Nishita Lam
2101CS53

Lab-1 Symmetric Encryption

* Introduction

Symmetric encryption, also known as private key encryption, is one of the oldest and most widely used methods for securing data. This encryption method relies on a single, shared key for both encryption and decryption. Its simplicity, speed, and efficiency makes it a critical component of modern cryptographic systems.

1. Core Mathematical Principles:

a.) Substitution and Permutation:- Symmetric Encryption algorithm uses substitution to replace plaintext elements with ciphertext elements and permutation to shuffle data

b.) Modular Arithmetic: Algorithms like AES often use modular arithmetic where numbers wrap around after reaching fixed modulus.

c.) XOR operation. The logical operator is a fundamental operation in symmetric encryption and decryption when combined with same key.

d.) Block and Stream Ciphers:
- Block cipher: Divide data into fixed-size blocks and encrypt them. Eg:- AES with 128-bit blocks.
- Stream Cipher: Encrypt data one bit or byte at a time.

One simple symmetric encryption is XOR operation

Encryption:- $C = P \oplus K$

Decryption:- $P = C \oplus K$

Notice that both encryption and decryption use same operation & key.

2. Key Management Challenges

a.) Key Distribution: The biggest challenge is securely distributing the shared key between parties. If intercepted, the entire system is compromised.

b) Key storage: Storing keys securely is critical - as unauthorised access can break encryption.

c) Scalability: In systems with multiple users, the number of required keys grow rapidly. Trivially, for n users, $\frac{n(n-1)}{2}$ keys are needed.

d) Key rotation and Revocation :- Regularly rotating keys minimizes the risk of compromise but this adds operational complexity.

3. Performance Statistics

a) Efficiency: Symmetric Encryption is faster than asymmetric encryption because it uses simpler mathematical operations. It is ideal for encrypting large volumes of data, such as in storage systems and file transfers.

b.) Hardware Optimization: Many algorithms (like AES) are optimized for hardware, offering even faster performance in secure chips.

c) Resource usage: Requires minimal computational resources compared to asymmetric encryption making it suitable for embedded systems.

4. Security Strengths and Vulnerabilities

→ Strengths :-

- Speed : Efficient for both encryption & decryption
- Simplicity : Straightforward implementation with well defined standards.
- Low computation. Overhead : Ideal for real-time and high-throughput systems.

→ Vulnerabilities

- Key exposure : A single key is a point of failures. If compromised, the entire communication is at risk.
- Brute force Attacks : Older Algorithms (like DES) with smaller key sizes are vulnerable. Modern algorithms mitigate this with longer key lengths.
- Replay Attacks : Without proper initialization, encryption data may be replaced by attackers.

5. Real World Applications and Use cases :

a) Data Storage Encryption: Encrypting files, hard drives, and cloud storage (eg: Bitlocker)

b) Secure Communication : VPNs (Virtual Private Networks) use symmetric encryption to protect transmitted data. Encrypted messaging apps rely on symmetric algorithms for message confidentiality.

c) Payment Systems :- Symmetric secures transactions in payment systems and ATMs. (eg. encrypting PINs during transmission).

d) Wireless Network Security: Protocols like WPA-2 use ~~using~~ symmetric encryption to secure WiFi communications.

e.) Embedded Systems: Resource constrained devices like IoT sensors rely on symmetric encryption for secure data exchange.