Week 8 revision notes

IFB240 Week 8 - Symmetric Cryptography

Overview

This week's focus is on **symmetric cryptography**, which is fundamental for providing confidentiality and integrity in information security. We explore various types of symmetric ciphers, such as **stream ciphers** and **block ciphers**, as well as how these methods can provide confidentiality, integrity, and authentication. We also look at practical applications, common issues, and the differences between symmetric and asymmetric cryptography.

1. What is Cryptography?

Definition

- **Cryptography** is the practice of transforming messages into an unintelligible form and recovering them to ensure information security.
  - It is a subset of **cryptology**, which encompasses both **cryptography** (making ciphers) and **cryptanalysis** (breaking ciphers).
- Derived from the Greek words meaning **"hidden writing"**.
- The goal is to protect data by making it unreadable to unauthorized users.

Key Terms

- **Plaintext (P)**: The original message or data (cleartext).
- **Encryption (E)**: The process of transforming plaintext into ciphertext using an algorithm and a key.
- **Cryptographic Key (K)**: Secret knowledge used in the encryption and decryption processes.
- **Ciphertext (C)**: Encrypted plaintext; data is now "hidden."
- **Decryption (D)**: The process of converting ciphertext back into plaintext using the algorithm and the key.

Example: Caesar Cipher

- **Encryption**: Shifts each character in the plaintext forward by a fixed number of positions (e.g., "how are you today" becomes "krz duh brx wrgdb").
- **Decryption**: Shifts back by the same number of positions to recover the original message.
- Note: The **Caesar Cipher** is not secure by modern standards and is easily broken.

2. Types of Symmetric Ciphers

Symmetric Cryptography Overview

- **Symmetric ciphers** use the **same key** for both encryption and decryption.
- Used primarily for providing **confidentiality**.

Types of Symmetric Ciphers

1. **Stream Ciphers**:
    - Encrypt plaintext one character at a time using a keystream.
    - **Example**: The **One-Time Pad (OTP)**, which provides perfect secrecy when:
        1. The key is truly random.
        2. The key length is the same as the message.
        3. The key is used only once.
    - **Practical Issues**: OTP is impractical due to difficulties in key management.
2. **Block Ciphers**:
    - Encrypts fixed-size blocks of data (e.g., 64-bit or 128-bit blocks).
    - **Common Modes**:
        - **Electronic Codebook (ECB)**: Each block is encrypted independently; **issues** include leakage of patterns in repeated plaintext.
        - **Cipher Block Chaining (CBC)**: Each plaintext block is XOR'd with the previous ciphertext block before encryption, making ciphertext for the same plaintext different each time.
    - **Examples**:
        - **Data Encryption Standard (DES)**: Uses 56-bit keys, now considered insecure due to vulnerability to brute-force attacks.
        - **Advanced Encryption Standard (AES)**: Modern standard with key sizes of **128, 192, or 256 bits**; used in a wide range of applications.

3. Confidentiality with Symmetric Ciphers

Symmetric Encryption for Confidentiality

- **In Storage**: Data at rest can be encrypted to prevent unauthorized access.
    - **Example**: **BitLocker** (Windows) and **FileVault** (Mac) use symmetric encryption to secure files on disk.
- **In Transmission**: Data in transit is encrypted to prevent eavesdropping.
    - **Examples**:

- **SSL/TLS**: Used for secure web communication.
- **WPA2**: Encrypts data over wireless networks.

Stream Ciphers for Confidentiality

- **Binary Additive Stream Cipher**: Encrypts each bit of plaintext by XOR'ing it with a bit from the keystream.
  - **One-Time Pad**: Provides perfect secrecy but is not practical for most uses.
- **Stream Cipher Application**: Often used in real-time applications such as mobile telephony (e.g., **A5/1** in GSM).

Block Ciphers for Confidentiality

- **Electronic Codebook (ECB)**:
  - Encrypts each block independently.
  - **Issue**: Repeated blocks in plaintext result in repeated blocks in ciphertext, which can reveal patterns.
- **Cipher Block Chaining (CBC)**:
  - Adds randomness by XOR'ing plaintext blocks with the previous ciphertext.
  - **Example**: **AES** in CBC mode provides confidentiality with additional integrity benefits due to the chaining process.

4. Integrity Assurance with Symmetric Ciphers

Hash Functions

- **Hash Functions** produce a fixed-length output from an arbitrary-length input.
- **Properties**:
  1. **Fixed Length**: Output length is constant regardless of input size.
  2. **One-Way**: Easy to compute the hash from input but infeasible to reverse.
  3. **Collision Resistance**: Hard to find two inputs with the same hash.
  4. **Avalanche Effect**: A small change in input results in a drastically different hash output.
- **Use for Integrity**: Hashes are often used to verify message integrity.
  - **Example**: Using **SHA-256** to generate a hash value and compare it to ensure data has not been altered.

Keyed Hash Functions (MACs)

- **Message Authentication Code (MAC)**: Provides both integrity and authentication.
- **How It Works**:

- **Alice** generates a message and computes a **MAC** using a secret key shared with **Bob**.
- **Bob** verifies the MAC on receipt to confirm that the message has not been altered.
- **Examples**: **HMAC-SHA256**, **CBC-MAC**.

CBC-MAC

- Uses a **block cipher in CBC mode** to compute a MAC value from the last ciphertext block.
- Used for integrity assurance and also provides authentication since only the parties with the shared key can generate a valid MAC.

5. Key Management in Symmetric Cryptography

Challenges

- **Key Distribution**: How to securely distribute symmetric keys before communication begins.
- **Key Storage**: Ensuring that secret keys remain secure during their lifecycle.
  - **Example**: Keys can be stored in encrypted files, forming a hierarchy of keys (e.g., **master keys**, **session keys**).

Importance of Key Security

- **Confidentiality**: If an attacker gains access to the key, all data encrypted with that key can be compromised.
- **Integrity**: Altering a key can render all ciphertext produced using that key unusable.
- **Availability**: Destroying a key would make all information encrypted with it permanently inaccessible.

6. Summary and Key Takeaways

- **Symmetric Cryptography** uses the same key for encryption and decryption.
- **Stream Ciphers**: Encrypt data as a stream; fast and suitable for real-time use.
- **Block Ciphers**: Encrypt fixed-size blocks of data; commonly used for secure file storage and communication.
- **Confidentiality**: Ensured by encrypting data in storage or during transmission.
- **Integrity Assurance**: Achieved using **hash functions** or **MACs**; prevents unauthorized modification.
- **Key Management**: Central to symmetric cryptography; if key security fails, the entire encryption scheme becomes ineffective.

Practical Tips

- Use **AES** in CBC mode for strong encryption and integrity.
- Always manage keys securely; consider using hardware modules (e.g., **HSMs**) for key protection.
- Combine **MACs** with symmetric encryption to ensure both confidentiality and integrity.

These notes are designed to provide a comprehensive understanding of symmetric cryptography, the types of symmetric ciphers, their applications, and practical concerns such as key management and integrity assurance. They ensure that even if you missed the lectures, you can still grasp the core concepts and their practical relevance in cybersecurity.