CHAPTER 3

A POLYNOMIAL BASED MASKED S-BOX TECHNIQUE IN AES FOR SIDECHANNEL ATTACKS

3.1 OVERVIEW

The Advanced Encryption Standard (AES) NIST (2001) is a block cipher with a key of 128 bit, is one of the widely accepted encryption algorithm in the world and is extensively used for both business and government purposes. There are three types AES based on the number of bits in the secret keys. As the length of key increases the safety level of AES also increases.

The AES algorithm can be implemented in software or hardware arrangement. The execution may depend on multiple factors like the reason, the circumstances, the equipment that are used, etc. The algorithm must be used in concurrence with a FIPS approved or NIST suggested mode of procedure FIPS Publication on AES in 2001. This chapter describes the types of SCAs and its prevention by using polynomial based masked S-box technique in AES.
3.2 AES ENCRYPTION

The encryption scheme consists of four stages; the first stage is “Sub Bytes” transformation which is a non-linear byte substitution in which each byte is replaced with its equivalent byte from S-Box look up table. The stage two which shows the “Shift Rows” conversion at regular intervals moves (permutes) the bytes within the block. The stage three “Mix Columns” conversion collects 4-bytes together forming 4-term polynomials and multiplies the polynomials with a predetermined mod (x4+1) polynomial. The fourth step “Add Round Key” modification adds the 128 bit key with the block of information as shown in the above Figure 3.1. The states are the in-between form of the cipher or original information typically displayed as a rectangular table of bytes with 4 rows and 4 columns. The key is lengthened into a quantity of separate round keys using particular key schedule known as the Rijndael key schedule. The majority of AES calculations are done in a particular finite field. AES operates on a 4×4 array of bytes, named as the state.

During encryption, every round of AES (excluding the final round) possess four steps:

![Figure 3.1 Overall block diagram of AES Encryption](image-url)
3.2.1 Add Round Key

Every byte of the state is xor-ed with the round key; In each round 128 bit key is generated from the cipher key using a key schedule as shown in the Figure.3.2

![Add Round Key Transformation](image)

Figure 3.2 Add Round Key Transformation

This is the simple “xor” function. Each byte in state matrix is “xor” ed with equivalent programmed round key matrix.

3.2.2 Shift Rows

A transposition step in which each row of the state is shifted at regular intervals with certain number of steps. In this stage of AES, the first row is left unaffected. Every byte of the row two is shifted by one position to the left. Similarly, in the third and fourth rows are moved by offsets of positions two and three positions correspondingly. For blocks with size of 128 bits and 192 bits, the shifting prototype is the similar. It is suggested in the FIPS Publication on AES in 2001.
The shift rows alteration is shown in Figure 3.4 below.

![Figure 3.3 Shift Rows Transformation](image)

Shifting procedure is performed based on the row number.

### 3.2.3 Sub Bytes

The Sub Byte transformation is done by taking the multiplicative inverse in GF $(2^8)$ followed by an affine transformation.

1. Obtain the multiplicative inverse in the finite field GF$(2^8)$

2. Relate the subsequent affine transformation (over GF$(2^8)$):

$$ b_i'=b_i \oplus b_{(i+4)} \mod 8 \oplus b_{(i+5)} \mod 8 \oplus b_{(i+6)} \mod 8 \oplus b_{(i+7)} \mod 8 \oplus c_i \text{ for } 0 \leq i < 8 ,$$

where $b_i$ indicates the $i$th bit of the byte $b$, and $c_i$ gives the $i$th bit of a byte $c$ with the value $\{63\}$ or $\{01100011\}$. In the above equation the prime on a variable indicates that the variable has to be restructured with the value on the right side as suggested by FIPS Publication on AES in 2001. This is a step of non-linear substitution where each byte is replaced with another according to a lookup Table 3.1.
Table 3.1 S-Box

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63</td>
<td>7c</td>
<td>77</td>
<td>7b</td>
<td>f2</td>
<td>6b</td>
<td>6f</td>
<td>c5</td>
<td>30</td>
<td>01</td>
<td>67</td>
<td>2b</td>
<td>fe</td>
<td>d7</td>
<td>ab</td>
<td>76</td>
</tr>
<tr>
<td>1</td>
<td>ca</td>
<td>82</td>
<td>c9</td>
<td>7d</td>
<td>fa</td>
<td>59</td>
<td>47</td>
<td>f0</td>
<td>ad</td>
<td>d4</td>
<td>a2</td>
<td>af</td>
<td>9c</td>
<td>a4</td>
<td>72</td>
<td>c0</td>
</tr>
<tr>
<td>2</td>
<td>b7</td>
<td>fd</td>
<td>93</td>
<td>26</td>
<td>36</td>
<td>3f</td>
<td>f7</td>
<td>cc</td>
<td>34</td>
<td>a5</td>
<td>e5</td>
<td>f1</td>
<td>71</td>
<td>d8</td>
<td>31</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>04</td>
<td>c7</td>
<td>23</td>
<td>c3</td>
<td>18</td>
<td>96</td>
<td>05</td>
<td>9a</td>
<td>07</td>
<td>12</td>
<td>80</td>
<td>e2</td>
<td>eb</td>
<td>27</td>
<td>b2</td>
<td>75</td>
</tr>
<tr>
<td>4</td>
<td>09</td>
<td>83</td>
<td>2c</td>
<td>1a</td>
<td>1b</td>
<td>6e</td>
<td>5a</td>
<td>a0</td>
<td>52</td>
<td>3b</td>
<td>d6</td>
<td>b3</td>
<td>29</td>
<td>e3</td>
<td>2f</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>53</td>
<td>d1</td>
<td>00</td>
<td>ed</td>
<td>20</td>
<td>fc</td>
<td>b1</td>
<td>5b</td>
<td>6a</td>
<td>cb</td>
<td>be</td>
<td>39</td>
<td>4a</td>
<td>4c</td>
<td>58</td>
<td>cf</td>
</tr>
<tr>
<td>6</td>
<td>d0</td>
<td>ef</td>
<td>aa</td>
<td>fb</td>
<td>43</td>
<td>4d</td>
<td>33</td>
<td>85</td>
<td>45</td>
<td>f9</td>
<td>02</td>
<td>7f</td>
<td>50</td>
<td>3c</td>
<td>9f</td>
<td>a8</td>
</tr>
<tr>
<td>7</td>
<td>51</td>
<td>a3</td>
<td>40</td>
<td>8f</td>
<td>92</td>
<td>9d</td>
<td>38</td>
<td>f5</td>
<td>bc</td>
<td>b6</td>
<td>da</td>
<td>21</td>
<td>10</td>
<td>ff</td>
<td>f3</td>
<td>d2</td>
</tr>
<tr>
<td>8</td>
<td>cd</td>
<td>0c</td>
<td>13</td>
<td>ec</td>
<td>5f</td>
<td>97</td>
<td>44</td>
<td>17</td>
<td>c4</td>
<td>a7</td>
<td>7e</td>
<td>3d</td>
<td>64</td>
<td>5d</td>
<td>19</td>
<td>73</td>
</tr>
<tr>
<td>9</td>
<td>60</td>
<td>81</td>
<td>4f</td>
<td>dc</td>
<td>22</td>
<td>2a</td>
<td>90</td>
<td>88</td>
<td>46</td>
<td>ee</td>
<td>b8</td>
<td>14</td>
<td>de</td>
<td>5e</td>
<td>0b</td>
<td>db</td>
</tr>
<tr>
<td>a</td>
<td>e0</td>
<td>32</td>
<td>3a</td>
<td>0a</td>
<td>49</td>
<td>06</td>
<td>24</td>
<td>5c</td>
<td>c2</td>
<td>d3</td>
<td>ac</td>
<td>62</td>
<td>91</td>
<td>95</td>
<td>e4</td>
<td>79</td>
</tr>
<tr>
<td>b</td>
<td>e7</td>
<td>c8</td>
<td>37</td>
<td>6d</td>
<td>8d</td>
<td>d5</td>
<td>4e</td>
<td>a9</td>
<td>6c</td>
<td>56</td>
<td>f4</td>
<td>ea</td>
<td>65</td>
<td>7a</td>
<td>ae</td>
<td>08</td>
</tr>
<tr>
<td>c</td>
<td>ba</td>
<td>78</td>
<td>25</td>
<td>1e</td>
<td>a6</td>
<td>b4</td>
<td>c6</td>
<td>e8</td>
<td>dd</td>
<td>74</td>
<td>1f</td>
<td>4b</td>
<td>bd</td>
<td>8b</td>
<td>8a</td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>7b</td>
<td>3e</td>
<td>b5</td>
<td>6b</td>
<td>4b</td>
<td>03</td>
<td>f6</td>
<td>0e</td>
<td>61</td>
<td>35</td>
<td>57</td>
<td>b9</td>
<td>86</td>
<td>c1</td>
<td>1d</td>
<td>9e</td>
</tr>
<tr>
<td>e</td>
<td>5e</td>
<td>f3</td>
<td>9b</td>
<td>11</td>
<td>69</td>
<td>d9</td>
<td>8e</td>
<td>94</td>
<td>9b</td>
<td>1e</td>
<td>87</td>
<td>e9</td>
<td>ce</td>
<td>55</td>
<td>28</td>
<td>df</td>
</tr>
<tr>
<td>f</td>
<td>8c</td>
<td>a1</td>
<td>89</td>
<td>0d</td>
<td>bf</td>
<td>e6</td>
<td>42</td>
<td>68</td>
<td>41</td>
<td>99</td>
<td>2d</td>
<td>0f</td>
<td>b0</td>
<td>54</td>
<td>bb</td>
<td>16</td>
</tr>
</tbody>
</table>

The substitute byte transformation is shown in Figure 3.4.

![Figure 3.4 Substitute Byte transformation](image)

3.2.4 Mix Columns

This is a mixing function which operates on the columns of the state that joins the four bytes in each column using a linear conversion. Each column of the state is multiplied with a predetermined polynomial \( c(x) \). Here, the four bytes of each column of the state are united using a reversible linear
transformation. In this step four bytes are assigned as input and four bytes are taken as output, in which every input byte affects all four output bytes.

*Figure 3.5 MixColumn Conversion*

The above Figure 3.5 shows how the Mix Column transformation takes place. In this conversion each column of the state matrix is multiplied by a constant fixed matrix. Each column is treated as a polynomial over $\text{GF}(2^8)$ and then multiplied modulo $x^4+1$ with affixed polynomial $c(x) = 3x^3 + x^2 + x + 2$. The Mix column stage can also be considered as a matrix multiplication in Rijndael’s finite field, where the Mix column block is given below:

\[
\begin{align*}
S'_{0,c} &= \begin{bmatrix} 02 & 01 \\ 01 & 01 \\ 01 & 02 \\ 03 & 01 \\ \end{bmatrix} \\
S'_{1,c} &= \begin{bmatrix} 03 & 01 \\ 02 & 03 \\ 01 & 02 \\ 01 & 03 \\ \end{bmatrix} \\
S'_{2,c} &= \begin{bmatrix} 02 & 01 \\ 03 & 01 \\ 02 & 03 \\ 01 & 02 \\ \end{bmatrix} \\
S'_{3,c} &= \begin{bmatrix} 01 & 01 \\ 03 & 01 \\ 01 & 03 \\ 01 & 01 \\ \end{bmatrix}
\end{align*}
\]

When the above matrix is extended the output of this block can be given by:

\[
\begin{align*}
S'_{0,c} &= ((0 2) \cdot S_{0,c}) \oplus ((0 3) \cdot S_{1,c}) \oplus S_{2,c} \oplus S_{3,c} \\
S'_{1,c} &= S_{0,c} \oplus ((0 2) \cdot S_{1,c}) \oplus ((0 3) \cdot S_{2,c}) \oplus S_{3,c} \\
S'_{2,c} &= S_{0,c} \oplus S_{1,c} \oplus ((0 2) \cdot S_{2,c}) \oplus ((0 3) \cdot S_{3,c}) \\
S'_{3,c} &= ((0 3) \cdot S_{0,c}) \oplus S_{1,c} \oplus S_{2,c} \oplus ((0 2) \cdot S_{3,c})
\end{align*}
\]
3.2.5 Key Expansion

The algorithm for generating the secret round key for all the 10 rounds is explained in the following steps: The column four of the i-1 key is rotated in such way that each element is moved upwards by one row. It then puts this result through Substitute Byte algorithm which replaces each 8 bits of the matrix with a analogous 8-bit value from S-Box.

Here the byte substitution is given by Figure 3.6.

![Figure 3.6 Byte Substitution](image)

In order to generate the column one of the ith key, this result is XOR-ed with the column one of the i-1th key as well as a constant (Row constant or Rcon) which is dependent on i. The table 3.2 shows the row constant (Rcon) is generated. The Figure 3.7 gives the method of generation of Expanded key1

![Table 3.2 RCON](image)
The column two is formed by XOR-ing the 1st column of the ith key with the column two of the i1th key. The Figure 3.8 gives the method of generation of expanded key2.

This continues iteratively for the other two columns in order to produce the entire ith key. In addition the entire process iteratively continues for generating all 10 keys. In the finishing process, all the generated keys are stored statically once they have been calculated initially as the ith key generated is required for the (10-i)th round of decryption. The Key Expansion matrix is shown in Figure 3.9.
3.3 AES DECRYPTION

In the decryption of AES mostly reverse of the encryption algorithm in the opposite direction is followed. The basic steps of AES Decryption are explained in the following block diagram shown as Figure 3.10. There are exactly Nr-1 rounds.

![Figure 3.10 AES Decryption](image)

In the above block diagram, it is found that AES decryption initially does the key-expansion process in which the 128-bit key block that generates all in-between keys Ashwini et al (2009) (which are generated from the original key during Encryption of every round). The generate roundkey module performs the algorithm that generates a round key. Its input is multiplexed between the user provided key and the previous round's key. The output is stored in a register to be used as input during the next iteration of the algorithm. RAM is used for the storage of the expansion keys and it also stores the original key and the keys generated for all ten rounds during the decryption algorithm.
3.3.1 Inverse Add Round Key

In the Inverse Add Round key step XOR operation is performed between the cipher text and intermediate expanded key corresponding to that particular iteration as shown by M.R.M. Rizk in the year 2007. E.g., if the figures on the left indicate the cipher and the key values, the ending value after it has been created by this step is shown on the right which is depicted in Figure 3.11.

\[
\begin{array}{cccc}
1A & A4 & 95 & 3E \\
A9 & B9 & 4C & 3A \\
13 & CD & 78 & 27 \\
86 & F1 & 33 & A8 \\
\end{array}
\begin{array}{cccc}
D0 & C9 & E1 & B6 \\
14 & EE & 3F & 63 \\
F9 & 25 & 0C & 0C \\
A8 & 89 & C8 & A6 \\
\end{array}
\begin{array}{cccc}
CA & 6D & 74 & 88 \\
BD & 57 & 73 & 59 \\
EA & E8 & 74 & 2B \\
2E & 78 & FB & 0E \\
\end{array}
\]

Figure 3.11 Inverse add round key

3.3.2 Inverse Shift Row

This inverse shift row step rotates I elements in every ith row by right wise, as shown in the Figure 3.12. In first row (zeroth row) as usual no shifting.

\[
\begin{array}{cccc}
S_{r,0} & S_{r,1} & S_{r,2} & S_{r,3} \\
S'_{r,0} & S'_{r,1} & S'_{r,2} & S'_{r,3} \\
\end{array}
\]

Figure 3.12 Inverse Shift rows
3.3.3 Inverse Mix Column

In the Inverse MixColumns stage, as performed by the Rijndael cipher, after the inverse shift-rows step, which is the main source of all the 10 rounds of diffusion in Rijndael. Each column in this step is treated as a polynomial over Galois Field ($2^8$) and is then multiplied modulo $x^4 + 1$ with a fixed inverse polynomial is $c^{-1}(x) = 11x^3 + 13x^2 + 9x + 14$. The Multiplication is done as shown in the following matrix.

$$
\begin{bmatrix}
    r_0 \\
    r_1 \\
    r_2 \\
    r_3 \\
\end{bmatrix} = 
\begin{bmatrix}
    14 & 11 & 13 & 9 \\
    9 & 14 & 11 & 13 \\
    13 & 9 & 14 & 11 \\
    11 & 13 & 9 & 14 \\
\end{bmatrix} 
\begin{bmatrix}
    a_0 \\
    a_1 \\
    a_2 \\
    a_3 \\
\end{bmatrix}
$$

The AES decryption originally performs key-expansion step on the 128-bit key block. Then the round key indicates the beginning of the actual decryption process once the information process is completed.

3.3.4 Inverse Sub Bytes

In the inverse sub bytes step, which replaces each entry in the matrix from the corresponding entry in the inverse S-Box as shown in Table 3.3.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>D</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

**Table 3.3 Inverse S-Box**
After completing these steps, the AES decryption repeats nine times the inverse shift row, inverse sub, inverse add round key and inverse mix column steps. In the final iteration, it does an inverse shift row, inverse sub bytes and inverse add round key to reproduce the original data.

3.4 SIDE CHANNEL ATTACKS AND ITS POSSIBILITIES

Today, in everyday life, the use of cell phones, smart cards and other wireless applications became inevitable. Meanwhile, the security of data transfer should also be taken into account. Some hackers technically reveal the secret key with the help of leakage of information like power consumption, output timing, electromagnetic and thermal emission and even sound. (Figure 3.13) Side Channel Attacks are called so because the information processed is based on ‘side channel information’.

![Figure 3.13 Information Leakage from Cryptographic Devices](image)

Side channel information is nothing but the data that can be retrieved from the encryption device that is neither the original text to be encrypted nor the cipher text that resulted from the encryption procedure.
Side Channel Attack is a division of cryptography which deals with the responsive information gained from the target cryptographic mechanism. All Electronic devices disclose information in different ways which are passive.

The features of Side Channel Attacks are:

- Simple ethics
- No need for sophisticated equipment, hence easy to implement
- Consistent
- Low cost, easily affordable
- Risky for secure chip makers

3.5 TYPES OF SIDE CHANNEL ATTACKS

The most common method used by attackers is power analysis attack. The leakage power or the power fluctuation occurring during the cryptographic process is captured by attackers. In general, this is done by inserting a small resistor in series with the power source or the ground and monitoring it. Simple power analysis (SPA) is the most prevalent and helpful only when the algorithm is known. The DPA is the substantial method based on statistical investigation and characteristics of the captured power traces. It is effective even for the unknown algorithm. The execution is done usually by comparing the power traces captured using the known/unknown key. They are correlated within each other to estimate the correct key.

When analysis is made to decide the causes of the attacks, it is found that Usage of registers Availability of pure cipher text Power fluctuation/
correlation provided during dispreading with matched key even when architecture is unknown. Approximately 9000 power traces are required. In general, the side channel attacks are divided into the following types based on the type and behaviour of the device used:

1. Active – the internal circuit of the cryptographic device is reached by the attacker.
   - Probing attack: insertion of sensors into the device
   - Fault induction attack: disturbing the behaviour of the device

2. Passive – The physical and electrical effects of the functionality of the device are used for the attack.

3. Invasive – Straight access to the components within the chip is made possible in invasive attacks. A wire connected to a information bus to see the data transfer is an example.

4. Non - Invasive - A non-invasive attack only exploits superficially available information (the Emission of which is often unintentional) such as operation time, power utilization.

Encryption is a mechanism which is considered as a unit that receives plaintext input and produces cipher text output. Attacks are usually based on either having the knowledge of cipher text or the plaintext. But today, everyone knows that encryption procedures have additional outputs and additional inputs which are neither the plaintext nor the ciphertext. The encryption device produces timing information (data about the timing taken by the operations) that is effortlessly computable.
The classification of side channel attacks is shown in Figure 3.14.

**Figure 3.14 Types of Side Channel Attacks**

- **Timing Attack** — attacks based on measuring how much time various computations take to act upon.

- **Power monitoring attack** — attacks based on the information of the power consumed by the hardware during various operations.

- **Electromagnetic attacks** — attacks which are based on the leaked electromagnetic emission which can directly give plaintexts and other information. Such dimensions are supportive to understand cryptographic keys using techniques corresponding to those in power consumptions, or can be used in non-cryptographic attacks.

- **Acoustic Cryptanalysis** — attacks which make use of sound generated during a computation (like power study).

- **Differential Fault Analysis** — the study in which leaks the secret by introducing faults in a calculation.


3.6 PROPOSED POLYNOMIAL BASED MASKED S-BOX TECHNIQUES

In the Boolean masking execution, the middle value $x$ is hidden by "exclusive-ORing" it with the indiscriminate mask $m$. For each round operation of AES, Shift-Rows, Mix-Columns, and Add-RoundKey are linear conversions, whereas SubBytes is the only nonlinear conversion of the AES. We demonstrate the linear transformations as $\text{Oper}$; then, the masked $\text{Oper}$ can be illustrated as $\text{Oper}(x \oplus m) = \text{Oper}(x) \oplus \text{Oper}(m)$, though, the masked nonlinear transformation Sub-Bytes has the significance as $S\text{-box}(x - m)' = S\text{-box}(x) \oplus S\text{-box}(m)$. In order to hide or mask the nonlinear transformation, a new S-box, denoted as $S\text{-box}'$, is recalculated as $S\text{-box}'(x \oplus m) = S\text{-box}(x) m'$, where $m$ and $m'$ are the input mask and output mask of Sub bytes. In order to mask a 128-bit AES, it typically needs 6-byte inconsistent values. These 6 values are defined as $m$, $m'$, $m_1$, $m_2$, $m_3$, and $m_4$. For simple calculation, we define $m_{1234} = \{m_1,m_2,m_3,m_4\}$ as the mask for one 32-bit Mix-Columns transformation and it holds that $m'_{1234} = \text{Mix-Columns}(m_{1234})$. The field $GF(2^8)$ is an extension of the field $GF(2^4)$, over which to perform a modular reduction requires an irreducible polynomial of degree 2, $x^2 + \{1\}x + \{e\}$ and an additional irreducible polynomial of degree 4, $x^4 + x + 1$. In order to lower the hardware resources, we approximate the masked AES algorithm mainly over $GF(2^4)$.

The resource reduction is further achieved by mapping with $GF((22)^2)2$. The adjustment of the masked Sub-bytes and masked Mix Columns is discussed in the subsequent and the masked Shift-Rows and masked Add-Round key remains unchanged.

In order to shift the mapping and inverse mapping exterior to AES's round operation, we interchange the computational cycle of masked affine and inverse mapping functions within masked S-box. The masked affine
function needs to be accustomed with new scaling factors as shown in figure 3.15.

Figure 3.15 Proposed Masked S-Box AES

\[(z+m+m)' = \text{map} \ (z+m+m)\]

where \( (z + m)' = \{a^* h + m_h, a^* l + m_l \} \) and \( m' = \{m_h, m_l \} \).

Affine transformation mapping is done using the equation given below.

\[A(z+m+m)+b = \text{Amap}^{-1}(z+m+m)'+b\]

Mapping is done in the Galois field GF \( (2^8) \) to GF\((2^4)^2\) to GF\((2^2)^3)^2\) so that information is masked in the S-box and prevented from attacks.
3.7 SIMULATION RESULTS

ModelSim SE is our UNIX, Linux, and Windows-based simulation tool and debugging situation, combining high performance with the most powerful and instinctive GUI in the business.

Using ModelSim software the simulation of Add Round Key has been done and the simulation result is shown below in figure 3.16.

![Figure 3.16 Add Round Key](image)

The simulation result of shift rows in the Figure 3.17.

![Figure 3.17 Shift Rows](image)
Using model simulator the simulation of sub bytes is done and the simulation results are shown in Figure 3.18.

![Figure 3.18 Sub bytes](image1.png)

The simulation results of mixed columns are given by the Figure 3.19.

![Figure 3.19 Mixed Columns](image2.png)
Figure 3.20 Synthesized result of polynomial based masked S-box Technique

The synthesized results of polynomial based masked S-box and general masked S-box is shown in the figures 3.20 and 3.21 respectively.
3.8 SUMMARY

Advanced Encryption Standard is found to be the popularly used cryptographic algorithm nowadays. Hence, it becomes necessary to analyze about the security of this algorithm during the transformation processes. Side-channel analysis is observing the behaviours of the electronic circuit. The analysis is often applied to encryption devices in order to investigate the leakage of secret data, such as keys and PIN codes. We have analyzed this truth based on simulation results of all proposed polynomial based masked S-Box technique implementations.