Report on AES-256 Encryption: Theoretical Aspects and Practical Implementation

Aadit Sharma
2101AI02

1 Introduction to AES-256 Encryption

1.1 What is AES?

The Advanced Encryption Standard (AES) is a symmetric block cipher established by the National Institute of Standards and Technology (NIST) in 2001. AES is widely used for secure data encryption and adheres to a robust encryption mechanism. It replaced the Data Encryption Standard (DES) due to its superior security and performance capabilities.

AES operates on fixed-size blocks of data (128 bits) and supports key lengths of 128, 192, and 256 bits. AES-256, which employs a 256-bit key, is the most secure variant and is suitable for high-security applications.

1.2 Characteristics of AES-256

- **Symmetric Encryption**: The same key is used for encryption and decryption.
- **Block Cipher**: Data is encrypted in fixed-size blocks (128 bits).
- **Key Length**: A 256-bit key provides a higher level of security against brute-force attacks.
- **Resistance to Cryptanalysis**: AES-256 is resistant to known cryptanalytic attacks when implemented correctly.

2 Theoretical Background

2.1 AES Structure

AES encryption consists of a series of transformations on the plaintext block. These transformations are repeated for 14 rounds in AES-256, with each round employing a unique round key derived from the main encryption key.

The primary steps in AES encryption and decryption include:

1. **Key Expansion**: The 256-bit key is expanded into 15 round keys using the Rijndael key schedule.
2. **Initial Round**: *AddRoundKey*, which XORs the plaintext with the first round key.

3. **Main Rounds (13 rounds)**:
    - *SubBytes*: Perform byte substitution using an S-box.
    - *ShiftRows*: Rotate rows of the state array.
    - *MixColumns*: Combine data from each column.
    - *AddRoundKey*: XOR the state array with the round key.
4. **Final Round**:
    - *SubBytes*
    - *ShiftRows*
    - *AddRoundKey (no MixColumns).*

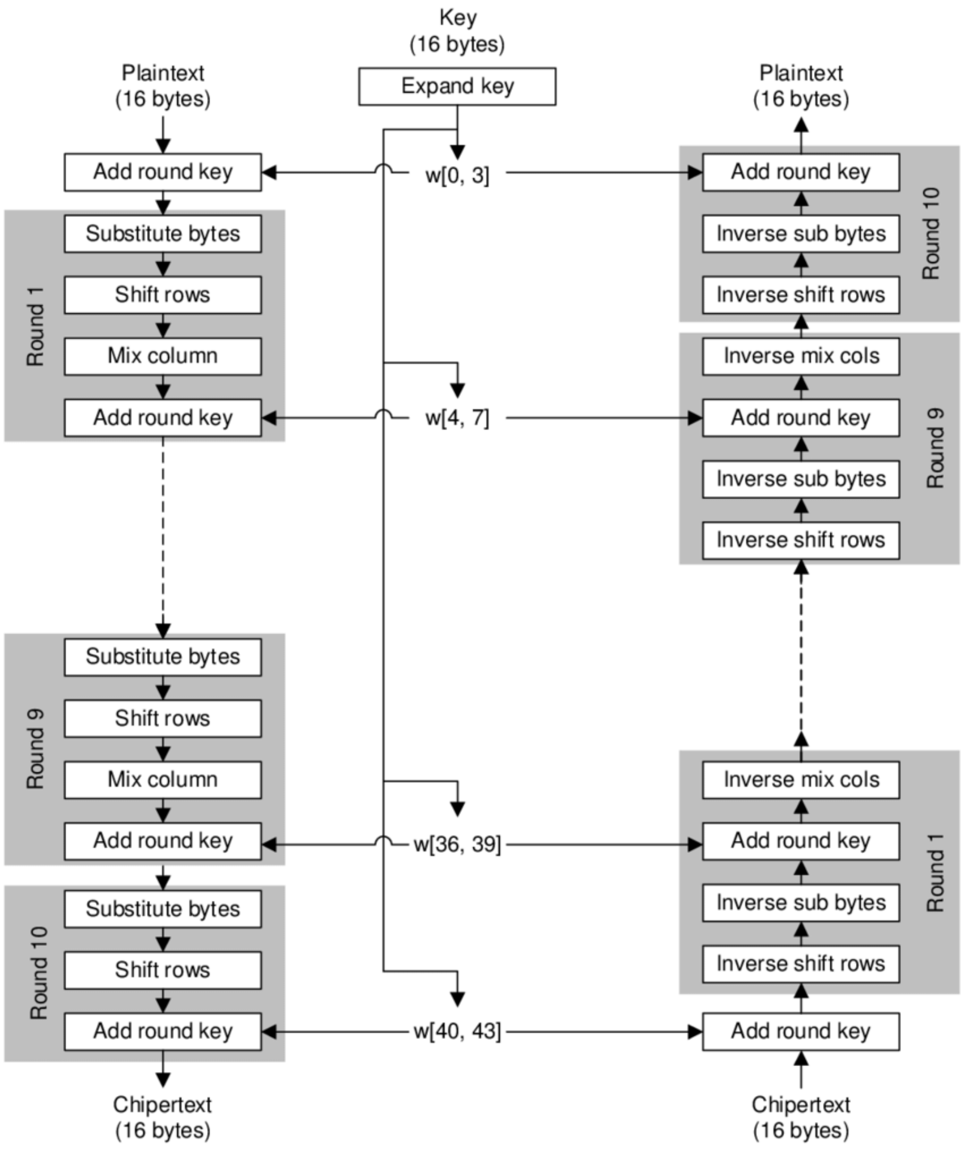

Figure 1: Flow Diagram of the AES Encryption Process

3 Identification of Potential Vulnerabilities

While AES-256 is highly secure, vulnerabilities may arise due to improper implementations or operational practices:

- **Key Management Issues**: Hardcoding keys or improper key storage can expose sensitive encryption keys.
- **Weak Initialization Vectors (IVs)**: Reusing IVs across encryption sessions can compromise data confidentiality.
- **Padding Oracle Attacks**: Improper padding validation during decryption can leak information about plaintext.
- **Side-Channel Attacks**: Attacks exploiting power consumption, electromagnetic leaks, or timing variations during encryption.

4 Proposed Mitigation Strategies

- **Secure Key Management**:
  - Use hardware security modules (HSMs) or key management systems (KMS) for key storage.
  - Avoid hardcoding keys in source code or storing them in plaintext.
- **Randomized IVs**:
  - Generate a unique, cryptographically secure IV for every encryption session.
- **Use Authenticated Encryption Modes**:
  - Employ modes like Galois/Counter Mode (GCM) or Counter with CBC-MAC (CCM) for data integrity and confidentiality.
- **Mitigating Side-Channel Attacks**:
  - Use constant-time algorithms to prevent timing analysis.
  - Shield hardware devices to reduce electromagnetic emissions.
  - Employ noise injection or data obfuscation techniques.

5 Impact of Different Key Sizes

AES supports three key sizes: 128, 192, and 256 bits. The security and performance implications of each are as follows:

- **AES-128**:
  - Fastest among the three variants.
  - Adequate security for most applications.
- **AES-192**:

  - Higher security than AES-128.
  - Moderate performance impact compared to AES-128.
- **AES-256**:
  - Maximum security, resistant to brute-force attacks even against quantum computers.
  - Slightly slower than AES-128 and AES-192 due to additional rounds.

6 Discussion on Potential Side-Channel Attacks

Side-channel attacks exploit physical properties of cryptographic operations to extract sensitive information:

- **Timing Attacks**: Measure execution time variations to deduce secret keys.
- **Power Analysis Attacks**: Analyze power consumption during encryption to infer keys.
- **Electromagnetic Analysis Attacks**: Capture electromagnetic emissions to gather cryptographic secrets.

6.1 Mitigation Strategies for Side-Channel Attacks

- Implement constant-time cryptographic algorithms to prevent timing attacks.
- Add random delays or noise to power consumption patterns.
- Use hardware shielding to prevent electromagnetic data leakage.

6.2 Performance Analysis

6.2.1 Encryption and Decryption Speeds

Performance was tested for various input sizes ranging from 1 KB to 10 MB. The average encryption and decryption times are as follows:

Input Size (KB)	Encryption Time (ms)	Decryption Time (ms)
1	0.05	0.07
10	0.10	0.08
100	0.45	0.45
1024	15.26	1.19
10240	35.90	31.08

Table 1: Encryption and Decryption Times for Various Input Sizes

6.2.2 Visualisations

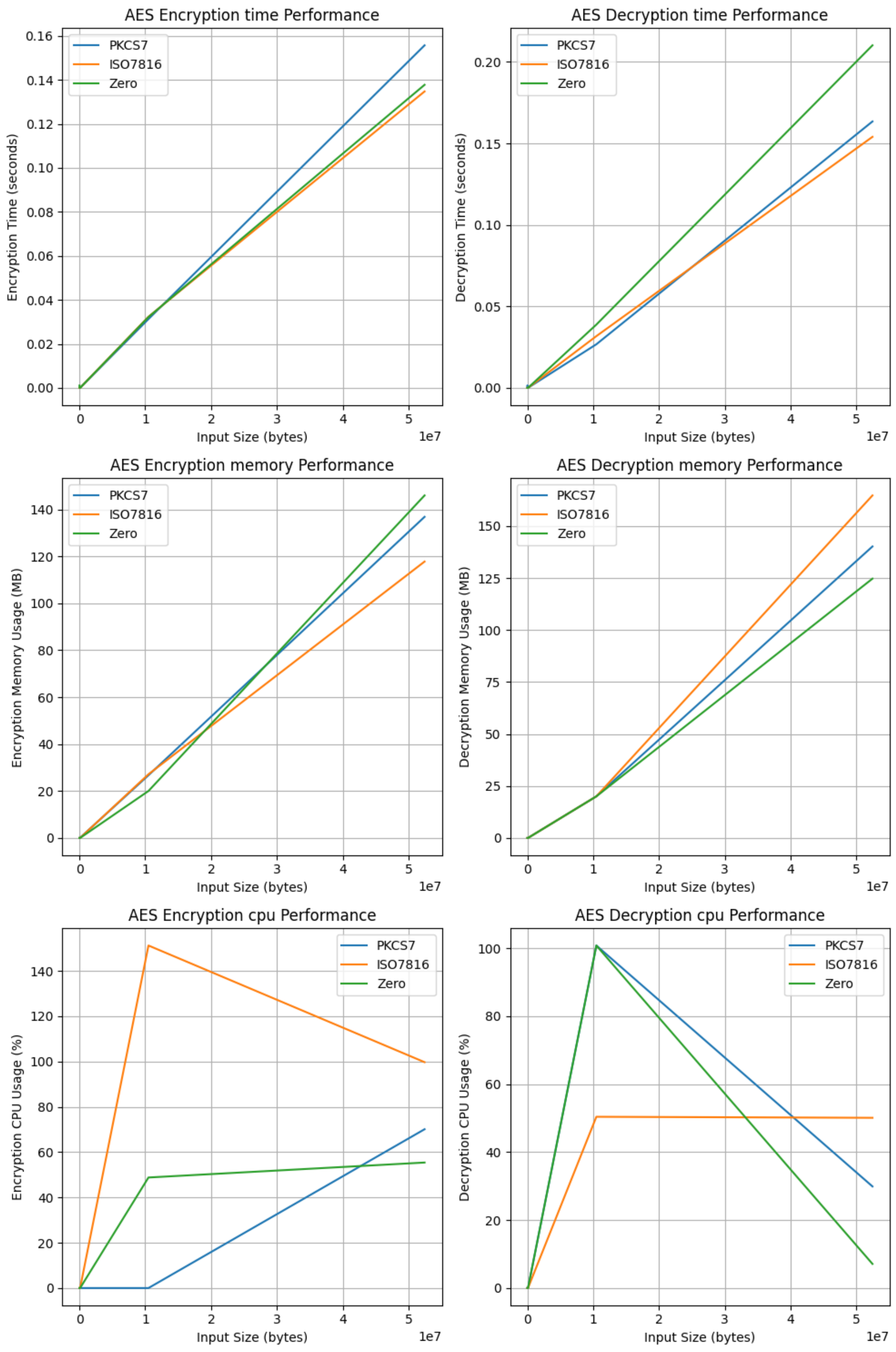

Figure 2: Encryption Times for AES-256

7 Conclusion

AES-256 remains a cornerstone of modern encryption standards due to its robust security and efficiency. The practical implementation in Python demonstrates its ease of use and adaptability across applications. When combined with sound security practices, AES-256 can safeguard sensitive data against evolving threats.