Integral Cryptanalysis of AES
Saad Khan(2021068)

Motivation

Integral cryptanalysis is implemented on AES to evaluate and enhance its security by identifying potential vulnerabilities. This method exploits the algebraic structure of AES, focusing on the relationships between multiple plaintexts and their corresponding ciphertexts. By doing so, it helps in understanding how AES resists various cryptographic attacks and aids in developing more robust encryption standards. This analysis is crucial for ensuring the reliability of AES in securing sensitive data against increasingly sophisticated threats.

Aim

The aim of implementing integral square cryptanalysis on AES is to rigorously test and validate the algorithm's resilience against cryptographic attacks. By analyzing the propagation of plaintext differences through the encryption process, researchers can identify weaknesses and potential attack vectors. This helps in refining the AES algorithm, ensuring it remains a robust and secure standard for encrypting sensitive information. Ultimately, the goal is to maintain the integrity and confidentiality of data in the face of evolving cyber threats.

1. Introduction

Cryptographic algorithms like the Advanced Encryption Standard (AES) form the backbone of modern data security. This report delves into two Python-based implementations: one demonstrating the AES encryption process and another performing a cryptanalytic integral, SQUARE attack on a reduced-round AES. The purpose of this report is to provide a detailed analysis of both implementations, showcasing their workflow, strengths, limitations, and practical relevance.

- **AES Implementation:** Explains the step-by-step encryption process.
- **Integral SQUARE Attack:** Demonstrates a cryptanalytic technique to recover the key used in reduced-round AES.

2. Objective

AES Implementation

To provide a Python implementation of AES encryption, enabling encryption of plaintext using 128-bit keys. The goal is to illustrate the key components of AES, such as key expansion and transformations, and demonstrate how these steps combine to produce secure ciphertext.

SQUARE Attack

To implement the SQUARE attack, a known cryptanalytic technique targeting reduced-round AES. This implementation simulates the attack to recover the last round key and, subsequently, the original key, exploiting structural weaknesses in reduced-round AES.

3. AES Implementation

Features of AES

AES is a symmetric block cipher standardized by NIST. It encrypts data in fixed-size 128-bit blocks and supports key sizes of 128, 192, and 256 bits. Our AES is implemented for 128-bit keys. The key features include:

1. **SubBytes:** Non-linear byte substitution using an S-box.
2. **ShiftRows:** Cyclic row shifts to achieve diffusion.
3. **MixColumns:** Column-wise transformations in Galois Field to enhance diffusion.
4. **AddRoundKey:** XOR operation between the state and the round key. 5. **Key Expansion:** Generates multiple round keys from the original key using rotations, substitutions, and XOR operations.

Workflow

The AES implementation encrypts plaintext in 10 rounds using a 128-bit key. The steps include:

1. **Initialization:**
    - Convert plaintext and key into 4x4 matrices (state format).
    - Generate 11 round keys through key expansion.
2. **Rounds:**
    - **Initial Round:** XOR the plaintext matrix with the first round key (AddRoundKey).
    - **Main Rounds (1–9):**
        - Perform SubBytes, ShiftRows, MixColumns, and AddRoundKey transformations.
    - **Final Round (10):**
        - Perform SubBytes, ShiftRows, and AddRoundKey (MixColumns is skipped).
3. **Output:** The final state matrix is converted into a hexadecimal ciphertext.

Key Functions

1. **`substitution`:** Applies the S-box for byte substitution.
2. **`shiftRows`:** Performs cyclic row shifts.
3. **`mix_columns`:** Multiplies the state matrix with a fixed polynomial in Galois Field.
4. **`add_round_key`:** XORs the state with the round key.
5. **`key_expansion`:** Expands the original key into multiple round keys

4. Integral SQUARE Attack

The INTEGRAL SQUARE attack is a cryptanalytic technique targeting block ciphers like AES. It relies on the properties of Λ-sets (balanced sets of plaintexts) to deduce key information in reduced-round AES implementations. This attack is ineffective against full-round AES but serves as a partial cryptanalysis for 4 rounds of AES.

Workflow

1. **Λ-Set Generation:**

    - Generate a set of 256 plaintexts where one byte cycles through all possible values, while others remain constant.
2. **Encryption:**
    - Encrypt the Λ-set using a reduced-round AES with the known key.
3. **Final Round Reversal:**
    - Reverse the AddRoundKey and SubBytes transformations of the last round using a guessed key byte.
4. **Validation:**
    - Check the balance property of the reversed Λ-set (XOR of all bytes in an index equals 0).
5. **Key Recovery:**
    - Iterate over all key byte positions, refine guesses, and recover the last round key.
    - Reverse the AES key schedule to reconstruct the original key.

Key Functions

1. `create_active_set`**:** Generates the Λ-set with active and passive bytes. 2. `generate_encrypted_set`**:** Encrypts the Λ-set using the AES implementation. 3. `reverse_last_round`**:** Reverses the last round transformations for a guessed key byte.
4. `validate_guess`**:** Validates a guessed key byte using the balance property.
5. `recover_round_key`**:** Recovers the last round key by iteratively validating guesses.
6. `reverse_key_schedule`**:** Reconstructs the original key from the last round key.

‘

5. Observations for SQUARE Attack

1. **Targeting Reduced-Round AES:**
    - The SQUARE attack is specifically designed for reduced-round AES implementations (e.g., 4 rounds). It exploits structural weaknesses that are mitigated in full-round AES (10 rounds for AES-128). The reduced number of transformations allows patterns in the encryption process to emerge,

making it possible to deduce key information.

2. **Λ-Set Properties:**
   - A Λ-set is a balanced set of plaintexts where one byte (active byte) varies across all 256 possible values, and the remaining bytes (passive bytes) are fixed. This structure ensures that certain properties, like balanced XOR sums, persist through encryption rounds under specific conditions. ○ In the attack, the Λ-set helps identify patterns that lead to the recovery of the key by analyzing the XOR balance property after reversing the final round.
3. **Dependency on AddRoundKey and SubBytes:**
   - The attack leverages the linear nature of the AddRoundKey step and the deterministic behavior of SubBytes. By guessing one byte of the key at a time, the attack reverses these transformations and validates the guess using the balance property.
4. **Iterative Key Byte Recovery:**
   - Each byte of the last round key is recovered independently. By validating each guessed byte using the balance property of the Λ-set, the attack progressively reconstructs the last round key. This modular approach simplifies the overall recovery process.
5. **Reversal of the Key Schedule:**
   - After recovering the last round key, the attack uses the AES key expansion algorithm in reverse to reconstruct the original encryption key. This step highlights the deterministic nature of AES key expansion, which allows backward inference of the original key.
6. **Computational Overhead:**
   - The attack requires iterating through all 256 possible values for each byte of the last round key, followed by additional checks for validating each guess. This leads to a computational complexity of approximately $256 \times 16 = 4096$ operations for the key byte recovery phase alone, excluding the cost of key schedule reversal. 7.

Balance Property Verification:

   - The attack relies heavily on the assumption that the XOR of all bytes in a specific position within the Λ-set remains balanced (i.e., equals 0) after transformations. This property is crucial for validating guesses during key recovery.
8. **Scalability Challenges:**
   - While the attack is effective for 4-round AES, it becomes computationally infeasible for full-round AES due to the additional transformations (e.g., MixColumns) that obscure the balance property. The attack's effectiveness diminishes as the number of rounds increases.

6. Conclusion

The integral SQUARE attack implementation effectively demonstrates how structural weaknesses in reduced-round AES (e.g., 4 rounds) can be exploited to recover the encryption key using active sets and balance properties. By iteratively reversing the last round transformations and leveraging the deterministic nature of AES key expansion, the attack reconstructs the original key. However, its practicality is limited to reduced-round AES, as the additional complexity and diffusion in full-round AES (10 rounds for AES-128) neutralize such vulnerabilities. This implementation serves as both an educational tool and a reminder of the importance of adhering to cryptographic standards, such as sufficient rounds and key lengths, to ensure encryption security.