Cryptography

Block Cipher and Data Encryption Standard

M S Vilkhu

14 Oct 2024 C1/C3

05 Oct 2024 C1/C3/C5

18 Oct 2024 C5

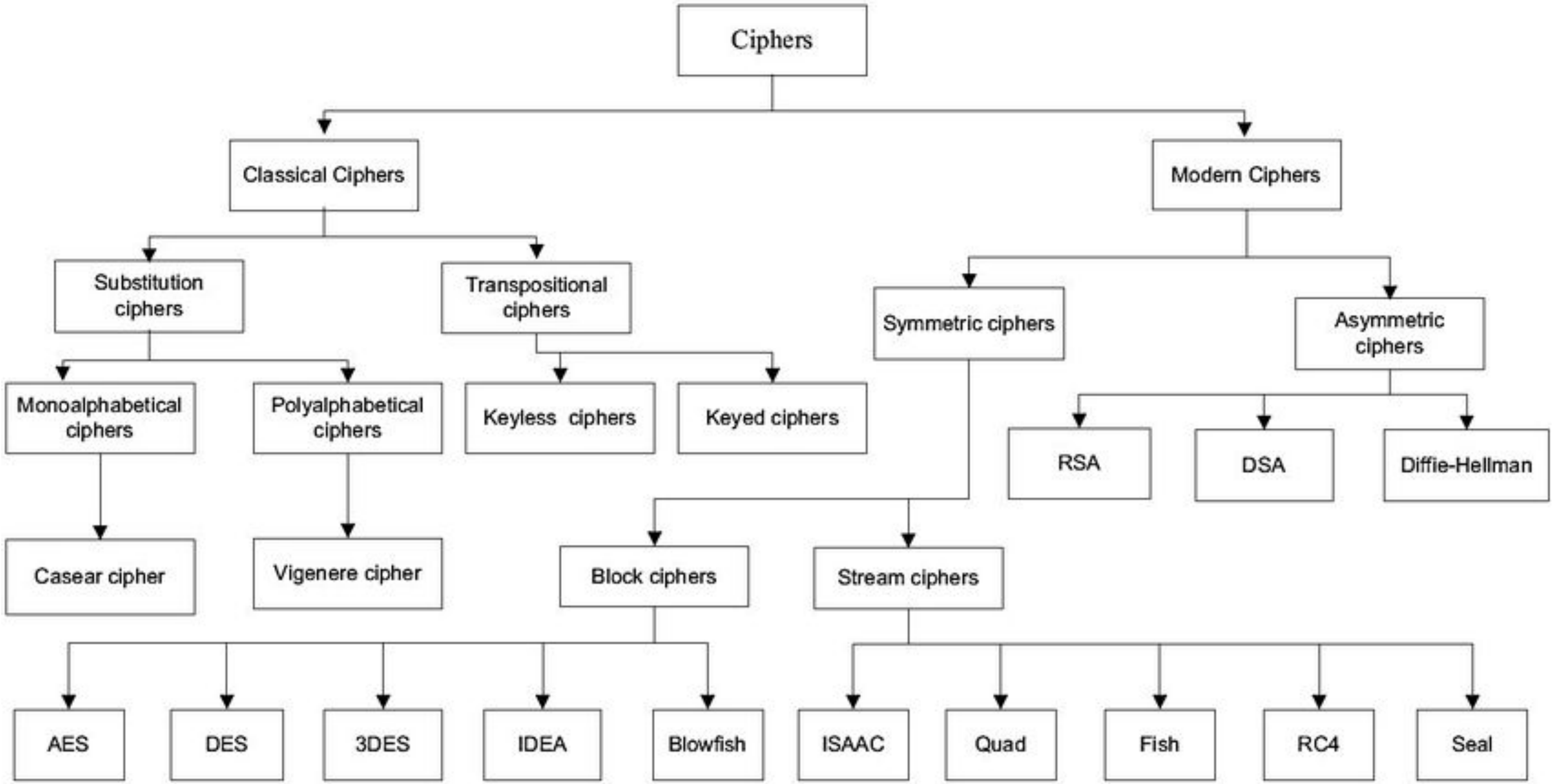
--07-Sep-24 (C1)

10 Oct 2024 C1/C3/c5

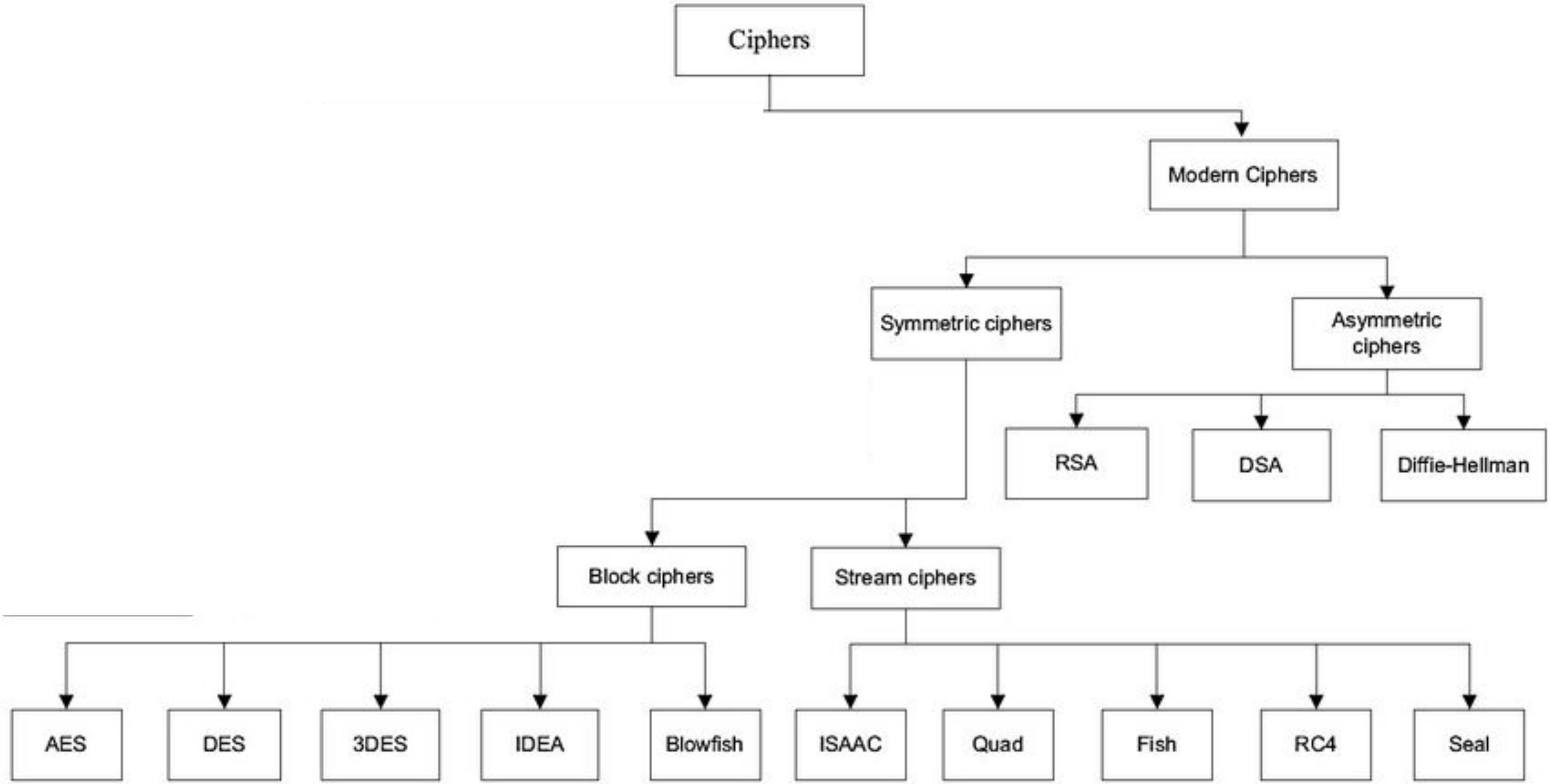
Learning Objectives

- Understand the distinction between **stream ciphers and block ciphers**.
- Present an overview of the **Feistel cipher** and explain how decryption is the inverse of encryption.
- Present an overview of **Data Encryption Standard (DES).**
- Explain the concept of the **avalanche effect.**
- Discuss the cryptographic **strength of DES.**
- Summarize the principal block cipher design principles.

Classification



Classification



Introduction

- Focus on **Modern cipher system**
- Focus on **Data Encryption Standard** (DES)
- DES **most important** though replaced by Advanced Encryption Standard (AES)
- Detailed study of DES provides an **understanding principle used** in other symmetric cipher

- begins with a discussion of the **general principles of symmetric block ciphers**, which are the principal type of symmetric ciphers studied in cryptography course.
- The other form of symmetric ciphers, **stream ciphers**, later.

Introduction

- Next, we cover full DES.
- Compared to public-key ciphers, such as RSA,
  - the **structure of DES and most symmetric ciphers is very complex**
- cannot be explained as easily as RSA and similar algorithms.

Traditional block cipher structure

- Several important **symmetric block encryption algorithms** in current use are based on a structure referred to as a **Feistel block cipher** [FEIS73].
- For that reason, it is important to **examine the design principles of the Feistel cipher**.
- We begin with a **comparison of stream ciphers and block ciphers**.

Stream Ciphers and Block Ciphers

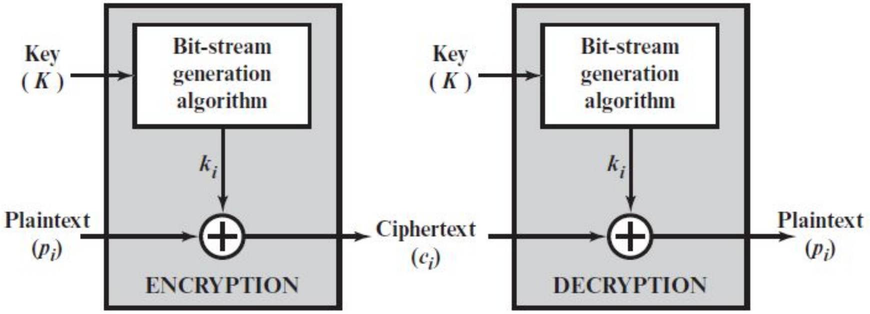
Stream Ciphers

- **A stream cipher** is one that **encrypts** a digital data stream **one bit or one byte** at a time.
- Examples of classical stream ciphers are the
  - **autokeyed Vigenère cipher and the Vernam cipher**.
  - In the ideal case, a **one-time pad** version of the Vernam cipher would be used in which the keystream (ki) is as long as the plaintext bit stream (pi).
- If the **cryptographic keystream is random**, then this cipher is **unbreakable** by any means other than acquiring the keystream.
- the **keystream** must be provided to **both users in advance** via some independent and secure channel.
- This introduces **insurmountable logistical problems** if the intended **data traffic is very** large.

Stream Ciphers and Block Ciphers

Stream Ciphers

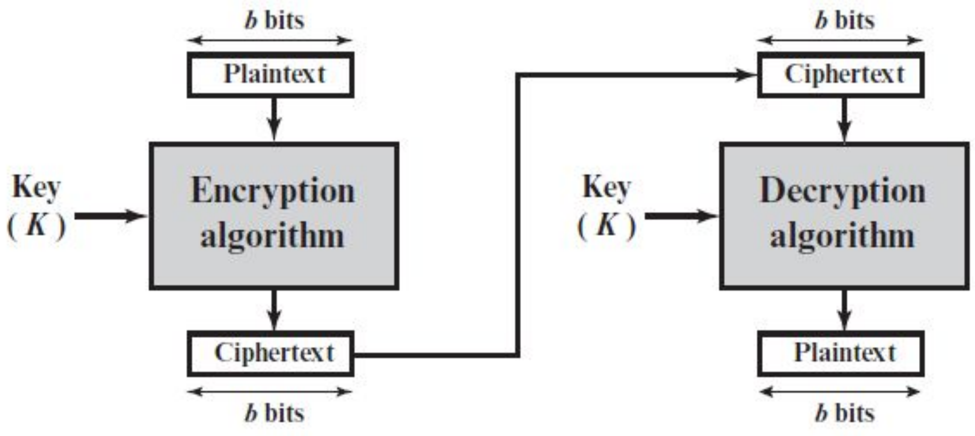
- for practical reasons, the **bit-stream generator must be implemented** as an **algorithmic procedure**, so that the **cryptographic bit stream** can be produced by both users.
- In this approach (Figure), the bit-stream generator is a **key-controlled algorithm** and must produce a bit stream that is cryptographically strong.
- That is, it must be **computationally impractical to predict future portions** of the bit stream based on previous portions of the bit stream.
- The two users need only share the **generating key**, and each can **produce the keystream.**



Stream Ciphers and Block Ciphers

Block cipher

- **A block cipher** is one in which a **block of plaintext** is **treated as a whole** and used to produce a **ciphertext block of equal length**.
- Typically, a **block size** of 64 or 128 bits is used.
- the **two users share** a symmetric encryption key (Figure)
- a block cipher can be used to achieve the same effect as a stream cipher.



Stream Ciphers and Block Ciphers

Block cipher

- Far more effort has **gone into analyzing block ciphers**.
- In general, they seem **applicable to a broader range of applications** than **stream ciphers**.
- The vast majority of **network-based symmetric cryptographic** applications make **use of block ciphers.**

- primarily **focus on block ciphers.**

Block Ciphers

Block cipher – Reversible or nonsingular transformation

- A block cipher **operates** on a **plaintext block of n bits** to produce a **ciphertext block of n bits.**
- There are **2^n possible different plaintext blocks** and, for the **encryption to be reversible (i.e., for decryption to be possible**), each **must produce a unique ciphertext block**.
- Such a transformation is called **reversible, or nonsingular.**
- The following examples illustrate nonsingular and singular transformations for n = 2.

we limit ourselves to **reversible mappings** - the number of different transformations

Reversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	00
11	01

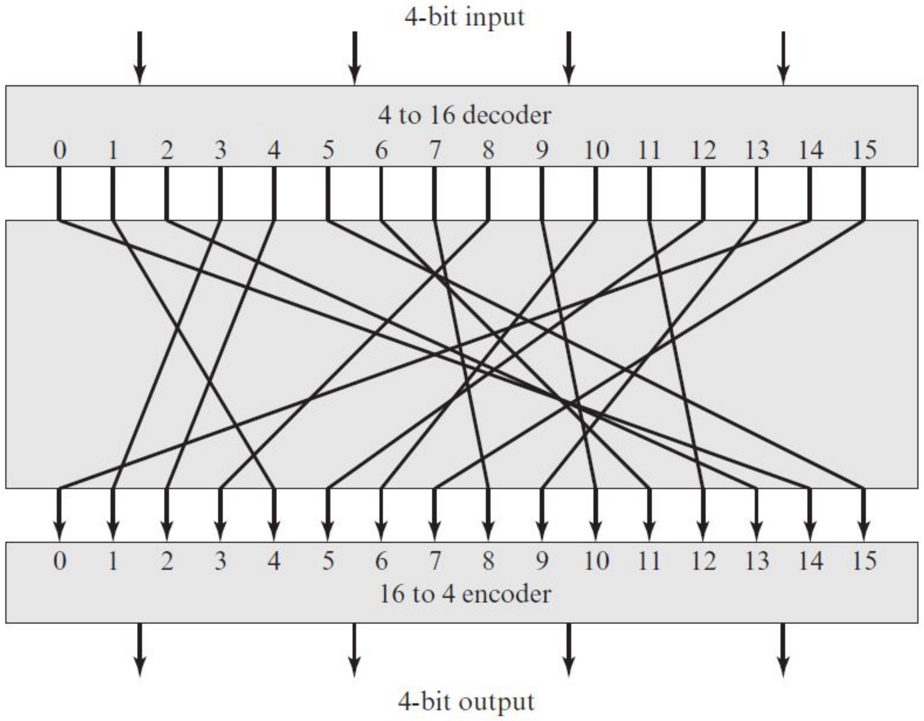
Irreversible Mapping

Plaintext	Ciphertext
00	11
01	10
10	01
11	01

ciphertext of 01 could have been produced by one of two plaintext blocks.

Block Ciphers : Ideal Block Cipher

- **What is ideal block cipher?**
- Figure illustrates the **logic of a general substitution cipher for n = 4.**
- A **4-bit input** produces **one of 16 possible input states**,
- which is **mapped by the substitution** cipher into a unique one of **16 possible output states**, each of which is represented by **4 ciphertext bits**.



Block Ciphers : Ideal Block Cipher

- **What is ideal block cipher?**
- The **encryption and decryption** mappings defined by a tabulation
- This is the **most general form of block cipher** and can be **used to define any reversible mapping between plaintext and ciphertext.**
- **Feistel** refers to this as the **ideal block cipher**, because it allows for the **maximum number of possible encryption mappings** from the plaintext block

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Block Ciphers : Ideal Block Cipher

- **What is ideal block cipher?**
- **An arbitrary reversible substitution cipher (the ideal block cipher)** for a large block **size is not practical**, - from an implementation and performance point of view.
- For such a transformation, the mapping itself constitutes the key.
- plaintext to ciphertext for n = 4. The mapping can be defined by the entries in the second column, which show the **value of the ciphertext for each plaintext block**
- the **key that determines** the specific mapping from among all possible mappings.

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Block Ciphers : Ideal Block Cipher

- **Ideal block cipher**
- In this case, using this straightforward method of defining the key, the **required key length** is
  - **(4 bits) * (16 rows) = 64 bits**. **[(4 bits) * (2^4 rows)]**
- In general, for an **n-bit ideal block cipher**, the length of the **key defined** in this fashion is
  - **$n * 2^n$ bits**.
- For a **64-bit block,** which is a **desirable length to thwart statistical attacks**, the required key length is
  - $64 * 2^{64} = 2^{70} \approx 10^{21}$ bits.
- considering these difficulties, **Feistel points** out that what is **needed is an approximation** to the **ideal block cipher system for large n**, built up out of components that are easily realizable

Cryptography

Block Cipher and Data Encryption Standard

M S Vilkhu

-18 Oct 2024 C5

14 Oct 2024 C1/C3

05 Oct 2024 C1/C3/C5

-19 Oct 2024 C1/C3/c5

--07 Sep 24 (C1)

Block cipher principles

- Block cipher principles
- **most symmetric block ciphers** are based on a **Feistel Cipher Structure** needed since must be **able to decrypt ciphertext to recover messages** efficiently.
- block ciphers look like an **extremely large substitution** would **need table of 264 entries for a 64-bit block**
- Instead create from **smaller building blocks**

- using **idea of a product cipher** in **1949 Claude Shannon** introduced idea **of**
  - **substitution-permutation (S-P) networks** called **modern substitution-transposition product cipher**
- this form the basis of modern block ciphers

Block cipher principles

- **Feistel Cipher Structure**
- Feistel cipher is a design model designed to **create different block ciphers**, **such as DES.**
- The model uses **substitution and permutation alternately**.

- **Substitution replaces** plain text elements with ciphertext.
- **Permutation** changes the **order** of the plain text elements rather than being replaced by another element as done with substitution.

- The **Feistel structure** is based on the **Shannon structure** proposed in 1945, demonstrating the **confusion and diffusion** implementation processes.

- **Confusion** produces a complex relationship between the **ciphertext and encryption key**, which is done by using a substitution algorithm.
- **diffusion** creates a complex relationship between **plain text and cipher text** by using a permutation algorithm.

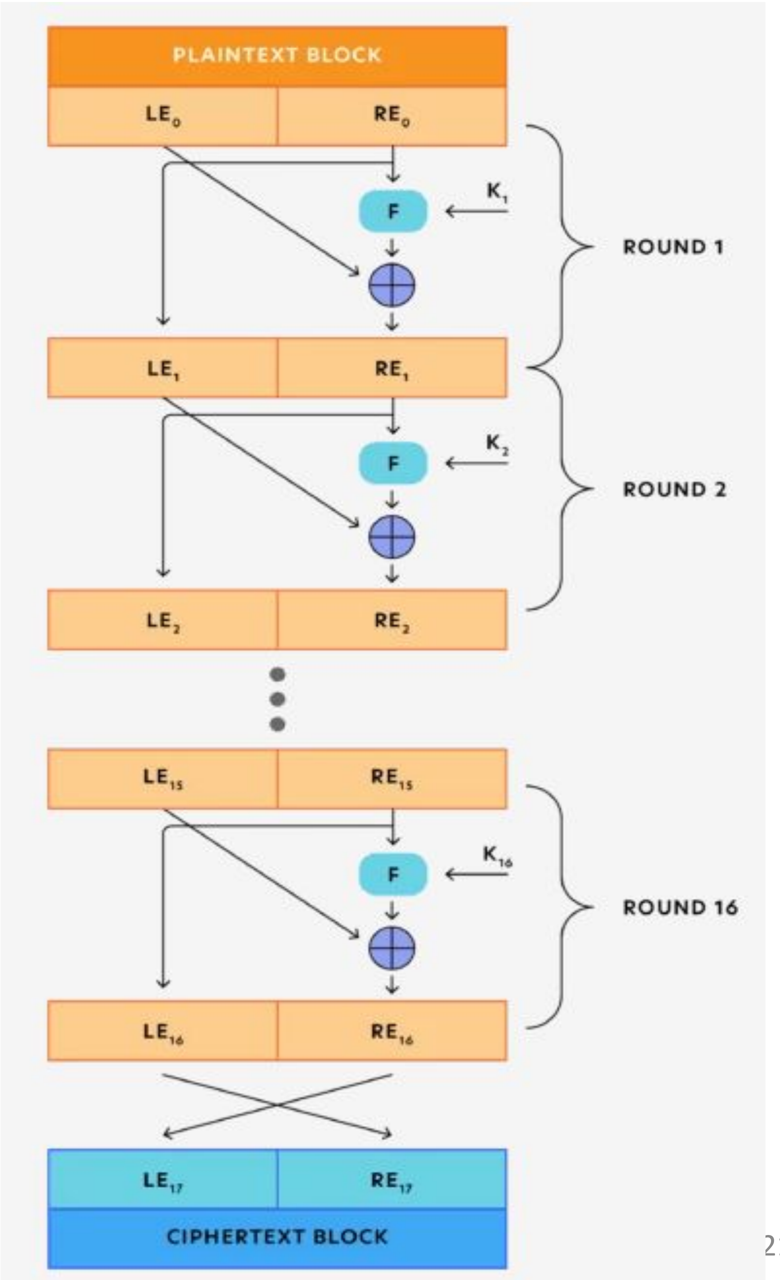
Feistel Cipher

- **Feistel Cipher Structure**
- Feistel Cipher Encryption Example
- The Feistel cipher encryption process involves **numerous rounds of processing** plain text.
- **Each round** includes the **substitution** step and then the **permutation** step.

- Check out the following example describing the encryption structure used for this design model.

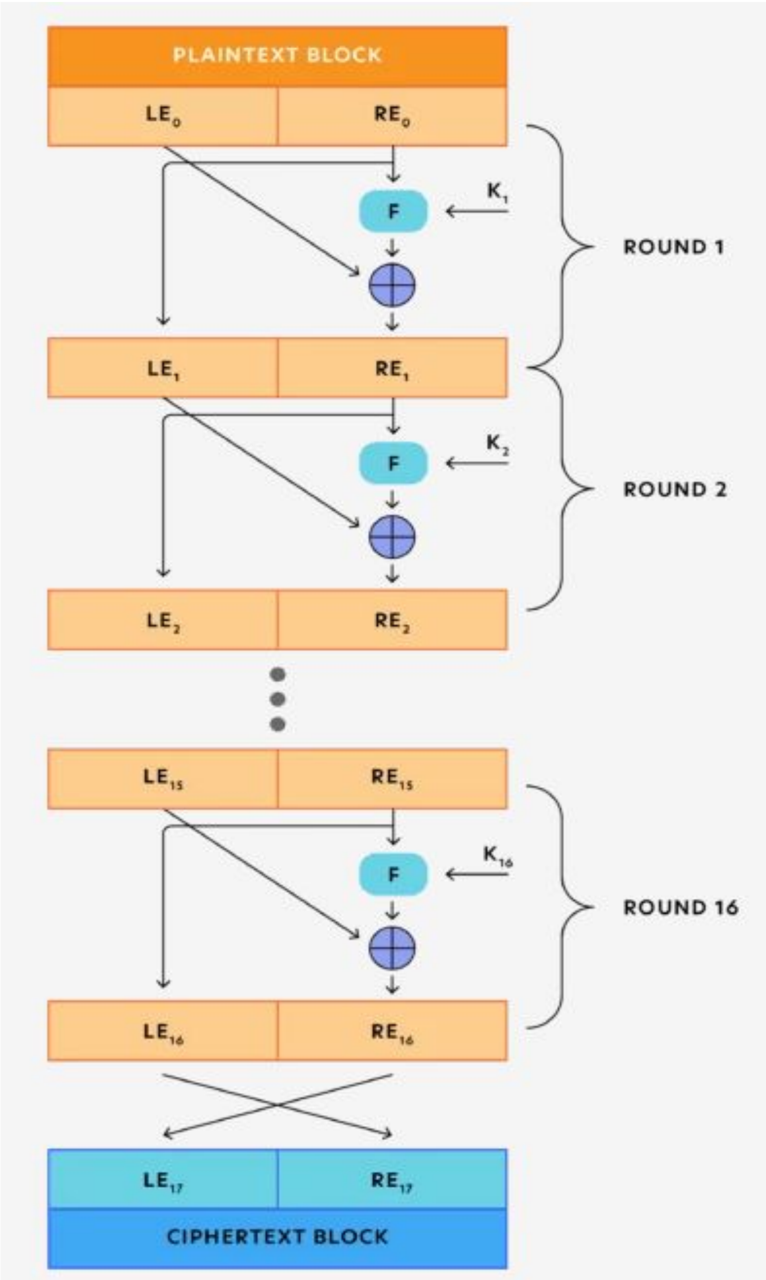
Feistel Cipher

- **Feistel Cipher Structure**
- **Step 1** – The **first step involves** the plain text being divided into **blocks of a fixed size**,
- with only **one block being processed at a time**.
- The encryption algorithm input consists of a plain text block and a key K.
- **Step 2 –**
- The left half of the plain text block will be represented as LE0, and the right half of the block will be RE0.
- Both halves of the plain text block (LE0 and RE0) will go through numerous rounds of processing plain text to produce the ciphertext block.



Feistel Cipher

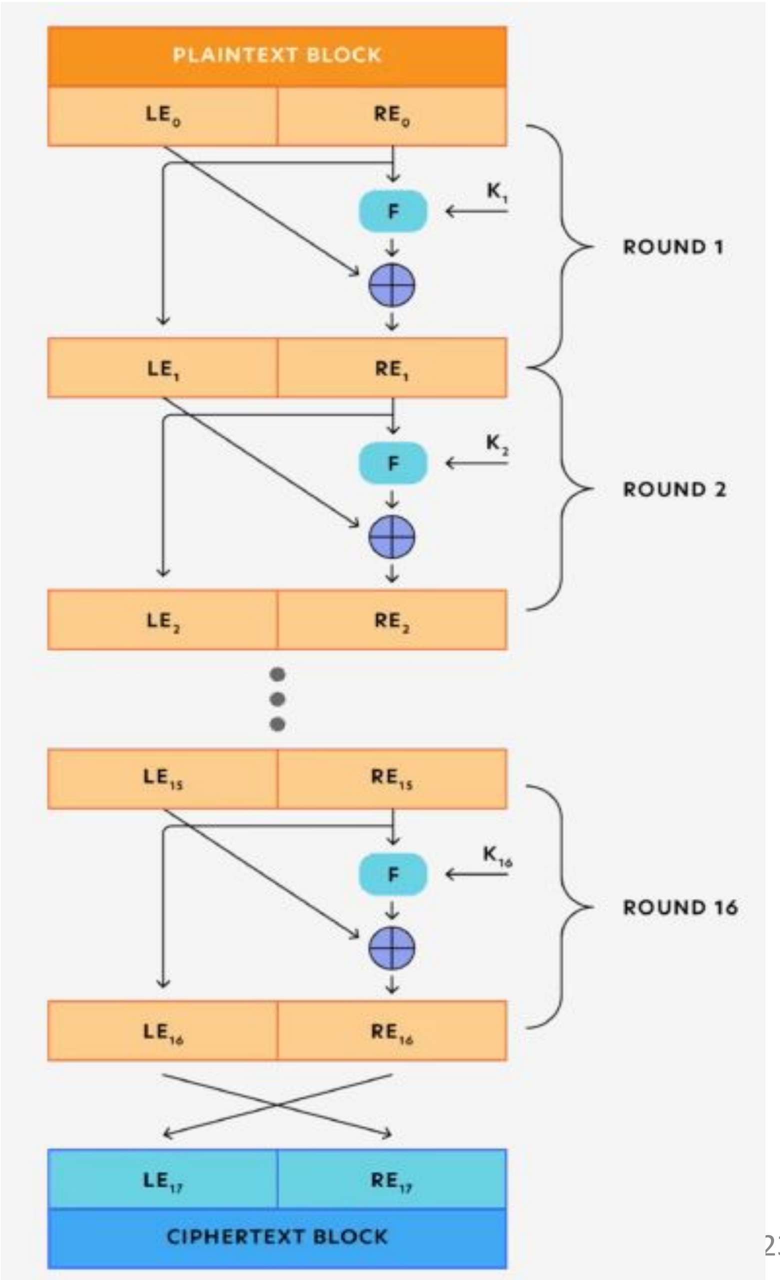
- **Feistel Cipher Structure**
- For every round, the **encryption function** is applied on the **right half REi** of the plain text block plus the key Ki.
- The function results are then **XORed** with the left half LEj.
- XOR is a logical operator used in cryptography that **compares two input bits and produces one output bit.**
- The XOR function results become the new right half for the next round RE i+1.
- The previous right half REi becomes the new left half LEi+1 for the next round.



Feistel Cipher

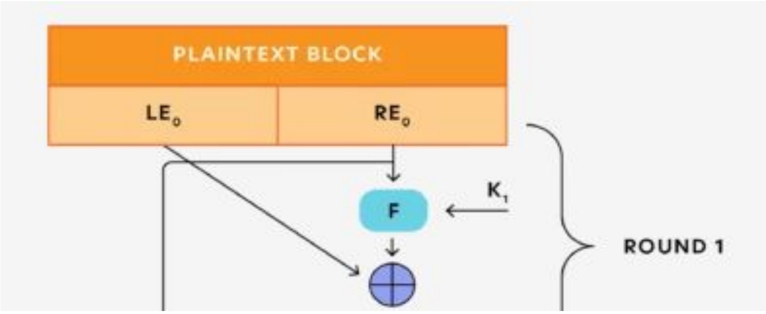
- **Feistel Cipher Structure**
- **substitution function** is implemented by using the **round function** to the **right half** of the plain text block
- The result of this function is **XORed** by using the left half of the block.
- **permutation** function is used by **switching the two halves**

- Feistel cipher model **implements** the **substitution** and **permutation** steps alternately,

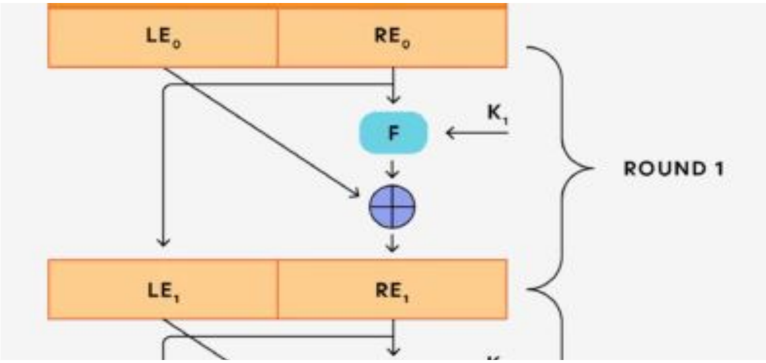


Feistel Cipher

- **Feistel Cipher Structure**
- For every round, the **encryption function (F)** is applied on the right **half REi** of the **plain text block plus the key Ki.**
- The function results are then **XORed with the left half LEj**.

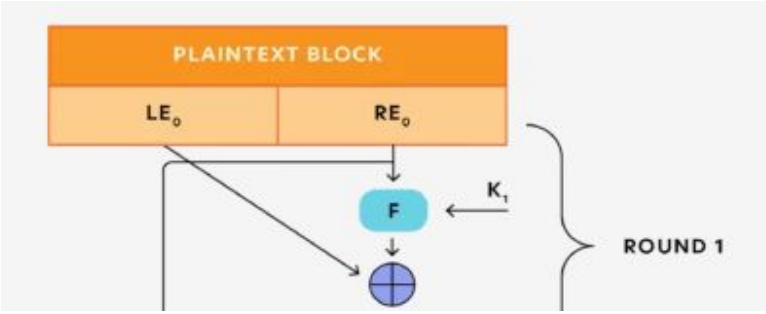


- The **XOR function results become** the **new right half** for the next round REi+1.
- The previous **right half REi becomes** the new **left half LEi+1** for the next round.

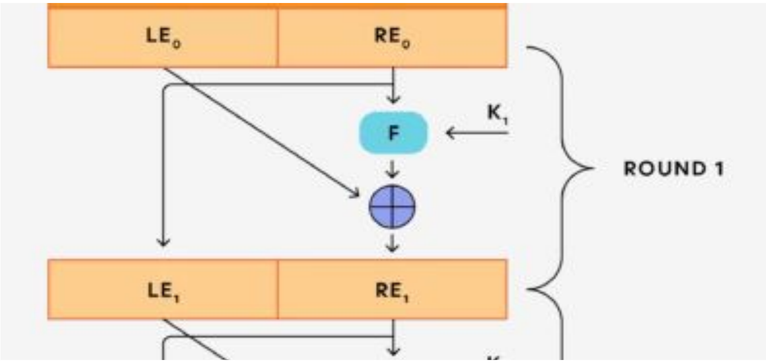


Feistel Cipher

- **Feistel Cipher Structure**



- Every round will **execute the same function**,
- **substitution function** is implemented by using the round **function to the right half** of the plain text block. The result of this function is XORed by using the left half of the block.
- Then, a **permutation function** is used by **switching the two halves**.



- The permutation results are provided for the next round.
- This is how the Feistel cipher model implements the **substitution and permutation steps alternately**, similar to the Shannon structure.

Feistel Cipher

- **Feistel Cipher Structure**
- Feistel cipher design features that are considered when using block ciphers:
- **Block size**
- **Easy analysis**
- **Key size**
- **The number of rounds**
- **Round function**
- **Subkey generation function**
- **Quick software encryption and decryption**

Feistel Cipher

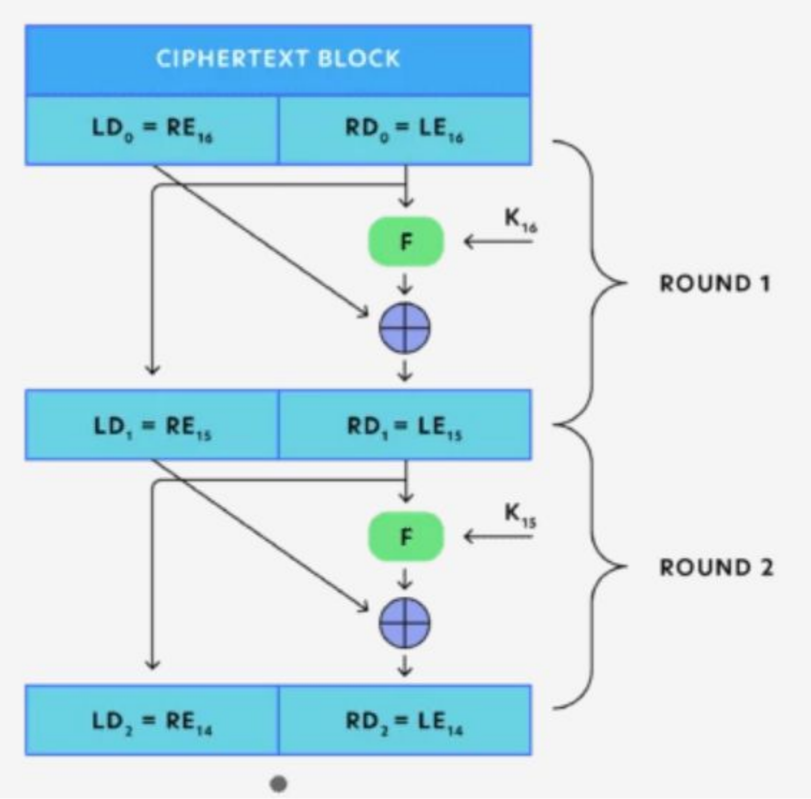
- **Feistel Cipher Structure**
- Feistel cipher design features that are considered when using block ciphers:
- **Block size –**
  - Block ciphers - **more secure** when the **block size is larger**.
  - Though, **larger block** sizes **reduce the execution speed** for the encryption and decryption process.
  - Typically, block ciphers have a block size of 64-bits, but modern blocks like AES (Advanced Encryption Standard) are 128-bits.
- **Easy analysis –** Block ciphers should be easy to analyze, which can help identify and address any **cryptanalytic weaknesses** to create more robust algorithms.
- **Key size –**
- Like the block size, **larger key sizes** are considered **more secure** at the cost of potentially **slowing down the time it takes to finish the encryption and decryption process**.

Feistel Cipher

- **Feistel Cipher Structure**
- Feistel cipher design features that are considered when using block ciphers:
- **The number of rounds –**
  - The number of rounds can also impact the security of a block cipher.
  - While **more rounds increase security**, the cipher is **more complex to decrypt**.
  - Thus, the **number of rounds depends** on a **business's desired level** of data protection.
- **Round function –** A **complex round function** helps boost the block cipher's security.
- **Subkey generation function –** The **more complex a subkey generation** function is, the more difficult it is for expert cryptanalysts to decrypt the cipher.
- **Quick software encryption and decryption –** It's helpful to use a software application that can help produce faster execution speeds for block ciphers.

Feistel Cipher

- **Feistel Cipher Structure**
- **Feistel Cipher Decryption Example**
- It may be surprising that the Feistel cipher model uses the same algorithm for encryption and decryption.
- There are a couple of key rules to consider during the decryption process:
- As shown in the figure the cipher text block contains two **halves, the left (LD0) and the right (RD0).**
- Like the encryption algorithm, **the round function is executed on the right half** of the cipher block **with the key K16.**
- The **function's result is XORed with the left half** of the cipher text block. The XOR function's **output becomes the new right half (RD1),**
- while **RD0 switches with LD0** for the next round.
- every round **uses the same function**, and once the fixed number of rounds is executed, the plain text block is achieved



Block cipher principles

- using idea of a product cipher in **1949 Claude Shannon** introduced idea **of substitution-permutation (S-P) networks** called **modern substitution-transposition product cipher** these form the basis of modern block ciphers
- S-P networks are based on the **two primitive cryptographic operations** we have seen before:
- **substitution (S-box)**
- **permutation (P-box)**

- provide **confusion and diffusion** of message

- **diffusion** – **dissipates statistical structure** of **plaintext** over bulk of ciphertext
- **confusion** – makes **relationship between ciphertext and key** as **complex as possible**

Diffusion and Confusion

Diffusion and confusion

- The terms diffusion and confusion were introduced by Claude Shannon to capture the **two basic building blocks for any cryptographic** system.
- Shannon's concern **was to thwart** **cryptanalysis** based on **statistical analysis**.
- The reasoning is
- Assume the attacker has some **knowledge of the statistical characteristics** of the plaintext. For example, the **frequency distribution** of the various letters may be known. Or there may be **words or phrases likely to appear** in the message (probable words).
- If these **statistics are in any way reflected in the ciphertext**, the cryptanalyst may be able to **deduce the encryption key**, part of the key, or at least a set of keys likely to contain the exact key.
- In what Shannon refers to as a **strongly ideal cipher**, **all statistics of the ciphertext are independent of the particular key used**.
- **This is impractical**

Block cipher principles

Diffusion and confusion

- recourse to ideal systems, Shannon suggests **two methods for frustrating statistical** cryptanalysis: **diffusion and confusion.**

- **In diffusion**, the statistical **structure of the plaintext is dissipated** into long-range statistics of the ciphertext.
- This is achieved by having each **plaintext digit affect the value of many ciphertext digits**;

- Every block cipher involves a transformation of a block of plaintext into a block of ciphertext, where the **transformation depends on the key**.

- The **mechanism of diffusion** seeks to make the **statistical relationship between the plaintext and ciphertext as complex as possible** in order to **thwart attempts to deduce the key.**

Block cipher principles

Diffusion and confusion

- On the other hand, **confusion** seeks to make the **relationship between the statistics of the ciphertext** and the **value of the encryption key** as **complex as possible**, again to **thwart attempts to discover** the key.
- Thus, even if the attacker can get some handle on the statistics of the ciphertext, the way in which the key was used to produce that ciphertext is so complex as to **make it difficult to deduce the key.**
- This is achieved by the **use of a complex substitution algorithm**. In contrast, a **simple linear substitution** function would add little confusion.

Important

so successful are **diffusion and confusion** in capturing the essence of the desired attributes of a block cipher that they have become the **cornerstone of modern block cipher design**.

Block cipher principles

- **Confusion and diffusion** are two fundamental principles in cryptography that enhance the security of encryption algorithms. They were introduced by Claude Shannon and are essential for creating strong cryptographic systems.
- Confusion
- **Definition**:
  - **relationship** between the **plaintext** (the original message) and the **ciphertext** (the encrypted message) **as complex as possible**.
  - The goal is to **obscure how the plaintext maps to the ciphertext**.
- **Purpose**:
  - the algorithm ensures that an **attacker cannot easily determine the key used for encryption just by analyzing the ciphertext**.
  - For **example**, if a specific **change in the plaintext** results in a **predictable change in the ciphertext**, it reduces security.
  - Confusion **ensures that small changes** in the plaintext **lead to unpredictable changes** in the ciphertext.

Block cipher principles

- **Confusion and diffusion** are two fundamental principles in cryptography that enhance the security of encryption algorithms. They were introduced by Claude Shannon and are essential for creating strong cryptographic systems.
- Confusion
- **Implementation**:
- various operations such as
  - **substitution** (replacing elements of the plaintext with others),
  - **complex key scheduling**, and
  - **non-linear transformations**.

Block cipher principles

- **Confusion and diffusion** are two fundamental principles in cryptography that enhance the security of encryption algorithms. They were introduced by Claude Shannon and are essential for creating strong cryptographic systems.
- **Diffusion**
- **Definition**:
  - spreading out the **influence of a single plaintext bit** over **many bits of ciphertext**. This means that **changing one bit of the plaintext should affect many bits of the ciphertext**.
- **Purpose**:
  - ensure that even a **small change in the plaintext (or the key) results in significant and unpredictable changes in the ciphertext,** making it harder for attackers to **deduce patterns**.
  - For example, if an attacker changes a single bit in the plaintext and only a small number of bits in the ciphertext change, the algorithm is less secure.
- **Implementation**:
  - Diffusion is often **implemented through permutation** operations and mixing functions that mix bits of the input in various ways across rounds of encryption.

XOR Operation

XOR (exclusive OR) is a fundamental operation in cryptography that **enhances security in several ways:**

1. Confusion and Diffusion

Confusion: XOR helps obscure the relationship between the plaintext (original data) and ciphertext (encrypted data). By combining plaintext with a key, the output is less predictable.

Diffusion: Changes in the input (plaintext or key) lead to significant changes in the output, making patterns harder to detect.

2. Key Addition

In symmetric key algorithms (like the One-Time Pad), XOR is used to combine the plaintext with a secret key. The resulting ciphertext looks random if the key is truly random and used only once:

$\text{Ciphertext} = \text{Plaintext} \oplus \text{Key}$

XOR Operation

XOR (exclusive OR) is a fundamental operation in cryptography that enhances security in several ways:

3. **Reversibility**

XOR operations are easily reversible. To decrypt, you simply XOR the ciphertext with the same key:

Plaintext=Ciphertext⊕Key\text{Plaintext}

This property is essential for both encryption and decryption processes.

4. **Bit-Level Manipulation**

XOR operates at **the bit level**, allowing for **efficient and fast computations**. This efficiency is crucial in performance-sensitive applications like secure communications and data encryption.

5. **Resistance to Linear Cryptanalysis**

Because XOR produces **outputs that are not linearly related to inputs**,

it adds a **layer of resistance** against certain types of attacks, particularly linear cryptanalysis.

XOR Operation

XOR is a simple yet **powerful** tool in cryptography that enhances security by providing

confusion,

diffusion, and

efficient bit manipulation.

Its **effectiveness**, especially when **combined with other cryptographic** techniques, contributes to the overall strength of encryption systems.

XOR operation on a word

Performing an XOR operation on a word (string of characters) with a key involves applying the XOR operation to each character of the word using the corresponding character in the key.

If the key is shorter than the word, it is typically repeated or cycled to match the length of the word.

Example

Let's say we have the word "HELLO" and the key "KEY".

Convert Characters to Binary:

Each character can be represented in binary using its ASCII value.

H = 72 = 01001000

E = 69 = 01000101

L = 76 = 01001100

L = 76 = 01001100

O = 79 = 01001111

K = 75 = 01001011

E = 69 = 01000101

Y = 89 = 01011001

XOR operation on a word

Extend the Key: To match the length of "HELLO", we can repeat the key:KEYKE

XOR Each Character: Now perform the XOR operation bit by bit.

Character	ASCII	Binary		Key	Key ASCII	Key Binary		XOR Result	Result ASCII	Result Char
H	72	01001000		K	75	01001011		00000011	3	\x03
E	69	01000101		E	69	01000101		00000000	0	\x00
L	76	01001100		Y	89	01011001		00010101	21	\x15
L	76	01001100		K	75	01001011		00000111	7	\x07
O	79	01001111		E	69	01000101		00001010	10	\x0A

Final Result:

The resulting string after the XOR operation is a combination of the characters corresponding to the XOR results:

\x03\x00\x15\x07\x0A

XOR operation on a word

Summary

The XOR operation on a word with a key produces a transformed word where each character is the result of the XOR operation between the corresponding characters of the word and the key.

This operation is a fundamental technique used in various cryptographic applications!

XOR the output with KEY will provide the original plaintext

Linear cryptanalysis

Linear cryptanalysis is a method of **cryptanalysis that exploits linear approximations** to describe the behavior of a cipher. It was first introduced by Mitsuru Matsui in the early 1990s and is particularly **applicable to symmetric-key block ciphers**.

Key Concepts

Linear Approximations:

Linear cryptanalysis relies on **finding linear relationships** between the plaintext, ciphertext, and key bits. For a cipher, a linear approximation might look something

like: $P1 \oplus P2 \oplus K1 = C1 \oplus C2P$

Here, P represents plaintext bits, C represents ciphertext bits, and K represents key bits. The goal is to **find such relations that hold with a probability better than random guessing**.

Bias:

If the **linear approximation** holds true with a **probability significantly greater than 0.5**, it **indicates a bias in the cipher that can be exploited**.

The **strength** of this bias helps **determine how effective the linear attack** can be.

Linear cryptanalysis

Data Collection:

To **successfully perform linear cryptanalysis**, an attacker **collects a large number of plaintext-ciphertext pairs.** The more pairs available, the better the chances of finding a bias that can be exploited.

Key Recovery:

Once biases are identified, the attacker can **use statistical methods to recover key bits**, often focusing on the bits that influence the linear approximation significantly.

Applications

Linear cryptanalysis is particularly effective against ciphers that **do not have strong diffusion properties**.

It has been notably applied to ciphers like DES (Data Encryption Standard), where certain linear approximations were found that allowed for key recovery with fewer data than brute force attacks.

Linear cryptanalysis

Defense Strategies

To defend against linear cryptanalysis, modern ciphers often incorporate:

- **Nonlinear operations**: This increases the complexity of any linear relationship.
- **Strong diffusion**: Ensuring that **changes in the plaintext significantly affect the ciphertext across many bits.**
- **Complex key schedules**: Making it **difficult to derive key bits** based on the input-output relationships.

Overall, linear cryptanalysis highlights the importance of designing cryptographic algorithms with a **focus on resisting statistical attacks.**

The DES Timeline

- The main events that took place during the lifetime of DES are as follows:
  - 1973-74: The DES algorithm is **developed by IBM**.
  - 1974: The **NSA adopts DES as a government-wide standard** for encryption.
  - 1976: DES is approved in the United States as a federal standard.
  - 1983, 1988, 1993, and 1999: Federal approval is reaffirmed by the NSA.
  - 1999: The more secure **triple DES algorithm** is recommended by NIST.
  - 2005: NIST withdraws affirmation of DES. However, Triple DES is given confirmation for sensitive government information.

- Meanwhile, in 2002, the more secure **advanced encryption standard** (AES) was becoming the algorithm of choice.

DES

- The data encryption standard (DES) was **endorsed by the National Security Agency** (NSA) from **1974 to 2002**.
- For **around 30 years**, the DES algorithm **ruled the cryptography** world as the go-to encryption algorithm.
- But what is DES encryption exactly, and what were the reasons behind its discontinuation?

- DES is based on an earlier cipher by **cryptographer Horst Feistel, called Lucifer**.

- Developed in the 1970s, **Lucifer** was one of the **earliest block ciphers**.

> The **Lucifer cipher** was a precursor to the **DES** algorithm. While **Lucifer** itself wasn't based on a Feistel structure, its design influenced the development of **DES**, which IBM later submitted as the U.S. encryption standard.

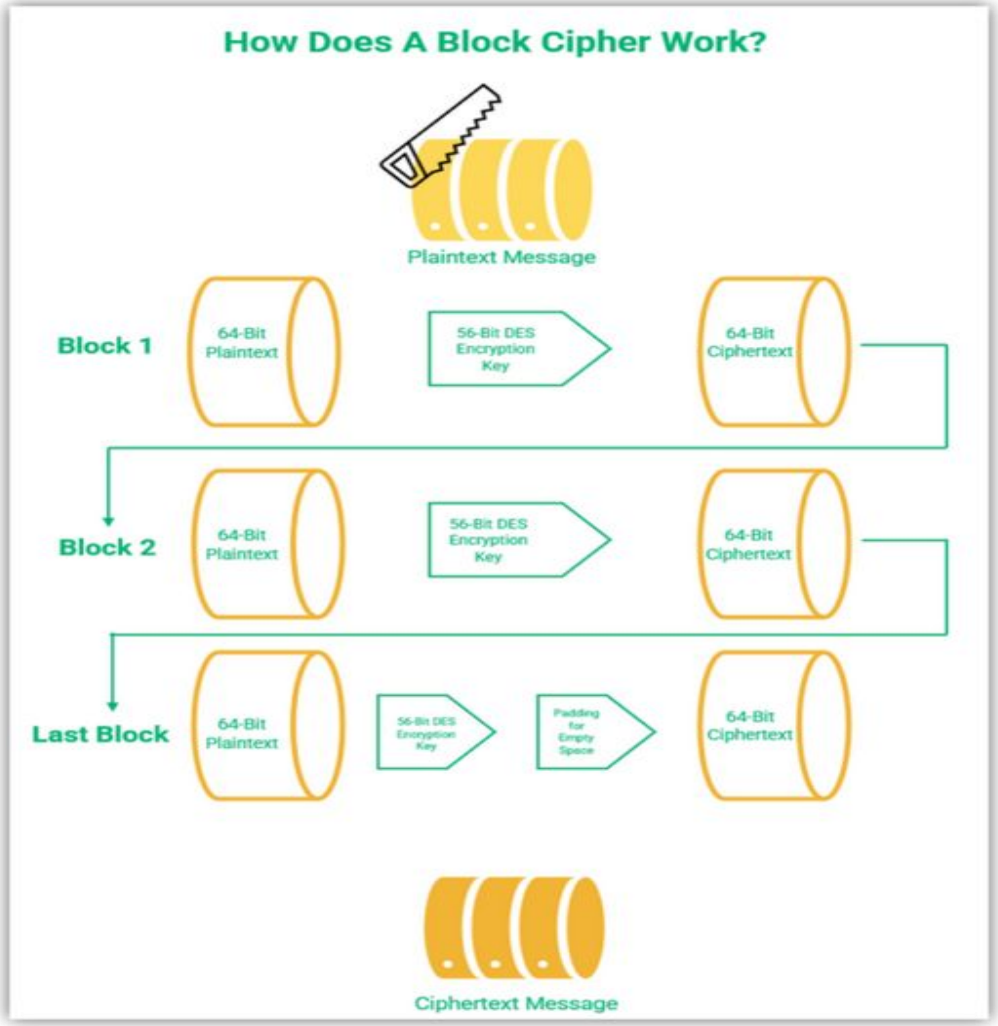
DES

- Why Is It **Important to Learn** About DES?

- You might be wondering why you should learn about DES if it’s already obsolete.
- Well, although it’s true that the DES algorithm is **no longer used for security purposes**, it is **still the basis for other security algorithms**.
- your **knowledge of DES** will help you **understand how other encryption algorithms work**.
- So, learning what DES encryption is and how it works will also **benefit your cybersecurity knowledge and understanding of cryptography as well**.

DES

- How DES works?
- look at how DES as a **block cipher works by breaking** your input down **into 64-bit blocks** and encrypting each block using a 56-bit key + 8 parity bits.



DES

- How DES works?
- look at how **DES as a block cipher works** by breaking your input down into 64-bit blocks and encrypting each block **using a 56-bit key** + **8 parity bits**.
- The **basis of DES is bits**, that is, binary numbers — i.e., 0s and 1s. Hexadecimal, or base 16 numbers, are made up of four bits.
- To encrypt a message, the data encryption standard:
    - Creates **blocks of 16 hexadecimal numbers** (equal to 64 bits) using an encryption key.
    - **Encrypts 64 bits of plaintext** (blocks) and **returns 64 bits of ciphertext**.
    - **Performs 16 rounds of processing** using **Feistel function**
    - **Fortifies the encryption** with **additional initial and final permutations**

DES

- How DES works?
- **Key**
- The **initial length** of the key is **64 bits**.
- every **eighth bit is dropped**, effectively making it a **56-bit key**.
- **Before being dropped**, these eight bits, **known as parity bits**, are used to **check the two versions of the message** and **detect errors in the code**.

- So, with DES a message is divided into blocks of 64 bits.

DES

- How DES works?
- **Padding**
- So, with DES a message is divided into blocks of 64 bits.
- One **problem** with this is that **not all messages have a length exactly** divisible by 64, so the last block might be smaller than 64 bits.
- This means that the **last part of the message** has to be **padded** with **extra bits to fill the space**. There are **different methods of padding** the messages.
- One method is to **use 0s at the end of the message** to fill the gap and the numbers are then removed upon decryption.

DES Encryption is a Six-Step Process

this block cipher method of encryption is, let's quickly break down how DES encryption works:

1. The **message** is divided into **64-bit blocks**.
2. An **initial permutation** is carried out on the plain text blocks.
3. Permuted blocks are divided **into two halves**, each of which is **32 bits – left plain text** (**LPT) and right plain text (RPT**).
4. Both LPT and RPT go **through 16 rounds of encryption**.
5. **Each round** of encryption **has five steps:**

   Step 1. Key transformation

   Step 2. Expansion permutation

   Step 3. S-Box permutation

   Step 4. P-Box permutation

   Step 5. ExclusiveOR (XOR) and swap

DES Encryption is a Six-Step Process

Each round of encryption **has five steps:**

Step 1. Key transformation —

Key transformation is a process wherein **16 different subkeys measuring 48-bits** each are **derived from the main key to encrypt plaintext**. The **key schedule** is used to derive **these keys**.

Step 2. **Expansion permutation** — A **half-block of 32-bits** is **expanded to 48 bits** using **expansion permutation**.

It adds **adjacent bits** from each **side of the block to the 32-bits** of the block to create a 48-bit block.

DES Encryption Is a Six-Step Process

Each round of encryption **has five steps:**

Step 3. **S-Box permutation** — A **substitution box permutation**, or S-box, is the **only non-linear component** **in the DES algorithm**.

It provides **additional security to the cipher**.

After the **block is mixed with the subkey**, it is divided **into eight 6-bit parts**.

The **S-box process** uses a **lookup table to convert the eight 6-bit parts into 4-bit output** each, resulting in **32-bit output in total**

Step 4. **P-Box permutation** — The **32-bit output** from the S-box permutation is **rearranged according to the P-box permutation.**

The design of the P-box permutation **ensures that the output of each S-box is spread across** **four different S-boxes** for the next round of encryption.

DES Encryption Is a Six-Step Process

Each round of encryption **has five steps:**

Step 5. ExclusiveOR (XOR) and swap — XOR is a mathematical function that compares two sets of bits that can be either 1s or 0s. If the bits from both sets match, the XOR output is 0. On the other hand, if they don't match, the output is 1.

This bit-wise comparison **results in stronger encryption**

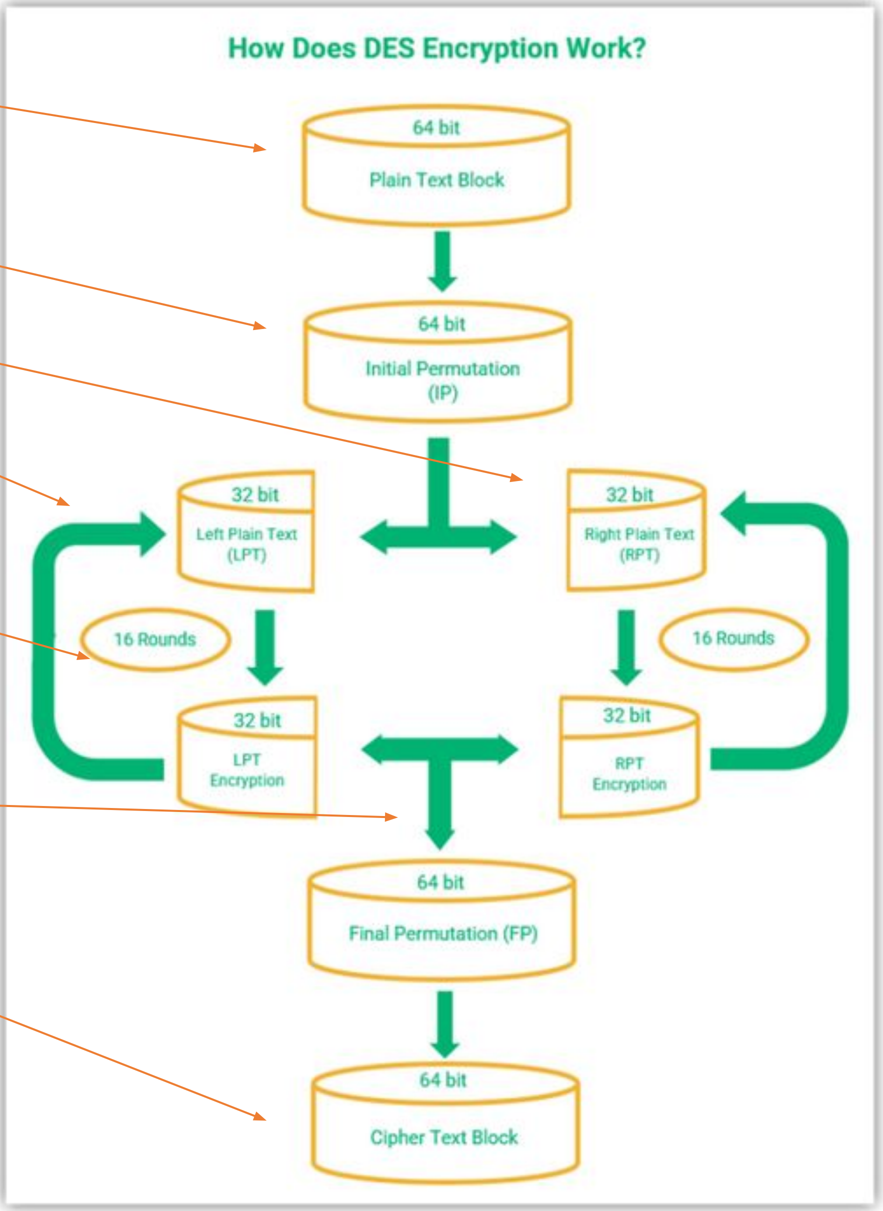
DES Encryption Is a Six-Step Process

this block cipher method of encryption is, let's quickly break down how DES encryption works:

1. The **message** is divided into **64-bit blocks**.
2. An initial permutation is carried out on the plain text blocks.
3. Permuted blocks are divided **into two halves**, each of which is **32 bits** – left plain text (**LPT) and right plain text (RPT**).
4. **Both LPT and RPT** go **through 16 rounds of encryption.**
5. Each round of encryption **has five steps:** (explained above)
6. LPT and RPT are **combined.**
7. The **final permutation** is performed on the combined LPT and RPT, resulting in the **final ciphertext**.

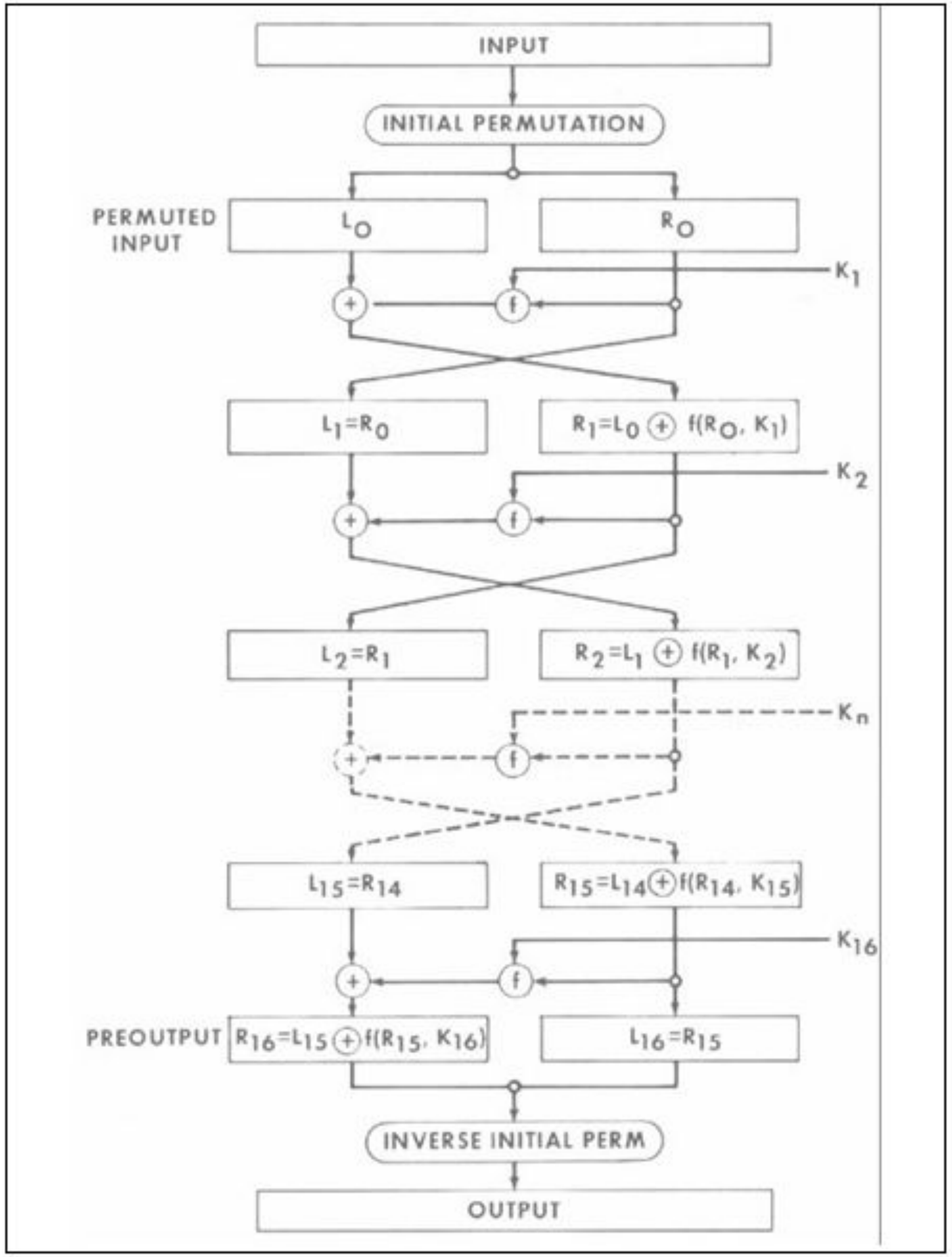
DES Encryption Is a Six-Step Process

1. The **message** is divided into **64-bit blocks**.
2. An initial permutation is carried out on the plain text blocks.
3. Permuted blocks are divided **into two halves,** each of which is **32 bits** – left plain text **(LPT) and right plain text (RPT)**.
4. Both LPT and RPT go **through 16 rounds of encryption**.
5. Each round of encryption **has five steps:** (explained earlier)
6. LPT and RPT are **combined.**
7. The **final permutation** is performed on the combined LPT and RPT, resulting in the **final ciphertext**.



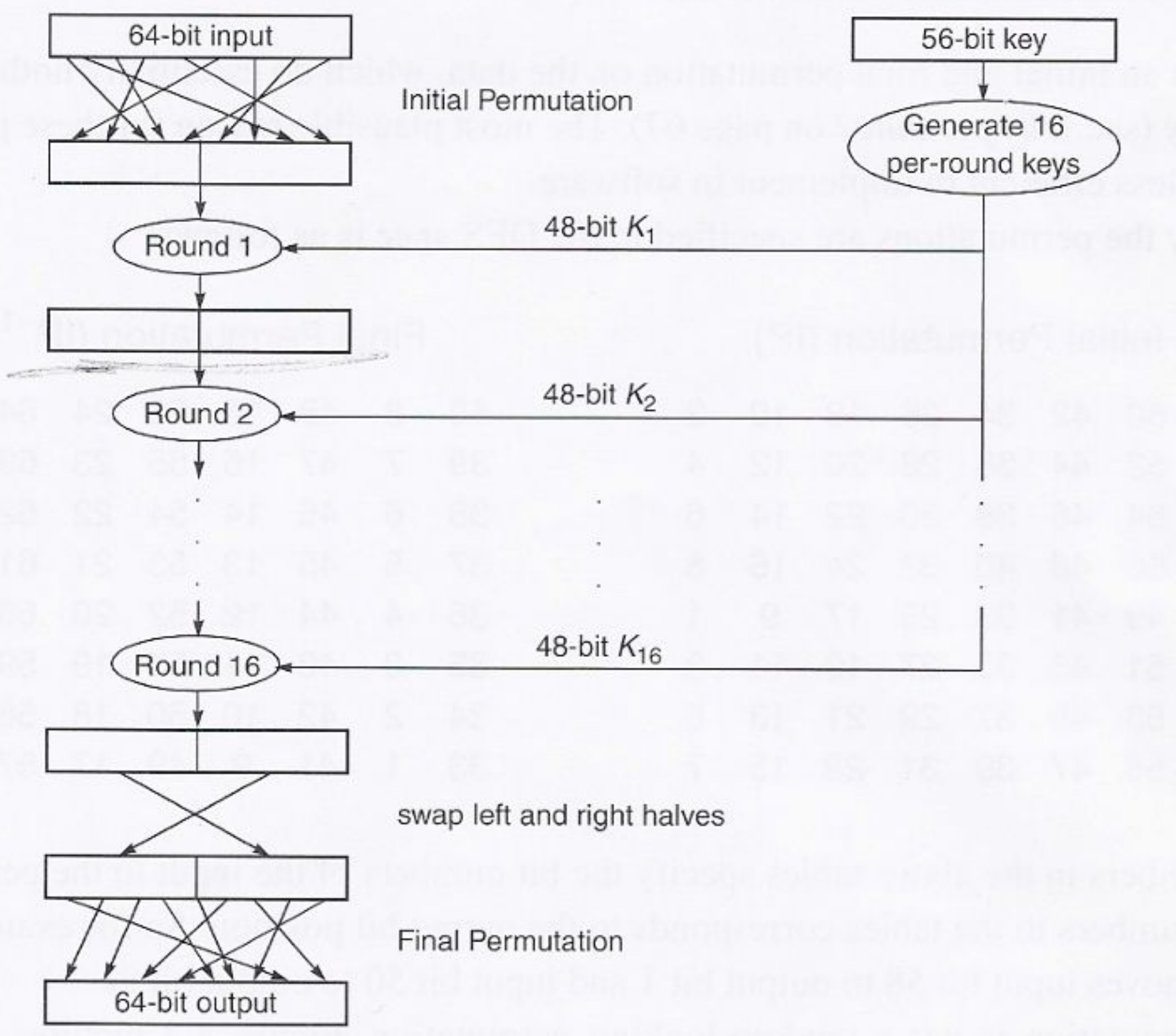
DES

Encryption



DES

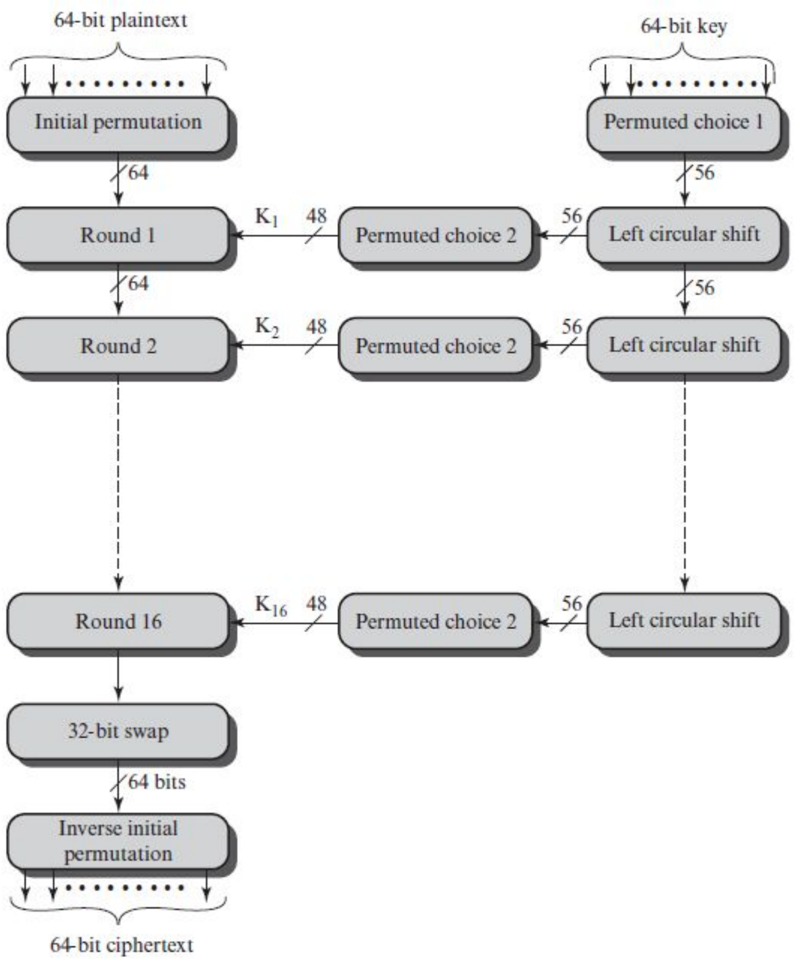
Encryption



DES

Encryption

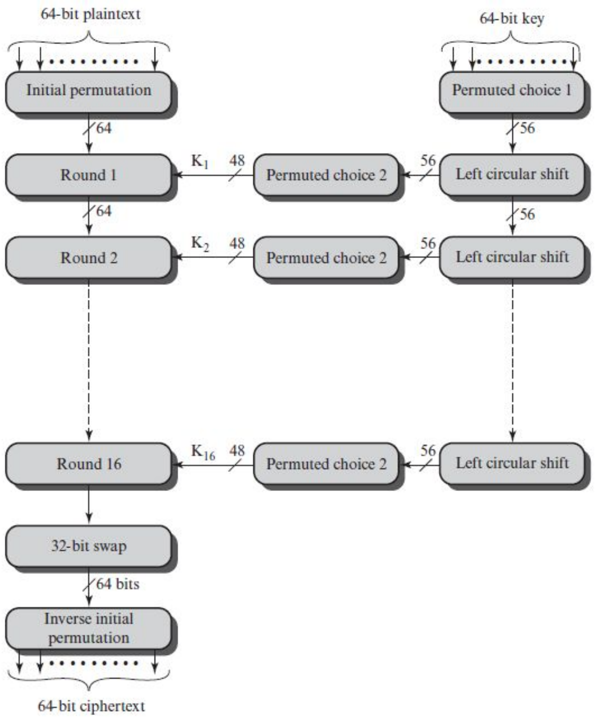
- The overall scheme for DES encryption is illustrated in Figure .
- As with any encryption scheme, there **are two inputs to the encryption function**: the plaintext to be encrypted and the key.
- In this case, the plaintext must be 64 bits in length and the key is 56 bits in length.
- Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.
- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*



DES

Encryption

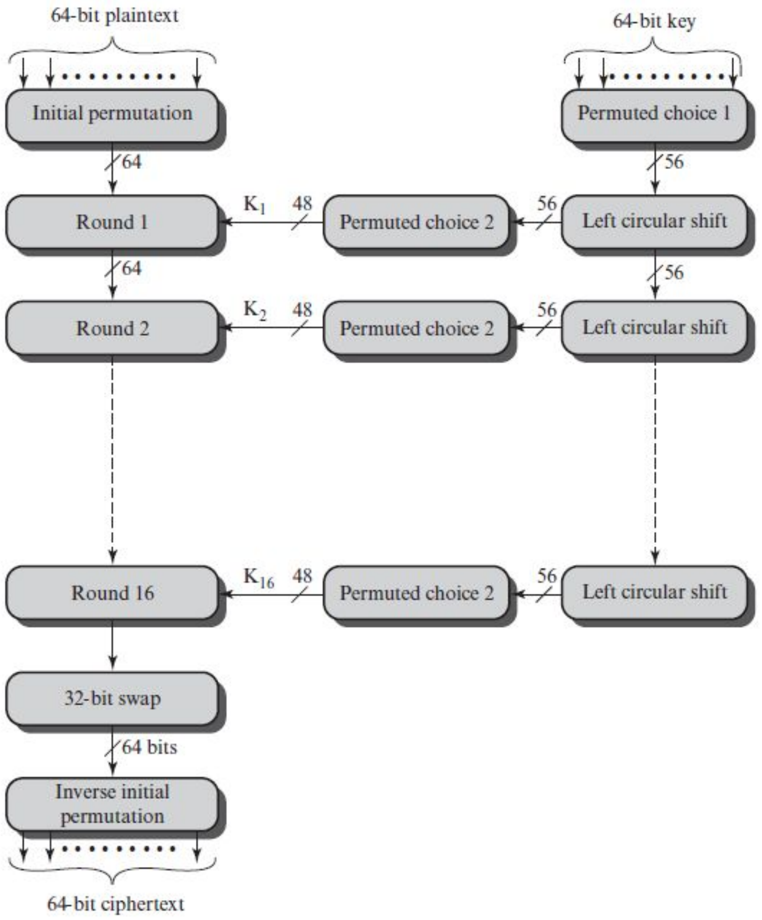
- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*
- This is followed by a phase consisting of **sixteen rounds** of the same function, which involves **both permutation and substitution** functions.
- The output of the l**ast (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key**.
- The left and right halves of the output **are swapped to produce the preoutput.**
- Finally, the **preoutput is passed through a permutation** [IP-1] that is the **inverse of the initial permutation function**, to produce the 64-bit ciphertext.
- With the exception of the initial and final permutations, DES has the **exact structure of a Feistel cipher**,



DES

Encryption

- The right-hand portion of Figure shows the way in which the 56-bit key is used.
- Initially, the key is passed **through a permutation function**.
- Then, **for each of the sixteen rounds**, a **subkey (Ki) is produced** by the **combination of a left circular** shift and a **permutation**.
- The **permutation function is the same for each round**, but a **different subkey is produced** because of the **repeated shifts of the key bits**.



DES

- DES is a symmetric encryption algorithm.
- Therefore, the **very key that is used to encrypt your plaintext data can also be used to decrypt it.**
- In a basic sense, **decryption requires the same steps as encryption** but **runs through them in reverse order.**
- application of the subkeys is reversed.
- Additionally, the initial and final permutations are reversed.

Why 56 bits?

- Use of **a 56-bit key** is one of the **most controversial aspects of DES**. Even before DES was adopted, people outside of the intelligence community complained that 56 bits provided inadequate security.
- **why were only 56 of the 64 bits of a DES key used in the algorithm**?
- The **disadvantage** of using **8 bits of the key for parity checking** is that it makes DES considerably less secure (256 times less secure against exhaustive search).
- what is the **advantage of using 8 bits of the key for parity?**
- Well, uh, let's say you receive a **key electronically**, and you want to sanity-check it to see if it could actually be a key. If you check the parity of the quantity, and it winds up not **having the correct parity**, then you'll know something went wrong.
- There are **two problems** with this reasoning. One is that there is a I in 256 chance (given the parity scheme) that even if you were to get 64 bits of garbage, that the result will happen to have the correct parity and therefore look like a key. That is way too large a possibility of error for it to afford any useful protection to any application. The other problem with the reasoning is that there is nothing terribly bad about getting a bad key. You'll discover the key is bad when you try to use it for encrypting or decrypting.

Why 56 bits?

- The key, at **56 bits, is pretty much universally acknowledged** to be too small to be secure.
- Perhaps one might argue that a **smaller key is an advantage** because it **saves storage**—but that **argument doesn't hold since nobody does data compression** on the 64-bit keys in order to fit them into 56 bits.
- So what benefits are there to usurping 8 bits for parity that offset the loss in security? People (not us, surely!) have suggested that our government consciously decided to weaken the security of DES just enough so that NSA would be able to break it.
- We would like to think there is an alternative explanation, but we have never heard a plausible one proposed.

Cryptography

DES Key Schedule, Strength

21 Oct 2024 C1/C3

-19 Oct 2024 C1/C3/c5

-18 Oct 2024 C5

14 Oct 2024 C1/C3

05 Oct 2024 C1/C3/C5

M S Vilkhu

-25 Oct 2024 C5

Topics

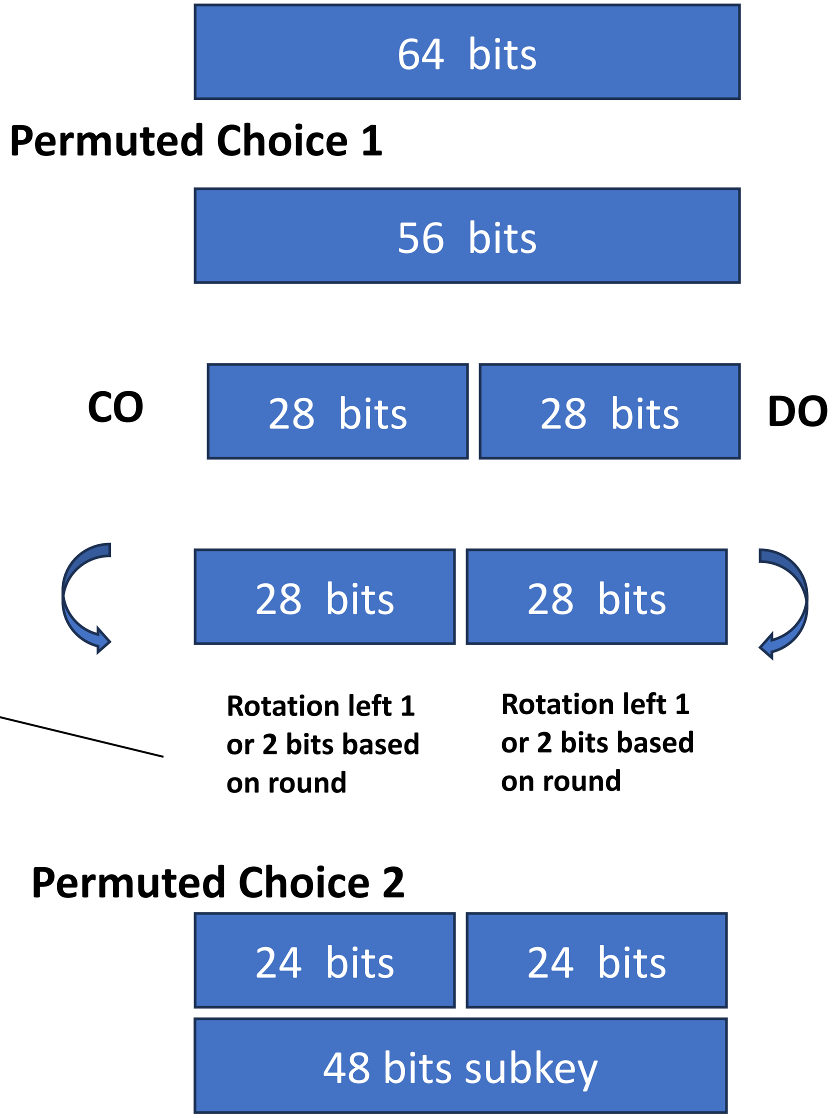
- Key Schedule
- What is S Box & P Box?
- Product cipher
- Avalanche Effect Vs Diffusion
- Strength & weakness of DES

Key Schedule

The **key schedule** in the Data Encryption Standard (DES) algorithm is a **series of steps that generates round keys** from **an initial key** to be used in encryption and decryption

1. **Permuted Choice 1 (PC-1)**: The initial 64-bit key is permuted to produce a 56-bit intermediate key. The remaining 8 bits are discarded or used for error detection
2. **Split into halves**: The 56-bit intermediate key is split into two 28-bit halves, C0 and D0.
3. **Rotations**: In each of 16 iterations, the halves are rotated left by 1 or 2 bits.
4. **Permuted Choice 2 (PC-2)**: 48 subkey bits are selected from the rotated halves. 24 bits are selected from the left half and 24 from the right
5. **Subkeys used in encryption and decryption**: The resulting subkeys are used in each round of encryption and decryption

iteration Number	number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1



Key Schedule

The **key schedule is a** **critical component** of DES because it makes the encryption and decryption process **more secure and effective**.

The key schedule **expands the short master key** to make the cryptosystem more difficult to attack.

The **key schedule for** **decryption is similar** to the one for encryption, but the **subkeys are in reverse order**.

S Box

Substitution Box or S-box

- An S-box (Substitution box) is a **fundamental component** used in many modern cryptographic algorithms, including block ciphers like DES (Data Encryption Standard) and AES (Advanced Encryption Standard).
- Its **main role** is to provide **non-linearity in the encryption process**, which makes it difficult for **attackers to reverse-engineer the cipher through cryptanalysis**
- An **S-box performs a substitution operation**
- where **input bits** are replaced with corresponding **output bits**. It takes a fixed number of input bits (say 6 bits in DES) and transforms them into a different number of output bits (say 4 bits in DES)

S Box

Substitution Box or S-box

- **Non-Linearity:**
- S-boxes **introduce non-linearity** into the encryption process, which is crucial to make the **relationship between the plaintext, ciphertext, and encryption key complex**.
- **Non-linearity means** the **output of the S-box is not a straightforward or predictable function** of its input
- **S-boxes contribute** to **confusion** by **obscuring this relationship**, preventing easy analysis by attackers.
- **Fixed Transformation**:
- An **S-box can be thought** of as a **lookup table or function**.
- For example, in DES, the input to the S-box might be 6 bits long, and the output will be 4 bits long. The 6-bit input is **used to index** into the S-box table, which gives the 4-bit output.

S Box

Substitution Box or S-box

- S-Box Example in DES:
- In DES, there are **8 different S-boxes,** each with a specific substitution rule.
- These S-boxes are crucial in making DES **resistant to cryptanalytic attacks** like **linear and differential cryptanalysis**.
- Example:
- If an S-box takes the **6-bit input 011011**, it might output the **4-bit value 1001** based on a predefined **substitution table.**
- Each of the 8 S-boxes in DES has a different substitution table.

P Box

P-box (Permutation box)

- A P-box (Permutation box) is a **fundamental** component used in many block ciphers, including DES.
- The P-box is responsible for **permuting or rearranging the bits** of its input.
- Unlike an S-box, which performs substitution (changing one value to another), **a P-box only rearranges** the bits of the input without changing their values.
- This operation **enhances the diffusion property of a cipher**, which helps to obscure the relationship between the input plaintext and the output ciphertext.

P Box

P-box (Permutation box)

- this is a simple **bit-shuffling operation** that mixes the bits without performing any mathematical operations
- Example of a P-Box in DES:
- In DES, the P-box is used in the permutation step to shuffle the bits after they have been processed by the S-boxes.
- Specifically, after the output of the **8 S-boxes** (which transforms 48 bits into 32 bits), the **P-box performs a straight permutation on the 32-bit output**, further **scrambling the data before the next round begins**

A Product cipher

- A product cipher is a type of **cryptographic algorithm** that **combines multiple simple encryption operations**, such as **substitution and permutation**, to create a more secure cipher.
- By combining these operations, **product ciphers achieve stronger security** than using just one operation alone.
- Product ciphers are widely **used in modern cryptography**, with many famous encryption algorithms, like **DES (Data Encryption Standard) and AES (Advanced Encryption Standard**), being based on the product cipher principle.

- **Substitution-Permutation Network (SPN)**:
- Many product **ciphers follow the Substitution-Permutation Network (SPN**) model, which combines two fundamental techniques substitution and permutation.

The Avalanche Effect

- A **desirable property of any encryption algorithm**
- is that a **small change** in either the **plaintext** or the **key** should **produce a significant change in the ciphertext**.
- In particular, a change in **one bit** of the plaintext or one bit of the key should produce a change **in many bits** of the ciphertext. This is referred to as the **avalanche effect**.
- If the **change were small**, this might provide a **way to reduce the size of the plaintext** or **key space to be searched**.

The Avalanche Effect

- **Change in plaintext**
- The second column of the table shows the intermediate 64-bit values at the end of each round for the two plaintexts.
- The third column shows the number of bits that differ between the two intermediate values.
- The table shows that,
  - after **three rounds**, **18 bits differ** between the two blocks.
  - On completion, the two ciphertexts differ In **32 bit positions**.

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

when the fourth bit of the plaintext is changed, so that the plaintext is **12468aceeca86420**

intermediate 64-bit values at the end of the round

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

The Avalanche Effect

- **Change in Key**
- Again, the results show that about **half of the bits in the ciphertext differ** and that the **avalanche effect** is **pronounced** after just a few rounds.

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

the original key, **0f1571c947d9e859**, and the altered key, **1f1571c947d9e859**

intermediate 64-bit values at the end of the round

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	27
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

The Avalanche Effect Vs Diffusion

- While **diffusion** and the **avalanche** effect are **closely related concepts** in cryptography, they are **not exactly the same**.
- **Both are important** in the context of block ciphers like DES (Data Encryption Standard) and other symmetric encryption algorithms, but they **serve different purposes**. Let's break them down:

The Avalanche Effect Vs Diffusion

1. Diffusion:

- **What it means**: Diffusion refers to the process by which **small changes in the input** (plaintext or key) **affect many parts of the output** (ciphertext).
- In essence, **it means spreading the influence** of **each bit of the input** across the **entire ciphertext** to hide statistical relationships in the plaintext.

- **How it works in DES**: In DES, diffusion is achieved through **multiple rounds of permutation and substitution**. Specifically, DES uses a **permutation function** (called the **P-box**) that **rearranges** the bits of the data block. This ensures that **bits from the input** are **diffused across multiple parts of the output**, making it harder to **deduce patterns** from the ciphertext.
- **Goal**: The purpose of diffusion is to ensure that the **ciphertext doesn't exhibit patterns** that can be **traced back to the plaintext**, thereby **making statistical attacks difficult**.

The Avalanche Effect Vs Diffusion

2. Avalanche Effect:

- **What it means**: The avalanche effect refers to a property of cryptographic algorithms where a **small change in the input** (such as flipping a single bit in the **plaintext or key**) causes a **drastic and unpredictable change in the output** (ciphertext).
- Ideally, a **minor change** should affect **roughly half of the output bits**.
- **How it works in DES**: DES achieves the avalanche effect **by using its Feistel structure** with **multiple rounds of substitution and permutation.**
- A **small change** in the input (plaintext or key) **propagates through the rounds**, and because of the diffusion and confusion operations in DES, this change will **affect many bits** in the final ciphertext. After just a few rounds, the output becomes vastly different from the original output.
- **Goal**: The purpose of the avalanche effect is to **make cryptanalysis harder.** Even if an attacker knows most of the input, the unpredictable change in the ciphertext caused by a small alteration makes it difficult to **reverse-engineer the encryption** or guess the key.

The Avalanche Effect Vs Diffusion

3. Are Diffusion and Avalanche Effect the Same?

- No, they are not exactly the same, though they are **closely related**:
- **Diffusion** is a **design principle** aimed at **spreading the impact of each input bit** across the ciphertext.
- It works to **eliminate any simple relationship** between the input and output, **preventing statistical attacks.**

- **The Avalanche Effect** is a **measurable property** **of a cipher** that indicates **how sensitive the output is to small changes** in the input.
- It's a **result of good diffusion and confusion**. A cipher with strong diffusion should exhibit a strong avalanche effect, meaning a small input change leads to a large, seemingly random change in the output.

The Avalanche Effect Vs Diffusion

5. Relationship in Symmetric Ciphers (Like DES):

- In symmetric ciphers, **diffusion** contributes to achieving the **avalanche effect**.
- In DES, the combination of **substitution** and **permutation** in **multiple rounds** ensures that even a **single bit change** in the plaintext or key **causes widespread changes** in the ciphertext (avalanche effect).
- **Both properties** are critical for strong encryption.
- Good **diffusion** ensures that **patterns in the plaintext** are **not reflected** in the ciphertext, and
- the **avalanche effect** ensures that **even small input changes** result in **vastly different ciphertexts**, making it more secure against cryptanalysis.
- The **avalanche effect** is the result of good diffusion and confusion, ensuring that small changes in the input produce significant changes in the output.
- In DES and other symmetric ciphers, diffusion helps achieve the avalanche effect, but they are **distinct concepts** that work together to enhance the security of the encryption.

The Avalanche Effect Vs Diffusion

Aspect	Diffusion	Avalanche Effect
Definition	The process of **spreading the influence** of each input bit (plaintext or key) across the entire ciphertext.	The phenomenon where a **small change in the input (e.g., flipping a single bit) causes a significant and unpredictable change** in the output (ciphertext).
Objective	**To hide statistical patterns** in the plaintext by distributing the impact of input bits widely across the ciphertext.	To ensure that **even minor changes in the input result in a significantly different outpu**t, enhancing security.
How it works in DES	Achieved through multiple rounds of permutation and substitution **(e.g., using the P-box and S-boxes).** This ensures that each bit of the plaintext affects many bits of the ciphertext.	A small change in the plaintext or key results in large changes in the ciphertext **after just a few rounds.** This is due to the combination of confusion and diffusion in the encryption rounds.
Main Function	To **obscure relationships between** the plaintext and the ciphertext, making it difficult for attackers to analyze patterns.	To create a **drastic difference** in the ciphertext even when the input is changed slightly, making cryptanalysis much more difficult.
Cryptographic Role	A **design principle** aimed at making the ciphertext appear as random as possible by diffusing the input's influence.	A **measurable property** of a cipher that reflects its sensitivity to small input changes. Good diffusion helps achieve a strong avalanche effect.
Importance	Prevents **statistical attacks** by eliminating clear correlations between input and output.	**Enhances security** by ensuring that a small change in input doesn't lead to predictable or minimal changes in output, making it harder to reverse-engineer the encryption.
Relationship	A **critical factor** that helps **produce** the avalanche effect.	A result of **good diffusion and confusion mechanisms** in the cipher.
Example	In DES, diffusion is achieved by the **Feistel structure** using substitution and permutation.	A single bit change in DES input causes around 50% of the output bits to change **after a few rounds.**

Strength of the DES

- Security provided by DES. The concerns, fall into two areas:
  1. key size and
  2. the nature of the algorithm.

- **Use of 56 bit keys**

Strength of the DES

- **Use of 56 bit keys**
- With a key length of 56 bits, there **are 2^{56} possible keys**, which is approximately $7.2 * 10^{16}$ keys.
- Thus, on the face of it, a **brute-force attack appears impractical.**
- Assuming that, on average, **half the key space has to be searched**, a **single machine** performing one **DES encryption per microsecond** would take more than a **thousand years to break the cipher.**
- However, the assumption of **one encryption per microsecond** is overly **conservative**.
- In 1977, **Diffie and Hellman postulated** that the technology existed to build a **parallel machine** with **1 million encryption devices**, each of which could perform **one encryption per microsecond** [DIFF77]. This would bring the average search time down to **about 10 hours**. The authors estimated that the cost would be about $20 million dollars in 1977.

Strength of the DES

- **Use of 56 bit keys**
- With current technology, it is **not even necessary to use special, purpose-built hardware.**
- Rather, the **speed of commercial, off-the-shelf processors** threaten the security of DES. A recent paper from Seagate Technology [SEAG08] suggests that a rate of **1 billion (10^9) key combinations per second** is reasonable for today's multicore computers.
- Table shows how much time is required for a brute-force attack for various key sizes.

Key Size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 Decryptions/s	Time Required at 10^{13} Decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns $= 1.125$ years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns $= 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns $= 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns $= 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns $= 1.8 \times 10^{60}$ years	1.8×10^{56} years
26 characters (permutation)	Monoalphabetic	$2! = 4 \times 10^{26}$	2×10^{26} ns $= 6.3 \times 10^{9}$ years	6.3×10^{6} years

Strength of the DES

- **the nature of the algorithm**
- Another concern is the possibility that cryptanalysis is possible by **exploiting the characteristics of the DES algorithm**.
- The focus of concern has been on the eight **substitution tables, or S-boxes**

Strength of the DES

- The Data Encryption Standard (DES) was a widely used **symmetric-key block** cipher that became a standard for encrypting sensitive data.
- However, its strength has been significantly **diminished over time** due to advancements in **computational power** and **cryptanalysis techniques**.
- Here’s an overview of DES's strengths and weaknesses:
- **Strengths of DES**

1. **Historical Importance**:
    1. DES was one of the first algorithms to gain **widespread acceptance as a secure method** for data encryption.
    2. Its adoption by the National Institute of Standards and Technology (NIST) established it as a foundational element of cryptography.

Strength of the DES

Strengths of DES

2. Block Cipher Structure:

1. DES is a symmetric-key block cipher that operates on **64-bit blocks of data**, using a 56-bit key (although the key is technically 64 bits, 8 bits are used for parity checks).
2. The algorithm **utilizes a Feistel network structure**, which is known for its **strong diffusion and confusion properties.**
3. **Feistel Network**: This structure allows the **cipher to be reversible**.
4. It processes the input data through **multiple rounds** (**16 in the case of DES**), using **permutations and substitutions** to obscure the **relationship between the plaintext and ciphertext.**

Strength of the DES

- **Strengths of DES**

3. Mathematical Foundations:

1. DES employs **S-boxes (substitution boxes) that provide** non-linear transformations, making it **resistant to certain types of attacks**.
2. The permutations and the way the algorithm combines these **transformations contribute to its** **overall complexity**.

4. Ease of Implementation:

3. DES has been relatively easy to implement in **both hardware and software**.
4. This accessibility contributed to its **widespread adoption** in various applications, including banking and telecommunications.

5. Standardization:

5. Being a standardized algorithm, DES has **undergone extensive scrutiny and testing** by the cryptographic community, leading to a certain **level of trust** in its **implementation** and **theoretical foundations**.

Strength of the DES

- **Strengths of DES**

6. Resilience to Differential Cryptanalysis:

- DES was one of the first ciphers designed to withstand **differential cryptanalysis**, a powerful attack technique developed after DES was created.
- Even though **differential cryptanalysis can theoretically attack** DES, it still **requires a large amount of data and time to be successful.**

Strength of the DES

Weaknesses of DES

1. **Key Length**:
   1. The primary vulnerability of DES is its 56-bit key length. With advances in computational power, brute-force attacks have become feasible. Theoretical estimates indicate that a brute-force attack on DES could be accomplished in hours or less using modern hardware.
   2. **Brute Force**: In **1998**, the **EFF (Electronic Frontier Foundation**) demonstrated this vulnerability by building a machine specifically designed to crack DES, completing the task in about 56 hours.
2. **Vulnerability to Cryptanalysis**:
   1. DES is **susceptible to various cryptanalysis techniques**, including:
      1. **Differential Cryptanalysis**: This technique analyzes how differences in input can affect the resultant difference at the output. Although DES was designed to resist differential attacks, it can still be vulnerable under certain conditions.
      2. **Linear Cryptanalysis**: This method exploits linear approximations to describe the relationship between plaintext, ciphertext, and key bits. It has been shown that linear attacks can be effective against DES, particularly when large amounts of plaintext-ciphertext pairs are available.

Strength of the DES

Weaknesses of DES

3. Limited Block Size (64-bit):

- DES operates on **64-bit blocks**, which are considered small by today's standards.
- This smaller block size increases the likelihood of **repeating** blocks (especially in large datasets), which can reveal patterns in the ciphertext

Modern Solutions and Alternatives to DES

1. **Triple DES (3DES)**:
   1. To overcome the short key length problem of DES, **Triple DES (3DES)** was introduced. It effectively **applies DES three times**, increasing the effective key length to **112 or 168 bits**.
   2. However, 3DES is slower than modern ciphers like AES, and even though it improves security, it is still less efficient and secure than AES. In fact, 3DES has also been deprecated by the NIST (National Institute of Standards and Technology).
2. **AES (Advanced Encryption Standard)**:
   1. The most common replacement for DES is **AES**,
   2. offers much stronger security with larger key sizes (128, 192, and 256 bits), **larger block sizes** (128 bits), and a **more efficient design**.
   3. AES is now the standard for symmetric encryption.

Conclusion

- While DES was a **groundbreaking** and highly **influential** encryption algorithm in its time, its **weaknesses**, especially the **short key length** and **vulnerability to brute force attacks**, made it obsolete.
- The **primary strength** of DES lies in its **well-tested Feistel structure**, but its **56-bit key length is insufficient** for modern cryptographic security.
- Today, AES is the widely used standard for secure encryption, offering superior security and performance.

Thank You