Design and Analysis of Cellular Automata Based Cryptographic Algorithms

Debdeep Mukhopadhyay

Design and Analysis of Cellular Automata Based Cryptographic Algorithms

Thesis submitted
by

Debdeep Mukhopadhyay

under the guidance of

Dr. Dipanwita Roy Chowdhury

For the award of the Degree of

Doctor of Philosophy
in
Computer Science and Engineering

Department of Computer Science and Engineering
Indian Institute of Technology
Kharagpur
March 2007

CERTIFICATE

This is to certify that the thesis entitled **Design and Analysis of Cellular Automata Based Cryptographic Algorithms** submitted by **Debdeep Mukhopadhyay**, a research scholar in the *Department of Computer Science and Engineering, Indian Institute of Technology, Kharagpur*, for the award of the degree of **Doctor of Philosophy** is an original research work carried under my supervision and guidance. The thesis has fulfilled all requirements as per the regulations of the Institute and in my opinion has reached the standard needed for submission. The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Dr. Dipanwita Roy Chowdhury
Dept. of Computer Science and Engg
Indian Institute of Technology
Kharagpur 721302 India
Dated: March, 2007

ACKNOWLEDGMENT

Now, as I have reached this juncture of my work where I may pen down this page, I find that I still have an uphill task ahead of me. Indeed,this dissertation would not have been possible without close interaction, inspiration and advice from numerous individuals.

First, I would like to express deep sense of gratitude and indebtedness to my advisor, Dr. Dipanwita Roy Chowdhury, for her constant involvement and guidance throughout the research work. Indeed her technical insight and intuition have helped me in exploring several interesting problems in the area of Cryptography. Overall, her guidance has greatly improved my fervour to carry on with research activities.

I must also extend my thanks to Prof N. B. Chakrabarty for his constant inspiration and bringing into my notice many latest scientific advancements in the world. From him I learnt that a true student of science and engineering should always be excited about acquiring more and more knowledge. I also sincerely thank Dr. Abhijit Das, for his various mathematical puzzles which have helped me in developing my aptitude in problem solving. Also I thank him for his teachings in music, which I never really grasped but nevertheless had nice escapes from daily routines.

I had the fortune of being blessed by encouraging and caring parents and brother. No words of thanks can describe their love and affection in handling me, while I was busy with my thesis work. Thanks also to my Grandmother, for her constant prayers for my well being.

No journey of man is possible without having friends, and I was extremely fortunate to have many of them both in the Department of Computer Science and Engineering and Advanced VLSI Design Laboratory, IIT Kharagpur. Special thanks to Arijit Mondal, not only for listening but also for responding to my strange questions in $GF(2)$ matrices. Heartiest thanks to Subrat, Abhishek, Shibaji and Santosh for being constantly aside during several crucial periods of this work. I also thank

my juniors and co-researchers, Jacob, Harish, Chitti, Shitikant, Amit, Baijayanta, Guruji, Chandranshu, Pallavi, Roshni, Gaurav, Kundan and Debojyoti for several enlightening discussions.

Finally, I must thank my Hero Honda Street motor-cycle for having saved valuable time and being constantly with me during my PhD life!

Debdeep Mukhopadhyay

Abstract

With the advent of electronic commerce and portable devices for communications, cryptology has become an exceedingly important science in the present day. The diversity of applications in which crypto-algorithms have to operate have increased and hence the requirement for efficient algorithms have grown. The present thesis deals primarily with the design and analysis of Cellular Automata (CA) based cryptosystems. Motivated by the fact that all prior ciphers based on CA have been found to be vulnerable to attacks, the present work first develops cryptographic primitives using Cellular Automata. Various CA based structures have been identified and characterized. The primitives exhibit interesting properties which are necessary to build crypto-applications like block ciphers and key agreement protocols.

Subsequently, the thesis develops a programmable CA based architecture called CASBox to generate cryptographically robust Substitution Boxes (S-Boxes). The work then investigates the effect of key mixing in block ciphers through addition modulo 2^n inplace of xoring. Finally the above concepts developed are combined to give rise to a block cipher named SPAMRC. In this construction the permutation layer is also designed using CA structures and is also self-invertible. The number of rounds of SPAMRC required to provide security against Linear and Differential Cryptanalysis is computed taking into consideration the effect of key mixing through addition.

The thesis then characterizes a special class of CA to generate expander graphs and then uses it to construct a hardware efficient one-way function, whose hardness depends on the combinatorial properties of expander graphs. The one-way function is then used to develop a new Cellular Automata based key agreement protocol. Since the key agreement technique does not use modular exponentiation it is suitable for low power appliances. The security of the protocol is proved in the formal model of Bellare and Rogaway.

The last part of the thesis deals with the evaluation of two standard cryptographic algorithms and their implementations. AES-Rijndael has been adopted by NIST as the worldwide standard for block ciphers. The present work proposes a fault based attack against Ri-

jndael. A step by step approach has been described how an attacker induces a fault, detects the nature of the fault induced and accordingly use different strategies to evaluate the key. The results of the attack have been compared with existing research in this area and has been found to be the strongest fault attack on AES. Finally the thesis deals with the customization of Cellular Message Encryption Algorithm (CMEA) developed by the Telecommunications Industry Association and widely used for wireless networks. Results show that with suitable modifications CMEA can be transformed into a strong cipher, which is essential for wireless security.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In today's world security of information is a fundamental necessity not only for military and diplomatic messages but also for private communications. Today's era of communication has increased the importance of financial data exchange, image processing, biometrics and e-commerce transactions which in turn has made data security an important issue. Cryptology is defined as the science concerned with communications in secure form. The goal of cryptology is thus construction of schemes, which can maintain desired security, even after malicious attempts. Cryptology consists of cryptography and cryptanalysis. The former involves the study and application of various techniques through which information may be rendered unintelligible to all but the intended receiver. On the other hand cryptanalysis is the science and art of breaking cryptosystems and recovering the secret information.

Design of cryptographic schemes is a daunting task. One cannot rely on intuitions regarding the typical state of the environment. Cryptography attempts to prove that the obtained designs are secure, using all available knowledge about possible attacks. On the other hand, cryptanalysis examines the cryptosystems for vulnerabilities and finds out practical attacks against the cryptographic schemes.

Recently, there has been a shift of focus in cryptology. Today's cryptology not only provides confidentiality, authentication, data integrity and non-repudiation, but has also the added task of providing security in menacing environments. Cryptology is a strange field of science. As opposed to other fields of science which has to work mostly against nature, cryptology works against a powerful, malicious adversary, often referred to as the eaves-dropper. The adversary attacking the system will try to manipulate the environment into conducive states and try to break the system by adopting strategies which the designer may have not envisioned. For the attacker it suffices to show a single successful weakness of the cryptosystem. A secured cryptosystem has to withstand all such types of attacks. This tussle between the cryptographer and the cryptanalyst has continued for ages.

1.1 Motivation

With the ever increasing growth of data communication in the field of e-commerce transactions and mobile communication data security has gained utmost importance. However the conflicting requirements of power, area and throughput of such applications make hardware cryptography an ideal choice. The advantages of hardware for cryptography are as follows:

1. Dedicated hardware devices can run encryption routines concurrently with the host computer which runs other applications. This parallel processing helps to significantly speed up the processes.

2. As hardware solutions are tamper proof, hardware based cryptography ensures confidentiality, integrity and authentication of cryptographic keys. Thus the keys can be safely stored inside the hardware. On the other hand software can be manipulated and the key can be reverse engineered by a determined attacker.

However, hardware is more expensive than software, especially if many hardware devices are required. Memory is yet another constraint for hardware designs. Hence the algorithms must be tailored properly for hardware implementations. They should be compact, scalable and modular to reduce the overall cost of production.

With this motivation the thesis analyzes both conventional and standard cryptographic algorithms, which are conducive for hardware implementations. In this search for hardware friendly cryptographic algorithms the primary tool used is the *Cellular Automata* (CA). The CA based architectures are regular, cascadable and also with lesser inter-connects, a feature helpful for VLSI designs. This motivates to employ CA for the design of cryptographic algorithms. Further, both the inter-connections and the number of clock cycles of the CA based structures may be programmed. This aids in the development of reconfigurable architectures. Last but not the least, the randomness provided by CA based functions passes also the test set by Donald Knuth[1]. Inspite of its many advantages, endeavours of obtaining ciphers based on Cellular Automata have been successfully cryptanalyzed. The primary reasons behind such attacks are the affine structures of CA and the lack of well-defined primitives required for cryptographic algorithms. The present work first develops CA based primitives like agreement functions, non-linear transformations, S-Boxes, one-way functions etc. The design of S-Box architectures, which are cryptographically robust and are also scalable, are of interest to the crypto-community. The work shows how to apply Cellular Automata in the design of such S-Boxes. Although, S-Boxes provide non-linearity and much needed security against cryptanalysis to block ciphers, an interesting technique to increase the security margin is through the use of arithmetic operations, like integer addition. Such operations are easy to implement in hardware

and may reduce the total number of rounds required to prevent attacks. With this motivation, key mixing through integer addition in place of the conventional xor have been studied. The development of hardware efficient one-way functions are of utmost requirement, as conventional one-way functions based on modular exponentiation are not satisfactory in terms of area-delay product. The thesis also shows how to realize an efficient one-way function, combining the algebraic properties of CA and the combinatorial properties of a special class of graph, known as Expander Graphs. The CA based primitives are applied to construct block ciphers and key agreement algorithms.

The thesis subsequently looks at standard cryptographic algorithms like AES (Advanced Encryption Standard), which is the world-wide standard for block ciphers and Cellular Message Encryption Algorithm, which is the standard algorithm for wireless CDMA networks. The complexity of the implementation of algorithms like the AES raises concerns regarding their reliability. Research is therefore needed to analyze cryptographic algorithms in an environment where fault is induced, either accidentally or intentionally. The work adds to the existing literature on fault attacks on the AES algorithm.

Finally, the thesis customizes the successful cryptanalysis of the CMEA algorithm. Cellular Message Encryption Algorithm (CMEA) [2] has been developed by the Telecommunications Industry Association (TIA) to encrypt digital cellular phone data. CMEA is one of the four cryptographic primitives specified for telecommunications and is designed to encrypt the control channel, rather than the voice data of cellular phones. In March 1997, Counterpane Systems and UC Berkeley jointly [3] published attacks on the cipher showing it had several weaknesses. The present work revisits the CMEA algorithm and customizes it into a modified algorithm called CMEA-I which prevents the previous attacks on CMEA and also provides better security margin against Linear and Differential cryptanalysis compared to CMEA.

1.2 Contribution of the Thesis

The contribution of the thesis may be summarised as follows:

- **Characterization of Complemented Cellular Automata:** In the present work a class of complemented null boundary Cellular Automaton has been characterized. One particular class of complemented CA with rule 153 has been studied and it has been proved that the cycle lengths are equal irrespective of the number of cells. The exact dependence of the cycle lengths on the number of cells has been formulated. There are no existing rules to relate the separate cycles formed by the CA. Thus the entire state space cannot be used and this leads to worries for the application of the CA in cryptography. The current

work develops new rules which govern the transition between the cyclic subspaces. These properties promise the CA to function as an ideal candidate for cryptographic algorithms.

- **Design of CASBox, a programmable structure to generate S-Boxes using CA:** In this work the Theory of Cellular Automata has been applied, to the best of our knowledge for the first time to generate robust S-Boxes on hardware. The work discusses how a special class of linear group CA, called the maximum length CA can generate cryptographically strong S-Boxes. In short the generated S-Boxes are balanced, satisfy Strict Avalanche Criterion, have high non-linearity of the component functions as well as their non-zero linear combinations, have high algebraic degree and are robust against linear and differential cryptanalysis. The work shows that the CA based S-Box, called CASBox, is programmable, modular, and generates a large number of S-Boxes all of which are equally cryptographically strong. The architecture proposed in the work has been implemented on a Xilinx XCV-1000 FPGA platform and the results show that the structure is efficient and scalable. The proposed CA based S-Box construction is extremely efficient due to the inherent parallelism in Cellular Automata transformations. Also as the chosen maximum length CA has a three neighbourhood cell the length of the interconnects would be less compared to an LFSR based S-Box, a feature helpful for VLSI implementations.

- **Key Mixing in Block Ciphers through Addition modulo 2^n and its effect on linear cryptanalysis:** In Substitution-Permutation Network (SPN) like AES, DES the key mixing step is performed by key xoring where the key bits are simply xored (that is added without carry) with the data bits before each round and after the last round. The present work investigates if the replacement of key xoring step in block ciphers by modular integer addition of the key shall help to reduce the bias of linear expressions and hence foil Linear Cryptanalysis.

- **Design of SPAMRC, Construction of a CA based Block Cipher:** A block cipher named SPAMRC (Substitution Permutation Addition Message Round Cipher) is proposed using the above constructions. The diffusion layer, named as CAMixColumn is a self-invertible linear MDS (Maximum Distance Separable) mapping implemented using a maximum length CA. Finally the number of rounds required to provide sufficient security margin against Linear and Differential cryptanalysis are computed taking into account the effect of key mixing through addition modulo 2^n in place of xoring.

- **Expander Graphs based on Cellular Automata and its application in:**

  1. **The development of one-way functions:** The one-way function is based on the combinatorial properties of expander graphs generated by a

special class of Cellular Automata, called Two Predecessor Single Attractor CA (TPSA-CA). The proposed architecture has been implemented in Verilog HDL and has been verified using RTL simulations using Mentor Graphics ModelSim SE. The Verilog RTL has been prototyped on a Xilinx Virtex XCV1000 FPGA (pkg bg560). Analysis show that the resource utilization increases linearly with the input size and the forward transformation is efficient. However, the complexity of inverting the one-way function appears to be intractable.

2. **The development of Key Agreement algorithm using Cellular Automata:** The proposed protocol performs the initial key agreement with authentication and key confirmation using three message exchanges. The designed one-way function is used by two communicating parties A and B to derive the session key K_{AB} with the desired property of keyfreshness. The one-way function is also used for authentication and key confirmation. Finally, the protocol has been shown to be secured in the Bellare Rogaway model.

- **Fault Based Side-Channel Attack on AES-Rijndael:**

  In this work we present a fault based side-channel attack on AES. Attack strategies have been proposed, when the attacker is able to induce byte level faults in the last three rounds of the block cipher AES. Extensive experimentations have been performed on a PC and it has been found that the key can be obtained using only two faulty ciphertexts in few seconds. The experimental results have been presented and compared with previous fault attacks against AES.

- **Customization of Cellular Message Encryption Algorithm (CMEA):** Cellular Message Encryption Algorithm (CMEA) proposed by the Telecommunications Industry Association has been successfully cryptanalyzed in March 1997 jointly by Counterpane Systems and UC Berkeley. This motivates the requirements for alternatives. In this work the algorithm has been first studied to comprehend the reasons for the successful attacks and finally customized so that the existing attacks do not work. The customized CMEA algorithm, named CMEA-I has been found to have better security margins against Linear and Differential cryptanalysis than CMEA without compromising the efficiency in implementation.

1.3 Organization of the Thesis

The rest of the dissertation is organized as follows:

- **Chapter 2:** The chapter presents a survey on works related to the construction of ciphers. In particular, reports on existing works in the design of block ciphers have been presented. A brief note on side-channel attacks and their countermeasures has also been reported. The chapter further mentions works on the design and analysis of key agreement protocols and one-way functions. The chapter finally comments on the future directions of cryptography.

- **Chapter 3:** The design and analysis presented in this work are based on the Theory of Cellular Automata (CA). Hence in this chapter the mathematical background of the work is presented. The chapter also includes preliminaries regarding some desirable cryptographic properties which are often used in the cryptanalysis of block ciphers.

- **Chapter 4:** The chapter presents the characterization of complemented CA with rule 153 and generalizes it to a class of CA structures with the same characterization. The work reveals that the properties may find useful applications in cryptography.

- **Chapter 5:** The chapter deals with the construction of S-Box based on Cellular Automata based structures. The chapter presents both theoretical and practical foundations to show that such CA based S-Boxes (named as CASBox) generate programmable, modular, scalable and cryptographically strong S-Boxes.

- **Chapter 6:** The chapter deals with the effect of key mixing through addition modulo 2^n instead of xoring on Linear Cryptanalysis (LC). The chapter theoretically estimates the effect of such a key mixing on the security of block ciphers against LC and also experimentally verifies the results with the help of an example block cipher.

- **Chapter 7:** The chapter develops a CA based block cipher, named SPAMRC based on the developed results. The chapter also presents the construction of an MDS diffusion layer using CA based rules. The work computes the security margin of four rounds of the cipher against Linear and Differential cryptanalysis, taking into consideration the extra security margin provided by key mixing using addition in place of xoring.

- **Chapter 8:** The chapter presents the construction of one-way functions based on expander graphs using Cellular Automata. The chapter also presents a key

agreement protocol using the one-way function. The protocol has been proved to be secure in the Bellare Rogaway model.

- **Chapter 9:** The chapter presents a new fault based attack on the AES algorithm. The work shows through examples how induced faults at the input of eighth, ninth and tenth rounds may be used to compute the key. Finally the work has been compared to existing fault attacks.

- **Chapter 10:** The chapter studies the successful cryptanalysis of the CMEA algorithm and customizes it to prevent the attacks. The security margin of the customized cipher has been evaluated in the light of Linear and Differential cryptanalysis.

- **Chapter 11:** The chapter concludes the dissertation and suggests some future directions of research in this area of work.

Chapter 2

A Survey

... Of Secrecy I am Silence...[*]

2.1 Introduction

The art of keeping message secret is cryptography, while cryptanalysis is the study attempted to defeat cryptographic techniques. *Cryptography* is used to protect information from illegal access. It largely encompasses the art of building schemes (*ciphers*) which allow secret data exchange over insecure channels[4]. The need of secured information exchange is as old as civilization itself. It is believed that the oldest use of cryptography was found in non-standard hieroglyphics carved into monuments from Egypt's Old Kingdom. In 5 BC the Spartans developed a cryptographic device, called *scytale* to send and receive secret messages. The code was the basis of Transposition ciphers, in which the letters remained the same but the order is changed. This is still the basis for many modern day ciphers. The other major ingredient of many modern day ciphers is Substitution ciphers, which was used by Julius Caesar and is popularly known as Caesar's shift cipher. In this cipher, each plaintext character was replaced by the character 3 places to the right in the alphabet set modulo 26. However in the last three decades cryptography has grown beyond designing ciphers to encompass also other activities like design of signature schemes for signing digital contracts. Also the design of cryptographic protocols for securely proving one's identity has been an important aspect of cryptography of the modern age. Yet the construction of encryption schemes remains, and is likely to remain,

[*] *"Maunam Caivasmi Guhyanam Jnanam"*, Bhagavad Gita, Vibhuti-yoga, Sloka 38

a central enterprise of cryptography [5]. The primitive operation of cryptography is hence *encryption*. The inverse operation of obtaining the original message from the encrypted data is known as *decryption*. Encryption transforms messages into representation that is meaningless for all parties other than the intended receiver. Almost all cryptosystems rely upon the difficulty of reversing the encryption transformation in order to provide security to communication [6]. *Cryptanalysis* is the art and science of breaking the encrypted message. The branch of science encompassing both cryptography and cryptanalysis is cryptology and its practitioners are cryptologists. One of the greatest triumph of cryptanalysis over cryptography was the breaking of a ciphering machine named Enigma and used during World war 2. In short cryptology evolves from the long lasting tussle between the cryptographer and cryptanalyst.

For many years many fundamental developments in cryptology outpoured from military organisations around the world. One of the most influential cryptanalytic papers of the twentieth century was William F. Friedman's monograph [7] entitled *The Index of Coincidence and its Applications in Cryptography*. For the next fifty years research in cryptography was done in a secret fashion, with the exception of the revolutionary contribution of Claude Shannon's paper "*The communication Theory of Secrecy Systems*", which appeared in the *Bell System Technical Journal* in 1949[8].

However after the world wars cryptography became a science of interest to the research community. The *Codesbreaker* by David Kahn produced the remarkable history of cryptography [9]. The significance of this classic text was that it raised the public awareness of cryptography. The subsequent development of communication and hence the need of privacy in message exchange also increased the impetus on research in this field. A large number of cryptographers from various fields of study began to contribute leading to the rebirth of this field. Horst Fiestel [10] began the development of the US Data Encryption Standard (DES) and laid the foundation of a class of ciphers called as private or symmetric key algorithms. The structure of these ciphers became popular as the Fiestel Networks in general. Symmetric key algorithms use a single key to both encrypt and decrypt. In order to establish the key between the sender and the receiver they required to meet once to decide the key. This problem commonly known as the key exchange problem was solved by Martin Hellman and Whitfield Diffie [11] in 1976 in their ground breaking paper *New Directions in Cryptography*. The developed protocol allows two users to exchange a secret key over an insecure medium without any prior secrets. The work not only solved the problem of key exchange but also provided the foundation of a new class of cryptography, known as the public key cryptography. As a result of this work the RSA algorithm, named after the inventors Ron Rivest, Adi Shamir and Leonard Adleman, was developed[12]. The security of the protocol was based on the computational task in factoring the product of large prime numbers.

Cryptology has evolved further with the growing importance of communications

and the development in both processor speeds and hardware. The modern day cryptographer has thus more work than merely jumbling up messages. He has to look into the application areas in which the cryptographic algorithms have to work. The transistor has become more powerful. The development of the VLSI technology (now in submicrons) have made the once cumbersome computers faster and smaller. The more powerful computers and devices will allow the complicated encryption algorithm run faster. The same computing power is also available to the cryptanalysts who will now try to break the ciphers with a straight forward brute force analysis. The world has thus changed since the DES was adopted as the standard cryptographic algorithm and DES was feeling its age. Large public literature on ciphers and the development of tools for cryptanalysis urged the importance of a new standard. The National Institute for Standards and Technology (NIST) organized a contest for the new Advanced Encryption Standard (AES) in 1997. The block cipher Rijndael emerged as the winner in October 2000 due to its features of security, elegance in implementations and principled design approach. Simultaneously Rijndael was evaluated by cryptanalysts and a lot of interesting works were reported. Cryptosystems are inherently computationally complex and in order to satisfy the high throughput requirements of many applications, they are often implemented by means of either VLSI devices or highly optimized software routines. In recent year such cryptographic implementations have been attacked using a class of attacks which exploits leaking of information through side-channels like power, timing, intrusion of faults etc. In short as technology progresses new efficient encryption algorithms and their implementations will be invented which in turn shall be cryptanalyzed in unconventional ways. Without doubt cryptology promises to remain an interesting field of research both from theoretical and application point of view.

2.2 Cryptography - Some Technical Details

The aim of the cryptographer is to find methods to secure and authenticate messages. The original message is called the plaintext and the encrypted output is called the ciphertext. A secret key is employed to generate the ciphertext from the plaintext. The process of converting the plaintext to the cipher text is called encryption and the vice versa is called decryption. The cryptographer tries to keep the massages secret from the attacker or intruder. A cryptosystem is a communication system encompassing a message source, an encryptor, an insecure channel, a decryptor, a message destination and a secure key transfer mechanism. The scenario of a cryptographic communication is illustrated in **Fig. 2.1**.

The goal of the cryptanalyst is to thwart the efforts of the cryptographer by breaking the cipher. He is a powerful entity who studies the cipher and uses algebraic

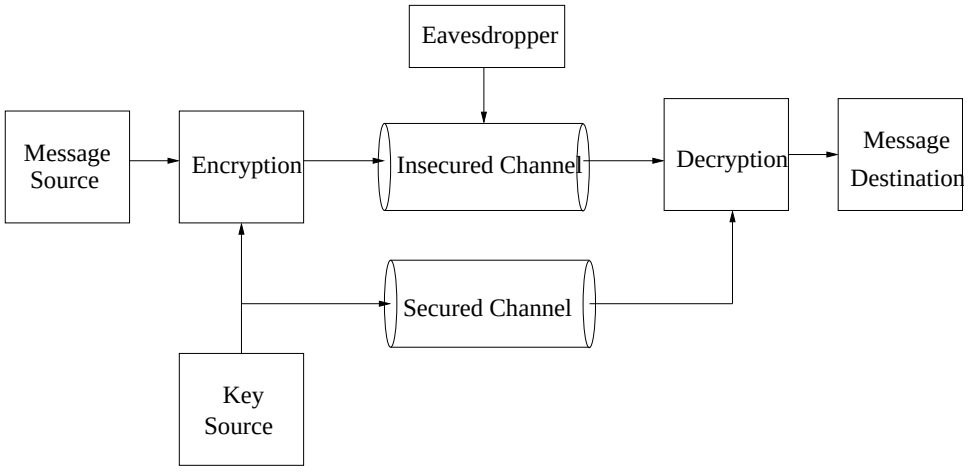


Figure 2.1: **Secret Key Cryptosystem Model**

and statistical techniques to attack a cryptographic scheme. A cryptanalytic attack is a procedure through which the cryptanalyst gains information about the secret key. Attacks are classified according to the level of a-priori knowledge available to the cryptanalyst.

A **Ciphertext-only attack** is an attack where the cryptanalyst has access to ciphertexts generated using a given key but has no access to the corresponding plaintexts or the key. A **Known-plaintext attack** is an attack where the cryptanalyst has access to both ciphertexts and the corresponding plaintexts, but not the key.

A **Chosen-plaintext attack** is an attack where the cryptanalyst can choose plaintexts to be encrypted and has access to the resulting ciphertexts, again their purpose being to determine the key.

A **Chosen ciphertext attack** is an attack in which the cryptanalyst can choose ciphertexts and can obtain the corresponding plaintext. The attacker has access to the decryption device.

The attacks are measured against a worst case referred to as the brute force method. The method is a trial and error approach, whereby every possible key is tried until the correct one is found. Any attack that permits the discovery of the correct key faster than the brute force method, on average, is considered successful. An important principle known as the Kerckhoff's principle states that, the secrecy of a cipher must reside entirely in the key. Thus an enemy will have a complete knowledge of the cipher but shall not know the key. A secured cryptographic scheme should withstand the attack of such a well-informed adversary.

There are two distinct types of ciphers.

- Private-key (or secret-key or symmetric) ciphers.
- Public-key ciphers.

These types differ in the manner in which keys are shared. In symmetric-key or private-key cryptography both the encryptor and decryptor use the same key. Thus, the key must somehow be securely exchanged before secret key communication can begin (through a secured channel, **Fig. 2.1**). In public key cryptography the encryption and decryption keys are different.

In such algorithms we have a key-pair, consisting of:

- Private Key, which must be kept secret and is used to decrypt messages.
- Public Key, which can be freely distributed and is used to encrypt messages.

The two parties namely Alice and Bob are communicating with each other and have their own key pair. They distribute their public keys freely. Alice encrypts the message intended for Bob, using Bob's public key. The resulting ciphertext can then be decrypted only using Bob's private key.

Public and Private (or symmetric) Key algorithms have complementary advantages and disadvantages. They have their specific application areas. Symmetric Key ciphers have higher data throughput but the key must remain secret at both the ends. Thus in a large network there are many key pairs that should be managed. Sound cryptographic practice dictates that the key should be changed frequently for each communication session. The throughputs of the most popular public-key encryption methods are several orders of magnitude slower than the best known symmetric key schemes. In a large network the number of keys required are considerably smaller and needs to be changed less frequently. In practice thus public-key cryptography is used for efficient key management while symmetric key algorithms are used for bulk data encryption. Since the present thesis deals with symmetric key algorithms, it has been elaborated in the subsequent discussions.

2.2.1 Symmetric-Key Cryptography

The symmetric-key algorithms can be divided into two types:

- Block Ciphers
- Stream Ciphers

Block ciphers operate on blocks of plaintexts and ciphertexts. Identical plaintext blocks always encrypt to identical ciphertext blocks for a given key. The algorithm DES uses 64 bit as block size whereas Rijndael uses 128 bits. The security of the cipher depends on the block sizes for the data and the key. From the Kerckhoff's principle as the key space becomes bigger the probability of success of the adversary reduces, thus increasing the security of the scheme. Stream ciphers are a class of encryption algorithms that encrypt individual characters (usually binary digits) of a plaintext message one at a time, using an encryption transformation which varies with time. A Vernam Cipher is a stream cipher in which the key length is the same as the message. The ciphertext is generated by adding the message to the key digit by digit. For the special case where the key is used only once the cipher is termed as one time pad. Shannon proved in 1949 that the cipher was perfectly secure and hence is unbreakable. However such a one-time pad is not feasible as the key size becomes too large.

As a significant portion of our work is based on block ciphers, we focus on such type of cryptographic schemes.

Block Ciphers

The block ciphers are a widely researched topic in the present cryptoworld. Security and efficient implementations are two of the most important design objectives of such ciphers. Shannon's principle of confusion and diffusion are applied in the designs of block ciphers. *Confusion* obscures the relationship between the plaintext and the ciphertext. This work to make the relationship between the statistics of the plaintext and the ciphertext as complicated as possible. *Diffusion* dissipates the redundancy of the plaintext by spreading it over the ciphertext. *Product* or *iterated* cipher is one whereby confusion and diffusion are achieved by the repeated application of the same simple ciphers. This helps the iterated cipher achieve security goals while being easily implementable at the same time. The cipher structure which is repeated is known as the *round*.

Mathematically a block cipher is the following pair of functions:

$$E : Z_2^k \times Z_2^n \rightarrow Z_2^n$$
$$D : Z_2^k \times Z_2^n \rightarrow Z_2^n$$

E is the encryption function which converts an n bit plaintext to an n bit ciphertext using a k bit key. D is the corresponding decryption function, which converts the n bit ciphertext to the n bit plaintext. Here, n is referred to as the block size,

while k is the size for key. The block ciphers are product ciphers and comprise of repetition of the same structure denoted by the function E_i. The number of rounds in the block cipher is r. Each round of the block cipher has an independent share of key K_i, where K_i is known as the round key and is generated by a *Key Scheduling Algorithm* from the input key K. Thus, for an n bit plaintext and k bit input key $E(P, K) = E_r(\ldots(E_2(E_1(P, K_1), K_2))\ldots), K_r)$. The rounds are more or less identical with minimal variations in the first and last rounds. Also it may be noted that there must be a key mixing layer at the beginning of the block cipher as otherwise the first encryption may be undone as it is invertible.

In modern ciphers there are two different types of confusion. One of them is key dependent while the other one is independent of key. Key dependent confusions are generally obtained by the bit-wise xor of key bits and the plaintexts. Non-linear transformations are used to obtain key independent confusions. The non-linear step is often implemented by means of a look-up table, often named as the substitution box or the S-Box. The S-Box design is one of the most important aspects of the design of ciphers. If the S-Box did not provide non-linearity to the cipher the cascading of the rounds could be represented by a simple step. Thus with a linear S-Box any number of rounds is equivalent to a single application of a different S-Box. The S-Box provides non-linearity to the cipher which provides the much needed confusion and helps to ward off cryptanalysis.

Cryptanalysis of Block Ciphers

The strength of a cipher can be measured by its resistance to known cryptanalytic techniques. The attacks types, as mentioned earlier, can be broadly categorised into (i) ciphertext only, (ii) known plaintext, (iii) chosen plaintext and (iv) chosen ciphertext.

Various attacks were developed in order to evaluate the strengths of block ciphers. Among the notable attacks were Hellman's time-memory trade-off attack [13], meet-in-the-middle attack [14], key degeneracy attack [15, 16], maximum likelihood estimation [17], method of formal coding [18] and related key [19]. In recent years several cryptanalysis of block ciphers have been published to assess the security of various ciphers. Reduced variants of the Advanced Encryption Standard (AES) has been cryptanalyzed and has led to large number of attacks against block ciphers [20, 21, 22, 23]. Recently a class of attacks have been proposed against AES Rijndael which uses overdefined equations [24, 25, 26]. The new kind of attack known as the algebraic attack [27, 24, 26] brings new design criteria to the surface. The algebraic attacks exploit the existence of low degree equations and once a system of nonlinear multivariate equations of low degree is obtained, it is solved by efficient methods like XL [28]. However two of the most powerful cryptanalytic techniques are:

- Differential Cryptanalysis [29]: In 1990 Eli Biham and Ali Shamir introduced the method. Let us consider a pair of known plaintexts (x_1 and x_2) maintaining a fixed difference, measured by the bitwise xor of x_1 and x_2. Due to the nature of the round transform the corresponding difference in the ciphertext depends upon the key. Thus by encrypting a large number of pairs of plaintexts, all with a given difference, and examining the difference in the ciphertexts it is possible to gain knowledge about the key.

- Linear Cryptanalysis [30]: Linear Cryptanalysis was proposed by Mitsuru Matsui in 1993. The approach was based on the linear approximations to the non-linear S-Box. The basic premise is that the linear approximation of a non-linear S-Box will hold with a certain probability. By chaining such linear approximations one approximate the entire cipher through a linear equation.

Several attacks combine these cryptanalytic techniques to obtain new attacks e.g., differential-linear attacks [31], miss-in-the-middle attacks [32, 33], boomerang attacks [34] and bilinear attacks [35]. Recently Eli Biham has further proposed some new kinds of attacks named as differential-bilinear attacks, combinations of differential and linear attacks and combinations of differential, bilinear and boomerang attacks [36].

Side Channel Attacks on Block Ciphers

From around mid 90's, a new research area that has gained focus is *side channel cryptanalysis*. The research area analyzes the *unintended* information leakage that is provided by naive implementations of a *secure* mathematical algorithm and its impact on the extraction of the secret key material. Research shows that while the efforts to carry out such attacks are low, the development of proper countermeasures is nontrivial. Related implementation based attacks are known as *fault analysis* that try to exploit *intended* leakage in implementations of cryptographic algorithms. Both the kinds of attacks are of practical importance for the development of secured products.

The attacks are of two nature: *active*, which change the environmental conditions and also sometimes physically open the cryptographic device. Another class of attack is the passive attack, which simply observes the inherent leakage of the cryptographic device and evaluates the secret key used. The leaked information may be the timing information (*timing attack*) [37, 38] or power consumption (*power attack*) [39]. In 1996 Paul Kocher described a method in which the keys of RSA were compromised by measuring the execution time of the overall cryptographic operation [37]. In 1999, a more effective side-channel cryptanalysis technique was introduced by Paul Kocher etal, where the power consumption of the device was analyzed to evaluate the key

[39]. There are two variants of the attack: simple power analysis and differential power analysis [40]. After the discovery of these kinds of attacks several improved attacks which exploit these side-channels [41, 42, 43, 44, 45, 46, 47] and their combinations have been discovered [48, 49, 50]. New design criteria for the important components of cipher, like the S-Boxes, are proposed to prevent side-channel attacks [51, 52]. The effect is so strong that modifications are being suggested in the normal Electronic Design Automation (EDA) of Integrated Circuits [53]. In order to prevent the resistance against side-channel analysis different logic styles are also proposed [54].

After the advent of smart cards, the implementations of the cryptographic algorithms inside the card were evaluated in the light of side channel cryptanalysis. Various countermeasures have been suggested to make the implementations robust against these kinds of attacks. An interesting technique proposed to counter the side-channel attacks is the concept of masking [55, 56, 57]. Essentially this method randomize the computations depending on the secret key and make the information leakage of an implementation unpredictable to an attacker.

Another class of side-channel attacks is known as fault based cryptanalysis. Fault attacks aim to cause errors during the processing of a cryptographic device. Hence the cryptographic device enters a modified execution path or returns erroneous cryptograms. Finally mathematical cryptanalysis is applied to exploit the wrong cryptograms and determine the secrets.

The attacks are based on assumptions regarding the control on fault location and fault occurence time and which are formally known as *fault models*.

In 1996, Dan Boneh, Richard A. DeMillo and Richard J. Lipton reported that certain implementations of RSA and other algorithms are vulnerable assuming that a certain transient fault occurred during the processing[58]. Shortly after in 1996, Eli Biham and Adi Shamir announced the approach of Differential Fault Analysis against secret key cryptosystems[59]. Here the same plaintext is encrypted and it is assumed that faults occur in the last three rounds of DES. The faulty cryptograms thus obtained are used to derive the key used in the encryptions. In 1999, Oliver Kommerling and Markus Kuhn reported that for the generation of faults, glitches in the external power lines in the internal clock lines are quite useful in practice[60]. A new kind of fault inductions were revealed in 2002, as semi-invasive optical fault inductions were presented by Sergei Skorobogatov and Ross Anderson [61]. These attacks allow specific control on a single register and gives great armoury to a fault based cryptanalysis. An alternative semi-invasive approach using eddy currents were proposed by Jean-Jacques Quisquater and David Samyde[62].

In short there are two kinds of fault based attacks.

- **Simple Fault Analysis:** It exploits a direct relationship between a faulty re-

sult and the secret key in the implementation. Often the sequence of operations are disturbed in order to develop new attacks.

- **Differential Fault Analysis:** This class of attacks need a certain number of faulty results using the same cryptographic key which are caused in a transient way. The faulty results are used to reduce the key space. Note, that the practical relevance of an attack strongly depends on the fault model used.

The impact of fault attacks have grown much recently after the work proposed in [61]. The Advanced Encryption Standard (AES) has also been analyzed in the perspective of fault based attacks. Differential Fault Analysis (DFA) on AES was reported in [63]. Another fault based attack on AES was reported in [64]. P. Dusart etal [65] performs a Differential Fault Analysis on AES and shows that using a byte level fault induction anywhere between the eighth round and ninth round the attacker is able to break the key with 40 faulty ciphertexts. Finally, [66] shows that when a byte level fault occurs at the input of the eighth round or the input of the ninth round of a ten round AES-128 algorithm, an attacker can retrieve the whole AES-128 key with only two faulty ciphertexts.

In order to prevent fault based side channel attacks it is required that the cryptographic device itself checks that the result obtained is correct. In the simplest way, this can be done by computing the same operation twice. The implementation should be capable to detect modifications of security variables. For block ciphers which are invertible, the result can be checked by calculating the inverse operation and checking whether the cryptogram gives the message back.

For an underlying hardware countermeasure, Sergei P. Skorobogatov and Ross J. Anderson proposed self-timed dual-rail-logic circuits that include an alarm mechanism.

Finally note, that there are not any sufficient countermeasures in case the attacker has an ideal fault control using short-timed multiple fault injections. This makes the design and analysis of fault attack preventing cryptographic implementations an important field of research.

2.2.2 Design of SPN ciphers resistant to Linear and Differential Attacks

Substitution-Permutation Networks (SPN) are practical product ciphers. In these structures a basic module is repeated and each such modules are known as *rounds*. The rounds are composed of the following operations:

1. Key mixing layer $\sigma(k)$, in which the key (k) is mixed to the data using simple key xoring $(\oplus)$
2. Non-linear layer, commonly known as Substitution Boxes or S-Boxes. The S-Boxes in SPN ciphers are necessarily bijective in SPN ciphers as opposed to Feistel networks where they need not be so.
3. The diffusion layer θ, which is linear with respect to $\oplus$.

The first part of the thesis deals with the construction of an SPN cipher using Cellular Automata (CA) based structures. Though the application of CA to realize substitution and permutation blocks with proven margin against Linear Cryptanalysis (LC) and Differential Cryptanalysis (DC) is novel and central to the present dissertation, the history of SPN ciphers and their design criteria have been set from around mid 70's by many eminent researchers. We provide a brief survey on some of the important works in this area.

2.2.3 Design of S-Boxes: The Substitution Layer

Design of the block ciphers is largely dependent on the robustness of its S-Box. Weakness of the S-Box is often compensated by increasing the number of rounds of a block cipher. But if the objective is to design a cipher which is fast and also secure, the number of rounds has to be less and at the same time the S-Boxes must be amenable to efficient implementations. Thus it is imperative to design efficient S-Boxes which are defiant against cryptanalysis. In order to design *proper* S-Boxes design criteria were searched for by several researchers [67, 68, 69, 70, 71]. The component functions of the above constructions were quadratic and did not satisfy the Strict Avalanche Criterion (SAC). The works were improved in the methodology proposed in [72].

One of the constructions proposed in [73] is based on what is known as the Maiorana-McFarland method.

Definition 2.1 *[74] The class of Maiorana-McFarland (n, m)-functions is the set of those functions F which can be written in the form:*

$$F(x,y) \;=\; x \times \begin{pmatrix} \varphi_{11}(y) & \dots & \varphi_{1m}(y) \\ \vdots & \ddots & \vdots \\ \varphi_{r1}(y) & \dots & \varphi_{rm}(y) \end{pmatrix} + H(y), (x,y) \in F_2^r \times F_2^s$$

where r and s are two integers satisfying $r+s=n$, H is any (s,m)-function and, for every index $i \leq r$ and every index $j \leq m$, φ_{ij} is a Boolean function on F_2^s.

Any Maiorana-McFarland's (n, m)-function F can be written in the form:

$$F(x, y) = (\bigoplus_{i=1}^{r} x_i \varphi_{i1}(y) \oplus h_1(y), \ldots, \bigoplus_{i=1}^{r} x_i \varphi_{im}(y) \oplus h_m(y))$$

where $H = (h_1, \ldots, h_m)$. After denoting, for each $i \leq m$, by ϕ_i the (s, r) function which admits the Boolean functions $\varphi_{1i}, \ldots, \varphi_{ri}$ for coordinate functions, we can rewrite the above relation as:

$$F(x, y) \quad = \quad (x.\phi_1(y) \oplus h_1(y), \ldots, x.\phi_m(y) \oplus h_m(y))$$

The construction proposed in [73] is built out of Linear Feedback Shift Registers (LFSRs) and is based on the Maiorana-McFarland construction technique. The construction may be briefly stated as follows: Let f_i, $i = 1, 2, \ldots, m$ be Maiorana functions and so has the form, $f_i(x) = f_i(x_1, x_2) = \pi_i(x_1).x_2 + g_i(x_1)$, where π_i is a permutation of the space $Z_p^{n/2}$ and g_i is a function from $Z_p^{n/2}$ to Z_p. Then $f = (f_1, f_2, \ldots, f_m)$ is perfect nonlinear if every nonzero linear combination of the permutation π_i, $i = 1, 2, \ldots, m$ is again a permutation of $Z_p^{n/2}$. In the above method $n \geq 2m$.

One way of constructing was presented through LFSRs, thus making the implementation efficient. The length of the LFSR was $n/2$ with a primitive feedback polynomial. Then A as well as its power represent permutations in $Z_p^{n/2}$.

Also the LFSRs with primitive feedback polynomials generates maximal length sequences. The principle to generate a perfect nonlinear S-Box with n input variables and m output variables, where $n \geq m$ was therefore to take a $n/2$ shift register with a primitive feedback polynomial. The input block was divided into two equal halves of $n/2$ bits, x_1 and x_2. The first bit of the output block of length m is obtained by calculating the dot product $x_1.x_2$. To obtain the second bit the shift register is applied and the dot product of the LFSR's new output state and the x_2 is computed. In this manner every shift of the register produces a new output digit.

Another interesting method of systematically generating cryptographically robust S-Boxes was presented in [72]. The method is based on an interesting combinatorial structure called group Hadamard matrices. The authors demonstrated that their method is superior to previous approaches and generates promising S-Boxes in terms of defiance against Linear and Differential cryptanalysis, SAC fulfilling properties, high nonlinearity and algebraic degrees. In [75] authors show that strong S-Boxes may be generated using two basic techniques:

1. Concatenating Bent Functions

2. Concatenating Linear Functions

However it was shown in [76] that all the proposed constructions for good candidates for S-Boxes are based on, what is known as the *Maiorana-McFarland Construction.* All the primary constructions presented in [77, 78, 79, 80] are based on this principle.

While the constructions of strong S-Boxes is an important aspect of cryptographic design, the other important aspect is the design, analysis and implementation of large strong S-Boxes. Although many desirable properties of S-Boxes have been studied, it is also a challenge to develop efficient scalable architectures for S-Boxes. For example the design techniques presented in [81] to realize $n \times n$ S-Boxes does not scale properly with the value n. In [82] a practical S-Box design has been described where it has been stated that the construction of large cryptographically strong S-Boxes are difficult and requires huge computational resources. The paper suggests that the creation of large cryptographically good S-Boxes with bent functions as ingredients have proven more difficult that originally expected. The algorithm proposed and the supporting experimentations showed that it took between 15 and 30 days on a Pentium 90 for an 8×32 S-Box with good cryptographic properties.

In [83] a method has been described for obtaining cryptographically strong 8×8 S-Boxes. However the performance of the generated S-Boxes are much inferior compared to possible S-Boxes that can be constructed for such dimensions. Recently an interesting construction of S-Boxes based on the Maiorana-McFarland methodology has been proposed in [84]. The software implementation of the proposed method has been presented in [85] where the authors claim that the work was the first practical software implementation of a general framework to generate cryptographically robust $n \times m$ S-Boxes. However, the construction requires table lookup of the order 2^m, which makes it infeasible for a hardware implementation where m is large. Indeed we require bijective S-Boxes where $m = n$ in block ciphers and the current algebraic attacks [86, 87] obviates the requirement that the size of the S-Boxes are equal to or more than eight. Hence the design of scalable architectures for cryptographically robust S-Boxes is an important future area of research.

2.2.4 Design of D-Boxes: The Diffusion/Permutation Layer

The other important component of an SPN block cipher is the permutation layer or D-Box. The purpose of the diffusion layer is to provide avalanche effect. It implies that there should be no correlation between linear combinations of the input and the output. It also implies that small input changes should cause large output changes. Also to produce a small output change a large input change is necessary. This effect is quantized with the parameter *branch number* β. Let $w_h(a)$ denote the Hamming

weight of a, i.e., the number of non-zero components of a. Then the branch number of the linear transformation θ, is

$$\beta(\theta) \quad = \quad min_{a \neq 0}(w_h(a) + w_h(\theta(a)))$$

Here β gives a measure of the worst case diffusion and is a lower bound of number of operational S-Boxes, more formally known as active S-Boxes. Hey and Tavares [88, 89, 90] showed that when the permutation layer of an SPN cipher is a diffusive linear transformation it improves the avalanche characteristics of a cipher and improves the security margin of the block cipher against Linear and Differential Cryptanalysis [30, 29]. Informally avalanche characteristics implies that for a good cipher a change in one bit of the input should change half of the output bits. It was shown that with the help of such linear transformations one can find upper bounds for differential characteristic probability [29] and also probabilities for linear approximations [30] as a function of number of rounds of the cipher. An interesting class of linear transformations is the Maximum Distance Separable (MDS) codes [91]. The use of such linear transformations was first proposed by Vaudenay in [92] and then utilized in the cipher *Shark* [93] and the cipher *Square* [94]. This class of linear transformations has the advantage that the number of S-Boxes involved in any two rounds of the linear transformation is maximum theoretically possible. For a linear (n, k, d) code over any field, $d \leq n-k+1$. Codes with $d = n-k+1$ are called Maximum Distance Separable Codes or MDS codes for short [91]. The MDS codes are described by the following result[91]: An (n, k, d) code with generator matrix $G = [I|A]$, where A is a $k \times (n-k)$ matrix, is MDS if and only if every square submatrix (formed from any i rows and i columns, for any $i = 1, 2, \ldots, min\{k, n-k\}$) of A is nonsingular.

In [95] two construction methods for involutional linear transformations based on MDS (Maximum Distance Separable) Codes were provided. The first method is a random construction. They proposed that the $m \times m$ matrix

$$A = \begin{pmatrix} A_{11} & A_{11}^{-1} \\ A_{11}^3 \oplus A_{11} & A_{11} \end{pmatrix}$$

where A_{11} is a random $m/2 \times m/2$ matrix, is an involution over $GF(2^n)$. For $n = 8$ a random search for a matrix A with the above structure and which has all the square submatrix nonsingular takes a few seconds.

The other method is based on an algebraic construction which is described by the following result:

Lemma 2.1 *Given* $x_0, \ldots, x_{n-1}$ *and* $y_1, \ldots, y_{n-1}$, *the matrix* $A = [a_{ij}]$, $0 \leq i, j \leq n-1$ *where* $a_{ij} = \frac{1}{x_i + y_j}$ *is called a Cauchy matrix. It is known that:*

$$det(A) = \frac{\prod_{0 \leq i < j \leq n-1}(x_j - x_i)(y_j - y_i)}{\prod_{0 \leq i < j \leq n-1}(x_i + y_j)}$$

Hence provided the x_i and y_i are distinct and $x_i + y_j \neq 0$ for all i, j it follows that any square submatrix of the Cauchy matrix is nonsingular. There are various other interesting works in literature

Literature provides various other construction techniques to generate linear transformations which may be efficiently implemented for block ciphers [96]. In the search of optimized MDS mappings the use of Hadamard matrices are also used [97, 96]. Given k elements $\alpha_0, \ldots, \alpha_{k-1}$, a Hadamard matrix A is constructed with each entry $A_{i,j} = \alpha_{i \oplus j}$. Each Hadamard matrix A over a finite field has the following properties: $A^2 = \gamma I$, where γ is a constant and I is the identity matrix. When γ is 1, the mapping is involutional. The circulant nature of the Hadamard matrices help to reduce the search space in which the MDS mappings are searched.

2.3 Design of Key Establishment Protocols

The underlying principle behind the design of secured systems is that secrecy must lie entirely in the key. The principle is known as the Kerckhoff's principle and was first postulated by a nineteenth century cryptographer named A. Kerckhoffs [9].

Hence establishing the key between the sender and receiver in a two party scenario and among all the legal players in a multiple party scenario is a challenging task. In this work we concentrate on two party key establishment. Such a multi-party algorithm is known as *protocol*, defined as a sequence of steps precisely specifying the actions required of two or more parties in order to achieve a specified objective.

Technically key establishment is defined as follows[98]:

Definition 2.2 Key Establishment *is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use.*

There are two broad categories of key establishment protocols:

- *Key Transport protocol* is a process or protocol whereby a shared secret becomes available to or more parties, for subsequent cryptographic use.

- *Key Agreement protocol* is a key establishment technique where one party creates or otherwise obtains a secret value, and secretly transfers it to the other parties.

Key establishment can also be categorised depending upon the manner in which the key is generated. They are known as:

- **Key pre-distribution** is a method to establish the key whereby the resulting established keys are completely determined a priori by initial keying material.

- **Dynamic key establishment** is a method in which the key established by a fixed pair (or group) of users varies on subsequent executions of the protocols. This makes the protocols immune to known-key attacks.

Further the key establishment protocols may be categorised into two types: (i) Trusted server based and (ii) without trusted servers.

In the first type of protocol there is a centralized or trusted party. This party is referred to by a variety of names: *trusted third party, trusted server, authentication server, key distribution center (KDC), key translation center (KTC) and certification authority (CA)*. Example of such type of protocols are [99, 100, 101]. The other type of key establishment protocols are those which do not depend on a trusted server, but rather depend on an apriori shared secret[102, 103].

However there are certain central goals which a secured protocol must satisfy. We enlist them in the following for a two party key establishment protocol. We assume that the protocol is played by two players, Alice (A) and Bob(B). The aims of the protocol are[98]:

1. At the end of the protocol the value of K_{AB} should be known to both A and B, but no other parties (except a trusted central server S if there is any).

2. A and B should know that K_{AB} is newly generated.

The eaves-dropper, Eve (E) observes the messages exchanged between A and B. The attacker E and the environment is modelled by certain assumptions and the protocol is proved to be secured in such a model. The strength of such a proof however depends on the practicality of such assumptions.

The basic assumptions are:

1. The adversary is able to eavesdrop on all messages sent in a cryptographic protocol.

2. The adversary is able to alter all messages sent in a cryptographic protocol using any information available. In addition the adversary can re-route any message to any other principal. This includes the ability to generate and insert completely new messages.

3. The adversary may be a legitimate protocol participant (an insider) or an external party (an outsider) or a combination of both.

4. An adversary is able to obtain the value of the session key K_{AB} used in any sufficiently odd previous run of the protocol.

5. The adversary may start any number of parallel protocol runs between any principals including different runs involving the same principals and with same principals taking the same or different protocol roles.

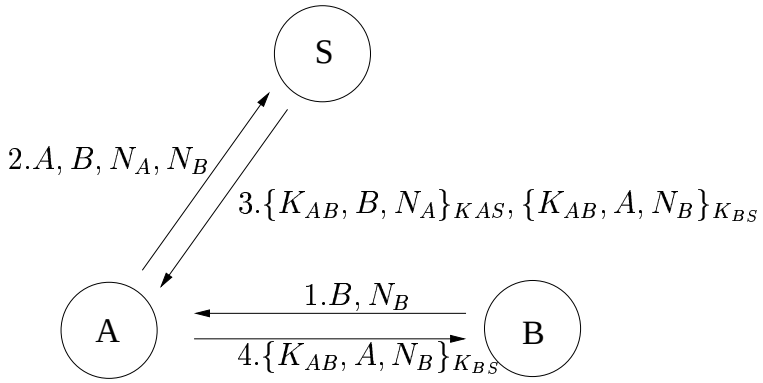


Figure 2.2: **A Key Establishment Protocol**

Fig. 2.2 provides an example protocol[98] among three entities A, B and a trusted server S. The aim of the protocol is to provide A and B with an established key K_{AB} at the end of the protocol, which they can use subsequently for communication. Any other party except the trusted server (i.e the adversary) should not be able to know the value of the key. Note that in the protocol the message format $\{M\}_K$ is used to indicate that the message M is encrypted with the key K. As a matter of general principle it is not possible to establish an authenticated session key without existing secure channels already being available[104]. The principle can be stated formally and proved. Therefore for the secure establishment of keys it is essential that keys are already shared between the different principles or certified public keys are available. In the protocol depicted in **Fig. 2.2** the keys K_{AS} and K_{BS} are used to indicate the shared keys between A and S and between B and S respectively.

A cryptographic protocol is made of various components, each of which has got their separate functionalities. For example in the above protocol, the encryption

function is used to provide security against the adversary under the first security assumption. In order to safe-guard against the second assumption, the messages transfered should be authenticated so that the attacker does not alter the messages. Hence the identifier (A and B) are included in the encrypted messages. Since the attacker does not have the keys K_{AS} and K_{BS}, he cannot alter the messages. The property is again based on the assumption that the encryption function is a strong encryption algorithm and satisfies a property known as *non-malleability*. The *identity* of the senders of the messages should be included to take care of the third assumption.

The session key is ideally changed in each session. This reduces the chance of cryptanalysis, as it provides less cryptogram which has been encrypted with the same key to the attacker. Also the key might be leaked due to various reasons, like insecure storage, betrayal of insiders and negligence on one of the parties involved. In short it should not be possible for any one to establish forcefully an old key by replaying old messages. For this the technique used, as shown in the messages of the depicted protocol, is called the *challenge-response* technique using *nonces*.

Definition 2.3 *A nonce is a random value generated by one party and returned to that party to show that the message is newly generated.*

In the protocol shown the values N_A and N_B are known as nonces.

Cryptographic protocols are built out of various cryptographic functions. There are essentially four kinds of cryptographic functions or primitives, as they are often called, being used in the operation of the protocols.

- **Encryption Function:** They include both private and public key encryption mechanisms, in order to provide confidentiality to the input message. Public Key encryption is denoted by the symbol $E_A(M)$, for the encryption of message M with the public key A. Similarly, $\{M\}_K$ denotes symmetric encryption of message M with shared key K.

  A strong encryption provides security to the message against chosen plaintext and chosen ciphertext attacks.

- **Message authentication code (MAC):** They are a family of functions parameterised by a key K such that $MAC_K(M)$ takes a message of arbitrary length and outputs a fixed length value. The transformation satisfies the following two conditions:

  1. It is computationally easy to calculate $MAC_K(m)$ given K and m.
  2. Given any number of MAC values for a given K, it is computationally hard to find any valid MAC value for any new message, without the knowledge of K.

The MAC function is used to provide authentication and data integrity by appending a MAC to a message which may be either in clear or encrypted. On receipt of the MAC, only the recipient who has the correct key can recompute the MAC and verify that it is the same as that received.

- **Digital Signature:** These class of algorithms consist of three set of components, a key set K, a message set M, and a signature set S together with the following three algorithms:

  1. A key generation algorithm, which outputs a valid signature key $k \in K$ and a valid verification key $k^{-1} \in K$.
  2. A signature generation algorithm, which takes an element $m \in M$ and a signature key $k \in K$ and outputs an element $s \in S$, the signature of the message m using the signature key.
  3. A verification function, which takes a signature $s \in S$, a message $m \in M$, and a verification key $k^{-1} \in K$ and outputs an element $v \in \{0, 1\}$. If $v = 1$ then we say the signature is valid or if $v = 0$ then the signature is invalid.

  Digital signature provides *non-repudiation*. Although it is not a typical property of authentication and key establishment protocols, they are sometimes necessary for some kind of protocols.

- **One-way Functions:** A function $f : X \rightarrow Y$ is one-way if it is computationally easy to calculate $y = f(x)$ given $x \in X$, but computationally difficult to find any x with $f(x) = y$ for almost all values of $y \in Y$.

  The one-way functions find several uses in protocol designs. They are used as MAC (message authentication codes) [105]. They are also used as MDC (manipulation detection codes). In such cases without the knowledge of the message the adversary is unable to alter the message without destroying the correctness of the MDC.

  Apart from these, one-way functions provide key freshness to the agreed key. It means that for the key establishment protocols, each user of the key is able to verify that it is new and not replayed from an old session.

  For example, let two users A and B both choose inputs N_A and N_B to derive a new session key. Let the form of the function be, $K_{AB} = f(N_A, N_B)$.

  A desirable property of the function f is that it should not be possible for A or B to force an old value of K_{AB} even if the other's input is known. This means that each user has independent assurance that the session key is new or fresh. Thus, if B knows $A's$ contribution N_A, he should be able to compute his share

N_B so that an old key K_{AB}^{old} is generated. Mathematically he should not be able to solve, N_B in the equation:

$$\begin{aligned} K_{AB}^{old} &= f(N_A, N_B) \\ &= g(N_B) \end{aligned}$$

The above condition implies that the function g must be a one-way function.

2.4 Key Agreement Protocols and their Security Analysis

The majority of the key establishment protocols deal with the scenario where there are exactly two players, A and B. This is commonly known as the *two-party* case. The key establishment protocols depend on symmetric key or public key cryptography. As mentioned, the protocols use various cryptographic algorithms or primitives in order to achieve data integrity, authentication, non-repudiation and other features. More specifically in this thesis we shall be dealing with a two party key agreement protocol which also provides authentication and key confirmation.

The definition of key agreement protocol is first stated[106, 107].

Definition 2.4 Key Agreement Protocol: *A key agreement protocol or mechanism is a key establishment technique in which a shared secret is derived by two (or more) parties as a function of information contributed by, or associated with, each of these (ideally) such that no party can predetermine the resulting value.*

The general format of such protocols requires each principal to select an independent input to the key. In the two party scenario between A and B let the contributions from A and B be r_A and r_B respectively. The players (A and B) then send each other messages constituted of their contributions (r_A or r_B) along with other messages (associated with their identities etc.). The session key is then derived usually in two stages:

1. The random inputs r_A, r_B and the long term keys are combined to form a shared secret, Z_{AB}.

2. The shared secret and possibly other inputs are transformed by a key derivation function, which is typically a one-way hash function.

Table 2.1: **Meet in the middle attack on the basic D-H protocol**

A		C		B
$r_A \in_R Z_q$				
$t_A = g^{r_A}$	$--t_A-->$	$r_C \in_R Z_q$		
		$t_C = g^{r_C}$	$--t_C-->$	$r_B \in_R Z_q$
	$<--t_C--$	$Z_{AC} = t_A^{r_C}$	$<--t_B--$	$t_B = g^{r_B}$
$Z_{AB} = t_C^{r_A}$		$Z_{CB} = t_B^{r_C}$		$Z_{AB} = t_C^{r_B}$

Recently the goal of such protocols has become more difficult. It is commonly known as the *authenticated key agreement (AK) problem.* In the AK problem A merely desires that only B can possibly compute the key, and not that B has actually computed the key. If A wants to make sure that B really has computed the agreed key, then key confirmation is incorporated into the key agreement protocol. The resulting goal is then called *authenticated key agreement with key confirmation (AKC)*[108].

The most celebrated key agreement protocol is the Diffie-Hellman (D-H) key exchange[11]. In the basic D-H protocol, two principals A and B agree publicly on an element g that generates a multiplicative group G. They select random variables r_A and r_B respectively in the range between 1 and the order of G. A computes $t_A = g^{r_A}$ and B calculates $t_B = g^{r_B}$. They subsequently exchange the values and both of them derive the shared secret $Z_{AB} = g^{r_A r_B}$.

This value can be calculated by A (or B) using the private information r_A (r_B) and the public information received g^{r_B} (g^{r_A}). The security of the D-H protocol against passive eaves-dropping, lies on the assumption that it is infeasible to compute the value of Z_{AB} from the values of g^{r_A} and g^{r_B}. This is known as the *D-H assumption.*

However the $D-H$ protocol provides no authentication and can break in the face of *meet-in-the-middle attack* (**table 2.1**). Adversary C masquerades as B to A and as A to B. Both A and B complete a normal run of the protocol, but both share keys with C, namely $g^{r_A r_C}$ and $g^{r_C r_B}$.

Several key agreement protocols have been developed subsequently which are based on the D-H protocol. The protocols based on the D-H assumption have the following advantages:

- The protocols may be generalized to work in any abelian group.
- The protocols based on D-H provide *forward secrecy*, which means that disclosure of the long-term secret key does not compromise on the security of exchanged keys from earlier runs.

However there are some protocols which are not dependent on the D-H assumption.

The advantage of the non D-H based protocols are:

- There can be computational savings over D-H by reducing the number of expensive operations
- Keys can be guaranteed to be random even if one of the inputs become known, a property which is lacking in the D-H based protocols

An example of a key agreement protocol, widely used for the internet, is the SKEME (Secure Key Exchange Mechanism for Internet) protocol[109]. It supports both D-H based and non D-H based key derivation. It is a compact protocol that supports a variety of realistic scenarios and security models over Internet. It provides clear tradeoffs between security and performance as required by the different scenarios without incurring in unnecessary system complexity. The SKEME protocol brings to the surface various requirements of a key agreement protocol:

1. Periodic Key Refreshment: It should be possible to update the session key used periodically at low computational cost.
2. Forward Secrecy: This mode though sometimes essential may be occasionally sacrificed at the cost of efficiency.
3. Key Separation: Different cryptographic functions should use different keys, so that the compromise of one key does not lead to the compromise of the others.
4. Privacy and anonymity: Along with confidentiality of the data, it may be necessary to hide the identity of the communicating parties. One of the goals of the protocol is to hide the identity of the attackers.
5. Efficiency and Simplicity: Key exchange operations may strongly vary on their computational requirements. Protocol design involves delicate tradeoffs between security and performance.
6. Algorithm independence: The protocol needs to define the underlying cryptographic primitives in a functional level, but not to depend on particular implementations.

As large number of cryptographic protocols for key agreement have been proposed, several attacks (**table 2.2**) against them have also been developed. Abadi and Needham have proposed a set of principles intended to provide rules of thumb for designers of protocols[110]. They were derived from the common errors of protocols and by following them the chances of attacks against protocols reduce. However

Table 2.2: **Some common attacks against cryptographic protocols**

Eavesdropping	The adversary captures the message transferred
Modification	Adversary alters the messages transferred
Replay	Adversary records a message and then transfers to a different party, possibly during a later instance
Preplay	Adversary engages in a run of the protocol prior to a run by the legitimate principals
Reflection	Adversary sends back messages to the sender
Denial of Service	Adversary prevents or hinders legitimate principals from completing the protocol
Typing Attack	Adversary replaces a protocol message with another type of message

there can be no guarantee regarding the fact that after following the principles the protocols cannot be attacked. This emphasises the need for *provable security* of the cryptographic protocols.

The approach uses a mathematical model that defines a protocol in which a powerful adversary plays a central role. The adversary essentially controls all the principals and can initiate protocol runs between any principals at any time. Insider attacks are modelled by allowing the adversary to corrupt any principals and the adversary can also obtain previous keys. Cryptographic algorithms are also modelled as specific transformations. For example many complexity-theoretic proves rely on a technique known as *random oracles*[111]. This model assumes that a hash function works in an idealised fashion, it returns true random values each time it is used. However if the input is the same then it returns the same output. Security of protocols is defined in terms of matching conversations (for authentication) and indistinguishability (for confidentiality of keys).

The treatment of computational complexity analysis for key establishment protocols was made popular by Bellare and Rogaway[112]. They provided the first formal model of adversary capabilities with an associated definition of security. Recently a related model has been developed in [113], which introduces modular proof models. However some drawbacks were subsequently found and modified in[114]. Fundamentally all the proving techniques have two basic requirements for security. First, two parties who have completely matching sessions (or partners) are required to accept the same session key. Secondly, no adversary or anyone other than the legitimate parties involved will learn about the session key at the end of the protocol. However the Bellare Rogaway model is one of the most popular models for proving the

security of key agreement protocols [108, 98] and it is always desirable to prove a new cryptographic protocol in this model. The model has been described in details in **section 8.7** where a proposed protocol has also been proved to be secure in the model.

2.5 Design of efficient one-way functions

Several attempts to develop practical and secure symmetric-key algorithms and key agreement protocols have been published in literature. However the current techniques to develop symmetric-key algorithms do not prove that the problem of breaking the cryptosystem is as hard as that of solving any hard theoretic problem[115]. Likewise, although formal proofs reduce the problem of attacking a protocol into well known problems, like the $D-H$ assumptions, the number of such problems are limited in literature. Further, considering the demand of increased traffic in on-line communication the need for such hard problems have increased. An ideal candidate for such type of functions are known as *one-way functions*. Informally, one-way functions are class of mathematical operations in which the forward transformation is computationally efficient but the inverse is not. Classical examples of such kinds of functions which have been employed for building cryptosystems are enlisted below[106]:

1. *Integer Factoring Problem:* Given a positive integer n, find its prime factors.

2. *RSA Problem:* Given a positive integer n that is a product of two distinct odd primes p and q, a positive integer e such that $gcd(e, (p-1)(q-1)) = 1$, and an integer c find an integer m such that $m^e = c(\text{mod } n)$.

3. *Discrete Logarithm Problem (DLP):* Given a prime p, a generator α of Z_p^* (multiplicative group), and an element $\beta \in Z_p^*$, find the integer x, $0 \leq x \leq (p-2)$, such that $\alpha^x = \beta(\text{mod } p)$. The *Generalized Discrete Logarithm Problem (GDLP)* is that given a finite cyclic group G of order n, a generator α of G, and an element $\beta \in G$, find an integer x, $0 \leq x \leq (n-1)$, such that $\alpha^x = \beta$.

4. *Diffie Hellman Problem (DHP):* Given a prime p, a generator α of Z_p^*, and the elements $\alpha^a \text{mod } p$ and $\alpha^b \text{mod } p$ find $\alpha^{ab} \text{mod } p$.

5. *Subset Sum Problem:* Given a set of positive integers $\{a_1, a_2, \ldots, a_n\}$ and a positive integer s, determine whether or not there is a subset of the $a_j's$ that sums to s.

However the majority of the cryptographic algorithms based on the first three four number theoretic problems can be shown to be dependent on solving the DLP.

However the cryptographic algorithms based on such type of problems use modular exponentiation and are computationally expensive. On the other hand cryptographic algorithms developed on the Subset Sum Problem, like the famous Merkle-Hellman Knapsack[116] are computationally efficient. Hence at first there was lot of optimism regarding the Knapsack cryptosystem. The reasons were not only due to its efficiency, but also the fact that its security seemed to be guaranteed due to the NP-completeness of the Subset Sum problem. However the system was actually shown to be based on a subproblem of the the Subset Sum problem and was hence cryptanalyzed by Shamir[117]. Soon after Brickell [118] and others showed how to undermine the security of variants and generalizations of the Merkle-Hellman technique. The dramatic break of most knapsack based crypto algorithms, made cryptosystems based on *combinatorics* less popular.

An additional possible reason for this was due to the theorem of Bassard[119]. It states that the cracking problem for a cryptosystem based on one-way function cannot be NP-hard unless NP=coNP. However as the problems based on combinatorics tend to either have a polynomial time algorithm or have NP-hard complexity, it was felt that one-way functions (or hard problems) based on combinatorics were probably of no use for cryptography. Such an interpretation was probably premature as there are several problems in combinatorics which are hybrid combinatorial/algebraic problems and hence does not satisfy the theorem proposed in [119]. The fact that such type of one-way functions can lead to efficient algorithms have motivated researchers to continue their search. Literature shows that two types of one-way functions based on combinatorics have been recently studied. The first one is the *T-functions* proposed by Alexander Klimov and Adi Shamir[120], while the later one is based on a special type of graphs, known as expander graphs and is proposed by Goldreich[115].

A T-function is a mapping from m n-bit words to k n-bit words in which each bit i of any of the outputs (for $0 \leq i < n$) can depend only on bits $0, 1, \ldots, i$ of the inputs. These class of functions have been applied to develop various cryptographic primitives, like stream ciphers, MDS mappings, invertible mappings and also one-way functions[121, 122, 123, 120]. The work proposed in [120] shows efficient techniques to find T-functions which are invertible.

Let $f(x)$ and $g(x)$ be arbitrary permutations over n-bit words. Although it seems that it is difficult to construct "random looking" $f(x)$ and $g(x)$ such that $f(x)$, $g(x)$ and $f(x) \oplus g(x)$ are all permutations, using the techniques proposed in [120] one can construct such T-functions. As a special case if the function $g(x)$ is an identity, then the resultant function is $x \oplus f(x)$. It is known from [124], that if $f(x)$ is a random function then $f(x) + x$ is hard to invert. However if such a proposition also applies for a T-function was to be inspected.

The one-way function proposed in [115] is a function (rather a collection of functions), f_n, that maps strings in $\{0,1\}^n$ to $\{0,1\}^n$. It uses as parameters two sets of

values:

- A collection (of size n) of small overlapping subsets of $[n]$, $C = \{S_i : |S_i| = d; S_i \in [n]; i \in [n]\}$ (typically d is chosen to be of the order of $log(n)$)
- A predicate $\Pi : \{0,1\}^d \rightarrow \{0,1\}$

For every string, $x = x_1 x_2 \ldots x_n$, in $\{0,1\}^n$, computing $f_n(x)$ involves first projecting x onto each of the subsets on the collection (if $S_i = \{i_1, i_2, \ldots, i_d\}$ then the projection of x on S_i, denoted by x_{Si} is a string of length d which is given by $x_{i1} x_{i2} \ldots x_{id}$) and then evaluating the predicate Π on each of the n projections, thus giving us the n bit values which represent the output string. In other words, $f_n(x)$ is the bit string in $\{0,1\}^n$ equal to:

$$\Pi(x_{S1})\Pi(x_{S2}) \ldots \Pi(x_{Sn})$$

It was shown that although the function in the forward direction is quite simple, inverting the function is apparently a hard problem if the collection C has some desirable combinatorial properties. Specifically, it was shown that if C is an expanding collection i.e for some k, every subset of C containing k subsets of $[n]$ be such that the union of these k subsets is of the form $k + \theta(n)$, then the problem of inverting the function does not seem to have an efficient solution.

The function was experimented with a special type of graphs, known as the expander graphs. Various types of expander graphs were experimented upon, like random, Alon's Geometric Expanders[125], Ramanujan Expanders of Lubotzky, Phillips and Sarnak[126], Gaber-Galil's [127] expander.

The security analysis of the one-way function show that the inverting algorithm's time complexity was exponential in the expansion rate of the graph[115, 128]. It was also shown that the performance of inverting algorithms deteriorated with the increase in the degree d of the graph. It was clearly pointed out in [128] that the security of the one-way function increases with the increase in the value of the parameter d. However the implementation of such a scheme becomes a worry. Thus construction of good expander graphs which has large values of d and yet be easily implemented shall be interesting and conducive for the practical realization of such one-way functions.

2.6 Conclusions: Where are we heading to?

Cryptography has evolved with time. According to Bruce Schneier many of the attacks like linear, boomerang and slide are more often mathematical arguments rather

than computer demonstrations. Thus no AES traffic can be decrypted using the present techniques. The security margin in the ciphers are sufficient to make these attacks irrelevant. Thus it is an open question how long will the tools remain theoretical. It is the diversity of the application areas which forces the modern cryptographer to design different algorithms, rather than its security concerns. At present it is tough to design ciphers which are fast and secure at the same time. This is because most ciphers are secure after sufficiently many rounds. But then the ciphers become slow. This makes the development of security algorithms under conflicting requirements of high throughput, low area and less power a daunting task.

In this work we deal with the design and analysis of various cryptographic algorithms. We search for cryptographic primitives, which are amenable to efficient implementations and still provide sufficient security margin. In this endeavour we have used a special type of machine, known as the *Cellular Automata*. The basic mathematical background of this class of machine has been dealt with in the next chapter.

Chapter 3

Mathematical Background

The thesis deals with the development of efficient cryptographic algorithms suited specifically for hardware. The primary tool used in this effort is the *Cellular Automata* (CA). A detailed treatment on the theory of CA may be found in [129]. The current chapter presents a preliminary overview on the theory of CA. Finally, we also detail the cryptographic properties of robust S-Boxes which are used in the cryptanalysis of block ciphers.

3.1 Preliminaries on Cellular Automata

Stephen Wolfram [130] pioneered the investigation of Cellular Automata (CA) as mathematical models for self-organizing statistical systems. The structure of CA can be investigated as a discrete lattice of cells where each cell has a storage element (DFF) and a combinational logic (CL) (**Fig. 3.1**).

The DFFs can assume either the value 0 or 1. The next state of a cell **(Fig. 3.2)** is assumed to depend on itself and on its two neighbours (left or right) and this is known as three neighbourhood dependency. The logic of the CL block determines the *rule* of the CA. The state of the CA at any instant of time is then denoted by the concatenation of the values held by the individual DFFs at that instant. The CA evolves in discrete time steps according to the rule of the automaton which in turn depends on the logic of the CL blocks. In our construction we have used a null boundary CA, which means that the left (right) neighbour of the leftmost (rightmost) terminal cell is connected to logic 0 state. If the CA has n cells then the state at any instant may be represented as X_t, where $X_t = \{q_0(t), q_1(t), \ldots, q_{n-1}(t)\}$. Here, $q_i(t)$ denotes the state of the i^{th} cell at the t^{th} instant of time. Clearly the state of the i^{th} cell at the $(t+1)^{th}$ instant of time is denoted by $q_i(t+1)$ and can be expressed

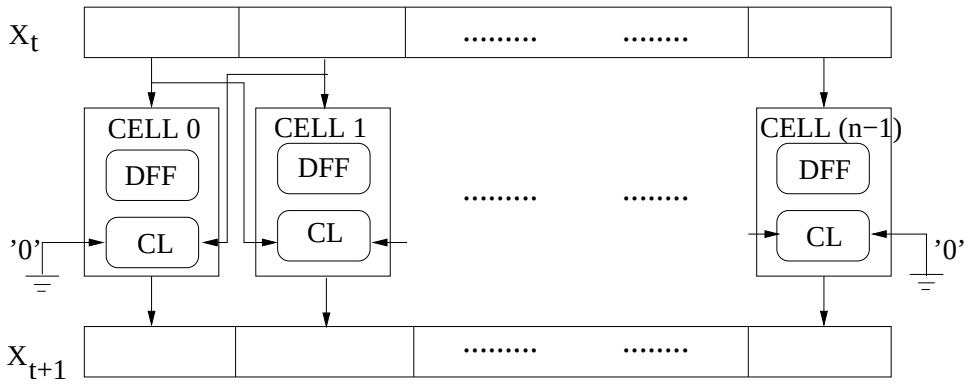


Figure 3.1: **The CA Structure**

as: $q_i(t+1) = f[q_{i-1}(t), q_i(t), q_{i+1}(t)]$, where f is the local transition function realized by the CL[129]. Each combinational logic thus represents a function f, which is inherently a boolean function and can be expressed in the form of a truth table. The decimal equivalent of the output is conventionally called the rule number for the cell.

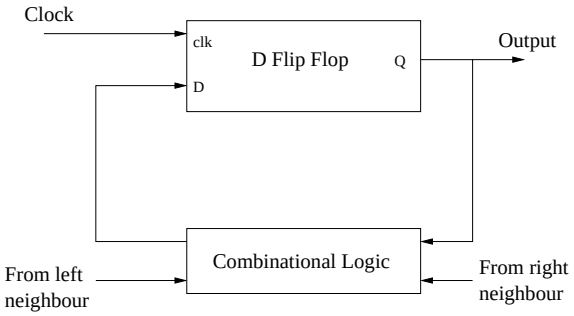


Figure 3.2: **CA cell**

Definition 3.1 Rule of a CA: *If the next state function of a cell is expressed in the form of a truth table, then the decimal equivalent of the output is called rule number of the Cellular Automaton.*

Example 3.1 *As an example rules* 90 *and* 150 *are described in* **table 3.1**.

Table 3.1 *shows the next states computed according to rules* 90 *and* 150. *The top row shows all the possible configurations of the left, self and right cells at instant* t. *The states at the instant of time* $(t+1)$ *are computed according to the rules.*

Table 3.1: **Rules** 90 **and** 150

Neighbourhood State:	111	110	101	100	011	010	001	000	
Next State:	0	1	0	1	1	0	1	0	(rule 90)
Next State:	1	0	0	1	0	1	1	0	(rule 150)

From the above table it is evident that the combinational logic for rule 90 *and* 150 *are respectively:*

$$\textit{rule 90:} \quad q_i(t+1) = \quad q_{i-1}(t) \oplus q_{i+1}(t)$$
$$\textit{rule 150:} \quad q_i(t+1) = \quad q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$$

On minimization, the truth tables for the rules $15, 51, 60, 85, 90, 102, 105, 150, 165, 170, 195, 204$ and 240 result in the logic functions noted in the **table 3.2**, where $q_i(t)$ denotes the state of the i^{th} CA cell at the t^{th} time instant, q_{i-1} and q_{i+1} refers to the state of its left and right neighbours.

Table 3.2: **Additive CA rules**

Complement		Dependency	Non-complement	
Rule	Logic function		Rule	Logic function
195	$\overline{q_{i-1}(t) \oplus q_i(t)}$	left & self	60	$q_{i-1}(t) \oplus q_i(t)$
165	$\overline{q_{i-1}(t) \oplus q_{i+1}(t)}$	left & right	90	$q_{i-1}(t) \oplus q_{i+1}(t)$
153	$\overline{q_i(t) \oplus q_{i+1}(t)}$	self & right	102	$q_i(t) \oplus q_{i+1}(t)$
105	$\overline{q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)}$	left & self & right	150	$q_{i-1}(t) \oplus q_i(t) \oplus q_{i+1}(t)$
85	$\overline{q_{i+1}(t)}$	right	170	$q_{i+1}(t)$
51	$\overline{q_i(t)}$	self	204	$q_i(t)$
15	$\overline{q_{i-1}(t)}$	left	240	$q_{i-1}(t)$

The following definitions of the theory of CA have been used in the discussion on the work.

Definition 3.2 Additive and Non-additive CA: *If the rule of a CA cell involves only XOR logic, then it is called a linear rule. A CA with all the cells having linear rules is called a linear CA. Rules involving XNOR logic are referred to as complemented rules. A CA having a combination of XOR and XNOR rules is called an additive CA. The rules with AND-OR logic are non-additive rules.*

Definition 3.3 *If all the CA cells obey the same rule then the CA is said to be a uniform CA, otherwise it is a hybrid CA.*

Definition 3.4 Null Boundary CA: *A CA is said to be a Null Boundary CA if the left (right) neighbour of the leftmost (rightmost) terminal cell is connected to logic* 0 *state.*

Definition 3.5 Periodic Boundary CA: *A CA is said to be a Periodic Boundary CA if the extreme cells are adjacent to each other.*

Definition 3.6 Characteristic Matrix of a CA: *An n-cell CA is characterized by an $n \times n$ matrix operating over $GF(2)$. The characteristic matrix T is constructed as:*

$$T[i,j] = \begin{cases} 1, & \textit{if the next state of the } i^{th} \textit{ cell} \\ & \textit{depends on the present state of the } j^{th} \textit{ cell} \\ 0, & \textit{otherwise} \end{cases}$$

Example 3.2 *A four cell null boundary CA with rule vector $< 90, 150, 90, 150 >$ is characterized by the characteristic matrix,*

$$T = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The characteristic polynomial of the matrix is obtained by constructing the matrix $[T] + x[I]$ and computing the corresponding determinant.

$$[T + xI] = \begin{pmatrix} x & 1 & 0 & 0 \\ 1 & 1+x & 1 & 0 \\ 0 & 1 & x & 1 \\ 0 & 0 & 1 & 1+x \end{pmatrix}$$

The characteristic polynomial of the Cellular Automaton is $p(x) = x^4 + x + 1$.

With the definition of the characteristic matrix T, thus the state transition of a linear CA can be described by the equation:

$$Y = T(X)$$

In the above equation T is an $n \times n$ matrix to represent the characteristic matrix of an n cell CA. The input state of the CA is denoted by the n-bit vector X while Y denotes the output bit vector of the CA. The CA based structure for the characteristic matrix shown in **example 3.2** is depicted in **Fig. 3.3**.

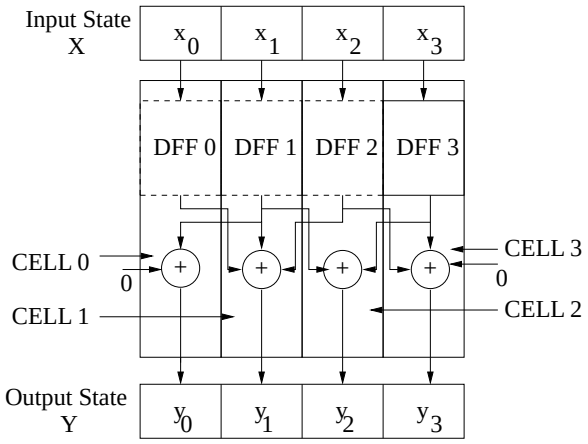


Figure 3.3: **A Cellular Automaton Based Structure**

Definition 3.7 Group CA: *If the CA under the transformation of operation with T forms a cycle and there exists an integer m such that $T^m = I$ (identity matrix) the CA is called a group CA.*

A Group CA is characterized by the following theorem:

Theorem 3.1 *A CA is a group CA iff the determinant det T=$|T|$=1, where T is the characteristic matrix of the CA.*

It is evident that the rule shown in **example 3.2** is of a group CA. In fact the CA of **example 3.2** has an additional property, that the transition matrix is of a maximum length CA.

Definition 3.8 Maximum Length CA: *An n-cell maximum length CA is characterized by the presence of a cycle of length* $(2^n - 1)$ *with all non-zero states. The characteristic polynomial of such a CA is primitive.*

The CA illustrated in **example 3.2** is that of a four cell maximum CA. This can be noted by the fact that the characteristic polynomial $p(x) = x^4 + x + 1$ is a primitive polynomial of $GF(2^4)$. Thus all the non-zero states lie in one cycle of $2^4 - 1 = 15$ elements. **Table 3.3** enlists the hybrid rules for maximum length CA for various sizes of the Automaton. In this table $'1'$ refers to rule 150 while $'0'$ means rule 90.

In contrast to the maximum length CA, when all the non-zero states does not lie in a cycle, the CA is called non-maximum length CA.

Theorem 3.2 *A group CA has cycle lengths of p or factors of p with a non-zero starting state iff* $det[T^p \oplus I] = 0$.

Proof: If there exists a cycle of length p for a state X then,

$$\begin{aligned} T^p(X) &= X \\ \Rightarrow (T^p \oplus I)(X) &= 0 \\ \Rightarrow det\ (T^p \oplus I) &= 0 \end{aligned}$$

Conversely, from the theory of matrices it is known [131] that if $det\ (T^p \oplus I) = 0$, then there exists at least one non-zero state X such that

$$\begin{aligned} (T^p \oplus I)(X) &= 0 \\ \Rightarrow T^p(X) &= X \end{aligned}$$

Hence the result. □

Lemma 3.1 *If the order* (o_G) *of the group CA characterized by T is a non-prime number, then the cycle lengths are equal to its factors only.*

Proof: Let the order of the group CA be $o_G = m$. Assume that there exists a cycle of length p such that p does not divide m. Thus, m can be expressed as $m = p * q + r$; q, r being integers and $0 < r < p$. Since, there exists a cycle of length p, there is a non-zero state X for which p is the smallest integer such that $T^p(X) = X$.

$$\begin{aligned} \text{Now, } T^m(X) &= T^{pq}.T^r(X) \\ &= T^r.T^{(q-1)p}(T^p(X)) \end{aligned}$$

Table 3.3: **Maximum length hybrid CA rules**

Number of CA cells	Rule	Cycle Length
4	0101	15
5	11001	31
6	010101	63
7	1101010	127
8	11010101	255
9	110010101	511
10	0101010101	1,023
11	11010101010	2,047
12	010101010101	4,095
13	1100101010100	8,191
14	01111101111110	16,383
15	100100010100001	32,767
16	1101010101010101	65,535
17	01111101111110011	131,071
18	010101010101010101	262,143
19	0110100110110001001	524,287
20	11110011101101111111	1,048,575
21	011110011000001111011	2,097,151
22	010101010101010101010101	4,194,303
23	11010111001110100011010	8,388,607
24	111111010010110101010110	16,777,215
25	101111010101010011110100	33,554,432
26	010110101101000101111011000	67,108,863
27	000011111000001100100001101	134,217,727
28	010101010101010101010101010101	268,435,455
29	101010010101110010100010000011	$2^{29}-1$
30	111010001001101100101000111101	$2^{30}-1$
31	010011001010110111110111001100 0	$2^{31}-1$
32	010001100000100110111011110101 01	$2^{32}-1$
33	000011000100111001110010110000101	$2^{33}-1$
34	001111000010110100001100011011 1010	$2^{34}-1$
35	010101111011110110011101010010100 11	$2^{35}-1$
36	101001100100100011111010110000100011	$2^{36}-1$
37	001001011001111010110101100001011 0011	$2^{37}-1$
38	000111001010111101100110011110000100 11	$2^{38}-1$
39	110100010111110110111100110011101101100	$2^{39}-1$
40	000011101100101010111110010000101110 0101	$2^{40}-1$
41	011010111111101000010110011000111100 00111	$2^{41}-1$
42	001001111110110011100101001001100111100 110	$2^{42}-1$
43	001110101110001011100010000101101011 0010010	$2^{43}-1$
44	001111001111011010110111000010010101 1000010	$2^{44}-1$
45	001101001011001101101001000100110001 101001101	$2^{45}-1$
46	000100101001100101000110100010110011 1011010110	$2^{46}-1$
47	001110010111111001110010101001000101 11000001101	$2^{47}-1$
48	000110000110111110010010100111010001 1110000011 11	$2^{48}-1$
49	001011011110110010001100101111100010 1110110011001	$2^{49}-1$
50	100110100110110000001100011010001011 00100010010110	$2^{50}-1$
51	000100001011101010100001011010011101 0001010000 10111	$2^{51}-1$
52	001100100011011110111011111110001000 1111010111000110	$2^{52}-1$
53	100001110010100010000010010011001011 1011111010110010101	$2^{53}-1$

$$\begin{aligned} &= T^r.T^{(q-1)p}(X) \\ &= T^r.T^{(q-2)p}(T^p.(X)) \\ &= T^r.T^{(q-2)}(X) \\ &\vdots \\ &= T^r.T^p(X) \\ &= T^r(X) \\ \Rightarrow T^m(X) &= T^r(X) \end{aligned}$$

Since, by hypothesis p is the smallest integer for a specific X, and since $r < p$, it immediately follows that such an r cannot exist. Thus p is a factor of n. □

3.1.1 Cycle set characterization of group CA

The characteristic polynomial of a linear machine like linear CA is given by the determinant of the matrix $(T + xI)$, where T is the characteristic matrix representing the global state transition function of the machine. Factors of this characteristic polynomial are called the *Factor polynomials*.

Elspas [132] has characterized the cycle sets in the state-transition diagram of linear machines by considering the factors of their characteristic polynomial. Some of the important results are noted below.

1. *Nonrepeated factors:* Let the characteristic polynomial $\phi(x)$ be factored into irreducible factors:

   $$\phi(x) = \phi_1(x)\phi_2(x)\ldots\phi_r(x)$$

   The cycle structure S_i corresponding to the factor polynomial $\phi_i(x)$ is given by 1-cycle plus μ_i cycles of length k_i, where $\mu_i = (2^{n_i} - 1)/k_i$, n_i being the degree of $\phi_i(x)$, and k_i, the smallest integer such that, $\phi_i(x)$ divides $x^{k_i} + 1$. This cycle structure is represented as $[1, \mu_i(k_i)]$.

   If $[1, \mu_1(k_1)]$ and $[1, \mu_2(k_2)]$ be the cyclic structures corresponding to two irreducible factors $\phi_1(x)$ and $\phi_2(x)$, then the cycle structure of the product $\phi_1(x)\phi_2(x)$ is given by $[1, \mu_1(k_1), \mu_2(k_2), \mu(k)]$, where $\mu = \mu_1\mu_2 gcd(k_1, k_2)$, and $k = lcm(k_1, k_2)$.

2. *Repeated factors:* If $\phi(x)$ be an irreducible polynomial with period k, then the cycle structure of a machine with minimal polynomial $[\phi(x)]^r$ is given by $[1, \mu'(k')]$, where, $k' = k2^{r_1}$, such that, $2^{r_1-1} < r < 2^{r_1}$, and $\mu' = \frac{2^{d(r-1)}(2^d-1)}{k'}$, where d is the degree of $\phi(x)$.

3. Finally, each matrix T is completely characterized by a set of *elementary divisors* [131, 133]:

$$[p_i(x)]^{e_{i1}}, [p_i(x)]^{e_{i2}}, \ldots, [p_i(x)]^{e_{ir_i}};\ i = 1,\ 2,\ \ldots\ r, \quad r \text{ being the rank of } T.$$

Each $p_i(x)$ is a distinct irreducible factor polynomial of the characteristic polynomial $\phi(x)$, and $e_{i1} \geq e_{i2} \geq \ldots \geq e_{ir_i}$. The minimal polynomial [131, 133] $m(x)$ of T, is the product of the highest degree elementary divisors $[p_i(x)]^{e_{i1}}$. Thus the cycle set can be easily computed by knowing the *elementary divisors* and applying the rules for product and repeated factors as noted in the earlier discussions.

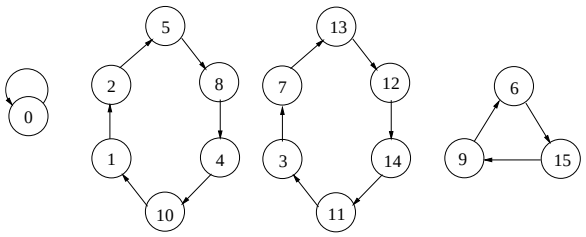


Figure 3.4: **The State Transition Diagram of a Nonmaximum Length CA**

For example, the characteristic matrix of the CA shown in **Fig 3.4** is

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

On diagonalization,

$$T + Ix = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & (x^2+x+1)^2 \end{bmatrix}$$

Hence its characteristic (as well as minimal) polynomial is $(x^2 + x + 1)^2$. Now the cycle structure corresponding to $(x^2 + x + 1)$ is [1,1(3)]. This is because k_i is equal to 3 (since the polynomial divides $x^3 + 1$) and $\mu_i = (2^2 - 1)/3 = 1$ (Case of *Nonrepeated factors*). To derive the cycle structure for the whole CA, the case of *Repeated factors* needs to be considered. Here the value of r is 2. Hence, $r_1 = 1$. Thus, $k' = 3.2^1 =$

6; and $\mu' = \frac{2^{2(2-1)}(2^2-1)}{6} = 2$. Hence, the cycle structure of the whole CA is [1, 1(3), 2(6)], that is, one cycle of length 1, one cycle of length 3, and two cycles of length 6.

Let us consider another 3 cell CA whose characteristic polynomial is given by $(x+1)(x^2+x+1)$. The cyclic structure corresponding to $(x+1)$ is [1,1(1)] since k_i is equal to 1 (since it divides $x+1$) and $\mu_i = (2^1-1)/1 = 1$. As noted earlier, the cyclic structure corresponding to (x^2+x+1) is [1,1(3)]. Therefore, the cyclic structure corresponding to $(x+1)(x^2+x+1)$ is given by [1, 1(1), 1(3)].

Characterization of non-group CA

In the state-transition graph of a group CA, every state has got a unique predecessor. Thus, all states lie on a disjoint set of cycles. There is another class of CA, in which some of the states are not reachable from any state. Moreover, the reachable states have got 2^k $(k > 0)$ predecessors. This type of cellular automata are known as *non-group cellular automata*. The characteristic matrix of such a CA is singular in nature, in contrast to a group CA for which the matrix is non-singular.

A non-group CA is characterized by the following theorem:

Theorem 3.3 *A CA is a non-group CA iff the determinant det T=$|T|$=0, where T is the characteristic matrix of the CA.*

Fig 3.5 displays the structure of a 4-cell non-group CA along with its characteristic matrix T and its state-transition behaviour. Other relevant features of the state transition behaviour are also noted in **Fig 3.5**. Another non-group CA in which the cyclic states lie on cycles of length unity only is shown in **Fig 3.6** while **Fig 3.7** displays a non-group CA with complemented rules.

The state-transition graphs of linear non-group CA, as illustrated in **Fig 3.5** and **3.6**, consist of a number of cyclic states lying on one or more cycles. Other states form inverted trees rooted at one of the cyclic states. Such inverted trees are henceforth referred to as simply trees.

Definition 3.9 *The cycles in the state-transition diagram of a non-group CA are referred to as* **attractors**. *Thus an* **attractor** *is a cycle of length $i (i \geq 1)$. A state with a self-loop is referred to as a* **graveyard state (or single cycle attractor)**. *A single cycle attractor is simply referred to as an attractor in subsequent discussions. For linear CA, the 0-state (that is all 0's binary pattern, represented as decimal '0') always forms a cycle of length 1 (that is an attractor).*

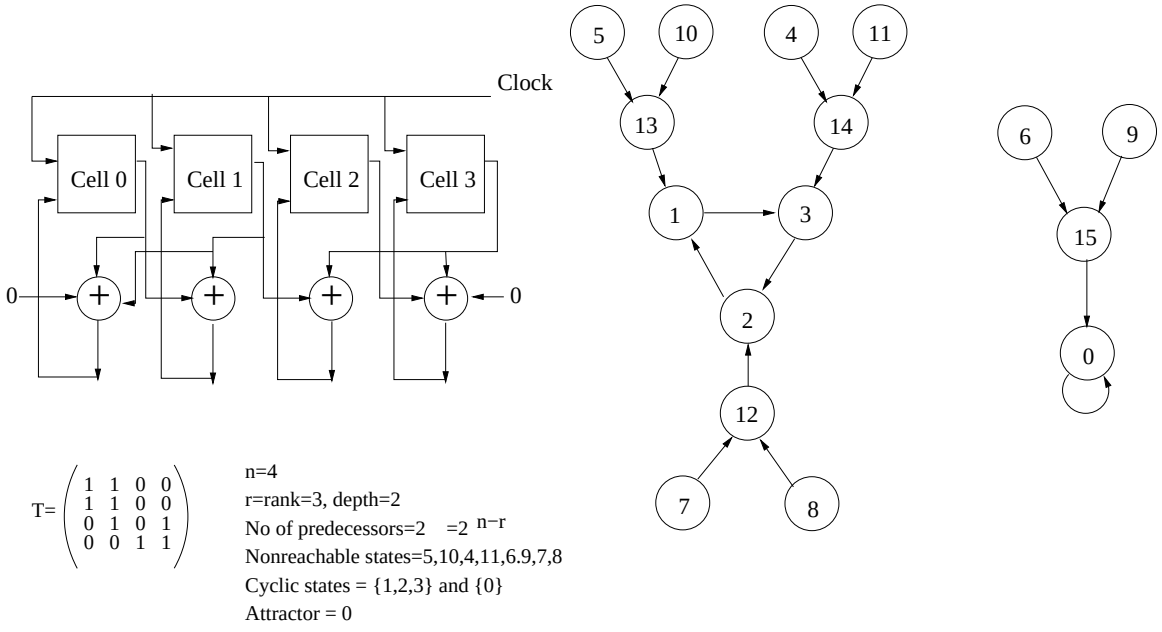


Figure 3.5: **Structure and state-transition graph of a 4-cell linear CA with rule** $< 102, 60, 90, 60 >$

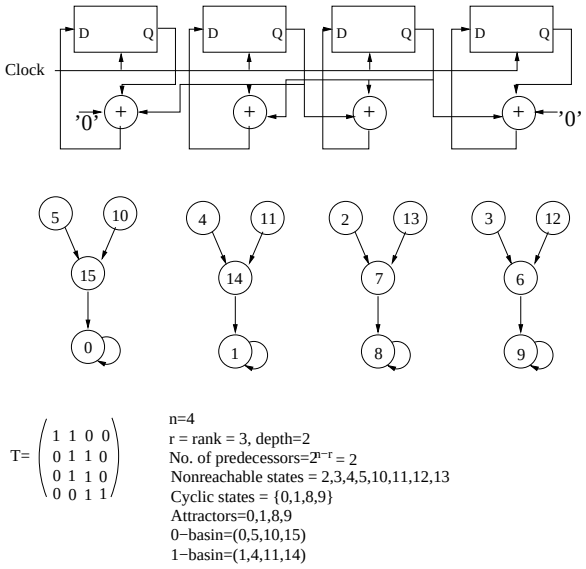


Figure 3.6: **Structure and state-transition graph of a 4-cell linear CA with rule** $< 102, 102, 60, 60 >$

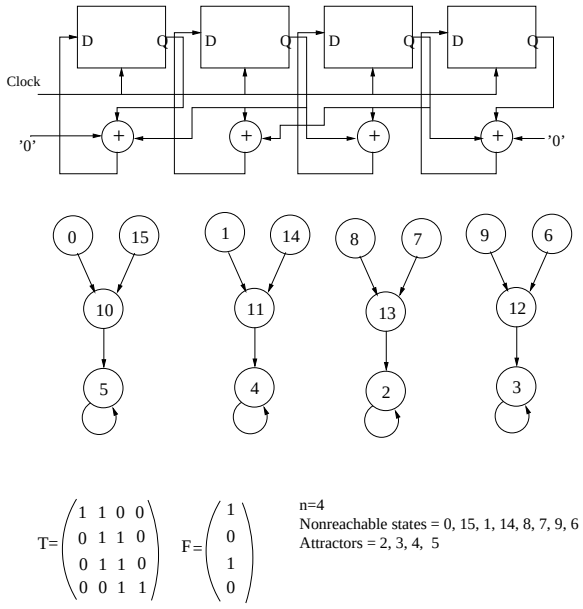


Figure 3.7: **Structure and state-transition graph of a 4-cell complemented CA**

Definition 3.10 *The tree rooted at a cyclic state α is denoted as* α-**tree**. *The set of states in* α-tree *is also referred to as* α-**basin** *(***Fig 3.6***). The terms* basin *and* tree *are used interchangeably.*

Definition 3.11 *The* **depth** *of a CA is defined to be the minimum number of clock cycles required to reach the nearest cyclic state from any non-reachable state in the state-transition diagram of the CA. The depth of both the CA illustrated in* **Figs 3.5, 3.6** *is 2.*

A few fundamental results characterizing the number of predecessors of a reachable state and the number of attractors in a non-group CA are noted below. The first lemma specifies the number of predecessors of the all-zero state. It may be noted here that the *null-space* [131] of a matrix consists of all such vectors which are transformed to the all-zero vector when premultiplied by the matrix.

Lemma 3.2 *If d is the dimension of the null space of the characteristic matrix of a non-group CA, then the total number of predecessors of the all-zero state (state 0) is* 2^d.

Theorem 3.4 *The number of predecessors of a reachable state and that of the state '0' in a non-complemented non-group CA are equal.*

Next result specifies the number of reachable/non-reachable states in the state-transition diagram of a non-group CA. Fraction of all reachable/non-reachable states has been identified by *Martin* [134] for uniform CA with the help of polynomial algebra. However, in general for any additive CA (uniform/hybrid), the fraction of reachable/non-reachable states can be enumerated from the knowledge of the number of predecessors of a reachable state, or the rank of the corresponding characteristic matrix.

Lemma 3.3 *If a state is reachable from k predecessors for a CA, then only $1/k$ of the total states are reachable.*

An attractor state x for a n-cell non-group CA is characterized by the relation,

$$\begin{aligned} Tx &= x \\ \Rightarrow (T \oplus I)x &= 0 \text{ where, } I \text{ is the } n \times n \text{ identity matrix} \end{aligned}$$

Lemma 3.4 *In an n cell non-complemented CA with characteristic matrix T, the number of attractors is 2^{n-r}, where r is the rank of the $(T \oplus I)$ matrix.*

The *depth* of a CA, as illustrated in **Fig 3.6**, can be computed directly from the *minimal polynomial* [131] of the characteristic matrix. This is elaborated in the following theorem.

Theorem 3.5 *If for the largest value k, x^k divides the minimal polynomial of the characteristic matrix of an n-cell CA, then the depth of the state-transition graph of the CA is k.*

A special class of non-group CA is the *Two Predecessor Single Attractor Cellular Automata* (TPSA-CA), where the number of attractors is two. There is a single attractor state and all the elements form an inverted tree like structure.

In the following section we present an overview regarding the cryptographic properties of a robust S-Box which are used in the security analysis of block ciphers.

3.2 Definitions related to the Security Analysis of a Block Cipher

Definition 3.12 *Balancedness: The vector space of n tuples of elements from GF(2) is denoted by V_n. Let f be a (Boolean) function from V_n to $GF(2)$. The truth table of f is defined as ($f(\alpha_0)$, $f(\alpha_1)$,...,$f(\alpha_{2^n-1})$), where α_i, $i = 0, 1, ..., 2^n - 1$, denote vectors in V_n. f is said to be balanced if its truth table has an equal number of zeroes and ones.*

Definition 3.13 *Affine Function: We call $h(x)=a_1x_1 \oplus \ldots \oplus a_nx_n \oplus c$ an affine function where $x = (x_1, \ldots, x_n)$ and a_j,$c \in GF(2)$. In particular, h will be called a linear function if $c = 0$.*

Definition 3.14 *Hamming Weight: The Hamming weight of a vector x, denoted by $W(x)$, is the number of ones in x.*

Definition 3.15 *Hamming Distance: Let f and g be functions on V_n. Then $d(f, g) = \sum_{f(x)\neq g(x)} 1$, where the addition is over the reals, is called the Hamming distance between f and g.*

Definition 3.16 *Non-linearity: Let ψ_0,... $\psi_{2^{n+1}-1}$ be the affine functions on V_n. Then $N_f = min_{i=0,\ldots,2^{n+1}-1}$ d(f,ψ_i) is called the non-linearity of f. It is well-known that the non-linearity of f on V_n satisfies $N_f \leq 2^{n-1}$-$2^{n/2-1}$, when n is even.*

Definition 3.17 *Bias of linear approximation: Let the linear approximation be of the form:*

$$< X_{i1} \oplus X_{i2} \oplus \ldots X_{iu} > \oplus < Y_{j_1} \oplus Y_{j_2} \ldots \oplus Y_{j_v} >= 0$$

where X_i represents the i-th bit of the input $X = [X_1, X_2, \ldots]$ and Y_j represents the j-th bit of the output $Y = [Y_1, Y_2, \ldots]$. This equation is representing the exclusive OR of u input bits and v output bits.

If the bits are chosen randomly then the above approximated linear expression will hold with probability 1/2. If p_l is the probability with which the expression holds then the bias is defined as $|p_l - 1/2|$.

Definition 3.18 *Robustness of S-Box [72]: Let $F=(f_1,...,f_s)$ be an $n \times s$ S-box, where f_i is a function on V_n, $i = 1, ..., s$, and $n \geq s$. We denote by L the largest value in the difference distribution table of F, and by R the number of non-zero entries in the first column of the table. In either case the value 2^n in the first row is not counted. Then we say that F is ϵ-robust against differential cryptanalysis, where ϵ is defined by*

$$\epsilon = (1 - R/2^n)(1 - L/2^n) \tag{3.1}$$

3.3 Conclusion

Chapters 2 and *3* report the background on which the present dissertation is based. *Chapters 4* to *10* describe the contribution of the thesis. The characterization of a special class of Cellular Automata and its applicability to the construction of cryptographic algorithms have been dealt with in the next chapter.

Chapter 4

Characterization of a Class of Complemented Group Cellular Automata

4.1 Introduction

Almost all cryptographic applications depend on the underlying strength of primitives. Certain basic blocks are repeatedly applied in order to build various cryptographic applications like block ciphers and key agreement protocols. The requirement on these sub-components or functions are varied and depends on the applications which are built out of them. The functions must satisfy certain cryptographic properties like non-linearity and avalanche criterion. For the functions to be used for the development of block ciphers the mappings must be invertible. With the advent of electronic commerce and portable devices for communications, cryptographic implementations have become exceedingly important. Hence, it is also imperative for the designers of crypto-algorithms that the functions are easily amenable to both software and hardware realizations. In the present work the fundamental block used to develop the cryptographic primitives is the Cellular Automaton (CA).

The Cellular Automaton (CA) was first introduced by *J. von Neumann* for modelling self-reproducible configurations. Wolfram [135] suggested simplification of the automaton with local inter-connections. The authors of [134] have used algebraic techniques to give an extensive analysis of the global properties of non-complemented Cellular Automaton (CA) with periodic boundary conditions. The systems were found to be irreversible and found to evolve through transients to attractors consist-

ing of a large number of configurations (**Fig. 4.1**).

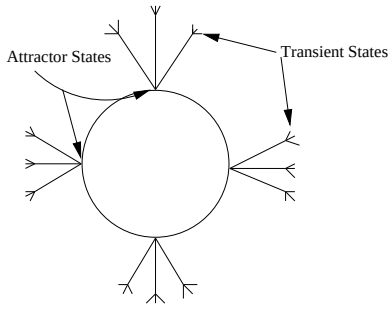


Figure 4.1: **Cellular Automaton Evolution**

The work reported in [134] concentrates on a class of Cellular Automata which exhibit the simplifying feature of "additivity". The configurations of such a Cellular Automaton satisfy an "additive superposition" principle, which allows a natural representation of the configurations by characteristic polynomials. The time evolution of the configurations is represented by iterated multiplication of their characteristic polynomials by fixed polynomials. The algebraic properties of these polynomials are then analyzed to determine the dependence of cycle lengths on the number of cells (or sites). In [136] and [129] the complemented Cellular Automaton has been proposed and studied using matrix algebra. However the works provide no final conclusion regarding the dependence of the length of the cycle spaces on the number of cells of a complemented Cellular Automaton. In summary, the nature of the complemented CA was not easily amenable to both algebraic and matrix analysis. In this work complemented CA has been characterized and it has been proved that the cycle lengths are equal irrespective of the number of cells. The exact dependence of the cycle lengths on the number of cells has been formulated. There are no rules in existing literature which can relate the separate cycles that are formed by the CA. Without such rules a Cellular Automaton does not allow for the transition from one cycle to another. Thus the entire state space cannot be effectively used and this leads to worries for the application of the CA in cryptography [129]. The current work develops new rules which govern the transitions between the cyclic subspaces of complemented CA. These properties promise the CA to function as an ideal candidate for cryptographic algorithms [137, 138].

The chapter also shows a novel method how to extend the rule of the Cellular Automaton to a more generalized structure with the same characterization and same rules. The extension shows that the same set of properties hold for CA structures which does not complement the entire state vector of the automaton. Possible applications of the developed properties in cryptography have been finally proposed in the work.

The outline of the chapter is as follows: *Section 4.2* details the characterization of the complemented CA. The state spaces of the complemented Cellular Automaton are related in *section 4.3*. *Section 4.4* extends the characterization of the complemented CA to a more generalized structure. *Section 4.5* proposes possible applications of the CA in design of cryptographic algorithms. The chapter is concluded in *section 4.6*.

4.2 Characterization of Complemented CA

A *Cellular Automaton (CA)* consists of a number of cells arranged in a regular manner, where the state transitions of each cell depends on the states of its neighbours. The next state of a particular cell is assumed to depend only on itself and on its two neighbours (3-neighbourhood dependency). The state x of the i^{th} cell at time $(t+1)$ is denoted as $x_{t+1}^i = f(x_t^{i-1}, x_t^i, x_t^{i+1})$, where x_t^i denotes the state of the i^{th} cell at time t and f is the next state function called the rule of the automata [135]. Since f is a function of 3 variables, there are 2^{2^3} or 256 possible next state functions. The decimal equivalent of the output column in the truth table of the function is denoted as the rule number. The next state function for different rules are stated below as examples:

$$\begin{aligned}
\text{Rule 90} &: \quad x_{t+1}^i = x_t^{i-1} + x_t^{i+1} \\
\text{Rule 150} &: \quad x_{t+1}^i = x_t^{i-1} + x_t^i + x_t^{i+1} \\
\text{Rule 51} &: \quad x_{t+1}^i = \overline{x_t^i} \\
\text{Rule 153} &: \quad x_{t+1}^i = \overline{x_t^i + x_t^{i+1}} \\
\text{Rule 195} &: \quad x_{t+1}^i = \overline{x_t^{i-1} + x_t^i}
\end{aligned}$$

In the above next state functions + denotes bitwise xor.

Definition 4.1 Complemented CA: *If the rules of the CA involve XNOR logic then the rules are called complemented rules. For example the rules* 153 *and* 195 *are instances of such rules. The corresponding CA is known as complemented Cellular Automaton, the state transition of which is characterized by the transition matrix* $(\overline{T})$*. This class of Cellular Automata have also been referred to as the fundamental transformations[129].*

The complemented CA with rule 153 has been characterized in this chapter and it has been verified that the rule 195 has also the same characterization and hence omitted. Also the characterization has been extended to a more general class of complemented CA in **section 4.4**.

The rule 153 represented as $(\overline{x_t^i + x_t^{i+1}})$, is the additive complement of rule 102 represented as $(x_t^i + x_t^{i+1})$. It is known that [129] if a cellular automaton with rule 153 is fed with an initial seed of X, then the cellular automaton produces an output $\overline{T}(X) = T(X) + IF$, where I is a unit matrix and F is all one vector. Hence, we have $X, \overline{T}(X)$ and $\overline{T}^2(X)$ members of the same cycle. Physically, an n-cell uniform CA having rule 153 evolves with equal number of cyclic states.

The following existing lemmas are cited without proof for convenience.

Lemma 4.1 *[133] If R is a space generated by a linear operator G (say), then the space can be decomposed into cyclic subspaces*

$$R = I_1 + I_2 + \ldots + I_t$$

such that the minimal polynomial of each of these cyclic subspace is a power of an irreducible polynomial.

Lemma 4.2 *[139] If the degree of an irreducible polynomial is m, then the period of this polynomial is $(2^m - 1)$.*

Lemma 4.3 *[132] If the period of an irreducible polynomial $p(x)$ is k then the period of $[p(x)]^j$ is kq^r, where $q^{r-1} < j \leq q^r$, q being the modulus.*

Based on the above results we prove the following results, in order to characterize the state transitions of the complemented Cellular Automata.

Theorem 4.1 *The CA having rule T and number of cells ranging from $(2^{k-1}+1)$ to 2^k have maximum cycle length of $m = 2^k$.*

Proof: Let us consider the cycle structure of $Y = T(X)$, where T is the $n \times n$ matrix of rule 102.

$$T = \begin{bmatrix} 1 & 1 & 0 & . & . & 0 \\ 0 & 1 & 1 & . & . & 0 \\ 0 & 0 & 1 & 1 & . & 0 \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & 1 \end{bmatrix}$$

For the linear CA having characteristic matrix T, R is the space generated by the linear operator T. The characteristic polynomial of the n-cell CA is given by $p_2(x) = (x+1)^n = [p_1(x)]^n$, where $p_1(x) = (1+x)$.

Thus according to **lemma 4.1**, the space generated by the linear operator T can be decomposed into cyclic subspaces. The minimal polynomial of each of the cyclic subspace is a power of the irreducible polynomial $(1+x)$. Clearly, the cyclic subspace with maximum length is due to $p_2(x) = (1+x)^n$.

By **lemma 4.2**, period of $p_1(x)$=$(1+x)$ is $k = 2^1 - 1 = 1$. Period of $p_2(x)$ can be found by **lemma 4.3**. In this case, the modulus $q = 2$ and $k = 1$. So the period of $p_2(x)$ is 2^r where $2^{r-1} < n \leq 2^r$
or, $\lceil log(n) \rceil \leq r < \lceil log(n) \rceil + 1$.

So, the period of $p_2(x)$ and hence the maximum cycle length of the CA generated by the linear operator T is always some power of 2.

From the inequality $2^{r-1} < n \leq 2^r$, if the number of cells n is of the form 2^k we have $2^{r-1} < 2^k \leq 2^r$. Thus, k=r and hence the maximum cycle length is 2^k.

If, n is of the form 2^k+1, we have $2^{r-1} < (2^k+1) \leq 2^r \Rightarrow r > k$ and $r-1 < k+1$ $\Rightarrow k < r < k+2$. Since r is an integer, $r = k+1$ and the period of $p_2(x)$ is 2^{k+1} (since the period of $p_2(x)$ is a power of 2). Hence the maximum size of a CA that evolves cycle length of 2^k is 2^k. Thus the maximum size of the CA that evolves cycle length of 2^{k-1} is 2^{k-1}.

Thus, if the number of cells range from $2^{k-1}+1$ to 2^k the maximum cycle length is 2^k.

□

Thus the maximum cycle length of a CA with characteristic matrix T is 2^k, where the number of cells (n) range from $(2^{k-1}+1)$ to 2^k.

Let the global state of an n-cell CA be represented by the vector X.

Theorem 4.2 *[129]* If $\overline{T}^p$ denotes p times application of the complemented CA operator $\overline{T}$, then,

$$\overline{T}^p(X) = (I + T + T^2 + \ldots + T^{p-1}).F + T^p.(X) \tag{4.1}$$

where T is the matrix of the corresponding non-complemented CA and F is the all one vector.

In order to evaluate the cycle structure of the complemented CA with rule $\overline{T}$ the following lemmas are developed.

Lemma 4.4 *If there exists a maximum cycle of length k for the linear CA characterized by operator T, then there will exist either a cycle of length k or $2k$ for the complemented CA characterized by $\overline{T}$.*

Proof: Since, k is the maximum length of the cycles in the state transitions of the CA characterized by operator T, $T^k = I$. Let, the length of a cycle for the complemented CA be $nk + r$, where $r < k$.

Thus, $\overline{T}^{nk+r}(X) = X$ and $T^{nk} = I$.

$$\begin{aligned}
\text{Hence, } X &= T^{nk+r}(X) + [I + T + T^2 + \ldots + T^{nk+r-1}]F \\
&= T^r(X) + [I + T + T^2 + \ldots + T^{nk-1}]F \\
&\quad + T^{nk}[I + T + T^2 + \ldots + T^{r-1}]F \\
\Rightarrow (I + T^r)X &= [I + T + T^2 + \ldots + T^{nk-1}]F \\
&\quad + T^{nk}[I + T + T^2 + \ldots + T^{r-1}]F \\
\Rightarrow (I + T)(I + T^r)X &= (I + T)[I + T + T^2 + \ldots + T^{nk-1}]F \\
&\quad + (I + T)[I + T + T^2 + \ldots + T^{r-1}]F \\
\Rightarrow (I + T^r)(I + T)X &= (I + T^{nk})F + (I + T^r)F \\
\Rightarrow (I + T^r)[(I + T)X + IF] &= 0
\end{aligned}$$

If $(I + T^r) \neq 0$, for solution of X to exist rank$(I + T)$=rank$(I + T, F)$. Here $(I + T, F)$ is the augmented matrix constructed from the matrix $(I + T)$ and the all one vector F.

But the lower row of $I + T$ is zero, so the rank is less than n, the order of the matrix T. But F is an all one vector, so the ranks cannot be equal.

Thus, $(I + T^r)$=0. Since $r < k$, hence this is not possible. Thus we arrive at a contradiction. Thus the maximal cycle length of the complemented CA is of the form nk, where n is a natural number.

Let X be the current state. The state of the complemented CA after m cycles is given by $\overline{T}^m(X)$=$[I + T + T^2 + \ldots T^{m-1}]F + T^m(X)$. If k is the order of the group generated by T, then $T^k = I$. Now to show $\overline{T}$ also generates a group it is needed to show that there exists an $m = k'$ such that $[I + T + T^2 + \ldots + T^{k'-1}] = 0$ and $T^{k'} = I$. For $k' = 2k$,

$[I + T + T^2 + T^3 + \ldots + T^{k-1} + T^k + T^{k+1} + \ldots + T^{2k-1}]$
$=[I + T + T^2 + T^3 + \ldots T^{k-1}] + [I + T + T^2 + T^3 + \ldots T^{k-1}]T^k$
$=[I + T + T^2 + T^3 + \ldots T^{k-1}][I + T^k]$

$=0$, since $T^k = I$.
Also, $T^{2k} = (T^k)^2 = I$. So, the length of the cycles generated by $\overline{T}$ is less than or equal to $2k$. Thus combining with the previous part, the cycle length of the complemented CA is either k or $2k$. □

Next, the condition for determining whether the order is k or $2k$ is stated, where k denotes the maximum cycle length of the linear CA characterized by the matrix T. Let, the term $diff_k$ be defined as the difference between the states of the complemented CA (characterized by $\overline{T}$) and the linear CA (characterized by T) after k clock cycles. If both the linear CA (T) and the complemented CA ($\overline{T}$) have initial states X and k clock cycles are applied we have:

$$\begin{aligned} diff_k &= \overline{T}^k(X) + T^k(X) \\ &= [I + T + T^2 + \ldots T^{k-1}]F + T^k(X) + T^k(X) \\ &= [I + T + T^2 + \ldots T^{k-1}]F \end{aligned}$$

Thus if $diff_k = 0$ and the maximum cycle length of the linear CA is k, we have:

$$\overline{T}^k(X) = T^k(X) + diff_k = T^k(X) = X.$$

Therefore if $diff_k = 0$ both the state transitions $\overline{T}$ and T will have length k, else the cycle lengths of the state transition $\overline{T}$ is $2k$.

Theorem 4.3 *Let k be the maximum cycle length generated by the linear CA characterized by the matrix T. Then if the number of cells of the Cellular Automaton is not a power of 2, $diff_k = 0$, whereas if the number of cells is a power of 2, $diff_k \neq 0$.*

Proof: Case 1: $diff_k \neq 0$ if the number of cells is a power of 2 (n=2^m)

Let the base case be denoted by m=1 and hence the number of cells of the Cellular Automaton is 2. Thus the corresponding characteristic matrix T is a 2×2 matrix denoted by:

$$\mathbf{T} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

The maximum cycle length generated by the 2 cell CA is 2 by **theorem 4.1** and thus the value of k is 2.

$$diff_2 = (I+T)F = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \right) F = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} F$$

which is not zero, thus the base case is satisfied.

Let, the number of cells be $n = 2^m$ and hence the characteristic matrix T is of size $n \times n$. By **theorem 4.1** the maximum cycle length generated by the CA is n. Thus $T^n = I$ and we need to prove that $diff_n \neq 0$.

For convenience of proof we denote the characteristic matrix of an n-cell CA by the matrix T_n. Similarly, the corresponding matrix for an $n/2$ cell CA is denoted by $T_{n/2}$ and so on.

Also since $n = 2^m$, we have $T_n^n = I_n$, where I_n is the identity matrix of order n. So, $n/2 = 2^{m-1}$ and hence, by **theorem 4.1** the maximum cycle length generated by the $n/2$ cell CA is $n/2 \Rightarrow T_{n/2}^{n/2} = I_{n/2}$. Here $I_{n/2}$ is the identity matrix of order $n/2$. Thus,

$$I_{n/2} + T_{n/2}^{n/2} = 0 \tag{4.2}$$

The $n \times n$ matrix T_n can be written recursively in the form:

$$\mathbf{T_n} = \begin{bmatrix} T_{n/2} & A_{n/2} \\ 0 & T_{n/2} \end{bmatrix}$$

where $A_{n/2}$ is the following $n/2 \times n/2$ matrix denoted as:

$$\mathbf{A_{n/2}} = \begin{bmatrix} 0 & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 0 & 0 & \dots & \dots & 0 \end{bmatrix}$$

The corresponding powers of T_n are computed as follows:

$$\mathbf{T_n^2} = \begin{bmatrix} T_{n/2}^2 & (T_{n/2}A_{n/2} + A_{n/2}T_{n/2}) \\ 0 & T_{n/2}^2 \end{bmatrix}$$

$$\mathbf{T_{n/2}^3} = \begin{bmatrix} T_{n/2}^3 & (T_{n/2}^2 A_{n/2} + T_{n/2}A_{n/2}T_{n/2} + A_{n/2}T_{n/2}^2) \\ 0 & T_{n/2}^3 \end{bmatrix}$$

Similarly,

$$\mathbf{T_n^{n-1}} = \begin{bmatrix} T_{n/2}^{n-1} & (T_{n/2}^{n-2}A_{n/2} + T^{n-3}A_{n/2}T_{n/2} + \ldots + T_{n/2}A_{n/2}T_{n/2}^{n-3} + A_{n/2}T_{n/2}^{n-2}) \\ 0 & T_{n/2}^{n-1} \end{bmatrix}$$

$$\begin{aligned} \text{Thus, } \mathit{diff}_{\mathbf{n}} &= (\begin{bmatrix} I_{n/2} & 0 \\ 0 & I_{n/2} \end{bmatrix} + \begin{bmatrix} T_{n/2} & A_{n/2} \\ 0 & T_{n/2} \end{bmatrix} + \begin{bmatrix} T_{n/2}^2 & T_{n/2}A_{n/2} + A_{n/2}T_{n/2} \\ 0 & T_{n/2}^2 \end{bmatrix} + \ldots + \\ &\quad \begin{bmatrix} T_{n/2}^{n-1} & T_{n/2}^{n-2}A_{n/2} + T_{n/2}^{n-3}A_{n/2}T_{n/2} + \ldots + T_{n/2}A_{n/2}T_{n/2}^{n-3} + A_{n/2}T_{n/2}^{n-2} \\ 0 & T_{n/2}^{n-1} \end{bmatrix})F \\ &= \begin{bmatrix} 0 & B_{n/2} \\ 0 & 0 \end{bmatrix} F \end{aligned}$$

It may be noted that in the above form we have used the following facts and notations:

- 0 denotes the corresponding zero matrix.
- $(I_{n/2}+T_{n/2}+T_{n/2}^2+\ldots+T_{n/2}^{n-1}) = (I_{n/2}+T_{n/2}^{n/2})(I_{n/2}+T_{n/2}+T_{n/2}^2+\ldots+T_{n/2}^{n/2-1}) = 0$, since $I_{n/2} + T_{n/2}^{n/2} = 0$ (by **equation 4.2**).
- Also, $B_{n/2}$ is an $n/2 \times n/2$ matrix and is equal to:

$$\begin{aligned} B_{n/2} &= A_{n/2} + (T_{n/2}A_{n/2} + A_{n/2}T_{n/2}) + \ldots + (T_{n/2}^{n-2}A_{n/2} + \ldots + A_{n/2}T_{n/2}^{n-2}) \\ &= (I_{n/2} + T_{n/2} + T_{n/2}^2 + \ldots + T_{n/2}^{n/2-1})A_{n/2}(I_{n/2} + T_{n/2} + T_{n/2}^2 + \ldots + T_{n/2}^{n/2-1}) \\ &\quad (\text{since } T_{n/2}^{n/2} = I_{n/2}) \\ &= \begin{bmatrix} 0 & B_{n/4} \\ 0 & 0 \end{bmatrix} \circ A_{n/2} \circ \begin{bmatrix} 0 & B_{n/4} \\ 0 & 0 \end{bmatrix}, \text{ where } \circ \text{ represents matrix multiplication} \end{aligned}$$

In the above form $B_{n/4}$ is an $n/4 \times n/4$ matrix, similarly obtained in the second stage of the decomposition.

$$\begin{aligned} \text{Thus, } B_{n/4} &= (I_{n/4} + T_{n/4} + T_{n/4}^2 + \ldots + T_{n/4}^{n/4-1})A_{n/4}(I_{n/4} + T_{n/4} + T_{n/4}^2 + \ldots + T_{n/4}^{n/4-1}) \\ &\quad (\text{since } T_{n/4}^{n/4} = I_{n/4}) \\ &= \begin{bmatrix} 0 & B_{n/8} \\ 0 & 0 \end{bmatrix} \circ A_{n/4} \circ \begin{bmatrix} 0 & B_{n/8} \\ 0 & 0 \end{bmatrix} \end{aligned}$$

Applying this decomposition recursively after i stages we have:

$$\begin{aligned}
\text{Thus, } B_{n/2^i} &= (I_{n/2^i} + T_{n/2^i} + T_{n/2^i}^2 + \ldots + T_{n/2^i}^{n/2^i-1}) A_{n/2^i} (I_{n/2^i} + T_{n/2^i} + T_{n/2^i}^2 + \ldots \\
&\quad \ldots + T_{n/2^i}^{n/2^i-1}) \\
&\quad (\text{since } T_{n/2^i}^{n/2^i} = I_{n/2^i}) \\
&= \begin{bmatrix} 0 & B_{n/2^{i+1}} \\ 0 & 0 \end{bmatrix} \circ (A_{n/2^i}) \circ \begin{bmatrix} 0 & B_{n/2^{i+1}} \\ 0 & 0 \end{bmatrix}
\end{aligned}$$

Thus after $m-1$ stages:

$$\begin{aligned}
B_2 &= (I_2 + T_2) A_2 (I_2 + T_2) \\
&= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \right) \cdot \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \right) \\
&= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.
\end{aligned}$$

$$\begin{aligned}
\text{Thus, } B_4 &= \begin{bmatrix} 0 & B_2 \\ 0 & 0 \end{bmatrix} \circ A_4 \circ \begin{bmatrix} 0 & B_2 \\ 0 & 0 \end{bmatrix} \\
&= \left(\left[\begin{array}{cc|cc|c} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \cdot \left[\begin{array}{cc|cc|c} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{array} \right] \cdot \left[\begin{array}{cc|cc|c} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right] \right) \\
&= \left[\begin{array}{cc|cc|c} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right].
\end{aligned}$$

Using this recursively,

$$diff_{\mathbf{n}} = \begin{bmatrix} 0 & B_{n/2} \\ 0 & 0 \end{bmatrix} F = \left[\begin{array}{ccc|ccc} 0 & \dots & 0 & \dots & \dots & 1 \\ 0 & \dots & 0 & \dots & \dots & 0 \\ 0 & \dots & 0 & \dots & \dots & 0 \\ - & - & - & - & - & - \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & 0 \end{array}\right] F \neq 0$$

This proves that when n is a power of 2, the maximum cycle length k generated by the linear CA is n. Thus, $k = n$ and $diff_k = diff_n \neq 0$. The result when n is not a power of 2 is proved in **case 2**.

Case 2: $diff_k = 0$ **if the number of cells is not a power of** 2 ($n = 2^{m+1} - l$, $l < 2^m$)

The characteristic matrix of a CA with size $n = 2^{m+1}$ can be written as

$$\mathbf{T_{2^{m+1}}} = \left[\begin{array}{ccccc|c} 1 & 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & 1 & \dots & \dots & 0 \\ 0 & 0 & 1 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & 1 & 1 \\ - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & \dots & 1 \end{array}\right] = \left[\begin{array}{ccccc|c} & & & & & . \\ & & & & & . \\ & & T_{(2^{m+1}-1)} & & & . \\ & & & & & . \\ & & & & & . \\ - & - & - & - & - & - \\ 0 & 0 & \dots & 0 & & 1 \end{array}\right]$$

Here $T_{(2^{m+1}-1)}$ is the characteristic matrix of the Cellular Automaton with size $2^{m+1} - 1$ and the number of cells is not a power of 2.

From **theorem 4.1** the maximum cycle length of both the automata having $2^{m+1} - 1$ and 2^{m+1} cells is 2^{m+1}.

Since the number of cells in the automata characterized by the matrix T_{2^m} is a power of 2 we may use the result proved in **case 1** to evaluate:

$$(I_{2^{m+1}} + T_{2^{m+1}} + \ldots + T_{2^{m+1}}^{2^{m+1}-1}) = \left[\begin{array}{ccccc|c} & & & & & . \\ & & & & & . \\ & & & & & . \\ & & I_{2^{m+1}-1} & & & . \\ & & & & & . \\ & & & & & . \\ & & & & & . \\ - & - & - & - & - & - \\ 0 & 0 & \ldots & 0 & & 1 \end{array}\right] +$$

$$\left[\begin{array}{ccccc|c} & & & & & . \\ & & & & & . \\ & & & & & . \\ & & T_{2^{m+1}-1} & & & . \\ & & & & & . \\ & & & & & . \\ & & & & & . \\ - & - & - & - & - & - \\ 0 & 0 & \ldots & 0 & & 1 \end{array}\right] + \ldots + \left[\begin{array}{ccccc|c} & & & & & . \\ & & & & & . \\ & & & & & . \\ & & T_{2^{m+1}-1}^{2^{m+1}-1} & & & . \\ & & & & & . \\ & & & & & . \\ & & & & & . \\ - & - & - & - & - & - \\ 0 & 0 & \ldots & 0 & & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccccc|c} 0 & 0 & 0 & \ldots & \ldots & 1 \\ 0 & 0 & 0 & \ldots & \ldots & 0 \\ 0 & 0 & 0 & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ - & - & - & - & - & - \\ 0 & 0 & \ldots & \ldots & 0 & 0 \end{array}\right]$$

From **theorem 4.1** it is evident that the maximum cycle length generated by the linear automata characterized by $T_{2^{m+1}-1}$ is $k = 2^{m+1}$.
Thus $diff_k$=$diff_{2^{m+1}}$ =$(I_{2^{m+1}-1} + T_{2^{m+1}-1} + T_{2^{m+1}-1}^2 + \ldots + T_{2^{m+1}-1}^{2^{m+1}-1})F = 0$.
Likewise for a CA of size $2^{m+1} - l$ $(l < 2^m)$ and characteristic matrix $T_{2^{m+1}-l}$ the maximum cycle length $k = 2^{m+1}$.
Thus the value of $diff_k = (I_{2^{m+1}-l} + T_{2^{m+1}-l} + T_{2^{m+1}-l}^2 + \ldots + T_{2^{m+1}-l}^{2^{m+1}-1})F = 0$.
Thus if the size of the CA is not a power of 2, $diff_k = 0$. This completes the proof.

□

Theorem 4.4 *The length of cycle for an n-cell CA, having rule $\overline{T}$, is*

$$l = 2^{\lfloor logn \rfloor + 1} \quad n \geq 2 \ . \tag{4.3}$$

Proof: The result is trivial for $n = 2$. Using the result that if the number of cells of the linear CA (with transition matrix T) range from $2^{k-1} + 1$ to 2^k the maximum cycle length is 2^k (**table 4.1**).

Table 4.1: **Maximum Cycle Length**

Range for the number of cells	Maximum Cycle Length
2	2
3 – 4	4
5 – 8	8
9 – 16	16
17 – 32	32
⋮	⋮
$(2^{n-1}+1) - (2^n)$	2^n

Thus, the maximum length of the cycles generated by the linear CA is $2^{\lceil log(n) \rceil}$. From **theorem 4.3**, if k is the maximum cycle length of the linear CA and the number of cells is not a power of 2, $diff_k = 0$. In that case by **lemma 4.4**, the length of the complemented CA is also k. But if the number of cells is a power of 2, $diff_k \neq 0$. Then, the length of the complemented CA is $2k$. Now,

$$\lceil log(n) \rceil = \begin{cases} \lfloor log(n) \rfloor + 1, & n \neq 2^k \\ \lfloor log(n) \rfloor, & n = 2^k \end{cases} \tag{4.4}$$

Thus, the length of an n-cell complemented CA ($n > 2$) is formulated by

$$l = \begin{cases} 2^{\lceil log(n) \rceil} = 2^{\lfloor log(n) \rfloor + 1} & \text{n} \neq 2^k \\ 2.2^{\lceil log(n) \rceil} = 2^{\lfloor log(n) \rfloor + 1} & \text{n} = 2^k \end{cases} \tag{4.5}$$

This completes the proof for $n > 2$. □

In order to explain the characteristics derived so far in the chapter the following example is provided.

Example 4.1 *In this example we consider a* 5 *cell complemented CA with rule* 153. *Thus, the state transition of the CA is defined as:*

$$Y = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = T(X) + IF = \overline{T}(X).$$

The cycle structure of the CA with transition matrix T (rule 102*) is derived using the above properties. The notations used and the theoretical background of the computation are described in* **section 3.1.1**. *To restate, the characteristic polynomial of the characteristic matrix of the group CA solely decides the cycle structure. Each factor of the characteristic polynomial,* $\phi_i(x)$ *contributes* μ_i *cycles of length* k_i. *The characteristic polynomial of the CA with characteristic matrix T is* $(1+x)^5$. *Hence the cycle structure depends on the five factors,* $(1+x), (1+x)^2, (1+x)^3, (1+x)^4$ *and* $(1+x)^5$.

Contribution of the term $(1+x)$**:** $\mu_1 = 1$ *and* $k_1 = 1$. *Thus, this term contributes one cycle of length* 1.

Contribution of the term $(1+x)^2$**:** $\mu_2 = 1$ *and* $k_2 = 2$. *Thus, this term contributes one cycle of length* 2.

Contribution of the term $(1+x)^3$**:** $\mu_3 = 1$ *and* $k_3 = 4$. *Thus, this term contributes one cycle of length* 4.

Contribution of the term $(1+x)^4$**:** $\mu_4 = 2^{4-1-\lceil log(4) \rceil} = 2$ *and* $k_4 = 2^{\lceil log(4) \rceil} = 4$. *Thus, this term contributes two cycles of length four.*

Contribution of the term $(1+x)^5$**:** $\mu_5 = 2^{5-1-\lceil log(5) \rceil} = 2$ *and* $k_5 = 2^{\lceil log(5) \rceil} = 8$. *Thus, this term contributes two cycles of length eight.*

Thus, the overall cycle structure is $[1, 1(1), 1(2), 3(4), 2(8)]$, *the first cycle being for the all zero data. Also it may be noted that the number of cells is* 5, *which is between* $2^2 + 1$ *and* 2^3. *Hence the maximum length of a cycle is* $2^3 = 8$, *according to* **theorem 4.1**. *The cycle structure is depicted in* **Fig. 4.2**.

The state spaces of the corresponding complemented CA is depicted in **Fig. 4.3**. *The length of each cycle is as characterized in* **theorem 4.4**, $2^{\lfloor log(5) \rfloor + 1} = 2^3 = 8$.

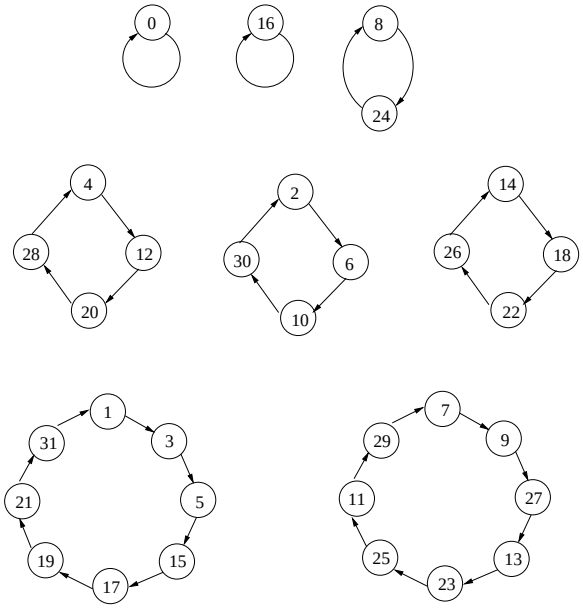


Figure 4.2: **The State Space of a** 5 **cell CA with rule** 102

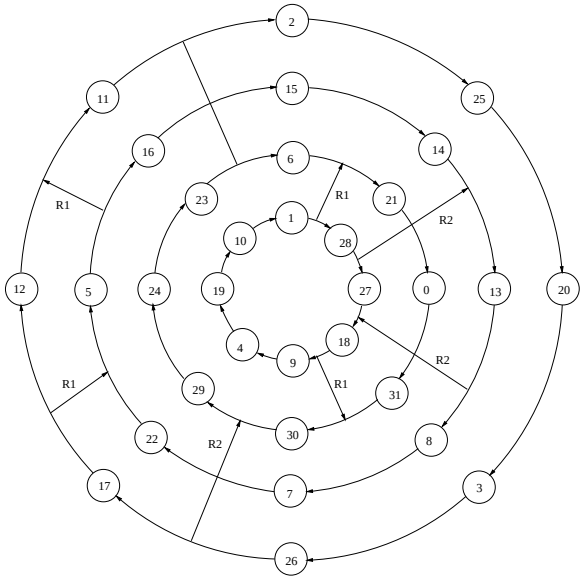


Figure 4.3: **The State Space of a** 5 **cell complemented CA with rule** 153

The following theorems find out the inter-relationships among these cycles (marked as $R1$ and $R2$ in **Fig. 4.3**). The objective is to find out a new set of rules which can help elements to migrate from any position of the state space to another. The relation between the state spaces is hence searched subsequently in the chapter.

4.3 Relation between the state spaces of Complemented CA

Theorem 4.5 *Every element, X of a cycle generated by CA with rule $\overline{T}$ when mapped by the rule, $X + \overline{T}(X) + \overline{T}^2(X)$, also lie in a cycle.*

Proof: Given $X, \overline{T}(X), \overline{T}^2(X)$ and $\overline{T}^3(X)$ are members of the same cycle. In the following equations, $+$ means simple xor operation.

$$\begin{aligned} \overline{T}(X) &= T(X) + IF \\ \overline{T}^2(X) &= T^2(X) + (I+T)F \end{aligned} \tag{4.6}$$

Let the rule map X to an element ϵ_1, such that :

$$\epsilon_1 = X + \overline{T}(X) + \overline{T}^2(X) \tag{4.7}$$

$$\text{or, } \epsilon_1 = X + T(X) + T^2(X) + TF \tag{4.8}$$

The same rule when applied on the next element, $\overline{T}(X)$ we have,

$$\begin{aligned} \epsilon_2 &= [T(X) + IF] + [T^2(X) + (I+T)F] + [T^3(X) + (I+T+T^2)F] \\ \text{or, } \epsilon_2 &= \overline{T}(X + T(X) + T^2(X) + TF) \\ \epsilon_2 &= \overline{T}(\epsilon_1) \end{aligned} \tag{4.9}$$

Therefore, ϵ_1 and ϵ_2 lie in the same cycle. □

Theorem 4.6 *If X, $\overline{T}(X), \overline{T}^2(X), \ldots$ lie in one cycle, then $X + \overline{T}(X) + \overline{T}^2(X)$, $\overline{T}(X) + \overline{T}^2(X) + \overline{T}^3(X), \ldots$ lie on a different non-intersecting cycle.*

Proof: From, **theorem 4.5**, $\epsilon_2 = \overline{T}(\epsilon_1)$, as they are members of the same cycle.

To prove that ϵ_1 and X do not belong to the same cycle, it is required to prove that no λ exists such that, $\epsilon_1 = \overline{T}^{\lambda}(X)$. Here λ is lesser than the cycle length (l) of

the complemented CA. Thus, $0 \leq \lambda \leq 2^{\lfloor (logn) \rfloor + 1}$, where n is the number of cells of the CA.
In other words, there does not exist any λ such that:

$X + T(X) + T^2(X) + TF = \overline{T}^{\lambda}(X) = T^{\lambda}(X) + [I + T + \ldots + T^{\lambda-1}]F.$

We prove the theorem by contradiction.

Case 1: Let λ exist and be even.

$$\begin{aligned} \text{LHS} &= X + T(X) + T^2(X) + TF \\ &= (I + T + T^2)(X) + TF \\ &= \begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ \ldots & 1 & 1 & 1 \\ \ldots & \ldots & 1 & 1 \\ 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \begin{bmatrix} \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \vdots \\ \overline{x}^{n-1} \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \text{RHS} &= T^{\lambda}(X) + [I + T + \ldots + T^{\lambda-1}]F \\ &= \begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \Big(\begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix} + \ldots + \\ &\quad \text{(even number of terms)} + \ldots + \begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \Big) F \\ &= \begin{bmatrix} \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \begin{bmatrix} \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \vdots \\ x^{n-1} \end{bmatrix} \end{aligned}$$

Hence, we have a contradiction, that is $x^{n-1} = \overline{x^{n-1}}$.

Case2: Let λ be odd. It may be observed from the evolution of the CA that the 0^{th} bit of $T^{\lambda}(X)$ can be represented as:
$(x^0 + x^1) + \binom{\lambda-1}{1}(x^1 + x^2) + \binom{\lambda-1}{2}(x^2 + x^3) + \ldots + (x^{\lambda-1} + x^{\lambda})$.

Here, $\binom{n}{r}$ denotes the number of ways selecting r elements from a set of n elements also commonly represented as ${}^nC_r$.

As λ is odd, we may express it as $2t+1$, for some positive integer t. Hence, the 0^{th} bit of $T^{\lambda}(X)$ can be expressed as:

$$(x^0+x^1)+(2t)(x^1+x^2)+t(2t-1)(x^2+x^3)+\ldots \quad = \quad (x^0+x^1)+t(2t-1)(x^2+x^3)+\ldots$$
$$\text{(as 2t} = 0 \bmod 2)$$

Therefore the 0^{th} bit of $T^{\lambda}(X)$ depends on x^0 and x^1. But it depends upon x^2 if $t(2t-1)$ is odd $\Rightarrow t$ is odd (as $2t-1$ is always odd).

Thus the 0^{th} bit of $T^{\lambda}(X)$ depends on x^0, x^1 and x^2 if t is odd i.e if λ is of the form $4p+3$, where p is a positive integer. If t is even, i.e if λ is of the form $4p+1$, the 0^{th} bit of $T^{\lambda}(X)$ depends on x^0, x^1 but not x^2.

Similarly, if λ is of the form $4p+1$, the i^{th} bit of $T^{\lambda}(X)$ depends on x^i, x^{i+1} but not x^{i+2}. On the contrary if λ is of the form $4p+3$, the i^{th} bit of $T^{\lambda}(X)$ depends on x^i, x^{i+1} and also x^{i+2}.

Likewise, if the exponent of T, λ had been even and of the form $4p$, then the i^{th} bit of $T^{\lambda}(X)$ depends on x^i. Similarly, if the form of λ be $4p+2$ then the i^{th} bit of $T^{\lambda}(X)$ depends on x^i and x^{i+2}. We tabulate the dependence of the i^{th} bit of $T^{\lambda}(X)$ on the bits x^0, x^1 and x^2 depending on the nature of the exponent λ in **table 4.2**.

Table 4.2: **Dependence of i^{th} bit of $T^{\lambda}(X)$ on input bits**

Exponent (λ)	Input Bit Position
$4p$	x^i
$4p+1$	x^i, x^{i+1}
$4p+2$	x^i, x^{i+2}
$4p+3$	x^i, x^{i+1}, x^{i+2}

Thus, the i^{th} bit of $[I+T+T^2+T^3](X)$ depends on $x^i+(x^i+x^{i+1})+(x^i+x^{i+2})+(x^i+x^{i+1}+x^{i+2})$ and hence none of x^i, x^{i+1} and x^{i+2}, as we are having modulo 2 sum.

$\Rightarrow$ if λ is of the form $4p+1$, then the i^{th} bit of $[I+T+\ldots+T^{\lambda-1}](X) = [I+T+\ldots+T^{4p}](X) = ([I+T+T^2+T^3]+T^4[I+T+T^2+T^3]+\ldots+T^{4p-4}[I+T+T^2+T^3]+T^{4p})(X)$, thus depends only on x^i.

Similarly, if λ is of the form $4p+3$, the i^{th} bit of $[I+T+\ldots+T^{\lambda-1}](X) = [I+T+\ldots+T^{4p+2}](X) = ([I+T+T^2+T^3]+T^4[I+T+T^2+T^3]+\ldots+T^{4p-4}[I+T+T^2+T^3]+T^{4p}[I+T+T^2])(X)$, thus depends on $x^i+x^{i+1}+x^{i+2}$.

Hence, we have for $\lambda = 4p+1$,

$$\begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & 1 & 1 & 1 \\ \dots & \dots & 1 & 1 \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \begin{bmatrix} \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & 1 & 1 & 0 \\ \dots & \dots & 1 & 1 \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \begin{bmatrix} \vdots \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Hence, we have a contradiction, that is
$(x^{n-2} + x^{n-1}) = \overline{(x^{n-2} + x^{n-1})}$.

Similarly, we have for $\lambda = 4p + 3$,

$$\begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & 1 & 1 & 1 \\ \dots & \dots & 1 & 1 \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \begin{bmatrix} \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \dots & \dots & \dots & \dots \\ \dots & 1 & 1 & 1 \\ \dots & \dots & 1 & 1 \\ 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} x^0 \\ \vdots \\ x^{n-1} \end{bmatrix} + \begin{bmatrix} \vdots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Hence, we have a contradiction, that is
$(x^{n-3} + x^{n-2} + x^{n-1}) = \overline{(x^{n-3} + x^{n-2} + x^{n-1})}$. This completes the proof. □

Corollary 4.1 *$X + \overline{T}(X) + \overline{T}^2(X), X + \overline{T}(X) + \overline{T}^3(X)$, construct non-intersecting cycles.*

In order to migrate from one state transition cycle of the complemented CA to another we require some *rules*. Next, we construct two rules $R1$ and $R2$ for the same.

Construction of rules: The following operations are defined as rules:

$$R1(X) = X + \overline{T}(X) + \overline{T}^2(X) \tag{4.10}$$

$$\text{and, } R2(X) = X + \overline{T}(X) + \overline{T}^3(X) \tag{4.11}$$

Corollary 4.2 *$R1$ and $R2$ are commutative i.e $R1(R2(X)) = R2(R1(X))$. Applying, the rules on X shows that both the operations produce the same result.*

Proof: The proof follows directly from the definition of the rules $R1$ and $R2$.

$$\begin{aligned} \text{Thus, } R1(X) &= X + \overline{T}(X) + \overline{T}^2(X) \\ &= X + [T(X) + IF] + [T^2(X) + (I + T)F] \\ &= (I + T + T^2)X + TF \end{aligned}$$

$$\begin{aligned} \text{Similarly, } R2(X) &= X + \overline{T}(X) + \overline{T}^3(X) \\ &= X + [T(X) + IF] + [T^3(X) + (I + T + T^2)F] \end{aligned}$$

$$= (I + T + T^3)X + (T + T^2)F$$

$$\begin{aligned} \text{Thus, LHS} &= R1[(I + T + T^3)(X) + (T + T^2)F] \\ &= (I + T + T^2)[(I + T + T^3)(X) + (T + T^2)F] + TF \\ &= (I + T + T^2)(I + T + T^3)(X) + T^4(F) \end{aligned}$$

$$\begin{aligned} \text{Similarly, RHS} &= R2[(I + T + T^2)(X) + TF] \\ &= (I + T + T^3)[(I + T + T^2)(X) + TF] + (T + T^2)F \\ &= (I + T + T^3)(I + T + T^2)(X) + T^4(F) \end{aligned}$$

Hence, $LHS = RHS$. □

Corollary 4.3 *$R1(\overline{T}^a(X)) = \overline{T}^a(R1(X))$, where a is any index.*

Proof:

$$\begin{aligned} \text{LHS} &= R1(\overline{T}^a(X)) \\ &= (I + T + T^2)(\overline{T}^a(X)) + TF \\ &= (I + T + T^2)(T^a(X) + (I + T + \ldots + T^{a-1})F) + TF \\ &= T^a(I + T + T^2)(X) + [(I + T + T^2)(I + T + \ldots + T^{a-1}) + T]F \\ &= T^a(I + T + T^2)(X) + [I + T + T^2 + \ldots + T^a + T^{a+1}]F \end{aligned}$$

$$\begin{aligned} RHS &= \overline{T}^a(R1(X)) \\ &= T^a(R1(X)) + (I + T + \ldots + T^{a-1})F \\ &= T^a[(I + T + T^2)X + TF] + (I + T + \ldots + T^{a-1})F \\ &= T^a(I + T + T^2)X + [I + T + T^2 + \ldots + T^a + T^{a+1}]F \end{aligned}$$

Hence, $LHS = RHS$. □

Corollary 4.4 *$\overline{T}^a(R1(R2(\overline{T}^b(X)))) = \overline{T}^b(R2(R1(\overline{T}^a(X))))$, where a and b are indices.*

Proof: Using **corollary 4.3**, we have the following:

$$LHS = \overline{T}^a(R1(R2(\overline{T}^b(X))))$$

$$\begin{aligned} &= R1(\overline{T}^a(R2(\overline{T}^b(X)))) \\ &= R1(R2(\overline{T}^a(\overline{T}^b(X)))) \end{aligned}$$

Likewise,

$$\begin{aligned} RHS &= \overline{T}^b(R2(R1(\overline{T}^a(X)))) \\ &= R2(\overline{T}^b(R1(\overline{T}^a(X)))) \\ &= R2(R1(\overline{T}^b(\overline{T}^a(X)))) \\ &= R1(R2(\overline{T}^a(\overline{T}^b(X)))), \text{applying } \textbf{corollary 4.2} \end{aligned}$$

In the above deduction we have used the fact that applying a times and then b times the complemented CA, is the same as that applying b times followed by the a times application of the CA. This may be easily followed if we consider that the state transition of the complemented CA are cyclic in nature $\Rightarrow \overline{T}^a(\overline{T}^b(X)) = \overline{T}^b(\overline{T}^a(X))$. Combining the above facts, $LHS = RHS$. □

Corollary 4.5

$$\begin{aligned} R1(a+b) &= R1(a) + R1(b) + TF \\ R2(a+b) &= R2(a) + R2(b) + (T+T^2)F \end{aligned}$$

Proof: The proof follows directly from the definition of $R1$ and $R2$.

$$\begin{aligned} R1(a+b) &= (I+T+T^2)(a+b) + TF \\ &= [(I+T+T^2)a + TF] + [(I+T+T^2)b + TF] + TF \\ &= R1(a) + R2(b) + TF \end{aligned}$$

Likewise,

$$\begin{aligned} R2(a+b) &= (I+T+T^3)(a+b) + (T+T^2)F \\ &= [(I+T+T^3)a + (T+T^2)F] + [(I+T+T^3)b + (T+T^2)F] + (T+T^2)F \\ &= R2(a) + R2(b) + (T+T^2)F \end{aligned}$$

□

The transition rules derived in the chapter is explained with the help of the following example.

Example 4.2 *Let us consider the cyclic subspaces generated by the complemented CA in* **example 4.1**. *Let, the value of* X *in* **corollary 4.4** *be* 6 *and let the values of* $a = 2$ *and* $b = 4$. *Thus, the operation of* $\overline{T}^2R1R2(\overline{T}^4(6)) = \overline{T}^2R1R2(30) = \overline{T}^2R1(20) = \overline{T}^2(13) = 7$. *The corresponding operation* $\overline{T}^4R2R1(\overline{T}^2(6))$ *also yields the value* $\overline{T}^4R2R1(0) = \overline{T}^4R2(1) = \overline{T}^4(15) = 7$. *The result thus supports* **corollary 4.4**.

4.4 Generalization of the Automaton

The above characterization is for complemented Cellular Automaton where all the bits of the state vector are complemented (xoring with the all one vector F). In the present section we show a technique to construct other Cellular Automata transitions for which the same characterization holds. The state transition of the general automaton is represented by T_g.

Lemma 4.5 *If* A *is a linear and invertible matrix, the cycle structure of the transformation* $\overline{T}$ *is the same as* $T_g = A.(\overline{T}).A^{-1}$.

Proof: Let the length of the cycles of the transformation $\overline{T}$ be l. Thus $\overline{T}^l = I$, where I is the identity matrix.

Let us evaluate

$$\begin{aligned} Y &= T_g^l(X) \\ &= (A.\overline{T}.A^{-1})(A.\overline{T}.A^{-1})\ldots(l \text{ times})(A.\overline{T}.A^{-1})(X) \\ &= (A.\overline{T}^2.A^{-1}).(A.\overline{T}.A^{-1})\ldots(A.\overline{T}.A^{-1})(X) \\ &= (A.\overline{T}^l.A^{-1})(X) \\ &= (A.A^{-1})(X)[\text{since, } \overline{T}^l(A^{-1}(X)) = A^{-1}(X)] \\ &= A.A^{-1}(X) \\ &= X \end{aligned}$$

Thus the cycle structure of $\overline{T}$ and T_g are same. □

Hence, the cycle lengths of the transitions T_g have the same dependence as that of the CA with transition defined by $\overline{T}$. The Cellular Automaton constructed from

the original CA with rule $\overline{T}$ has a transition defined by $T_g = A.(\overline{T}).A^{-1}$. Here the operator A is linear and represented by any invertible matrix. The following lemma shows how the next state can be computed from an initial seed X.

Lemma 4.6 *If a Cellular Automaton with transition defined by $T_g = A.(\overline{T}).A^{-1}$ is formed we can estimate the next state of a seed X by the equation $T_g(X) = ATA^{-1}(X) + AF$.*

The following lemmas show that the same rules $R1$ and $R2$ are applicable for the Cellular Automaton with transition rule T_g.

Lemma 4.7 *Every element, X of a cycle generated by CA with rule T_g when mapped by the rule, $X + T_g(X) + T_g^2(X)$, also lie in a cycle.*

Proof: Given $X, T_g(X), T_g^2(X)$ and $T_g^3(X)$ are members of the same cycle.

$$\begin{aligned} T_g(X) &= ATA^{-1}(X) + AF \\ T_g^2(X) &= AT^2A^{-1}(X) + A(I+T)F \\ T_g^3(X) &= AT^3A^{-1}(X) + A(I+T+T^2)F \end{aligned}$$

Let the rule map X to an element ϵ_1, such that :

$$\begin{aligned} \epsilon_1 &= X + T_g(X) + T_g^2(X) \\ \text{or, } \epsilon_1 &= A(I+T+T^2)A^{-1}(X) + ATF \end{aligned}$$

The same rule when applied on the next element, $T_g(X)$ we have,

$$\begin{aligned} \epsilon_2 &= [ATA^{-1}(X) + AF] + [AT^2A^{-1}(X) + A(I+T)F] + \\ &\quad [AT^3A^{-1}(X) + A(I+T+T^2)F] \\ \text{or, } \epsilon_2 &= T_g(A(I+T+T^2)A^{-1}(X) + ATF) \\ \epsilon_2 &= T_g(\epsilon_1) \end{aligned} \tag{4.12}$$

□

Lemma 4.8 *If X, $T_g(X), T_g^2(X), \ldots$ lie in one cycle, then $X + T_g(X) + T_g^2(X)$, $T_g(X) + T_g^2(X) + T_g^3(X), \ldots$ lie on another non-intersecting cycle.*

Lemma 4.9 *$X + T_g(X) + T_g^2(X), X + T_g(X) + T_g^3(X)$, construct non-intersecting cycles.*

Construction of rules: The following operations are defined as rules:

$$\begin{aligned} R1(X) &= X + T_g(X) + T_g^2(X) \\ \text{and, } R2(X) &= X + T_g(X) + T_g^3(X) \end{aligned}$$

Lemma 4.10 *$R1$ and $R2$ are commutative i.e $R1R2(X)$=$R2R1(X)$. Applying, the rules on X shows that both the operations produce the same result.*

Lemma 4.11 *$R1(T_g^a(X)) = T_g^a(R1(X))$, where a is any index.*

Lemma 4.12 *$T_g^a R1R2(T_g^b(X)) = T_g^b R2R1(T_g^a(X))$, where a and b are indices.*

The rules are used to migrate from one cyclic subspace generated by T_g to another. The inter-relations promise the development of CA-based cryptographic algorithms which are illustrated in the next section.

4.5 Possible Applications in Cryptography

The fact that the simple underlying rules of the CA can be efficiently implemented and repeated applications of these simple rules can demonstrate complex behaviors have lured researchers to develop CA based ciphers [140, 141, 142, 143, 144, 145, 137, 138]. However the previous efforts have been successfully cryptanalyzed. The reasons behind the negative results should not be attributed to the CA, which on the contrary is a wonderful machine conducive for cipher design. The properties discovered and characterized in the chapter may be used to develop or construct new encryption and key agreement algorithms. For the purpose of completeness we conclude with the application of the generalized CA to the construction of key agreement protocols and Block Ciphers.

4.5.1 Key Agreement Protocols

Key agreement protocols are mechanisms which establish the keys before the onset of private key algorithms. The private key algorithms use the key which is settled

between the legitimate sender and receiver through a key agreement protocol. The shared key or secret is derived by two or more parties as a function of the information provided by each party. Ideally the key should not be derivable by either of the parties alone. Such partial key algorithms thus establish key on the fly. Key freshness is important, that is the key should change frequently. Thus less data is encrypted by the same key, thus the eavesdropper has lesser probability of success. Further the amount of damage done due to the revelation of a key is reduced if the key is frequently changed.

The class of the group CA characterized with the matrix T_g (characterized in this chapter) exhibit an interesting agreement property which may be used to vary the session key fast at a very low overhead. **Fig. 4.4** shows the state space cycles generated by the transformation T_g. $P1$ is an initial point of agreement of two parties. From **lemma 4.12** it is evident that if X is processed according to the transformations, $T_g^a R1 R2 T_g^b$ or $T_g^b R2 R1 T_g^a$, then the state X follows two distinct paths and finally converge at a point P_2, which is the second point of agreement.

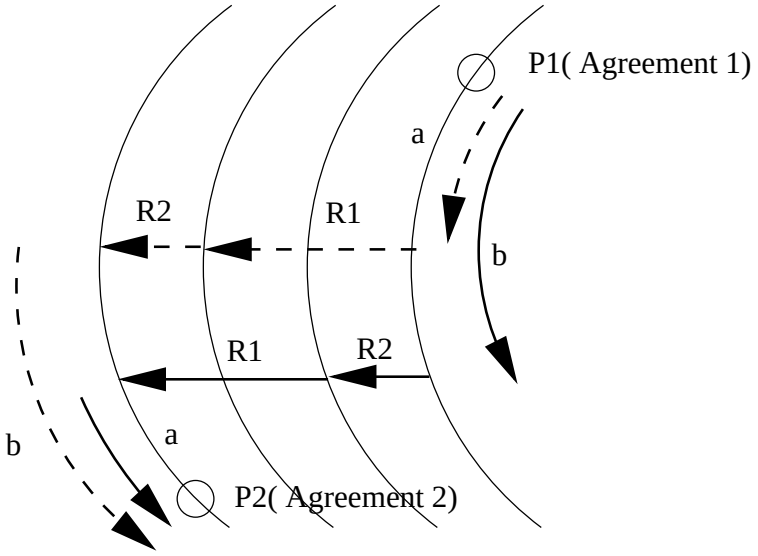


Figure 4.4: **Agreement Property of the State Spaces**

The agreement property may be used to vary an agreed key frequently at a low overhead. Let A and B be two parties with an agreed key K_1. In order to alter the key, A generates $r_A \in_R [0, l-1]$ and computes $x_A = R1[T_g^{r_A}(K_1)] + f(K_1)$, where f is a one-way function. Here l is the length of the cycles in the state transition of the generalized automata and $R1$ is one of the rules of the generalized complemented automata.

Likewise, B generates $r_B \in_R [0, l-1]$ and computes $x_B = R2[T_g^{r_B}(K_1)] + f(K_1)$.

$R2$ is the other rule described in the previous section.

After x_A and x_B are exchanged both the principals compute the new session key using an agreement property. The new session key is $K_2 = R1[T_g^{r_A}\{R2(T_g^{r_B}(K_1)\}]$. The agreement property can be formally stated with the following theorem.

Theorem 4.7 *A computes K_2' from the knowledge of x_B (the public information), r_A (the private information) and the initially agreed key K_1. B computes K_2'' from the knowledge of x_A (the public information), r_B (the private information) and the initially agreed key K_1. Then $K_2' = K_2''$, thus they both agree on a new derived key.*

Proof: The proof follows directly from the characterization presented in the previous section. A calculates $K_2' = T_g^{r_A}[R1\{R2[T_g^{r_B}(K_1)]\}]$ and B computes $K_2'' = T_g^{r_B}[R2\{R1[T_g^{r_A}(K_1)]\}]$.

From the properties of the generalized automata, $K_2 = K_2' = K_2''$. Thus they both agree on a new key K_2. □

The generalized automata helps to introduce randomness into the session key and the smaller cycles make the process fast. Finally the pseudo-randomness of x_A and x_B has been verified using standard tests for randomness. In **Fig. 4.5** the results of the avalanche analysis have been presented. The X-axis shows the number of bits affected, while the Y-axis shows the corresponding frequencies. The Avalanche Test is based upon the fact that a 1-bit change in the plain text should change half of the bits in the ciphertext[105]. A pair of input sequences have been taken and the function is applied upon both of them. The number of bit-differences in the output sequence are noted. The mean of the distribution should be $n/2$ and the standard deviation, $\sqrt{npq}$, where n is the block-size and $p = q = 1/2$ (binomial probability). The simulation has been performed on a huge set of data with $n = 128$ and on the average 60 bits ($\approx 128/2$) get affected. The result thus supports the criterion.

However the above sequence of operations depends on the assumption that a session key has already been established. In order to achieve the initial key agreement we use a specially designed Cellular Automata structure and is discussed in **chapter 8**.

4.5.2 Block Ciphers

Diffusion is a necessary step for the security of block ciphers [146]. The Diffusion Layer (D-Box) may be efficiently realized through the generalized complemented Cellular

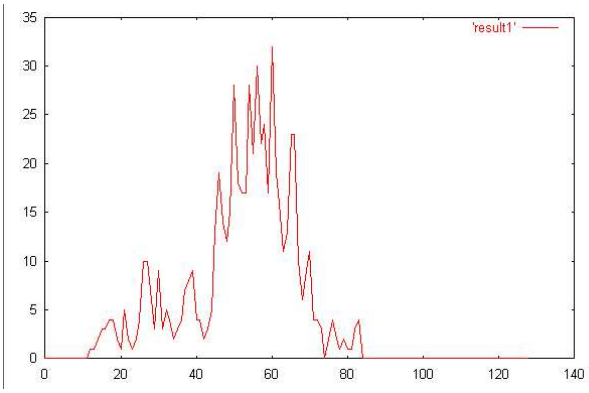


Figure 4.5: **Avalanche Test Results**

Automata, characterized by the transition matrix T_g. The cyclic property of the CA leads to the fact that the same transformation may be used both to encrypt as well as to decrypt the data. For example, if l is the group length of the complemented CA, we have $T_g^l = I$. Thus if the D-Box is characterized by the transformation $D = T_g^{l/2}$, the same transformation performs both encryption and decryption. In this case two points may be observed:

1. The value of l should be constant for a fixed n. In other words all the cycles should be of equal length and independent of input data.

2. The length of the cycles should be linear with the number of cells, which in turn is proportional to the block size of the cipher. Since, the block size decides the security parameter of the cipher, it has to be kept large. The linear dependency of the cycle length and the block size (n) is required to make both the encryption and decryption operations fast (as the number of clock cycles required both to encrypt and decrypt are linear with n).

The characterized Cellular Automata satisfies both the requirements. However $D = T_g^{l/2}$ cannot be used because of Avalanche Property [105]. In order to obtain good Avalanche effect the matrix A of T_g is to be appropriately chosen and the encryption and decryption algorithms should be made respectively:

$$\begin{aligned} T_g(enc) &= T_g^{l/2-1} \\ T_g(dec) &= T_g^{l/2+1} \end{aligned}$$

Thus the diffusion layer helps in developing transformations which can be used both to encrypt and decrypt data. Also the component has good Avalanche Property as elaborated in[138].

However the diffusion obtained by the linear layer has to be combined appropriately with non-linear components (often referred to as S-Boxes) to protect the block cipher against Linear and Differential Cryptanalysis. However the non-linearity should not disturb the cyclic property and hence the self-invertibility of the generalized complemented CA. We propose to construct the round as $r = f^{-1} \circ c^i \circ f$, where f is the non-linear layer which has an inverse f^{-1}. Here c is the generalized CA and i the number of clock cycles it is applied.

One of the interesting properties of the construction is that the structure can be programmed easily to perform both encryption and decryption. Encryption is achieved when $i = n-1$, while the round performs decryption when $i = n+1$. Apart from having self-invertibility, the non-linear components do not disturb the cyclic nature of the linear CA as the cyclic structure of the transformation $T_1 = c^i$ is the same as that of $T_2 = r = f^{-1} \circ c^i \circ f$. Further, the round has a fast forwardness

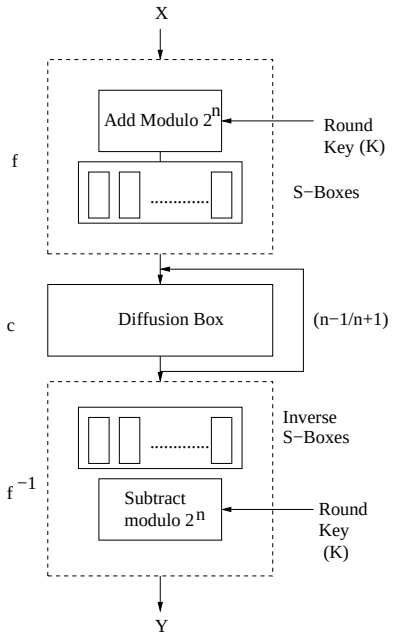


Figure 4.6: **The CA based round**

property [137] resulting in high speed ciphers.

Fast Forwardness: The transformation T_2 has the advantage of fast forwardness. This may be observed as follows:

$$\begin{aligned} T_2^m(X) &= (f.T_1.f^{-1})^m \\ &= f(T_1(f^{-1}(f(T_1(f^{-1}(\ldots(m \text{ times}) f(T_1(f^{-1})))...)(X) \\ &= fT_1^m f^{-1}(X) \end{aligned}$$

Thus the transformation T_2 can be made to iterate without requiring the non-linear function to be iterated. Since the non-linear function is more computation intensive this reduces the cost of the operation and is amenable to efficient implementations.

Below we present construction of the non-linear S-Boxes using a very simple non-linear rule. The claim is that although much stronger S-Boxes are possible with Cellular Automata (as we show subsequently), here we present an approach to build S-Boxes using extremely simple underlying rules. We show that this first level attempt to generate S-Boxes is quite interesting and also satisfy some of the important properties of a good S-Box.

A Cursory attempt to generate S-Boxes using Simple Rules

In order to construct the S-Box using Cellular Automata, we first require a non-linear Cellular Automata rule which is reversible. Also, the Cellular Automata based S-Box should satisfy the various properties required for a robust S-Box design [72]. A skewed version of rule 30 is:

$$x_{t+1}^i = x_t^i \oplus (x_t^{i+1} + x_t^{i+2}).$$

The inverse rule for the non-linear CA may be obtained, as the above rule is invertible, unlike rule 30 CA. However, a serious flaw with it is that the last bit in it passes unchanged. To avoid this, we propose to use "cycles" of this simple rule. After each cycle the output is completely reversed, that is the last bit becomes the first, the second last the second and so on. It is interesting to observe that the robustness of the thus developed S-Box is a function of the number of clock cycles. We fix the number of cycles of the simple non-linear rule required to obtain a cryptographically strong S-Box. Indeed, we show that the robustness of the S-Box is comparable to some of the strongest S-Boxes obtained in the paper of [72], with the advantage that the implementation is very simple. The idea of repeating simple rules to obtain the behavior is also in conformation with the concept envisaged by [135].

Table 4.3: **Non-linearity for different bits and rounds for the S-Box**

No of bits	Number of cycles	Non-linearity (N_f)	Maximum non-linearity (N_f^{max})	Ratio (N_f/N_f^{max})
4	4	4	6	0.67
4	8	4	6	0.67
8	4	64	120	0.53
8	8	82	120	0.68

Evaluation of the S-Box

The S-Box is a crucial component of Substitution Permutation Networks. The S-Boxes are supposed to be defiant against Linear Cryptanalysis (LC) and Differential Cryptanalysis (DC). For this they should satisfy various properties like high non-linearity, balancedness, robustness against differential cryptanalysis and small biases of linear approximations.

Non-linearity and Balancedness:

First, the non-linearity of the S-Box output bits should be high and also they should represent a balanced boolean function. If we consider an m bit S-Box, the non-linear CA (mentioned earlier) gives us a permutation on V_m, that is, we have an invertible mapping. So the enumeration of the truth tables of the output bits show that all the possible elements of V_m appear. Thus the truth table of each output bit has an equal number of zeros and ones, resulting in the balancedness of the output function. Now, a balanced boolean function cannot attain the highest non-linearity, but it should be close to the maximum non-linearity value. As already mentioned in **chapter 3** the maximum non-linearity value for a function on V_m is $2^{m-1} - 2^{m/2-1}$, when m is even. As described before we iterate the simple non-linear rule over some "cycles". Our observation is that as the number of cycles increase the non-linearity of the S-Box varies and gradually approaches its maximum value. The non-linearity of the S-Box for different number of bits and cycles is tabulated in **table 4.3**.

Based on the above result we choose to construct a S-Box on 8 bits. Thus each bit is a boolean function of 8 bits and has a high non-linearity. We, next perform a Linear Cryptanalysis and show that the biases obtained are less. In order to facilitate the representation we tabulate the result for a 4 bit S-Box, although the 8 bit Linear Approximation Table is even better.

Linear Cryptanalysis:

Linear Cryptanalysis essentially deals with the probability of approximating the input and output of non-linear functions, used in the block cipher with linear expres-

sions [147, 148]. The approach in linear cryptanalysis is to determine expressions of the form below which have a high or low probability of occurrence [147, 148]

Let us consider an expression of the form:

$< X_{i1} \oplus X_{i2} \oplus \ldots X_{iu} > \oplus < Y_{j1} \oplus Y_{j2} \ldots \oplus Y_{jv} >= 0$

where X_i represents the i-th bit of the input $X = [X_1, X_2, \ldots]$ and Y_j represents the j-th bit of the output $Y = [Y_1, Y_2, \ldots]$. This equation represents the exclusive OR of u input bits and v output bits.

If the bits are chosen randomly then the above approximated linear expression will hold with probability 1/2. It is the deviation from the probability of 1/2 (bias) for an expression to hold that is exploited in linear cryptanalysis: the further away a linear expression is from holding with a probability of 1/2, the better the cryptanalyst is able to apply linear cryptanalysis. We thus prepare a linear approximation **table 4.4** for the non-linear S-Box of the cipher round. Each element in the table represents the number of matches between the linear equation represented in hexadecimal as "Input Sum" and the sum of the output bits represented in hexadecimal as "Output Sum" minus 8. The hexadecimal value representing a sum, when viewed as a binary value indicates the variables involved in the sum. As we can see from **table 4.4** the biases are small and thus the corresponding equations will offer no considerable help for the cryptanalysis. This table is comparable to the table that we obtain while doing a similar analysis for the S-Boxes of DES.

Differential Cryptanalysis:

Next, we evaluate the robustness of the S-Box against Differential Cryptanalysis (DC). Differential cryptanalysis takes the advantage of entries with high values in the difference distribution tables of S-Boxes employed by block ciphers. The difference distribution table for a $n \times s$ S-Box is a $2^n \times 2^s$ matrix. The rows of the matrix, indexed by the vectors in V_n, represent the change in the input, while the columns, also indexed by the vectors in V_n represent the change in the output of the S-Box.

An entry in the table indexed by (ΔX, ΔY) indicates the number of input vectors which when changed by ΔX (bitwise XOR), result in a change in the output by ΔY (bitwise XOR).

From the definition of robustness against DC, stated in **chapter 3**, it is evident that the values of L and R have to be less. That is, in addition to the requirement of having no large values, the difference distribution table of an S-Box should also contain as less non-zero entries as possible in its first column [72].

We again observe that the robustness against Differential Cryptanalysis varies with the number of "cycles" of the S-Box. The robustness of the non-linear transformation of our cipher for different values of the number of bits in a block and the number of cycles are shown in **table 4.5**. It follows from the above tables that among all the

Table 4.4: **Linear Approximation Table for 4 bit S-Box**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	-2	2	0	0	2	2	4	0	-2	-2	4	0	2	-2	0
2	0	2	2	0	2	0	-4	-2	0	2	-2	4	2	0	0	2
3	0	-4	0	0	2	-2	2	2	0	4	0	0	2	-2	2	2
4	0	0	2	2	2	2	0	0	2	-2	4	0	0	-4	-2	2
5	0	2	0	2	2	0	-2	4	2	0	-2	-4	0	2	0	2
6	0	2	-4	2	0	-2	0	2	2	0	2	4	-2	0	2	0
7	0	0	-2	2	0	0	2	-2	2	-2	0	0	6	2	0	0
8	0	0	0	0	-2	2	2	-2	4	4	0	0	-2	2	-2	2
9	0	-2	2	0	2	0	0	-2	0	-2	2	0	-2	4	4	2
10	0	2	2	0	4	-2	2	0	0	2	2	0	0	2	-2	-4
11	0	4	0	0	0	0	4	0	-4	0	0	0	0	0	0	4
12	0	0	2	2	-4	0	-2	2	-2	2	4	0	2	2	0	0
13	0	2	0	-6	0	2	0	2	2	0	2	0	2	0	2	0
14	0	2	4	2	-2	0	2	0	2	0	-2	0	0	-2	4	-2
15	0	0	-2	2	2	6	0	0	-2	2	0	0	0	0	2	-2

values, the ratios for non-linearity and ϵ-robustness against differential cryptanalysis are the highest when the block size is 8 bits and the number of cycles is 8.

The performance of the S-Box, with respect to the powerful Differential Cryptanalysis may be compared with that of the some of the standard S-Boxes in literature (**table 4.6**). The table shows that the robustness of the DES S-Boxes are very poor with respect to that of the S-Boxes constructed in [72]. We see that the S-Box constructed out of the simple non-linear rule of a CA is comparable to that presented in [72]. The elegance of the CA based S-Box is that it is simple and easy to implement.

Table 4.5: **ϵ-robustness against Differential Cryptanalysis**

Number of bits	Number of cycles	ϵ	L	R	Maximum $\epsilon(\epsilon_m)$ value of	Ratio(ϵ/ϵ_m)
4	4	0.6250	6	0	0.875	0.71
4	8	0.5000	8	0	0.875	0.57
8	4	0.8125	48	0	0.992	0.82
8	8	0.9297	18	0	0.992	0.94

Table 4.6: **Comparison of robustness against Differential Cryptanalysis**

S-Box	Robustness (ϵ)
S_1 (DES)	0.316
S_2 (DES)	0.363
S_3 (DES)	0.316
S_4 (DES)	0.469
S_5 (DES)	0.387
S_6 (DES)	0.367
S_7 (DES)	0.340
S_8 (DES)	0.328
[72]	0.875
[72]	0.96875
[72]	0.992
CA based S-Box	0.9297

4.6 Conclusion

In summary, the present work investigates the state spaces of a class of complemented cellular automaton. New properties have been proved which relate the state spaces of the cellular automaton. The properties help to use the entire state space. The characterization has been extended to a more general class of CA structures. In this chapter it is shown that the state transitions of the generalized CA has an agreement property which may be applied to vary the session key generated by a key agreement algorithm at a fast rate with a low overhead. Further a technique has been proposed to develop a self invertible round of a block cipher whose diffusion layer may be implemented through the generalized CA. The chapter also presents a novel way to introduce non-linearity into the diffusion layer through repeated applications of a non-linear rule of a CA. Results have been furnished to demonstrate that the non-linear layers (S-Boxes) are cryptographically robust. However Cellular Automata may be applied more systematically to develop programmable S-Boxes with provable security margins, as discussed in the next chapter.

Chapter 5

CASBox : A Programmable Structure to Generate S-Boxes

5.1 Introduction

The security of block ciphers largely depends on the cryptographic robustness of the S-Boxes. Thus the construction of good S-Boxes are an extremely important component of cipher design. In [149] Gordon and Retkin first focused on the statistical properties of random, reversible S-Boxes. In literature subsequently several works [68, 69, 70, 71] have been published in defining the desirable properties of S-Boxes. However the drawbacks of all these proposals were pointed out in [72]. The main weaknesses were that the component functions of these S-Boxes were quadratic and thus could be vulnerable to many classic as well as the recent algebraic attacks. Further these techniques are based on permutations and thus have an equal number of input and output bits. The suggestion to drop an appropriate number of component functions from a permutation yields an S-Box with lesser output bits but without any guarantee on its robustness against Differential Cryptanalysis. None of these constructions satisfy the Strict Avalanche Criterion (SAC), which is an important property of modern day ciphers. One of the constructions proposed in [73] is based on Maiorana-McFarland method and is built out of Linear Feedback Shift Registers (LFSRs). Apart from the above drawbacks the class of circuits built around LFSRs can be found to have the following inherent disadvantages (i) irregularity of the interconnection structure (ii) larger delay and (iii) lack of modularity and cascadability. Also the resultant S-Box was not balanced and had the restriction that the first half of the input that goes to the LFSRs was not zero. This restricts the usage of the generated S-Boxes. In [72] the authors describe various properties of cryptographi-

cally robust S-Boxes and also discusses their construction. However, the work does not describe how to implement the S-Boxes in real life. It has been shown in [76] that all the proposed constructions for good candidates for S-Boxes are based on, what is known as the *Maiorana-McFarland Construction*, refer **section 2.2.3**. All the prior works[77, 78, 79, 80, 84, 85] are based on this principle. Although several theoretical works have been done in the construction of S-Boxes, there are relatively few works on efficient hardware or software for the generation of cryptographically robust S-Boxes. While the construction of strong S-Boxes is an important aspect of cryptographic design, the other important aspect is the design, analysis and implementation of large strong S-Boxes. Although many desirable properties of S-Boxes have been studied, it is also a challenge to develop efficient scalable architectures for S-Boxes.

In [69] a method was presented for $n \times n$ S-Box design. However, for the S-Box created by this method, its inverse S-Box is almost completely linear (it has only one non-linear function) and its diffusion property cannot be ensured. In [81] Adams and Tavares proposed a design methodology for $n \times n$ S-Boxes. Since, the method is an exhaustive search method the complexity of the method grows as the value of n increases. The method of [150] makes use of near bent Boolean functions of five variables to create 5×5 S-Boxes in order to resist differential attacks [29]. However the method is restricted for S-Boxes whose input bits are odd. In [82] a practical S-Box design has been described where it has been stated that the construction of large cryptographically strong S-Boxes are difficult and requires huge computational resources. In [83] a method has been described for obtaining cryptographically strong 8×8 S-Boxes. However the performance of the generated S-Boxes are much inferior compared to possible S-Boxes that can be constructed for such dimensions. Recently an interesting construction of S-Boxes based on the Maiorana-McFarland methodology has been proposed in [84]. The software implementation of the proposed method has been presented in [85] where the authors claim that the work was the first practical software implementation of a general framework to generate cryptographically robust $n \times m$ S-Boxes. However, the construction requires table lookup of the order 2^m, which makes it infeasible for a hardware implementation where m is large. Indeed we require bijective S-Boxes where $m = n$ in block ciphers and the current algebraic attacks obviate the requirement that the size of the S-Boxes are equal to or more than eight.

In this work the Theory of Cellular Automata has been applied, to the best of our knowledge for the first time to generate S-Boxes as good as those of [72] on hardware. The work discusses how a special class of linear group CA, called the maximum length CA can generate S-Boxes which satisfy the requirements and properties described in [72]. In short the generated S-Boxes are balanced, satisfy Strict Avalanche Criterion, have high non-linearity of the component functions as well as their non-zero linear

combinations, have high algebraic degree and are robust against linear and differential cryptanalysis. The work shows that the CA based S-Box, called CASBox, is programmable, modular, and generates a large number of S-Boxes all of which are equally cryptographically strong. The architecture proposed in the chapter has been implemented on a Xilinx XCV-1000 platform and the results show that the structure is efficient and scalable. Compared to the work done by S. Mister and C. Adams [82] the proposed CA based S-Box construction is extremely efficient with respect to time, due to the inherent parallelism in Cellular Automata transformations. Also as the chosen maximum length CA has a three neighbourhood cell [135, 129] the length of the interconnects would be less compared to a LFSR based S-Box [73], a feature helpful for VLSI implementations[151]. Further it may be pointed out that since the S-Boxes generated can be simply reconfigured by programming the number of cycles of the internal CA, the CASBox may be used as a real life implementation of a randomly selected S-Box leading to the realization of a true random cipher [152].

The chapter is organized as follows: The overall design of the CASBox has been described in *section 5.2*. In *section 5.3* a closed set of invertible linear transformations have been constructed using Cellular Automata rules. The CASBox have been mathematically formulated in *section 5.4* and its various characteristics have been described in *section 5.5*. The cryptographic strengths of the CASBox have been analyzed in *section 5.6*. A specimen example of 8×8 S-Box has been presented in *section 5.7* while FPGA implementations of CASBox have been presented in *section 5.8*. *Section 5.9* outlines how CASBox helps to prevent attacks on SPN ciphers. The work is concluded in *section 5.10*.

5.2 Construction of CA Based S-Box (CASBox)

The present section constructs an $n \times n$ mapping using 2^{n-k} $(k > \frac{n}{2})$ k cell maximum length Cellular Automata. The characteristic matrix of each group CA is represented by a $k \times k$ matrix with elements in $GF(2)$ (i,e 0 or 1). **Fig. 5.1** depicts the overall construction of the $n \times n$ mapping which is used for the CA based S-Box (CASBox). The linear Cellular Automata are represented by the linear operators $L_0, \ldots, L_{2^{n-k}-1}$. The input z to the S-Box is a vector of n bits and has two parts y and x. Thus, $z = (y, x)$ where x is a vector of size k bits and y is that of $(n-k)$ bits. The input seed to each Cellular Automaton (CA) is a k bit vector $x = \{x_1, \ldots, x_k\}$ where each element belongs to $GF(2)$. Each linear operator L_i $(0 \leq i < 2^{n-k})$ transforms a k bit input to a k bit output. The output of the 2^{n-k} linear Cellular Automata are passed through a multiplexer, controlled by a $(n-k)$ bit vector $y = \{y_1, \ldots, y_{n-k}\}$ which serves as the select line. The select line is used to multiplex out one of the outputs of 2^{n-k} Cellular Automata. The $(n-k)$ bit vector y is processed by a non-linear

balanced permutation nl, to output $nl(y)$. The k bit output is then transformed by an operator B (which includes a linear transformation A and xoring with $nl(y)$) to map the k bits into an n bit output. The input to the CASBox is $z = (y, x)$ and the output is denoted by $Q(z) = \{q_1(z), \ldots, q_n(z)\}$, where $q_i(z)$ are the component functions of the CASBox. The properties satisfied by the overall construction are:

1. The S-Box is robust against Differential and Linear Cryptanalysis.
2. Any non-zero linear combination of the component boolean functions have high degree of non-linearity.
3. The sum of any subset of the component functions are non-linearly balanced functions and hence the component functions are all uncorrelated[153].
4. The structure of the CASBox is extremely programmable. For a fixed value n and k the technique described in the chapter gives ${}^{2^k-1}P_{2^{n-k}}$ S-Boxes which are regular many-to-one mappings from vector space of n bits to that in k bits (**theorem 5.5**). By regular mapping it is meant that for each k bit vector output there are exactly 2^{n-k} input vectors possible. Simply changing the number of clock cycles of the same CA transformation various S-Boxes are obtained, all of which are cryptographically robust.
5. The $n \times n$ S-Box is invertible and satisfies SAC (Strict Avalanche Criterion).

5.3 The Chosen Set of Cellular Automata Transformations

In this section we present the construction of a set of group CA generated by a maximum length CA [129]. The linear transformations of the S-Boxes are the elements of the constructed set (S). The set is analyzed first and used in the subsequent development of the CASBox.

5.3.1 Set of CA Transformations

The characteristic matrix of a k cell CA is represented by a $k \times k$ matrix T with elements in $GF(2)$. When $|T| = 1$ then the CA is a group CA. We choose a k cell maximum length CA. The characteristic matrix of the maximum length group CA is denoted by a $k \times k$ matrix T_k, with the elements also in $GF(2)$. In the maximum length CA all the non-zero elements lie in one cycle [129]. Hence, except the all zero

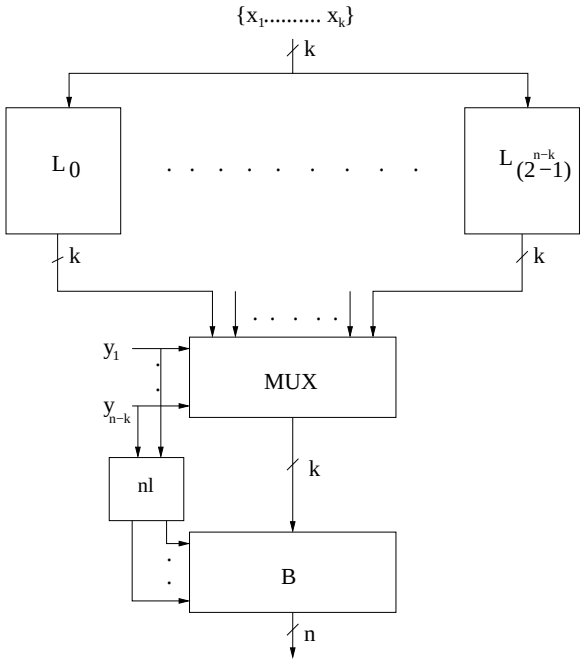


Figure 5.1: **Structure of the CA Based S-Box (CASBox)**

vector (which maps to itself) the other $2^k - 1$ non-zero elements form a cycle in the state transition diagram of the CA. Thus, $T_k^{(2^k-1)} = I$, where I is a $k \times k$ identity matrix. Thus, the set $S = \{I, T_k, \ldots, T_k^{2^k-2}\}$ contains $(2^k - 1)$ invertible matrices of dimension $k \times k$ and with elements in $GF(2)$. The properties of the set S are described next.

5.3.2 Properties of the set S

Lemma 5.1 *The transformation $T_k^{i-1} \in S$ is invertible ($1 \leq i < 2^k$)*

Proof: We denote the determinant of the matrix T_k^{i-1} by $det(T_k^{i-1})$. Since, T_k represents the characteristic matrix of a group CA $det(T_k) = 1$ (by **theorem 3.1**). Thus, $det(T_k^{i-1}) = (det(T_k))^{i-1} = 1 \neq 0$, since $det(AB) = det(A).det(B)$, where A and B are square matrices.

Hence T_k^{i-1} is invertible. □

Thus S is a set of invertible matrices (linear transformations) which form characteristic matrices of various linear group Cellular Automata.

Lemma 5.2 *Set S is closed under addition modulo 2*

Proof: Let, $T_k^{i-1}, T_k^{j-1} \in S$, where $i \neq j$ and $1 \leq (i,j) < 2^k \Rightarrow$ for an input vector X,

$Y = (T_k^{i-1} \oplus T_k^{j-1})X = T_k^{i-1}(X) \oplus T_k^{j-1}(X)$.

From the definition of maximum length group CA, it is evident that when $i \neq j$, $T_k^{i-1}(X) \neq T_k^{j-1}(X)$.

Thus, $Y \neq 0$ and so must lie in the maximum length cycle. Thus, it can be expressed as some $T_k^{w-1}(X)$ $(0 < w < 2^k)$.

Clearly, $det(T_k^{w-1}) = 1$ and hence is invertible and thus $T_k^{w-1} \in S$. This completes the proof. □

Lemma 5.3 *If $T_k^{i-1}, T_k^{j-1} \in S$, rows of the matrices T_k^{i-1} and T_k^{j-1} are pairwise distinct when $i \neq j$.*

Proof: Since, $(T_k^{i-1} \oplus T_k^{j-1}) = T_k^{w-1} \in S \Rightarrow (T_k^{i-1} \oplus T_k^{j-1})$ is invertible.

Thus, rank $(T_k^{i-1} \oplus T_k^{j-1}) = k$ and so no row of $(T_k^{i-1} \oplus T_k^{j-1})$ is the zero vector of size $k \Rightarrow$ rows of T_k^{i-1} and T_k^{j-1} are not equal. □

5.4 Mathematical Formulation of CASBox

The input to the CASBox is denoted by $z = (y, x)$ where $y \in V_{n-k}$ and $x \in V_k$ (vector space of n tuples of elements from $GF(2)$ is denoted by V_n). Here, $k > \frac{n}{2}$. The output of CASBox is denoted by $Q(z) = (Q_2(z), Q_1(z))$, where $Q_2(z) \in V_{n-k}$ and $Q_1(z) \in V_k$.

5.4.1 Determination of $Q_1(z)$

The 2^{n-k} linear transformations chosen from the set S are indicated by the linear transformations $L_0, \ldots, L_{2^{n-k}-1}$.

The linear transformations are the following $k \times k$ matrices:

$$L_0 = \begin{pmatrix} l_{01} \\ \vdots \\ l_{0k} \end{pmatrix}, \ldots, L_i = \begin{pmatrix} l_{i1} \\ \vdots \\ l_{ik} \end{pmatrix}, \ldots, L_{2^{n-k}-1} = \begin{pmatrix} l_{2^{n-k}-1,1} \\ \vdots \\ l_{2^{n-k}-1,k} \end{pmatrix}$$

The elements of the matrices L_i $(0 \leq i < 2^{n-k})$ are k length vectors, l_{ij}, $(1 \leq j \leq k)$. The elements of the vectors are $GF(2)$ elements, hence the final matrices are all $k \times k$ matrices with elements $(0, 1)$. Also, the vectors of a particular matrix are

linearly independent, as the matrices are invertible. For example, if the value of k is 5, then the matrix L_0 could be represented as the following 5×5 matrix:

$$L_0 = \begin{pmatrix} l_{01} \\ l_{02} \\ l_{03} \\ l_{04} \\ l_{05} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

It may be noted that the elements of the matrix L_0 are $\{0,1\}$ vectors of length 5 and they are linearly independent. For instance, the values of the elements $l_{01} = \{11000\}$, $l_{02} = \{11100\}$, $l_{03} = \{01010\}$, $l_{04} = \{00101\}$ and $l_{05} = \{00011\}$.

The $Q_1(z)$ component of the matrix is the concatenation of the output of 2^{n-k} Cellular Automata selected according to the control line of the multiplexer.

Thus mathematically the vector $Q_1(z) \in V_k$ is:

$$Q_1(z) = \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y) L_\sigma(x) \tag{5.1}$$

In the above notation [72] the vector space of k tuples of elements from $GF(2)$ is denoted by V_k. $D_\sigma(.)$ is a function on V_{n-k} defined by $D_\sigma(y) = (\bar{i}_1 \oplus y_1) \ldots (\bar{i}_{n-k} \oplus y_{n-k})$ where $\sigma = (i_1, \ldots, i_{n-k})$ and $y = (y_1, \ldots, y_{n-k})$.

5.4.2 Determination of $Q_2(z)$

The other part of the output indicated by the vector $Q_2(z)$, which is of $(n-k)$ bits is obtained as follows (i.e $Q_2(z) \in V_{n-k}$):

$Q_2(z) = AQ_1(z) \oplus nl(y)$, where the matrix A is non-degenerate

$$= \begin{pmatrix} a_1 \\ \vdots \\ a_{n-k} \end{pmatrix} Q_1(z) \oplus nl \begin{pmatrix} y_1 \\ \vdots \\ y_{n-k} \end{pmatrix}$$, where nl is a non-linear permutation on V_{n-k}. Thus the function $nl(y)$ has $(n-k)$ components, $(nl_1(y), \ldots, nl_{n-k}(y))$

each of which is a non-linear boolean transformation on $y = (y_1, \ldots, y_{n-k})$.

The vectors a_i $(1 \leq i \leq (n-k))$, which constitutes the matrix A are linearly independent.

Thus, the output $Q(z) \in V_n$ may be represented as:

$Q(z) = \begin{pmatrix} Q_2(z) \\ Q_1(z) \end{pmatrix} = BQ_1(z)$ (B denotes the transformation depicted in **Fig. 5.1**)

$$= \begin{pmatrix} A_{(n-k)\times k} \\ I_{k\times k} \end{pmatrix} Q_1(z) \oplus \begin{pmatrix} nl \\ 0_{k\times(n-k)} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_{n-k} \end{pmatrix}$$

Thus the output $Q(z)$ may be enumerated as $Q(z) = \{Q_2(z), Q_1(z)\}$, where $Q_1(z) = \{q_1(z), \ldots, q_k(z)\}$ and $Q_2(z) = \{q_{k+1}(z), \ldots, q_n(z)\}$.

Hence,

$$\begin{array}{lcl} q_1(z) & = & D_0(y)l_{01}(x) \oplus \ldots \oplus D_j(y)l_{j1}(x) \ldots \oplus D_{(2^{n-k}-1)}(y)l_{(2^{n-k}-1,1)}(x) \\ \vdots & & \vdots \\ q_i(z) & = & D_0(y)l_{0i}(x) \oplus \ldots \oplus D_j(y)l_{ji}(x) \ldots \oplus D_{(2^{n-k}-1)}(y)l_{(2^{n-k}-1,i)}(x) \\ \vdots & & \vdots \\ q_k(z) & = & D_0(y)l_{0k}(x) \oplus \ldots \oplus D_1(y)l_{jk}(x) \ldots \oplus D_{(2^{n-k}-1)}(y)l_{(2^{n-k}-1,k)}(x) \end{array} \quad (5.2)$$

The matrix $A = \begin{pmatrix} a_1 \\ \vdots \\ a_{n-k} \end{pmatrix} = \begin{pmatrix} a_{11} & \ldots & a_{1k} \\ \ldots & \ldots & \ldots \\ a_{n-k,1} & \ldots & a_{n-k,k} \end{pmatrix}$

The remaining part of the output is enumerated after the following discussion on the non-linear permutation on V_{n-k}.

5.4.3 Non-linear permutation on V_{n-k}

As discussed, CASBox uses a non-linear permutation, nl operating in V_{n-k}.

$$nl : V_{n-k} \rightarrow V_{n-k}$$

Thus it is evident that nl comprises of $(n-k)$ non-linear boolean functions, $(nl_1, \ldots, nl_{n-k})$, which satisfy the following conditions:

1. Each function $(nl_1, \ldots, nl_{n-k})$ is non-linear and balanced
2. Any non-zero linear combination of $nl_1, \ldots, nl_{n-k}$ is non-linear and balanced.
3. nl is a permutation and hence is invertible

Possible constructions of the non-linear function nl

In the construction of the non-linear function nl we shall use two basic methods. The first technique is based on the inverse operation in Galois field in $GF(2^{n-k})$.

- **The mapping $nl(y) = y^{-1}$, where $y \in GF(2^{n-k})$:** The function has been treated in details in [154]. The mapping is:

$$nl(y) = \begin{cases} x^{-1}, & \text{if } x \neq 0 \\ 0, & \text{if } x = 0 \end{cases}$$

The properties of the mapping may be enlisted as follows:

1. The non-linearity of nl is greater than or equal to $2^{n-k-1} - 2^{\frac{(n-k)}{2}}$
2. A mapping is called differentially uniform if for every non-zero input difference and any output difference the number of possible inputs has a uniform upper bound. If $(n-k)$ is odd then the function nl is differentially 2-uniform and if $(n-k)$ is even then the function is differentially 4-uniform.

Example 5.1 *For a* 10×10 *CASBox if one chooses the value of* k *to be* 6*, then* $n - k = 4$*. Thus, we require a non-linear permutation in the space* V_4*. The non-linear permutation thus comprises of* 4 *boolean component functions,* $nl(y) = \{nl_1, nl_2, nl_3, nl_4\}$ *and thus also outputs* 4 *bits.*

If we consider the inverse mapping in the field $GF(2^4)$ *with primitive polynomial* $x^4 + x + 1$ *the boolean equations are:*

$$\begin{aligned} nl_1 &= y_1 \oplus y_2 \oplus y_3 \oplus y_4 \oplus y_1y_3 \oplus y_2y_3 \oplus y_1y_2y_3 \oplus y_1y_3y_4 \\ nl_2 &= y_1y_2 \oplus y_1y_3 \oplus y_2y_3 \oplus y_4 \oplus y_2y_4 \oplus y_1y_2y_3 \\ nl_3 &= y_1y_2 \oplus y3 \oplus y_1y_3 \oplus y_4 \oplus y_1y_4 \oplus y_1y_3y_4 \\ nl_4 &= y_2 \oplus y_3 \oplus y_4 \oplus y_1y_4 \oplus y_2y_4 \oplus y_3y_4 \oplus y_2y_3y_4 \end{aligned}$$

1. *It may be verified that the algebraic degree is* 3 *and the non-linearity of any non-zero linear combination,* $c_1nl_1(y) \oplus c_2nl_2(y) \oplus c_3nl_3(y) \oplus c_4nl_4(y)$ *where* $c_1, c_2, c_3, c_4 \in GF(2)$*, is* 4*.*
2. *Also, if* α *is any non-zero vector in* $GF(2^4)$ *then* $nl(y) \oplus nl(y \oplus \alpha)$ *also belongs to* $GF(2^4)$ *and hence can take* 16 *values. However since* α *is non-zero, the resultant value cannot be zero. Of the remaining* 15 *values,* $nl(y) \oplus nl(y \oplus \alpha)$ *does not take* 8 *values and it assumes* 7 *values in* $GF(2^4)$ *twice each.*

3. The inverse mapping in $GF(2^4)$ is differentially 4 uniform.

- **From the m-sequences generated by a maximal length LFSR:** In this construction we consider a LFSR of size $(n-k)$ with feedback polynomial $p(x)$. The LFSR is loaded with the pattern $(1,\ldots,0)$ and we obtain the $(2^{n-k}-1)$ patterns for each of the $(n-k)$ bits. When the feedback polynomial is primitive, each of these sequences are called m-sequences or maximal length sequences. The truth table of each of the $(n-k)$ bits with an extra zero added at the beginning is converted to an equivalent and-xor form using Reed-Muller expansion[155, 72]. This gives the non-linear permutation, nl where each component function is balanced and non-linear with respect to the input y.

Example 5.2 *For an 8×8 CASBox, when the parameter k is set to the value 5, we require the non-linear permutation in the field V_3.*

Thus we consider an LFSR of 3 bits with feedback polynomial x^3+x+1. We load the LFSR with the initial pattern of $(1,0,0)$. The evolution of the pattern may be observed in **Fig. 5.2**. *The and-xor forms may be obtained from the evolution of the patterns after adding an extra row of all zeroes at the beginning.*

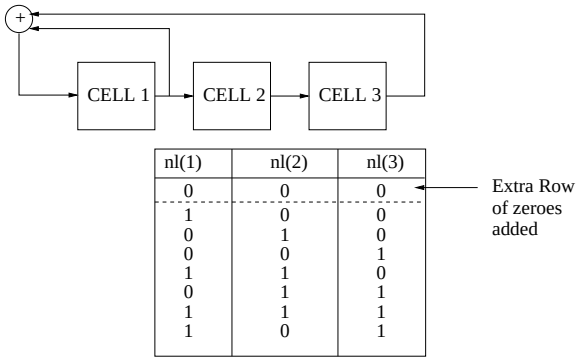


Figure 5.2: **The LFSR based construction of non-linear permutation in V_3**

Minimizing the truth-table shown in **Fig. 5.2** *the final boolean equations are:*

$$\begin{aligned} nl_1 &= y_1 \oplus y_3 \oplus y_2y_3 \\ nl_2 &= y_1 \oplus y_2 \oplus y_1y_2 \oplus y_2y_3 \\ nl_3 &= y_1y_2 \oplus y_2y_3 \oplus y_1y_3 \end{aligned}$$

Thus, the non-linear permutation in V_3 is $nl(y) = \{nl_1, nl_2, nl_3\}$. It may be verified that the final function $nl(y)$ is a non-linear permutation with algebraic degree 2. The non-linearity of any non-zero linear combination is 2, which is the maximum non-linearity for a boolean function in V_3. Also the resultant boolean function is balanced. When y takes all the values in $GF(2^4)$, $nl(y) \oplus nl(y \oplus \alpha)$ assumes 4 values in V_3 twice each. It does not take values of the other 4 vectors.

Having discussed about the construction of the function $nl(y)$ we continue with the description of the remaining $n - k$ bits of the CASBox.

$$\begin{array}{lcl} q_{k+1}(z) & = & [a_{11}q_1(z) \oplus \ldots \oplus a_{1i}q_i(z) \ldots \oplus a_{1k}q_k(z)] \oplus nl_1(y) \\ \vdots & & \vdots \\ q_n(z) & = & [a_{n-k,1}q_1(z) \oplus \ldots \oplus a_{n-k,i}q_i(z) \ldots \oplus a_{n-k,k}q_k(z)] \oplus nl_{n-k}(y) \end{array} \tag{5.3}$$

In the following section the various cryptographic properties of the CASBox have been analyzed.

5.5 Characteristics of the CASBox

The following theorems characterize the CASBox and estimates various cryptographic properties which are required for a robust S-Box.

Theorem 5.1 *The non-linearity of each component function $q_i(z)$ $(1 \leq i \leq n)$ is atleast $2^{n-1} - 2^{k-1}$, where $k > n/2$*

Proof:

Non-linearity of the component functions of $Q_1(z)$:

For $1 \leq i \leq k$,

$$q_i(z) = \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y) l_{\sigma i}(x)$$

As all the rows of the matrices $L_0, \ldots, L_{2^{n-k}-1}$ are pairwise distinct (by **lemma 5.3**), hence $l_{\delta i} \neq l_{\delta' i}$, if $\delta \neq \delta'$. Also, none of the vectors is a zero vector.

Thus, $q_i(z)$ is a boolean function whose truth-table is obtained by the concatenation of the truth-tables of 2^{n-k} non-zero and pairwise distinct k variable linear boolean equations.

Let g be a linear approximation of n variables for $f = q_i(z)$. The linear approximations are represented by the rows of the Sylvester-Hadamard matrix H_n [156, 153]. From the fact that $H_n = H_k \otimes H_{n-k}$, the linear approximations can be represented by the concatenation of 2^{n-k} linear functions, all of which are the same or complement.

Thus, the sequences of f and g are respectively denoted by

$$\begin{aligned} \eta_f &= \{\tau_{l_0}, \ldots, \tau_{l_{2^{n-k}-1}}\} \\ \eta_g &= \{c_0 l_t, \ldots, c_{2^{n-k}-1} l_t\} \end{aligned}$$

where l_t is the sequence of a linear function of k variables (obtained from the rows of H_k) and $c_j \in \{-1, +1\}$, $0 \leq j \leq 2^{n-k} - 1$.

The distance between the two function on V_n, f and g is denoted by $d(f, g) = 2^{n-1} - \frac{1}{2} < \eta_f, \eta_g >$, where $< \eta_f, \eta_g > = \#(f = g) - \#(f \neq g)$ [153].

Here, $\#(f = g)$ denote the number of cases when f and g match and $\#(f \neq g)$ denote the number of cases when f and g does not match.

Since the linear equations $l_{\delta,i}$, $\delta \in V_{n-k}$ are all pairwise distinct, the truth-table of f can match with that of g in atmost one component linear function, say $\tau_{lq} = c_q l_t$. In the other cases, the component function in the truth-table of f and in that of g are different. Thus, they match in half of the cases and does not match in half of the cases.

Therefore,

$$\begin{aligned} < \eta_f, \eta_g > &= \#(f = g) - \#(f \neq g) \\ &= \#_q(f = g) - \#_q(f \neq g) + \sum_{0 \leq u < 2^{n-k}, u \neq q} (\#_u(f = g) - \#_u(f \neq g)) \\ &= 2^k + \sum_{0 \leq u < 2^{n-k}, u \neq q} (0), \\ &\qquad \text{(as it matches } 2^{k-1} \text{ times and does not match } 2^{k-1} \text{ times)} \\ &= 2^k. \end{aligned}$$

Thus, the non-linearity of $f = q_i(z)$ $(1 \leq i \leq k)$is

$$N_f = d(f, g) = 2^{n-1} - \frac{1}{2} < \eta_f, \eta_g >$$

$$= \quad 2^{n-1} - 2^{k-1}$$

Non-linearity of the component functions of $Q_2(z)$:

$$\begin{aligned}
q_{k+i}(z) &= a_{i1}q_1(z) \oplus \ldots a_{ij}q_j(z) \oplus \ldots \oplus a_{ik}q_k(z) \oplus nl_i(y) \\
&= a_{i1}[\bigoplus_{j=0}^{2^{n-k}-1} D_j(y)l_{j1}(x)] \oplus \ldots \oplus a_{ik}[\bigoplus_{j=0}^{2^{n-k}-1} D_j(y)l_{jk}(x)] \oplus nl_i(y), \\
&\qquad\qquad \text{(using } \textbf{equation 5.2}\text{)} \\
&= D_0(y)[\bigoplus_{j=1}^{k} a_{ij}l_{0j}(x)] \oplus \ldots \oplus D_{2^{n-k}-1}(y)[\bigoplus_{j=1}^{k} a_{ij}l_{2^{n-k}-1,j}(x)] \oplus nl_i(y) \\
&= \bigoplus_{m=0}^{2^{n-k}-1} D_m(y)[\bigoplus_{j=1}^{k} a_{ij}l_{mj}(x)] \oplus nl_i(y) \\
&= term_1 \oplus term_2
\end{aligned}$$

It is easy to check that $term_1 = \bigoplus_{m=0}^{2^{n-k}-1} D_m(y)[\bigoplus_{j=1}^{k} a_{ij}l_{mj}(x)]$ is not equal to zero. Since in that case, each of the functions $q_1(z), \ldots, q_k(z)$ become linearly dependent which in turn implies that the rows of the matrices $L_0, \ldots, L_{2^{n-k}-1}$ are linearly dependent. This contradicts the initial construction.

Non-linearity due to $term_1$:

If $L_0 = \begin{pmatrix} l_{01} \\ \vdots \\ l_{0k} \end{pmatrix}, \ldots, L_{2^{n-k}-1} = \begin{pmatrix} l_{2^{n-k}-1,1} \\ \vdots \\ l_{2^{n-k}-1,k} \end{pmatrix}$

and since the matrices are elements of the set S $(L_0 \oplus L_{2^{n-k}-1})$ is also invertible (by **lemma 5.2**).

So, $\text{rank}(L_0 \oplus L_{2^{n-k}-1}) = k$, i.e $\text{rank} \begin{pmatrix} l_{01} \oplus l_{2^{n-k}-1,1} \\ \vdots \\ l_{0k} \oplus l_{2^{n-k}-1,k} \end{pmatrix} = k$

Therefore all the rows are non-zero vectors and are linearly independent.

Thus there does not exist any non-zero vector $(a_{i1}, \ldots, a_{ik})$ such that

$$\begin{aligned}
& a_{i1}(l_{01} \oplus l_{2^{n-k}-1,1}) \oplus \ldots \oplus a_{ik}(l_{0k} \oplus l_{2^{n-k}-1,k}) = 0 \\
\textit{or}, \quad & a_{i1}l_{01} \oplus \ldots \oplus a_{ik}l_{0k} = a_{i1}l_{2^{n-k}-1,1} \oplus \ldots \oplus a_{ik}l_{2^{n-k}-1,k}
\end{aligned}$$

So it is not possible that for any non-zero linear combination two linear functions are pairwise equal. Hence, the non-linearity of the term $term_1$ is $2^{n-1} - 2^{k-1}$.

Non-linearity due to $term_2$:

The non-linearity of $term_2 = nl_i(y)$ $(1 \leq i \leq n-k)$ with respect to the input $z = (y, x)$ is obtained from the non-linearity of the boolean functions of the non-linear permutation with respect to the input y.

Thus, if the maximum non-linearity of the function $nl_i(y)$ is m_{nl} with respect to the input y, then the non-linearity of the function with respect to the input z is $\frac{2^n}{2^{n-k}} m_{nl} = 2^k m_{nl}$.

For example, if $k = n-3$ and if one uses the LFSR based non-linear permutation shown in the previous section for V_3 the non-linearity of the resultant function is $2^{n-3}2 = 2^{n-2}$. Thus the non-linearity of $term_1$ is more than that of $term_2$ as $n/2 < k < n$. More generally, for the inverse based non-linear permutation the non-linearity of the term $nl(y)$ is $2^{n-k-1} - 2^{(n-k)/2}$.

More specifically as $n > k$ we have,

$$\begin{array}{lccc} & n+2 & > & k \\ or, & n+k & > & 2k-2 \\ or, & 2^{(n+k)/2} & > & 2^{k-1} \\ or, & 2^{n-1} - 2^{k-1} & > & 2^k(2^{n-k-1} - 2^{(n-k)/2}) \\ or, & \text{non-linearity } (term_1) & > & \text{non-linearity } (term_2) \end{array}$$

Thus, the non-linearity of the first term is more than that of the second term as $n/2 < k < n$. Hence the resultant non-linearity of the component functions, $q_{k+i}(z)$ $(1 \leq i \leq n-k)$ is atleast $2^{n-1} - 2^{k-1}$. □

Theorem 5.2 *Any component function $q_i(z)$ $(1 \leq i \leq n)$ is balanced.*

Proof: From the previous proof, the i^{th} component function $q_i(z)$ $(1 \leq i \leq k)$ is the concatenation of 2^{n-k} non-zero linear functions. Since each of the non-zero linear functions are balanced, so $q_i(z)$ is also balanced.

The remaining component functions are obtained by the xoring of the boolean functions obtained by the concatenation of 2^{n-k} non-zero linear functions (which is balanced by the previous logic for the first k bits) and a non-linear boolean function. Hence the remaining component functions are also balanced as the xoring of a balanced boolean function with another boolean function (balanced or un-balanced) results in a balanced boolean function. □

Theorem 5.3 *The non-linearity of any non-zero linear combination of the component functions $q_i(z)$ $(1 \leq i \leq n)$ is $2^{n-1} - 2^{k-1}$ $(k > n/2)$ except in $2^{n-k}-1$ cases when the non-linearity is $2^k m_{nl}$ where m_{nl} is the non-linearity of the component functions of the non-linear permutation in V_{n-k}. The resulting functions are always balanced.*

Proof: Let us consider a non-zero linear combination of the component functions. For a non-zero vector $(c_1, \ldots, c_k, c_{k+1}, \ldots, c_n)$ there can be 2^n such linear combinations.

For $1 \leq j \leq 2^n$,

$$
\begin{aligned}
d_j = \bigoplus_{i=1}^{n} c_i q_i \quad &= \quad c_1[\bigoplus_{j=0}^{2^{n-k}-1} D_j(y) l_{j1}(x)] \oplus \ldots \oplus c_k[\bigoplus_{j=0}^{2^{n-k}-1} D_j(y) l_{jk}(x)] \\
&\quad \oplus c_{k+1}[\bigoplus_{j=0}^{2^{n-k}-1} D_j(y)(\bigoplus_{m=1}^{k} a_{1m} l_{jm}(x)) \oplus n l_1(y)] \oplus \ldots \oplus \qquad (5.4) \\
&\quad c_n[\bigoplus_{j=0}^{2^{n-k}-1} D_j(y)(\bigoplus_{m=1}^{k} a_{n-k,m} l_{jm}(x)) \oplus n l_{n-k}(y)] \\
&\quad (\text{using } \textbf{equation 5.2 and 5.3}) \\
&= \quad D_0(y)[\bigoplus_{j=1}^{k} r_j l_{0j}(x)] \oplus \ldots \oplus D_{2^{n-k}-1}(y)[\bigoplus_{j=1}^{k} r_j l_{2^{n-k}-1,j}(x)] \bigoplus_{j=1}^{n-k} c_{k+i} n l_i(y) \\
\text{where, } r_j \quad &= \quad c_j \oplus c_{k+1} a_{1j} \oplus \ldots \oplus c_n a_{n-k,j}
\end{aligned}
$$

Case 1: $d_j \neq \bigoplus_{i=1}^{n-k} c_{k+i} n l_i(y)$

Thus as discussed in **theorem 5.1**, non-linearity of the non-zero linear combinations of the component boolean functions in such a case is due to the concatenation of the 2^{n-k} linear functions. Each of the linear functions $\bigoplus_{j=1}^{k} r_j l_{uj}(x)$ $(0 \leq u \leq 2^{n-k}-1)$ operate on x which belongs to V_k. Similar to the proof in **theorem 5.1** the linear functions are non-zero and also pairwise distinct. That is, there does not exist a non-zero vector $(r_1, \ldots, r_k)$ such that $\bigoplus_{j=1}^{k} r_j l_{uj}(x) = \bigoplus_{j=1}^{k} r_j l_{vj}(x)$, where $u \neq v$.

Hence, using the same logic as in **theorem 5.1** the non-linearity of the non-zero linear combination of the component functions is also $2^{n-1} - 2^{k-1}$.

Also the functions representing the non-zero linear combinations of the component functions are balanced. This is because they are formed by the xoring of the boolean functions which are obtained by the concatenations of non-zero linear functions in V_k, which are themselves balanced and a non-linear term. Since the boolean function obtained by the xoring of a balanced function and another function (balanced or

unbalanced) leads to a balanced function, so the resultant function is balanced.

Case 2: $d_j = \bigoplus_{i=1}^{n-k} c_{k+i} nl_i(y)$

From the properties of the non-linear permutations, discussed previously, the non-linearity is $2^k m_{nl}$.

For example, if $k = n-3$ and if one uses the LFSR based non-linear permutation, shown in the previous section for V_3, the non-linearity of the resultant function is $2^{n-3}2 = 2^{n-2}$. For the inverse based non-linear permutation the non-linearity of the term $nl(y)$ is similarly $2^k(2^{n-k-1} - 2^{(n-k)/2}) = 2^{n-1} - 2^{\frac{(n+k)}{2}}$, where $n-k$ is even. Number of such cases is clearly $2^{n-k} - 1$.

Also, since the function nl represents a permutation so the component functions are balanced. Hence the non-zero linear combination of the balanced functions gives a balanced function. Thus the resultant function is balanced also in this case. □

An important requirement in S-Box design is to have a regular S-Box. Informally it means that each output symbol should appear an equal number of times when the input is varied over all possible values. Formally, a mapping from the vector space V_n to V_k is said to be a regular mapping if for each element $y \in V_k$, there are exactly 2^{n-k} elements in V_n which are mapped to y. The following theorem characterizes a regular mapping.

Theorem 5.4 *[72] A mapping $(f_1, \ldots, f_k)$, where each f_i is a function on V_n and $n \geq k$, is regular if and only if all nonzero linear combinations of $f_1, \ldots, f_k$ are balanced.*

It is extremely important that an $n \times k$ S-Box be a regular mapping. Otherwise the absence of regularity may be exploited to develop attacks against the S-Box. The following theorem proves that the component functions of Q_1 realize a regular mapping.

Theorem 5.5 *The mapping $Q_1(z) = \{q_1(z), \ldots, q_k(z)\}$ is a regular mapping from V_n to V_k.*

Proof: From **theorem 5.3**, all non-zero linear combinations of the k component boolean functions are balanced. Hence, the mapping is regular by **theorem 5.4**. □

Theorem 5.6 *The mapping* $Q(z) = \{Q_2(z), Q_1(z)\}$ *is invertible.*

Proof:

The output $Q(z) \in V_n$ is represented as:

$$Q(z) = \begin{pmatrix} Q_2(z) \\ Q_1(z) \end{pmatrix} = BQ_1(z)$$

$$= \begin{pmatrix} A_{(n-k)\times k} \\ I_{k\times k} \end{pmatrix} Q_1(z) \oplus \begin{pmatrix} nl \\ 0_{k\times(n-k)} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_{n-k} \end{pmatrix}$$

Hence, $Q_2(z) = A_{(n-k)\times k}Q_1(z) \oplus nl(y)$. Since $nl(y)$ is a permutation, it has an inverse $nl^{-1}(y)$.

Thus, $y = nl^{-1}(Q_2(z) \oplus A_{(n-k)\times k}Q_1(z))$.

Having retrieved the vector y one can obtain the remaining part of the input, i,e x as follows.

We have,

$$Q_1(z) = \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y) L_\sigma(x) \tag{5.5}$$

Let the obtained value of y be δ. Since, $D_\delta(y) = 1$ if and only if $y = \delta$ and so $Q_1(z) = L_\delta(x)$.

Hence, $x = L_\delta^{-1}(Q_1(z))$, which can be obtained as the matrix L_δ is invertible being an element of the set S.

Further the properties of the group CA can be used to develop a very efficient inverting structure.

That is if $L_\delta = T_k^{i-1} \in S$, it can be realized by applying $(i-1)$ clock cycles to the Cellular Automaton, characterized by the matrix T_k. Let this be the forward transformation depicted in **Fig. 5.3**. Since, T_k represents a maximum group CA, we have $T_k^{2^k-1} = I$. Thus the corresponding inverting transformation is represented by $L_\delta^{-1} = (T_k^{i-1})^{-1} = T_k^{((2^k-1)-(i-1))}$. Thus the same structure can be used to invert the output.

□

Theorem 5.7 *The algebraic degree of each component functions of the CASBox is* $(n-k+1)$ *except in* $2^{n-k}-1$ *cases when it will be* $deg(nl)$ *where* $deg(nl)$ *is the algebraic degree of the permutation* $nl(y)$.

Proof:

Case 1: $d_j \neq \bigoplus_{i=1}^{n-k} c_{k+i} nl_i(y)$

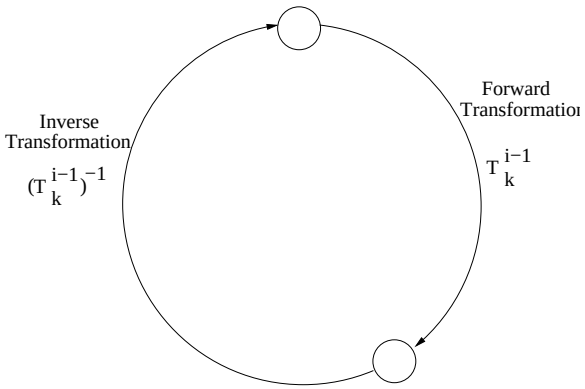


Figure 5.3: **Inversion using Cyclic structure of Group CA**

In this case the algebraic degree of each component function of the CASBox is computed by calculating that of the equation:

$$q_i(y,x) \quad = \quad D_0(y) f^l_{0i}(x) \oplus \ldots \oplus D_{(2^{n-k}-1)} f^l_{(2^{n-k}-1),i}(x),$$

where i ranges from 0 to $(n-1)$.

Here, the functions $f^l_{\sigma,i}(x)$ are linear functions which are obtained from the rows of the characteristic matrices of the Cellular Automaton or from the non-zero linear combinations of the rows. Thus, none of them are zero and also the degree is 1.

The resultant degree of the component functions are obtained by the degree of the terms $D_\sigma(x)$, whose degree is $(n-k)$. Hence the degree of the component functions is $(n-k+1)$.

Case 2: $d_j = \bigoplus_{i=1}^{n-k} c_{k+i} nl_i(y)$

In these $2^{n-k}-1$ cases, the algebraic degree is due to that of the degree, $deg(nl)$ of the component functions of the non-linear permutation $nl(y)$. In order to express more quantitatively the value of $deg(nl)$ in the sample constructions provided is $deg(nl) = (n-k-1)$. □

5.6 Cryptographic Strength of CASBox

In this section the strength of the S-Boxes generated by the Cellular Automata transforms is evaluated.

5.6.1 Strength against Linear Cryptanalysis

Linear Cryptanalysis (LC) tries to take advantage of high probability occurrences of linear expressions involving plaintext bits, ciphertext bits and subkey bits. The basic idea is to approximate a portion of the cipher with an expression that is linear, where linearity refers to a mod-2 bitwise exclusive or operation [147]. The approach in LC is to determine expressions of the form which have a high or low probability of occurrence.

With respect to the CASBox consider an expression of the form:

$< X_{i1} \oplus X_{i2} \oplus \ldots X_{iu} > \oplus < Y_{j_1} \oplus Y_{j_2} \ldots \oplus Y_{j_v} >= 0$

where X_i represents the i-th bit of the input $X = [X_1, X_2, \ldots]$ and Y_j represents the j-th bit of the output $Y = [Y_1, Y_2, \ldots]$. This equation is representing the exclusive OR of u input bits and v output bits.

If the bits are chosen randomly then the above approximated linear expression will hold with probability 1/2. If p_l is the probability with which the expression holds then the bias is defined as $|p_l - 1/2|$. A S-Box is supposed to be defiant against LC if the bias is small.

From the above theorems in all but $(2^{n-k} - 1)$ cases the non-linearity of any non-zero linear combination of the outputs of the CASBox is $2^{n-1} - 2^{k-1}$.

Thus in such cases the probability of match for the best linear approximation is $1 - \frac{2^{n-1}-2^{k-1}}{2^n} = \frac{1}{2} + 2^{k-n-1}$. Hence the bias of any linear approximation of the CASBox is atmost 2^{k-n-1}.

If we set the parameter k such that $\frac{n}{2} < k < \mu n$, where $\frac{1}{2} < \mu < 1$, then the bias of any linear approximation of the CASBox is atmost $\frac{1}{2}\frac{1}{2^{n(1-\mu)}}$. Thus the bias against linear cryptanalysis reduces exponentially with n if one sets the parameter μ appropriately. That is the value of k should be close to $\frac{n}{2}$ to reduce this bias.

However in the remaining cases the non-linearity is $2^k m_{nl}$ where m_{nl} is the maximum non-linearity of the component functions of the non-linear permutation, $nl(y)$.

Thus the bias in such cases is atmost $1 - \frac{2^k m_{nl}}{2^n} - \frac{1}{2} = \frac{1}{2} - 2^{k-n} m_{nl}$.

For the LFSR based construction when $k = n - 3$, $m_{nl} = 2$, thus bias $= \frac{1}{4}$. Hence the parameter k or n has no effect on the bias in such cases.

For the inverse based construction when $(n - k)$ is even the bias is:

$$\begin{aligned} \text{bias} &= \frac{1}{2} - 2^{k-n}(2^{n-k-1} - 2^{\frac{(n-k)}{2}}) \\ &= 2^{(k-n)/2} \end{aligned}$$

In this case also if one sets the parameter k such that $\frac{n}{2} < k < \mu n$, where

$\frac{1}{2} < \mu < 1$, then the bias is atmost $\frac{1}{2^{n/2(1-\mu)}}$. Thus the bias against linear cryptanalysis reduces exponentially with n if the value μ and hence k is properly set. Thus the value of k is ideally set close to $n/2$.

5.6.2 Strength against Differential Cryptanalysis

For the input $z = (y, x)$ to the CASBox ($x \in V_k$, $y \in V_{n-k}$) , the output is $Q(z) = \{Q_2(z), Q_1(z)\}$ where $Q_1(z) \in V_k$ and $Q_2(z) \in V_{n-k}$.

Now, $Q_1(z)$ is formed by the concatenation of 2^{n-k} linear functions on V_k. Likewise, $Q_2(z)$ is also expressible in the form of concatenation of 2^{n-k} linear functions on V_k xored with the vector $y \in V_{n-k}$.

Thus, mathematically,

$$\begin{aligned} Q_1(z) &= \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y) L_\sigma(x) \\ Q_2(z) &= AQ_1(z) \oplus nl(y) \end{aligned}$$

Let $\gamma = (\beta, \alpha)$, $\beta \in V_{n-k}$, $\alpha \in V_k$.

From the definition of the function $D_\sigma(y)$, it is clear that $D_\sigma(y) = 1$ iff $y = \sigma$. Also, it is evident that $D_\sigma(y \oplus \beta) = D_{\sigma \oplus \beta}(y)$. The results have also been proved in [153].

Thus using these facts we obtain the following:

$$\begin{aligned} Q_1(z \oplus \gamma) &= \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y \oplus \beta) L_\sigma(x \oplus \alpha) \\ &= \bigoplus_{\sigma \in V_{n-k}} D_{\sigma \oplus \beta}(y) L_\sigma(x \oplus \alpha) \\ &= \bigoplus_{\sigma' \in V_{n-k}} D_{\sigma'}(y) L_{\sigma' \oplus \beta}(x \oplus \alpha) \\ &= \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y) L_{\sigma \oplus \beta}(x \oplus \alpha) \end{aligned}$$

As,

$$\begin{aligned}\Delta Q_1(z) &= Q_1(z) \oplus Q_1(z \oplus \gamma) \\ &= \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y)[L_\sigma(x) \oplus L_{\sigma \oplus \beta}(x \oplus \alpha)]\end{aligned}$$

Likewise,

$$\begin{aligned}\Delta Q_2(z) &= Q_2(z) \oplus Q_2(z \oplus \gamma) \\ &= [AQ_1(z) \oplus nl(y)] \oplus [AQ_1(z \oplus \gamma) \oplus nl(y \oplus \beta)] \\ &= A[Q_1(z) \oplus Q_1(z \oplus \gamma)] \oplus (nl(y) \oplus nl(y \oplus \beta)) \\ &= A\Delta Q_1(z) \oplus (nl(y) \oplus nl(y \oplus \beta))\end{aligned}$$

The profile of the distribution table may be obtained through the following analysis.

Case 1: Let, $\beta = 0$. Thus the differential of the output $Q(z)$ is denoted by $\Delta Q(z) = \{\Delta Q_2(z), \Delta Q_1(z)\}_{\beta=0}$. Thus, we have $\Delta Q_1(z) = \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y) L_\sigma(\alpha)$.

Since, $D_\sigma(y) = 1$ iff $y = \sigma \Rightarrow \Delta Q_1(z)|_{y=\delta} = L_\delta(\alpha)$.

For $y = \delta$ and $y = \delta'$ (where $\delta \neq \delta'$), we have $L_\delta(\alpha) \neq L_{\delta'}(\alpha)$.

This can be easily observed from the characteristics of a maximum length CA as $T_k^{\delta-1}(\alpha) \neq T_k^{\delta'-1}(\alpha)$, where $(\delta, \delta') \in V_{n-k}$ and the cycle length of a k cell maximum length group CA is $2^k - 1$. Thus, the values of $\delta, \delta' \leq 2^{n-k} - 1 < 2^k - 1$ (as $k > n/2$).

Also, $\Delta Q_2(z) = A[\Delta Q_1(z)]|_{y=\delta} = AL_\delta(\alpha)$. Thus, $\Delta Q(z) = \{AL_\delta(\alpha), L_\delta(\alpha)\}$, which does not depend upon x. Thus, keeping y held at δ if x is varied (for all the possible 2^k cases) the output is constant. Therefore the frequency of $\Delta Q(z)$ in the difference distribution table is 2^k.

Case 2:

When $\beta \neq 0$, $\Delta Q_1(z)|_{y=\delta} = [L_\delta \oplus L_{\delta \oplus \beta}](x) \oplus L_{\delta \oplus \beta}(\alpha)$.

By the closure property of the set S, $L_\delta \oplus L_{\delta \oplus \beta} = L_{\delta''}$, for some $0 \leq \delta'' \leq 2^k - 1$. Thus, $\Delta Q_1(z)|_{y=\delta} = L_{\delta''}(x) \oplus L_{\delta \oplus \beta}(\alpha)$.

$$\begin{aligned}\text{Similarly, } \Delta Q_2(z)|_{y=\delta} &= A\Delta Q_1(z)|_{y=\delta} \oplus (nl(\delta) \oplus nl(\delta \oplus \beta)) \\ &= A(L_{\delta''}(x) \oplus L_{\delta \oplus \beta}(\alpha)) \oplus (nl(\delta) \oplus nl(\delta \oplus \beta)) \\ &= AL_{\delta''}(x) \oplus (AL_{\delta \oplus \beta}(\alpha) \oplus (nl(\delta) \oplus nl(\delta \oplus \beta)))\end{aligned}$$

From the properties of CA and the fact that A is a non-degenerate matrix (as

rows of A are linearly independent) when x takes the values x_1 and x_2 ($x_1 \neq x_2$) and the value of y is held at δ, $\Delta Q_1(\delta, x_1) \neq \Delta Q_1(\delta, x_2)$ and $\Delta Q_2(\delta, x_1) \neq \Delta Q_2(\delta, x_2)$.

Thus varying x through the 2^k values, $\Delta Q(z)$ also takes the values of 2^k vectors in V_n. Now, for the other values of y (there can be 2^{n-k} values of y), $\Delta Q(z)$ also takes 2^k distinct values. In worst case all the 2^k vectors are repeated for each value of y. In that case the frequency of the value of $\Delta Q(z)$ is 2^{n-k}, which is the largest frequency in the difference distribution table in Case 2.

Since $k > n/2$, $\Rightarrow$ the frequency of an entry in the difference distribution table in Case 1 (2^k) $>$ the frequency of an entry in the difference distribution table in Case 2 (2^{n-k}). Thus, $L_{max} = 2^k$.

Since, the $n \times n$ mapping obtained through the CASBox is one-one and hence invertible, $R = 0$. Thus, the robustness of any S-Box generated through the CASBox is atleast:

$$\begin{aligned} \epsilon_{min} &= 1 - \frac{L_{max}}{2^n} \\ &= 1 - \frac{1}{2^{n-k}} \end{aligned} \tag{5.6}$$

5.6.3 Satisfaction of Strict Avalanche Criterion

A boolean function f on V_n is said to satisfy SAC (Strict Avalanche Criterion) iff $f(x) \oplus f(x \oplus \alpha)$ is balanced for all $\alpha \in V_n$. In order to verify whether the S-Boxes generated by CASBox satisfy SAC we observe $\Delta Q_1(z)$ and $\Delta Q_2(z)$. We have:

$$\begin{aligned} \Delta Q_1(z) &= \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y)[(L_\sigma(x) \oplus L_{\sigma \oplus \beta}(x)) \oplus L_{\sigma \oplus \beta}(\alpha)], \\ \Delta Q_2(z) &= A\Delta Q_1(z) \oplus (nl(y) \oplus nl(y \oplus \beta)) \end{aligned}$$

Thus when $\beta \neq 0$, $\Delta Q_1(z) = \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y)[L_{\sigma''}(x) \oplus L_{\sigma \oplus \beta}(\alpha)]$.

Because of the closure property of set S, $L_\sigma(x) \oplus L_{\sigma \oplus \beta}(x) = L_{\sigma''}(x)$ where $L_{\sigma''} \in S$. Now, $L_{\sigma''}(x) \oplus L_{\sigma \oplus \beta}(\alpha)$ is an output of an invertible linear CA xored with a constant term. Hence, the component functions of $L_{\sigma''}(x) \oplus L_{\sigma \oplus \beta}(\alpha)$ are balanced. Thus the component functions of $\Delta Q_1(z)$, which are concatenations of balanced functions are also balanced. Similarly, the component functions of $\Delta Q_2(z)$ are balanced.

However when $\beta = 0$, $\Delta Q_1(z) = \bigoplus_{\sigma \in V_{n-k}} D_\sigma(y)[L_{\sigma \oplus \beta}(\alpha)]$. It is evident that the component functions of $\Delta Q_1(z)$ and hence also of $\Delta Q_2(z)$ are unbalanced. Thus, the component functions of the CASBox does not satisfy SAC when $\beta = 0$.

However a simple linear transformation on the input variables can make $Q(z)$'s component functions satisfy SAC [72]. We have $Q(z) = \{q_1(z), \ldots, q_n(z)\}$. Let, W be a non-degenerate $n \times n$ matrix with entries from $GF(2)$. Let, γ_i denote the i^{th} row of W. If $q_i(z) \oplus q_i(z \oplus \gamma_i)$ is balanced for $i \in \{1, \ldots, n\}$, then $q_i(zW)$ satisfies SAC[153]. From, the previous discussions on balancedness of the boolean functions, $q_i(z)$, it is evident that such a matrix W must have non-zero entries in the first $(n-k)$ columns of each of the n rows. Such a W may be[72]:

$$W = \begin{pmatrix} I_{(n-k)\times(n-k)} & 0_{(n-k)\times k} \\ D_{k\times(n-k)} & I_k \end{pmatrix}$$

and

$$D = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & \ldots & 0 \end{pmatrix}_{k\times(n-k)}$$

The same CASBox $Q(z)$ when operated on the transformed input, $z' = zW$, satisfies SAC. The non-degenerate linear mapping also does not alter the other cryptographic properties developed in the chapter.

Theorem 5.8 *The number of S-Boxes generated by a CASBox with parameters n and k is $N_{CASBox}(n,k) =^{(2^k-1)} P_{(2^{n-k})} \prod_{i=0}^{n-k-1}(2^k - 2^i)$.*

Proof: The mappings from V_n to V_k are generated by varying the number of cycles $(n_1, n_2, \ldots, n_{2^{n-k}})$ such that no two are pairwise distinct. The problem reduces to computing the number of ways of choosing 2^{n-k} elements from the set S. It is clear that the number of ways are ${}^{(2^k-1)}P_{(2^{n-k})}$, where $k > n/2$ and ${}^aP_b$ is equal to $\frac{a!}{(a-b)!}$.

In order to generate the other $(n-k)$ output bits one can vary the matrix $A_{(n-k)\times k}$ such that A is a non-degenerate matrix. Essentially the rows of the matrix A are vectors of length k and elements $\in GF(2)$. The first row can be chosen in $(2^k - 1)$ ways, excluding the zero vector. The second row is chosen in $(2^k - 2)$ ways, excluding the first row also. The third row cannot be the linear combination of the first two rows and hence number of possible third row is $(2^k - 2^2)$. Thus the possible number of ways of choosing A is $\prod_{i=0}^{(n-k-1)}(2^k - 2^i)$. Hence, the total number of S-Boxes generated by the CASBox is ${}^{(2^k-1)}P_{(2^{n-k})} \prod_{i=0}^{n-k-1}(2^k - 2^i)$. □

In the following section we provide an example construction of S-Box through the CASBox architecture.

5.7 Specimen Design of CASBox to Generate S-Boxes

In this section we present a sample construction of S-Box through an 8×8 CASBox. It becomes clear from the example that the CASBox is a highly programmable and modular technique to generate cryptographically robust S-Boxes.

It may be noted that the technique may be used to generate any $n \times k$ S-Box, where n and k are not equal. However if $n \times n$ S-Box is required then, as per the construction we require a non-linear permutation in V_{n-k}. Since, a non-linear permutation is only possible when $(n-k) \geq 3$ and we also have $k > n/2$. By these two inequalities we have $n > 6$.

Example 5.3 *Design of CASBox structure to generate* 8×8 *S-Boxes*

Design Parameters: From the theory of the CASBox developed, we have $n = 8 \Rightarrow k > \frac{8}{2} = 4$. *We choose* $k = 5$. *Hence, we require a* 5 *cell maximum length CA to generate the set* S. *The rule of such a CA is* $\{150, 150, 90, 90, 150\}$ *(***table 3.3***). Hence the matrix,*

$$T_5 = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The length of the maximum cycle of the CA is 31 *and thus the set* $S = \{I, T_5, \ldots, T_5^{30}\}$. *In order to construct the CASBox we thus use* $2^{8-5} = 8$ *Cellular Automata denoted by* $\{U_1, \ldots, U_8\}$ *all of which have the same characteristic matrix* T_5 *only operated for different clock cycles.*

The other matrices involved are:

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \textit{and } A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

As depicted in **Fig. 5.4** *the input is* $z = (y, x)$ *where* y *is of* 3 *bits while* x *comprises of* 5 *bits. The input is transformed into a vector* $z' = zW$ *where* $z' = \{y', x'\}$. *The* 5 *bits* x' *are input to the eight Cellular Automata* U_1 *to* U_8. *During the generation of the S-Boxes each automaton is evolved for different clock cycles. For instance automaton* U_1 *is clocked* n_1 *times,* U_2 *for* n_2 *times and so on. All the quantities* $n_1, n_2, \ldots, n_8$ *are pairwise distinct. In this example, hence number of such choices is* ${}^{31}P_8$. *Thus varying the number of cycles of the CA we can get various S-Boxes, all of which satisfy the desired cryptographic properties. The outputs from the eight automata are multiplexed out using the three bit select lines,* y'. *The five bit output is passed as it is as the five LSBs of the CASBox output. The first three bits of the CASBox are generated from the five bits by the linear transformation* A *and the non-linear permutation* $nl(y)$. *In this construction* $nl(y)$ *is the LFSR based function/ non-linear permutation in* V_3.

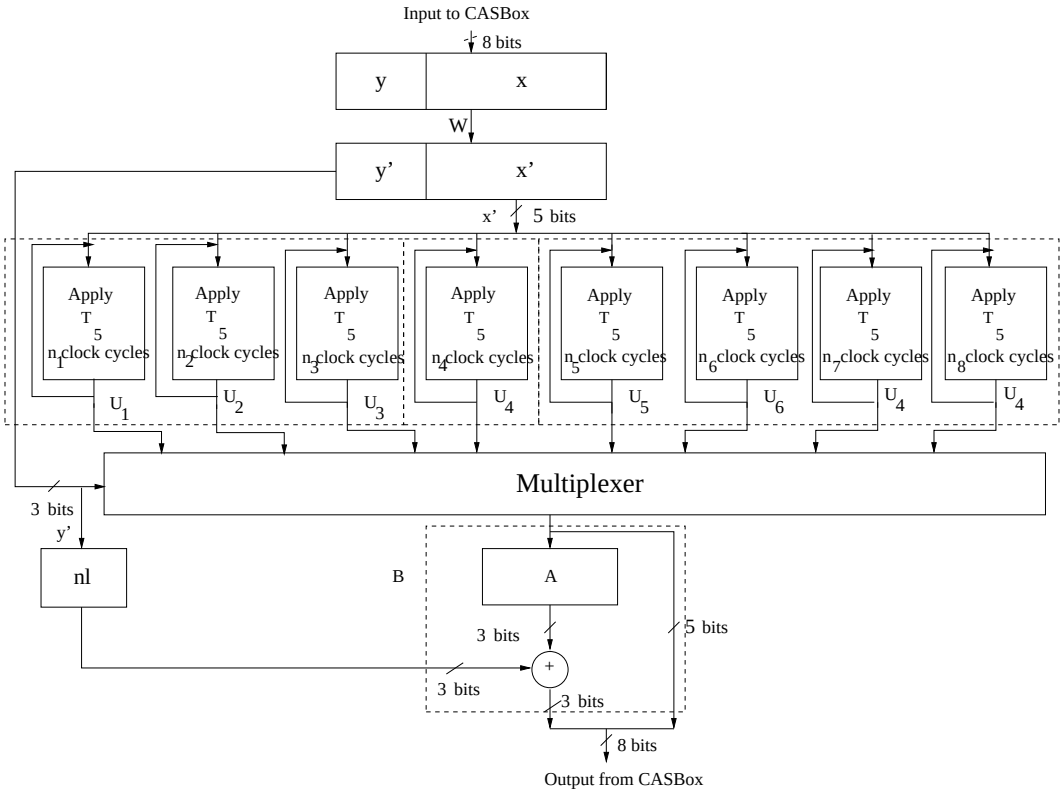


Figure 5.4: **An** 8×8 **CASBox**

As an example, let us operate the first automaton once, the second automaton twice, the third thrice and likewise continuing the eighth automaton for eight clock

cycles.

*The S-Box generated with these parameters is represented by the following table (***table 5.1***):*

Table 5.1: **A generated** 8×8 **S-Box**

0	120	60	198	150	9	207	210	75	95	153	15	86	165	225	144
39	27	221	99	18	201	141	212	68	20	80	130	250	77	139	190
128	36	142	79	23	223	16	189	90	168	103	240	247	105	195	56
12	2	205	166	93	63	149	146	42	114	216	229	65	117	186	235
64	74	211	208	52	104	248	173	193	188	44	88	148	191	38	4
182	37	181	47	13	200	81	157	217	61	164	73	67	241	97	218
32	57	222	152	178	220	100	85	3	206	118	228	50	136	111	138
215	41	145	48	213	92	187	109	199	237	10	127	102	59	131	129
192	163	101	132	170	46	106	108	252	243	183	58	29	245	51	89
184	159	219	126	113	177	119	53	172	231	33	232	238	66	6	40
96	70	137	202	25	83	249	214	110	30	180	161	45	49	254	135
196	72	226	11	123	155	84	209	54	236	35	156	179	5	175	124
224	143	31	121	167	158	7	55	115	107	242	227	21	91	203	140
234	28	133	122	98	194	82	251	151	22	134	14	62	233	112	174
160	176	8	71	76	43	204	244	94	154	125	230	17	162	26	246
185	78	169	1	197	69	253	34	116	87	239	147	171	255	24	19

The properties of the S-Box are:

1. *The S-Box is one-to-one and hence invertible. The inverse CASBox can also be implemented using a programmable maximum length CA based structure and is straightforward from* **theorem 5.6**. *The architecture is depicted in* **Fig. 5.5**.

2. *The non-linearity of any non-zero linear combination of the* 8 *boolean equations is* 112 *except in* $2^{8-5} - 1 = 7$ *cases where it is* $2^{8-2} = 64$, *due to the LFSR based non-linear permutation in* V_3. *The maximum non-linearity for an* 8 *variable boolean function is* 120, *thus the achieved non-linearity is very high.*

3. *Any non-zero linear combination of the* 8 *boolean functions is a balanced boolean function.*

4. *The maximum bias value of an entry in the linear approximation table is either* 16 *or* 64 *as expected in the theoretical results.*

5. *Robustness against Differential Cryptanalysis is* 0.875 *as expected.*

6. The S-Box satisfies SAC.

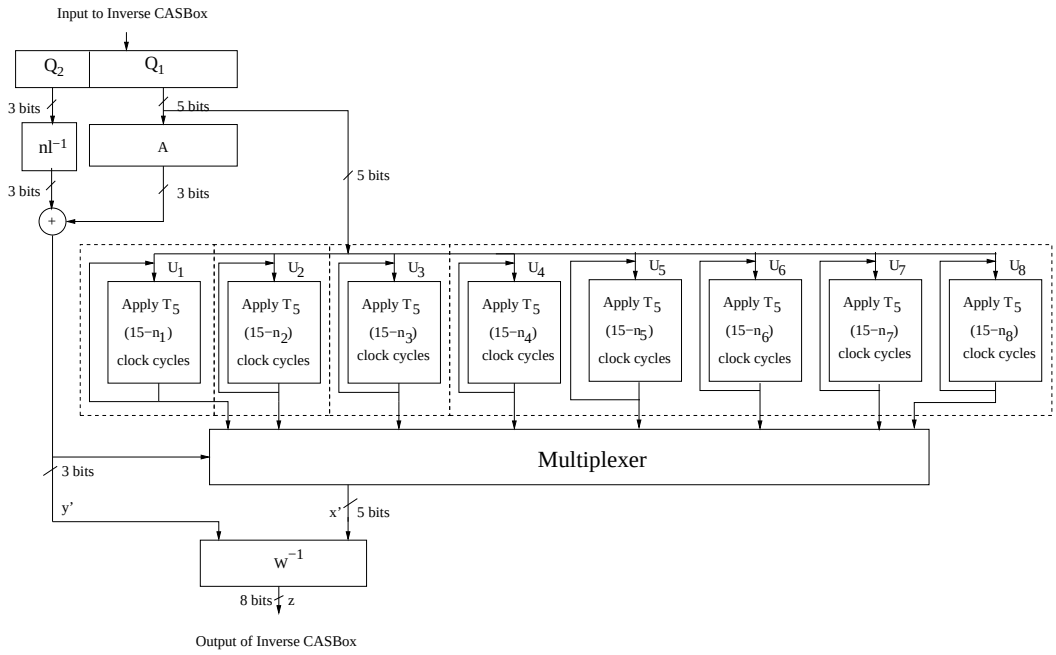


Figure 5.5: **An** 8×8 **Inverse CASBox**

The elegance of CASBox is its programmability i.e, varying the number of clock cycles one can derive large number of S-Boxes, all of which are cryptographically robust.

5.8 VLSI Design of the CASBox

In the present section we outline the VLSI design of the CASBox. The architecture of the design as depicted in **Fig. 5.6** is a scalable and modular design of the CASBox. The n bit input is first converted by the W transformation. The n bit output of the transformation W is split into two parts, of k and $(n-k)$ bits respectively. The k bit output is processed by the k cell CA Based Transform, which is programmed by the rule of a maximum length group CA through the k bit bus *rule*. Contrary to the basic architecture described previously, instead of the 2^{n-k} linear machines, the design uses only one single k-bit maximum length group CA. In the theory developed for the CASBox (**section 5.7**) each of the 2^{n-k} CA runs for pairwise distinct clock cycles $N_1, N_2, \ldots, N_{2^{n-k}}$ which serves as a key to the S-Box. The information is passed to the device as k bit inputs, $iter[1], \ldots, iter[2^{n-k}]$. The k-bit upcounter checks whether

the present count value *count* equals any of the values $iter[1], \ldots, iter[2^{n-k}]$. If count is equal to $iter[i]$, then the line $cntl[i]$ goes high and the output of the CA based transform is transferred to *Buffer[i]*. Likewise all the 2^{n-k}, k bit outputs of the CA based transform are stored in the buffers. The other $(n-k)$ bits of the input is used to select one of the 2^{n-k} outputs of the group CA. Finally the k bit output is transformed by B and the non-linear permutation $nl(y)$ to yield the n bit output. It may be pointed out that an alternative architecture would have been to check the $(n-k)$ bits first and immediately decide the number of clock cycles for which the group CA has to be run to yield the k bit output. Finally the k bit output is converted by the transformation B and $nl(y)$ to yield the n bit output. However in such a case the time required to compute the output of the CASBox depends on the input, which can lead to a timing based side-channel cryptanalysis of the block cipher built out of the CASBox. In our proposed architecture however the time required is constant and independent on either the input or number of iterations.

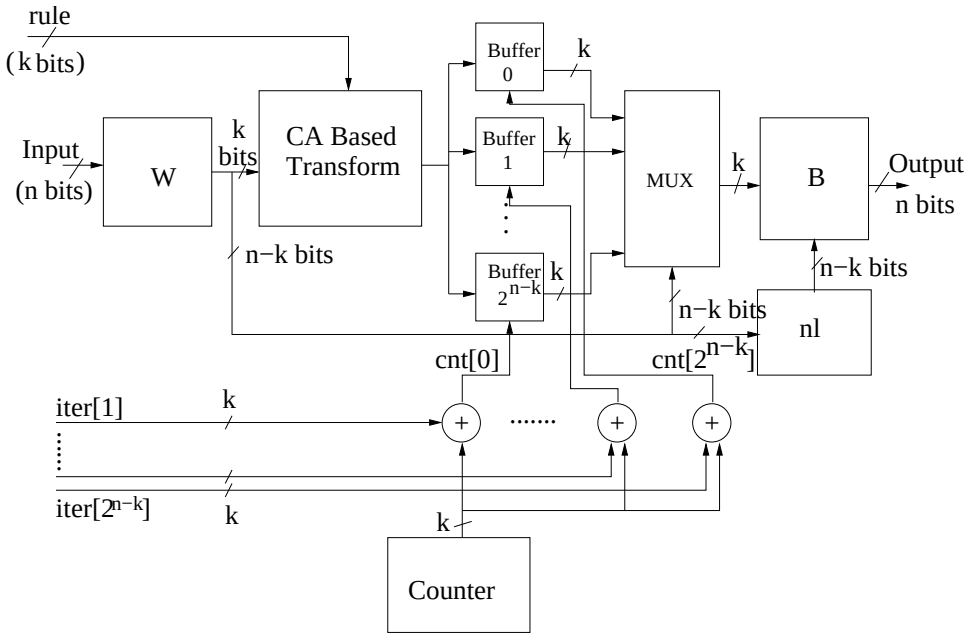


Figure 5.6: **Top-level architecture of the CASBox**

In **table 5.2** the results of the implementation have been furnished.

From the point of view of hardware complexity the Cellular Automaton used is a k bit maximal group CA which uses three neighbourhood rule. The number of clock cycles required is of the order of 2^k. In order to have maximum non-linearity and robustness against Linear Cryptanalysis the value of the parameter k has been kept just greater than $n/2$. As the value of k approaches n, the hardware complexity reduces with a corresponding reduction in the security margin (refer **table 5.3**).

Table 5.2: **Performance of the CASBox implemented on Xilinx XCV-1000 platform**

Dimension	XOR	NAND	NOT	FF	MUX	Slices	Slice FF	LUT	Freq
8×8	11	0	0	51	7	65	51	115	102.828 MHz
10×10	14	0	0	109	8	122	109	204	93.79 MHz
16×16	23	0	0	1189	11	1401	1189	1606	94.233 MHz

Table 5.3: **Dependence of efficiency in implementation and the parameter k for an 8 bit CASBox**

k	XOR	NAND	NOT	FF	MUX	Slices	Slice FF	LUT	Freq
5	11	0	0	51	7	65	51	115	102.828 MHz
6	10	0	0	37	8	55	37	94	106.417 MHz
7	9	0	0	29	8	48	29	85	108.944 MHz

5.9 CASBox helps in preventing attacks on SPN ciphers

The design of CASBox has some inherent features which help in preventing attacks on the SPN ciphers built out of it. The reasons may be listed as follows:

1. **CASBox helps to randomize SPN ciphers:**

   An $n \times n$ S-Box generated by the CASBox has two parameters (n, k) where $k > n/2$. Here we require 2^{n-k} linear CA chosen from the set S. Inherently each of the automata runs for different clock cycles, say $N_1, N_2, \ldots$ such that they are pairwise distinct. Changing the number of clock cycles results in a different S-Box, which in turn gives rise to a different SPN cipher. So, CASBox has an inherent scope of keying and hence implements randomly selected S-Boxes. The selection of S-Boxes can be synchronized between the sender and receiver according to some algorithm (output of a pseudo-random algorithm having the same input seed and running at both ends). As reported in [152], if the substitution boxes of an SPN are randomly selected, then the expected value of any expected linear probability (ELP) converges to that of a true random cipher, as the number of rounds is increased. The CASBox gives a practical implementation to realise such a cipher. Such a randomly selected S-Box helps

to prevent even unknown attacks, as when there is no systematic design there can be no systematic weakness.

2. **CASBox has no Algebraic description:** The S-Boxes generated by the CASBox lacks any straight-forward algebraic description. Even the Rijndael S-Box, though very complex when regarded as a function, can be characterized in several ways by algebraic relations, being true with high probability, usually 1[27]. On page 6 of [93], the designers of AES say:*".. The disadvantage of these boxes is that they have a simple description in $GF(2^m)$, which is also the field in which the diffusion layer is linear. This may create uneasy feelings, but we are not aware of any vulnerability caused by this property. For the time being we challenge cryptanalysts to demonstrate any vulnerability caused by this property. Should such a vulnerability exist, one can always replace the S-Boxes by S-Boxes with similar properties, that are not algebraic over $GF(2^m)$."* The idea of algebraic attacks have been employed over stream ciphers and also block ciphers. The seminal idea of Patarin [157] and later improved by [158, 159] has lead to the breaking of several stream ciphers [160, 86, 87, 161] and also is probing the security of AES [24, 26]. Jakobsen clearly makes his point showing to obtain that to obtain secure ciphers *"...it is not enough that round functions have high Boolean complexity. Likewise, good properties against differential and linear properties are no guarantee either. In fact, many almost perfect non-linear functions should be avoided exactly because they are too simple algebraically...."*. In [27] the author makes extensive investigations on the inverse function in $GF(2^n)$ and some of its linear equivalents and uses to build them both as components of highly insecure ciphers and as the algebraic structure that can be exploited in attacks. The disadvantage of S-Boxes which does not have an algebraic description is implementation. The fact that the Rijndael S-Box may be efficiently implemented both in hardware [162, 163, 164, 165, 166, 167] and software [168] makes it ideal for high throughput applications like wireless. Our CASBox is an ideal alternative in the fact that it may be efficiently implemented using the programmable Cellular Automata based structure and the S-Boxes generated lack easily obtainable algebraic relations and hence may be resistant against algebraic attacks. However deeper investigation is required to draw a conclusion on the topic.

5.10 Conclusion

In the present chapter we have developed a programmable Cellular Automata based S-Box, called CASBox. We have shown analytically that the group properties of a maximum length CA can be appropriately multiplexed to generate cryptographically

robust S-Boxes. The VLSI design of the CASBox show that the design is efficient, scalable and conducive for high speed designs. The next chapter inspects whether the key mixing step in SPN block ciphers, traditionally performed by xoring, can be replaced favorably with modular integer addition.

Chapter 6

Key Mixing in Block Ciphers through Addition modulo 2^n

6.1 Introduction

Linear Cryptanalysis [30] is one of the most powerful and significant attacks applicable to symmetric key block ciphers. The block ciphers have to be designed so that they provide resistance to Linear cryptanalysis (LC). Although some design methodologies have been proposed in [169, 170, 171, 172], the systematic development of block ciphers with resistance against linear cryptanalysis is still a challenging task.

Linear Cryptanalysis essentially deals with the probability of approximating the input and output of non-linear functions, used in the block cipher with linear expressions [148]. The objective of LC is to obtain the last round key of a R round block cipher from the linear approximations of $(R-1)$ rounds. The linear approximation is achieved by combining the smaller linear expressions with large bias [148]. The bias of the linear expression is obtained using the Piling-Up lemma and has to be suitably high for the attack to successfully reveal the last round keys. From this lemma it is evident that for a linear expression with a large bias, the biases of each individual sub-expressions have to be significant. If one of them is negligible (almost zero), then the bias of the resultant expression is also negligible (almost zero) and does not lead to a successful linear cryptanalysis.

In Substitution-Permutation Network (SPN) like AES, DES the key mixing step is performed by key xoring where the key bits are simply xored (that is added without carry) with the data bits before each round and after the last round. In [173] the linear approximations of addition modulo 2^n (with carry) was studied. The author derived

an $\theta(logn)$-time algorithm to compute the correlation of linear approximations of addition modulo 2^n. The algorithm is optimal and generates all linear approximations with a given non-zero correlation coefficient, and also determines the distribution of the correlation coefficients. The present work computes the reduced bias of linear expressions in the cipher when key mixing is performed in an n bit block cipher by addition modulo 2^n. Indeed some block ciphers like MARS[174], IDEA[175] and FEAL[176] use addition modulo 2^n inside their rounds. Also in the design of *"SEA: A Scalable Encryption Algorithm for Small Embedded Applications"* the designers have used addition modulo 2^n [177]. One of the reasons behind the use of such a step is the improvement in non-linearity. However the security margin provided by the addition step was not computed analytically and was hence under-estimated. In the cryptanalysis of FEAL[178] the best linear approximations of the arithmetic addition step was computed. In [179] the cryptographic significance of the carry term involved in integer addition was studied. The aim of the work was to investigate the probability distribution of the carry for integer addition with arbitrary number of integers.

The present work revisits the analysis of the integer addition step but with a different objective, to observe the effect of a key mixing by addition modulo 2^n on Linear Cryptanalysis. The work presents a new approach to analytically determine the bias of linear approximations of the addition step and to estimate the best possible bias. Finally, it shows experimentally that such a key mixing operation can help to foil the powerful linear cryptanalysis. It has been experimentally demonstrated that an SPN block cipher, named GPig1 which performs key mixing through xoring breaks when Linear Cryptanalysis is applied. However, when the key mixing step is performed through addition modulo 2^n, the modified SPN cipher named GPig2 does not reveal the key when attacked using Linear Cryptanalysis (LC).

In the next section (*section 6.2*) the maximum bias of linear approximations for addition modulo 2^n has been evaluated. *Section 6.3* presents the construction of the block ciphers GPig1 and GPig2. *Section 6.4* shows theoretically that the bias of sample linear approximations for GPig2 is much less compared to those in GPig1. *Section 6.5* compares the linear attack on GPig1 with that over GPig2 and demonstrates that GPig2 is a stronger cipher. Finally *section 6.6* summarises the advantages of performing key mixing in block ciphers through addition modulo 2^n compared to bitwise xor. The work is concluded in *section 6.7*.

6.2 Linear Approximation of Addition modulo 2^n

Block Ciphers use simple bit-wise exclusive OR between the key bits associated with a round and the data block input to a round. Also at the end there is a key xoring step with a round key, so that a cryptanalyst cannot easily work his way backwards.

Linear Cryptanalysis (LC) tries to take advantage of high probability occurrences of linear expressions involving plaintext bits, ciphertext bits and subkey bits. The basic idea is to approximate a portion of the cipher with an expression that is linear, where linearity refers to a mod-2 bitwise exclusive or operation. The approach in LC is to determine expressions of the form which have a high or low probability of occurrence. Let us consider an expression of the form:

$< X_{i1} \oplus X_{i2} \oplus \ldots X_{iu} > \oplus < Y_{j_1} \oplus Y_{j_2} \ldots \oplus Y_{j_v} >= 0$

where X_i represents the i-th bit of the input $X = [X_1, X_2, \ldots]$ and Y_j represents the j-th bit of the output $Y = [Y_1, Y_2, \ldots]$. This equation is representing the exclusive OR of u input bits and v output bits.

If the bits are chosen randomly then the above approximated linear expression will hold with probability 1/2. If p_l is the probability with which the expression holds then the bias is defined as $|p_l - 1/2|$.

In order to extract the key bits the cryptanalyst forms linear approximations for $R-1$ rounds (if R is the total number of rounds) with large probability bias. Then the cryptanalyst attacks the last round subkeys or round keys. The probability of various linear expressions are formed and are collected using the Piling-Up Lemma to form bigger equations. The lemma is stated underneath without proof.

Lemma 6.1 *[30] For n independent, random binary variables $X_1, X_2, \ldots, X_n$, with bias $\epsilon_1, \epsilon_2, \ldots, \epsilon_n$,*

$$Pr(X_1 \oplus \ldots \oplus X_n = 0) = 1/2 + 2^{n-1} \prod_{i=1}^{n} \epsilon_i$$

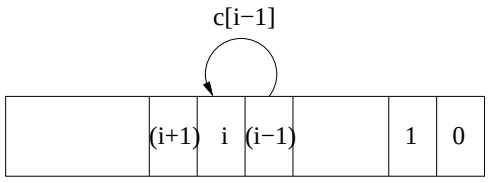
Thus if $X_1, X_2, \ldots, X_n$ are n linear approximations then the bias of the linear approximation made out of these n equations is denoted by [148, 147]:

$$\epsilon_{1,2,\ldots,n} = 2^{n-1} \prod_{i=1}^{n} \epsilon_i$$

Thus it is evident that if the bias of any of the linear approximations fall then the bias of the resultant equation also reduces. In the following theorems we compute the maximum bias of all possible linear approximations of addition modulo 2^n. Hence we obtain the best linear approximation of addition modulo 2^n in order to perform LC. Subsequently the biases are used to perform Linear Cryptanalysis (LC) against an SPN cipher, where the key is mixed using addition modulo 2^n. Results show that such a cipher becomes stronger against Linear Cryptanalysis.

Theorem 6.1 *If two n bit numbers, x and k generate an n bit number* $y = (x + k)$ *mod* 2^n*, then the probability* p_i *of denoting* $y[i]$*, each output bit of y by the linear function* $x[i] \oplus k[i]$ *is* $p_i = 1/2 + (1/2)^{i+1}$ *and* p_i *lies in the range* $1/2 < p_i \leq 1$ *as i lies in* $0 \leq i < n$*.*

Proof: Let $c[i]$ denote the carry out from the addition of x and k after i bits, (refer **Fig. 6.1**). Clearly, $y[0] = x[0] \oplus k[0]$, with probability 1. Thus $p_0 = 1$.



1. The Output Register y which stores the sum of two registers x and k
2. 0, 1, ... ,(i−1), i, (i+1), ... indicates the bit positions of y
3. c[i−1] indicates the carry out after the addition of (i−1) bits are complete

Figure 6.1: **The Output State of the sum**

Now, $y[1] = x[1] \oplus k[1]$ when there is no carry $c[0]$, fed from the 0^{th} bit.

Now, $c[0] = 0$, with probability $3/4$ and hence $p_1 = 3/4$.

Let, the event that the i_{th} bit of y can be expressed as the linear expression in $x[i]$ and $k[i]$ has a probability p_i. Similarly the $(i+1)^{th}$ bit can be linearly expressed with a probability p_{i+1}.

Now, we note the following fact. The $(i+1)^{th}$ bit cannot be linearly expressed if there is a carry from the i^{th} bit, that is if $c[i]$=1.

This can be divided into two mutually exclusive cases.

- First, the event say A is the case when $c[i-1]$=0 and the addition of $x[i]$ and $y[i]$ generates a carry. Now, when $c[i-1] = 0$, then $y[i]$ must have been linearly expressed (using the above fact) and the probability by definition is p_i. Thus the probability that A is true is $1/4.p_i$.

- The other event B is the case where $c[i-1]$=1 and the addition of $x[i]$ and $y[i]$ propagates the carry. The probability that B is true is $3/4.(1 - p_i)$.

Clearly if the event $(A \cup B)$ occurs then the $(i+1)^{th}$ bit cannot be linearly expressed. The probability that the $(i+1)^{th}$ bit cannot be linearly expressed is $(1 - p_{i+1})$.

Thus, $(1 - p_{i+1}) \quad = \quad P(A \cup B)$

$$\begin{aligned} &= P(A) + P(B) \text{ (because A and B are mutually exclusive)} \\ &= 1/4.p_i + 3/4.(1 - p_i) \\ \text{or, } p_{i+1} &= 1/4 + p_i/2 \end{aligned}$$

Using the recurrence relation we have,

$$\begin{aligned} p_{i+1} &= 1/4 + p_i/2 \\ &= 1/4 + 1/2(1/4 + p_{i-1}/2) \\ &= 1/4[1 + 1/2] + (1/2)^2 p_{i-1} \end{aligned}$$

Thus continuing we have,

$$\begin{aligned} p_{i+1} &= 1/4[1 + (1/2) + (1/2)^2 + \ldots + (1/2)^i] + (1/2)^{i+1} p_0 \\ &= 1/2[1 + (1/2)^{i+1}], \text{ since } p_0 = 1 \end{aligned}$$

Thus, $p_i = 1/2[1 + (1/2)^i] = 1/2 + (1/2)^{i+1}$.

Using the equation we have $p_0 = 1, p_1 = 3/4, p_2 = 5/8, p_3 = 9/16$ and so on.

Clearly, $1/2 < p_i \leq 1$.

□

Therefore, the bias of the linear approximation relating to the i^{th} bit position is $(p_i - 1/2) = 1/2^{i+1}$.

In the following theorems we compute the maximum value of the biases of all possible linear approximations of the sum bits.

Theorem 6.2 *If two n bit numbers, x and k generate an n bit number $y = (x + k)$ mod 2^n, then the largest bias of a linear approximation of each output bit of y is $1/4$.*

Proof: It is evident that, $y[i] = x[i] \oplus k[i] \oplus c[i-1]$, where $c[i-1]$ is the carry in of the i^{th} bit of the addition. The carry in is the non-linear part of the equation. Thus in order to obtain various linear approximations for the non-linear part linear approximations have to be found out for the carry in term. Each possible approximation of $c[i]$, denoted by $L[i]$ will give rise to different biases which are equal to the bias of a linear approximation of $y[i]$.

The equation for $c[0] = x[0]k[0]$, which is a boolean function for two variables.

Likewise, $c[1] = \text{majority}(x[1], k[1], c[0])$

$$= \text{majority}(x[1], k[1], x[0]k[0])$$
$$= x[1]k[1] \oplus x[1]x[0]k[0] \oplus k[1]x[0]k[0].$$

Thus $c[1]$ is a boolean function of four variables.

Likewise, $c[i]$ is a boolean function for $2(i+1)$ variables.

The maximum non-linearity for an m variable boolean function, where m is even, is $2^{m-1} - 2^{m/2-1}$. Such a function is known as a bent function. Hence, the probability of match for the best linear approximation of a boolean function with maximum non-linearity operating on an even number of variables is: $1 - \frac{2^{m-1}-2^{m/2-1}}{2^m} = \frac{1}{2} + 2^{-(m/2+1)}$.

Thus, the probability of matching for the best linear approximation for a bent function is $1/2 + 2^{-(i+2)}$, substituting $m = 2(i+1)$.

The output $y[i] = x[i] \oplus k[i] \oplus c[i-1]$ can thus be approximated by a linear equation, $y'[i] = x[i] \oplus k[i] \oplus L[i-1]$, where $L[i-1]$ is the best linear approximation for $c[i-1]$.

Hence, if the term $c[i-1]$ was a bent function, the largest probability of $L(i-1)$ matching $c[i-1]$ is $1/2 + 2^{-(i-1+2)} = 1/2 + 2^{-(i+1)}$.

However, the non-linearity of $c[i-1]$ is not maximum. We note that $c[i-1] = x[i-1]k[i-1] \oplus \ldots$.

The non-linearity of a two variable *and* function is 1. But since $c[i-1]$ is a boolean function for $2(i-1+1) = 2i$ variables, the number of times the truth table of the term $x[i-1]k[i-1]$ appears in the truth table of $c[i-1]$ is 2^{2i-2}. Thus the resultant non-linearity of $x[i-1]k[i-1]$ is $2^{2i-2}.1 = 2^{2i-2}$.

It may be noted that since the other terms of $c[i-1]$ are also *and* terms of more than two variables (and thus have less non-linearity than $x[i-1]k[i-1]$), the non-linearity of $c[i-1]$ is the same as that of $x[i-1]k[i-1]$, that is 2^{2i-2}.

Therefore the largest probability of $L(i-1)$ matching $c(i-1)$ is $1 - \frac{2^{2i-2}}{2^{2i}} = 3/4$.

Thus, the largest bias of a linear approximation for $c[i-1]$ and hence $y[i]$ is $1/4$.
□

Corollary 6.1 *The best linear approximation for $s[i]$ is $a[i] \oplus k[i] \oplus k[i-1]$, where the bias is $1/4$. If $y[i]$ is approximated by the equation $a[i] \oplus k[i]$, then the bias is $2^{-(i+1)}$.*

We see that the bias of the linear approximations involving the key bits reduces considerably and makes the finding of linear approximations in the cipher with a large bias more difficult. Discovering the key through Linear Cryptanalysis becomes improbable.

In order to observe the effects of key mixing through addition on linear cryptanalysis we choose two SPN ciphers, GPig1 and GPig2. The cipher, named GPig1, has

been chosen from [147, 148]. GPig2 is constructed in such a way that it differs from GPig1 so that the key mixing is performed through addition modulo 2^n. First, the construction of the two block ciphers are highlighted in the following section.

6.3 Construction of the SPN Ciphers : GPig1 and GPig2

In this section we present the construction of Substitution-Permutation networks, GPig1 and GPig2, which is subsequently cryptanalyzed using linear cryptanalysis. The cipher GPig1 is essentially a traditional SPN block cipher, where the key mixing is performed by xoring between the data and the round keys. GPig1 is modified into another cipher and named GPig2, the only modification in the latter cipher being that the key mixing step is performed through addition modulo 2^n. In subsequent sections linear cryptanalysis against the modified cipher has been compared with that of the original cipher to demonstrate the benefit of the change.

6.3.1 The Substitution-Permutation Network-GPig1

In **Fig. 6.2** the unmodified block cipher *GPig1* is illustrated. The cipher takes a 16-bit input block and processes the block by repeating the basic operations of a round four times. Each round consists of

- Substitution
- a Transposition of bits (Permutation)
- a Key Mixing Step

This basic structure is the Fiestel Network and the basic operations are similar to those found in DES and in many modern ciphers, including Rijndael. Thus, the experimentation performed on the SPN cipher with respect to linear cryptanalysis is also applicable in case of standard and more practical block ciphers, without loss of generality.

The various blocks used in the block cipher are detailed next.

Substitution

In the cipher, the 16 bit data block data is subdivided into four groups (sub-blocks). Each sub-block forms an input to a 4×4 S-Box (a substitution with 4 input and 4

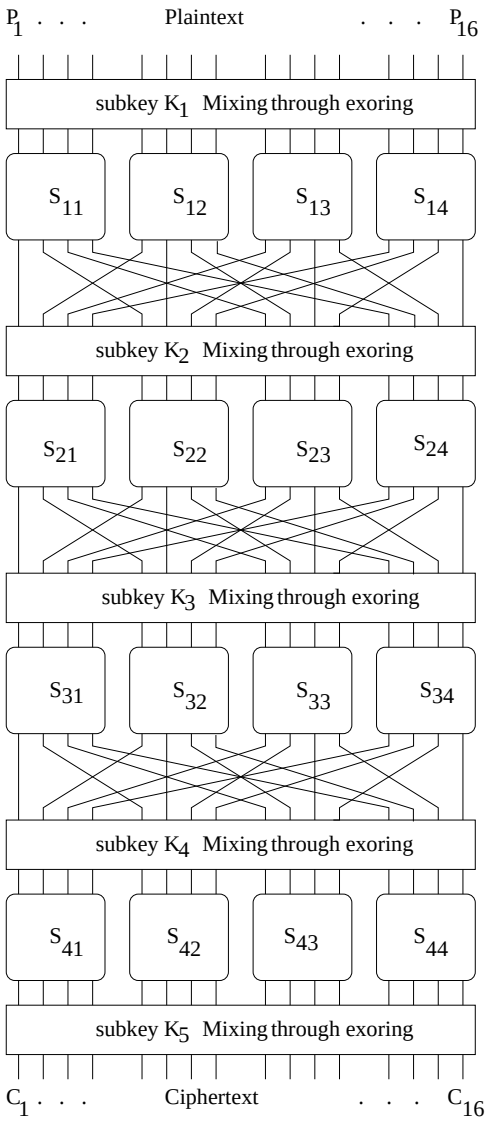


Figure 6.2: **The Structure of GPig1**

output bits), which can be implemented easily with a table lookup of sixteen 4-bit values, indexed by the integer represented by the 4 input bits. For the cipher, the same S-Box is chosen for all the rounds and is chosen from the S-Boxes of DES. It is the first row of the first DES S-Box. In **table 6.1**, the most significant bit of the hexadecimal notation represents the leftmost bit of the S-Box in **Fig. 6.2**.

Table 6.1: **S-Box Representation (in hexadecimal)**

input	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
output	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Permutation

The permutation portion of a round is simply the transposition of the bits or the permutation of the bit positions. The permutation of **Fig. 6.2** is given in **table 6.2** (where the numbers represent bit positions in the block, with 1 being the leftmost bit and 16 being the rightmost bit) and can be simply described as: the i^{th} input bit is connected to the j^{th} output bit (see **Fig. 6.2**).

Table 6.2: **Permutation**

input	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
output	1	5	9	13	2	6	10	14	3	7	11	15	4	8	12	16

Key Mixing

The key mixing is achieved in the block cipher through bit-wise exclusive-OR between the key bits associated with a round (referred to as a subkey) and the data block input to a round. The subkey for a round is derived from the master's key through a process known as the key schedule. In the cipher, we shall assume that all the subkeys are independently generated and are unrelated.

Decryption

In order to decrypt, data is essentially passed backwards through the network. However the S-Boxes have to be bijective. Also, the subkeys have to be applied in the reverse order for proper decryption.

6.3.2 The modified SPN Cipher-GPig2

GPig2 is a similar block cipher as GPig1 with the only difference being in the key mixing step. Instead of xor operations between the data of the i^{th} round (X_i) and the i^{th} round key (K_i), the key mixing in GPig2 is performed through addition modulo 2^{16}. Thus, we replace the key mixing step of the i^{th} round:

$$Y_i = X_i \oplus K_i$$

with, $Y_i = (X_i + K_i)\%2^{16}$, where $'+'$ represents the arithmetic addition operation. The symbol $'\%'$ is the modulo operation, st $0 \leq Y_i \leq 2^{16}$.

It is clear that the step is a reversible step, since $X_i = (Y_i - K_i)\%2^{16}$, where $'-'$ refers to signed arithmetic subtraction.

In the present section both GPig1 and GPig2 are analyzed under the light of linear attack. In order to start with the analysis we first need to analyze the S-Box components and obtain linear approximations for the S-Box, which is the same in both the ciphers.

6.4 Linear Cryptanalysis of GPig1 and GPig2

The linear approximations of the S-Box is presented in [148, 147]. We summarise the result with a brief description. As **Fig. 6.3** shows, the input bits of the S-Box are represented by X_1, X_2, X_3, X_4 and the output by Y_1, Y_2, Y_3, Y_4. A linear approximation involving the input bits is denoted by $a_1X_1 \oplus a_2X_2 \oplus a_3X_3 \oplus a_4X_4$, where $a_i \in \{0,1\}$. The approximation can be represented by a hexadecimal value $a_1a_2a_3a_4$, where a_1 is the most significant bit. Similarly the linear approximation involving the output bits, $b_1X_1 \oplus b_2X_2 \oplus b_3X_3 \oplus b_4X_4$, where $b_i \in \{0,1\}$, is denoted by the hexadecimal value $b_1b_2b_3b_4$. In order to obtain the probability of a linear approximation, all the 16 possible input values for X are applied, and the corresponding output values of Y are examined. The number of matches between the output Y and the linear approximation of the output is obtained (N). Thus the bias is $\frac{N}{16} - \frac{1}{2}$.

For example, for the expression,

$$X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4,$$

it is observed that out of the 16 cases, 12 is the number of matches. Thus the probability of the linear approximation is $\frac{12}{16} = \frac{3}{4}$ and the bias is $\frac{3}{4} - \frac{1}{2} = \frac{1}{4}$.

A complete enumeration of all the linear approximations of the S-Box in the cipher is given in **table 6.3** [148]. The entries of the table are filled up with the values $N - 8$. Thus, the bias for a linear approximation is obtained by dividing an entry in the table by 16. Hence, for the above example the input sum in hexadecimal is 6 and

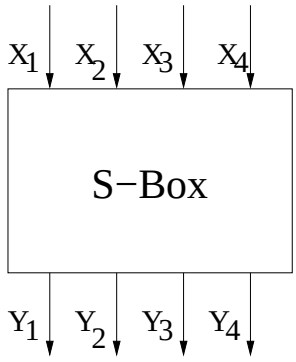


Figure 6.3: **S-Box Mapping**

the corresponding output sum is B. Thus the corresponding entry in the table is +4 and therefore the bias is $\frac{+4}{16} = \frac{1}{4}$.

Table 6.3: **Linear Approximation Table (LAT) of S-Box**

		Output Sum															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	0	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	-2	-2	0	0	-2	+6	+2	+2	0	0	+2	+2	0	0
	2	0	0	-2	-2	0	0	-2	-2	0	0	+2	+2	0	0	-6	+2
I	3	0	0	0	0	0	0	0	0	+2	-6	-2	-2	+2	+2	-2	-2
n	4	0	+2	0	-2	-2	-4	-2	0	0	-2	0	+2	+2	-4	+2	0
p	5	0	-2	-2	0	-2	0	+4	+2	-2	0	-4	+2	0	-2	-2	0
u	6	0	+2	-2	+4	+2	0	0	+2	0	-2	+2	+4	-2	0	0	-2
t	7	0	-2	0	+2	+2	-4	+2	0	-2	0	+2	0	+4	+2	0	+2
	8	0	0	0	0	0	0	0	0	-2	+2	+2	-2	+2	-2	-2	-6
S	9	0	0	-2	-2	0	0	-2	-2	-4	0	-2	+2	0	+4	+2	-2
u	A	0	+4	-2	+2	-4	0	+2	-2	+2	+2	0	0	+2	+2	0	0
m	B	0	+4	0	-4	+4	0	+4	0	0	0	0	0	0	0	0	0
	C	0	-2	+4	-2	-2	0	+2	0	+2	0	+2	+4	0	+2	0	-2
	D	0	+2	+2	0	-2	+4	0	+2	-4	-2	+2	0	+2	0	0	+2
	E	0	+2	+2	0	-2	-4	0	+2	-2	0	0	-2	-4	+2	-2	0
	F	0	-2	-4	-2	-2	0	+2	0	0	-2	+4	-2	-2	0	+2	0

6.4.1 Linear Approximations for the complete Ciphers

The biases of the linear approximations have been obtained for the S-Boxes of the SPN networks. By concatenating appropriate linear approximations of the S-Boxes, the linear approximations of the complete cipher involving plaintext bits and data

bits from the output of the second last round of S-Boxes are obtained, using the Piling-Up lemma. Following is an example of the calculation of the bias of a linear approximation of both the ciphers. It is evident from the results that the bias of a linear approximation for GPig2 is much lesser than that for GPig1.

In the following example, $U_i(V_j)$ represents the 16-bit block of bits at the input (output) of the round i S-Boxes and $U_{i,j}$ ($V_{i,j}$) represent the j^{th} bit of block $U_i(V_j)$ (where the bits are numbered from 1 to 16 from left to right in **Fig. 6.2**). In case of GPig1 the 16-bit block key for the i^{th} round , K_i, is exclusive-ORed at the input to round i. However, K_5 is the key exclusive-ORed at the output of round 4. In the case of GPig2, instead of exclusive-OR, as already pointed out, the key bits are added modulo 2^n to the data blocks.

Example 6.1 *Comparison of the probability biases of linear approximations for the first* 3 *rounds of GPig1 and GPig2*

Sample Linear Approximation: $U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0$

GPig1:

In order to obtain the linear approximation for the first two rounds we consider the following linear expressions:

1. $V_{1,6} = U_{1,5} \oplus U_{1,7} \oplus U_{1,8}$, *with bias* $\frac{1}{4}$
2. $U_{1,5} = P_5 \oplus K_{1,5}$, *with bias* $\frac{1}{2}$
3. $U_{1,7} = P_7 \oplus K_{1,7}$, *with bias* $\frac{1}{2}$
4. $U_{1,8} = P_8 \oplus K_{1,8}$, *with bias* $\frac{1}{2}$
5. $U_{2,6} = V_{2,6} \oplus V_{2,8}$, *with bias* $-\frac{1}{4}$
6. $U_{2,6} = V_{1,6} \oplus K_{2,6}$, *with bias* $\frac{1}{2}$

The concatenation of the above expressions lead to the following approximation:

$$V_{2,6} \oplus V_{2,8} \oplus P_5 \oplus P_7 \oplus P_8 \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} = 0 \qquad (6.1)$$

The Piling-Up Lemma predicts that the bias of **equation 6.1** *is equal to* $\beta_1 = 2^5(\frac{1}{4}\frac{1}{2}\frac{1}{2}\frac{1}{2}(-\frac{1}{4})\frac{1}{2}) = -\frac{1}{8}$.

Similarly, in order to obtain the linear approximation for the third round we consider the following expressions:

1. $U_{3,6} = V_{2,6} \oplus K_{3,6}$, *with bias* $\frac{1}{2}$
2. $U_{3,14} = V_{2,8} \oplus K_{3,14}$, *with bias* $\frac{1}{2}$
3. $U_{3,6} = V_{3,6} \oplus V_{3,8}$, *with bias* $-\frac{1}{4}$
4. $U_{3,14} = V_{3,14} \oplus V_{3,16}$, *with bias* $-\frac{1}{4}$

Combining the equations we arrive at the expression:

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus V_{2,6} \oplus K_{3,6} \oplus V_{2,8} \oplus K_{3,14} = 0 \tag{6.2}$$

with a bias of $\beta_2 = 2^3(\frac{1}{2}\frac{1}{2}(-\frac{1}{4})(-\frac{1}{4})) = +\frac{1}{8}$.

Combining **equation 6.1** *and* **equation 6.2** *we get the expression:*

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus K_{3,14} = 0 \tag{6.3}$$

The following expressions:

1. $U_{4,6} = V_{3,6} \oplus K_{4,6}$
2. $U_{4,8} = V_{3,14} \oplus K_{4,8}$
3. $U_{4,14} = V_{3,8} \oplus K_{4,14}$
4. $U_{4,16} = V_{3,16} \oplus K_{4,16}$

each having a bias of $\frac{1}{2}$, *are combined with* **equation 6.3** *to finally obtain:*

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus \sum_K = 0 \tag{6.4}$$

Here, $\sum_K = K_{1,5} \oplus K_{1,7} \oplus K_{1,8} \oplus K_{2,6} \oplus K_{3,6} \oplus K_{3,14} \oplus K_{4,6} \oplus K_{4,8} \oplus K_{4,14} \oplus K_{4,16}$.

Hence, using Piling-Up Lemma the bias of the equation is:

$\beta_3 = 2^5(\beta_1\beta_2(\frac{1}{2^4})) = 2^5((-\frac{1}{8})(\frac{1}{8})(\frac{1}{2^4})) = -\frac{1}{32}$.

Now, since $\sum_K$ is fixed (that is either 0 or 1 depending on the key bits), the linear approximation

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0 \tag{6.5}$$

holds with probability $\frac{1}{2} - \frac{1}{32} = \frac{15}{32}$. or $1 - \frac{15}{32}$, depending on whether $\sum_K$ is 0 or 1.

*Thus, the bias of the linear expression (***equation 6.5***) has a magnitude of $\frac{1}{32}$.*

Next, we compute the bias of the linear expression in the case of the cipher GPig2.

GPig2:

In order to obtain the bias of the linear approximation a similar calculation is performed.

The biases of the linear approximations of the S-Boxes are identical for both the ciphers. Only the biases of the linear expressions involving the key bits are different for GPig2 and are computed using **theorems 6.1 and 6.2**. *We first enumerate the linear expressions involving the key bits and the corresponding biases:*

1. $U_{1,5} = P_5 \oplus K_{1,5} \oplus K_{1,4}$, *with bias* $\frac{1}{4}$
2. $U_{1,7} = P_7 \oplus K_{1,7} \oplus K_{1,6}$, *with bias* $\frac{1}{4}$
3. $U_{1,8} = P_8 \oplus K_{1,8} \oplus K_{1,7}$, *with bias* $\frac{1}{4}$
4. $U_{2,6} = V_{1,6} \oplus K_{2,6} \oplus K_{2,5}$, *with bias* $\frac{1}{4}$

Thus, the bias of **equation 6.1** *in case of GPig2 is :*

$\beta_1' = 2^5(\frac{1}{4}\frac{1}{4}\frac{1}{4}\frac{1}{4}(-\frac{1}{4})\frac{1}{4}) = -\frac{1}{128}$.

In order to obtain the linear approximation for round 3, the linear expression involving the key bits are:

1. $U_{3,6} = V_{2,6} \oplus K_{3,6} \oplus K_{3,5}$, *with bias* $\frac{1}{4}$
2. $U_{3,14} = V_{2,8} \oplus K_{3,14} \oplus K_{3,13}$, *with bias* $\frac{1}{4}$

Thus the bias of **equation 6.2** *becomes:*

$\beta_2' = 2^3(\frac{1}{4}\frac{1}{4}(-\frac{1}{4})(-\frac{1}{4})) = \frac{1}{32}$.

*In order to arrive at the final expression (***equation 6.5***), the expressions involving the key bits of round 4 are*

1. $U_{4,6} = V_{3,6} \oplus K_{4,6} \oplus K_{4,5}$, *with bias* $\frac{1}{4}$
2. $U_{4,8} = V_{3,14} \oplus K_{4,8} \oplus K_{4,7}$, *with bias* $\frac{1}{4}$
3. $U_{4,14} = V_{3,8} \oplus K_{4,14} \oplus K_{4,13}$, *with bias* $\frac{1}{4}$
4. $U_{4,16} = V_{3,16} \oplus K_{4,16} \oplus K_{4,15}$, *with bias* $\frac{1}{4}$

Thus, the bias of **equation 6.5** *is:*

$\beta_3{}' = 2^5(\beta_1{}'\beta_2{}'\frac{1}{4}\frac{1}{4}\frac{1}{4}\frac{1}{4}) = 2^5((-\frac{1}{128})(\frac{1}{32})(\frac{1}{4}^4)) = \frac{1}{2^{20}} \approx 0.$

The above example demonstrates that when the key mixing step in the SPN block cipher is performed with the help of addition modulo 2^n, the biases of linear expressions are almost zero, whereas for GPig1 they are as high as $\frac{1}{32} = 0.03125$. Thus while GPig1 breaks in the face of Linear Cryptanalysis, GPig2 is much resistant against the attack.

In the following section we perform a linear attack on both the ciphers, GPig1 and GPig2 and evaluate the strength of the second cipher against the cryptanalysis.

6.5 Experimental Extraction of Key Bits

In this section it is experimentally shown that GPig1 is successfully cryptanalyzed using the linear expression, mentioned in the example. It is also demonstrated that for GPig2 such an attack does not work. The reason being, in order for linear cryptanalysis to be successful the bias of $(R-1)$ round linear expressions (approximations) for an R round block cipher has to be large. However, in the case of GPig2 the biases of linear expressions are very less and hence such equations cannot be exploited in a conventional linear attack.

6.5.1 Experimental Setup

The procedure adopted to evaluate the last round keys are as follows:

1. A large number $(10,000)$ of cipher-texts are obtained by encrypting plaintexts i,e we generate $10,000$ known plaintext/ ciphertext pairs.
2. The attacker considers the linear approximation (mentioned in the example) of the first 3 rounds of the ciphers. To restate the expression is:

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 = 0 \tag{6.6}$$

The terms $U_{4,6}$, $U_{4,8}$ and $U_{4,14}$ affects the S-Boxes S_{42} and S_{44} (refer **Fig. 6.2**). Hence, the attacker guesses $(K_{5,5}, \ldots, K_{5,8})$ and $(K_{5,13}, \ldots, K_{5,16})$. In case of GPig1, he xors them with the ciphertext bits to obtain $(V_{5,5}, \ldots, V_{5,16})$. Then he performs the inverse of the S-Box operations to obtain the values of $U_{4,6}$, $U_{4,8}$ and $U_{4,14}$. If their values satisfy **equation 6.6** then a count is incremented for the guessed key bits $(K_{5,5}, \ldots, K_{5,8}, K_{5,13}, \ldots, K_{5,16})$. The partial subkey which has the count which differs greatest from half the number of plaintext/ ciphertext samples $(50,000)$ is assumed to represent the correct values of the guessed key bits. An incorrect subkey is assumed to be equivalent to a random guess of the bits of the linear expression and this holds with probability close to 1/2. The same attack is also performed on GPig2. Only we assume that the attacker knows the values of the key bits $(K_{5,9}, \ldots, K_{5,12})$. Thus here he guesses the partial keys $(K_{5,5}, \ldots, K_{5,8}, K_{5,13}, \ldots, K_{5,16})$ and subtracts the key bits from the ciphertext to arrive at the required values of $(V_{5,5}, \ldots, V_{5,16})$ and finally the values of $U_{4,6}$, $U_{4,8}$ and $U_{4,14}$. The rest of the attack is similar. This gives a best case scenario to the attacker.

From **table 6.4**, we see that the attack works fine for GPig1, where the correct subkey bits (last round) keys $(K_{5,5}, \ldots, K_{5,8}, K_{5,13}, \ldots, K_{5,16}) = [2,4]$ leads to the largest bias of 0.0308 and is thus detected. The bias is also close to the calculated bias of 1/32=0.03125.

However the same attack on GPig2 shows that the bias of the expression for the correct key bits [2,4] is only 0.0010 which is less than the biases of the incorrect key bits. The result implies that the probability of linear expressions to hold in case of GPig2 is much close to 1/2 and is thus very hard to differentiate from a random guess. Thus GPig2 offers a much better resistance to linear cryptanalysis than GPig1. Also note that in the experimentation it was observed that the highest bias (0.0139) occurred for a key bit = $E9$, which is an incorrect key.

In the following section we outline the advantages of performing key mixing using addition modulo 2^n over bitwise xor.

Table 6.4: **Experimental Results for Linear Attack**

Partial SubKey	Bias		Partial SubKey	Bias	
$[K_{5,5}, .., K_{5,8}, .., K_{5,13}]$	XOR	ADD	$[K_{5,5}, .., K_{5,8}, .., K_{5,13}]$	XOR	ADD
1C	0.0023	0.0027	2A	0.0099	0.0030
1D	0.0042	0.0084	2B	0.0053	0.0044
1E	0.0013	0.0006	2C	0.0060	0.0120
1F	0.0055	0.0034	2D	0.0107	0.0034
20	0.0011	0.0023	2E	0.0074	0.0061
21	0.0061	0.0053	2F	0.0024	0.0012
22	0.0028	0.0049	30	0.0137	0.0002
23	0.0075	0.0067	31	0.0151	0.0043
24	0.0308	0.0010	32	0.0104	0.0048
25	0.0156	0.0079	33	0.0151	0.0010
26	0.0148	0.0022	34	0.0090	0.0025
27	0.0011	0.0003	35	0.0130	0.0048
28	0.0266	0.0009	36	0.0078	0.0034
29	0.0107	0.0046	37	0.0025	0.0020

Max Bias for XOR: 0.0308 for the correct Key 24H
Max Bias for Add: 0.0139 for an incorrect Key E9H

6.6 Effects of Key mixing using addition modulo 2^n

1. The diffusion process is improved inside the bytes
2. Non-linearity is improved
3. The bias of linear approximations falls and helps in preventing linear cryptanalysis
4. The step provides an additional security margin to the cipher and thus helps to reduce the number of rounds of the block cipher. Mathematically, let p be the security margin against Linear Cryptanalysis (LC) provided by a single round of a cipher, performing key mixing using xor. If r is the total number of rounds of the cipher, the total security margin against LC is p^r.

   Similarly, let r_1 be the total number of rounds of the block cipher when key mixing is performed by addition modulo 2^n. In such a case the total security

margin provided against LC is atleast $(\frac{p}{4})^{r_1}$. Assuming both the ciphers provide equal security against LC,

$$\begin{aligned} p^r &= (\frac{p}{4})^{r_1} \\ \text{or, } r_1 &= r\frac{log(p)}{log(p)-2} \end{aligned}$$

Thus, as expected $r_1 < r$.

5. Key addition also helps in preventing differential cryptanalysis (discussed in the next chapter)

6. However the increase in security comes at the cost of additional computational overhead. When n bits of key are mixed by xoring, an n bit XOR gate is necessary which may be operated in parallel. Thus the input output delay is that of 1 XOR gate (which is roughly equal to 1.5 AND gate delay). However for a modulo addition operation there are $(2n-1)$ XOR gates, $(2n-3)$ AND gates and $(n-2)$ OR gates necessary. The input output delay, in such a case is equal to that of $(2.5n+0.5)$ AND gates. However if we keep the value of n within 16 all modern day processors would incur almost same cost/ speed for both the methods of key mixing. Further as shown above if we are able to reduce the number of rounds in the block cipher using the modular addition operation then we may compensate the increased overhead involved in the key addition step.

6.7 Conclusion

In the present chapter, the effect of performing key mixing by addition modulo 2^n instead of the more popular xor has been revisited. The largest bias of linear approximations for the output bit of such a key mixing have been computed. Both theoretically and experimentally it has been demonstrated that such a modification makes the cipher strong against Linear Cryptanalysis. Employing the concepts developed in *chapters 5* and *6*, namely the CASBox architecture and key mixing through modular integer addition, the ensuing chapter introduces a new block cipher.

Chapter 7

SPAMRC: A Cellular Automata Based SPN Cipher

The fact that the simple underlying rules of the Cellular Automata (CA) can be very efficiently implemented and repeated applications of these simple rules can demonstrate complex behaviors, have lured researchers to develop CA based ciphers. In [140], non-homogeneous Cellular Automata, which have different rules for different cells, were proposed for public-key cryptography. But the paper lacks specifications like key-size, key generation procedures and also real life examples. This makes it difficult to perform cryptanalysis of the cipher, thus leaving its security untested. The block ciphers and stream ciphers proposed in [141] were broken in [180] due to the affine property of the used Cellular Automata. In [181] another block cipher was proposed but it was unable to get rid of the affine property and thus could not achieve the claimed security. In [142] an extended Cellular Automaton ($GF(2^8)$) was used to develop a cryptosystem which used the Galois Field multiplication as the non-linear step. However the paper also lacked detailed cryptanalysis. The recent CA based block cipher, proposed in [143] mixes an affine CA with non-affine mappings. However the block cipher has been successfully cryptanalyzed in [182]. In [183], the authors show that though some one-dimensional CA exhibit high randomness, still does not provide desirable security due to their reversibility.

The reasons behind the failure should not be attributed to the Cellular Automaton, which is on the contrary a wonderful machine conducive for cipher design. In [137] the CA has been revisited and a generalized block cipher round has been composed using a special technique. The elegance of the composition was the fact that the combination of the linear and non-linear parts did not disturb the cyclic structure of the linear part. However in order to develop a complete block cipher many details like block sizes (of both the data and the key), number of rounds required and a detailed

security analysis have to be performed, which were missing in the previous attempts of designing CA based cryptosystems [182]. In the present work we thus adopt a "tame" approach [120] of cipher design, by combining the cryptographic primitives developed so far and some other new methods.

In this work it is shown for the first time in the literature of Cellular Automata based cipher design how the features of a cipher based on Substitution and Permutation can be derived using CA rules. SPAMRC (Substitution Permutation Addition Message Round Cipher) is a block cipher built out of Cellular Automata (CA) based operations. As the name suggests, SPAMRC is a block cipher based on substitution and permutation network (SPN) ciphers. The components performing substitution and permutation are developed from Cellular Automata based transformations. The S-Box (Substitution Box) used in SPAMRC is based on the CASBox architecture developed in *chapter 5*. The diffusion layer is based on the combination of ShiftRow used in AES [184] and a permutation layer designed using Cellular Automata. The permutation layer is based on Maximum Distance Separable (MDS) codes and is a self-invertible transformation and thus may be used for both encrypting and decrypting data. The key mixing in the block cipher is performed through addition modulo 2^b, where b is the word size and is typically equivalent to a byte of data. Finally using the technique proposed in [185] the chapter computes the number of rounds required to provide sufficient security margin against Linear and Differential cryptanalysis. In order to perform the analysis the results proposed in [185] are extended to take into account the effect of key mixing through addition modulo 2^b in place of xoring.

The chapter is organised as follows: *Section 7.1* describes the complete algorithm for SPAMRC while the round transformations are explained in *section 7.2*. The parameters required to estimate the number of rounds are computed in *section 7.3*. *Section 7.4* shows that the provable security margin of four rounds of SPAMRC is comparable to that of AES-Rijndael. The work is concluded in *section 7.5*.

7.1 Algorithms for SPAMRC

SPAMRC is a block cipher with both variable block and key lengths. The block and key length is a multiple of 32 bits, with a minimum of 128 bits. In the present work the algorithm focuses upon the 128 bit design, both for the key and block length, however the design is easily scalable to other block and keysizes. The data processed by each round is stored in the form of state matrix, where each element is a word. Before stating the algorithm we first define the following parameters of the block cipher.

Parameters of SPAMRC:

- N: plaintext size, key size
- b: size of each word
- n_r: number of rounds
- $m = N/b$: number of words in each state matrix

Key Mixing in SPAMRC is performed by adding the round keys modulo 2^b to the state of the block cipher. The following pseudo-code describes the i^{th} round of SPAMRC. The input to the round is $state(i)$ and the $roundkey(i-1)$. The output of the round is $state(i+1)$. The data transformed by the rounds of SPAMRC are stored in the form of the state matrix. The state matrix has an order $4 \times l$, where l represents the columns and 4 indicates the number of rows. Each element is a word of b bits. Thus, we have $N = 4lb$. Typically, when $N = 128$, the values of l and b are respectively 4 and 8. Number of words in the state matrix is thus $m = 128/8 = 16$. Each word in SPAMRC is a byte which makes implementation suitable on any processor platform.

Algorithm 7.1 *Input : state(i), roundkey(i-1)*
Roundi(state(i), roundkey(i-1))
{
temp1 = CASBox(state(i));
temp2 = Shift Row(temp1);
state(i+1) = CAMixColumn(temp2);
}

The above steps are briefed as follows:

- **CASBox** An 8×8 CASBox structure is used to generate cryptographically robust S-Boxes. The construction is discussed in **chapter 5** and is found to give efficiency and security to the block cipher against linear, differential and algebraic attacks.
- **Shift Row and CAMixColumn** The diffusion layer named as CAMixColumn is a self-invertible linear mapping $GF((2^8)^4) \rightarrow GF((2^8)^4)$ based on the $[8, 4, 5]$ MDS (Maximum Distance Separable) code with generator matrix $G_H = [IH]$, where $H = T^{174}(had(I, T^2, T^4, T^8))$. Here T is the characteristic matrix of an 8 cell maximum length CA, I is the identity matrix and had represents

a circulant Hadamard matrix. The ShiftRow is a diffusion optimal step and the CAMixColumn is an MDS transformation, leading to very fast inter byte diffusion [184].

In the following section the individual steps are elaborately described.

7.2 The round transformations of SPAMRC

7.2.1 Key mixing using addition modulo 2^b

As discussed before the state matrix of block cipher, SPAMRC comprises of words of size b. The key scheduling algorithm generates N bit round keys which are mixed with the N bit state matrix at the input of the particular round. Finally, there is one extra key mixing layer at the end of the last round of the block cipher.

The $(i,j)^{th}$ value of the state matrix of the r^{th} round is denoted by $s^r_{i,j}$. The corresponding value of the round key is $k^r_{i,j}$, which is mixed to the state matrix through the function $(s^r_{i,j} + k^r_{i,j})\text{mod}2^8$, as typically the value of b is 8. Here the symbol $'+'$ represents integer addition of two 8 bit numbers.

The motivation for using the key addition layer compared to the more popular bitwise key xoring is: (i) high diffusion, (ii) high non-linearity, (iii) improving the security margin against Linear Cryptanalysis (LC) and Differential Cryptanalysis (DC), (iv) same cost in terms of speed/ area in case of hardware and code size/ speed in case of a software implementation and thus incurring no penalty from the implementation point of view. In this chapter we later quantify the security margin provided by the key addition layer against LC and DC.

7.2.2 CASBox: Implementation of S-Box in SPAMRC

The S-Box in SPAMRC is based on the CASBox architecture described in *chapter 5*. The CASBox operates on 8 bits of input and results in an 8 bit output. The CAS-Box is bijective and re-orientation of the basic block results in the inverse CASBox (**theorem 5.6**). The CASBox operates on 8 bits of input and results in another 8 bits of output.

The features of the 8×8 CASBox are as follows:

1. The values of the parameters n and k, which characterize the CASBox are set respectively to the values 8 and 5. Here, it may be noted that n and b are same.

2. The non-linearity of each component function and also their non-zero linear combinations is very high.

3. CASBox provides high security against LC and DC.

The margins against DC and LC are respectively defined as follows[185]:

For any given $\Delta x(\neq 0), \Delta y$ representing the input and output differentials of an n bits S-Box,

$$DP^S(\Delta X \rightarrow \Delta y) = (1/2^n)(\#\{x \in Z_2^n : S(x) \oplus S(x \oplus \Delta x) = \Delta_y\})$$

For any given $\Gamma y(\neq 0), \Gamma x$ representing the output and input masks of the S-Box,

$$LP^S(\Gamma y \rightarrow \Gamma y) = [(1/2^{n-1})(\#\{x \in Z_2^n : \Gamma x.x = \Gamma y.S(x)\}) - 1]^2$$

Thus the following parameters define the resistance of the S-Box against Linear and Differential Cryptanalysis.

$$\begin{aligned} DP^S_{max} &= max_{\Delta x \neq 0, \Delta y \neq 0} DP^S(\Delta x \rightarrow \Delta y) \\ LP^S_{max} &= max_{\Gamma x \neq 0, \Gamma y \neq 0} LP^S(\Gamma y \rightarrow \Gamma x) \end{aligned}$$

For CASBox, from **section 5.6.2** thus $DP^S_{max} = \frac{1}{2^n}2^k = 2^{k-n}$. As discussed in **theorem 5.3** in the description of CASBox, there are two kinds of non-linearity in the non-zero linear combinations of the component functions. Using the LFSR based non-linear permutation over V_3 we obtain the value of LP^S_{max} as follows:

$$\begin{aligned} LP^S_{max} &= [\frac{1}{2^{n-1}}(2^n - \text{minimum mismatch}) - 1]^2 \\ &= [\frac{1}{2^{n-1}}(2^n - \text{non-linearity}) - 1]^2 \\ &= [\frac{1}{2^{n-1}}(2^n - 2^{n-2}) - 1]^2 \\ &= [\frac{1}{2^{n-1}}(4.2^{n-2} - 2^{n-2}) - 1]^2 \\ &= [\frac{1}{2^{n-1}}(3.2^{n-2}) - 1]^2 \\ &= [\frac{3}{2} - 1]^2 = \frac{1}{4} \end{aligned}$$

With the chosen parameters $n = 8$ and $k = 5$, thus $DP^S_{max} = 2^{-3}$ and $LP^S_{max} =$

2^{-2}.

The algebraic degree of the non-zero linear combinations of CASBox from **theorem 5.7** is $(n-k+1)=4$ except in 7 cases where it is 2.

7.3 Construction of the involutional diffusion layer: CAMixColumn

Generally SPN cipher (e.g Rijndael) require two different permutation modules for encryption and decryption. SPAMRC uses a diffusion layer which is self-invertible and is based on MDS (Maximum Distance Separable) codes.

A block code of length n and 2^k codewords is called a *linear* (n,k) code if and only if its 2^k codewords form a k-dimensional subspace of the vector space of all the n-tuples over the field $GF(2)$. A binary block code is linear iff the modulo-2 sum of two codewords is also a codeword. The *hamming distance* d of a system of codewords determines the number of errors that can be detected and corrected. If $(n-k)$ check bits are appended to k information bits to get a distance-d code, it is called (n,k,d) code[91, 186].

For a linear (n,k,d) code over any field, $d \leq n-k+1$. Codes with $d=n-k+1$ are called MDS codes.

Lemma 7.1 *An (n,k,d) code with generator matrix $G=[I|A]$, where A is a $k \times (n-k)$ matrix is MDS iff every square submatrix (formed from any i rows and any i columns) from $i=1,2,\ldots,min(k,n-k)$ of A is non-singular.*

In the construction of the CA based MDS code we have used the concept of circulant Hadamard matrices.

Given k elements $\alpha_0,\alpha_1,\ldots,\alpha_{k-1}$, a Hadamard matrix A is constructed with each entry $A_{i,j}=\alpha_{i\oplus j}$. Each submatrix over finite field has the following property, $A^2=\gamma I$, where γ is a constant. When $\gamma=1$, A is an involution matrix.

7.3.1 Realization of Hadamard Matrix using Cellular Automata

The CAMixColumn step, like the MixColumn step of AES, operates on a column, i,e 4 words at a time. When $b=8$, it operates on 4 bytes (i,e 32 bits). Thus the diffusion

layer may be represented by a mapping,

$$D : GF((2^b)^4) \rightarrow GF((2^b)^4)$$

The mapping is based on an $[8, 4, 5]$ MDS code with generator matrix $G_A = [I|A]$. Let, T be the characteristic matrix of a maximum length CA of $b = 8$ cells. In such a CA all the non-zero states lie in one cycle. Thus $T^{255} = I$. A possible rule for such a CA is $(150, 150, 90, 150, 90, 150, 90, 150)$ where 90 and 150 are the rule numbers of the corresponding CA cell, refer **table 3.3**. We denote the Hadamard matrix A as:

$$A = \begin{pmatrix} I & T^2 & T^4 & T^8 \\ T^2 & I & T^8 & T^4 \\ T^4 & T^8 & I & T^2 \\ T^8 & T^4 & T^2 & I \end{pmatrix}$$

Thus, squaring the matrix A we have,

$$\begin{aligned} A^2 &= (I \oplus T^4 \oplus T^8 \oplus T^{16}) \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix} \\ &= (I \oplus T^4 \oplus T^8 \oplus T^{16}) I_s \text{ ,where } I_s \text{ is the identity state matrix} \\ &= T^{162}(I_s) \text{ ,from simulation results on the matrix } T \end{aligned}$$

It may be noted that any square submatrix of A is nonsingular, thus implying that $G = [I|A]$ is MDS.

In order to obtain a self-invertible mapping, D we require:

$$\begin{aligned} D^2 &= I_s \\ \text{or, } A^2 &= T^{162} D^2 \\ \text{or, } D &= \frac{1}{T^{81}} A \end{aligned}$$

Since, $T^{255} = I$, $T^{-81} = T^{174}$. Thus, we define the mapping CAMixColumn as:

$$D = T^{174} A$$

Let a column of the state matrix, which is in $GF(2^b)^4$ be represented as $X = (X_3, X_2, X_1, X_0)$ where each element $X_i \in GF(2^b)$. The corresponding output is the column $Y = (Y_3, Y_2, Y_1, Y_0)$, where each element $Y_i \in GF(2^b)$.

Thus, the output may be represented as $Y = AX$. Hence, we have the following equations:

$$\begin{aligned} Y_3 &= I(X_3) \oplus T^2(X_2) \oplus T^4(X_1) \oplus T^8(X_0) \\ Y_2 &= I(X_2) \oplus T^2(X_3) \oplus T^4(X_0) \oplus T^8(X_1) \\ Y_1 &= I(X_1) \oplus T^2(X_0) \oplus T^4(X_3) \oplus T^8(X_2) \\ Y_0 &= I(X_0) \oplus T^2(X_1) \oplus T^4(X_2) \oplus T^8(X_3) \end{aligned}$$

Generalizing the above equations,

$$Y_i = I(X_{i\oplus 0}) \oplus T^2(X_{i\oplus 1}) \oplus T^4(X_{i\oplus 2}) \oplus T^8(X_{i\oplus 3})$$

The architecture of the CAMixColumn is easily derived from the above generalized equation and is depicted in **Fig. 7.1**.

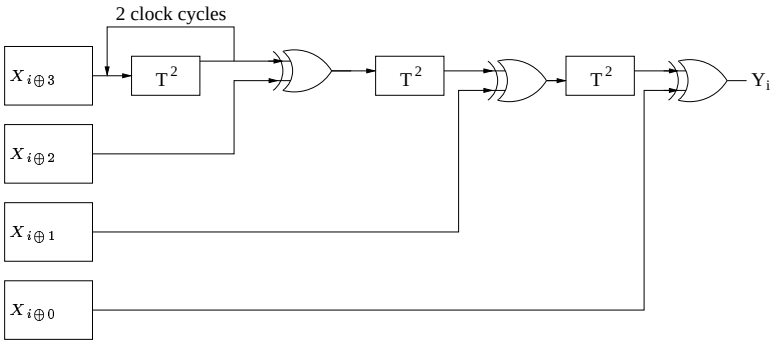


Figure 7.1: **Architecture of CAMixColumn**

7.3.2 Features of the architecture

The proposed architecture of CAMixColumn has the following salient features:

1. There is only one Cellular Automata block, T^2 which is being reused.

2. The critical delay to obtain Y_i is independent of input and is equal to four clock cycles of the Cellular Automata, whose characteristic matrix is denoted by T^2. Such an implementation helps to prevent timing based attacks.

3. Finally each Y_i which belongs to $GF(2^8)$ is transformed by a combinational network denoted by:

$$T^{174} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

7.4 Estimation of number of rounds of SPAMRC

In order to complete the number of rounds we consider the minimum number of differentially and linearly active S-Boxes.

Definition 7.1 Differentially and Linearly Active S-Boxes: *A differentially active S-Box is defined as an S-Box with a non-zero input difference and a linearly active S-Box is an S-Box with a non-zero output mask value.*

We may consider the structure of 2 rounds of SPAMRC and compare with the two rounds SDS (Substitution Diffusion Substitution) layer presented in [187]. The major difference is due to the key mixing layer in SPAMRC, which is done through key addition. In the previous analysis [185, 187], the key mixing layer was performed by xoring and had no influence on the number of active S-Boxes and thus was ignored in the computation of rounds. In this work we also evaluate the security margin provided by the key addition layer against Linear and Differential Cryptanalysis and use it to compute the number of rounds of SPAMRC.

For the sake of analogy to the representation of [185, 187] we denote the two rounds of SPAMRC as $ASDAS$ (Addition Substitution Diffusion Substitution Addition). If we combine the addition layer (A) and the substitution layer (S) and represent by a combined keyed substitution layer (S'), then the two rounds are denoted by the term $S'DS'$ (**Fig. 7.2**).

If the input of the Diffusion layer (D) be x and the corresponding output is y, then $y = Dx$.

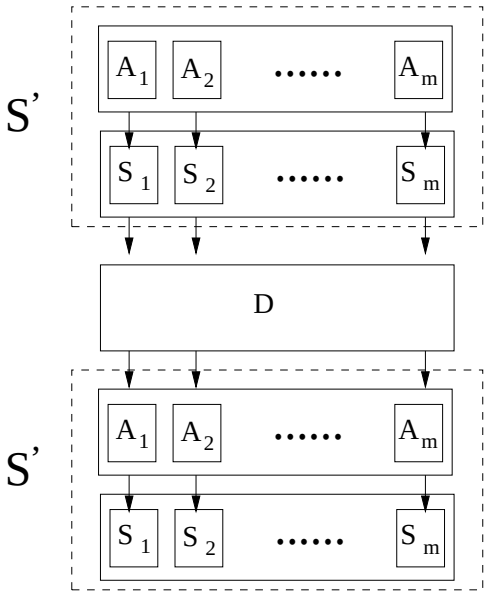


Figure 7.2: $S'DS'$ **function**

Therefore, input difference of D is Δx and output difference is $\Delta y = D\Delta x$. Similarly, the input and output masks are Γx and Γy respectively.

Lemma 7.2 *The minimum number of differentially and linearly active S-Boxes of the $S'DS'$ is:*

$$\begin{aligned} \beta_d(D) &= min_{\Delta x \neq 0} H(\Delta x) + H(\Delta y) \\ \beta_l(D) &= min_{\Gamma y \neq 0} H(\Gamma x) + H(\Gamma y), \end{aligned}$$

respectively. In the above equations, for $x = (x_1, x_2, \ldots, x_m)$, the symbol $H(x)$ stands for the cardinality of the set $\{1 \leq i \leq m : x_i \neq 0\}$.

As previously discussed the diffusion layer is implemented using the CAMixColumn, which is based on MDS codes. Equivalently, it was pointed out that the matrix D of the CAMixColumn step has all submatrices non-singular. The following result thus gives an estimate of the number of active S-Boxes.

Lemma 7.3 *[187] Let D be an $n \times n$ matrix representing the Diffusion layer. Then $\beta_d(D) = \beta_l(D) = n + 1$ iff rank of each $k \times k$ submatrix of D is k for all $1 \leq k \leq n$.*

In order to estimate the number of rounds of the block cipher we require to compute the LP and the DP parameters of the modified S-Box represented by S'. The modified S-Box, as previously stated has a key addition layer instead of key xor, which provides an extra security margin over the CASBox layer. It may be noted that the previous block ciphers, like MARS, SEA [174, 177] which also uses key addition modulo 2^n, does not incorporate the key addition layer in the analysis of security margin. However, in this work we have computed the security margin provided by 4 rounds of SPAMRC against Linear and Differential Cryptanalysis taking into account the key addition layer.

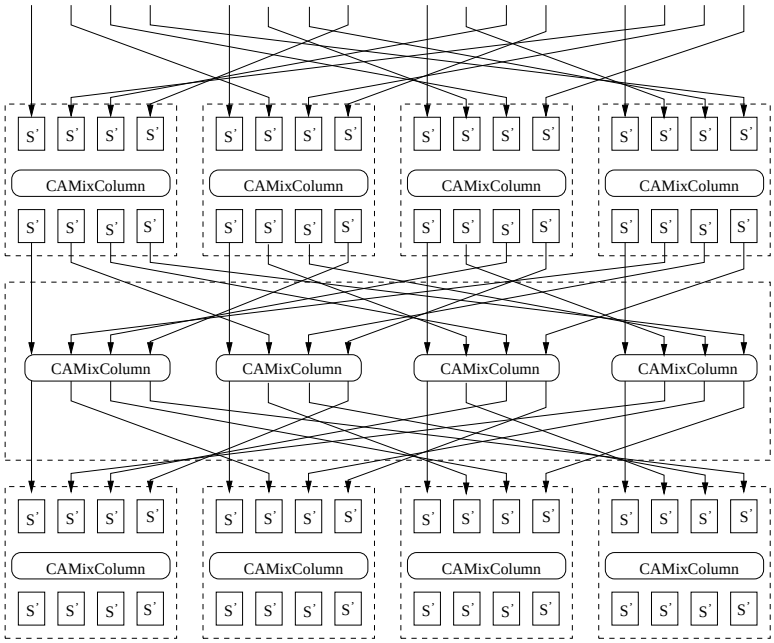


Figure 7.3: **Branching of 4 Rounds of SPAMRC**

The security margin provided by 4 rounds of SPAMRC against Linear and Differential cryptanalysis is computed by finding the number of linearly and differentially active S-Boxes (or S' boxes) (**Fig. 7.3**). If one bit of the input to the $S'DS'$ is disturbed, then due to good avalanche effect of the CASBox (S layer) almost 4 bits of the differential in the output of S Box is affected. The DP of the combined S-Box may be estimated from the product of the DP of the CASBox and the maximum probability that the key addition layer passes a differential, assuming that the key is non-trivial as may be expected from a properly designed key scheduling algorithm.

Let, the inputs to the key addition layer be x and x' such that $\Delta x = x \oplus x'$. The corresponding outputs are:

$$y \;=\; (x + k) \bmod 2^8$$

$$y' = (x' + k) \bmod 2^8$$

Representing the equations at bit level:

$$\begin{aligned} y[0] &= x[0] \oplus k[0] \\ y[1] &= x[1] \oplus k[1] \oplus c[0] \\ &\vdots \\ y[7] &= x[7] \oplus k[7] \oplus c[6] \end{aligned}$$

where, $c[i]$ represents the carry out from the i^{th} bit level.
Thus, $\Delta y = y \oplus y' = \Delta x \oplus \Delta c$, where $\Delta c = (\Delta c[6], \Delta c[5], \ldots, \Delta c[0])$.
Now, we have:

$$\Pr(\Delta x \rightarrow \Delta y) = \Pr(\Delta y | \Delta x) = \Pr(\Delta C) \tag{7.1}$$

At any i^{th} $(0 \leq i \leq 6)$ bit position the carry outs may be represented as:

$$\begin{aligned} c[i] &= \text{majority}(x[i], k[i], c[i-1]) \\ c'[i] &= \text{majority}(x'[i], k[i], c'[i-1]) \end{aligned}$$

By **theorem 6.1** we have the following recursive relation:

$$p_i = \Pr(c[i] = 0) = \frac{1}{4} + \frac{p_{i-1}}{2}$$

Therefore, $q_i = \Pr(c[i] = 1) = 1 - p_i = \frac{3}{4} - \frac{p_{i-1}}{2}$.
Thus, the probability that $\Delta c[i] = 0$ is:

$$\begin{aligned} \Pr(\Delta c[i] = 0) &= \Pr(c[i] = 0 \& c'[i] = 0) + \Pr(c[i] = 1 \& c'[i] = 1) \\ &= (\frac{1}{4} + \frac{p_{i-1}}{2})^2 + (\frac{3}{4} - \frac{p_{i-1}}{2})^2 \\ &= \frac{p_{i-1}^2}{2} - \frac{p_{i-1}}{2} + \frac{5}{8} \end{aligned}$$

Hence the probability that $\Delta c[i] = 1$ is $1 - (\frac{p_{i-1}^2}{2} - \frac{p_{i-1}}{2} + \frac{5}{8}) = -\frac{p_{i-1}^2}{2} + \frac{p_{i-1}}{2} + \frac{3}{8}$.

From the above equations the following is easily deduced:

$$\begin{aligned} \Pr_{max}(\Delta c[i] = 0) &= 1 \\ \Pr_{max}(\Delta c[i] = 1) &= \frac{5}{8} \end{aligned}$$

Since from **equation 7.1**, the probability distribution of Δy given Δx is identical to that of Δc and due to the Avalanche effect of the S-Boxes, there are an equal numbers of zeros and ones in Δy, the additional security margin provided by the key addition layer against Differential Cryptanalysis is roughly $(\frac{5}{8})^4(1)^4 \approx 2^{-3}$.

Thus, the modified value of DP of CASBox is:

$$\begin{aligned} DP &= 2^{-3} \times (\text{DP of CASBox}) \\ &= 2^{-6} \end{aligned}$$

From **theorem 6.2**, proved in **section 6.2** of **chapter 6** the bias of any linear approximation involving any bit position of the key addition layer, except the 0^{th} bit is maximum $\frac{1}{4}$.

If a linear approximation of the CASBox involves atleast 1 bit position of the input, then using Piling up lemma the bias of the resultant approximation is $\frac{1}{4} \times$ largest bias of CASBox $= \frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$. Thus the probability of matching is $\frac{1}{2} \pm \frac{1}{16}$.

Thus the modified value of LP of CASBox is:

$$\begin{aligned} LP &= [\frac{1}{2^{8-1}} 2^8 (\text{prob of matching }) - 1]^2 \\ &= [2(\frac{1}{2} \pm \frac{1}{16}) - 1]^2 \\ &= \frac{1}{2^6} \end{aligned}$$

7.5 Security Margin of 4 rounds of SPAMRC against Linear and Differential Cryptanalysis

Having estimated the values of maximum linear and differential probabilities (LP and DP), we can compute the security margin against Linear and Differential (LC and DC) cryptanalysis provided by four rounds of SPAMRC (**Fig. 7.3**).

For this purpose we also require to compute the number of active S-Boxes in 4 rounds of SPAMRC. Due to the MDS property of the step *CAMixColumn* the number of active S-Boxes due to one CAMixColumn is atleast 5 by **lemma 7.3**. For every input active S-Box, the 1^{st} CAMixColumn step, due to its MDS property results in 4 active S-Boxes(**Fig. 7.3**). Each active S-Box in turn affects all the S-Boxes in the next round. Hence in this round, each CAMixColumn receives all 4 active input S-Boxes and therefore results in atleast one active output S-Box. Thus, if we follow the propagation of active S-Boxes, shaded in **Fig. 7.3**, we find that there are 5 CAMixColumns each of which has 5 active input output S-Boxes. Using this fact the minimum number of active S-Boxes in the four rounds of SPAMRC is 25. Thus both the branch numbers (linear and differential) are equal to 25. The branch numbers $\beta_l = \beta_d = 5$ and they are equal because the diffusion layer *CAMixColumn* is MDS.

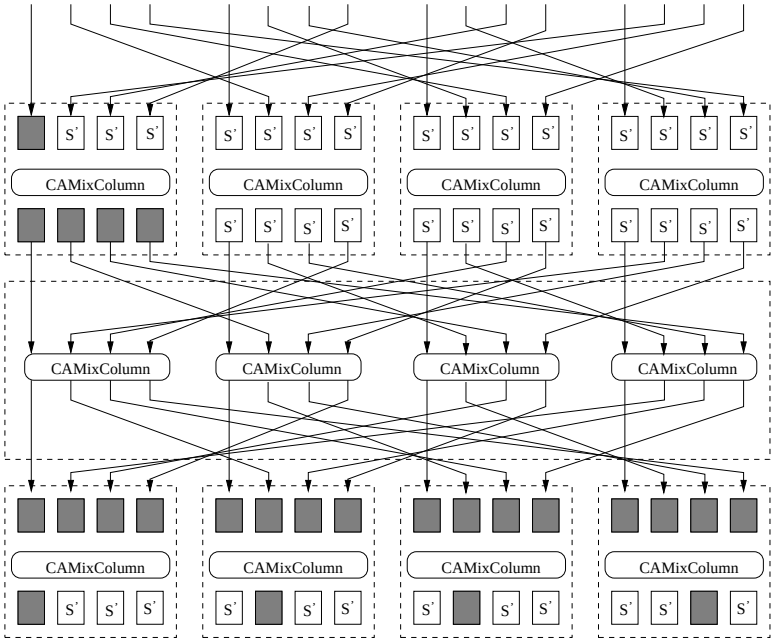


Figure 7.4: **Computing minimum number of active S-Boxes in 4 rounds of SPAMRC**

From the above results we may compute the value of MDCP (maximum differential characteristic probability) and MLCP (maximum linear characteristic probability) which are defined as follows [185].

Definition 7.2 *The maximum differential characteristic probability (MDCP) and the maximum linear characteristic probability (MLCP) are:*

$$MDCP = \sum_{\Delta w}[\prod_{i=1}^{m} DP^{S_i}(\Delta x_i \rightarrow \Delta y_i)]$$
$$MLCP = \sum_{\Gamma w}[\prod_{i=1}^{m} LP_{S_i}(\Gamma y_i \rightarrow \Gamma x_i)]$$

In the above definition Δx_i and Γx_i are respectively the input differentials and masks of the i^{th} S-Box. Similarly, Δy_i and Γy_i are the output differentials and masks of the i^{th} S-Box. As mentioned before m denotes the number of words in the state matrix. Δw and Γw denotes the input differentials and masks of the trail under observations. The following result is used to compute the values of $MDCP$ and $MLCP$.

Lemma 7.4 *In* **Fig. 7.4** *the number of differential/ linear active S-Boxes is no less than* 25

Lemma 7.5 *If the maximum differential/linear probability for the S-Boxes S' is p/q, the $MDCP$ and $MLCP$ satisfy the following inequalities:*

$$MDCP \leq p^{25}, MLCP \leq q^{25}$$

Here, $p = DP = 2^{-6}$ and $q = LP = 2^{-6}$. Thus using the above facts the margins of four rounds of SPAMRC against Linear and Differential cryptanalysis are:

$$MDCP_{SPAMRC} \leq (2^{-6})^{25} = 2^{-150}$$
$$MLCP_{SPAMRC} \leq (2^{-6})^{25} = 2^{-150}$$

The values of the parameters $MDCP$ and $MLCP$ are comparable to those of four rounds of AES-Rijndael where both the values are equal to 2^{-150}.

7.6 Conclusion

In this chapter the design of a Cellular Automata based block cipher, named SPAMRC has been outlined. The block cipher is based on a Rijndael like structure but the

underlying operations are based on Cellular Automata based transformations. The chapter uses the CASBox structure described previously, thus leading to provable security and also high programmability. The chapter also presents a novel technique to construct MDS mappings using CA based transformations. The key mixing is performed through the use of addition modulo 2^8 operation. The security margin provided by the operation against LC and DC has been computed. Finally it has been proved that four rounds of SPAMRC provide similar security against LC and DC as that provided by AES. Key establishment is a fundamental problem in usage of block ciphers. The next chapter explores a possible key agreement technique.

Chapter 8

Cellular Automata based Expander Graphs and their Cryptographic Applications

8.1 Introduction

Expander Graphs have been a significant tool both in theory and practice. It has been used in solving problems in communication and construction of error correcting codes as well as a tool for proving results in number theory and computational complexity. The combinatorial properties of the expander graphs can also lead to the construction of one-way functions [115]. Informally, the one-way functions are a class of functions in which the forward computation is easy, but the inverse is hard to find. The one-way functions form an important core of all key agreement algorithms which are an important step in secure electronic communication. The well known Diffie-Hellman key exchange algorithm [11] provides a ground-breaking solution to the problem of secured key distribution. However the security of the algorithm depends on the one-wayness of the modular exponentiation, which is a costly process in terms of computational resources. Since the seminal paper of Diffie-Helmann, there has been efforts in developing key exchange protocols whose security lies on one-way functions which are computationally efficient. However designing a strong one-way function which is computationally strong and yet hardware efficient is a challenging task.

The present work characterizes a special class of Cellular Automata (CA) [129], known as the *Two Predecessor Single Attractor Cellular Automata* (TPSA-CA) to generate expander graphs on the fly. The elegance of the scheme is that it uses regular,

cascadable and modular structures of CA to generate random d regular graphs of good expansion property with very less storage. The state transitions of each TPSA is captured in a single state, which is known as the graveyard of the CA. Finally, the expander graphs have been used to construct the one-way function according to the proposal of [115]. The entire algorithm has been prototyped on a Vertex Xilinx XCV-1000 platform. Results show that the scheme is very elegant in the fact that very high throughputs are achievable with a minimal cost of hardware and power. The performance has been compared with that of efficient implementations of modular exponentiation in $GF(2^m)$ and results show that the proposed one-way function is much superior.

The one-way function based on the expander graphs may also find applications in the design of authentication and key establishment protocols, which are fundamental building blocks for securing electronic communication. Cryptographic algorithms for encryption and decryption cannot perform their specified functions unless secure keys have been established. Key establishment [106] is a process or protocol whereby a shared secret becomes available to two or more parties, for subsequent cryptographic use. Key agreement mechanism is a key establishment technique in which a shared secret is derived by two or more parties as a function of information contributed by or associated with each of these (ideally) such that no party can predetermine the resulting value. Since the seminal paper of Diffie and Hellman in 1976, several solutions have been proposed for key agreement whose security lies on the Diffie-Hellman (D-H) problem (either computational or decisional) in finite groups. The security of D-H protocol lies on the intractability of the well known discrete log problem, but the use of the modular exponentiation is a concern for applications with limited computational power. Typical examples of such devices are mobile terminals and embedded hardware. A popular solution proposed is to develop a protocol with unbalanced computational loads[98]: the server bears an increased load in order to ease the load at the client side. Another technique that is effective is to allow the client side to pre-compute values which can be used during the protocol execution phase. Some of the protocols proposed for such constrained networks may be found in [188, 189, 190, 191]. Horn et al. [192] provide a survey and comparisons on protocols for mobile devices.

However modular exponentiation is an expensive operation. Despite the increase in availability of computational resources considerable demand exists in the development of protocols that can be implemented on devices with limited computational power. Achieving a secured key agreement with lesser computational overhead compared to exponentiation is a challenging task. The present work develops a key agreement protocol with authentication and key confirmation using the one-way function based on expander graphs generated by Cellular Automata. In an authenticated key agreement with key confirmation (AKC) a party i wants to make sure that the other

party j has really computed the agreed key. The security of a protocol depends on two aspects: first it depends upon the cryptographic strength of the underlying functions or commonly known as primitives. Secondly it depends on the messages exchanged between the parties in the key agreement protocol.

In these lines the present work first develops a Cellular Automata based one way function based on the combinatorial properties of expander graphs. The one-way function is used to derive the session key K_{AB} with the desired property of keyfreshness [98]. The one-way function is also used for authentication and key confirmation. The protocol has been shown to be secured in the Bellare Rogaway model [112].

The outline of the chapter is as follows: *Section 8.2* describes some of the preliminaries of expander graphs. The TPSA-CA is characterized in *section 8.3* and is used to generate expander graphs. Subsequently, *section 8.4* presents the construction of the CA based one-way function which is implemented on Xilinx XCV-1000 platform in *section 8.5*. Using the proposed one-way function a CA based key algorithm is presented in *section 8.6*. The security of the protocol is proved in Bellare Rogaway model in *section 8.7*. The chapter is concluded in *section 8.8*.

8.2 Preliminaries on Expander Graphs

Informally *expander graphs* are a class of graphs $G = (V, E)$ in which every subset S of vertices expands quickly, in the sense that it is connected to many vertices in the set $\overline{S}$ of complementary vertices. It may be noted that the graph may have self loops and multiple edges. The following definition states formally the *expansion property* of these class of graphs [193].

Definition 8.1 *The edge boundary of a set $S \in G$, denoted $\delta(S)$ is $\delta(S) = E(S, \overline{S})$ is the set of outgoing edges from S. The* expansion parameter *of G is defined as:*

$$h(G) \quad = \quad min_{S:|S|\leq n/2} \frac{|\delta(S)|}{|S|}$$

where $|S|$ denotes the size of a set S.

There are other notions of expansion, the most popular being counting the number of neighbouring vertices of any small set, rather than the number of outgoing edges.

Following are some examples of expander graphs [194].

Example 8.1 *Let G be a complete graph on n vertices i.e, the graph in which every vertex is connected to every other vertex. Then for any vertex in S, each vertex is*

connected to all the vertices in $\overline{S}$*, and thus* $|\delta S| = |S| \times |\overline{S}| = |S|(n - |S|)$*. Thus the expansion factor is given by:*

$$h(G) \quad = \quad min_{S:|S|\leq n/2}(n - |S|) = \lceil \frac{n}{2} \rceil$$

Example 8.2 *Let* G *be a random* d*-regular graph, in which each of* n *vertices is connected to* d *other vertices chosen at random. Let* S *be a subset of atmost* $n/2$ *vertices. Then a typical vertex in* S *will be connected to roughly* $d \times |\overline{S}|/n$ *vertices in* $\overline{S}$*, and thus* $|\delta S| \approx d \times |S||\overline{S}|/n$*, and so*

$$\frac{|\delta(S)|}{|S|} \quad \approx \quad d\frac{|\overline{S}|}{n}$$

Since, $|\overline{S}|$ *has its minimum at approximately* $n/2$ *it follows that* $h(G) \approx d/2$*, independently of the size* n*.*

Definition 8.2 *A* **family of expander graphs** *(*G_i *where* $i \in n$*) is a collection of graphs with the following properties:*

- *The graph* G_i *is a* d*-regular graph of size* n_i *(*d *is the same constant for the whole family).* $\{n_i\}$ *is a monotone growing series that does not grow too fast (e.g* $n_{i+1} \leq n_i^2$*)).*
- *For all* i*,* $h(G_i) \geq \epsilon \geq 0$*.*

Although d-regular graph random graphs on n vertices define an expander, for real life applications it is necessary to have more explicit constructions on $O(2^n)$ vertices, where n is the parameter defining the problem size. This is because to store a description of a random graph on so many vertices requires exponentially much time and space. Two well known constructions are given below:

Example 8.3 *In this example the family of graphs is indexed by a prime number* p*. The set of vertices for the graph* G_p *is just the set of points in* Z_p*, the field of integers modulo* p*. A* 3*-regular graph is constructed by connecting each vertex* $x \neq 0$ *to* $x - 1, x + 1$ *and* x^{-1}*. The vertex* $x = 0$ *is connected to* $p - 1, 0$ *and* 1*. This was proven to be an expander by Lubotsky, Phillips and Sarnak in* 1988*[195].*

Example 8.4 *A similar but slightly complex construction was due to Margulis[196, 197]. The vertex set is $Z_m \times Z_m$, where m is some positive integer, and Z_m is the additive group of integers modulo m. The degree is 4 and the vertex (x, y) has edges to $(x \pm y, y)$ and $(x, x \pm y)$, where all operations are done modulo m.*

The properties of the eigenvalue spectrum of the adjacency matrix $A(G)$ can also be used to understand properties of the graph G.

The **adjacency matrix** of a graph G, denoted by $A(G)$ is an $n \times n$ matrix such that each element (u, v) denotes the number of edges in G between vertex u and vertex v[193]. For a d-regular graph, the sum of each row and column in $A(G)$ is d. By definition the matrix $A(G)$ is symmetric and therefore has an orthonormal base $v_0, v_1, \ldots, v_{n-1}$, with eigenvalues $\mu_0, \mu_1, \ldots, \mu_{n-1}$ such that for all i we have $Av_i = \mu_i v_i$. Without loss of generality we assume the eigenvalues sorted in descending order $\mu_0 \geq \mu_1 \geq \ldots \geq \mu_{n-1}$. The eigenvalues of $A(G)$ are called the spectrum of G. The following two results are important in estimating the expansion properties of the graph.

1. $\mu_0 = d$
2. $\frac{d-\mu_1}{2} \leq h(G) \leq \sqrt{2d(d-\mu_1)}$

Thus, the parameter $d - \mu_1$, also known as the *Spectral Gap* gives a good estimate on the expansion of the graph G. The graph is an expander if the spectral gap has a lower bound ϵ' such that $d - \mu_1 > \epsilon'$.

A graph G_1 has better expansion properties than graph G_2, implies that for any subset S, $|S| \leq n/2$ of the graph G_1 has a larger number of neighbouring elements outside the set S, compared to that in G_2. Mathematically, the value of $h(G_1) > h(G_2)$. Informally, it implies that the graph G_1 expands faster compared to graph G_2. The hardness of inverting the one-way function based on expander graphs increases with the expansion of the expander graph [115]. A random regular graph has good expansion properties. However the problem of realizing such a graph is in its description which grows exponentially with the number of vertices.

In the next section we present the construction of a family of random d regular graph using the properties of a special class of CA, known as the Two Predecessor Singe Attractor Cellular Automaton (TPSA CA). It has been shown that the graph has good expansion properties. The merit of the construction lies in the fact that the generation is extremely simple and leads to an efficient design as developed in subsequent sections.

8.3 Expander Graphs using TPSA CA

TPSA CA are a special class of non-group CA in which the state transition graph forms a single inverted binary routed tree at all zero state (**Fig. 8.1**). Every reachable state in the state transition graph has exactly two predecessors. The only cyclic state is the all zero state (for a non-complemented TPSA CA), which is an attractor (or graveyard). If T_n is the characteristic matrix of an n cell automaton then the necessary and sufficient conditions to be satisfied by the Transition matrix for the CA to be TPSA CA is[129]:

1. Rank(T_n)=$n-1$
2. Rank($T_n + I_n$)=n, I_n being an $n \times n$ identity matrix
3. Characteristic Polynomial = x^n
4. Minimal Polynomial = x^n

The following results [129] characterize the state transition of the non-complemented TPSA CA.

Lemma 8.1 *[129] For an n cell TPSA CA with characteristic polynomial x^n and minimal polynomial x^n, (i) the number of attractors is 1, the all zero state, (ii) the number of states in the tree is 2^n.*

Lemma 8.2 *For an n cell TPSA CA having $m(x) = x^n$ the depth of the tree is n.*

Following is an example of a 4 cell non-complemented TPSA CA.

Example 8.5 *The state transition matrix for a 4 cell CA is denoted by:*

$$T_4 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The state transition diagram of the TPSA CA is shown in **Fig. 8.2(a)**. *As an example let us compute the next state of 14, which in binary form is $X = (1110)^T$. Thus the next state is obtained as $Y = T_4 \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = 1$ Thus the next state of 14*

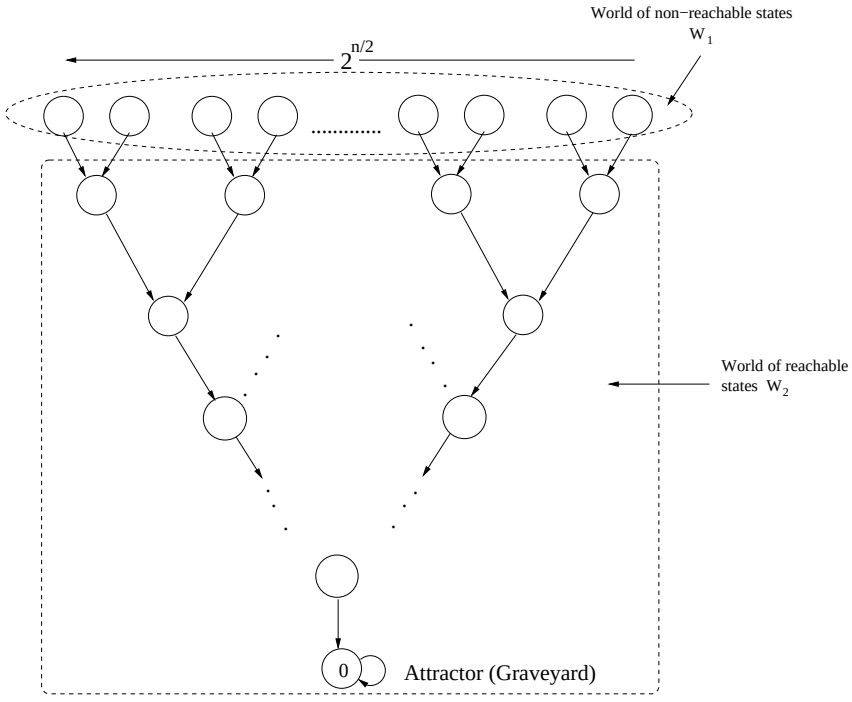


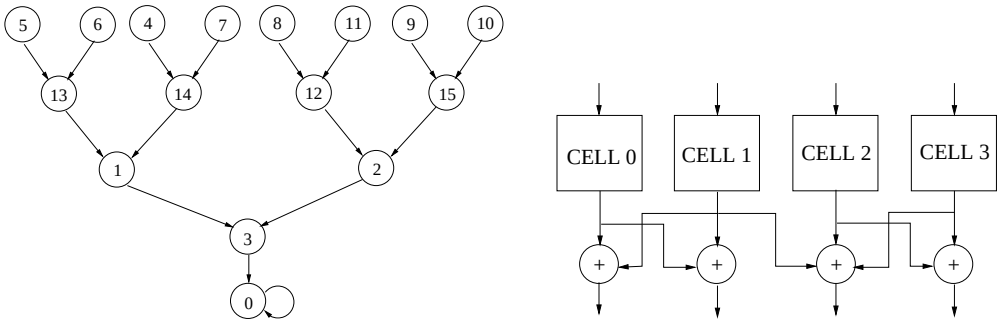
Figure 8.1: **The state transition graph of a non-complemented TPSA CA**

is 1, *which may be observed in* **Fig. 8.2(a)**. *Here* {*0*} *is the* attractor *or* graveyard *state. The states* {*5, 6, 4, 7, 8, 11, 9, 10*} *make the* non-reachable world, *while the states* {*13, 14, 12, 15, 1, 2, 3, 0*} *make the* reachable *world. The corresponding interconnection is given in* **Fig. 8.2(b)**. *As may be observed that the structure comprises of local interconnections leading to efficient designs.*

Next, we present an method to recursively synthesize an n cell TPSA. The state transition matrix of the n cell TPSA is denoted by T_n and is generated from an $n-1$ cell TPSA CA characterized by the matrix T_{n-1}. The following theorem describes the property exploited in the construction.

Theorem 8.1 *Given that* T_{n-1} *is the characteristic matrix of an* $(n-1)$ *cell TPSA, the matrix* T_n *denoted by:*

$$T_n = \left(\begin{array}{cccccc|c} & & & & & & 0 \\ & & & T_{n-1} & & & \vdots \\ & & & & & & 0 \\ \hline - & - & - & - & - & - & - \\ 0 & \ldots & 0 & 1 & & & 0 \end{array}\right)$$



(a) The state transition graph of a 4 cell non-complemented TPSA CA

(b) The interconnection of a 4 cell non-complemented TPSA CA

Figure 8.2: **A 4 cell non-complemented TPSA CA**

represents the characteristic matrix of an n cell TPSA.

Proof: It is evident that since the element at the n^{th} row and $(n-1)^{th}$ column is 1 and by the construction methodology all the rows have 0 in the $(n-1)^{th}$ columns the row added is linearly independent from the other rows of T_n. Hence it adds by 1 to the rank of T_{n-1}. Thus, $rank(T_n) = rank(T_{n-1}) + 1 = n - 1 + 1 = n$
Similarly, using the fact that $rank(T_{n-1} \oplus I_{n-1}) = n - 1$ (where I_{n-1} is the identity matrix of order $n-1$), we have $rank(T_n \oplus I_n) = n$. The characteristic polynomial of the matrix T_n, denoted by $\phi_n(x)$ is evaluated as $det(T_n \oplus xI_n)$, where det denotes the determinant. Thus we have,

$$\begin{aligned}
\phi_n(x) &= \det \left(\begin{array}{cccc|c} & & & & 0 \\ & & & & 0 \\ & & & & \vdots \\ & T_{n-1} \oplus xI_{n-1} & & & 0 \\ & & & & \vdots \\ & & & & 0 \\ & & & & 0 \\ \hline - & - & - & - & - \\ 0 & \ldots & 0 & 1 & x \end{array} \right) \\
&= x\phi_{n-1}(x), (\phi_{n-1}(x) \text{ denotes the characteristic polynomial of } T_{n-1}) \\
&= x.x^{n-1} \\
&= x^n
\end{aligned}$$

In order to evaluate the minimal polynomial we make use of the following proposition.

Lemma 8.3 *Let $\phi_n(x)$ and $\psi_n(x)$ be the characteristic polynomial and the minimal polynomial of the matrix T_n, respectively. Let the greatest common divisor (gcd) of the matrix $(T_n \oplus I_n x)^\vee$ that is the matrix of algebraic complements of the elements of the matrix $(T_n \oplus I_n x)$ be $d(x)$. Then, $\phi_n(x) = d(x)\psi_n(x)$.*

From the matrix $(T_n \oplus I_n x)^\vee$ it may be observed that the element at the position $(0, n)$ is 1 and thus the gcd $d(x)$ is also 1. Thus the minimal polynomial is equal to the characteristic polynomial which is x^n. Thus, we observe that the construction follows all the four necessary and sufficient requirements of a TPSA CA. This completes the proof. □

Example 8.6 *Given the fact that*

$$T_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

is the characteristic matrix of a 2 cell TPSA CA. Thus, using the above theorem it is evident that:

$$T_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

is the characteristic matrix of a 3 cell TPSA CA.

An alternative construction when number of cells is even: It may be observed that when number of cells is even the characteristic matrix can be constructed as follows:

$$T_{2n} = \left(\begin{array}{cccc|cccc} & & & & & & & \\ & & T_n & & & & 0 & \\ & & & & & & & \\ \hline - & - & - & - & - & - & - & - \\ 0 & 0 & \dots & 1 & & & & \\ 0 & 0 & \dots & 0 & & & & \\ \vdots & & & \vdots & & & T_n & \\ 0 & 0 & \dots & 0 & & & & \\ 0 & 0 & \dots & 0 & & & & \end{array}\right)$$

In the above construction T_n denotes the characteristic matrix of an n-cell TPSA-CA. Hence, the rank of T_n is $(n-1)$ and hence there are two rows in T_n which are linearly dependent. It is assumed in the above methodology to construct the characteristic matrix of a TPSA-CA of $2n$ cells, that the first two rows are linearly dependent. This is without loss of generality as otherwise the non-zero row in the lower-left submatrix would shift to one of the dependent row positions.

The proof of the above construction is as follows: Rank of the matrix T_{2n} is $\text{rank}(T_n) + n = 2n - 1$. This is because the non-zero row of lower-left submatrix shall make all the lower n rows of the matrix T_{2n} independent. Rank of $(T_{2n} \oplus I_{2n})$ is $\text{rank}(T_n \oplus I_n) + n = 2n$. Similarly, characteristic polynomial of T_{2n} is $det(T_{2n} \oplus xI_{2n}) = (T_n \oplus xI_n)^2 = (x^n)^2 = x^{2n}$. That the minimal polynomial and characteristic polynomials are the same is for the same reason as shown in **theorem 8.1**. Thus the characteristic matrix T_{2n} satisfies all the necessary and sufficient conditions for a TPSA-CA of size $2n$.

The above method gives a much faster construction methodology.

As an example using the characteristic matrix of a 2 cell TPSA CA,

$$T_2 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

the characteristic matrix of a 4 cell TPSA CA is constructed as:

$$T_4 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

It may be noted that this is precisely the characteristic matrix of the 4 cell TPSA CA of **example 8.5**.

We have seen above that the state transition in the above class of TPSA CA is governed solely by the characteristic matrix. This class of CA is known as the non-complemented TPSA CA. On the contrary when the next state is obtained by the application of the characteristic matrix and then xoring with a vector F, the CA is known as the complemented TPSA-CA.

The following results show how complementing the state transition function of the non-complemented CA generates a class of automaton with the same properties as

the original TPSA CA.

Lemma 8.4 *Corresponding to a non-complemented TPSA CA M_1 and a state Z, there exists a complemented CA M_2 with state Z as an attractor. If the characteristic matrix M_1 be indicated by T_n and it is required to build a complemented TPSA CA such that Z is the graveyard (attractor) then the characteristic matrix of the complemented CA, $\overline{T}_n$ is related to T_n by*

$$\overline{T}_n(X) = T_n(X) \oplus (I_n \oplus T_n)Z$$

where X is the seed to the CA and I_n is the identity matrix of order n.

Lemma 8.5 *A complemented TPSA CA has the same structure as a non-complemented TPSA CA. To emphasize*

- *Number of attractors in the complemented CA is the same as that in the original non-complemented CA.*
- *Number of reachable states and non-reachable states are same as that in the original non-complemented CA.*

Lemma 8.6 *If any state Z in the non-reachable world of a non-complemented CA is made the graveyard in a complemented TPSA, then the non-reachable elements become elements of the reachable world in the complemented CA and viceversa. Thus the reachable world (W_1) and the non-reachable world (W_2) exchange themselves (***Fig. 8.3***).*

Proof: Let X and Z be two non-reachable elements in the n cell non-complemented CA with characteristic matrix T_n. Let X be the l^{th} level sister of Z. In all cases $l < n$. Thus, we have:

$$T_n^l(X) \;=\; T_n^l(Z)$$

Let us consider the state transition diagram of the complemented CA with Z as the graveyard. The state transition of the complemented CA is indicated by $\overline{T}_n$. We shall prove that in this state transition graph X is a reachable state. Let, the depth

of X in the graph of the complemented CA be t. If t is less than n then X is a reachable state. Since, Z is the graveyard of this graph we have:

$$\begin{aligned} \overline{T}_n^t(X) &= Z \\ T_n^t(X) \oplus (I_n \oplus T_n^t)Z &= Z \\ T_n^t(X) &= T_n^t(Z) \end{aligned}$$

Thus, X and Z are t^{th} level sisters in the state transition graph of the non-complemented CA. But we know that they are l^{th} level sisters. Thus $t = l < n$. Thus, the depth of X is lesser than n and hence X is a reachable state in the state transition graph of the complemented CA. □

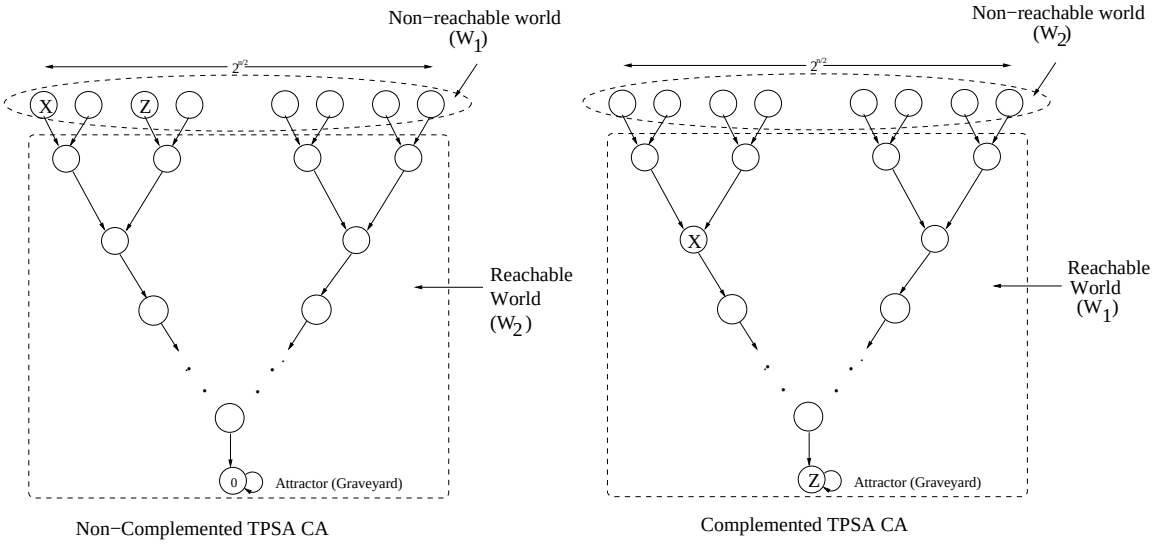


Figure 8.3: **The exchange of the worlds in a complemented TPSA CA**

8.3.1 Construction of expander graph using the TPSA CA

The TPSA CA can be effectively used to generate a random d regular graph on the fly. It may be noted that the entire nature of the graph is stored in the graveyard state, thus leading to a very compact storage of the graph. This is because given the graveyard state, the entire transition graph can be obtained.

In order to construct the d regular graph we proceed as follows: Let $Z_1 \in W_1$ (non-reachable world in the non-complemented TPSA CA) and $Z_2 \in W_2$ (reachable world in the non-complemented TPSA CA). Let, G_1 and G_2 be the state transition graphs with Z_1 and Z_2 as the graveyards respectively.

Table 8.1: **Spectrum of a 4 cell TPSA based regular graph**

No. of Union (t)	Graveyards	Degree	First Eigen Value	Second Eigen Value	Spectral Gap (g)	g/t
1	{0},{4}	4	4	3.2361	0.76	0.76
3	{0,15},{4,8}	8	8	4.899	3.10	1.03
5	{0,15,3},{4,8,10}	12	12	6.3440	5.66	1.14
7	{0,15,3,2},{4,8,10,9}	16	16	5.2263	10.77	1.54

Clearly, in G_1 if $X \in W_1$, $degree(X) = 3$ and if $X \in W_2$, $degree(X) = 1$. Similarly, in G_2 if $X \in W_1$, $degree(X) = 1$ and if $X \in W_2$, $degree(X) = 3$. Here *degree* is defined as the sum of the *indegree* and the *outdegree* in the corresponding graph.

Thus, in the graph G obtained by a union operation in the graphs G_1 and G_2, allowing multiple edges and self loops, we have for $X \in G$, $degree(X) = 4$. If we continue the union operation in the above method we have $degree(X) = 2(t + 1)$, where t is the number of union operations. **Table 8.1** shows the result of an experimentation performed with the TPSA based regular graph. It measures the value of the two largest eigen values for random TPSA based graphs for degree $4, 8, 12$ and 16. The difference between the largest two eigen values is known as the spectral gap and should be large for good expansion of the graph. Results show that the spectral gap and hence the expansion increases proportionately with the number of union operations (t).

8.3.2 Setting parameters of the d regular TPSA based graph for good Expansion Properties

In the present section we compute the expansion obtained in the d regular TPSA based graph in terms of the parameters of the graph G. Let the number of nodes in the graph be n and the degree of each node is d. Let us consider a random d regular graph (**Fig. 8.4**) the subset A with αn vertices $(0 < \alpha < 1)$.

For the graph G to have good expansion properties the set A should have more than βn $(0 < \beta < 1)$ neighbours outside A. The probability of such an event should be high.

Equivalently, we may state that the probability that the number of neighbours of the vertices of A outside A is less than βn is negligible. Let us fix A and B such that

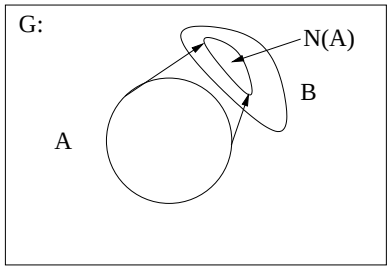


Figure 8.4: **Computing the expansion of the random d regular graph**

$|A| = \alpha n$ and $|B|{=}\beta n$.

It is required that the vertices of A be matched to $N(A)$ (neighbours of A outside A) s.t $N(A) \subset B$. If we first consider a single matching, αn vertices can have maximum αn neighbours in $N(A)$. The probability that the neighbours of A are in B is :

$$\begin{aligned} p &= \frac{\text{no. of ways in which N(A) can lie inside B}}{\text{no. of ways in which N(A) may be chosen outside A}} \\ &= \frac{\binom{\beta n}{\alpha n}}{\binom{n-\alpha n}{\alpha n}} \end{aligned}$$

Hence, if we consider a d-matching, assuming all the edges to be independent we have

$$\begin{aligned} \Pr[N(A) \subset B] &= \left(\frac{\binom{\beta n}{\alpha n}}{\binom{n-\alpha n}{\alpha n}} \right)^d \\ &\approx (\tfrac{\beta}{1-\alpha})^{\alpha n d} \end{aligned}$$

Thus, we have the following probability:

$$\begin{aligned} \Pr[\exists A \in G \text{ s.t } |A| = \alpha n, |N(A)| \leq \beta n] &\leq \sum_{|A|=\alpha n} \sum_{|B|=\beta n} (\tfrac{\beta}{1-\alpha})^{\alpha n d} \\ &\leq \binom{n}{\alpha n} \binom{n}{\beta n} (\tfrac{\beta}{1-\alpha})^{\alpha n d} \end{aligned}$$

Next, we use the following approximations: $\binom{n}{\alpha n} = 2^{nH(\alpha)}$, $\binom{n}{\beta n} = 2^{nH(\beta)}$.

Here, $H(\alpha) = -\alpha \log_2 \alpha - (1-\alpha)\log_2(1-\alpha)$. Thus, setting $\alpha = \frac{1}{m}$ and $\beta = \frac{1}{2}$ and some simplifications we have:

$$\Pr[\exists A \in G \text{ s.t } |A| = \alpha n, |N(A)| \leq \beta n] \quad \leq \quad 2^{n[\log_2 m - (1-\frac{1}{m})\log_2(m-1) + 1 + \frac{d}{m}\log\frac{m}{2(m-1)}]} \approx 2^{-cn}$$

Hence in order to make the probability negligible we may choose parameters m and d such that c becomes positive. The value of d gives us an estimate of the number of union operations t to be performed.

8.3.3 Mathematical Formulation of adjacency in the TPSA based Graph

The graph $G(V, E)$ can thus be described as a union operation between the graphs G_1 and G_2 where $Z_1 \in W_1$ and $Z_2 \in W_2$ are the respective graveyard states.

The following algorithm computes the four neighbours of a given state in the graph G.

Algorithm 8.1 Computing neighbourhood of a vertex X in G

Input: *Z_1, Z_2, a state $X \in G$. Any state X is an n-tuple, $(x_1, x_2, \ldots, x_n)$*

Output: *The four neighbours of X (U, V, W, Y)*

Step 1: Computation of neighbours U and V,

$$\begin{aligned} U &= T_n(X) \oplus (I_n \oplus T_n)Z_1 \\ V &= T_n(X) \oplus (I_n \oplus T_n)Z_2 \end{aligned}$$

Computation of neighbours W and Y

Step 2:

Compute, $X \oplus (I_n \oplus T_n)Z_1 = (\, x_1, x_2, \ldots, x_n \,)^T$

If$(x_1 = x_2)\{/ * X \in G_1 * /$*from the matrix defined in* **theorem 8.1**

set $w_n = 0$, Compute $w_{n-1} = x_n, w_{n-2} = x_{n-1}, \ldots, w_2 = x_3, w_1 = w_2 \oplus x_1$

set $y_n = 1$, Compute $y_{n-1} = x_n, y_{n-2} = x_{n-1}, \ldots, y_2 = x_3, y_1 = y_2 \oplus x_1$ $\}$

Step 3:

else$\{/ * X \in G_2 * /$

repeat Step 2 with Z_2 replacing Z_1

$\}$

It is evident that the distribution of the neighbourhood is identical to that of Z_1 or Z_2 which are randomly chosen. Thus, we have a four regular pseudo-random graph when we have only one union operation. The degree may be increased with the number of union operations. The advantage lies in the fact that in general the description of a random graph grows exponentially with the number of vertices. However, using the TPSA based construction one may generate graphs which exhibit randomness and also may be described using polynomial space, as we require to store only the graveyard state. For an N cell TPSA, in order to compute the neighbourhood of any state, Algorithm 1 is applied in constant time, as each of the steps $1, 2$ and 3 may be applied in constant time parallely on the N bit input vector. Herein lies the efficacy of the TPSA based expander graphs.

8.4 Application of TPSA based Expander Graphs in Constructing One-way functions

The one-way function using the d-regular graph generated by the TPSA is based on the construction proposed in [115]. The one-way function maps a string in $\{0,1\}^n$ to another in $\{0,1\}^n$. The algorithm is based on a d regular graph generated by the TPSA. As already mentioned the graveyard states are from W_1 and W_2. Let Z_1 denotes the graveyard states from W_1 and Z_2 denotes the graveyard states from W_2. If there are i elements in Z_1 and Z_2 then the number of union operations required to generate the d regular graph is $2i-1$ and so we have $d = 2(2i-1+1)$. Thus we arrive at the number of graveyard states required as $i = \frac{d}{4}$.

The evaluation of the one-way function is as follows:

Algorithm 8.2 One-way function (f) using the state transitions of a TPSA

Input: *$Z_1 \in W_1$, $Z_2 \in W_2$, a state $X \in \{0,1\}^n$. Thus the TPSA based regular graph has degree 4.*

Output: *The one-way output $Y \in \{0,1\}^n$ such that $Y = f(X)$*

Step 1 *Consider an N cell TPSA, where $N = \log_2(n)$. Generate a collection C, which constitutes of the neighbours of each node in the regular graph generated by the state transitions of the TPSA using* **Algorithm 8.1**. *Mathematically, $C = \{S_i,$ if $v \in S_i, E(v,i) = 1, i \in \{1,\ldots,n\}\}$.*

Step 2 *For $i = 1$ to n, project X onto each of the subsets S_i. If $S_i = \{i_1, i_2, \ldots, i_d\}$,*

then the projection of X on S_i denoted by X_{S_i}, is a string of length d indicated by $\{X_{i_1}, X_{i_2}, \ldots, X_{i_d}\}$.

Step 3 *Evaluates a non-linear boolean function Π on each of the n projections thus giving an n bit output, $\{\Pi(X_{S_1}), \Pi(X_{S_2}), \ldots, \Pi(X_{S_n})\}$. The non-linear function Π is $\Pi(z_1, z_2, \ldots, z_d) = (\sum_{i=1}^{d/2} z_i z_{i+d/2}) \ mod\ 2$.*

The forward transformation is very efficient as the total time required is $O(n)$. This may be observed as the time required to perform **Step 1** is due to that required to apply Algorithm 1 at all the 2^N vertices. Hence the total time is also proportional to $2^N = n$. The time required to apply **Step 2** and **Step 3** is also proportional with n and hence the total time required. However computing the inverse seems to be intractable even when the collection C is known. As proved in [115] the complexity of a proposed inverting algorithm is atleast exponential in $min_\pi\{max_i\{|\cup_{j=1}^{i} S_{\pi(j)}| - i\}\}$.

We have shown in **section 8.3.2** that the probability that the size of the neighbourhood of any subset S is proportional to n is very high. Thus for all cases the value of $min_\pi\{max_i\{|\cup_{j=1}^{i} S_{\pi(j)}| - i\}\}$ is $O(n)$ and hence the complexity of a possible inverting algorithm proposed in is atleast exponential in $O(n)$[115]. Thus the problem of inverting the one-way function seems to be intractable.

In the following section we present the implementation of the proposed architecture on a Xilinx XCV-1000 platform.

8.5 Implementation of the TPSA based One-way Function

In this section, we propose the architecture for n = 128 bit one-way function. The size of the TPSA CA is thus $N = log_2 n = log_2 128 = 7$. Using the 7-bit TPSA CA, we construct a 16-regular random expander graph. It has been computed that with these parameters, the expander graph has good expansion properties which in turn make the one-way function strong [115].

Fig. 8.5 shows the top-level view of the architecture. The $N = 7$ bit up-counter counts from 0 to 127 ($n = 128$) and for each of these values the block Gen-Neighbours calculates the $d = 16$ neighbours in the TPSA based d-regular graph. Each neighbour selects a particular bit from the input X. The 16 bit output of the block Gen-Neighbours is thus the 16 bits selected from the input X. These 16 bits are taken by the block Projection which calculates the Boolean function Π on those bits. Finally the output bit of the Projection block (Q) is clocked into the proper flip-flop of the

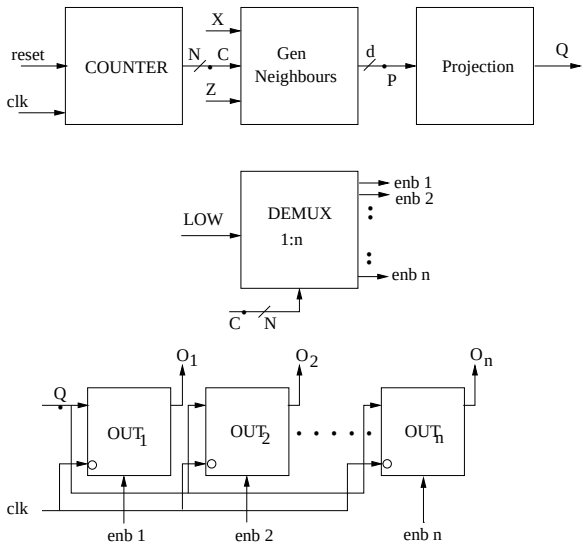


Figure 8.5: **Top-level view of the architecture**

Table 8.2: **Performance of the Design on Xilinx XCV-1000 platform**

No of Slice Flip Flops	223, 1% of total
No of 4 Input LUTs	1343, 5% of total
No of Slices	6% of total
Maximum delay	0.842 ns
Power Consumption	32mW at $V_{cc} = 3.3V$
Throughput	3.5 Gbps @28.6 MHz

output register (OUT_1 to OUT_n). The output of the counter (c) determines which of the flip-flops (OUT_1 to OUT_n) in the output register is to be enabled (enb_1 to enb_n) and consequently the output of the Projection block (Q) gets clocked into that enabled flip-flop. The Z input to Gen-Neighbours is the set of the N-bit graveyards of all those TPSA CA whose union is to be carried out.

The proposed architecture has been implemented in Verilog HDL and has been verified using RTL simulations using Mentor Graphics ModelSim SE. The Verilog RTL has been prototyped on a Xilinx Virtex XCV1000 FPGA (pkg bg560). The characteristics of the design have been summarized in **table 8.2**.

Analysis show that the resource utilization increases linearly with the input size. **Table 8.3** gives a comparison of the resources used, the delay and the area-delay product of the proposed one-way function with those of an efficient implementation of the exponentiation operation in $GF(2^n)$ [198]. In the table, n denotes the number

Table 8.3: **Comparison of the proposed One-way function with respect to Implementation**

	Proposed One-Way Function	**Efficient Implementation of exponentiation in $GF(2^n)$[198]**
Order of resource usage	$8n$ LUT+ n FF	$15.7n$ LUT + $3n$ FF
Order of delay	n	n^2
Order of area-delay product	n^2	n^3

of bits in the input and output.

8.5.1 Message Authentication Code using Cellular Automata based one-way function

A message authentication code (MAC) is generated by a function of the form $MAC = C_K(M)$ The technique assumes that the two communication partners say A and B, share a common secret key K. When A has a message M to send to B, it calculates the MAC as a function of the message and the key using an algorithm denoted by C_K.

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate the new MAC. If the MAC received matches with the computed result the receiver is assured of the following:

- The receiver is assured that the message is not altered.
- The receiver is assured that the message is from the alleged sender, under the assumption that no one else has the shared key K.
- If the message has an nonce or a sequence number, then the receiver can be assured of the proper sequencing of the messages.

In order to realize the function C_K we reuse the CA based one-way function f. In order to make it work as a keyed function we use the graveyard states Z_1 and Z_2 as a secret. Since, the state transitions in the graph gets randomly configured depending on Z_1 and Z_2, in order to compute the MAC, one needs to know the graveyard states.

8.6 The Key Agreement Protocol

The key agreement protocol has been described in this section. The protocol uses the TPSA CA based one-way function which serves the protocol in two ways. First it helps to derive a common key from the contributions of both the parties, also called the half keys. The advantage of the one-wayness is that it provides key-freshness to the keying material, in the fact that no party can force an old key to be generated after knowing the half key of the other party. The other purpose is when the function is used to develop a message authentication code (MAC). While using the one-way function for developing the MAC, the function needs to be key-dependent. For this purpose we use the graveyard states which are kept secret, being known to only the parties exchanging the keys.

The outline of the protocol is presented next.

Parameter: n : The size of the TPSA CA and the derived session key

Both A and B share the secret graveyard states $Z_1 \in W_1$, $Z_2 \in W_2$ and hence can compute the one-way function f, which depends on the graveyard states. The graveyards define the adjacency of the expander graph, which in turn defines the one-way function.

A generates nonce $N_A \in_R [0, 2^n - 1]$, half key $K_A \in_R [0, 2^n - 1]$

Message 1: $A \rightarrow B : A, K_A, N_A$

B generates $N_B \in_R [0, 2^n - 1]$, half key $K_B \in_R [0, 2^n - 1]$ and computes $K_{AB} = f(K_A \oplus K_B)$ and the MAC $f(A, B, N_A, K_A, K_B)$.

Message 2: $B \rightarrow A : f(A, B, N_A, K_A, K_B), (K_B, N_B, B)$

Now, A also computes $K_{AB} = f(K_A \oplus K_B)$ and the MAC $f(A, B, N_B, K_A, K_B)$.

Message 3: $A \rightarrow B$, $f(A, B, N_B, K_A, K_B)$

Final agreed key is K_{AB}.

The protocol is explained in the following description. Both A and B share the graveyard states Z_1 and Z_2, which serve as the key to the one-way function. Since, only A and B share this knowledge they only can compute the MAC using the one-way function f.

- Principal A sends his identifier A, the nonce N_A and his share to the key (half key) K_A.
- Principal B then sends his contribution K_B and nonce N_B to A along with his identifier. Meanwhile B computes the value of $K_{AB} = f(K_A \oplus K_B)$ and the MAC $f(A, B, N_A, K_A, K_B)$ and sends it to A.
- A on receiving the message from B checks whether the message is transferred

properly by verifying the MAC. Since only A and B agrees the common graveyard states, A is satisfied about the authenticity of B. It also satisfies A that B has actually received $A's$ half key K_A properly and thus have generated the same session key. This serves key confirmation. Then A accepts and computes $K_{AB} = f(K_A \oplus K_B)$. He also sends the MAC $f(A, B, N_B, K_A, K_B)$ to B.

Thus both A and B computed the session key K_{AB} and thus both the principals agree upon a session key.

8.7 Security Analysis of the Protocol

Informally, a secure key agreement protocol should not allow an active resource bounded adversary to manipulate the message flows in any polynomial number of protocol executions between honest parties, in such a way that information is leaked on the session key (or any specific goal of the protocol is compromised). We present a proof based on the formal model shown in [112] and the goals of a successful key agreement with authentication and key confirmation defined in [108].

In this model a protocol is a pair $P = (\Pi, \mathcal{G})$ of probabilistic polytime computable functions, where Π specifies protocol actions, message formats while $\mathcal{G}$ specifies how long term keys are generated (the secret graveyard states in our case). The number of principals involved are indicated by $I = \{1, \ldots, N_1\}$. Here N_1 is a polynomial in the security parameter n so that $N_1 = T_1(n)$ for some polynomial function T_1.

The adversary E is defined to be a probabilistic machine that is in control over all communications between the parties by interacting with a set of oracles $\Pi^i_{A,B}$ (i.e i^{th} session of a principal A in a specific protocol run with B being the other principal involved in the key agreement). The oracle queries are summarised underneath:

- **Send**(A, B, i, M): This query allows the adversary to send a message M from B to A in session i. If the client oracle $\Pi^i_{A,B}$ has either accepted with some session key or terminated, this will be made known to the adversary.
- **Reveal**(A, B, i): The client oracle upon receiving this query and if it has accepted and holds some session key, it will send back the session key to E.
- **Corrupt**(A, K): This query allows E to corrupt the principal A at will, and thereby learn the complete internal state of the corrupted principal. The query also gives E to over-write the long-term key of the corrupted principal with any specific value, K.
- **Test**(A, B, i): This query does not correspond to any of $A's$ ability. If $\Pi^i_{A,B}$ has accepted with some session key and is being asked this query, then depending

on a randomly chosen bit b, E is given either the actual session key or a session key drawn randomly from the session key distribution. The adversary's job is now to guess b. Let the output of the adversary be a bit **Guess**. Let, $\mathbf{GoodGuess^E(k)}$ be the event that **Guess**=b. Then we define advantage$^E(k) = |\Pr[\mathbf{GoodGuess^E(k)}] - \frac{1}{2}|$.

Security depends on the notions of partnerships of oracles and indistinguishability of of session keys. The definition of partnership is used in the definition of security to restrict the adversary's Reveal and Corrupt queries to oracles that are not partners of the oracles whose key the adversary is trying to guess. The notion of security is through the following definition of *matching conversations.* Conversation is defined to be a sequence of messages sent and received by an oracle which is recorded in a transcript T. At the end of a protocol run, T contains the record of Send queries and its responses.

Definition 8.3 MatchingConversations[*112*]*: Let us consider an initiator oracle* $\Pi^i_{A,B}$ *and a responder oracle* $\Pi^j_{B,A}$ *who engage in conversations* C_A *and* C_B *respectively.* $\Pi^i_{A,B}$ *and* $\Pi^j_{B,A}$ *are said to be have matching conversations if: there exists* $\tau_0 < \tau_1 < \tau_2 < \tau_3 < \ldots$ *and* $\alpha_1, \beta_1, \alpha_2, \beta_2, \ldots$ *such that* $C_A = (\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), \ldots$ *and* $C_B = (\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), \ldots$. *Here,* λ *is the empty string used to initiate the protocol. The term* $(\tau_0, \lambda, \alpha_1)$ *in* C_A *means that at time* τ_0*, the oracle* $\Pi^i_{A,B}$ *receives the message* λ *and responds with the message* α_1*. Similarly, the term* $(\tau_1, \alpha_1, \beta_1)$ *in* C_B*, means that at time* τ_1 *the oracle* $\Pi^j_{B,A}$ *receives the message* α_1 *and responds with the message* β_1*. Likewise, the message exchange between the initiator and the responder oracle continues.*

Finally we state the goals of a *secure AKC protocol.*

Definition 8.4 *[108] A protocol* $P = (\Pi, \mathcal{G})$ *is a* secure authenticated key agreement with confirmation (AKC) protocol *if:*

1. *In the presence of a benign adversary on* $\Pi^i_{A,B}$ *and* $\Pi^j_{B,A}$*, both oracles always accept holding the same session key* K^{ij}_{AB}*, and this key is uniformly distributed at random in* $\{0,1\}^n$

   and if for every adversary E:

2. *If uncorrupted oracles* $\Pi^s_{A,B}$ *and* $\Pi^t_{B,A}$ *have matching conversations then both oracles accept and hold the same session key* K^{st}_{AB}.

3. *The probability of the event* $\mathbf{NoMatching^E(n)}$ *(denotes the event that there exists an oracle* $\Pi^s_{A,B}$ *which accepted but there is no oracle* $\Pi^t_{B,A}$ *with which it has had a matching conversation) is negligible.*

4. $advantage^{\mathbf{E}}(\mathbf{n})$ *is negligible.*

In the following proof we show that the proposed protocol satisfies the above goals in this model.

8.7.1 Formal Proof of Security of the Protocol

In the present analysis we show that the CA based Key agreement is a secured AKC protocol with the help of the following theorem.

Theorem 8.2 *The proposed CA based Key Agreement Protocol is a secured AKC if the MAC generated by the one-way function is secured.*

Proof: It is trivial to show that the conditions 1 and 2 of **definition 8.4** are satisfied.

We first prove that $\Pr(\mathbf{NoMatching^E(n)})$ is negligible.

The adversary E succeeds if at the end of $E's$ experiment there exists an oracle $\Pi^s_{A,B}$ (where A and B are not corrupted) which has accepted but there is no $\Pi_{B,A}$ which has had a matching conversation. Let, Pr(E succeeds) $= p(n)$ and by contradiction let $p(n)$ be non-negligible. We may divide the case into two smaller subparts.

Case 1: Suppose that $p_1(n)$ is the probability that E succeeds against $\Pi^s_{A,B}$ and $p_1(n)$ is say non-negligible. We shall construct an adversary F from E such that it violates the semantic security of the MAC generated by the CA based one-way function f.

Define F: F picks $(A, B) \in_R I$ and $s \in_R \{1, \ldots, T_2(n)\}$, for some polynomial function T_2, guessing that E will succeed against $\Pi^s_{A,B}$ oracle.

F answers all $E's$ queries itself. When the long-term keygenerator $\mathcal{G}$ is queried by E or any oracle that is not $\Pi_{A,B}$ or $\Pi_{B,A}$, F gives up. $F's$ actions when $\mathcal{G}$ is queried by on the shared graveyards Z_1, Z_2 by $\Pi_{A,B}$ or $\Pi_{B,A}$ are as follows:

1. F answers $E's$ reveal queries as usual.

2. If E asks corrupt A or B, F gives up.

3. F also answers send queries of $\Pi_{A,B}$ or $\Pi_{B,A}$ as per Π, except that instead of calculating the long term shared key $K = \mathcal{G}(Z_1, Z_2)$ and using this key to generate the MAC by computing $f(.)$, F calls its own MACing oracle to compute the response. Thus F generates the key randomly and computes the one-way function (using a random set of graveyards) on behalf of $\Pi_{A,B}$ and $\Pi_{B,A}$ oracles and also decides whether such oracles should accept.

4. If E does not invoke $\Pi^s_{A,B}$ oracle as an initiator F gives up.

5. On the other hand if E does invoke $\Pi^s_{A,B}$ as an initiator oracle, then at some time τ_0, $\Pi^s_{A,B}$ receives λ and responds with A, K_A, N_A as its output message. If $\Pi^s_{A,B}$ does not at some later time receives the message $f(A, B, N_A, K_A, K_B), K_B, N_B, B$ for some N_A, F gives up.

6. However if $\Pi^s_{A,B}$ is to accept, it must later receive a flow of this form. In this event provided F has not called its random oracle to generate the MAC for (A, B, N_A, K_A, K_B) then F stops and outputs a guess. If F has previously called its oracle to compute the MAC value, then F gives up. This is because of the fundamental definition of random oracles which states that such a function outputs a random output $H(x)$ for an input x, if x has not been queried before. However if x has been queried before it repeats the same previous output. Thus if the function f has been computed before using the random oracle used to generate the MAC and E has not been successful, then it will not be successful even now.

Suppose, E does succeed against $\Pi^s_{A,B}$. In this event F outputs a valid forgery for the MAC f and wins the experiment, provided E or some other oracle apart from $\Pi^s_{A,B}$ or $\Pi^t_{B,A}$ has not:

a Computed $\mathcal{G}$ on Z_1 and Z_2 and

b F has not previously calculated the flow that makes $\Pi^s_{A,B}$ accept on behalf of some $\Pi_{A,B}$ or some $\Pi_{B,A}$ oracle.

Since, it is assumed that the long term key has been generated by a secured exchange $\mathcal{G}$ by inputting the graveyards Z_1 and Z_2 uniformly selected from the worlds W_1 and W_2 the probability of (a) is negligible.

The probability of (b) is also negligible, because F could only have called in this message on behalf of a responder $\Pi^t_{B,A}$ which received (A, K_A, N_A) as its first flow or on behalf of an initiator $\Pi^u_{A,B}$ $(u \neq s)$ which also chose K_A, N_A and needs to decide whether or not to accept. The probability that the call was made by the responder $\Pi^t_{B,A}$ before τ_0 is negligible since K_A, N_A was chosen at random and if the call was

made after τ_0, then $\Pi^t_{B,A}$ has had a matching conversation to $\Pi^s_{A,B}$ (and thus E fails). The probability that the call was made by $\Pi^u_{A,B}$ is negligible since in this event $\Pi^u_{A,B}$ and $\Pi^s_{A,B}$ have independently chosen the same K_A, N_A.

Thus, the success probability of F is:

$$\begin{aligned}\Pr[\text{F succeeds in forging the MAC}] &= \Pr[\text{E succeeds}]\Pr[\text{of choosing A, B, s}] - \lambda(n) \\ &\quad \text{where } \lambda(n) \text{ is negligible} \\ &= \frac{p_1(n)}{T_1(n)^2 T_2(n)} - \lambda(n)\end{aligned}$$

which is also non-negligible. Thus we arrive at a contradiction.

Case 2: Similarly, we can show that the probability that E succeeds against $\Pi^t_{B,A}$ is negligible. Combining these two cases we obtain that $\Pr(\mathbf{NoMatching}^{\mathbf{E}}(\mathbf{n}))$ is negligible.

Next we show that $\text{advantage}^{\mathbf{E}}(\mathbf{n})$ is negligible. As a contradiction let us assume that $\Pr(\text{E succeeds})$ is $\frac{1}{2} + q(n)$, where $q(n)$ is not negligible.

Let M denote the event that E picks up some $\Pi^s_{A,B}$ oracle to ask its Test query s.t some $\Pi^t_{B,A}$ has had a matching conversation to $\Pi^s_{A,B}$.

$$\Pr(E \text{ succeeds}) = \Pr(E \text{ succeeds} \mid M)\Pr(M) + \Pr(E \text{ succeeds} \mid \overline{M})\Pr(\overline{M})$$

Since, $\Pr(\overline{\text{M}}) = \lambda(n)$ is negligible by the previous proof. Thus,

$$\begin{aligned}\frac{1}{2} + q(n) &\leq \Pr(\text{E succeeds} \mid \text{M})\Pr(\text{M}) + \lambda(n) \\ &\leq \Pr(\text{E succeeds} \mid \text{M})(1 - \lambda(n)) + \lambda(n)\end{aligned}$$

Thus, $\Pr(\text{E succeeds} \mid \text{M}) = \frac{1}{2} + q_1(n)$, where $q_1(n)$ is non-negligible. Given the event M, session key will be of the form $f(K_A \oplus K_B)$. Let N denote the event the adversary or some other oracle has computed $f(K_A \oplus K_B)$. Thus,

$$\begin{aligned}\Pr(E \text{ succeeds} \mid M) &= \Pr(E \text{ succeeds} \mid \text{M} \wedge \text{N})\Pr(\text{N}|\text{M}) + \Pr(E \text{ succeeds} \mid M \wedge \overline{N})\Pr(\overline{N}|M) \\ q_1(n) + \frac{1}{2} &\leq \Pr(\text{N}|\text{M}) + \frac{1}{2}\end{aligned}$$

This is because $\Pr(E \text{ succeeds} \mid M \wedge \overline{N}) = \frac{1}{2}$, as $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ are not open by definition

$$\text{Thus, } \Pr(N|M) \geq q_1(n)$$

Thus, the probability that the adversary or some other oracle apart from A or B has computed $f(K_A \oplus K_B)$ is non-negligible. This is contradictory to the assumption that without the knowledge of the graveyards Z_1 or Z_2 it is infeasible to compute the function f. □

Apart from authentication and key confirmation, the CA based AKC protocol also provides **key freshness** due to the following reason:

If one parties say B, comes to know the other parties half key K_A, and wants to contribute K_B such that an old key is forced he needs to solve $x \in [0, 2^n - 1]$ the equation $K_{old} = f(K_A \oplus x)$ which is not possible due to the one-wayness of f inspite of knowing the graveyards Z_1, Z_2.

8.8 Conclusion

The chapter proposes a method to generate expander graphs with good expansion properties based on the state transitions of a special class of Cellular Automata, known as the Two Predecessor Single Attractor CA (TPSA-CA). The expander graph has been finally used to compose a one-way function whose security lies on the combinatorial properties of the expander graphs. The design has been implemented on a Xilinx XCV-1000 platform and the performance of the design has been compared with that of efficient designs of the popular modular exponentiation based one-way function. Results have been furnished to prove that the design is superior in terms of area delay product and power consumption. The one-way function is then used to design a key agreement protocol which provides authentication, key confirmation and key freshness. Finally, the security of the protocol has been proved in the Bellare Rogaway model.

In the following two chapters, we perform cryptanalysis of two standard ciphers: AES Rijndael, which is the world-wide standard for block ciphers and CMEA, which is the standard block cipher for the digital Cellular phones.

Chapter 9

Fault Based Attack of the Rijndael Cryptosystem

9.1 Introduction

In order to satisfy the security requirements of various information disciplines e,g networking, telecommunications, data base systems and mobile applications, applied cryptography has gained immense importance now-a-days. To satisfy the high throughput requirements of such applications, the complex cryptographic systems are implemented by means of either VLSI devices (crypto-accelerators) or highly optimized software routines (crypto-libraries). The high complexity of such implementations raises concerns regarding their reliability. Hence in this scenario it is imperative that the crypto-algorithms should not only prevent conventional cryptanalysis but also should prevent the deduction of the keys from accidental faults or intentional intrusions. Such attacks are known as fault attacks and was first conceived in September 1996 by Doneh, Demillo and Lipton [58, 199] from Bellcore. The fault attack was applicable to public key cryptosystems and was extended to various secret key ciphers like DES, the technique being known as Differential Fault Analysis (DFA)[59]. On 2^{nd} October 2000, the US National Institute of Standards and Technology (NIST) selected Rijndael [184] as the Advanced Encryption Standard (AES) and thus replaced DES as a world-wide standard for symmetric key encryption. Thus smart cards and secure micro-controllers are designed using AES to protect both the confidentiality and the integrity of sensitive information. With the work on optical fault induction reported in [61], research in the field of fault-based side channel cryptanalysis of AES has gained considerable attention. DFA on AES was reported in [63] by inducing faults at byte level to the input of 9^{th} round of Rijndael using 250 faulty

ciphertexts. In the fault based attack on AES reported in [64] around 128 to 256 faulty ciphertexts are required to discover the key. P. Dusart etal [65] performs a Differential Fault Analysis on AES and shows that using a byte level fault induction anywhere between the eighth round and ninth round the attacker is able to break the key with 40 faulty ciphertexts. Finally, [66] shows that when a byte level fault occurs at the input of the eighth round or the input of the ninth round of a ten round AES-128 algorithm, an attacker can retrieve the whole AES-128 key with two faulty ciphertexts.

In the present work we present a fault based side-channel attack on AES using a byte level fault. In the proposed attack the attacker intends to induce a byte level fault at the input of the 9^{th} round by affecting a byte of the round key. It may happen, accidentally due to imprecise control over the induction of fault, he ends up in inducing the fault at the input of eighth round or last round. Based on the fault signature in the ciphertext the attacker identifies the round in which the fault has been induced. Accordingly he chooses a strategy from three options. The first strategy, Strategy 1 is applied if the fault is induced at the input of last round. It uses a filter based on the differential property of the S-Box of Rijndael and uses computation in $GF(2^4)$. Strategy 2 and 3, are adopted when the fault is induced at the input of the ninth and eighth rounds respectively. Extensive experimentations have been performed on a PC and it has been found that the key can be obtained using only two faulty ciphertexts. The time required is a few seconds and the worst case complexity for the attacks is 2^8 for strategy 1 and 2^{16} for strategy 2 and 3. The idea of using algebraic equations (as in strategy 2 and 3) have also been adopted in [65]. But the equations proposed in the present work leads to much simpler analysis and reduces the number of faulty ciphertexts required from 40 to 2. The experimental results have been presented to demonstrate that the attack is a better fault attack than the previous fault attacks against AES.

The chapter is organised as follows: *section 9.2* describes the AES Rijndael algorithm. The fault model and the attack environment is stated in *section 9.3*. The three strategies of the fault attack on AES is elaborated in *sections 9.4, 9.5* and *9.6*. Finally the results of the work is compared to existing research and the work is concluded in *section 9.7*.

9.2 The Description of AES Rijndael Algorithm

The description of the Rijndael algorithm may be found in [184]. The typical round is described in the current subsection. The 128 bit message and key sizes have been considered, but the discussion is valid for all the sizes.

The 128 bit input block to Rijndael is arranged as a 4×4 array of bytes, known as the state matrix, refer to **Fig. 9.1**. The elements of the matrix are represented by the variable,b_{ij}, where $0 \leq i, j \leq 3$ and i, j refers to the row and column positions.

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

Figure 9.1: **The State Matrix of Rijndael**

The algorithm has ten rounds and the keys of each round are generated by a keyscheduling algorithm. The design of the key scheduling algorithm of Rijndael is such that the knowledge regarding any round key reveals the original input key (named as the master key) from which the round keys are derived. The input state matrix (plaintext) is transformed by the various round transforms. The state matrix evolves as it passes through the various steps of the cipher and finally emerges in the form of ciphertext.

The rounds of Rijndael use the following steps(**Fig. 9.2**):

1. The SubByte Step: The SubByte is the only non-linear step of the cipher. It is a bricklayer permutation consisting of an S-box applied to the bytes of the state. Each byte of the state matrix is replaced by its multiplicative inverse, followed by an affine mapping. Thus the input byte x is related to the output y of the S-Box by the relation, $y = A.x^{-1} + B$, where A and B are constant matrices[184].

2. The ShiftRows Step: Each row of the state matrix is rotated by a certain number of byte positions. This is a byte transposition step.

3. The MixColumn Step: The MixColumn is a bricklayer permutation operating on the state column by column. Each column of the state matrix is considered as a 4-dimensional vector where each element belongs to $GF(2^8)$. A 4×4 matrix M whose elements are also in $GF(2^8)$ is used to map this column into a new vector. This operation is applied on all the 4 columns of the state matrix [184].

4. AddRoundKey: Each byte of the array is exclusive-ored with a byte from a corresponding array of round subkeys.

The first 9 rounds of Rijndael are identical except the last round in which the Mix-Column step does not exist.

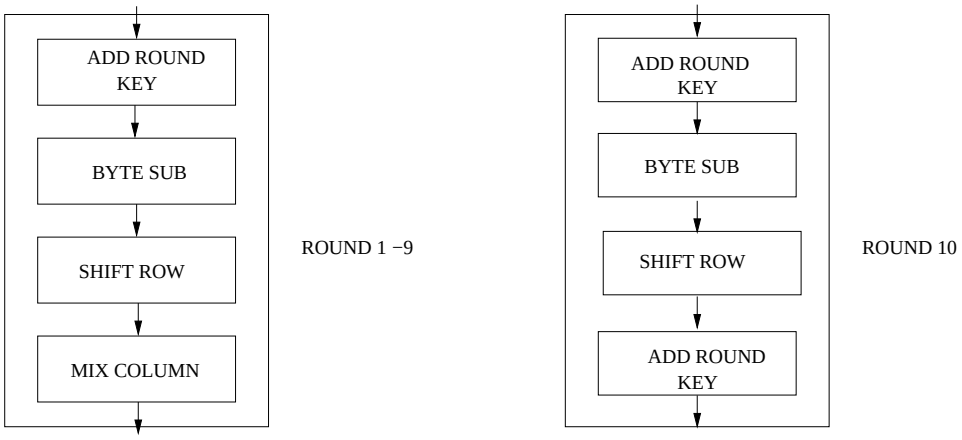


Figure 9.2: **The Round Transforms of Rijndael**

9.3 Fault Model Used and the Attack Environment

In this work, the fault assumed is a single byte fault. Single byte fault means the fault f_{ij} is injected in one particular byte b_{ij}, where i refers to the row position and j refers to the column position in the state matrix ($0 \leq i, j \leq 3$). The number of bits in the byte which are affected by the fault is indicated by $w(f_{ij})$, where $0 \leq w(f_{ij}) \leq 8$. If the fault f_{ij} is injected at the input of the r^{th} round then it is denoted as f_{ij}^{r}.

As the attacker does not have full control over the round in which the fault is injected, the value of r can vary. Keeping in mind the present technology of fault injection it is practical to assume that the value of r can deviate by atmost $\pm$ 1. If the attacker wishes to inject fault at the input of the ninth round, he may end up doing so at the input of the eighth round or the last round. In this work we therefore develop the attacking strategy so that he can detect the location of the fault (that is the round in which the fault is induced) from the output ciphertexts. Based upon the round in which the fault is injected the attacker uses three different strategies to reveal the key.

Before stating the fault attack techniques we outline two practical scenarios where the attack may be carried out. The scenarios show that depending upon the implementation of cryptographic hardware there are two different requirements on the

attacker in order to inject the fault at any precise round location.

- Scenario 1: Certain implementations of Rijndael requires pipelining at all stages (unrolled rounds), due to the requirement of throughput. Thus each key requires access to an unshared key memory. Hence, the entire key has to be stored in a key register or memory. In such a case the attacker wishes to cause faults in the value that is being read from the memory while leaving the value stored in the memory unaffected. This does not hamper the normal functionality of the device and is thus undetectable. Further, in such an implementation large number of faulty ciphertexts can be obtained, since the key stored in the memory is unaffected. Thus the requirement on the attacker in such a case is *Control on Fault Location.*
- Scenario 2: The other way in which block ciphers like AES-Rijndael are implemented is through iterative structures (rolled rounds) or a combination of unrolled and rolled rounds. In such a case the key is not stored in the memory and thus the requirement on the attacker is *Control on Fault Timing.*

Imprecise control over fault location or fault timing hinders the attacker to be able to inject a fault at a predetermined round key. In our present attack, we assume that the attacker intends to inject fault in a byte at the input of the 9^{th} round, but may end up in inducing the fault at the input of 8^{th} round or at the input of last round. In the following sections we present strategies through which the attacker identifies the round in which the fault is injected and accordingly he applies three alternate strategies to discover the key. The three alternate strategies are:

1. **Strategy 1:** When the attacker injects fault at the input of the last round
2. **Strategy 2:** When the attacker injects fault at the input of the penultimate round
3. **Strategy 3:** When the attacker injects fault at the input of the eighth round

9.4 Attack Based on Strategy 1

In this section we are going to describe a fault attack on AES where the fault is induced at the input of the last round AES S-Box.

The last round of AES Rijndael does not have a diffusion step and is a bytewise operation. In this attack a fault is induced at a byte position in the last round S-Box input. The attacker obtains a pair of ciphertexts (C_1, C_1'), where C_1 refers to the fault

free encryption and C_1' is the faulty ciphertext. The byte in which the fault has been induced can be trivially identified from the difference of C_1 and C_1'.

We assume in this attack that the attacker can induce a fixed byte fault. A filter based on $GF(2^4)$ operations is built using a differential property of the S-Box of AES. The attacker uses the filter to reduce the key space drastically. Results show that the filter pass only 2 keys, out of which one is the correct key. The correct key is confirmed using another pair of ciphertexts. Thus the key byte is revealed using only 2 faulty ciphertexts. The differential property and the developed filter is first stated, followed with the description of the actual attack.

9.4.1 The differential property of the Galois Field Inverse

The filter for strategy 1 is based on a differential property of the inverse step of AES Rijndael. The inverse mapping is a differentially 4-uniform mapping [154], means that for any non-zero input difference and any output difference the number of possible inputs have an upper bound of four. In this case we use a special case of the Galois Field Inverse to develop a filter using computations in the sub-field. This makes the acceptance or rejection of the guessed keys fast.

The inverse of a $GF(2^8)$ element can be computed by expressing it as a polynomial of the first degree with coefficients from $GF(2^4)$, [200]. In order to map an element in $GF(2^8)$ to an element in the composite field $GF((2^4)^2)$, the element is multiplied with a transformation matrix, T.

One such matrix T is as follows [162],

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

The current work uses $(x^2 + x + w^{14})$ as the irreducible polynomial for Galois Multiplication in the composite field. Here w is the primitive element of $GF(2^4)$ and the field polynomial for computation in $GF(2^4)$ is $x^4 + x + 1$. Thus, $w^{14} = w^3 + 1$. The multiplicative inverse for the arbitrary polynomial $(bx + c)$, where (b, c) are in $GF(2^4)$, can be computed as[200]:

$$dx + e = (bx + c)^{-1} = b(b^2w^{14} + bc + c^2)^{-1}x + (b + c)(b^2w^{14} + bc + c^2)^{-1} \quad (9.1)$$

The architecture for the computation of inverse is shown in **Fig. 9.3**.

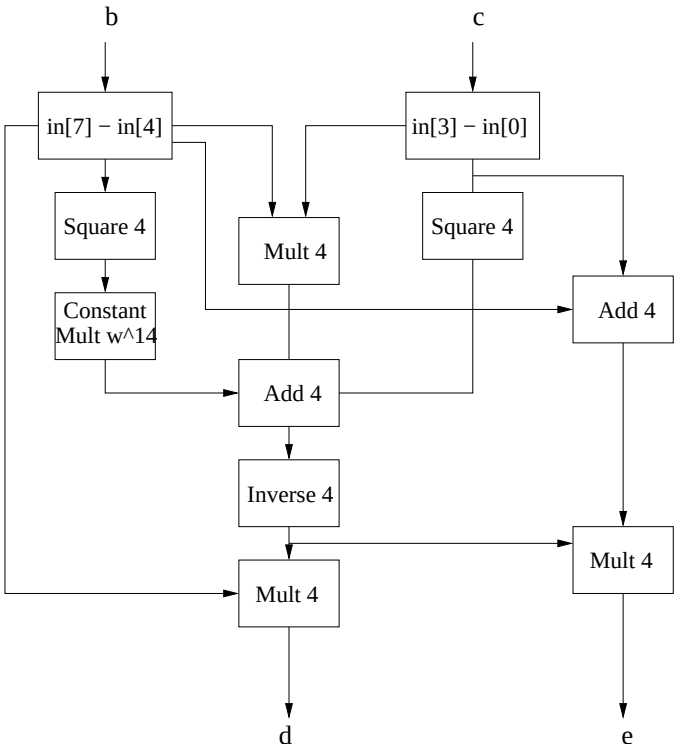


Figure 9.3: **Computation of the Inverse in $GF((2^4)^2)$.**

Let two such parallel inverse blocks be considered(**Fig. 9.4**). One of them operates on X, an element in $GF((2^4)^2)$. The other one operates on the complement of X which is X'.

Let $X = bx + c$ and therefore $X' = b'x + c'$ where b, b', c and c' all belongs to $GF(2^4)$. Also b is the complement of b'. Similarly c is the complement of c'.

The inverses are $f(X) = dx + e = X^{-1}$ and $f(X') = d'x + e' = (X')^{-1}$, where d, d', e and e' all belongs to $GF(2^4)$.
Hence the differential, $\delta e = e + e'$

$= (b + c)(b^2w^{14} + bc + c^2)^{-1} + (b' + c')((b')^2w^{14} + b'c' + (c')^2)^{-1}$

$= (b + c)\delta B$, as $(b + c) = (b' + c')$.

Here $\delta B = (b^2w^{14} + bc + c^2)^{-1} + ((b')^2w^{14} + b'c' + (c')^2)^{-1}$.
Thus, the Galois Field inverse yields a differential property in its lower nibble(4 bits)

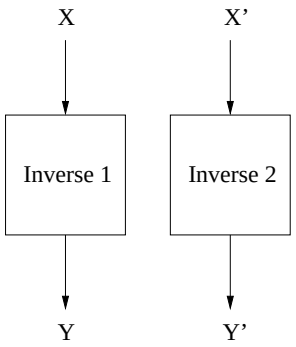


Figure 9.4: **Differential Property of the Inverse module in $GF((2^4)^2)$.**

as the following equation is obtained,

$$\delta e = (b + c)\delta B \tag{9.2}$$

9.4.2 Attack Using the Filter

The filter developed in the previous subsection may be used to attack Rijndael using the fact that the field $GF(2^8)$ is isomorphic to the composite field $GF((2^4)^2)$. The filter developed is used to reduce the key space drastically. The operations of Rijndael in the final round are bytewise. Hence in the last round of the algorithm (**Fig. 9.5**) the transformation of each of the bytes may be observed independently. The attacker obtains a pair of ciphertexts, out of which one is the faulty ciphertext. The faulty ciphertext is obtained when there is a byte fault at the input of the last round S-Box. The Rijndael algorithm has transformations in $GF(2^8)$.

Let us consider one byte of the state matrix after the 9^{th} round of Rijndael algorithm, say the $(i,j)^{th}$ byte of the state matrix. The byte is xored with the 9^{th} round key byte, $K9_{i,j}$. The attacker induces a fault in which he converts the 9^{th} round key byte to $K9'_{i,j}$ by finding the complement of some of the bits. The attacker replaces byte $K9_{i,j}$ by:

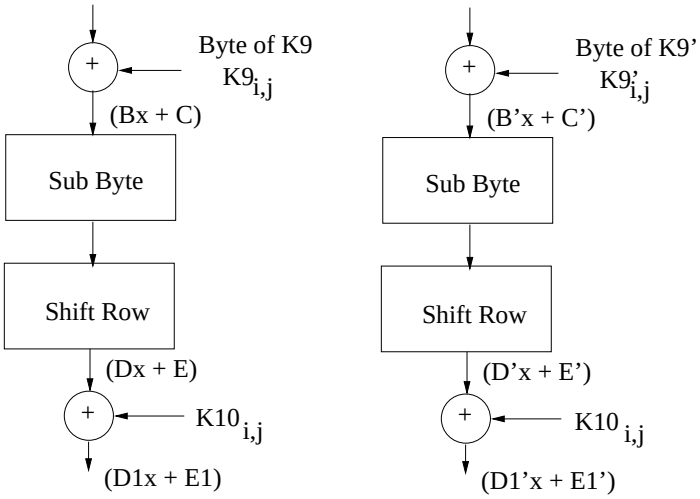
$K9'_{i,j} = K9_{i,j} \oplus T^{-1}F$, where F is an all one vector.
Since, $T^{-1}F = [11000100]^t$ the attacker only gets the $1^{st}, 2^{nd}$ and 6^{th} bits of the 9^{th} round key complemented.

In order to perform the fault attack, the cryptanalyst studies two sets of the cipher. In one set, **Fig. 9.5** he considers the actual cipher where the elements belongs to $GF(2^8)$. In the figure, the first block (**Fig. 9.5(a)**) refers to the fault free tenth round, whereas the second block (**Fig. 9.5(b)**) refers to the faulty last round of Rijndael.

The other set, **Fig. 9.6** is a reference cipher where the elements belong to $GF((2^4)^2)$. In this figure, also as in the first set, the first block refers to the fault free case, whereas the second block refers to the case where fault has been injected at the input of the last round.

Due to the isomorphism between $GF(2^8)$ and $GF(2^4)^2$ the two sets (**Fig. 9.5** and **Fig. 9.6**) can be related by the matrix T.

The reference cipher, as already described is built by describing all the states and the transformations of the original cipher in the composite field $GF((2^4)^2)$. In



a. Fault Free Cipher b. Faulty Cipher

Figure 9.5: **Last round of Rijndael in** $GF(2^8)$

the reference ciphers $k9_{i,j}$ is the $(i,j)^{th}$ byte of the 9^{th} round key. It has a one-one mapping with $K9_{i,j}$, the corresponding key byte in the original cipher, due to the isomorphism between $GF(2^8)$ and $GF((2^4)^2)$.

Thus, $k9_{i,j} = T(K9_{i,j})$ and $k9'_{i,j} = T(K9'_{i,j}) = T(K9_{i,j} \oplus T^{-1}F) = T(K9_{i,j}) \oplus F$. Thus in the reference cipher the 9^{th} round key is complemented. Hence, $k9'_{i,j}$ is the bit complement of $k9_{i,j}$. Thus in the reference cipher (**Fig. 9.6**) $(b'x + c')$ is the bit complement of $(bx + c)$.

Let the affine transformation in $GF(2^8)$ be represented as $Y = A(X) + B$ and in $GF((2^4)^2)$ reference cipher be represented as $y = A'(x) + B'$, where $(X, Y) \in GF(2^8)$ and $(x, y) \in GF(2^4)^2$. Then, $y = T(Y)$ and $x = T(X)$.
Thus, $y = T(Y) = TA(X) + TB$
$= TAT^{-1}(x) + TB$.
Hence, $A' = TAT^{-1}$ and $B' = TB$. The SubByte operation in the reference cipher is

$y = A'(x^{-1}) + B'$, where the inverse is in $GF(2^4)^2$.

We compute the differentials of the output bytes from the two blocks of the reference ciphers depicted in **Fig. 9.6**. The differential from the reference ciphers (**Fig. 9.6**) is denoted by,
$(\delta dx + \delta e) = (d1x + e1) + (d1'x + e1') = A'[(bx + c)^{-1} + (b'x + c')^{-1}]$.
The differential in the reference ciphers can be related to that in the the original cipher in $GF(2^8)$ (**Fig. 9.5**) by,

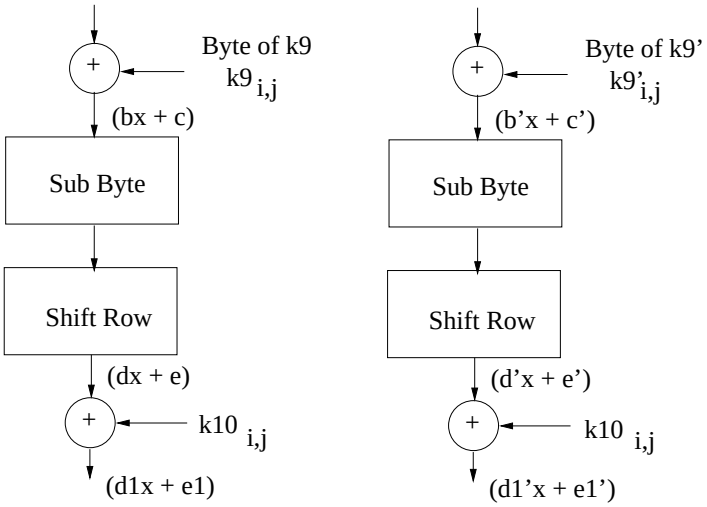
$$\delta Dx + \delta E = T^{-1}A'[(bx + c)^{-1} + (b'x + c')^{-1}].$$

In other words,

$$\begin{aligned} (bx + c)^{-1} + (b'x + c')^{-1} &= A'^{-1}T(\delta Dx + \delta E) \\ or, (b + c)\delta B &= \text{lower nibble}\{A'^{-1}T(\delta Dx + \delta E)\}, \text{from } \textbf{equation 9.2} \quad (9.3) \\ or, (b + c)\delta B &= \text{lower nibble}\{TA^{-1}(\delta Dx + \delta E)\} \end{aligned}$$

since $(bx + c)$ and $(b'x + c')$ are bit complements of each other and $A' = TAT^{-1}$. In the above equations *lowernibble* refers to the least significant 4 bits.

Thus we arrive at a filter for rejecting the wrong guesses while trying to attack full round Rijndael.



a. Fault Free Cipher b. Faulty Cipher

Figure 9.6: **Last round of Rijndael in** $GF(2^4)^2$

Based on the above filter an algorithm to attack the full round Rijndael Cryptosystem is outlined below. It may be noted that the revelation of any round key of Rijndael leads to the knowledge of the original master key.

Algorithm 9.1 : To obtain the tenth round Key byte: $K10_{i,j}$ **(that is the** $(i,j)^{th}$ **byte of the tenth round key).** *Let K be the expanded round keys for all the ten rounds in the fault free cipher. Thus, K=$\{K0, K1, \ldots, K9, K10\}$. Likewise for the faulty cipher, K' =$\{K0, K1, \ldots, K9', K10\}$, indicating that the ninth round round key is injected with the fault*

- *Step1: Encrypt P using K to obtain C. Thus $C = E(P, K)$. One of the bytes of the state matrix of C is $(D1x + E1)$.*
- *Step2: Encrypt P using K' to obtain C'. The key K' differs from K only in the fact that the $(i,j)^{th}$ byte of the 9th round key is modified. The $1^{st}, 2^{nd}$ and 6^{th} bits of $K9_{i,j}$ are complemented. The corresponding byte of the faulty state matrix is $D1'x + E1'$*
- *Step3: Xor the output bytes to obtain $\delta Dx + \delta E$; $\delta Dx + \delta E$ = (D1x + E1) + $(D1'x + E1')$*
- *Step4: Randomly guess (b,c) where $b, c \epsilon GF(2^4)$*
- *Step5: Use the filter $(b + c)\delta B$=lower nibble$\{TA^{-1}(\delta Dx + \delta E)\}$*
  where $\delta B = (b^2w^{14} + bc + c^2)^{-1} + ((b')^2w^{14} + b'c' + (c')^2)^{-1}$, where (b', c') are complements of (b, c)
  If the guess passes the filter evaluate possible tenth round key as
  $K10_{i,j} = T^{-1}[A'(bx + c)^{-1} + B'] + (Dx + E)$
  $= [AT^{-1}(bx+c)^{-1}+B]+(Dx+E)$, *since* $A' = TAT^{-1}$ *and* $B' = TB$.
- *Step6: Check whether $K10_{i,j}$ gives $(D'x + E')$ from $(B'x + C')$. If it is true then $K10_{i,j}$ is a possible key byte.*

The above algorithm is applied 16 times concurrently for faults at all the byte positions of the state matrix. Statistical analysis have been performed and it has been found that the developed filter effectively screens out wrong guesses. A significant reduction has been observed in the total number of keys, proving the efficacy of the filter designed to remove wrong guesses.

9.4.3 A Working Example

In the present section an example has been cited to explain the working of the above attack. The Rijndael algorithm is applied on a 128 bit plaintext PT. The plaintext

may be conceptualised as a 4×4 matrix, in which each element is a byte. In the present context let,

$$\mathbf{PT} = \begin{pmatrix} 00000000 & 10001000 & 00110001 & 11100000 \\ 01000011 & 01011010 & 00110001 & 00110111 \\ 11110110 & 00110000 & 10011000 & 00000111 \\ 10101000 & 10001101 & 10100010 & 00110100 \end{pmatrix}$$

The input key is

$$\mathbf{K0} = \begin{pmatrix} 00101011 & 00101000 & 10101011 & 00001001 \\ 01111110 & 10101110 & 11110111 & 11001111 \\ 00010101 & 11010010 & 00010101 & 01001111 \\ 00010110 & 10100110 & 10001000 & 00111100 \end{pmatrix}$$

The keyscheduling algorithm generates the round keys for the encryption algorithm. Thus the total set of keys for the 10 rounds is $K = \{K0, K1, K2, \ldots, K9, K10\}$. In the process the 9^{th} round key is denoted as

$$\mathbf{K9} = \begin{pmatrix} 10101100 & 00011001 & 00101000 & 01010111 \\ 01110111 & 11111010 & 11010001 & 01011100 \\ 01100110 & 11011100 & 00101001 & 00000000 \\ 11110011 & 00100001 & 01000001 & 01101110 \end{pmatrix}$$

The cryptanalyst obtains the ciphertext obtained by encrypting PT by K. Let the ciphertext be

$$\mathbf{CT} = \begin{pmatrix} 10010101 & 11110110 & 11011010 & 11101000 \\ 11101111 & 00100101 & 01111101 & 11100001 \\ 01110011 & 01101110 & 11001001 & 11110010 \\ 00101110 & 11101011 & 00000101 & 01000001 \end{pmatrix}$$

Now due to a malicious effort made by the attacker the 9^{th} round key is manipulated. Thus the $K9$ is replaced in K by $K9'$, where

$$\mathbf{K9'} = \begin{pmatrix} 10001111 & 00111010 & 00001011 & 01110100 \\ 01010100 & 11011001 & 11110010 & 01111111 \\ 01000101 & 11111111 & 00001010 & 00100011 \\ 11010000 & 00000010 & 01100010 & 01001101 \end{pmatrix}$$

Note that each byte of $K9$ are changed such that the $6^{th}, 2^{nd}$ and 1^{st} bits are complemented. Thus $K' = \{K0, K1, K2, \ldots, K9', K10\}$. The eavesdropper then observes the modified ciphertext obtained by encrypting P with K'. Thus he obtains another cipher

$$\mathbf{CT'} = \begin{pmatrix} 01100011 & 10100110 & 11100011 & 00011000 \\ 00011101 & 00110100 & 00111100 & 01000000 \\ 00110111 & 11111010 & 00111010 & 00011001 \\ 10111110 & 01001101 & 11000010 & 01111010 \end{pmatrix}$$

The attacker uses the knowledge of CT and CT' to arrive at the round keys. The keyscheduling algorithm of Rijndael has the property that the revelation of any round keys leads to the calculations of all the round keys. In the present attack the 10^{th} round key is evaluated. The present example has

$$\mathbf{K10} = \begin{pmatrix} 11010000 & 11001001 & 11100001 & \mathbf{10110110} \\ 00010100 & 11101110 & 00111111 & 01100011 \\ 11111001 & 00100101 & 00001100 & 00001100 \\ 10101000 & 10001001 & 11001000 & 10100110 \end{pmatrix}$$

The **table 9.1** shows the result of a run of the attack. The cryptanalyst uses the developed filter in reducing the possible key space. Further, let us take a random index of the state matrix to investigate the working of the attack. For example for index 03 in **table 9.1** the two possible keys passing the filter are {11000111, **10110110**}. It may be observed that the correct key byte is 10110110 which exists in the reduced set. In order to ascertain which of the two key bytes is the actual key the attacker obtains another pair of ciphertexts and intersects the two reduced sets. The workload in the present case may be calculated from **table 9.1**. Since, the last round of AES is bytewise, the entire key may be obtained through the parallel operation of the filter, and thus in 2 runs of the algorithm.

The working example shows that the application of the filter reduces the key space from 2^{128} to 2, with a complexity of 2^8. With the development of the work referred to in [61] the present attack becomes the most powerful of all the fault attacks in existing literature. The reasons are:

1. The number of faulty ciphertexts is only 2, with the time complexity of the attack being only 2^8.

2. The fault can be injected at any byte and the corresponding fault position is easily detected from the ciphertext pair, since the last round is bytewise

3. The filter based on which the pruning of keys is performed is based on $GF(2^4)$ operations and is thus very fast. The easy and fast rejection of wrong keys leads

Table 9.1: **Reduction of keyspace in Strategy 1**

Index $I_{i,j}$ of state matrix	Fault Free Byte $(Dx + E)$		Faulty Byte $(D'x + E')$		No of keys passing the filter
	D	E	D'	E'	
00	1001	0101	0110	0011	2
01	1111	0110	1010	0110	2
02	1101	1010	1110	0011	2
03	1110	1000	0001	1000	2
10	1110	1111	0001	1101	2
11	0010	0101	0011	0100	2
12	0111	1101	0011	1100	2
13	1110	0001	0100	0000	2
20	0111	0011	0011	0111	2
21	0110	1110	1111	1010	2
22	1100	1001	0011	1010	2
23	1111	0010	0001	1001	2
30	0010	1110	1011	1110	2
31	1110	1011	0100	1101	2
32	0000	0101	1100	0010	2
33	0100	0001	0111	1010	2

NB: Index $I_{i,j}$ means $(i+1)^{th}$ row and $(j+1)^{th}$ column of the state matrix.

to real time attack, with the key being cracked in few seconds in a PC.

9.5 Attack Based on Strategy 2

Our next fault attack is when the fault is induced in a byte of the state matrix before the SubByte of the penultimate round (i.e. the 9^{th} round).

The 9^{th} round has a diffusion step, so a disturbance in one byte affects 4 bytes at the output. The last round does not have a diffusion step and so the disturbances remain in 4 bytes of the state matrix. If one traces the disturbance in the state matrix through the last two rounds the following properties can be identified. These properties can be utilized to develop an attack against the block cipher, where the fault induced may be random.

9.5.1 Property of the State Matrix

Fig. 9.7 shows the propagation of the fault, when it is induced in a byte at the input of the ninth round Byte Sub. In the figure we show the case when the fault is induced at the 00^{th} byte of the state matrix. The fault value is say f, and after the Byte Sub, the fault value is transformed to a value f'. The byte fault is propagated to four byte positions. As the figure suggests, the faults at the input of the tenth round Byte Sub have values of $2f'$, f', f' and $3f'$. The faulty values after the tenth round Byte Sub gets transformed into F_1, F_2, F_3 and F_4. The attacker obtains a pair of ciphertext, (CT, CT'). The ciphertext CT is a fault free ciphertext and the ciphertext CT' is the ciphertext, when a fault is induced in a byte. When a bytewise fault is induced at the input of the ninth round Byte Sub, the difference of the ciphertexts CT and CT' has the pattern as shown in **Fig. 9.7** after the tenth round ShiftRow. Observing the signature the attacker gets confirmed about the fact that he has been able to induce fault before 9^{th} round MixColumn.

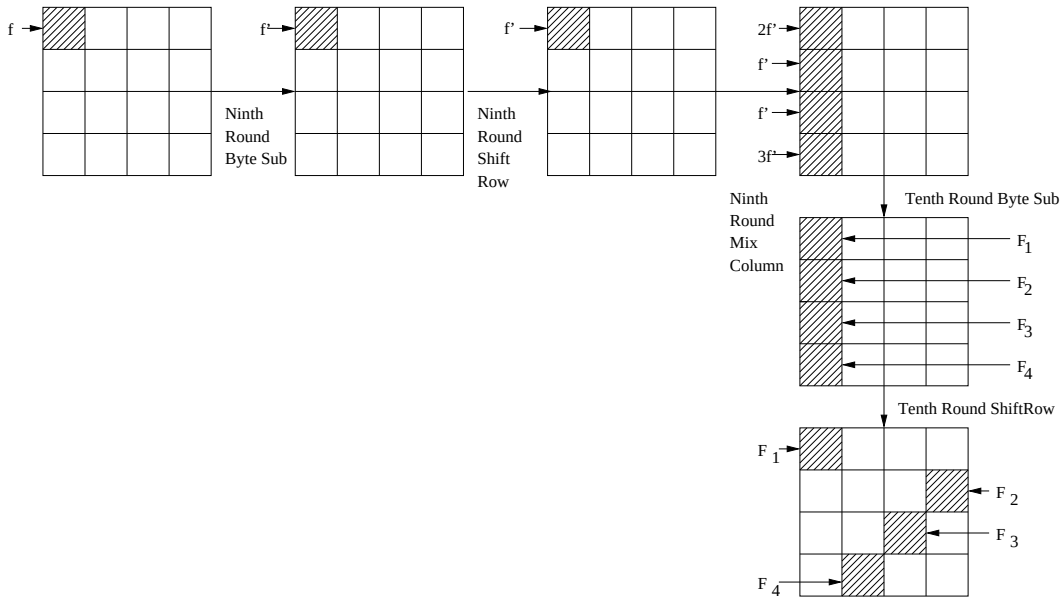


Figure 9.7: **Propagation of Fault Induced in the input of the ninth round of Rijndael**

The fault pattern as shown in **Fig. 9.7** depicts the difference between the fault free ciphertext CT and the faulty ciphertext CT'. Let the values of the bytes shaded in **Fig. 9.7** after the tenth round Shift Round Key in the ciphertext CT be denoted by x_1, x_2, x_3 and x_4. Then the corresponding values for the fault free ciphertext CT' is denoted by $x_1 + F_1$, $x_2 + F_2$, $x_3 + F_3$ and $x_4 + F_4$. Here the sign + stands for the bit-wise exclusive-or operation of two bytes. The corresponding key bytes are K_1,

K_2, K_3 and K_4.

The fault pattern gives the following set of equations:

$$\begin{aligned} ISB(x_1 + K_1) + ISB(x_1 + F_1 + K_1) &= 2[ISB(x_2 + K_2) + ISB(x_2 + F_2 + K_2)] \\ ISB(x_2 + K_2) + ISB(x_2 + F_2 + K_2) &= ISB(x_3 + K_3) + ISB(x_3 + F_3 + K_3) \\ ISB(x_4 + K_4) + ISB(x_4 + F_4 + K_4) &= 3[ISB(x_2 + K_2) + ISB(x_2 + F_2 + K_2)] \end{aligned}$$

In the above set of equations ISB stands for the Inverse Byte Sub operation, which is defined as the inverse of the Byte Sub step. The values of (x_1, x_2, x_3, x_4) and $(x_1 + F_1, x_2 + F_2, x_3 + F_3, x_4 + F_4)$ are known to the adversary. The attacker intends to compute the values of K_1, K_2, K_3 and K_4 from the equations. The attacker evaluates the keys as follows: The attacker guesses the bytes K_1 and K_2 and checks whether they satisfy the first equation. The solution sets for the second and third equations are searched in parallel. Thus the time complexity of the key conjuring is 2^{16}. Finally since the variable K_2 is in all the three equations, the solution set of K_2 from each equation is intersected to arrive at a reduced solution space for K_2. The reduced space of K_2 is then used to find a reduced set of K_1, K_3 and K_4 from the three equations. It may be noted that the above attack does not depend upon the value of the induced fault, which may be random. This completes a single pass of the algorithm with one (CT, CT') pair. In order to ascertain the key bytes, further passes of the algorithm are run with other (CT, CT') pairs. We have experimentally verified that the number of passes of the algorithm does not go beyond two, thus revealing the key with only two faulty encryptions.

9.5.2 A Working Example

In the present section we outline the working of the attack through an example. The example may be compared with the results shown in [65, 66] which are the two most recently reported works on Fault Attacks on AES. In this attack scenario we assume that the fault is induced in a byte at the input of the ninth round. In such a scenario the solution proposed in [65] requires about 40 to 50 faulty ciphertexts. The attack strategy proposed in [66] requires two faulty ciphertexts to evaluate the key. In both the work proposed in this work and [66] a single pass of the algorithm reduces the key space of four last round key bytes. In order to fully reveal the key the attack is applied again by considering another three faulty byte positions. The work proposed in [66] after one pass of the algorithm (i,e with a single (CT, CT') pair) the remaining number of candidate keys is about 1038. Thus if only one faulty encryption is possible the key space is reduced to approximately $1038^4 \approx 2^{40}$. We show in the following example that a single pass of the proposed strategy is better

than that of [66] and can also ascertain the key in exactly two runs of the algorithm.

Let the plaintext be:

$$\mathbf{PT_1} = \begin{pmatrix} 00000001 & 11111110 & 10000001 & 11111100 \\ 10100110 & 01101011 & 11100001 & 10100011 \\ 11110100 & 10100010 & 11110111 & 01111000 \\ 01000000 & 10100101 & 10001110 & 11110000 \end{pmatrix}$$

The plaintext is encrypted using Rijndael encryption algorithm. The key matrix is:

$$\mathbf{K0} = \begin{pmatrix} 11100111 & 00101000 & 10010101 & 01100001 \\ 01110110 & 10101110 & 11110111 & 11001111 \\ 00010101 & 11011010 & 00110101 & 01011111 \\ 00111110 & 10000010 & 10100100 & 01001100 \end{pmatrix}$$

and the tenth round key is:

$$\mathbf{K10} = \begin{pmatrix} \mathbf{00101101} & 00010100 & 00011101 & 11000101 \\ 11110000 & 11100010 & 11010111 & \mathbf{01000001} \\ 11100010 & 00000111 & \mathbf{10100010} & 10110010 \\ 01010101 & \mathbf{11101010} & 01110000 & 00111100 \end{pmatrix}$$

The corresponding ciphertext is:

$$\mathbf{CT_1} = \begin{pmatrix} 11101110 & 01111111 & 11110100 & 01100101 \\ 01011000 & 01001101 & 10110101 & 10110101 \\ 11111001 & 00101001 & 11010010 & 11100010 \\ 10000101 & 00111011 & 11111100 & 11110111 \end{pmatrix}$$

The faulty ciphertext for a random fault induced in the 00^{th} position at the input of the ninth round leads to the following faulty ciphertexts:

$$\mathbf{CT'_1} = \begin{pmatrix} \mathbf{00101111} & 01111111 & 11110100 & 01100101 \\ 01011000 & 01001101 & 10110101 & \mathbf{11111111} \\ 11111001 & 00101001 & \mathbf{01111000} & 11100010 \\ 10000101 & \mathbf{10010101} & 11111100 & 11110111 \end{pmatrix}$$

The bytes in the faulty ciphertexts which are bolded show how the fault has propagated. Using the above signature the attacker identifies the round input where the fault is induced. Accordingly, he applies the equations of strategy 2. Experiments

show that only 256 possible subkeys satisfy the equations. Thus, with a single faulty encryption the key space reduces to $256^4 \approx 2^{32}$, which is better than the result of [66].

The actual key can be ascertained if we consider another faulty encryption. Let another plaintext be:

$$\mathbf{PT_2} = \begin{pmatrix} 10101001 & 01010110 & 10001101 & 11001100 \\ 11110110 & 01001011 & 10100011 & 10000011 \\ 11110100 & 00000010 & 10100110 & 11110000 \\ 11110000 & 11100101 & 00111100 & 11110001 \end{pmatrix}$$

The corresponding ciphertext is:

$$\mathbf{CT_1} = \begin{pmatrix} 00001101 & 11111101 & 00111011 & 10001101 \\ 11000110 & 00100011 & 11110101 & 01110001 \\ 11001111 & 00101110 & 10100101 & 11011010 \\ 01110011 & 00001111 & 10101101 & 11000100 \end{pmatrix}$$

and the faulty ciphertext is

$$\mathbf{CT'_1} = \begin{pmatrix} \mathbf{01011100} & 11111101 & 00111011 & 10001101 \\ 11000110 & 00100011 & 11110101 & \mathbf{10100001} \\ 11001111 & 00101110 & \mathbf{11111011} & 11011010 \\ 01110011 & \mathbf{00011011} & 10101101 & 11000100 \end{pmatrix}$$

The signature of the fault indicates that the fault is induced at the input of the ninth round of Rijndael. Thus the equations of strategy 2 is applied and it is found that 272 different possible tenth round key satisfy the equations.

Intersection of the two solution sets leaves only one element, which is the correct solution. In this example we arrive at the key bytes: K_1=11010000, K_2=01100011, K_3=00001100 and K_4=10001001. The bold elements in the matrix of $K10$ show that the guesses are correct.

9.6 Attack Based on Strategy 3

In this attack we assume that the adversary has induced fault in a byte of the input to the eighth round. If the fault is induced in a byte of the state matrix, which is input to the eighth round, the disturbance spreads to the entire state matrix when

it emerges out after the tenth round. However there is a definite pattern in the state matrix after the ninth round which may be exploited to develop a fault attack. We assume that the attacker observes the difference between the fault free ciphertext and the faulty ciphertext (C and C') to comprehend where the fault has been induced. We have shown that the attacker is able to detect from the difference whether the fault is induced at the input of the ninth round or the last round. If neither of the cases occur we assume that the fault is induced at the input of the eighth round.

9.6.1 Property of the State Matrix

Fig. 9.8 shows the diffusion of a byte fault induced at the input of the eighth round. Similar to the previous section the various round operations transform the initial value of the fault f. The attacker observes, like in the previous cases two ciphers - one fault free and the other faulty. The difference of the state matrices of the two ciphers are depicted in **Fig. 9.8**.

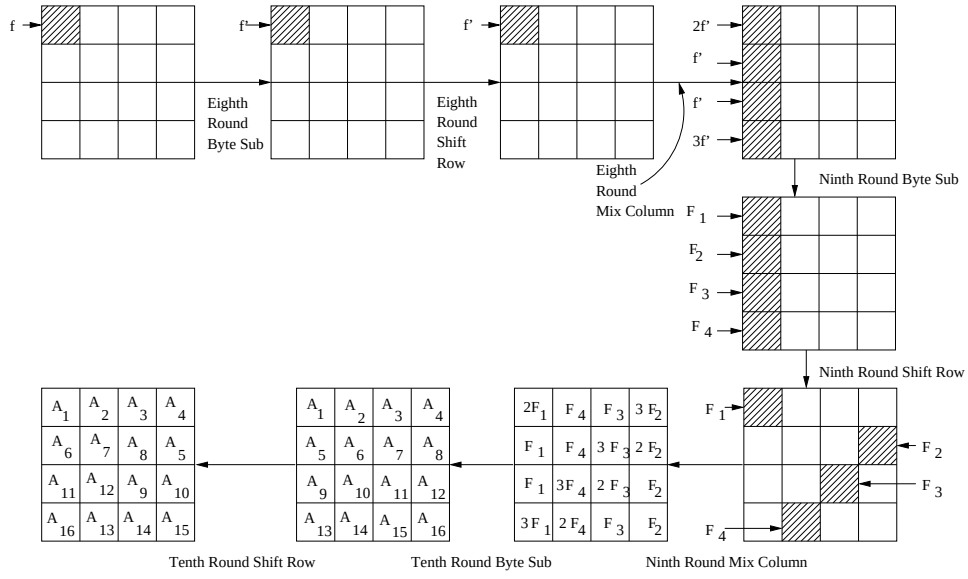


Figure 9.8: **Propagation of Fault Induced in the input of eighth round of Rijndael**

The attacker knows the value of CT and CT' from the two ciphertexts that he obtain. Let, the two ciphertexts be represented by:

$$\mathbf{CT} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{pmatrix}$$

and

$$\mathbf{CT'} = \begin{pmatrix} x_1 + A_1 & x_2 + A_2 & x_3 + A_3 & x_4 + A_4 \\ x_5 + A_6 & x_6 + A_7 & x_7 + A_8 & x_8 + A_5 \\ x_9 + A_{11} & x_{10} + A_{12} & x_{11} + A_9 & x_{12} + A_{10} \\ x_{13} + A_{16} & x_{14} + A_{13} & x_{15} + A_{14} & x_{16} + A_{15} \end{pmatrix}$$

The corresponding key matrix for the tenth round is:

$$\mathbf{K_{10}} = \begin{pmatrix} K_{00} & K_{01} & K_{02} & K_{03} \\ K_{10} & K_{11} & K_{12} & K_{13} \\ K_{20} & K_{21} & K_{22} & K_{23} \\ K_{30} & K_{31} & K_{32} & K_{33} \end{pmatrix}$$

We note the state of the differences after the ninth round shift row from **Fig. 9.8**. Combining the above facts we obtain the following set of equations to evaluate the values of the key bytes K_{00}, K_{13}, K_{22} and K_{31}:

$$\begin{aligned} ISB(x_1 + K_{00}) + ISB(x_1 + A_1 + K_{00}) &= 2[ISB(x_8 + K_{13}) + ISB(x_8 + A_5 + K_{13})] \\ ISB(x_8 + K_{13}) + ISB(x_8 + A_5 + K_{00}) &= ISB(x_{11} + K_{22}) + ISB(x_{11} + A_9 + K_{22}) \\ [ISB(x_{14} + K_{31}) + ISB(x_{14} + A_{13} + K_{31}) &= 3[ISB(x_8 + K_{13}) + ISB(x_8 + A_5 + K_{13})] \end{aligned}$$

The unknowns in the above set of equations is the value of the key bytes K_{00}, K_{13}, K_{22} and K_{31}. The attacker similar to the previous strategy obtains reduced solution spaces for the bytes K_{00}, K_{13}, K_{22} and K_{31} from the three equations. The worst case complexity for one pass of the algorithm is 2^{16} and is again independent of the value of the fault induced. Similar to strategy 2, another solution set for the key bytes is obtained with another CT and CT' pair. The two solution sets are intersected to arrive at the correct key bytes, as the intersection set leaves only one element.

It may be noted that the equations of strategy 2 and 3 are identical and thus the solutions are of similar nature. A one passes of the algorithm leaves on the average

Table 9.2: **Comparison of Existing Fault Attacks on AES**

Reference	Fault Model	Fault Location	No. of Faulty Encryptions
[64]	Force 1 bit to 0	Chosen	128
[64]	Implementation Dependent	Chosen	256
[63]	Switch 1 bit	Any bit of chosen bytes	≈ 50
[63]	Disturb 1 byte	Anywhere among 4 bytes	≈ 250
[65]	Disturb 1 byte	Anywhere between last two MixColumn	≈ 40
[66]	Disturb 1 byte	Anywhere between 7^{th} round and 8^{th} round MixColumn	2
This chapter	Disturb 1 byte	Anywhere between 7^{th} round MixColumn and last round input	2

of 256 possible candidate keys and with another faulty encryption identifies the exact candidate key.

9.7 Comparison of Results and Conclusions

In this section we compare the existing fault based attack on AES with the help of **table 9.2**.

Thus the comparisons show that the current fault attack requires the minimum of faulty encryptions in order to derive the key like [66]. Also the fault location can be anywhere between the seventh round MixColumn (effectively the input of the eighth round) and the input of the last round of Rijndael. Also, if the work reported in [66] be compared with the present attack based on the result with one faulty encryption, then the present attack reduces the number of candidate keys to an average of about 256 compared to 1036 required in the previous attack. Often the second fault induction may not be possible, in such a case the present attack is almost $(\frac{1036}{256})^4 \approx 1000$ times more powerful than the work reported in [66]. Thus the present attack is better than the previous fault attacks on AES. The class of attacks thus obviates the necessity of fault tolerance in cipher structures.

Chapter 10

Customizing Cellular Message Encryption Algorithm

10.1 Introduction

Cellular Message Encryption Algorithm (CMEA) [2] has been developed by the Telecommunications Industry Association (TIA) to encrypt digital cellular phone data. CMEA is one of the four cryptographic primitives specified for telecommunications and is designed to encrypt the control channel, rather than the voice data. It is a block cipher which uses a 64 bit key and operates on a variable block length. CMEA is used to encrypt the control channels of cellular phones. It is distinct from ORYX, an also insecure stream cipher that is used to encrypt data transmitted over digital cellular phones.

In March 1997, Counterpane Systems and UC Berkeley jointly [3] published attacks on the cipher showing it had several weaknesses. In the paper the authors have presented several attacks on CMEA which are of practical threat to the security of digital cellular systems. The authors describe an attack on CMEA which requires $40 - 80$ known plaintexts, has time complexity about $2^{24} - 2^{32}$, and finishes in minutes or hours of computation on a standard workstation. The authors point out that the cryptanalysis of CMEA underscores the need for an open cryptographic review process. Thus having faith on new algorithms which are designed close door is always dangerous. The use of such algorithms can lead to a total collapse of the cellular telephonic industry. CMEA is used to protect sensitive control data, such as the digits dialled by the cellphone user. A successful break of CMEA might reveal user calling patterns. Finally, compromise of the control channel contents could lead to the leaking of any confidential data (like credit card numbers, bank account numbers

and voice mail PIN numbers) that the user types on the keypad.

Following the revelation of the weakness of CMEA, a patchup algorithm called ECMEA was standardised by TIA. A further enhancement of ECMEA, called SCEMA, is also developed [2]. However according to [3] the previous cryptanalysis of all the crypto-algorithms proposed by TIA clearly demonstrate that there is a need of explicitly stating security assumptions during every step of the design. Also security components should not be reused without thoroughly examining the implications of reuse. Although it has been proposed that the future generation cellular networks (CDMA 2000 1X Revision A) will use AES(Rijndael)[184], the implementation constraints of a wireless network might prove to be a concern. This motivates the design of special ciphers for wireless telephones (networks) but at the same time which are evaluated meticulously. The security margins of such algorithms must be stated so as to increase confidence in the ciphers. In other words, dedicated as well as standard block cipher security analysis should be presented for the ciphers which are used to prevent frauds in such important networks. In these lines, the present work revisits the CMEA algorithm. The algorithm has been analyzed to understand the reasons of its insecurity. Based upon the analysis the CMEA has been modified to CMEA-I. The new algorithm has been analyzed and it has been shown that the original attacks does not work against the cipher. Also the diffusion and confusion properties of CMEA-I has been demonstrated by means of Avalanche analysis. The security of CMEA depends on the strength of the T-Box. Hence, security margins have been presented to establish that the T-Box provides sufficient security margins against linear and differential cryptanalysis.

The chapter is organised as follows. In *section 10.2* the preliminaries have been stated which details the original CMEA algorithm and the attacks against it. In *section 10.3* the CMEA algorithm has been analyzed to understand why the algorithm breaks in the face of the attacks detailed in *section 10.2*. *Section 10.4* presents the customized CMEA with necessary modifications to plague the existing weaknesses of CMEA. *Section 10.5* performs a security analysis of CMEA-I. The section shows how CMEA-I prevents the attack proposed in [3]. The diffusion and confusion properties of CMEA-I has also been analyzed in the section using Avalanche criterion. Linear and differential cryptanalysis has been performed on the T-Box in the section. The efficiency of the cipher has been discussed in *section 10.6*. Finally *section 10.7* concludes the work.

10.2 Preliminaries

This section describes the CMEA algorithm and the existing cryptanalysis of CMEA. CMEA is a byte-oriented variable width block cipher with a 64 bit key. Block sizes

may be any number of bytes. CMEA is optimized for 8-bit microprocessors with severe resource limitations.

10.2.1 The CMEA as it is

CMEA has three layers. The first layer performs one non-linear pass on the block, affecting left-to-right diffusion. The second layer is a purely linear, unkeyed operation intended to make changes in the opposite direction. One can think of the second step as xoring the right half of the block onto the left half. The third layer performs a final non-linear pass on the block from left to right. In fact, it is the inverse of the first layer.

CMEA obtains its non-linearity in the first and third layer from an 8-bit keyed lookup table known as the T-Box. The T-Box calculates its 8-bit output as $T(x) = C(((C(((C(((C((x \oplus K_0) + K_1) + x) \oplus K_2) + K_3) + x) \oplus K_4) + K_5) + x) \oplus K_6) + K_7) + x$, x is the input byte and $K_{0,\ldots,7}$ represents the 8 byte key. In this equation C is an unkeyed 8-bit lookup table known as the CaveTable. The operation $\oplus$ represents a bit-wise xor, while $+$ denotes binary addition on the operands. All the operations are 8 bit operations. The algorithm encrypts an n-byte message $P_{0,\ldots,n-1}$ to a ciphertext $C_{0,\ldots,n-1}$ under the key $K_{0\ldots7}$ as follows:

Algorithm 10.1 $y_0 = 0$
$for(i = 0; i < n; i++)$
{
$P'_i = P_i + T(y_i \oplus i)$
$y_{i+1} = y_i + P'_i$
}

$for(i = 0; i < \lfloor n/2 \rfloor; i++)$
$P''_i = P'_i \oplus (P'_{n-i-1} \bigvee 1)$

$z_0 = 0$
$for(i = 0; i < n; i++)$
{
$z_{i+1} = z_i + P''_i$
$C_i = P''_i - T(z_i \oplus i)$
}

Recovering the values of all the 256 T-Box entries is equivalent to the breaking of CMEA even if the keys are not recovered. The values of $T(0)$ occupies a position of special importance. $T(0)$ is always used to obtain C_0 from P_0. Without $T(0)$ one cannot trivially predict where other T-Box entries are likely to be used. Knowing $T(0)$ lets us learn the inputs to the T-Box lookups that modify the second byte in the message. The CAVE Table has very skewed statistical distribution. 92 of the possible 256 eight bit values never appear.

10.2.2 Attacks on CMEA

The attacks against CMEA are briefed next. The attacks [3] can be categorised into two broad types:

A Chosen Plaintext Attack

CMEA is weak against chosen-plaintext attacks; one can recover all the T-Box entries with about 338 chosen texts (on average) and very little work. The attacker does not have control over the block length. The attack has two steps.

1. Recovery of $T(0)$
   For each guess of x, where x is a byte, the message $P = (1-x, 1-x, 1-x, \ldots, 1-x)$ is encrypted, where the sign $-$ denotes binary subtraction. Each byte has the value $(1-x)$. If the result is of the form $C = (-x, \ldots)$ then with very high probability $T(0) = x$. There are only 256-92=164 possible values of $T(0)$, thus the correct value is expected to be guessed using on the average $164/2 = 82$ trials.

2. Recovery of the remaining T-Box entries
   For each byte j, to learn the value of $T(j)$ let $k = ((n-1) \oplus j) - (n-2)$, where the desired blocks are n bytes long. The encryption of $P = (1-T(0), 1-T(0), \ldots, 1-T(0), k-T(0), 0)$ is obtained. If the result is of the form $C = (t-T(0), \ldots)$ then with high probability $T(j) = t$, with a possible ambiguity in the LSB. The second phase requires 256 more chosen plaintexts, thus requiring 338 chosen plaintexts on the whole.

A Known Plaintext attack on 3-byte blocks

Because of the skewed distribution of the CAVE Table $T(0)$ can have 164 possibilities. For each guess at $T(0)$, a 256×256 array of $p_{i,j}$ is constructed which checks whether

$T(i) = j$ is possible for each i, j. All values for $T(i)$, $i > 0$, are initially listed as possible. Since, $T(i) - i$ is a CAVE Table output and the Cave Table has a non-uniform distribution, one can immediately rule out the 92 values for $T(i)$.

Using each known plaintext/ciphertext pair lets us establish implications of the form, $T(0) = t_0, T(i) = j => T(i') = j'$.

If we have eliminated $T(i') = j'$ as impossible, then we can conclude $T(i) = j$ is impossible. In this way $p_{i,j}$ is reduced. One either reaches a conclusion or moves to phase 2.

The second phase recovers the CMEA key from the information previously stored in the $p_{i,j}$ array. The key recovery is based on pruned search. First one guesses K_6 and K_7. Then the effect of the last one-fourth of the T-Box is peeled off and checked whether it is a valid T-Box entry. Because of the skew in the CAVE Table incorrect key guesses are easily identified. The pruned key search is continued by guessing K_4 and K_5. Though the pruned search complexity grows very fast, the T-Box can be subjected to a classic meet-in-the-middle attack. One can work halfway through the T-Box given only $K_{0...3}$, and one can work backwards up to the middle given just $K_{4...7}$ and look for a match. The combination of the pruned search and the meet-in-the-middle attack cryptanalysis recovers the entire CMEA Key with 40-80 known plaintexts.

10.3 Why is CMEA weak?

A detailed study of the CMEA algorithm shows why CMEA is susceptible to chosen plaintext and known plaintext attacks. In this section the properties of the algorithm which make the cipher weak have been identified. The CMEA algorithm is modified to a new algorithm named CMEA-I plaguing the weaknesses of the existing CMEA. The security of CMEA-I has been analyzed in the following section. Recovery of all values of the 256 T-Box entries is equivalent to the breaking of the cipher, so the strength of the T-Box requires special attention and hence has been treated subsequently in details.

- **Property 1** If the plaintext is of the form $P = \{1 - x, 1 - x, \ldots, 1 - x\}$ and the ciphertext is of the form $C = \{-x, \ldots\}$ then with very high probability $T(0) = x$.

  Analysis: $P'_0 = P_0 + T(0) = 1 - x + T(0)$.
  If $T(0) = x$, we have $P'_0 = 1$.

  Thus, $y_1 = y_0 + P'_0 = 0 + 1 = 1$.

Likewise, $P'_1 = P_1 + T(1 \oplus 1) = 1 - x + T(0)$.
If $T(0) = x$, we have $P'_1 = 1$.
Thus, $y_2 = y_1 + P'_1 = 1 + 1 = 2$.
Thus continuing we have $P'_{n-1} = 1$.

So, $P''_0 = P'_0 \oplus (P'_{n-1} \bigvee 1) = 1 \oplus 1 = 0$.

Hence, $C_0 = P''_0 - T(0) = -T(0) = -x$.

The probability when using the CaveTable is dependent on the fact that the initial guess for $T(0)$ is correct and the possible number of trials is thus only (256-92)/2 = 82 on the average.

- **Property 2** If the plaintext is of the form $P = \{1 - T(0), 1 - T(0), \ldots, 1 - T(0), k - T(0), 0\}$ and the ciphertext is $C = \{t - T(0), \ldots\}$ where $k = ((n - 1) \oplus j) - (n - 2)$ then with very high probability $t = T(j)$.

Analysis: It has been shown that $P'_i = 1$ and $y_{i+1} = (i + 1)$, where $0 \leq i \leq (n - 3)$.

Now, $P'_{n-2} = P_{n-2} + T(y_{n-2} \oplus (n - 2))$
$= P_{n-2} + T(0)$, since $y_{n-2} = n - 2$
$= k - T(0) + T(0) = k$.

Using this fact, $y_{n-1} = y_{n-2} + P'_{n-2}$
$= (n - 2) + k = (n - 1) \oplus j$.

Therefore, $P'_{n-1} = P_{n-1} + T(y_{n-1} \oplus (n - 1))$
$= 0 + T(j)$.

Thus, $C_0 = P''_0 - T(0)$
or, $t - T(0) = P'_0 \oplus (P'_{n-1} \bigvee 1) - T(0)$
or, $t = 1 \oplus (T(j) \bigvee 1)$
$= T(j)$,
with a very high probability, with some confusion with the LSB.

- **Property 3** The CMEA algorithm uses a skewed CAVE Table[3]. The CAVE Table is not a permutation and 92 of the possible 256 values does not occur.

- **Property 4** The CMEA algorithm uses a four round T-Box which can be subjected to meet-in-the-middle attack[3].

Using the above properties one can explain why the CMEA algorithm is weak against the chosen plaintext and known plaintext attacks. The causes of the attacks are enlisted as follows:

1. Chosen Plaintext Attack: The CMEA algorithm is weak against chosen plaintext attack because of properties 1 and 2.
2. Known Plaintext Attack: The known plaintext attack is powerful against the CMEA algorithm because of properties 3 and 4.

10.4 Customized Cellular Message Encryption Algorithm : CMEA-I

Analyzing the above properties the CMEA algorithm has been modified. The resultant cipher is presented in this section.

- **Modification 1** Clearly the update equation of P_i needs to be changed so that properties 1 and 2 work no more. The modified equation is of the form:

  $P'_i = P_i + T(y_i \oplus f(i,n))$

  such that as we vary i from 0 to $(n-1)$ (where n is the number of byte blocks in the plaintext) the T-Box is not predictably accessed. In the original CMEA property 1 exists because for a particular nature of the input plaintext and key the T-Box was always referred at the point 0. So, the function $f(i,n)$ should be such that the T-Box is accessed at different points. After considering several forms of the function $f(i,n)$ the proposed function is $f(i,n) = (2i)\%n$, where $\%$ represents the modulo operation.

  Hence the update equation is:

  $P'_i = P_i + T(y_i \oplus ((2i)\%n))$

  Thus the algorithm is transformed into:

  Algorithm 10.2 $y_0 = 0$
  $for(i = 0; i < n; i++)$
  {
  $P'_i = P_i + T(y_i \oplus ((2i)\%n))$
  $y_{i+1} = y_i + P'_i$
  }

  $for(i = 0; i < \lfloor n/2 \rfloor; i++)$
  $P''_i = P'_i \oplus (P'_{n-i-1} \bigvee 1)$

$z_0 = 0$
$for(i = 0; i < n; i++)$
{
$z_{i+1} = z_i + P_i''$
$C_i = P_i'' - T(z_i \oplus ((2i)\%n))$
}

- **Modification 2** The CAVE Table is replaced with the AES S-Box which can be efficiently implemented[200]. Thus the distribution is no more skewed and all the possible 256 values appear as a possibility.
- **Modification 3** The T-Box previously had 4 rounds. The number of rounds of the T-Box has been increased to 8 rounds to prevent meet-in-the-middle attack. The output of the 4 round T-Box is recycled again through the T-Box.

10.5 Security Analysis of CMEA-I

In the present section the security of CMEA-I has been analyzed. The analysis shows that the scheme does not break under a chosen plaintext and known plaintext attack. Avalanche analysis has been performed on CMEA-I. The results show that the scheme provides the necessary diffusion and confusion necessary for a strong cryptographic scheme. The security of the T-Box plays a vital role in the security of the cipher. So, the T-Box has been also analyzed using linear and differential cryptanalysis.

10.5.1 How CMEA-I prevents Chosen-Plaintext and Known-Plaintext attacks?

Due to the modifications incorporated in the cipher the original attack does not work for CMEA-I. For 50,000 variations of the key, plaintexts of the form $(1 - T(0), 1 - T(0), \ldots, 1 - T(0))$ gives ciphertext of the form $(-T(0), \ldots)$ only 0.766% of the time. However we present a modified attack in lines with the original attack and show that the cipher prevents the attack successfully.

Let the P_0 block of the plaintext be $(1 - x_0)$.

Thus $P_0' = P_0 + T(y_0 \oplus 0)$
$= 1 - x_0 + T(0 \oplus 0)$
$= 1 - x_0 + T(0).$

Let $x_0 = T(0)$. So $P'_0 = 1$ and $y_1 = y_0 + P'_0 = 1$.

Similarly, $P'_1 = P_1 + T(1 \oplus 2)$
$= P_1 + T(3)$.

It may be pointed out that the operator $\oplus$ stands for bit-wise xor of the binary representation of its operands. Hence, $1 \oplus 2$ means $\{00000001\} \oplus \{00000010\} = \{00000011\}$, which in decimal means 3.

Hence if we have P_1=$1 - x_1$ and let $x_1 = T(3)$.

So, $P'_1 = 1$ and $y_2 = y_1 + P'_1 = 1 + 1 = 2$.

Likewise, $P'_2 = P_2 + T(y_2 \oplus 4)$
$= P_2 + T(2 \oplus 4)$
$= 1 - x_2 + T(6)$, if $P_2 = 1 - x_2$
$= 1$, using the guess $x_2 = T(6)$.

$y_3 = y_2 + P'_2 = 2 + 1 = 3$.

For the fourth block, $P'_3 = P_3 + T(y_3 \oplus 6)$
$= P_3 + T(3 \oplus 6)$
$= 1 - x_3 + T(5)$, if $P_3 = 1 - x_3$
$= 1$, if $x_3 = T(5)$.

Thus if we have four blocks in the plaintext (without loss of generality) then

$$P''_0 = P'_0 \oplus (P'_3 \bigvee 1) = 0.$$

and hence, $C_0 = 0 - T(0) = -T(0)$.

Thus for 4 input blocks if one obtains chosen plaintexts of the form $P = (1 - T(0), 1 - T(3), 1 - T(6), 1 - T(5))$ then the ciphertext is of the form $C = (-T(0), \ldots)$. Then the number of trials on the average is $(256^4)/2$ which is equivalent to a brute force search on the entire plaintext space and is much larger than that required for original CMEA. (Note that as the CAVE Table has been replaced by the S-Box of Rijndael-AES the number of possible values of each T-Box access is 256).

The following proof shows that the attack is inefficient against CMEA-I.

Proof: During the attack we find that at each stage $y_i = i$ and $f(i, n) = (2i)\%n$, where % refers to the modulo operation. Let CMEA-I break in the face of the attack. For the attack to work the T-Box must be accessed at the same point for at least a single case. In other words there should be repetition in the point at which the T-Box is accessed.

Let us have two instances of i, namely i_1 and i_2 $(i_1 \neq i_2)$, for which the T-Box is accessed at the same point. Thus,

$$i_1 \oplus ((2i_1)\%n) = i_2 \oplus ((2i_2)\%n)$$

or, $(i_1 \oplus i_2) = 2(i_1 \oplus i_2)\%n$

If, $2(i_1 \oplus i_2) < n$, then the equation is possible if $i_1 = i_2$, contradicting our initial assumption.

Also, if $2(i_1 \oplus i_2) = kn + r > n$ (where $k \geq 1$ and $r < n$), we have

or, $(kn + r)/2 = (kn + r)\%n = r$

or, $kn = r$, which is not possible as $r < n$. Thus we arrive at a contradiction, and hence the T-Box is not accessed at the same point. Thus the attack does not work against CMEA-I.

Also the number of chosen plaintexts grows exponentially with the number of blocks. For an n byte block the number of chosen plaintexts is of the order of 256^n. Thus the number of plaintexts to be investigated is equal to that in a brute force search on the entire plaintext space. Such a large number of plaintext requirement makes the attack ineffective against CMEA-I. □

As the CAVE Table has been replaced by the AES S-Box the skewness of the CAVE Table does not exist. Also all the 256 values may appear. The T-Box has been extended to eight rounds and thus a meet-in-the-middle attack does not work. The known plaintext attack against the original CMEA was found to be ineffective against the customized CMEA (CMEA-I).

10.5.2 Diffusion and Confusion in the CMEA-I Algorithm

Diffusion and confusion are two important properties necessary for the security of block ciphers [146]. The current section of the chapter deals with diffusion and confusion in the CMEA-I algorithm. The CMEA-I algorithm has been subjected to Avalanche Attack to test the confusion and diffusion which the cipher provides. A function has a good avalanche effect when a change in one bit of the input results in a change of half of the outputs bits.

Diffusion criteria requires that a change in a single bit of the plaintext should cause a change in several bits in the ciphertext (the key is kept constant). In order to test the diffusion property the CMEA-I algorithm has been subjected on pairs of plaintext which differ by one bit. The number of output bits affected should have a mean of $n/2$ where n is the number of bits of the cipher. In other words it is expected that for a good cipher approximately half of the output bits should be affected. The experiments have been performed on a block size of three-bytes (24 bits). In **Fig. 10.1** the frequency of the number of bits affected has been plotted versus the number of bits affected. The plot shows that around 12 bits are affected for a maximum number of cases. Also the computed average is around 11.98. The plot shows that the algorithm provides sufficient diffusion property.

Confusion criteria requires that a change in a single bit in the key should cause a

change in several bits in the ciphertext (the plaintext is kept constant). In order to test the confusion property the CMEA-I algorithm has been used to encrypt plaintexts with pairs of keys which differ by one bit. The number of output bits affected according to the Avalanche criterion should be around $n/2$ where n is the number of bits of the cipher. The experiments have been performed again on a block size of three-bytes (24 bits). In **Fig. 10.2** the frequency of the number of bits affected has been plotted versus the number of bits affected. The plot shows that around 12 bits are affected for a maximum number of cases. Also the computed average is around 11.91. The plots show that the confusion property is satisfied by CMEA-I.

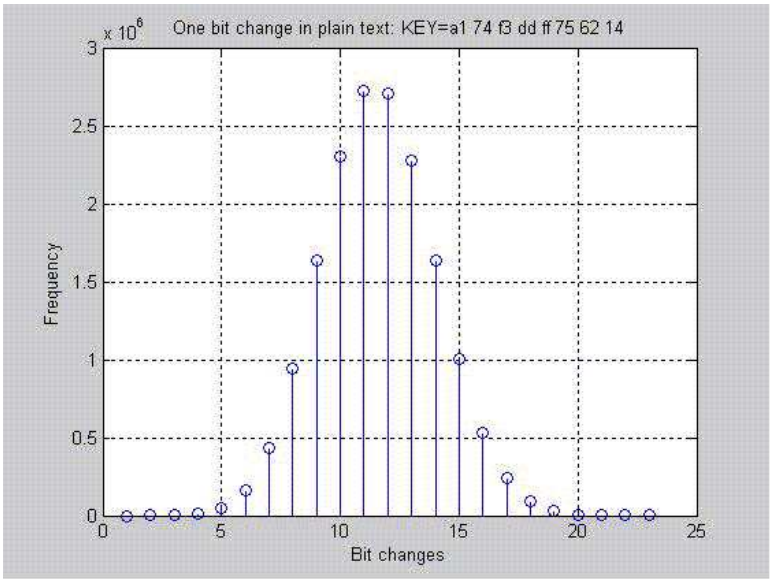


Figure 10.1: **Avalanche Effect to show diffusion**

10.5.3 Finer Issues of Security

The T-Box plays a central role in the cipher structure of CMEA. One can gather information about the T-Box entries from the known CMEA encryptions. Also if the T-Box is compromised and all the T-Box outputs can be identified then the CMEA algorithm is also broken. So, the problem reduces to the cryptanalysis of the T-Box algorithm, given information about the input and output of some of the elements. More formally in this section we shall inspect given the T-Box input and outputs for

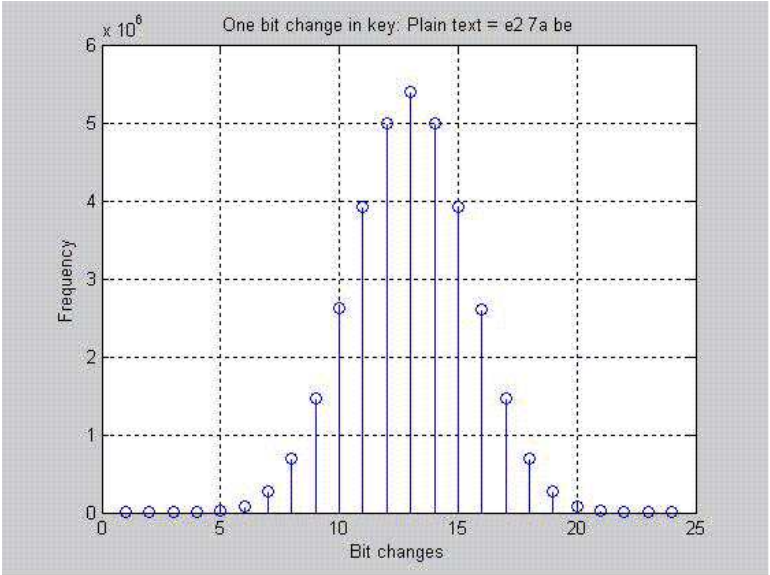


Figure 10.2: **Avalanche Effect to show confusion**

some values is it possible to obtain the other T-Box elements or to recover the key. We analyze the T-Box algorithm under linear and differential attacks [30, 29, 147, 148].

Differential Analysis of the T-Box

Differential Analysis on a block cipher observes that given a certain input difference if a particular output difference occurs with a high probability. In an ideal cipher for an n-bit block the probability should be of the order of $1/2^n$. Differential cryptanalysis seeks to exploit a scenario where a particular output difference δY occurs given a particular input difference δX with a very high probability. The pair $(\delta X, \delta Y)$ is referred to as a differential.

We first observe the security which a single round of the T-Box provides against a differential attack. **Fig. 10.3** shows the single round of T-Box.

Given x_1 and x_5 (the input and output pair at any point) one can calculate x_3. Thus the problem reduces to the cryptanalysis of the portion in the T-Box shown in **Fig. 10.4**. In **Fig. 10.4**, δx_1 and δx_2 are same and does not depend upon the key. Once we know δx_2 and δx_3 and observe the differential property of the addition block to obtain information about K_1 we can also infer information about K_0.

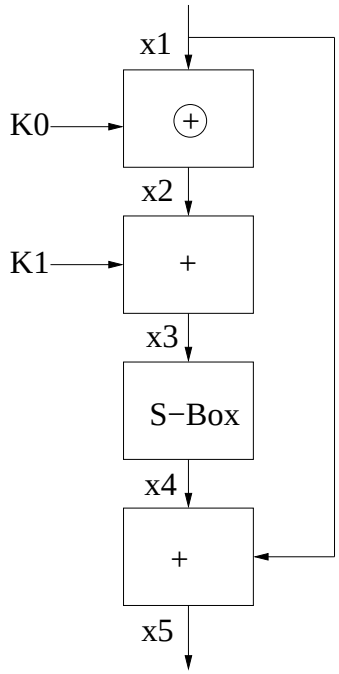


Figure 10.3: **One round T-Box**

From the differentials of the addition block we observe the following two facts:

1. For a fixed $(\delta x_2, \delta x_3)$ can certain keys be ruled out?
2. What is the worst case size of the reduced key space?

Analysis of one round of the T-Box brings the following observations to the surface. We have created tables for the entire key space and noted how many keys are possible for each pair of $(\delta x_2, \delta x_3)$. The tables show that the distribution is very sparse and there are large number of cases where a $(\delta x_2, \delta x_3)$ pair is not possible for any key. There are instances for which certain keys can be immediately ruled out. The remaining set of possible keys varies in size and ranges from as low as 2 to 254 (except the $(0, 0)$ pair where all the keys are possible). Thus in such worst case scenario a random search over only 2 values will reveal K_1 and hence K_0. Hence, one round of the T-Box shows weakness. So, we require to increase the number of rounds of T-Box.

Let us calculate the maximum probability of a differential to pass through one round of the T-Box. It was found that there exists weak keys for each possible δx_1. The weak key is defined to be a key for which there is a δx_3 which always occurs for the particular δx_1 and the key. Next the δx_3 which serves as an input to the S-Box was considered. The corresponding output differential δx_4 with the highest

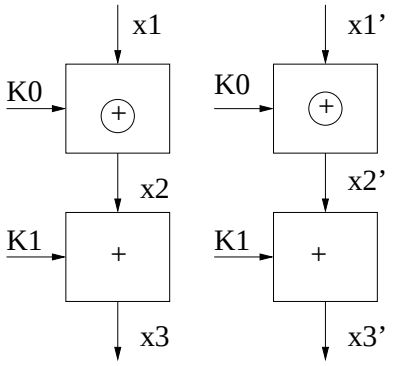


Figure 10.4: **Differential Analysis of T-Box**

probability was observed. Next for all these δx_1's and δx_4's the possible δx_5's were found which had the highest probability.

The above steps were done for all the possible δx_1's and their corresponding weak keys. The analysis results in the worst case maximum probability of obtaining a δx_5 for any given δx_1.

The probability worked to around 0.0078, so for 8 rounds of the T-Box the probability is around 1.37×10^{-17}, which is negligible. If we do not use the weak keys then the worst case probability of the passing of differential reduces to around 0.0039. But this reduces the key space.

Linear Cryptanalysis of the T-Box

The S-Box of AES is known to be resistant against Linear Cryptanalysis(LC). The current subsection works out the security margin which the T-Box provides against LC and shows that the scheme is at least as secured as the AES S-Box.

Linear Cryptanalysis tries to take advantage of high probability occurrences of linear expressions involving plaintext bits, the ciphertext bits and subkey bits. The difference of the probability from the probability 1/2 is known as the bias of the linear equation. Linear Cryptanalysis exploits linear approximations with a large bias. It is a known plaintext attack, that is the attacker does not choose which plaintexts are available. The basic idea is to approximate the operation of a portion of the cipher with a linear expression where the linearity refers to a mod-2 bitwise operation ($\oplus$).

We obtain a linear expression relating the bits of x_1, x_5 and the keys K_0 and K_1, refer **Fig. 10.3**. In order to do so, we follow the usual technique of forming smaller linear equations with large bias and then combine them using Piling-Up lemma to obtain the resultant bias.

The expressions may be derived as follows:

$$\begin{aligned} x_5[0] &= x_4[0] \oplus x_1[0] \text{ with probability 1,} \\ &= f(x_3[i_1], x_3[i_2], \ldots, x_3[i_k]) \oplus x_1[0], \end{aligned}$$

where f is a linear approximation for the S-box with the largest bias ϵ_{RD}.

It may be pointed out that any other expression involving bits of x_1, x_4 and x_5 has a smaller bias, the greatest among them being $\frac{1}{4}$ (**theorem 6.2**).

Using **theorem 6.2**, the biases of linear expressions where $x_3[i_1]$ is expressed in terms of $x_1[i_1], K_0[i_1]$ and $K_1[i_1 - 1]$ is:

$$\begin{aligned} x_3[i_1] &= x_2[i_1] \oplus K_1[i_1] \oplus K_1[i_1 - 1], \text{with bias } 1/4, \\ &= x_1[i_1] \oplus K_0[i_1] \oplus K_1[i_1] \oplus K_1[i_1 - 1], \text{with bias } 1/4. \end{aligned}$$

$$\begin{aligned} \text{Similarly, } x_3[i_2] &= x_1[i_2] \oplus K_0[i_2] \oplus K_1[i_2] \oplus K_1[i_2 - 1], \text{with bias } 1/4. \\ x_3[i_3] &= x_1[i_3] \oplus K_0[i_3] \oplus K_1[i_3] \oplus K_1[i_3 - 1], \text{with bias } 1/4 \\ &\ldots \\ x_3[i_k] &= x_1[i_k] \oplus K_0[i_k] \oplus K_1[i_k] \oplus K_1[i_k - 1], \text{with bias } 1/4 \end{aligned}$$

Combining the biases using Piling-Up lemma we obtain the bias of the complete linear expression:

$$\begin{aligned} x_5[0] = f(&x_1[i_1], x_1[i_2], \ldots, x_1[i_k], K_0[i_1], K_0[i_2], \ldots, K_0[i_k], K_1[i_1], K_1[i_1 - 1], K_1[i_2], \\ &K_1[i_2 - 1], \ldots, K_1[i_k], K_1[i_k - 1]) \oplus x_1[0] \end{aligned}$$

The bias of the linear trail of T-Box of CMEA-I denoted by ϵ_{CMEA-I} is thus:

$$\epsilon_{CMEA-I} = 2^k (\frac{1}{4})^k (\epsilon_{RD}) = \frac{\epsilon_{RD}}{2^k}$$

Thus the security margin provided by the modified T-Box of CMEA-I against Linear Cryptanalysis is thus at least as much as the AES S-Box. It also supports our claim made in **chapter 6** that mixing arithmetic operations, like addition with binary operations, like xor helps us to build ciphers which are more resistant against cryptanalytic attacks in general and linear attacks in particular.

10.6 Efficiency of CMEA-I

In this section we compare the efficiency of the design with respect to the original CMEA algorithm, which is known to be suited for the telecommunication industry. The CMEA algorithm is optimized for 8 bit micro-processors with severe resource limitations[3]. The CMEA algorithm has been modified in the following three places in order to prevent successful cryptanalysis of CMEA-I.

- The update equation of P_i has been changed to $P_i'=P_i + T(y_i \oplus ((2i)\%n))$.
- The CAVE Table is replaced with the AES S-Box.
- The number of rounds of T-Box has been increased to eight rounds.

The first change is a minor functional change and does not require any extra computation with respect to the CMEA algorithm. The multiplication by 2 is a simple shift left operation and hence has no negative effect on the efficiency of the original CMEA algorithm.

The second change advocates the replacement of the CAVE Table with the AES S-Box. The AES S-Box, unlike the DES S-Box and the CAVE Tables can be implemented through a compact algebraic equation [184]. The structured algorithm of AES S-Box makes it amenable to efficient implementations both in hardware and software [168, 165, 200, 167]. Still the S-Box of Rijndael is secured as it has withstood lot of cryptanalysis [184].

The third modification of CMEA-I is the increase of the number of rounds in T-Box from four to eight. Although this increases the computation slightly but is compensated by the fact that replacement of CAVE Table by AES S-Box can lead to extremely fast designs [166]. Further it may be added that like CMEA, CMEA-I is also a self-invertible encryption algorithm and hence conducive for implementations.

10.7 Conclusion

In the present work the original CMEA algorithm has been modified into CMEA-I. The chapter shows how the existing cryptanalysis of CMEA fails to break CMEA-I. It has been shown that the T-Box provides sufficient security margin to the cipher CMEA-I in the face of linear and differential cryptanalysis. In short, the work demonstrates that with suitable modifications the original CMEA algorithm can be made strong and hence can be suitable for wireless security.

Chapter 11

Concluding Remarks and Open Problems

The dissertation searches for cryptographic primitives, which are efficient to implement in hardware and also exhibit cryptographic strength. Finally, the primitives have been used to compose various ciphers and protocols. *Cellular Automata (CA)* have been employed extensively to realize the goals.

- The work shows both theoretically and through practical experiments that a special class of CA, the complemented CA possess some remarkable properties which may be used to develop block ciphers and vary the session key of a key agreement protocol.
- The work shows that CA may be employed to develop cryptographically robust S-Boxes. The thesis presents two such schemes: The first one is an informal approach, but extremely efficient method. It shows how repetition of simple rules may develop good S-Boxes. The second technique provides a more formal method to construct S-Boxes. Both theoretically and experimentally it has been confirmed that the S-Boxes made out of CA are cryptographically strong, efficient to implement and also programmable.
- The work also investigates the effect of key mixing using integer addition on linear cryptanalysis of block ciphers. It has been shown, theoretically and through experimentations using toy ciphers, that performing such a key mixing indeed improves the security margin of the cipher. Finally, a CA based MDS mapping has been designed which is also self-invertible. The CA based crypto-primitives and the methods developed have been finally integrated to form a new block cipher named SPAMRC. Theoretically the security margin of 4 rounds of the

cipher against Differential and Linear Cryptanalysis have been estimated. Results show that the cipher provides formidable resistance against two of the most powerful cryptanalytic techniques.

- The work develops a CA based technique to generate expander graphs using polynomial space. The graph has been used to develop an expanding collection which can be employed to realize a one-way function. Experimental results show that such a one-way function is extremely conducive for VLSI implementations and saves the area-delay product compared to conventional one-way functions. Finally the one-way function developed have been used to develop a key-agreement protocol with authentication, key confirmation and key freshness. The work assures the security of the protocol in the Bellare-Rogaway model.

- Next, the work analyzes two standard cryptographic algorithms: AES and CMEA. The work concludes, both theoretically and through simulations that AES when subjected to faults can be vulnerable. The work proposes the strongest attacks of its kind in literature to the best of our knowledge. Finally it has been shown that although the original CMEA algorithm has been broken by cryptanalysis, proper modifications in the algorithm can lead to a stronger version which can prevent the existing attacks on CMEA.

11.1 Open Problems and Future Scope of Work

Following are some interesting extensions of our current work which can be carried out in the future.

- In the construction of CASBox, it has been analyzed that the resulting S-Boxes are strong against Differential and Linear Cryptanalysis, have high non-linearity and algebraic degree and satisfies the properties of SAC and balancedness.

  A natural extension of the present work would be to consider the following properties of the S-Box:

  1. **Algebraic immunity:** Although the CASBox lacks any straight forward algebraic relation and the algebraic degree is high, one may further obtain boolean equations (Algebraic Normal Forms) of the component bits and compare it against a fixed S-Box.

     In these lines, we may interpret the CASBox as follows. The CASBox operates on the input $z = (y, x)$ and results in an output $Q(z) = (Q_2(z), Q_1(z))$.

The construction has been discussed in details in **chapter 5**. The construction may be analyzed in an equivalent manner.

The output $Q_1(z)$ is derived from the input x, by operating a maximum length CA, characterized by the matrix T. The number of clock cycles are the output of a function r, operating on the input y. Thus, the number of clock cycles to be applied is $r(y)$ and the output $Q_1(y, x) = T^{r(y)}(x)$. It might be interesting to further analyze the function in the light of algebraic analysis.

2. **Differential Power Analysis:** Prouff had introduced the notion of *Transparency Order* (T_p)[51]. The property deals with the ability of S-Boxes to thwart single-bit or multi-bit DPA attacks. If this parameter is sufficiently small, then the S-Box is able to withstand DPA attacks without modifications in implementation. The modifications are not desirable as they make the cipher about twice slower. In [201] a lower bound of T_p was calculated for various functions including the AES S-Box. It was shown that the AES S-Box has a high transparency order (≥ 7.8) and quite close to the worst case transparency order (8) for an 8-bit S-Box. In order to compute the resistance of CASBox against DPA it is an open problem to compute its transparency order.

- The proof of **theorem 6.2** shows that if the carry term is bent then the bias of any linear approximation is $2^{-(i+1)}$ and hence falls exponentially fast with the bit position. Such a key mixing would improve greatly the resistance of ciphers against Linear and possibly even Differential attacks. Hence, it would be interesting to define an ideal composition (denoted by $\circ$) between the key K and the data X:

$$Y = X \circ K$$

It is intended that the carry term propagating from bit i to bit $(i+1)$, denoted by c_i is a bent function of the variables $(x_i, y_i, x_{i-1}, y_{i-1}, \ldots, x_0, y_0)$. In other words, we require a recursive method to generate a bent function, c_i of $2(i+1)$ variables from x_i, y_i and also the previous carry term, c_{i-1} which is a bent function in $2i$ variables. It would be interesting to search whether such a key mixing is possible.

- Security margin of SPAMRC against Linear and Differential Cryptanalysis have been computed. However the key schedule for SPAMRC needs to be developed. Finally, the block cipher needs to be evaluated against more sophisticated attacks like Truncated Differential Cryptanalysis, Impossible Differential Cryptanalysis, Higher Order Differential Cryptanalysis, Boomerang, Related Key,

Interpolation and Slide Attacks.

- Protecting against a fault based side-channel attack by duplicating the hardware is a trivial solution and incurs a lot of wastage of hardware resources. In order to save on hardware, it may be inferred from the fault attack presented in the work and those compared with, that only the last three rounds of AES are required to be fault tolerant. It would be interesting to develop a complete AES hardware which can prevent fault attacks and yet impose minimal penalty on the performance of the device.

Bibliography

[1] D. E. Knuth, *The Art of Computer Programming — Seminumerical Algorithms*, Addison-Wesley, 1981.

[2] TIA Telecommunications Industry Association, "Common Cryptographic Algorithms, Revision D.1, Publication Version, September 13, 2003," http://ftp.tiaonline.org/TR-45/TR45AHAG/Public/ComCryptAlgD1.pdf.

[3] D. Wagner, B. Schneier and J. Kelsey, "Cryptanalysis of the Cellular Message Encryption Algorithm," in *Crypto 1997*, 2002, Also a NESSIE report, pp. 526–537.

[4] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, John Wiley & Sons, 2001.

[5] O. Goldreich, *Foundations of Cryptography*, vol. 2, Cambridge University Press, 2005.

[6] J. Seberry and J. Pieprzyk, *An Introduction to Computer Security*, Advances in Computer Science Series, 1988.

[7] W. F. Friedman, "The index of coincidence and its application in cryptography," in *Riverbank Publication, Riverbank Labs.* 1920, Reprinted by Aegian Park Press.

[8] C. E. Shannon, "Communication theory of secrecy systems, vol 28, no 4," in *Bell System Technical Journal.* 1949, pp. 656–715, Bell.

[9] D. Kahn, "The codebreakers: The story of secret writing," 1967, New York: Macmillan Publishing Co.

[10] H. Feistel, "Cryptography and Computer Privacy," *Scientific American*, vol. 228, no. 5, pp. 15–23, May 1973.

[11] W. Diffie and M. Hellman, "New Directions in Cryptography," in *IEEE Transactions on Information Theory (22)*. 1976, pp. 644–654, IEEE.

[12] R. Rivest, A. Shamir and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM, Previously released as an MIT "Technical Memo" in April 1977*, vol. 21, no. 2, pp. 120–126, 1978.

[13] M. E. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Proceedings*, , no. 4, pp. 401–406, 1980.

[14] R.L. Rivest B.S. Kaliski and A.T. Sherman, "Is the data encryption standard a group?," *Journal of Cryptology*, pp. 1–36, 1988.

[15] D. Chaum and J.H. Evertse, "Cryptanalysis of DES With a Reduced Number of Rounds," in *Proceedings of Crypto, 1985*. 1986, pp. 192–211, Springer Verlag.

[16] Y. Desmedt, J.J. Quisquater and M. Davio, "Dependence of the output on the input in DES: Small avalanche characteristics," in *Proceedings of Eurocrypt*. 1984, pp. 359–376, Springer Verlag.

[17] D. Andleman and J. Reeds, "On the cryptanalysis of rotor and substitution-permutation networks," *IEEE Transactions on Information Theory*, vol. 28, no. 4, pp. 578–584, 1982.

[18] I. S. Bichl, "Cryptanalysis of the Data Encryption Standard by method of formal coding," in *Proceedings of the workshop on Cryptology*. 1982, pp. 235–255, Springer Verlag.

[19] E. Biham, "New type of cryptanalytic attacks using related keys," in *Proceedings of Eurocrypt*. 1994, pp. 398–409, Springer Verlag.

[20] E. Biham and N. Keller, "Cryptanalysis of reduced variants of rijndael," http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html, 2000.

[21] S. Lucks, "Attacking Seven Rounds of Rijndael under 192-Bit and 256-Bit Keys," in *Proceeding of the Third Advanced Encryption Standard Candidate Conference, NIST*, 2000, pp. 215–229.

[22] L. R. Knudsen and D. Wagner, "Integral Cryptanalysis," in *Proceeding of the 9th International Workshop on Fast Software Encryption, LNCS 2365*, 2002, pp. 112–127.

[23] J. H. Cheon, M. Kim etal., "Improved Impossible Differential Cryptanalysis of Rijndael and Crypton," *Information Security and Cryptology-ICISC 2001*, vol. 2288 in Lecture Notes in Computer Science, pp. 39–49, 2001.

[24] N. T. Curtois and J. Piepryzk, "Cryptanalysis of block ciphers with overdefined systems of equations," in *Asiacrypt 2002*. 2002, Springer Verlag.

[25] D. CopperSmith, "XSL against Rijndael," in *Crypto-gram, 2002*, October 2002.

[26] D. CopperSmith, "Impact of Curtois and Piepryzk results," in *NIST AES Discussion Forum, 2002*, September 2002.

[27] N. T. Curtois, "The Inverse S-Box, Non-linear Polynomial Relations and Cryptanalysis of Block ciphers," in *AES 2004*, LNCS 3373, 2005, pp. 170–188, Springer Verlag.

[28] N. Courtois, A. Klimov, J. Patarin and A. Shamir, "Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations," in *Proceeding of Eurocrypt, LNCS 1807*, 2000, pp. 392–407.

[29] E. Biham and A. Shamir, "Differential Cryptanalysis of DES like CryptoSystems," *Journal of Cryptology*, vol. 4, pp. 3–72, 1991.

[30] Mitsuru Matsui, "Linear Cryptanalysis method for DES cipher," in *Advances in Cryptology-Eurocrypt 1993*. 1993, pp. 386–397, Springer, volume 765 of LNCS.

[31] S. K. Langford, M. E. Hellman, "Differential-linear Cryptanalysis," in *Proceeding of Crypto, LNCS 839*, 1994, pp. 17–25.

[32] E. Biham, A. Biryukov, A. Shamir, "Miss in the Middle Attacks on IDEA and Khufu," in *Proceeding of Fast Software Encryption, LNCS 1636*, 1999, pp. 124–138.

[33] E. Biham, A. Biryukov, A. Shamir, "Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials," in *Proceeding of Eurocrypt, LNCS 1592*, 1999, pp. 12–23.

[34] D. Wagner, "The Boomerang Attack," in *Proceeding of Fast Software Encryption, LNCS 1636*, 1999, pp. 156–170.

[35] N. T. Curtois, "Feistel Schemes and Bi-linear Cryptanalysis," in *Proceeding of Crypto, LNCS 3152*, 2004, pp. 23–40.

[36] E. Biham, O. Dunkelman and N. Keller, "New Combined Attacks on Block Ciphers," in *Proceeding of FSE, LNCS 3557*, 2005, pp. 126–144.

[37] P. C. Kocher, "Timing Attacks on Implementation of Diffie-Hellman, RSA, DSS and Other Systems," in *Proceeding of Crypto, LNCS 1109*, 1996, pp. 104–113.

[38] O. Acii, W. Schindler and C. K. Koc, "Improving Brumley and Boneh timing attack on unprotected SSL implementations," in *Proceedings of the 12th ACM conference on Computer and communications security, CCS '05*, New York, NY, USA, 2005, pp. 139–146, ACM Press.

[39] P. C. Kocher, J. Jaffe and B. Jun, "Differential Power Analysis," in *Proceeding of Crypto, LNCS 1666*, 1999, pp. 388–397.

[40] L. Goubin and J. Patarin, "DES and Differential Power Analysis - The "Duplication" Method," in *Cryptographic Hardware and Embedded Systems - CHES*. 1999, Springer-Verlag.

[41] A. Hevia and M. Kiwi, "Strength of Two Data Encryption Standard Implementations under Timing Attacks," vol. 2, pp. 416–437, 1999.

[42] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Side Channel Cryptanalysis of Product Ciphers," vol. 8, pp. 141–158, 2000.

[43] P. N. Fahn and P. K. Pearson, "IPA: A New Class of Power Attacks," in *Cryptographic Hardware and Embedded Systems-CHES, LNCS , 1717*. 1999, pp. 173–186, Springer-Verlag.

[44] M. L. Akkar and R. Bevan and P. Dischamp and D. Moyart, "Power Analysis, What is Now Possible...," in *Proceedings of ASIACRYPT 2000, Springer-Verlag, LNCS 1976*, 2000, pp. 489–502.

[45] S. M. Yen, "Amplified Differential Power Cryptanalysis on Rijndael Implementations with Exponentially Fewer Power Traces," in *Information Security and Privacy, Proceedings of ACISP 2003, Wollongong, Australia, LNCS 2727*, 2003, pp. 106–117.

[46] S. B. Ors and E. Oswald and B. Preneel, "Power-Analysis Attacks on an FPGA - First Experimental Results," in *Proceedings of Cryptographic Hardware and Embedded Systems - CHES 2003, LNCS 2779*, 2003, pp. 35–50.

[47] Y. Sakai and K. Sakurai, "A New Attack with Side Channel Leakage During Exponent Recoding Computations," in *Proceedings of CHES 2004, Springer-Verlag, LNCS 3156*, 2004, pp. 298–311.

[48] R. Novak, "SPA-Based Adaptive Chosen-Ciphertext Attack on RSA Implementation," in *Proceedings of PKC 2002, LNCS 2274*, 2002, pp. 252–262.

[49] W. Schindler, "A Combined Timing and Power Attack," in *Proceedings of PKC 2002, LNCS 2274*, 2002, pp. 263–279.

[50] K. Schramm and G. Leander and P. Felke and C. Paar, "A Collision-Attack on AES Combining Side Channel- and Differential-Attack," in *Proceedings of CHES 2004, Springer-Verlag, LNCS 3156*, 2004, pp. 163–175.

[51] E. Prouff, "DPA Attacks and S-Boxes," in *Proceedings of FSE 2005, Springer LNCS 3557*, 2005, pp. 424–441.

[52] S. Mangard and N. Pramstaller and E. Oswald, "Successfully Attacking Masked AES Hardware Implementations," in *Proceedings of CHES 2005, Springer LNCS 3659*, 2005, pp. 157–171.

[53] S. Guilley and P. Hoogvorst and Y. Mathieu and R. Pacalet, "The "Backend Duplication" Method," in *Proceedings of CHES 2005, Springer LNCS 3659*, 2005, pp. 383–397.

[54] T. Popp and S. Mangard, "Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints," in *Proceedings of CHES 2005, Springer LNCS 3659*, 2005, pp. 172–186.

[55] J. D. Goli and C. Tymen, "Multiplicative Masking and Power Analysis of AES," in *Proceedings of CHES 2002, Springer 2003, LNCS, 2523*, 2002, pp. 198–212.

[56] E. Trichina and D. De Seta and L. Germani, "Simplified Adaptive Multiplicative Masking for AES," in *Proceedings of CHES 2002, Springer 2003, LNCS, 2523*, 2002, pp. 187–197.

[57] J. Dj. Golic, "Techniques for random masking in hardware," Cryptology ePrint Archive, 2005.

[58] D. Boneh, R.A. DeMillo and R.J. Lipton, "On the Importance of checking cryptographic Protocols for Faults," in *Eurocrypt 1997*. 1997, LNCS 1233.

[59] E. Biham and A. Shamir, "Differential Fault Analysis of Secret Key Cryptosystems," in *Advances in Cryptology, Crypto 1997*. 1997, LNCS 1294.

[60] O. Kommerling and M. G. Kuhn, "Design Principles for Tamper-Resistant Smartcard Processors," in *Proceedings of the USENIX Workshop on Smartcard Technology*, 1999, pp. 9–20.

[61] S. Skorobogatov and R. Anderson, "Optical Fault Induction Attacks," in *CHES 2002*. 2002, LNCS 2523.

[62] J. J. Quisquater and D. Samyde, "Eddy Currents for Magnetic Analysis with Active Sensors," in *Proceedings of Esmart 2002, Nice, France*, 2002.

[63] C. Giraud, "DFA on AES," Cryptology ePrint Archive, Report 2003/008, 2003.

[64] J. Blomer and J. P. Seifert, "Fault Based Cryptanalysis of the Advanced Encryption Standard (AES)," in *FC 2003*. 2003, pp. 162–181, LNCS 2742.

[65] P. Dusart, G. Letourneux and O. Vivolo, "Differential Fault Analysis on A.E.S.," http://eprint.iacr.org/2003/010.

[66] G. Piret and J. J. Quisquater, "A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad," in *CHES 2003*. 2003, pp. 77–88, LNCS 2779.

[67] J. Kam and G. Davida, "Structured design of substitution-permutation encryption networks," *IEEE Transactions on Computers*, , no. 10, pp. 747–753, 1979.

[68] A. Webster and S. Tavares, "On the design of s-boxes," in *Advances in Cryptology-Crypto*, 1985, pp. 523–534.

[69] J. Pieprzyk and G. Finkelstein, "Towards effective non-linear cryptosystem design," *IEE Proceedings*, vol. E-135, no. 6, pp. 325–335, 1988.

[70] A. Youssef and S. Tavares, "Resistance of balanced s-boxes to linear and differential cryptanalysis," *Information Processing Letters*, vol. 56, pp. 249–252, 1995.

[71] A. Youssef and S. Tavares, "Number of non-linear regular s-boxes," *IEE Electronic Letters*, vol. 31, no. 19, pp. 1643–1644, 1995.

[72] J. Seberry, X. M. Zhang and Y. Zheng, "Systematic Generation of Cryptographically Robust S-boxes," 1^{st} Conference Computer and Communication Security, VA, USA, 1993, pp. 171–182.

[73] K. Nyberg, "Perfect non-linear s-boxes," in *Advances in Cryptology-Eurocrypt*, 1991, pp. 378–386.

[74] C. Carlet, *Boolean Methods and Models*, Cambridge University Press, 2005.

[75] J. Pieprzyk, T. Hardjono and J. Seberry, *Fundamentals of Computer Security*, chapter 3, pp. 144–167, Springer, 2002.

[76] C. Carlet and E. Prouff, "Vectorial Functions and Covering Sequences," in *Finite Fields and Applications, Fq7, Springer LNCS 3006*, 2004, pp. 215–248.

[77] T. Johansson and E. Pasalic, "A Construction of Resilient Functions with high Non-linearity," in *Proceedings of IEEE International Symposium on Information Theory*, 2000.

[78] T. Satoh K.Kurosawa and K. Yamamoto, "Highly Nonlinear t-resilent Functions," *Journal of Universal Computer Science*, vol. 3, no. 6, pp. 721–729, 1997.

[79] S. Maitra and E. Pasalic, "Linear Codes in Generalised Construction of Resilient Functions with Very High Nonlinearity," in *Proceedings of SAC 2001, Springer LNCS 2259*, 2001, pp. 60–74.

[80] K. Nyberg, "On the Construction of Highly Nonlinear Permutations," in *Advances in Cryptology-Eurocrypt, Springer LNCS 658*, 1992, pp. 92–98.

[81] C. Adams and S. Tavares, "Good S-Boxes are Easy to Find," Advances in Cryptology, Crypto, 1989, pp. 612–615.

[82] S. Mister and C. Adams, "Practical S-Box Design," Workshop on Selected Areas in Cryptography (SAC '96) Workshop Record, Queens University , 1996, pp. 61–76.

[83] X. Yi, S. X. Cheng, X. H. You and K. Y. Lam, "A method for obtaining cryptographically strong 88 S-boxes," Global Telecommunications Conference,GLOBECOM '97, IEEE, Nov 1997, pp. 689–693.

[84] K. C. Gupta and P. Sarkar, "Improved Construction of Non-linear Resilent S-Boxes," in *In Advances in Cryptology-Asiacrypt*. 2002, pp. 466–483, Springer Verlag.

[85] K. C. Gupta and P. Sarkar, "Efficient Representation and Software Implementation of Resilient Maiorana-McFarland and S-Boxes," in *In WISA 2004*. 2004, pp. 317–331, LNCS 3325, Springer Verlag.

[86] N. T. Curtois and W. Meier, "Algebraic Attacks on Stream Ciphers with Linear Feedback," in *Eurocrypt 2003*, LNCS 2656, 2003, pp. 345–359, Springer Verlag.

[87] F. Armknecht and M. Krause, "Algebraic Attacks on Combiners with Memory," in *Crypto 2003*, LNCS 2729, 2003, pp. 162–176, Springer Verlag.

[88] H.M. Heys and S.E. Tavares, "The design of substitution-permutation networks resistant to differential and linear cryptanalysis," in *In the Proceedings of* 2^{nd} *ACM Conference on Computer and Communications Security, Fairfax. Virginia*, 1994, pp. 148–155.

[89] H.M. Heys and S.E. Tavares, "The design of product ciphers resistant to differential and linear cryptanalysis," *Journal of Cryptology*, vol. 9, no. 1, pp. 1–19, 1996.

[90] H.M. Heys and S.E. Tavares, "Avalanche characteristics of substitution-permutation networks," *IEEE Transactions on Computer*, vol. 44, pp. 1131–1139, Sept. 1995.

[91] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error Correcting Codes*, North-Holland Publishing Company, 1977.

[92] S. Vaudenay, "On the need for multipermutations: Cryptanalysis of MD4 and SAFER," Proceedings of Fast Software Encryption (FSE), Springer, LNCS 1008, 1995, pp. 286–297, Springer Verlag.

[93] J. Daemen, V. Rijmen, B. Preneel, A. Bosselaers, "The Cipher SHARK," Proceedings of Fast Software Encryption (FSE), Springer, LNCS 1039, 1996, pp. 99–112, Springer Verlag.

[94] J. Daemen, L. Knudsen and V. Rijmen, "The Block Cipher Square.," 1997, pp. 149–165, proceedings of FSE'97, lecture notes in Computer science 1267.

[95] A. Youssef and S. Mister and S. Tavares, "On the design of linear transformations for substitution permutation encryption networks," Workshop on Selected Areas of Cryptography (SAC '96): Workshop Record, 1997, pp. 40–48.

[96] P. Junod and S. Vaudenay, "Perfect Diffusion Primitives for Block Ciphers-Building Efficient MDS Matrices," in *Selected Areas in Cryptography*. 2004, Springer Verlag.

[97] P. S.L.M. Barreto and V. Rijmen, "The anubis block cipher," http://paginas.terra.com.br/informatica/paulobarreto/AnubisPage.html.

[98] C. Boyd and A. Mathuria, *Protocols for Authentication and Key Establishment*, chapter 5, pp. 138–199, Springer, 2003.

[99] R. Needham and M. D. Schroeder, "Using Encryption for Authentication in large Networks of Computers," in *Communications of ACM, 21(2)*, 1978, pp. 993–999.

[100] B. C. Neuman and Theodore Ts'o, "Kerberos: An Authentication Service for Computer Networks," in *IEEE Communications Magazine, 32(9)*, 1994, pp. 33–38.

[101] D. Otway and O. Rees, "Efficient and Timely Mutual Authentication," in *ACM Operating Systems Review, 21(1)*, 1987, pp. 8–10.

[102] P. Janson and G. Tsudik, "Secure and Minimal protocols for Authenticated Key Distribution," *"Computer Communications"*, vol. 18, pp. 645–653, 1995.

[103] C. Boyd, "Towards a classification of key agreement protocols," in 8^{th} *IEEE Computer Security Foundations Workshop*, 1995, pp. 38–43.

[104] C. Boyd, "Security Architectures using Formal Models," *"IEEE journal on Selected Areas in Communication"*, vol. 11(5), pp. 694–701, 1993.

[105] W. Stallings, *Cryptography and Network Security*, Prentice Hall, 2003.

[106] A. Menezes, P. Van. Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, chapter 12, pp. 489–541, CRC Press, 1996.

[107] ISO, "Information technology-security techniques - key management-part 3: Mechanisms using assymetric techniques iso/iec 11770-3," International Standard, 1999.

[108] S. B. Wilson, D. Johnson and A. Menezes, "Key Agreement Protocols and their Security Analysis," in *Sixth IMA International Conference on Cryptography and Coding*, 1997, pp. 110–125.

[109] H. Krawczyk, "SKEME: A Versatile Secure Key Exchange for Internet," in *Proceedings of Symposium on Network and Distributed System Security, SNDSS*, Washington DC, USA, 1996, p. 114, IEEE Computer Society.

[110] M. Abadi and R. Needham, "Prudent Engineering Practice for Cryptographic Protocols," in *Proceedings of IEEE Symposium on Research in Security and Privacy 1994*, 1998, pp. 122–136.

[111] M. Bellare and P. Rogaway, "Random Oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of Annual Conference on Computer and Communications Security, ACM 1993*, 1993, pp. 62–73.

[112] M. Bellare and P. Rogaway, "Entity Authentication and Key Distribution," in *Advances in Cryptology-Crypto 1993*. 1993, pp. 110–125, Lecture Notes in Computer Science, Vol. 773, Springer Verlag.

[113] M. Bellare, R. Canetti and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols," in *Proceedings of ACM Symposium on the Theory of Computing-STOC 1998*, 1998, pp. 419–428.

[114] R. Canetti and H. Krawczyk, "Analysis of Key Exchange Protocols and their use for building secure Channels," in *Proceedings of Eurocrypt, 2001*, 2001, pp. 453–474.

[115] O. Goldreich, "Candidate One-Way Functions Based on Expander Graphs," Cryptology ePrint Archive, Report 2000/063, 2000.

[116] R. C. Merkle and M. E. Hellman, "Hiding Information and Signatures in Trapdoor Knapsacks," *"IEEE Transactions on Information Theory"*, vol. 24, pp. 525–530, 1978.

[117] A. Shamir, "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," *"IEEE Transactions on Information Theory"*, vol. 30, pp. 699–704, 1984.

[118] E.F. Brickell, "Breaking Iterated Knapsacks," in *Proceedings of Crypto, Springer*, 1984, pp. 342–358.

[119] G. Brassard, "A note on the Complexity of Cryptography," *"IEEE Transactions on Information Theory"*, vol. IT-25, pp. 232–233, 1979.

[120] A. Klimov, *Applications of T-functions in Cryptography*, Ph.D. thesis, The Weizmann Institute of Science, 2005.

[121] A. Klimov and A. Shamir, "A New Class of Invertible Mappings," in *Proceedings of CHES, Springer LNCS 2523*, 2003, pp. 470–483.

[122] A. Klimov and A. Shamir, "Cryptographic Applications of T-functions," in *Proceedings of SAC, Springer LNCS 3006*, 2004, pp. 248–261.

[123] A. Klimov and A. Shamir, "New Cryptographic Primitives Based on Multiword T-functions," in *Proceedings of FSE, Springer LNCS 3017*, 2004, pp. 1–15.

[124] R. S. Winternitz, "A Secure One-way Hash Function Built from DES," in *IEEE Symposium on Security and Privacy*, 1984.

[125] N. Alon, "Eigen Values, Geometric Expanders, Sorting in Rounds and Ramsey Theorem," *"Combinatorica"*, vol. 6, pp. 207–219, 1986.

[126] P. Sarnak A. Lubotzky, R. Phillips, "Ramanujan Graphs," *"Combinatorica"*, vol. 8, pp. 261–277, 1988.

[127] O. Gaber and Z. Galil, "Explicit Construction of Linear Size Superconcentrators," *"JCSS"*, vol. 22, pp. 407–420, 1981.

[128] S.K. Panjwani, "An Experimental Evaluation of Goldreich's One-Way Function," Cryptology ePrint Archive, Report 2000/063, 2001.

[129] P. Pal Chaudhuri, D.Roy Chowdhury, S. Nandi, and S. Chattopadhyay, *Additive Cellular Automata Theory and its Application*, vol. 1, IEEE Computer Society Press, 1997.

[130] Stephen Wolfram, *A New Kind of Science*, Wolfram Media Inc., 2004.

[131] F. R. Gantmatcher, *The Theory of Matrices, Vol I*, Chelsea Publishing Co., NY., 1959.

[132] B. Elspas, "The theory of autonomous linear sequential networks," *TRE Trans. on Circuits*, vol. CT-6, no. 1, pp. 45–60, March 1959.

[133] F. R. Gantmatcher, *The Theory of Matrices, Vol II*, Chelsea Publishing Co., NY., 1959.

[134] O Martin, A.M. Odlyzko and S. Wolfram, "Algebraic Properties of Cellular Automata," *Commun. Math. Phys.*, pp. 219–258, 1984.

[135] S. Wolfram, "Statistical mechanics of cellular automata," *Rev. Mod. Phys.*, vol. 55, no. 3, pp. 601–644, July 1983.

[136] A.K. Das, A. Ganguly, A. Dasgupta, S. Bhawmik and P.P Chaudhuri, "Efficient Characterization of Cellular Automata," *IEE Proceedings*, pp. 81–87, January 1990.

[137] D. Mukhopadhyay and D. RoyChowdhury, "Cellular automata: An ideal candidate for a block cipher," in *In the Proceedings of First International Conference on Distributed Computing and Internet Technology ICDCIT 2004, LNCS 3347*, December 2004.

[138] P. Joshi, D. Mukhopadhyay and D. Roy Chowdhury, "Design and Analysis of a Robust and Efficient Block Cipher Using Cellular Automata," in *Proceedings of* 20^{th} *International Conference on Advanced Information Networking and Applications, AINA*, 2006, vol. 2, pp. 67–71.

[139] S. W. Golomb, *Shift Register Sequences*, Holden Day, 1967.

[140] P. Guan, "Cellular automata public key cryptosystem," *Complex Systems*, vol. 1, pp. 51–57, 1987.

[141] B. Kar S. Nandi and P. Pal.Chaudhury, "Theory and applications of cellular automata in cryptography," *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1346–1357, Dec. 1994.

[142] D. Mukhopadhyay and D. Roy Chowdhury, "Cellular automata based cryptosystem employing galois field algebra," Yokohama, Japan, 2001, International Symposium on Cellular Automata.

[143] S. Sen, C. Shaw, D. R. Chowdhury, N. Ganguly, and P. Pal Chowdhury, "Cellular automata based cryptosystem (cac)," in 4^{th} *International Conference on Informations and Computer Security (ICICS 2002)*. Dec 2002, pp. 303–314, LNCS 2513.

[144] S. Wolfram, "Cryptography with cellular automata," 1986, pp. 429–432, Advances in Cryptology: Crypto '85 Proceedings, LNCS 218.

[145] P. Sarkar, "Hiji-bij-bij: A new stream cipher with a self-synchronizing mode of operation," 2003, pp. 36–51, INDOCRYPT.

[146] C. E. Shannon, "A mathematical theory of communication," in *Bell System Technical Journal*, http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html, July and October 1948, pp. 379–423 and 623–656, Bell.

[147] D. R. Stinson, *Cryptography : Theory and Practice*, chapter 3, pp. 79–88, 2002.

[148] H. M. Heys, "A Tutorial on Linear and Differential Cryptanalysis," www.engr.mun.ca/ howard/PAPERS/ldc_tutorial.ps.

[149] J. A. Gordon and H. Retkin, "Are big s-boxes best?," in *EUROCRYPT*, 1982, pp. 257–262.

[150] J. Detombe and S. Tavares, "Constructing Large Cryptographically Strong S-Boxes," Advances in Cryptology, Crypto, 1992, pp. 165–181.

[151] J. M. Rabaey, A. Chandrakasan, and B. Nicolic, *Digital Integrated Circuits, A Design Perspective*, New Jersey: Prentice Hall, 2003.

[152] Liam Keliher, H. Meijer and S. Tavares, "Toward the true random cipher: On expected linear probability values for spns with randomly selected s-boxes," *Communication, Information and Network Security*, pp. 123–146, 2003.

[153] J. Seberry, X. M. Zhang and Y. Zhang, "Cryptographic boolean functions via group hadamard matrices," *AJC: Australasian Journal of Combinatorics*, vol. 10, 1994.

[154] K. Nyberg, "Differentially uniform mappings for cryptography," in *Advances in Cryptology-Eurocrypt*, 1993, pp. 55–64.

[155] D. RayChaudhuri, *Digital Circuits*, vol. 1, chapter 6, pp. 254–258, 2001.

[156] W. de Launey, "Generalised hadamard matrices whose rows and columns form a group," *Combinatorial Mathematics X, Lecture Notes in Mathematics*, vol. 1036, pp. 154–176, 1983.

[157] J. Patarin, "Hidden Field Equations (HFE) and Isomporphism of Polynomials (IP) :two new families of Asymm. Algorithms," in *Eurocrypt 1996*. 1996, pp. 33–48, Springer Verlag.

[158] N. T. Curtois, "The Security of Hidden Field Equations (HFE)," in *Cryptographer's Track rsa Conference 2001*, LNCS 2020, 2001, pp. 266–281, Springer Verlag.

[159] N. T. Curtois, M. Daum and P. Felke, "On the Security of HFE, HFEvand Quartz," in *PKC 2003*, LNCS 2567, 2003, pp. 337–350, Springer Verlag.

[160] N. T. Curtois, "Higher Order Correlation Attacks, XL algorithm and Cryptanalysis of Toyocrypt," in *ICISC 2002*, LNCS 2587, 2002, pp. 182–199, Springer Verlag.

[161] N. T. Curtois, "Algebraic Attacks on combiners with Memory and Several Outputs," Cryptology ePrint Archive, Report 2003/125, 2003.

[162] A. Rudra, P. K. Dubey, C. S. Jutla etal., "Efficient Implementation of Rijndael Encryption with Composite Field Arithmetic," in *CHES*, CHES 2001:Paris, France, May 14-16 2001, pp. 171–184, Springer.

[163] H. Kuo and I. Verbauwhede, "Architectural Optimization for a 1.82 Gbits/sec VLSI implementation of the AES Rijndael Algorithm," in *CHES*, CHES 2001:Paris, France, May 14-16 2001, pp. 51–64, Springer.

[164] A.K. Lutz, J. Treichler and etal., "2 Gbits/s Hardware Realizations of Rijndael and Serpent:A comparative analysis," in *CHES*, CHES 2002:Hotel Sofitel, San Francisco Bay (Redwood City), USA, August 13-15 2002, Springer.

[165] S. Morioka and A. Satoh, "An Optimized S-Box Circuit Architecture for Low Power AES Design," in *CHES*, CHES 2002:, San Francisco Bay (Redwood City), USA, August 13-15 2002, Springer.

[166] S. Morioka and A. Satoh, "A 10-Gbps full-AES Crypto Design with a twisted BDD S-Box Architecture," *IEEE Transactions on VLSI Systems*, vol. 12, Issue: 7, pp. 686–691, July, 2004.

[167] D. Mukhopadhyay and D. RoyChowdhury, "An Efficient End to End Design of Rijndael Cryptosystem in 0.18 μ CMOS," 18^{th} International Conference on VLSI Design, Kolkata, India, January 3-7 2005, pp. 405–410, IEEE Computer Society.

[168] B. Gladman, "Implementations of AES (Rijndael) in C/C++ and Assembler," http://fp.gladman.plus.com/cryptography_technology/rijndael.

[169] F. Chabaud and S. Vaudenay, "Links between Differential and Linear Cryptanalysis," in *Advances in Cryptology-Eurocrypt 1994*. 1994, pp. 356–365, Springer, volume 950 of LNCS.

[170] M. Matsui, "New structure of block ciphers with provable security against linear and differential cryptanalysis," in *Fast Sofware Encryption 1996*. 1996, pp. 205–218, Springer, volume 1039 of LNCS.

[171] K. Nyberg, "Linear approximations of block ciphers ," in *Advances in Cryptology-Eurocrypt 1995*. 1995, pp. 439–444, Springer, volume 950 of LNCS.

[172] J. Daemen, *Cipher and Hash Function Design: Methods Based on Linear and Differential Cryptanalysis*, Ph.D. thesis, Katholieke Universiteit Leuven, March, 1995.

[173] J. Wallen, "Linear approximations of Addition Modulo 2^n ," in *Fast Sofware Encryption 2003*. 2003, pp. 261–273, Springer, volume 2887 of LNCS.

[174] C. Burwick, D.Coppersmith, E. D Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas, L. O Connor, M. Peyravian, D. Safford and N. Zunic, "Mars - a candidate cipher for AES," in *First Advanced Encryption Standard (AES) Conference, Ventura, CA*, 1998.

[175] X. Lai and J.L. Massey, "A Proposal for a New Block Encryption Standard," in *Advances in Cryptology-Eurocrypt'90*. 1991, pp. 389–404, Springer Verlag.

[176] A. Shimizu and S. Miyaguchi, "Fast Data Encipherment Algorithm FEAL," in *Proceedings of Eurocrypt 1987, Springer 1988, LNCS, 304*, 1987.

[177] F. X. Standaert, G. Piret, N. Gershenfeld, J. J. Quisquater, "SEA: a Scalable Encryption Algorithm for Small Embedded Applications," in *Seventh Smart Card Research and Advanced Application IFIP Conference* . 2006, Springer Verlag.

[178] A. Tardy Corfdir and H. Gilbert, "A Known Plaintext Attack of FEAL-4 and FEAL-6," in *Proceedings of Crypto 1991*, 1991, pp. 172–182.

[179] O. Staffelbach and W. Meier, "Cryptographic Significance of the Carry for Ciphers Based on Integer Addition," in *Proceedings of Crypto 1990*, 1990, pp. 601–614.

[180] S. Merphy S. Blackburn and K. Paterson, "Comments on theory and applications of cellular automata in cryptography," *IEEE Transactions on Computers*, vol. 46, no. 5, pp. 637–638, May 1997.

[181] N. Ganguly, A. Das, B. Sikdar and P. Pal.Chaudhury, "Cellular automata model for cryptosystem," in *Cellular Automata 2001*, Yokohama University, Japan, 2001, pp. 120–125.

[182] F. Bao, "Cryptanalysis of a new cellular automata cryptosystem," in *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, Wollongong, Australia, July 9-11 2003, vol. 2727 of *Lecture Notes in Computer Science*, Springer.

[183] C. K. Koc and A. M. Apohan, "Inversion of Cellular Automata Iterations," *IEE Proceedings - Computers and Digital Techniques*, vol. 144, no. 5, pp. 279–284, 1997.

[184] J. Daemen and V. Rijmen, *The Design of Rijndael*, Springer-Verlag, 2002.

[185] S. Hong, S. Lee, J. Lim, J. Sung and D. Cheon, "Provable security against differential and linear cryptanalysis for the spn structure," in *FSE-2000, LNCS 1636*, 2000.

[186] T.R.N Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*, Prentice Hall, Englewood Cliffs, NJ, 1989.

[187] S. Lee J. S. Kang, S. Hong and J. Lee, "Practical and provable security against differential and linear cryptanalysis for substitution-permutation networks," *ETRI Journal*, vol. 23, no. 4, pp. 158–167, December 2001.

[188] Y. Yacobi and Z. Shmuely, "On key distribution systems," in *In the Proceedings of Advances in Cryptology-Crypto '89, LNCS 435*, December 1989, pp. 344–355.

[189] D. S. Wong and A. H. Chan, "Efficient and mutually authenticated key exchange for low power computing devices," in *In the Proceedings of Advances in Cryptology-Asiacrypt '2001, LNCS 2248*, 2001, pp. 272–289.

[190] C. S. Park, "On certificate-based security protocols for wireless mobile communication systems," *IEEE Network*, vol. 11, no. 5, pp. 50–55, September/ October 1997.

[191] M. Jakobssen and D. Pointcheval, "Mutually authentication for low power mobile devices," in *Financial Cryptography, LNCS 2339*, 2001, pp. 178–195.

[192] K. Martin G. Horn and C. J. Mitchell, "Authentication protocols for mobile network environment value-added services," *IEEE Transactions on Vehicular Technology*, vol. 51, no. 2, pp. 383–392, March 2002.

[193] N. Linial and A. Wigderson, "Expander graphs and their applications," http://www.math.ias.edu/ boaz/ExpanderCourse/, 2003.

[194] M. A. Nielsen, "Introduction to expander graphs," http://www.qinfo.org/people/nielsen/, 2005.

[195] R. Phillips A. Lubotzky and P. Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, no. 3, pp. 261–277, 1988.

[196] G. A. Margulis, "Explicit constructions of expanders," *Problemy Peredači Informacii*, vol. 9, no. 4, pp. 71–80, 1973.

[197] G. A. Margulis, "Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators," *Problemy Peredachi Informatsii*, vol. 24, no. 1, pp. 51–60, 1988.

[198] G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar and T. Wollinger, "Efficient $gf(p^m)$ arithmetic architectures for cryptographic applications," in *CT-RSA 2003, LNCS 2612*, 2003, pp. 158–175.

[199] D. Boneh, R.A. DeMillo and R.J. Lipton, "On the Importance of Eliminating Errors in Cryptographic Computations," *Journal of Cryptology*, pp. 101–120, 2001.

[200] V. Rijmen, "Efficient Implementation of the Rijndael Sbox," http://www.esat.kuleuven.ac.be/ rijmen/rijndael.

[201] C. Carlet, "On Highly Nonlinear S-boxes and their inability to thwart DPA attacks (extended version)," Cryptology ePrint Archive, Report 205/387, 2005, http://eprint.iacr.org/.

Publications and Communications *

- **Conference Papers:**

1. † D. Mukhopadhyay, P. Joshi and D.RoyChowdhury, "An Efficient Design of Cellular Automata based Cryptographically Robust One-Way Function", Proceedings of 20^{th} International Conference on VLSI Design, VLSID 2007, pp 842-853, Bangalore, India.
2. † D. Mukhopadhyay and D. RoyChowdhury, "Generation of Expander Graphs Using Cellular Automata and its Applications to Cryptography", Proceedings of the 7^{th} International Conference on Cellular Automata for Research and Industry (ACRI 2006), LNCS 4173 pp 636-645, 20-23 September 2006, Perpignan, France
3. D. Bhattacharya, D. Mukhopadhyay and D. RoyChowdhury, "A Cellular Automata Based Approach for Generation of Large Primitive Polynomial and its Application to RS-coded MPSK Modulation", Proceedings of the 7^{th} International Conference on Cellular Automata for Research and Industry (ACRI 2006), LNCS 4173 pp 204-214, 20-23 September 2006, Perpignan, France
4. † D. Mukhopadhyay and D. RoyChowdhury, "Key Mixing in Block Ciphers through Addition Modulo 2^n, Proceedings of National Workshop in Cryptology, 2006.
5. † D. Mukhopadhyay and D. RoyChowdhury, "R6Crypt: A New Cryptosystem for Handheld Devices", Proceedings of International Conference on Computer & Communication Engineering, (ICCCE'06) May 2006, Kuala Lumpur, 9-11 May 2006.
6. † P. Joshi, D. Mukhopadhyay and D. RoyChowdhury, "Design and Analysis of a Robust and Efficient Block Cipher Using Cellular Automata", In the Proceedings of the 20^{th} International Conference on Advanced Networking and Applications (AINA'06), volume 2, pp 67-71, April 18-20, Vienna, Austria.
7. † D. Mukhopadhyay, A. Chaudhury, A. Nebhnani and D. RoyChowdhury, "CCMEA : Customized Cellular Message Encryption Algorithm for Wireless Networks", In the Proceedings of International Conference on Infor-

*The papers marked with † have been published out of the work reported in the thesis

mation Systems Security (ICISS 2005), LNCS 3803 pp 217-227, December 19-21, Kolkata, India.

8. D. Mukhopadhyay, S. Banerjee, D. RoyChowdhury and B. Bhattacharya, "CryptoScan: A Secured Scan Chain Architecture", in the Proceedings of Asian Test Symposium 2005, pp 348-353, December 18-21, Kolkata, India.
9. † D. Mukhopadhyay and D. RoyChowdhury, "Cellular Automata Based Key Agreement", 2^{nd} International Conference on E-Business and Telecommunication Networks", Microsoft Convention Centre at Reading U.K. (ICETE 2005), October 3-7, 2005.
10. D. Mukhopadhyay, S. Banerjee and D. RoyChowdhury, "Performing Scan Based Attack On a Stream Cipher Hardware", In the Proceedings of National Workshop on Cryptology, 12-14 August, pp 1-8, Shimoga, India.
11. † D. Mukhopadhyay and D. RoyChowdhury, "Secured Key Agreement Using Cellular Automata", In the Proceedings of National Workshop on Cryptology, 12-14 August, 2005, pp 85-94, Shimoga, India.
12. D. Mukhopadhyay and D. RoyChowdhury, "Programmable Galois Multiplier Using Cellular Automaton", 9^{th} VLSI Design and Test Symposium (VDAT 2005), pp 169-176, Bangalore, India.
13. Debdeep Mukhopadhyay and D. RoyChowdhury, "An Efficient End to End Design of Rijndael Cryptosystem in 0.18 μ CMOS", in the Proceedings of the 18^{th} International Conference on VLSI Design 2005 jointly held with 4^{th} International Conference on Embedded Systems Design, pp 405-410, Kolkata, India.
14. S. Banerjee, D. Mukhopadhyay and D. RoyChowdhury, "Computer Aided Test (CAT) Tool for Mixed Siganal SOCs", in the Proceedings of the comming 18^{th} International Conference on VLSI Design 2005, pp 787-790, Kolkata, India.
15. † D. Mukhopadhyay and D. RoyChowdhury, "New Observations on the Security of the Rijndael Cryptosystem", In the Proceedings of National Workshop on Cryptology 2004, pp 292-312, Kollam, India
16. D. Mukhopadhyay and D. RoyChowdhury, "An Efficient Galois Multiplier Using Cellular Automata", International Conference on Number Theory and Fourier Techniques-ICNFT 2004, Srinivas Ramanujam Centre, SASTRA Deemed University, Kumbakonam, India
17. † D. Mukhopadhyay and D. RoyChowdhury, "Cellular Automata : An Ideal Candidate for a Block Cipher", 1^{st} International Conference on Distributed Computing and Internet Technology (ICDCIT 2004), Lecture Notes in Computer Science, 3347, pp 452-457, Bhuvaneshwar, India.

18. † D. Mukhopadhyay and D. RoyChowdhury, "Characterization of a Class of Complemented Group Cellular Automata", 6^{th} International Conference on Cellular Automata for Research and Industry, ACRI'04, Lecture Notes in Computer Science, 3305, pp 775-784, Amsterdam, The Netherlands.
19. D. Mukhopadhyay and D. RoyChowdhury, "Design of a coprocessor for Galois Field Computation", In the Proceedings of the International Conference on Communication, Devices and Intelligent Systems, CODIS04, Kolkata, India.
20. S. Banerjee, D. Mukhopadhyay, and D. RoyChowdhury, "Testing of ADC Embedded in Mixed-Signal SOC", In the Proceedings of the International Conference on Communication, Devices and Intelligent Systems, CODIS04, Kolkata.
21. S. Banerjee, D. Mukhopadhyay and D. RoyChowdhury, " Automatic Generated Built-In-Self-Test for Embedded Memory", in the Proceedings of the IEEE Indicon 2004, pp 377-380, Kharagpur, India.
22. S. Banerjee, D. Mukhopadhyay and D. RoyChowdhury, "Best Repair: An Efficient Reconfiguration for RRAM", in the Proceedings of the IEEE Indicon 2004, pp 423-426, Kharagpur, India.
23. D. Mukhopadhyay and D. RoyChowdhury, "Design and Implementation of Cryptoattack on Secured Embedded Systems", In the Proceedings of the 6^{th} International Conference on Information Technology, CIT03, Bhubaneshwar, India.
24. C. V. Guru Rao and D. Mukhopadhyay and D. Roy Chowdhury, "A New Strategy and Design For Mixed signal SOC Testing", In the digest of the papers of 4^{th} International Workshop on RTL and High Level Testing (WRTLT'03) in conjunction with ATS'03 at Xi'an, P.R.China, November 20-21, 2003.
25. D. Mukhopadhyay and D. RoyChowdhury, "Smart Medical Service to the Rural World", In the Proceedings of the International Conference on Information Technology: Prospects and Challenges in the 21^{st} century, ITPC03, Kathmandu Nepal.
26. S.Basu, D. Mukhopadhyay, Dipanwita Roychoudhury, Indranil Sengupta, Sudipta Bhawmik, "Reformatting Test Patterns for Embedded Core Based Systems Using Test Access Mechanism (TAM) Switch", Proceedings of International Conference on ASP-DAC and VLSI Design 2002, pp 598-603, Bangalore, India.
27. C. V. Guru Rao, D. Mukhopadhyay, D. Roy Chowdhury, "A Design for test Technique for mixed mode SOC Design", 11^{th} Annual IEEE Symposium on System On a Chip, Bangalore, India, November 22-23, 2002.

28. D. Mukhopadhyay and D. RoyChowdhury, "Cellular Automata Based Cryptosystem Employing Galois Field (2^p) algebra", International Symposium on Cellular Automata, Yokohama, Japan, 2001.

- **Journal Papers**

1. † D. Mukhopadhyay and D. RoyChowdhury, "Theory of a Class of Complemented Group Cellular Automata and its Application to Cryptography", To appear in Journal of Cellular Automata
2. † D. Mukhopadhyay and D. RoyChowdhury, "Fault Based Attack on the Rijndael Cryptosystem", To appear in Journal of Discrete Mathematical Sciences & Cryptography
3. † D. Mukhopadhyay and D. RoyChowdhury, "Key Mixing in Block Ciphers through Addition modulo 2^n", To appear in International Journal of Computer, Mathematical Sciences and Applications.
4. † D. Mukhopadhyay and D. RoyChowdhury, "Customizing Cellular Message Encryption Algorithm", To appear in International Journal of Network Security.
5. D. Mukhopadhyay and D. RoyChowdhury, "Secured Flipped Scan Chain Model for Crypto-architecture", To appear in IEEE Transactions on CAD.
6. D. Mukhopadhyay, G. Sengar and D.RoyChowdhury, "Hierarchical Verification of Galois Field Circuits", To appear in IEEE Transactions on CAD.
7. S. Banerjee, D. Mukhopadhyay and D. RoyChowdhury, "A DFT Solution for Mixed Signal SOCs", IEEE Transactions on CAD, Volume 25, Issue 7, pp: 1368-1377, July 2006.

Biography of the Author

Debdeep Mukhopadhyay, was born on the 31^{st} of October, 1977 in Shibpur, Howrah, a twin town of Kolkata, West Bengal, India. He received his B. Tech degree in Electrical Engineering from the Indian Institute of Technology (IIT), Kharagpur in the year 2001. He then rejoined the Indian Institute of Technology, Kharagpur in the Department of Computer Science and Engineering and received his MS degree in the year 2004. The title of his thesis was "Hardware for Cryptography". He continued his study in the same department as a PhD student in the field of cryptography. During this period, he has worked as a Graduate Research Assistant in the Advanced VLSI Design Laboratory, IIT Kharagpur and subsequently as a Senior Project Officer in the Department of Computer Science and Engineering, IIT Kharagpur. He has worked in projects entitled, "End to End ASIC Design of AES Rijndael", sponsored by Indian Space Research Organization (ISRO), "Design and Implementation of Cryptosystems Resistant to Side-Channel Attacks", sponsored by the Department of Information Technology, India and "Testing of System on Chips", sponsored by Lucent Technologies, USA. His research interests include Cryptography, VLSI Design and Testing.