Cryptanalysis

- CaS -

Ji Yong-Hyeon

Department of Information Security, Cryptology, and Mathematics
College of Science and Technology
Kookmin University

January 19, 2024

List of Symbols

Contents

Chapter 1

The Linear Approximation Table

We will define the **Linear Approximation Table** $\mathcal{L}$ of any S-Box. We abbreviate “linear approximation table” by **LAT**.

Suppose we are given a S-Box S takes n-bit sequences to m-bit sequences. The LAT $\mathcal{S}$ of S is a table (or matrix) with 2^n-rows and 2^m-columns. The entries of the LAT $\mathcal{L}$ consist of integers. This includes both positive and negative integers,m as well as zero.

1.1 Motivation

We calculate the LAT’s of the S-Box’s used in given block cipher as the first step in designing an attack on that block cipher by linear cryptanalysis.

How does the LAT of a S-Box reveal potential vulnerabilities of a block cipher that uses it in its design?

If the LAT of a S-box contain some “large” integer value (either positive or negative) then a block cipher that uses that S-Box may be vulnerable to an attack by linear cryptanalysis.

1.2 Preliminaries

1.2.1 The Filed of Two Elements

Define $\mathbb{F}_2 = \{0, 1\}$ to be the filed of two elements. We interpret $\mathbb{F}_2$ as the set of bits (zero and one). We have two binary operation on $\mathbb{F}_2$ namely **addition** and **multiplication**, so that $\mathbb{F}_2$ becomes a **field** under these operations.

- $\oplus : \mathbb{F}_2 \times \mathbb{F}_2 \longrightarrow \mathbb{F}_2$.
- The **addition operation** on F_2 is the logical operator XOR.
- It is denoted by $\oplus$.

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

- $\& : \mathbb{F}_2 \times \mathbb{F}_2 \longrightarrow \mathbb{F}_2$.
- The **multiplication operation** on F_2 is the logical operator AND.
- It is denoted by &.

x	y	$x\&y\ (xy)$
0	0	0
0	1	0
1	0	0
1	1	1

1.2.2 Sequences of Bits

By a sequence of bits, we mean a sequence of 0's and 1's. Define $\mathbb{F}_2^n$ to be the st of bit sequences of length n, where n is a positive integer.

$$\mathbb{F}_2^n = \left\{(x_1, \ldots, x_n) : x_i \in \mathbb{F}_2\right\}.$$

- $\mathbb{F}_2^n$ has 2^n elements.
- $\mathbb{F}_2^n$ is in bijection with the set of integers $\{0, 1, \ldots, 2^n - 1\}$.

1.2.3 The Dot Product

We can define the **dot product** operation, which is a map from $\mathbb{F}_2^n \times \mathbb{F}_2^n$ to $\mathbb{F}_2$. That is,

$$\bullet : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2.$$

We define dot product $x \cdot y \in \mathbb{F}_2$ by

$$x \cdot y = \bigoplus_{i=1}^{n} x_i y_i,$$

where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in $\mathbb{F}_2^n$.

An expression that we will encounter in defining the linear approximation table is:

$$(a \cdot x) \oplus (\beta \cdot y).$$

Here $\alpha, x \in \mathbb{F}_2^n$ and $\beta, y \in \mathbb{F}_2^m$. Then

$$\begin{cases} \alpha \cdot x \in \mathbb{F}_2 \\ \beta \cdot y \in \mathbb{F}_2 \end{cases} \implies (a \cdot x) \oplus (\beta \cdot y) \in \mathbb{F}_2.$$

1.3 Definition of a S-Box

S-BOX

Definition 1.1. Let $n, m \in \mathbb{Z}^+$ be positive integers. Let

$$S : \mathbb{F}_2^n \to \mathbb{F}_2^m$$

be any function. We call S a **S-BOX**. Note that n and m represent the number of **input bits outputs bits**, respectively.

1.4 Linear Equations Associated to a S-Box

Let $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a given S-Box with n input bits and m output bits. Suppose $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_2^n$ and $\beta = (\beta_1, \ldots, \beta_m) \in \mathbb{F}_2^m$ are given. We are interested in solutions $x \in \mathbb{F}_2^n$ of the equation:

$$(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0.$$

We call α and β the **masks** of the equation. α is the **input mask** and β is the **output mask**.

We can write

$$S(x) = S(x_1, \ldots, x_n) = y = (y_1, \ldots, y_m)$$

where $x_1, \ldots, x_n \in \mathbb{F}_2$ are the **input variables** and $y_1, \ldots, y_m \in \mathbb{F}_2$ are the **output variables**. Note that y_i are not free variables but rather are determined by the choice of the free variables x_i.

Then the equation $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$ can be written as:

$$\left(\bigoplus_{i=1}^{n} \alpha_i x_i \right) \oplus \left(\bigoplus_{j=1}^{m} \beta_j y_j \right) = (\alpha_1 x_1 \oplus \cdots \oplus \alpha_n x_n) \oplus \left(\beta_1 y_1 \oplus \cdots \oplus \beta_m y_m \right) = 0.$$

This is a **linear equation** of the input variables $x_1, \ldots, x_n$ and the output variables $y_1, \ldots, y_m$.

Let $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a S-Box. For $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$, we define the set $\Sigma_{\alpha,\beta}$ by:

$$\Sigma_{\alpha,\beta} = \left\{ x \in \mathbb{F}_2^n : (\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0 \right\}.$$

So, $\Sigma_{\alpha,\beta}$ is the set of all values $x \in \mathbb{F}_2^n$ that satisfy the linear equation: $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$.

We also define the non-negative integer $e_{\alpha,\beta}$ to be the cardinality of the set $\Sigma_{\alpha,\beta}$, namely:

$$e_{\alpha,\beta} = \left| \Sigma_{\alpha,\beta} \right|.$$

Since $\Sigma_{\alpha,\beta}$ is a subset of $\mathbb{F}_2^n$, and the cardinality of $\mathbb{F}_2^n$ is 2^n, we get rough bound on $e_{\alpha,\beta}$

$$0 \leq e_{\alpha,\beta} \leq 2^n.$$

1.5 Bias and Probability

In general if p is a **probability** of an event then p is a real number between 0 and 1, that is $0 \leq p \leq 1$. From a given probability we can define an associated quantity called the **bias**, usually denoted ϵ.

Bias

Definition 1.2. The bias ϵ is defined in terms of the probability p by the formula

$$\epsilon = p - \frac{1}{2}.$$

Therefore, the bias is a real number in the range $-\frac{1}{2} \leq \epsilon \leq \frac{1}{2}$.

So, a bias can be negative whereas a probability can not. Conversely, if we are given the bias ϵ then the associated probability p is given by

$$p = \epsilon + \frac{1}{2}.$$

Just as $p = 0$ and $p = 1$ are the edge cases of a probability value, the values $\epsilon = -\frac{1}{2}$ and $\epsilon = \frac{1}{2}$ are the edge cases of a bias value.

Note that a probability of $p = \frac{1}{2}$ has a bias of $\epsilon = 0$. So, for example on the toss of a fair coin, both heads and tails have a bias 0.

1.6 Linear Equations and Bias

Given $\alpha \in \mathbb{F}_2^n$ and $\beta \in \mathbb{F}_2^m$ we have defined number of $x \in \mathbb{F}_2^n$ which satisfy the equation $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$ to be $e_{\alpha,\beta} = \left|\Sigma_{\alpha,\beta}\right|$. Therefore, the probability of $x \in \mathbb{F}_2^n$ satisfying the equation $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$ is:

$$p_{\alpha,\beta} = \frac{\left|\Sigma_{\alpha,\beta}\right|}{\left|\mathbb{F}_2^n\right|} = \frac{e_{\alpha,\beta}}{2^n}.$$

As with all probabilities we have $0 \leq p_{\alpha,\beta} \leq 1$.

So, the **bias** of the equation $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$ holding is:

$$\epsilon_{\alpha,\beta} = p_{\alpha,\beta} - \frac{1}{2}.$$

Since $p_{\alpha,\beta} = \frac{e_{\alpha,\beta}}{2^n}$ we can write the bias as

$$\epsilon_{\alpha,\beta} = \frac{e_{\alpha,\beta}}{2^n} - \frac{1}{2} = \frac{e_{\alpha,\beta} - 2^{n-1}}{2^n}.$$

We call the numerator of this expression the **bias integer** associated to α and β and denote it as:

$$e'_{\alpha,\beta} = e_{\alpha,\beta} - 2^{n-1}.$$

Recall that $0 \leq e_{\alpha,\beta} \leq 2^n$. Subtracting 2^{n-1} from this inequality gives:

$$\begin{aligned} 0 &\leq e_{\alpha,\beta} \leq 2^n \\ 0 - 2^{n-1} &\leq e_{\alpha,\beta} - 2^{n-1} \leq 2^n - 2^{n-1} \\ -2^{n-1} &\leq e_{\alpha,\beta} - 2^{n-1} \leq 2^n(1 - 2^{-1}) \\ -2^{n-1} &\leq e_{\alpha,\beta} - 2^{n-1} \leq 2^{n-1}. \end{aligned}$$

Which gives us rough bounds for the possible values of the **bias integer** $e'_{\alpha,\beta}$

$$-2^{n-1} \leq e'_{\alpha,\beta} \leq 2^{n-1}.$$

By definition, the bias is obtained from the bias integer by dividing it by 2^n:

$$\epsilon_{\alpha,\beta} = \frac{e'(\alpha,\beta)}{2^n}.$$

1.7 The LAT of a S-Box

The Linear Approximation Table (LAT)

Definition 1.3. Let $S : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$ be a S-Box. The **linear approximation table** (abbreviated **LAT**) of S is a table of integers with 2^n-rows and 2^m-columns. We denote it by $\mathcal{L}$.

- The rows are indexed by the elements $\alpha \in \mathbb{F}_2^n = \{0, \dots, 2^n - 1\}$.
- The columns are indexed by the elements $\beta \in \mathbb{F}_2^m = \{0, \dots, 2^m - 1\}$.
- The entry at row index α and column index β is given by the bias integer $e'_{\alpha,\beta}$.

$$\mathcal{L} = (e'_{\alpha,\beta}).$$

So, the LAT of a S-Box is just a table of all possible bias integer $e'_{\alpha,\beta}$.

Note. The **linear approximation table** $\mathcal{L}$ of a S-Box $S : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^m$:

$\mathcal{L}$	0	1	$\cdots$	β	$\cdots$	$2^m - 1$
0	2^{n-1}					
1	0					
$\vdots$						
α	0			$e'_{\alpha,\beta}$		
$\vdots$						
$2^n - 1$	0					

We will see that for any LAT the first column is all zeros except the first entry which is 2^{n-1}.

1.8 Properties of the LAT

We already proved that the entries $e'_{\alpha,\beta}$ of the LAT are integers between -2^{n-1} and 2^{n-1}. Consider the case $\alpha = 0$ and $\beta = 0$. Then every $x \in \mathbb{F}_2^n$ is a solution to $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$. Therefore, $\Sigma_{0,0} = \mathbb{F}_2^n$ and so $e_{0,0} = \left|\mathbb{F}_2^n\right| = 2^n$. The corresponding bias integer is

$$e'_{0,0} = e_{0,0} - 2^{n-1} = 2^n - 2^{n-1} = 2^{n-1}.$$

So, for any LAT, the value in the upper-left corner is always 2^{n-1}.

Lemma 1.1. *Let $\alpha \in \mathbb{F}_2^n$ with $\alpha \neq 0$ and $n > 1$. Define the following subset of $\mathbb{F}_2^n$:*

$$W = \left\{x \in \mathbb{F}_2^n : \alpha \cdot x = 0\right\}.$$

Then the cardinality of W is 2^{n-1}.

Proof. Since $\alpha = (\alpha_1, \ldots, \alpha_n) \neq 0$,

$$\exists i_0 \in \{1, \ldots, n\} : \alpha_{i_0} \neq 0.$$

Assume that $i_0 = 1$. We can write

$$\alpha = (1, \alpha') \quad \text{where} \quad \alpha' \in \mathbb{F}_2^{n-1}.$$

Define a map

$$\begin{array}{rccc} \phi & : & \mathbb{F}_2^{n-1} & \longrightarrow & W \\ & & x & \longmapsto & (\alpha' \cdot x, x) \end{array}.$$

Note that ϕ is well defined since

$$\alpha \cdot \phi(x) = \alpha \cdot (\alpha' \cdot x, x) = (\alpha' \cdot x) \oplus (\alpha' \cdot x) = 0.$$

Since ϕ is bijection, the cardinality of W equals that of $\mathbb{F}_2^{n-1}$, which is 2^{n-1}. □

Let us consider the values of the first column of the LAT (except for the first entry). This corresponds to the case $\alpha \neq 0$ and $\beta = 0$. The equation $(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0$ becomes $\alpha \cdot x = 0$. By the previous lemma the cardinality of the set of $x \in \mathbb{F}_2^n$ satisfying $\alpha \cdot x = 0$ is 2^{n-1}. Hence $e_{\alpha,0} = 2^{n-1}$. And so,

$$e'_{a,0} - 2^{n-1} = 2^{n-1} - 2^{n-1} = 0.$$

So, the first column of any LAT has its first entry as 2^{n-1} and then all zeros below that. This case where $\beta = 0$ is not of interest in applications since it does not involve the S-Box S.

Suppose S is a bijection (one-to-one and onto). Then necessarily $n = m$. So.

$$S : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n \quad \text{and} S^{-1} : \mathbb{F}_2^n \rightarrow \mathbb{F}_2^n.$$

Suppose $x \in \mathbb{F}_2^n$ and let $y = S(x) \in \mathbb{F}_2^n$. Then $x = S^{-1}(y)$. Then

$$(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0 \equiv (\beta \cdot y) \oplus (\alpha \cdot S^{-1}(y)) = 0.$$

This shows that for any α and β, we have $e^S_{\alpha,\beta} = e^{S^{-1}}_{\beta,\alpha}$. Hence:

$$\mathcal{L}_{S^{-1}} = (\mathcal{L}_S)^T.$$

This says the LAT of S^{-1} is the transpose of the LAT of S. Therefore, when S is a bijection the first row and the first column are all zeros, except for the upper-left entry which is 2^{n-1}.

Proposition 1.2. *If $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ is a bijective S-Box and $n > 1$ then the entries of the LAT are all* ***even integers****.*

Proof. $\square$

1.9 Complexity

Note that there are 2^n possible choices of α and 2^m possible choices of β. Therefore, there are $2^n \cdot 2^m = 2^{n+m}$ possible equations:

$$(\alpha \cdot x) \oplus (\beta \cdot S(x)) = 0.$$

There are 2^n possible inputs $x \in \mathbb{F}_2^n$ to each equation. So, to calculate the LAT of a S-Box will require $2^{n+m} \cdot 2^n = 2^{2n+m}$ calculations.

For example if $n = m = 8$ this gives $2^{3n} = 2^{3 \cdot 8} = 2^{24} = 16,777,216$ possibilities to look at (which is no problem computationally).

1.10 Coding the LAT

1.10.1 Dot Product

```c
int dot(int u, int v) {
    int w = u & v;
    int dot = 0;

    // Using Kernighan's algorithm to count the number of set bits
    while (w) {
        dot ^= 1;
        w &= w - 1;  // clear the least significant bit set
    }

    return dot;
}
int main() {

    int u = 0b101010;  // Example binary numbers
    int v = 0b110100;

    // Dot product: 1
    printf("Dot product: %d\n", dot(u, v));

    return 0;
}
```

```c
// S is a list of integers that gives the values of the S-Box
// n = number of input bits of the S-Box S
// alpha is the input mask
// beta is the output mask
int bias_integer(int S[], int alpha, int beta, int n) {
    int e = 0;
    int range = 1 << n;  // 2 ** n

    for (int x = 0; x < range; x++) {
        if (dot(alpha, x) ^ dot(beta, S[x]) == 0) {
            e++;
        }
    }

    return e - (range >> 1);  // range / 2 or 2 ** (n - 1)
}

int main() {
    // Example S-box (replace with actual values)
    int S[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int n = 3;  // Example number of input bits
    int alpha = 0b101;  // Example input mask
    int beta = 0b110;  // Example output mask

    // Bias integer: 2
    printf("Bias integer: %d\n", bias_integer(S, alpha, beta, n));
    return 0;
}
```

```c
// LAT = linear approximation table
// S = S-Box
// n = number of input bits
// m = number of output bits
int** lat(int S[], int n, int m) {
    int n_range = 1 << n;
    int m_range = 1 << m;

    // Dynamically allocate 2D array
    int** L = (int**)malloc(n_range * sizeof(int*));
    for (int i = 0; i < n_range; i++) {
        L[i] = (int*)malloc(m_range * sizeof(int));
    }

    // Compute the LAT
    for (int alpha = 0; alpha < n_range; alpha++) {
        for (int beta = 0; beta < m_range; beta++) {
            L[alpha][beta] = bias_integer(S, alpha, beta, n);
        }
    }

    return L;
}

int main() {
    // Example S-box (replace with actual values)
    int S[] = {1, 2, 3, 4, 5, 6, 7, 8};
    int n = 3;  // Example number of input bits
    int m = 3;  // Example number of output bits

    int** L = lat(S, n, m);

    // Print LAT matrix
    for (int i = 0; i < (1 << n); i++) {
        for (int j = 0; j < (1 << m); j++) {
            printf("%d ", L[i][j]);
        }
        printf("\n");
        free(L[i]);  // Free memory for each row
    }
    free(L);  // Free the top-level pointer

    return 0;
}
```

```c
void print_lat(int S[], int n, int m) {
    int n_range = 1 << n;
    int m_range = 1 << m;
    int** L = lat(S, n, m);

    for (int alpha = 0; alpha < n_range; alpha++) {
        for (int beta = 0; beta < m_range; beta++) {
            printf("%2d ", L[alpha][beta]);  // %2d ensures numbers are right-aligned in a field of width 2
        }
        printf("\n");

        // Free memory for each row
        free(L[alpha]);
    }

    // Free the top-level pointer
    free(L);
}

int main() {
    int S[] = {1, 2, 3, 4, 5, 6, 7, 8};  // Example S-box (replace with actual values)
    int n = 3;  // Example number of input bits
    int m = 3;  // Example number of output bits

    print_lat(S, n, m);

    return 0;
}
```

1.11 S-Box Analysis

Note (Parity Bits). Parity bits are a simple, yet powerful, method of error detection in digital communications and data storage. A parity bit is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) is even or odd. There are two types of parity bits:

- **Even Parity**: The parity bit is set so that the total number of 1-bits in the code is even.
- **Odd Parity**: The parity bit is set so that the total number of 1-bits in the code is odd.

The calculation of a parity bit depends on the desired parity (even or odd). For a given set of bits, the process is as follows:

1. Count the number of bits set to 1 in the data.
2. For even parity, if the count is odd, set the parity bit to 1. If the count is even, set the parity bit to 0.
3. For odd parity, if the count is even, set the parity bit to 1. If the count is odd, set the parity bit to 0.

Parity bits are widely used in various forms of data transmission and storage to detect errors. They are particularly useful in detecting single-bit errors.

While parity bits can detect single-bit errors, they are not capable of detecting all types of errors, such as two-bit errors or the exact location of the error. Parity bits are a simple, yet powerful, method of error detection in digital communications and data storage. A parity bit is a bit that is added to a group of source bits to ensure that the number of set bits (i.e., bits with value 1) is even or odd.

```c
int parity(int n) {
    int count = 0;
    while (n) {
        count ^= n & 1;
        n >>= 1;
    }
    return count;
}
```

Example 1.1. Consider

```
u8 s_box[8] = {0x7, 0x0, 0x6, 0x4, 0x5, 0x2, 0x1, 0x3};
```

That is,

$$\begin{aligned}
\texttt{000} &\mapsto \texttt{111}\\
\texttt{001} &\mapsto \texttt{000}\\
\texttt{010} &\mapsto \texttt{110}\\
\texttt{011} &\mapsto \texttt{100}\\
\texttt{100} &\mapsto \texttt{101}\\
\texttt{101} &\mapsto \texttt{010}\\
\texttt{110} &\mapsto \texttt{001}\\
\texttt{111} &\mapsto \texttt{011}.
\end{aligned}$$

Let $S(1,1,0) = (0,0,1)$ where

$$\begin{aligned}
x_2 &= 1, & x_1 &= 1, & x_0 &= 0,\\
y_2 &= 0, & y_1 &= 0, & y_0 &= 1.
\end{aligned}$$

$$x_2 + x_1 + x_0 = y_2 + y_1 + y_0$$

Appendix A

Additional Data A

A.1 Substitution-BOX

Bibliography

[1] “The Linear Approximation Table (LAT) of a S-Box” YouTube, uploaded by JacksonInfoSec, 22 October 2022, https://www.youtube.com/watch?v=hHG_Ife-of0

[2] “Cryptanalysis - L8 Linear Cryptanalysis” YouTube, uploaded by Maria Eichlseder, 6 May 2021, https://www.youtube.com/watch?v=RE6xu5THyJA