CS406 Project
Decrypting SPN Cipher using Linear Cryptanalysis

Midathana Pranay - 210050096
Magham Dipen Anjan - 210050092

May 2024

GitHub Repository: `https://github.com/pranay5677/CS406-Linear_Cryptanalysis`

S-Box & P-Box

In our cipher, we divide the 16-bit data block into four 4-bit sub-blocks. Each sub-block forms an input to a 4×4 S-box (a substitution with 4 input and 4 output bits), which can be easily implemented with a table lookup of sixteen 4-bit values, indexed by the integer represented by the 4 input bits. The most fundamental property of an S-box is that it is a nonlinear mapping, i.e., the output bits cannot be represented as a linear operation on the input bits.

The permutation portion of a round is simply the tranposition of the output bits from the S-Box of the current round to give input bits of the next round. Each sub-blocks produced by the output of S-Box are glued to be taken as input of P-Box which produces a 16-bit output.It is to be noted that this output is not the input to next S-Box , we XOR this with next round's Key to get the input to next S-Box.

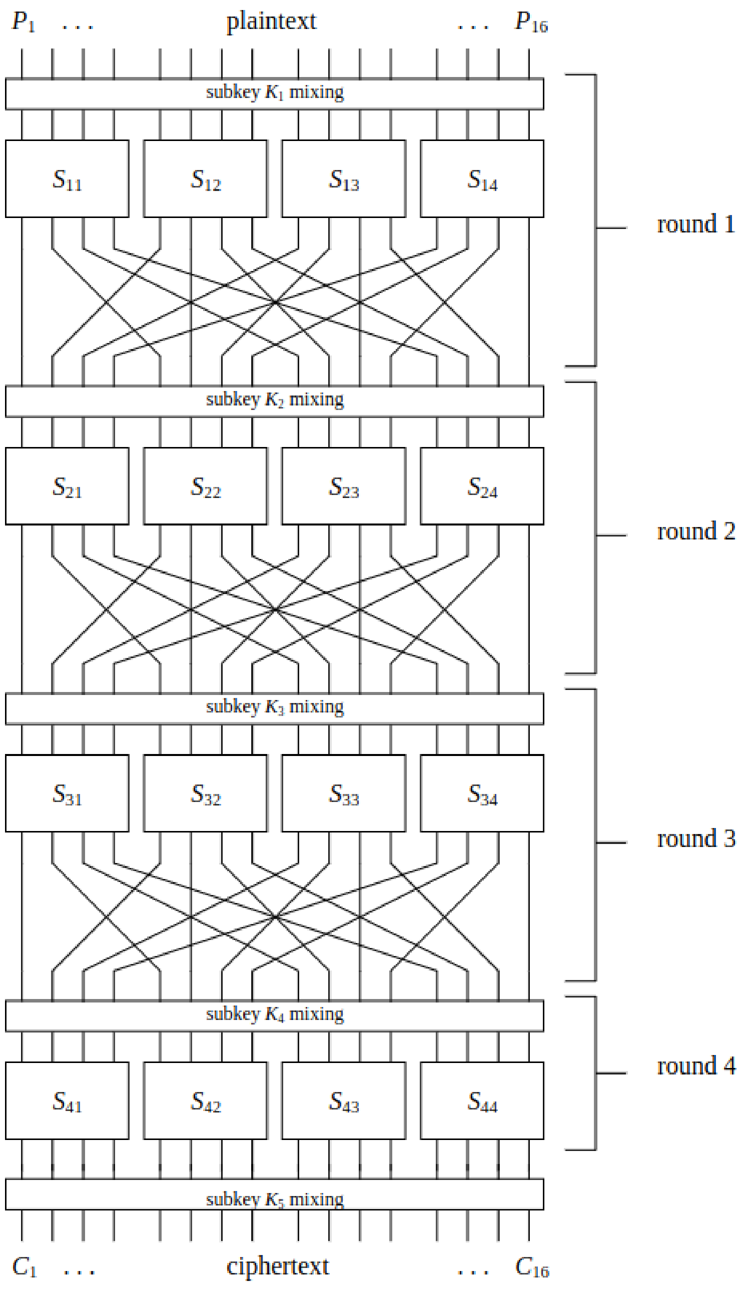
Encryption & Decryption

Encryption in an SPN cipher involves several rounds as follows:

1. **Key Expansion**: The encryption key is expanded into round keys.
2. **Initial Permutation**: Rearrange the plaintext bits.
3. **Round Function**: Substitute and permute the plaintext.
4. **Key Mixing**: Combine the round output with the round key.
5. **Final Permutation**: Rearrange the output bits.
6. **Output**: The final output is the ciphertext.

In an SPN cipher, decryption is the reverse of encryption:

1. **Key Expansion (Reverse)**: Expand the encryption key into round keys in reverse order.
2. **Initial Permutation (IP)**: Rearrange the ciphertext bits.
3. **Round Function (Reverse)**: Perform the inverse of substitution and permutation operations.
4. **Key Mixing (Reverse)**: Combine the round output with round keys in reverse order.
5. **Final Permutation (FP)**: Rearrange the output bits.
6. **Output**: The final output is the plaintext.



Linear Cryptanalysis of SPN Cipher

1. **Construct Linear Approximation for individual S-Boxes**:
   - Each entry in the Linear Approximation table represents the number of matches between the linear equation represented in hexadecimal as ”Input Sum” and the sum of the output bits represented in hexadecimal as ”Output Sum” minus 8 for that S-Box
   - Find the output which has more bias for given input of the S-Box to get an linear equation relating input and output bits with computed bias
2. **Query Data**:
   - Collect a sufficient number of plaintext-ciphertext pairs for cryptanalysis of the given cipher.
   - Store initial inputs for every S-box to initiate the attack on the cipher.

3. **Compute Paths & Linear Equations**:
   - Iterate through the following procedure until reaching the last round:
     - For a given input of a round, select the output with the maximum bias for that input. The input to the next round is the output after passing the maximum bias output through a P-box. Also, record the bias obtained in this round.
     - In the last round, calculate the final linear equation for the given input using the piling lemma for the biases stored and the output of the last round.
4. **Attack to Obtain Last Sub Key**:
   - Choose the linear equation with the maximum bias in the above step
   - Utilize this linear equation for a brute force attack on key bits for all set bits in the output of the last rounds as follows:
     - For each key, iterate through all possible plaintext-ciphertext pairs. Invert the XOR of the ciphertext and the key, and call this inverted_output. Calculate the bias with which this key satisfies the previously obtained linear equation.
     - From all biases obtained, select the sub key whose bias is closest to the computed bias of the linear equation
   - Iterate through the linear equations in decreasing order of the bias until we obtain the complete subkey
5. **Extending it to obtain complete key**:
   - Repeat the above procedure by replacing the ciphertext with s_inv[XOR(last_subkey,ciphertext)]
   - Make sure to use P-Box on the input obtained to the last round.

Differential Cryptanalysis

We will now discuss a similar and yet clever,powerful technique compared to Linear Cryptanalysis. Differential Cryptanalysis exploits the likelihood of specific plaintext differences producing particular ciphertext differences, particularly focusing on the final round of the cipher.
Here's how it typically works:

1. **Input and Output Differences**: Let's say you have an encryption system where you input a plaintext $X = [X_1, X_2, \ldots, X_n]$ and get an output ciphertext $Y = [Y_1, Y_2, \ldots, Y_n]$. The input difference is denoted as $\Delta X = X_v \oplus X_w$, where $\oplus$ represents a bitwise XOR operation and X_v and X_w are two different inputs. Similarly, the output difference is denoted as $\Delta Y = Y_v \oplus Y_w$, where Y_v and Y_w are the corresponding outputs.
2. **Probability in Randomized Ciphers**: In an ideal randomized cipher, if you have a specific input difference ΔX, the probability of obtaining a particular output difference ΔY is $\frac{1}{2^n}$, where n is the number of bits in X.
3. **Exploiting Differential**: Differential cryptanalysis comes into play when a specific output difference ΔY occurs with a very high probability p_D, which is significantly greater than $\frac{1}{2^n}$. This high probability of a specific output difference given a specific input difference is what attackers leverage to break the encryption.
4. **Identifying Differentials**: A pair $(\Delta X, \Delta Y)$ with a high probability p_D is called a "differential." Differential cryptanalysis aims to identify such differentials and use them to deduce information about the encryption key or even to break the encryption entirely.

By analyzing these differentials, attackers can gain insights into the internal workings of the cipher and potentially recover the plaintext or the encryption key. It's a powerful technique used in cryptanalysis, especially against block ciphers like DES and AES.

Advantages using Differential Cryptanalysis

The ability to attack ciphers that use non-linear S-boxes is one of the key advantages of the differential method. It's also relatively efficient and requires fewer plaintext-ciphertext pairs than other cryptanalytic techniques. Additionally, differential cryptanalysis has shown to be successful in breaching a variety of block ciphers and successfully attacking a wide range of cryptographic systems,cryptographic hash functions.

Limitations using Differential Cryptanalysis

Despite its many advantages, the differential technique has several limitations. For instance, this method is less effective against cyphers with a small block size or a limited number of rounds.

As with linear cryptanalysis, caution must be exercised in "proving" immunity to differential cryptanalysis. The computation of the differential characteristic probability is premised on the independence of the S-boxes involved in the approximation and in a real cipher, there is a dependence between the data entering different S-boxes. Hence, the probability p_D is an estimate only. In practice, in many ciphers it has proven to be reasonably accurate.

Approaches to providing resistance to differential cryptanalysis have focused on the S- box properties (i.e., minimizing the difference pair probability of an S-box) and finding structures to maximize the number of active S-boxes. Rijndael is a good example of a cipher designed to provide high resistance to differential cryptanalysis.

Finally, it's susceptible to noise and randomisation, significantly affecting the attack's success rate.

References

1. Howard M. Heys. *A Tutorial on Linear and Differential Cryptanalysis*
   `https://ioactive.com/wp-content/uploads/2015/07/ldc_tutorial.pdf`
2. Christopher Kruegel. *CS177: Project 6 - Linear Cryptanalysis*.
   `https://sites.cs.ucsb.edu/~chris/teaching/cs177/projects/proj6.html`
3. D. Mukhopadhyay *Linear CryptAnalysis*. nptelhrd
   `https://www.youtube.com/watch?v=93PD9CqcZTg`