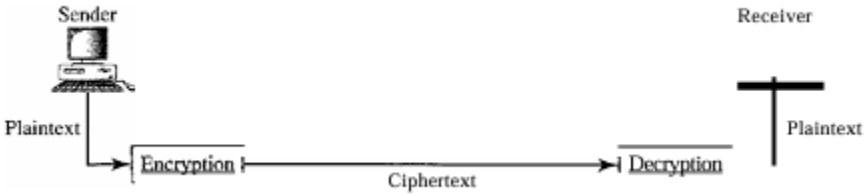
CRYPTOGRAPHY

- Transforming messages to make them safe and immune to attacks.



PLAINTEXT

- The original message before transformation is called PlainText.

CIPHERTEXT

- The message after transformation is called ciphertext

Sender = . uses encryption algorithm

Receiver = uses decryption algorithm.

CIPHER

- The encryption and decryption algorithms are called Ciphers.
- Also referred to different categories of algorithm.
- One cipher serves millions of communicating pairs.

KEY

- The number(or set of number) the cipher operates on is called Key.
- For encryption = encryption algorithm + encryption key + plaintext
- For decryption = decryption algorithm + decryption key + ciphertext

ALICE, BOB, EVE

The three characters in the information exchange scenario

ALICE - needs to send secure data

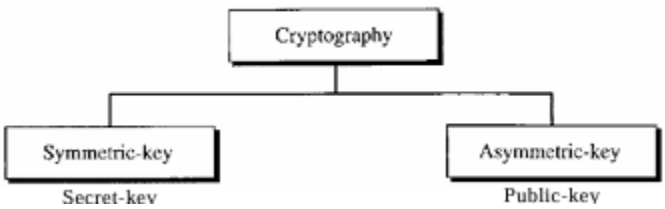
BOB - recipient of data

EVE - disturbs the communication between alice and bob.

THE THREE TYPES OF KEYS -

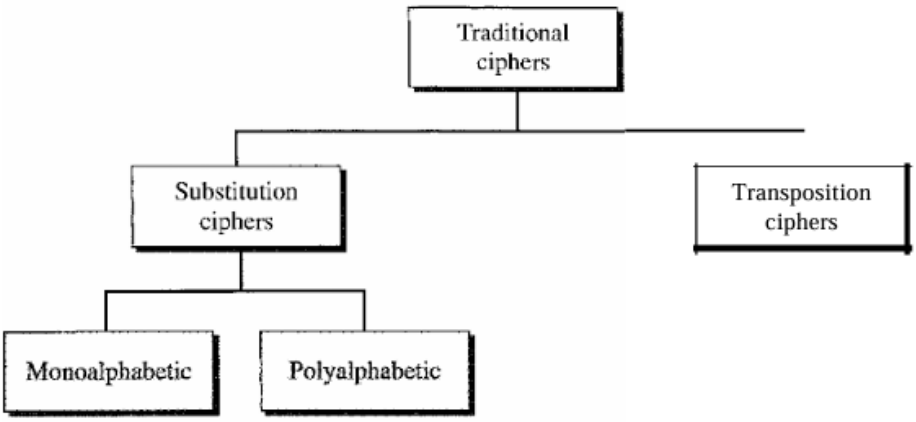
1. **SECRET KEY**
    - Used in symmetric key cryptography
2. **PUBLIC KEY –** used in asymmetric key cryptography
3. **PRIVATE KEY**

THE TWO CATEGORIES



SYMMETRIC KEY CRYPTOGRAPHY

1. **TRADITIONAL CIPHER**
    - Character-oriented cipher.



SUBSTITUTION CIPHER

- Substitution of one symbol by another.

a) MONOALPHABETIC CIPHER

- a character (or a symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position in the text.

b) POLYALPHABETIC CIPHER

- each occurrence of a character can have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is a one-to-many relationship.
- For example, character A could be changed to D in the beginning of the text, but it could be changed to N at the middle. It is obvious that if the relationship between plaintext characters and ciphertext characters is one-to many, the key must tell us which of the many possible characters can be chosen for encryption.
- To achieve this goal, we need to divide the text into groups of characters and use a set of keys. For example, we can divide the text "THISISANEASYTASK" into groups of3 characters and then apply the encryption using a set of3 keys. We then repeat the procedure for the next 3 characters

SHIFT CIPHER

- Simplest monoalphabetic cipher.
- Assumption = plain and cipher text contains only uppercase letters (A TO Z).
- Encryption = "shift key characters down," key = defined by user.
- Decryption = "shift key characters up." , key = as per encryption.
- Also called **CAESAR CIPHER. - key = 3**

TRANSPOSITION CIPHER

- No substitution, but the location of the text changes.
- A transposition cipher reorders (permutes) symbols in a block of symbols.

KEY

- the key is a mapping between the position of the symbols in the plaintext and cipher text.

SIMPLE MODERN CIPHERS

- Ciphers needed to be bit-type, because we send audios, music, videos as well.
- It is convenient to convert these types of data into a stream of bits, encrypt the stream, and then send the encrypted stream
- when text is treated at the bit level, each character is replaced by 8 (or 16) bits, which means the number of symbols becomes 8 (or 16).
- Mingling and mangling bits provides more security than mingling and mangling characters.

XOR CIPHERS

- Made of -set of simple ciphers.
- Uses of exclusive-or operation.
- Needs two inputs, 1 = plaintext , 2 = key
- Size of key, plaintext and ciphertext is same.
- Imp. property - the encryption and decryption are the same.

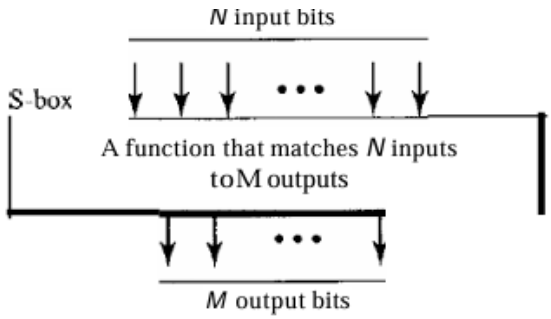
ROUNDED CIPHERS

- Input bits are rotated to left or right.
- Two types = keyed or keyless.
- **Keyed =** value of key = number of rotations.
- **Keyless =** no. of rotations is fixed.
- Considered as a special case of transpositional cipher instead of characters, bits are transposed instead of characters.

Example - length of original string = N
Number of rotations = 1 - (N-1)

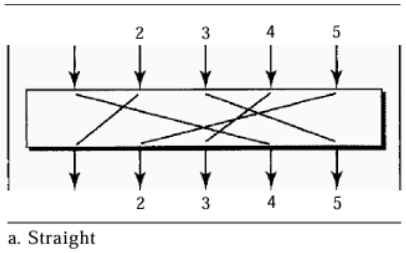
SUBSTITUTION CIPHER : S-BOX

- parallels the traditional substitution cipher for characters.
- The input to an S-box is a stream of bits with length N; the result is another stream of bits with length M. And N and M are not necessarily the same
- Figure 30.11 shows an S-box. The S-box is normally keyless and is used as an intermediate stage of encryption or decryption. The function that matches the input to the output may be defined mathe matically or by a table

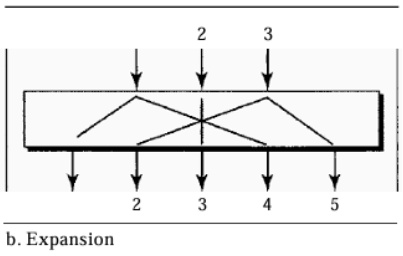


TRANSPOSITION CIPHER : P-BOX

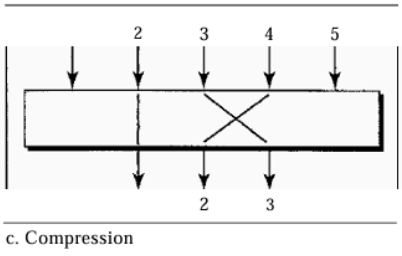
- A P-box (permutation box) for bits parallels the traditional transposition cipher for char acters. It performs a transposition at the bit level; it transposes bits.
- It can be implemented in software or hardware, but hardware is faster. P-boxes, like S-boxes, are nunnally key less. We can have three types of pennutations in P-boxes: the straight permutation, expansion permutation, and compression permutation.
- A straight permutation cipher or a straight P-box has the same number of inputs as outputs. In other words, if the number of inputs is N, the number of outputs is also N. In an expansion pennutation cipher, the number of output ports is greater than the number of input ports. In a compression pennutation cipher, the number of output ports is less than the number of input ports



a. Straight



b. Expansion



c. Compression

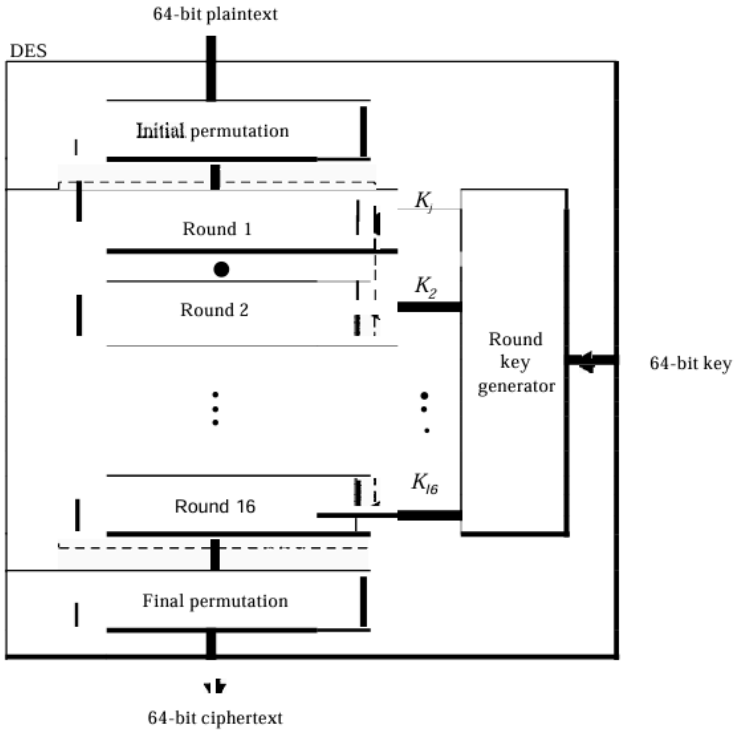
MODERN ROUND CIPHERS

- The ciphers of today are called round ciphers because they involve multiple rounds, where each round is a complex cipher made up of the simple ciphers that we previously described.
- The key used in each round is a subset or variation of the general key called the round key.
- If the cipher has N rounds, a key generator produces N keys, Kb , Kz, ..., KN, where K1 is used in round 1, K2 in round 2, and so on.

we introduce **two modem symmetric-key ciphers: DES and AES**. These ciphers are referred to as **block ciphers** because they divide the plaintext into blocks and use the same key to encrypt and decrypt the blocks. DES has been the de facto standard until recently. AES is the formal standard now.

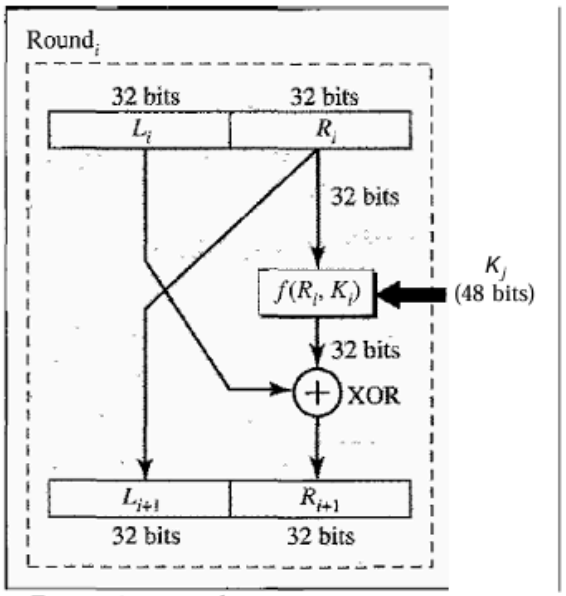
DES - DATA ENCRYPTION STANDARD

The algorithm encrypts a 64-bit plaintext block using a 64-bit key

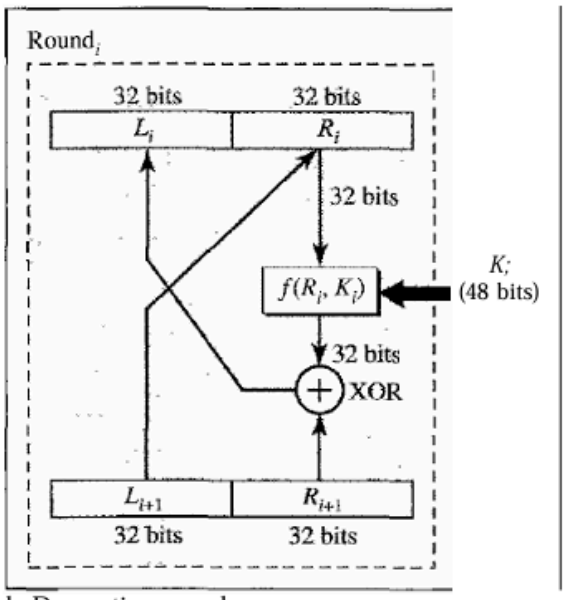


- 2 blocks (P-Boxes) + 16 round ciphers.
- The initial and final permutations are keyless straight permutations that are the inverse o feach other. The permutation takes a 64-bit input and permutes them according to predefined values.
- Each round of DES is a complex round cipher,as shown in Figure30.14.

- Note that the structure of the encryption round ciphers is different from that the decryption one.
- **Main Components:**
    - **Initial Permutation (IP):** A keyless permutation of the 64-bit plaintext before entering the rounds.
    - **16 Rounds of Encryption:** Each round uses a different subkey derived from the original 64-bit key.
    - **Final Permutation (FP):** Another keyless permutation applied to the output of the 16 rounds



a. Encryption round



b. Decryption round

How DES Works

1. **Initial Permutation (IP):**
   The 64-bit plaintext undergoes an initial permutation, rearranging the bits according to a predefined pattern.
2. **Splitting into Halves:**
   The permuted block is divided into two 32-bit halves, called the **Left (L)** and **Right (R)** halves.
3. **Round Function:**
   Each of the 16 rounds involves:
    - Expanding the right half from 32 bits to 48 bits using an **expansion permutation**.

- Combining the expanded right half with a 48-bit subkey using an **XOR operation**.
- Passing the result through a set of **S-boxes** (substitution boxes), which reduce the 48-bit input back to 32 bits.
- Applying a **straight permutation** to the output of the S-boxes.
- XORing the result with the left half to produce the new right half.

4. **Swapping:**
   After each round, the right half becomes the new left half, and the output of the round function becomes the new right half.
5. **Final Permutation (FP):**
   After 16 rounds, the left and right halves are combined, and a final permutation is applied to produce the 64-bit ciphertext.

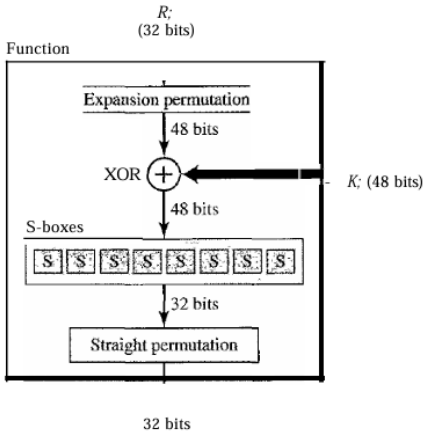
Subkey Generation

The key used in DES is not directly applied to the encryption rounds. Instead:

- The original 64-bit key undergoes an **initial permutation**.
- The key is divided into two 28-bit halves.
- For each round, the halves are shifted left by 1 or 2 bits (depending on the round) and combined.
- A **key compression permutation** reduces the combined key from 56 bits to 48 bits.

DES FUNCTION

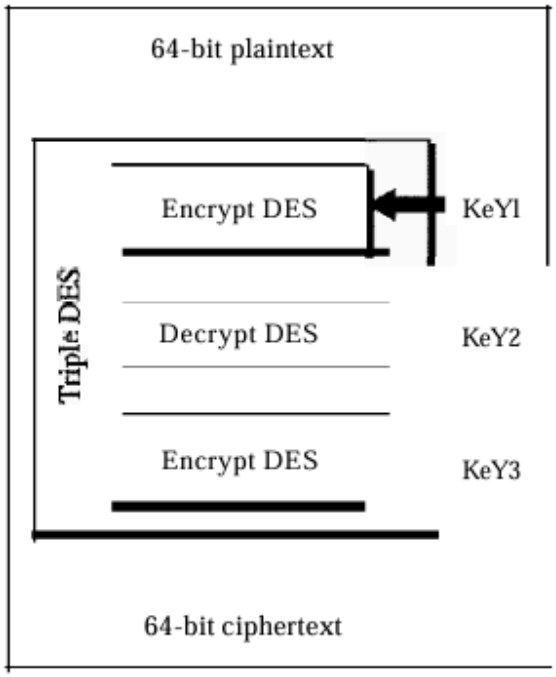
- Heart of DES.
- Applies to 48-bit key to the rightmost 32 bits R(i) to produce a 32-bit output.
- Made up of 4 operations = XOR + expansion permutation + group of S-Boxes + straight permutation.



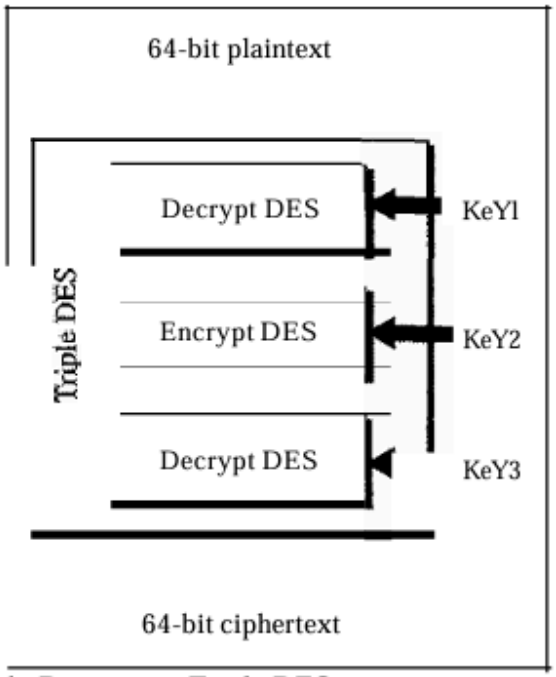
Security of DES

Despite its complexity, DES has certain vulnerabilities:

- The **effective key length** of 56 bits makes it susceptible to brute-force attacks.
- Cryptographic analysis and advances in computing power made DES less secure over time.



a. Encryption Triple DES



b. Decryption Triple DES

Triple DES (3DES)

To address the weaknesses of DES, **Triple DES (3DES)** was developed. 3DES enhances security by applying the DES algorithm three times to each data block.

How 3DES Works

1. **Encryption-Decryption-Encryption (EDE) Process:**
    - In encryption mode:
        1. Encrypt the plaintext using the first DES key.
        2. Decrypt the result using the second DES key.
        3. Encrypt the result again using the first DES key.
    - In decryption mode:
        1. Decrypt the ciphertext using the first DES key.
        2. Encrypt the result using the second DES key.
        3. Decrypt the result again using the first DES key.
2. **Key Variants:**
    - **Two-Key 3DES:** Uses two keys, effectively creating a 112-bit key size. The first and third keys are the same, making it backward-compatible with single DES.
    - **Three-Key 3DES:** Uses three independent keys, resulting in a 168-bit key size for stronger security.

Strengths and Limitations

- **Strengths:**
    - DES introduced innovative cryptographic techniques and served as a foundation for modern ciphers.
    - 3DES extended the usability of DES by improving security without requiring entirely new hardware.
- **Limitations:**
    - DES's 56-bit key is too short by modern standards.
    - 3DES, while more secure, is computationally expensive and slower than newer algorithms like AES.

AES - ADVANCED ENCRYPTION STANDARD
It was designed because the key size in DES was too small.
Although the key size was increased in the triple DES, the process was too slow.
AES = complex round cipher.

Size ofData Block	*Number ofRounds*	*Key Size*
128 bits	10	128 bits
	12	192 bits
	14	256 bits

AES has three different configurations with respect to the number ofrounds and key size.

Key Features of AES

1. **Block Size:**
    - AES uses a fixed block size of 128 bits.

2. **Key Sizes:**
    - AES supports three key sizes:
        - **128 bits:** 10 rounds
        - **192 bits:** 12 rounds
        - **256 bits:** 14 rounds
    - Each configuration has a corresponding number of rounds for encryption and decryption.

3. **Rounds:**
    - Each round includes a series of cryptographic transformations to ensure data security.
    - The final round differs slightly as it omits one operation.

General Structure of AES

AES begins with an initial XOR operation and is followed by a series of rounds. The general structure includes:

1. **Initial AddRoundKey Operation:**
    - The plaintext block is XORed with the initial key (round key K0K_0).

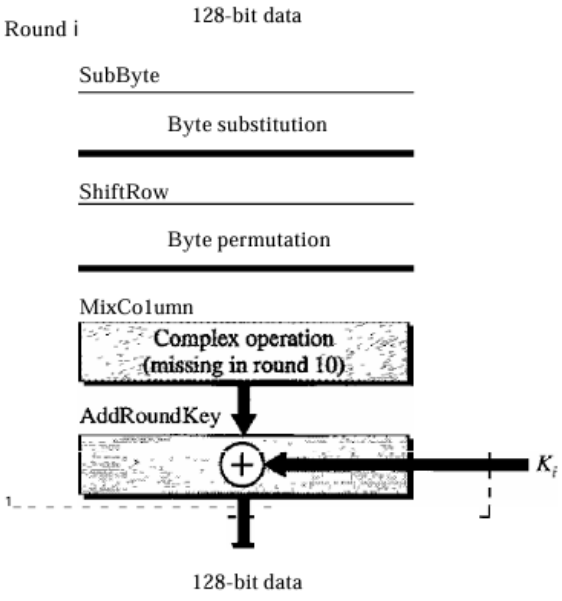
2. **Main Rounds:**
    - A total of n−1 rounds, where nn depends on the key size (10, 12, or 14 rounds).

3. **Final Round:**
    - The last round omits the **MixColumns** step present in the main rounds.

Structure of Each Round

Each round in AES (except the final round) consists of the following operations:

1. **SubBytes (Byte Substitution):**
    - Each byte in the data block is replaced with a corresponding byte from a predefined **Substitution Box (S-box)**.
    - This step provides non-linearity, making the cipher resistant to cryptanalysis.



2. **ShiftRows (Row Shifting):**
    - The rows of the data block are shifted by a specific number of positions.
    - This operation provides diffusion by rearranging the data within the block.

3. **MixColumns:**
    - Columns of the data block are mixed using a mathematical transformation in a finite field.
    - This step further ensures diffusion by spreading out the influence of each byte.

4. **AddRoundKey:**
    - The current data block is XORed with the round key derived from the original key.

Final Round

The final round is similar to the other rounds but omits the **MixColumns** step. This ensures the correct decryption process and maintains symmetry.

Key Expansion in AES

AES uses a **key schedule** to derive round keys from the original key. Key expansion ensures that each round has a unique key, which enhances security:

- The key schedule includes a series of rotations, substitutions (using the S-box), and XOR operations.
- The number of round keys generated depends on the key size (128, 192, or 256 bits).

Comparison with DES

Feature	DES	AES
Block Size	64 bits	128 bits
Key Size	56 bits (effective)	128, 192, 256 bits
Number of Rounds	16	10, 12, or 14
Security	Vulnerable	Highly secure
Efficiency	Slower	Faster

Other Symmetric Block Ciphers

1. **IDEA (International Data Encryption Algorithm):**
    - Developed by Xuejia Lai and James Massey.
    - Features a 64-bit block size and a 128-bit key.
    - Can be implemented in both hardware and software.

2. **Blowfish:**
    - Created by Bruce Schneier.
    - Block size: 64 bits.
    - Key size: Variable (32 to 448 bits).
    - Known for its speed and simplicity.
    -
3. **CAST-128:**
    - Designed by Carlisle Adams and Stafford Tavares.
    - Feistel cipher with 16 rounds.
    - Block size: 64 bits.
    - Key size: 128 bits.

4. **RC5 (Rivest Cipher 5):**
    - Designed by Ron Rivest.
    - Highly flexible with variable block sizes, key sizes, and rounds.

Conclusion

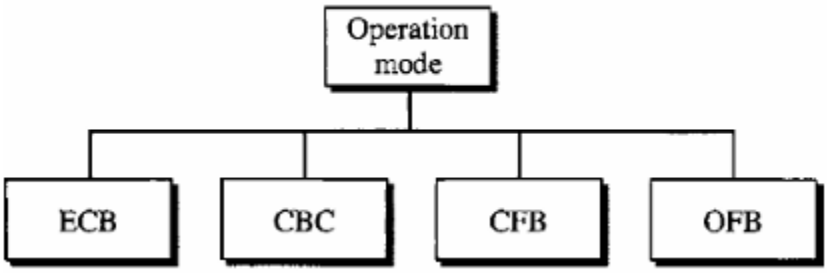
AES has become the standard for secure encryption worldwide due to its robustness, adaptability, and efficiency.

Its support for multiple key sizes and resistance to cryptanalysis makes it ideal for applications in industries ranging from finance to telecommunications.

While other symmetric block ciphers like IDEA, Blowfish, CAST-128, and RC5 have their uses, AES remains the gold standard for modern encryption systems.

MODE OF OPERATION

- A mode of operation is a technique that employs the modern block ciphers such as DES andAES.

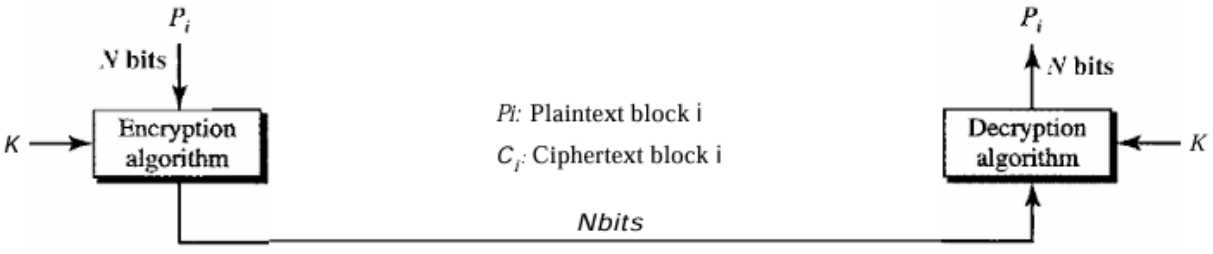


ECB - Electronic Code Book

- Purely block cipher technique.
- PLaintext is divided into blocks of N bits.
- Ciphertext = clocks of N bits.
- Value of N depends on type of cipher used.

CHARACTERISTIC TRAITS -

1. Because the key and the encryption/decryption algorithm are the same, equal blocks in the plaintext become equal blocks in the ciphertext. For example, if plaintext blocks 1, 5, and 9 are the same, ciphertext blocks I, 5, and 9 are also the same. This can be a security problem; the adversary can guess that the plaintext blocks are the same if the corresponding ciphertext blocks are the same.
2. If We Reorder the plaintext block, the ciphertext is also reordered.
3. Blocks are independent of each other. Each block is encrypted or decrypted inde pendently. A problem in encryption or decryption of a block does not affect other blocks.
4. An error in one block is not propagated to other blocks. If one or more bits are corrupted during transmission, it only affects the bits in the corresponding plaintext after decryption. Other plaintext blocks are not affected. This is a real advantage if the channel is not noise-free



CBC - CIPHER BLOCK CHAINING .

The **Cipher Block Chaining (CBC)** mode is a block cipher mode of operation designed to improve the security of encryption methods like the Electronic Codebook (ECB) mode. In CBC, each plaintext block is influenced by the encryption of the previous ciphertext block, providing better diffusion and making patterns in the plaintext harder to detect in the ciphertext.

How CBC Works

1. **Encryption:**
    - For the first block (i=1i = 1):
        - The plaintext block (P1P_1) is XORed with an **initialization vector (IV)** before encryption.
        - The resulting block is encrypted using the block cipher algorithm to produce the first ciphertext block (C1C_1).

    - C1=E(P1⊕IV)C_1
    - For subsequent blocks (i>1):

        - The plaintext block (P i) is XORed with the previous ciphertext block (Ci−1).

        - The result is encrypted using the block cipher to produce the current ciphertext block (C i).

    - Ci=E(Pi⊕C i−1) C_i
2. **Decryption:**

    - The process reverses encryption. Each ciphertext block is decrypted, and the previous ciphertext block (or IV for the first block) is XORed with the result to recover the plaintext:

    - For the first block (i=1):
      P1=D(C1) ⊕ IV P_1
    - For subsequent blocks (i>1):
      Pi=D(Ci) ⊕ C i−1 P_i

Initialization Vector (IV)

- **Purpose:** The IV introduces randomness into the encryption process to ensure that identical plaintext blocks encrypt to different ciphertext blocks.
- **Shared Information:** The IV must be known to both the sender and receiver. It is typically sent along with the ciphertext or agreed upon beforehand.
- **Security:** The IV should be unpredictable to prevent attacks that exploit repeated plaintext patterns.

Advantages of CBC

1. **Enhanced Security:**
    - By chaining the encryption of blocks, CBC prevents patterns in the plaintext from appearing in the ciphertext, a common weakness in ECB mode.
2. **Error Propagation Control:**
    - Errors in one block of ciphertext affect only two blocks during decryption (the current and subsequent block).
3. **Widely Supported:**
    - CBC is commonly implemented in protocols like TLS and IPsec.

Limitations of CBC

1. **Error Sensitivity:**
    - A single bit error in a ciphertext block will corrupt two plaintext blocks during decryption.

2. **Sequential Processing:**
    - CBC encryption and decryption are sequential processes, as each block depends on the previous one. This makes parallel processing difficult.

3. **IV Handling:**
    - Securely generating and transmitting the IV is critical. If the IV is predictable, it can compromise the security of the system.

CHARACTERISTIC TRAITS -

1. Even though the key and the encryption/decryption algorithm are the same, equal blocks in the plaintext do not become equal blocks in the ciphertext. For example, if plaintext blocks 1, 5, and 9 are the same, ciphertext blocks I, 5, and 9 will not be the same. An Adversary will not be able to guess from the ciphertext that two blocks are the same.
2. Blocks are dependent on each other. Each block is encrypted or decrypted data on a previous block. A problem in encryption or decryption of a block affects other blocks.
3. The error in one block is propagated to the other blocks. If one or more bits are corrupted during the transmission, it affects the bits in the next blocks of the plain text after decryption.

CFB - CIPHER FEEDBACK

The cipher feedback (CFB) mode was created for those situations in which we need to send or receive r bits of data, where r is a number different from the underlying block size of the encryption cipher used. The value of r can be 1, 4, 8, or any number of bits.

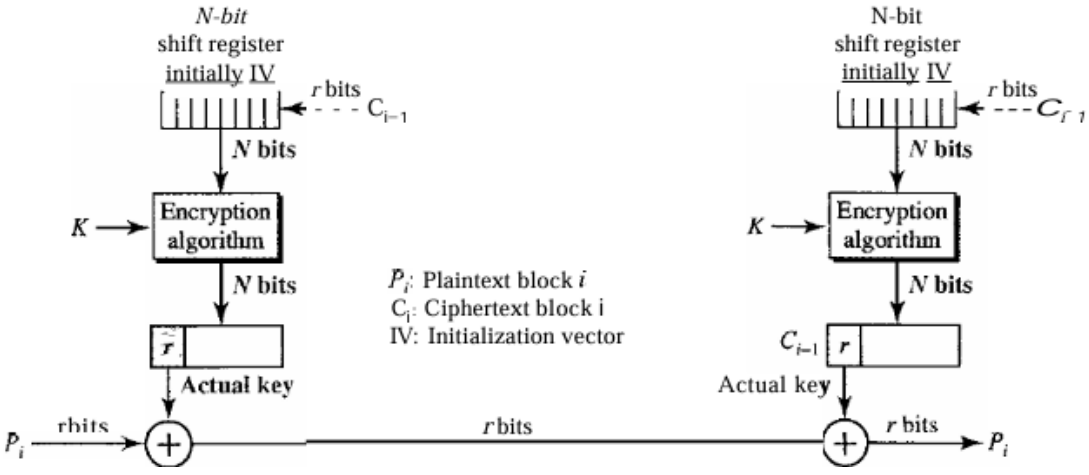
Since all block ciphers work on a block of data at a time, the problem is how to encrypt just r bits.

The solution is to let the cipher encrypt a block of bits and use only the first r bits as a new key (stream key) to encrypt the r bits of user data.

Figure 30.22 shows the configuration.

The following are some characteristics of the CFB mode:

1. If We change the IV from one encryption to another using the same plaintext, the ciphertext is different.
2. The ciphertext Ci depends on both Pi and the preceding ciphertext block.
3. Errors in one or more bits of the ciphertext block affect the next ciphertext blocks.

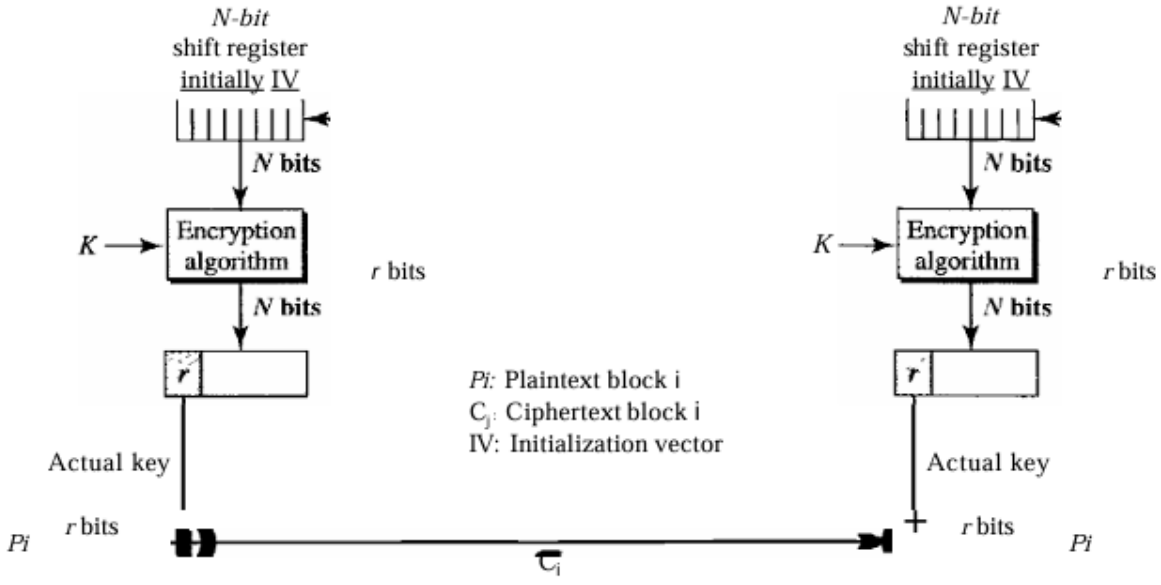


OFB - OUTPUT FEEDBACK

- The output feedback (OFB) mode is very similar to the CFB mode with one difference.
- Each bit in the ciphertext is independent of the previous bit or bits.
- This avoids error of propagation.
- as in CFB, both the sender and the receiver use the encryption algorithm. Note also that in OFB, block ciphers such as DES or AES can only be used to create the key stream.
- The feedback for creating the next bit stream comes from the previous bits of the key stream instead of the ciphertext. The ciphertext does not take part in creating the key stream.

CHARACTERISTIC TRAITS -

1. If we change the IV from one encryption to another using the same plaintext, the ciphertext will be different.
2. The ciphertext Ci depends on the plaintext Pi'
3. Errors in one or more bits of the ciphertext do not affect future ciphertext blocks.



ASYMMETRIC-KEY CRYPTOGRAPHY

RSA - RIVEST , SHAMIR , ADLEMAN

The **RSA algorithm** is one of the most widely used public-key cryptographic systems. It was named after its inventors—Ron Rivest, Adi Shamir, and Leonard Adleman. RSA uses a pair of keys: one for encryption (public key) and another for decryption (private key). The mathematical foundation of RSA relies on the difficulty of factoring large composite numbers into their prime components.

Key Generation in RSA

To create the public and private keys:

1. **Choose Two Large Prime Numbers:**
    - Select two large prime numbers, pp and qq.
    - Example: p=7, q=11. .

2. **Calculate n:**
    - Compute $n=p\times q$.
    - n is used as the modulus for both encryption and decryption.
    - Example: $n=7\times 11=77$.

3. **Calculate ϕ(n):**
    - Compute $\phi(n)=(p-1)\times(q-1)$, where ϕ(n) is the totient of n.
    - Example: $\phi(n)=(7-1)\times(11-1)=6\times 10=60$. .

4. **Choose Public Exponent (e):**
    - Select an integer ee such that $1 < e < \phi(n)$ and $\gcd(e, \phi(n))=1$
    - Example: e=13

5. **Calculate Private Exponent (d):**
    - Compute d such that $d \times e \equiv 1 \pmod{\phi(n)}$
    - d is the modular multiplicative inverse of e modulo ϕ(n)
    - Example: d=37, as $13\times 37\equiv 1 \pmod{60}$

6. **Publish and Keep Keys:**
    - Publish e and n as the **public key**.
    - Keep d and ϕ(n) as the **private key**.

Encryption in RSA

1. Convert the plaintext message (P) into a numerical representation.
    - For short messages, map characters to numbers (e.g., A = 0, B = 1, ..., Z = 25).
2. Compute the ciphertext (C) using the formula:
   $C=P^e \pmod n$
3. Send C to the recipient.

Example:

- Plaintext: P=5
- Public Key: e=13, n=77
- Ciphertext:
  $C=5^{13} \pmod{77}=26$

Decryption in RSA

1. The recipient uses their private key (d) to decrypt the ciphertext (C) back into plaintext (P) using the formula:
   $P=C^d \pmod n$
2. The result (P) is the original plaintext message.

Example:

- Ciphertext: C=26
- Private Key: d=37, n=77
- Plaintext:
  $P=26^{37} \pmod{77}=5$

Practical Considerations

1. **Block Size Restriction:**
    - RSA requires that the plaintext (PP) be less than nn.
    - For long messages, divide the plaintext into blocks where each block is smaller than nn.

2. **Encoding and Decoding:**
    - Convert plaintext to numeric form and back after decryption.
    - Example: Use ASCII or a custom mapping scheme.

Advantages of RSA

1. **Public Key Cryptography:**
    - Secure communication without the need to exchange secret keys.

2. **Versatile Applications:**
    - RSA is used for encrypting short messages, digital signatures, key exchange, and authentication.

3. **High Security:**
    - The security of RSA relies on the difficulty of factoring large composite numbers.

Limitations of RSA

1. **Efficiency:**
    - RSA is computationally expensive for large messages.
    - Modern systems use RSA to encrypt symmetric keys rather than the actual data.

2. **Key Size:**
    - Security increases with larger key sizes, but this also increases computational costs.
    - Recommended key sizes are 2048 bits or more for secure applications.

3. **Quantum Threat:**
    - RSA may become vulnerable to quantum computers due to Shor's algorithm.

Applications of RSA

1. **Digital Signatures:**
    - RSA is used to verify the authenticity of messages and documents.

2. **Key Exchange:**
    - RSA is used in protocols like SSL/TLS for securely exchanging symmetric keys.

3. **Authentication:**
    - RSA ensures the identity of the sender in secure communication.

4. **Encryption of Short Messages:**
    - RSA is often used for encrypting short messages or small data, such as session keys in hybrid cryptosystems.

RSA remains a cornerstone of modern cryptography due to its robustness and wide applicability, though it's often combined with symmetric cryptography for efficiency.

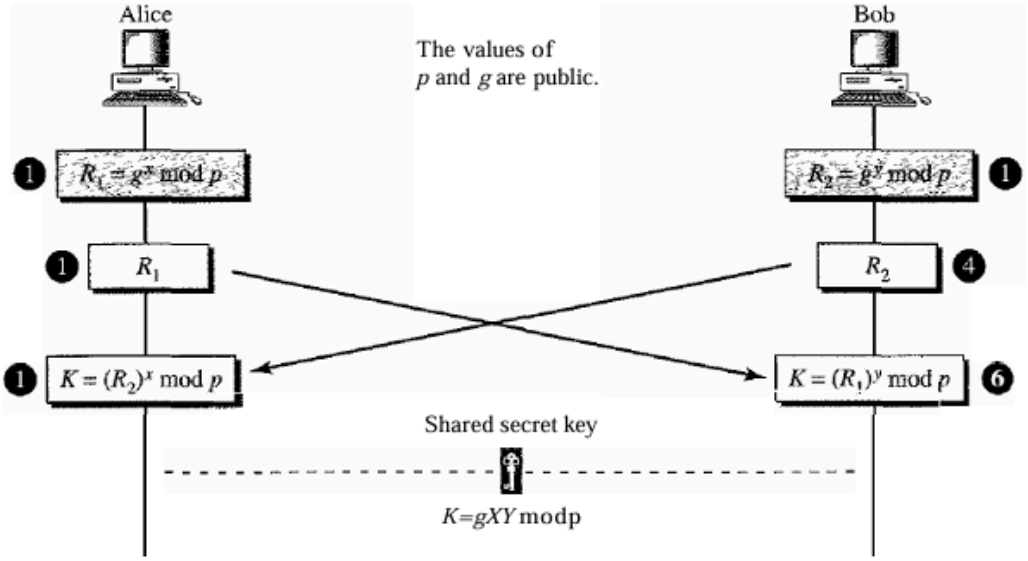
Diffie-Hellman Key Exchange

The **Diffie-Hellman Key Exchange** is a cryptographic protocol primarily designed for securely exchanging cryptographic keys over a public channel. Unlike RSA, which encrypts messages, Diffie-Hellman focuses on generating a shared symmetric key for secure communication. This allows two parties to establish a shared secret key without meeting in person or relying on a pre-shared key.

How Diffie-Hellman Works

The Diffie-Hellman protocol relies on two fundamental mathematical concepts:

1. **Modulo Arithmetic**: Calculations are done modulo a large prime number pp.
2. **Exponentiation**: Large powers of numbers are computed within the modulo system.



Steps in Diffie-Hellman Protocol

1. **Public Values**:
    - Alice and Bob agree on a large prime number pp and a generator gg (a primitive root modulo pp).

- These values are public and can be shared openly over the Internet.

2. **Private Values**:
   - Alice chooses a private number x (known only to her).
   - Bob chooses a private number y(known only to him).

3. **Calculate Public Keys**:
   - Alice calculates her public value R1=g^x mod p.
   - Bob calculates his public value R2=g^y mod p.

4. **Exchange Public Keys**:
   - Alice sends R1 to Bob.
   - Bob sends R2 to Alice.

5. **Calculate Shared Secret Key**:

   - Alice uses Bob's public value R2 and her private number x to calculate the shared key:
     K=(R2)^x mod p
   - Bob uses Alice's public value R1 and his private number y to calculate the same shared key:
     K=(R1)^y mod p

   - Since (R2^)x=(g^y)^x=(g^x)^y=(R1)^y mod p , both Alice and Bob arrive at the same symmetric key K.

Example

Let’s walk through an example using small values to demonstrate the protocol.

1. **Public Values**:
   - p=23 (a prime number).
   - g=7 (a generator modulo pp).

2. **Private Values**:
   - Alice chooses x=3x = 3 (private key).
   - Bob chooses y=6y = 6 (private key).

3. **Calculate Public Keys**:
    - Alice calculates R1=g^x mod p=7^3 mod 23=21.
    - Bob calculates R2=g^y mod p=7^6 mod 23=4

4. **Exchange Public Keys**:
    - Alice sends R1 = 21 to Bob.
    - Bob sends R2=4to Alice.

5. **Calculate Shared Secret Key**:

    - Alice computes K=(R2)^x mod p=4^3 mod 23=18
    - Bob computes K=(R1)^y mod p=21^6 mod 23=18

Both parties arrive at the same shared key K=18K = 18.

Security of Diffie-Hellman

The security of Diffie-Hellman depends on the difficulty of the **Discrete Logarithm Problem**:

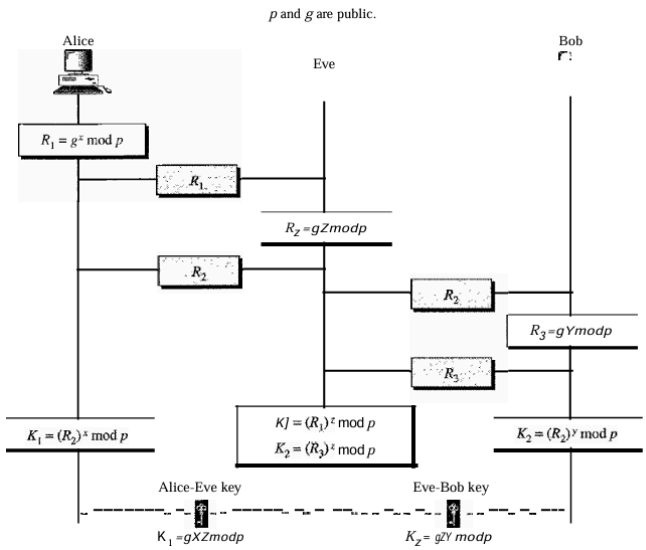
- Even if an attacker intercepts pp, gg, R1R_1, and R2R_2, it is computationally infeasible to deduce xx or yy due to the large size of pp (typically 1024 bits or more in practical implementations).

Weakness: Man-in-the-Middle Attack

While Diffie-Hellman is secure against brute force, it is vulnerable to a **Man-in-the-Middle Attack** if authentication is not used.

Scenario:

1. Eve intercepts R1 from Alice and replaces it with her own RE=g^z mod pp, where z is a number chosen by Eve.
2. Eve sends RE to Bob.



3. Bob responds with R2=g^y mod p, which Eve intercepts and replaces with another R^E=g^z mod p before sending it to Alice.

This creates two shared keys:

- K1 between Alice and Eve.
- K2 between Eve and Bob.

Eve can now decrypt messages from Alice, alter them, and re-encrypt them for Bob (or vice versa).

Solution: Authentication

To prevent man-in-the-middle attacks:

- Use **digital signatures** to authenticate the public values exchanged between Alice and Bob.
- Ensure that R1R_1 and R2R_2 are tied to their respective parties through verifiable credentials.

Applications of Diffie-Hellman

1. **Secure Key Exchange**:
    - Commonly used in protocols like TLS (Transport Layer Security) to securely establish a symmetric session key for communication.

2. **Ephemeral Key Exchange**:
    - In **Ephemeral Diffie-Hellman (DHE)**, new keys are generated for each session, ensuring forward secrecy.

3. **Cryptographic Protocols**:
    - Often combined with RSA or digital certificates for authenticated and secure communication.

By enabling two parties to securely generate a symmetric key over an insecure channel, Diffie-Hellman remains a cornerstone of modern cryptography.