A STUDY ON COUNTERMEASURES ON AES AGAINST SIDE CHANNEL ATTACKS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DAMLA ÇENESIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

AUGUST 2019

Approval of the thesis:

A STUDY ON COUNTERMEASURES ON AES AGAINST SIDE CHANNEL ATTACKS

submitted by **DAMLA ÇENESIZ** in partial fulfillment of the requirements for the degree of **Master of Science in Cryptography Department, Middle East Technical University** by,

Prof. Dr. Ömür Uğur
Director, Graduate School of **Applied Mathematics** ________________

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography** ________________

Prof. Dr. Ferruh Özbudak
Supervisor, **Cryptography, METU** ________________

Examining Committee Members:

Assoc. Prof. Dr. Ali Doğanaksoy
Mathematics Department, METU ________________

Prof. Dr. Ferruh Özbudak
Mathematics Department, METU ________________

Assoc. Prof. Dr. Sedat Akleylek
Computer Engineering Department, Ondokuz Mayıs University ________________

Date: ________________

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: DAMLA ÇENESIZ

Signature :

ABSTRACT

A STUDY ON COUNTERMEASURES ON AES AGAINST SIDE CHANNEL ATTACKS

Çenesiz, Damla
M.S., Department of Cryptography
Supervisor : Prof. Dr. Ferruh Özbudak

August 2019, 37 pages

Side Channel Attacks have a important role for security of cryptographic algorithm. There are different method which include Threshold Implementation to protect against these kind of attacks. In this thesis, we study certain countermeasures to side channel attacks for AES. We start with a survey on Side Channel Attacks for block ciphers and we mentioned attack models for AES. We give also partical attention Treshold Implementation properties and construction methods. We also give some details of subfield construction and Threshold Implementation of AES.

Keywords: Side channel attack, S-box, AES, Subfield

ÖZ

YAN KANAL ANALİZLERİNE KARŞI AES İÇİN GELİŞTİRİLEN KORUMA YÖNTEMLERİ ÜZERİNE BİR ÇALIŞMA

Çenesiz, Damla
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Prof. Dr. Ferruh Özbudak

Ağustos 2019, 37 sayfa

Yan kanal analizi atakları, günümüz kriptografik algoritmaları için tehdit oluşturmaktadır. Altsınır gerçeklemesinin de içinde olduğu yan kanal analizi ataklarına karşı birçok yöntem bulunmaktadır. Bu çalışmada belirli yan kanal analizi saldırılarına karşı, AES şifreleme yöntemi için geliştirilen belirli bir koruma yöntemi çalışılmıştır. Öncelikli olarak blok şifrelere uygulanan yan kanal analizi ataklarıyla ilgili araştırma yapılmıştır ve AES için oluşturulan bazı atak modelleri incelenmiştir. Daha sonrasında Altsınır Gerçeklemesi'nin özellikleri incelenmiştir ve AES için kullanılan Altsınır Gerçeklemeleri ve AES algoritmasının altcisim yapılanması ile ilgili detaylı bilgiye yer verilmiştir.

Anahtar Kelimeler: Yan kanal Analizi, S-kutusu, AES, Altcisim

ACKNOWLEDGMENTS

I would like to my very thanks to my thesis supervisor Prof. Dr. Ferruh Özbudak for his encouragement and valuable advices during this thesis. Also, I would like to express my gratitude to Assoc. Prof. Dr. Ali Doğanaksoy for his encouragements, committee member Sedat Akleylek and all my other teachers for all the information they taught me.

I would like to thank my family for their support during this thesis preparation time. Many thanks goes to İsmail T. Aslıyüce, Erkan Uslu and Kübra Kaytancı for their all supports and encouragements. Finally, I thank Şeyma Bodur who was my comrade in this period.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

CPA	Correlation Power Analysis
SPA	Simple Power Analysis
DPA	Differential Power Analysis
TI	Threshold Implementation
FPGA	Field Programmable Gate Array
EM	Elektro Magnetic
AES	Advanced Encryption Standard
S-Box	Byte Substition
GF	Galois Field

CHAPTER 1

INTRODUCTION

Cryptographic algorithms include encryption algorithm, plaintext-ciphertext pairs and key and must provide four properties such that confidentially, data integrity, authentication and non-repudation. If a device use cryptographic algorithm for security of information, then this device is called a crytographic device. These devices can be smart cards, FPGA, id card, computer and some other devices.

Kerchoff's law assummes that cryptographic algorithm process is known. Only key is kept as a secret. Therefore, breaking a cryptographic algorithm generally means obtaining secret key. If there is no attack to get secret key, then the algorithm is considered secure in practice. If the currect technologies is not enough to break a cryptographic algorithm, the algorithm is called computationally secure.

Not only cryptograhic algorithm security but also device characteristic are so important for the security of an algorithm. Because there are some methods to obtain secret key which are called Side Channel Attacks. The most known of these attack is Power Analysis Attack[16] and the attack is applicable with very few and cheap equipment. This method is first shown by Kocher in 1998 [11], then this attack type has been popular. After that, protection methods against these attack has been developed for algorithms used today. One of the protection algorithm is Threshold Implementation [19] method which was proved reliability against first order power analysis attacks. [9] [1]

In this thesis, first chapter is consists of side channel attacks on cryptographic devices. Also, power analysis attacks method and model construction of attacks on AES are detailed. In second chapter, AES algorithm and subfield construction [12] [7] details of AES Substitution Box are given. In the third chapter, Threshold Implementation properties are explained.[14] Construction method of Threshold Implementation are shown with explanatory examples. In the last part of thesis, a Threshold Implementation of AES [13] [1] is given in detailed and functions are used in this implementation and analyzed from the points of Threshold Implementation properties and construction methods.

1.1 Side Channel Attacks on Cryptograhic Devices

All efforts try to obtain key are named an attack. These attack are divided into two groups: passive and active attacks.

Passive attacks attempt to obtain secret information by examining the chracteristic of a cryptographic device such as power, EM , and time consumption without interfering the device.

Active attacks attempt to device by direct intervention. Behaviours as a result of this attack inform about secret information.

There is an also different classification of attack types. ***Invansive attack***, can be embedded to cryptographic devices. There is no restriction in this attack type to obtain secret key. Invansive attack start with analyzing different part of device. By using probe, different part of device is attained. If probing is just used to observe data signal, this attack is passive attack also. In ***Semi-invansive attack***, secret information is attempted to obtain from memory cells without probing. Active semi-invansive attack cause a fault in the device by using electromagnetic field, x-ray etc.

Non-invansive attacks are also called side channel attacks. Some non-expensive devices would be enough for these kind of attacks. Side channel attacks does not affect the algortihm process.In this thesis, power analysis attack, one of the most important side channel attack, is mentioned in detail. Power analysis attack is big threat for cryptographic devices because the attack use only a oscilloscope and computer to attack.[8]

1.1.1 Power Analysis Attacks

In 1998, Power analysis attack is introduced by Kocher.[11] This attack tries to get the secret key by measuring power consumption. The attacker needs some equipments; a oscilloscope to collect power consumption and a computer to analyze obtaining data for revealing secret key. There are mainly three types of power analysis attacks simple power analysis, differential power analysis and correlation power analysis. [25]

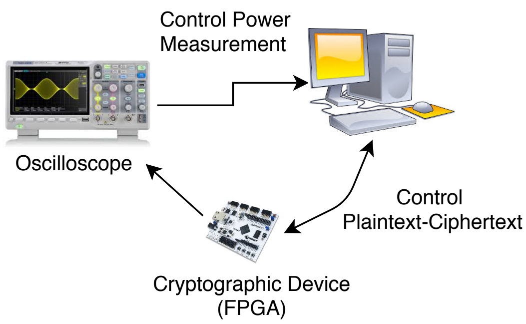

Figure 1.1: Side Channel Attack General Concept

1.1.1.1 Simple Power Analysis

Simple power analysis attack to obtain secret key by using power consumption of cryptographic devices. For this attack, details of implementation of cryptographic algorithm must be known and get a trace or few traces. In practice, this type of attack is not enough for succesful attack. At the same time, this attack helps to understand which algorithm works in the device. The attack is used with the other attack types.

1.1.1.2 Differential Power Analysis

By using large number of traces, this attack does not need information about the cryptographic device. It is enough to know which algorithm works in the device. This attack search for data dependency with power consumption. The attacker use statistical techniques after measuring power consumption.

Firstly, depending on algorithm characteristic, the attacker tries to decide intermediate value. Intermediate value which must depend on known plaintexts or ciphertexts and a part of secret key. After the deciding intermediate value of attack, power consumption are measured during encrypting known plaintexts.[8]

Let the attacker has n different plaintexts and $P = (P_1, P_2, ..., P_n)$ be the set of plaintexts , $T = (T_1, T_2, ..., T_n)$ be traces set and length of all T_i block is t and $T_i = (T_{i1}, T_{i2}, ..., T_{it})$ represent point on trace.

Next, power models are constituted for every possible key values according to intermediate values $f(p, k)$ of this encryption. To find model power traces, Hamming Distance model or Hamming Weight model is used.

P_1	P_2	...	P_{n-1}	P_n

Table 1.1: Set of Plaintexts

K_1	K_2	...	K_{m-1}	K_m

Table 1.2: Key possibilities

For all key possibilities, intermediate values are calculated by

$$\begin{pmatrix} f(P_1, K_1) & f(P_1, K_2) & ... & f(P_1, K_m) \\ f(P_2, K_1) & f(P_2, K_2) & ... & f(P_2, K_m) \\ \vdots & \vdots & \vdots & \vdots \\ f(P_n, K_1) & f(P_n, K_2) & ... & f(P_n, K_m) \end{pmatrix}_{nxm}$$

Table 1.3: Intermediate values of the attack

and traces for every plaintext are ;

$$\begin{pmatrix} T_{1,1} & T_{1,2} & ... & T_{1,t} \\ T_{2,1} & T_{2,2} & ... & T_{2,t} \\ \vdots & \vdots & \vdots & \vdots \\ T_{n,1} & T_{n,2} & ... & T_{n,t} \end{pmatrix}_{nxt}$$

Table 1.4: Traces $T_{i,j}$ of all plaintexts

Results by statistical analysis for all P_i by correlation coefficient ;

$$\begin{pmatrix} R_{1,1} & R_{1,2} & ... & R_{1,t} \\ R_{2,1} & R_{2,2} & ... & R_{2,t} \\ \vdots & \vdots & \vdots & \vdots \\ R_{m,1} & R_{n,2} & ... & R_{m,t} \end{pmatrix}_{mxt}$$

Table 1.5: Results

The highest values of the results show which key is probably used to encrypt selected intermediate value. If the all results are raughly same, attacker must measure more power consumption to reveal secret key.

Difference of Means : examines the relationship between power measurement and intermediate values of the attack by take into account least significant bit or most significant bit. [22]

Let intermediate value of this algorithm be output of s-box. Firstly, the s-box output are calculated for all key possibilities by known plaintext. Two groups are set according to least significant bit. Mean of measurements are calculated for two groups. After difference of these means for all key hypothesis are calculated, analyzing of these differences give best key hypothesis.

1.1.1.3 Correlation Power Analysis

Correlation power analysis attack is statistical power analysis attack by using Pearson correlation coefficient. Compared to differential power analysis, CPA attack need less power traces. [4]

Using plaintext(or ciphertext) and a part of key, intermediate value $f(p, k)$ are generated. Power models of intermediate values are calculated by Hamming Weight or Hamming Distance model for all key possibilities.

Definition 1. *Hamming Weight: is the number of ones in the binary sequence and denoted by*

$$HW(x) = \#1$$

where $x \in F_2^n$

Definition 2. *Hamming Distance: is the number of different bits between two binary sequences*

$$HD(x,y) = \#1 \quad \textit{of} \quad x \oplus y$$

where $x, y \in F_2^n$

Power measurements are taken during cryptographic algorithm. After measurement of power consumption, Pearson correlation coefficient is used for relation between power model and real power consumption.

Definition 3. ***Pearson Correlation Coefficient:*** *Let* x_i *and* y_i *are in different data groups,* n *is sample size and* $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ *,* $\bar{y}$ *has also same structure. Then,*

$$r = \frac{\sum_i x_i y_i + \bar{x}\bar{y}n - \bar{x}\sum_i y_i - \bar{y}\sum_i x_i}{\sqrt{\left(\sum_i {x_i}^2 + n\bar{x}^2 - 2\bar{x}\sum_i x_i\right)\left(\sum_i {y_i}^2 + n\bar{y}^2 - 2\bar{y}\sum_i y_i\right)}} \tag{1.1}$$

is Pearson Correlation Coefficient. r can have a value between -1 and 1 and shows that relationship between two data groups;

- *is strong negative when r is -1.*
- *is strong positive when r is 1.*
- *does not exist when 0.*

If r values are close enough to these values, they give information about the relationship between two groups.

1.1.2 Power Analysis Attacks Models for AES

AES algorithm is resistant to mathematical attacks and used in many cryptographic devices. This algorithm is sensitive to side channel analysis due to the features of the device used and how it is implemented. There are hypothesis power models using Hamming weight and Hamming distance which can be used for both DPA and CPA attacks for AES.

1.1.2.1 First Round Attack Model

First round attack model is constituted by using plaintext and S-box operation. AES has a 0^{th} round which is just consist of adding round key.

$P =$	P_0	P_1	...	P_{14}	P_{15}

Table 1.6: A plaintext 128 bit length with 16 byte representation

Let P be a plaintext and 128 bit length . Firstly, inverse shift operation is applied.

All key possibilities k is applied for every plaintext. Let $K_i = i$ where $i \in F_2^8$;

$K_i =$	k_0	k_1	...	k_{14}	k_{15}

Table 1.7: A key possibilities 128 bit length with 16 byte representation

Then,

$P + K_i =$	$P_0 + k_0$	$P_1 + k_1$	$P_2 + k_2$	...	$P_{14} + k_{14}$	$P_{15} + k_{15}$

After the adding key operation, output of S-box A.5 is calculated with the table for every key hypothesis.

```
Input: byte plaintext[nx1], int n , int out[nx256]
begin
byte state[nx1]
state=plaintext
for counter=0 to n-1 do
for key=0 to 255 do
state[counter]+key
sbox (state)+plaintext
hamming weight(state)
end for
end for
out state
end
```

Then, Hamming Distance of plaintexts and S-box [A.5] output of $P + k_i$'s are calculated. For every plaintexts, hypothesis power models are calculated with 256 key possibilities. Relation between power traces and hypothesis power models are examined with Pearson Correlation Coefficient[1.1].

1.1.2.2 Last Round Attack Model

The lack of column mixing in the final round of AES is a weakness for side channel analysis. The power model should be created with the S-box, which is the nonlinear, dependent to key and encrypted text operation of the AES algorithm.

Let C be a ciphertext and 128 bit length . Firstly, inverse shift operation is applied.

$C =$	C_0	C_1	...	C_{14}	C_{15}

Table 1.8: Set of Ciphertexts

$\Longrightarrow$ By Inverse Shift Operation

$C_{inv-shift} =$	C_0	C_5	C_{10}	C_{15}	C_4	C_9	C_{14}	C_3	C_8	C_{13}	C_2	C_7	C_{12}	C_1	C_6	C_{11}

Table 1.9: Inverse Shift Results

After the inverse shift operation, all key possibilities k apply to every ciphertext. Let $K_i = i$ where $i \in F_2^8$ and all length of K_i is 8 bit.,

$K_i =$	k_0	k_1	...	k_{14}	k_{15}

Then,

$C_{inv-shift} + K_i =$	$C_0 + k_0$	$C_5 + k_1$	$C_{10} + k_2$	...	$C_6 + k_{14}$	$C_{11} + k_{15}$

The inverse of this table of this process for all entries is given in Appendix A because inverse s-box [A.6] is necessary to use in the last round attack. Let index of S-box be x then x^{th} entry give the result inverse S-box of $x \in GF(2^8)$

```
Input: byte ciphertext[nx1], int n , int out[nx256]
begin
  byte state[nx1]
  state=ciphertext
  for counter=0 to n-1 do
    for key=0 to 255 do
      st1=invshift(state[counter])+key
      st2=inverse sbox (state)
      state +st1
      hamming weight(state)
    end for
  end for
  out state
end
```

Then Hamming Distance of $C_{inv-shift} + k_i$ and S-box outputs of $C_{inv-shift}$ is calculated. For every 8 bit of ciphertexts, there exist 256 hypothesis power models. Real power measurements and the relationship between these hypothesis power models are examined with Pearson Correlation Coefficient 1.1.

1.1.2.3 S-Box Input Output Model

This is also an Hamming Distance model 2. Since the confusion part of the AES in both the key schedule and the algorithm itself is provided by the s-box, all hypothesis power models are generated by the substitution box of AES.

Likewise 1.1.2.1, all key possibilities k is added to every plaintext. After the adding key operation, output of S-box is calculated with the table [A.5] for every key hypothesis. For the construction of the hypothesis power model, s-box output and input are summed.

```
Input: byte plaintext[nx1], int n , int out[nx256]
begin
  byte state[nx1]
  state=plaintext
  for counter=0 to n-1 do
    for key=0 to 255 do
      state[counter]+key
      sbox (state)
      state+plaintext+key
      hamming weight(state)
    end for
  end for
  out state
end
```

Three power analysis attack models for AES which is implemented as unprotected against side channel attacks are given in above. These attacks also depend number of power traces and characteristic of cryptographic devices. Since there is no mix column operation in the last round of AES, it creates a weakness against side channel attacks. Therefore, the last round attack amoung these three attack models provides the best correlation with power consumptions. In other attack models, correlation will be more powerful if more suitable power traces are taken.

CHAPTER 2

AES ALGORITHM AND SUBFIELD S-BOX CONSTRUCTION

2.1 Preliminaries

Definition 4. ***Field***:*By commutative ring R, an object $R \neq \varnothing$ together with second binary operation*

$$+ : R \times R \to R$$

$$. : R \times R \to R$$

Having this properties;

- *$\{R, +\}$ is an abelian group : $+(a, b = a + b)$*
- *$.(a, b) = a.b$ under this operation $\exists$ unit element denoted by 1, with denoted the property that $a.1 = 1.a = a \; \forall \in R$. R is commutative ring with $R^* = R \backslash 0$. Then, R is called a field and it is usually denoted by F. If F has finite elements, then it is called finite field. [15]*

Definition 5. ***Extention of field:*** *K is an extension of F, then K is a vector space over F. dim K over F is called degree of the extension $F \subset K$ and denoted by $[K : F] = dim_{F \backslash K}$*

$F \subset K$ field extension, α is algebraic over F of $\exists$ monic polynomial

$$g(x) = x^m + a_1 x^{m-1} + ... + a_n \in F[x] \quad \text{such that } g(\alpha) = 0$$

Definition 6. ***Trace:*** *Let F be a finite field and $F = F_{q^n}$ and $K = F_q$, then trace of F over K is*

$$Tr_{F \backslash K}(\alpha) = \sum_{i=0}^{n-1} \alpha^{q^i}$$

Definition 7. ***Norm:*** *Let $F = F_{q^n}$ be a finite field over $K = F_q$ and norm of F over K is*

$$N_{F \backslash K}(\alpha) = \prod_{i=0}^{n-1} \alpha^{q^i}$$

Definition 8. ***Polynomial Basis:*** *Let α be a primitive element of F over K, then*

$$\{1, \alpha, \alpha^2, \alpha^3, ..., \alpha^{n-1}\}$$

is a polynomial basis of F over K.

Definition 9. ***Normal Basis:*** *Let F be a extension of K with degree n. Then, $\alpha \in F$*

$$\{\alpha, \alpha^q, \alpha^{q^2}, ..., \alpha^{q^{n-1}}\}$$

is a normal basis of F over K.[17]

Definition 10. ***Boolean Function:*** *Let $f(x) : F_2^n \to F_2$ is a Boolean function which maps n bits to a single bit.*

$$f(x) = c_1 f_1(x) + c_2 f_2(x) + ... + c_n f_n(x)$$

where c_i are constant.

Definition 11. ***S-box:*** *S-box can be considered as a vector of Boolean functions. Let $S(x)$: $F_2^n \to F_2^m$ be a S-box which maps n bits to m bits. Each entry of $S(x)$ is a Boolean function.*

Definition 12. ***Affine function:*** *Let $f(x) = c_1 f_1(x) + c_2 f_2(x) + ... + c_n f_n(x) + C$ is an affine function where $f(x) : F_2^n \to F_2$ and C is a constant in F_2. If $C = 0$, then this function is called linear function and denoted by $l_c = cx$*

2.2 Block Ciphers

Block cipher is a cryptographic encryption method which works by dividing plaintext into blocks with fixed length. All blocks divided according to this method will be encrypted seperately, and the ciphertext will be obtained by the sequence of these blocks. [18]

For obtaining good block cipher depends on diffusion and confusion properties. ***Diffusion*** means that a character of a plaintext is changed then several characters of ciphertext should change. ***Confusion*** each character of the ciphertext should depend on several parts of key.

Permutation and substitution satisfy these two properties. There are two main structures of block ciphers. One of the structures is Substitiution Permutation Network(SPN). The structure is constituent of Advanced Encryption(AES). [23]

2.2.1 Advanced Encryption Standard and S-box Construction

AES which is the most widely used algorithm in block cipher, is a symmetric encryption algorithm. [3] In 2002, AES found a place among the encryption algorithms. AES is called Rijndael by the developers of this algorithm Vincent Rijmen and John Daemen. AES with 128 block length uses 128 bit, 192 bit and 256 bit length key alternatively. All operations are applied to 4x4 matrices. According to key length, the number of cycle change.

Key length	Number of Rounds
$128bit$	10
$192bit$	12
$256bit$	14

Table 2.1: Key sizes of AES algorithm

Each round consists of four layers in the AES algorithm. The algorithm's input output and matrices are 128 bits. These matrix is 4x4 and each entry is 8 bit length. Firstly, 128 bit data is converted to a 4x4 matrix.

Let plaintext be $[P_0, P_1, P_2, ..., P_{15}]$ and all P_i is 8 bit. Matrix form is

$$\begin{bmatrix} P_0 & P_1 & P_2 & P_3 \\ P_4 & P_5 & P_6 & P_7 \\ P_8 & P_9 & P_{10} & P_{11} \\ P_{12} & P_{13} & P_{14} & P_{15} \end{bmatrix}$$

Table 2.2: 128 bits plaintext matrix construction

There are four basic steps called layers respectively.

1. ByteSub(S-Box)
2. ShiftRow
3. MixColumn
4. AddRoundKey

Rjindael Encryption 0^{th} round is consists of AddRoundKey, 9 rounds of all four layers and the final round without Mixcolumn.

2.2.1.1 Substitution Box

For the subfield construction, we examine S-box construction of AES. We can describe the operations in $GF(2^8) = F_2[x]/ < x^8 + x^4 + x^3 + x + 1 >$. For computing ByteSub, we first compute the inverses of the entries of our matrix start with plaintext $x = x_7x_6x_5x_4x_3x_2x_1x_0$ where $x_i \in \{0, 1\}$ and $x \in GF(2^8)$. Then, compute the inverse of x, i.e compute $x^{-1} = y_7y_6y_5y_4y_3y_2y_1y_0 = y$.

Let $Sbox(x) = S = s_7s_6s_5s_4s_3s_2s_1s_0$, then

$$\begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Note that if $\alpha = \alpha_0 + \alpha_1 x + ... + \alpha_7 x^7 \in GF(2^8)$ then, $\alpha^{-1} = \beta_0 + \beta_1 x + ... + \beta_7 x^7 = \beta$ such that $\alpha.\beta = 1 mod f(x)$. In order to compute α^{-1}, Euclidean algorithm for polynomials is used.

2.2.1.2 Shift Row

The four rows of the state matrix are shifted cyclically. The method is that i^{th} row is shifted i times

$$\begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{bmatrix} \Longrightarrow \begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{11} & S_{12} & S_{13} & S_{10} \\ S_{22} & S_{23} & S_{20} & S_{21} \\ S_{33} & S_{30} & S_{31} & S_{32} \end{bmatrix}$$

Table 2.3: Shift Row operation

2.2.1.3 Mix Columns

Mix columns operation, which is a polynomial multiplication operation, is used for diffusion of this algorithm.

$$\begin{bmatrix} x & x+1 & 1 & 1 \\ 1 & x & x+1 & 1 \\ 1 & 1 & x & x+1 \\ x+1 & 1 & 1 & x \end{bmatrix} \begin{bmatrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{11} & S_{12} & S_{13} & S_{10} \\ S_{22} & S_{23} & S_{20} & S_{21} \\ S_{33} & S_{30} & S_{31} & S_{32} \end{bmatrix} = \begin{bmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{11} & M_{12} & M_{13} & M_{10} \\ M_{22} & M_{23} & M_{20} & M_{21} \\ M_{33} & M_{30} & M_{31} & M_{32} \end{bmatrix}$$

Table 2.4: Mix Column operation

2.2.1.4 Add Round Key

The main 128 bit key creates 4x4 matrix of key bytes. For the other round keys are obtained by 4 columns of this matrix.

$$\begin{bmatrix} K_{00} & K_{01} & K_{02} & K_{03} \\ K_{11} & K_{12} & K_{13} & K_{10} \\ K_{22} & K_{23} & K_{20} & K_{21} \\ K_{33} & K_{30} & K_{31} & K_{32} \end{bmatrix}$$

Table 2.5: Key expanded matrix

Let columns of the matrix be numarized by C_i where $i \in \{0, 1, ..., 43\}$. Then the construction of the other round keys;

- $i \not\equiv 0(mod4) \to C_i = C_{i-4} \oplus C_{i-1}$
- $i \not\equiv 0(mod4) \to C_i = C_{i-4} \oplus T(C_{i-1})$

T is a transformation which is consists of cyclic, substitution box and addition round constant. Firstly, take a column of the key matrix then shift cyclically. Let

$$C_i = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \Longrightarrow \begin{bmatrix} b \\ c \\ d \\ a \end{bmatrix} \Longrightarrow \begin{bmatrix} S(b) \\ S(c) \\ S(d) \\ S(a) \end{bmatrix}$$

After this operation, round constant of key operation is calculated by $r(i) = (00000010)^{(\frac{i-4}{4})} \in GF(2^8)$

2.3 S-box with Subfield Construction

S-box of AES is consists of $GF(2^8)$ multiplication with polynomial basis and constant addition. Irreducible function is $x^8 + x^4 + x^3 + x + 1$ and α be a root of this polynomial. Then the polynomial basis is $[\alpha^7, \alpha^6, \alpha^5, \alpha^4, \alpha^3, \alpha^2, \alpha, 1]$. Finding inverse element in $GF(2^8)$ is a hard operation and calculated by Euclidean algorithm. The inverse operation in $GF(2^8)$ can be calculated by combination of some subfield operations. [5] [6]

Firstly, a element $Y \in GF(2^8)$ can be shown over $GF(2^4)$ as $Y = y_1x + y_0$ and multiplication is calculated modular $f(x) = x^2 + rx + v$ 2 degree irreducible polynomial. Polynomial basis of $GF(2^8)/GF(2^4)$ is $[x, 1]$ and normal basis is $[x^{2^4}, x] = [x^{16}, x]$ [12] [20]

$$f(x) = x^2 + rx + \nu = (x + X)(x + X^{16})$$

so trace and norm equal to

$$Tr_{GF(2^8)/GF(2^4)} = r = X + X^{16} \quad \text{and} \quad Norm_{GF(2^8)/GF(2^4)} = \nu = (X)(X^{16}).$$

2.3.1 Inverse $GF(2^8)$ over $GF(2^4)$

Firstly, inverse operation in $GF(2^8)$ over $GF(2^4)$ be defined for construction of the S-box. Let $Y = y_1y + y_0$ and $D = d_1y + d_0$ be inverse of g $\in GF(2^8)$. For this inversion in the subfield $GF(2^4)$ an inversion, three multiplication, bitwise sum ($\oplus$), squaring and multiplication with norm are necessary. By the normal basis construction;

If D is inverse of Y, then $YD \equiv mod(x^2 + rx + v)$

$$\begin{aligned}
YD &= (y_1x + y_0)(d_1x + d_0)mod(x^2 + rx + v)\\
1 &= y_1d_1x^2 + y(y_1d_0 + y_0d_1) + y_0d_0mod(x^2 + rx + v)\\
1 &= y_1d_1x^2 + y(y_1d_0 + y_0d_1) + y_0d_0 + y_1d_1(x^2 + rx + v)\\
1 &= (y_1d_0 + y_0d_1 + y_1d_1r)x + (y_0d_0 + y_1d_1v)\\
1 &= 0x + 1
\end{aligned}$$

Because of the $YD = 1 = 0x + 1$;

$$0 = (y_1d_0 + y_0d_1 + y_1d_1r) \tag{2.1}$$

$$1 = (y_0d_0 + y_1d_1v) \tag{2.2}$$

by 2.1 and 2.2 equations are multiplied by y_0 and y_1 respectively

$$0 = y_1y_0d_0 + (y_0^2 + y_1y_0r)d_1 \tag{2.3}$$

$$y_1 = y_1y_0d_0 + y_1^2vd_1 \tag{2.4}$$

By equation 2.2 multiply with y_1, $y_1y_0d_0 = y_1 + y_1^2d_1v$ and from equaiton 2.3, equations in below are obtained.

$$\begin{aligned}
y_1 &= (y_0^2 + y_1y_0r + y_1^2v)d_1\\
y_1d_0 &= (y_0^2 + y_1y_0r + y_1^2v)d_1
\end{aligned}$$

Then, the inverse of Y in figure 2.1;

$$d_1 = (y_0^2 + y_1y_0 + y_1^2v)^{-1}y_1 \tag{2.5}$$

$$d_0 = (y_1^2v + y_1y_0r + y_0^2)^{-1}(y_0 + y_1r) \tag{2.6}$$

By 2.5 and 2.6 equation $[d_1, d_0]$,which represent the element Y, are ;

$$Y^{-1} = (y_1X^{16} + y_0X)^{-1} = (d_1X^{16} + d_0X) \tag{2.7}$$

$$= [((\nu \times (y_1 + y_0)^2) + y_1y_0)^{-1} + y_0]X^{16} \tag{2.8}$$

$$+ [((\nu \times (y_1 + y_0)^2) + y_1y_0)^{-1} + y_1]X \tag{2.9}$$

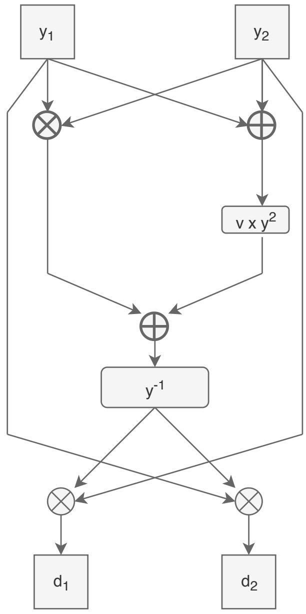

Figure 2.1: $GF(2^8)$ inversion on subfield $GF(2^4)$

2.3.2 Inverse $GF(2^4)$ over $GF(2^2)$

Equations 2.8 and 2.9 and figure 2.1 show that inverse in $GF(2^8)$ include $GF(2^4)$ inverse operation.

For the inversion $GF(2^4)$ over $GF(2^2)$, irreducible polynomial $s(z) = z^2 + Tz + N$ is used for multiplication operations. In $GF(2^4)$ of $y = G_1z + G_0$ and $d = D_1z + D_0$ be inverse of y then,

$$yd = (G_1D_0 + G_0D_1 + G_1D_1T)z + G_0D_0 + G_1D_1N$$

Then

$$\begin{aligned} D_1 &= (G_1^2N + G_1G_0T + G_0^2)^{-1}G_1 \\ D_0 &= (G_1^2N + G_1G_0T + G_0^2)^{-1}(G_0 + G_1T) \end{aligned}$$

2.3.3 Inverse $GF(2^2)$ over $GF(2)$

Similarly $GF(2^2)$ of $G = g_1w + g_0$ $D = h_1w + h_0$ be inverse of G and the irreducible polynomial is $t(w) = w^2 + w + 1$

$$\begin{aligned} 1 = GD &= (g_1h_0 + g_0h_1 + g_1h_1)w + (g_0h_0 + g_1h_1) \\ h_1 &= (g_1^2 + g_1g_0 + g_0^2)^{-1}g_1 \\ h_2 &= (g_1^2 + g_1g_0 + g_0^2)^{-1}(g_0 + g_1) \end{aligned}$$

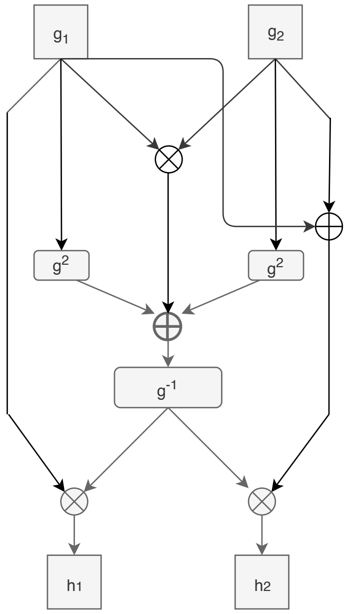

Figure 2.2: $GF(2^2)$ inversion on subfield $GF(2)$

2.3.4 Multiplication $GF(2^4)$ over $GF(2^2)$

For the $GF(2^4)$ multiplication, operations in $GF(2^2)$ are necessary. This operation 3 multiplication, 4 addition and multiplication with norm. Other operation in $GF(2^4)$ is combination of squaring and multiplication with scalar.

$$\begin{aligned} yd &= (G_1Z^4 + G_0) \times (D_1Z^4 + D_0) \\ &= [N \times [(G_1 + G_0) \times (D_1 + D_0)] + (G_1 \times D_1)]\, Z^4 \\ &+ [N \times [(G_1 + G_0) \times (D_1 + D_0)] + (G_0 \times D_0)]\, Z \\ &= P_1Z^4 + P_0Z \end{aligned}$$

where

$$\begin{aligned} P_1 &= [N \times [(G_1 + G_0) \times (D_1 + D_0)] + (G_1 \times D_1)] \\ P_0 &= [N \times [(G_1 + G_0) \times (D_1 + D_0)] + (G_0 \times D_0)] \end{aligned}$$

2.3.5 Multiplication $GF(2^2)$ over $GF(2)$

Multiplication in the $GF(2^2)$ has the same structure only multiplication by norm is different. Irreducible polynomial of $GF(2^2)/GF(2)$ is $t(x) = w^2+w+1$. Then, $Norm_{GF(2^2)/GF(2)} =$

$Tr_{GF(2^2)/GF(2)} = 1$

$$\begin{aligned}
GD &= (g_1w^2 + g_0w)(d_1w^2 + d_0w) \\
&= g_1d_1(w^2 + 1) + w^2w(g_1d_0 + g_0d_0) + g_0d_0w^2 \\
&= w^2g_1d_1 + w[(g_0 + g_1)(g_0 + d_1)] + g_0d_0w \\
&= w^2g_1d_1 + (w^2 + w)(g_0 + g_1)(d_0 + d_1) + g_0d_0w \\
&= w^2(g_1d_0 + g_0d_1 + g_0d_0) + w(g_0d_1 + g_1d_0 + g_1d_1)
\end{aligned}$$

2.3.6 Squaring $GF(2^4)$

Squaring in $GF(2^4)$ is the last operation to calculate inverse in $GF(2^8)$ where irreducible polynomial is $z^2 + Tz + N = (z + Z)(z + Z^4)$, $T = Z + Z^4$ and $N = ZZ^4$. For the squaring, some calculations are necessary;

$$\begin{aligned}
z^2 &= Tz + N \\
z^4 &= T^2z^2 + N^2 \\
&= z + T^2N + N^2
\end{aligned}$$

$$T = z^4 + z \tag{2.10}$$

$$NT = Nz^4 + Nz \tag{2.11}$$

$$N = \frac{N}{T}z^4 + \frac{N}{T}z \tag{2.12}$$

Let $Y = G_1z^4 + G_0z$

$$\begin{aligned}
Y^2 &= G_1^2z^8 + G_0^2z^2 \\
&= G_1^2z^8 + G_0^2(Tz + N) \\
&= G_1^2(Tz^4 + N) + G_0^2(Tz + N) \\
&= G_1^2Tz^4 + G_0^2Tz + N(G_1^2 + G_0^2) \\
&= G_1^2Tz^4 + G_0^2Tz + (\frac{N}{T}z^4 + \frac{N}{T}z)(G_1^2 + G_0^2) \quad \text{By 2.12} \\
&= z^4\left[G_1^2T + (G_1^2 + G_0^2)\frac{N}{T}\right] + z\left[G_0^2T + (G_1^2 + G_0^2)\frac{N}{T}\right] \\
\text{Let} \quad T = 1 & \\
&= z^4\left[G_1^2 + (G_1^2 + G_0^2)N\right] + z\left[G_0^2 + (G_1^2 + G_0^2)N\right]
\end{aligned}$$

For the squaring in $GF(2^4)$, squaring in $GF(2^2)$ is also needed. Squaring in $GF(2^2)$ where irreducible polynomial is $w^2 + w + 1 = 0$ and $G = g_0w + g_1$

$$\begin{aligned}
G^2 &= g_0^2w^2 + g_1^2 \\
&= g_0w^2 + g_1 \\
&= g_0(w + 1) + g_1 \\
&= g_0w + (g_0 + g_1)
\end{aligned}$$

Substitution Box of AES was constructed by using normal basis subfield operations. $GF(2^8)$ inverse operation with a normal basis was consists of an inversion and three multiplication in $GF(2^4)$, bitwise sum $(\oplus)$, squaring and multiplication with norm in $GF(2^2)$

CHAPTER 3

THRESHOLD IMPLEMENTATION

3.1 Threshold Implementation

In this section, some definitions and properties about Threshold Implementation. Then, these properties are applied to some examples. Threshold Implementation is a method applied to function. Input of these function must be satisfied two properties which are *Correctness* and *Uniform Masking*. These properties are also used by all masking methods. Functions of Threshold Implementation depends on three properties constitutively.[2]. Threshold implementation is based two information sharing methods in below.

Definition 13. ***Multiparty Computation:*** *Let n different parties* $[P_1, P_2, ..., P_n]$ *has different input* $[x_1, x_2, ..., x_n]$*. Then, Multiparty computation is a protokol let that* P_i *only learns the value* y_i *where* $f(x_1, x_2, ..., x_n) = (y_1, y_2, ..., y_n)$

Definition 14. ***Shamir Secret Sharing Scheme:*** *[24] Let secret information S be* $[S_1, S_2, ..., S_n]$*. This information is shared that if the knowledge of k parts of secret are enough to know secret S, then* $k-1$ *part does not reveal any information about S.*

This method is called threshold scheme and denoted by $[k, n]$*. The case of* $k = n$ *requires all parts of secret to compute S. Threshold implementation use the case of* $k = n$

These two definitions are the basis of Threshold Implementation. Let $f(x) = y$ where F_2^n to F_2^m, firstly sensitive variable x is shared;

Definition 15. *(Sharing) Let* $X \in F^m$ *and* s *be number of shares.To share all entities of* $\vec{X} = (x, y, z, ..., t)$*;*

1. *Generate random bit shares of entity up to* $s-1$ *and then,*
2. s^{th} *share be equal to* $\sum_{s-1}^{i=1} = x_i$ *to satisfy* $x = x_1 + x_2 + ... + x_s$*,* $y = y_1 + y_2 + ... + y_s$ *,...,*$t = t_1 + t_2 + ... + t_s$*.*

This method is also used in Boolean Masking. Then, $\vec{\bar{x}} = (x_1, x_2, .., x_s)$ is called share vector of sensitive variable of x and x_i denote the share vector without x_i term where $i \in 1, 2, ..., s$.

Property 1. *Let* $N(\vec{\tilde{x}}) = \#\{\tilde{x} = (x_1, x_2, ..., x_s) : x_1 + x_2 + .. + x_s = x \in F\}$*. If* $N(\vec{\tilde{x}}) = n$ *where* $n \in Z \quad \forall x \in F^m$*, then the masking is uniform.*

In words, if for each sensitive value x, number of share vectors of x is constant, then this masking is uniform.

A d^{th} order masking of a variable x is obtained by seperating $d + 1$ random x_i where $i \in \{1, 2, ..., d + 1\}$. Given sharing of input, threshold implementation can apply linear and nonlinear function with using this sharing. [10]

$F = (f_1, f_2, ..., f_t)$ is vector of functions where f_i component function and t is the number of component functions. For the threshold implementation, $\forall f_i : F_q^m \to F_q$ must satisfied three properties.

Property 2. (Correctness) *Let* $F(X) = Y = Y_1 + Y_2 + ... + Y_t = f_1 + f_2 + ... + f_t$, $\forall$ $Y \in F_2^n$ $Y = f_1(x) + ... + f_t(x)$ $\forall x \in F_2^m$

Example 1. *Let* $F(x, y, z) = xy + z$
$F(x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) = f_1 + f_2 + f_3$ *where*

$$f_1 = x_2y_2 + x_2y_3 + x_3y_2 + z_2$$
$$f_2 = x_1y_3 + x_3y_1 + x_3y_3 + z_3$$
$$f_3 = x_1y_2 + x_2y_1 + x_1y_1 + z_1$$
$$f_1 + f_2 + f_3 = (x_1 + x_2 + x_3)(y_1 + y_2 + y_3) + z_1 = xy + z$$

3.1.1 Threshold Implementation of Linear Functions

Let $l(x) = y$ be linear function over $GF(2)$. If s is the number of shares. Then input and output shares;

$$x = x_1 \oplus x_2 \oplus ... \oplus x_s \quad \text{and} \quad y = y_1 \oplus y_2 \oplus ... \oplus y_s.$$

By the linearity of the function;

$$y = y_1 \oplus y_2 \oplus ... \oplus y_s = l(x_1) \oplus l(x_2) \oplus ... \oplus l(x_s) = l(x)$$

Like above equation, threshold implementation of linear function is constructed by applying the function to different shares of x.

3.1.2 Threshold Implementation of Nonlinear Functions

To construct a nonlinear function $F(X) = Y$ where $X \in F_2^m$ and $Y \in F_2^n$ according to Threshold Implementation need two more properties *Noncompleteness* and *Uniformity of* *function*.

Property 3. (Noncompleteness) *If all F_i is independent from at least one x_i, then F satisfy non-completeness property of Threshold Implementation.*

$$
\begin{aligned}
z_1 &= F_1(x_2, x_3, ..., x_n, y_2, y_3, ..., y_n, ...) \\
z_2 &= F_2(x_1, x_3, ..., x_n, y_1, y_3, ..., y_n, ...) \\
&\vdots \\
z_n &= F_n(x_1, x_2, ..., x_{n-1}, y_1, y_2, ..., y_{n-1}, ...)
\end{aligned}
$$

In Example 1, all component are independent from at least a share of input.

Corrollary 1. *A d^th degree function can be shared with at least $d+1$ shares to satisfy* noncompletenesss *property.*

While these properties are easily satisfied, it is not easy to provide the next feature that gives near-linear qualification to the Threshold Implementation functions.

Property 4. *Let $X \in F_2^m$ and $Y \in F_2^n$ with $F(X) = Y$*
$N(a, b, c, ...) = \#\{(x_i, y_j, ...) : F_1(...) = a, F_2(...) = b, , ..., F_t(...) = k$ where $i, j \in \{1, ..., s\}\}$ and t is number of shares. F has uniformity if and only if $N(a_i, b_i, ..., k_i) = N(a_j, b_j, ..., k_j)$ where $a_i \oplus b_i \oplus c_i \oplus ... \oplus k_i = a_j \oplus b_j \oplus c_j \oplus ... \oplus k_j$ where $\forall i, j \in \{1, .., t\}$

Example 2. *$F(X, Y) = XY$ with four shares First order noncompleteness Threshold Implementation)*

$$
\begin{aligned}
f_1 &= (x_3 + x_4)(y_2 + y_4) + y_2 + x_2 + y_4 \\
f_2 &= (x_1 + x_3)(y_1 + y_4) + y_1 + x_1 + y_4 \\
f_3 &= (x_2 + x_4)(y_1 + y_4) + y_2 + y_3 + x_2 + x_3 + x_4 \\
f_4 &= (x_1 + x_2)(y_2 + y_3) + y_1 + y_3 + x_1 + x_3 + x_4
\end{aligned}
$$

$f = f_1 \oplus f_2 \oplus f_3 \oplus f_4$	*Times of appearance*
0	*20*
1	*12*

3.1.3 Methods for Construction Threshold Implementation

3.1.3.1 Direct Sharing

Direct Sharing is a method to construct functions which satisfy three properties of Threshold Implementation. By this method, first two properties are easily satisfied. However, this method does not guarantee to give component functions which satisfy uniformity.

Construction 1. *First order direct sharing construction for quadratic functions: Let* $f : F_2^n \to F_2^m$ *be a quadratic function.* $f = \sum_{i=1}^{t} f_i$ *, where* t *is the share of share number of function and* $x = \sum_{i=1}^{s} x_i$ *,* $y = \sum_{i=1}^{s} y_i$ *where* s *is the share number of input. Then,*

- *If the linear term exists, then* $\{i\}$ *and* $\{i+1\}$ *share of term, these shares are in* $\{i-1\}^{th}$ *component function.*
- *Quadratic terms with only* $\{i\}$ *shares and* $\{i, i+1\}$ *shares are in also* $\{i-1\}^{th}$ *component function.*

Example 3. *(Quadratic Example) Let* $F(x, y) = xy + y$ *and* $x = \sum_{i=1}^{3} x_i$ *and* $y = \sum_{i=1}^{3} y_i$

$$f_1 = x_2y_2 + x_2y_3 + x_3y_2 + y_2;$$
$$f_2 = x_3y_3 + x_3y_1 + x_1y_3 + y_3;$$
$$f_3 = x_1y_1 + x_1y_2 + x_2y_1 + y_1;$$

$\{f_1, f_2, f_3\}$	*Total*	*x=y=0*	*x=0,y=1*	*x=1,y=0*	*x=1,y=1*
000	*21*	*7*	*0*	*7*	*7*
001	*5*	*0*	*5*	*0*	*0*
010	*5*	*0*	*5*	*0*	*0*
011	*9*	*3*	*0*	*3*	*3*
100	*5*	*0*	*5*	*0*	*0*
101	*9*	*3*	*0*	*3*	*3*
110	*9*	*3*	*0*	*3*	*3*
111	*1*	*0*	*1*	*0*	*0*

For the higher degree functions, there is no method directly. However, similar method can be used for higher degree function.

Construction 2. First Order Direct Sharing for Cubic Functions:

Let $f : F_2^n \to F_2^m$ be a cubic function, $f = \sum_{i=1}^{t} f_i$, where t is the share of share number of function and $x = \sum_{i=1}^{s} x_i$, $y = \sum_{i=1}^{s} y_i$, and $z = \sum_{i=1}^{s} z_i$ where s is the share number of input. Then;

- *If the linear term exists, then $\{i + 1\}$ share of inputs are in $\{i^{th}\}$ component function.*
- *If the quadratic term exists, then $\{i + 1, i + 1\}$,$\{i + 1, i + 2\}$,$\{i + 1, i + 3\}$ and $i + 3, i + 2$ are in also $\{i^{th}\}$ component function.*
- *Cubic terms whose first and second entries indexes are $\{i+1, i+2\}$, with mixed indexes such as $\{i+1, i+2, i-1\}$ and the last one $\{i-1, i-1, i+2\}$ are also in $\{i^{th}\}$ component function.*

Example 4. *(Cubic Example) Let $F(x, y) = xyz + yz$ and $x = \sum_{i=1}^{4} x_i$, $y = \sum_{i=1}^{4} y_i$ and $z = \sum_{i=1}^{4} z_i$*

$$
\begin{aligned}
f_1 = {} & x_2y_2z_2 + x_2y_3z_2 + x_2y_2z_3 + x_2y_3z_4 + x_2y_4z_3 + x_2y_2z_4 + \\
& x_2y_4z_2 + x_2y_4z_4 + x_2y_3z_3 + x_4y_3z_2 + x_3y_4z_2 + x_4y_2z_3 + \\
& x_3y_2z_4 + x_4y_3z_3 + x_4y_4z_3 + x_4y_3z_4 + y_2z_2 + y_2z_3 + y_2z_4 + y_4z_3 \\
f_2 = {} & x_3y_3z_3 + x_3y_4z_3 + x_3y_3z_4 + x_3y_4z_1 + x_3y_1z_4 + x_3y_3z_1 + \\
& x_3y_1z_3 + x_3y_1z_1 + x_3y_4z_4 + x_1y_4z_3 + x_4y_1z_3 + x_1y_3z_4 + \\
& x_4y_3z_1 + x_1y_4z_4 + x_1y_1z_4 + x_1y_4z_1 + y_3z_3 + y_3z_4 + y_3z_1 + y_3z_4 \\
f_3 = {} & x_4y_4z_4 + x_4y_1z_4 + x_4y_4z_1 + x_4y_1z_2 + x_4y_2z_1 + x_4y_4z_2 + \\
& x_4y_2z_4 + x_4y_2z_2 + x_4y_1z_1 + x_2y_1z_4 + x_1y_2z_4 + x_2y_4z_1 + \\
& x_1y_4z_2 + x_2y_1z_1 + x_2y_2z_1 + x_2y_1z_2 + y_4z_4 + y_4z_1 + y_4z_2 + y_2z_1 \\
f_4 = {} & x_1y_1z_1 + x_1y_2z_1 + x_1y_1z_2 + x_1y_2z_3 + x_1y_3z_2 + x_1y_1z_3 + \\
& x_1y_3z_1 + x_1y_3z_3 + x_1y_2z_2 + x_3y_2z_1 + x_2y_3z_1 + x_3y_1z_2 + \\
& x_2y_1z_3 + x_3y_2z_2 + x_3y_3z_2 + x3y_2z_3 + y_1z_1 + y_1z_2 + y_1z_3 + y_3z_2
\end{aligned}
$$

After constructing component functions, the hardest part is obtaining uniformity so, there are some methods to provide uniformity of shared functions.

3.1.3.2 Remasking

Definition 16. Remasking: *Let $F : F_2^n \to F_2^m$ be function. Then component functions of F will be $\{f_1, f_2, ..., f_t\}$ where $F = \sum_{i=1}^{t} f_i$ and the components f_i does not satisfy uniformity.*

- *Generate $t-1$ random number such that $m_1, m_2, ..., m_{t-1}$ to add f_i components where $i \in \{0, 1, ..., t-1\}$*
- *Add m_t to last component f_t where $m_t = \sum_{i=1}^{t-1} m_i$*

$$\begin{aligned} f_1^* &= f_1 + m_1 \\ f_2^* &= f_2 + m_2 \\ \vdots &= \vdots \\ f_t^* &= f_1 + \sum_{i=1}^{t_1} m_i \end{aligned}$$

İt is last choice to construct uniform sharing functions for Threshold Implementation because of the fact that finding fresh random numbers is expensive operation.

3.1.3.3 Increasing the number of input shares

To keep cost low, share number must be kept minimum but it is not possible for every function. Increasing number of input creates new spaces to find function construction. By direct sharing method, uniform function construction cannot be obtained everytime. By increasing the shares, function which is satisfied all properties can be founded.

Example 5. *Let $F(x, y) = yz + x$ and $x = \sum_{i=1}^{4} x_i$, $y = \sum_{i=1}^{4} y_i$ and $z = \sum_{i=1}^{4} z_i$*

$$\begin{aligned} f_1 &= x_2 + (y_2 + y_3 + y_4)(z_2 + z_3 + z_4) \\ f_2 &= x_3 + y_1(z_3 + z_4) + z_1(y_3 + y_4) + y_1 z_1 \\ f_3 &= x_4 + y_1 z_2 + y_2 z_1 \\ f_4 &= x_1 \end{aligned}$$

3.1.3.4 Correction Terms

After satisfying property 1 and 2, using correction terms expand the possible sharing functions. In some cases, it may be more useful than increasing share of input. [21]

Definition 17. *Correction term is a term which is that can be added than more than one component to construct uniform functions. Let $F: F_2^n \rightarrow F_2^m$ and $deg(F) = d$. Then;*

- *For the noncompleteness property, (w.l.o.g) terms with (i,j) indices, can be used as a correction term in all components except f_i, f_j component function*
- *If there is no bound about term degree, higher degree terms is usable as correction term.*

Corrollary 2. *For the first order direct sharing of function with degree d, up to d-1 degree terms are used as a correction term to satisfy all properties.*

Example 6. *Let $F = XY$ and because of the noncompleteness property x_4 and y_4 can be used as a correction term.*

$$F_1 = (x_2 \oplus x_3 \oplus x_4)(y_2 \oplus y_3) \oplus y_4$$
$$F_2 = (x_1 \oplus x_3)(x_1 \oplus y_4) \oplus (x_1 y_3) \oplus x_4$$
$$F_3 = (x_2 \oplus x_4)(x_1 \oplus y_4) \oplus (x_1 y_2) \oplus x_4 \oplus y_4$$

3.2 Threshold Implementation of AES algorithm

Security of Threshold Implementation is proved against first order power analysis attacks and is applicable for all algorithms used. Providing all the features of Threshold Implementation becomes difficult as degree of function increases. AES algorithm works in $GF(2^8)$ field. Even if the first order Threshold Implementation is used for the AES algorithm, at least 9 shares function must be used. Assuming that this function exists, it will need large area on embedded devices such as a smart card.

After Canright construction, it was possible to protect the AES algorithm with Threshold Implementation. There different types of Threshold Implementation for AES. In this section, A Threshold Implementation is applied to the AES algorithm and S-box construction will be discussed.

3.2.1 Raw Implementation

Threshold Implementation has become feasible after Canright construction for AES. In applications in implemented in embedded devices, it is important that the algorithm takes up little area and works in a short time. Therefore, there is a trade off between share numbers and time consuming of Threshold Implementations of AES. In Raw Implementation, the number of shares was kept small and as constant as possible.

AES is consist of four layers. Before these four operation plaintext are shared by four shares. The most important reason is that all operation is in $GF(2^4)$. After that two shares is used for Add Round Key operation which is the 0^{th} round operation, then these two shares are added to any two shares of plaintexts. By the same way, Mix column operation is worked for two shares simultaneously.

Implementation of Substition Box, which is important and nonlinear part of AES, is detailed and functions of Raw Implementation of AES is analyzed in terms of constructiom method and properties of Threshold Implementation. Lastly, Shift row operation is implemented as normal version.

3.2.1.1 Raw Implementation of AES S-Box

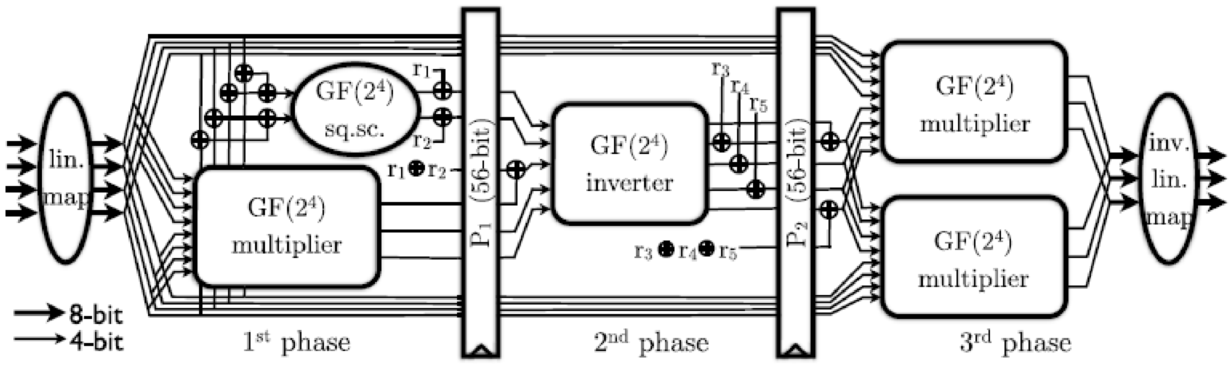

Figure 3.1: Raw Implementation of AES S-box[1]

Raw implementation of S-box use $GF(2^4)$ tower field construction for nonlinear operation. It is known that 3 times $GF(2^4)$ multiplication, a $GF(2^4)$ inverse and a $GF(2^4)$ square scalar are used for construction of S-box.

Linear Map and Inverse Linear Map(Change of Basis): Linear map and inverse linear map in figure represent transformation of basis from polynomial basis to normal basis and vice versa. Any x element in $GF(2^8)$is represented by polynomial basis in AES algorithm. However, Raw Implementation use normal basis construction so input of S-box must be represented by normal basis before S-box operation.

Let x be element $GF(2^8)$ then x can be represented as a vector over $GF(2)$.

$$x = x_0 + x_1\alpha + x_2\alpha^2 + x_3\alpha^3 + x_4\alpha^4 + x_5\alpha^5 + x_6\alpha^6, x_7\alpha^7$$

where $\{\alpha, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7\}$ is polynomial basis and normal basis from $GF(2^8)$ to $GF(2^4)$ be $[y^{16}, y]$. Then, for construction of normal basis in Appendix A [Table A.1].

$$
\begin{aligned}
y &= y_1x^{16} + y_0x \\
&= (G_1z^4 + G_0z)x^{16} + (G_{1,2}z^4 + G_{0,2}z)x \\
&= [(\beta_7w^2 + \beta_6w)z^4 + (\beta_5w^2 + \beta_4w)z]x^{16} + [(\beta_3w^2 + \beta_2w)z^4 + (\beta_1w^2 + \beta_0w)z]y \\
&= \beta_7w^2z^4y^{16} + \beta_6wz^4y^{16} + \beta_5w^2zy^{16} + \beta_4wzy^{16} + \beta_3w^2z_4y + \beta_2wz^4y + \beta_1w^2zy + \beta_0wzy
\end{aligned}
$$

Inverse linear map shows changing basis from normal to polynomial in Appendix A [Table A.2].

After this converting, Square scalar and $GF(2^4)$ Multiplication are applied to 4 shares of 8-bit inputs.

GF($\mathbf{2^4}$) Multiplication: Firstly 4 shares of the s-box input ,which are shared at the beginning of algorithm, are used as input of $GF(2^4)$. Every input of multiplication has 4 shares such as $x_1 = x_{11} \oplus x_{12} \oplus x_{13} \oplus x_{14}$. Then, at the end of $GF(2^4)$ multiplication, there exist 3 output shares for all component $GF(2^4)$.

Let $x = (x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ is vectorial representation of any element x in $GF(2^8)$ and x_1 be most significant bit and x_8 be least significant bit. Then,

$$F = (x_1, x_2, x_3, x_4)(x_5, x_6, x_7, x_8)$$

is $GF(2^4)$ multiplication

$$
\begin{aligned}
F_1 &= x_1x_5 + x_3x_5 + x_4x_5 + x_2x_6 + x_3x_6 + x_1x_7 + x_2x_7 + x_3x_7 + x_4x_7 + x_1x_8 + x_3x_8 \\
F_2 &= x_2x_5 + x_3x_5 + x_1x_6 + x_2x_6 + x_4x_6 + x_1x_7 + x_3x_7 + x_2x_8 + x_4x_8 \\
F_3 &= x_1x_5 + x_2x_5 + x_3x_5 + x_4x_5 + x_1x_6 + x_3x_6 + x_1x_7 + x_2x_7 + x_3x_7 + x_1x_8 + x_4x_8 \\
F_4 &= x_1x_5 + x_3x_5 + x_2x_6 + x_4x_6 + x_1x_7 + x_4x_7 + x_2x_8 + x_3x_8 + x_4x_8
\end{aligned}
$$

where F_1, F_2, F_3, F_4 are component functions of F. Threshold Implementation function is applied to all terms in this components such that x_2x_5 is shared by 4-3 TI function below. After the sharing operation, there are 3 shares for every component function so, there are 12 shares for F_1, F_2, F_3, F_4 end of the sharing

$F = F_1 + F_2 + F_3 = X_iX_j$ where $X_i = x_{i1} \oplus x_{i2} \oplus x_{i3} \oplus x_{i4}$ and $X_j = x_{j1} \oplus x_{j2} \oplus x_{j3} \oplus x_{j4}$ [1]

$$
\begin{aligned}
F_1 &= (x_{i2} \oplus x_{i3} \oplus x_{i4})(x_{j2} \oplus x_{j3}) \oplus x_{j4} \\
F_2 &= (x_{i1} \oplus x_{i3})(x_{j1} \oplus x_{j4}) \oplus (x_{i1}x_{j3}) \oplus x_{i4} \\
F_3 &= (x_{i2} \oplus x_{i4})(x_{j1} \oplus x_{j4}) \oplus (x_{i1}x_{j2}) \oplus x_{i4} \oplus x_{j4}
\end{aligned}
$$

This function is constructed by using increasing number of input and decreasing number of output. At the same time, correction term is used to satisfy the all properties.

$f = f_1 \oplus f_2 \oplus f_3 \oplus f_4$	Times of appearance
0	48
1	16

When implementing this cascaded and parallel functions, one of the things that need attention is uniformity of function is still satisfied. The same function as a composite function may result in loss of uniformity.

Example 7. *Let $F(X,Y) = XY + YZ$ and $X = \sum_{i=1}^{4} x_i$,$Y = \sum_{i=1}^{4} y_i$ and $Z = \sum_{i=1}^{4} z_i$*

$$f_1 = (x_2 + x_3 + x_4)(y_2 + y_3) + y_4 + (y_2 + y_3 + y_4)(z_2 + z_3) + z_4$$
$$f_2 = ((x_1 + x_3)(y_1 + y_4)) + x_1y_3 + x_4 + ((y_1 + y_3)(z_1 + z_4)) + y_1z_3 + y_4$$
$$f_3 = ((x_2 + x_4)(y_1 + y_4)) + x_1y_2 + x_4 + y_4 + ((y_2 + y_4)(z_1 + z_4)) + y_1z_2 + z_4 + y_4$$

$f = f_1 \oplus f_2 \oplus f_3 \oplus f_4$	*Times of appearance*
0	*768*
1	*256*

Let $F(X,Y) = XY + ZY = (X + Z)Y$ and $X = \sum_{i=1}^{4} x_i$,$Y = \sum_{i=1}^{4} y_i$ and $Z = \sum_{i=1}^{4} z_i$

$$f_1 = (x_2 + x_3 + x_4)(y_2 + y_3) + y_4 + (z_2 + z_3 + z_4)(y_2 + y_3) + y_4$$
$$f_2 = ((x_1 + x_3)(y_1 + y_4)) + x_1y_3 + x_4 + ((z_1 + z_3)(y_1 + y_4)) + z_1y_3 + z_4$$
$$f_3 = ((x_2 + x_4)(y_1 + y_4)) + x_1y_2 + x_4 + y_4 + ((z_2 + z_4)(y_1 + y_4)) + z_1y_2 + z_4 + y_4$$

$\{f_1, f_2, f_3, f_4\}$	*Times of appearance*
000	*1152*
001	*384*
010	*384*
011	*1152*
100	*128*
101	*384*
110	*384*
111	*128*

Threshold implementation of $GF(2^4)$ inverse operation need 5 shares. Before inverse operation there are three output shares from $GF(2^4)$ multiplication. Two more shares is necessary for invertion. Square scalar is a linear operation and work in parallel for two shares. However, there is no guarantee for the uniformity of these two functions' outputs. Because if outputs of these two functions are supposed same functions output, uniformity is not satisfied. It is known that $GF(2^4)$ multiplication is uniform so masking output of square scalar is enough. Therefore, random variables $[r_1, r_2]$are added for two output of square scalar. Then, $r_1 + r_2$ is added to one output of $GF(2^4)$ multiplication.

$\mathbf{GF(2^4)}$ **Inverse Function** Inverse for the $x \in GF(2^4)$ where the component functions are

$$\begin{aligned}
Y_1 &= x_2x_3x_4 + x_2x_3 + x_2 + x_1x_3 + x_3 \\
Y_2 &= x_1x_3x_4 + x_1x_3 + x_4 + x_2x_3 + x_2x_4 \\
Y_3 &= x_1x_4x_2 + x_1x_4 + x_2 + x_1x_3 + x_1 \\
Y_4 &= x_1x_3x_2 + x_1x_3 + x_2 + x_1x_4 + x_2x_4
\end{aligned}$$

By using 5 shares Threshold Implementation below, first order resist implementation of inverse operation is formed. This Threshold Implementation is applied to bitwise operation.

Let $F = XYZ + XY + Z$ [1]

$$\begin{aligned}
F &= F_1 + F_2 + F_3 + F_4 + F_5 \\
F_1 &= [(x_2 + x_3 + x_4 + x_5)(y_2 + y_3 + y_4 + y_5)(z_2 + z_3 + z_4 + z_5)]+ \\
&\quad [(x_2 + x_3 + x_4 + x_5)(y_2 + y_3 + y_4 + y_5)] + z_2 \\
F_2 &= [x_1(y_3 + y_4 + y_5)(z_3 + z_4 + z_5) + y_1(x_3 + x_4 + x_5)(z_3 + z_4 + z_5)+ \\
&\quad z_1(x_3 + x_4 + x_5)(y_3 + y_4 + y_5) + x_1y_1(z_3 + z_4 + z_5)+ \\
&\quad x_1z_1(y_3 + y_4 + y_5) + y_1z_1(x_3 + x_4 + x_5) + x_1y_1z_1]+ \\
&\quad [x_1(y_3 + y_4 + y_5) + y_1(x_3 + x_4 + x_5) + x_1y_1] + z_3 \\
F_3 &= (x_1y_1z_2 + x_1y_2z_1 + x_2y_1z_1 + x_1y_2z_2 + x_2y_1z_2+ \\
&\quad x_2y_2z_1 + x_2y_1z_4 + x_1y_2z_4 + x_1y_4z_2 + x_2y_4z_1+ \\
&\quad x_4y_1z_2 + x_4y_2z_1 + x_1y_2z_5 + x_2y_1z_5 + x_1y_5z_2+ \\
&\quad x_2y_5z_1 + x_5y_1z_2 + x_5y_2z_1) + (x_1y_2 + y_1x_2) + z_4 \\
F_4 &= (x_1y_2z_3 + x_1y_3z_2 + x_2y_1z_3 + x_2y_3z_1 + x_3y_1z_2 + x_3y_2z_1) + z_5 \\
F_5 &= z_1
\end{aligned}$$

$f = f_1 \oplus f_2 \oplus f_3 \oplus f_4$	Times of appearance
0	768
1	1280

Table 3.1: Uniformity of 5-5 shares function

This first order noncompleteness 5 shares Threshold Implementation is also constructed by increasing input and output shares but correction term is not used. Squaring scalar part is just copied for two different shares because multiplication has 3 output and inverse operation need 5 shares. For the output of square scalar algorithm two random numbers in $GF(2^4)$ are used.

For the $GF(2^4)$ multiplication after inverse operation, 3 random variables are added to outputs of inverse operation because multiplication inputs must be uniform and need 4 shares. By adding one output to another one, 4 input shares is obtained and by the 3 random 4-bit numbers uniformity is satisfied. In totally, 5 random 4-bit numbers are used for the S-box operation.

After the last two $GF(2^4)$ multiplication, 3 output shares are gotten. The last operation is the inverse linear operation is converting basis from normal basis to polynomial basis. The other operations of AES are needed two shares so, one of the three shares whic are the S-box output is added to one of the others.

Operation except s-box works on two shared input. 24 random bits are used to increase number of shares for substitution box of other rounds.

There are three types for the construction of AES with Threshold Implementation. Raw Implementation of AES use minimum share numbers for the implementation and based for the others.

CHAPTER 4

CONCLUSION

Side channel attack is a parameter for testing security of an cryptographic algorithms and devices. Power analysis attack is an important role symmetric and asymmetric cryptographic algorithm. There are so many side channel attack types for AES which is most widely used cryptographic algorithm.

In this thesis, in Chapter 1, we research side channel attack and we investigate that power analysis attack are used especially for AES. We mentioned and compare according to reveals of AES algorithm that three of these attacks when the algorithm is implemented without any protection for SCA.

In Chapter 2, we give details of subfield construction for substitution box of AES which is necessary to countermeasure Threshold Implementation against side channel attack for AES.

In Chapter 3, we focused on properties and construction methods of Threshold Implementation with explanatory examples.

In conclusion, Threshold Implementation functions which are used for AES are examined. A Threshold Implementation of AES whose security is proved against first order power analysis attack and base for other type TI of AES is detailed.

REFERENCES

[1] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, Trade-offs for threshold implementations illustrated on aes, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34(7), pp. 1188–1200, July 2015, ISSN 0278-0070.

[2] B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, Threshold implementations as a countermeasure against higher order differential power analysis, University of Twente, 7, 05 2015.

[3] B. Bilgin, S. Nikova, V. Nikov, V. Rijmen, N. Tokareva, and V. Idrisova, Threshold implementations of small s-boxes, Cryptography and Communications, 7, 03 2015.

[4] E. Brier, C. Clavier, and F. Olivier, Correlation power analysis with a leakage model, Springer, 3156, pp. 16–29, 03 2004.

[5] D. Canright, A very compact rijndael s-box, 2004.

[6] D. Canright, A very compact s-box for aes, Cryptographic Hardware and Embedded Systems, 3659, pp. 441–455, 2005.

[7] D. Canright and L. Batina, A very compact "perfectly masked" s-box for aes, Springer, 5037, pp. 446–459, 06 2008.

[8] C. Clavier, J. Coron, and N. Dabbous, Differential power analysis in the presence of hardware countermeasures, Springer, 1965, pp. 252–263, 01 2000.

[9] T. De Cnudde, B. Bilgin, O. Reparaz, V. Nikov, and S. Nikova, Higher-order threshold implementation of the aes s-box, Springer, 9514, pp. 259–272, 03 2016.

[10] M. Dworkin, E. Barker, J. Nechvatal, J. Foti, L. Bassham, E. Roback, and F. James, Advanced encryption standard (aes), NIST Pubs, 197, November 2001.

[11] P. Kocher, J. Jaffe, and J. B., Differential power analysis, Advances in Cryptology, 1666, pp. 388–397, December 1999.

[12] P. Kocher, J. Jaffe, and B. Jun, A very compact s-box for aes, Springer, 3659, pp. 441–455, 08 2005.

[13] P. Kocher, J. Jaffe, and B. Jun, Pushing the limits: A very compact and a threshold implementation of aes, Springer, 6632, pp. 69–88, 11 2011.

[14] P. Kocher, J. Jaffe, and B. Jun, A more efficient aes threshold implementation, Springer, 8469, pp. 267–284, 11 2014.

[15] R. Lidl, H. Niederreiter, and T. Popp, *Finite Fields*, Cambridge University Press, 1996.

[16] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*, Springer, 2007.

[17] A. Menezes, I. Blake, and et all, *Applications of Finite Fields*, Springer, 1993.

[18] J. Menezes, S. Vanstone, and P. Oorschot, *Handbook of Applied Cryptography*, CRC Press Inc., 1996.

[19] S. Nikova, C. Rechberger, and R. V., Threshold implementations against side-channel attacks and glitches, International Conference on Information and Communications Security, 4307, p. 3–33, December 2014.

[20] S. Nikova, V. Rijmen, and M. Schläffer, Using normal bases for compact hardware implementations of the aes s-box, Springer, 5229, pp. 236–245, 10 2008.

[21] S. Nikova, V. Rijmen, and M. Schläffer, Using normal bases for compact hardware implementations of the aes s-box, Springer, 5229, pp. 236–245, 10 2008.

[22] L. Owen, W. Buchanan, and D. Carson, Power analysis attacks on the aes-128 s-box using differential power analysis (dpa) and correlation power analysis (cpa), Journal of Cyber Security Technology, 1(2), pp. 88–107, 2017.

[23] C. Paar and J. Pelzl, *Understanding Cryptography*, Springer, 2010.

[24] A. Shamir, How to share a secret, ACM, 22, pp. 16–29, 03 1979.

[25] F. Standaert, S. Örs, and B. Preneel, Power analysis of an fpga, Springer, 3156, pp. 30–44, 08 2004.

APPENDIX A

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \end{pmatrix}$$

Table A.1: Converting Polynomial Basis to Normal Basis

$$\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \end{pmatrix}$$

Table A.2: Converting Polynomial Basis to Normal Basis

$$
\begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \end{pmatrix}
=
\begin{pmatrix}
1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1
\end{pmatrix}
\cdot
\begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{pmatrix}
$$

Table A.3: Converting Normal Basis to Polynomial Basis

$$
\begin{pmatrix}
0 & 1 & 1 & 2 & 1 & 2 & 2 & 3 & 1 & 2 & 2 & 3 & 2 & 3 & 3 & 4 \\
1 & 2 & 2 & 3 & 2 & 3 & 3 & 4 & 2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 \\
1 & 2 & 2 & 3 & 2 & 3 & 3 & 4 & 2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 & 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\
1 & 2 & 2 & 3 & 2 & 3 & 3 & 4 & 2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 & 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 & 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\
3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 & 4 & 5 & 5 & 6 & 5 & 6 & 6 & 7 \\
1 & 2 & 2 & 3 & 2 & 3 & 3 & 4 & 2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 & 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 & 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\
3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 & 4 & 5 & 5 & 6 & 5 & 6 & 6 & 7 \\
2 & 3 & 3 & 4 & 3 & 4 & 4 & 5 & 3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 \\
3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 & 4 & 5 & 5 & 6 & 5 & 6 & 6 & 7 \\
3 & 4 & 4 & 5 & 4 & 5 & 5 & 6 & 4 & 5 & 5 & 6 & 5 & 6 & 6 & 7 \\
4 & 5 & 5 & 6 & 5 & 6 & 6 & 7 & 5 & 6 & 6 & 7 & 6 & 7 & 7 & 8
\end{pmatrix}
$$

Table A.4: Hamming Weight Table For All Elements in $GF(2^8)$

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

Table A.5: Substition Box of AES

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

Table A.6: Inverse Substition Box of AES