Assignment 2 Report: Design and Implementation of AES Encryption System

Contributors:
Paras Dhiman (2021482)
Aekansh Kathunia (2021127)

February 21, 2024

Abstract

This report presents the design and implementation of an Advanced Encryption Standard (AES) encryption system. AES is a widely used symmetric encryption algorithm known for its security and efficiency. The system is implemented in Python and includes key components such as key expansion, byte substitution, row shifting, column mixing, and round key addition. Sample inputs and outputs are provided to demonstrate the functionality of the system, along with validation checks to ensure the correctness of the encryption and decryption processes.

1 Introduction

In today's digital age, secure communication and data protection are of paramount importance. Encryption algorithms play a crucial role in safeguarding sensitive information from unauthorized access and tampering. AES, adopted as a standard by the U.S. government, is widely used in various applications ranging from securing communications over the internet to encrypting stored data on devices.

This report outlines the design and implementation of an AES encryption system, providing insights into its key components, functionality, and sample usage scenarios. The system is developed in Python programming language, offering a practical demonstration of AES encryption and decryption processes.

2 System Design:

The AES encryption system consists of several key components:

- **Key Expansion**: The original cryptographic key undergoes expansion to generate a series of round keys used in encryption and decryption rounds.
- **SubBytes**: Each byte of the state matrix is substituted with another byte from a fixed S-box, enhancing confusion.
- ShiftRows: Rows of the state matrix are cyclically shifted to provide diffusion and prevent patterns in the output.
- **MixColumns**: A mixing operation is performed on the columns of the state matrix to further increase diffusion and prevent attacks.
- **AddRoundKey**: Each byte of the state matrix is combined with the round key using bitwise XOR operation.

3 Implementation Details:

The AES encryption system is implemented in Python, utilizing functions and data structures to represent various components of the algorithm. Key functions include bytes-to-matrix conversion, matrix-to-bytes conversion, key expansion, encryption, and decryption. The system takes a plaintext

and a cryptographic key as inputs and produces ciphertext through the encryption process. Decryption reverses this process to recover the original plaintext. Sample inputs and outputs are provided to illustrate the functionality of the system.

4 Sample Input and Output

The original plaintext input is `AREUAMANAREUAMAN`, and after decryption, it remains `AREUAMANAREUAMAN`, indicating that the decryption process successfully recovered the original plaintext. This confirms that the decryption was successful, and the original plaintext matches the decrypted plaintext. Furthermore,

the outputs of the 1st round encryption and the 9th round decryption are compared to validate the encryption-decryption process. The outputs are as follows:

For 1st Round Encryption:

```
state_after_1st_round(ENCRYPTION):
[[[112, 63, 54, 111], [253, 245, 52, 212], [252, 187, 60, 17], [3, 129, 231, 17]]]
```

For 9th Round Decryption:

```
state_after_9th_round(DENCRYPTION):
[[[112, 63, 54, 111], [253, 245, 52, 212], [252, 187, 60, 17], [3, 129, 231, 17]]]
```

The outputs of the 1st round encryption and the 9th round decryption are indeed identical, confirming that the output of the 1st round is equal to the output of the 9th round, as expected.

Similarly, the outputs of the 9th round encryption and the 1st round decryption are compared:
For 9th Round Encryption:

```
state_after_9th_round(ENCRYPTION):
[[[247, 142, 48, 221], [252, 144, 92, 54], [210, 77, 169, 170], [14, 20, 181, 24]]]
```

For 1st Round Decryption:

```
state_after_1st_round(DENCRYPTION):
[[[247, 142, 48, 221], [252, 144, 92, 54], [210, 77, 169, 170], [14, 20, 181, 24]]]
```

Similarly, the outputs of the 9th round encryption and the 1st round decryption are identical, confirming the validity of the encryption and decryption processes.

5 Performance Analysis

The AES encryption and decryption processes involve various computational operations such as substitution, permutation, and key mixing. Understanding the performance characteristics of these operations is essential for assessing the efficiency of the AES algorithm in practical scenarios.

5.1 Computational Complexity

The computational complexity of AES encryption and decryption primarily depends on the number of rounds and the size of the data blocks. AES operates on 128-bit blocks and supports key sizes of 128, 192, and 256 bits. Each round of AES consists of four main operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey.

- **SubBytes**: The SubBytes operation substitutes each byte of the state matrix with another byte from a fixed S-box. This operation involves table lookups and is executed in constant time, resulting in $O(1)$ complexity per byte.
- **ShiftRows**: ShiftRows performs cyclic shifts on the rows of the state matrix, providing diffusion and preventing patterns in the output. The complexity of this operation is linear, with $O(n)$ complexity per block, where n is the number of bytes in the block.
- **MixColumns**: MixColumns applies a mixing operation to the columns of the state matrix, further increasing diffusion and preventing attacks. This operation involves matrix multiplications and additions and has a complexity of $O(n^2)$ per block.
- **AddRoundKey**: AddRoundKey combines each byte of the state matrix with the round key using bitwise XOR operation. Like SubBytes, this operation executes in constant time, resulting in $O(1)$ complexity per byte.

5.2 Efficiency Considerations

While AES provides strong security guarantees, the efficiency of its implementation is crucial for practical applications. Efficient implementations leverage parallelism, hardware acceleration, and optimization techniques to minimize computational overhead.

- **Parallelism**

  Modern CPUs and GPUs support parallel execution of instructions, enabling concurrent processing of multiple data blocks. Parallel implementations of AES can exploit this capability to enhance throughput and performance.

- **Hardware Acceleration**

  Specialized hardware accelerators such as AES-NI instructions in Intel processors and dedicated cryptographic co-processors improve the efficiency of AES operations. Offloading encryption and decryption tasks to such accelerators can significantly reduce processing time.

- **Optimization Techniques**

  Optimization techniques such as loop unrolling, instruction scheduling, and memory access optimization can further improve the performance of AES implementations. Compiler optimizations and algorithmic enhancements contribute to reducing latency and improving overall efficiency.

5.3 Real-World Performance

In real-world scenarios, the performance of AES encryption and decryption depends on various factors such as hardware capabilities, software optimizations, and workload characteristics. Benchmarking and profiling tools help assess the performance of AES implementations under different conditions and identify optimization opportunities. Overall, AES remains a viable choice for secure data encryption due to its balance of security, efficiency, and widespread support across platforms and devices. Continual advancements in hardware and software technologies further enhance the performance and scalability of AES-based cryptographic systems.

6 Conclusion

The AES encryption system presented in this report demonstrates the effectiveness and reliability of the Advanced Encryption Standard (AES) algorithm in securing sensitive information. By implementing key components such as key expansion, byte substitution, row shifting, column mixing, and round key addition, the system provides a comprehensive illustration of AES encryption and decryption processes. Throughout the implementation, it was evident that AES offers a robust framework

for cryptographic operations, ensuring data confidentiality and integrity in various applications. The systematic approach to encryption and decryption, coupled with the inherent complexity of the algorithm, reinforces its resilience against cryptographic attacks. Moreover, the flexibility of AES allows

for customization and optimization to meet specific use cases and performance requirements. Further enhancements, such as parallelization and hardware acceleration, can significantly improve the efficiency of AES-based encryption systems, making them suitable for high-throughput applications. Additionally, integration with existing software systems and frameworks can extend the applicability of

AES encryption in real-world scenarios, ranging from secure communication protocols to data storage and transmission mechanisms. In conclusion, the AES encryption system serves as a cornerstone in

modern cryptography, offering a reliable and versatile solution for protecting data privacy and confidentiality. As digital threats continue to evolve, AES remains a critical tool for safeguarding sensitive information and ensuring secure communication and information exchange in the digital era.