S-RADG: A stream cipher RADG cryptography

Article *in* ARPN Journal of Engineering and Applied Sciences · January 2018

DOI: 10.3923/jeasci.2018.2317.2321

CITATION
1

READS
1,431

3 authors:

Salah Albermany
University Of Kufa
74 PUBLICATIONS **140** CITATIONS
SEE PROFILE

Duha Amer
Al-Furat Al-Awsat Technical University
4 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Sawsan Kamal
Al-Nahrain University,College of Science
11 PUBLICATIONS **30** CITATIONS
SEE PROFILE

Republic of Iraq
Ministry of Higher Education and Scientific Research
Al-Nahrain University
College of Science
Department of Computer Science

Improvement for Stream RADG Cipher Automata Algorithms

A Thesis

Submitted to the College of Science/Al-Nahrain University as a partial Fulfillment of the requirements for the Degree of Master of Science in Computer Science.

By:

Duha Amer Mahdi

B.Sc. in Computer Science Dept. / Faculty of Computer Science and Mathematics / University of Kufa

(2013-2014)

Supervised by

Prof. Dr. Salah Abdulhadi Albermany

Dr. Sawsan Kamal Thamer

2018 A.D. 1439 A.H.

بِسْمِ اللّهِ الرَّحْمَنِ الرَّحِيمِ

قَالَ

إِنَّمَا أَشْكُو بَثِّي وَحُزْنِي

إِلَى اللّهِ وَأَعْلَمُ مِنَ اللّهِ

مَا لَا تَعْلَمُونَ ﴿٨٦﴾

صَدَقَ اللهُ العَلِيُّ العَظِيمُ

(سورة يوسف)

Dedication

I would like to dedicate this work to God's Crown and the brother of messenger Mohammed, princess of believers **Al-Imam Ali (as)**, who is my way to God

To the man that was his dream to see me at the top of the mattress, **My Dad**.

To the woman that give me affection and love, **My Mom**.

To the spirit of **my grandmother** Allah bless her soul.

To my lovers, my three sisters, my **Aunt** and my cousin **"Amel".**

To my life, my two brothers.

To everyone who helped me and encouraged me to finish this work.

Acknowledgements

I thank God for his infinite grace, including Almighty to complete this work. Peace and blessings are upon to the Messenger Mohammed and his holy progeny.

*I would like to express my gratitude to my supervisor "**Professor Dr. Salah Abdulhadi Albermany** ", who honored me with his acceptance of the supervision of this thesis and his support, his guidance a valuable and for teaching me how to be a successful person. God bless him. My second supervisor "**Dr.Sawsan Kamal Thamer**" also made many helpful suggestions, many thanks.*

*I thank my uncle "**Professor Dr. Bayan.M.Sabbar** ", who was the reason after God in my acceptance of the master study. "Thanks Dad".*

My family , they have always supported my dreams and aspirations, I'd like to thank them for all they are.

*I thank my friend and my sister "**Baneen Khalid**" for her permanent moral support.*

Duha

Summary

Web has enormous number of digital information, so making sensitive information accessible on the web requires a massive efforts in security controls and checking of access to the contents, to achieve this many cryptographic algorithms are introduced.

This thesis introduces two algorithms to protect the data transmitted through networks by developing existing keyless algorithm called RADG into S-RADG (Stream RADG {Reaction Automata Direct Graph}) algorithm. Where RADG is algorithm depend on graph theory and do not require any key to encrypt data depending on randomness and original message in the encryption. The new algorithm (S-RADG algorithm) designed based on the same characteristics of RADG (graph theory and randomness encryption) and depend on stream cipher to generates key .A new algorithm encrypts one bit at time like stream cipher.

The developed algorithm deals with data like a block cipher cryptography, so it encrypts a large size of data. The permutation process in Permutation S-RADG algorithm make it stronger and increases the randomness making the algorithm complicate, hard to break by hacker and need more work during decryption process to return the plaintext. The key of two algorithms is generated randomly by using one of stream cipher algorithm, which is LFSR (Linear Feedback Shift Register).

The results of S-RADG and Permutation S-RADG algorithms are different ciphertexts with same plaintext, since depended on randomness encryption (RADG algorithm).

The ciphertexts differ significantly a lot, making the process of attacking within the large systems very difficult compared to the schemes that rely on

the classical cryptography. As compared to RADG method, the proposed algorithms are designed for security purpose with highly efficient data encryption and the analysis for the two proposed algorithms proved terms of confidentiality and data integrity and authentication.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

Abbreviation	Meaning
AES	Advanced Encryption Standard
BRADG	Block Cipher Reaction Automata Direct Graph
DES	Data Encryption Standard
LFSR	Linear Feedback Shift Register
IT	Information Technology
MAC	Message Authentication Code
MRADG	Multi Reaction Automata Direct Graph
RADG	Reaction Automata Direct Graph
RBC	Random Block Cipher
RC4	Rivest Cipher 4
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
S-RADG	Stream Reaction Automata Direct Graph cipher
TPA	Third Auditor Party
WNs	Wireless Networks

List of Symbols

Symbol	Description
addr	Address of S-RADG and Permutation S-RADG design
B_i	Block of S-RADG algorithm
C	Cipher Text
I	Index of S-RADG state
J	Jump set of RADG
K	Key of encryption process and decryption process
k	Size of jump set
L	Length of stages of LFSR method
M	The message to be encrypted
m	Size of reaction states
n	Size of standard states
P_j	The value after permutation
P_n	Number of permutation
Q	Standard set of RADG
R	Reaction set of RADG
r_i	The result after XOR in Permutation S-RADG method
S_n	The sequences of the output of the LFSR
T	Transition function of RADG
V	State value of S-RADG
x	Size of input block
Y_i	The initial value of primitive polynomial
Σ	Input data
Ψ	Output data
Ω	Number of states distributions in reaction partitions

λ	Number of Values in the State

Chapter One

Introduction and General Review

Chapter One: Introduction and General Review

1.1 Introduction

The development of information technology and the rapid growth of computer networks allowed large files to be easily transmitted in open networks such as the internet. This information may exposed to attacks to theft sensitive data or modified for this reason require the protection of information.

Cryptography is the science of using mathematics to encrypt and decrypt data, and thus it provides a way to store sensitive information or transmit it across insecure networks such as the internet, so that it cannot be read by anyone except the intended recipient.

1.2 Literature Survey

This section represents a study of previously existing work that related to the stream cipher and the current work.

Patrik Ekdahl and Thomas Johansson (2003) proposed a new version of SNOW, called SNOW 2.0. The new version of the cipher does not only appear to be more secure, but its implementation is also a bit faster in software where SNOW 1.0, SNOW 2.0, and SNOW 3G are word-based synchronous stream ciphers developed by Thomas Johansson and Patrik Ekdahl . They found the implementation of SNOW 2.0 is easier and encryption is faster than SNOW 1.0. A complete description of SNOW 2.0 was given and the design differences from SNOW 1.0 and how they apply to the known attacks were discussed. Some implementation aspects of the new design were discussed, in particular how to get a fast implementation of the LFSR and the S-box [Pat3].

Hossam El-din H et al (2005) this paper presented an efficient chaos based feedback stream cipher (ECBFSC) for image cryptosystems. The proposed

stream cipher is based on the use of a chaotic logistic map and an external secret key of 256-bit. A new features of the proposed stream cipher include the heavy use of data-dependent iterations, data-dependent inputs, and the inclusion of three independent feedback mechanisms. These proposed features are verified to provide high security level. According to the results of the security analysis, they concluded that the proposed ECBFSC is expected to be useful for real-time image encryption and transmission applications [Hos7].

Salah Albermany and Ghazanfar A. Safdar (2014) introduced a new method in security that not depends on keys cryptography, which is keyless cryptography method that called Reaction Automata Direction Graph (RADG). This method based on automata direct graph and reaction states. RADG method gives different ciphertext with same plaintext. The mathematical model and implementation of RADG algorithm explained by providing example of encryption and decryption. The security elements confidentiality, authentication, non-repudiation and integrity achieved by the RADG security analysis. Proven validity of RADG method by hamming distance to measure certainty of the algorithms. RADG method needs more process to hack a ciphertext within large system [Sal14].

Aissa Belmeguena et al (2015) proposed design consists of a 128-bit nonlinear feedback shift register (NLFSR), a 128-bit linear feedback shift register (LFSR) and a Boolean function. The developed program is used to transform the original speech data wav file into positive data, and then transform the positive data into positive digital data file. Finally, used implemented program to encrypt and decrypt the speech data. The proposed scheme is compared to a similar one of Grain-128. The results show that the proposed scheme was secure and the encrypted speech was very different than the original speech and no significant information on the original speech

could be retrieve and the results given by the modified Grain-128 scheme are better than the results given by the original Grain-128 [Ais15].

Salah Albermany and Ali Hasan Alwan (2016) improved RADG algorithm to be more secure in Wide region systems, for example, subjective Radio System "CRN" (Cognitive Radio Network). The improved algorithm is called MRADG algorithm(Multi Reaction Automata Direct Graph) .The reason for this paper is to build up the RADG configuration to be more secure in bigger system. By using RSA algorithm as a transition function in RADG and change the ciphertext into point in particular elliptic curve, with applied pseudorandom generated key [Sal16].

Salah Albermany et al.(2017) proposed an another algorithm based on RADG technique called Random Block Cipher (RBC). A new secret key comprises of a 64-bit block size and a 128-bit key length were generated. The new algorithm uses a new S-boxes generated from DES S-boxes and uses a bijective function, bitwise operations and a carefully key schedule design. The new algorithm are used in wireless network, RBC depends on uses session keys to perform requirements security privacy and data integrity. RBC algorithm is very efficient large data and can gives more than differ ciphertext for the same plaintext and this is proving that the process of breaking code within the large system will be very difficult [Sal17].

Ali H. Kashmar and Eddie S. Ismail (2017) presented the design of an improved efficient and secure stream cipher called Blostream, which is more secure than conventional stream ciphers that use XOR for mixing. The proposed cipher comprises two major components: the Pseudo Random Number Generator (PRNG) using the Rabbit algorithm and a nonlinear invertible round function (combiner) for encryption and decryption. A complete description of the algorithm, evaluation of its performance in team of security properties, some advantages and disadvantages and comparison of

the proposed cipher with similar systems were presented The proposed cipher depicts some disadvantages, for instance, in case an error occurs during a ciphertext transformation, the round function combiner may address the wrong element; this operation affects subsequent parts of the message. Blostream revealed considerable resistance against a number of possible attacks, including brute-force attack, statistical attacks, deferential attacks, distinguishing and correlations, which were applied to the proposed cipher [Ali17].

1.3 Aim of Thesis

The main objective of this thesis is to design and generate random cipher efficient to provide secure data transition through networks. two new The role of the design can be summarized as the following steps:

1. Redesign RADG algorithm by combining RADG algorithm with stream cipher to generate random key by using LFSR.
2. Achieve a wide secure data transmitted through network between more than two communicating nodes.

1.4 Thesis Layout

The remaining chapters of the thesis are organized as the following:

- **Chapter Two:** shows the theoretical background of stream cipher , block cipher and RADG cryptography.
- **Chapter Three:** introduces the proposed algorithms and clarifies the two algorithms and explains how the encryption and decryption of these algorithms are work.
- **Chapter Four:** presents the performance and security analysis of the proposed design.
- **Chapter Five:** presents the conclusion and future work

Chapter Two

Theoretical Background

Chapter Two: Theoretical Background

2.1 Introduction

Cryptography is an essential tool for security. Previously, the purpose of the cryptography is to hide text messages during the war. But recently, becomes necessary to secure the transfer of information between online networks with complete secrecy .Cryptography is the science that used for encryption and decryption, so that the information is secured and the phenomenon for the sender and the receiver only. Cryptography goals are to satisfy confidentiality, integrity and authentication. In the modern cryptography, there are two types, symmetric key cryptography which uses one key for encryption and decryption, and asymmetric key (public key) cryptography which uses two different keys, one for encryption is called public key, another for decryption is called private key [Paa10, Suk12], as shown following in Figure 2.1.

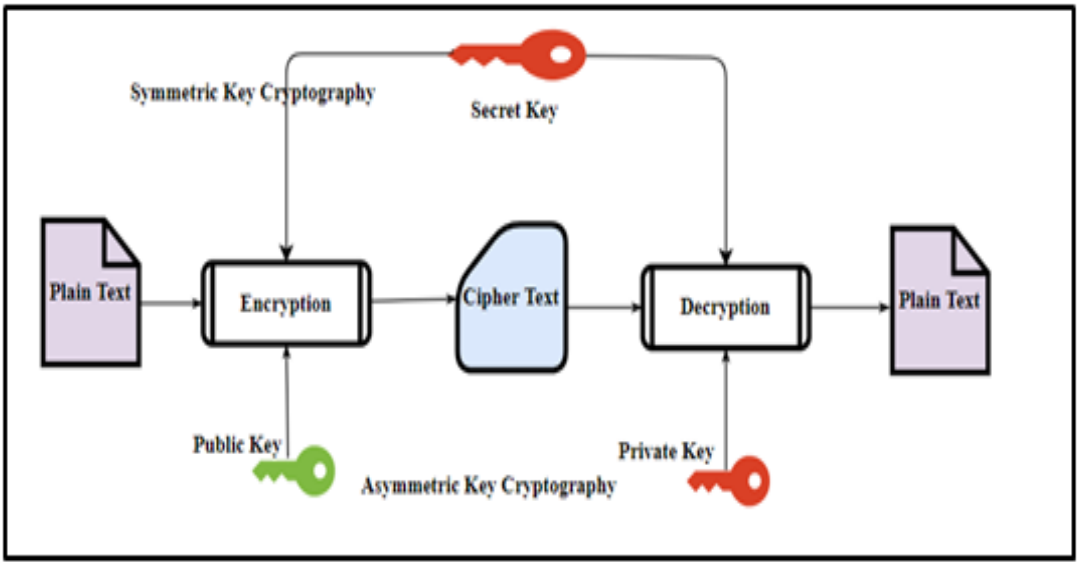


Figure 2.1 Symmetric and Asymmetric key Cryptography Scheme [Paa10]

In a symmetric key, there are two types of ciphering, stream cipher and block cipher [Paa10]. As following in Figure 2.2.

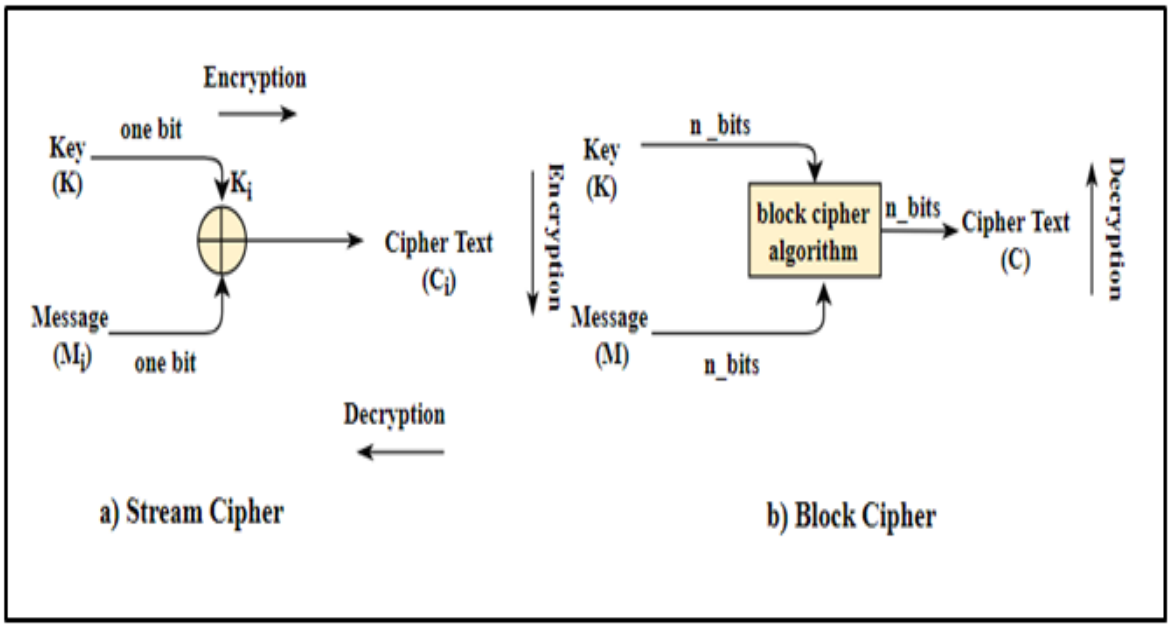


Figure 2.2 Encryption/Decryption in Stream Cipher and Block Cipher [Paa10]

2.2 Reaction Automata Direct Graph (RADG)

RADG is an algorithm that depends on reaction states and direct graph, which is keyless algorithm cryptography, this means, it is not used a key during encryption and decryption and there is no agreement between two parties of communication [Sal14].

RADG is represented by set of tuples $\{R, Q, \Sigma, \Psi, J, T\}$, where R is reaction set that have m of length, which have λ of values for each element, Q is standard design set that have n length, also have λ of values, Σ represents input data (or alphabet), Ψ represents output transition, J is jump set which is subset of Q set, that have k length, which is have no value, just transmit from one state to another in Q set and T represents transition function [Sal14]. The encryption process in RADG algorithm is begins by selecting a random state from Q set, if Q state transmitted to J state then select a new random state to continue encryption process [Sal14]. Example 2.1 illustrates how RADG method is working.

Example 2.1:

If $m = 2$, $n = 4, k = 1$ and suppose $\lambda = 2$, (this mean the number of value in each state is 2). Figure 2.3 illustrates the transitions of states of this example.

The state in this example have two values only, each state has two transitions to another state. The state in RADG design consists from three tuples, the first tuple is state address, and the two other are values of state. The state number of 4 and 5 are in R sets (only two states since $m = 2$) and the state number 0, 1, 2 and 3 are Q sets (since $n = 4$) and the state number 3 is J set in Q sets (since $k = 1$), the input data set as $\Sigma = \{0,1\}$, and the output transition as $\Psi = \{2, 5, 6, 16, 12, 13, 20, 25, 40\}$.

Suppose that the message or plaintext to be encrypted by using RADG is 0111. The transition function represented as T, where T take two parameters state address and message bit, this function represented as the following form:

$$T(\text{address state}, \text{message bit})$$

The encryption by using RADG method begins by selecting random state, in this example, started in 0 state, and then we have:

T (0, 0) = (1, 5)

T (1, 1) = (2, 2)

T (2, 2) = (0,12)

T (0, 1) = (3,16)

The ciphertext is 5, 2, 12 and 16 according to example in Figure 2.3

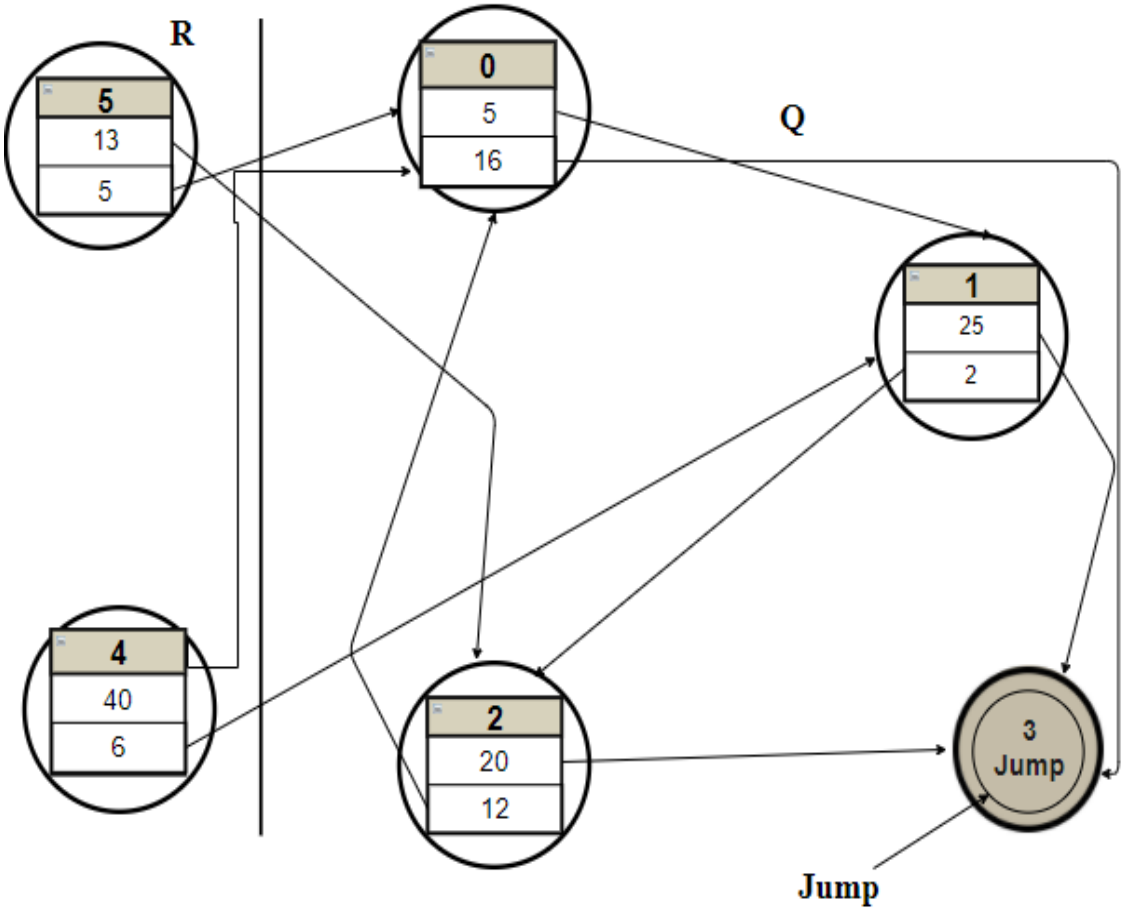


Figure 2.3 Reaction Automata Direct Graph Design Example

The following step to explain the results of example.

T (0, 0): The first value is address state which is *0*and the second value is the first bit message which is 0.

(1, 5): The first value is the state number of the previous state that transmitted to it according to message bit (in the state 0, the input 0 transmit to state 1) and the second value is the first value of the state 0 according to first message bit =0.

T (1, 1): The first value is address state that transmitted to it previous state, which is *1*and the second value, is the second bit message which is 1.

(2, 2): The first value is the state number of the previous state that transmitted to it according to message bit (in the state 1, the input 1 transmit to state 2)

and the second value is the second value of the state 2 according to the second message bit =0.

T (2, 1): The first value is address state which is *2*and the second value is the third bit message which is 1.

(0, 12): The first value is the state number of the previous state that transmitted to it according to the message bit (in the state 2, the input 1 transmit to state 0) and the second value is the second value of the state 2 according to the third message bit =0.

T (0, 1): The first value is address state which is *0*and the second value is the bit message which is 0

(3, 16): The first value is the state number of the previous state that transmitted to it according to the message bit (in the state 0, the input 1 transmit to state 3) and the second value is the second value of the state 3 according to the fourth message bit =1.

2.3 Symmetric key Stream cipher

Stream cipher started in using in 1917 by Gilbert Vernam, where it is designed by encrypting one bit at time, where used same key in encryption and decryption. The encryption done by adding one bit from message to one bit from keystream and continue until the message bits are finished.

Encryption: $\mathbf{C_i = M_i \oplus K_i}$

Decryption: $\mathbf{M_i = C_i \oplus K_i}$

Stream cipher can be categorized into synchronizing stream cipher and asynchronizing stream cipher [M.J.B95].

2.3.1 Synchronizing Stream Cipher

In a synchronizing stream cipher, the keystream is generated randomly in separate of plaintext, and then the keystream is XORed with plaintext in encryption process or XORed with ciphertext in decryption process. The synchronizing cipher does not result error propagation, this is one of its advantages. Because of this, it limits the presence of an error during decryption, although that, Truder can has simple effects on the encrypted text and he fully aware of what these changes are to the plaintext. The structure of the synchronizing stream cipher is shown in the Figure 2.4 [M.J.B95].

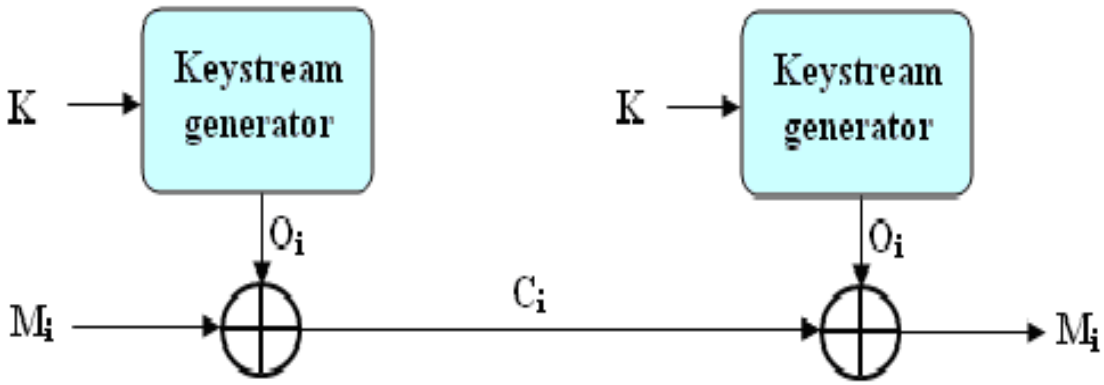


Figure 2.4 Synchronizing Stream Cipher Structure

2.3.2 Asynchronizing Stream Cipher

An asynchronizing stream cipher is opposite of a synchronizing stream cipher in which the computed keystream depends on previous ciphertext bits it is also called self-synchronizing or CipherText Auto Key (CTAK). A self-synchronizing carries out some error propagation, where any change made by the attacks on the plaintext will produce, an effect on all parts of ciphertext that result from the decryption .This type also has disadvantages where the adversary is aware of some variables that represent as inputs for the process which are taken from encrypted text, the structure of the self-synchronizing stream cipher is shown in the Figure 2.5 [M.J.B95].

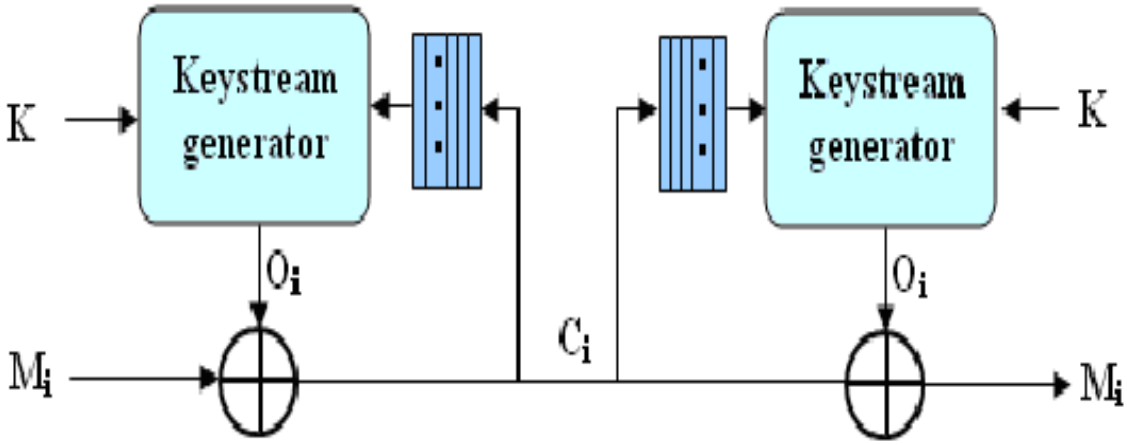


Figure 2.5 Self-Synchronizing Stream Cipher Scheme

2.4 Symmetric Key Block cipher

A block cipher is a techniques of cryptography system, in which cryptographic key and plaintext of the block cipher algorithms are applied on block of data at once rather than one bit. There are many block cipher algorithms like DES algorithm and AES algorithm [Jaw11].

2.4.1 Data Encryption Standard Algorithm (DES)

DES is block cipher, it was introduced by IBM depended on their Lucifer cipher and became used in 1974. DES algorithm encrypts 64 bits at time with 56 bits of key to result 64 bits as a ciphertext. The cipher key is normally given as a 64-bits key (in which 8 extra bits are the parity bits, which are dropped before the actual key-generation process), so if the data size to be encrypted is greater than 64 bits, the DES will encrypts the first 8 blocks then return to encrypt other blocks, so if the other blocks is less than 8 blocks, then extended it by using padding. The DES encrypts the first 64-bits then repeats the work for other 64-bits [Jaw11].

2.4.2 Advanced Encryption Algorithm (AES)

Also known as the Rijndael algorithm, is a symmetric block cipher that can encrypt data blocks of 128 bits using symmetric keys 128, 192, or 256. AES was introduced to replace the DES. Brute force attack is the only effective attack known against this algorithm [Jaw11].

2.5 Linear Feedback Shift Register

Linear feedback shift register is a stream cipher generate random sequences, it applied in hardware and others but surely, not at all.

LFSR is linear function, in equation (2.1) [Paa10, Kle13].

$$f(\vec{S}) = \sum_{i=0}^{n-1} Y_i S_i \qquad \textbf{(2.1)}$$

The equation (2.1) used to get the sequences of LSFR. The sequence of inner state is S_i, it is shifted one bit to right, and the rightmost is the output. The output is computed by XOR for summation of previous stage.

When the initial value generated (seed), and then it is used to determine the output. The input sequence with length n is $S_n, S_{n-1,} \ldots . S_0$. Equation (2.2) is used for linear function to determine the output is [Kle13]:

$$S_n + L = \sum_{i=0}^{L-1} Y_i S_n \ , \quad \forall n \geq 0 \qquad \textbf{(2.2)}$$

Table 2.1 Linear Feedback Shift Register Implement Notations

Notations	Details
L	The length of the stages
S_n	The sequences of the output
Y_i	The initial value

Table 2.1 explains the coefficients of equation 2.2. The number of stage known by the primitive polynomial, where number of stage equal to polynomial degree.

Supposed LFSR with degree d = 4, Figure 2.6 shows the path.

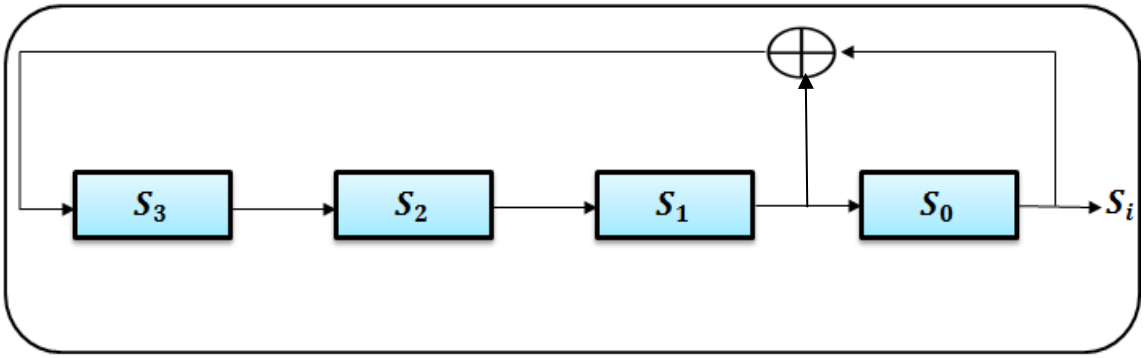


Figure 2.6 Linear Feedback Shift Register with Degree Four

Suppose seed = 1001, Table 2.2 illustrate how LFSR is generated.

Table 2.2 Stages of Linear Feedback Shift Register

t	S_3	S_2	S_1	S_0
0	1	0	0	1
1	0	1	0	0
2	0	0	1	0
3	1	0	0	1

2.5.1 The Berlekamp-Massey (BM) Algorithm

The BM algorithm determines a linear recurrence of least order L ≥ 0 which generates a given (finite) sequence s of length n ≥ 1. It is widely used in Coding Theory, Cryptography and Symbolic Computation. The applications and implementation of this algorithm were advanced and extended by Massey who used the physical interpretation of a linear feedback

shift register (LFSR) as a tool to better understand the algorithm. What the variation does is synthesize LFSR's that have a specified output sequence. This physical interpretation of LFSR's provides a physical explanation of the length of the encoded message needed to be able to decode it using the algorithm. The length of message needed is only twice the length of the LFSR used or 2n [Nor10].

- **Algorithm 2.1 The Berlekamp-Massey (BM) Algorithm**

// Given binary sequence $s = (s_0, s_1, s_2, \ldots, s_{n-1})$
// Find linear complexity L and connection polynomial $C(x)$
BM(s)
    $C(x) = B(x) = 1$
    $L = N = 0$
    $m = -1$
    while $N < n$ // n is length of input sequence
        $d = s_N \oplus c_1 s_{N-1} \oplus c_2 s_{N-2} \oplus \cdots \oplus c_L s_{N-L}$
        if $d == 1$ **then**
            $T(x) = C(x)$
            $C(x) = C(x) + B(x)x^{N-m}$
            if $L \leq N/2$ **then**
                $L = N + 1 - L$
                $m = N$
                $B(x) = T(x)$
            end if
        end if
        $N = N + 1$
    end while
    return(L)
end BM

Chapter Three

The Practical Design

Chapter 3: The Practical Design

3.1 Introduction

This chapter presents the design of implementation of the proposed algorithms for security of data. The two algorithms depend in their work on the principles of RADG algorithm. They use a key that is generated by a stream cipher (LFSR is used), i.e., the result of the stream cipher is used as a key for the new algorithms.

The first algorithm is S-RADG (Stream cipher RADG (Reaction Automata Direct Graph)) that deals with data as stream cipher, one bit at time. The second algorithm is Permutation S-RADG that deals with data as a block cipher, to encrypt a large size of data in comparison with S-RADG. The result of the encryption phase of two algorithms is different for the same plaintext, this is because they are depending on randomness concept during encryption process (since they are depending on the RADG properties). The two algorithms are a new addition to cryptography algorithm to increase security efficiency. Figure (3.1) illustrates the proposed cryptosystem.

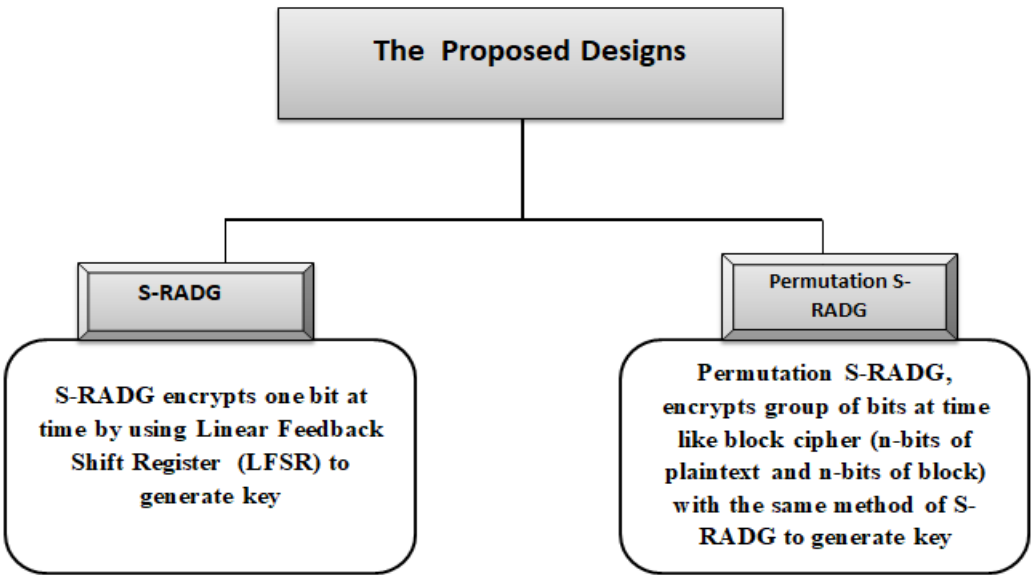


Figure 3.1 the structure of the proposed method

3.2 Stream cipher Reaction State Automata Direct Graph (S-RADG)

This algorithm generates the key by using LFSR method. Then process the data with the generated key bit by bit to produce the ciphertext as shown in Figure 3.2.

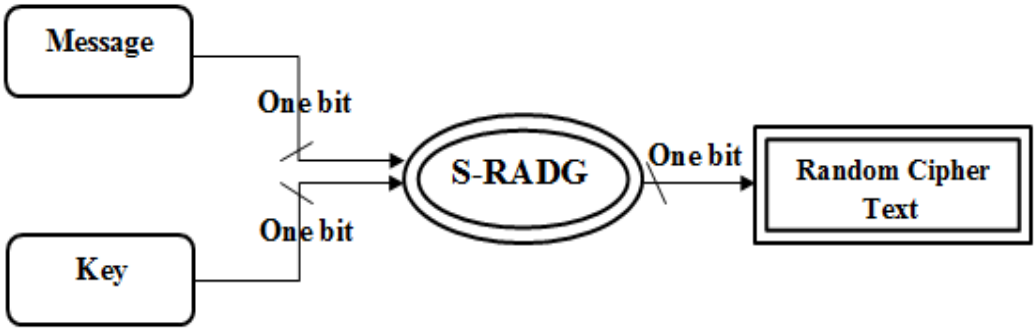


Figure 3.2 Stream cipher Reaction State Automata Direct Graph Algorithm

The S-RADG algorithm is based on RADG mathematical model; it keeps the same features and characteristics of the RADG algorithm with a stream cipher key.

In S-RADG, the transitions for ciphering all bits of plaintext will depend on an address for standard states Q and for reaction states R, denoted by the symbol addr, to determine each state during transitions. The size of the address (addr) of states can be calculated by the equation (3.1), where m represents a number of the elements in set Q and n represents a number of the elements in set R.

The address must be positive integer number, then used ⌈ ⌉ to convert real number to non-integer number by using the equation (3.2), where ⌈ ⌉ symbol represent a celling function.

$$\text{addr} = \lceil \log_2(n + m) \rceil \quad \textbf{(3.2)}$$

If n = 4, m =7.

The number of bits for each address of S-RADG in design is 4.

$$addr = \lceil \log_2(n + m) \rceil$$

$$addr = \lceil \log_2(4 + 7) \rceil$$

$$addr = \lceil \log_2 11 \rceil = 4$$

The S-RADG operations are performed on a one bit at time. The S-RADG algorithm encrypts each bit using RADG state selected randomly, where obtained ciphering values from transmit state to another. These values are used as plain text to DES algorithm by using the same S-RADG key which generated by LFSR method.

To determine the location of value (index of value) by equation (3.3),

$$M_i \oplus K_i = I_i \quad \textbf{(3.3)}$$

Where M_i is the message to be encrypted (one bit is encrypted at time), where $i = 1, \ldots, L$. L is a length of M. K_i is the key that exchanged between two them, I_i which is represents the location of the value.

Then calculate the block (B) by the equation (3.4).

$$B_i = I_i || V || addr \quad \textbf{(3.4)}$$

Where B_i is represent the plaintext of the DES algorithm, where the first bit is I_i =location of the state, | | represents concatenation, V is a value in a state, and addr which represents address of state (address number).

Then

$$C_i = \text{DES encryption } (B_i, K_i) \quad \textbf{(3.5)}$$

Where C_i represent the encrypted data (cipher text), K_i represent the key of DES algorithm which is the same key o S-RADG algorithm.

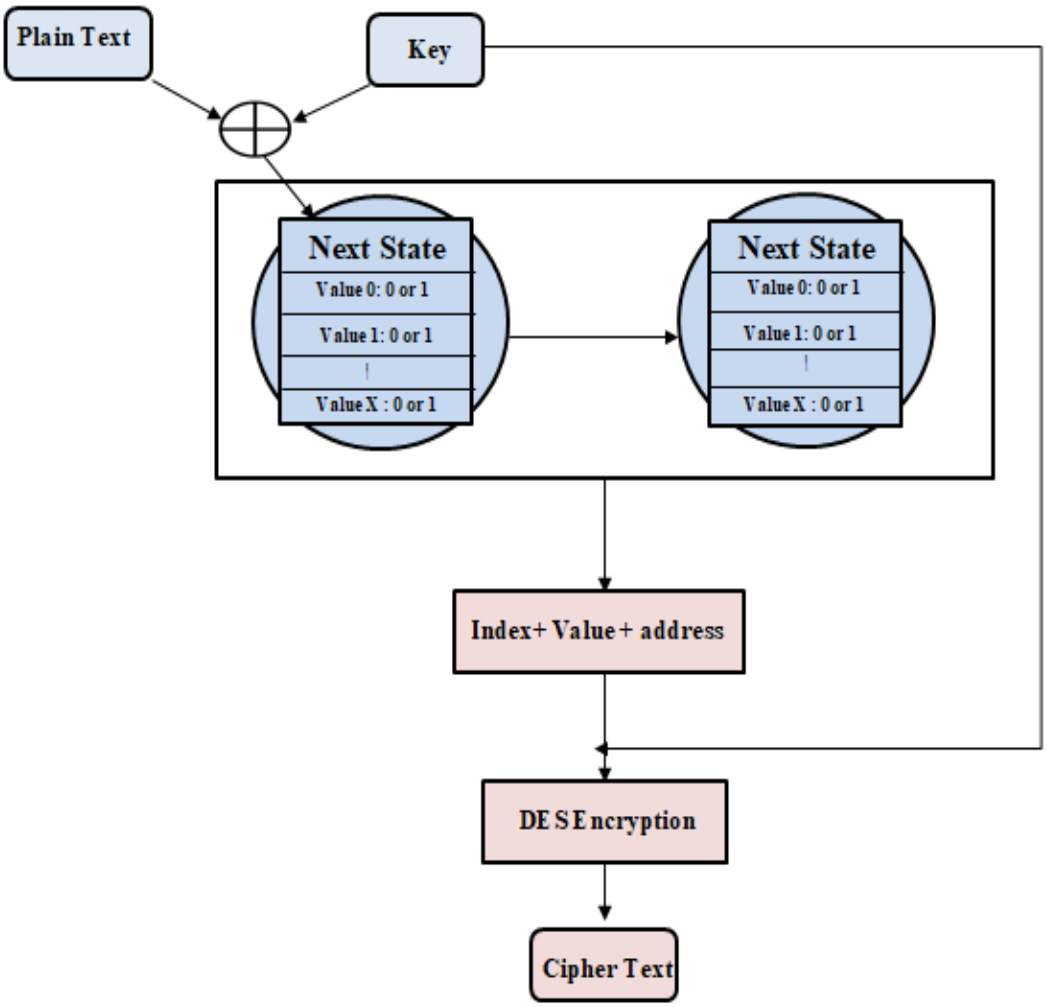


Figure 3.3 General Structure of Stream cipher Reaction State Automata Direct Graph Algorithm

3.2.1 S-RADG Implementation

This section provides algorithms of S-RADG encryption and decryption processes.

3.2.1.1 S-RADG Encryption Algorithm

In the encryption process using S-RADG algorithm, the message length starts from 0 to M where M represent the last bit of the message. Each bit from message with the relative bit from key by certain function to determine the value of selected random state. Algorithm 3.1 illustrates the encryption process.

- **Algorithm 3.1 Algorithm For Stream cipher Reaction State Automata Direct Graph Encryption**

Input: The message $M = \{m_0, m_1, \dots m_{|L|}\}$, where the key $K = \{K_0, K_1 \dots . K_{|L|}\}$

Output: The cipher text $C \leftarrow \{C_0, C_1, \dots .\}$

Steps

1. $C \leftarrow \emptyset$, $q_{current} = q_{rand}$, $L \leftarrow 0$
2. while $(L < |M|)$
3. $I \leftarrow (m_L \oplus K_L)$
4. $(V)_L \leftarrow \text{getvalue}(q_{current}, \text{index})$
5. $b_n \leftarrow (I, V, q_{current})$
6. $C_1 \leftarrow \text{DES}[b_n]$
7. Add (C_{L-1}, C_L)
8. if$(q_{current} \in \text{Jump})$

   $q_{new} \leftarrow q_r, \quad q_r \in R$

   $L = L + 1$
9. else

10. $q_{new} \leftarrow q_{current}$
11. End while
12. return C

3.2.1.2 S-RADG Decryption Algorithm

The decryption algorithm process is based on the same structure of the encryption process. It begins from the end of encryption, where it takes the ciphertext and the key as the input to DES algorithm to return the plain text as the output in the Algorithm 3.2.

- **Algorithm 3.2 Algorithm For Stream cipher Reaction State Automata Direct Graph Decryption**

Input: the cipher text $C \leftarrow \{C_0, C_1, \ldots., C_{|L|}\}$, the key $K \leftarrow \{K_0, K_1, \ldots., K_{|C|}\}$

Output: the plain text $M \leftarrow \{M_0, M_1, \ldots..\}$

Steps

1. $b_L \leftarrow \text{DESdecipher}[C_L]$, $L \leftarrow |C|$
2. Search (q_{old}, Q)
3. while $(L \leq= C)$
4. $m_L \leftarrow (I \oplus K_L)$
5. $\text{Add}(M, m_L)$
6. reverse sequence$\left(M_0, M_1, \ldots. M_{|m|}\right)$
7. return M

3.3 Permutation S-RADG Algorithm

Permutation Stream Cipher Reaction Automata Direct Graph method (Permutation S-RADG method) is a type of symmetric key block cipher algorithm which uses a similar key in cipher and decipher. Permutation S-RADG is based on permutation ciphers.

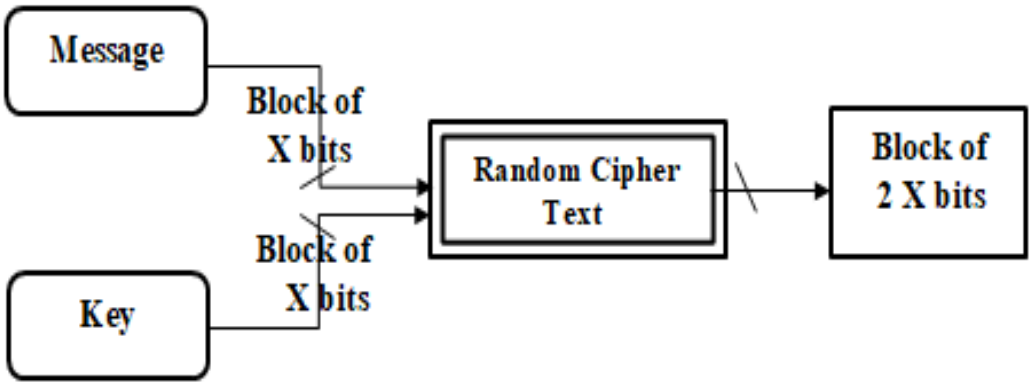


Figure 3.4 Permutation Stream cipher Reaction State Automata Direct Graph Algorithm

Basically the permutation S-RADG algorithm is derived from an existing algorithm called RADG algorithm (Reaction Automata Direct Graph) [Sal14]. Permutation S-RADG divides the plaintext into sub blocks of size($M_0, M_1, \ldots, M_{|B|/x}$). Also, it divides the key after generating into sub blocks of x bits size($K_0, K_1, \ldots, K_{|B|/x}$), where |B|/x represents the number sub blocks. Then the message is XORed with the key in equation (3.8).

$$(M_i \oplus K_i) = \text{Cipher}_{1_i} \ , i = 0,1, \ldots, B-1 \qquad \textbf{(3.8)}$$

Encryption starts by selecting a random state to be called initial state, then Cipher_{1_i} used as an input to the states, where in each state there are the state values which are represented by v_h and the permutation values which are represented by P_j are stored in the state. The cipher value after permutation (P_j) is used with the address (addr) as an input to DES algorithm. (See Figure 3.6.)

The transitions through states of the designed algorithm based on the transition function in the example.

The key of Permutation S-RADG algorithm is generated by LFSR as mentioned in section (3.2.5.1). The algorithm generates blocks of key, where each sub blocks must be equal to sub blocks of message.

$$\left| K_{|B|/x} \right| = \left| M_{|B|/x} \right|.$$

$$K_i \leftarrow K_{B_i}$$

The Permutation S-RADG algorithm uses at least key of size 128 bits at least when used DES algorithm during it work, if AES algorithm is used instead of DES algorithm, the key size will be 196 bits or 256 bits.

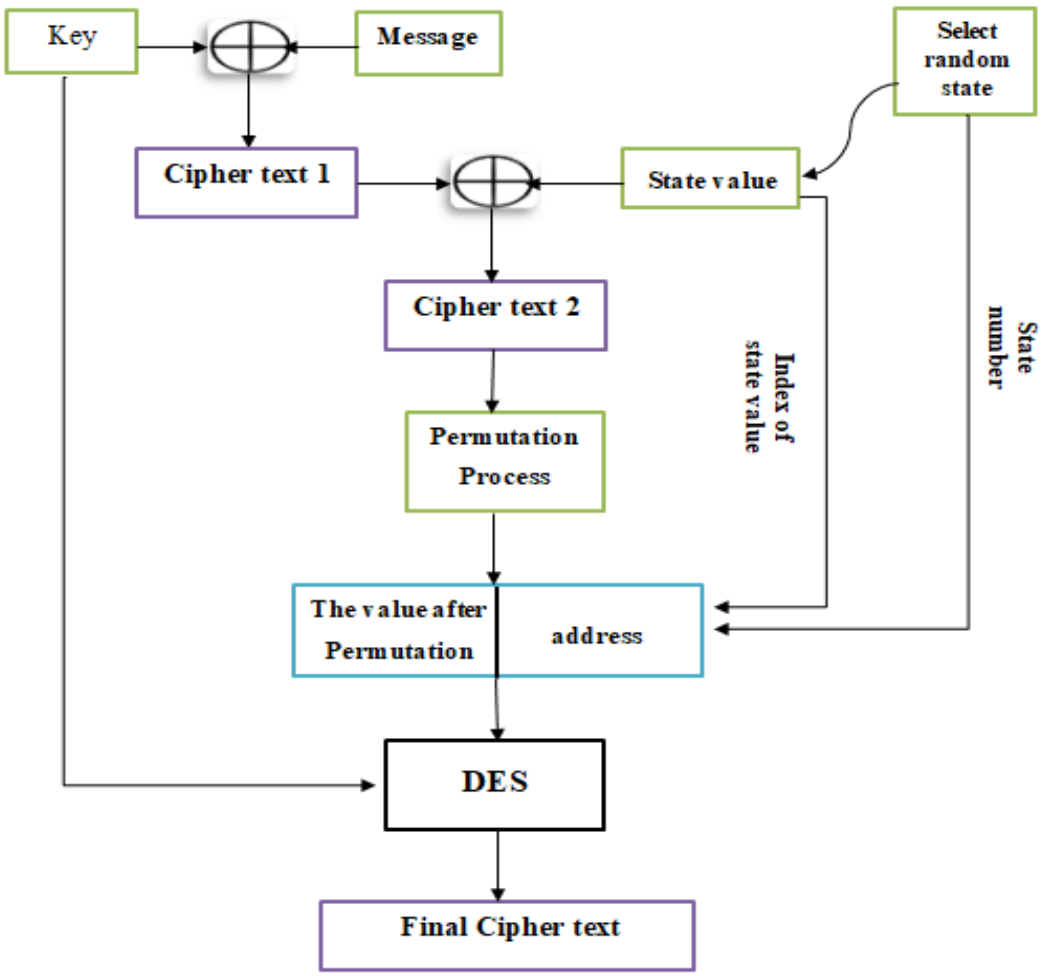


Figure 3.5 The General Structure of Permutation Stream cipher Reaction State Automata Direct Graph Algorithm

Permutation S-RADG method designed algorithm as RADG method consists of 6 tuples $\{Q, R, \Sigma, \Psi, J, T\}$.The states in Q and R sets have λ of permutation values. In the proposed method, the address bits can be calculated using equation (3.9).

$$addr = state_{no} \,||\, v_{location}$$

$$|state_{no}| = |v_{location}| = |x/2| \quad \textbf{(3.9)}$$

Where $state_{no}$ represents the number of the state, $v_{location}$ represents the location of state value, and x represents the size of the message block.

$$|addr| = |x| \quad \textbf{(3.10)}$$

So the address length is equal to x, the size of the message block.

In the designed algorithm, n = |Q|, m = |R| and k = |J|.

3.3.1 Permutation Process of Permutation S-RADG Algorithm

The permutation process used in Permutation S-RADG algorithm to change the positions of the $Cipher_{2_i}$, where $Cipher_{2_i}$ represents the result of XOR operation between $Cipher_1$(the cipher value resulted from XOR operation between the message and the key) and v (the value stored in the state after converted it into binary), where $|P_j| = |M_{|B|/x}| = |K_{|B|/x}|$,

where $j = 0,1, \ldots |B|/x$. The permutation values stored in the states of the design, where permutation number at each state equal to λ value (λ represent the number of value in each state except J state doesn't take any value).

Equation (3.13) is used to determine which permutation used from the permutation table.

$$P = \sum_{i=1}^{x} \text{Cipher}_{2i} \bmod \lambda \qquad \textbf{(3.13)}$$

The state Permutation S-RADG algorithm has a number, the state number and the index of state value which they represent the address of the state denoted by (addr). So each state in the designed algorithm, have address, state values and permutation values, where $|\text{addr}| = |x|$ bits.

Then, the block after permutation process becomes:

$$P_j \,||\, \text{addr}$$

Two blocks formed represent the block after permutation and the block that contains the number of state and the location of state value. Then, all blocks are encrypted by DES algorithm, where DES algorithm is block cipher algorithm that encrypt 64 bits block at time with 56 bits as a key, so the ciphering of Permutation S-RADG beings by encrypting the message by 64 bits with 56 bits key (used the same Permutation S-RADG key but just 56 bits of which). The Permutation S-RADG algorithm may be used AES algorithm for ciphering with the size of message as at least 128 bits and the key length as 256 bits (Permutation S-RADG algorithm key length by using AES algorithm become 512 bits).

3.3.2 Permutation S-RADG Implementation

This section provides the encryption process and the decryption process where the encryption and the decryption process used the same key.

3.3.2.1 Permutation S-RADG Encryption Algorithm

Permutation S-RADG encryption process starts by selecting a random state from Q set, except J state. Then by using the transitions of design, all blocks of XOR operation result (Cipher_1) are encrypted. If the transition reaches to J state, stope encryption, then reselect from R set. This transition is continued until complete all blocks of Cipher_{1_i} as shown in Algorithm 3.3.

- **Algorithm 3.3 Algorithm For Permutation Stream cipher Reaction State Automata Direct Graph Encryption**

Input: the message blocks $M = \{m_0, m_1, \dots m_{|B|/x}\}$, the key blocks $K = \{K_0, K_1 \dots . K_{|B|/x}\}$

Output: the cipher blocks $C \leftarrow \{C_0, C_1, \dots C_{|c|/2x}\}$

Steps

1. $C \leftarrow \emptyset$, set $L \leftarrow 0$
2. while $(L < |B|/x)$
3. $\mathrm{Cipher}_{1_i} \leftarrow (m_i \oplus K_i)$
4. $\mathrm{State}_{old} \leftarrow \mathrm{select\_random(Q\_states)}$
5. $\mathrm{Cipher}_2 \leftarrow (\mathrm{Cipher}_{1_{B_i}} \oplus v_{B_i})$
6.

7. addr ← concatination(q_{old}, $v_{location}$)
8. $B_j \leftarrow (P_{B_j}$, addr)
9. $C_1 \leftarrow$ DES cipher[B_j]
10. Add ($C_1, C_1(L)$)
11. if($q_{old} \in$ Jump)
12. $q_{new} \leftarrow R$

$$L = L + 1$$

13. else
14. $q_{new} \leftarrow q_{old}$
15. return C

3.3.2.2 Permutation S-RADG Decryption Algorithm

Decryption process is inverse of encryption process, where all steps are reversed to return plaintext. Decryption process adopts the block (b_n) to return address (addr) of state to reach to the state used for encryption to return the result($Cipher_{1_{B_i}}$), then by XORed sub blocks of $Cipher_{1_{B_i}}$ with sub blocks of the key(K_{B_i}), returned the plain text. A algorithm 3.4 illustrate how the decryption is done.

- **Algorithm 3.4 Algorithm For Permutation Stream cipher Reaction State Automata Direct Graph Decryption**

Input: the cipher block $C \leftarrow \{C_0, C_1, \ldots., C_{|c|/2x}\}$, the key block $K \leftarrow \{K_0, K_1, \ldots., K_{|B|/x}\}$

Output: the message block $M \leftarrow \{m_0, m_1, \ldots..\}$

Steps

1. $B_i \leftarrow \text{decipherDES}[C_1]$, $L \leftarrow |C|$
2. Left_half, Right_half $\leftarrow \text{split}(C_{|c|/2x})$, where Split operation divided the cipher text into two equals halves: the first half (left) represents the value after permutation process, and the second half (right) represents the address.

   Left_half $\leftarrow P_{B_j}$

   Right_half $\leftarrow$ addr;

3. $\text{Cipher}_{2_{B_i}} \leftarrow \text{Inv\_Permutation}(P_j)$
4. $P_{B_i} \leftarrow (v_{B_j} \oplus \text{Cipher}_{1_{B_j}})$
5. $m_i \leftarrow (\text{Cipher}_{1_i} \oplus K_i)$
6. $M \leftarrow \text{reverse sequence}(m_0, m_1, \ldots . m_{|L|})$
7. return M

3.4 Application of the Proposed Algorithms

The proposed algorithms are designed to protect data transmitted through networks to make this data more secure. Once the client sends a request to the server, the key will be exchanged for data transfer. The server encrypts the data and sends it to the client. The proposed system is used for encryption when the client uploads its data, and is used for decryption when the client loads data. Encryption and decryption are performed using the proposed algorithms where are symmetric key algorithms, use the same key in encryption and decryption by using LFSR to generate the key to increase security. Any two users connected to network used Third Party (TP) to manage the communication between two parties communications.

Chapter Four

Security Analysis

Chapter Four: Security Analysis

4.1 Introduction

This chapter introduces the performance and security analysis of the two proposed designed algorithms S-RADG (Stream Cipher Reaction State Automata and Direct Graph) and Permutation S-RADG.

4.2 Security Analysis of S-RADG Algorithm

The cryptography goals are: confidentiality, integrity and authentication. To evaluate any security system, the system should be analyzed and make to sure that it achieves these goals.

4.2.1 Confidentiality

Confidentiality means prevent wrong people to access to information, this means, it equally to privacy of information. Encryption and decryption can be used for confidentiality as illustrated in the example in Appendix (A).

4.2.2 Data Integrity

Integrity verifies the correctness of the message to the recipient, the information arrived from the allowed (correct) sender or not. In S-RADG algorithm, integrity is satisfied by notice that any modification on cipher text can detect easily. The hacker faces difficulty when he/she wants to modify cipher block marked by unauthorized user, since the S-RADG algorithm uses the secret key and transitions stored . If the hacker reaches the secret key, the decryption process of the cipher text will failed because the search process is failed where the value is not found which must give the correct plaintext and in this case cannot be continue to complete decryption process.

4.2.3 Authentication

Authentication means detection an unauthorized user in writing (i.e., modification of data). MAC (Message Authentication Code) stream cipher

based, can used to ensure authentication of ciphering message using S-RADG method where MAC method requires two parameters as inputs, message with symmetric key. In sender part, the message is encrypted using S-RADG algorithm, then MAC called MAC_1 is computed, and then sent the encrypted message and the sequences of MAC are sent to the receiver. In receiver part, the ciphertext is decrypted and the MAC of the decrypted message called MAC_2 is computed. The comparison between MAC_1 and MAC_2 is made to guarantee the authentication of message. If $MAC_1 = MAC_2$, then the message is authenticated.

Table 4.1 authentication of message M

Sender	**Receiver**
If Message M and the key K, then compute 1- A = Stream-MAC (M, K) 2- C = En-S-RADG (M, K) Send (A1, C) to receiver	If Received A, C Cipher C and Key K, then compute 1- M = De-S-RADG (C, K) 2- B= Stream-MAC (M, K) If B = A then authentic the message Else deny the message

4.3 Performance Analysis of S-RADG Algorithm

Entropy is used to measures the uncertainty in the system. Using entropy to analysis performance of S-RADG method, S-RADG is run 100 times for the same message to show how it performs in terms of entropy for different cipher text values. The Shannon entropy equation provides a way to estimate the average minimum number of bits needed to encode a the cipher text of based on the frequency of zeros values and ones values in cipher texts. Shannon entropy equation is:

$$H(X) = \sum_{i=0}^{N-1} p_i \log_2 p_i$$

Where $H(X)$ is represent the minimum average number of each bits of cipher texts and p_i represented the probability of occurrence of zero's value and one's value in the ciphertexts.

i	Cipher text
0	10011101000111101011101111000000101011101111001010110110110100101010011111111001101111010110011100010001110011010111010000011110
1	00111101100111111101001100111100100001100001000000011101111001000100011010010100001010010000101
2	11101001100101110010001000100000111011100110011110010000001000010011100011111001011010001010100000111010111001110111010011101
3	00010110100111001101000111001111001100100011011110101001100000010110001111101001011001110010101000110010110101000110101100010011
4	00001110010001000101111110111000110011000111101010001000101000110010101010001110000011110011000000001101010001110111111111110010
5	11011001010011111111010010100110010100100101100011101010000111010101100001111001100110001101100101010010101100111010011111101011
6	00100010111111111111011100011010100011110101000001110001000010111111101010010111100101101001011110101110011011001011101110111001100011
7	00001100100111001000111111010111011110100111110001101110111000100111001001111111001101111011001000001101110011001110011100011000100011
8	00011001000001111001000100010101011001100000000001101000000011100101001111111100110111101011001110001000111001101011101000011110
9	00111101100111111101001100111100100001100001000000011101111001000100011000100000111010100010001110111111001100010011010101

	10010100
10	11100010011110110011001111101110101010010000101100010001100101010001001000100000110010010010100100101011110110000010011110011010
11	10100000010011000111001110110110101111110111001101100101011111001011000111101001000001000010010010010000110001110001011010110110
12	01101100011000001001000111011000010000100110111111011000001000100111001011010110000010010010110001111111010101101011111101110010
	:
	:
	:
	:
	:
100	01111001001111010110001101011110111111011101011011001100000010011010101000001110101011011010111011000111010111101010001010111000

The total number of ciphertext analyzed by using entropy 100 ciphertext, Table (4.2) show the different ciphertexts for the same message of example explained in Appendix A.

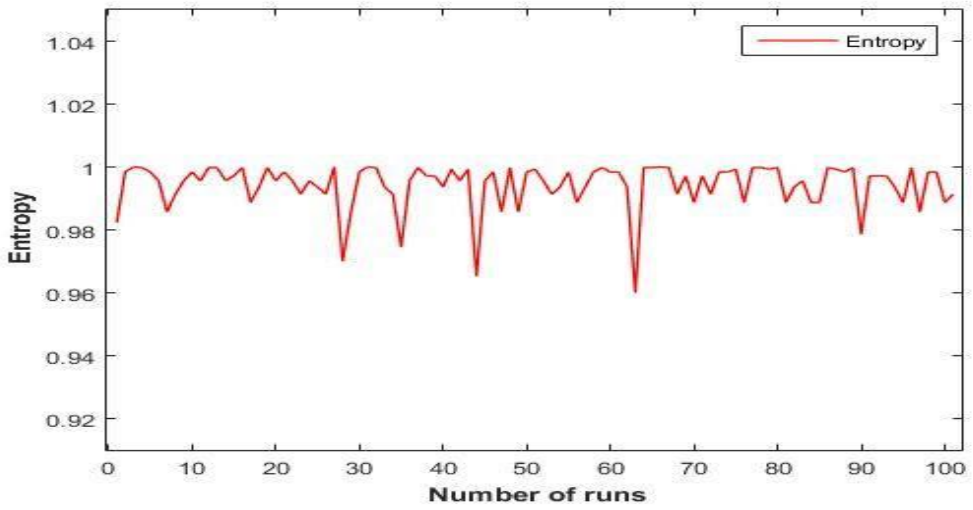


Figure 4.1 shows the entropy between individual ciphertexts for message length of 128 bits.

We can notice from the previous Figure that, the new S-RADG method is random and can result different ciphertexts for the same plaintext.

4.4 Security Analysis of Permutation S-RADG Algorithm

Also this security system must satisfy cryptography goals.

4.4.1 Confidentiality

Confidentiality goal is to ensure the privacy of data and this satisfies by using encryption and decryption techniques, this proven by example Appendix (B).

4.4.2 Data Integrity

Integrity protects the data from any modification that exposed by malicious. In Permutation S-RADG algorithm, there are many different approaches to confirm that, any modification in the cipher text, will give an error. When change any bit in the block of cipher text then decrypt it using DES algorithm to return $b_n$where b_n include the block value after permutation and the block value of the address state, will return wrong value after inverse permutation. The following example Table (4.3) explains how to detect any modification in ciphertext using transitions function, where the encryption and decryption process and the example in Table (4.3) explained in Appendix B.

The iteration of the block ciphertext marked by unauthorized user, the decryption operation of the ciphertext will go wrong since returns the wrong inverse permutation value, this leads to fault plaintext and for this situation cannot keep on completing decryption process.

Table 4.3 Integrity of The Permutation Stream cipher Reaction State Automata Direct Graph algorithm

	Cipher text	b_n	State number	jump	addr	P_{B_j}
10	1111110111110001	1000001101101000	3	-	00110000	10111110
9	1001100000000001	0110001001100001	6	-	01100010	01100001
8	1010100111010000	0011000010111110	8	J	10000011	01101000
7	1100000100101011	0000000000000000	6	-	00010011	11110100
6	1010101101001010	----------	---	--	--------	---------
5	0101010101101101	----------	---	--	---------	----------
4	0101010010100101	----------	---	--	----------	----------
3	1011100110000110	----------	---	--	----------	----------
2	0101001111111111	----------	---	--	----------	----------
1	1100101001000100	----------	---	--	----------	----------
0	0111011100101100	----------	---	--	----------	---------
	1111101111110011	----------				

4.4.3 Authentication

Authentication is a way toward deciding if a user ought to be enabled access to system. After authentication, the user allows to access the data but not all, only authenticated. A server can employ to verify the authenticity of message by agreeing two users on certain key and using random value can prove that.

Then the steps of authentication will be as following:

Step1: Two users represented as N1 and N2, they agreed to use session key via server to use in encryption and decryption data, N1 encrypts a random number (r_1) by using Permutation S-RADG algorithm and send it to the server.

Step2: N1 encrypts the same random number(r_1) and send it with encrypted data ($cipher_{N_1}$) as a message authentication code to N2.

Step3: N2 receiving the encrypted data ($cipher_{N_1}$) and (r_1) from N1, then use the same key to decrypt the ciphering to get plain text then generate the random number r_2.

Step5: The receiver (N_2) compared the two random numbers (N_2) with the random number that received(N_1), if the two random number equals, the request of sender is done (authenticated request) otherwise, the request is rejected. Figure 4.2shows the authentication steps.

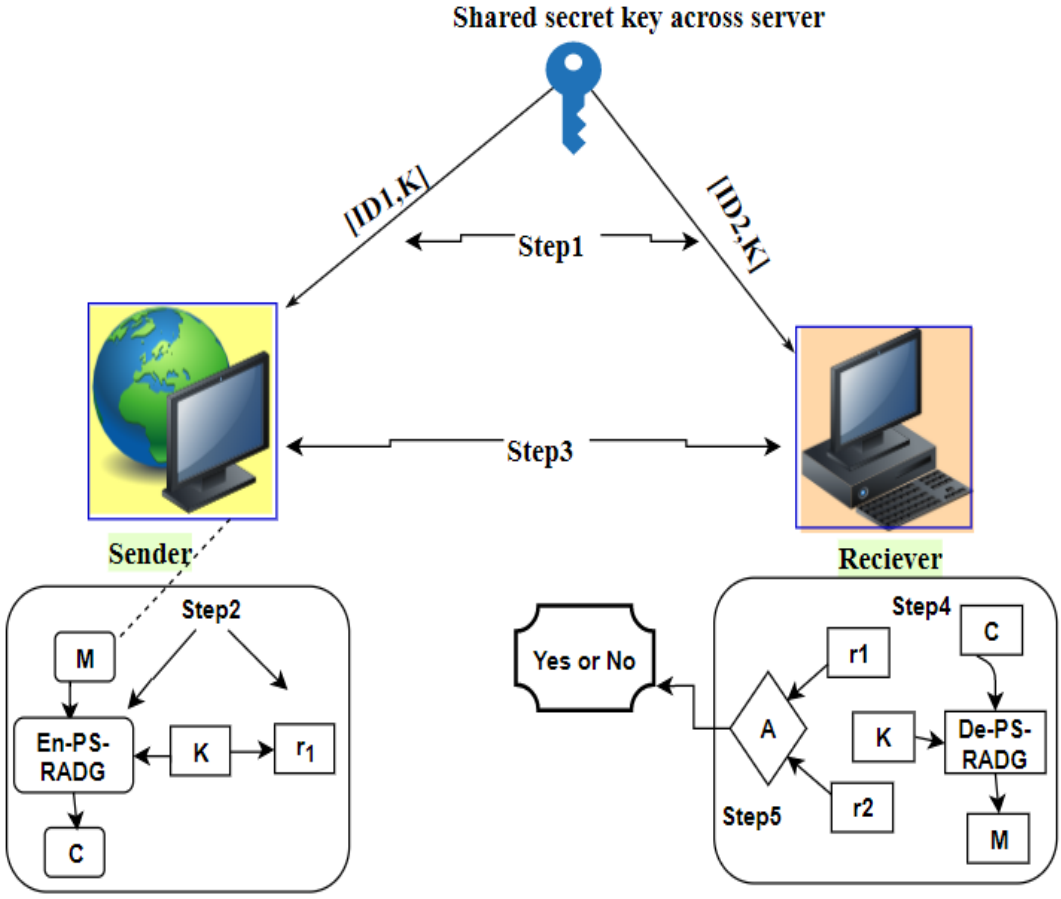


Figure 4.2 Permutation Stream cipher Reaction State Automata Direct Graph Authentication Steps

4.5 Performance Analysis of Permutation S-RADG Design

Using entropy to analyses performance of Permutation S-RADG method, Permutation S-RADG is run 100 times for the same message to show how it performs in terms of entropy for different ciphertext values. Figure 4.3 shows the entropy for the same plaintext (of length 192-bits) runs 100 times.

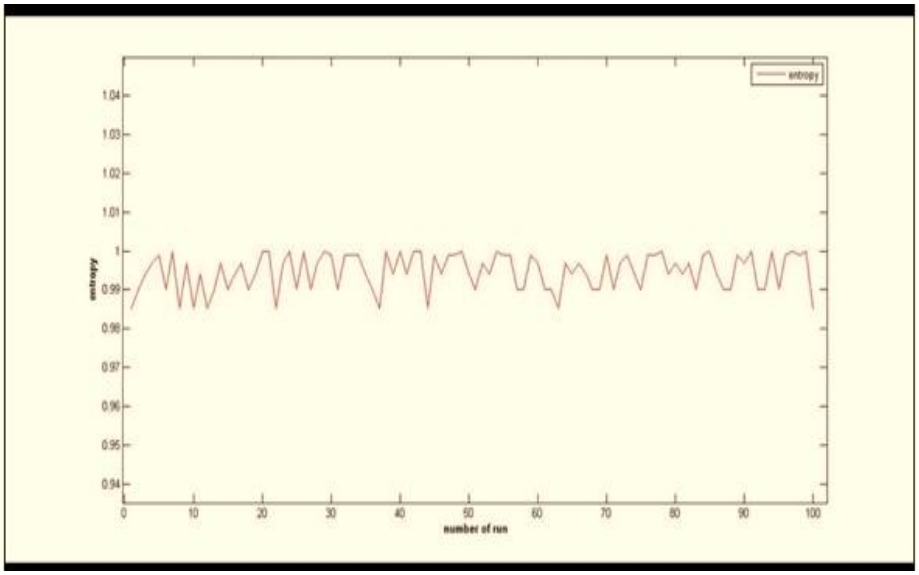


Figure 4.3 Entropy for Message Run 100 Times

We can notice from previous Figure that, the Permutation S-RADG method is random and can result different ciphertexts for the same plaintext. Table (4.4) show the different ciphertexts for the same message of example explained in Appendix B.

Table 4.4 Cipher Text resulted from Permutation Stream cipher Reaction State Automata Direct Graph Algorithm

i	Cipher text
0	1100100101000110111011001111010000101000011111001000110010100110100001100100101110101001011011110001010000100010110101111111010101001001010111000011111110111110100000110100000101011111110111101
1	0001101101000000001000010110001101100011010101100111111100100111100001100100101110101001011011110001010000100010110101111111010101001001010111000011111110111110100000110100000101011111110111101

	111110111101
2	1111010111011101010111010011000100110111110100101000000101101010100001100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101
3	0111110111101100110000110011101100100111011010001000110010011011100001100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101
4	1000010001011011001111010000111100110011011100100011100100111101000011100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101
5	1000010001011011001111010000111100110011011100100011100100111101000011100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101
6	0001101101000000001000010110001101100011010101100111111100100111000011001001011101010010110111100010100001000101101011111111010010010101110000111111101111101000001101000001010111111011101
7	0000110011101111101100000101111010011110111011100010110011100010100001100100101110101001011011110001010000100010110101111111010010010101110000111111101111101000001101000001010111111011101
8	1010010101111000011011000111111011001100100001011011111111001111100001100100101110101001011011110001010000100010110101111111010010010101110000111111101111101000001101000001010111111011101
9	0111110111101100110000110011101100100111011010001000110010011011100001100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101
10	1000010001011011001111010000111100110011011100100011100100111101000011100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101
	:
	:
	:
	:
	:

100	1100100101000110111011001111010000101000011111001000110010100110100001100100101110101001011011110001010000100010110101111111010100100101011100001111111011111010000011010000010101111110111101

4.6 Attack Cryptanalysis

This section presentes seven basic cryptanalytic methods which are used as a part of cryptographic attacks, likewise indicateed to as cryptanalysis.

4.6.1 Exhaustive Search Attack

This attack also called brute force attack, where the attacker searches all possible cases by using the key to decrypt the ciphertext. Where the possible cases of exhaustive search attack is 2^n.

In S-RADG method the size of key is 56-bits, so the number of possible search cases to find the correct key are 2^{56}. In the Permutation S-RADG method the number of possible search is 2^{128} if used DES algorithm or 2^{512} if used AES algorithm, the big size of the key makes the attacker meets obstruction in guess the key of Permutation S-RADG algorithm.

4.6.2 Guess-and-Determine- Attack

This type of attack involves several steps. First, the attacker guesses the first block of the plaintext and then the remaining blocks through their relationship to the ciphertext. Finally, the estimated portion compared with the ciphertext.

In S-RADG and Permutation S-RADG methods the plaintext is completely independent from ciphertext and the attacker face difficult when he tries to estimate.

4.6.3 Algebraic Attack

Algebraic attack is used in LFSR method where LFSR method is used in designed methods (S-RADG and Permutation S-RADG methods) as a key. The algebraic attacker tries to recover the key after trying to discover the initial state of the sequences bits of LFSR method (the key). This type of attack tries to find the key equation which it is generated the sequences that used as a key of designed methods. The key of S-RADG method and Permutation S-RADG method is generated randomly by selecting a random initial state to generate the sequence of the key, the algebraic attacker faces difficulty to guess the primitive polynomial of the LFSR used to generate the sequences and initial state that make the sequence 128 bits or 512 bits. By using the Berlekamp Massey Algorithm explained in section (2.5.1), will find the characteristic polynomial of the lowest possible degree that will generate the sequence of the LFSR of the proposed algorithms.

By using the same example explained in Appendix B, the initial state is:

$S = 10000101$

1-

$$C(x) = B(x) = 1$$

$$L = N = 0$$

$$m = -1$$

$$\text{While } 0 < 8$$

$$d = S0 = 1$$ **yes**

$$\text{if } d = 1$$

$$T(x) = 1$$

$$C(x) = 1 + x$$

$$\text{If } 0 \leq 0/2$$ **yes**

$$L = 0 + 1 - 0 = 1$$

$$m = 0$$

$$B(x) = 1$$

$$N = 1$$

2-

While $1 < 8$ **yes**

$$d = S_1 \oplus C_1\ S_0 = 0 \oplus 1.1 = 1$$

if $d = 1$ **yes**

$$\boldsymbol{T(x) = 1 + x}$$

$$C(x) = 1 + x + 1.x^{1-0} = 1$$

If $1 \leq 1/2$ **No**

$$N = 1 + 1 = 2$$

3-

While $2 < 8$ **yes**

$$d = S_2 \oplus 0.0 \oplus 0.1 = 1$$

if $d = 1$ **yes**

$$T(x) = 1$$

$$C(x) = 1 + 1.x^{2-0} = 1 + x^2$$

If $1 \leq 2/2$ **yes**

$$L = 2 + 1 - 1 = 2$$

$$m = 2$$

$$B(x) = 1$$

$$N = 3$$

4-

While $3 < 8$ **yes**

$$d = S_3 \oplus C_1\ S_2 \oplus C_2\ S_1 \oplus C_3\ S_0$$

$$d = 0 \oplus 0.0 \oplus 1.0 \oplus 0.1 = 0$$

$$\text{If } d = 1 \; N = 0$$

$$N = 3 + 1 = 4$$

5-

$$\text{While } 4 < 8$$ **yes**

$$d = S_4 \oplus C_1 S_3 \oplus C_2 S_2 \oplus C_3 S_0$$

$$d = 0 \oplus 0.0 \oplus 1.0 \oplus 0.0 = 0$$

$$\text{If } d = 1 \text{ No}; \; N = 4 + 1 = 5$$

6-

$$\text{while } 5 < 8$$ **yes**

$$d = S_5 \oplus C_1 S_4 \oplus C_2 S_3 \oplus C_3 S_2 \oplus C_4 S_1 \oplus C_5 S_0$$

$$= 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$$

$$\text{If } d = 1 \text{ yes;}$$

$$T(x) = 1 + x^2$$

$$C(x) = 1 + x^2 + x^{5-2}$$

$$\boxed{= 1 + x^2 + x^3}$$

$$\text{If } 2 \leq 5/2$$

$$3L = 5 + 1 - 2 = 4; \; m = 5; \; B(x) = 1 + x^2; \; N = 6$$

7-

$$\text{while } 6 < 8$$

$$d = S_6 \oplus C_1 S_5 \oplus S_4 C_2 \oplus S_3 C_3 \oplus S_2 C_4 \oplus S_1 C_5 \oplus S_0 C_6$$

$$= 0 \oplus 0.1 \oplus 0.1 \oplus 0.1 \oplus 0.0 \oplus 0.0 \oplus 1.0 = 0$$

$$\text{If } d = 1 \text{ No}$$

$$N = 6 + 1 = 7$$

8-

$$\text{while } 7 < 8 \quad \text{yes}$$

$$d = S_7 \oplus S_6 C_1 \oplus S_5 C_2 \oplus S_4 C_3 \oplus S_3 C_4 \oplus S_2 C_5 \oplus S_1 C_6 \oplus S_0 C_7$$

$$= 1 \oplus 0.0 \oplus 1.1 \oplus 0.1 \oplus 0.0 \oplus 0.0 \oplus 0.0 \oplus 1.0$$

$$= 1 \oplus 1 = 0$$

$$\text{If } d = 1 \text{ No}$$

$$N = 1 + 8$$

This trace prove the difficulty of discovering the primitive polynomial that used in the proposed algorithms, where the primitive polynomial used in example explained in Appendix B is $x^8 + x^2 + 1$, while the primitive polynomial determined by the Berlekamp-Massey algorithm to the same initial state is $1 + x^2 + x^3$.

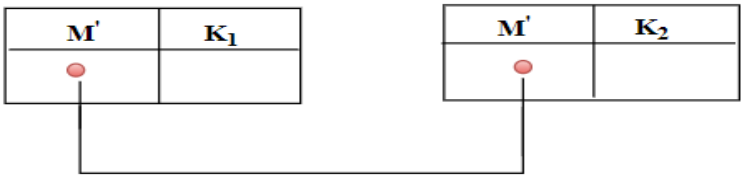
4.6.4 Meet-in-the-Middle Attack

The idea of this attack is that a middle text is created for the encryption process of the encryption or decryption process and must be equal to both processes.

$$M' = E_{K_1}(P), \quad M' = D_{K_2}(C)$$

- Encrypt the plaintext using all possible values from K_1 and records the resulting values to M'.
- Decrypt the ciphertext using all possible values from K_2 and records the resulting values to M'.

Created two tables are created to store the all resulting values that are similar.



needed $\mathbf{2^k}$ key search test, so the two design need $\mathbf{2^{128}}$ or $\mathbf{2^{512}}$ key seacrh test.

4.6.5 Differential Cryptanalysis

The attacker selects two plaintexts, then use XOR operation to this two inputs and show the difference for the results.

The encryption process of S-RADG method and Permutation S-RADG method is random as well as the Permutation S-RADG based on permutation process add strong point to the ciphering.

Differential cryptanalysis at first, compute the characteristics of inputs and outputs of permutation process. So suppose the permutation process with A as input and B as output, then :

$$A' = A \oplus A^*$$

Where A' represents the differential of the two plaintexts, A represents the first plaintext and A^* represent the second plaintext.

For the outputs, a similar process is done :

$$B' = B \oplus B^*$$

Where B and B^* represents the two outputs, recorded all values of A, A^*,A',B, B^* and B' in table. Then, analyze the characteristics of the deferential, A' of input and the differential, B' of output to know how A' affects B'.

Both S-RADG method and Permutation S-RADG method are based on ciphering process as well as the Permutation S-RADG based on permutation process this gives a clear idea that the two methods are strong in the face of differential attack because the random encryption which means each plaintext give more than one ciphertext so it is hard to compare two plaintexts and get the differential.

4.6.6 Divide-and-Conquer Attack

This attacker exploits the linear relationships between inputs and outputs (plaintexts and ciphertexts) during encryption process. Since Permutation S-RADG method uses permutation process, making it non-linear method, so the attacker has difficulty for finding the relationship between the plaintext and the ciphertext.

4.7 Time Complexity

The time complexity of algorithm measures the amount of time taken by an algorithm to run as a function of the length of the string representing the input. The most well-known metric for computing time complexity is big O notation. This deletes every constant factors with the goal that the running time can be assessed in connection to n as n approaches infinity. The running time of the loop is directly proportional to n, so according to S-RADG encryption algorithm (3.1).

1. $C \leftarrow \emptyset$, $q_{current} = q_{rand}$, $L \leftarrow 0$
2. $while\ (L < |M|)$
3. $\mathrm{I} \leftarrow (m_L \oplus K_L)$
4. $(\mathrm{V})_{\mathrm{L}} \leftarrow getvalue(q_{current}, index)$
5. $b_n \leftarrow (I, V, q_{current})$
6. $C_1 \leftarrow DES[\, b_n]$
7. $\mathrm{Add}\ (\mathrm{C_{L-1}}, \mathrm{C_L})$
8. $\mathrm{if}(q_{current} \in Jump)$

   $q_{new} \leftarrow q_r, \quad q_r \in R$

The time complexity of S-RADG encryption algorithm is $O(n)$, according to the length of input and the loop.

For S-RADG decryption algorithm, the time complexity is $O(mn)$. Where n represents the input that will be decrypted and m represents the λ value. Since the value of λ is constant then the time complexity is $O(n)$. The best case, the worst case, average case and best case of S-RADG method for encryption and decryption process is $O(n)$.

The time complexity of permutation S-RADG encryption algorithm (3.3) is $O(n \log n)$, since the input is divided into sub blocks with same size and then merge them in n size. If number of blocks equal to 1 then the time complexity is $O(\log n)$.

1. $C \leftarrow \emptyset$, set Length $\leftarrow 0$
2. while $(L < |B|/x)$
3. $Cipher_{1_i} \leftarrow (m_i \oplus K_i)$
4. $Cipher_{1_{|B|/x}} \leftarrow$ split $(Cipher_{1_i})$
5. $State_{old} \leftarrow$ select_random(Q_states)
6. $Cipher_2 \leftarrow (Cipher_{1_{B_i}} \oplus v_{B_h})$
7. $P_{B_j} \leftarrow$ Permutation $(Cipher_{2_{B_i}})$
8. addr $\leftarrow q_{old} + v_{location}$

For Permutation S-RADG decryption algorithm the time complexity is $O(mn \log n)$, where m represents λ value where the value of λ is constant then the time complexity is $O(n \log n)$ in the worst and average case.

4.8 Comparison RADG with Proposed Designs

The proposed designs (S-RADG and Permutation S-RADG) are development of RADG design. All designs are affected by graph theory.

Encryption process starts by selecting randomly initial state from Q states except jump state. There are some differences between RADG and proposed designs as clarify in Table 4.5.

Table 4.5 Comparison between Reaction Automata Direct Graph Method and the Proposed Designs

RADG Method		S-RADG and Permutation S-RADG Methods
1	Keyless cryptography method	Symmetric key cryptography methods
2	Output is hexadecimal number	Output is binary number
3	$\lambda = 2$, this means each state has two values	$\lambda = x$, this means each state has numbers of values
4	Efficient to protect wireless network	Efficient to protect cloud computing network
5	Effective with peer to peer communication and don't have to utilize server in the network and there is no agreement among users.	Effective with peer to peer and multi users communicate where utilize CIS as a server to distribute the services among users and to concurrence on keys between any two users on the network
6	Depend on search in decryption process	Do not need to search in decryption process, since depended on the key and the address of the state to know the correct state.

4.9 Comparison S-RADG Algorithm with Permutation S-RADG Algorithm

The two proposed algorithms stream cipher reaction state automata and direct graph (S-RADG) and Permutation S-RADG are depend on the same features of RADG mathematical model and used the same stream cipher

method called LFSR (Linear Feedback Shift Register) to generate the key that used in encryption and decryption process. The following Table 4.6 shows the differences between them.

Table 4.6 Comparison between the Proposed Designs

	S-RADG Method	Permutation S-RADG Methods
1	Input binary message one bit at time (0,1)	Input binary message as block bits
2	Input key as one bit at time (0,1)	Input key as block bits
3	The structure of S-RADG state consists of state number and λ of values.	The structure of Permutation S-RADG state consists of state number , λ of state values and λ of permutations table, don't have values in the state
4	S-RADG algorithm doesn't have any process in its work	Permutation S-RADG algorithm have permutation process in its work
5	S-RADG depends on XOR operation between one bit of message and one bit of key to determine the index of the value	Permutation S-RADG depends on the sum of the output after XOR operation between block bits of message with block bits of key module λ to determine the number of permutation
6	The address of S-RADG state is state number after converting into binary number and depends on the function to determine the size of address block. The address block is computed by :	The address of Permutation S-RADG state represents the state number after converting it into binary number and the location of state value that selected, and the block size depends on the block size of message.

	$addr = \lceil \log_2(n + m) \rceil$ Where addr represents address of state.	

Chapter Five

Conclusions and Future works

Chapter Five: Conclusions and Future Works

5.1 Introduction

This chapter introduces the conclusions of the new two designs, Stream Cipher RADG (S-RADG) and Permutation S-RADG designs based on testing and implementing them to protect data. The conclusion discussed in section (5.*2*). Also, in this chapter there are two suggestions to improve the efficiency of the new proposed designs presented in section (5.*3*) as a future work.

5.2 Conclusions

Designing and implementing Stream Cipher Reaction Automata Direct Graph resulted many conclusions.

1. The S-RADG and Permutation S-RADG are cascade encryption, since there are more than one encryption process.
2. In implementation of both the S-RADG and Permutation S-RADG methods, the encryption of plaintext will be randomly dependent on using mathematical modeling.
3. Stream cipher based on using different key at every time and this make the cipher weak but the proposed algorithms (S-RADG and Permutation S-RADG) used the same key at every time.
4. Both the S-RADG and Permutation S-RADG methods achieved security requirements, such as confidentiality, data Integrity, authentication.
5. The permutation process in Permutation S-RADG increases the randomness which makes the method more complicate, more random and hardest to break by attacker than S-RADG method.

5.3 Future works

There are two suggestions to improve the ability and the efficiency of the proposed designs listed as future work:

1- key management is necessary to the security of a cryptosystem, so the next work possible be the key management of S-RADG and permutation S-RADG algorithms.

2- Improvement S-RADG and Permutation S-RADG algorithms by attempting to decrease the size of ciphertext.

References

References

[Pat3] Patrik Ekdahl , T.J., *A New Version of the Stream Cipher SNOW.* Springer, 2003.

[Hos7] Hossam El-din H. Ahmed, H.M.K., and Osama S. Farag Allah, *An Efficient Chaos-Based Feedback Stream Cipher (ECBFSC) for Image Encryption and Decryption.* Informatica 31, 2007.

[Sal16] Salah, A.A.; and Ghazanfar A.; "*Keyless Security in Wireless Networks*", Springer, 25 July 2014.

[Ais15] Aissa Belmeguenai, K.M., Mohamed Lashab, *Speech Encryption Using Stream Cipher.* British Journal of Applied Science & Technology, 2015. 8.

[Sal14] Salah, A.A.; and Ali.H.; "*MRADG design on Elliptic Curve Cryptography*", American Research Foundation, ICCIIDT London - UK, October 2016.

[Sal17] Salah, A.A.; and Fatema, R.; "*New Block Cipher Key with RADG Automata*", Asian Journal of Information Technology, Vol.16, 2017

[Ali17] Ali H. Kashmar , E.S.I., *BLOSTREAM: A HIGH SPEED STREAM CIPHER.* Journal of Engineering Science and Technology, 2017. **12**.

[Paa10] C. Paar and J.Pelzl; "*Understanding Cryptography*", Springer, 2010.

[Suk12] Sukalyan, S.; and Saikat, G.; "*A Stream Cipher Cryptosystem Based on Linear Feedback Shift Register*", International Journal of Mathematical Archive, P. 362-372, February 2012.

[M.J.B95] M.J.B, R.; "*Stream Ciphers*", RSA Laboratories, a division of RSA Data Security, July 25, 1995.

[Jaw11] Jawahar, T.; and Nagesh, K.; "*DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis"*, International Journal of Emerging Technology and Advanced Engineering, Vol.1, Issue.2, December 2011.

[Kle13] Klein, A.; "*Stream Ciphers*", Springer, 2013.

[Nor10] Norto, G.H., *The Berlekamp-Massey Algorithm via Minimal Polynomials.* 2010.

[How94] Howard, M.H.; and Stafford, E.T.; "*Substitution-Permutation Networks Resistant to Differential and Linear Cryptanalysis*", The Natural Sciences and Engineering Research Council of Canada and the Telecommunications Research Institute of Ontario., 1994.

[Chi15] Chris, C.; "*Permutation Ciphers*", Springer, 2015.

Appendices

APPENDIX (A): S-RADG ALGORITHM EXAMPLE

Let $m = 6$, $n = 4$,$k = 2$, all states in Figure A.1, have λ=2.

Where $m = |R|, n = |Q|$, $k = |J|$. Using the message "hi" to generate key and then it will be the input to the encryption process.

- **Key Generation**

The key is generated by LFSR method, which used for both encryption and decryption, at first, the key is generated by using equation of LFSR method.

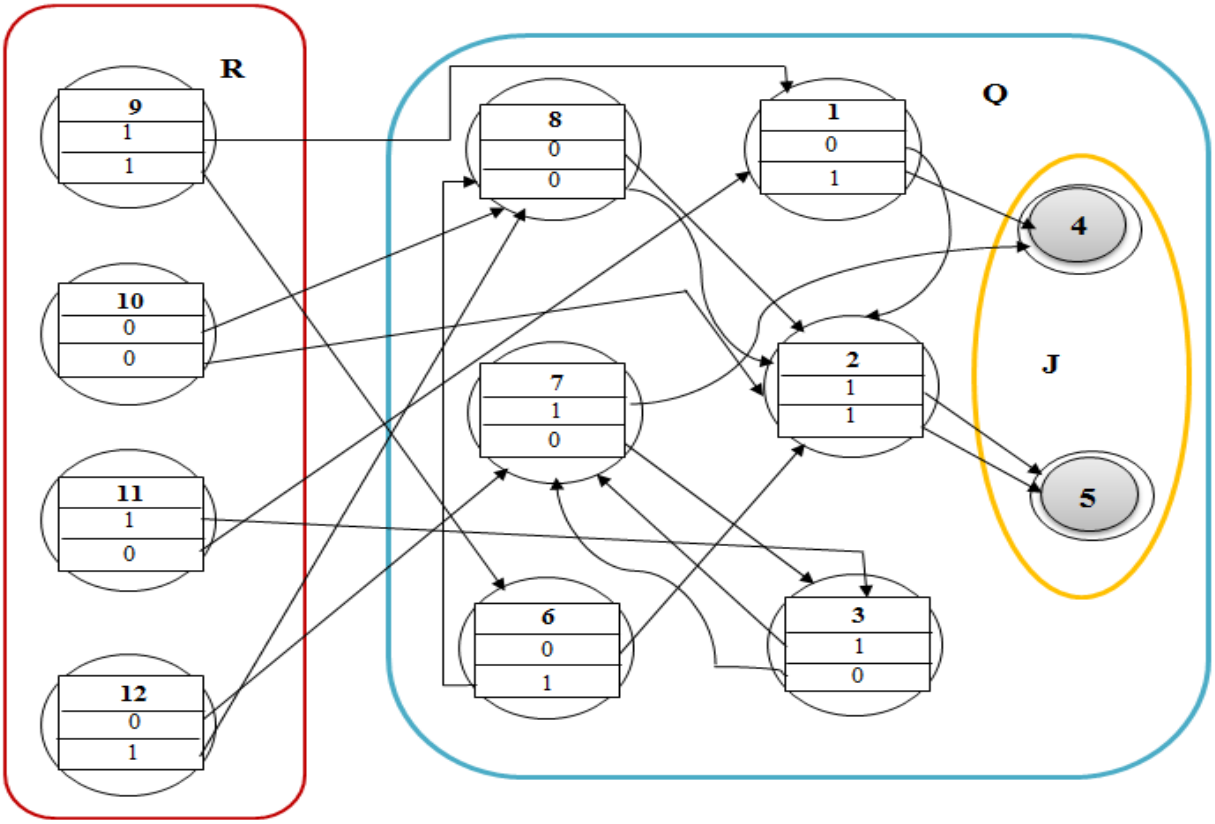


Figure A.1 Stream Reaction Automata Direct Graph Transitions

Suppose that the primitive polynomial is $P(x) = x^4 + x + 1$.The stages of LFSR are illustrated in Figure A.2.

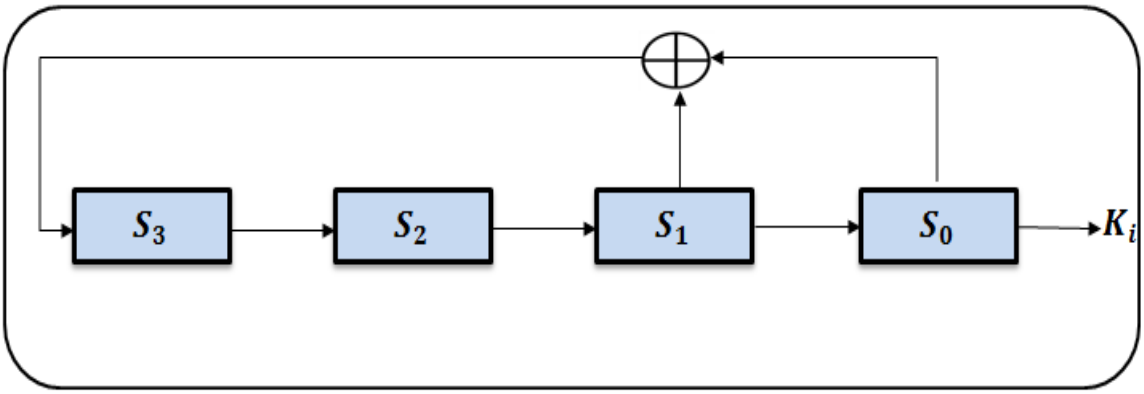


Figure A.2 Linear Feedback Shift Register Length

Suppose that the initial value of seed = 1010, by using the equation 2.1. The key is generated by LFSR method as shown in Table A.1.

Table A.1 The key of Stream Reaction Automata Direct Graph method

Result of LFSR				
t	S_3	S_2	S_1	S_0
1	1	0	1	1
2	1	1	1	0
3	1	1	1	1

The key is: 1101 1110 1111

- **Data Encryption**

First, convert the message into binary as the below:

hi: 0110 1000 0110 1001

Then, XORed the plaintext with the key as the below:

$$\begin{array}{r} 0110\ 1000\ 0110\ 1001 \\ 1101\ 1110\ 1111\ \ 1101 \quad \oplus \\ \hline 1011\ 0110\ 1001\ 0100 \end{array}$$

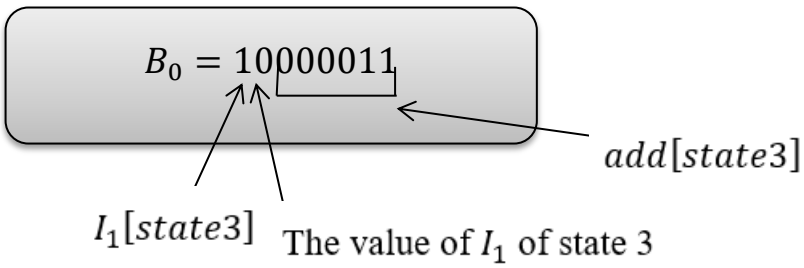
This result is used as I_i of the value for each state.

Then, by Figure A.1, start by selecting random state from **Q** state, the selected initial state in this example is the state 3, and then calculate the address to each state by equation (3.1).

Table A.2 Encryption of Stream Reaction Automata Direct Graph Algorithm

S-RADG Encryption							
i	**State number**	**Message**	**Key**	**I(index)**	**Jump**	**B_i**	**Cipher data**
0	*3*	0	1	1	-	10000011	00000011
1	*1*	1	1	0	-	00000001	10000001
2	2	1	0	1	-	11 000010	11000010
3	11	0	1	1	J	10 001011	00001011
4	1	1	1	0	-	00 000001	10000001
5	2	0	1	1	-	11 000010	01000010
6	11	0	1	1	J	10 001011	00001011
7	1	0	0	0	-	00 000001	00000001
8	2	0	1	1	-	11000010	01000010
9	10	1	1	0	J	00100000	10001010
10	8	1	1	0	-	00100000	10001000
11	*2*	0	1	1	-	11000010	01000010
12	10	1	1	1	J	10100000	10001010
13	8	0	1	0	-	00100000	00001000
14	2	0	0	0	-	01000010	01000010
15	*9*	1	1	0	J	01100100	11001001

The state 3 have address = 000011, and so on the remaining states. The beginning by the first bit of message represents as data in the Table A.2, the first bit is 0 with the first bit of key is 1 to result 1, this represents the index of the state 3 value, where the value in index 1 is 1. Then, the block become



Then, according to Figure 1, transmitted to state 7 and repeat the steps that processed on step 3, if accessed to jump state where the jump state in this example is state 4 and state 5, and then back select a random state from R state.

Since DES algorithm uses 64 bits as input, so dividing the 128 bits into two parts to become 64 bits at each part, then encrypt each part independently by using the same key of S-RADG that generated by LFSR method. The key of DES algorithm is 56 bits (the cipher key is normally given as a 64-bits key in which 8 extra bits are the parity bits, which are dropped before the actual key-generation process), so, if the data size to be The DES encrypt the first 64-bits then repeat the work for other 64-bits, by using same S-RADG key.

- **Data Decryption**

The decryption process is a reverse of the encryption process, where the receiver receives the encryption data from the sender and begins to decrypt it, where the encryption data includes cipher data and the key exchanged between the two parties.

First, the receiver begins to decrypt the cipher data by using DES algorithm to result B_i raw in Table 3.3.

Then from B_0, the state number is extracted which the sender uses it as the initial state, where B_0= 10000011, according to S-RADG algorithm, the sixth right bits represent the address of the state.

$$\text{addr} = 000011$$

After that convert the binary number to decimal number

$$\text{addr} = 3$$

Then from the sameB_0, extracted the index of the value of state 3 according to S-RADG algorithm the index (I):

B_0= 10000011

Index of state

$$I[\text{state } 3] = 1$$

Then the finally the state value (V) is:

B_0= 10000011

Value of state

$$V[\text{state } 3(I_0)] = 0$$

Then by XORed the key with index of state (I), the plain text is retrieved

$$I_0 \oplus K_0 = P_0$$

$$1 \oplus 1 = 0$$

Apply all the mentions steps above of decryption until reaching the last bit of cipher data. After completion decryption, all cipher data then inverse the result to become as shown in Table 3.

Table A.3 Decryption of Stream Reaction Automata Direct Graph Algorithm

S-RADG Decryption							
i	Cipher data	B_i	State number	State value	I(index)	Key	Message
15	11001001	01100100	9	0	0	1	1
14	01000010	01000010	2	1	0	0	0
13	00001000	00100000	8	0	0	1	0
12	10001010	10100000	10	0	1	1	1
11	01000010	11000010	2	1	1	1	0
10	10001000	00100000	8	0	0	1	1
9	10001010	01000000	10	1	0	1	1
8	01000010	11000010	2	1	1	1	0
7	00000001	00000001	1	0	0	0	0
6	00001011	10001011	11	1	1	1	0
5	01000010	11000010	2	0	1	1	0
4	10000001	00000001	1	1	0	1	1
3	10001011	10001011	11	0	1	1	0
2	11000010	11000010	2	0	1	0	1
1	10000001	00000001	1	1	0	1	1
0	00000011	10000011	3	0	1	1	0

APPENDIX (B)

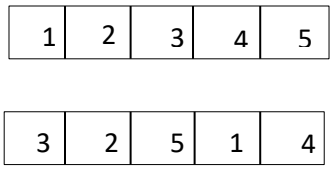
APPENDIX (B.1):PERMUTATION PROCESS AND PERMUTATION S-RADG ALGORITHM EXAMPLE

Permutation Process: Permutation is a process that reorders the sequence of letters of the message. For example: The plaintext is : alphabet to encipher the plaintext using permutation process, let divide the message into blocks at each block there is 5- letters [How94, Chi15].

a	l	p	h	a

b	e	t	β	β

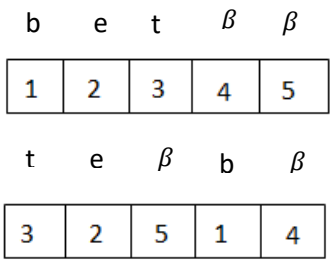
The number of letters in the second block is not composed of 5, so this block must be padded. Suppose that the following permutation to rearrange the message letters.



Reorder the letters according to the above permutation, the letter in the first position in the message moves to the fourth position in the block, the letter in the second position also stay in the second position of block, the letter in the position 3 moves to the first position, the letter in the position 4 moves to fifth position and the letter in the position 5 move to the position 3 of the block permutation.

Applied the permutation of the first block:

Then the permutation applied to the second block.



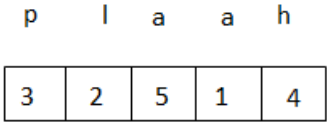
The message after the permutation using permutation set becomes:

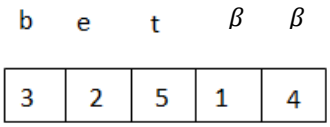
plaah teβbβ

The cipher message is: plaahteβbβ

For decryption this message to return the original message, by using inverse permutation:

The letter in the first position in the block moves to the position 3, the letter in the second position remain in the same position, the letter in the position 3 moves to the position 5, the letter in the position 4 moves to the first position and the letter in the position 5 moves to the position 4.





The plain text is alphabetββ

- **Permutation S-RADG Algorithm Example**

Let $m = 4$, $n = 2$, $k = 2$, all states in Appendix (B.2), have λ=2. Where $m = |Q|$, $n = |R|$, $k = |J|$. Used the message "hellopeople" for encryption process, where $x = 8$ bits.

The key generated by LFSR method where used for encryption and decryption process. The key generated as sub blocks, where $|B| = x$.

Suppose the primitive polynomial is $P(x) = x^8 + x^2 + 1$ and the initial value = 10000101.

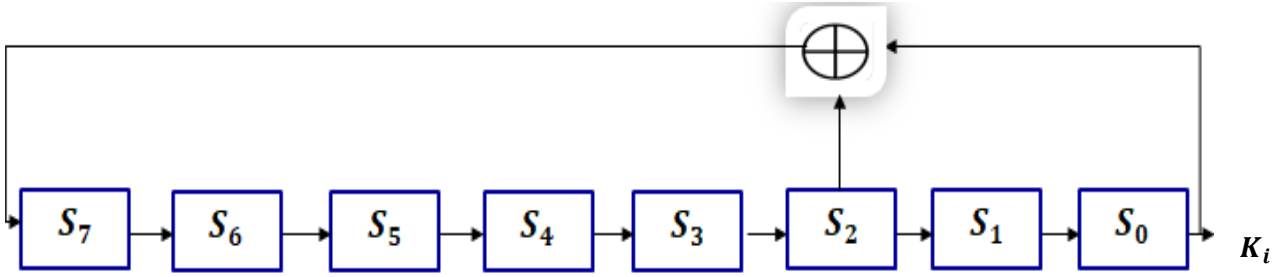


Figure B.1 Linear Feedback Shift Register Length of the Example

At i = 29 in Table B.1, the sequence equal to seed (initial state= 10000101), so stopped at this sequence. Since the key size of Permutation S-RADG method is 128 bit, at i = 15 the sequence length is 128, so ignored the sequence after i = 15.

The generated key is: 01000010001000011001000001001000 00100100100100100100100110100100110100100110100110110100110110 10011011010011011010011011111001101.

Table B.1 Linear Feedback Shift Rigister Generation of the Example

i	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0
0	0	1	0	0	0	0	1	0
1	0	0	1	0	0	0	0	1
2	1	0	0	1	0	0	0	0
3	0	1	0	0	1	0	0	0
4	0	0	1	0	0	1	0	0
5	1	0	0	1	0	0	1	0
6	0	1	0	0	1	0	0	1
7	1	0	1	0	0	1	0	0
8	1	1	0	1	0	0	1	0
9	0	1	1	0	1	0	0	1
10	1	0	1	1	0	1	0	0
11	1	1	0	1	1	0	1	0

12	0	1	1	0	1	1	0	1
13	0	0	1	1	0	1	1	0
14	1	0	0	1	1	0	1	1
15	1	1	0	0	1	1	0	1
16	0	1	1	0	0	1	1	0
17	1	0	1	1	0	0	1	1
18	1	1	0	1	1	0	0	1
19	1	1	1	0	1	1	0	0
20	1	1	1	1	0	1	1	0
21	1	1	1	1	1	0	1	1
22	1	1	1	1	1	1	0	1
23	0	1	1	1	1	1	1	0
24	1	0	1	1	1	1	1	1
25	0	1	0	1	1	1	1	1
26	0	0	1	0	1	1	1	1
27	0	0	0	1	0	1	1	1
28	0	0	0	0	1	0	1	1
29	1	0	0	0	0	1	0	1

Since x = 8 bits then $\left|B\right|$ = 8 bits. The sub blocks are:

K_{B_0}=01000010, K_{B_1} = 00100001, K_{B_2} = 10010000, K_{B_3} = 01001000, K_{B_4} =00100100, K_{B_5} =10010010, K_{B_6} =01001001, K_{B_7} =10100100, K_{B_8} = 11010010, K_{B_9} = 01101001, $K_{B_{10}}$ = 10110100, $K_{B_{11}}$ = 11011010, $K_{B_{12}}$ = 01101101, $K_{B_{13}}$ = 00110110, $K_{B_{14}}$ = 10011011, $K_{B_{15}}$ = 11001101.

- **Data Encryption**

  Convert the message into binary as below:

M: hellopeople

The message after converting it to the binary:

0110100001100101011011000110110001101111011100000110010101101111011100000110110001100101.

Then the message is divided into sub blocks, x = 8 bits this mean $|B[M]| =$ 8 bits. The sub blocks of message are:

M_{B_0}= 01101000, M_{B_1}= 01100101, M_{B_2}= 01101100, M_{B_3}= 01101100, M_{B_4}= 01101111, M_{B_5}= 01110000, M_{B_6}= 01100101, M_{B_7}= 01101111, M_{B_8}= 01110000, M_{B_9}= 01101100, $M_{B_{10}}$= 01100101.

Then calculate $Cipher_i$. The sub blocks $Cipher_{B_i}$:

$Cipher_{1_{B_0}}$= 01101010, $Cipher_{1_{B_1}}$= 01000100, $Cipher_{1_{B_2}}$= 11111100, $Cipher_{1_{B_3}}$= 00100100, $Cipher_{1_{B_4}}$= 01001011, $Cipher_{1_{B_5}}$= 11100010, $Cipher_{1_{B_6}}$= 00101100, $Cipher_{1_{B_7}}$= 11001011, $Cipher_{1_{B_8}}$= 10100010, $Cipher_{1_{B_9}}$= 00000101, $Cipher_{1_{B_{10}}}$= 11010001.

The encryption steps implemented on sub blocks $Cipher_{1_{B_i}}$ in the Table 3-5 below with the sub blocks, start by select random state from Q set. See Appendix A and Table 3-10 to understand the encryption process.

The steps of the encryption as follows:

1- Select state 1, v_0= 5= 00000101,01101010⊕00000101=01101111=V_{B_1}.

2- State 2, v_2 =22 =00010110, 01000100⊕00010110 =01010010 =V_{B_2}.

3- Sate 1, v_2 =3 =00000011, 11111100⊕00000011 =11111111 =V_{B_3}.

4- State 3, v_2 =11 =00001011, 00100100⊕00001011 =00101111 =V_{B_4}.

5- State 7, v_0 =6 =00000110, 01001011⊕00000110 =01001101 =V_{B_5}.

6- State 6, v_0 =15 =00001111, 11100010⊕00001111 =11101101 =V_{B_6}.

7- State 1, v_3 =29 =00011101, 00101100⊕00011101 =00110001 =V_{B_7}.

8- State 6, v_1 =30 =00011110, 11001011⊕00011110 =11010101 =V_{B_8}.

9- State 8, v_3 =35 =00100011, 11100010⊕00100011 =11000001 =V_{B_8}.

10- State 6, v_2 =19 =00010011, 00000101⊕00010011 =00010110 =V_{B_8}.

11- State 3, v_0 =40 =00101000, 11010001⊕00101000 =11111001 =V_{B_8}.

Then get the values after the permutation process:

1- V_{B_1}=01101111, P_2 =8 6 1 2 4 3 5 7, P_{B1} = 11010111.

2- V_{B_2}=01010010, P_3 =1 6 2 5 8 3 4 7, P_{B2} = 00100011.

3- V_{B_3}=11111111, P_0 =1 4 6 8 3 5 2 7, P_{B3} = 11111111.

4- V_{B_4}= 00101111, P_1=4 7 3 2 1 8 6 5, P_{B4} = 01100111.

5- V_{B_5}= 01001101, P_0=2 7 5 8 1 4 6 3, P_{B5} = 10110010.

6- V_{B_6}= 11101101, P_2=3 8 2 1 6 4 7 5, P_{B6} = 11111001.

7- V_{B_7}= 00110001, P_3=1 5 6 3 8 2 4 7, P_{B7} = 00011010.

8- V_{B_8}= 11010101, P_1=8 1 2 4 7 6 5 3, P_{B8} = 11110100.

9- V_{B_9}= 11000001, P_3=6 1 2 4 8 7 3 5, P_{B9} = 01101000.

10- $V_{B_{10}}$=00010110, P_3=1 4 6 8 2 5 3 7, P_{B10}= 01100001.

11- $V_{B_{11}}$=11111001, P_2=5 7 4 3 1 8 2 6, P_{B11}= 10111110.

Table B.2 Encryption of The Permutation Stream Reaction Automata Direct Graph algorithm

	P_{B_j}	State number	jump	add2	b_n	Cipher text
0	11010111	1	-	00010000	0001000011010111	0101001111111111
1	00100011	2	J	00100010	0010001000100011	1100101001000100
2	11111111	1	-	00010010	0001001011111111	0111011100101100
3	01100111	3	-	00110010	0011001001100111	1111101111110011
4	10110010	7	J	01110000	0111000010110010	0011100001001100
5	11111001	6	-	01100000	0110000011111001	0101010101101101
6	00011010	1	-	00010011	0001001100011010	0101010010100101
7	11110100	6	-	01100001	0110000111110100	1011100110000110
8	01101000	8	J	10000011	1000001101101000	1111110111110001
9	01100001	6	-	01100010	0110001001100001	1001100000000001
10	10111110	3	-	00110000	0011000010111110	1010100111010000
					0000000000000000	1100000100101011

- **Data Decryption**

In the decryption process, the receiver work in reverse order for the steps of the encryption process. The steps of decryption process mention in the section (3.3.5.2) illustrated in the Table B.2, See Appendix (B.2) also to understand the decryption process.

Table B.3 Decryption of The Permutation Stream Reaction Automata Direct Graph algorithm

	Cipher text	**b_n**	**State number**	**jump**	**add2**	**P_{B_j}**
10	1111110111110001	1000001101101000	3	-	00110000	10111110
9	1001100000000001	0110001001100001	6	-	01100010	01100001
8	1010100111010000	0011000010111110	8	J	10000011	01101000
7	1100000100101011	0000000000000000	6	-	00010011	11110100
6	0011100001001100	0111000010110010	1	-	01100000	00011010
5	0101010101101101	0110000011111001	6	-	01100000	11111001
4	0101010010100101	0001001100011010	7	J	01110000	10110010
3	1011100110000110	0110000111110100	3	-	00110010	01100111
2	0101001111111111	0001000011010111	1	-	00010010	11111111
1	1100101001000100	0010001000100011	2	J	00100010	00100011
0	0111011100101100	0001001011111111	1	-	00010000	11010111
	1111101111110011	0011001001100111				

Then excuted inverse permutation at each block in colmun P_{B_j}. Then XORed the inverse permutation value with the value stored in the state, the resulted value XORed with the key to return the orginal message.

APPENDIX (B.2): TABLES OF PERMUTATIONS S-RADG ALGORITHM WITH PERMUTATION VALUES AND THE TRANSITIONS

State no.	State value	P0								Transition
1	9	2	6	7	4	5	8	1	3	4
2	3	3	6	7	5	4	1	2	8	3
3	12	2	4	6	7	5	3	1	8	1
4	-	-	-	-	-	-	-	-	-	4
5	-	-	-	-	-	-	-	-	-	5
6	23	2	6	7	5	8	3	1	4	1
7	27	*2*	3	6	4	1	8	5	7	6
8	28	3	7	2	5	1	8	6	4	3

State no.	State value	P1								Transition
1	2	7	5	2	8	6	3	1	4	5
2	2	1	4	8	2	3	6	7	5	6
3	1	4	7	3	2	1	8	6	5	1
4	-	-	-	-	-	-	-	-	-	4
5	-	-	-	-	-	-	-	-	-	5
6	21	8	1	2	4	7	6	5	3	5
7	22	1	3	8	5	7	2	6	4	7
8	16	2	8	6	4	1	3	5	7	1

State no.	State value	P2								Transition
1	3	7	5	3	4	2	*1*	6	8	7
2	11	8	5	7	1	3	2	4	6	1
3	13	5	7	4	3	1	8	2	6	2
4	-	-	-	-	-	-	-	-	-	4
5	-	-	-	-	-	-	-	-	-	5
6	20	3	8	2	1	6	4	7	5	3
7	24	8	4	1	2	5	3	6	7	3
8	18	1	4	5	8	2	7	3	6	6

State no.	State value	P3								Transition
1	3	1	5	6	3	8	2	4	7	6
2	4	1	6	2	5	8	3	4	7	4
3	19	8	1	2	3	5	6	4	7	2
4	-	-	-	-	-	-	-	-	-	4
5	-	-	-	-	-	-	-	-	-	5
6	15	1	4	6	8	2	5	3	7	1
7	26	6	2	1	3	8	5	7	4	2
8	11	6	1	2	4	8	7	3	5	6

الملخص

يحتوي الويب على عدد هائل من المعلومات الرقمية ، لذا فإن امكانية الوصول الى المعلومات الحساسة عبر الويب تتطلب جهودًا هائلة من حيث التحكم في الأمان والتحقق من الوصول إلى المحتويات ، و من أجل تحقيق ذلك قدمت العديد من خوارزميات التشفير.

تقدم هذه الأطروحة خوارزميتين لحماية البيانات المتنقلة عبر الشبكات من خلال تطوير خوارزمية بدون مفتاح تسمى RADG الى خوارزميةReaction} S-RADG (Stream RADG Automata Direct Graph}). حيث تعتمد خوارزمية RADG على نظرية الرسم البياني ولا تتطلب أي مفتاح لتشفير البيانات اعتمادًا على العشوائية والرسالة الأصلية في التشفير. الخوارزمية الجديدة خوارزمية (S-RADG) مصممة على أساس نفس خصائص(RADG نظرية الرسوم البيانية والتشفير العشوائي) وتعتمد على التشفير السيلي لتوليد المفتاح. تشفر الخوارزمية الجديدة bit واحد في الوقت ذاته مثل التشفير السيلي.

تم تطوير خوارزمية S-RADG الى خوارزمية S-RADG بوجود عملية التبادل، تتعامل الخوارزمية المطورة مع البيانات مثل التشفير الكتلي، حيث تقوم بتشفير مجموعة من البتات في وقت واحد تسمى كتلة ، لذلك فهي تشفر حجمًا كبيرًا من البيانات. إن عملية التبادل في خوارزمية التقليب-S RADG تجعلها أقوى وتزيد من العشوائية مما يزيد من تعقيد الخوارزمية ، ويصعب كسرها من قِبل الهاكر وتحتاج إلى مزيد من العمل أثناء عملية فك التشفير لإعادة النص العادي. يتم إنشاء مفتاح الخوارزميات عشوائياً باستخدام أحد خوارزمية التشفير السيلي ، وهي(LFSR).

إن نتائج خوارزميات S-RADG و Permutation S-RADG هي نصوص مشفرة مختلفة لنفس النص العادي ، لأنها تعتمد على التشفير العشوائي.

تختلف النصوص المشفرة بشكل كبير كثيرًا ، مما يجعل عملية كسر الشفرات داخل الأنظمة الكبيرة صعبة للغاية مقارنة بالبرامج التي تعتمد على التشفير التقليدي بالمقارنة مع طريقة RADG. تم تصميم الخوارزميات المقترحة لغرض أمني مع تشفير عالي الكفاءة للبيانات ، كما أن تحليل الخوارزميتين المقترحتين أثبت شروط السرية وسلامة البيانات والتوثيق.

جمهوريه العراق
وزارة التعليم العالي والبحث العلمي
جامعة النهرين
كليه العلوم
قسم الحاسبات

تحسين خوارزميات الرادج السيلية

رسالة مقدمة الى كلية العلوم في جامعة النهرين كجزء من متطلبات نيل شهادة الماجستير في علوم الحاسبات

من قبل

ضحى عامر مهدي

(بكالوريوس علوم حاسبات، 2014)

بأشراف

أ.د. صلاح عبد الهادي البيرماني

د. سوسن كمال ثامر

1439 2018