparison too, we have assumed the time for plaintext processing and transformations in round function are constant as no modification has done on them. Therefore, the Table 4.2 compares only the time taken for the key expansion process.The time complexity for the key expansion in Original AES is given by O(n) and for RK-AES it is O(L log n)

Table 4.2: *Time consumption comparison*

Hardware specification for computation: CPU: 2.6Ghz, i3 6th,Gen with 16 GB RAM			
	Average Time Consumption (in milliseconds)		
Schemes	32 bit key words w_0 to w_7	32 bit key words w_8 to w_{59}	g function
Original, AES	3.67	5.73	6.32
RK-AES	6.78	8.87	9.33

The time comparison results show that using the SRFG in AES key expansion modification is increasing the time consumption in generating the key words and thus contributing to the trade-off between security and time consumption. To support this trade-off and overcome with the security issues, we have also compared the attack for both, the original AES and the modified AES.

Table 4.3: *Comparison of cost of attacks on RK-AES*

	Differential cryptanalysis	Linear cryptanalysis	Related key analysis	Fault injection attack in key space
Original AES	2^{32}	$2^{14}-1$	2^{32}	2^{26}
RK-AES	$2^{52.7}$	2^{60}	$2^{52.7}$	$\approx 2^{129}$

Table 4.3 describes the fact that, the cost of the attacks for RK-AES is much higher than the original AES due to the use of randomness with SRFG in several layer. This signifies that RK-AES is better in terms of security.

Lastly, we have compared two prime evaluation parameter of encryption algorithms: confusion and avalanche effect. Confusion property requires the statistical relationship of between the ciphertext and key to be more complex. Besides, avalanche effect requires change in the cipher text bits if any single bit is changed in the key. We have calculated confusion property in terms of non-linearity and resiliency. The avalanche effect is measured in terms of propagation criterion, correlation immunity and algebraic immunity. The calculation formula for confusion and avalanche effect have been given below.

$$\begin{aligned} Confusion = \\ & \omega \times Nonlinearity + \\ & \omega \times Resiliency + \\ & \omega \times Balancedness \end{aligned} \tag{4.39}$$

$$\begin{aligned} Avalanche = \\ & \omega \times Propagation\ criterion + \\ & \omega \times Correlation\ immunity + \\ & \omega \times Algebraic\ immunity \end{aligned} \tag{4.40}$$

where, ω is the weights assigned to the features. We have considered for our experimentation of RK-AES, $\omega = 0.33$ and n=32 bit.

Therefore, following the Table 4.1, the values for confusion and avalanche effect in RK-AES are:

$$Confusion_{RK-AES} = 0.33 \times (\ \tfrac{n}{2} \times 0.65\ + log_2\ 5) + (0.33 \times \tfrac{n}{2}) + 0.33 \times (\tfrac{n}{2} - 2) \cong 22.621$$

$$Avalanche_{RK-AES} = 0.33 \times (\text{ n} \times 0.73 + log_2\ 5)\ +\ (0.33 \times \text{n}) + (0.33 \times \tfrac{n}{2}) \cong 24.31$$

Similarly, we have calculated the values for confusion and avalanche effect in original AES. Considering the orders in Table 4.1, the values are as follows.

$Confusion_{AES}$ = 0.33 $\times(\frac{n}{2}$ - log_2 n) + (0.33 × 0) + 0.33 × $(\frac{n}{2} - 2) \cong 7.59$

$Avalanche_{AES}$ = 0.33 ×(log_2 n + $\frac{n}{4}$) + (0.33×$\frac{n}{10}$) + (0.33 ×$\frac{n}{6}$)≅ 15.816

The similar result of Avalanche effect is also experimented in the bit values of the data samples. Table 4.4 compares the avalanche effect. The comparison results of

Table 4.4: *Avalanche effect comparison*

	Weighted value of Avalanche	Average number of bits changed
Original AES	15.816	17.3 bits out of 32 bits
RK-AES	24.310	≈25 bits out of 32 bits

confusion property and avalanche effect also show the improvement of the parameters as compared to the original AES algorithm.

4.7 Conclusion

AES is a popular symmetric block cipher used by different commercialization sectors. But this algorithm is facing a number of cryptanalysis effects as we have seen in the literature review. Therefore, in this chapter we have tried to solve the problem by incorporating the changes in key expansion module. The highlight of this work is to apply randomness in the key generation. Moreover, as per our previous work, using SRFG as a cryptographic function in AES has been proved beneficial. The justification for the same has already shown in the chapter. The results show that RK-AES is having three times better confusion property and 53.7% better avalanche effect as compared to the original AES. Moreover, the cost of the attacks is also higher in Rk-AES in terms of number of bits required to be exhausted which shows that RK-AES is efficient in withstanding the attacks. The limitation of our present work is about the time taken by the modified key expansion module which is actually creating a trade-off between security and time. It is also known that both these two cannot be achieved simultaneously. Therefore, if we ignore the time complexity, RK-AES is efficient in all respects of cryptographic algorithms. Furthermore, being a symmetric key algorithm AES uses the single key for both encryption and decryption. In our present work, the round keys are stored separately as each round keys are generated randomly and are used for decryption accordingly. In our future work, we shall try to work on the trade-off and also about the storage process of round keys.

CHAPTER 5

MRC4: A Modified version of RC4 with SRFG

5.1 Introduction

Cryptography is one of the major domain in network security for providing confidentiality, authentication and other security services [162]. Along with the development of the ciphers, cryptanalysis is gets attraction to the cryptographers as these processes identify the vulnerabilities in the ciphers for further improvement. The trend of converging to IoT shows a need of improving the cryptographic algorithms for applications to be secure[187][188][191]. Cryptographic algorithms are broadly categorized in different ways such as block ciphers and stream ciphers, symmetric and asymmetric ciphers [162]. Designing such algorithms requires a number of considerations such as key size, message size, functions and so on. The selection of key, its size and using a key scheduling algorithm are some major concerning factor in cryptography. A weak key can reveal the plaintext message with least time. Though we know that, cryptographic algorithms face brute-force attack, but brute-force is not considered as its complexity is higher than any other process of cryptanalysis.

Cryptographic algorithms primarily depend on the structure of the algorithms and their corresponding functions [163]. Apart from using basic gates such as AND, OR, NOT, XOR in the algorithms, researchers also have shown some specialized Boolean functions for the use. The generic Boolean functions have created the basic functionalities of generating any cryptographic function. However, the technology progress and enhancing computational ability of the attackers has urged a need of introducing new features in the functions used in ciphers for providing more strength. Balancedness, non-linearity, resiliency, immunity, correlation and propagation characteristics are some of the important parameters to evaluate the strength of the ciphers. In this chapter, we have considered Rivest Cipher 4 (RC4) for our experimentation of randomness feature. Though RC4 prohibits pseudo randomness but the use of random function generator in such stream ciphers is a novel approach in this domain.

We have attributed the key scheduling module of RC4 undergoing a modification using our Symmetric Random Function generator (SRFG) [189]. We have evaluated this new version of modified RC4 with the parameters said above.

5.2 RC4 Algorithm

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security [162][71]. It is a variable keysize stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation with a pseudo-random effect. The base of the algorithm is depending on a pseudo-random key scheduling process. A variable key length in between of 1 byte to 256 bytes (8 bit to 2048 bits) is allowed by this algorithm. It is used to initialize the state vector S. For each encryption decryption a byte k is generated that works as a key. This byte generated by a systematic method from *S*. Finally XOR operation used between the plaintext byte and the *k* byte. The algorithm of original RC4 has been summarized below.

5.2.1 Initialization of S and T

To initialize the process, the entries of state vector *S* are set equal to the values from 0 through 255 in ascending order; that is; S[0] = 0, S[1] = 1, . . . , S[255] = 255. A temporary vector, *T*, is also created which is followed from the key bytes. If the length of the key *K* is 256 bytes, then *K* is directly transferred to T byte wise. Otherwise, for a key of length *keylen* bytes, the first *keylen* elements of *T* are directly copied from *K* and then bytes from *K* is repeated as many times as necessary to complete out *T*.

5.2.2 Initial permutation in S

We use *T* generated from the above step to produce the initial permutation of *S*. This involves starting with S[0] and going through to *S[255]*, and, for each *S[i]*, swapping *S[i]* with another byte in S according to a scheme controlled by *T[i] as:*

```
for i = 0 to 255 do
    j = (j + S[i] + T[i]) mod 256;
    Swap (S[i], S[j]);
```

Use of 'mod 256' function is confining the randomness in between of 256 bytes of *S* itself. Once this S is initialized the key *K* is not used again.

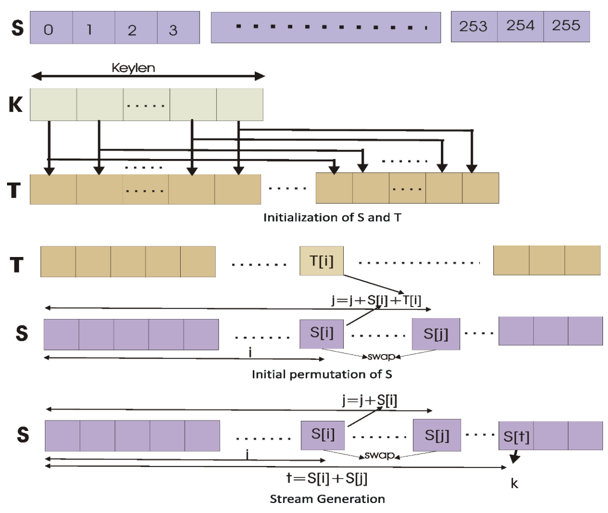


FIGURE 5.1: *Working principle of RC4*

5.2.3 Stream generation

It is used to generate the byte stream of *k*. Starting with *S[0]* and going through to *S[255]*, and, for each *S[i]*, swapping *S[i]* with another byte in S according to a scheme dictated by the current configuration of S. After *S[255]* is reached, the process continues, starting over again at *S[0]*:

```
while (true)
$i = (i + 1) \bmod 256;$
$j = (j + S[i]) \bmod 256;$
$Swap (S[i], S[j]);$
$t = (S[i] + S[j]) \bmod 256;$
$k = S[t];$
```

Finally, XOR is used between k and plaintext byte to get the ciphertext and vice-versa. The overall logic of RC4 key scheduling is diagrammatically shown in Figure 5.1.

5.3 Modified RC4 (MRC4)

The analysis of the literature review exhibits the fact the RC4 faces the cryptanalysis problems due to its key scheduling algorithm. Therefore we have modified the very first stage of RC4 key scheduling. In this modification, we have added SRFG for each byte transferred from K to T. For this, SRFG will get two inputs from two consecutive bytes of K. The bytes in K is considered to be iterative till the completion of bytes of T. For the last byte of T to be generated the last byte of K (after expanding) and the first byte K is considered as inputs of SRFG. At this time, we have stored the key stream accordingly to byte for decryption process. The modification has been summarized in Algorithm 2. All the others modules are kept same and therefore only the modified process is shown in Figure 5.2.

Algorithm 2 Algorithm for Modified RC4(MRC4)

```
1: procedure MRC4 KEY EXPANSION
2:     $for\ i = 0\ to\ 255$ do
3:     if ($keylen == 256$) then
4:         $for\ j = 0\ to\ 255$
5:         $T'[j] = k[i]$
6:         end loop
7:     else
8:         $for\ j = 0\ to\ keylen$
9:         $recursively\ copy\ k[j]\ to\ T[i]$
10:        end loop
11:    end if
12:    end loop
13:    $for\ i = 0\ to\ 255$ do
14:    $T[i] = T'[i] \otimes T'[i+1]\ where \otimes represents\ SRFG$
15:    end loop
16:    T[255]= T'[255] ⊗ T'[0]
17: end procedure
```

Figure 5.2.The modification in RC4 scheme

SRFG produces the symmetric balanced output without considering any specific input string which makes it adaptable in all types of input variables. It uses a

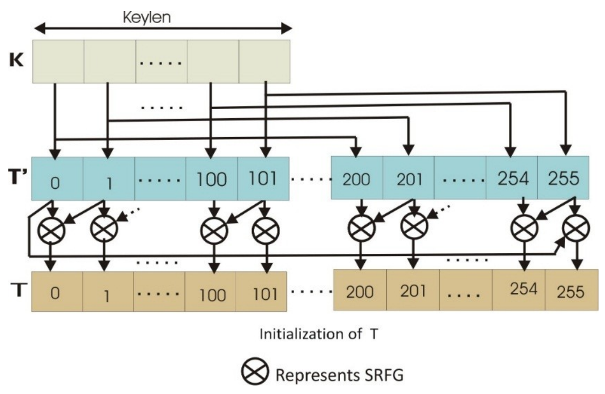


FIGURE 5.2: *The modification in RC4 scheme*

random combination of logic GATEs (AND, OR, NOT and XOR). The expression for the modified key scheduling with our SRFG can be given as:

$$f_c = \otimes f_i^L \tag{5.1}$$

where, i={1, 2,..4} four logic GATES : AND, OR, NOT and XOR; L represents the expression length (Number of terms in the combined function f_c) and $\otimes$ represents the random combination. In our experiments we have used L= 5. To emphasize the randomness in modified key scheduling, the equation 5.1 can be further expressed in terms of two input variables', as shown in Equation 5.2.

$$f_c\ (T'_j, T'_{j+1}) = \otimes f^{L_i}(T^{'}_{\ j}, T^{'}_{\ j+1}) \tag{5.2}$$

The main objective of adding SRFG in RC4 is to enable the key expansion module from K to T with randomness feature. This will help RC4 to prevent the inner state information leakage problem.

5.4 Feature Analysis of MRC4

We have experimented on the key scheduling algorithm (KSA) of RC4 with our previously developed SRFG. It removes the biasness from the key stream bytes.

This bias generates cryptanalysis attacks deducing linear or differential relations among the states or key stream bytes and backtracks the secret key. For the cryptanalysis process, it is not always necessary to have the whole key in hand; rather a single part of key is sufficient in revealing the overall key space. With the progress of cryptanalysis technologies, generating such relations or deducing keys from key streams is getting more sophisticated with less complexity.

We have explained some important features for the outputs of KSA in MRC4 such as: balancedness, non-linearity, resiliency, propagation criterion and immunity. Each byte b_i in the initial state space T' is comprised of 8 bits and is considered as 8 bit byte vector b in our experimentation.

Let $\mathcal{B}_2$ the set of all symmetric random combined functions on two variables input to the SRFG stage in generation of T for all the functions from F_2^2 into F_2 where F_2^2={ $(b_1,b_2)|\ b_i \in F_2\}$. F_2 is the finite field of two elements {0,1} and $\bigoplus$ is any operation of the field F_2.

Any combined function $f_c \in \mathcal{B}_2$ of $five$ terms is expressed as a polynomial which is basically termed as Algebraic Normal Form (ANF) of the function and given as:

$$f_c(b_1,b_2) = \oplus\lambda_u \left(\prod_{i=1}^{2} rand\ (b_i)^{u_i}\right)^5, \quad \lambda_u \in F_2\,,\ \mathrm{u} \in F_2^2 \tag{5.3}$$

with,

$$\lambda_u = \oplus f_c(b),\ \ b \preccurlyeq u, \forall\ b_i = \{b_{i_1},\ b_{i_2},\ \ldots,\ b_{i_8}\} \tag{5.4}$$

where,

$$(b_{i_1},b_{i_2},\ldots,\ b_{i_8}) \preccurlyeq (u_1,u_2,\ldots,\ u_8)\ iff\ \ \forall i,j\ ,\ w_{i_j} \le u_i\ and\ j=1,2,..,8 \tag{5.5}$$

The output of f_c depends on the weight of its input variables. Weight is calculated as total number of 1s in the variable. As a result, f_c corresponds to a function

$\mathfrak{g}_c$: $\{0,,1,\ldots,8\} \rightarrow F_2$ such that $\forall x \in F_2^2$, $f_c(x) = \mathfrak{g}_c(wt(x))$.
The sequence $\mathfrak{g}_c(f_c) = (\mathfrak{g}_c(0), \mathfrak{g}_c(1)\ldots, \mathfrak{g}_c(7))$ for 8-bit byte vector is considered as simplified value vector of f_c. To establish the relation between simplified value vector and arithmetic normal form (ANF), the equation 5.3 can be rewritten as shown in Equation 5.6.

$$f_c(b_1, b_2) = \oplus\lambda_f\ (j) \oplus \left(\prod_{i=1}^{2} rand\ (b_i)^{u_i} \right)^5 = \oplus\ \lambda_f\ (j)\ \mathcal{X}_{j,N} \tag{5.6}$$

where, $\lambda_f\ (j)$, u $\in F_2^2$ *and* $L \in \mathbb{Z}$, j=$\{1,2\}$. $\mathcal{X}_{j,N}$ is the elementary polynomial of degree j with 2 variables. The coefficients of arithmetic normal form of f_c is represented by 8$-bit$ vector, $\lambda\ (f_c)$=$\{\ \lambda_f\ (0), \lambda_f\ (1),\ldots, \lambda_f\ (8)\}$, called as simplified vector of ANF of f_c.

5.4.1 Balancedness

Balanced property in KSA in MRC4 exists if its simplified value vector $\mathfrak{g}_c$ satisfies the following condition.

$$\forall\ i = \{1,2\}\ \ ,\ \ \mathfrak{g}_c(i) = \ \mathfrak{g}_c\ (2-i)\ \boxplus 1\ ,\ where\ \boxplus is\ sum\ over\ F_2 \tag{5.7}$$

The equation 5.7 also provides the feature of trivial balancedness corresponding to symmetric functions. Therefore, f_c verifies the condition $D_1 f_c$= 1. The functions having $D_1 f_c$= 1 do not exist for even values of n (here n =8 for bytes) because for any byte vector b, $wt(b)$=$n/2$ ($where$, $wt(b)$ is the weight of byte vector defined as number of 1s in it), we can calculate the $D_1 f_c$ as:

$$[D_1 f_c = f_c\ (b) \boxplus f_c\ (b+1) = \mathfrak{g}_c(\ n/2) \boxplus \mathfrak{g}_c\left(\frac{n}{2}\right) = 0\ ] \tag{5.8}$$

As each byte is balanced, the overall bytes in initial states are also balanced. Therefore, this balancedness property helps to prevent attacks depending upon the weight or hamming distances.

5.4.2 Nonlinearity

We have measured nonlinearity of the modified KSA. This feature depends on the bits of the byte vectors b_i. b_i is also considered as the affine transformations of the functions generated from the SRFG used in KSA. The nonlinearity is calculated by the hamming distance between two affine transformations. For example, two byte vectors are: b_i and b_j of 8 bits each.

$$\mathcal{N}\ell\left(b_{i_k}, b_{j_k}\right) = \sum_{k=1}^{n} b_{i_k} \neq b_{j_k}, \; where \; n = 8 \tag{5.9}$$

This property of non-linearity will remove the probability of bias and reducing the chances deducing relations with all 0s or all 1s inputs. Each byte follows this non-linearity to generate the initial state of T.

5.4.3 Resiliency

The KSA in MRC4 is *m-resilient* if the output remains balanced when any m input bits are fixed and remaining *(8-m)* bits are altered in next sequence. The function is more resilient if m is higher. The property of resiliency is related to the weights of the restrictions of the $\mathrm{f_c}$ to some subspaces.

$\forall\ \mathrm{f_c} \in \mathcal{B}_2$ and any affine any subspace $\mathcal{S} \subset F_2^2$, the restriction of $\mathrm{f_c}$ to $\mathcal{S}$ is the function given as:

$$\mathrm{f}_{\mathcal{S}} : \quad \mathcal{S} \rightarrow F_2 \tag{5.10}$$

The subspace $\mathcal{S}$ is spanned by k canonical basis vectors and its supplementary subspace is $\overline{\mathcal{S}}$. The restrictions of $\mathrm{f_c}$ to $\mathcal{S}$ and to all its cosets are given by $b' +$

$\mathcal{S}$ where, $b' \in \overline{\mathcal{S}}$. Being $\mathrm{f_c}$ symmetric and balanced, $\mathcal{S}$ is represented as: $\mathcal{S} = \langle s_1, s_2, \ldots, s_k \rangle$ and $\mathrm{f}_{a+\,\mathcal{S}}$ becomes symmetric and balanced too. Moreover, for all $s \in \mathcal{S}$, we can write the following.

$$\mathrm{f}_{b'+\,\mathcal{S}}(s) = f(b'+s) = \mathfrak{g}_c\ (wt(b') + wt(s)) \tag{5.11}$$

The simplified value vector and the simplified ANF vector of $\mathrm{f}_{a+\,\mathcal{S}}$ can be deduced from f_c as given below.

$$\mathfrak{g}_{c\mathrm{f}_{b'+\,\mathcal{S}}}\ (i) = \mathfrak{g}_c\ (i + wt(b')),\ \forall i,\ 0 \le i\ \le k \tag{5.12}$$

$$\lambda_{f_{b'+\,\mathcal{S}}}(i) = \oplus\ \lambda_f\ (i+j), \qquad \forall i,\ 0 \le i\ \le k\ \ and\ \ j \preccurlyeq wt(b') \tag{5.13}$$

Proposition 7: For the KSA in MRC4 with m-resiliency factor, two consecutive key stream bytes with 8 bit each have maximum non-linearity of 5. In such cases, the difference between simplified value vectors of the two consecutive bytes of T equals to the sum of the non-linearity of two consecutive bytes $b_{i,}b_{j,}$ in *T*.

$$\max[\mathcal{N}\ell\ (b_{\mathrm{i}}\ ,\ b_{\mathrm{i+1}})]\ \rightarrow 5 \tag{5.14}$$

$$[\ \mathfrak{g}_{c\mathrm{KSA}}\ (T_k) - \mathfrak{g}_{c\mathrm{KSA}}\ (T_{k+1})] = \sum_{i,j=0}^{255} \mathcal{N}\ell\left(b_{\mathrm{i}}\ ,\ b_{\mathrm{j}}\right) \tag{5.15}$$

5.4.4 Propagation criterion

Following the basic properties of SRFG as shown in our previous work [189], the modified KSA satisfies the propagation criterion of degree k and order m as the outputs of KSA can be represented by affine function keeping m input bits constant and satisfies the propagation criterion of degree k. Considering each key stream byte for experimentation, Let $f_c \in \mathcal{B}_2$ and let $b_i, b_j \in F_2^2$, $\forall\ i,j = 1,2,..,\ 256$, such that $wt(b_i){=}wt\left(b_j\right){=}\frac{n}{2}$. Then, $D_{b_i}f_c$ and $D_{b_j}f_c$ are linearly equivalent. This signifies that if we change the input variables with a linear permutation μ of F_2^2, such that $D_{b_i}f_c = D_{b_j}f_c \circ \mu$, where $\circ$ is composite function. The permutation μ exists on the variable in a way so that that $b_j = \mu(b_i)$. Since, f_c used in KSA is symmetric and balanced, we can have,

$$D_{b_j}f_c\ (\mu(\mathrm{a})) = D_{b_i}f_c\ (a),\ where\ a\ \in \overline{\mathcal{S}} \tag{5.16}$$

Let k be an integer, $1 \leq k \leq n{-}1$, $\mathrm{z} \in \overline{b_i}{=} \left\langle b_{i_1},\ b_{i_2},\ldots,\ b_{i_{n-k}}\right\rangle$ and $\varepsilon_k{=}\ b_{n-k+1}{+}\cdots{+}w_n$. Then for any $z = a + b_j$, with $a\ \in\ \overline{\mathcal{S}}$, then we can have the following.

$$wt(z){=}wt(a){+}wt\left(b_j\right) \tag{5.17}$$

$$wt(z{+}\ \varepsilon_k){=}wt(a){+}wt\left(b_j{+}\ \varepsilon_k\right){=}wt(a){+}k{-}wt(b_j) \tag{5.18}$$

Thus,$\forall\ a{\in}B,\ B = \{b_0, b_i \ldots,\ b_{255}\}$

$$D_{\varepsilon_k}f_c\ (a{+}y){=}f_c(a{+}b)\ \boxplus f_c\left(a{+}\varepsilon_k{+}b_j\right){=}wt(a){+}wt\left(b_j{+}\ \varepsilon_k\right){=}wt(a){+}k{-}wt\left(b_j\right)$$

$$={\mathfrak{g}}_c\left(wt(a){+}w\left(b_j\right)\right)\boxplus\ \left(wt(a){+}k{-}w\left(b_j\right)\right) \tag{5.19}$$

Equation 5.19 signifies that $\mathfrak{g}_c$ follows the symmetric property. This means that partial derivatives of our modified KSA outputs are also propagated with the symmetric features.

5.4.5 Immunity

Correlation immunity and algebraic immunity are examined for the KSA in MRC4. For, correlation immunity, considering each of the two input bytes b_i as 8 bit binary vector the outputs are correlation immune if :

$$Prob(\, f_c = b_i) = \frac{1}{2}\,,\ 1 \leq i \leq 8 \tag{5.20}$$

The probability distribution must be equal for all the bits and therefore, the output byte $b_o \in$T , having the following property.

$$|\min[M_0\ (b_o, (\, b_o)^r) - M_1\ (b_o, (\, b_o)^r)\,]| = \min[m] \mapsto 0 \tag{5.21}$$

where, $[M_0\ (b_o, (\, b_o)^r)]$ is the matching of output bytes from KSA and its reverse with respect to value 0 and $[M_1\ (b_o, (\, b_o)^r)]$ is the matching of output words from KSA and its reverse with respect to value 1.

Following the above property, an interesting feature of our modified key expansion module has been identified and the proposition has been given as:

Proposition 8: In MRC4, if $[M_0\ (b_o, (\, b_o)^r)] = m_0$ *and* $[M_1\ (w_o, (\, w_o)^r)] = m_1$, then the maximum non-linearity between two successive T_k will be the sum of all m_0 and m_1 for all the b_o in T.

$$max[\mathcal{N}\ell\ (T_i\ , T_{i+1}\ )]\ = \sum_{1}^{256} m_0 + m_1,\ for\ all\ b_o \text{ in T}$$

Algebraic immunity is related with the annihilator of a function. To evaluate this property for the modified KSA we have considered the following. Given, $f_c \in \mathcal{B}_2$, any function of the set $A(f_c) = \{\, g \in \mathcal{B}_2 \,|gf = 0\,\}$ is defined as the annihilator of the function f_c . The algebraic immunity of f_c is denoted by $AI(f_c)$ is minimum degree of all nonzero annihilators of f_c or f_c +1. The value of $AI(f_c)$ is given as:

$$AI(f_c) = \min[\deg(g)\,]\,|g \neq 0,\, g \in A(f_c) \cup A(f_c{+}1) \tag{5.22}$$

As we have used SRFG to generate the output bytes, minimum degree is always n/2. Therefore, the algebraic immunity of the outputs from it is always n/2 which is always optimal.

We can rewrite the equation 5.22 for two consecutive state of T as:

$$AI(f_c)=\min[\deg(T_i)] \mid f_c \rightarrow T_i g\neq 0,\ A(f_c)\cup A(f_c+1)\neq 0 \tag{5.23}$$

$$\vdash AI(f_c)-AI(f_{c+1})=0 \tag{5.24}$$

5.5 Security Analysis

The analysis of the previous work of cryptanalysis processes on RC4 show that the cipher suffers from the problem of biased distinguisher which eventually deduce the relational statistics for identifying the secret key. The biasedness is generated with all 0s or all 1s values of bytes of inputs. Moreover, related key attacks are also responsible for linear and differential cryptanalysis on RC4. Therefore, in this section, we have analysed the security features of MRC4 in the above said perspective.

5.5.1 Related key attack analysis

We focus on the S-box initialization responds to a single-byte difference in its input. Assume any random key $k^{'}(i)=k(i)$ where, $k(i)$ is the original key except when $i=t$ where $k(t)\neq k'(t)$. In this case, $j=(j+S(i)+k(i)) mod\ 256$, will result in different values of *j*. If *t* is close to zero, the resulting S-boxes will be completely different. If t is close to 255, however, the S-boxes will be substantially similar because the first *t-1* iterations through the initialization loop performed exactly the same work. We have used a twiddled key $k^{'}(t)=k(t)+\mu$ and therefore $k^{'}(t+1)=k(t+1)-\mu$, then the value of j at i-th iteration will be $j'_i=j_i+\mu$ and j'_{i+1} will be same as j_i. This *k'* and *k* are called related keys. With this key-setup, in the initialization phase, the RC4 output for the original key and the related key will proceed in lock step for $b'_i = b_i$ still $s'_{i-1} \neq s_{i-1}$. At this point we define it as derailment of RC4 systems and identical bytes produced by the two keys is defined as the length of the derailment.

In our experiment, for two randomly chosen keys, in the initialization phase of the KSA, the probability of two bytes to be similar is given as:

$$P(b'_i = b_i) = \sum_{i=0}^{255} \bigoplus \prod_{4}^{5} P(b'_i) \times P(b_i) = 256 \times \frac{1}{3125} \times \frac{1}{256} \times \frac{1}{256} = 0.00000125 \tag{5.25a}$$

To generalize the equation 5.25a for any x bytes of key, the equation 5.25a can be re-written as:

$$P(b'_i = b_i) = \sum_{i=0}^{i=x-1} \bigoplus \prod_{4}^{5} P(b'_i) \times P(b_i) \tag{5.25b}$$

The probability of the derailment length of d is calculated as:

$$P(d) = P(b'_i = b_i)^d \rightarrow 0 \ \ \textit{due to its least value} \tag{5.26}$$

5.5.2 Linear cryptanalysis

We have followed the bit-advantage concept for analysis of linear attacks. If an attack is executed on an n-bit key and recovers the correct value of the key ranked among the top m out of 2^n possible candidates, the attack obtains an *(n - log (m))* -bit advantage over exhaustive search. In such attacks, linear equations are made from he known plaintexts functioning with cipher text to get the key bytes. Linear cryptanalysis is very common in RC4 as the key bytes are directly XORed with plaintext bytes and key bytes are achievable. But, in MRC4 we have processed KSA with SRFG so that for any two random bytes of key stream the total key is not going to be deduced. Therefore, we have analysed the linear relation approximation on two related keys rather than on known plaintexts as below.

Following to the theorem 2 in the research work [192], if P_s be the probability that a linear attack on an n-bit key stream byte (8 bits here), with a linear approximation of probability p, with N known related key stream bytes, delivers an m-bit or higher advantage. Assuming that the linear approximation's probability to hold is independent for each key tried and is equal to $\frac{1}{2}$ for all wrong keys, we can have:

$$P_s = p^N (n-m) \left(1 - 2^{m-1}\right) \tag{5.27}$$

Now, to calculate the linear approximation of probability p of the attack we need to evaluate hat whether any two wrong keys k_1 and k_2 is deriving two similar bytes or not. The probability of such similarity has been calculated in equation 5.25. This shows that, the success probability P_s is also reduced and approximates to 0 if $m \rightarrow n$. Therefore, equation 5.27 can be re-written as:

$$P_s = p^N (n-m)\left(1 - 2^{m-1}\right) = (\approx 0)^N (n-n)\left(1 - 2^{n-1}\right) = 0 \tag{5.28}$$

Therefore the proposition can be written as:

Proposition 9: In MRC4, if $P(b'_i = b_i) \rightarrow 0$ *and n-bit random key stream byte attempts to generate an n-bit advantage, the success probability* $P_s = 0$.

5.5.3 Differential Cryptanalysis

We concentrate on keys of 256 bits as these keys are very common in WEP implementations. The differential cryptanalysis deduces the output streams that are expected to be the same in the first few bytes even though input has been changed. Generally, this cryptanalysis is a type of chosen plaintext. Therefore, we need to check the probability of getting any two similar cipher text bytes for two chosen plaintext bytes.

Let pt' *and* pt'' are two chosen plaintext of l bytes with a known difference MRC4 and given as:

$$pt^{'} - pt^{''} = \mathcal{N}\ell(\text{ pt}', \text{ pt}'') \tag{5.29}$$

Due to the convenience of calculation, we assume that the non-linearity is uniformly distributed among all the bytes of the chosen plaintexts and therefore, from equation 5.29 we can write the following.

$$\mathcal{N}\ell\ (\text{ pt}^{'}_{i}, \text{pt}^{''}_{i}) = \frac{\mathcal{N}\ell(\text{ pt}^{'},\ \text{pt}^{''})}{l} = \Delta_i,\ for\ i = 0, 1, 2, .., l-1 \tag{5.30}$$

Following the equation 5.25b, the probability of any two ciphertext bytes to be similar can be calculated as:

$$P(ct^{'}_{i} = ct''_{i}) = \frac{1}{\frac{l!}{2!\ \times (l-2)!}} \times P(b^{'}_{i} = b_i) \times \frac{1}{\Delta_i} \tag{5.31a}$$

Assuming *a* key size of 256 bits (32 bytes) and plaintext size of 64 bytes, we can write from the equation 5.31a is:

$$\begin{aligned} P(ct^{'}_{i} = ct^{''}_{i}) & \\ &= \frac{1}{\frac{l!}{2!\ \times (l-2)!}} \times P(b^{'}_{i} = b_i) \times \frac{1}{\Delta_i} \\ &= \frac{1}{2016} \times 32 \times \frac{1}{3125} \times \frac{1}{32} \times \frac{1}{32} \times \frac{1}{\Delta_i} \\ &= 5 \times 10^{-9} \times \frac{1}{\Delta_i} \end{aligned} \tag{5.31b}$$

Table 5.1: *Comparison of Features of MRC4 and other algorithms*

	Order of Non-linearity	Order of Balancedness	Order of Resiliency	Order of Propagation criterion	Order of Correlation immunity	Order of Algebraic immunity
Original RC4	$\frac{n}{10}$	0	0	0	$\frac{n}{10}$	0
Weerasinghe [158]	$\frac{n}{10}$	0	0	0	$\frac{n}{4}$	0
RC4-M3 [159]	$\frac{n}{4}$	0	$log_2 n$	$\frac{log_2 n}{n}$	n	$\frac{n}{8}$
MRC4	$\frac{n}{3}$	$\frac{n}{2}$	$\frac{n}{2}-2$	$log_2 n$	n	$\frac{n}{2}$
n is the value of bits in cipher text considering all the bits, orders are calculated based upon maximum degree						

The equation 5.31b shows that using the two chosen plaintext the probability of two random similar cipher bytes is impractical. Therefore, differential analysis is not possible on our MRC4. From, this analysis, we can infer the following proposition.

Proposition 10: In MRC4, the probability of the two similar ciphertext bytes is inversely proportional to the number of plaintext bytes and inversely proportional to nonlinearity and given as:

$$P(ct'_i = ct''_i) \propto \frac{1}{\Delta_i} \tag{5.32a}$$

$$P(ct'_i = ct''_i) \propto \frac{1}{l} \tag{5.32b}$$

From the above, we can write:

$$P(ct'_i = ct''_i) \propto \frac{1}{\Delta_i \times l}$$

$$\begin{aligned} &\vdash P(ct'_i = ct''_i) \\ &= k \times \frac{1}{\Delta_i \times l} \text{, where } k \text{ is constant and given as } k \\ &= P(b'_i = b_i) = 0.125 \times 10^{-4} \end{aligned} \tag{5.33}$$

5.6 Performance Evaluation

We have implemented MRC4 in software with hardware specification as: CPU: 2.6Ghz, i3 6th Gen with 4 GB RAM. We have compared our experiment results with the original RC4 and two other improvements of RC4 in recent times. We have saved the key bytes generated from the key stream generator and we shall consider this as a feature work to deal with key saving process. The key stream generation process is only modified in MRC4. But, still we have measured the results with the overall process of encryption and decryption. We have varied the key sizes from 64 bits to 2048 bits in all the performance metrics.

The first comparison is done on the basis features extracted in the section 5: non-linearity, balancedness, resiliency, propagation criterion, correlation immunity and algebraic immunity. We have measured the order of the features stated above and compared accordingly as in Table 5.1.

The comparison results in Table 5.1 signifies that the modification of key stream generation is efficient in terms of the above said features. We can see from the table that, original RC4 was lacking behind in acquiring the important features of cryptographic functions and therefore, the attacks as considered in the previous work are executable on this cipher. Better results are noticed in improvement proposed in [158] as compared to original RC4 but still it is not enough for the purpose. Further, the improvement shown in [159] provides far better result as compared of previous two algorithms. MRC4 has provided improved features and the optimality of balancedness which is useful for preventing bitsum attacks [172].The high correlation immunity will also help the MRC4 to prevent correlation attacks [193].

TABLE 5.2: *Time Consumption of KSA*

Hardware specification for computation: CPU: 2.6Ghz, i3 6th,Gen with 4 GB RAM						
Scheme	*Key size*					
	64 bit	*128 bit*	*256 bit*	*512 bit*	*1024 bit*	*2048 bit*
Original RC4	20.32	18.39	17.58	17.23	16.42	16.13
Weerasinghe [158]	25.35	20.15	19.7	18.71	17.45	16.57
RC4-M3 [159]	44.42	43.73	43.09	42.66	40.78	40.32
MRC4	43.47	42.83	41.16	40.27	39.50	39.20

Moreover, we have compared the computation time for our experiments with the original RC4 and other two algorithms [158][159] as shown before. In this comparison too, we have assumed the time for XORing operation is a constant for both encryption and decryption and therefore not considered in the time consumption calculation. Therefore, the Table 5.2 only compares the time taken for the Key Scheduling Algorithm (KSA) by varying the key size. The values are given in microseconds. The time complexity of key scheduling algorithm for Original RC4 is

O(n) and MRC4 is O(log nL). The time comparison results show that use of the

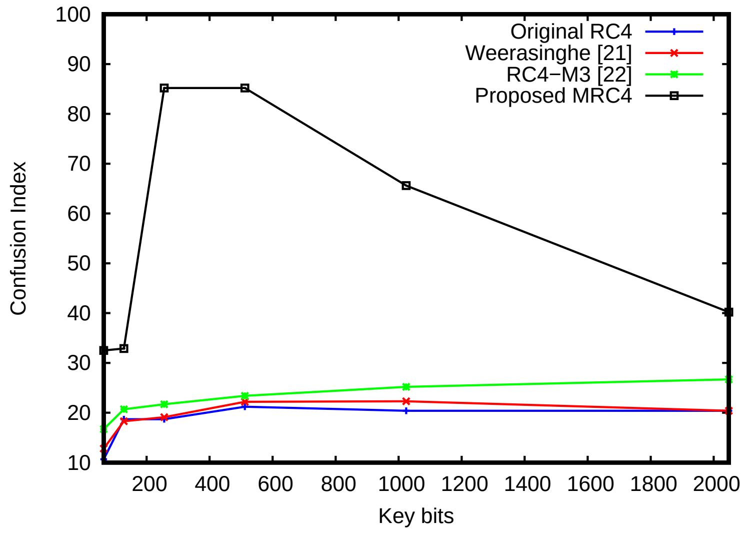


FIGURE 5.3: *Confusion index comparison*

SRFG in RC4 key scheduling modification is increasing the time consumption in generating the key stream bytes and thus contributing to the trade-off between security and time consumption. The same has also been addresses by the authors in [159]. To support this trade-off and overcome with the security issues, we have also compared the attacks on the algorithms in terms of cost of bits as shown in Table 5.3. For this comparison we have considered 50 plaintexts of 128 bytes each with 128 bit (16byte) keys.

Table 5.3 describes the fact that, the cost of the attacks for MRC4 is much higher than the other algorithms due to the use of randomness with SRFG in KSA. This signifies that MRC4 is better in terms of security.

Lastly, we have compared two prime evaluation parameter of cryptographic algorithms: confusion and diffusion. According to Shannon's theory [1], confusion property exhibits the statistical relationship of between the ciphertext and key to be more complex whereas, diffusion property yields the relationship between plaintext and ciphertext such that a single bit change in plaintext must change at least half of the cipher text bits. We have introduced two new metrics in this point

TABLE 5.3: *Comparison of cost of attacks on MRC4*

	Differential cryptanalysis	**Linear cryptanalysis**	**Related key attacks**
Original RC4	2^{128}	2^{64}	2^{128}
Weerasinghe [158]	2^{128}	2^{128}	2^{128}
RC4-M3 [159]	2^{512}	2^{128}	2^{512}
MRC4	2^{1024}	2^{1024}	2^{3125}

to have a bounded range of the metric and given as:

$$Confusion\ index = \frac{key\ bits \times nonlinearity\ in\ two\ simulatenous\ ciphertext}{ciphertext\ bits} \tag{5.34}$$

$$Diffusion\ index = \frac{nonlinearity\ in\ two\ simulatenous\ ciphertext}{plaintext\ bits} \tag{5.35}$$

The maximum value of the confusion index in Equation 5.34 can be $\propto$ which is not possible in the real life implementation scenarios as all the cipher text bits cannot be changed for a single bit change in key. If it happens then, it will work as a bias in next state initialization. We have varied the key bits from 64 to 2048 with a fixed plaintext of 128 bytes (1024 bits) and as an output we have received 128 bytes (1024 bits). The results are compared with other three algorithms as previous shown in Figure 5.3 and Figure 5.4 respectively.

From the Figure 5.3, we can observe that MRC4 is better in providing confusion. Moreover, an interesting behaviour of MRC4 has been observed here. It shows that upto 128 bit keys, the confusion index is almost static and with the key size of 256 bits to 512 bits it increases with a high slope and with higher than 512 bits of keys, confusion index decreases rapidly. This fact signifies that for MRC4 using

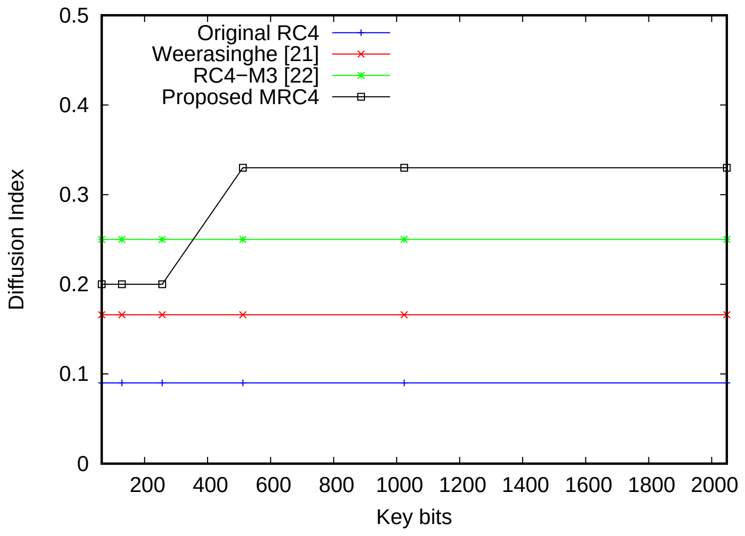


FIGURE 5.4: *Diffusion index comparison*

the key sizes in between of 256 to 512 bits is most efficient. Similarly, 5.4 shows that original RC4 and the algorithm shown in reference [158] are having very low diffusion property. The algorithm in reference [159] is having a better diffusion index. Like confusion, here also MRC4 follows a constant diffusion index up to the key size of 256 bits but drastically changes with the use of 512 bits of keys and gets constant after with higher bits of keys. Analysing the behaviour of MRC4 in confusion and diffusion, we can say that use of 512 bits key is considered to be the most efficient use in MRC4 to get high cryptographic features.

5.7 Conclusion

RC4 is a popular symmetric stream cipher used by different network security technologies. But this algorithm has got a setback due to a number of cryptanalysis attacks. Therefore, in this chapter we have tried to solve the problem by incorporating the changes in key scheduling structure. The highlight of the present work is to apply randomness in the key stream bytes. The experimental results show that MRC4 possess 60% better confusion property and 50% better diffusion as compared to the original RC4 and other algorithms. The limitation of our present work is about the time taken by the KSA which is actually creating a trade-off between security and time. It is also known that both these two cannot be achieved simultaneously. Therefore, if we ignore the part of the time, MRC4 is efficient in all respects of cryptographic features. Furthermore, being a symmetric key algorithm RC4 uses the single key for both encryption and decryption. In our present work, the key stream bytes are stored separately as each key stream bytes are generated randomly and are used for decryption accordingly. In our future work, we shall try to work on analysing this trade-off between storing of random keys and its security and its related space complexity.

CHAPTER 6

Conclusion and Future Work

6.1 General

Cryptography, in its present form, has been considered as one of the integral part of information theory and coding. The variety of applications and its requirements for security services have given a scope for the crypto designers to design cryptographic algorithms. All these algorithms considers a mathematical model to be applied in the systems and therefore, information theory has been directly involved with it. Thus cryptography also provides a mathematical and a deterministic way to measure the security issues with a particular system. A number of algorithms has been developed so far considering all their corresponding pros and cons. Some of the algorithms are obsolete due to their severe weakness and lack of abilty of reviving. Moreover, some algorithms are facing problems with cryptanalysis break down, reconfiguring themselves for more security strength. This process of cryptographic algorithm design and cryptanalysis is therefore a cyclic process. In our present work, we have experimented this and have evaluated the results and have observed some of the new findings of cryptographic behavior.

6.2 Summary of Important Findings

In the present work, we have examined and developed a symmetric random function generator using the basic GATEs (AND, OR, XOR and NOT) for providing randomness in the algorithms used in cryptography. This randomness protects the algorithms from cryptanalysis attacks with a cost of large number of bits. As the IoTs, sensor networks, cloud computing are advancing day by day with number of technological developments, the design of cryptographic algorithms must be set strict. SRFG is able to provide true randomness of bits in the outputs and therefore cryptanalysis using such a process is critical. The primary focus of the

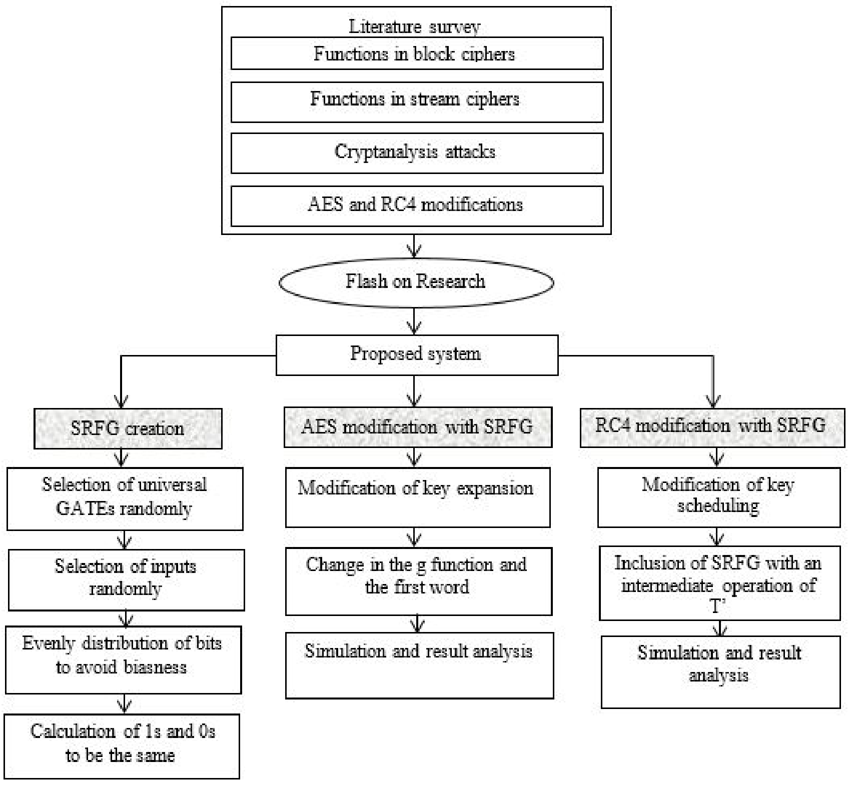

Literature survey
Functions in block ciphers
Functions in stream ciphers
Cryptanalysis attacks
AES and RC4 modifications
Flash on Research
Proposed system
SRFG creation
AES modification with SRFG
RC4 modification with SRFG
Selection of universal GATEs randomly
Selection of inputs randomly
Evenly distribution of bits to avoid biasness
Calculation of 1s and 0s to be the same
Modification of key expansion
Change in the g function and the first word
Simulation and result analysis
Modification of key scheduling
Inclusion of SRFG with an intermediate operation of T'
Simulation and result analysis

FIGURE 6.1: *Summarization of Contribution*

thesis is Symmetric Random Function Generator (SRFG). Firstly, we have generated a function generator which combines the basic GATEs randomly and outputs a function. The inputs to the basic GATEs are also selected randomly. Therefore, two stages of randomness have been included in our developed system. We have analyzed the basic behavior of the SRFG. We found that SRFG shows relatively better results as compared to other random structures. Secondly, we have applied this SRFG to modify the key expansion module of two popular algorithms: AES as the block cipher and RC4 as the stream cipher. In both the cases we have identified some of the improved results which can be used as mainstream applications. The findings of the overall thesis have been shown in the Figure-6.1.

The salient features of the findings from the research work have been summarized in Table 6.1.

TABLE 6.1: *Salient features of the research work*

	Salient features	**Issues addressed**
Development of Symmetric Random Function Generator	• Random selection of basic GATEs • Random selection of inputs • Outputs possess cryptographic features: balancedness, non linearity, propagation, immunity, symmetric, resiliency • Expandable to any number of bits	• Balance property of outputs • Randomness

RK-AES	• Use of SRFG in key expansion module • g function has been modified • Round construction array has been used to use the SRFG for the very first column of the key matrix	• Randomness in key bits • Avoidance of biased bits • Strength to withstand cryptanalytic attacks
MRC4	• Use of SRFG in key expansion module • Initiating a intermediate T matrix for key bits to be utilized for SRFG	• Randomness in key bits • Avoidance of biased bits • Best usage of a key-size • Strength to withstand cryptanalytic attacks

The comparison of the developed system and modification with the existing methods is shown in Table 6.2

TABLE 6.2: *Comparison of features of the present research work with existing work*

Proposed method	**Existing Methods**	**Remarks**
SRFG	• Zhang et. al. [181] • Li et. al. [182] • Wang et. al. [186]	• The results show that the present approach provides more efficient cryptographic features: balancedness, resiliency, symmetric, propagation and immunity. • Approximately twice better than the existing systems in the overall performance.

RK-AES	• Original AES [36]	• Original AES is only compared as randomness in key expansion of AES is new in this domain. • Thrice better confusion property • 53.7% better Avalanche effect • Higher cost of bits for cryptanalysis attacks
MRC4	• Original RC4 [71] • Weerasinghe [158] • RC4-M3 [144]	• 50% better diffusion property • 60% better confusion property • Higher cost of bits for cryptanalysis attacks

6.3 Future Work

The symmetric ciphers use the same key for both encryption and decryption. In our present research of the modifications in AES and RC4, we have faced the challenges regarding the storage of the keys. We have used SRFG in both the key

management process and the modifications are showing very good cryptographic features. But, we have not considered the storage of the individual keys for decryption in our present work and we have stored the keys separately for corresponding messages (both plaintext and ciphertext) which has significantly increased the storage complexities. Therefore, we have considered this part as a future work and we shall try to develop a dynamic key management process where the same key will be dynamically used for encryption and decryption.
Some of the extensible future research problems are listed below.

- Generation of Hash algorithm and performance analysis of SRFG in cryptographic hashes.
- Performance of SRFG based algorithms in withstanding Quantum Computing attacks.
- Monte Carlo experimentation can be done with SRFG outputs with any feasible source by integrating a function output over a large parameter space and selecting random values within that parameter space and averaging the resulting function values. This is often used in finance and engineering.
- Digital display with the effect of rotation and scaling can be analysed using the randomization of pixel area using the concept of SRFG.
- In Machine Learning and Statistical Learning, random number generators are the basis to obtain samples from interesting distributions (Gamma, Beta, etc).
- CAPTCHA technology can be used with SRFG by randomly choosing the images or texts or even merging the text and images.
- SRFG can be used in amortized searching and sorting algorithms.

This open and extensible future research work will help the upcoming researchers to identify the problem definition and flourish the potentialities and multidimensional features of or SRFG.

BIBLIOGRAPHY

[1] Shannon, Claude E. 1948. A Mathematical Theory of Communication. The Bell System Technical Journal 27(July 1928): 379423. http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf.

[2] Van Tilborg, H. C. A. 2010. Fundamentals Of Cryptology - A Professional Reference and Interactive Tutorial. FUNDAMENTALS OF CRYPTOLOGY 76(5): 503. http://www.ncbi.nlm.nih.gov/pubmed/20545048.

[3] Schneier, Bruce. 1997. Applied Cryptography, 1996. John Wiley & Sons Inc.,: 12547.

[4] Kartalopoulos, Stamatios V. 2006. A Primer on Cryptography in Communications. IEEE Communications Magazine 44(4): 146-51.

[5] Diffie, Whitfield. 1988. The First Ten Years of Public-Key Cryptography. Proceedings of the IEEE 76(5): 560-77.

[6] H. M. Heys. 2004. ”A Tutorial on Linear and Differential Cryptanalysis”. University of Newfoundland, St. John’s, NF, Technical report, Electrical and Computer Engineering.

[7] Shannon, C. E. 1949. Communication Theory of Secrecy Systems. Bell System Technical Journal 28(4): 656-715.

[8] Blaze, M, W Diffie, Rl Rivest, and Bruce Schneier. 1997. Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security, January 1996.

[9] https://en.wikipedia.org/wiki/Substitutionpermutation_network

[10] http://www.wikiwand.com/simple/Feistel_cipher

[11] J. Smith. 1971. ”The design of Lucifer: a cryptographic device for data communications”. IBM T.J. Watson Research Center, Yorktown Heights, N.Y., USA, Technical report.

[12] NIST, National Institute of Standards and Technology. 1999. Data Encryption Standard (DES). Federal Information Processing Standards Publication (FIPS PUB 46-3) 25(10): 1-22.

[13] Kilian, Joe. 2001. How to Protect Des against Exhaustive Key Search (an Analysis of DESX). Journal of Cryptology 14(1): 17-35.

[14] Shimizu, Akihiro, and Shoji Miyaguchi. 1988. FEAL: Fast Data Encipherment Algorithm. Systems and Computers in Japan 19(7): 2034.

[15] R. Rivest. 1998. A description of the RC2(r) encryption algorithm. RFC editor, United States.

[16] Merkle, Ralph C. 2014. Advances in Cryptology—CRYPTO 90 Fast Software Encryption FSE 2014.

[17] S. Miyaguchi. 1990. ”The FEAL cipher family”. Advances in Cryptology. Springer-Verlag, pp: 627-638.

[18] Kwan, M, J Pieprzyk, J Brown, and L Seberry. 1991. Improving Resistance to Differential Cryptanalysis and the Redesign of LOKI. In Advances in Cryptology - ASIACRYPT91, , 36-50.

[19] M. Wood and T. Cusick. 1990. ”The RedocII cryptosystem”. Advances in Cryptology - Crypto’90, pp: 545-563.

[20] Lai, Xuejia, James L. Massey, and Sean Murphy. 1991. Markov Ciphers and Differential Cryptanalysis. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 17-38.

[21] Bruce Schneier. 1993. ”Description of a new variable length key, 64-bit block cipher (Blowfish)”. Fast Software Encryption, Cambridge Security Workshop Proceeding Springer-Verlag, pp: 191-204

[22] Massey, J L. 1994. SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm. Fast Software Encryption 809(809): 1-17.

[23] W. Wlfovics and A. Di Porto. 1994.”VINO: a block cipher including variable permutations”. Fast Software Encryption, Springer-Verlag, pp: 205-210.

[24] Dinur, Itai, Orr Dunkelman, and Adi Shamir. 2012. Improved Attacks on Full GOST. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9-28.

[25] M. Blaze and B. Schneier. 1995. ”The MacGuffin cipher algorithm”. Software Encryption: Second International Workshop, Springer-Verlag, pp: 97-110.

[26] R. Rivest. 1995. ”The RC5 encryption algorithm”. Fast Software Encryption: Second International Workshop, Springer-Verlag, pp: 86-96.

[27] D.J. Wheeler and R.M. Needham. 1994 ”TEA, a Tiny Encryption Algorithm”. in Proc. FSE, pp: 363-366.

[28] Matsui, Mitsuru. 1996. Block Encryption Algorithm MISTY. Technical report of IEICE ISEC 96 167: 35-48. http://ci.nii.ac.jp/naid/110003296962.

[29] G. Alvarez, D. de la Guia, F. Montoya, and A. Peinado. 1996. ”Akelarre: a new block cipher algorithm”. SAC’96, Workshop Record, Queen’s University, Canada, pp: 1-14.

[30] Anderson, Ross, and Eli Biham. 1996. Two Practical and Provably Secure Block Ciphers: BEAR and LION. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 113-20.

[31] Carlisle, Adams. 1997. Constructing of Symmetric Ciphers Using the CAST Design Procedure. Designs, Codes, and Cryptography 12: 283316.

[32] Rijmen, Vincent et al. 1996. The Cipher SHARK. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 99-111.

[33] Kwan, Matthew. 1997. The Design of the ICE Encryption Algorithm. In Proceedings of the 4th International Workshop on Fast Software Encryption, , 69-82. http://dl.acm.org/citation.cfm?id=740726.

[34] Daemen, Joan, Lars Knudsen, and Vincent Rijmen. 1997. The Block Cipher SQUARE. Lecture Notes in Computer Science 1267: 149-65.

[35] D. M'Raihi, D. Naccache, J. Stern, and S. Vaudenay.1997. "XMX: A firmware-oriented block cipher based on modular multiplications". Fast Software Encryption, Springer-Verlag, 1997, pp:166- 171.

[36] Daemen, Joan, and Vincent Rijmen. 2002. New York The Design of Rijndael: AES - The Advanced Encryption Standard. http://portal.acm.org/citation.cfm?id=560131.

[37] H. Heys, C. Adams, S. Tavares, and M. Wiener.1998."CAST256: a submission for the Advanced Encryption Standard". First AES Candidate Conference (AES1), USA, 1998.

[38] Vaudenay, Serge. 1999. On the Security of CS-Cipher. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp: 260-74.

[39] C. Lim. 1998. ”Crypton: a new 128-bit block cipher”. First AES Candidate Conference (AES1), USA.

[40] Lucks, Stefan. 1999. On the Security of the 128-Bit Block Cipher DEAL. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 60-70.

[41] Wu, Wenling, Bao Li, Denguo Feng, and Sihan Qing. 2000. On Decorrelated Fast Cipher. Journal of Electronics (China) 17(1): 94-96. http://dx.doi.org/10.1007/s11767-000-0028-6.

[42] M. Matsui and T. Tokita. 1999. ”Cryptanalysis of a reduced version of the block cipher E2”. 6th International workshop on Fast Software Encryption, Springer Verlag, pp: 71-80.

[43] D. Georgoudis, D. Leroux, and B. Chaves. 1998. ”The FROG encryption algorithm”. First AES Candidate Conference (AES1), USA.

[44] R. Schroeppel and H. Orman, 1998.”Overview of the hasty pudding cipher”. First AES Candidate Conference (AES1), USA.

[45] J. Pieprzyk, and J. Seberry L. Brown. 1998.”Introducing the new LOKI97 block cipher”. First AES Candidate Conference (AES1), USA.

[46] M. Jacobson and K. Huber. 1998. ”The Magenta block cipher algorithm.First AES Candidate Conference (AES1), USA.

[47] Burwick, Carolynn, Don Coppersmith, and Edward DAvignon. 1998. MARS-a Candidate Cipher for AES. NIST AES Proposal. http://www.encryption.de/docs/Mars.pdf.

[48] Rivest, RL, M.J.B. Robshaw, R. Sidney, and Y.L. Yin. 1998. The RC6 Block Cipher. in First Advanced Encryption .

[49] Biham, E, R Anderson, and L Knudsen. 1998. Serpent: A New Block Cipher Proposal. Fast Software Encryption 1372: 222-38.

[50] National Institute of Standards and Technology. (1998) Skipjack and KEA algorithm specifications. http://csrc.nist.gov/CryptoToolkit/skipjack/skipjack.pdf. [Online].

[51] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, and C. Hall. 1998. Twofish: A 128-Bit Block Cipher. NIST AES Proposal 15(1): 1-27. http://www.counterpane.com/twofish-paper.html.

[52] Diffie, Whitfield, and Martin E. Hellman. 1977. Exhaustive Cryptanalysis of the NBS Data Encryption Standard. Computer 10(6): 74-84.

[53] S. Vaudenay and H. Handschuh. 2000. ”A universal encryption standard. Selected Areas in Cryptography, Springer-Verlag, pp: 1-12.

[54] P. Barreto and V. Rijmen. 2000. ”The Khazad legacy-level block cipher”. First Open NESSIE Workshop, Leuven, Belgium.

[55] P. Barreto and V. Rijmen. 2000. Anubis block cipher, http://www.larc.usp.br/pbarreto/AnubisPage.htm.

[56] Aoki, K, T Ichikawa, M Kanda, and M Matsui. 2001. Camellia: A 128-Bit Block Cipher Suitable for Multiple PlatformsDesign andAnalysis. In Proceedings of the 7th Annual International Workshop on Selected Areas in Cryptography, , 39-56. http://www.springerlink.com/index/ytjujxaukr6w0nqd.pdf.

[57] P. Nguyen, F. Noilhan, and S. Vaudenay L. Granboulan. 2001. ”DFCv2”. Selected Areas in Cryptography. Springer-Verlag, pp: 57-71.

[58] J. Borst. 2000. ”The block cipher: GrandCru. First Open NESSIE Workshop, Leuven, Belgium.

[59] Toshiba Corporation. 2000.”Specification on a block cipher: Hierocrypt-L1. First Open NESSIE Workshop, Leuven, Belgium.

[60] ETSI/SAGE. 1999. Kasumi Specification,Part of the Specification of the 3GPP Confidentiality and Integrity Algorithms. http://www.etsi.org.

[61] A. Machado. 2000. ”The Nimbus cipher: a proposal for NESSIE. First Open NESSIE Workshop, Leuven, Belgium.

[62] M. Peeters, G. Van Assche, and V. Rijmen J. Daemen. 2000. ”The Noekeon block cipher”. First Open NESSIE Workshop, Leuven, Belgium.

[63] A. Volchkov and A. Labedev.2000. ”NUSH”. First Open NESSIE Workshop, Leuven, Belgium.

[64] L. McBride. 2000. ”Q: a proposal for NESSIE”. First Open NESSIE Workshop, Leuven, Belgium.

[65] Shimoyama, Takeshi et al. 2002. The Block Cipher SC2000. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 312-27.

[66] Biham, Eli, Orr Dunkelman, and Nathan Keller. 2003. Rectangle Attacks on 49-Round {SHACAL}-1. In Fast Software Encryption – FSE, 22-35.

[67] Bogdanov, A et al. 2007. PRESENT: An Ultra-Lightweight Block Cipher. Springer Berlin Heidelberg: 450-66.

[68] De Cannire, Christophe, Orr Dunkelman, and Miroslav Kneevi. 2009. KATAN and KTANTAN - A Family of Small and Efficient Hardware-Oriented Block Ciphers. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 272-88.

[69] Guo, Jian, Thomas Peyrin, Axel Poschmann, and Matt Robshaw. 2011. The LED Block Cipher. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 326-41.

[70] Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers Ray Beaulieu. 2015. ”The SIMON and SPECK lightweight block ciphers”. 52nd Annual Design Automation Conference (DAC ’15), ACM, pp: 1-6.

[71] Basu, Riddhipratim, Shirshendu Ganguly, Subhamoy Maitra, and Goutam Paul. 2008. A Complete Characterization of the Evolution of RC4 Pseudo Random Generation Algorithm. Journal of Mathematical Cryptology 2(3): 257-89.

[72] Mahdi Madani and Salim Chitroub. 2014. Enhancement of A5/1 Stream Cipher Overcoming its Weakneses. Proceedings of The Tenth International Conference on Wireless and Mobile Communications.

[73] U.Blocher and M.Dichtl. 1994. Fish: A fast software stream cipher. Fast Software encryption, Vol 809, LNCS, pp:41-44.

[74] Pudovkina, M. 2001. Analysis of chosen plaintext attacks on the WAKE stream cipher. Available at: http://citeseer.ist.psu.edu/452427. html., 2001.

[75] RJ Anderson. 1994.On Fibonacci Keystream Generators. Fast Software Encryption, Springer LNCS vol 1008 pp: 346352.

[76] Robert J. Jenkins Jr. 1996. ISAAC. Fast Software Encryption, pp: 41-49.

[77] Software-efficient pseudorandom function and the use thereof for encryption Patent no: US 5454039 A

[78] J. Daemen and C. Clapp. 1998.Fast hashing and stream Encryption with PANAMA. Fast Software Encryption, LNCS 1372, Springer-Verlag, pp: 60-74.

[79] Watanabe, Dai et al. 2004. A New Keystream Generator MUGI. In IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, , 37-45.

[80] Lu, Yi, and Serge Vaudenay. 2008. Cryptanalysis of an E0-like Combiner with Memory. Journal of Cryptology 21(3): 430-57.

[81] Halevi, Shai, Don Coppersmith, and Charanjit S Jutla. 2002. Scream: A Software-Efficient Stream Cipher. In Fast Software Encryption, 9th International Workshop, FSE 2002, Leuven, Belgium, February 4-6, 2002, Revised Papers, , 195-209. http://www.iacr.org/cryptodb/archive/2002/FSE/3065/3065.pdf.

[82] Boesgaard, Martin et al. 2003. Rabbit: A New High-Performance Stream Cipher. Fast Software Encryption: 30729. http://www.springerlink.com/index/M65525J3CKAG208U.pdf.

[83] Ekdahl, Patrik, and Thomas Johansson. 2001. SNOW - a New Stream Cipher. RST open Nessie Workshop, Heverlee, Belgium (Iv): 1-17.

[84] Watanabe, Dai, Soichi Furuya, and Toshinobu Kaneko. 2005. A MAC Forgery Attack on SOBER-128. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E88A(5): 1166-72.

[85] Rose, Gregory G, and Philip Hawkes. 2003. Turing: A Fast Stream Cipher. Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers 2887: 290306. http://www.iacr.org/cryptodb/archive/2003/FSE/3190/3190.pdf.

[86] Canniere, C De, and B P Trivium. 2008. New Stream Cipher Designs. In The eSTREAM Finalists, , 244-66.

[87] Berbain C. O. Billet et al. 2008. Sosemanuk, a Fast Software-Oriented Stream Cipher. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4986 LNCS: 98-118.

[88] Bernstein, Daniel J. 2008. The salsa20 Family of Stream Ciphers. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4986 LNCS: 84-97.

[89] E. Biham and J. Seberry. Roo: a fast and secure stream cipher using rolling arrays," eSTREAM, http://www.ecrypt.eu.org/stream/papersdir/081.pdf.

[90] Whiting, Doug, Bruce Schneier, Stefan Lucks, and Frederic Muller. 2005. Phelix Fast Encryption and Authentication in a Single Cryptographic Primitive. ECRYPT Stream Cipher Project: 1-17. https://www.schneier.com/paper-phelix.html.

[91] Liu, Yunyi, and Tuanfa Qin. 2009. The Key and IV Setup of the Stream Ciphers HC-256 and HC-128. In Proceedings - International Conference on Networks Security, Wireless Communications and Trusted Computing, NSWCTC 2009, , 430-33.

[92] Hell, M., Johansson, T., Meier, W. 2007. Grain A Stream Cipher for Constrained Environments. International Journal of Wireless and Mobile Computing, 2(1): 86-93.

[93] Makoto Matsumoto, Mutsuo Saito, Takuji Nishimura and Mariko Hagita. 2006. CryptMT version 2.0 : A Large State Generator with Faster Initialization. eSTREAM report 2006/023. URL: http://www.ecrypt.eu.org/stream/papers.html.

[94] S. O'Neil, B. Gittins and H. Landman. 2005. "VEST Hardware Dedicated Stream Ciphers". Report 2005/032, eSTREAM - ECRYPT - Stream Cipher Project, http://www.ecrypt.eu.org/stream , 2005.

[95] B. Gammel, R. Gottfert, and O. Kniffler. 2007. Achterbahn-128/80: Design and analysis. SASC2007: Workshop Record of The State of the Art of Stream Ciphers, pp: 152165.

[96] Berbain, Cme, Henri Gilbert, and Jacques Patarin. 2006. QUAD: A Practical Stream Cipher with Provable Security. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 109-28.

[97] Nawaz, Yassir, and Guang Gong. 2008. WG: A Family of Stream Ciphers with Designed Randomness Properties. Information Sciences 178(7): 1903-16.

[98] Cid, Carlos, Shinsaku Kiyomoto, and Jun Kurihara. 2009. The RAKAPOSHI Stream Cipher. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 32-46.

[99] Kitsos, Paris, Nicolas Sklavos, and Athanassios N. Skodras. 2011. An FPGA Implementation of the ZUC Stream Cipher. In Proceedings - 2011 14th Euromicro Conference on Digital System Design: Architectures, Methods and Tools, DSD 2011, , 814-17.

[100] Babbage, S, and M Dodd. 2006. The Stream Cipher MICKEY 2.0. ECRYPT Stream Cipher Project, Report: 1-12.

[101] Rivest R.L., and Schuldt J.C.N. 2014. SpritzA spongy RC4-like stream cipher and hash function. Proceedings of the Charles River Crypto Day, USA.

[102] Dubrova, Elena, and Martin Hell. 2017. Espresso: A Stream Cipher for 5G Wireless Communication Systems. Cryptography and Communications 9(2): 273-89.

[103] E.K. Grossman and B. Tuckerman .1977. "Analysis of a Feistel-like cipher weakened by having no rotating key". IBM Thomas J. Watson Research Report RC 6375.

[104] Biham, Eli, and Adi Shamir. 1991. Differential Cryptanalysis of DES-like Cryptosystems. Journal of Cryptology 4(1): 3-72.

[105] Matsui, M. and Yamagishi, A. 1992. "A new method for known plaintext attack of FEAL cipher". Advances in Cryptology - EUROCRYPT.

[106] Davies, D., and S. Murphy. 1995. Pairs and Triplets of DES S-Boxes. Journal of Cryptology: The Journal of the International Association for Cryptologie Research (IACR) 8(1): 1-25.

[107] Biham, Eli, and Alex Biryukov. 1997. An Improvement of Davies Attack on Des. Journal of Cryptology 10(3): 195-205.

[108] Lipton, Richard J., and Jeffrey F. Naughton. 1993. Clocked Adversaries for Hashing. Algorithmica 9(3): 239-52.

[109] Biham, Eli. 1994. New Types of Cryptanalytic Attacks Using Related Keys. Journal of Cryptology 7(4): 229-46.

[110] Harpes, Carlo, Gerhard G. Kramer, and James L. Massey. 1995. A Generalization of Linear Cryptanalysis and the Applicability of Matsuis Piling-up Lemma. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp:24-38.

[111] C. Harpes and J. Massey. 1997. Partitioning Cryptanalysis. 4th International Workshop in Fast Software Encryption (FSE ’97), Springer-Verlag. pp: 1327.

[112] Kocher and Paul. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. Advances in CryptologyCRYPTO96, Lecture Notes in Computer Science 1109: 104113.

[113] Lars Knudsen and David Wagner. 2001. ”Integral Crytanalysis”. 9th International Workshop on Fast Software Encryption (FSE ’02), Springer-Verlag. pp: 112127.

[114] Jakobsen, Thomas, and Lars R Knudsen. 1997. The Interpolation Attack on Block Ciphers. Building (February 1996): 2840.

[115] Kelsey, John, Tadayoshi Kohno, and Bruce Schneier. 2000. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent (Abstract Only). The Third Advanced Encryption Standard Candidate Conference, April 13–14, 2000, New York, NY, USA: 10.

[116] Kelsey, John, Bruce Schneier, and David Wagner. 1999. Mod N Cryptanalysis, with Applications against RC5P and M6. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 13955.

[117] Kelsey, John, Tadayoshi Kohno, and Bruce Schneier. 2000. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent (Abstract

Only). The Third Advanced Encryption Standard Candidate Conference, April 13–14, 2000, New York, NY, USA: 10.

[118] Eli Biham, Orr Dunkelman, and Nathan Keller. 2001. The Rectangle Attack Rectangling the Serpent. Advances in Cryptology, Proceedings of EUROCRYPT, Springer-Verlag. pp: 340357.

[119] Courtois, Nicolas T., and Josef Pieprzyk. 2002. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 26787.

[120] Khovratovich, Dmitry et al. 2015. Rotational Cryptanalysis of ARX Revisited. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 51936.

[121] Bruce Schneier . 2010. Schneier on Security: New Attack on Threefish.

[122] Van Loan and Charles. 1992. ”Computational frameworks for the Fast Fourier Transform. SIAM.

[123] Pan and Jeng-Shyang. 2009. ”Data Encryption Method using Discrete Fractional Transformation”. US Patent Application: 20090136023.

[124] Michiel Hazewinkel. 1994. ”Affine Transformation”. Encyclopaedia of Mathematics. Springer.

[125] V. Dolotin and A. Morozov. 2008. ”Introduction to non linear algebra”. arXiv:hep-th/0609022v

[126] Wang, Qian, An Wang, Liji Wu, and Jiliang Zhang. 2016. A New Zero Value Attack Combined Fault Sensitivity Analysis on Masked AES. Microprocessors and Microsystems 45: 35562.

[127] Piret, Gilles. 2003. A Differential Fault Attack Technique against SPN Structures, with Application to the AES and KHAZAD.

In Hardware and Embedded Systems-CHES 2003, , 7788. http://www.springerlink.com/index/WQ7JX5HB6XGBU3X7.pdf.

[128] J. Blomer and J.-P. Seifert. 2003. ”Fault based cryptanalysis of the Advanced Encryption Standard”. Financial Cryptography 03, LNCS. Springer.

[129] Dusart, P, G Letourneux, and O Vivolo. 2002. Differential Fault Analysis on A.E.S . Information Sciences 2846(19): 1-10. http://arxiv.org/abs/cs/0301020.

[130] Giraud, Christophe. 2005. DFA on AES. 4th International Conference, AES 2004 (April): 27-41.

[131] Mukhopadhyay, Debdeep. 2009. An Improved Fault Based Attack of the Advanced Encryption Standard. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pp: 421-34.

[132] Kim, Chong Hee. 2010. Differential Fault Analysis against AES-192 and AES-256 with Minimal Faults. In Fault Diagnosis and Tolerance in Cryptography - Proceedings of the 7th International Workshop, FDTC 2010, , 3-9.

[133] Tunstall, M, D Mukhopadhyay, and S Ali. 2011. Differential Fault Analysis of the Advanced Encryption Standard Using a Single Fault. Information Security Theory and Practice. Security and Privacy of Mobile Devices in Wireless Communication: 224-233.

[134] Biryukov, Alex et al. 2010. Key Recovery Attacks of Practical Complexity on AES-256 Variants with up to 10 Rounds. In Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), , 299-319.

[135] Cui, Jie, Liusheng Huang, Hong Zhong, and Wei Yang. 2010. Improved Related-Key Attack on 7-Round AES-128/256. In ICCSE 2010 - 5th International Conference on Computer Science and Education, Final Program and Book of Abstracts, , 462-66.

[136] Barenghi, Alessandro, Guido M. Bertoni, Luca Breveglieri, and Gerardo Pelosi. 2013. A Fault Induction Technique Based on Voltage Underfeeding with Application to Attacks against AES and RSA. Journal of Systems and Software 86(7): 1864-78.

[137] Farhady Ghalaty, Nahid, Bilgiday Yuce, and Patrick Schaumont. 2016. Analyzing the Efficiency of Biased-Fault Based Attacks. IEEE Embedded Systems Letters 8(2): 33-36.

[138] Kang, Jinkeon et al. 2013. Collision Attacks on AES-192/256, Crypton-192/256, mCrypton-96/128, and Anubis. Journal of Applied Mathematics 2013.

[139] Sahmoud, Shaaban. 2013. ”Enhancement the Security of AES Against Modern Attacks by Using Variable Key Block Cipher”. Int. Arab J. e-Technol. 3. pp: 17-26.

[140] Zhao, Xinjie et al. 2013. A Comprehensive Study of Multiple Deductions-Based Algebraic Trace Driven Cache Attacks on AES. Computers and Security 39(PART B): 173-89.

[141] Roetteler, Martin, and Rainer Steinwandt. 2015. A Note on Quantum Related-Key Attacks. Information Processing Letters 115(1): 40-44.

[142] Mestiri, Hassen, Fatma Kahri, Belgacem Bouallegue, and Mohsen Machhout. 2016. A High-Speed AES Design Resistant to Fault Injection Attacks. Microprocessors and Microsystems 41: 47-55.

[143] Patranabis, Sikhar, Abhishek Chakraborty, Debdeep Mukhopadhyay, and Partha Pratim Chakrabarti. 2017. Fault Space Transformation: A Generic Approach to Counter Differential Fault Analysis and Differential Fault Intensity Analysis on AES-Like Block Ciphers. IEEE Transactions on Information Forensics and Security 12(5): 1092-1102.

[144] Poonam Jindal and Brahmjit Singh.2014. “RC4 Encryption-A Literature Survey”. International Conference on Information and Communication Technologies (ICICT 2014), Procedia Computer Science 46, pp:697-705.

[145] Alexander L. Grosul and Dan S. Wallach. “A Related-Key Cryptanalysis of RC4”. Technical report TR-00-358, Department of Compyter Science, Rice University.

[146] Fluhrer S.R. and McGrew D.A. 2000.“Statistical Analysis of the Alleged RC4 Keystream Generator”. Fast Software Encryption, pp:19-30.

[147] Fluhrer S.R., Mantin I. and Shamir A. 2001.“Weaknesses in the Key Scheduling Algorithm of RC4”. International Workshop on Selected Areas in Cryptography, pp:1-24.

[148] Souradyuti Paul and Bart Preneel.2004.“A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher”.International Association for Cryptologic Research, FSE 2004, LNCS 3017, pp. 245-259.

[149] Violeta Tomašević, Slobodan Bojanić, and Octavio Nieto-Taladriz. 2007. ”Finding an internal state of RC4 stream cipher”. Inf. Sci.,177(7):1715-1727.

[150] Biham, Eli and Dunkelman. 2007.“Differential Cryptanalysis in Stream Ciphers”.IACR Cryptology ePrint Archive.

[151] Goutam Paul and Subhamoy Maitra.2007.”Permutation After RC4 Key Scheduling Reveals the Secret Key”. International Workshop on Selected Areas in Cryptography,Selected Areas in Cryptography, pp 360-377.

[152] Matsui M.2009.“Key Collisions of the RC4 Stream Cipher”.In: Dunkelman O. (eds) Fast Software Encryption. Lecture Notes in Computer Science, vol 5665. Springer, Berlin, Heidelberg.

[153] Jiageng Chen and Atsuko Miyaji. 2013.“Novel strategies for searching RC4 key collisions”. Computers and Mathematics with Applications, 6, pp:81-90.

[154] Mohammad Ali Orumiehchiha, Josef Pieprzyk, Elham Shakour, and Ron Steinfeld. 2013."Cryptanalysis of RC4(n, m) stream cipher". In Proceedings of the 6th International Conference on Security of Information and Networks (SIN '13), ACM, pp:165-172.

[155] Santanu Sarkar.2014. "Proving empirical key-correlations in RC4". Information Processing Letters 114 (2014): 234–238.

[156] Jian Xie and Xiaozhong Pan."An improved RC4 stream cipher". International Conference on Computer Application and System Modeling,pp: V7-156-V7-159.

[157] Bhargvi H. Kamble and Dr. B. B. Meshram. 2012."Robustness of RC4 against Differential attack". International Journal of Advanced Research in Computer Engineering & Technology, Vol.14.

[158] T.D.B Weerasinghe. 2013. "An Effective RC4 Stream Cipher". IEEE 8th International Conference on Industrial and Information Systems, ICIIS 2013,pp:18-20.

[159] Poonam Jindal and Brahmjit Singh.2017."Optimization of the Security-Performance Tradeoff in RC4 Encryption Algorithm". Wirel. Pers. Commun. 92(3): 1221-1250.

[160] Ahmed T. Sadiq, Alaa K. Farhan, Shaimaa A. Hassan. 2016. "A proposal to improve rc4 algorithm based on hybrid chaotic maps". Journal of Advanced Computer Science and Technology Research, 6(4), pp:74-81.

[161] Sura M. Searan, Ali M. Sagheer. 2016. "Modification of RC4 Algorithm by using Two State Tables and Initial State Factorial". I. J. Computer Network and Information Security.12. pp:1-8.

[162] W. Stallings, Cryptography and Network Security: Principles and Practices, 2005. doi:10.1007/11935070.

[163] T.W. Cusick, P. Stnic, Cryptographic Boolean Functions and Applications, 2009. doi:10.1016/B978-0-12-374890-4.00012-4.

[164] F. Max, S. Marc, Reverse Engineering of the Cryptanalytic Attack used in the Flame Super-Malware, Advances in Cryptology, 21st International Conference on the Theory and Application of Cryptology and Information Security. (2015) 586-611.

[165] A. Canteaut, M. Videau, Symmetric Boolean Functions, IEEE Trans. Inf. Theory. 51 (2005) 2791-2811.

[166] P. Savicky, On the bent Boolean functions that are symmetric, Europ. J. Combin. 15 (1994) 407-410.

[167] S. Maitra, P. Sarkar, Maximum nonlinearity of symmetric Boolean functions on odd number of variables, IEEE Trans. Inf. Theory. 48(9) (2002) 2626-2630.

[168] S. Picek, D. Jakobovic, J.F. Miller, L. Batina, M. Cupic, Cryptographic Boolean functions: One output, many design criteria, Appl. Soft Comput. J. 40 (2016) 635-653. doi:10.1016/j.asoc.2015.10.066.

[169] X.M. Zhang, Y. Zheng, Cryptographically resilient functions, IEEE Trans. Inf. Theory. 43 (1997) 1740-1747. doi:10.1109/18.623184.

[170] C. Blondeau, K. Nyberg, Perfect nonlinear functions and cryptography, Finite Fields Their Appl. 32 (2015) 120-147. doi:10.1016/j.ffa.2014.10.007.

[171] N. Nisan, A. Wigderson, Hardness vs randomness, J. Comput. Syst. Sci. 49 (1994) 149-167. doi:10.1016/S0022-0000(05)80043-1.

[172] Amandeep, G. Geetha, Analysis of bitsum attack on block ciphers, J. of Disc. Math. Sci. and Crypt. 19(4) (2016) 875-885.

[173] G. Horvth, C.L. Nehaniv, Length of polynomials over finite groups, J. Comput. Syst. Sci. (2015) 614-1622. doi:10.1016/j.jcss.2015.05.002.

[174] A. C. Grove, An introduction to Walsh functions and their applications, Intl. J. of Math. Edu. in Sci. and Tech. 14(1) (1983) 43-53.

[175] P. Feinsilver, J. Kocik, Krawtchouk polynomials and Krawtchouk matrices, in: Recent Adv. Appl. Probab., (2005) 115-141.

[176] L. Comtet, Advanced Combinatorics: The art of finite and infinite expansions. (1974) doi:10.1007/978-94-010-2196-8.

[177] C. J. Mitchell, Enumerating Boolean functions of cryptographic significance, J. Cryptol. 2(3) (1990) 155-170.

[178] Q. Meng, H. , M. Yang, Z. Wang, Analysis of affinely equivalent Boolean functions, Sci. China, Ser. F Inf. Sci. 50 (2007) 299-306. doi:10.1007/s11432-007-0030-9.

[179] T. Siegenthaler, Correlation-immunity of nonlinear combining functions for cryptographic applications (Corresp.), IEEE Trans. Inf. Theory. 30 (1984) 776-780. doi:10.1109/TIT.1984.1056949.

[180] V. Chepyzhov, B. Smeets, On a Fast Correlation Attack on Certain Stream Ciphers, Lect. Notes Comput. Sci. 547 (1991) 176-185.

[181] W.G. Zhang, F.Q. Jiang, D. Tang, Construction of highly nonlinear resilient Boolean functions satisfying strict avalanche criterion, Sci. China Inf. Sci. 57 (2014) 1-6. doi:10.1007/s11432-014-5073-0.

[182] L. Li, Y. Sun, W. Zhang, Construction of balanced Boolean functions with high nonlinearity, good local and global avalanche characteristics, Front. Math. China. 11(2) (2016) 339-352.

[183] Oliveira O. R. B. de. (2011). An Alternative Method for the Undetermined Coefficients and the Annihilator Methods, https://arxiv.org/abs/1110.4425 , retrieved on 21.02.2017.

[184] Armknecht, F. Improving Fast Algebraic Attacks. In Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7 (2004) Revised Papers 65-82.

[185] Courtois N, Meier W. Algebraic attacks on stream ciphers with linear feedback. Advances in Cryptology: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT03), LNCS 2656, (2003) 345-359.

[186] Wang, Q., & Tan, C. H. A new method to construct Boolean functions with good cryptographic properties. Information Processing Letters, (2013) 113 14-16, 567-571. https://doi.org/10.1016/j.ipl.2013.04.017

[187] Sciancalepore S, Piro G, Boggia G, Bianchi G. Public Key Authentication and Key Agreement in IoT Devices with Minimal Airtime Consumption. IEEE Embed Syst Lett 2017 9, 1-4. doi:10.1109/LES.2016.2630729.

[188] Raza S, Seitz L, Sitenkov D, Selander G. S3K: Scalable Security with Symmetric Keys - DTLS Key Establishment for the Internet of Things. IEEE Trans Autom Sci Eng 2016;13:1270-80. doi:10.1109/TASE.2015.2511301.

[189] Rahul S., Geetha G. Symmetric random function generator (SRFG): A novel cryptographic primitive for designing fast and robust algorithms. Chaos, Solitons & Fractals, Volume 2017;104: 371-377.

[190] Wei Y., Hu, Y. Linear-Differential Cryptanalysis for SPN Cipher Structure and AES. Wuhan Univ. J. of Nat. Sci. 2007;12:37. https://doi.org/10.1007/s11859-006-0166-2

[191] Fulong Chen, Yonglong Luo, “Industrial IoT Technologies and Applications”, Second EAI International Conference, Industrial IoT 2017, Wuhu, China, March 25–26, 2017.

[192] Selçuk, A.A. On Probability of Success in Linear and Differential Cryptanalysis, J Cryptol (2008) 21: 131. https://doi.org/10.1007/s00145-007-9013-7.

[193] A. Canteaut and M. Naya-Plasencia. “Correlation attacks on combination generators”, Cryptography and Communications, vol. 4, n. 3-4, pp. 147-171, 2012.

[194] https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf (Accessed on 26-05-2018)

Publications

[1] Rahul Saha, G. Geetha, ”Symmetric random function generator (SRFG): A novel cryptographic primitive for designing fast and robust algorithms,” Chaos, Solitons & Fractals (Elsevier). 104, (2017), 371-377. (SCI/SCIE Indexed with Impact Factor 1.5)

[2] Rahul Saha, G.Geetha, Hye-jin Kim, ”Indentifying Open Research Problems in Cryptography by Surveying Cryptographic Functions and Operations” International Journal of Grid and DIstributed Computing. 10, (2017), 79-98. (Scopus Indexed)

[3] Rahul Saha, G. Geetha, ” On the Propagation and Immunity Characteristics of Symmetric and Balanced Random Function Generator (SRFG)”, Security and Communication Networks. (SCI/SCIE Indexed) [Communicated]

[4] Rahul Saha, G. Geetha, ” RK-AES: An Improved Version of AES using a New Key generation Process with Symmetric Random Function Generator”, IETE Journal of Research (Taylor and Francis). (SCI/SCIE Indexed) [Communicated]

[5] Rahul Saha, G. Geetha, Gulshan Kumar ” MRC4: A Modified RC4 Algorithm using Symmetric Random Function Generator for Improved Cryptographic Features”, Fundamenta Informaticae. (SCI/SCIE Indexed) [Communicated]