MODULE - 2 SYLLABUS:

SYMMETRIC CIPHERS

Stream ciphers and block ciphers, Data Encryption Standard (DES), Advanced Encryption Standard (AES) - Structure, Transformation Functions; Block Cipher Operation - Multiple encryption and triple DES, Cipher block chaining mode, Cipher feedback mode, Output feedback mode, Counter mode.

1. Stream Ciphers and Block Ciphers

1.1 Stream Ciphers

Stream ciphers are a class of encryption algorithms that encrypt data one bit or byte at a time. The idea behind stream ciphers is to generate a pseudorandom keystream (a sequence of random-looking bits) that is combined with the plaintext using an operation like XOR to produce the ciphertext. Stream ciphers are a type of symmetric encryption, meaning the same key is used for both encryption and decryption.

1.1.1 Historical Development of Stream Ciphers

Stream ciphers have a rich history that dates back to early ciphering techniques used during wartime communications. The first stream ciphers appeared in the early 20th century, where they were employed in military communications to secure radio transmissions. The development of electronic ciphers led to the creation of algorithms such as **RC4**, which became one of the most widely used stream ciphers.

The **RC4** cipher was designed by **Ron Rivest** in 1987 and became widely popular for securing wireless networks (WEP), SSL/TLS, and other communication protocols. Despite its initial success, **RC4** was later found to have several vulnerabilities, and as of today, it is largely considered insecure for many applications, leading to the development of more secure alternatives.

1.1.2 Key Features of Stream Ciphers

1. **Encryption Process**: The basic process involves the generation of a keystream that is typically created using a pseudorandom number generator (PRNG). The keystream is then XORed with the plaintext to produce the ciphertext.
    - **Keystream Generation**: A key is used to initialize the PRNG, which produces a stream of pseudorandom bits. This stream of bits is then XORed with the plaintext bits to produce the ciphertext.

- **XOR Operation**: The XOR operation is the main mathematical function used in stream ciphers. If the plaintext bit is "1" and the keystream bit is "1", the result will be "0". If the plaintext bit is "0" and the keystream bit is "0", the result will also be "0". XOR is reversible, which means the same operation can be applied during decryption to recover the original plaintext.

2. **Speed and Efficiency**: Stream ciphers are often faster than block ciphers, especially when the data is small or arrives continuously, as they can be processed in a bitwise or bytewise fashion. This makes them particularly efficient in scenarios like real-time communications (e.g., voice or video streaming).
3. **Error Propagation**: Stream ciphers exhibit minimal error propagation. This means that a single bit error in the ciphertext will only affect the corresponding bit in the plaintext during decryption. This makes stream ciphers suitable for environments where data is transmitted over unreliable channels, such as wireless networks.
4. **Key Reuse**: One of the primary weaknesses of stream ciphers is key reuse. If the same key is used to encrypt multiple messages or streams, it can lead to vulnerabilities, such as **keystream reuse attacks**. Reusing the same keystream for different pieces of data compromises the security of the entire encryption process.

1.1.3 Examples of Stream Ciphers

1. **RC4**: As mentioned, **RC4** was widely used for many years. The RC4 algorithm generates a pseudorandom keystream based on a secret key, which is XORed with the plaintext. However, due to several cryptographic weaknesses, it is now considered deprecated.
2. **Salsa20 and ChaCha20**: These are modern stream ciphers that are considered secure and are often used in scenarios where high-speed encryption is required. The Salsa20 family of ciphers, including **ChaCha20**, was designed by **Daniel J. Bernstein** to address vulnerabilities found in RC4. These ciphers are used in various cryptographic protocols, including TLS (Transport Layer Security).

1.1.4 Advantages of Stream Ciphers

- **High Speed**: Stream ciphers are generally faster and require less computational power compared to block ciphers. This makes them suitable for hardware implementations and resource-constrained environments, such as mobile devices.

- **Low Latency**: Stream ciphers have low latency, making them ideal for real-time applications where data needs to be encrypted and decrypted in small chunks.
- **No Error Propagation**: The key property of stream ciphers is that errors in one bit of the ciphertext only affect that specific bit during decryption, which does not affect the entire message.

1.1.5 Disadvantages of Stream Ciphers

- **Key Management**: The major drawback of stream ciphers is related to key management. If the key is compromised or reused, the encryption becomes insecure. It is essential to generate a unique keystream for each message or session.
- **Vulnerability to Attacks**: Stream ciphers are more susceptible to attacks like **known-plaintext** and **chosen-plaintext attacks** when the key is reused or weak.
- **Limited Security Assurance**: Compared to block ciphers, stream ciphers often lack the same level of security assurance, especially when the keystream generation process is not cryptographically strong.

1.2 Block Ciphers

Block ciphers operate on fixed-size blocks of plaintext, typically 64 bits or 128 bits, and transform these blocks into ciphertext using a symmetric key. Block ciphers apply a series of transformations and operations to the data, such as substitution and permutation, in multiple rounds, with each round depending on the key.

1.2.1 Historical Context of Block Ciphers

The development of block ciphers began in the 1970s with the **Data Encryption Standard (DES)**, which was adopted by the U.S. government as a standard for encrypting sensitive but unclassified information. DES used a 56-bit key and was based on the **Feistel network**. Although DES was highly influential, it became insecure as computational power increased, leading to the development of more secure algorithms such as **Advanced Encryption Standard (AES)**.

1.2.2 Block Cipher Characteristics

1. **Fixed Block Size**: Block ciphers operate on fixed-length blocks of data, which can range from 32 bits to 256 bits, but common block sizes are 64 and 128 bits.

2. **Multiple Rounds of Encryption**: Block ciphers generally perform multiple rounds of encryption. In each round, the plaintext block undergoes operations such as substitution (using S-boxes) and permutation (P-boxes), followed by a key mixing process. The number of rounds typically ranges from 10 to 20, depending on the algorithm (e.g., AES performs 10 rounds for a 128-bit key).
3. **Key Size**: The strength of a block cipher is largely determined by its key size. Larger keys offer higher security by increasing the number of possible keys and making brute-force attacks infeasible. Modern block ciphers use key sizes ranging from 128 bits to 256 bits, as seen in **AES**.
4. **Transformation Functions**: The security of block ciphers relies on the application of transformation functions such as:
    - **Substitution**: Replaces a byte or block with another value using a substitution box (S-box).
    - **Permutation**: Rearranges the bits or bytes in a fixed manner to provide diffusion.
    - **Key Mixing**: Combines the encryption key with intermediate values at each round to ensure that the key influences every part of the ciphertext.

1.2.3 Example: Advanced Encryption Standard (AES)

AES is the most widely used block cipher today. AES operates on 128-bit blocks and supports key sizes of 128, 192, or 256 bits. The AES algorithm involves multiple rounds of encryption, depending on the key size:

- **10 rounds** for 128-bit keys
- **12 rounds** for 192-bit keys
- **14 rounds** for 256-bit keys

1.2.4 Block Cipher Security

Block ciphers are considered secure when used with appropriate key sizes and modes of operation. For example, AES-256 (256-bit key size) is widely regarded as highly secure and resistant to cryptanalysis, while DES (56-bit key) is now vulnerable to brute-force attacks.

1.2.5 Advantages and Disadvantages of Block Ciphers

- **Advantages**:
    - **Strong Security**: Block ciphers provide strong security guarantees, especially when used with long keys and secure modes of operation.

    - **Resistant to Cryptanalysis**: Block ciphers like AES are highly resistant to attacks like differential and linear cryptanalysis, which makes them suitable for a wide range of applications.
    - **Versatility**: Block ciphers can be used in many encryption schemes, including file encryption, disk encryption, and secure communication protocols like TLS.
- **Disadvantages**:
    - **Slower than Stream Ciphers**: Block ciphers are generally slower than stream ciphers, especially when encrypting small amounts of data.
    - **Error Propagation**: In some modes of operation, an error in one block of ciphertext can affect subsequent blocks of plaintext.
    - **Complexity**: Block ciphers are more complex than stream ciphers and require more computational resources.

Certainly! Here's the combined and expanded explanation, incorporating all the details for **Data Encryption Standard (DES)**, **Advanced Encryption Standard (AES)**, and **Block Cipher Modes of Operation**.

2. Data Encryption Standard (DES)

2.1 Overview of DES

The **Data Encryption Standard (DES)**, developed by IBM and adopted by the U.S. National Institute of Standards and Technology (NIST) in 1977, was one of the first widely used block ciphers. It operates on 64-bit blocks with a 56-bit key. Although DES was considered secure in its early days, it became vulnerable to brute-force attacks as computational power increased. Today, DES is regarded as outdated and insecure for most applications, but it provides a foundational understanding of symmetric-key block ciphers.

2.1.1 Structure of DES

1. **Block Size**: DES operates on 64-bit blocks. Each block of plaintext is encrypted to produce a 64-bit block of ciphertext.
2. **Key Size**: DES uses a 56-bit key for encryption, but the key is actually 64 bits in length, with 8 bits used for parity checks.
3. **Rounds**: The encryption process consists of 16 rounds of encryption, each involving a combination of substitution, permutation, and key mixing operations.

2.1.2 Transformation Functions in DES

1. **Initial Permutation (IP)**: The input block undergoes a fixed permutation that rearranges the bits before any rounds of encryption.
2. **Round Function**: The core of DES consists of 16 rounds of processing, where each round involves:
    - **Expansion (E) Permutation**: Expands a 32-bit half-block into 48 bits.
    - **Substitution (S-boxes)**: The 48-bit data is split into eight 6-bit chunks and substituted using predefined substitution boxes (S-boxes).
    - **Permutation (P)**: After substitution, the data undergoes a permutation to spread the influence of each bit over many others.
3. **Key Mixing**: Each round uses a unique subkey derived from the original key. These subkeys are generated through the **key schedule** process.
4. **Final Permutation (FP)**: After all rounds, the resulting bits undergo a final permutation before producing the 64-bit ciphertext.

2.1.3 DES Key Generation and Subkey Schedule

The **key schedule** generates 16 subkeys, each 48 bits in length, from the original 56-bit key. The process includes:

1. **Initial Key Permutation**: A permutation (PC-1) is applied to the 56-bit key.
2. **Splitting**: The key is split into two 28-bit halves.
3. **Shifting**: Each half is cyclically shifted to create different subkeys for each round.
4. **Permutation (PC-2)**: The 28-bit halves are combined and permuted to generate the final 48-bit subkey.

2.1.4 Weaknesses of DES

The primary weakness of DES is its 56-bit key size. As computational power increased, brute-force attacks became feasible. In 1998, the **Electronic Frontier Foundation (EFF)** demonstrated that DES could be cracked in a few days using a custom-built machine. Other weaknesses, such as vulnerability to **differential cryptanalysis** and **linear cryptanalysis**, further compromised its security.

2.1.5 Attacks on DES

1. **Brute-Force Attacks**: The 56-bit key length is small enough to allow a brute-force search of all possible keys, especially with modern hardware.

2. **Differential Cryptanalysis**: This attack exploits the relationships between input differences and corresponding output differences across multiple rounds.
3. **Linear Cryptanalysis**: This method seeks linear approximations for the relationships between plaintext, ciphertext, and the key.
4. **Meet-in-the-Middle Attack**: In multi-layer encryption scenarios like **Double DES**, this attack can significantly reduce the effective key size.

2.2 Transition to AES: The Need for a New Standard

Due to the vulnerabilities in DES, there was a need for a more secure encryption algorithm. This led to the development of **AES (Advanced Encryption Standard)**, which was adopted by NIST in 2001.

3. Advanced Encryption Standard (AES)

3.1 Overview of AES

AES is a symmetric-key block cipher standardized by NIST in 2001 to replace DES. It operates on 128-bit blocks and supports key sizes of 128, 192, or 256 bits. AES is widely used in applications such as **SSL/TLS** for secure web traffic, **VPNs**, and **file encryption**. Its security is considered robust, and it is resistant to all known practical cryptographic attacks.

3.1.1 Structure of AES

1. **Block Size**: AES operates on 128-bit blocks of data, which is more efficient for most applications than the 64-bit block size used by DES.
2. **Key Sizes**: AES supports three key sizes:
    - **128 bits** for lower security
    - **192 bits** for medium security
    - **256 bits** for high security
3. **Rounds**: The number of rounds in AES varies based on the key length:
    - **10 rounds** for 128-bit keys
    - **12 rounds** for 192-bit keys
    - **14 rounds** for 256-bit keys

3.1.2 AES Transformation Functions

AES involves a series of operations to transform plaintext into ciphertext:

1. **SubBytes**: A substitution step where each byte of the state matrix is replaced using the AES S-box.
2. **ShiftRows**: The rows of the state matrix are shifted cyclically to introduce diffusion.
3. **MixColumns**: This step mixes the columns of the state matrix to further diffuse the data and strengthen the security.
4. **AddRoundKey**: The round key is added to the state matrix using the XOR operation. Each round uses a different round key generated from the original key.
5. **Final Round**: The last round omits the MixColumns step but retains SubBytes, ShiftRows, and AddRoundKey.

3.1.3 Key Expansion and Subkey Generation

AES generates a series of round keys through the **key expansion** process. The original key is expanded into a series of 4-byte words, and each word is used as part of the round key for encryption. The expansion process includes **rotations**, **substitutions**, and the application of the S-box to increase the complexity and security.

3.2 AES vs. DES: Performance and Security Comparison

- **Key Length**: AES offers stronger security due to its longer key sizes (up to 256 bits), whereas DES only uses a 56-bit key, making it vulnerable to brute-force attacks.
- **Efficiency**: AES is more efficient than DES, with faster encryption and decryption speeds due to its improved structure and design.
- **Security**: AES provides robust protection against all known attacks, including brute-force and cryptanalysis methods like **differential** and **linear cryptanalysis**.

4. Block Cipher Modes of Operation

Both DES and AES can be used with different **modes of operation** to encrypt data of arbitrary length. These modes define how to apply the block cipher to data that exceeds the block size and manage various encryption concerns like error propagation and parallelization.

4.1 Electronic Codebook (ECB) Mode

In **ECB mode**, each block of plaintext is encrypted independently using the same key. This simplicity makes it fast and efficient but also exposes the cipher to significant weaknesses.

4.1.1 Weaknesses of ECB

- **Pattern Leakage**: Identical plaintext blocks encrypt to identical ciphertext blocks. This reveals repeating patterns in the data, which can be exploited by attackers.
- **No Chaining**: Since each block is encrypted independently, ECB fails to achieve the desired diffusion and avalanche effect.

4.1.2 Use Cases for ECB

Despite its weaknesses, ECB can be used in situations where data does not exhibit repeating patterns or in cases where efficiency is paramount. An example of such use could be encrypting fixed-size records, where data patterns are controlled.

4.2 Cipher Block Chaining (CBC) Mode

CBC mode improves upon ECB by introducing **chaining**. In CBC, each plaintext block is XORed with the previous ciphertext block before encryption. The first block is XORed with an **Initialization Vector (IV)** to ensure that identical plaintext blocks produce different ciphertexts.

4.2.1 Characteristics of CBC

- **Error Propagation**: A single bit error in the ciphertext affects the current and next plaintext block during decryption.
- **IV Requirement**: CBC requires an IV to ensure that identical plaintexts encrypt to different ciphertexts.
- **Widely Used**: CBC is often used in **SSL/TLS** and **disk encryption** protocols due to its security properties.

4.3 Cipher Feedback (CFB) Mode

In **CFB mode**, the previous ciphertext block is encrypted and XORed with the plaintext to generate the ciphertext. CFB can be used in various feedback sizes, such as **CFB-1**, **CFB-8**, and **CFB-128**.

4.3.1 Characteristics of CFB

- **Stream-like Operation**: CFB operates similarly to a stream cipher, making it suitable for real-time encryption.
- **No Padding**: Since data is processed in smaller units, no padding is required.

4.4 Output Feedback (OFB) Mode

OFB mode generates a keystream by encrypting the IV and using the output to XOR with the plaintext. Unlike CBC, the encryption of the previous ciphertext block is not used; instead, the keystream is generated independently.

4.4.1 Characteristics of OFB

- **Error Propagation**: Errors in the ciphertext do not propagate to subsequent blocks.
- **Predictable Keystream**: The keystream is generated independently, making OFB susceptible to attacks if the IV is reused.

4.5 Counter (CTR) Mode

CTR mode converts the block cipher into a stream cipher by encrypting successive counter values. Each counter value is encrypted and then XORed with the plaintext to generate the ciphertext.

4.5.1 Characteristics of CTR

- **Parallelism**: CTR mode allows for parallel encryption and decryption, making it faster than other modes in hardware implementations.
- **Random Access**: CTR mode allows random access to encrypted data, making it useful in applications like disk encryption.