CHAOS-BASED ADVANCED ENCRYPTION STANDARD

Thesis by

Naif B. Abdulwahed

In Partial Fulfillment of the Requirements for the degree of

Master of Science

King Abdullah University of Science and Technology

Computer Science Program

Thuwal, Makkah Province, Kingdom of Saudi Arabia

May, 2013

The undersigned approve the thesis of Naif B. Abdulwahed

Dr. David Keyes Committee Chair (Thesis Supervisor)	Signature	Date
Dr. Khaled Salama Committee Member (Thesis Supervisor)	Signature	Date
Dr. Mohammed-Slim Alouini Committee Member	Signature	Date
Dr. Hussam A. H. Fahmy Committee Member	Signature	Date

King Abdullah University of Science and Technology
2013

ABSTRACT

Chaos-Based Advanced Encryption Standard

Naif B. Abdulwahed

This thesis introduces a new chaos-based Advanced Encryption Standard (AES). The AES is a well-known encryption algorithm that was standardized by U.S National Institute of Standard and Technology (NIST) in 2001. The thesis investigates and explores the behavior of the AES algorithm by replacing two of its original modules, namely the S-Box and the Key Schedule, with two other chaos-based modules. Three chaos systems are considered in designing the new modules which are Lorenz system with multiplication nonlinearity, Chen system with sign modules nonlinearity, and 1D multiscroll system with stair case nonlinearity. The three systems are evaluated on their sensitivity to initial conditions and as Pseudo Random Number Generators (PRNG) after applying a post-processing technique to their output then performing NIST SP. 800-22 statistical tests. The thesis presents a hardware implementation of dynamic S-Boxes for AES that are populated using the three chaos systems. Moreover, a full MATLAB package to analyze the chaos generated S-Boxes based on graphical analysis, Walsh-Hadamard spectrum analysis, and image encryption analysis is developed. Although these S-Boxes are dynamic, meaning they are regenerated whenever the encryption key is changed,

the analysis results show that such S-Boxes exhibit good properties like the Strict Avalanche Criterion (SAC) and the nonlinearity and in the application of image encryption.

Furthermore, the thesis presents a new Lorenz-chaos-based key expansion for the AES. Many researchers have pointed out that there are some defects in the original key expansion of AES and thus have motivated such chaos-based key expansion proposal. The new proposed key schedule is analyzed and assessed in terms of confusion and diffusion by performing the frequency and SAC test respectively. The obtained results show that the new proposed design is more secure than the original AES key schedule and other proposed designs in the literature. The proposed design is then enhanced to increase the operating speed using the divide-and-conquer concept. Such enhancement, did not only make the AES algorithm more secure, but also enabled the AES to be faster, as it can now operate on higher frequencies, and more area-efficient.

ACKNOWLEDGMENTS

First and foremost, all praises be to Allah, the most merciful, the lord of the worlds, who taught man what he knew not. I would like to thank him for bestowing upon me the chance, strength and ability to complete this work.

Second, I would like to express the deepest appreciation to Prof. Khaled Salama and Prof. David Keyes without whom this thesis would not be possible. To Prof. Khaled Salama for his keen sense of understanding and strong support for my work and providing me with a great environment at his Sensors Lab where I have been able to work productively. To Prof. David Keyes who provided all the support and cooperation to ease the birth of this thesis and for the effort and time he invested to ensure its overall quality. I would also like to sincerely thank my thesis committee members (Profs. Mohamed Slim-Alouini and Hussam A. H. Fahmy), for reviewing my thesis and for their invaluable time, feedback and support. It is with immense gratitude that I acknowledge the support and help of M. Affan Zidan for introducing me to my research in the Chaos theory and for his keen insight and his positive feedback to always motivate me to perform the best. Special thanks to Mahmoud Ouda and all the members of Sensors Lab group who made my experience at the research work enjoyable and educational.

Lastly, but certainly not the least, I would like to thank my family for their support and their prayers for me which have been the driving force that enabled me to finish this thesis.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
CSB-Boxes	Chaos Based Substitution Boxes
CB-PRNG	Chaos Based Pseudo Random Number Generator
DSP	Digital Signal Processing unit
FPGA	Field Programmable Gate Array
GF(2^8)	Galois Field
LUT	Look-Up Table
MAD	Mean of Absolute Deviation
NIST	National Institute of Standards and Technology
ROM	Read-Only Memory
SAC	Strict Avalanche Criteria
S-Box	Substitution Box

LIST OF SYMBOLS

α	Significance Level to accept the null hypothesis
Nr	Number of Rounds in the AES Algorithm

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter I

Introduction

Chaos theory is the study of the behavior of dynamical systems that are very sensitive to initial conditions. Started originally as field of study in mathematics [1-3], chaos theory has also been developed and investigated by other different research areas including physics, chemistry and biology [4-6]. Although these dynamical systems are based on deterministic models, their high sensitivity to initial conditions or control parameters cause their solutions to be unpredictable [7]. The chaotic behavior of such systems can be observed in many natural phenomena such as astronomy and weather [8, 9]. In fact, the early discovery of chaotic systems returns back to the 1880's by Henry Poincare in his attempt to prove the stability of the solar system through his work on the restricted three-body problem [10]. Subsequent work in weather prediction by Edward Lorenz in 1961 has also contributed to the chaos theory [11].

Since the discovery of the existence of such chaotic systems and the increasing number of the applications they are involved in, especially in communication engineering [12, 13], researchers have been proposing many circuit implementations and optimization techniques for chaos generators. The majority of such chaos circuit implementations are focused on analog designs [14-18]. In general, analog circuit implementations of chaos generators are sensitive to the

operating conditions, process variations and temperature. Furthermore, the fact that these analog implementations are realized by using capacitors makes such chaos generators area inefficient and also initial conditions cannot be set accurately. These limitations make the digital implementation of chaotic systems desirable, since they overcome such issues. In turn, the digital implementation of chaos generators are mostly concentrated to date on logistic maps [19, 20] and most recently on third-order Ordinary Differential Equations (ODE's) [21-23].

One of the interesting applications of chaos is in the field of cryptography. Cryptography is concerned with techniques to secure the transfer of messages between two parties by means of encrypting such messages. Many researchers have emphasized the strong relationship between chaos and cryptography [24-27]. Even more interestingly, the idea of using chaotic systems in cryptography predated the popularization of the word "chaos", when Shannon pointed out in his classical paper: "Communication Theory of Secrecy Systems", published in 1949 [28], that good mixing transformations in good secrecy systems are achieved by basic operations that are the heart of chaotic maps [29]. In fact, in the same paper he invented two terms that are considered the most fundamental concepts of block ciphers, namely "confusion" and "diffusion". The mixing property and sensitivity to initial conditions of chaos generators are mapped to the diffusion and confusion of cryptosystems. Accordingly, such tight relationship between chaos and cryptography has led to a new field of research called chaotic cryptography and much work has been published [26, 30-36].

The fact that chaos generators have been implemented in the digital domain have boosted and enriched the research works in their application in encryption

and in cryptography in general where many chaotic ciphers in the 21st century have been proposed [30, 37-43]. Chaos was exploited in the two types of modern encryption which are symmetric-key encryption in its both forms; stream ciphers and block ciphers, and public-key encryption. In stream ciphers, the digital chaos generators were mainly used to generate a stream of pseudo-random numbers that used as keys to mask the plaintext as in [44-46]. For block ciphers, the key or the plaintext were used as the initial conditions and control parameters of the digital chaos generators and then iterated for several cycles or rounds to generate the cipher text as the image encryption block ciphers proposed in [24, 39, 47]. On the other hand, the application of digital chaos generators in public-key encryption received much less attention and only few works have been proposed in such category [48, 49].

All the chaos-based encryption algorithms mentioned above, whether in private-key or in public-key encryption, are independent in their design of any previously existing algorithms. Meanwhile, as such chaos-based encryption system have been developed and proposed, cryptanalytic works, which is the science of breaking encryption algorithms and finding their weaknesses and flaws, have been developed [25, 33, 37, 41, 50-53]. The work of this thesis mainly investigates the behavior of digital chaotic systems in a strong well-known block ciphering algorithm by replacing two of its essential components with chaotic ones and thus adding more security step to the original algorithm instead of building new algorithms from scratch. This block cipher is the well-known Advanced Encryption Standard (AES) and the modified components are its S-Box and Key Schedule.

Our proposal to modify the originally existing block ciphers is not the first. Kocarev et al [25] proposed chaos based Data Encryptions Standard (DES) using chaotic maps. Similarly, the idea of constructing chaos-based S-Boxes to be used in AES, DES or in other algorithms in general introduced and proposed in [54-66] and few non-chaotic S-Boxes design were proposed in [67-69]. Equally important, new alternative chaos–based designs has been proposed for the AES Key Schedules in [70-72], and chaos-free key schedules were proposed in [73-78]. The above chaos-based proposed designs are all based on chaos maps where the work in this thesis, and as one of its contributions, uses chaos generators based on ODE's.

The organization of this thesis is as follows: The Advanced Encryption (AES) is explained and described in Chapter 2. This Chapter begins with a quick introduction to Cryptography and Encryption Algorithms and their categories and how they are evaluated. Then, the round operations performed by the AES algorithm which are Add Round Key, Shift Rows, Substitute Bytes and Mix Columns are explored. Moreover, more attention is given for describing the operation of the AES Key Schedule for its three versions of 128-, 192-, and 256-bits key lengths. The hardware implementation of the AES that will be used throughout the Thesis will be introduced at the end of this Chapter with emphasizing on the implementation of the S-Box and the Key Schedule.

Chapter 3 introduces the three chaos systems that will be used to modify the AES design. These systems are Lorenz, Chen, and 1D Multiscroll chaos generators. The chapter starts by describing the mathematics behind these systems and examines the shape of their attractors and their score of the Maximum Lyapunov Exponent (MLE). Then, the three systems are evaluated as they are

used on constructing Pseudo Random Number Generators (PRNG) after enhancing their outputs by applying a post-processing technique. The NIST SP. 800-22 statistical testing suite is used to test each output variable of these PRNG's and to help determine which output variable to use further given the used post-processing technique. Moreover, the tree chaos-based PRNGS (CB-PRNG) are evaluated for their sensitivity to the initial conditions. The chapter ends by providing the FPGA hardware utilization estimate in terms of area along with the operation frequency and throughput.

Chapter 4 implements a new design for dynamic chaos-base S-Boxes (CBS-Boxes) for AES. The chapter begins by briefly describing the properties of good cryptographic S-Boxes and a quick introduction to original S-Box of AES. The detailed design and hardware implementation of dynamic S-Boxes based on the three systems mentioned in Chapter 3 is provided in this Chapter. Furthermore, a complete platform to investigate and analyze the properties of the CBS-Boxes is developed. The analysis provided in this chapter is based on graphical analysis, Walsh-Spectrum analysis, and image encryption analysis, which are mainly used to test the quality of the CBS-Boxes in terms or randomness. The chapter ends by comparing the dynamically generated CBS-Boxes to existing proposed designs of chaos-based S-Boxes.

Chapter 5 introduces a new chaos based Key Schedule for AES. The chapter briefly describes the design philosophy of good key schedules. The defects of the original AES key schedule are also introduced which also gave the motivation behind our proposal. Detailed explanation of the design and the hardware implementation of the chaos based key schedule is given in this chapter. A parallel

architecture of the design is also proposed. In order, to assess the quality of the proposed chaos-based key schedule two tests have been performed which are SAC and frequency tests. These tests are also applied on the original AES key schedule where the obtained results support the claims about the discussed defects. As an attempt to increase the operation frequency of the proposed design, a new enhancement based on the concept of divide-and-conquer is introduced later to the design. The new enhanced design is revaluated in terms of security, speed and hardware utilization and compared to existing works.

Finally, a summary of the thesis is given in the conclusion along with some outlines for potential research work.

I.1 Objectives and Contribution

The main objective of this thesis is to investigate and study the application of chaos based random number generators in a widely used encryption algorithm, namely the Advanced Encryption Standard. Subsequently, we propose a new algorithm along with the digital implementation of the chaos-based encryption system.

The main contributions of this thesis include:

- FPGA implementation of chaos-based dynamic 8X8 S-boxes for AES.
- The analysis of the sensitivity of the initial condition bits of the chaos-based pseudo random generator.

- The implementation of a MATLAB package to compute the Walsh-Hadamard Transformation of the S-boxes Boolean functions to measure their cryptographic features.
- The analysis of the chaotic based pseudo random number generators as they are applied in the generation of encryption S-boxes using Walsh-Hadamard Spectrum and image encryption analysis.
- The implementation of a novel parallel chaos-based 128-, 192- and 256-bits key schedule for the Advanced Encryption Standard, that is shown to be more secure, faster and less hardware demanding.

Chapter II

Cryptography and AES

Cryptography is a research field of Mathematics and Computer Science concerned with techniques to secure the transfer of messages between two parties. It is always assumed that a third party, often called adversary, can listen to the communication channel carrying such a message and accordingly can acquire the message. Preventing such adversary from understanding the message is achieved by applying a cryptographic mechanism called encryption algorithm using a piece of information that is only know to the communicating parties and explicitly unknown to the adversary. This piece of information is called the key of the encryption algorithm and the algorithm along with the key is called a cryptosystem

II.1 Encryption Algorithms

The main mission of an encryption algorithm is to convert a plaintext to a ciphertext using an encryption key. This process must be invertible such that knowing the same key the ciphertext can be converted back to the original message or the plaintext, a process called decryption. In order to ensure that such conversion of the plaintext to ciphertext is effective, most encryption algorithms are designed based on the confusion and diffusion concepts proposed by Shannon in [28]. The term confusion means no statistical analysis on the ciphertext should

reveal any information regarding the encryption key. On the other hand, the term diffusion means that the information of the plaintext should be highly spread in the ciphertext such that no statistical analysis on the ciphertext can reveal any information about the plaintext. The easiest way to achieve such properties is to use substitution-permutation networks (SPN) in building encryption algorithms such as the Advanced Encryption Standard (AES) that will be described later in this chapter.

II.1.1 Types and Categories

Modern encryption algorithms can be divided into two types according to the encryption scheme used. The encryption scheme can either be symmetric by using a private key only, or asymmetric by using a public key and a private key. Symmetric encryption algorithms, as the name might indicate, use the same key in encryption and decryption, and hence the key in such scheme should be private and only shared between the two communicating parties. In such a scheme, the decryption process usually uses the inverse operations used in the encryption process to recover the plaintext from the ciphertext. Symmetric encryption algorithms can also be subcategorized to stream ciphers, where data is encrypted one bit at a time, or block ciphers where data is partitioned into blocks of fixed sizes and encrypted one block at a time. Examples of Symmetric block ciphers are Data Encryption Standard (DES) [79], Twofish [80], Serpent [81] and the AES [82], and stream ciphers like RC4 [83] and Sober-128 [84].

On the other hand, asymmetric key encryption algorithms uses a public know key to encrypt the message, however, the decryption is done by using a

private key. Therefore, the public key can be safely published so anyone can use it to encrypt the message but only a private secret key that is only known to the legitimate receiver can be used to decrypt the message. Examples of public key encryption algorithms are RSA [85] and ElGamal [48].

II.1.2 Evaluation and Performance Criteria

Encryption algorithms are usually measured and evaluated for two criteria: security and speed. Security is one of the difficult assessments of encryption algorithms since there is no absolute judgment on whether a certain algorithm is secure. Usually, the designers of encryption algorithms conduct some security analysis of their proposal at the same time they publish their design and claim that their design is secure until a successful cryptanalysis of the proposed design is published pointing out that the algorithm is broken and is not secure any more. Such cryptanalysis work often aims to find some weaknesses in the algorithm trying to exploit the part/round where the cipher fails to successfully satisfy the diffusion or confusion properties discussed in the beginning of this chapter. In general, the security analysis of encryption algorithms is performed by applying some statistical tests that measure at what level is the confusion and diffusion satisfied.

Speed, on the other hand, is easier to measure. Typical speed measurement of encryption algorithms is done by monitoring the amount of time in terms of clock cycles in which the algorithm finish encrypting or decrypting an arbitrary set of data. In addition to the speed, encryption algorithms also are evaluated according to their simplicity in both description and implementation.

II.2 Advanced Encryption Standard, AES

The Advanced Encryption Standard (AES) is a symmetric, private key block cipher that can encrypt data blocks of 128 bits using symmetric keys of 128, 192 or 256 bits. The algorithm was originally designed by Joan Daemen and Vincent Rijmen and was originally called Rijndael. The designers of Rijndael submitted a proposal of the cipher to be evaluated by the U.S. National Institute of Standards and Technology, (NIST). In 2001, it was announced by NIST as the AES to be the successor of the Triple Data Encryption System (3DES) that was universally used [82]. The only modification done to Rijndael cipher to become the AES was only fixing the length of data block to only be 128-bits, unlike the original algorithm that was allowing a data length of 128, 192 or 256-bits. Being the winner among 5 finalists ciphers, AES is believed to be reasonably secure and hence is used widely around the world in many products in both hardware and software and in many online application through the internet.

The AES algorithm design is based on the substitution-permutation network concept where the bytes of the plaintext are substituted and permuted at each round through four operations. These operations are Add Round Key, Substitute Byte, Mix Columns and Shift Rows. These four operations are repeated in every round, but however, their order differs depending whether the main operation is encryption or decryption. Figure 1 shows the general structure of the AES algorithm. The number of rounds (Nr) or iterations depends on the master key length. The algorithm performs Nr =10, 12, 14 rounds for key lengths 128-, 192- and 256-bits respectively. For each round, a fixed length of 128-bits key are

generated by the Key Schedule operation that expands the master key to a number of sub keys depending on the number of rounds.

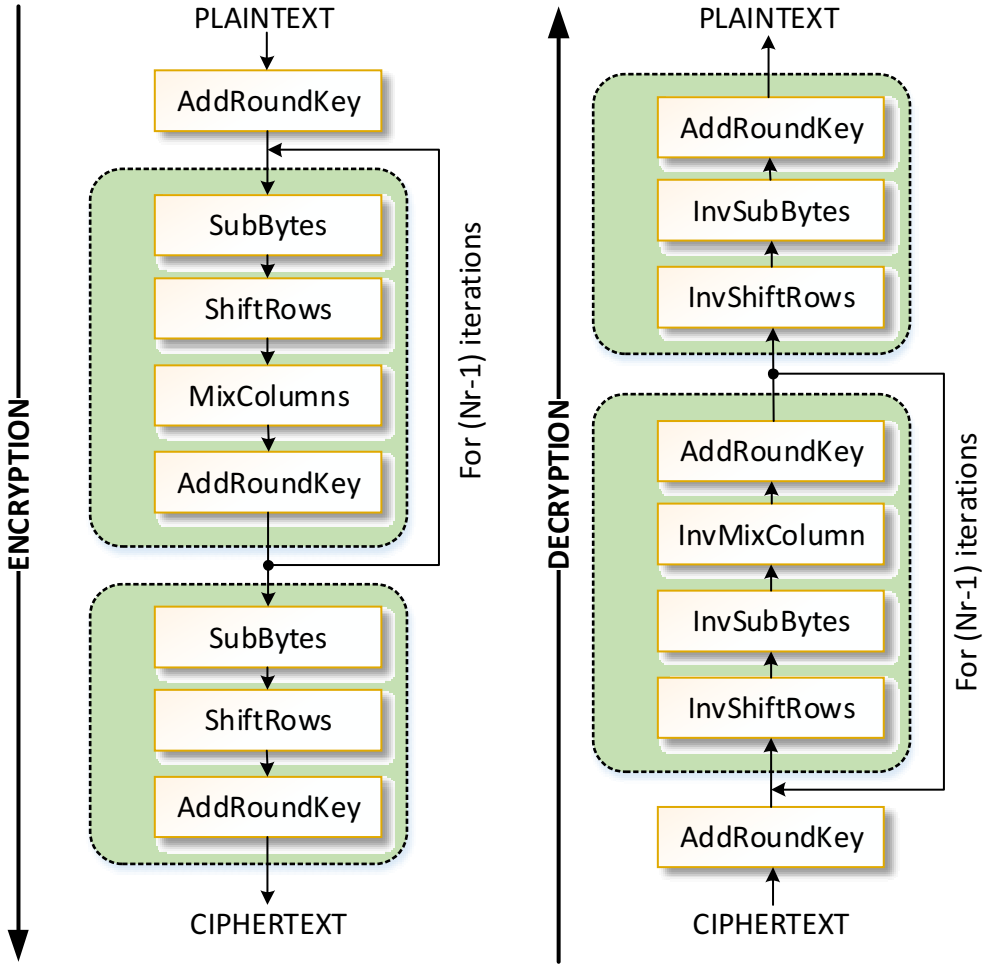


Figure 1 : AES's encryption and decryption operations.

II.2.1 Round Operations

The encryption process of the AES performs the four operations iteratively to the plaintext until the number of the required rounds is reached. When the data is fed to the AES algorithm, it is transformed to column-wise ordering before the round operations process the plain bytes. The plaintext is 128-bits long consisting of 16

bytes. If the bytes are numbered from B_{00} up to B_{15}, then the plaintext is stored as a 4x4 matrix in a structure called a state as Figure 2 illustrates and kept this way throughout the encryption process. The four operations are described next.

B_{00}	B_{04}	B_{08}	B_{12}
B_{01}	B_{05}	B_{09}	B_{13}
B_{02}	B_{06}	B_{10}	B_{14}
B_{03}	B_{07}	B_{11}	B_{15}

Figure 2: 4x4 data state array

II.2.1.1 Add Round Key

In the Add Round Key operation, the state (or the plaintext) is added with the sub key corresponding to the current round and which had been already created by the Key Schedule. This addition is merely an Exclusive OR (XOR) operation performed bitwise from the key and the state. Figure 3 illustrates this operation. In the AES, the Add Round Key operation is performed once with the master key before the algorithm proceeds with the regular rounds operations.

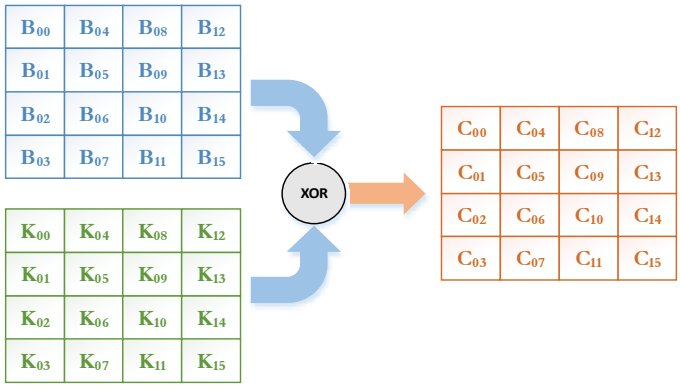


Figure 3: Add Round Operation

II.2.1.2 Sub Byte

In this operation, all the bytes in the state are substituted with another different bytes according to an 8-bit look-up table called the Substitution Box (S-Box). The sub byte operation is the nonlinear step in the AES algorithm and hence provides the confusion requirement in the encryption process. Since the expected values of a byte cannot be out of the range [0, 255], the S-Box have 256 entries to provide a substitution for every possible byte value. The S-Box is often visualized as a 16x16 hexadecimal table where a row is indexed by the most significant four bits of the input byte and a column is indexed by the other least significant bits. Finally, the entry will be the corresponding substitution for that byte. Figure 4 and 5 illustrate such Byte Sub operation and S-Box.

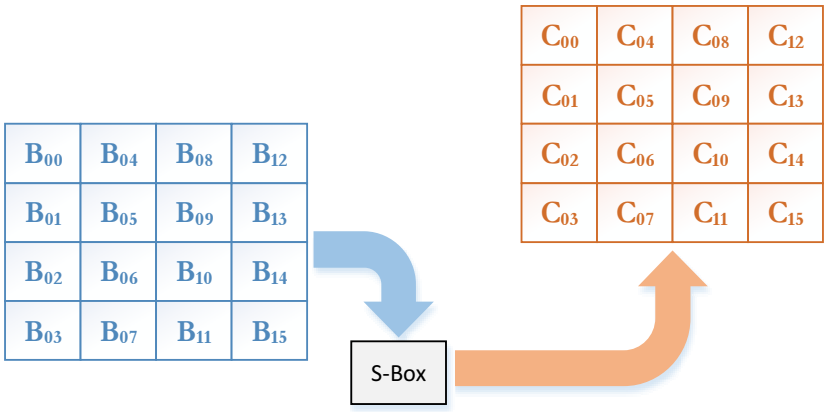


Figure 4: Substitute byte operation.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
10	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
20	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
30	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
40	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
50	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
60	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
70	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
80	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
90	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A0	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B0	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C0	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D0	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E0	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F0	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 5: S-Box used in substitute byte operation.

II.2.1.3 Shift Rows

This operation simply performs byte shifting on the current state. The operation is illustrated in Figure 6. The first row of the state is left untouched. However, the remaining three rows are shifted to the left each by different amount where the second row is shifted one byte, the third row is shifted by two bytes and finally the fourth row is shifted by three bytes.

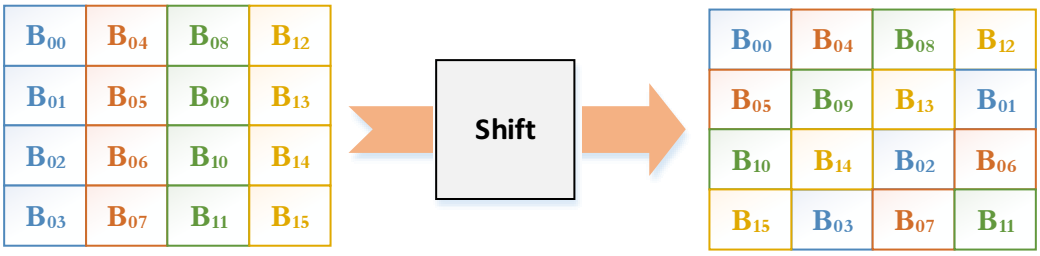


Figure 6: Shift Row operation.

II.2.1.4 Mix Columns

In this operation, each column which consists of four bytes is multiplied by a known 4x4 matrix defined as follows:

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

However, the multiplication performed on this matrix is not a normal multiplication. Rather, the multiplication is carried over a Finite Field called Galois-Field (GF), where the multiplication operation can be defined as follows [86]: Multiplication by 1 means do nothing, multiplication by 2 means shift to the left, and multiplication by 3 means shift to the left and XOR with the operand. Figure 7 illustrates the Mix Column operation. The Mix Column together with the Shift Rows operations provide the diffusion step in the AES encryption process.

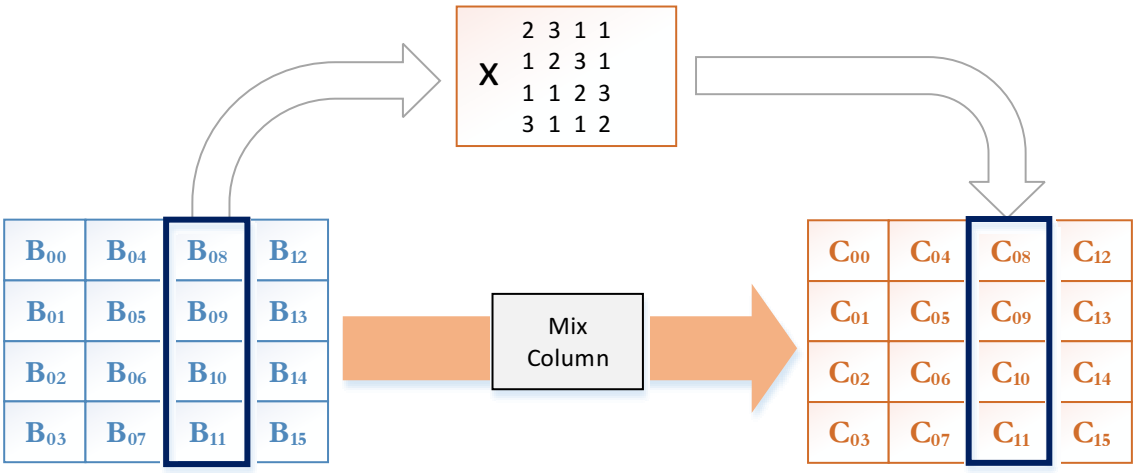


Figure 7: Mix Column operation.

II.2.2 Inverse Round Operations

The inverse round operations are the operation performed in the decryption process to transform the cipher text back to plaintext. This requires all the round operation of the AES algorithm to be reversible. The easiest and most straightforward operation to be reversed is the Add Round Key where the cipher text is XORed back with the same used sub-key to recover the original state. Reverse of Shift Rows operation is shifting in the opposite direction with the corresponding amount in each row.

The reverse of the Mix Column operation is achieved through a multiplication, over the finite field GF, by another special matrix defined as follows:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

After such multiplication takes place, the state of data before applying the Mix Column operation will be recovered.

Finally, the sub byte operation is reversed by substituting back all the bytes in the state to their original values. To do this, a substitution box called Inverse S-Box is used. This Inverse S-Box basically maps a byte back to its original value that was already substituted in the encryption process. The Inverse S-Box neutralizes the effect of the S-Box, such that if a certain byte is substituted using the S-Box and then substituted again using the inverse S-Box, this byte will not be changed and will stay as it is.

II.2.3 Key Schedule

The AES Key Schedule performs two main jobs: Key Expansion and Key Selection. In key expansion, the master key is expanded to create a number of sub-keys where the Key Selector chooses a sub-key for each round to be used in the Add Round Key operation. Therefore, the number of sub-keys is equal to the number of rounds to be performed and hence 10, 12 and 14 sub-keys will be generated for 128-bits, 192-bits and 256-bits encryption modes respectively. In general, the number of needed sub-keys is equal to Nr.

Similar to the data block state, a round key or the master key is stored in a 4x4 column-wise matrix. Each column represents a word, W where each word consists of four bytes. The key Expansion algorithm works on each column at time where it applies there operations which are rotating a word, substitute using the S-Box, and XORing. The algorithm is described next.

The creation of each sub-key can be divided into two parts. First part is creating the first word or column of the sub-key. The second part is creating the remaining columns of each sub-key. Figure 8, illustrates the operations involved in the first part. The creation of each word of the sub-key involves two words from the previous sub-key. If the first word of a certain sub-key is denoted by W_i, then the first and last words of the previous sub-key W_{i-1} and W_{i-4} respectively will contribute in creating W_i. Clearly, the first sub-key (sub-key 1) will use two words from the master key. Word W_{i-4} will be processed by three operations; rotating, substituting its individual bytes using the S-Box, then XOR'ing with word W_{i-4} and with a defined round constant from the Round Constant Array Rcon. The Rcon

array contains some defined constants in GF(2^8) field and is indexed by the sub-key number so that each sub-key uses only one constant from the array [87].

The second part of creating a sub-key involves only XOR operation with two words from the previous sub-key. Hence, if the second word the sub-key is W_{i+1}, then W_{i+1} is just the results of XORing W_i and W_{i-3} ($W_{i+1} = W_i$ XOR W_{i-3}). This is illustrated in Figure 10. The general pseudo code for the 128-bits, 192-bits and 256-bits AES Key Expansion algorithm [73] is described Figure 9. For the decryption process, the same sub-keys are generated but are scheduled to the Add Round Operation in the reverse order they were scheduled for the encryption process.

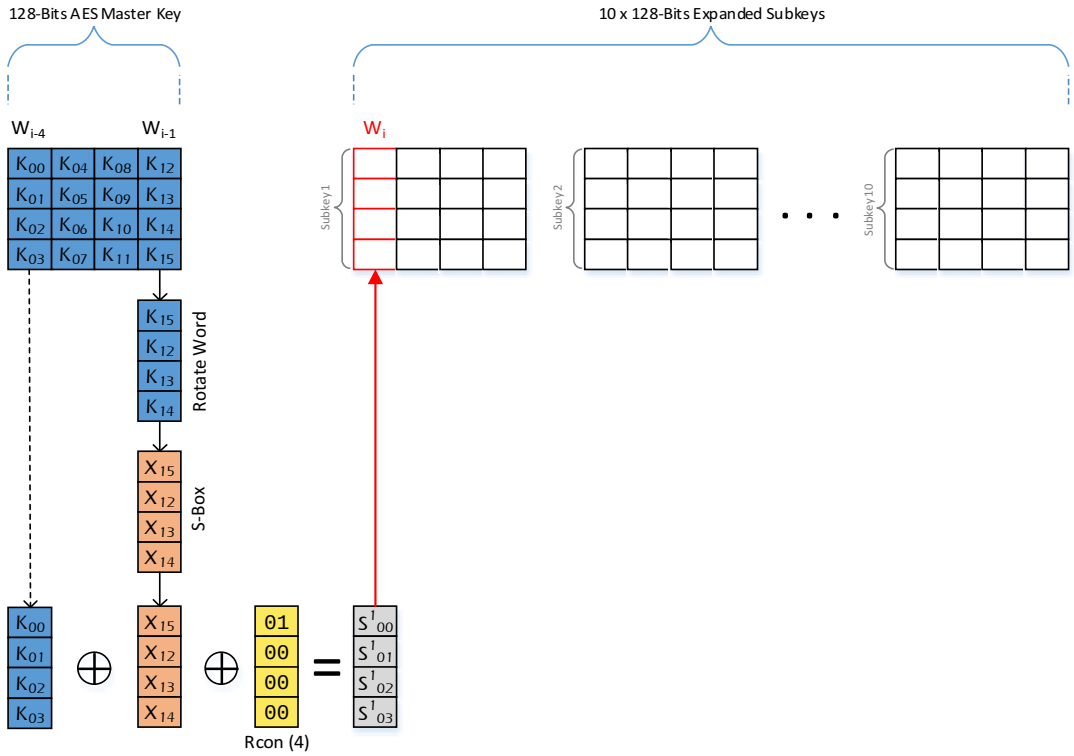


Figure 8: Key Expansion operation (first word of a sub-key).

128-bit Key	192-bit Key
for i = 0 to 3 W[i] = MasterKey[i] for j = 4 to 40 (in steps of 4) W[j] = W[j-4]. SubByte(Rotl(W[j-1])). Rcon[j/4] for i = 1 to 3 W[i+j] = W[i+j-4]. W[i+j-1]	for i = 0 to 5 W[i] = MasterKey[i] for j = 6 to 48 (in steps of 6) W[j] = W[j-6]⊕ SubByte(Rotl(W[j-1]))⊕ Rcon[j/6] for i = 1 to 5 (and while i+j<52) W[i+j] = W[i+j-6]⊕ W[i+j-1]

256-bit Key

```
for i = 0 to 7
  W[i] = MasterKey[i]
for j = 8 to 56 (in steps of 8)
  W[j] = W[j-8]. SubByte(Rotl(W[j-1])). Rcon[j/8]
for i = 1 to 3
  W[i+j] = W[i+j-8]. W[i+j-1]
  W[j+4] = W[j-4]. SubByte(W[j+1])
for i = 5 to 7
  W[i+j] = W[i+j-8]. W[i+j-1]
```

Figure 9: Pseudo-Code algorithm for 128-, 192-, 256-bits AES Key Schedule.

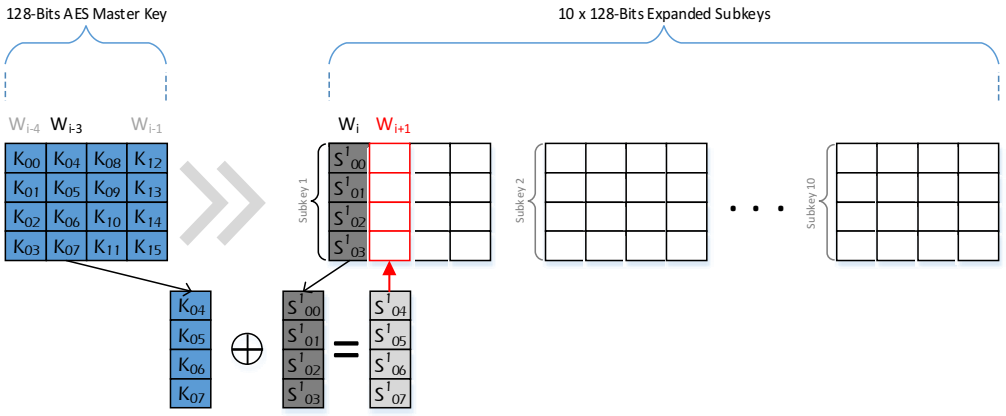


Figure 10: Key Expansion operation (remaining words of a sub-key).

II.3 Hardware Implementation of AES

Many hardware based implementation for AES have been proposed that optimize for speed [88], [89], [90], [91] or for area [92], [93], [94]. In our design for the proposed chaos based S-Box and the chaos based Key Expander, we have adopted the design of the FPGA Verilog based AES proposed in [95]. The proposed design looks similar to that shown in Figure 11.

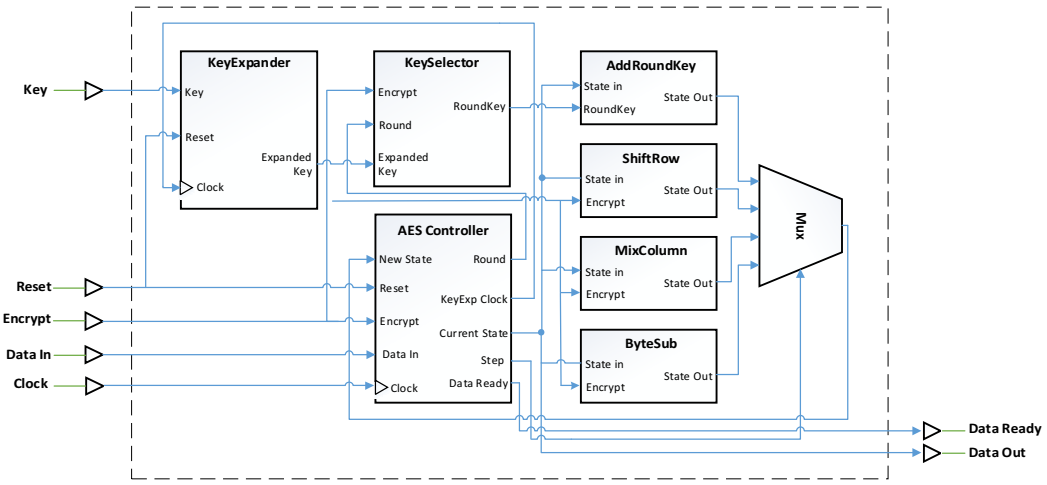


Figure 11: Verilog structure block diagram of AES.

In this design, each of the Round Operations mentioned in the previous sections is implemented in different module. The main controller then connects these different modules and synchronizes the encryption and the decryption operations. This simplifies our job in modifying the sub byte and the key schedule modules and replacing them with our proposed chaos based modules.

II.3.1 S-Box Implementation

S-Boxes are usually implemented using look-up tables. Such implementation on FPGA using HDL Verilog is done by hardwiring the substitution byte values in the Verilog code using wire assignments. This is done to implement both the S-Box and the Inverse S-Box. Moreover, this type of design is programmed on the FPGA by realizing Read Only Memory (ROM) units. Depending on the FPGA used and the ROM bit configuration supported, the number of look-up Tables (LUT) consumed will vary.

In the case of the Xilinx Virtex4 XC4VSX35-10FF668 FPGA, ROM units with bit configuration of 256x1 bit that consumes 16 LUTs is supported, and to support wider bit lengths these units can be cascaded. In order to store the S-Box using the look-up table approach, 256x8 bits are needed. Therefore, a total of 256x16 ROM units (about 256 LUTs) are required to realize the both the S-Box and its inverse. This number of LUT's will be multiplied by 16 resulting in a total of 4096 if parallel substitution of the 16 bytes of the state array takes place.

II.3.2 Key-Schedule Implementation

There are mainly two approaches to implement Key Schedules. First, the key schedule is fed with the master key and the main algorithm controller waits until all the sub-keys are generated before it starts the encryption process. The second approach is to generate the sub-keys on the fly. In other words, the algorithm controller does not have to wait for all sub-keys to be generated, instead it start the first encryption round whenever the first sub-key is generated, then second round starts when second sub-key is generated, and so on.

The original AES Key Schedule in this design uses the first approach and is implemented as a parameterized Verilog module. The module generates a new sub-key each 4 clock cycles by producing a word (4 bytes) per cycle. Moreover, since the AES key expansion uses the S-Box in its operation, the S-Box is embedded inside the key expansion module as a look-up table increasing the LUT's utilization of the key expansion module.

Chapter III

Digital Implementation of CB-PRNGs

In this chapter, we describe and investigate the chaos systems used in our work. We have used three chaos systems, where each system uses a different technique to achieve the nonlinearity. The chaos systems are: the Lorenz system, which is based on multiplication nonlinearity, then Chen system, which is based on Sign Modules nonlinearity, and a 1-Dimensional Multiscroll system based on Staircase nonlinearity. These systems have been demonstrated to generate chaotic dynamics by calculating the Maximum Lyapunov Exponent (MLE) of the output time series. Such MLE value should be positive and the greater this value is the greater the unpredictability and the chaotic sensitivity of corresponding chaos generator.

According to [96] and [97], chaos generators are one of the main techniques to generate random numbers. Therefore, in this thesis, we will not only exploit the chaos generators for their sensitivity to initial conditions and their mixing property, we will also take advantage of their ability to produce random numbers. Hence, in this chapter, the three systems are evaluated first for their suitability in building PRNG's by using a post-processing technique then applying the NIST SP. 800-22 statistical testing suite on their output. After that, a key-sensitivity test is proposed and applied to these generators to assess their sensitivity to the initial

conditions. Finally, the FPGA hardware resources utilization of the final chaos-based PRNG is computed using Xilinx synthesize tool.

III.1 Description of Chaotic Systems

Following are descriptions of the used three chaos systems. Each chaotic system uses a different set of differential equations to generate the chaos behavior. The chaotic dynamics are then realized by numerically solving these systems of equations in fixed point, for reproducibility. In general, there are many numerical techniques to solve such systems including Euler, Runge-Kutta and Midpoint methods [98]. The numerical technique used to solve the chaos system will affect its chaotic response [23]. It was shown that the Euler technique yields the best performance in terms of operating frequency and MLE for digital implementation of chaos systems. Therefore, we have used digital implementations of the three systems based on the Euler method.

III.1.1 System 1: Lorenz - Multiplication Nonlinearity

This system was developed by Edward Lorenz in 1960's at his attempt to model hydrodynamical phenomena such as the air flow in the atmosphere [11]. It represents a severe truncation of a spectral expansion of a fluid mechanical system. He was one of the first scientists to conduct a numerical study on chaos. In his model, numerical solutions of three ordinary differential equations (ODE) were used to prove the existence of deterministic non-periodic flow in the atmosphere. These three equations are defined as follows:

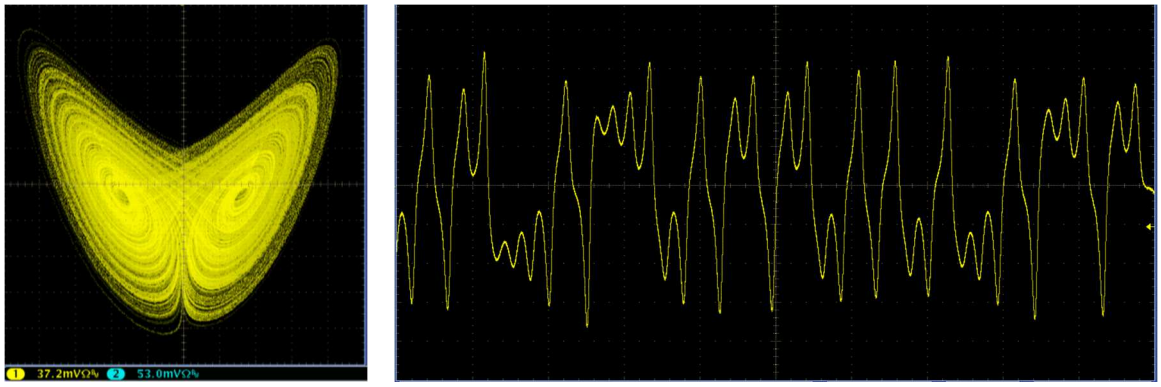
$$\dot{x} = -\sigma x + \sigma y \qquad \text{III.1-1}$$

$$\dot{y} = -xz + \rho x - y \qquad \text{III.1-2}$$

$$\dot{z} = xy - \beta z \qquad \text{III.1-3}$$

where x, y and z are the system state variables and $\sigma,\ \rho,$ and β are some constants that are used as the system parameters.

We have used the digital implementation of Lorenz system presented in [23] to solve the system equations using Euler approximation. To simplify the design of this implementation, the system parameters $\sigma,\ \rho,$ and β were chosen to be power-of-2 so that the multiplication can be realized as a shift register. For $\sigma = 8$, $\rho = 32$, and $\beta = 2$ and initial conditions $x_0 = 0,\ y_0 = 4$ and $z_0 = 4$, the $x - z$ projection producing the system unique attractor and x output versus time were captured using an oscilloscope and are shown in Figure 12. Furthermore, this implementation of the Lorenz system has scored an MLE of 1.667 using 250K iterations.



(a) (b)

Figure 12: Oscilloscope snapshots of Lorenz system. (a) Attractor projection on the x-z plane, (b) output x.

III.1.2 System 2: Chen – Sign, Modules Nonlinearity

This system was proposed in [99] to introduce a multiplication-free Lorenz system described in the previous section. The modifications come from observing that the multiplication terms in the original system can be replaced by other multiplication-free terms and still producing the same attractor as the original. In Equation **III.1-3**, the xy term is replaced by only the absolute value of y, while the multiplication by x in Equation III.1-2 is replaced by $sign$ function. Therefore, the absolute value function along with the $sign$ function provide the nonlinearity in the system instead of the multiplication. The ODE equations of the new system are then as follows:

$$\dot{x} = a(y - x) \qquad \textbf{III.1-4}$$

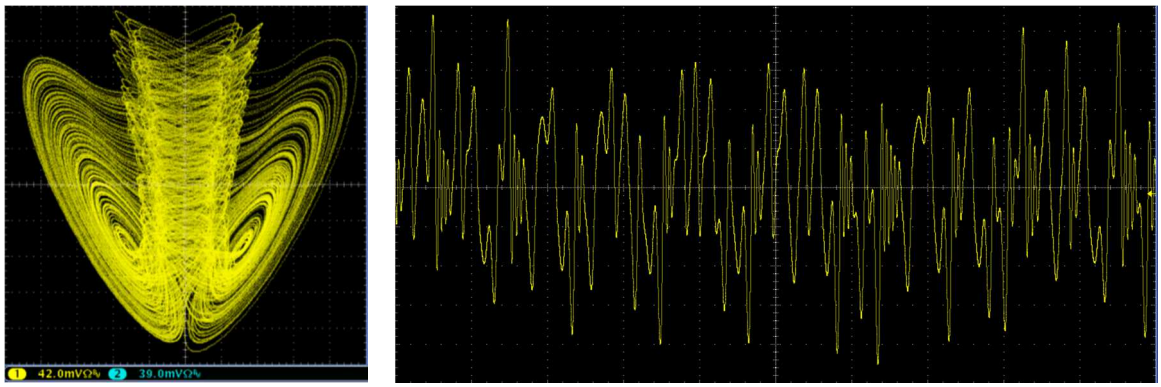
$$\dot{y} = sign(x)(b - z) + cy \qquad \textbf{III.1-5}$$

$$\dot{z} = |y| - dz \qquad \textbf{III.1-6}$$

where $a, b, c\ and\ d$ are the new system parameters and

$$sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \qquad \textbf{III.1-7}$$

The digital implementation of this systems that employs the Euler approximation to obtain the numerical solutions has been adopted from [23]. Specifying the system parameters as follows: $a = 1, b = 1, c = \frac{1}{2}, d = \frac{1}{8}$ and initial conditions $x_0 = 128,\ y_0 = 288,$ and $z_0 = 336$, the x-z projection producing the system attractor and x output versus time were captured using an Oscilloscope and are shown in Figure 13. Using the Euler approximation as the numerical solutions technique, the system scored an MLE of 0.299.



(a) (b)

Figure 13: Oscilloscope snapshots of Chen system. (a) Attractor projection on the x-z plane, (b) output x.

III.1.3 System 3: 1D Multiscroll – Staircase Nonlinearity

This system can be described by the following three ordinary differential equations that define a third-order Jerk system [100]:

$$\dot{x} = y \qquad \text{III.1-8}$$

$$\dot{y} = z \qquad \text{III.1-9}$$

$$\dot{z} = a\,(-x - y - z + F(x)) \qquad \text{III.1-10}$$

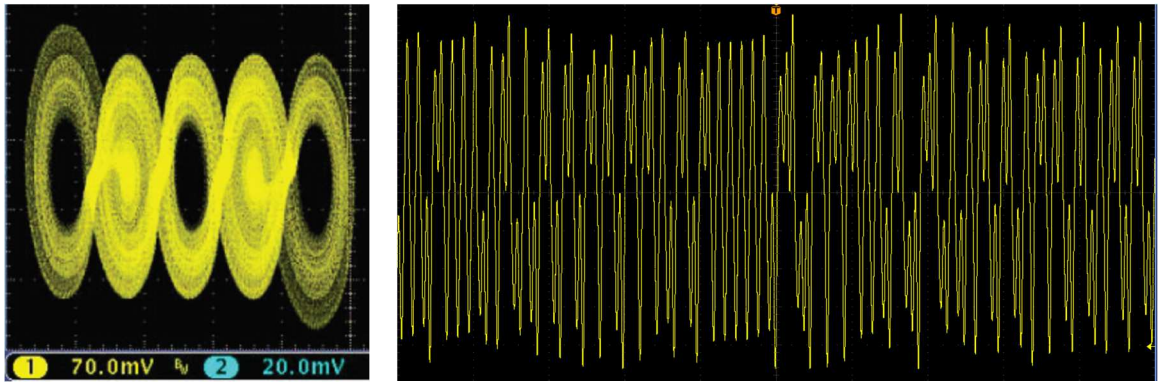
where $x, y,$ and z are system state variables, $a \in [0.47,\ 0.96]$, and $F(x)$ is a simple non-linear function. In this system, $F(x)$ is the stair case nonlinear function, which can be defined as:

$$F(x) = sgn(x) + \sum_{j=1}^{M} [sgn(x - 2j) + sgn(x + 2j)]$$

and $\boldsymbol{sgn(x)}$ is the sign function defined in III.1-7.

Solving these three equations numerically results in one-dimensional multi-scrolls, where the number of these multi-scrolls can be controlled by M. In our

design, we have adopted the digital implementation of 1D system that's proposed by [21]. This implementation uses the Euler approximation to solve the system numerically and provide a real-time controllable parameter to adjust the number of scrolls. For $a = 0.875$, and initial conditions $x_0 = 0, y_0 = 0.5$, and $z_0 = 0.5$, the x-y projection producing the attractor of five scrolls and the x output versus time were captured using an Oscilloscope and are shown in Figure 14. For the same parameters, the system scored an MLE value of 0.162554.



(a) (b)

Figure 14: Oscilloscope snapshots of 1D Multiscroll system. (a) Attractor projection on the x-y plane, (b) output x.

III.2 Evaluation of Chaotic Systems as PRND's

In this section, we evaluate the quality of the three chaos systems when implemented a PRNG's. First, we introduce the post-processing technique we have used to enable chaos generators to act as random generators. The NIST SP. 800-22 statistical testing suite [101] is used to measure the random output of these PRNG's. Then, the area and speed are measured for each system.

III.2.1 Post-Processing

In order to build a PRNG based on a chaos system, an additional step is required to enhance the statistical properties of the chaos generators. This step is called post-processing. This step is needed due to the fact that PRNG's based on chaos systems suffer from short term predictability [96]. The short term predictability can be realized by observing the output sequences from the raw output of the digital chaos generators. This causes certain bits of the output to be defective and consequently decrease the randomness quality of the system.

Some post-processing techniques including Von Neumann and bit-counting have been proposed [97]. However, these techniques do not always enable the chaos-based PRNG's to fully pass all the NIST SP. 800-22 statistical tests compared to the technique proposed in [102]. This technique basically suggests only suppressing the defective bits from the output of the digital chaos generators. By inspecting the output of the three chaos systems under our consideration, we can see that most of these defective bits are indeed apparent in the most significant half of the output. Thus, the post-processing we will be using to enable the three chaos system to act as good statistically PRNG's is to only suppress the most significant half of each output. For example, if the chaos generator produces a 32-bits output in each cycle, then we only consider the least significant 16-bits of the output. Therefore, if we want the chaos-based PRNG to generate an output of n-bits, we make its original output to be 2*n-bits and suppress the most significant n-bits.

III.2.2 NIST Randomness Test

In this section, we test the output of each variable of the CB-PRNG after applying the post-processing technique discussed in previous section using NIST SP. 800-22 statistical testing suite. The NIST tests were run on $32x10^6$ binary sequences from each CB-PRNG using 32-bits bus width and the initial conditions discussed in previous section. Tables III.2.1, III.2.2 and III.2.3 show metrics of test results of these sequences. Since each CB-PRNG provide three output, we have performed the tests for each variable alone as shown in the tables.

Table III.2.1: NIST SP. 800-22 test results for Lorenz PRNG.

NIST SP. 800-22 Results - Lorenz						
	X		Y		Z	
	PV	PP	PV	PP	PV	PP
Monobits	✓	1.0	✓	1.0	✓	1.0
Block Frequency	✓	1.0	✓	0.96	✓	1.0
Cumulative Sums	✓	1.0	✓	1.0	✓	1.0
Runs	✓	0.93	✓	1.0	✓	1.0
Longest Run	✓	1.0	✓	1.0	✓	1.0
Rank	✓	0.96	✓	1.0	✓	1.0
FFT	✓	0.93	✓	1.0	✓	0.96
N. O. Temp,	✓	0.93	✓	0.93	X	0.90
O. Temp.	✓	1.0	✓	0.96	✓	0.96
Universal	✓	1.0	✓	1.0	✓	0.96
App. Entropy	✓	1.0	✓	1.0	✓	1.0
R. Excur.	✓	0.93	✓	0.95	✓	0.94
R. Excur. Var.	✓	0.93	✓	1.0	X	0.90
Serial	✓	1.0	✓	0.93	✓	0.96
L. Complexity	✓	1.0	✓	0.96	✓	1.0

Table III.2.2: NIST SP. 800-22 test results for Chen PRNG.

NIST SP. 800-22 Results - Chen						
	X		Y		Z	
	PV	PP	PV	PP	PV	PP
Monobits	✓	1.0	✓	1.0	✓	0.96
Block Frequency	✓	0.96	✓	1.0	✓	0.96
Cumulative Sums	✓	1.0	✓	0.96	✓	0.96
Runs	✓	1.0	✓	1.0	✓	1.0
Longest Run	✓	1.0	✓	1.0	✓	1.0
Rank	✓	1.0	✓	1.0	✓	0.96
FFT	✓	1.0	✓	1.0	✓	0.96
N. O. Temp,	✓	0.93	✓	0.93	X	0.93
O. Temp.	✓	1.0	✓	0.96	✓	1.0
Universal	✓	0.96	✓	0.96	✓	0.93
App. Entropy	✓	1.0	✓	1.0	✓	0.96
R. Excur.	X	0.94	✓	0.93	✓	0.94
R. Excur. Var.	X	0.90	✓	1.0	✓	0.94
Serial	✓	0.96	✓	0.96	✓	1.0
L. Complexity	✓	0.96	✓	1.0	✓	1.0

Table III.2.3: NIST SP. 800-22 test results for 1D PRNG.

NIST SP. 800-22 Results – 1D						
	X		Y		Z	
	PV	PP	PV	PP	PV	PP
Monobits	✓	1.0	✓	1.0	✓	1.0
Block Frequency	✓	1.0	✓	0.96	✓	1.0
Cumulative Sums	✓	1.0	✓	1.0	✓	1.0
Runs	✓	0.96	✓	0.96	✓	1.0
Longest Run	✓	1.0	✓	1.0	✓	1.0
Rank	✓	0.96	✓	1.0	✓	1.0
FFT	✓	1.0	✓	1.0	✓	0.93
N. O. Temp,	✓	0.9	✓	0.93	✓	0.93
O. Temp.	✓	1.0	✓	0.93	✓	1.0
Universal	✓	1.0	✓	1.0	✓	1.0
App. Entropy	✓	1.0	✓	1.0	✓	0.96
R. Excur.	X	0.88	✓	0.95	✓	0.95
R. Excur. Var.	X	0.94	✓	0.95	✓	1.0
Serial	✓	0.93	✓	0.96	✓	0.96
L. Complexity	✓	1.0	✓	1.0	✓	1.0

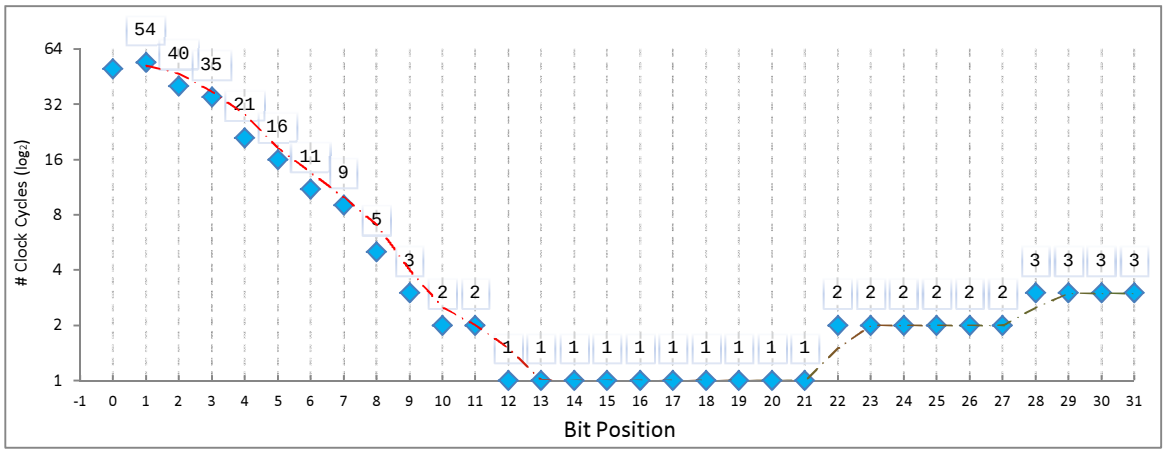
As we can see from, not all the output variables have passed the 15 NIST tests. In Lorenz system, X and Y passed but Z failed. In Chen System, only Y has passed. In 1D system, Y and Z have passed only. In an attempt to increase the throughput for the Lorenz and the 1D systems, we re-run the NIST tests again on binary sequences that are the concatenation of the passed variables. In the case or Lorenz, the test were performed on the concatenation of X and Y, and for 1D system concatenation of Y and Z were examined. The concatenations of these variables have indeed passed all the NIST tests causing the throughput of both systems to double.

III.2.3 Key Sensitivity Test

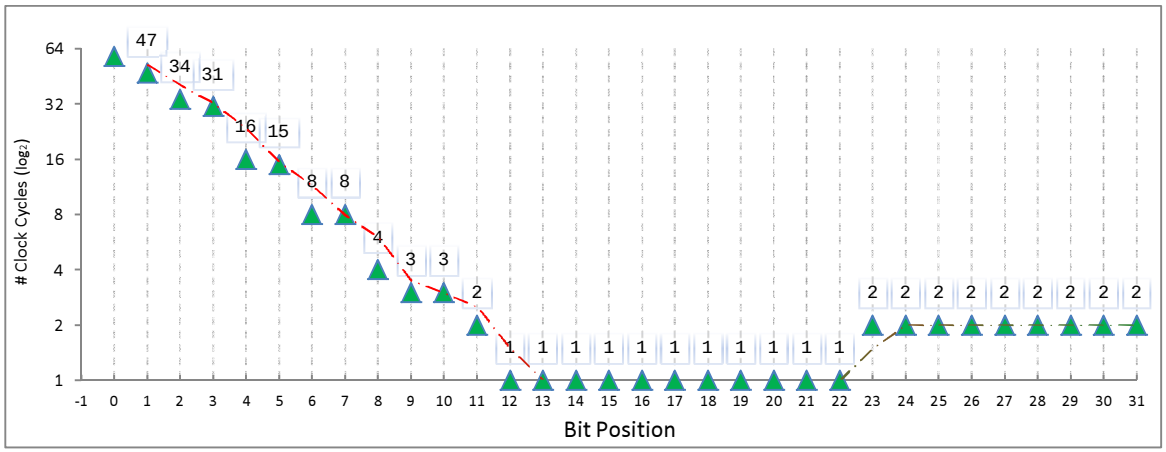
In this section, we have studied the sensitivity of each system to the initial conditions. This is an important assessment for CB-PRNG especially for the application of cryptography as in our case. In order to do this, we have evaluated the Avalanche criteria of the three CB-PRNG's. In other words, we have measured the number of clock cycle each CB-PRNG takes until it generates an output that differs in at least half the bits from an output that is generated by only flipping a certain bit in the input. We call this test the shift-and-flip test. We have performed this test by observing the output variable that has passed the NIST tests in the previous section. The test has been repeated for 64 randomly choose initial conditions and the average of clock cycles required was calculated. Figure 15 thorough Figure 17, show the obtained results on 16-bits bus width for this test.

While all the three systems are expected to be sensitive to the initial conditions, the shift-and-flip test shows that the Lorenz system is more sensitive to

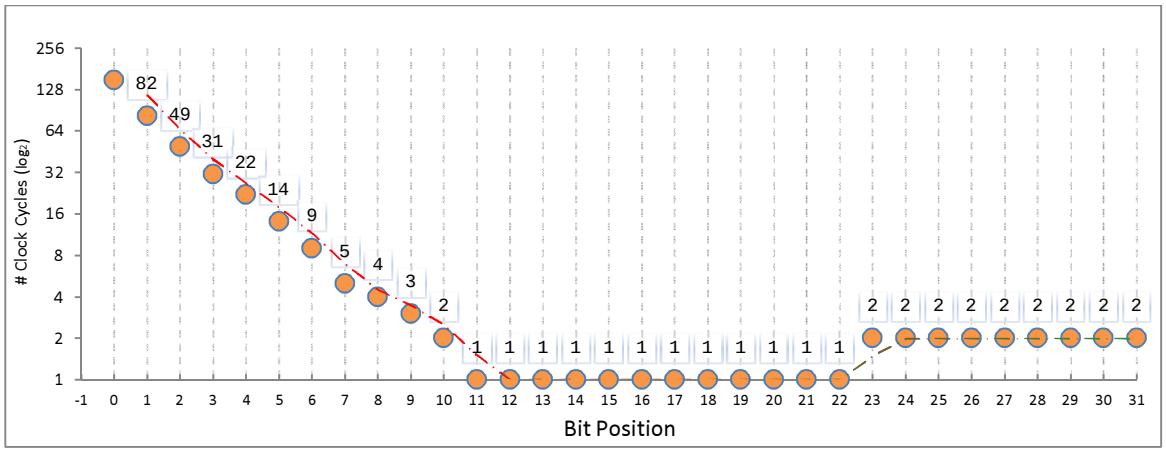
initial conditions than the other two systems, since it was requiring the an average of 1 clock cycle in order to change half its output bits. Such high sensitivity to initial conditions and high MLE score achieved by the Lorenz system is due to the multiplication operations in its equation system.



(a)

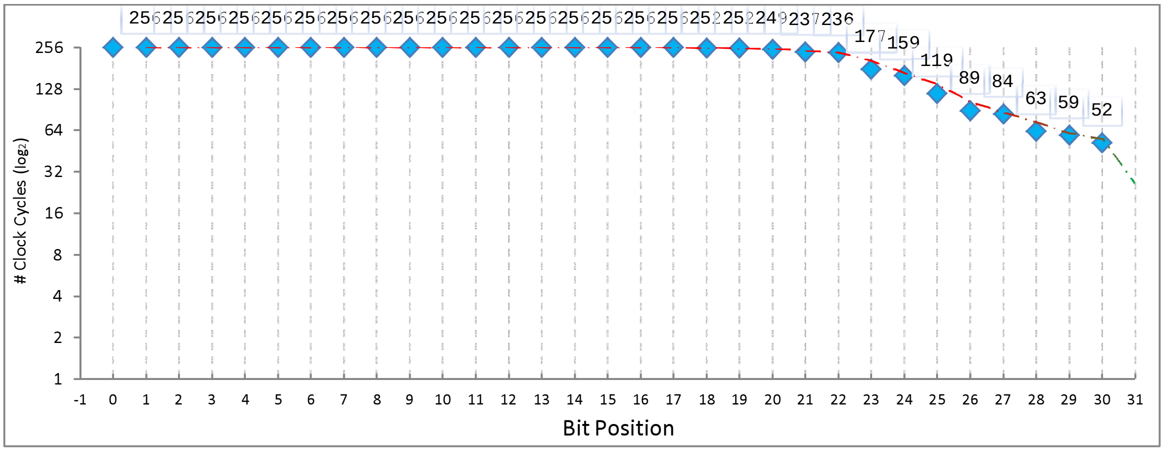


(b)

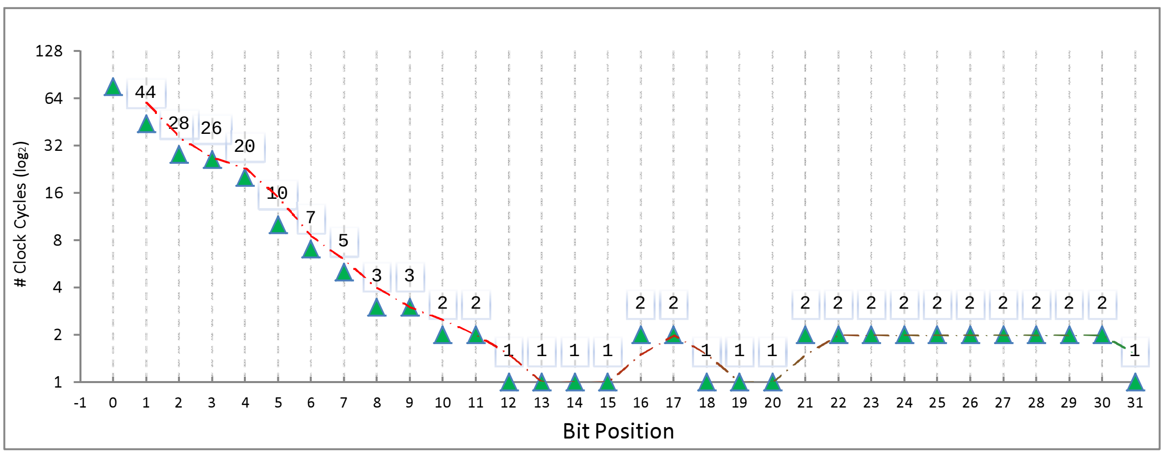


(c)

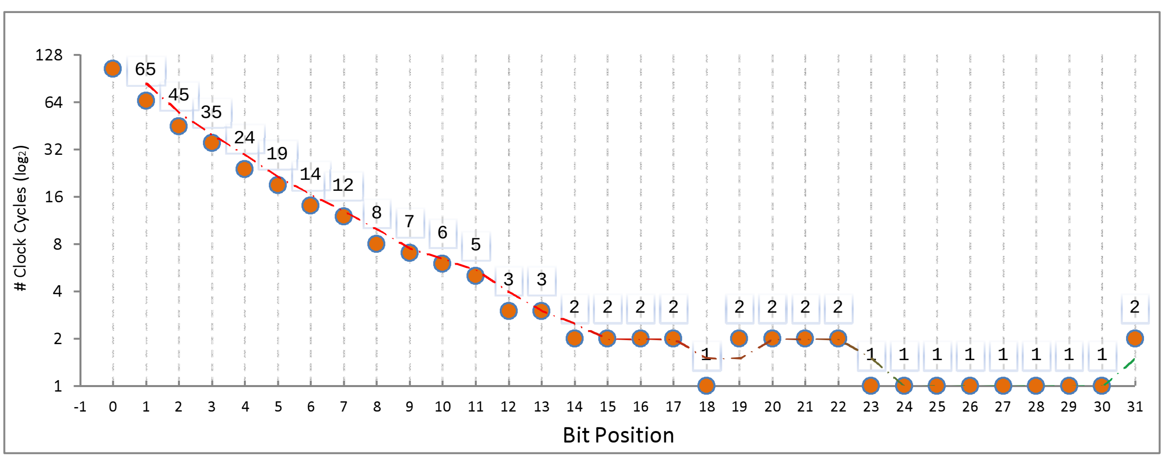
Figure 15: Number of clock cycles to satisfy Avalanche for 1D system. (a) input x, (b) input y, (c) input z



(a)

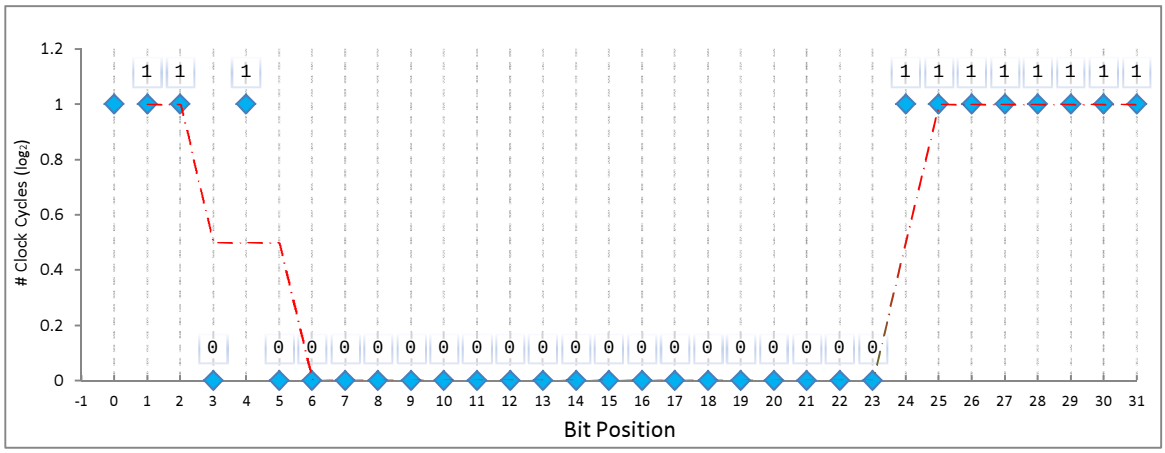


(b)

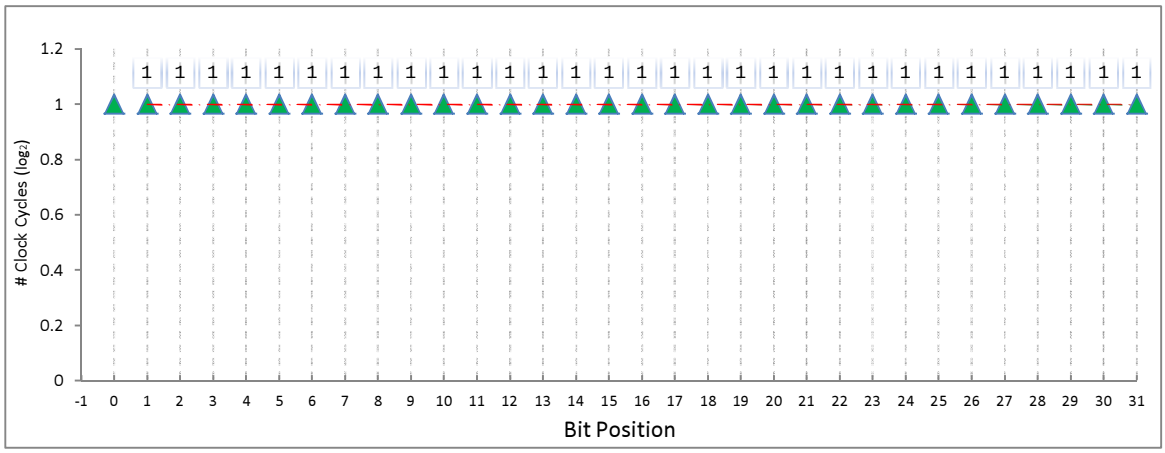


(c)

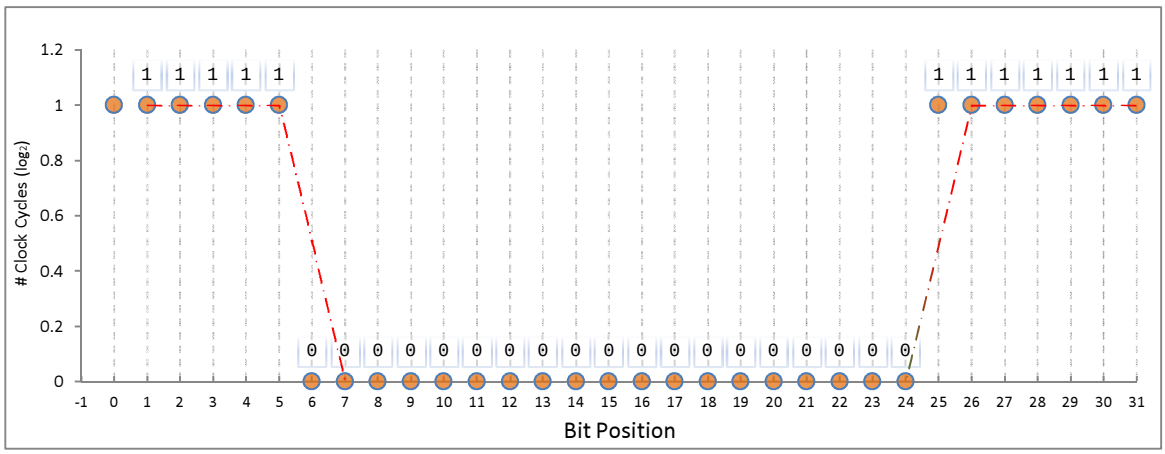
Figure 16: Number of clock cycles to satisfy Avalanche for Chen system. (a) input x, (b) input y, (c) input z



(a)



(b)



(c)

Figure 17: Number of clock cycles to satisfy Avalanche for Lorenz system. (a) input x, (b) input y, (c) input z

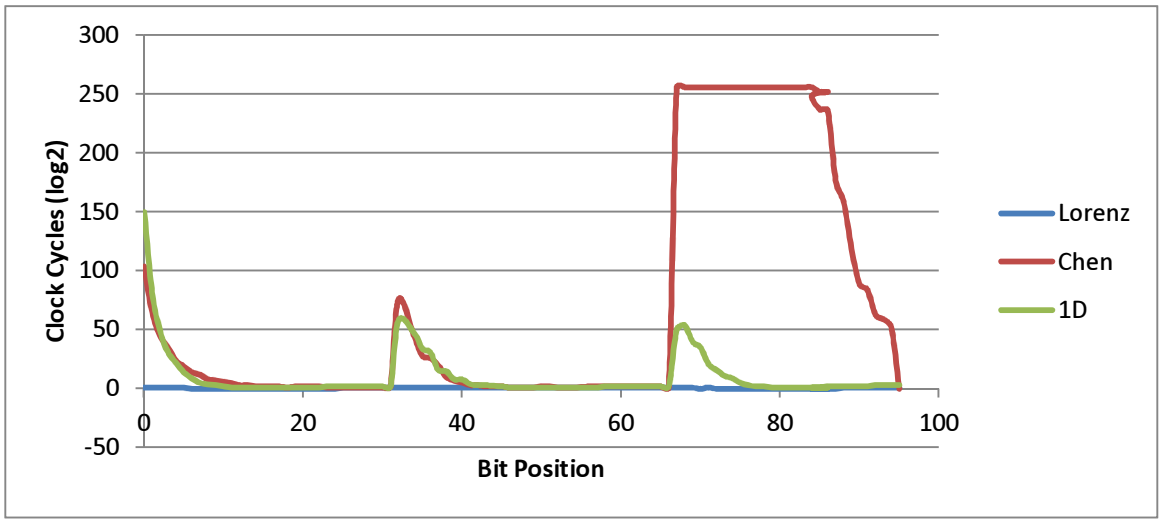


Figure 18: Summary of the key sensitivity Avalanche test. Bit position 0-31 corresponds with z, 32-63 to y and 64-95 to x.

III.2.4 Area and Speed

All the digital implementations of the CB-PRNG's using Euler approximation and the parameters defined in previous section have been synthesized on Xilinx Virtex4 XC4VSX35- 10FF668 FPGA. This FPGA has a total of 30,720 flip-flops (FF's), 30,720 4-input look-up tables (LUT's) and a total of 192 Digital Signal Processing (DPS) units. Table III.2.4 show the hardware utilization summary including area and speed. Because Lorenz system uses multiplication and then apply addition or subtraction, Xilinx synthesize tools optimize these operations by utilizing the DPS's included on the Virtex4 FPGA. However, Chen and the 1D multi-scroll systems do not involve such multiplication and hence the use of DSP's are not required.

Table III.2.4: Experimental area and throughput results from the Xilinx Virtex 4 FPGA optimized for speed.

	System 1- Lorenz	System 2 - Chen	System 3 - 1D
Bus Width [bits]	32	32	32
LUTs	316	415	331
FF's	96	96	101
DSP's	8	-	-
Output bits/cycle	32[1]	16[2]	32[3]
Frequency [MHz]	60.370	118.585	128.324
Throughput [Mb/s]	1931.84	1897.36	4106.368

[1] Concatenating outputs X and Y.

[2] Output Y.

[3] Concatenating outputs Y and Z.

Chapter IV

New Chaos Based S-Boxes for AES

This Chapter introduces a new hardware-based methodology for designing good cryptographic S-boxes for the AES algorithm and for any block ciphers that uses an 8X8 S-box module in its operations. The chapter begins by a brief explanation of S-boxes and their role and importance in nowadays block ciphers.

IV.1 S-Boxes Design Philosophy

The Substitution Box, or S-Box, is a means to replace a certain set of values with other different values according to a pre-defined mapping rule. An NxM S-Box, is a function that maps a set of values $P = \{0,1,\dots,2^{N-1}\}$ to another set $R = \{0,1,\dots,2^{M-1}\}$. Therefore, an 8x8 S-Box maps a byte value to another byte value. In general, S-Boxes are used in block ciphers as the step that provides the confusion in the ciphertext. The pre-defined rule used in mapping the values in the S-Boxes is what characterizes an S-Box and defines its properties. The rest of this section briefly describes the properties of a good cryptographic S-Box.

IV.1.1 Properties of Good Cryptographic S-boxes

Since block ciphers use S-Boxes to introduce confusion in the encryption process, the properties of the used S-Box determines how good such confusion is achieved.

In general, there are five properties that are considered when building S-Boxes for block ciphers [103]. These properties are Bijection, Nonlinearity, Strict Avalanche Criteria (SAC), and Independence of Output bits.

IV.1.1.1 Bijection

This property simply ensures that each input value to the S-Box is mapped to a unique output value forcing the S-Box to be a one-to-one function. Such property is necessary to enable the correct recovery (back substitution) of substitute values via the Inverse S-Box.

IV.1.1.2 Nonlinearity

Nonlinearity is always a desirable property in block ciphers as it adds more complexity to ciphertext and makes the block cipher more immune to the linear cryptanalysis [79]. S-Boxes are means to provide such nonlinearity in block ciphers. The nonlinearity property means that the mapping function of the S-Box is non-linear.

IV.1.1.3 Strict Avalanche Criterion

This property started originally with the term Avalanche only that was first introduced in 1973 by Feistel [104]. The Avalanche effect concept means that for any given function if the input is changed by one bit, there should be an Avalanche change in the output bits, at least half of the output bits should change. Completeness is another cryptographic property mainly for SPN block ciphers that means each output bit should depend on all the input bits [105]. Combining the

two concepts of Avalanche and Completeness, Webster and Tavares [103] have proposed the Strict Avalanche Criterion in 1985. The SAC basically means that when the input is changed by only one bit, then all the output bits should change 50% of the time for all the input vectors. When the SAC property is satisfied for an arbitrary function f, then f is said to exhibit a good diffusion.

IV.1.1.4 Independence of Output bits

This property means that the output bits behave independently from each other, where no statistical analysis can exploit a relation between them. To indicate how good such property is satisfied, another two properties are measured in S-Boxes which are Autocorrelation and Differential Uniformity.

The Autocorrelation measures the correlation of the input-outputs across the elements of the S-Box, which should be as small as possible. On the other hand, differential uniformity ensures that input differences are mapped uniformly to output differences. It is measured by the maximum number a certain output difference appears as the output when all the input differences are fed to the S-Box as the inputs. The less the Differential Uniformity and Autocorrelation between the elements of the S-Box, the more that S-Box is immune to differential cryptanalysis.

IV.1.2 AES S-Box

The original AES S-Box was derived using some algebraic manipulation of Boolean functions to be resistant against linear and differential cryptanalysis [82]. It is constructed by using multiplicative inverse operations over the Galois-Field

$GF(2^8)$ followed by some affine transformations [86]. The AES S-Box is believed to be one the S-Boxes that exhibit good cryptographic properties. Indeed, as we will show later in this chapter in the security analysis of S-Boxes, the AES S-Box showed excellent results for the Nonlinearity, SAC, Autocorrelation, and Differential uniformity tests.

IV.2 New Design and Hardware Implementation of CBS-Boxes

This section describe the design of a new dynamic key-dependent S-Box using CB-PRNDG's in Verilog. As discussed earlier, the S-Box is used at the Substitution Byte operation of the AES where every byte from the input state is replaced by its corresponding byte in the S-Box. This requires a searching mechanism where each byte is used as the search key to find its substitute.

In general, designing a dynamic S-Box imposes two hardware optimization challenges. First, the decision on how or where the substitution values for each byte (or the actual substitution table) will be stored should be made. On a typical FPGA, and depending on the device used, either a dedicated RAM architecture which is called Block RAM on FPGA's or a group of Look-Up tables, LUT's to exploit the architecture of a Distributed RAM can be used. Roughly speaking, the latter is desired when speed is an important factor and the data to be stored is not very big which is the case for dynamic S-boxes and is what we have used in our design. Second challenge is how you can search and access such dynamic tables

and provide the required substitution in a reasonable amount of time without an expensive hardware penalty.

Furthermore, the fact that in the AES there are two different tables are needed which are S-box itself (for the encryption process) and its inverse (for decryption) adds more difficulty to the design.

IV.2.1 Dynamic, Key-Dependent CBS-Boxes

IV.2.1.1 CB-PRNG Bus Width and Initial Condition

For S-Boxes, each entry consists of 8-bits that will be mapped to another 8-bits input. Therefore, to populate such a table we need 8-bits token at time. This implies that we should choose the CB-PRNG's bus width to be 16-bits in order to apply the post processing technique discussed earlier and have good statistically random sequences of 8-bits to fill up the S-Box.

The initial condition for the CB-PRNG is the AES 128-bit master key. Since we have used a bus width of 16-bits, assigning the initial condition is just a matter of dividing the 128-bits key among the 3x16-bits initial values for each internal chaos register. Our method to such distribution is shown in Figure 19. The most significant two bits of each initial value are left set to zeros to ensure that Chaos Generator does not diverges and stay on the attractor path.

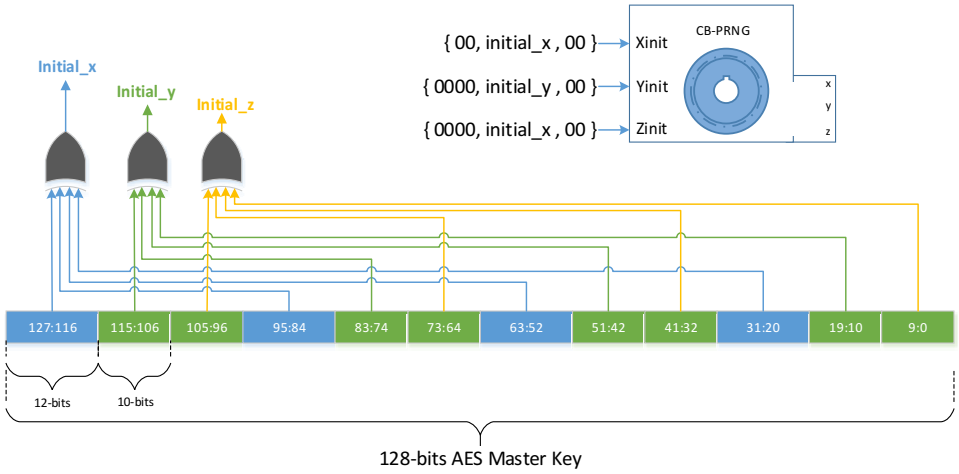


Figure 19: Distribution of the key bits on the initial conditions.

IV.2.1.2 Generating the S-Box

To dynamically store the 256 generated bytes from the CB-PRNG's, a group of 16 registers are used. Each register is 128-bits long where it can hold 16 bytes. Thus, we can visualize the storage of the 256 generated bytes as 16X16 2-dimentional array where a 4-bits index is needed to choose a row and another 4-bits index to choose a column. To do this, an 8-bit counter (*s_counter*) is used where the row index will be the most significant 4-bits and the column index is in the least significant 4-bits.

Using the clock signal, at each positive edge the output of the CB-PRNG is captured and assigned to the 8-bit entry addressed by the s_counter. Throughout this Chapter, we will use the word "token" to refer to output byte of the CB-PRNG and the word "index" to refer to the location in the S-Box where the token will be assigned to. Thus, index and s_counter can be used interchangeably.

The S-Box generation process is explained by the following example as in Figure 20. Assume the output of the CB-PRNG, a token, is the hexadecimal value "6C" and the value of s_counter, the index, at that cycle is "2E". This means that the row to be selected is the 2nd row, hence register number 1 is selected. Now, to assign the generated random value to column E (which the 14th position) of the 2nd register, we will have to keep shifting the value until it reaches the right position. However, this will cause the previously stored values in the second register to be misplaced or even lost. Instead, we create temporary 128-bit variable, initialize it to zeros, and then assign the random variable to it at the right position (14 in this case). The temporary variable is then XORed with the selected register which is the 2nd register in this case. Finally, the result of the XOR operation is stored back to the same register. Furthermore, at the end of such cycle and after the random value is stored successfully, the same *s_counter* is incremented. The generation finishes when the *s_counter* reaches value 256.

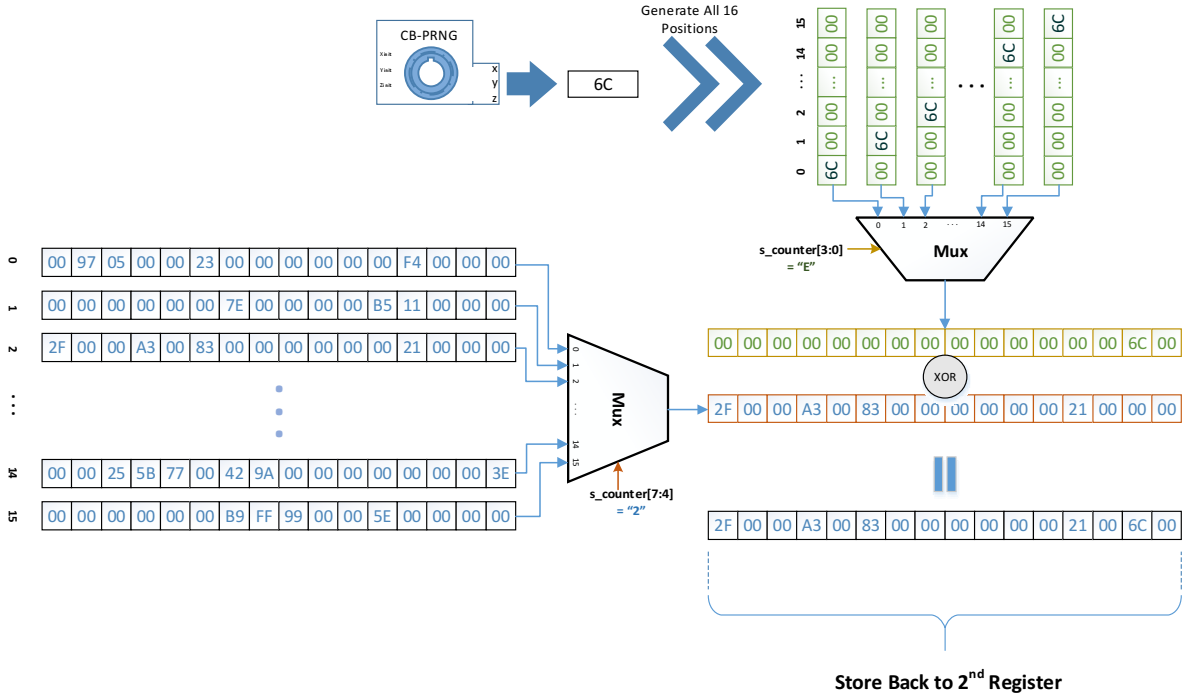


Figure 20: Submitting a new value to the S-Box.

IV.2.1.3 Avoiding Repeated Values

The previous design will work fine as all the 256 entries will be assigned a value from the CB-PRNG. However, the design will not prevent assigning the same value twice in case the CB-PRNG generated the token or the value more than once. Although, it's assumed that the CB-PRNG's have a big period and it is unlikely to produce the same token in the short time taken to create the S-Box, we should ensure that the design prevent such an event to ensure that the dynamically generated S-Boxes satisfy the Bijection property.

A first thought to solve this is to search the 16 register for each token to be assigned whether if it has been already stored or not. Of course, such operation is expensive and not practical especially when it comes to hardware implementations.

Instead, we perform the following trick. We create a record for each generated token as 256-bit register where each token corresponds to one bit in the record. All the bits in the record register are initialized to zeros at the start of the S-Box generation. After that, for each random token generated and assigned in the S-Box, its corresponding bit in the record is set to "1". This step is merely a one shifter where a value of one is assigned to the corresponding position and then XORed with the record register.

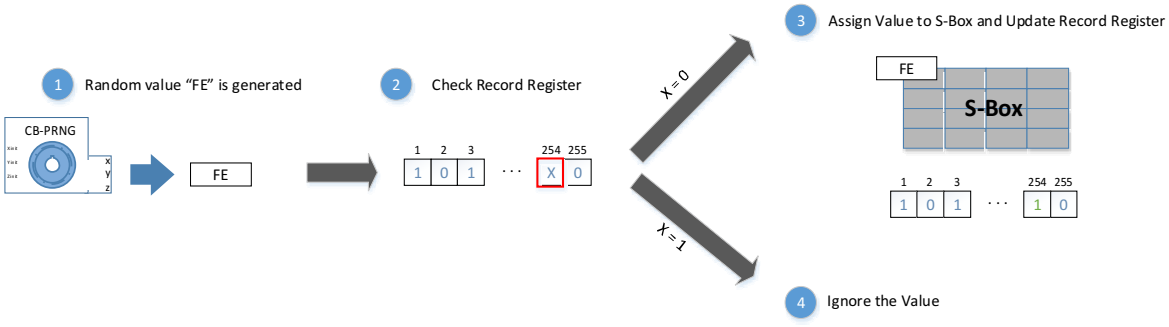


Figure 21: Redundancy check in the dynamic S-Box.

Now, the design is refined such that before assigning any randomly generated token to the S-Box, its corresponding bit in the record is checked first. If its corresponding bit is 0, then store the token in the S-Box, otherwise ignore it and get another token. This is illustrated in Figure 21. Equally important, the *s_counter* is only incremented when the token is accepted and assigned.

Furthermore, to ensure satisfying the SAC and nonlinearity to the optimal, we add one more checking condition for submitting the token to the S-Box. This condition is to prevent the event where the CB-PRNG generates a token equal to

the current value of the index which is the *s_counter*. Such event means that the S-Box will map a byte to itself leading to no confusion at all to the plaintext.

IV.2.1.4 Accessing the Dynamic S-Box

The second issue in dynamic S-Boxes after being created is the mechanism or the methodology of how they will be accessed. In fact, accessing S-Boxes both themselves and their inverse is what the algorithm will be using along the encryption and decryption processes. In encryption, a byte from the current state must be replaced from the S-Box. The easiest way to achieve this is to perform two multiplexing steps on the 16 S-Box registers as shown in Figure 22. The first step is to use the most significant 4-bit of the input byte to select a register among the 16 registers. The second step is to choose the right byte in the selected register using the least significant 4-bits of the same input byte.

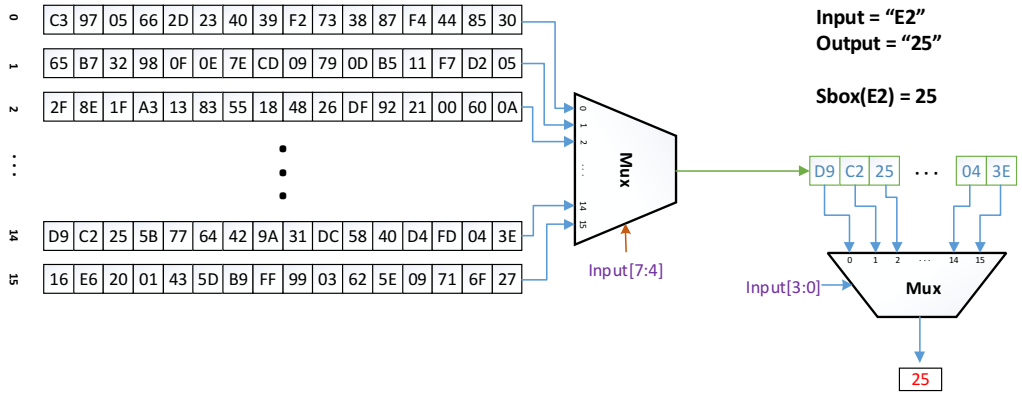


Figure 22: Direct access to the S-Box.

On the other hand, accessing the inverse S-Box is not as straight forward as the direct S-Box. In the case of the normal direct S-Box, the input will be the

index and the output is just the entry stored at that index. However, in the case of the inverse S-Box, the input is the entry and the output will be its inverse.

There are two approaches to provide the invers S-Box mapping. The first is to create another 16 by 128-bits registers that are populated at the same time the direct S-Box registers are populated and then use the same approach we mentioned for the direct S-Box to access the inverse S-Box. The second approach is to come up with a mechanism where you can provide inverse S-Box substitutions without creating another inverse registers and by only using the direct S-Box register. We've adopted the second approach in our design.

The access of the inverse S-Box using the direct S-Box in our design introduces two additional independent modules. The first one is the findByteBit module. This module takes the byte to be replaced by its inverse and use it as a key to a parallel hardware search in the direct S-Box registers. The search ends in one cycle only and outputs a 256-bits register *match* where all the bits are set to zero except the bit position where the key search byte and an entry in the S-Box registers match. Now, the job is done if we knew the position of that bit which was set to 1 in the *match* register. This position is the value of the inverse substitution of the input byte.

We have developed a fast and easy trick to get the position from the *match* register. The idea is similar to the Binary Search that uses a concept called Divide-and-Conquer. The *match* register is divided to halves till we find the bit that's set to one. This begins by dividing the register into two halves 128-bits each then check which register is not equal to zero. If the most significant half is not equal to

zero (which means that this half contains our bit), then add half the length of the divided register (which is 256 at the beginning, so add 128) to zero initialized sum variable. On the other hand, if the least significant half is the portion which is not equal to zero, then add zero to the sum variable. We then continue to divide the 128 half to 64, 32, 16, 8, 4, and 2 until we reach length of 1 bit and with each division we check and add to the cumulative sum variable. Finally, the sum variable will have the exact position of our bit. The full algorithm is described as follows:

```
Algorithm: Bit Position Finder
Input: keyByte b, 256-bits match register M
Output: invers byte b`
Begin
    Sum = 0;
    While(length(M)>1)
    Begin
        Lms = M[length(M)-1: length(M)/2];
        Lls = M [length(M)/2 -1: 0];
        If(Lms != 0)
            Sum = sum + length(M)/2;
            M = Lms;
        else
            M = Lls;
    End
    b` = sum;
End
```

As it can be shown from the algorithm, its hardware design will be very simple and only use few LUT's. The generation of the 256-bits match register involves parallel searching which is an expensive step in terms of area utilization. In fact, this step is what limits the efficiency of the dynamic inverse S-Box generation.

IV.2.2 Assignment Ordering Techniques

During the generation of the S-Box 16 registers, there are many methods of assigning the random tokens to the registers. In our design, we have used three techniques which are: assigning the random outputs to in sequence incrementing indices, assigning the random output to in-sequence incrementing Gray-encoded indices, and finally assigning the random output to randomly generated indices from another CB-PRNG.

IV.2.2.1 In-Sequence Ordering

Our original design without any changes implies an in-sequence assignment of the random tokens. This is because the *s_counter* is incremented by one in sequence. So, starting from zero, the first output token from the CB-PRNG is assigned to index 00, and then the second output is assigned to 01, and so on. The process continues till we reach FF which is the last position. In-Sequence Ordering is shown in Figure 23.

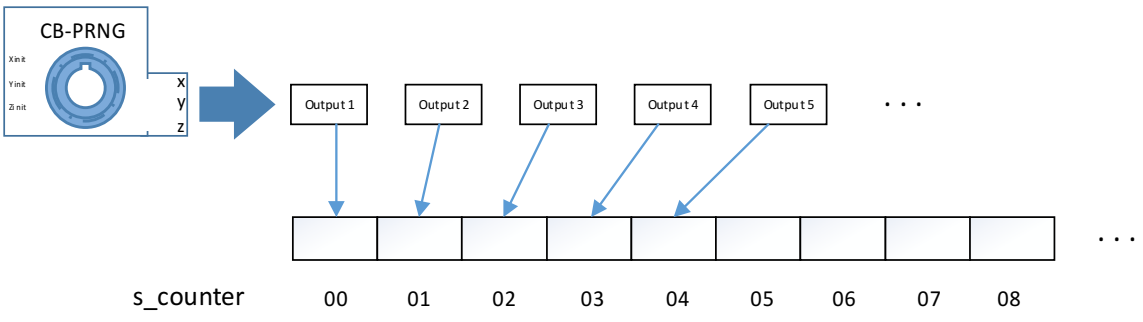


Figure 23: In-Sequence Ordering of outputs of the CB-PRNG.

IV.2.2.2 Gray-Encoding Ordering

In this ordering, the *s_counter* is incremented using Gray encoding. The Gray encoding is a binary encoding system where any two adjacent numbers in that system differs only in two bit positions [106]. This was believed to enhance SAC property of the S-Box. In any case, the only change in the design for this is making the *s_counter* increments in Gray codes rather than normal codes. This is illustrated in Figure 24. Hence we used a simple Binary to Gray converter that consists of simple XORing between the input bits as the following algorithm:

Algorithm: Binary to Gray Converter
Input: 8-bits binary-encoded b
Output: 8-bits Gray-encoded g
Begin
g[8] = b[8];
g[7] = b[8] XOR b[7];
g[6] = b[7] XOR b[6];
g[5] = b[6] XOR b[5];
g[4] = b[5] XOR b[4];
g[3] = b[2] XOR b[3];
g[2] = b[3] XOR b[2];
g[1] = b[2] XOR b[1];
End

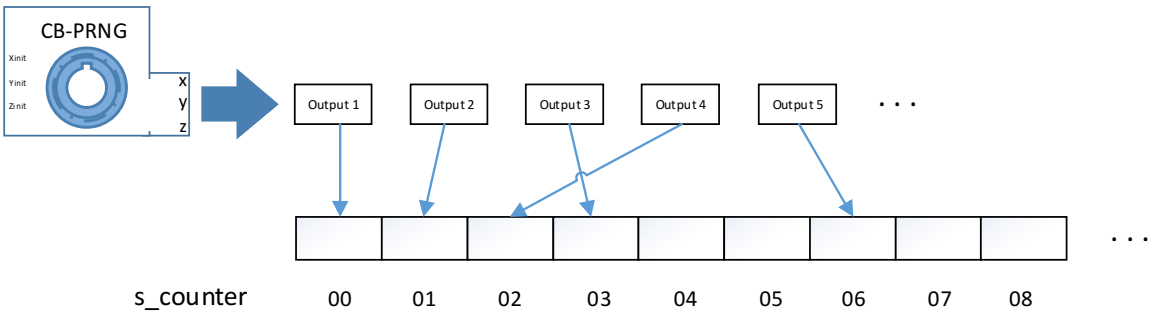


Figure 24: Gray-Encoding Ordering of the outputs of CB-PRNG.

IV.2.2.3 Random Ordering

In this technique, the random output from the CB-PRNG is assigned to random locations. These random locations are generated from another CB-PRNG where the later can be based on the same chaos system or based on a different system. In the case where the same system is used, then the initial conditions need not be the same otherwise will lead to a trivial S-Box where each byte maps to itself.

The modification to our design to incorporate the Random Ordering is as follows. Instead of assigning in-sequence incrementing values each time to *s_counter*, we assign random values generated by another CB-PRNG. However, before assigning this value to *s_counter*, we need to make sure that this value has not been assigned previously to avoid overwriting an entry in the S-Box registers. Hence, we create another bit record register and use the same repetition detection mechanism similar to the one already used before in Figure 21. Therefore, in Random Ordering we will have two bit record registers; one for the original values generated by the first CB-PRNG, and the second is for the values that will be assigned to the *s_counter* to index the S-Box registers.

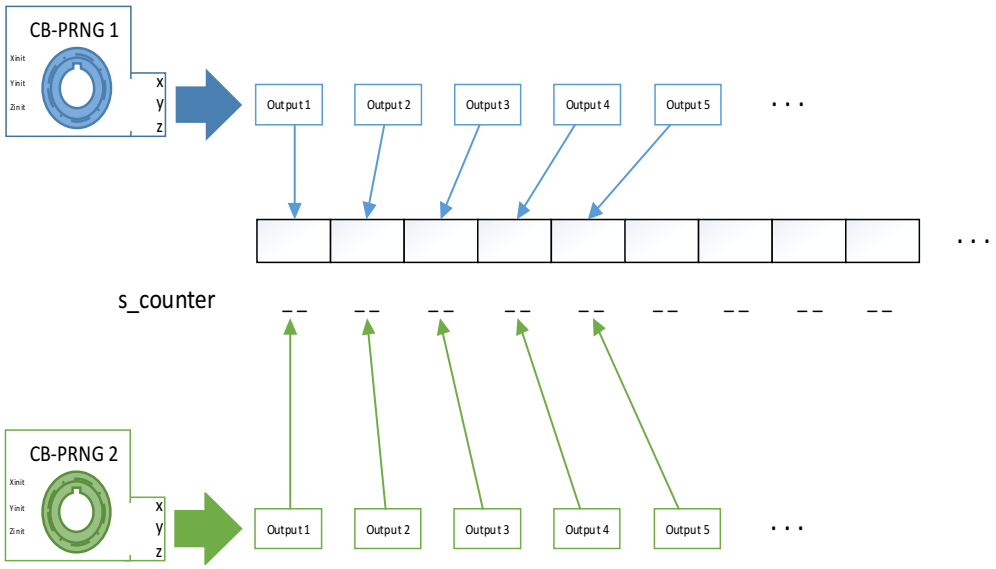


Figure 25: Random-to-Random Ordering of the outputs of CB-PRNG.

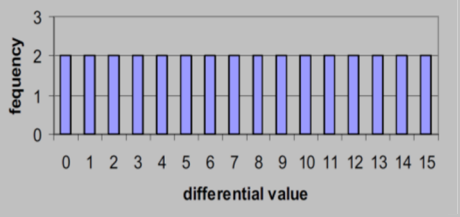
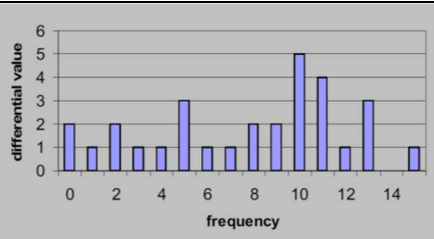
Generally speaking, Random Ordering technique is not efficient in terms of speed. This is because a lot of clock cycles will be wasted till all the 256 possible values of *s_counter* are covered. In fact, in some cases, the generation of the S-Box using this ordering diverges due to phenomena we call Unmatched-Random. This happens when there is only one unassigned entry in the S-Box and the CB-PRNG does not give that value or that index until it wastes a huge number of clock cycles. What makes this even worse is when the Unmatched-Random event happens to both CB-PRNG giving the values for *s_counter* or the token values.

IV.3 Analysis of Generated CBS-Boxes

In this section, we analyze the S-Boxes generated using the design approach described in earlier in this chapter. We use three analysis platforms to assess the quality of the dynamically generated S-Boxes. The three analyses are based on a graphical analysis, Walsh-Hadamard Spectrum, and image encryption analysis.

IV.3.1 Graphical Analysis

In this analysis, we adopt the methods proposed and described in [107]. The idea behind this analysis is to provide an initial judgment on a newly created S-Box whether it has good SAC properties or not, based on graphical statistical analysis. The end result of this analysis depends on the shape of the graph produced from processing some test cases from the S-Box. The analysis includes three tests where in order for the S-Box to pass a certain test, the produced graph needs to follow a certain shape. The three tests are the Completeness test, the Avalanche test, and the Strict Avalanche test. Figure 26 shows how to determine the results for each test.

Test	Good Results	Poor Results
Completeness		

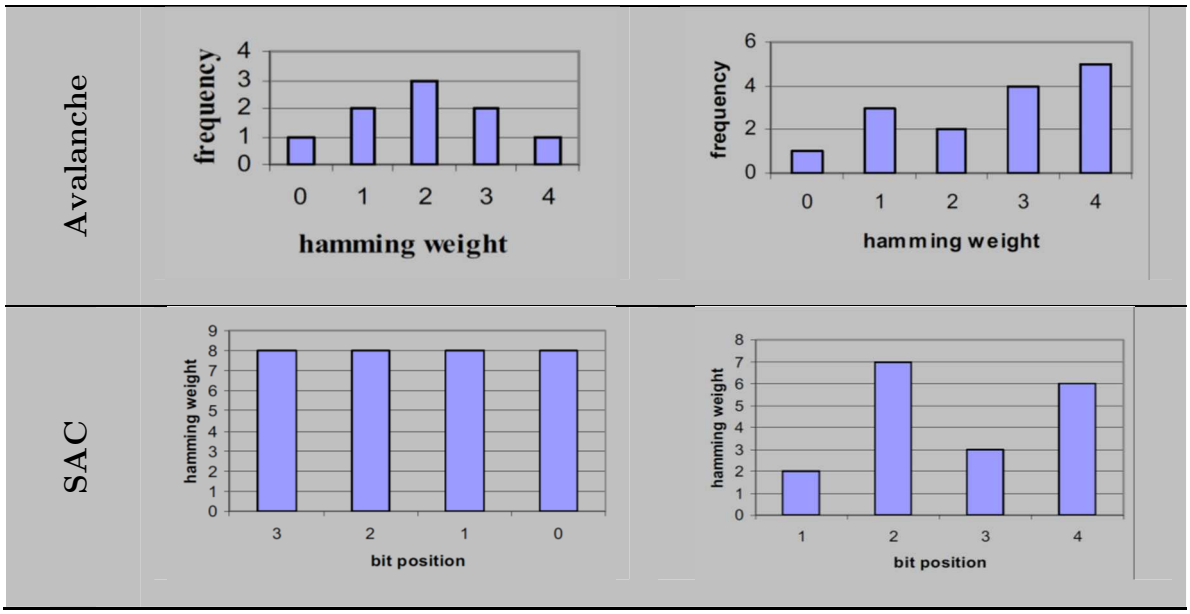
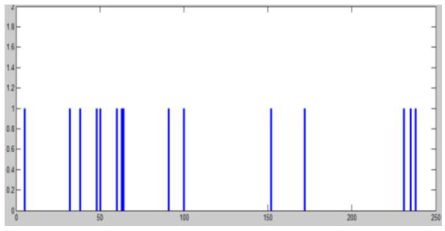
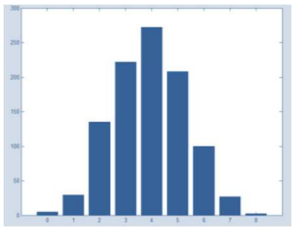
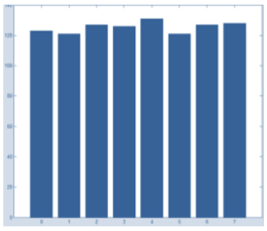
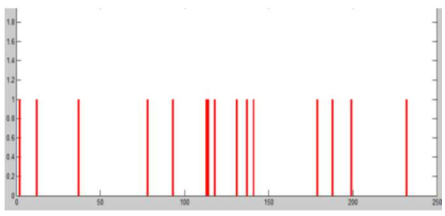
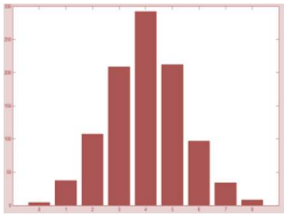
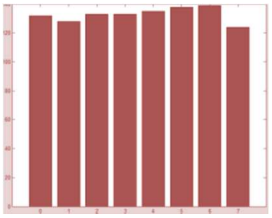


Figure 26: Decision rules based on graphs shape.

To apply these tests on the dynamically generated S-Boxes using the three systems, we randomly choose an S-Box from each category and from each Ordering Techniques. The results of the CBS-Boxes along and the original AES S-Box are shown in the next Figure.

System	Completeness	Avalanche	SAC
Original AES			
Lorenz			

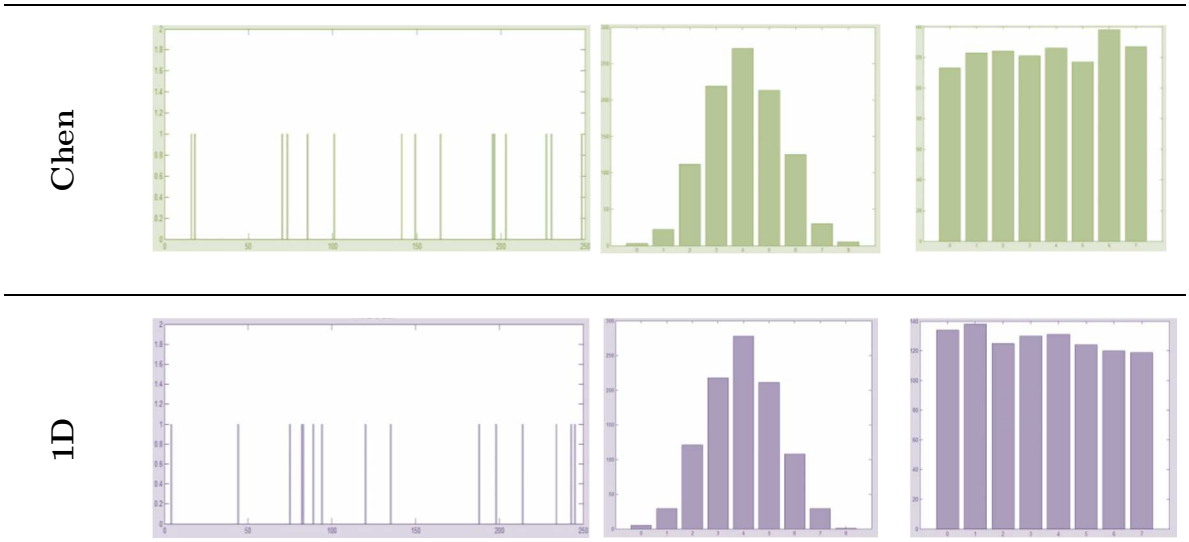


Figure 27: Graphical analysis test results, where S-Boxes chosen from Lorenz (random order), Chen (In-seq. Order), 1D (Gray-enc. Order).

IV.3.2 Walsh Spectrum Analysis

The analysis done using graphical analysis in previous section does not really specify which S-Box is better. Rather, it just gives an indication that these S-Boxes are adequate. Therefore, in order to investigate how much a certain S-Box is better than the other, we use another analysis tool. One of the important tools to perform such analysis on S-Boxes is the Walsh-Hadamard transformation [108].

Mainly, Walsh-Hadamard transformation works on Boolean functions [109] and is also used in encryption applications [110]. The idea to apply such tools in the analysis of S-Boxes is to treat the 8x8 S-Box as an 8 Boolean functions where each function takes 8-bits input. The cryptographic properties of these Boolean functions reflect the overall security of the S-Box based on them. In fact, the cryptographic properties extracted from the Walsh Spectrum analysis include SAC, Nonlinearity, Auto-correlation, and other features. Most of these properties are

what we are looking to investigate in any newly proposed S-Boxes and hence Walsh Spectrum analysis seems to be the right tool for this.

In this thesis, we have developed a full MATLAB package that compute the Walsh-Hadamard transformation of S-Boxes and to calculate their corresponding SAC, Nonlinearity, and Auto-correlation. For more details on the implementation of the Walsh-Hadamard transformation see Appendices Chapter VII on page 113.

Table IV.3.1: SAC and Nonlinearity results of dynamic CBS-Boxes.

System		SAC				Nonlinearity		
		Min	Max	Avg.	R. Error	Min	Max	Avg.
AES S-Box		**0.4531**	**0.5625**	**0.5048**	**0.1250**	**112**	**112**	**112**
Lorenz	In Seq.	0.3593	0.6406	0.5035	0.2312	96	110	103
	Gray	0.375	0.6406	0.5038	0.2223	96	110	103.5
	Random	0.3594	0.656	0.5030	0.2166	90	108	103
Chen	In Seq.	0.375	0.6250	0.5012	0.2166	92	110	103.9
	Gray	0.375	0.6406	0.5033	0.2223	96	110	103.5
	Random	0.375	0.6562	0.4999	0.2239	92	108	103.5
1D	In Seq.	0.375	0.6250	0.5012	0.2166	92	110	103.9
	Gray	0.343	0.625	0.5004	0.2333	92	108	103.3
	Random	0.359	0.640	0.5010	0.2187	94	108	103.4
Lorenz-1D		0.343	0.625	0.5014	0.2208	96	108	103.5
1D-Chen		0.359	0.640	0.5027	0.2375	90	108	103.43

Table IV.3.2: Autocorrelation and Differential Uniformity of the dynamic CBS-Boxes.

System		Auto-Correlation			Differential Uniformity		
		Min	Max	Avg.	Min	Max	Avg.
AES S-Box		**32**	**32**	**32**	**4**	**4**	**4**
Lorenz	In Seq.	48	88	68.4	10	14	11.5
	Gray	56	96	68.2	10	14	11.5
	Random	48	88	67.2	10	14	11.3
Chen	In Seq.	48	96	67.63	10	14	11.4
	Gray	56	96	68.18	10	14	11.5
	Random	56	96	68.3	10	14	11.6
1D	In Seq.	48	96	67.6	10	14	11.4
	Gray	56	96	69.4	10	14	11.6
	Random	56	88	68.2	10	14	11.0
Lorenz-1D		48	88	66.5	10	14	11.3
1D-Chen		48	96	67.3	10	14	11.6

IV.3.3 Image Encryption Analysis

Following the results obtained from the Walsh-Spectrum analysis, we can now decide how much a certain S-Box is better than the other. We next use an analysis based on image encryption statistics. In other words, we want to see how the CBS-Boxes perform in image encryption applications. To do this, a MATLAB software package has been developed to encrypt some grey-scale images using the best S-Boxes obtained from each chaos system and the original AES S-Box. Then, the encrypted images are fed to another MATLAB code that we also developed to implement the algorithms mentioned in [111]. The image analysis measures the quality of encryption, and hence the diffusion, when these S-Boxes are used to encrypt the images by only byte substitution. Table IV.3.3 list these tests and how

to judge the outcomes. These image analysis tests were performed on selected S-Boxes from each system and within each system one S-Box with good property is chosen from each ordering technique. The selection was based on the results obtained from the Walsh Spectrum analysis. The image analysis results performed on the S-Boxes using the cameraman.bmp file are shown in Table IV.3.4.

Table IV.3.3: Suggested image analysis test and the corresponding decision rule.

Encrypted Image Property/Test	**Outcome**
Pixels Correlation (Horizontal, Vertical, Diagonal)	the smaller the better
Entropy	the greater the better
Contrast	the greater the better
Homogeneity	the smaller the better
Energy	the smaller the better
Mean of Absolute Deviation, MAD	the greater the better

Table IV.3.4: Image analysis test results when applied to "cameraman.bmp" file.

System		Pixels Correlation			Entropy	Contr.	Homg.	Energy	MAD
		H	V	D					
Plain Image		0.9098	0.9443	0.9292	7.1028	-	-	-	-
AES S-Box		0.3071	0.2562	0.1044	7.1025	6125	**65078**	4.24518	**87.1**
Lorenz	In Seq.	0.3070	0.2407	0.2186	7.1025	1568	65252	4.25717	70.0
	Gray	0.3224	0.2330	0.1110	7.1025	8232	65133	4.23946	69.9
	Random	0.2638	0.1708	0.1447	7.1025	882	65264	4.25913	86.2
Chen	In Seq.	0.3029	0.2430	0.0978	7.1025	6223	65168	4.24492	79.2
	Gray	0.2455	0.1482	0.0735	7.1025	6517	65163	4.24375	77.3
	Random	0.2708	0.1820	0.1596	7.1025	9114	65117	4.23673	80.7
1D	In Seq.	0.2539	0.2020	0.1947	7.1025	8183	65133	4.23959	85.9
	Gray	0.3096	0.2289	0.1324	7.1025	1372	65255	4.25782	85.7
	Random	**0.2312**	0.1518	**0.0679**	7.1025	3038	65225	4.25339	82.8
Lorenz-1D		0.2481	**0.1373**	0.1848	7.1025	**11270**	**65078**	**4.23075**	79.6

From Table IV.3.4 we see that the original AES S-Box is not the dominant S-Box with the good results, unlike the analysis in the previous section. The original S-Box has scored better results only on Homogeneity and the MAD test, where the other remaining 6 tests showed CBS-Boxes perform better. All the S-Boxes scored almost the same in the Entropy test. Notice also that most of these image analysis tests by intuition will show better results if image was more randomly the encrypted. Therefore, the CBS-Boxes constructed using the Random Ordering technique, 1D-Random or Lorenz-to-1D, had most of the good results. Following are image encryption results using two 256x256-pixels grey-scale images; camaeraman.bmp and f14.

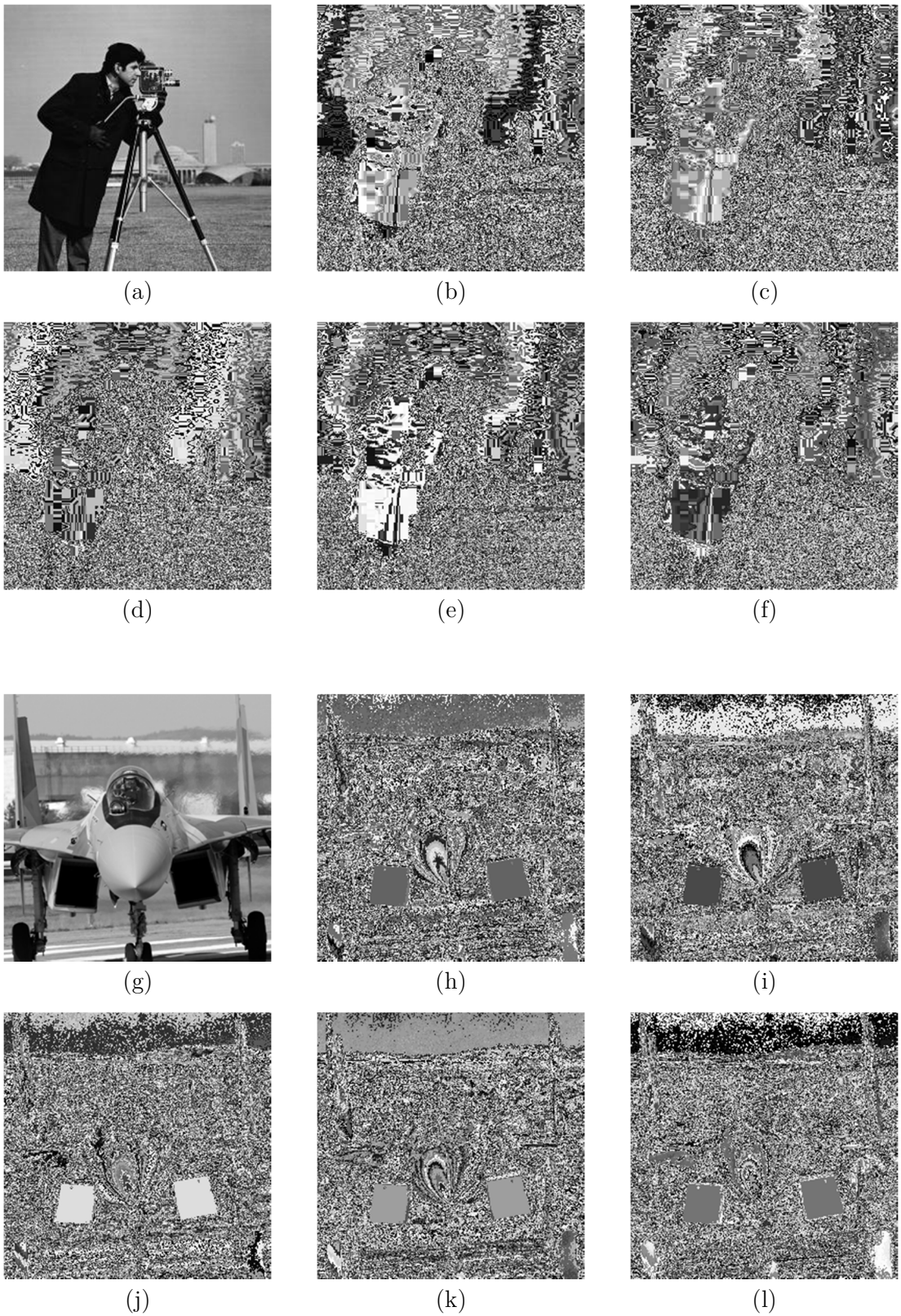


Figure 28: Image encryption results using S-Boxes. (a-f) results for cameraman, (g-l) results for f14, (b&h) original S-Box, (c&i) Lorenz, (d&j) Chen, (e&k) 1D, (f&l) Lorenz-to-1D.

IV.4 Comparison to Existing Literature

The comparison of dynamic S-Boxes is not straightforward. Most proposed S-Boxes, especially those based on Chaos, are static. In other words, the designers of such S-Boxes use the trial and error approach until they reach an S-Box with good properties or at least better than S-Boxes proposed by others. In such designs, the system initial conditions are closely investigated and observed and the ones that always produce good properties are used in generating the static S-Box. However, in dynamic S-Boxes such advantage is limited since the initial conditions are assigned from the encryption key. Table IV.4.1 compares the obtained nonlinearity and SAC results of the three proposed systems with existing dynamic and static S-Boxes.

Table IV.4.1: SAC and nonlinearity compariosn to existing static and dynamic S-Boxes.

System		SAC			Nonlinearity		
		Min	Max	Avg.	Min	Max	Avg.
		Static S-Boxes					
AES S-Box		0.453	0.562	0.505	112	112	112.0
Asim [58]		0.391	0.586	0.494	96	108	103.5
Jakimoski [55]		0.375	0.593	0506	100	108	103.4
		Dynamic S-Boxes					
Peng[4] [66]		0.485	0.519	0.490	95	110	103.6
This Work	Lorenz[5]	0.359	0.640	0.503	96	110	103.0
	Chen[6]	0.375	0.656	0.4999	92	108	103.5
	1D[7]	0.375	0.625	0.501	92	110	103.9
	Lorenz-1D	0.343	0.625	0.501	96	108	103.5
	1D-Chen	0.359	0.640	0.503	90	108	103.4

[4] Dynamic S-Boxes generated for 200 random keys.

[5] In-Sequence Ordering

[6] Random Ordering

[7] In-Sequence Ordering

Chapter V

New Improved Chaos-Based Key Expansion for AES

This chapter implements and introduces a new Key Expansion module for AES for the three 128-bits, 192-bits and 256-bits key length versions. The motivation behind our design is to overcome some of the defects the original AES Key Expansion suffers from. Therefore, this chapter starts with briefly describing the design philosophy of key schedules in general, and then the defects that are present in the AES Key Schedule are described. The chapter then introduces a novel chaos-based design along with the analysis and comparison to other works.

V.1 Key Schedulers Design Philosophy

Key Schedulers are important part of any encryption algorithm. In any block cipher, the Key Schedule is the unit that works on the secret key extracting a group of sub-keys to be used in each round or iteration in encryption/decryption operations. Thus, the Key Schedulers must not be weak in such way they should not help improving the chances of key related attacks to be successful.

V.1.1 Properties of Strong Key Schedulers

In general, there are no well-defined standard or agreement on the criteria that any good schedule should have. However, block cipher designers have proposed some properties that should exists in Key Schedules to be prone to the key-related attacks [112] [86] [73]. These properties can be summarized as follows:

- The Key Schedule should create mutually independent sub-keys that exhibit random features.
- The Key Schedule should be a One-Way function.
- No correlations should exist in the sub-keys such that the knowledge of a sub-key or (a part of a sub-key) does not give any information about the master key or any other sub-key.
- The Key Schedule should exhibit enough nonlinearity and high diffusion on the sub-keys.
- The Key Schedule should eliminate symmetry between the sub-keys.
- The Key Schedule should be simple to describe and efficient in implementation.

V.1.2 AES Key Schedules Defects

The AES algorithm can consists of two main parts: the cipher itself and its key schedule. If we return back to the description of the AES algorithm in Chapter 2, we can see that cipher algorithm uses and performs more robust round operations unlike the key schedule which only uses XOR operations in addition to the S-Box substitutions. In fact, the designers of Rijndael have claimed in their proposal for the AES that “Knowledge of a part of the Cipher Key or Round Key bits shall not

allow to calculate many other Round Key bits" [87]. However, according to the analysis preformed in [113], the AES key schedule does not seem to achieve this goal. In this analysis, the AES key expansion was shown to have two undesired properties which are partial key guessing and key splitting. These two properties cause the AES key expansion to exhibit low level of diffusion and nonlinearity which were also claimed by AES proposal. The analysis performed in [73] also emphasized the partial key guessing defect by demonstrating a similar defect called bit leakage. The bit leakage defect can be evident by observing the key schedule operation illustrated in Figure 10. The knowledge of words W_i and W_{i+1} causes immediate knowledge of word W_{i-3}.

Therefore, the AES original key schedule violates the first four properties mentioned in the previous section and that are suggested by cryptographers. In fact, the designers of Rijndael have stated that the main purpose of this key schedule design was to remove symmetry between sub-keys, be immune against related key attacks and have efficient implementations [86]. Their design indeed eliminated the symmetry between the sub-keys and can be implemented efficiently. However, the fact that it suffers from well-known defects have reduced the number of rounds were related key attacks are effective and also assisted the published attacks that exploit some weaknesses in the cipher system including attacks based on guessing round keys or exploiting some relations between round keys and the master key as in [73, 74, 113-116].

Our design proposal will aim to eliminate these defects and provide more secure key schedule module for AES for all the three key lengths. Furthermore, we

will elaborate more and measure how it performs in terms of diffusion and confusion compared to the original AES key expansion.

V.2 Design and Implementation of New Chaos-Based Key Expansion

In this section, we describe the implementation of our proposed chaos-based AES Key Expansion for the three key lengths 128-bit, 192-bit and 256-bit. Knowing the defects that the AES Key Schedule suffers from, we want our design to overcome these issues and strengthen the overall security of the AES algorithm. By looking back at the properties of good Key Schedules in the previous sections, we have anticipated that using CB-PRNG's in our approach will lead to a superior design. This is because CB-PRNG's incorporates these properties in their nature starting from randomness up to the efficient and simple implementations. As we have done in our design for the CBS-Box, we will only modify the module related to the Key Expansion function shown in Figure 11.

In designing our key expansion system, the main focus was to ensure that each bit in the master key should always affect all the bits in any given sub-key. Since Lorenz system showed the highest sensitivity to the initial conditions, we will focus on building our Key Schedule based on this generator from now on. In the case of the Lorenz system or any CB-PRNG, the master key will be the initial condition for that generator. This implies that we should use all the master key bits in the initial condition of the CB-PRNG. In fact, this was one of the challenges we have faced in designing our system.

In the first attempt to solve this issue, since the three CB-PRNG's are implemented as parameterized modules, we made the bus width of the output equal to 256-bits, where half of these bits will be neglected according to the post-processing technique we are using (see Post-Processing). This way the initial condition can cover up to 256-bit master key length and the output will result in a full 128-bit sub-key that depends on all the bits of the master key. However, this will be infeasible and impractical to implement as the hardware utilization will significantly increase and the clock cycle will be much longer resulting to inefficient design. To overcome this issue, we have used a trick that depends on the results obtained from Chapter 3.

Instead of using one CB-PRNG to hold all the master key bits as the initial condition, we use more than one generator of the same system and distribute the master key bits among these generators. We have fixed the bus width of the CB-PRNG's to be 32-bits wide, resulting in statistically acceptable sequences of 16-bits random outputs that can be used to construct a sub-key. This way, the CB-PRNG's will only have FPGA hardware utilization as shown in Table III.2.4. The number of generators used then depends on the length of the master key used in the encryption. This is explained in next section.

Now, each generator will produce a random output that is independent of the random outputs produced from the other generators and also each generator's output is affected only by the portion of the master key bits used as the initial condition for the corresponding generator. Therefore, we XOR the outputs of the generators creating what we call a token as shown in Figure 29. The XORing of the random outputs provides two advantages. First, it enables a token to

encapsulate any change in the master key bits. Second, XOR'ing two random outputs results a more statistically random outputs most of the time. Finally, the token is submitted to a big shift register that hold all the words of all the sub-keys in order.

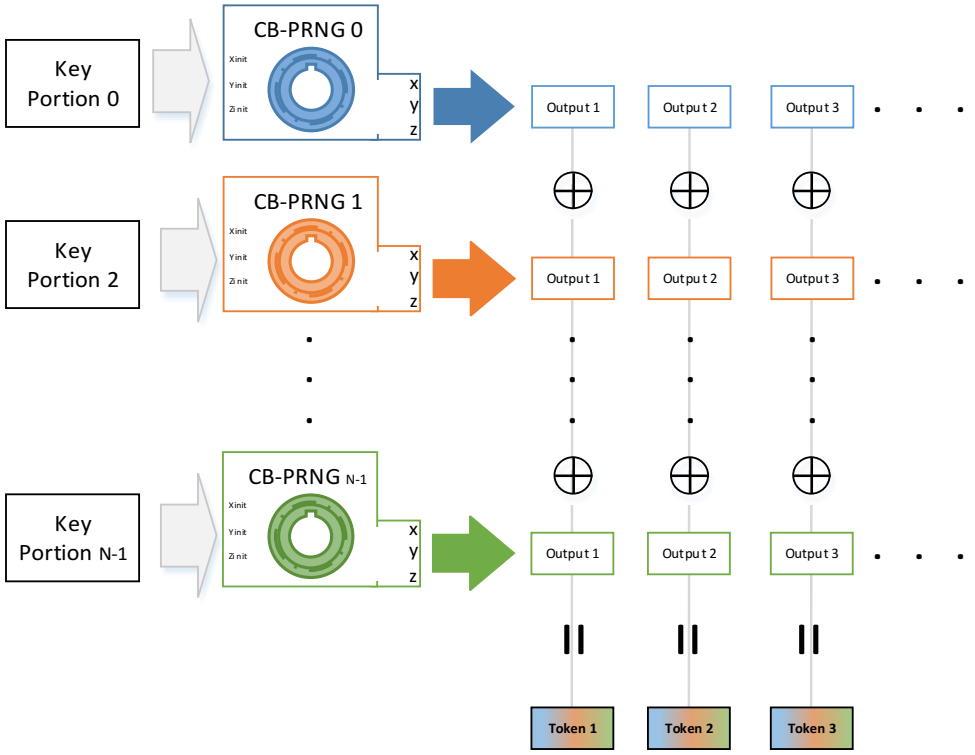


Figure 29: Creating tokens by XORing the outputs of the CB-PRNG's.

V.2.1 Master Key Distribution

The use of multi small-sized CB-PRNG's in creating the tokens that will construct the sub-keys along with the XOR operation has guaranteed that each token now changes when one bit of the master key changes. Now we turn our attention to the distribution of the master key bits among the initial condition of theses multi CB-

PRNG's. Following is the description of these distributions for 128-bits, 192-bits and 256-bits versions of Chaos-based AES Key Expansion.

For all the three key lengths, the key distribution among the three initial conditions was as follows: For initial condition of X, the most two significant and the six least significant bits were set to zeros and are left unchanged and the remaining 24-bits will be assigned a 24-bits portion of the master key. For the initial conditions of Z and Y, the most two significant and the least ten significant bits were left set to zeros and the remaining 20-bits will be assigned a 20-bits portions of the master key. The most significant bits of each initial condition were left unchanged to ensure that each CB-PRNG stays and operates in the expected attractor shape. Moreover, initial condition of X were given more bits of the master key than initial conditions of Y and Z because as Figure 17 shows initial conditions of X show more sensitivity to bits changes in the Lorenz based PRNG.

V.2.1.1 128-Bit Key Schedule

For the 128-bits Key Schedule, two CB-PRNG's have been used. Two 32-bits output are generated each clock cycle and XORed producing a 32-bits token that is submitted to the expanded key to form a word of a sub-key. Figure 30 illustrate this operation. The operation is repeated for (11*128)/32 = 44 clock cycles in order to create all the 11 sub-keys for AES-128 encryption mode.

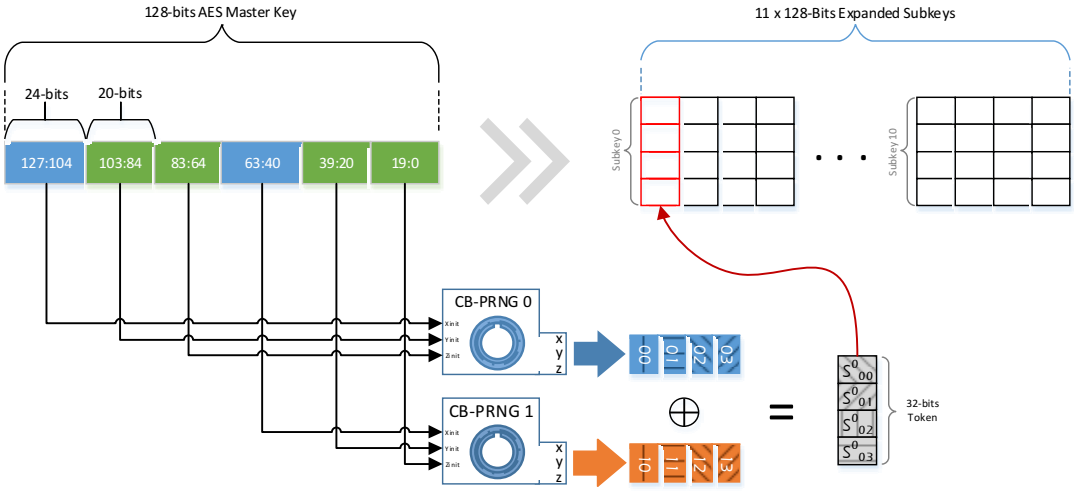


Figure 30: 128-bit Chaos Key Schedule.

V.2.1.2 192-Bit Key Schedule

In this version, the 192-bits of the master key are distributed among three CB-PRNG's. Accordingly, three random outputs will be XORed to produce a 32-bit token as shown in Figure 31. The 192-bit version of the chaos based key expansion is expected to generate 13 sub-keys and hence it requires a total of (13*128)/32=52 clock cycles to finish.

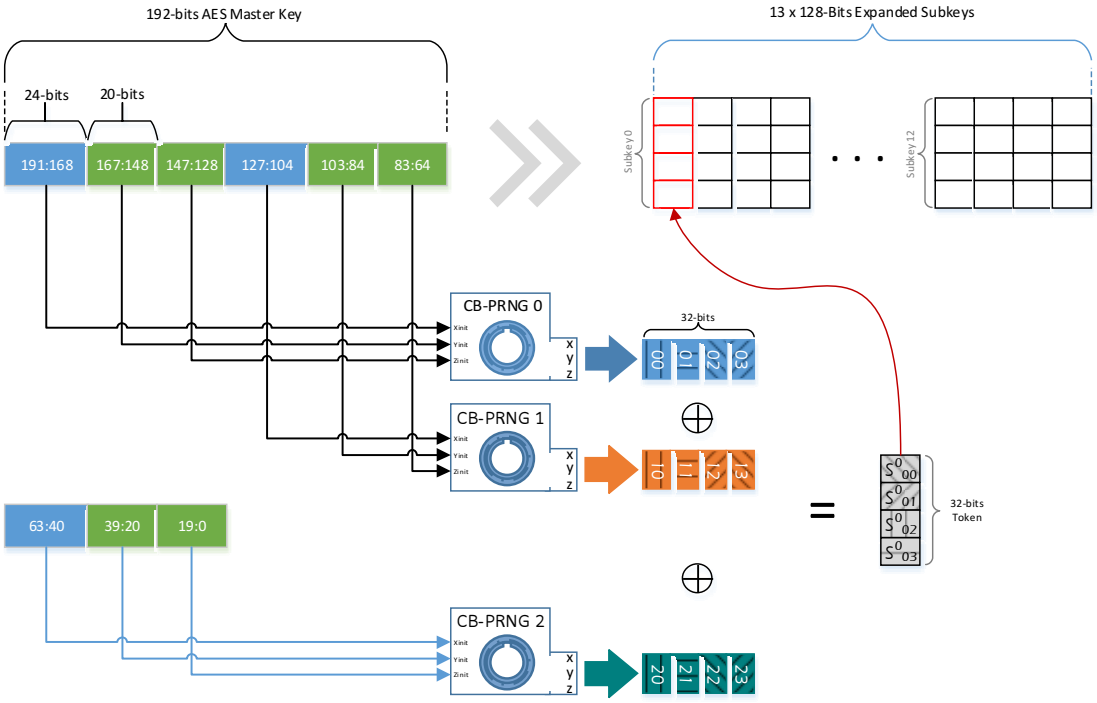


Figure 31: 192-bit Chaos Key Schedule.

V.2.1.3 256-Bit Key Schedule

For the 256-bit version of chaos based key expansion, four CB-PRNG's are used. This means that four outputs will be XORed at any given clock cycles to produce a 32-bit column for each one of the 15 sub-keys. The number of clock cycles required for this is equal to (15*128)/32=60. The operation of the 256-bit key expansion design is illustrated in Figure 32.

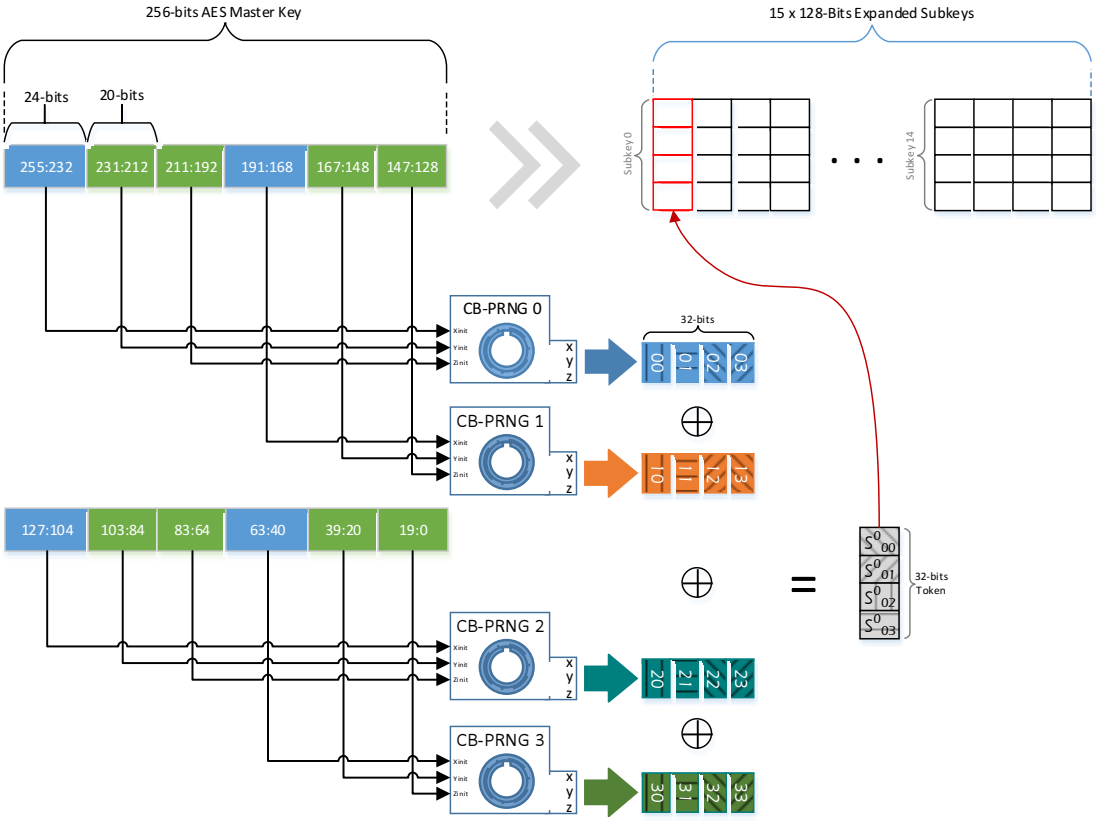


Figure 32: 256-bit Chaos Key Schedule.

V.2.2 Parallel Architecture for Speed

In the previous design, each one of the version of the Key Expansion takes a different number of clock cycles to provide the required number of sub-keys in each case. As we've shown that it takes exactly 44, 52, and 60 clock cycles to generate 11, 13, and 15 sub-keys respectively. However, the fact that the Lorenz CB-PRNG has area-efficient implementation encourages us to exploit parallelism in our design. In other words, we can double the speed for a price of small increase of area utilization.

Doubling the speed of the Chaos-based Key Expansion implies doubling the number of CB-PRNG's in each version. So, the number of CB-PRNG's will be doubled and will be distributed in two groups where each group will create half the number of required sub-keys. However, this will take us back to an issue we have encountered before which is the issue of distributing the initial conditions to doubled number of CB-PRNG's. Clearly, we cannot assign the same bits of the master key to the two groups as this will lead to creating two groups of the same sub-keys. The trick to solve this is to still assign the same bits of the master key to each group but change the order of which they are assigned. Again, this was inspired by the fact that the Lorenz-Based PRNG is very sensitive to initial conditions.

The idea of distributing the master key bits to the new redundant group of CB-PRNG's is as follows. Instead of assigning one chunk of 24-bits, and two chunk of 20-bits of the master key to the initial conditions of X, Y and Z starting from most significant bit of the master key as we did originally with first group, we assign these chunks for the redundant group starting from the least significant bit of the master key. This will results in a total different group of sub-keys. Figure 33 shows the parallel architecture for expanding the 128-bit master key. We can see from the figure that sub-key number 5, which is the middle key among the 11 sub-keys, is generated by the two groups where each group contribute by exactly 2 words.

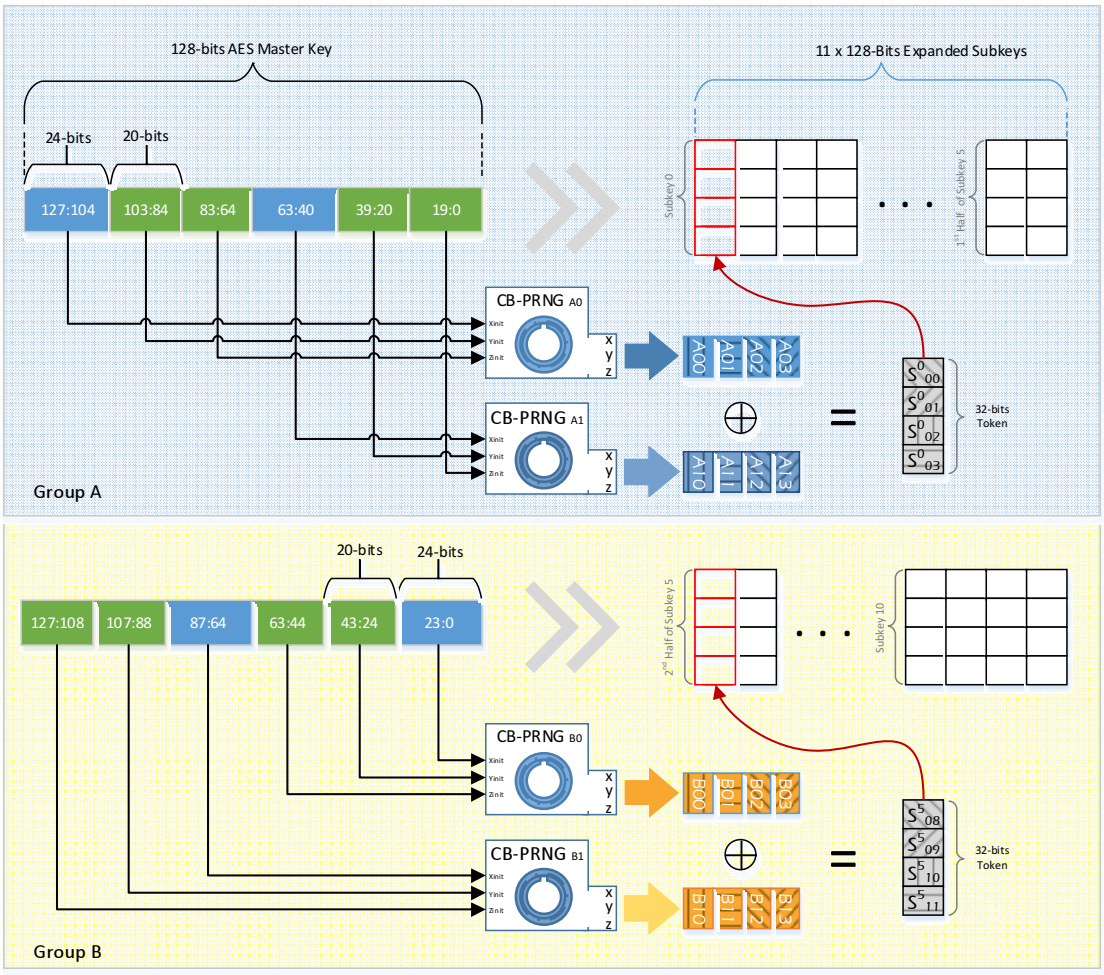


Figure 33: Parallel architecture of 128-bit Chaos Key Schedule.

Similar architecture is implemented for the 192-bit and 256-bit keys where the number of CB-PRNG's will double in each case from three to six in the 192-bit version and from four to eight in the 256-bit version. As a result of this, the number of clock cycles will reduce to 22, 26 and 30 for the 128-bit, 192-bit and 256-bit cases respectively.

V.3 Security Analysis of Proposed Chaos-based Key Schedule

In order to make a strong and fair security analysis of any proposed key schedule, we treat the key schedule algorithm as a ciphering algorithm by itself. We then evaluate the key schedules according to the security criteria that we always use to evaluate the ciphering algorithms. Thus, we can conclude robust judgment on the proposed key schedule and ensure it produces strong and good statistically random sub-keys. Two of the well-known security criteria that most ciphering algorithm are based up on are confusion and diffusion [28]. According to [73], the confusion and diffusion properties can be measure using the frequency test and the SAC test respectively.

V.3.1.1 Frequency Test

The purpose of the frequency test is to measure the proportion of the ones and zeros in each sub-key and hence it asses the bit mixing criteria as a measure for the confusion property. In our evaluation for the proposed chaos based key schedule, we have used the frequency test provided in the NIST SP. 800-22 statistical testing suite [101]. The output of this test is a p-value that conclude that a certain sequence is random or not for a given significance level α. If the resulted p-value $<$0.01 or 0.001, then conclude the sequence to be non-random. Otherwise, conclude the sequence to be random. In cryptography, the significance level is usually 1% or 0.1% corresponding to values of α of 0.01 and 0.001, respectively.

V.3.1.2 SAC Test

The Strict Avalanche Criterion is used here to evaluate the diffusion property of the generated sub-keys from the chaos based key schedule. We have developed a full MATLAB program to evaluate SAC property of every generated sub-key from the 128-bit, 192-bit and 256-bit chaos based key expansion. The program implements the algorithm described by [117]. Before the MATLAB program is used, a simulation using Xilinx ISE 11.1 ISim is used to produce all the avalanche vectors for all sub-keys. These avalanche vectors have information about how each sub-key changes when each bit of the master key is flipped. The avalanche vectors are them submitted to MATLAB to perform the Kolmogorov-Smirnov test statistics as described in [118]. The output of the Kolmogorov-Smirnov test is D* statistics value for each sub-key, where a value of D* less than 1.628 and 1.949 implies that the bit diffusion for that sub-key is satisfied at the 1% and 0.1% critical level respectively [73, 117].

V.3.1.3 Obtained Results

To support the findings that were concluded by [73] and [113] regarding the slow diffusion of the original AES key schedule, we have performed the frequency test and the SAC test on the sub-keys generated by the original key schedule. In performing these tests, 4K of random keys generated by MATLAB were used as the input to create the sub-keys. The results of the 128-bit original key expansion are shown in Table V.3.1.

Table V.3.1: SAC and Frequency Test of the original AES Key Schedule

Round	Frequency		SAC	
	P/F	p	P/F	D^*
1	✗	0.0001	✗	119.241
2	✓	0.5038	✗	93.758
3	✓	0.9493	✗	65.82
4	✓	0.5901	✗	40.766
5	✓	0.4026	✗	29.654
6	✓	0.5361	✗	28.025
7	✓	0.3591	✗	28.025
8	✓	0.7319	✗	28.078
9	✓	0.3254	✗	28.078
10	✓	0.5920	✗	28.001

We can see from Table V.3.1 that, even though the SAC is not satisfied at any critical level, the bit diffusion in the sub-keys is very slow. Furthermore, not all the sub-keys pass the frequency test as sub-key 1 which is the most related key to the master key fails to score p-value greater than 0.001. Therefore, these results are consistent with the claim that AES original key schedule does not provide sufficient diffusion in in the generated sub-keys.

Unlike the original AES key expansion, our proposed chaos based key expansion have satisfied the Frequency and SAC tests for all round keys and for the 128-bit, 192-bit and 256-bit master keys. Table V.3.2 through Table V.3.4 show the test results for our proposed key schedule for the three master key lengths.

Table V.3.2: SAC and Frequency Test of the proposed chaos-bsed 128-bit AES Key Schedule

Round	Frequency		SAC	
	P/F	p	P/F	D^*
0	✓	0.1689	✓	1.445
1	✓	0.9427	✓	1.318
2	✓	0.0263	✓	1.221
3	✓	0.2789	✓	1.494
4	✓	0.7908	✓	1.296
5	✓	0.1394	✓	1.448
6	✓	0.7633	✓	1.442
7	✓	0.1750	✓	1.899
8	✓	0.4279	✓	1.675
9	✓	0.2098	✓	1.805
10	✓	0.1732	✓	1.501

Table V.3.3: SAC and frequency test of the proposed chaos-based 192-bit AES Key Schedule.

Round	Frequency		SAC	
	P/F	p	P/F	D^*
0	✓	0.7372	✓	1.178
1	✓	0.8074	✓	1.230
2	✓	0.7899	✓	1.018
3	✓	0.7788	✓	1.434
4	✓	0.8614	✓	1.237
5	✓	0.0396	✓	1.669
6	✓	0.1921	✓	1.164
7	✓	0.6330	✓	1.398
8	✓	0.8713	✓	1.402
9	✓	0.5274	✓	1.243
10	✓	0.9856	✓	1.529
11	✓	0.5765	✓	1.672
12	✓	0.4373	✓	1.194

Table V.3.4: SAC and frequency test of the proposed chaos-based 256-bit AES Key Schedule.

Round	Frequency		SAC	
	P/F	p	P/F	D^*
0	✓	0.9195	✓	1.242
1	✓	0.5346	✓	1.028
2	✓	0.9701	✓	1.005
3	✓	0.4662	✓	1.231
4	✓	0.4032	✓	1.346
5	✓	0.3085	✓	1.109
6	✓	0.6510	✓	1.363
7	✓	0.9067	✓	1.393
8	✓	0.0213	✓	1.128
9	✓	0.0101	✓	1.094
10	✓	0.7399	✓	1.274
11	✓	0.9515	✓	1.203
12	✓	0.5899	✓	1.273
13	✓	0.1566	✓	1.365
14	✓	0.5995	✓	1.824

V.3.1.4 Results Discussion

We can see that our proposed key schedule has better confusion and diffusion than the original AES key schedule. In fact, our proposal adds more security advantages to the AES algorithms beyond better diffusion and confusion. The proposed key schedule satisfies all the properties we mentioned in the previous sections that a strong key schedule should satisfy. Equally important, another security advantage is that the modified chaos based AES that uses this proposed key schedule does not use the master key in first Add Round operation unlike the original algorithm. Finally, the entropy of the sub-keys in the chaos based AES is maximized now since all the master key bits are all now used in generating them. According to [119], the later promotes the chaos based AES now to rank first in the key schedules classifications as most secure where the original key schedule was causing AES to rank fourth in the same classification.

V.3.1.5 Comparison to Existing Key Schedules

In this section we compare the chaos based key schedule to other proposed key schedule that was originally proposed to solve the bit leakage defect we have discussed before [73]. Table V.3.5 show the comparison between three sub-keys for 128-bit, 192-bit and 256-bit master key lengths.

Table V.3.5: SAC and Frequency Test of the proposed design and existing work.

Round	Chaos-Based Key Schedule				May et al. [73]			
	Frequency		**SAC**		**Frequency**		**SAC**	
	P/F	p	P/F	D^*	P/F	p	P/F	D^*
128-bit master key								
1	✓	0.9427	✓	1.318	✓	0.1557	X	15.775
2	✓	0.0263	✓	1.221	✓	0.8757	✓	1.212
3	✓	0.2789	✓	1.494	✓	0.3498	✓	1.689
192-bit master key								
1	✓	0.8074	✓	1.230	✓	0.5593	X	11.891
2	✓	0.7899	✓	1.018	✓	0.2002	✓	1.268
3	✓	0.7788	✓	1.434	✓	0.2041	✓	1.155
256-bit master key								
1	✓	0.5346	✓	1.028	✓	0.8900	X	21.158
2	✓	0.9701	✓	1.005	✓	0.6766	✓	1.196
3	✓	0.4662	✓	1.231	✓	0.9029	✓	1.89

V.4 Area and Performance Results

Our proposed chaos-based Key Expansion for AES was synthesized and implemented on Xilinx Virtex4 XC4VSX35 FPGA. The design is fully parameterized, meaning all the three versions corresponding to different key lengths have been implemented in one Verilog file. An external include file has some defined constants that could be used to select the key lengths.

V.4.1 Key-to-Encryption Benchmark

A benchmark for measuring and comparing the efficiency of Key Schedules is the key-to-encryption ratio [73]. This ratio is calculated by dividing the number of cycles the key schedule takes to finish generating all sub-keys by the clock cycles

taken to encrypt one block only. One important note here about our proposed chaos based key expansion is that it only takes few clock cycles by Lorenz system to start generating highly diffused sub-keys. For the results obtained for the SAC and frequency tests, only four clock cycles were skipped before the start of the real chaos key expansion operation. Thus, the total number of clock cycles required by the design is 22+4=26, 26+4=30 and 30+4=34 clock cycles for the 128-bit, 192-bit and 256-bit versions, respectively. Other than the key schedule, the new chaos based AES requires the same number of clock cycles as required by the original AES. Table V.4.1 shows and compares the key-to-encryption ratio of the different key schedules.

Table V.4.1: K:E ratio comparison of proposed design and existing work.

	Key Exp. Chaos	Key Exp. Original [95]	Key Exp. May et al. [73]
128-bit master key			
Key Setup	26	42	-
Encrypt Block	126	142	-
K:E	**0.2**	**0.295**	**3.30**
192-bit master key			
Key Setup	30	52	-
Encrypt Block	150	172	-
K:E	**0.2**	**0.3**	**3.25**
256-bit master key			
Key Setup	34	58	-
Encrypt Block	173	198	-
K:E	**0.19**	**0.29**	**3.21**

V.4.2 Area and Speed Estimation

Table V.4.2 shows the hardware utilization when the design is synthesized using Xilinx Virtex4 XC4VSX35 FPGA for the smallest key length 128-bits and the longest key length 256-bits. The frequency of the chaos based key expansion and accordingly the chaos based AES is limited frequency of the Lorenz chaos generator. So, we would expect the frequency of the design to be 60.73 MHz as computed by Xilinx synthesize tool in Table III.2.4,which is the case here indeed.

Table V.4.2: Hardware utilization of the proposed design and the original key schedule.

	Chaos		Original	
	K. Exp. Only	Full AES	K. Exp. Only	Full AES
	128-bit master key			
LUTs	2793	5725	4969	7811
FF's	1824	4750	2832	5764
DSP48s	32	32	-	-
FIFO16/RAMB16s	-	16	-	16
Frequency [MHz]	60.370	60.370	85.854	93.936
	256-bit master key			
LUTs	4571	8402	6816	10397
FF's	2720	6416	3856	7566
DSP48s	64	64	-	-
FIFO16/RAMB16s	-	16	-	16
Frequency [MHz]	60.370	60.370	82.414	80.046

Optimized for speed

V.5 Speed Enhancement for Proposed Chaos Key Expansion

In this section, we propose a new approach to enhance the frequency speed of the 128-bits chaos-based key schedule implemented in the previous section. The previous design was based on the Lorenz chaos generator that in turn uses 32-bit multipliers. Such 32-bit multiplier is the unit that introduces the longest delay in the system.

Therefore, and as an attempt to eliminate such latency, we again adopt the concept of divide-and-conquer in the new approach. The basic idea is as follows: instead of using four 32-bit wide chaos generators in each group creating one half of the sub-keys as in Figure 33, we use eight 16-bit wide chaos generators. Hence, we replaced each wide multiplier by two narrower multipliers working in parallel and generating the same number of bits per cycle but, more importantly, in much less time.

However, doing this will take us back to the same issue we encountered on how to ensure that all the used eight 16-bit chaos generators now create outputs that depends on all the bits of the master key. Again, this can be resolved by coming up with a way to distribute the master key bits among all the chaos generators in such a way no two generators produce the same output. Then, the same trick of XORing all the outputs in one token so that a token carry all the changes the might happen in any bit of the master key bits. Such setup and distribution of the master key bits is also shown in Figure 34.

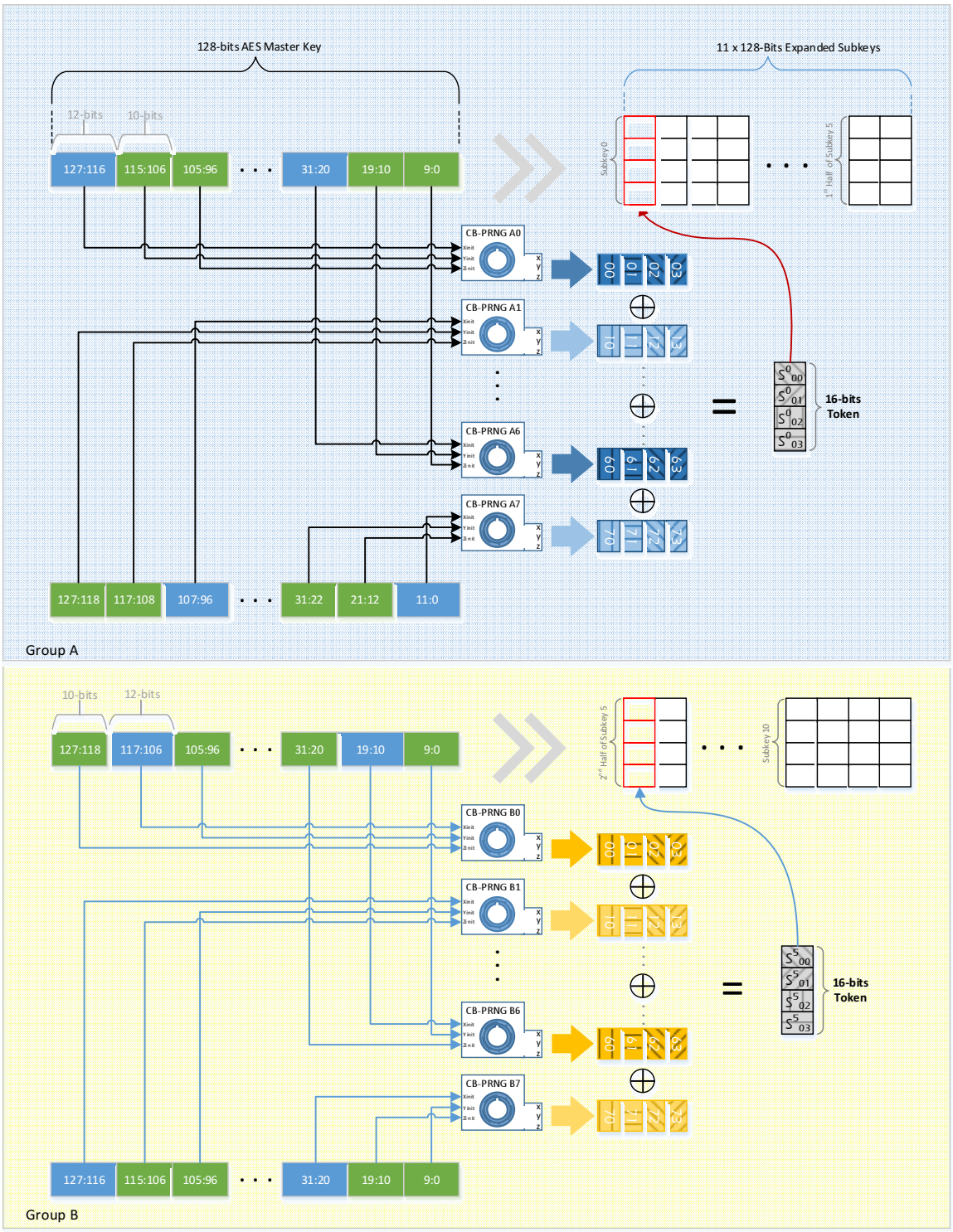


Figure 34: Parallel architecture of the enhanced 128-bit chaos-based key schedule.

V.5.1 Security Analysis and K:E Ratio

The new system have been tested for the quality of diffusion and confusion it imposes on the generated 11 sub-keys using the same Frequency test and SAC test mentioned in before. The new results of the system are shown in Table V.5.1.

Table V.5.1: SAC and Frequency test results for the enahnced 128-bit chaos based key schedule.

Round	Frequency		SAC	
	P/F	p	P/F	D^*
0	✓	0.8596	✓	1.254
1	✓	0.5806	✓	1.418
2	✓	0.1913	✓	1.418
3	✓	0.8727	✓	1.424
4	✓	0.5977	✓	1.201
5	✓	0.8770	✓	1.681
6	✓	0.2493	✓	1.437
7	✓	0.4743	✓	1.436
8	✓	0.3693	✓	1.342
9	✓	0.8294	✓	1.512
10	✓	0.4120	✓	1.663

In order to achieve such excellent results on both tests, two main modifications have performed on the original system besides making the output now 16-bits wide and doubling the number of generators. First, the parameters of the Lorenz generator is changed as follows: α is now changed from 8 to 32 which means now instead shifting by 3 bits we shift by 5 bits, and r is changed from 32 to 64 which indicates shifting by 6 bits instead of 5 and finally, β is left unchanged. Furthermore, the most significant bits of the initial conditions of x and y were changed from 00 for x and 00 for y to be now 10 and 01 respectively. Second

modification is that the number of clock cycles skipped by the Lorenz generator is now 11 clock cycles instead of 4 cycles as in the previous system.

Such modifications were necessary because reducing the multipliers width to 16-bits caused the diffusion rate to be slower now. The number of clock cycles now taken by the new 128-bit chaos based key expansion to generate all the 11 sub-keys is 22+11=32 clock cycles. Table V.5.2 shows the K:E ratio for the new enhanced design comparing with the previous and original designs.

Table V.5.2: K:E ratio of the enhanced design compared with the previous designs.

	Enhanced Key Exp. Chaos	Key Exp. Chaos	Key Exp. Original [95]	Key Exp. May et al. [73]
128-bit master key				
Key Setup	32	26	42	-
Encrypt Block	126	126	142	-
K:E	**0.254**	**0.2**	**0.295**	**3.30**

V.5.2 Area and performance Results

The main purpose of this design was to increase the operating frequency of the chaos based key expansion without compromising the good results of the diffusion and confusion tests. Indeed, the operating frequency of the new system increased from 60.37 MHz up to 103.697 MHz achieving a speed-up of 1.72. Hardware utilization of the proposed system after and before frequency enhancements along with the original AES results are show in Table V.5.3. The enhanced chaos-based key expansion now operates on a frequency faster than the original AES and

consumes less area as well. Thus the enhanced chaos system now is still better than the original AES but more area demanding than the previous design.

Table V.5.3: Hardware utilization of the proposed system for 128-bit after and before frequency enhancements.

	Enhanced Chaos		Chaos		Original	
	K. Exp. Only	Full AES	K. Exp. Only	Full AES	K. Exp. Only	Full AES
LUTs	3840	6769	2793	5725	4969	7811
FF's	2160	5087	1824	4750	2832	5764
DSP48s	30	30	32	32	-	-
FIFO16/RAMB16s	-	16	-	16	-	16
Frequency [MHz]	**103.697**	**103.697**	60.370	60.370	85.854	93.936

Optimized for speed.

However, if the area is such a concern in a certain application employing the AES on the FPGA chip, the parallel architecture of the new enhanced chaos-based key expansion can be left out. This will introduce more clock cycles in the new key expansion leading to less key-to-encryption ration, but nevertheless the whole new AES will operate on a higher frequency now. The new clock cycles taken by the chaos-based key expansion will now be 44+11=55 clock cycles, since now Group B, as was illustrated in Figure 34, will now be deactivated. This seems like an attractive solution because 36% of area has been saved and a speed up of 10% in frequency in Full AES algorithm is obtained only at a cost of increasing the number of clock cycles (41% more) taken by the key expansion alone. The new hardware estimation results with K:E ratio is shown in Table V.5.4.

For encrypting a big amount of data using the same key, the key expansion is used only once at the beginning, unlike the full cipher which will iterate for a given number of rounds. Thus, increasing the number of cycles in the key expansion only in returns for minimizing speed and area for the whole AES algorithm is a reasonable compromise.

Table V.5.4: New results of hardware utilization and frequency along with K:E ratio.

		Enhanced Chaos (Un-Parallel Arch.)		Enhanced Chaos (Parallel Arch.)		Original	
		K. Exp. Only	Full AES	K. Exp. Only	Full AES	K. Exp. Only	Full AES
K:E	Key Setup	55		32		42	
	Enc. Block	126		126		142	
	Ratio	0.437		**0.254**		**0.295**	
LUTs		2378	5035	3840	6769	4969	7811
FF's		1472	4046	2160	5087	2832	5764
DSP48s		16	16	30	30	-	-
FIFO16/RAMB16s		-	16	-	16	-	16
Frequency [MHz]		**103.697**	**103.697**	**103.697**	**103.697**	**85.854**	**93.936**

Chapter VI

Conclusion

VI.1 Summary

This thesis introduces a new chaos-based Advanced Encryption Standard with two of its original modules replaced by new chaos-based versions. These two modules are the S-Box and the Key Schedule. Three chaos generators are considered for this mission, which are the Lorenz system that is based on multiplication nonlinearity, the Chen system that is based on sign modules nonlinearity, and one-dimensional multiscroll system that is based on staircase nonlinearity. Before these systems are introduced in the new designs, they are assessed and evaluated as PRNG's after applying post-processing techniques, and are also tested for their sensitivity to the initial conditions. The three systems are then used to dynamically create new S-Boxes for the AES that change whenever the master key changes. Moreover, the detailed design and hardware implementation of the key-dependent CBS-Boxes using three ordering techniques were given. The generated S-Boxes were analyzed using well-known tools such as basic graphical analysis, Walsh-Hadamard Spectrum analysis, and image encryption analysis. The results obtained show that the CBS-Boxes exhibit good cryptographic properties that can be compared to other work. Furthermore, the Lorenz system is used to design a new chaos-based

key schedule for AES. The new proposed design has shown to be more secure, faster, and more area-efficient than the original AES key schedule.

VI.2 Future Research Work

One possible direction of future work is to optimize the design of the dynamic key-dependent chaos-based S-Boxes for AES. Extensive analysis performed in Chapter 4 suggests that such chaos-based S-Boxes exhibit good cryptographic properties in addition to the fact that they are created dynamically. This adds more security and complexity to the AES algorithm since new different S-Boxes are generated whenever the secret key changes. However, the hardware implementation of such dynamic look-up tables is expensive in terms of area utilization.

Other possible future work is to explore more options to introduce the chaos generators in the AES or in any encryption algorithm. Chapter 5 has shown an example of such successful integration between chaos and encryption. Thus, this can be such a motivation to further investigate more options to involve chaos in already existing encryption algorithms.

APPENDICES

Chapter VII

Walsh-Hadamard Transformation for Boolean Functions

Walsh-Hadamard Transform (WHT) is a generalized class of Fourier Transform that can provide another representation for Boolean functions. Similar to Fourier Transform that can express a periodic signal in terms of sine and cosine functions with different frequencies, WHT can also express a Boolean function in terms of its correlation with all linear functions of different Hamming weights.

For a given Boolean function $f(x): Z_2^n \rightarrow Z_2$, where $x = \{x_1, x_2, \dots, x_n\}$, then its WHT $S_f(w)$ is given by [108]:

$$S_f(w) = \sum_{x=0}^{2^n-1} (-1)^{w \bullet x} f(x)$$

where $w \in Z_2^n$, and $w \bullet x = w_0 x_0 + \cdots + w_{n-1} x_{n-1}$.

The following are the algorithms description along with the MATLAB implementation to extract the cryptographic properties of S-Boxes.

VII.1 WHT and SAC Computation

The computation of the WHT is combined with SAC as they require same amount of iteration. Thus, the SAC matrix of the eight Boolean functions of the S-Box is calculated in the same time the WHT is computed. To calculate the SAC, we need to compute the probability of change in each output bit when a certain bit in the input is flipped.

The S-Box can be seen to be composed of 8 Boolean functions $f_1, f_2, \ldots, f_8$. Let P_j^i denotes the probability of change of the output of f_j when only the ith bit $x \in Z_2^n$ is flipped. Then, P_j^i can be calculated using the following formula [120]:

$$P_j^i = \frac{1}{2} - 2^{-(2n-1)} \sum_{w \in Z_2^n} \hat{S}_{(j)}^2(w)(-1)^{w_i}$$

The following is the MATLAB code for WHT and SAT:

```
%  Calculating Walash-Hadamard transform and SAC Matrix
%-------------------------------------------------------
Sh = zeros(8, 256);
for i=1:1:8 %--> This loops all 8 bit positions
    for j=1:1:8 %--> This loops all 8 boolean functions
        SWtemp = 0;
        for w=0:1:255
            Sjh = 0;
            w_bin = dec2bin(w, 8);
            for x=0:1:255
                x_bin = dec2bin(x, 8);
                xw = x_bin(1)*w_bin(1);
                for u=2:1:8
                    xw = bitxor(xw, x_bin(u)*w_bin(u));
                end
                Sjh = Sjh + (sh(j,x+1) * (-1)^xw);
            end
            Sh(j, w+1) = Sjh;
            SWtemp = SWtemp + (Sjh*Sjh * (-1)^w_bin(9-i));
        end
        P(i,j) = 0.5 - ( 0.5^(2*8+1) * SWtemp);
        Paval(i) = Paval(i) + P(i,j)/8;
    end
end
```

VII.2 Nonlinearity

The nonlinearity of the S-Box is extracted from computing the nonlinearity of its corresponding eight Boolean functions. For a given Boolean function f, its nonlinearity N_f can be computed from its WHT by the following formula [108]:

$$N_f = 2^{n-1}(1 - \underset{w \in Z_2^n}{Max} |S_{(f)}(w)|)$$

The MATLAB code to calculate the nonlinearity of the S-Box is shown below:

```
%%
% Calculating Nonlinearity N
Nk = zeros(8, 1);
for k=1:1:8
    Sh_abs = abs(Sh(k,:));
    max_Sh_abs = max(Sh_abs);
    Nk(k) = (2^7)*(1-(2^-8)*max_Sh_abs);
end
N = max(Nk);
```

VII.3 Autocorrelation

The autocorrelation of Boolean functions can be computed from the WHT using The Wiener-Kintchine Theorem [121] that relates the autocorrelation of Boolean functions to their power spectrum as follows:

$$\sum_{x \in Z_2^n} ac(x)(-1)^{x.w} = \hat{S}(w)^2$$

The MATLAB Code to calculate the Autocorrelation is shown below:

```
%%
% Autocorrelation (AC)
% using Wiener-Kinthine theorem (from Walsh)
%-----------------------------------------
AC = zeros(8, 256);
for u=1:1:8
    AC(u,:) = Sh(u,:).^2;
    i = 1;
    while(i<256)
        j = 0;
        while(j<256)
           k = j;
            while(k<j+i)
               a = AC(u,k +1) - AC(u,k+1 +i);
               AC(u,k +1) = AC(u,k +1) + AC(u,k+1 +i);
               AC(u,k+1 +i) = a;
               k = k + 1;
            end
           j = j + bitshift(i,1);
        end
       i = bitshift(i,1);
    end
    AC(u,:) = bitshift(AC(u,:),-8);
end
% Calcualting MAX AC values matrix
AC_max = zeros(1,8);
for i=1:1:8
    AC_max(i) = max(abs(AC(i,2:256)));
end
ACm = max(AC_max);
```

REFERENCES

[1] A. Sharkovskii, "Coexistence of Cycles of a Continuous Map of the Line Into Itself," *International Journal of Bifurcation and Chaos,* vol. 5, p. 849, 1995.

[2] Y. Oono, "Period three implies chaos," *Progress of Theoretical Physics,* vol. 59, pp. 1028-1030, 1978.

[3] D. Ruelle, "What is a Strange Attractor?," *Notices of the AMS,* vol. 53, 2006.

[4] O. E. Rossler, "An equation for continuous chaos," *Physics Letters A,* vol. 57, pp. 397-398, 1976.

[5] R. M. May and others, "Simple mathematical models with very complicated dynamics," *Nature,* vol. 261, pp. 459-467, 1976.

[6] A. Zaikin and A. Zhabotinsky, "Concentration wave propagation in two-dimensional liquid-phase self-oscillating system," *Nature,* vol. 225, pp. 535-537, 1970.

[7] E. N. Lorenz, *The essence of chaos*: Routledge, 1995.

[8] A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers," presented at the Dokl. Akad. Nauk SSSR, 1941.

[9] M. Henon, "A two-dimensional mapping with a strange attractor," *Communications in Mathematical Physics,* vol. 50, pp. 69-77, 1976.

[10] H. Poincare, "Sur le probleme des trois corps et les equations de la dynamique divergence des series de M. Lindstedt," *Acta Mathematica,* vol. 13, pp. 1-270, 1890.

[11] E. Lorenz, "Deterministic Nonperiodic Flow," *Journal of Atmospheric Sciences,* vol. 20, pp. 130-148, 1963.

[12] M. Sushchik, L. S. Tsimring, and A. R. Volkovskii, "Performance analysis of correlation-based communication schemes utilizing chaos," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 47, pp. 1684-1691, 2000.

[13] F. Lau, M. Yip, C. Tse, and S. Hau, "A multiple-access technique for differential chaos-shift keying," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 49, pp. 96-104, 2002.

[14] K. Sun and J. Sprott, "Periodically Forced Chaotic System with Signum Nonlinearity," *International Journal of Bifurcation and Chaos,* vol. 20, pp. 1499-1507, 2010.

[15] M. Delgado-Restituto and A. Rodriguez-Vazquez, "Integrated chaos generators," *Proceedings of the IEEE,* vol. 90, pp. 747-767, 2002.

[16] K.-S. Tang, K. F. Man, G.-Q. Zhong, and G. Chen, "Generating chaos via x |x|," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 48, pp. 636-641, 2001.

[17] S. Ozoguz and A. Elwakil, "On the realization of circuit-independent nonautonomous pulse-excited chaotic oscillator circuits," *Circuits and Systems II: Express Briefs, IEEE Transactions on,* vol. 51, pp. 552-556, 2004.

[18] A. Radwan, A. Soliman, and A. El-Sedeek, "MOS realization of the modified Lorenz chaotic system," *Chaos, Solitons & Fractals,* vol. 21, pp. 553-561, 2004.

[19] S.-L. Chen, T. Hwang, and W.-W. Lin, "Randomness enhancement using digitalized modified logistic map," *Circuits and Systems II: Express Briefs, IEEE Transactions on,* vol. 57, pp. 996-1000, 2010.

[20] C.-Y. Li, T.-Y. Chang, and C.-C. Huang, "A nonlinear PRNG using digitized logistic map with self-reseeding method," presented at the VLSI Design Automation and Test (VLSI-DAT), 2010 International Symposium on, 2010.

[21] A. S. Mansingka, A. G. Radwan, and K. N. Salama, "Design, implementation and analysis of fully digital 1-D controllable multiscroll

chaos," presented at the Microelectronics (ICM), 2011 International Conference on, 2011.

[22] A. S. Mansingka, A. G. Radwan, and K. N. Salama, "Fully digital 1-D, 2-D and 3-D multiscroll chaos as hardware pseudo random number generators," presented at the Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on, 2012.

[23] M. A. Zidan, A. G. Radwan, and K. N. Salama, "The effect of numerical techniques on differential equation based chaotic generators," presented at the Microelectronics (ICM), 2011 International Conference on, 2011.

[24] J. Fridrich, "Symmetric ciphers based on two-dimensional chaotic maps," *International Journal of Bifurcation and Chaos,* vol. 8, pp. 1259-1284, 1998.

[25] L. Kocarev, G. Jakimoski, T. Stojanovski, and U. Parlitz, "From chaotic maps to encryption schemes," presented at the Circuits and Systems, 1998. ISCAS'98. Proceedings of the 1998 IEEE International Symposium on, 1998.

[26] G. Alvarez, P. Montoya, G. Pastor, and M. Romera, "Chaotic cryptosystems," presented at the Security Technology, 1999. Proceedings. IEEE 33rd Annual 1999 International Carnahan Conference on, 1999.

[27] M. Gotz, K. Kelber, and W. Schwarz, "Discrete-time chaotic encryption systems. I. Statistical design approach," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 44, pp. 963-970, 1997.

[28] C. Shannon, "Communication Theory of Secrecy Systems," *Bell System Technical Journal, Vol 28, pp. 656–715,* Oktober 1949.

[29] L. Shujun, "Analyses and New Designs of Digital Chaotic Ciphers," *School of Electronic and Information Engineering, Xi'an Jiaotong University,* 2003.

[30] L. Shujun, M. Xuanqin, and C. Yuanlong, "Pseudo-random bit generator based on couple chaotic systems and its applications in stream-cipher cryptography," in *Progress in Cryptology—INDOCRYPT 2001*, ed: Springer, 2001, pp. 316-329.

[31] T. Beth, D. E. Lazic, and A. Mathias, "Cryptanalysis of cryptosystems based on remote chaos replication," presented at the Advances in Cryptology—CRYPTO'94, 1994.

[32] R. Matthews, "On the derivation of a "chaotic" encryption algorithm," *Cryptologia,* vol. 13, pp. 29-42, 1989.

[33] D. D. Wheeler, "Problems with chaotic cryptosystems," *Cryptologia,* vol. 13, pp. 243-250, 1989.

[34] T. Habutsu, Y. Nishio, and I. Sasase, "A secret key cryptosystem using a chaotic map," *IEICE TRANSACTIONS (1976-1990),* vol. 73, pp. 1041-1044, 1990.

[35] T. Habutsu, Y. Nishio, I. Sasase, and S. Mori, "A secret key cryptosystem by iterating a chaotic map," presented at the Advances in Cryptology—EUROCRYPT'91, 1991.

[36] S. Li, Q. Li, W. Li, X. Mou, and Y. Cai, "Statistical properties of digital piecewise linear chaotic maps and their roles in cryptography and pseudo-random coding," in *Cryptography and Coding*, ed: Springer, 2001, pp. 205-221.

[37] M. I. Sobhy and A.-E. Shehata, "Methods of attacking chaotic encryption and countermeasures," presented at the Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on, 2001.

[38] S. Sen, C. Shaw, D. R. Chowdhuri, N. Ganguly, and P. P. Chaudhuri, "Cellular automata based cryptosystem (CAC)," in *Information and Communications Security*, ed: Springer, 2002, pp. 303-314.

[39] M. Jessa, "Data encryption algorithms using one-dimensional chaotic maps," presented at the The 2000 IEEE International Symposium on Circuits and Systems (ISCAS 2000), Geneva, 2000.

[40] G. Alvarez, F. Montoya, M. Romera, and G. Pastor, "Cryptanalysis of a chaotic encryption system," *Physics Letters A,* vol. 276, pp. 191-196, 2000.

[41] G. Jakimoski and L. Kocarev, "Analysis of some recently proposed chaos-based encryption algorithms," *Physics Letters A,* vol. 291, pp. 381-384, 2001.

[42] J.-C. Yen and J.-I. Guo, "A new image encryption algorithm and its VLSI architecture," presented at the Signal Processing Systems, 1999. SiPS 99. 1999 IEEE Workshop on, 1999.

[43] S. Su, A. Lin, and J.-C. Yen, "Design and realization of a new chaotic neural encryption/decryption network," presented at the Circuits and Systems, 2000. IEEE APCCAS 2000. The 2000 IEEE Asia-Pacific Conference on, 2000.

[44] V. A. Protopopescu, R. T. Santoro, and J. S. Tolliver, "Fast and secure encryption-decryption method based on chaotic dynamics," 5479513, 1995.

[45] S. Tao, W. Ruli, and Y. Yixun, "Perturbance-based algorithm to expand cycle length of chaotic key stream," *Electronics Letters,* vol. 34, pp. 873-874, 1998.

[46] N. Masuda and K. Aihara, "Cryptosystems with discretized chaotic maps," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 49, pp. 28-40, 2002.

[47] Y. Kenji and K. Tanaka, "Image encryption scheme based on a truncated Baker transformation," *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences,* vol. 85, pp. 2025-2035, 2002.

[48] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," presented at the Advances in Cryptology, 1985.

[49] P. Rio Piedras, "Cellular automaton public-key cryptosystem," *Complex Systems,* vol. 1, pp. 51-57, 1987.

[50] D. D. Wheeler and R. A. Matthews, "Supercomputer investigations of a chaotic encryption algorithm," *Cryptologia,* vol. 15, pp. 140-152, 1991.

[51] E. Biham, "Cryptanalysis of the chaotic-map cryptosystem suggested at EUROCRYPT'91," presented at the Advances in Cryptology—EUROCRYPT'91, 1991.

[52] L. Kocarev, "Chaos-based cryptography: a brief overview," *Circuits and Systems Magazine, IEEE,* vol. 1, pp. 6-21, 2001.

[53] G. Hu, Z. Feng, and L. Wang, "Analysis of a type of digital chaotic cryptosystem," presented at the Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on, 2002.

[54] L. Kocarev and G. Jakimoski, "Logistic map as a block encryption algorithm," *Physics Letters A,* vol. 289, pp. 199-206, 2001.

[55] G. Jakimoski and L. Kocarev, "Chaos and cryptography: block encryption ciphers based on chaotic maps," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 48, pp. 163-169, 2001.

[56] J. Urias, E. Ugalde, and G. Salazar, "A cryptosystem based on cellular automata," *Chaos: An Interdisciplinary Journal of Nonlinear Science,* vol. 8, pp. 819-822, 1998.

[57] S. Li, X. Zheng, X. Mou, and Y. Cai, "Chaotic encryption scheme for real-time digital video," presented at the Electronic Imaging 2002, 2002.

[58] M. Asim and V. Jeoti, "Efficient and simple method for designing chaotic S-boxes," *ETRI journal,* vol. 30, p. 170, 2008.

[59] G. Tang, X. Liao, and Y. Chen, "A novel method for designing S-boxes based on chaotic maps," *Chaos, Solitons & Fractals,* vol. 23, pp. 413-419, 2005.

[60] G. Chen, Y. Chen, and X. Liao, "An extended method for obtaining S-boxes based on three-dimensional chaotic baker maps," *Chaos, Solitons \& Fractals,* vol. 31, pp. 571-579, 2007.

[61] F. Ozkaynak and A. B. Ozer, "A method for designing strong S-Boxes based on chaotic Lorenz system," *Physics Letters A,* vol. 374, pp. 3733-3738, 2010.

[62] I. Hussain, T. Shah, and M. A. Gondal, "A novel approach for designing substitution-boxes based on nonlinear chaotic algorithm," *Nonlinear Dynamics,* vol. 70, pp. 1791-1794, 2012.

[63] Y. Wang, L. Yang, M. Li, and S. Song, "A method for designing S-box based on chaotic neural network," presented at the Natural Computation (ICNC), 2010 Sixth International Conference on, 2010.

[64] G. Xu, G. Zhao, and L. Min, "An Extended Method for Obtaining S-Boxes Based on Discrete Chaos Map System," presented at the Computational Intelligence and Security, 2009. CIS'09. International Conference on, 2009.

[65] G. Zaibi, F. Peyrard, A. Kachouri, D. Fournier-Prunaret, and M. Samet, "A new design of dynamic S-Box based on two chaotic maps," presented at the

Computer Systems and Applications (AICCSA), 2010 IEEE/ACS International Conference on, 2010.

[66] J. Peng, S. Jin, L. Lei, and X. Liao, "Construction and analysis of dynamic S-boxes based on spatiotemporal chaos," presented at the Cognitive Informatics Cognitive Computing (ICCI*CC), 2012 IEEE 11th International Conference on, 2012.

[67] I. Abd-ElGhafar, A. Rohiem, A. Diaa, and F. Mohammed, "Generation of AES Key Dependent S-Boxes using RC4 Algorithm," presented at the 13th International Conference on Aerospace Sciences & Aviation Technology, 2009.

[68] M. T. Tran, D. K. Bui, and A. D. Duong, "Gray S-box for advanced encryption standard," presented at the Computational Intelligence and Security, 2008. CIS'08. International Conference on, 2008.

[69] J. Liu, B. Wei, X. Cheng, and X. Wang, "An AES S-box to increase complexity and cryptographic analysis," presented at the Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on, 2005.

[70] D. Chen, D. Qing, and D. Wang, "AES Key Expansion Algorithm Based on 2D Logistic Mapping," presented at the Chaos-Fractals Theories and Applications (IWCFTA), 2012 Fifth International Workshop on, 2012.

[71] S. Kazmi and N. Ikram, "Chaos based key expansion function for block ciphers," *Multimedia Tools and Applications,* pp. 1-15, 2011.

[72] K. Faraoun, "Chaos-based key stream generator based on multiple maps combinations and its application to images encryption," *The International Arab Journal of Information Technology,* vol. 7, pp. 231-240, 2010.

[73] L. May, M. Henricksen, W. Millan, G. Carter, and E. Dawson, "Strengthening the Key Schedule of the AES," presented at the Proceedings of the 7th Australian Conference on Information Security and Privacy, London, UK, UK, 2002.

[74] X. L. Jialin Huang, "Transposition of AES Key Schedule," ed: Cryptology ePrint Archive, Report 2012/260, 2012.

[75] I. Saberi, B. Shojaie, and M. Salleh, "Enhanced Key Expansion for AES-256 by Using Even-Odd Method," presented at the Research and Innovation in Information Systems (ICRIIS), 2011 International Conference on, 2011.

[76] L. Xin-she, Z. Lei, and H. YU-pu, "A novel generation key scheme based on DNA," presented at the Computational Intelligence and Security, 2008. CIS'08. International Conference on, 2008.

[77] Z. Muda, R. Mahmod, and M. Sulong, "Key transformation approach for Rijndael security," *Information Technology Journal,* vol. 9, pp. 290-297, 2010.

[78] S. Sulaiman, Z. Muda, and J. Juremi, "The new approach of Rijndael key schedule," presented at the Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on, 2012.

[79] D. Coppersmith, "The Data Encryption Standard (DES) and its strength against attacks," *IBM journal of research and development,* vol. 38, pp. 243-250, 1994.

[80] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish encryption algorithm: a 128-bit block cipher*: John Wiley & Sons, Inc., 1999.

[81] E. Biham, R. Anderson, and L. Knudsen, "Serpent: A new block cipher proposal," presented at the Fast Software Encryption, 1998.

[82] N. I. of Standards and Technology, "Advanced Encryption Standard," in *NIST FIPS PUB 197*, ed, 2001.

[83] G. Paul and S. Maitra, *RC4 Stream Cipher and Its Variants*: CRC PressI Llc, 2011.

[84] P. Hawkes and G. Rose, "Primitive specification for SOBER-128," *IACR ePrint Archive,* vol. 473, p. 476, 2003.

[85] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM,* vol. 21, pp. 120-126, 1978.

[86] J. Daemen and V. Rijmen, *The Design of Rijndael*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002.

[87] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," presented at the First Advanced Encryption Standard (AES) Conference, 1998.

[88] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Efficient and High-Performance Parallel Hardware Architectures for the AES-GCM," *Computers, IEEE Transactions on,* vol. 61, pp. 1165-1178, 2012.

[89] N. Nedjah, L. M. Mourelle, and M. P. Cardoso, "A Compact Piplined Hardware Implementation of the AES-128 Cipher," presented at the Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on, 2006.

[90] D. Wang and X. Li, "Improved method to increase AES system speed," presented at the Electronic Measurement Instruments, 2009. ICEMI '09. 9th International Conference on, 2009.

[91] H. Yin, H. Debiao, K. Yong, and F. Xiande, "High-speed ASIC implementation of AES supporting 128/192/256 bits," presented at the Test and Measurement, 2009. ICTM '09. International Conference on, 2009.

[92] K. Gaj, "Very compact FPGA implementation of the AES algorithm," presented at the Proceedings of 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES), number 2779 in Lecture Notes in Computer Science, 2003.

[93] P. Hamalainen, T. Alho, M. Hannikainen, and T. D. Hamalainen, "Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core," presented at the Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on, 2006.

[94] P.-C. Liu, H.-C. Chang, and C.-Y. Lee, "A 1.69 Gb/s area-efficient AES crypto core with compact on-the-fly key expansion unit," presented at the ESSCIRC, 2009. ESSCIRC '09. Proceedings of, 2009.

[95] S. Thomas. (2007, Verilog Implementation of Rijndael Encryption Algorithm). Available: http://tstout.net/highlight.php?filename=.%2FSchool%2FEE+5751%2FAES+Project%2F&language=text&root=%2E

[96] M. E. Yalcin, J. A. K. Suykens, and J. Vandewalle, "True random bit generation from a double-scroll attractor," *Circuits and Systems I: Regular Papers, IEEE Transactions on,* vol. 51, pp. 1395-1404, 2004.

[97] T. Stojanovski, J. Pihl, and L. Kocarev, "Chaos-based random number generators. Part II: practical realization," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on,* vol. 48, pp. 382-385, 2001.

[98] C. M. Close and D. K. Frederick, *Modeling and Analysis of Dynamic Systems*: Houghton Mifflin School, 1993.

[99] G. Chen and J. Lu, "Dynamics of the Lorenz system family: analysis, control and synchronization," *SciencePress, Beijing,* 2003.

[100] G. Xie, P. Chen, and M. Liu, "Generation of Multidirectional Multiscroll Attractors under the Third-Order Jerk System," presented at the Information Science and Engineering, 2008. ISISE '08. International Symposium on, 2008.

[101] A. S. Rukhin, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; Dray, J.; Vo, S.; Bassham, L. E., "A statistical test suite for random and pseudorandom number generators for cryptographic application," *NIST Special Publication 800-22, 2010.*

[102] M. A. Zidan, A. G. Radwan, and K. N. Salama, "Random number generation based on digital differential chaos," presented at the Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on, 2011.

[103] A. Webster and S. E. Tavares, "On the design of S-boxes," presented at the Advances in Cryptology—CRYPTO'85 Proceedings, 1986.

[104] H. Feistel, "Cryptography and computer privacy," *Scientific american,* vol. 228, pp. 15-23, 1973.

[105] J. B. Kam and G. I. Davida, "Structured design of substitution-permutation encryption networks," *Computers, IEEE Transactions on,* vol. 100, pp. 747-753, 1979.

[106] A. K. Maini, *Digital Electronics: Principles, Devices and Applications.* New York, NY, USA: John Wiley & Sons, Inc., 2007.

[107] P. P. Mar and K. M. Latt, "New Analysis Methods on Strict Avalanche Criterion of S- Boxes," presented at the World Academy of Science, Engineering and Technology 48, 2008.

[108] B. Wei, D. Liu, and X. Wang, "Analysis of AES S-box with Walsh Spectrum," *Informatica (Slovenia),* vol. 26, 2002.

[109] S. Gao, "Walsh Spectrum of Cryptographically Concatenating Functions and its Application in Constructing Resilient Boolean Functions," *J. Comput. Inf. Syst,* vol. 7, pp. 1074-1081, 2011.

[110] J.-s. Y. L.-j. H. H.-c. Pan, "Data Encryption Method Using Discrete Fractional Hadamard Transformation," ed, 2009.

[111] T. H. Shah, I; Gondal, M.A.; Mahmood, H, "Statistical analysis of S-box in image encryption applications based on majority logic criterion," *International Journal of Physical Sciences,* vol. 6, pp. 4110-4127, 2011.

[112] L. R. Knudsen and T. About, "A Key-schedule Weakness in SAFER K-64," presented at the Advances in Cryptology, Proceedings Crypto'95, LNCS 963, 1995.

[113] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner*, et al.*, "Improved Cryptanalysis of Rijndael," presented at the Proceedings of the 7th International Workshop on Fast Software Encryption, London, UK, 2001.

[114] C. Jie, H. Liusheng, Z. Hong, and Y. Wei, "Improved related-key attack on 7-round AES-128/256," presented at the Computer Science and Education (ICCSE), 2010 5th International Conference on, 2010.

[115] A. Biryukov, D. Khovratovich, and Nikolic, Ivica, "Distinguisher and Related-Key Attack on the Full AES-256," presented at the Proceedings of the 29th Annual International Cryptology Conference on Advances in Cryptology, Berlin, Heidelberg, 2009.

[116] A. Biryukov and Nikolic, Ivica, "Automatic search for related-key differential characteristics in byte-oriented block ciphers: application to AES, Camellia, Khazad and others," presented at the Proceedings of the 29th Annual international conference on Theory and Applications of Cryptographic Techniques, Berlin, Heidelberg, 2010.

[117] E. Dawson, H. Gustafson, and A. Pettitt, "Strict Key Avalanche Criterion," *Australasian Journal of Combinatorics,* vol. 6, pp. 147-153, 1992.

[118] R. B. D'Agostino and M. A. Stephens, "Goodness-of-fit Technicques," in *Statistics: a Series of Textbooks and Monographs*. vol. 68, ed: Marcel Dekker Incorporated, 1986, pp. 97-193.

[119] G. Carter, E. Dawson, and L. Nielsen, "Key Schedule of iterative block ciphers," presented at the Proceedings of Information Security and Privacy - 3rd Australasian Conference, ACISP'98, 1998.

[120] F. Sattar and M. Mufti, "Spectral Characterization and Analysis of Avalanche in Cryptographic Substitution Boxes using Walsh-Hadamard Transformations," *International Journal of Computer Applications,* vol. 28, 2011.

[121] C. Carlet, "Partially-bent functions," *Designs, Codes and Cryptography,* vol. 3, pp. 135-145, 1993.