Correlation Power Analysis (CPA) on Romulus-N Encryption

Deepak S (MM22B011) Monish M (CS23M003)

November 15, 2024

1 Introduction

This report presents a Correlation Power Analysis (CPA) attack on the Romulus-N encryption algorithm, based on the SKINNY block cipher. The CPA attack utilizes a set of 500 plaintexts along with their corresponding power traces to recover the encryption key. Our focus is primarily on the first round of encryption to extract key bytes K_0 to K_7, and we extend our analysis to the second round to derive the remaining key bytes K_8 to K_{15}.

1.1 Encryption Overview

Romulus-N is a lightweight authenticated encryption scheme designed in accordance with the NIST LWC standard and employs the SKINNY block cipher. The encryption process takes the following inputs:

- Message M: 16-byte blocks
- Nonce N: 16-byte blocks
- Associated Data (AD): Fixed as 00
- Key K: 16-byte block

The encryption process consists of 80 rounds of SKINNY encryption, divided into two phases:

1. The first 40 rounds operate with $AD = 00$, Nonce N, and Key K.
2. The subsequent 40 rounds use the message M, Nonce N, and Key K.

The SKINNY implementation of Romulus utilizes a Tweak Key of 48 bytes generated from the actual key and nonce through specific operations. Each round includes the following operations:

- **SubCells** – Each byte of data is substituted based on a lookup table.
- **AddConstants** – Each byte in the state is XORed with a fixed constant matrix.
- **AddRoundTweakey (ART)** – The Tweak Key consists of:

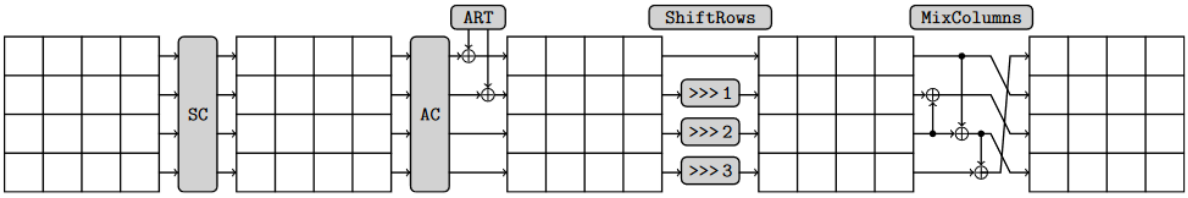

Figure 1: One round encryption of SKINNY Implementation in Romulus

  - The first 16 bytes as **TK1**.
  - The next 16 bytes as the **Nonce**, referred to as **TK2**.
  - The final 16 bytes as the **Derived Key**, denoted as **TK3**.

  The Derived Key for the first round is the same as the actual key, while for subsequent rounds, it is derived from a permuted version of the previous round's TK3 combined with an LFSR (Linear Feedback Shift Register).

- **ShiftRows** – The state matrix is shifted to the right.
- **MixColumns** – The results from ShiftRows are further diffused using a different matrix than AES.

The first two rows of the Tweak Key are used for the encryption in the current round, with the subsequent bytes applied in the next round. Therefore, key bytes K_0 to K_7 are relevant for Round 1, while K_8 to K_{15} will be used in Round 2.

2 Methodology

The CPA attack is conducted in two main phases: Round 1 and Round 2.

2.1 Correlation Power Analysis (CPA)

CPA leverages power consumption data during encryption to identify key bytes. It compares the Hamming weight of intermediate values (derived from key guesses) against actual power traces.

Round 1 Analysis

We focus on extracting key bytes K_0 to K_7 by correlating power traces with Hamming weights from the ART output of the first round. The following steps are performed:

1. **SubCells** – Input is set to 00 (since $AD = 00$). The operation is performed on the first 8 bytes.
2. **AddConstants** – The resulting state is XORed with a constant matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

3. **AddRoundTweakey** – Tweak Keys (**TK1**, **TK2**, **TK3**) are extracted, with:
   - **TK1**:
   $$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 26 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$
   - **TK2**: Nonce from the provided plaintext file.
   - **TK3**: The 16 bytes of keys to be guessed.

Correlation Steps

For each byte of the key guess, we calculate the Hamming weight of the ART output and correlate it with the power traces. The process includes:

1. **Divide and Conquer** – Each key byte guess has 256 possible values. We compute Hamming weights for each byte and correlate with power traces.
2. **Discarding False Positives** – Patterns in key guesses help narrow the search space. Identifying consistent differences in successive key guesses can reduce false positives.

Round 2 Analysis

To recover K_8 to K_{15}, we apply a similar CPA approach to the second round of encryption after performing the necessary operations from Round 1. The Tweak Keys for Round 2 are obtained similarly, and we also need to guess the LFSR value for maximum correlation.

2.2 Implementation Details

The attack was implemented in Python using NumPy and power traces loaded from .npy files. The core code snippet responsible for correlation calculation is as follows:

```
for key_guess in range(256):
    for point in range(search_start, search_end):
        correlation_value = correlation_array[key_guess, point]
        results.append((byte_index, key_guess, point, correlation_value))

        if correlation_value > max_correlation:
            max_correlation = correlation_value
            max_point = point
            max_key_guess = key_guess
```

This script analyzes correlation points for key bytes and identifies the strongest key guesses.

3 Results

The CPA attack yielded key guess correlations showing significant patterns among:

- Differences between Byte 0 and Byte 1
- Differences between Byte 1 and Byte 2
- Differences between Byte 2 and Byte 3

The analysis revealed consistent differences in successive key guesses, aiding in the recovery of the key bytes.

4 Conclusion

The CPA attack on Romulus-N encryption successfully identified key byte correlations using power traces from the first round. By filtering results and analyzing differences between key guesses, we effectively recovered the key bytes K_0 to K_7. The final retrieved key is:

$$\text{key} = \texttt{"93 4d 67 9d 1f 0a 5c 62 df 0f 00 a7 0b e8 ae 86"}$$

5 References

- Romulus-N GitHub repository: https://github.com/romulusae/romulus
- Points document on CPA results (attached).