INVARIANT ATTACKS

A NEW TOOL FOR CRYPTANALYSIS TOOLSET

By

Khurram Shahzad

A thesis submitted to the faculty of Information Security Department,
Military College of Signals, National University of Sciences and Technology,
Islamabad, Pakistan, in partial fulfillment of the requirements for the
degree of MS in Information Security

September 2021

Declaration

I certify that this research work titled "Interpolation Analysis: New Criteria for Cryptanalytic attack Toolset" is my own work. No portion of this work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere. The material that has been used from other sources has been properly acknowledged / referred.

Signature of Student

Khurram Shahzad

00000325326

ABSTRACT

Security of block ciphers has always remained the focus of crypto research in order to establish the degree of confidence we can have on one or an entire family of block ciphers. For long, linear and differential cryptanalytic attacks provided the basis for most of the attacks against block ciphers, however, no major cipher in its full form could be successfully broken. Evolving technological landscape calls for lightweight block ciphers with simpler algorithms, key schedules, and round constants to achieve security as well as economy of size, energy, and cost in modern communication systems. This security compromise creates vulnerability of these lightweight ciphers towards modern attacks e.g., invariant attacks, interpolation attacks, boomerang attacks and many more.

Newly introduced Invariant Attacks try to map a single round of an SPN cipher in the form of a polynomial under a weak key setting using a step-by-step approach. Such polynomial must be invariant (unchanging) to the linear and non-linear components over multiple rounds of the underlying cipher. These attacks have successfully been applied to break lightweight SPNs like Midori64, Scream, iScream, Print and more. Until now, individual ciphers were attacked using various forms of invariant attacks e.g., nonlinear invariant attacks, invariant subspace attacks, generalized nonlinear invariant attacks and invariant hopping attacks exploiting numerous vulnerabilities.

This thesis will focus on invariant attacks against various ciphers, exploited vulnerabilities and present a cryptanalytic toolset that can be utilized to safeguard a cipher against the invariant attacks family. The toolset will provide a set of properties that if satisfied by all linear and non-linear components of a cipher in general and S-Box component in particular, will provide safety against these attacks.

COPYRIGHTS STATEMENT

DEDICATION

This thesis is dedicated to

MY LOVING FAMILY

for their love, endless support, and encouragement

ACKNOWLEDGEMENTS

I am grateful to Allah Almighty for giving me strength to keep going on with this thesis, irrespective of many challenges and troubles. All praises for HIM and HIM alone.

I am very grateful to my Project Supervisor Asst Prof. Dr. Fawad Khan, Co-Supervisor Asst Prof Dr Shahzaib Tahir and GEC members who supervised the thesis / research in a very encouraging and helpful manner. They always guided me with their profound and valuable support that helped me in achieving my research aims.

I would also like to extend my feelings of gratitude towards Brigadier Humayun Ajmal Bhatti, Major Umar Malik and Major Wajahat Sultan for their continuous guidance, motivation, and endless support.

Finally, I would like to express my appreciation to all the people who have provided valuable support to my study and whose names I could not bring to memory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Ever increasing reliance on network-based systems for sensitive communication has revitalized the need for information security exponentially. It is inevitable for the organizations to ensure protection of confidentiality, integrity, and availability of their sensitive data round the clock. Loss of significant information may cause financial, security and reputation damage to the entire system. It thus becomes imperative for every organization to guarantee information assurance through tiers of physical as well as cyber security. Cryptology as a field has evolved over the decades to be the major stakeholder in provision of information security [1]. Cryptology in nutshell deals with the development as well as security evaluation of various encryption systems. Cryptology may be broadly classified into two major branches, i.e., Cryptography and Cryptanalysis. Cryptography deals with development of secure and efficient cryptosystems, whereas Cryptanalysis as a field remains focused on methods to break or find vulnerabilities of these cryptosystems. In simple words, cryptanalysis as a branch of cryptography deals with the process of looking for security vulnerabilities in the observed ciphers aiming to eradicate these loopholes and make the desired cryptosystem secure, else, declare the cryptosystem insecure or obsolescent and thus put it out of service [2]. Cryptography and cryptanalysis techniques complement each other in order to provide the requisite data security for our communication systems. Major classes of crypto systems fall in symmetric and asymmetric crypto systems categories [3]. Symmetric key block ciphers distribute the data among fixed length data blocks and encrypt the data while processing it through iterative rounds of fixed or variable nature [4]. Modern symmetric block ciphers comprise unique combinations of Substitution and Permutation components and can be categorized as SPNs [5]. Since their inception, the dimension of research in the field has focused on two key areas, i.e., firstly, how to develop lightweight block ciphers with low processing and high-speed requirements suiting modern devices and secondly, explore cryptanalytic tools to identify loopholes in these block ciphers so that these can be bridged and made secure.

Two most significant block cipher cryptanalysis methodologies have been linear cryptanalysis and differential cryptanalysis [5, 6]. These two techniques have provided us with the basic understanding and procedure of carrying out these attacks against block ciphers [6]. Although these techniques have significantly reduced the brute force effort required to break few of the low complexity toy ciphers, nothing substantial has been proven regarding breaking the security of standard DES and AES algorithms [7]. Basing on the findings of linear and differential cryptanalysis techniques, research community have agreed upon the Difference Distribution Table (D.D.T) and the Linear Approximation Table (L.A.T) from the prospect of S-Box design as a design criterion that can help design a secure S-Box and cipher. Modern day technological advancements warrant development of small size, low complexity and low processing requirement cryptosystems that can make the communications secure without burdening or slowing down the system. Such cryptosystems normally reduce the processing overhead by trading off in the key alternating domain. Key schedule is either kept very simple or it is omitted altogether by using identical master key in all the rounds, which makes the systems vulnerable to analytical and structural attacks [8]. This leaves the onus of security to the S-Box, which is the only non-linear component of a standard Substitution-Permutation Network (SPN) [9]. Numerous analysis techniques rooting from the mathematical basis of S-Box have emerged since then. One of these attacks is the Invariant Attack which attempts to approximate the cipher under observation in the form of a polynomial through a structured approach. Such polynomial must be invariant (unchanging) to the linear and non-linear components over multiple rounds of the underlying cipher. This technique starts the analysis with a single round and accumulating to a wholesome round invariant polynomial representing the system. So far, four main types of invariant attacks have been proposed, i.e., the invariant sub-space attack, non-linear invariant attack, invariant hopping attack and generalized nonlinear invariant attacks [8]. These attacks will be studied in depth in order to create in-depth understanding and formalize a method to carryout invariant attack against generalized block ciphers.

1.1 Problem Statement

Block ciphers are a vital branch of secret-key crypto systems. They are employed either in isolation or in combination with other crypto variants to ensure information security among communicating parties in the presence of adversaries with malintent of eavesdropping on the contents communicated. Cryptanalysis of block ciphers has been a well-researched area among the academia, industry, and defense establishments for numerous reasons. Major reasons include measures to locate vulnerabilities of block ciphers in whole or in part including their mathematical structures and to assure the users on the level of security provided by these ciphers. Earlier proposed techniques including linear and differential cryptanalysis have provided little success into analysis of light weight versions of block ciphers with much reduced rounds, block size, key length and complexity, however, there is no substantial proof on the universal viability of these techniques to decode any block cipher of standard dimension. Among analysis of standard SPN ciphers, the major resistance is posed by the S-Box that provides the non-linearity property through a well-articulated combination of confusion and diffusion attributes proposed by Shannon. The invariant attack endeavours to map the various rounds of a cipher in the form of a function and then expand the function to represent the entire cipher, however, up till now, there has been no standard approach to implement these attacks. Each cipher has its own independent structure and therefore the representation would also be unique. The core aim of current research is to create a deep insight into the invariant attacks with an aim to draw pertinent conclusions regarding analysis of modern block ciphers so that we can generalize our findings applicable to a broad set of block ciphers. This shall in turn lead us to find the strengths and vulnerabilities of employed S-boxes and help us in designing secure/ fail safe S-boxes custom tailored for specific applications. This research will also contribute to propose an organized set of tools required to analyze an SPN with regards to invariant attacks.

1.2 **Motivation**

Pakistan Army is involved in military operations of varying natures during war as well as peace. The vast areas of deployment call for secure communication channels for speedy and safe transfer of sensitive information among various operational elements. Block ciphers have been in use among militaries around the world for secure communication. Due to same reason, immense research is being carried into this area of cryptology. For example, during 1980's, a simple but robust block cipher T-310 (consisting of 150 rounds) remained employed by American forces as a military grade cipher to encrypt secret government communication among various sensitive elements deployed all over Europe [10]. Compared to other counterparts from symmetric and asymmetric domains, block ciphers offer numerous overriding advantages that include simplicity of design, high level security due to large key-space and time/ resource requirements required to brute-force the key in real-time scenarios. By the time, the key would be retrieved, the intelligence might have lost its value for the adversary. Another inherent advantage while employing block ciphers in military systems is the availability of alternative secure channels for sharing of secret key among various elements. Moreover, the similarity of encryption and decryption processes supports speedy point-to-point and point-to-multipoint communications. It is in this backdrop, that the focus of cryptographic research has focused on cryptanalysis of block ciphers specially the non-linear components, i.e., the S-Box. Since no major breakthrough has been achieved through major cryptanalysis techniques including differential cryptanalysis and linear cryptanalysis of full-scale block ciphers, it is hoped that these new generation of attacks, i.e., the invariant attacks might pave way to structuralize the cryptanalysis of these block ciphers rendering these ciphers obsolete OR raise our confidence and trust on block ciphers declaring them secure enough for our information security needs. Keeping this in mind, this research will benefit in improving our insight into invariant attacks and provide basis for further research by concerned stakeholders from industry, academia, and defense research organizations.

1.3 **Research Objectives**

Following are the objectives of this research.

- Detailed literature review and exploring the existing mathematical structures.
- Mathematical design of S-Box and evaluation methods.
- Define a method based upon invariant attacks for the S-Box evaluation.
- Pseudocode and implementation for different structures.

1.4 **Contribution**

The research being undertaken in the field of S-Box evaluation and the applicability of invariant attacks for S-Box analysis offers multi-faceted advantages as under: -

- It is aimed to evaluate the existing work done on the subject thus providing a platform for further work.
- Considering the scarcity of research and the complexity of the subject, it will be structured to take the form of a simple tutorial for beginners with a view to enhance understanding.
- In addition to locating the vulnerabilities in the block ciphers, this research shall also examine ways to construct fail-safe and secure S-Boxes for the future which will contribute towards boosting our confidence in simple and lightweight SPN ciphers with secure S-Boxes.
- The research work can help analyze, evaluate and design secure ciphers not vulnerable to invariant attacks, and be benefited by military, commercial and general public communications.

1.5 **Thesis Outline**

The research work has been organized and distributed in following chapters:

- **Chapter 1**: A brief introduction is given, problem statement is highlighted, followed by motivation behind the research and research objectives are identified / explained. Furthermore, the contributions made through this research are highlighted. A short tutorial giving basic understanding of Invariant Attacks is also provided.
- **Chapter 2**: A birds-eye view of existing / recent research in the field of S-Box analysis and invariant attacks is discussed.
- **Chapter 3**: An insight into the S-Box properties, criteria for a strong Boolean function involved in a cryptographically strong S-Box.
- **Chapter 4**: In-depth analysis of Invariant Attack, its overview, basic types, ciphers which are vulnerable to invariant attack and reasons for that. SageMath analysis of

ciphers attacked by invariant attack. Design criteria to safeguard ciphers from invariant attacks.

- **Chapter 5**: Future work basing on proposed design criteria, need for formalization of the methodology and publication of tool kit to check and ensure resistance against invariant attacks.
- **Chapter 6**: Concluding remarks.

1.6 Brief Tutorial on Invariant Attacks

1.6.1 Defining and Understanding Non-Linear Invariant Attacks

Step-I : Cipher Selection and Assumption Phase

Suppose there is a block cipher defined by: $\boldsymbol{E_k}: \boldsymbol{F_2^N} \rightarrow \boldsymbol{F_2^N}$

- Implementing an Invariant Attack on this block cipher is to calculate an effectively calculable Boolean function i.e., $\boldsymbol{g}: \boldsymbol{F_2^N} \rightarrow \boldsymbol{F_2}$ such that $\boldsymbol{g(P) + g(E_k(P))}$ remains constant for any Plaint Text P and a large number of Keys from the key space. Such keys are known as weak keys. The function *g* is known as Non-Linear Invariant for the cipher *E_K.*
- For *h* number of plaint text, ciphertext pairs $\{\mathbf{P}, \mathbf{E_k(P)}\}$, the probability that these pairs will possess the invariant property is computed to be approximately $\mathbf{2^{-h+1}}$ where *g* remains balanced.

Now, on the first look, this idea seems complex. This tutorial will attempt to simplify the concept for a common reader. For ease of assimilation, suppose there is 8 rounds simple SPN cipher with **12 bits** both input as well as output. Suppose there **are 4 S-Boxes** S_{11}, S_{12}, S_{13} and S_{14} with **3 bits input / output**.

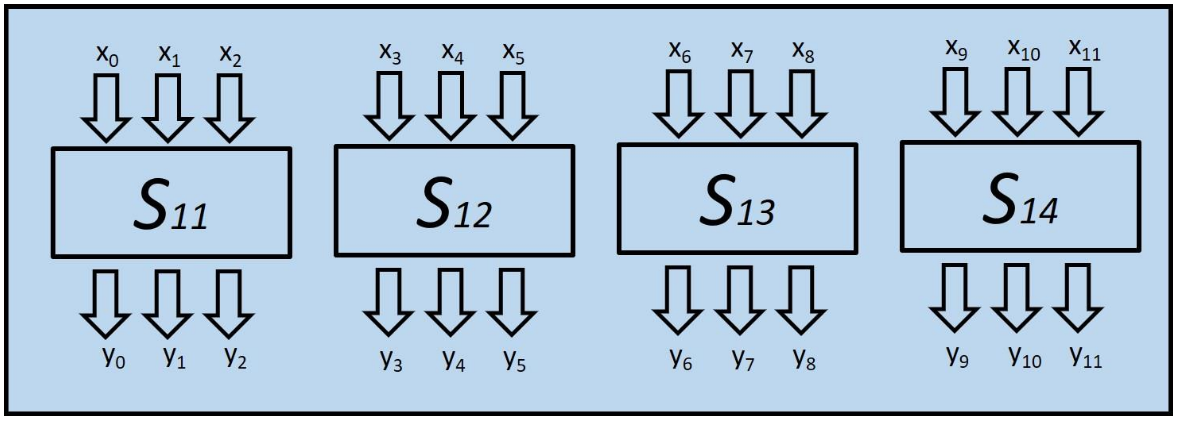


Figure 1.1 S-Box Layer of Single Round of simple SPN

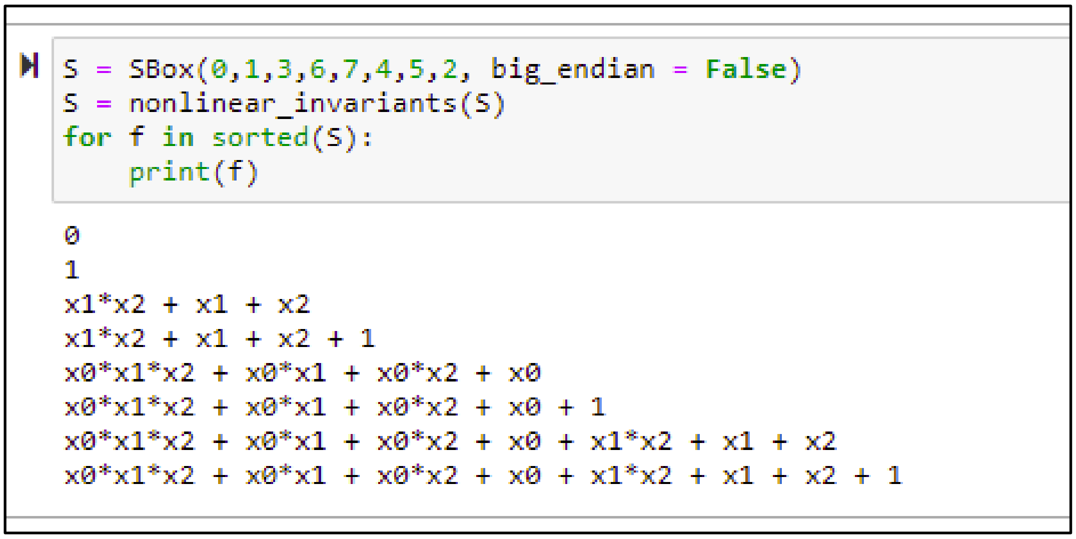
For simplicity, we consider single S-Box S_{11} and also assume that all 8 rounds of the cipher use same S-Box S_{11}. Look up table of the S-Box we have used is as under:-

x	0	1	2	3	4	5	6	7
S(x)	**0**	**1**	**3**	**6**	**7**	**4**	**5**	**2**

Table 1.2 S-Box Lookup Table

Step-II : Invariant Computation Phase

For this phase, we calculate an invariant for the S-box. There can be various combinations of invariants. We will use SageMath tool to compute the invariant of the cipher. The Sage function as well as list of output invariants is as under:-



```
S = SBox(0,1,3,6,7,4,5,2, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

```
0
1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure 1.2 Nonlinear Invariant of S-Box S_{11} Calculated by SageMath

Let the Invariant polynomial representing the S_{11} be as under: -

$$Y_{11} = x_2.x_1 \oplus x_1 \oplus x_2$$

Step-III : Attack Implementation Phase

Now, we will try to implement invariant attack on part of this cipher using same and different key combinations.

1.6.1.1 **Using a Single 4-Bit Master Key in Each Round**.

- **Input**: We initialize our input to $\mathbf{S_{11}}$ as **110**.
- **Key**: Let the single key common for each round be **101**.
- **Output**: **CT = PT ⊕ Key**.
- **Constant C**: **g (PT) ⊕ g (CT) = C**.

The results of 8 rounds iteration are summarized in the table below:

Round	**Input Bits (x)**			**G(x)**	**Key Bits (k) Fixed Key**			**Output Bits (y) y = x + k**			**G(y)**	**Constt C=G(x) + G(y)**
	x_2	**x_1**	**x_0**		**k_2**	**k_1**	**k_0**	**y_2**	**y_1**	**y_0**		

1	1	1	0	1	1	0	1	0	1	1	1	**0**
2	0	1	1	1	1	0	1	1	1	0	1	**0**
3	1	1	0	1	1	0	1	0	1	1	1	**0**
4	0	1	1	1	1	0	1	1	1	0	1	**0**
5	1	1	0	1	1	0	1	0	1	1	1	**0**
6	0	1	1	1	1	0	1	1	1	0	1	**0**
7	1	1	0	1	1	0	1	0	1	1	1	**0**
8	0	1	1	1	1	0	1	1	1	0	1	**0**

Table 1.2 Results of 8-Round Iteration with Fixed Key

Observation

- Now basic equation of the Invariant Attack i.e $\boldsymbol{g(P)\oplus g\big(E_k(P)\big)}$ is constant has been satisfied since in each round, the output of the polynomial.

$$\boldsymbol{Y_{11} = x_3.x_1 \oplus x_2.x_0 \oplus x_2 \oplus x_1 \oplus x_0}$$

  Value of constant comes equal to “0” each time.

- Therefore, the polynomial is true round invariant for the S-box and key is a ***weak key***.

- Cipher in question is vulnerable to invariant attack.

Inference. From above observation, it can be inferred that, “***all lightweight ciphers that do not employ a key-alternating schedule or use single key in each round are vulnerable to invariant attacks***”.

1.6.1.2 **STEP-2**: Using Randomly Generated / Different Key in Each Round. Now we provide randomly generated keys to each round and observe the results. The results are summarized in the table: -

Round	**Input Bits (x)**			**G(x)**	**Key Bits (k) Random Key**			**Output Bits (y) y = x + k**			**G(y)**	**Constt C=G(x) + G(y)**
	x_2	x_1	x_0		k_2	k_1	k_0	y_2	y_1	y_0		
1	1	1	0	1	1	1	1	0	0	1	0	**1**
2	0	0	1	0	1	1	0	1	1	1	1	**1**
3	1	1	1	1	0	1	0	1	0	1	1	**0**
4	1	**0**	1	1	0	0	1	1	0	0	1	**0**
5	1	0	0	1	1	0	0	0	0	0	0	**1**
6	0	0	0	0	0	1	1	0	1	1	1	**1**
7	0	1	1	1	0	0	0	0	1	1	1	**0**
8	0	1	1	1	1	0	1	1	1	0	1	**0**

Table 1.3 Results of 8-Round Iteration with Different Keys

Observation

The output of the polynomial

$$Y_{11} = x_3.x_1 \oplus x_2.x_0 \oplus x_2 \oplus x_1 \oplus x_0$$

Is random i.e it comes as “0” four times and “1” for four times out of eight rounds.

Inference. From above observation, it can be inferred that, “*all ciphers that do employ a key-alternating schedule or use random keys in each round are safe from invariant attacks*”.

1.6.2 How to Create a Cumulative Round Invariant Polynomial for Complete Cipher

Previously, we illustrated the basic concept of Invariant Attacks with the help of two examples in which each round used an identical S-Box. So, we used a single polynomial for entire cipher to prove the core idea. This was just for ease of assimilation. In actual scenarios, this may not be the case. S-Boxes used in various rounds may be different, therefore, for each round, a different polynomial will be formulated. Once we have done that, the resulting polynomials will be combined to give a single polynomial representing the entire cipher. Now we will try to elaborate this concept using same 8-rounds SPN.

Let $\boldsymbol{R}: \boldsymbol{F}_2^N \rightarrow \boldsymbol{F}_2^N$ represents the round function of a cipher. After including the key-XOR operation, the equation of the round function is represented as $\boldsymbol{(x) = R\ (x \oplus k)}$. Therefore, if the cipher is composed of r number of rounds, the Ciphertext C will be computed as under:-

$$\boldsymbol{x_0 = P,}$$

$$\boldsymbol{x_{i+1} = R_{ki}(x_i) = R(x_i \oplus k_i);\ 0 \leq i \leq r - 1,}$$

$$\boldsymbol{C = x_r}$$

With this structure of a round function in a cipher, an invariant attack tries to find out an invariant function g for each round such that, for any number of existing weak keys, the function $\boldsymbol{g(R(x \oplus k)) = g(x \oplus k) \oplus c = g(x) \oplus g(k) \oplus c \oplus x}$, this simply implies that once a function g is applied onto a round function R, it amounts to XOR of function *g* calculated on the Plaintext and function *g* calculated on the key bits. In addition, it creates constant bit for each round which is also XORed with the output. This inference, once applied to a simple 8-round SPN, provides an interesting analogy as under: -

$$g(C) = g(R(x_{r-1} \oplus k_{r-1}))$$

$$= g(x_{r-1}) \oplus g(k_{r-1}) \oplus c$$

$$= g(R(x_{r-2}) \oplus g(k_{r-2}) \oplus g(k_{r-1}) \oplus c$$

$$= g(x_{r-2}) \oplus g(k_{r-2}) \oplus g(k_{r-1})$$

..

$$= g(P) \oplus \bigoplus_{i=0}^{r-1} g(k_i) \quad \oplus \bigoplus_{i=0}^{r-1} c$$

1.6.3 **Proof**.

- ***Round-1:*** $g(x_1) = g(PT) \oplus g(k_0) \oplus C_0$
- ***Round-2:*** $g(x_2) = g(x_1) \oplus g(k_1) \oplus C_1$
- ***Round-3:*** $g(x_3) = g(x_2) \oplus g(k_2) \oplus C_2$
- ***Round-4:*** $g(x_4) = g(x_3) \oplus g(k_3) \oplus C_3$
- ***Round-5:*** $g(x_5) = g(x_4) \oplus g(k_4) \oplus C_4$
- ***Round-6:*** $g(x_6) = g(x_5) \oplus g(k_5) \oplus C_5$
- ***Round-7:*** $g(x_7) = g(x_6) \oplus g(k_6) \oplus C_6$
- ***Round-8:*** $g(x_8) = g(CT) = g(x_7) \oplus g(k_7) \oplus C_7$

Now, we know that $g(x_8) = g(CT)$, so incorporating values of previous rounds in Round-8 equation, we can conclude that:

7 **7**

$$G(C) = g\ (PT) \oplus \bigoplus_{i = 0} g\ (k_i) \oplus \bigoplus_{i = 0} C$$

Conclusion. We can conclude that, “***the final invariant polynomial for a simple SPN cipher can be computed by separately computing the function g over the PT XORed with the cumulated g(K*$_i$*) and XORed with cumulated constant C of all rounds***”.

CHAPTER 2

EXISTING RESEARCH ON S-BOX ANALYSIS AND INVARIANT ATTACKS

2.1 Introduction

In this chapter, we will carry out review of work done in the domain of block cipher cryptanalysis with specific focus on the research carried out on invariant attacks. Up till now, different researchers have launched invariant attacks on few lightweight ciphers like SCREAM, iSCREAM, Midori64, T-310 and others basing on certain assumptions. The results are encouraging, however, are localized to specific ciphers and no universal methodology could be proposed.

2.2 Existing Research

Block ciphers were introduced as a cryptographic technique for data security in early 1970's. Since then, block ciphers have remained an active area of application and research for industry and academia alike. Their design protocols have been actively scrutinized and evaluated using standard cryptanalysis tools and techniques including differential attacks [11], linear attacks [12], and different other variants of these approaches [13]. Thereafter, the focus of cryptanalysis research evolved towards Generalized Linear Cryptanalysis (GLC) which proposed existence of non-linear polynomials. This was introduced at Eurocrypt'95 [14]. It was argued that the iterated structure of block ciphers was vulnerable to round invariant attacks similar to Linear Cryptanalysis [15]. Harpes *et al.* [16] and Knudsen *et al.* [17] were the first to carry out detailed research on non-linear cryptanalysis. The major focus of researchers was to find a polynomial which is round invariant attacks, i.e. the value of which remains constant after one round, however, the same polynomial even if it exists is very complex to find from such a large space [18]. Moreover, it would be computationally not feasible to calculate this polynomial with conventional cryptanalytic methods [19]. Recently, researchers have been able to locate very limited number of attacks and that too with impossibility results [20]. Initially, once invariant attacks were proposed for analysis of block ciphers, these attacks were able to trivially locate polynomials of degree 2 only [21]. This led to the initial application of non-linear cryptanalysis of full-scale block ciphers which uses nonlinear round invariants [22]. An important observation was highlighted in this context with regards to wise choice of round constants that could weaken the attacks against light weight block ciphers. Authors also presented certain countermeasures to ensure robustness of these ciphers against invariant attacks, however, same have not been properly proven afterwards [23]. In recent years, the crypto community has focused on cryptanalyzing specific families of block ciphers. It was proven that well established block cipher families are quite robust against invariant attacks. Further research highlighted that rather than full-fledged block ciphers, their light-weight variants were more vulnerable to invariant and similar kind of attacks mainly due to their weak or no key alternating schedules [24].

In the same context of attempting to attack specific block cipher families, during ASIACRYPT 2016, Todo *et al.* [25] presented their findings on results of non-linear invariant attacks against full round ciphers, e.g., SCREAM [26], iSCREAM [27] and Midori64 [28]. These invariant attacks against full round ciphers could be launched with an assumption that the key for the attack be chosen out of weak key class. This could substantially reduce the brute-force effort by restricting to a limited key space [29]. This attack, however, weakens due to smart choice of round constants which can affect the non-linear invariants of ciphers. With an aim to overcome the impact of round constants from invariant attack proposed by Todo *et al*, authors presented another attack as Generalized Non-linear

Invariant Attack (GNIA) which applies a pair of constants to the input of a full round invariant attack against iSCREAM cipher employing weak keys. In general, authors proposed a new concept of closed loop S-box invariants and proved that the choice of strong round constants is linked to the existence of linear structure of the closed-loop invariants of the substitution layer. GNIA proved to be conditionally successful against SPN block ciphers, specifically it was able to launch distinguishing attack against a modified version of iSCREAM employing weak key class. In comparison, the nonlinear attack proposed by Todo *et al.* was not successful against iSCREAM.

In the same context, Beierle *et al.* took the study of impact of round constants on the resistance of block ciphers to the next stage. The major outcome in [17] is that the we can independently chose round constants regardless of the substitution layer. Same assumption could not be found optimal in case of many block ciphers. Further research showed that few famous block ciphers like PRESENT, PRINCE, and L-block do not include above mentioned closed-loop invariants and are thus resilient against GNIA.

As per research, invariant attacks have been characterized into two major categories. Firstly, the non-linear invariant attacks and secondly the invariant subspace attacks. The nonlinear type of invariant attacks was proposed in [30] and are considered as a further extension of sub-space invariant attacks. Sub-space invariant attacks also take the assumption that if the selected round-key is part of the weak-key class, then both input and output will belong to common affine sub-space through multiple encryption rounds. Second main type of invariant attacks, i.e., the non-linear invariant attacks were proposed/ demonstrated by Todo *et al*. These attacks try to locate non-linear invariants of each round separately and then combined to form a single invariant polynomial representing the entire cipher. There are two main difficulties in this scenario, one that such a comprehensive polynomial is very hard to find and second, that all round keys must belong to same weak key class. It can be stated with confidence that certain lightweight forms of block ciphers are vulnerable to the invariant attacks. For ease of assimilation, the crux of research work on the subject is summarized in Table-3 below:-

Subject	Cryptanalysis Technique Employed / Major Focus	Ref
Differential CA	Concepts of Differential CA on DES and DES-like Cryptosystems proposed	[11]
Linear CA	Concepts of Linear CA on DES and DES-like Cryptosystems proposed	[12]
Generalized Linear CA	Argued that the iterated structure of block ciphers was vulnerable to round invariant attacks similar to Linear Cryptanalysis	[15]
Non-Linear Approximations in Linear CA	First to carry out detailed research on non-linear cryptanalysis. The major focus of researchers was to find a polynomial which is round invariant representing the cipher	[16], [17]
Non-Linear Characteristics in Linear CA	Concluded that if an invariant polynomial does exist, it is very complex to find from such a large space. Moreover, it would be computationally infeasible to calculate this polynomial.	[18], [19]
GNIA and a New Design Criterion for Rd Constants	Invariant attacks were proposed for analysis of block ciphers, these attacks were able to trivially locate polynomials of degree 2 only	[20]
Non-linear Invariant Attack against SCREAM, iSCREAM & Midori64	Initial successful implementation of non-linear cryptanalysis of full-round block ciphers which uses nonlinear round invariants	[22]
Resistance Against Invariant Attacks: How to Choose RCs	Authors proposed that wise choice of round constants that could weaken the attacks against light weight block ciphers. Presented certain countermeasures to ensure robustness of these ciphers against invariant attacks.	[23]
Nonlinear Invariant Attack: Practical Attack on Full SCREAM, iSCREAM, and Midori64	Full versions of block ciphers are secure against invariant attacks; however, their lightweight variants are more vulnerable due to weak or no key alternating schedules. Todo et al. presented their findings on results of successful non-linear invariant attacks against full round ciphers, e.g., SCREAM, iSCREAM and Midori 64 using weak-key class	[24], [25]
Nonlinear Invariant Attack: Practical Attack on Full SCREAM, iSCREAM, and Midori64	Invariant attacks have been characterized into two major categories. Firstly, the Non-linear invariant attacks and secondly the invariant subspace attacks. Non-linear invariant attacks try to locate non-linear invariants of each round separately and then combined to form a single invariant polynomial representing the entire cipher. There are two main difficulties in this scenario, one that such a comprehensive polynomial is very hard to find and second, that all round keys must belong to same weak key class.	[30]

Table 2.1 Summary of Literature Review Findings

2.3 **Conclusion**

The scarcity of research on the subject and absence of a common agreed upon framework to launch invariant attacks against different variations of block ciphers including lightweight versions still needs a lot of work. Our research will explore the mathematical structures, design, and properties of S-Box with a view to develop deep understanding of invariant attacks to propose a standardized methodology for further work.

CHAPTER 3

MATHEMATICAL DESIGN OF S-BOX AND EVALUATION METHODS

3.1 **Introduction**. Block ciphers, among all cotemporaries, have proven to the most popular category of crypto algorithms [31]. In block cipher operation, entire plaintext stream is distributed into fixed equal length data blocks, which are then treated separately for entire encryption-decryption cycle. A plaintext block when encrypted with same cipher and same key, will always produce the same ciphertext. The data blocks are processed through repeated encryptions known as round function. Each block cipher comprises different number of rounds depending upon the design structure. Many such encryption schemes employ quite simpler operations repeated through multiple number of rounds. Block ciphers are mostly better suited for software implementations [32].

3.2 Basic Block Cipher Mechanism and Components

- **Block Cipher Components**. There are five main components of a block cipher namely [33]:-

Plaintext message-space	-	M
Ciphertext message-space	-	C
Key Space	-	K
Encryption Algorithm	-	$E = \{E_k: k \varepsilon k\}$
		$E_k = M \longrightarrow C$
Decryption Algorithm	-	$D = \{D_k: k \varepsilon k\}$
		$D_k = C \longrightarrow M$

Basic symmetric encryption equation: $\boldsymbol{D_k(E_k(m)) = m}$

3.3 **Mathematical Structures**. The design structure of the block cipher is of paramount importance in its operation. It describes the length of data block, length of key, number of encryption rounds, components of the cipher including the key alternating algorithm, round function, and the structure of substitution / permutation components etcetera. Broad types of block cipher structures are described as under [34]:-

(1) Feistel structure.

(2) Substitution - Permutation structure (focus of this thesis).

(3) MISTY structure.

(4) L-M structure.

(5) Generalized Feistel structure.

- **Basic Concept of Feistel Structures**. The Feistel structure was introduced in 1970's and forms the basis of many modern ciphers [35]. The working of Feistel ciphers is explained in under mentioned steps:-

  (1) The plaintext block is initially distributed into two halves i.e., left and right.

  (2) Then application of round function to one half.

  (3) Finally, outcome from the round function is Xor-ed with other half.

  (4) The Feistel structure is explained in Figure 3.1 below:-

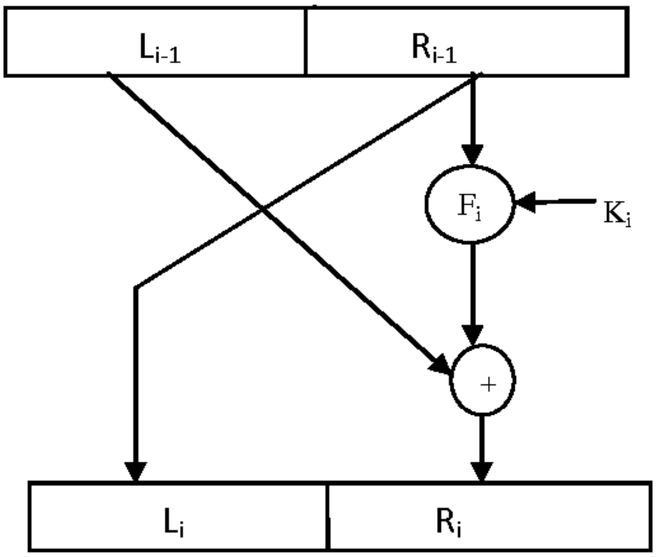


Figure 3.1 Feistel Structure

- **Substitution – Permutation Network Structures (SPNs)**.

  (1) The Substitution-Permutation networks form the core of most modern block ciphers due to their inherent security introduced by the confusion-diffusion criteria proposed by Claude Shannon in 1949 [36]. In SPN ciphers, the permutation and substitution operations are applied to the entire data block at one time. Then the output of current round is fed to the next round as input.

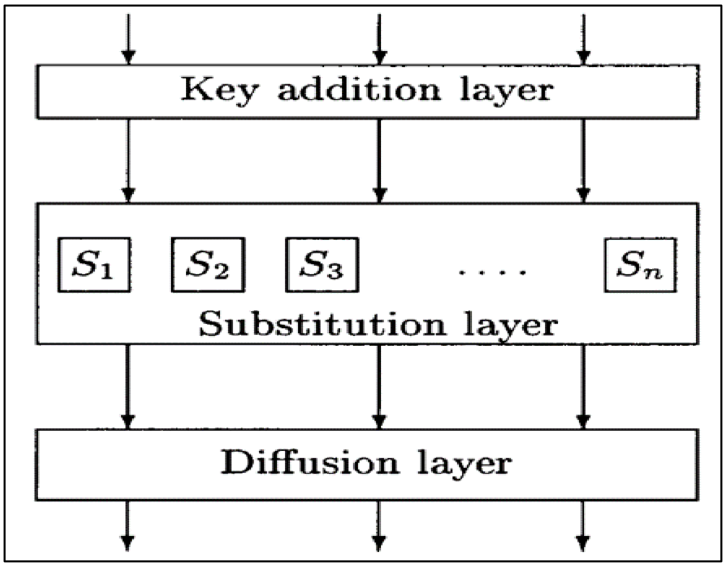


Figure 3.2 SPN structure

- **Permutation Box**. The permutation or the diffusion layer is the linear layer of SPN ciphers. It is simply linear displacement of data bits to predefined positions. This operation can be realized either by a simple look-up permutation table as in case of DES, or by mix column operation by multiplying with a linear matrix. Either way, this layer does not offer much security to the block cipher, however, once applied with the correctly balanced components, it becomes crucial to the cipher's security.

- **Substitution Box**. The substitution or confusion layer of the SPN is the most important as well as critical layer of the entire cipher with regards to security. It is also realized as a lookup table; however, all substitutions are carried out with appropriate Boolean functions satisfying security criteria. The substitution box or S-Box structure is kept secret for security reasons. The diagrammatic layout of S-Box, P-Box combination is represented in Figure 3.3 below:-

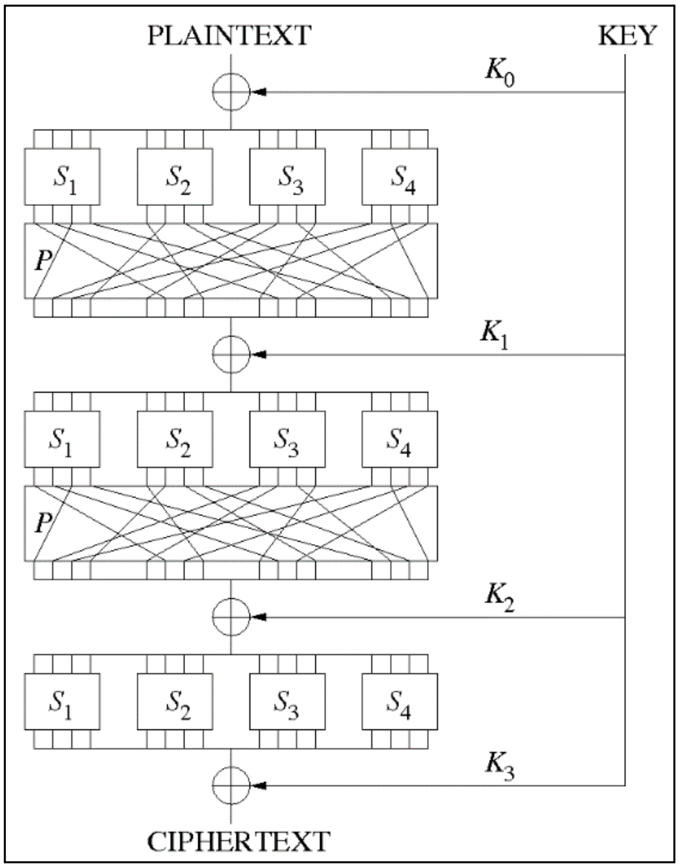


Figure 3.3 Combination of S-Box and P-Box in SPN Cipher

- **Round Function**. A round function is the processing of data through single round of encryption. It takes as input the plaintext bits and processes this plaintext including round key XOR, substitution and permutation. It produces as output the ciphertext for that specific round. Block diagram of round function of Data Encryption Standard (DES) is explained in Figure 3.4 below:-

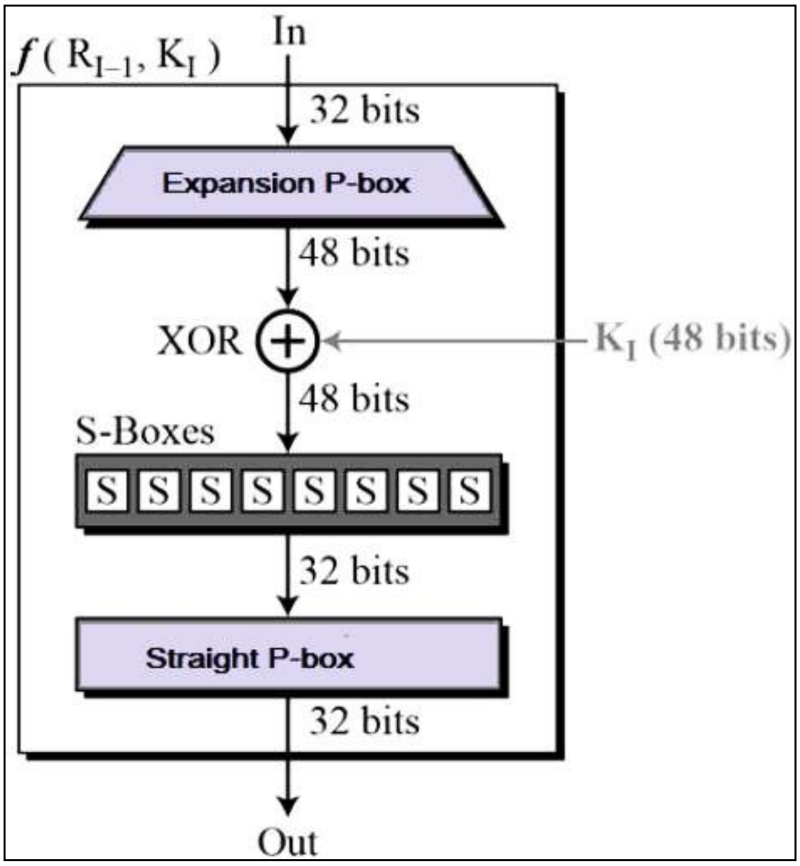


Figure 3.4 Round Function of DES

- **Key Schedule**. In secret-key cryptosystems, a common secret key is employed at both encryption and decryption ends. This secret key or master key is passed through a key-generation algorithm for creation of round sub-keys to provide a secure key for each round of encryption. The methodology adopted to generate these round keys differs with each cipher depending upon the security requirements and design.

3.4 **Understanding of Boolean Functions – A Prelude to Understanding S-Box**. Boolean functions were pioneered in 19th Century by G. Boole, who established the application of mathematics in logic. In cryptography, Boolean functions form the basis of large number of systems including block ciphers and stream ciphers. Knowledge of Boolean functions is critical to the understanding of S-Box design. An S-box is constituted of numerous output Boolean functions; in other words, each output bit from one S-Box is outcome of a separate Boolean function. Moreover, Boolean functions exhibit few important cryptographic properties, which ensure the resilience of an S-Box against many families of attacks. In the subsequent paragraphs, few important terms related to Boolean functions will be defined in order to set the stage for S-Box analysis.

3.5 Few Related Definitions

- **Boolean Function** Suppose an n-dimensional vector space ***$GF(2)^n$***, then, a Boolean function **f(x)** can be defined as a mapping ***$f: GF(2)^n \rightarrow GF(2)$***; Where, $x = (x_n,x_n-1,...,x_{n.}1)$ and ***$GF(2)^n$*** represents a Galois field of order 2^n. In this case, the total number of distinct n-variable Boolean functions will be ***$2^{2^\wedge n}$***. In case we enhance number of inputs 'n', the number of possible output Boolean functions will increase exponentially.

- **The Truth Table.** For a Boolean function f(x) comprising n-variables, a truth table is the binary output vector representing f(x) and contains 2^n elements. There exists a unique truth table for every Boolean function. Suppose a Boolean function ***$f : \{0,1\}^2 \rightarrow \{0,1\}$***. A truth table represents all possible combinations of inputs x_1 and x_2 with linear function F. The truth table is represented in **Table 3.1** below:-

x_1	***x_2***	***F***
0	0	1
0	1	1
1	0	0
1	1	0

Table 3.1 Truth Table Representation

- **Algebraic Normal Form**. In ANF, we represent a Boolean function as a polynomial in ***$F_2(x_1,\cdots,x_n)$*** with its bitwise sum of its input bits. For example, the ANF representation of Table 2.1 will be

$$f(x_1,x_2) = x_1 \oplus 1.$$

- **Hamming Weight (HW)**. For an n-variable Boolean function f(x), hamming weight ***HW(f)*** is the total number of ones in the truth table of ***f(x)***.

- **Hamming Distance (HD)**. Hamming distance ***HD(f , g)*** between two n-variable Boolean functions, ***f(x)*** and ***g(x)*** represents the total count of differing elements in corresponding positions between the two truth tables of ***f(x)*** and ***g(x)***.

- **Algebraic Degree (AD)**. For a Boolean function f(x), algebraic degree is the total number of variables in the largest product term (having a nonzero coefficient) of the function's ANF.

- **Equivalent Boolean Functions.** Two distinct n-variable Boolean functions, f and g, are equivalent functions iff there exist some a,b,c, and M, such that following equivalence relation holds true:

$$g(x) = f(Mx \oplus a)b.x \oplus c;$$

where a and b are binary n-variable vectors, c is a binary scalar, and M is an n × n invertible binary matrix.

3.6 S-box theory

With brief introduction to the Boolean functions, we shift our focus to the main subject of this thesis i.e., S-Box, which is a unique combination of one or more Boolean functions. Before getting to the crux, it is important to create basic understanding of S-Box concepts. This Section will cover basic definitions of S-Box and related terminologies.

Definition of S-Box. S-box with n x m configuration is a mapping from n input bits to m output bits, S : $\{0,1\}^n \rightarrow \{0,1\}^m$. An S-box is a combination of m single bit output Boolean functions combined in a predefined order. There can be 2^n inputs and 2^m possible outputs for an n × m S-box. S-Box is often realized in the form of a look-up table without disclosing the Boolean functions.

Cryptographic properties of S-boxes

Since an S-Box is composed of numerous Boolean functions (one for each bit output), in order to study the security properties of an S-Box, the effect of all linear combinations of component Boolean functions needs to be studied.

Few important properties of S-Box are elaborated below:-

- **Balance (B)**. An S-Box $S:\{0,1\}^n \rightarrow \{0,1\}^m$ is balanced, if Hamming Weight $HW(f) = 2^{n-1}$, which means that number of 1's and 0's in the truth table is equal. An S-box, which is balanced is one whose component Boolean functions and their linear combinations, are all balanced. The significance of the balance property is that the higher the magnitude of function imbalance, a high probability of linear approximation of the S-Box exists.

- **Non-Linearity (NL)**. Confusion property implies a complex and unpredictable relation between secret key and the ciphertext. Confusion property is embedded in a cipher design through use of nonlinear components specifically, the S-box. In order to check the resilience of an S-Box against cryptanalysis attacks, it is important to quantify the nonlinearity of the S-box. Nonlinearity requires that the S-box be not a linear mapping from input to output. This would make the cryptosystem susceptible to attacks [37].

- **Algebraic Degree or Complexity (AD).** The AD is a quantitative analysis of the strength of an S-Box against higher order differential and algebraic cryptanalyses. An S-box with a higher algebraic degree will be resilient against cryptanalytic attacks.

- **Strict Avalanche Criteria (SAC).**

Strict Avalanche Criteria implies that if we change single bit of S-box input, it must change in at least half of output bits. An S-Box that satisfies SAC will be secure against many families of attacks [38].

- **Correlation Immunity (CI).** A Boolean function f on $\boldsymbol{F_2^n}$ is said to be correlation immune of order $\boldsymbol{m}$, with 1 ≤ m ≤ n, if the output of f and any m input variables are statistically independent [39]. Higher the value of correlation immunity, the more resistant will be the S-Box.

- **Differential Uniformity (DU)**. Differential cryptanalysis was proposed by Biham and Shamir. It exploits the imbalance in XOR distribution among inputs and outputs of an S-Box. Differential uniformity means that inputs and outputs of an S-Box map uniquely with uniform probabilities. S-Boxes with smaller Differential Uniformity possess better resistance against differential cryptanalysis [40].

- **Linear Approximation (LA).** Attacks based on Linear cryptanalysis compute a linear approximation between bits of plaintext, ciphertext and the key. These estimates are approximated from linear approximation tables of the nonlinear non-linear elements in a block cipher, i.e., the S-box. In principle, lower the Linear Approximation value, the higher the resistance of S-box's against linear cryptanalysis [41].

- **Fixed (Fp) and Opposite Fixed Points (OFp).** Fixed point in an S-box means the points for which input directly maps to output, while opposite fixed points means that output is the complement of input. The number of fixed and opposite fixed points should be minimum possible in order to enhance resistance against statistical analysis [42].

- **Bit Independence Criterion (BIC).** Bit independence implies the independence of output bits on previous bits. The higher the bit independence, the more unpredictable and secure will be the S-box.

- **Confusion Coefficient Variance (CCV)**.

CCV is also a measure of resistance of an S-box against side channel analysis. This is a probabilistic model that allows the attacker to explore cipher design. Heuser et al. [43] in their research have concluded that higher the value of CCV, suggested that higher CCV value, the higher will be resistance level against side-channel analysis.

- **Signal-to-Noise Ratio (SNR) OR Differential Power Analysis (DPA)**. DPA is a category of side channel attacks that involves statistical analysis of power measurements obtained from a crypto system. These attacks exploit power consumption of different hardware components of a system using different keys. DPA uses signal processing and error correction attributes of a system to extract valuable intel about a system. An S-box with lower SNR (DPA) will be considered better resistant against side-channel analysis [44].

3.7 **Conclusion**. In this chapter, we have tried to build the foundation of in-depth study of block ciphers and specially the nonlinear component i.e., the S-box and its main properties. The list is inexhaustive since the mathematical properties of S-box cannot be covered in a short thesis. However, an effort has been made to take a note of important properties of S-box which have a serious impact on the security of S-box and the whole cipher. After this, a detailed account of invariant attacks and their countermeasures will be covered in the subsequent parts of this thesis.

CHAPTER 4

INVARIANT ATTACK AND DESIGN CRITERIA TO SAFEGUARD BLOCK CIPHERS AGAINST INVARIANT ATTACKS

4.1 Introduction

Major focus of the designers of symmetric-key for instance Substitution-Permutation-Networks (SPN) remains on design criteria for the substitution boxes (S-Box) of the cryptosystem. Main reason behind this fact is that S-Box is the only non-linear component of an SPN [45]. In case of a weak S-Box, the entire cipher will act linearly, and its security will be easily compromised through cryptanalysis [46]. The design and evaluation criteria of S-Box are interlinked processes since both the steps complement each other towards security of a block cipher [45]. Study of cryptanalytic attacks against full and lightweight versions of block ciphers is an ongoing process. The basic endeavour of this research is to consider the vulnerabilities in block ciphers exploited by Invariant Attacks with an aim to stipulate a design criterion to safeguard against this attack. The flow of work has been outlined in Fig 4.1 below.

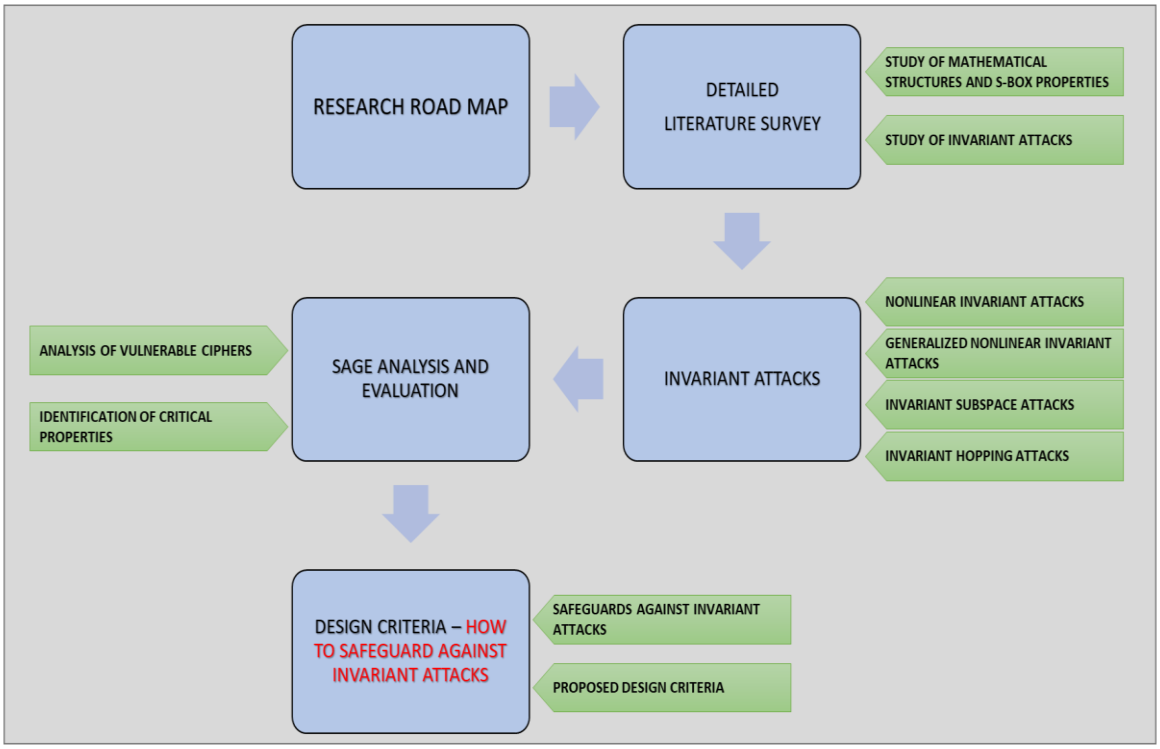

RESEARCH ROAD MAP
DETAILED LITERATURE SURVEY
STUDY OF MATHEMATICAL STRUCTURES AND S-BOX PROPERTIES
STUDY OF INVARIANT ATTACKS
INVARIANT ATTACKS
NONLINEAR INVARIANT ATTACKS
GENERALIZED NONLINEAR INVARIANT ATTACKS
INVARIANT SUBSPACE ATTACKS
INVARIANT HOPPING ATTACKS
SAGE ANALYSIS AND EVALUATION
ANALYSIS OF VULNERABLE CIPHERS
IDENTIFICATION OF CRITICAL PROPERTIES
DESIGN CRITERIA – HOW TO SAFEGUARD AGAINST INVARIANT ATTACKS
SAFEGUARDS AGAINST INVARIANT ATTACKS
PROPOSED DESIGN CRITERIA

Figure 4.1 Research Flow of Work

4.2 **Overview of Invariant Attack**. The word 'Invariant' literally means unchanging. In cryptography, invariant attack is a newly proposed attack system in which the properties of a block cipher round function are preserved (or remain unchanging) in arbitrary number of rounds. Suppose there is a block cipher defined by: $\boldsymbol{E_k : F_2^N \rightarrow F_2^N}$

Implementing an Invariant Attack on this block cipher is to find an efficiently computable Boolean function: $\boldsymbol{g : F_2^N \rightarrow F^N}$

such that $\boldsymbol{g(P) \oplus g\big(E_k(P)\big)}$ is constant for **any Plaint Text P** and for a **number of Keys**. Such keys are known as **weak keys**. The function ***g*** is known as **Non-Linear Invariant** for the cipher $\boldsymbol{E_K}$.

4.3 **Major Types of Invariant Attacks**. Till date, four main types of invariant attacks have come to light. Brief overview and attack methodology are elaborated in ensuing paragraphs.

4.3.1 Nonlinear Invariant Attack

4.3.1.1 **Background**. The nonlinear invariant attacks were first introduced in ASIACRYPT-2016 [8] as a new type of cryptanalytic technique carried out under weak key setting. It was developed from two main lines of research dating back to 1993. The schematic layout of the development ladder of the attack is explained in Figure 4.2 below.

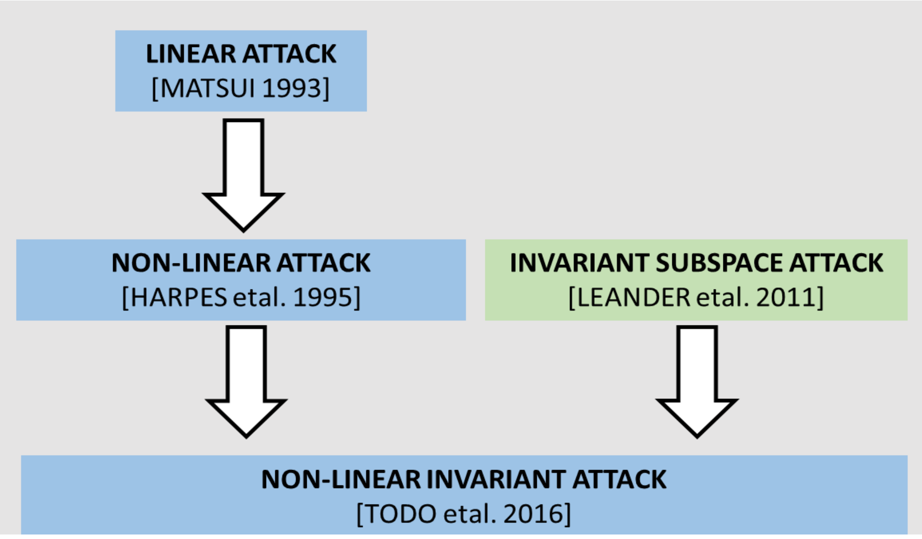


Figure 4.2 Research Chronology of Invariant Attacks

4.3.1.2 **Overview of Nonlinear Invariant Attack**. The nonlinear invariant attack is a practical type of attack against vulnerable block ciphers and can be considered as an advanced form of linear cryptanalysis. Initially it acts as a distinguishing attack, however, with suitable assumptions, the attack can be extended to a Ciphertext Only attack. This attack can be modeled against most of the existing block ciphers under specific assumptions, however, these have proved more effective against **key-alternating ciphers** and **substitution permutation networks (SPN) ciphers.** Suppose a typical block cipher consisting of round function **F** for ***i*** number of rounds. We have determined a Boolean Function ***g*** which is a nonlinear invariant for a single round. Now if, the keys used in each round are weak keys, the invariant function ***g*** will be invariant over arbitrary number of rounds. In such case, the nonlinear approximation is possible with probability 1.

4.3.1.3 **Basic Working of SPN Block Cipher**. Consider a key alternating block cipher represented by Round Function $R : FN^2 \rightarrow FN^2$.

Then, round function with key XOR can be expressed as under:

$$R_k(x) = R\,(\,x \oplus k)$$

For this cipher comprising ***r*** number of rounds, the ciphertext C will be computed as under:-

$$x_0 = P,$$

$$x_{i+1} = R_{ki}\,(x_i) = R(x_i \oplus k_i); \qquad 0 \leq i \leq r - 1,$$

$$C = x_r$$

4.3.1.4 **How Nonlinear Invariant Attack Works.** The basic premise of the invariant attack is to determine a nonlinear Boolean function *g* such that

$$g(R(x \oplus k)) = g(x \oplus k) \oplus c = g(x) \oplus g(k) \oplus c \oplus x$$

where *c* is a constant in F2. All such Keys for which this equation satisfies are known to be ***weak keys***, which the determined function ***g*** is called the ***nonlinear invariant.***

4.3.1.5 **Analysis of Application of Nonlinear Invariant Attack Against Practical Ciphers**.

- **What Sort of Ciphers are Vulnerable to Nonlinear Invariant Attack?** As observed, most lightweight block ciphers are designed for lightweight applications, therefore, they have a minimal or sometimes even non-existent key alternating schedule for example for the ciphers using same round key in each round. Since full scale ciphers like AES employ a complex key alternating function, nonlinear invariant attack has not been successful against such ciphers.
- **Weak Key Space**. Nonlinear Invariant Attacks work under the assumption of weak keys such that, for a Boolean function to be invariant over arbitrary number of rounds, all round keys must be weak.
- **Weak Round Constants**. In case the weak key condition is fulfilled (all round keys used are weak keys), then, suppose Boolean function ***g*** is invariant with respect to the round function ***R.*** Now, if all round constant $\boldsymbol{RC_i}$ are involved only in linear terms of the function ***g***, then, the function g becomes nonlinearly invariant to the round constant addition step. Such constants are called weak constants.
- **Non-Trivial Invariant Computation**. The attack unfolds sequentially, such that, initially, invariant function for each single round is computed individually and then combined mathematically. In case, the degree of the polynomial increases, the computation will become more and more complex thus increasing computation cost.
- **Analysis of Attack Against Ciphers SCREAM and iSCREAM**. Both these ciphers [8] are tweakable block ciphers and possess weak keys to the tune of 2^{96} and 2^{97} respectively. In case of a successful attack against SCREAM, the attacker can recover 32 bits from the

last data block in case the final block length is between 12 and 15 bytes. The vulnerability of this cipher stems from the L-function which is based on orthogonal matrix which can be manipulated. In case of iSCREAM, the weak key space is doubled to 2^{97} because there are two independent invariant functions for the cipher.

- **Attack Against Midori64**. Midori is a low energy block cipher [8] and has 2^{64} number of weak keys. If used in well-known operating modes like Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB) and Counter Mode, the attacker can successfully recover 32 bits in each data block of plaintext.

4.3.2 **Invariant Subspace Attack**.

4.3.2.1 **Overview of the Attack**. Invariant Subspace Attack was proposed by Leander et al. at CRYPTO 2011. It is similar in form to the nonlinear invariant attack and was proposed for cryptanalysis of a specific cipher known as PRINT cipher [47]. The core concept of the attack is the identify invariant subspaces in the key space which satisfy self-similarity property. These attacks are successful against lightweight ciphers employing weak or non-existent key alternating schedule.

4.3.2.2 **How Invariant Subspace Attack Works**. We assume that the round function of a cipher maps a Coset **A** of some vector subspace **U** of the inner state to a Coset **B** of the same space, and a fixed key belonging to **A - B** is added in every round. Then the set A is preserved by the round function, and hence remains stable through the whole encryption process. The process of preservation of subspace properties is explained in Figure 4.3 below:-

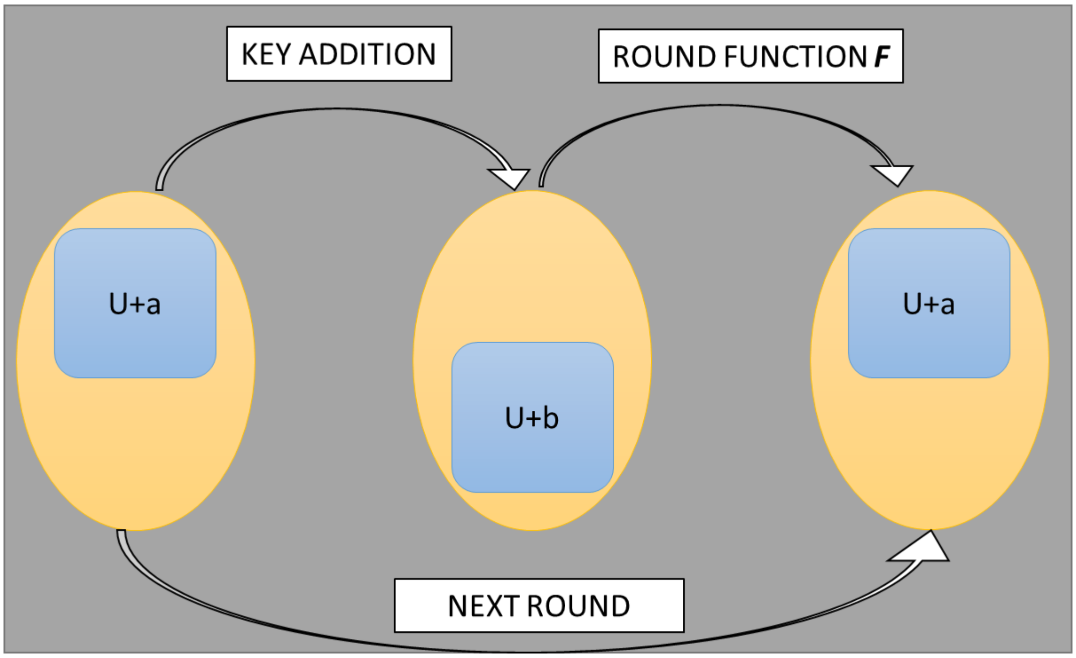


Figure 4.3 Layout of Invariant Subspace Attack

4.3.2.3 **Steps of the Attack**

- First, the input plaintext $\mathbf{x_i}$ is chosen from the subspace **U+a**.
- After key-XOR stage, this element maps to subspace **U+b**.
- Once the round function F is applied, the element again maps to subspace **U+a**.
- The resulting ciphertext will also belong in the subspace **U+a**.
- This way, the subspace **U+a** is preserved.

4.3.2.4 **Analysis of Application of Invariant Subspace Attack Against Practical Ciphers**.

- **Vulnerability**. Substitution Permutation Network based ciphers employing simple key schedule are vulnerable to subspace attacks. This type of attack seems particularly well-suited to substitution-permutation networks (SPN) with a minimal key schedule or cryptographic permutations with highly structured round constants.
- **Weak Key Space**. Like other variants of the invariant attack, this attack works on the basic assumption of working in the weak key space using weak round key in each round. strongest attack shows the existence of a weak key set of density 2^{-32}. This weak key space apparently seems low compared to the nonlinear invariant attack, however, once found,

the weak keys will lead to the plaintext going through all rounds of encryption with probability of 1.

- **Round Constants**. Block ciphers employing permutation functions with well-structured round constants are especially vulnerable to subspace invariant attacks.

- **Invariant Subspace Size**. The attack is more successful against ciphers with large invariant subspaces.

- **Outcome of Invariant Subspace Attack against ROBIN, ZORRO and iSCREAM**. Invariant subspace attack successfully identifies weak keys in Robin, Zorro and iScream in a chosen tweak scenario. In related key setting, these ciphers are easily broken without weak-key requirement even.

- **Outcome of Invariant Subspace Attack against LED, Noekon and Fantomas Ciphers**. Once these ciphers were attacked, no invariant subspaces could be identified in their structure, thus these ciphers were found safe against this attack.

- **Outcome of Invariant Subspace Attack against LS-Design Ciphers**. LS design-based ciphers were found resistant against this category of attack.

- **Outcome of Invariant Subspace Attack against Midori64 Cipher**. Midori64 cipher was found especially vulnerable to subspace attack because of weak combinations of three factors including round constants, S-box, and existence of orthogonal matrix in the linear layer. Once the conditions are present, the cipher can be distinguished with single query and key can be recovered in 2^{16} time complexity using two queries. In Midori64, use of round constants is tricky, if not chosen wisely, weak round constants will add to the vulnerability of the cipher. In order to eliminate the vulnerability of weak round constants, the S-Boxes may be tailored in line with the key schedule.

- **Conditions for the Use of S-Boxes**. The ciphers with identical S-Boxes in each round are broken easily. This fact has been proven by the fact that the attack could not be launched against Midori128 because of the use of four different S-Boxes in the round.
- **The Case of Involution and Non-Involution S-Boxes**. With simple key schedule, involution type S-Boxes can be considered less secure compared to non-involution type S-Boxes.

4.3.3 **Generalized Nonlinear Invariant Attack**

4.3.3.1 **Basic Idea of the Attack**. The concept of Generalized Nonlinear Invariant attack originates from basic Nonlinear Invariant Attack [48]. Although the nonlinear invariant attack was successful against full block ciphers like Midori, Scream and iScream, its major drawback was with the choice of round constants [49]. If round constants were chosen wisely to nonlinearly affect the invariants, the attack would be impossible to mount. The generalized nonlinear invariant attack overcomes the issue of round constants by utilizing a pair of constants in the input of the invariants.

4.3.3.2 **Attack Methodology of Generalized Nonlinear Invariant Attacks**.

Suppose a standard block cipher which inputs Plaintext P and round subkey K_i to output Ciphertext C. The input output relation can be expressed as under:-

$$x_0 = P,$$

$$x_{i+1} = FK_i(x_i) = F(x_i) \oplus K_i,$$

$$x_r = C$$

The generalized nonlinear invariant attacks evaluate a nonlinear Boolean function

$g : GF(2)^n \rightarrow GF(2)$ along with two n-bit constants $(a1, a2) \in GF(2)n \times GF(2)^n$ such that:-

$$g(x \oplus a_1) \oplus g(F_{Ki}(x) \oplus a_2) = c$$

(where *c* is a binary constant) holds for any *x*.

In order to compute generalized nonlinear invariant of a cipher, attacker needs set of weak keys for each round along with pair of constants (*a*1*, a*2) contained in nonlinear terms of the nonlinear invariant *g*(*x*).

4.3.3.3 **Analysis of Generalized Nonlinear Invariant Attacks against iSCREAM Cipher**. The efficiency of newly proposed Generalized Nonlinear Invariant Attack can be carried out with respect to iSCREAM cipher which was also attacked by basic nonlinear invariant attack. The criteria of round constants proposed by Beierle *et al*. [23] makes the cipher resistant against nonlinear invariant attack proposed by Todo *et al*. [8] In contrast, same criteria do not suffice against generalized nonlinear invariant attacks. In these attacks, concept of closed loop invariants of the substitution layer creates a linear structure in the S-Box invariant which can be exploited. Instead of creating nonlinear resistance, round constants can even create a vulnerability for generalized nonlinear invariant attack to be successful.

4.3.4 **Invariant Hopping Attack**.

4.3.4.1 **Basic Idea of the Attack**. Available invariant attacks exploit vulnerability in the iterated round structure of the block ciphers, however, until recently, only few of such attacks were launched with impossibility results [50]. Available invariant attacks have been able to construct polynomial invariants of degree 2 only [51]. Invariant hopping attacks were proposed in order to construct stronger invariants. These attacks work for entire key space by exploiting roots of the fundamental equation and have success probability of 1 [52]. Hopping attacks make use of the existing invariant methodology by upgrading a simple attack against a weak cipher to a higher degree invariant attack using a step-by-step approach. This step-by-step approach is known as Invariant Hopping.

4.3.4.2 **Attack Methodology of Invariant Hopping Attacks against T-310 Block Cipher**. Invariant Hopping attack was launched on a classical block cipher T-310 which was actually employed by US Government during Cold War era. It had a block size of 36 bits and key length of 240 bits. In first step single round of encryption was represented in the form of 36 Boolean Polynomials of degree 6. The technique adopted to solve the Fundamental Equation for the cipher uses simple relation: ***P*(*Inputs*) = *P*(*Outputs*)**

The same is mathematically equivalent to

$$\boldsymbol{P(a; b; c; d; e; f; g; h; \ldots\ldots) = P(b; c; d; F + i; f; g; h; F + Z1 + e; \ldots..)}.$$

In the next step, the equations are translated into their ANF by using a new set of variables. Technique of attack hopping is employed by transforming attack on one cipher into another cipher [36].

4.3.4.3 **Analysis of Invariant Hopping Attacks against T-310 Block Cipher.** The purpose of choice of this cipher was that its internal wiring was flexible. The invariant hopping attacks attack a weak cipher to break stronger ciphers. Here it was demonstrated by manipulating the internal wiring of the cipher to own advantage. In this way, the complexity and degree of Boolean function is being enhanced progressively. Using this technique, complex invariants of degree 8 can be constructed methodically [53].

4.4 **Analysis of Vulnerable Lightweight Ciphers Using SAGEMATH Tool**. There has been extensive research in the field of cryptography. New ciphers are exposed to cryptanalysis techniques in order to test their security credibility. Various cryptanalytic tools are employed to determine properties of various components of ciphers contributing towards their security. SageMath is a state-of-the-art open-source math-based utility which has well developed cryptographic tools in addition to numerous other applications [54]. SageMath provides functionality of multiple cryptographic functions using few lines of code. SageMath was utilized to determine important properties of S-Boxes of vulnerable ciphers, list of reviewed properties is as under:-

- Differential Uniformity (DU).
- Fixed Points and opposite fixed points (F.P).
- Linear Structure or not (L.S).
- Balanced function or not (Bal).
- Bent function (Bent).
- Involution function or not (Inv).
- Linearity (Lin).
- Non-Linearity (NL).
- Maximum Degree (Max.D).
- Minimum Degree (Min.D).
- Maximum Differential Probability (M.D.P).
- Maximum Linear Bias (M.L.B).
- S-Box Invariants (newly introduced functions which takes input of S-Box and provides in output all possible invariants of the S-Box.

4.4.1 **SageMath Analysis**. Basic aim of this study is to analyse various properties of S-Boxes that impact their resilience against invariant attacks. For this purpose, ten lightweight ciphers that had been cryptanalysed by various variants of invariant attack were shortlisted. These ciphers had S-Boxes of configuration 3x3, 4x4 and 8x8 respectively. At the end of Sage analysis, we shall be able to draw certain conclusions which will lead us to formulation of a security toolkit to assure resilience of an S-Box against invariant attacks.

The output of various functions applied to target ciphers is as under:-

Ser	CIPHER	DU	F.P	L.S	BAL	BENT	INV	LIN	NL	MAX. D	MIN. D	M.D.P	M.L.B
3x3 S-Box													
1.	**PRINT**	2	[0,1]	T	T	F	F	4	2	2	2	0.25	2
2.	**SEA**	2	[0,4]	T	T	F	F	4	2	2	2	0.25	2
4x4 S-Box													
3.	**PRESENT**	4	No	T	T	F	F	8	4	3	2	0.25	4
4.	**PRINCE**	4	No	F	T	F	F	8	4	3	3	0.25	4
5.	**ELEPHANT**	4	No	T	T	F	F	8	4	3	2	0.25	4
6.	**MIDORI**	4	[3,7, 8,9]	T	T	F	T	8	4	3	2	0.25	4
8x8 S-Box													
7.	**iSCREAM**	8	[*]	T	T	F	T	64	96	6	4	0.0625	32
8.	**SCREAM**	8	[ ]	T	T	F	F	64	96	6	3	0.0312	32
9.	**ZORRO**	10	[ ]	F	T	F	F	64	96	7	5	0.0390	32
10.	**SKINNY_8**	64	[255]	T	T	F	F	128	64	6	2	0.25	64

Note: Properties names have been coded as defined in Section 4.4 above; in addition, True=T and False=F.

[*] = [0, 24, 38, 61, 69, 95, 103, 124, 132, 158, 162, 179, 201, 209, 235, 250]

Table: 4.1 Important SageMath Properties of Vulnerable Ciphers Vulnerable to Invariant Attacks

4.4.2 **Computation of Nonlinear Invariants for S Box in SageMath**. In line with Nonlinear Invariant Attack proposed by Todo *et al*. in [22], a patch was introduced in SageMath [54] to compute nonlinear invariants for an S-Box. For an mxm S-Box S, the attack attempts to compute m-variables Boolean functions **g** such that ***g(x) + g(S(x))*** is a constant function. The implementation of this patch is based on the method proposed by authors in Section 3.1 of [22]. It was a very useful tool, however, it had certain compatibility / operation issues in the latest version of SageMath i.e., 9.3. After necessary modification with help of our Thesis Advisor, the patch was made functional in the current SageMath version. The function code to compute nonlinear invariants is opensource and has been copied at ***Appendix-A***.

4.5 **Conclusions from Sage Analysis**. At this stage, no wholesome tool set is available to analyse entire cipher with regards to invariant attacks, SageMath tool can help us in analyzing individual components with respect to certain important properties so that we can comment on the behaviour of the ciphers. Salient outcomes of the above tabulated Sage analysis are as under:-

- **Impact of Balancedness on Invariant Attack**. All vulnerable S-Boxes display the property of balance. Therefore, where trait of being balanced reveals vulnerability towards linear and differential cryptanalysis, it casts no major influence against invariant attacks.

- **Impact of Number and Location of Fixed Points in an S-Box**. Fixed point means that input bit on that specific position of S-Box gives out the same bit in output. Whereas, if output is the compliment of input, it is opposite fixed point. This is an important property which impacts the security of S-Box. We shall study the impact of number and location of fixed points on the security of a S-Box in relation to the invariant attack. As per understanding, the less the number of fixed points, the better the security and vice versa.

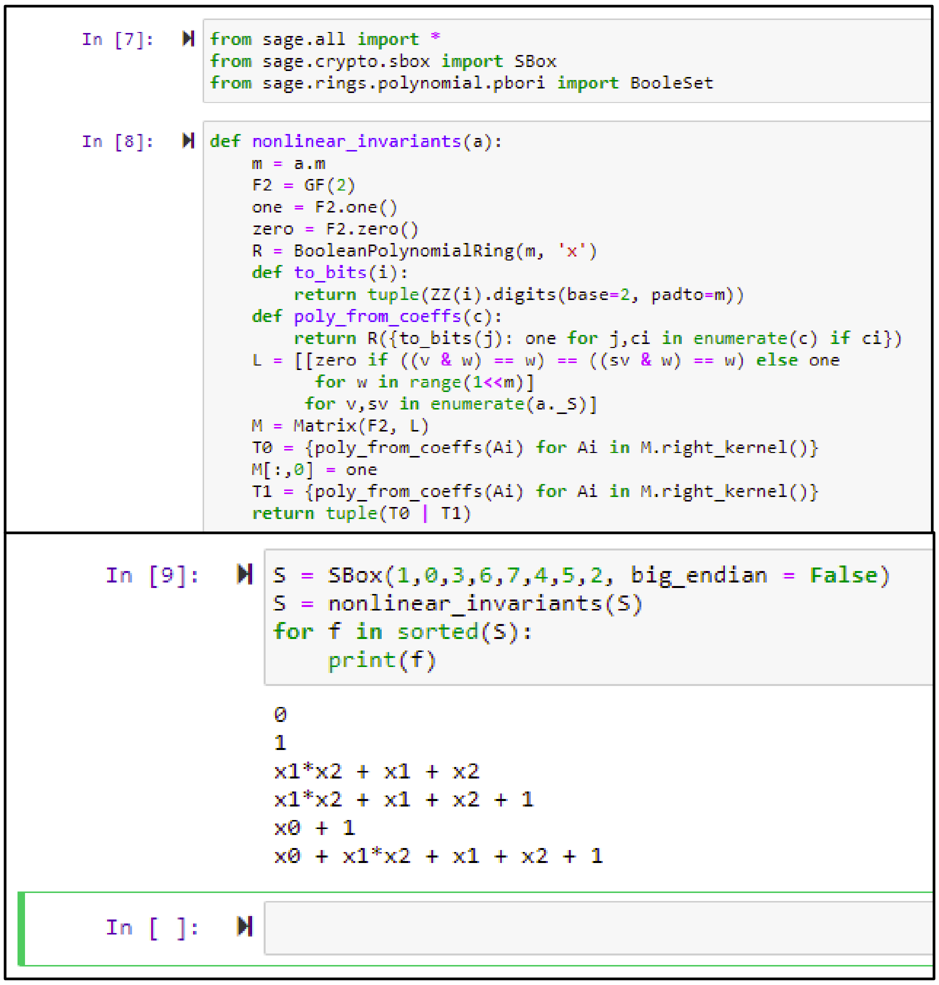
We are going to use the nonlinear invariant function built in the **SageMath** tool to confirm our point. Let us take the example of a hypothetical 3-bit S-Box. We shall start taking number of fixed points from 0 uptill 7. Input S-Box with varying fixed points are depicted in **Table 4.2** through **Table 4.9** below. Output of the non-linear invariant function with different number of fixed points is depicted in **Fig 4.4** through **Fig 4.11** below:-

- <u>**No Fixed Point**</u>

x	0	1	2	3	4	5	6	7
S(x)	1	0	3	6	7	4	5	2

Table: 4.2 S-box with No Fixed Points

<u>**Output of Invariant Function**</u>



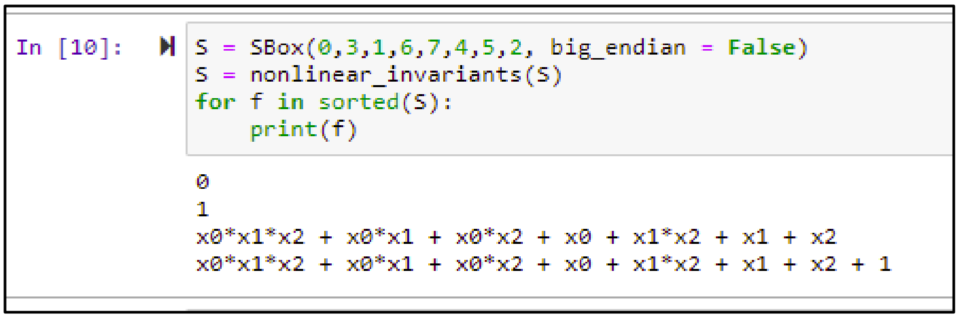
```
In [7]:  from sage.all import *
         from sage.crypto.sbox import SBox
         from sage.rings.polynomial.pbori import BooleSet

In [8]:  def nonlinear_invariants(a):
             m = a.m
             F2 = GF(2)
             one = F2.one()
             zero = F2.zero()
             R = BooleanPolynomialRing(m, 'x')
             def to_bits(i):
                 return tuple(ZZ(i).digits(base=2, padto=m))
             def poly_from_coeffs(c):
                 return R({to_bits(j): one for j,ci in enumerate(c) if ci})
             L = [[zero if ((v & w) == w) == ((sv & w) == w) else one
                   for w in range(1<<m)]
                  for v,sv in enumerate(a._S)]
             M = Matrix(F2, L)
             T0 = {poly_from_coeffs(Ai) for Ai in M.right_kernel()}
             M[:,0] = one
             T1 = {poly_from_coeffs(Ai) for Ai in M.right_kernel()}
             return tuple(T0 | T1)

In [9]:  S = SBox(1,0,3,6,7,4,5,2, big_endian = False)
         S = nonlinear_invariants(S)
         for f in sorted(S):
             print(f)

         0
         1
         x1*x2 + x1 + x2
         x1*x2 + x1 + x2 + 1
         x0 + 1
         x0 + x1*x2 + x1 + x2 + 1

In [ ]:
```

Figure: 4.4 S-box with No Fixed Points

- **1 Fixed Point**

x	0	1	2	3	4	5	6	7
S(x)	0	3	1	6	7	4	5	2

Table: 4.3 S-box with 1 Fixed Points



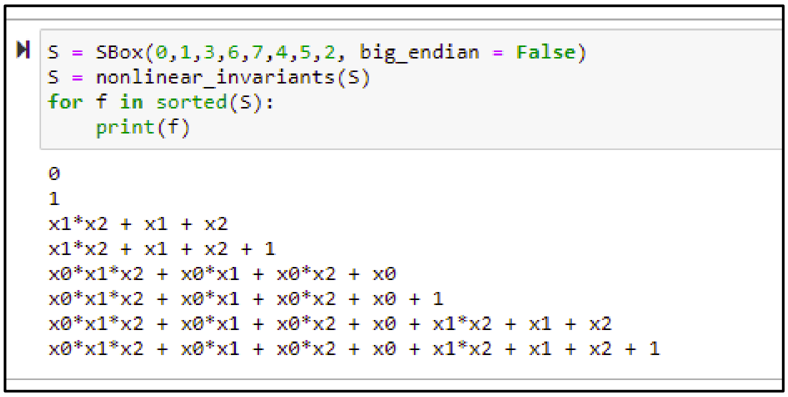
```
In [10]:  S = SBox(0,3,1,6,7,4,5,2, big_endian = False)
          S = nonlinear_invariants(S)
          for f in sorted(S):
              print(f)

          0
          1
          x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
          x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.5 S-box with 1 Fixed Points

- **2 Fixed Point**

x	0	1	2	3	4	5	6	7
S(x)	0	1	3	6	7	4	5	2

Table: 4.4 S-box with 2 Fixed Points



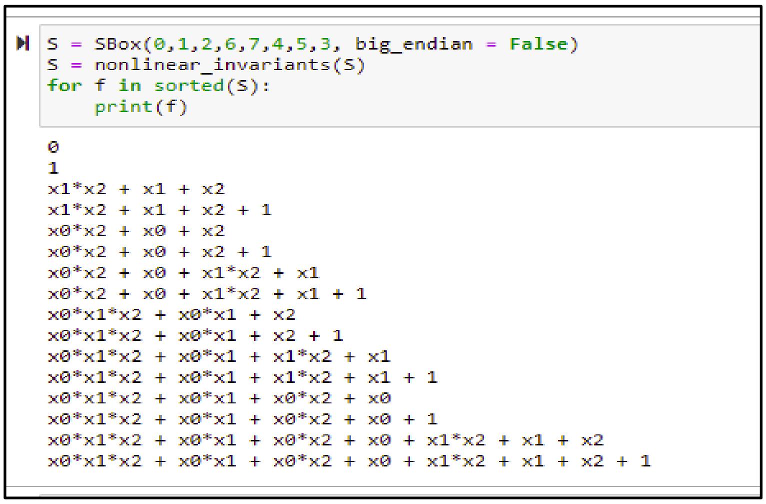
```
S = SBox(0,1,3,6,7,4,5,2, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)

0
1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.6 S-box with 2 Fixed Points

- **3 Fixed Point**

x	0	1	2	3	4	5	6	7
S(x)	0	1	2	6	7	4	5	3

Table: 4.5 S-box with 3 Fixed Points



```
S = SBox(0,1,2,6,7,4,5,3, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

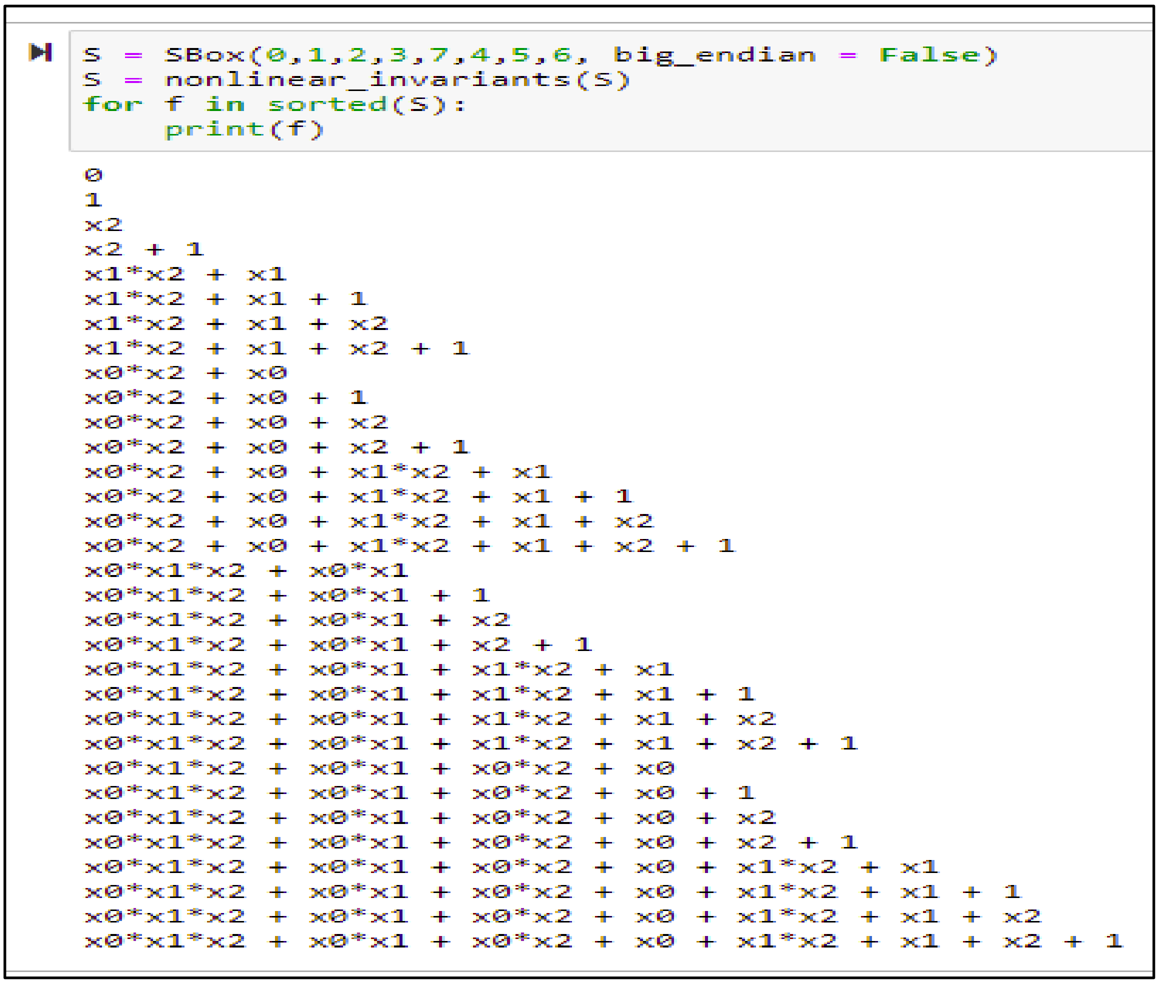
```
0
1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x2 + x0 + x2
x0*x2 + x0 + x2 + 1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x2
x0*x1*x2 + x0*x1 + x2 + 1
x0*x1*x2 + x0*x1 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.7 S-box with 3 Fixed Points

- **<u>4 Fixed Point</u>**

x	0	1	2	3	4	5	6	7
S(x)	0	1	2	3	7	4	5	6

Table: 4.6 S-box with 4 Fixed Points



```
S = SBox(0,1,2,3,7,4,5,6, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

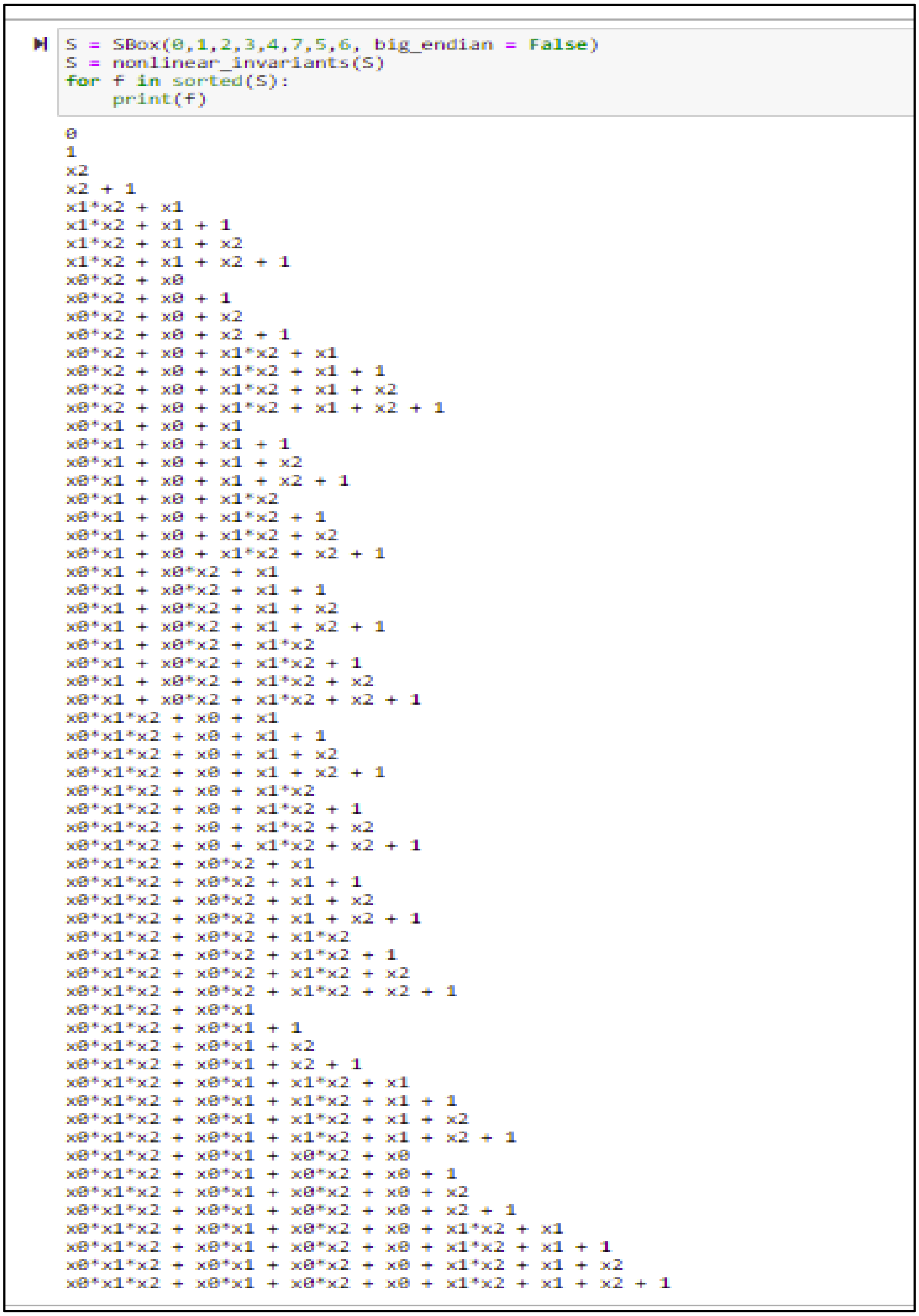
```
0
1
x2
x2 + 1
x1*x2 + x1
x1*x2 + x1 + 1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x2 + x0
x0*x2 + x0 + 1
x0*x2 + x0 + x2
x0*x2 + x0 + x2 + 1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x2 + x0 + x1*x2 + x1 + x2
x0*x2 + x0 + x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1
x0*x1*x2 + x0*x1 + 1
x0*x1*x2 + x0*x1 + x2
x0*x1*x2 + x0*x1 + x2 + 1
x0*x1*x2 + x0*x1 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.8 S-box with 4 Fixed Points

- **5 Fixed Point**

x	0	1	2	3	4	5	6	7
S(x)	0	1	2	3	4	7	5	6

Figure: 4.7 S-box with 5 Fixed Points



```
S = SBox(0,1,2,3,4,7,5,6, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

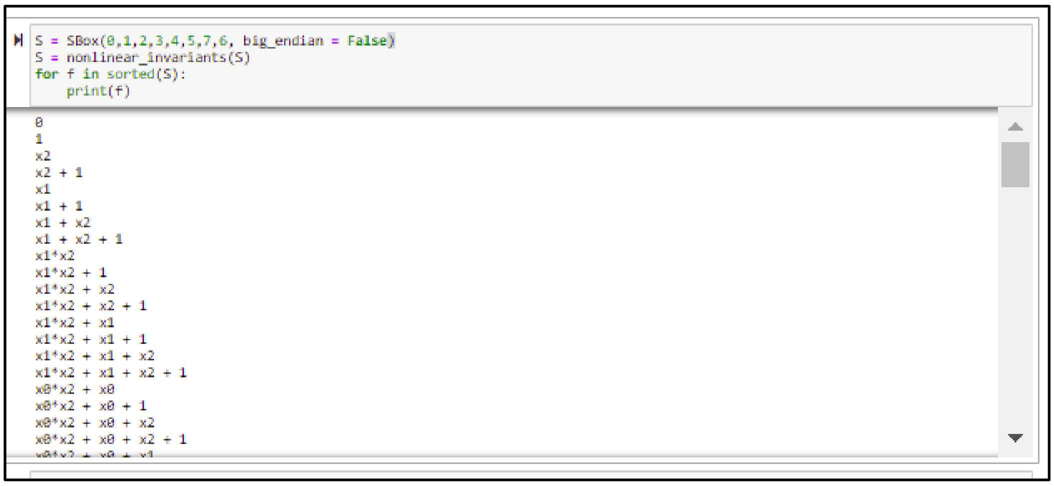
```
0
1
x2
x2 + 1
x1*x2 + x1
x1*x2 + x1 + 1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x2 + x0
x0*x2 + x0 + 1
x0*x2 + x0 + x2
x0*x2 + x0 + x2 + 1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x2 + x0 + x1*x2 + x1 + x2
x0*x2 + x0 + x1*x2 + x1 + x2 + 1
x0*x1 + x0 + x1
x0*x1 + x0 + x1 + 1
x0*x1 + x0 + x1 + x2
x0*x1 + x0 + x1 + x2 + 1
x0*x1 + x0 + x1*x2
x0*x1 + x0 + x1*x2 + 1
x0*x1 + x0 + x1*x2 + x2
x0*x1 + x0 + x1*x2 + x2 + 1
x0*x1 + x0*x2 + x1
x0*x1 + x0*x2 + x1 + 1
x0*x1 + x0*x2 + x1 + x2
x0*x1 + x0*x2 + x1 + x2 + 1
x0*x1 + x0*x2 + x1*x2
x0*x1 + x0*x2 + x1*x2 + 1
x0*x1 + x0*x2 + x1*x2 + x2
x0*x1 + x0*x2 + x1*x2 + x2 + 1
x0*x1*x2 + x0 + x1
x0*x1*x2 + x0 + x1 + 1
x0*x1*x2 + x0 + x1 + x2
x0*x1*x2 + x0 + x1 + x2 + 1
x0*x1*x2 + x0 + x1*x2
x0*x1*x2 + x0 + x1*x2 + 1
x0*x1*x2 + x0 + x1*x2 + x2
x0*x1*x2 + x0 + x1*x2 + x2 + 1
x0*x1*x2 + x0*x2 + x1
x0*x1*x2 + x0*x2 + x1 + 1
x0*x1*x2 + x0*x2 + x1 + x2
x0*x1*x2 + x0*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x2 + x1*x2
x0*x1*x2 + x0*x2 + x1*x2 + 1
x0*x1*x2 + x0*x2 + x1*x2 + x2
x0*x1*x2 + x0*x2 + x1*x2 + x2 + 1
x0*x1*x2 + x0*x1
x0*x1*x2 + x0*x1 + 1
x0*x1*x2 + x0*x1 + x2
x0*x1*x2 + x0*x1 + x2 + 1
x0*x1*x2 + x0*x1 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.9 S-box with 5 Fixed Points

- **6 Fixed Point**

x	0	1	2	3	4	5	6	7
S(x)	0	1	2	3	4	5	7	6

Table: 4.8 S-box with 6 Fixed Points



```
S = SBox(0,1,2,3,4,5,7,6, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

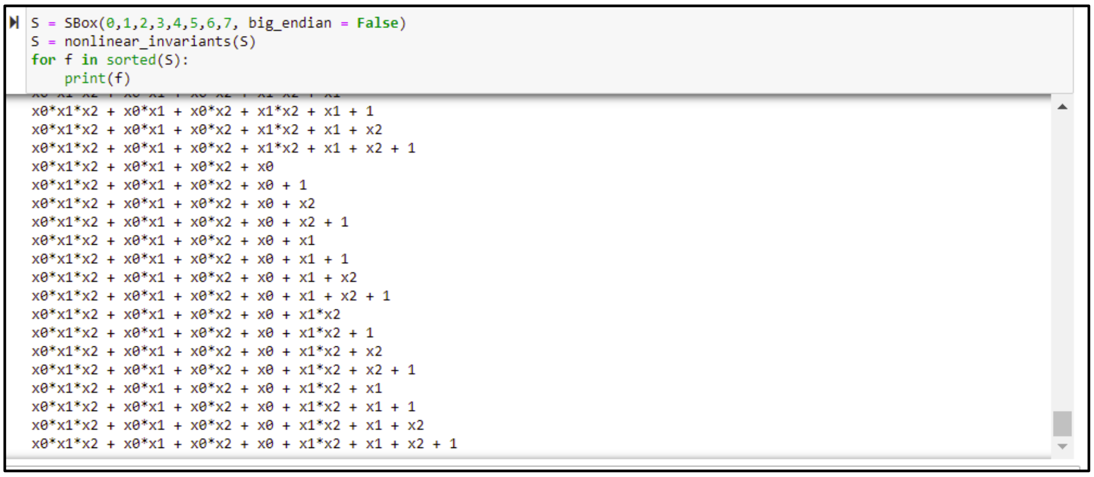
```
0
1
x2
x2 + 1
x1
x1 + 1
x1 + x2
x1 + x2 + 1
x1*x2
x1*x2 + 1
x1*x2 + x2
x1*x2 + x2 + 1
x1*x2 + x1
x1*x2 + x1 + 1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x2 + x0
x0*x2 + x0 + 1
x0*x2 + x0 + x2
x0*x2 + x0 + x2 + 1
x0*x2 + x0 + x1
```

Figure: 4.10 S-box with 6 Fixed Points

- **8 Fixed Point**

x	0	1	2	3	4	5	6	7
S(x)	0	1	2	3	4	5	6	7

Table: 4.9 S-box with 8 Fixed Points



```
S = SBox(0,1,2,3,4,5,6,7, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

```
x0*x1*x2 + x0*x1 + x0*x2 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x0*x2 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.11 S-box with 8 Fixed Points

- **Summary of Results**

Ser	No. of Fixed Points	No. of Invariants
1.	0	6
2.	1	4
3.	2	8
4.	3	16
5.	4	32
6.	5	64
7.	6	128
8.	8	256

Table: 4.10 Summary of Results No of Fixed Pts Vs No. of Invariants

- **Graph**

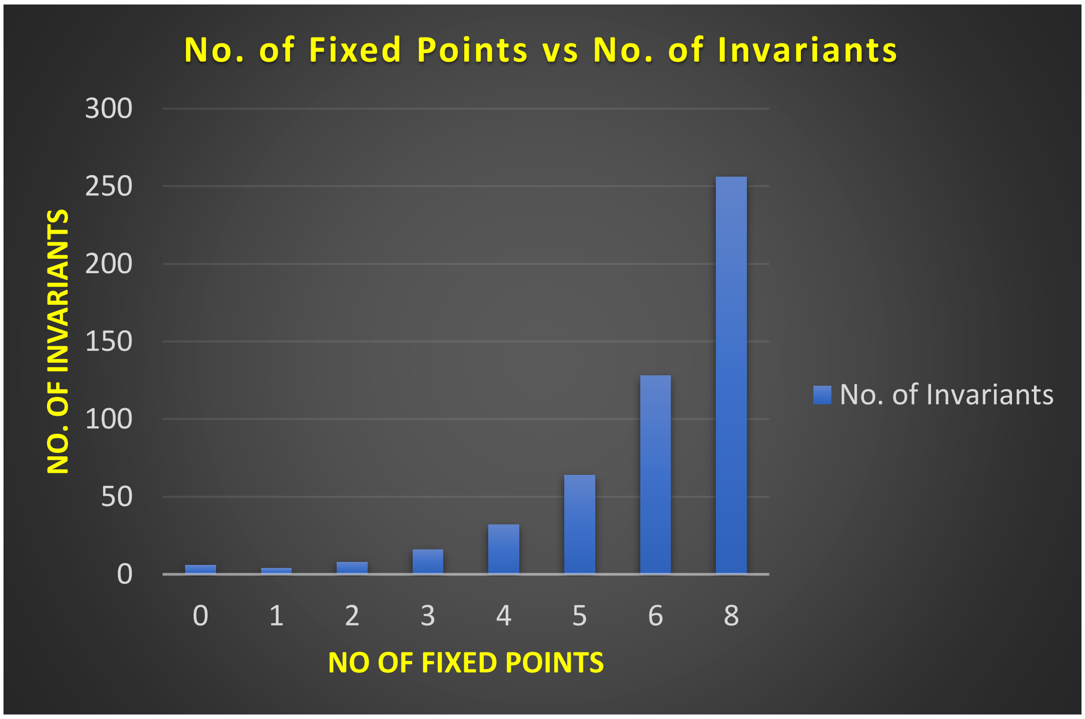


Figure: 4.12 Summary of Results No of Fixed Pts Vs No. of Invariants

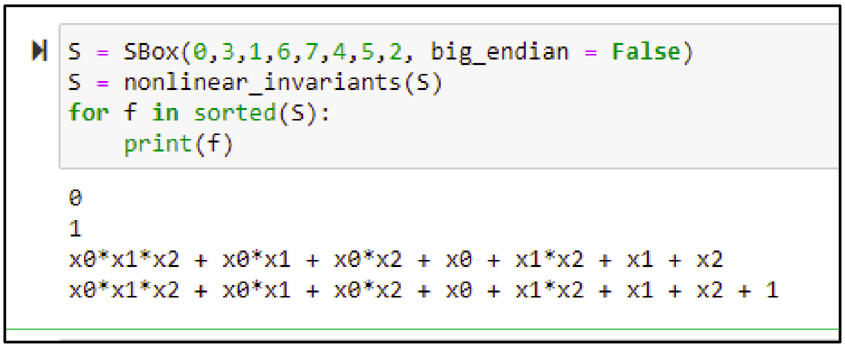
- **Inference**. The **Table 4.10** and graph in **Fig 4.12** clearly indicate that as we increase the number of fixed points in an S-Box, the possible number of invariants for that S-Box also increase exponentially. Therefore, in order to reduce the vulnerability, the number of fixed points must be kept to the minimum.

- **Impact of Location of Fixed Point on Security of S-Box.** Input S-Box with varying location of fixed points are depicted in **Table 4.11** to **Table 4.18** below. Output of the non-linear invariant function with different number of fixed points is depicted in **Fig 4.13** through **Fig 4.20** below:-

  - **Fixed Point at 0**

x	0	1	2	3	4	5	6	7
S(x)	0	3	1	6	7	4	5	2

Table: 4.11 S-box with Fixed Point at '0' Location



```
S = SBox(0,3,1,6,7,4,5,2, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

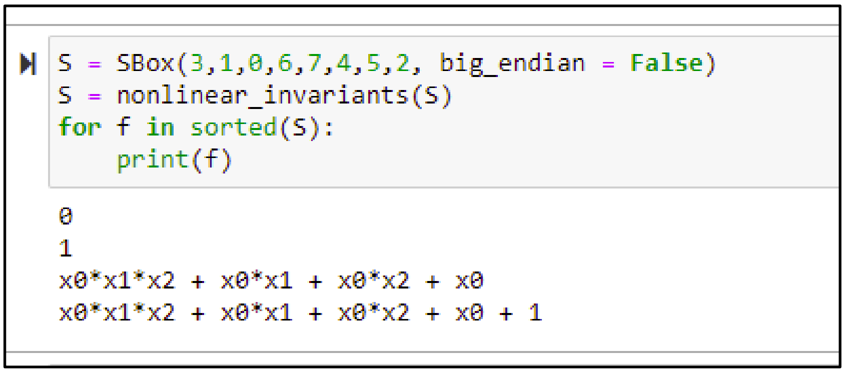
```
0
1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.13 S-box with Fixed Point at '0' Location

  - **Fixed Point at 1**

x	0	1	2	3	4	5	6	7
S(x)	3	1	0	6	7	4	5	2

Table: 4.12 S-box with Fixed Point at '1' Location



```
S = SBox(3,1,0,6,7,4,5,2, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

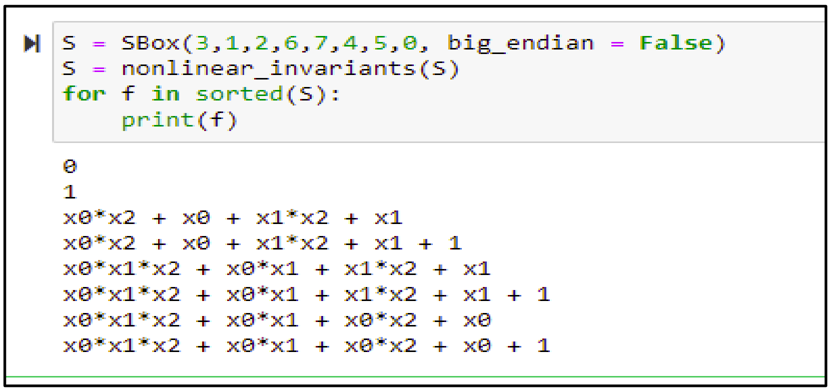
```
0
1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
```

Figure: 4.14 S-box with Fixed Point at '1' Location

  - **Fixed Point at 2**

x	0	1	2	3	4	5	6	7
S(x)	3	6	2	16	7	4	5	0

Table: 4.13 S-box with Fixed Point at '2' Location



```
S = SBox(3,1,2,6,7,4,5,0, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

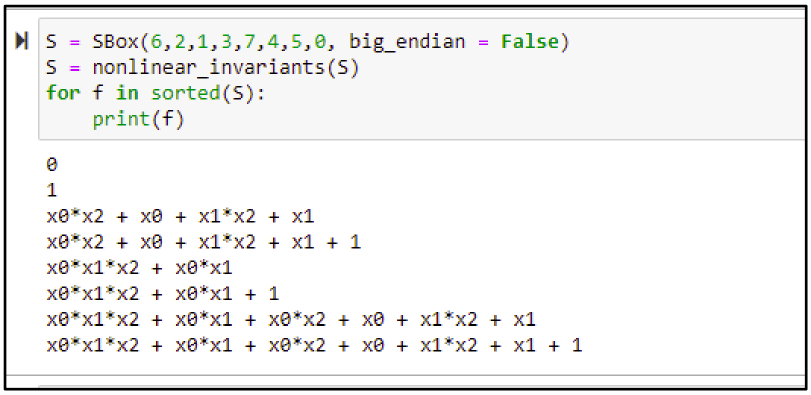
```
0
1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
```

Figure: 4.15 S-box with Fixed Point at '2' Location

- **Fixed Point at 3**

x	0	1	2	3	4	5	6	7
S(x)	6	2	1	3	7	4	5	0

Table: 4.14 S-box with Fixed Point at '3' Location



```
S = SBox(6,2,1,3,7,4,5,0, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

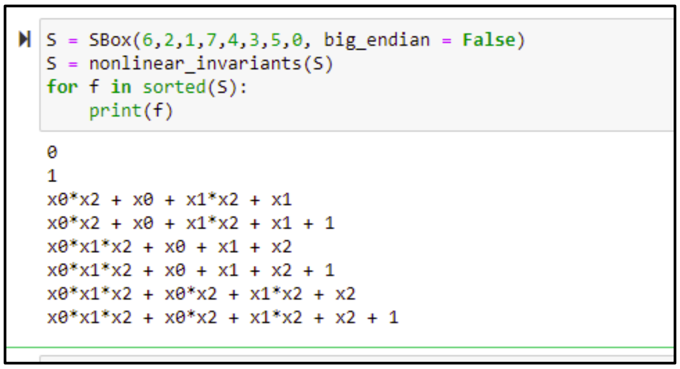
```
0
1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x1
x0*x1*x2 + x0*x1 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + 1
```

Figure: 4.16 S-box with Fixed Point at '3' Location

- **Fixed Point at 4**

x	0	1	2	3	4	5	6	7
S(x)	6	2	1	7	4	3	5	0

Table: 4.15 S-box with Fixed Point at '4' Location



```
S = SBox(6,2,1,7,4,3,5,0, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

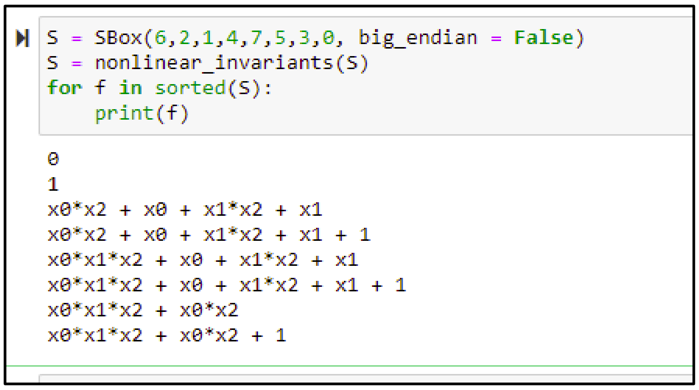
```
0
1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0 + x1 + x2
x0*x1*x2 + x0 + x1 + x2 + 1
x0*x1*x2 + x0*x2 + x1*x2 + x2
x0*x1*x2 + x0*x2 + x1*x2 + x2 + 1
```

Figure: 4.17 S-box with Fixed Point at '4' Location

- **Fixed Point at 5**

x	0	1	2	3	4	5	6	7
S(x)	6	2	1	4	7	5	3	0

Table: 4.16 S-box with Fixed Point at '5' Location



```
S = SBox(6,2,1,4,7,5,3,0, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

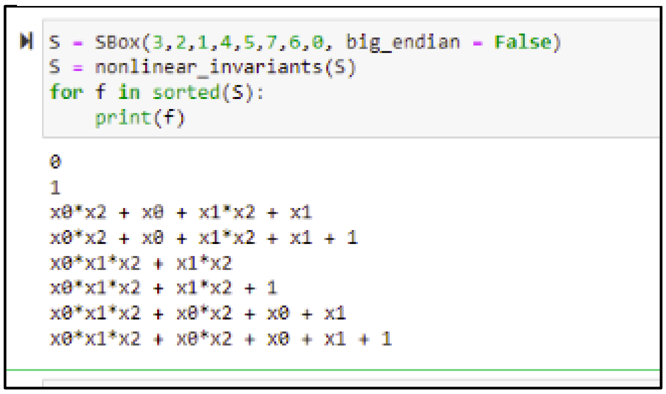
```
0
1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0 + x1*x2 + x1
x0*x1*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x0*x2
x0*x1*x2 + x0*x2 + 1
```

Figure: 4.18 S-box with Fixed Point at '5' Location

- **Fixed Point at 6**

x	0	1	2	3	4	5	6	7
S(x)	3	2	1	4	5	7	6	0

Table: 4.17 S-box with Fixed Point at '6' Location



```
S = SBox(3,2,1,4,5,7,6,0, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

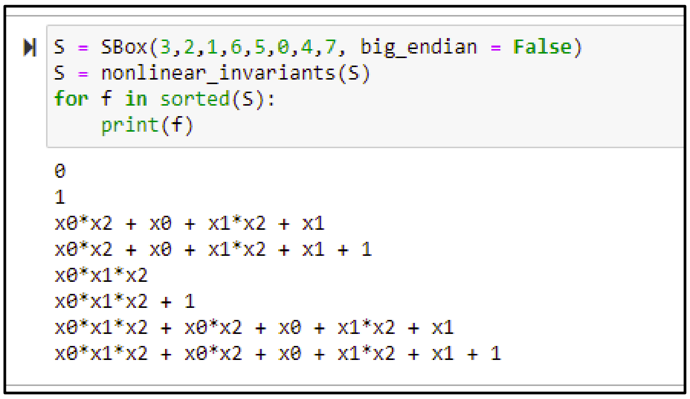
```
0
1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2 + x1*x2
x0*x1*x2 + x1*x2 + 1
x0*x1*x2 + x0*x2 + x0 + x1
x0*x1*x2 + x0*x2 + x0 + x1 + 1
```

Figure: 4.19 S-box with Fixed Point at '6' Location

- **Fixed Point at 7**

x	0	1	2	3	4	5	6	7
S(x)	3	2	1	6	5	0	4	7

Table: 4.18 S-box with Fixed Point at '7' Location



```
S = SBox(3,2,1,6,5,0,4,7, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

```
0
1
x0*x2 + x0 + x1*x2 + x1
x0*x2 + x0 + x1*x2 + x1 + 1
x0*x1*x2
x0*x1*x2 + 1
x0*x1*x2 + x0*x2 + x0 + x1*x2 + x1
x0*x1*x2 + x0*x2 + x0 + x1*x2 + x1 + 1
```

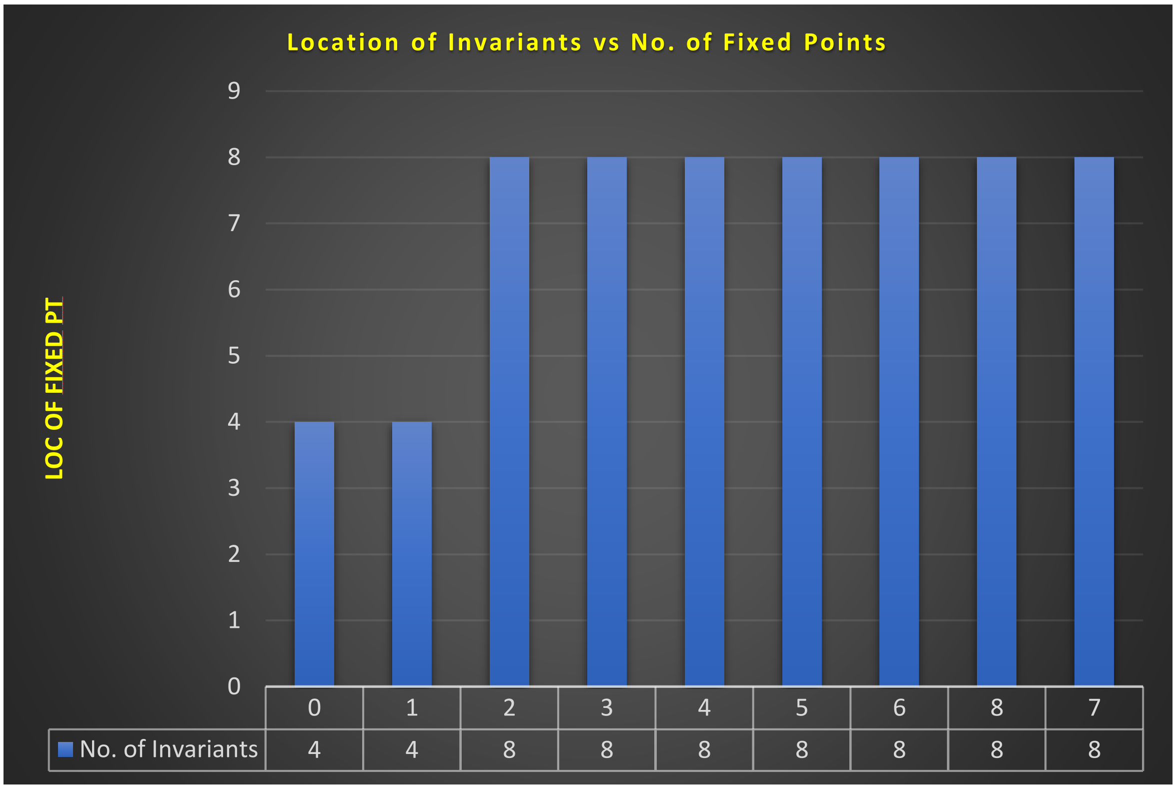
Figure: 4.20 S-box with Fixed Point at '7' Location

- **Summary of Results**

Ser	Location of Fixed Points	No. of Invariants
1.	0	4
2.	1	4
3.	2	8
4.	3	8
5.	4	8
6.	5	8
7.	6	8
8.	8	8

Table: 4.19 Summary of Results Location of Fixed Pts Vs No. of Invariants

- **Graph**



	0	1	2	3	4	5	6	8	7
■ No. of Invariants	4	4	8	8	8	8	8	8	8

Figure: 4.21 Summary of Results Location of Fixed Pts Vs No. of Invariants

- **Inference**. **Table 4.19** and graph in **Fig 4.21** clearly indicate that as we advance the location of fixed points in an S-Box, the possible number of invariants is not affected.

- **Impact of Use of Involution Type S-Boxes**. Most S-Boxes attacked by various types of Invariant attacks possess the property of being non-involution except Midori and iScream. Therefore, using majority vote system, it may be beneficial to use involution S-Boxes to guard against Invariant attacks.

- **Impact of Nonlinearity**. Most S-Boxes that were found vulnerable against invariant attacks possessed a linear structure less Prince and Zorro. It may be concluded that to safeguard against invariant attacks, non-linear S-Boxes will stand stronger chances of security.

- **Impact of Bent Functions**. None of the S-Boxes attacked were based on bent functions.

4.6 Safeguarding Against Invariant Attacks

4.6.1 **Methods to Enhance Resistance Against Non-Linear Invariant Attacks**. The question of whether or not a cipher is secure against invariant attacks, is of immense importance. The number of possible invariants for a given round function might be large. Instead of checking security of a cipher against each variant of the attack in question, it may be beneficial to outline important postulates to safeguard against invariant attacks.

- **Use of Complex Round Constants**. To ensure strong resistance against nonlinear invariant attacks, the **use of complex round constant is strongly recommended**. Criteria for strong round constants as proposed by Beierle *et al*. in [23] can be utilized as a basic step towards achievement of provable security.

- **Avoiding Use of Binary Orthogonal Matrix in Linear Layer of Block Ciphers**. Recent research on cryptanalysis of lightweight block ciphers suggests that uptill now, vulnerable nonlinear invariants are restricted to quadratic degree only. Similarly, use of binary orthogonal matrix is considered mandatory in order to overcome the linear layer. It is also noteworthy that while using 4x4 bit S-Boxes, quadratic nonlinear invariants cannot be avoided. Therefore, while using 4x4 S-Boxes, one must try to avoid use of binary orthogonal matrix in the linear layer of cipher. This fact must also be borne in mind that orthogonality property is a strong safeguard against linear and differential cryptanalysis, while same is a potential loophole with regards to invariant attacks. In case application of orthogonal matrix is inevitable in the cipher design, it must be non-binary one.

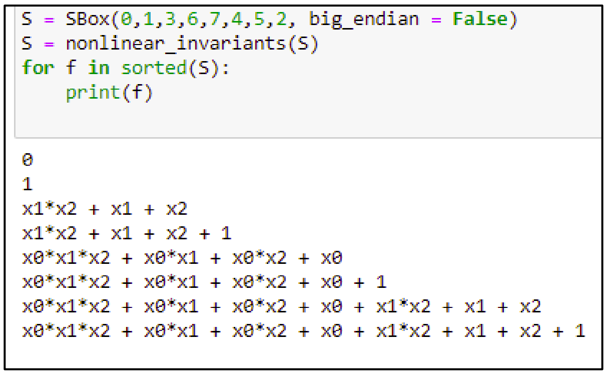
- **S-Box Configuration**.

    - As explored by the existing research, all ciphers composed of 3x3 S-Boxes like Print Cipher, SEA Cipher are vulnerable to invariant attacks. Therefore, it may be implied that 3x3 bit S-Boxes may not be utilized by designers.

- As of ciphers with 4x4 S-Boxes, few ciphers like Prince, Midori, Elephant and Present Ciphers are vulnerable basing on the combination of design components including key schedule (especially choice of round constants) and the linear layer, while few others may be secure.

- As we enhance the configuration of S-Box in the overall structure, i.e., instead of 3x3 S-Box, we use 4x4 or 8x8 S-Box, the complexity of the invariant functions is going to enhance, which is likely to reduce the vulnerability of that cipher against invariant attacks. Let us take the example of 3x3, 4x4 and 8x8 S-Box to compare the complexity and nature of invariants. SageMath results are appended below in **Fig 4.22** through **Fig 4.24** below for reference:-

- **<u>Invariant for Print Cipher S-Box (3x3) Computed by SageMath</u>**

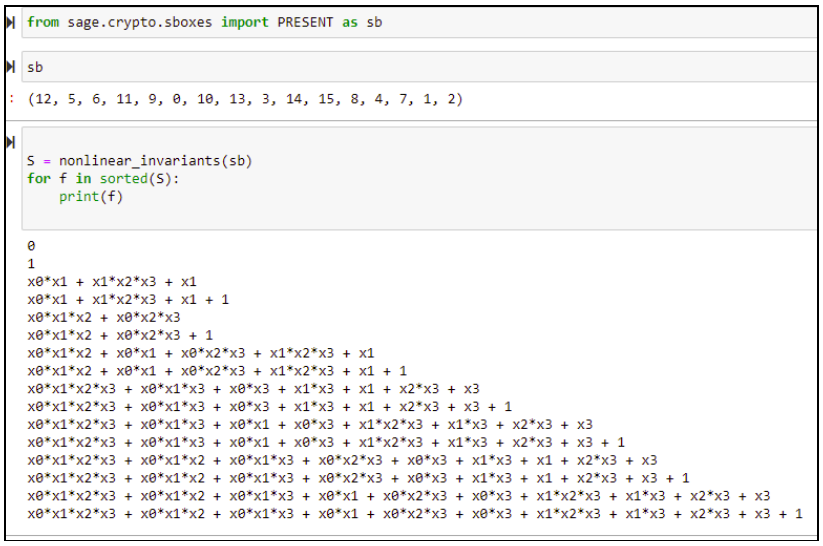


```
S = SBox(0,1,3,6,7,4,5,2, big_endian = False)
S = nonlinear_invariants(S)
for f in sorted(S):
    print(f)
```

```
0
1
x1*x2 + x1 + x2
x1*x2 + x1 + x2 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0
x0*x1*x2 + x0*x1 + x0*x2 + x0 + 1
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2
x0*x1*x2 + x0*x1 + x0*x2 + x0 + x1*x2 + x1 + x2 + 1
```

Figure: 4.22 Invariants of Print Cipher S-box Computed by SageMath

- **<u>Invariant for PRESENT Cipher S-Box (4x4) Computed by SageMath</u>**



```
from sage.crypto.sboxes import PRESENT as sb
```

```
sb
```

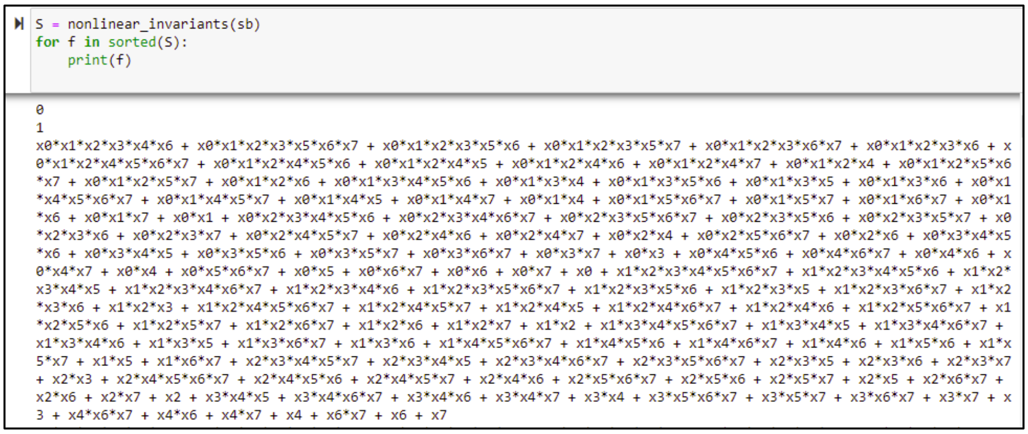
```
(12, 5, 6, 11, 9, 0, 10, 13, 3, 14, 15, 8, 4, 7, 1, 2)
```

```
S = nonlinear_invariants(sb)
for f in sorted(S):
    print(f)
```

```
0
1
x0*x1 + x1*x2*x3 + x1
x0*x1 + x1*x2*x3 + x1 + 1
x0*x1*x2 + x0*x2*x3
x0*x1*x2 + x0*x2*x3 + 1
x0*x1*x2 + x0*x1 + x0*x2*x3 + x1*x2*x3 + x1
x0*x1*x2 + x0*x1 + x0*x2*x3 + x1*x2*x3 + x1 + 1
x0*x1*x2*x3 + x0*x1*x3 + x0*x3 + x1*x3 + x1 + x2*x3 + x3
x0*x1*x2*x3 + x0*x1*x3 + x0*x3 + x1*x3 + x1 + x2*x3 + x3 + 1
x0*x1*x2*x3 + x0*x1*x3 + x0*x1 + x0*x3 + x1*x2*x3 + x1*x3 + x2*x3 + x3
x0*x1*x2*x3 + x0*x1*x3 + x0*x1 + x0*x3 + x1*x2*x3 + x1*x3 + x2*x3 + x3 + 1
x0*x1*x2*x3 + x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x0*x3 + x1*x3 + x1 + x2*x3 + x3
x0*x1*x2*x3 + x0*x1*x2 + x0*x1*x3 + x0*x2*x3 + x0*x3 + x1*x3 + x1 + x2*x3 + x3 + 1
x0*x1*x2*x3 + x0*x1*x2 + x0*x1*x3 + x0*x1 + x0*x2*x3 + x0*x3 + x1*x2*x3 + x1*x3 + x2*x3 + x3
x0*x1*x2*x3 + x0*x1*x2 + x0*x1*x3 + x0*x1 + x0*x2*x3 + x0*x3 + x1*x2*x3 + x1*x3 + x2*x3 + x3 + 1
```

Figure: 4.23 Invariants of PRESENT Cipher S-box Computed by SageMath

- **Invariant for AES S-Box (8x8) Computed by SageMath**



```
S = nonlinear_invariants(sb)
for f in sorted(S):
    print(f)
```

```
0
1
x0*x1*x2*x3*x4*x6 + x0*x1*x2*x3*x5*x6*x7 + x0*x1*x2*x3*x5*x6 + x0*x1*x2*x3*x5*x7 + x0*x1*x2*x3*x6*x7 + x0*x1*x2*x3*x6 + x
0*x1*x2*x4*x5*x6*x7 + x0*x1*x2*x4*x5*x6 + x0*x1*x2*x4*x5 + x0*x1*x2*x4*x6 + x0*x1*x2*x4*x7 + x0*x1*x2*x4 + x0*x1*x2*x5*x6
*x7 + x0*x1*x2*x5*x7 + x0*x1*x2*x6 + x0*x1*x3*x4*x5*x6 + x0*x1*x3*x4 + x0*x1*x3*x5*x6 + x0*x1*x3*x5 + x0*x1*x3*x6 + x0*x1
*x4*x5*x6*x7 + x0*x1*x4*x5*x7 + x0*x1*x4*x5 + x0*x1*x4*x7 + x0*x1*x4 + x0*x1*x5*x6*x7 + x0*x1*x5*x7 + x0*x1*x6*x7 + x0*x1
*x6 + x0*x1*x7 + x0*x1 + x0*x2*x3*x4*x5*x6 + x0*x2*x3*x4*x6*x7 + x0*x2*x3*x5*x6*x7 + x0*x2*x3*x5*x6 + x0*x2*x3*x5*x7 + x0
*x2*x3*x6 + x0*x2*x3*x7 + x0*x2*x4*x5*x7 + x0*x2*x4*x6 + x0*x2*x4*x7 + x0*x2*x4 + x0*x2*x5*x6*x7 + x0*x2*x6 + x0*x3*x4*x5
*x6 + x0*x3*x4*x5 + x0*x3*x5*x6 + x0*x3*x5*x7 + x0*x3*x6*x7 + x0*x3*x7 + x0*x3 + x0*x4*x5*x6 + x0*x4*x6*x7 + x0*x4*x6 + x
0*x4*x7 + x0*x4 + x0*x5*x6*x7 + x0*x5 + x0*x6*x7 + x0*x6 + x0*x7 + x0 + x1*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*x6 + x1*x2*
x3*x4*x5 + x1*x2*x3*x4*x6*x7 + x1*x2*x3*x4*x6 + x1*x2*x3*x5*x6*x7 + x1*x2*x3*x5*x6 + x1*x2*x3*x5 + x1*x2*x3*x6*x7 + x1*x2
*x3*x6 + x1*x2*x3 + x1*x2*x4*x5*x6*x7 + x1*x2*x4*x5*x7 + x1*x2*x4*x5 + x1*x2*x4*x6*x7 + x1*x2*x4*x6 + x1*x2*x5*x6*x7 + x1
*x2*x5*x6 + x1*x2*x5*x7 + x1*x2*x6*x7 + x1*x2*x6 + x1*x2*x7 + x1*x2 + x1*x3*x4*x5*x6*x7 + x1*x3*x4*x5 + x1*x3*x4*x6*x7 +
x1*x3*x4*x6 + x1*x3*x5 + x1*x3*x6*x7 + x1*x3*x6 + x1*x4*x5*x6*x7 + x1*x4*x5*x6 + x1*x4*x6*x7 + x1*x4*x6 + x1*x5*x6 + x1*x
5*x7 + x1*x5 + x1*x6*x7 + x2*x3*x4*x5*x7 + x2*x3*x4*x5 + x2*x3*x4*x6*x7 + x2*x3*x5*x6*x7 + x2*x3*x5 + x2*x3*x6 + x2*x3*x7
+ x2*x3 + x2*x4*x5*x6*x7 + x2*x4*x5*x6 + x2*x4*x5*x7 + x2*x4*x6 + x2*x5*x6*x7 + x2*x5*x6 + x2*x5*x7 + x2*x5 + x2*x6*x7 +
x2*x6 + x2*x7 + x2 + x3*x4*x5 + x3*x4*x6*x7 + x3*x4*x6 + x3*x4*x7 + x3*x4 + x3*x5*x6*x7 + x3*x5*x7 + x3*x6*x7 + x3*x7 + x
3 + x4*x6*x7 + x4*x6 + x4*x7 + x4 + x6*x7 + x6 + x7
```

Figure: 4.24 Invariants of AES S-box Computed by SageMath

- **Summary**

Ser.	Cipher	S-Box Size	No. of Terms in S-Box	No. of Invariants	Max No. of Terms in Invariant	Complexity (Max Product Terms) in Invariant
1.	Print	3x3	8	8	8	3
3.	Present	4x4	16	16	11	4
4.	AES	8x8	256	32	134	8

Table: 4.20 Comparison of S-box Size Vs Complexity of Invariant

- **Graphs**

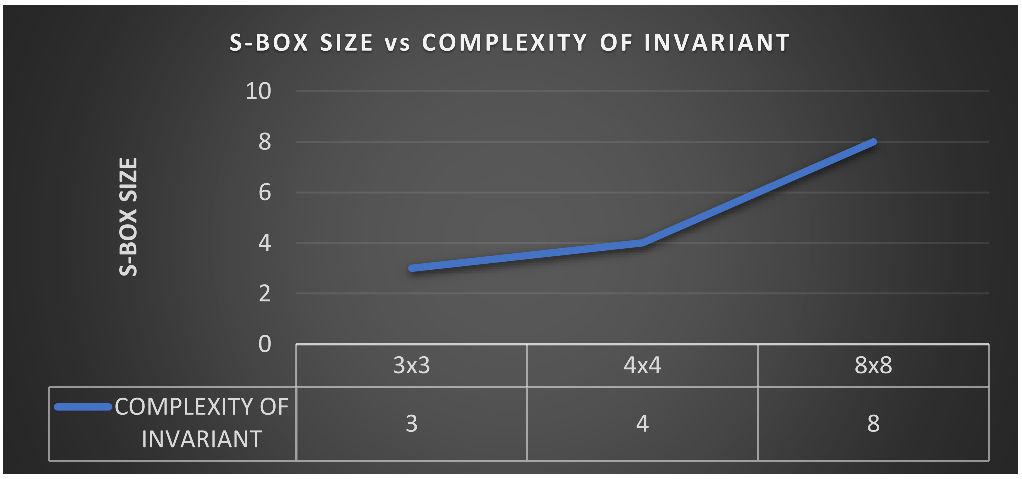


Figure: 4.25 Comparison of S-box Size Vs Complexity of Invariants

<u>Results</u>. Results summarized in **Table 4.20** and graphs in **Fig 4.25** and **4.26** reveal following:-

- As we enhance the configuration of the S-Box, the complexity of the invariant and max number of terms in single invariant increase exponentially.
- We can clearly observe that complexity of invariant of a single 8x8 S-Box is very difficult to solve. Same is going to get more complex with addition of invariants coming from several rounds. Thus, with S-Box of greater configuration, the possibility of invariant attack becomes questionable.

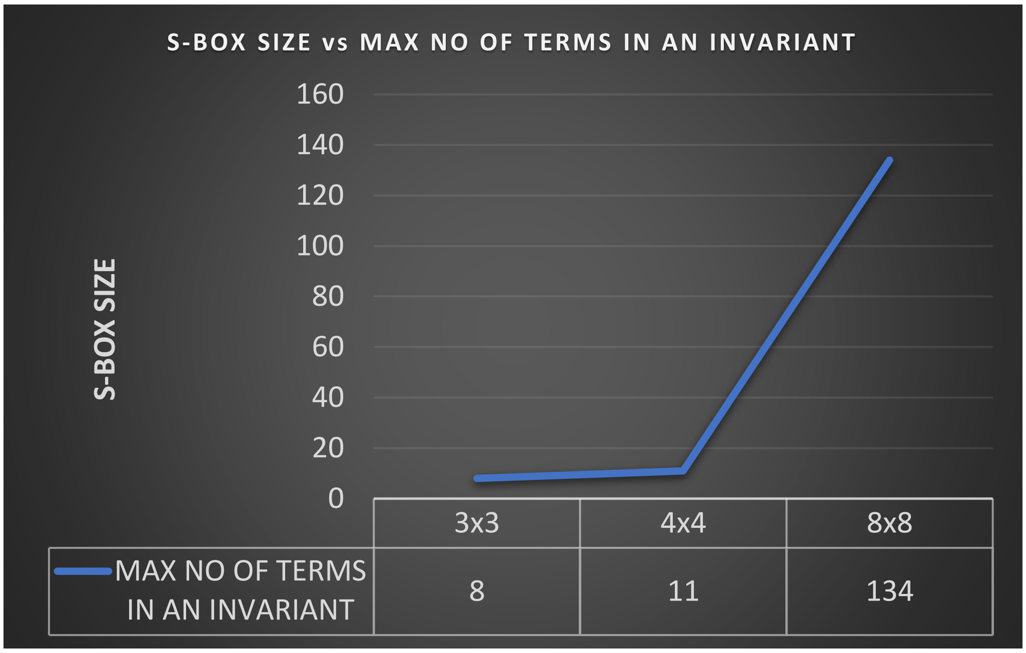


Figure: 4.26 Comparison of S-box Size Vs Max No of Terms in An Invariants

4.6.2 Methods to Enhance Resistance Against Invariant Subspace Attacks

- A baseline formula to enhance resistance invariant subspace attacks is to employ complex key alternating schedules or random / complex round constants. In contrast however, there are certain ciphers without employing any key schedule, yet they are resistant to the invariant subspace attack, i.e., Fantomas, Led and Noekon.
- Possibility of existence of invariant subspaces is dependent upon linear relations in the S-Boxes. Furthermore, the dimension of possible invariant can be bounded to control the existence of subspaces.

- As similar to nonlinear invariant attack, all 3x3 bit S-Boxes are vulnerable to invariant subspace attacks. As of 4x4 bit S-Boxes, certain variants like Serpent and present are vulnerable owing to the construction and properties, while others like Rinjdael may be considered safe since they possess no such weak equation that may be exploited.

4.7 **Proposed Design Criteria**. Having gone through the basic construct of all major variants of the invariant attack and delved into the analyses of various ciphers attacked by these variants, it is deemed appropriate that a common criterion may be defined which can assure provable security against invariant attacks against SPN ciphers. These defined properties are inclusive of and in addition to the agreed upon set of properties of a good S-Box as defined by NIST [43].

4.7.1 Criteria of Cryptographically Strong S-Box

As defined by NIST, following five design criteria must be satisfied by Boolean functions used in deigning cryptographically strong S-Boxes [55].

- **Bijection.** In case of n x n bit S-box, if the input satisfies one-to-one and onto mapping to the output.
- **Strict Avalanche Criterion**. It implies that change in a single bit in the input vector gives rise to a significant change in the output bits of the vector with probability of one half**.**
- **Bit Independence Criterion**. It implies that there should not be any statistical relationship among the output bits from the output vectors.
- **Non-Linearity**. Non-linearity implies that S-Box is not a linear mapping from input to output, because in case of such mapping, the S-Box would be vulnerable to attacks [44].
- **Balance**. Balance means that the Boolean Vectors combined in the S-Box have equal number of 0's and 1's in the truth table.

4.7.2 **Characteristics of S-Box Required to Resist Invariant Attacks**

Keeping in view the diversity of block ciphers and the large number of attacks being attempted to break their security, it is imperative that a common toolset be proposed in addition to the previously proposed criteria. Main characteristics are as under:-

- **S-Box Configuration**. Each S-Box employed in the cipher must be at-least 4x4 configuration, however, used in the right combination with linear layer and key alternating algorithm.

- **Use of Non-Binary Orthogonal Matrix in Linear Layer**. In case of 4x4 S-Boxes used in the substitution layer, use of binary orthogonal matrix in linear layer must be avoided.

- **Fixed and Opposite Fixed Points in S-Box**. Fixed and opposite fixed points provide vulnerability in the cipher by giving output same (or compliment in case of opposite fixed points) as of input on specific fixed points. Therefore, S-Boxes with fixed and opposite fixed points must be avoided in the system.

- **Key Alternating Algorithm**. In lightweight ciphers, key alternating algorithms are kept either very simple or sometimes even non-existent in order to improve energy consumption, however, same is a serious security hazard especially with regards to weak keys contributing to invariant attacks. It is recommended that strong key alternating schedule must be employed in a cipher so that the quantum of weak keys is kept to the minimum. A balance may be struck among security and energy consumption.

- **Use of Complex Round Constants**. Round constants are an important factor in the security of a cipher. The intelligent choice of round constants is a must, since even few complex round constants result in vulnerability. Round constants must be chosen wisely in relation to the linear and substitution layers of the cipher.

- **Use of Involution S-Boxes**. S-Box designed with involution functions are more security robust compared to involution ones. This must be factored while checking resistance against invariant attacks.

4.7.3 **Schematic Representation of Proposed Design Criteria**

For ease of assimilation, the proposed design criteria to safeguard a block cipher from invariant attacks family has been schematically represented in Fig 4.27 below. At the top, we consider a standard SPN block cipher in question which we intend to make secure against invariant attack. Then we analyse the cipher with regards to its vulnerabilities and work out set of assumptions that must hold for the attack to succeed. Then we analyse which sub-class of invariant attacks Then we apply our worked-out criteria for S-Box design as well as for other components of the cipher. These set of properties will in turn make the cipher secure against invariant attacks. This can further be re-confirmed by again checking for vulnerabilities and repeating the loop. Diagrammatic layout is as under:-

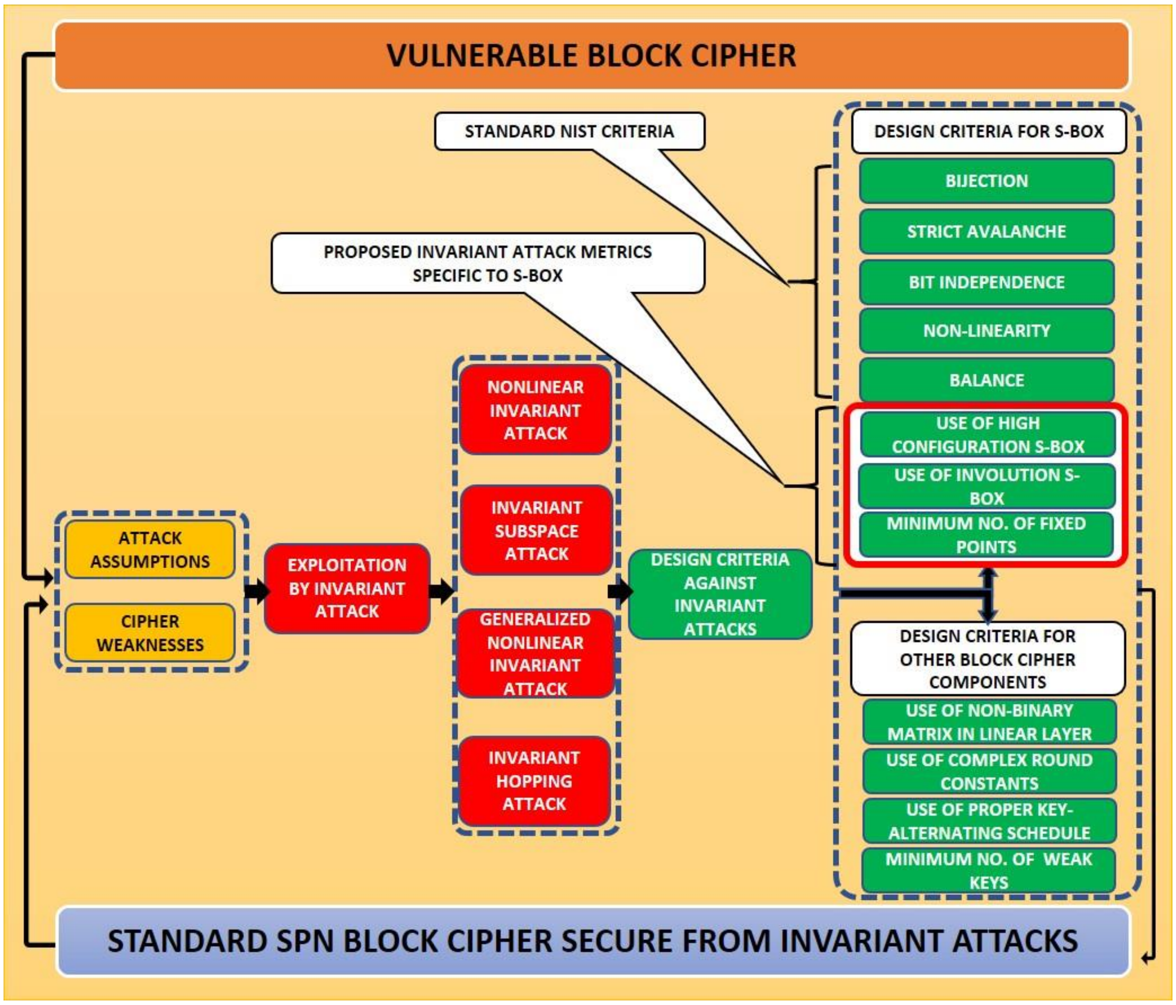

VULNERABLE BLOCK CIPHER
STANDARD NIST CRITERIA
DESIGN CRITERIA FOR S-BOX
BIJECTION
STRICT AVALANCHE
BIT INDEPENDENCE
NON-LINEARITY
BALANCE
PROPOSED INVARIANT ATTACK METRICS SPECIFIC TO S-BOX
USE OF HIGH CONFIGURATION S-BOX
USE OF INVOLUTION S-BOX
MINIMUM NO. OF FIXED POINTS
NONLINEAR INVARIANT ATTACK
INVARIANT SUBSPACE ATTACK
GENERALIZED NONLINEAR INVARIANT ATTACK
INVARIANT HOPPING ATTACK
ATTACK ASSUMPTIONS
CIPHER WEAKNESSES
EXPLOITATION BY INVARIANT ATTACK
DESIGN CRITERIA AGAINST INVARIANT ATTACKS
DESIGN CRITERIA FOR OTHER BLOCK CIPHER COMPONENTS
USE OF NON-BINARY MATRIX IN LINEAR LAYER
USE OF COMPLEX ROUND CONSTANTS
USE OF PROPER KEY-ALTERNATING SCHEDULE
MINIMUM NO. OF WEAK KEYS
STANDARD SPN BLOCK CIPHER SECURE FROM INVARIANT ATTACKS

Figure: 4.27 Schematic Layout of Proposed Design Criteria

4.7 **Conclusion**. Above mentioned are few of the considerations that can assure provable security against invariant attacks. Invariant attacks are still in the nascent phase and are evolving at a fast pace, however, there has not been a major breakthrough with regards to any full-scale cipher like AES. In addition, invariant attacks work under lot of assumptions like weak key schedule, weak round constants, requirement of large number of plaintext-ciphertext pairs and so on. If these vulnerabilities are addressed in the S-Box design phase, lot of such attacks can be avoided.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

Invariant attack family has recently been researched and proposed. Recent research has revealed four main types of invariant attacks. Till now, all four sub types have their peculiar characteristics, assumptions, and results. Similarly, all these types attack different types of block ciphers using different methods. The purpose of this thesis was to develop an in-depth insight into invariant attacks, so that, a common toolset can be defined to safeguard against these attacks. An effort has been made to study the previously defined toolsets and find additional set of properties to add security and resilience to the S-boxes in general and entire block ciphers in particular. Keeping these properties in sight, we can easily safeguard against invariant attacks through preventive security. In retrospect, a lot remains to be researched on to the various categories of invariant attacks already discovered and those yet to be found. The base line rests on the basic structure of an SPN cipher which includes a round function, key-alternating schedule (including the round constant), the confusion component (S-box layer) and the diffusion component (permutation layer). In case, these components are used in a balanced composition with compromising the security features and at the same time, keeping the cipher lightweight to economize energy and storage consumption, a secure working trade-off can be worked out providing all required functionalities. In nutshell, for a cipher to be secure with regards to invariant attacks, we propose that in addition to satisfying already published NIST criteria of bijection, non-linearity, balance, strict avalanche criteria and bit independence, a good cipher must also have S-Boxes which are involution type, be of high configuration, and have minimum number of fixed points. Other than S-Box, the cipher must have minimum number of weak keys and use complex round constants to be totally secure.

5.2 **Future Work**

Block ciphers are one of the most important primitives of symmetric key algorithms which have widespread application in modern communications. Lightweight block ciphers are often employed in energy efficient applications like internet of things and artificial intelligence etcetera. In such systems, a trade-off among security and energy efficiency is worked out to meet the optimum requirements of the system. Lightweight versions of block ciphers are created by reducing the number of rounds, key lengths, simplifying key-alternating algorithms, reducing block lengths, using S-boxes with simple Boolean functions and many more. These energy conserving measures at one hand enhance the workability, while on other hand these security compromises give way to vulnerabilities for different types of attacks. Till now, many families of attacks have surfaced that include linear cryptanalysis, differential cryptanalysis, statistical attacks, structural attacks and many more. However, basing on these attacks, no full version of cipher like AES could be broken. Recently, various forms of invariant attacks came to surface with success against full versions of few lightweight ciphers depending upon the weaknesses in the S-box or combinational vulnerabilities stemming from other weak components. As researched, the invariant attacks work with assumptions of weak keys and weaknesses in the cipher structure. As elaborated in Chapter-4, four main variants of invariant attacks have surfaced namely nonlinear invariant attacks, invariant subspace attacks, invariant hopping attacks and generalized nonlinear attacks. In-depth study revealed that all variants of the attacks require different types of assumptions and conditions to materialize.

As crystallized in the start of the thesis, the main aim is to understand the anatomy of attack, so that a toolkit can be devised in order to safeguard against the invariant attacks. This is more economical with regards to the research. It is identified that invariant attacks differ from other families of attacks like linear and differential cryptanalysis. For a start, NIST defined five set of properties required for strong Boolean functions (which constitute an S-box). Later, researchers defined different toolsets to guard against different families of attacks, however, most research was focused against the most promising attacks i.e., linear, and differential cryptanalysis. During course of this research, it has been determined that although NIST criteria for strong Boolean functions (for S-box design) does ensure resilience against most attacks, it is not sufficient to guard against newer families of attacks like invariant attacks. Nevertheless, basic criteria are essential step for S-box design and further properties as defined in this Chapter-4 are mandatory to ensure resilience.

Foregoing in view, there is an essential requirement of a well-tested toolset to further proceed with our inquiry on Invariant Attacks. Here mention of the cryptographic tool embedded in SageMath application is noteworthy. Initial versions of SageMath were able to provide various properties of imported S-boxes, such as Difference Distribution Table, Auto-Correlation Table, Boomerang Connectivity Table, Non-linearity, Differential Uniformity, Maximum Differential Probability, Maximum Linear Bias and many more. All these inquiries of S-Box properties were carried out in our thesis and tabulated to accrue certain conclusions. In the latest version of SageMath, a newer function to calculate Nonlinear Invariants has been proposed, although not yet finally embedded. This function can take S-Box values as input and provide all possible invariants of the S-Box for further analysis. However, the feature is not yet final, and it lacks the capability of analysis of multiple S-Boxes in the cipher. Similarly, there is no feature to calculate invariants of permutation stage or full round functions. SageMath has turned out to be a perfect toolkit, however with the combination of all previous properties discussed above with the invariant function, a full-fledge toolset can be worked out to facilitate work.

During the course of research, four different sub-categories (mostly interlinked with each other) came to fore. They are conceptually same, however, differ in the pre-requisites and manifestation of attacks. Similarly, there are ciphers which are vulnerable to one category of invariant attacks while same cipher is secure against the other category (or categories). This would be a great field of further research to crystallize all these types of invariant attacks, carry out their mathematical and conceptual comparisons to clarify similarities and differences of all these types. This would further pave way to tell us why certain ciphers are vulnerable to invariant subspace attacks while secure against nonlinear invariant attacks, and vice-versa. This comparative study will also help us consolidate a common cryptographic toolset against all variants of the invariant attack. So far, maximum degree of the Boolean polynomials for which final fundamental equation can be computed is limited to 6. In contrast the space of possible polynomials Z and P is extremely large. In future research, endeavours can be made to enhance this maximum degree of Boolean polynomial and to be able to exactly locate the weak key space, like in the case of finding the invariants [59].

As of now, only linear, and differential cryptanalysis techniques were considered to be a substantial threat to light weight block ciphers, while full scale ciphers like AES are still considered safe against these techniques. A major line of work could be to pitch the toolsets of linear and differential properties against those of invariant attacks and find out whether if ciphers safe against linear and differential cryptanalysis were also secure against invariant attacks. Similarly, with the inception of new attack categories and sub-categories, it would be prudent to classify the newer attack types into various categories for future research.

BIBLIOGRAPHY

[1] Trappe, W. and L. Washington, Introduction to Cryptography with Coding Theory. Pearson International Edition ed. second, India: Pearson Practice Hall, 2006.

[2] E. Laskari, G. Meletiou, Y. Stamatiou and M. Vrahatis, "Cryptography and Cryptanalysis Through Computational Intelligence", Studies in Computational Intelligence, pp. 1-49, 2007. Available: 10.1007/978-3-540-71078-3_1 [Accessed 8 November 2020].

[3] G. Simmons, "Symmetric and Asymmetric Encryption", ACM Computing Surveys, vol. 11, no. 4, pp. 305-330, 1979. Available: 10.1145/356789.356793 [Accessed 8 November 2020].

[4] M. Ubaidullah and Q. Makki, "A Review on Symmetric Key Encryption Techniques in Cryptography", International Journal of Computer Applications, vol. 147, no. 10, pp. 43-48, 2016. Available: 10.5120/ijca2016911203.

[5] H. Heys and S. Tavares, "Substitution-permutation networks resistant to differential and linear cryptanalysis", Journal of Cryptology, vol. 9, no. 1, pp. 1-19, 1996. Available: 10.1007/bf02254789.

[6] C. Blondeau, G. Leander and K. Nyberg, "Differential-Linear Cryptanalysis Revisited", Journal of Cryptology, vol. 30, no. 3, pp. 859-888, 2016. Available: 10.1007/s00145-016-9237-5.

[7] C. Harpes, G.G. Kramer, J.L. Massey, A generalization of linear cryptanalysis and the applicability of Matsui's piling-up lemma, in L.C. Guillou, J.-J. Quisquater, editors, EUROCRYPT. LNCS, vol. 921 (Springer, 1995), pp. 24–38.

[8] Y. Todo, G. Leander and Y. Sasaki, "Nonlinear Invariant Attack: Practical Attack on Full SCREAM, iSCREAM, and Midori64", Journal of Cryptology, vol. 32, no. 4, pp. 1383-1422, 2018. Available: 10.1007/s00145-018-9285-0.

[9] G. Hatzivasilis, K. Fysarakis, I. Papaefstathiou and C. Manifavas, "A review of lightweight block ciphers", Journal of Cryptographic Engineering, vol. 8, no. 2, pp. 141-184, 2017. Available: 10.1007/s13389-017-0160-y [Accessed 8 November 2020].

[10] N. Courtois, "A nonlinear invariant attack on T-310 with the original Boolean function", Cryptologia, pp. 1-15, 2020. Available: 10.1080/01611194.2020.1736207.

[11] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems", Journal of Cryptology, vol. 4, no. 1, pp. 3-72, 1991. Available: 10.1007/bf00630563.

[12] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of

Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings, pages 386–397, 1993.

[13] Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In Advances in Cryptology - CRYPTO ’94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings, pages 17–25, 1994.

[14] C. Harpes, G. Kramer, and J. Massey: A Generalization of Linear Cryptanalysis and the Applicability of Matsui’s Piling-up Lemma, Eurocrypt’95, LNCS 921, Springer, pp. 24–38.

[15] Carlo Harpes, Gerhard G. Kramer, and James L. Massey. A Generalization of Linear Cryptanalysis and the Applicability of Matsui’s Piling-Up Lemma. In Advances in Cryptology - EUROCRYPT ’95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding, pages 24–38, 1995.

[16] Lars R. Knudsen and Matthew J. B. Robshaw. Non-Linear Approximations in Linear Cryptanalysis. In Advances in Cryptology - EUROCRYPT ’96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding, pages 224–236, 1996.

[17] Nicolas T. Courtois: On the Existence of Non-Linear Invariants and Algebraic Polynomial Constructive Approach to Backdoors in Block Ciphers, https://ia.cr/2018/807, last revised 27 Mar 2019.

[18] Nicolas T. Courtois: On the Existence of Non-Linear Invariants and Algebraic Polynomial Constructive Approach to Backdoors in Block Ciphers.

[19] Lars R. Knudsen, Matthew J. B. Robshaw: Non-Linear Characteristics in Linear Cryptoanalysis, Eurocrypt’96, LNCS 1070, Springer, pp. 224–236, 1996.

[20] Yongzhuang Wei, Tao Ye, Wenling Wu, Enes Pasalic: Generalized Nonlinear Invariant Attack and a New Design Criterion for Round Constants, In IACR Tr. on Symm. Crypt. Vol. 2018, No. 4, pp.62-79.

[21] Nicolas Courtois: Feistel Schemes and Bi-Linear Cryptanalysis, in Crypto 2004, LNCS 3152, pp.23–40, Springer, 2004.

[22] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64. In Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II, pages 3–33, 2016.

[23] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving Resistance Against Invariant Attacks: How to Choose the Round Constants. In Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology

Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II, pages 647–678, 2017.

[24] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack: Practical Attack on Full SCREAM, iSCREAM, and Midori64. Journal of Cryptology, pages 1432–1378, Apr 2018.

[25] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64. In Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II, pages 3–33, 2016.

[26] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. SCREAM v3. Submission to CAESAR competition. 2015.

[27] Vincent Grosso, Gaëtan Leurent, François-Xavier Standaert, and Kerem Varici. SCREAM v1. Submission to CAESAR competition. 2014.

[28] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A Block Cipher for Low Energy. In Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 – December 3, 2015, Proceedings, Part II, pages 411–436, 2015.

[29] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack. In Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings, pages 206–221, 2011.

[30] Yosuke Todo, Gregor Leander, and Yu Sasaki. Nonlinear Invariant Attack: Practical Attack on Full SCREAM, iSCREAM, and Midori64. Journal of Cryptology, pages 1432–1378, Apr 2018.

[31] Carlet, C.: On the coset weight divisibility and nonlinearity of resilient and correlation immune functions. In: Sequences and Their Applications-SETA, 01. Discrete Mathematics and Theoretical Computer Science, pp. 131–144. Springer, Berlin (2000)

[32] Meier, W., Stafelbach, O.: Fast correlation attacks on certain stream ciphers. J. Cryptol. **1**(3), 159–176 (1989)

[33] Cohen, G., Honkala, I., Litsyn, S., Lobstein, A.: Covering Codes. Elsevier, Amsterdam (1997)

[34] Millan, W.L.: Analysis and design of Boolean functions for cryptographic applications. Ph.D. thesis, Queensland University of Technology (1997)

[35] Feistel, H.: Cryptography and computer privacy. Sci. Am.**228**(5), 15–23 (1973)

[36] Lai, X.: Higher Order Derivative and Differential Cryptanalysis. Communications and Cryptography: Two Sides of One Tapestry, pp. 227–233. Kluwer Academic, Dordrecht (1994)

[37] Preneel, B., Leekwijck, W.V., Linden, L.V., Govaerts,R., Vandewalle, J.: Propagation characteristics of Boolean functions. In: Advances in Cryptology—EUROCRYPT 90. Lecture Notes in Computer Science, vol. 473, pp. 161–173. Springer, Berlin (1991)

[38] Webster, A. F.; Tavares, Stafford E. (1985). "On the design of S-boxes". *Advances in Cryptology - Crypto '85*. Lecture Notes in Computer Science. **218**. New York, NY: Springer-Verlag New York, Inc. pp. 523–534. ISBN 0-387-16463-4.

[39] T. Siegenthaler, Correlation-immunity of nonlinear combining functions for cryptographic applications, IEEE Transactions on Information theory, V. IT-30, No 5, 1984, p. 776–780.

[40] Biham, E., Shamir, A. Differential cryptanalysis of DES-like cryptosystems. *J. Cryptology* **4,** 3–72 (1991). https://doi.org/10.1007/BF00630563.

[41] Youssef, A. & Tavares, Stafford & Mister, S. & Adams, C.. (1996). Linear approximation of injective s-boxes. Electronics Letters. 31. 2165-2166. 10.1049/el:19951466.

[42] T. Muenzenherger and R. Smithson, "Fixed points and proximate fixed points", *Fundamenta Mathematicae*, vol. 63, no. 3, pp. 321-326, 1968. Available: 10.4064/fm-63-3-321-326.

[43] Heuser A, Rioul O and Guilley S. A theoretical study of Kolmogorov-Smirnov distinguishers: side-channel analysis vs. differential cryptanalysis.

[44] P. Kocher, J. Jaffe, B. Jun, Differential Power Analysis, technical report, 1998; later published in Advances in Cryptology – Crypto 99 Proceedings, Lecture Notes in Computer Science Vol. 1666, M. Wiener, ed., Springer-Verlag, 1999.

[45] C. Adams and S. Tavares, "The structured design of cryptographically good s-boxes", Journal of Cryptology, vol. 3, no. 1, pp. 27-41, 1990. Available: 10.1007/bf00203967.

[46] E. F. Brickell, J. H. Moore, and M. R. Purtill, Structure in the s-boxes of the DES (extended abstract), in Advances in Cryptology: Proc. of CR YPTO '86, Springer-Verlag, New York, 1987, pp. 3-8.

[47] J. Guo, J. Jean, I. Nikolic, K. Qiao, Y. Sasaki and S. Sim, "Invariant Subspace Attack Against Midori64 and The Resistance Criteria for S-box Designs", *IACR Transactions*

on Symmetric Cryptology, pp. 33-56, 2016. Available: 10.46586/tosc.v2016.i1.33-56.

[48] C. Adams and S. Tavares, "The structured design of cryptographically good s-boxes", *Journal of Cryptology*, vol. 3, no. 1, pp. 27-41, 1990. Available: 10.1007/bf00203967.

[49] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. Proving Resistance Against Invariant Attacks: How to Choose the Round Constants. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 647–678, 2017.

[50] Nicolas T. Courtois: *On the Existence of Non-Linear Invariants and Algebraic Polynomial Constructive Approach to Backdoors in Block Ciphers,* https://eprint.iacr.org/2018/807.pdf, revised 3 Dec 2018.

[51] Marco Calderini: *A note on some algebraic trapdoors for block ciphers,* last revised 17 May 2018, https://arxiv.org/abs/1705.08151.

[52] Lars R. Knudsen, Matthew J. B. Robshaw: *Non-Linear Characteristics in Linear Cryptoanalysis,* Eurocrypt'96, LNCS 1070, Springer, pp. 224{236, 1996.

[53] Courtois, Nicolas. (2020). Invariant Hopping Attacks on Block Ciphers.

[54] Nicolas T. Courtois: *Structural Nonlinear Invariant Attacks on T-310: Attacking Arbitrary Boolean Functions,* https://eprint.iacr.org/2018/1242.pdf, received 28 Dec 2018. H. Knospe, *A course in cryptography*

[55] Beierle C., Canteaut A., Leander G., Rotella Y. (2017) Proving Resistance Against Invariant Attacks: How to Choose the Round Constants. In: Katz J., Shacham H. (eds) Advances in Cryptology – CRYPTO 2017. CRYPTO 2017. Lecture Notes in Computer Science, vol 10402. Springer, Cham. https://doi.org/10.1007/978-3-319-63715-0_22.

[56] C. Adams and S. Tavares. The structured design of cryptographically good s-boxes. Journal of Cryptology, 3(1):27–41, 1990.

[57] D. Coppersmith. The data encryption standard and its strength against attacks. IBM Journal of Research & Development, 38(3):243, May 1994.

[58] "21252 (Computing nonlinear invariants in mq.SBox) – Sage", *Trac.sagemath.org*, 2021. [Online]. Available: https://trac.sagemath.org/ticket/21252. [Accessed: 04-Apr- 2021].

[59] Nicolas Courtois, Maria-Bristena Oprisanu and Klaus Schmeh: Linear cryptanalysis and block cipher design in East Germany in the 1970s, in Cryptologia, 05 Dec 2018, https://www.tandfonline.com/doi/abs/10.1080/01611194.2018.1483981.

Appendix-I

SAGEMATH CODE FOR COMPUTING NONLINEAR INVARIANTS IN MQ.SBOX

```
def nonlinear_invariants(self):
   m = self.m
    F2 = GF(2)
    one = F2.one()
    zero = F2.zero()
    R = BooleanPolynomialRing(m, 'x')
    def to_bits(i):
        return tuple(ZZ(i).digits(base=2, padto=m))
    def poly_from_coeffs(c):
        return R({to_bits(j): one for j,ci in enumerate(c) if ci})
    L = [[zero if ((v & w) == w) == ((sv & w) == w) else one
          for w in range(1<<m)]
         for v,sv in enumerate(self._S)]
    M = Matrix(F2, L)
    T0 = {poly_from_coeffs(Ai) for Ai in M.right_kernel()}
    M[:,0] = one
    T1 = {poly_from_coeffs(Ai) for Ai in M.right_kernel()}
    return tuple(T0 | T1)
```