Correlation Power Analysis (CPA) Attack on AES

Introduction

This report documents an implementation of a Correlation Power Analysis (CPA) attack on the Advanced Encryption Standard (AES). Side-channel attacks exploit physical leakages such as power consumption during cryptographic operations to recover secret keys. Here, AES encryption power traces are analyzed to recover the 128-bit secret key.

Code Overview

The Python script takes two input files: - plaintexts.npy: Contains plaintext blocks used for AES encryption. - power_traces.npy: Contains measured power consumption traces. It also defines a target ciphertext (hexadecimal string) and attempts to recover the AES key by analyzing the correlation between hypothetical power consumption and measured traces.

Key Components of the Script

- SBOX and Hamming Weight (HW) models are defined for AES. - cpa_recover_byte(): Performs correlation for each possible key byte guess (0-255). - recover_key(): Repeats CPA for all 16 bytes of the AES key. - try_decrypt_key(): Uses the recovered key to attempt decryption of the given ciphertext.

Attack Methodology

1. Load plaintexts and power traces from numpy arrays. 2. For each key byte, compute hypothetical power consumption using the model: HW[SBOX[plaintext_byte XOR key_guess]] 3. Correlate hypothetical power with actual power traces. 4. The key guess with the highest correlation is selected. 5. Repeat for all 16 key bytes. 6. Attempt to decrypt the given ciphertext using the recovered key.

Expected Output

- Recovered AES key (hexadecimal format). - Correlation scores for each key byte. - Decrypted plaintext (if the correct key is found).

Conclusion

This CPA attack demonstrates the feasibility of recovering AES secret keys by analyzing side-channel information. Such attacks pose significant risks to cryptographic implementations if not properly protected. Countermeasures such as masking, hiding, and secure hardware design are essential to prevent leakage.