Mustansiriyah University

College of Science – Department of CS/Cybersecurity

Cryptography Course

2023-2024

By

Prof. DR. Bashar AL-Esawi

Definition of cryptography:

Cryptography is the practice of securing communication from third-party eavesdropping by converting it into a coded language, using mathematical algorithms and secret keys, to protect the *confidentiality, integrity, and availability* of the information being transmitted.

Here are some common terms used in cryptography:

- **Plaintext:** The original message or data that is intended to be kept secret.
- **Encryption:** A method used to transform plaintext into ciphertext.
- **Ciphertext:** The encrypted message or data that has been transformed using a cipher.
- **Key:** A secret value used to encrypt or decrypt messages.
- **Encryption:** The process of transforming plaintext into ciphertext using a cipher and a key.
- **Decryption:** The process of transforming ciphertext back into plaintext using a cipher and a key.
- **Cryptosystem:** Combination of a cipher, key, and any related algorithms or protocols used to secure data.
- **Cryptanalysis:** The study of techniques used to break or weaken cryptographic systems.
- **Brute force attack:** A cryptanalysis technique where all possible keys are tried to decrypt a message.
- **Protocol:** Set of rules and procedures used to secure communication between 2 or more parties.

What is the role of cryptography in cybersecurity?

Cryptography plays a crucial role in cybersecurity by providing a means to protect sensitive information and communications from unauthorized access, interception, or modification. Here are some key roles of cryptography in cybersecurity:

1. **Confidentiality:** Cryptography helps ensure that data is kept secret and protected from unauthorized disclosure.

2. **Integrity:** Cryptography helps ensure that data is not tampered with or modified in transit, providing data integrity and trustworthiness.

3. **Authentication:** Cryptography helps establish the identity of the communicating parties, ensuring that data is exchanged only between trusted parties.

4. **Non-repudiation:** Cryptography helps ensure that parties cannot deny their involvement in the exchange of information or transactions, providing proof of the exchange.

Overall, cryptography provides the foundation for many cybersecurity technologies, including secure communication protocols, digital signatures, and data encryption, that protect sensitive information and critical systems from cyber threats.

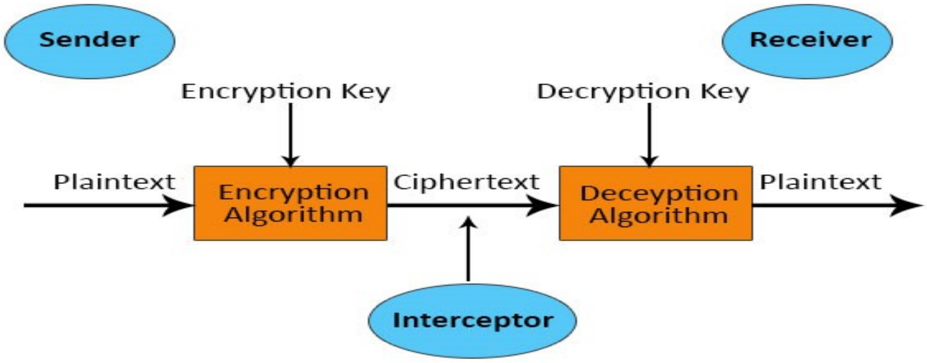


Figure (1): The Cryptosystem Architecture.

Description of Confidentiality, Integrity, and Authenticity (CIA):

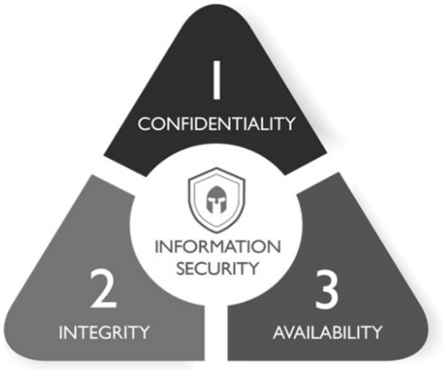
Confidentiality, Integrity, and Authenticity (CIA) are three important principles that form the basis of information security.

Confidentiality refers to the protection of sensitive information from unauthorized access or disclosure. It involves ensuring that only authorized users have access to the information and that the information is protected from interception or eavesdropping during transmission.

Integrity refers to the protection of the accuracy and completeness of data and information. It involves ensuring that data is not altered or destroyed in an unauthorized or unintended manner and that the data remains consistent and accurate over time.

Availability refers to the assurance that data or information is genuine and can be trusted. It involves verifying the identity of the user or the source of the information and ensuring that the data has not been tampered with or modified.

Together, these principles form the basis of a secure and trustworthy information system that protects against unauthorized access, malicious attacks, and unintentional errors. They are critical to maintaining the confidentiality, accuracy, and reliability of sensitive information and are essential for maintaining the trust of users and stakeholders.



Information security, Data security, Network security, and Cyber security:

Information security, data security, network security, and cyber security are all related but distinct fields within the broader realm of computer security.

Information security refers to the protection of information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction. The goal of information security is to ensure confidentiality, integrity, and availability of information by applying a risk management process and giving assurance that information security requirements are met.

Data security, on the other hand, is the practice of protecting data, both in storage and in transit, from unauthorized access or alteration. This involves implementing various technologies, processes, and policies to secure sensitive data and prevent data breaches.

Network security focuses on protecting the infrastructure of computer networks, including the devices, protocols, and data that traverse them. The aim of network security is to prevent unauthorized access, use, disclosure, disruption, modification, or destruction of network resources and to ensure the confidentiality, integrity, and availability of data transmitted over the network.

Cyber security, also known as computer security, refers to the protection of internet-connected systems, including hardware, software, and data, from attack, damage, or unauthorized access. This field encompasses a wide range of technologies, processes, and practices aimed at safeguarding computer systems and networks from cyber threats such as malware, ransomware, and hacking.

Each of these fields is critical for protecting the security and privacy of information and data in the digital age, and they often overlap and intersect with one another, See Figure (2).

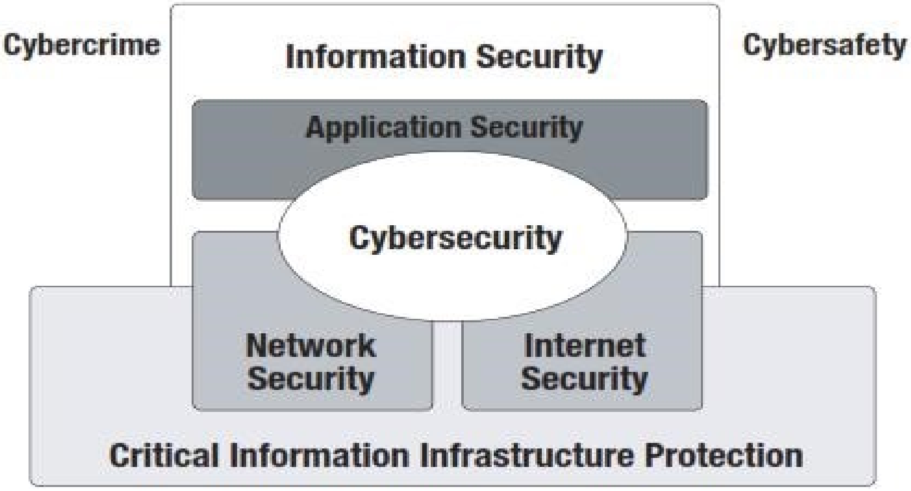


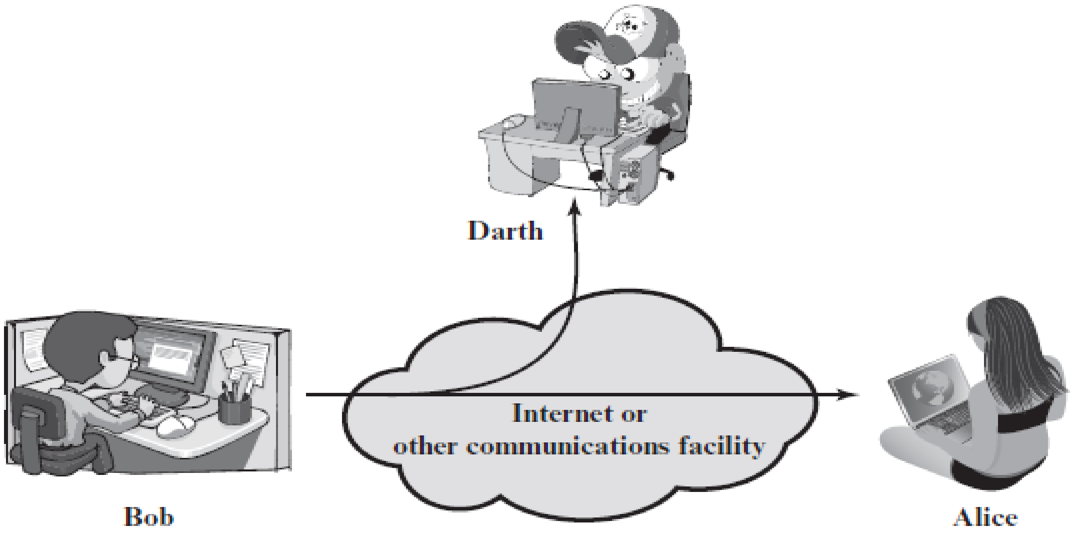
Figure (2): Critical Information Infrastructure Protection.

OSI Security Architecture:

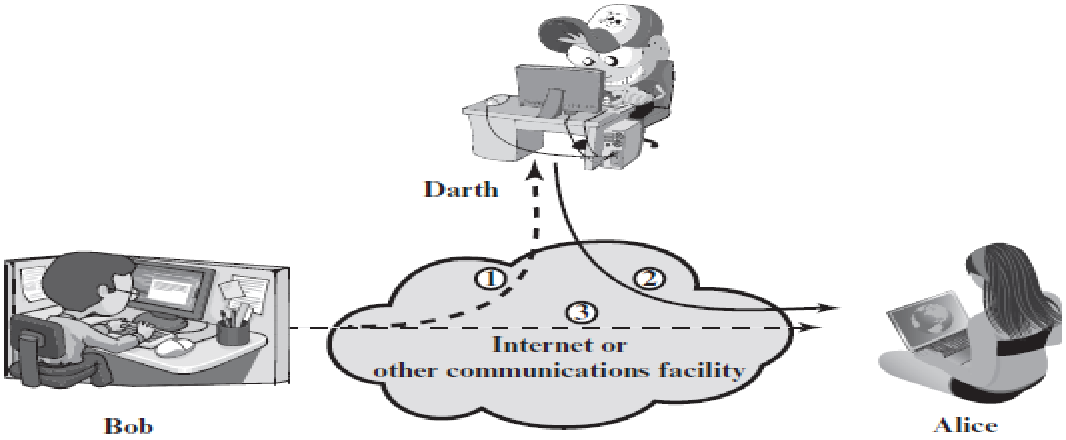
the OSI security architecture provides a useful, if abstract, overview of many of the concepts that this book deals with. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as

■ **Security attack:** Any action that compromises the security of information owned by an organization.

1. **Passive Attacks:** Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Passive attacks are very difficult to detect, because they do not involve any alteration of the data. Two types of passive attacks are:
    a. **Release of message contents:** A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. *We would like to prevent an opponent from learning the contents of these transmissions.*
    b. **Traffic analysis:** Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. *The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.*
2. **Active Attacks:** Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: M*asquerade, replay, Modification of messages, and Denial of service.*



(a) Passive attacks



(b) Active attacks

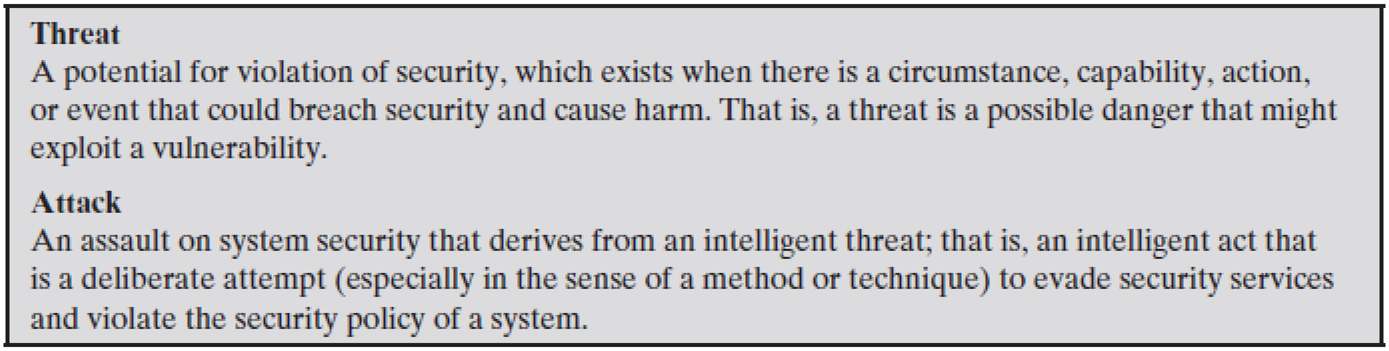
- A **Masquerade** takes place when one entity pretends to be a different entity (path 2 of Figure b is active).
- **Replay** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect (paths 1, 2, and 3 active).
- **Modification** of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect (paths 1 and 2 active).
- **Denial of service** prevents or inhibits the normal use or management of communications facilities (path 3 active). This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

■ **Security Service:** a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers. A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service. These services can be described as follow:

AUTHENTICATION, ACCESS CONTROL, DATA CONFIDENTIALITY,

DATA INTEGRITY, NONREPUDIATION, AVAILABILITY SERVICE

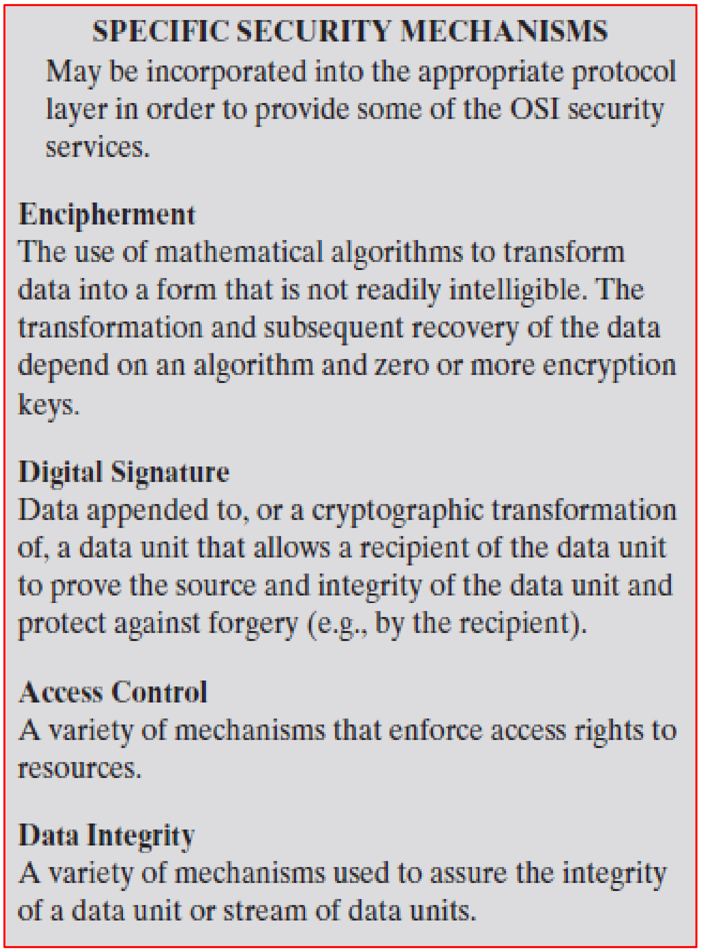
In the literature, the terms threat and attack are commonly used to mean more or less the same thing.



Threat
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Attack
An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

■ **Security mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.



SPECIFIC SECURITY MECHANISMS

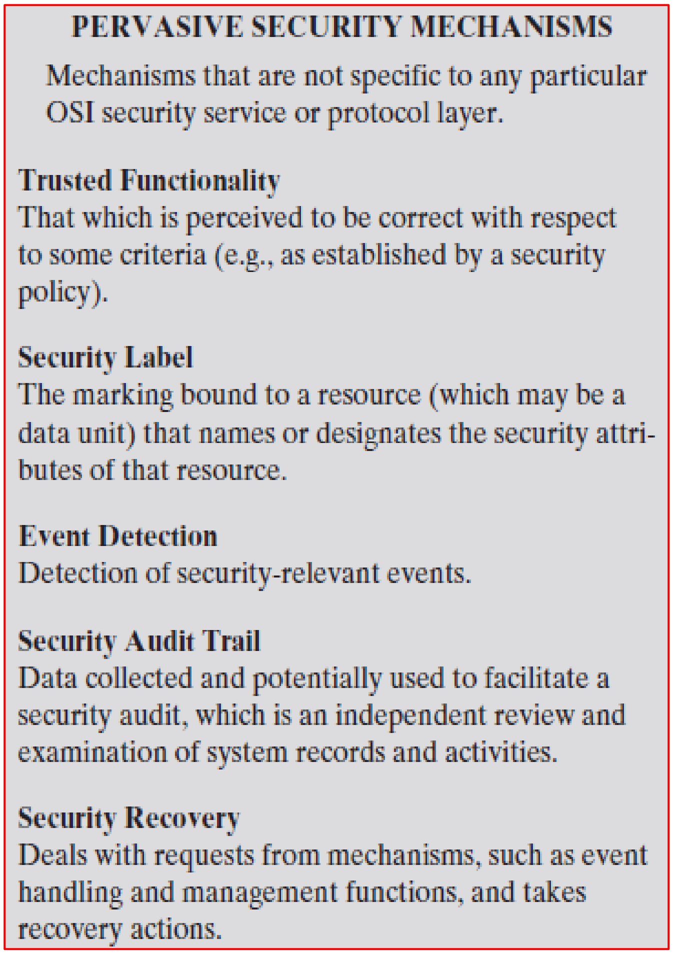
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment
The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

Digital Signature
Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control
A variety of mechanisms that enforce access rights to resources.

Data Integrity
A variety of mechanisms used to assure the integrity of a data unit or stream of data units.



PERVASIVE SECURITY MECHANISMS

Mechanisms that are not specific to any particular OSI security service or protocol layer.

Trusted Functionality
That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label
The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection
Detection of security-relevant events.

Security Audit Trail
Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery
Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

Basic Number Theory

DIVISIBILITY AND THE DIVISION ALGORITHM:

Divisibility is a mathematical concept that describes the ability of one number to be divided exactly by another number without leaving a remainder. In other words, if one number is divisible by another number, it means that the first number is a multiple of the second number.

For example, 15 is divisible by 3 because 15 can be divided exactly by 3, which gives a quotient of 5 and a remainder of 0. Similarly, 10 is divisible by 5 because 10 can be divided exactly by 5, which gives a quotient of 2 and a remainder of 0.

The Division Algorithm is a method used to find the quotient and remainder when one integer (the dividend) is divided by another integer (the divisor). The algorithm states that if a and b are any two integers, with b being non-zero, then there exist unique integers q (the quotient) and r (the remainder). **THE FOLLOWING SNIP OF CODE IN C#:**

```
using System;
class Program
{
   static void Main (string [ ] args)
          {
           int a = 15; // the dividend
           int b = 3; // the divisor
          if (a % b == 0)
                    Console.WriteLine("{0} is divisible by {1}", a, b);
          else
                    Console.WriteLine("{0} is not divisible by {1}", a, b);
          int q = a / b; // the quotient
          int r = a % b; // the remainder
          Console.WriteLine("{0} = {1}({2}) + {3}", a, b, q, r);
          }
}
```

Greatest Common Divisor:

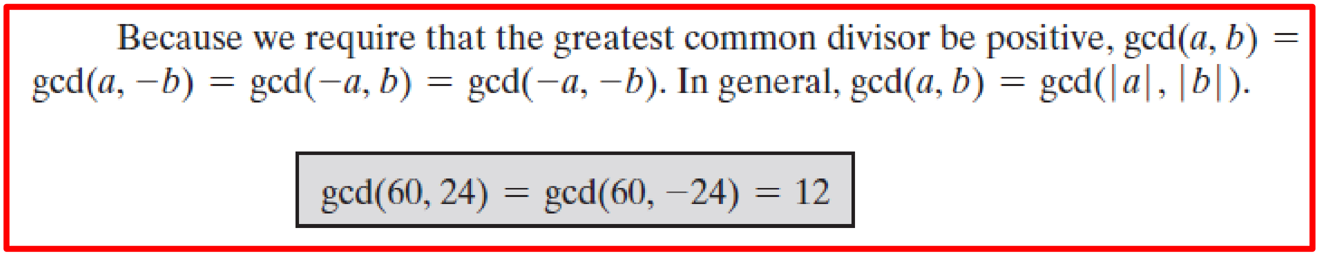
The Greatest Common Divisor (GCD) is the largest positive integer that divides two or more numbers without leaving a remainder. It is also known as the Highest Common Factor (HCF). The GCD of two or more integers can be found by identifying the common factors of the integers and selecting the largest one.

For example, consider the numbers 24 and 36. The factors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24, while the factors of 36 are 1, 2, 3, 4, 6, 9, 12, 18, and 36. The common factors of 24 and 36 are 1, 2, 3, 4, 6, and 12. Therefore, the GCD of 24 and 36 is 12.

Another example is the numbers 18, 24, and 36. The factors of 18 are 1, 2, 3, 6, 9, and 18, the factors of 24 are 1, 2, 3, 4, 6, 8, 12, and 24, and the factors of 36 are 1, 2, 3, 4, 6, 9, 12, 18, and 36. The common factors of 18, 24, and 36 are 1, 2, 3, 6, and 9. Therefore, the GCD of 18, 24, and 36 is 3.

THE FOLLOWING SNIP OF CODE IN C#:

```
using System;
class Program
{
  static void Main(string[] args)
        {
      int a = 24;
      int b = 36;
      int gcd = FindGCD(a, b);
      Console.WriteLine("The GCD of {0} and {1} is {2}", a, b, gcd);
        }
  static int FindGCD(int a, int b)
        {
      while (b != 0)
                 {
                  int temp = b;
                  b = a % b;
                  a = temp;
                  }
      return a;
        }
}
```



Because we require that the greatest common divisor be positive, $\gcd(a, b) = \gcd(a, -b) = \gcd(-a, b) = \gcd(-a, -b)$. In general, $\gcd(a, b) = \gcd(|a|, |b|)$.

$$\boxed{\gcd(60, 24) = \gcd(60, -24) = 12}$$

MODULAR ARITHMETIC:

If a is an integer and n is a positive integer, we define a mod n to be the remainder when a is divided by n. The integer n is called the modulus.

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

To calculate -23 mod 13, we can first calculate the remainder of the division of 23 by 13, which is 10. Since the dividend is negative, we then subtract the divisor from the remainder and obtain:

-23 mod 13 = 10 - 13 = -3, **Therefore, -23 mod 13 is -3.**

Another Examples:

$$\begin{aligned}
&11 \bmod 8 = 3;\ 15 \bmod 8 = 7\\
&[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = 10 \bmod 8 = 2\\
&(11 + 15) \bmod 8 = 26 \bmod 8 = 2\\
&[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = -4 \bmod 8 = 4\\
&(11 - 15) \bmod 8 = -4 \bmod 8 = 4\\
&[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = 21 \bmod 8 = 5\\
&(11 \times 15) \bmod 8 = 165 \bmod 8 = 5
\end{aligned}$$

One common method is the modular exponentiation algorithm, which is designed specifically for this purpose.

The modular exponentiation algorithm works by repeatedly squaring the base and taking the remainder modulo the modulus, which reduces the number of multiplications needed. Here's how the algorithm works, **using your example of calculating the remainder of 11^7 modulo 5:**

Step 1: Convert the exponent to binary form. In this case, 7 in binary is 111.

Step 2: Initialize a variable to hold the result, set it to 1.

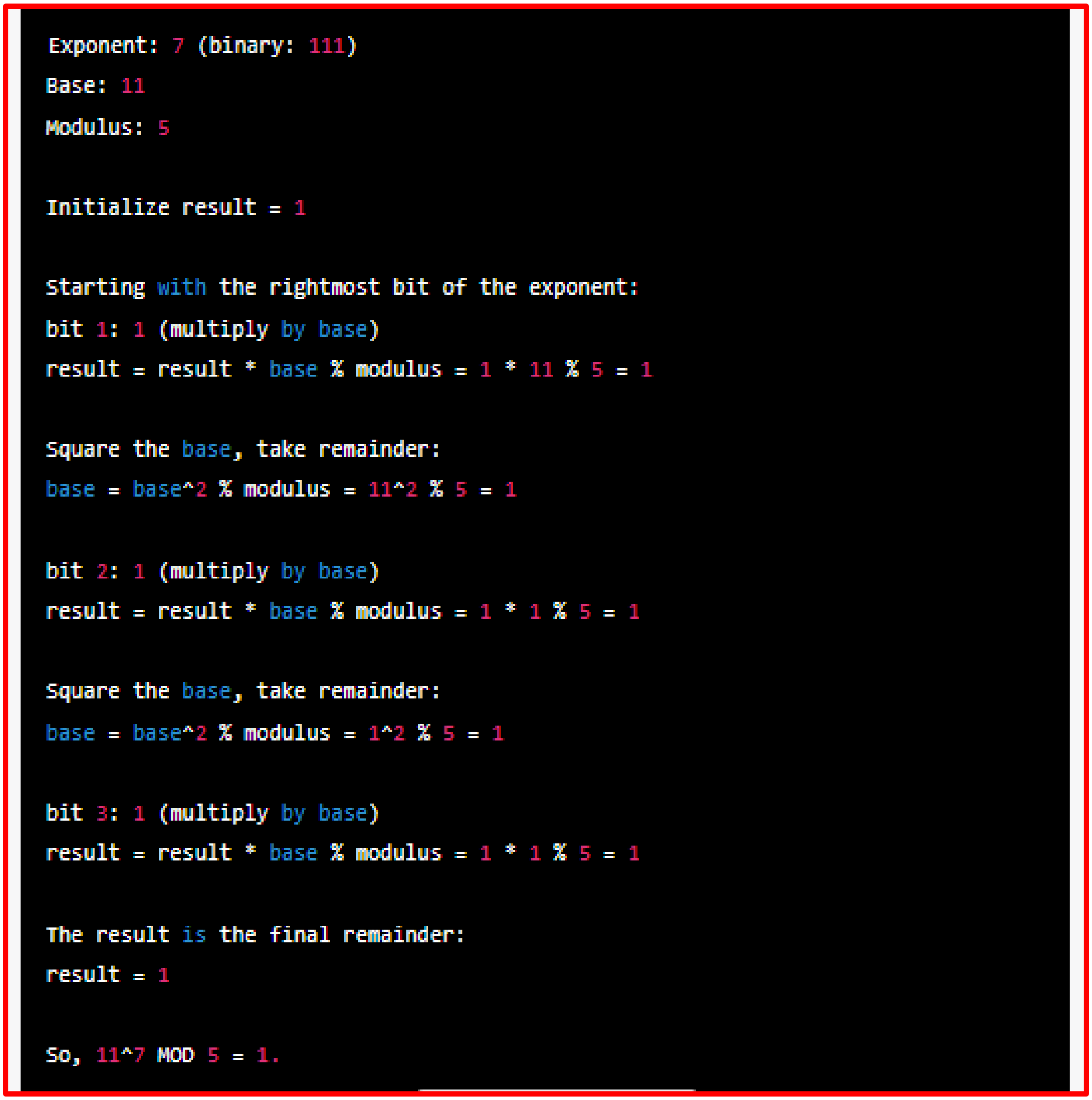
Step 3: For each bit in the binary form of the exponent, from right to left:

a) Square the result.

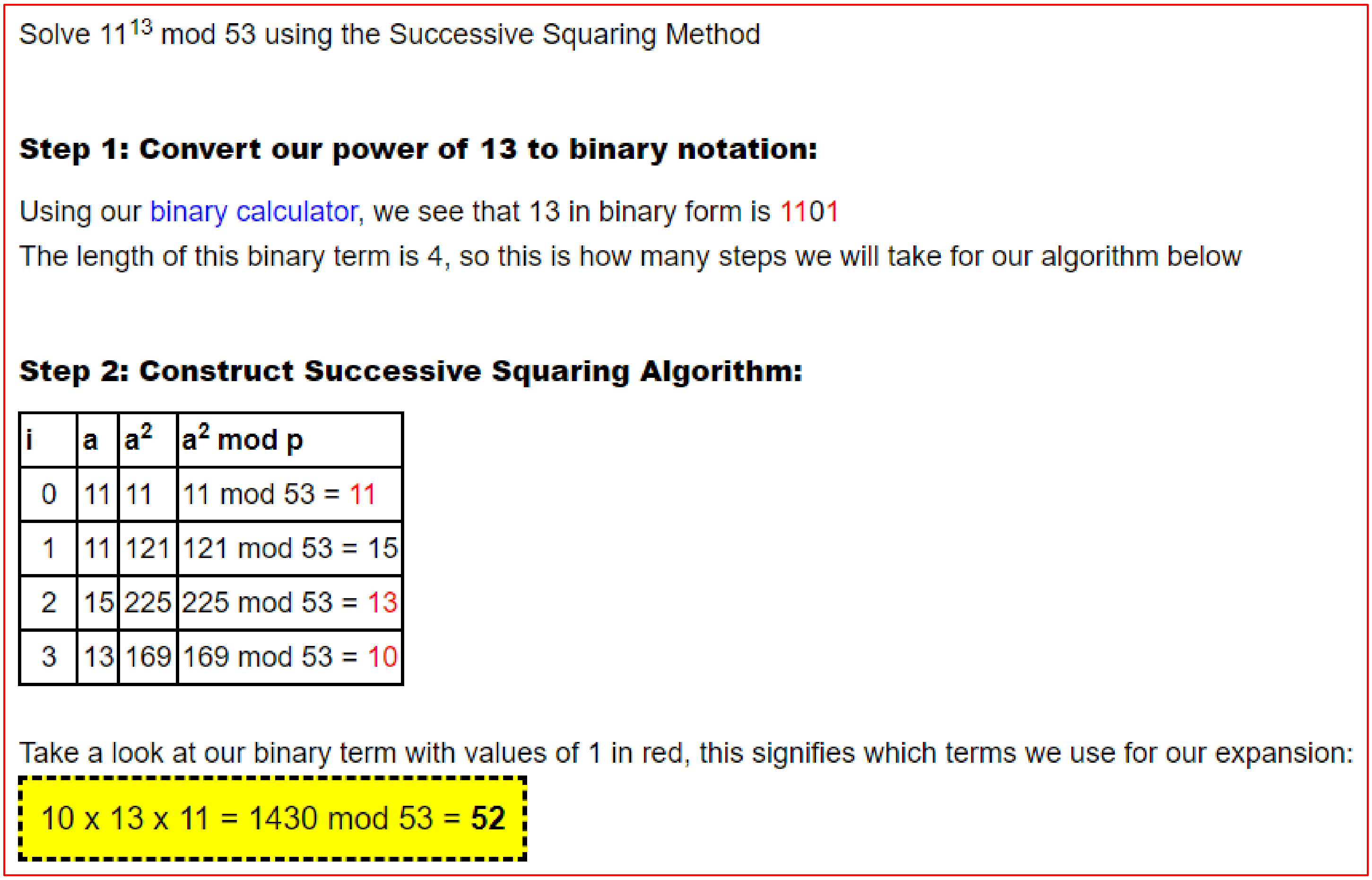
b) If the current bit is 1, multiply the result by the base.

c) Take the remainder of the result modulo the modulus.

Here's how this works in practice:



```
Exponent: 7 (binary: 111)
Base: 11
Modulus: 5

Initialize result = 1

Starting with the rightmost bit of the exponent:
bit 1: 1 (multiply by base)
result = result * base % modulus = 1 * 11 % 5 = 1

Square the base, take remainder:
base = base^2 % modulus = 11^2 % 5 = 1

bit 2: 1 (multiply by base)
result = result * base % modulus = 1 * 1 % 5 = 1

Square the base, take remainder:
base = base^2 % modulus = 1^2 % 5 = 1

bit 3: 1 (multiply by base)
result = result * base % modulus = 1 * 1 % 5 = 1

The result is the final remainder:
result = 1

So, 11^7 MOD 5 = 1.
```



Solve 11^{13} mod 53 using the Successive Squaring Method

Step 1: Convert our power of 13 to binary notation:

Using our binary calculator, we see that 13 in binary form is 1101
The length of this binary term is 4, so this is how many steps we will take for our algorithm below

Step 2: Construct Successive Squaring Algorithm:

i	a	a^2	a^2 mod p
0	11	11	11 mod 53 = 11
1	11	121	121 mod 53 = 15
2	15	225	225 mod 53 = 13
3	13	169	169 mod 53 = 10

Take a look at our binary term with values of 1 in red, this signifies which terms we use for our expansion:

10 x 13 x 11 = 1430 mod 53 = **52**

https://www.mathcelebrity.com/modexp.php?

PRIME NUMBERS:

A prime number is a positive integer greater than 1 that has no positive integer divisors other than 1 and itself. In other words, a prime number is a number that is only divisible by 1 and itself.
For example, the first few prime numbers are:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, ...

These numbers have only two positive integer factors, which are 1 and themselves. For example, 2 can only be divided by 1 and 2, 3 can only be divided by 1 and 3, 5 can only be divided by 1 and 5, and so on.

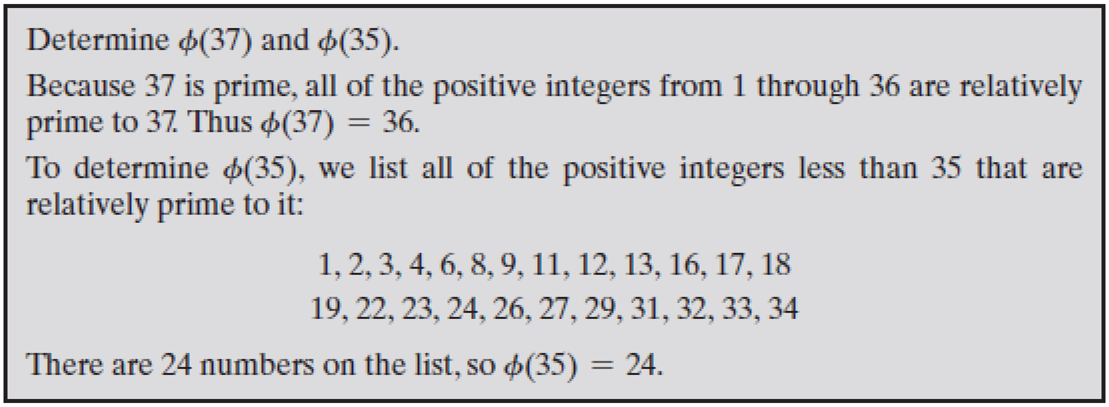
Prime numbers are important in many areas of mathematics and computer science. They are the building blocks of the integers, and many complex mathematical structures and algorithms rely on their properties. For example, public-key cryptography, which is used to secure online transactions, relies on the fact that it is very difficult to factor large composite numbers into their prime factors.

Q) Find all Prime numbers that in range [313-31313]?

https://planetcalc.com/9003/

Euler’s Totient Function vs Euler Function:

"Euler's function" and "Euler's totient function" are two different names for the same mathematical concept. Both names refer to the function that counts the number of positive integers less than or equal to n that are relatively prime to n. The function is denoted by the symbol φ(n) or sometimes by ϕ(n).



Determine $\phi(37)$ and $\phi(35)$.

Because 37 is prime, all of the positive integers from 1 through 36 are relatively prime to 37. Thus $\phi(37) = 36$.

To determine $\phi(35)$, we list all of the positive integers less than 35 that are relatively prime to it:

$$1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18$$
$$19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34$$

There are 24 numbers on the list, so $\phi(35) = 24$.

It should be clear that, for a prime number p, ϕ (p) = p - 1

Now suppose that we have two prime numbers p and q with p ≠ q. Then we can show that, for n = p*q,

$$\phi(n) = \phi(p*q) = \phi(p) * \phi(q) = (p - 1) * (q - 1)$$

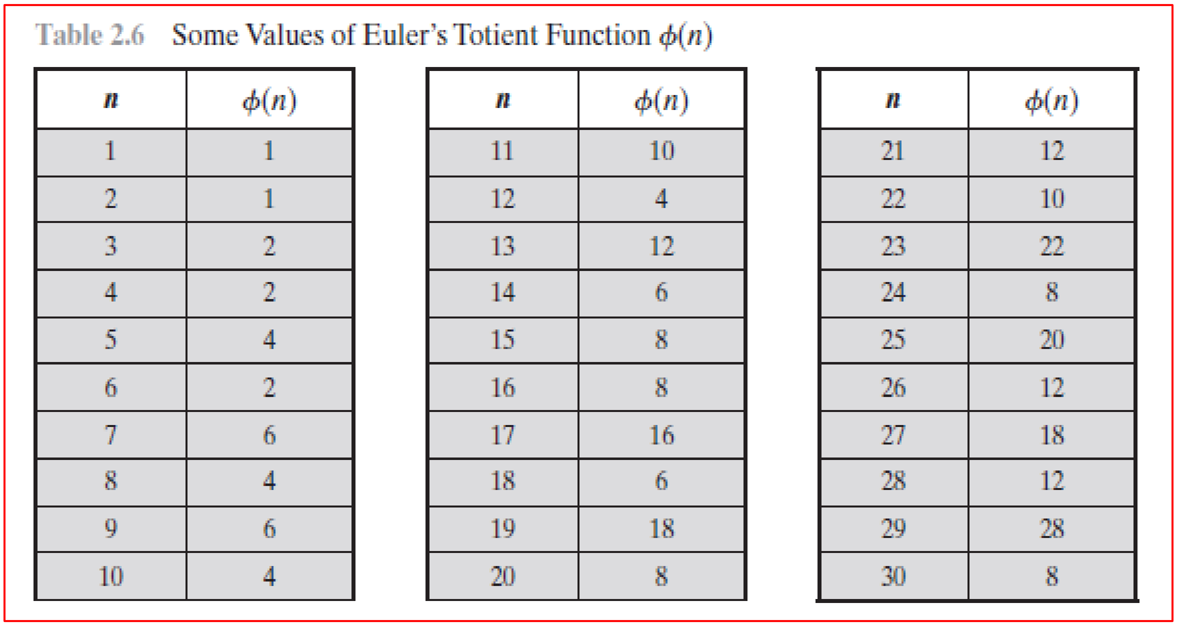
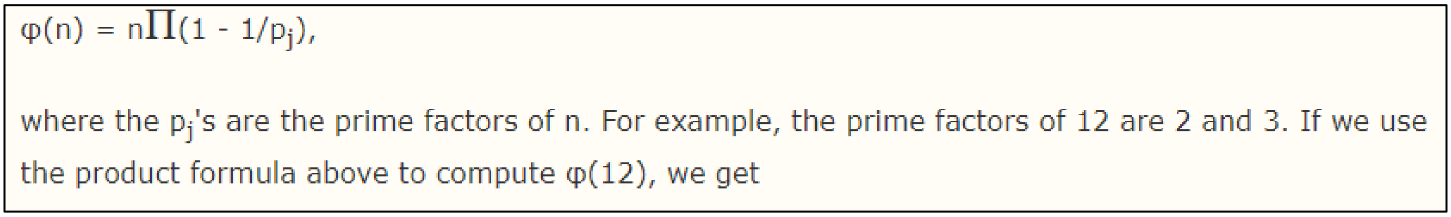


Table 2.6 Some Values of Euler’s Totient Function $\phi(n)$

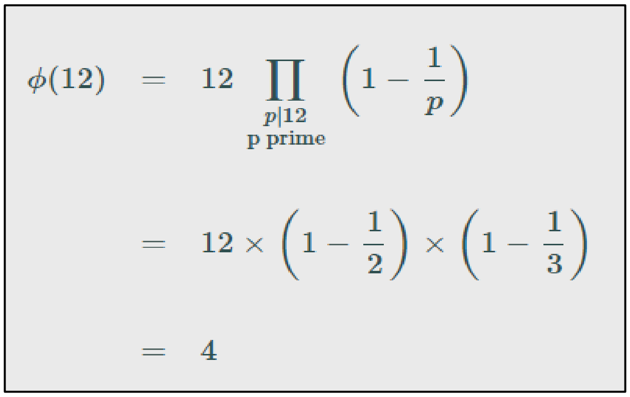
n	$\phi(n)$	n	$\phi(n)$	n	$\phi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

In number theory, the Euler Phi Function or Euler Totient Function φ(n) gives the number of positive integers less than n that are relatively prime to n, i.e., numbers that do not share any common factors with n. For example, φ(12) = 4, since the four numbers 1, 5, 7, and 11 are relatively prime to 12.

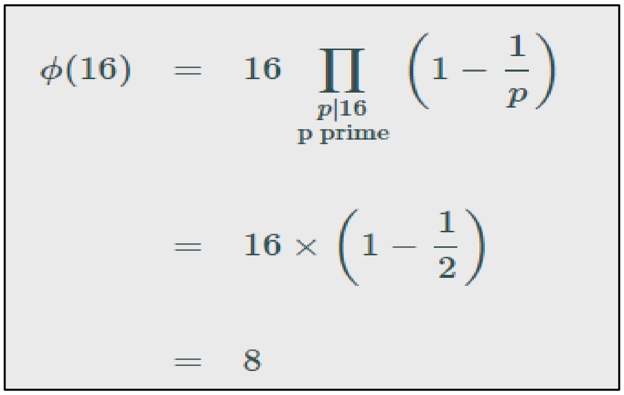


$\varphi(n) = n\prod(1 - 1/p_j)$,

where the p_j's are the prime factors of n. For example, the prime factors of 12 are 2 and 3. If we use the product formula above to compute φ(12), we get



$$\begin{aligned} \phi(12) &= 12 \prod_{\substack{p|12 \\ \text{p prime}}} \left(1 - \frac{1}{p}\right) \\ &= 12 \times \left(1 - \frac{1}{2}\right) \times \left(1 - \frac{1}{3}\right) \\ &= 4 \end{aligned}$$



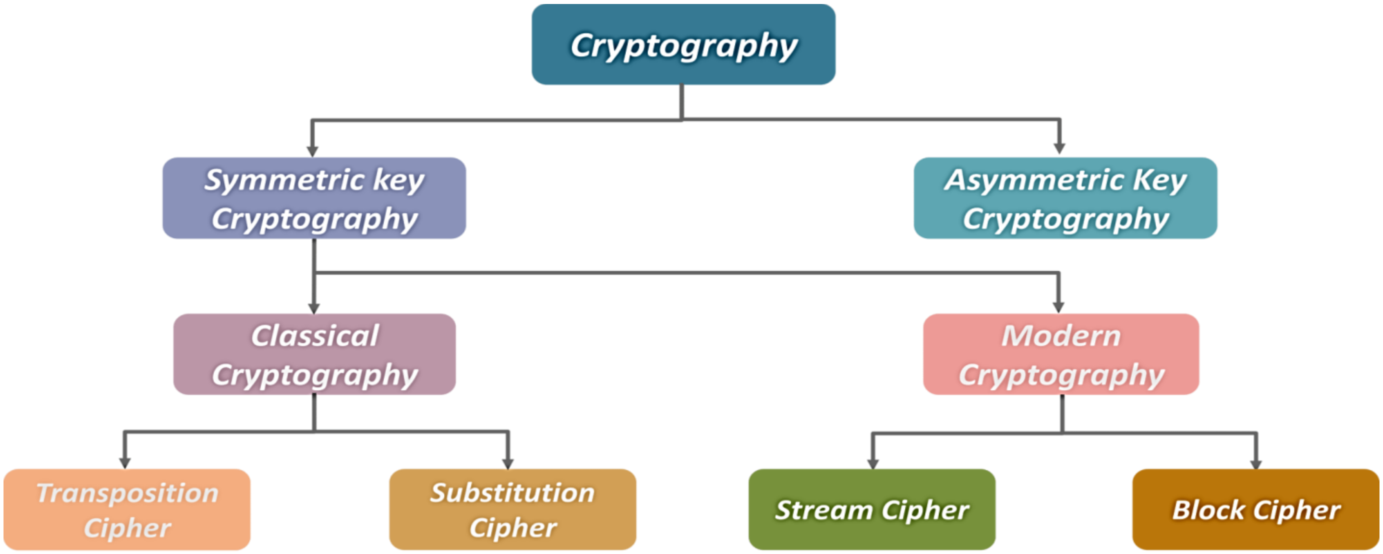
$$\begin{aligned} \phi(16) &= 16 \prod_{\substack{p|16 \\ \text{p prime}}} \left(1 - \frac{1}{p}\right) \\ &= 16 \times \left(1 - \frac{1}{2}\right) \\ &= 8 \end{aligned}$$

https://www.had2know.org/academics/euler-totient-function-calculator.html

https://mathtools.lagrida.com/arithmetic/euler_totient.html

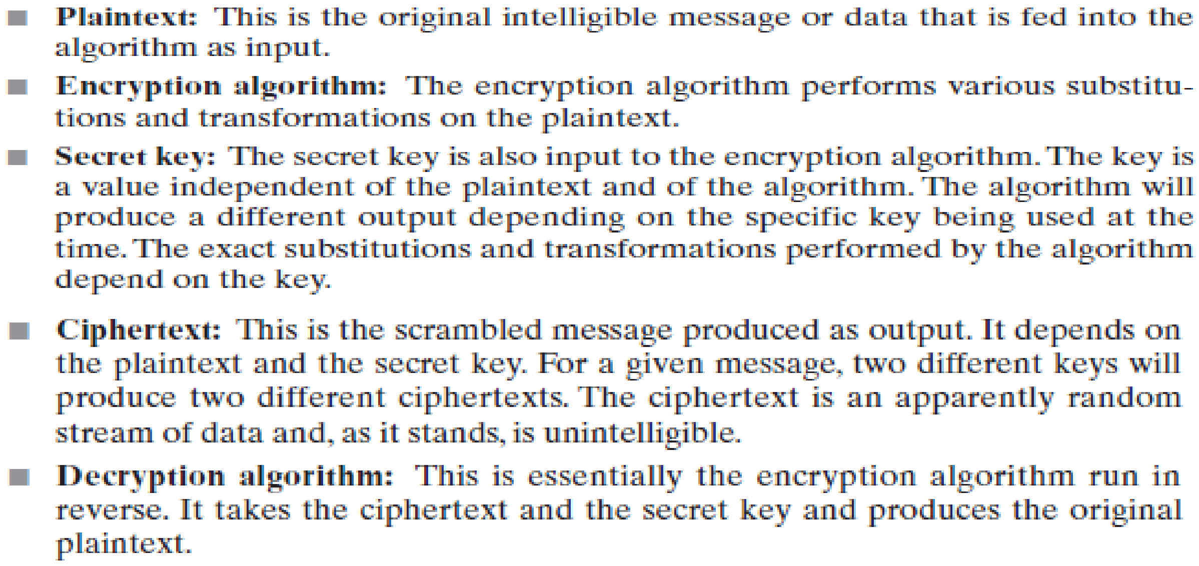
Classical Encryption Techniques

We can describe Cryptography simply by the following Framework:



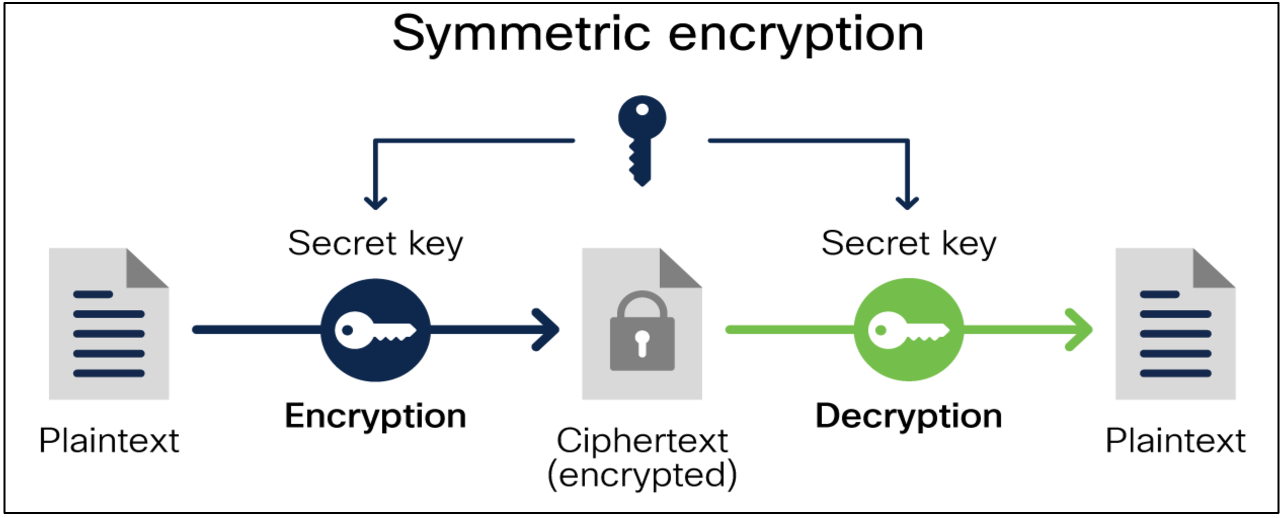
SYMMETRIC CIPHER MODEL:

A symmetric encryption scheme has five ingredients:



- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

The following figure describe the Symmetric Encryption System in general:



Note: All Encryption Methods depends on Alphabet, so the English alphabet as follow:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

SUBSTITUTION TECHNIQUES:

A substitution cipher is a type of encryption technique where each letter in the plaintext is replaced by another letter or symbol. There are several types of substitution ciphers, and here are a few examples:

1. **Caesar Cipher**: This is one of the simplest and most well-known substitution ciphers. It involves shifting each letter in the plaintext by a fixed number of positions in the alphabet. For example, with a shift of 3, "A" would become "D", "B" would become "E", and so on.

2. **Monoalphabetic Cipher**: In this cipher, each letter in the plaintext is replaced by a corresponding letter in the ciphertext. The substitution is determined by a **fixed key** or a **predetermined pattern**. It uses a fixed key which consist of the 26 letters of a “**shuffled alphabet**”.

3. **Polyalphabetic Cipher**: This type of cipher uses multiple substitution alphabets to encode the plaintext, making it more secure than monoalphabetic ciphers. One example is the **Vigenère cipher**, which uses a series of interwoven Caesar ciphers with different shift values based on a repeating keyword.

4. **Homophonic Cipher**: In this cipher, each letter in the plaintext is replaced by one or more symbols or letters in the ciphertext. This adds an extra layer of security, as there are multiple possible substitutions for each letter. However, it also makes the ciphertext longer and harder to decrypt.

5. **Polygraphic Cipher**: Instead of substituting letters one at a time, polygraphic ciphers substitute multiple letters or symbols at once. Examples include **Playfair cipher**, which substitutes pairs of letters, and Hill cipher, which uses matrix algebra to encrypt blocks of letters.

CAESAR CIPHER:

Caesar Cipher is a type of substitution cipher, which replaces each letter in the plaintext (the message to be encrypted) by a letter a fixed number of positions down the alphabet. For example, if the shift value is 3, then the letter 'A' in the plaintext would be replaced by the letter 'D' in the ciphertext, 'B' would be replaced by 'E', and so on.

Write down the alphabet:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Shift the letters by KEY that agree between Sender and Receiver places to the right:

Let Key =3; Write down the plaintext to be encrypted:

Plaintext=" HELLO"; C= E (P, Key) = (P + Key) mod 26;

Replace each letter in the plaintext with the corresponding letter in the shifted alphabet:

H becomes K E becomes H L becomes O L becomes O O becomes R

The Resulting Ciphertext is "KHOOR".

To Decrypt the ciphertext, you simply shift the letters back by 3 places to the left.

P = D (C, Key) = (C - Key) mod 26; So, in this case, "**KHOOR**" would be **decrypted** to "**HELLO**".

MONOALPHABETIC CIPHER:

Using a simple monoalphabetic cipher depend on the following algorithm:

- **Write down the alphabet:**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

- **Sender and Receiver agree about Create a new alphabet by randomly shuffling the letters:**
  - **MAY BE AS KEYWORD OR NEW ALPHABET: Let Keyword= "MUSTANSIRIYAH"**

M	U	S	T	A	N	I	R	Y	H	B	C	D	E	F	G	J	K	L	O	P	Q	V	W	X	Z

- **Write down the plaintext to be encrypted: HELLO**
- **Replace each letter in the Plaintext=" HELLO" with the corresponding letter in the new alphabet:**

H becomes R, E becomes A, L becomes C, L becomes C, O becomes F

The Resulting Ciphertext is "RACCF"**.**

NOTE: To decrypt the ciphertext, you simply use the inverse substitution rule to replace each letter in the ciphertext with its corresponding letter in the original alphabet.

PLAYFAIR CIPHER:

The **Playfair** Cipher is a type of **Polygraphic Substitution** cipher, which uses pairs of letters instead of single letters to create the ciphertext. The best-known multiple-letter encryption cipher is the Playfair, which treats diagrams in the plaintext as single units and translates these units into ciphertext diagrams. The Playfair algorithm is based on the use of a 5 * 5 matrix of letters constructed using a keyword.

1. First, we need to set up a key square or a key matrix, which is a 5x5 grid of letters used to encrypt the plaintext. The key square is usually created using a keyword, where all repeated letters are removed, and then the remaining letters are arranged in a 5x5 grid. In this example, we will use the keyword "LEMON":

L	**E**	**M**	**O**	**N**
A	B	C	D	F
G	H	**I/J**	K	P
Q	R	S	T	U
V	W	X	Y	Z

2. Write down the plaintext to be encrypted, and break it into pairs of letters. If there is an odd number of letters, add an "**X**" at the end:

   HELLO becomes HE LX LO

3. For each pair of letters, apply the following encryption rules:
   a. If the two letters are the same, insert an "X" between them.
   b. If the two letters appear in the same row of the key square, replace each letter with the letter to its right (wrapping around to the beginning of the row if necessary).
   c. If the two letters appear in the same column of the key square, replace each letter with the letter below it (wrapping around to the top of the column if necessary).
   d. If the two letters form a rectangle, replace each letter with the letter in the same row and opposite corner of the rectangle.

4. Applying these rules to the plaintext pairs, we get the following ciphertext pairs:
   HE becomes RB, **LX becomes MV,** **LO becomes EN**
   The resulting ciphertext is " RBMVEN".

NOTE: To decrypt the ciphertext, you simply apply the inverse of the encryption rules to each pair of letters in the ciphertext using the same key square. In this case, " **RBMVEN** " would be decrypted to "HELLO" by replacing each pair of letters in the ciphertext with the corresponding pair of letters in the plaintext, as determined by the encryption rules used during encryption.
https://planetcalc.com/7751/

ONE TIME PAD ENCRYPTION (SUBSTITUTION- POLY-ALPHABETIC CIPHER):

One-time pad is a type of encryption where each bit or character of the plaintext is combined with a corresponding bit or character from a secret random key using the XOR operation. The key is at least as long as the plaintext and is only used once, hence the name "one-time pad".

The resulting ciphertext appears completely random and does not reveal any information about the plaintext. This makes one-time pad encryption theoretically unbreakable, as long as the key remains secret and is used only once.

- **Alphabet** such as ABCDEFGHIJKLMNOPQRSTUVWXYZ OR can be Customized.
- **Generate a key:** The key should be a random string of the same length as the plaintext. Each character in the key should be chosen uniformly and independently from the set of possible characters.
- **Convert the plaintext and key into binary:** The plaintext and key should be converted into binary strings using some fixed encoding, such as ASCII or Unicode.
- **Perform the XOR operation:** The binary plaintext and key should be XORed together bit-by-bit to produce the ciphertext. The XOR operation is performed as follows:
- **Convert the ciphertext back into a text format:** The resulting ciphertext should be converted back into a text format using the same encoding as the plaintext.

Here is an example of how this algorithm would work using the **plaintext** "HELLO" and a randomly generated **key** "KRGJK":

- Generate a **key**: "KRGJK" and in Hexa=”4B 52 47 4A 4B” therefore in binary:
  01001011 01010010 0100 0111 01001010 01001011
- Convert the **plaintext** into binary:
  Plaintext: 01001000 01000101 01001100 01001100 01001111
  Key: 01001011 01010010 0100 0111 01001010 01001011
- Perform the **XOR** operation:
  Plaintext XOR Key: 00000011 00010111 00001011 00000110 00000100
- Convert the ciphertext back into a text format:
  Ciphertext: 00000011 00010111 00001011 00000110 00000100
  Encoded: '\x03\x17\x0B\x06\x04'=1p°`

Therefore, the resulting ciphertext is '\x03\x17\x0B\x06\x04'. This ciphertext can only be decrypted using the original key, which is known only to the sender and receiver.

https://xor.pw/#
https://codebeautify.org/text-to-binary
https://www.rapidtables.com/convert/number/binary-to-ascii.html

AFFINE CIPHER (SUBSTITUTION- **MONOALPHABETIC** CIPHER):

The Affine Cipher is a type of monoalphabetic substitution cipher that uses mathematical operations to encrypt and decrypt messages. It involves two keys: a multiplicative key (a number) and an additive key (another number). Here's an example of how the Affine Cipher works step by step:

1. **S/R Agree about Alphabet such that:**

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

2. **Choose the two keys**: a multiplicative key **(a)** and an additive key **(b)**. For example, let's choose **a = 5** and **b = 8**.
3. **Choose a message to encrypt**. Let's choose the message **"HELLO WORLD".**
4. **Convert each letter in the message to a number**, using a standard numerical mapping. For example, A=0, B=1, C=2, and so on. Using this mapping, the message "**HELLO WORLD**" becomes: **7 4 11 11 14 22 14 17 11 3**
5. **Apply the encryption formula to each number in the message: C = (a * P + b) mod 26** where C is the encrypted number, P is the plaintext number, and mod 26 ensures that the result is always between 0 and 25.
6. Using the keys from step 1, the encryption formula becomes: C = (5 * P + 8) mod 26
7. Applying this formula to each number in the message, we get: **17 22 3 3 2 12 2 15 3 23**
8. Convert the encrypted numbers back into letters, using the same numerical mapping. For example, 0=A, 1=B, 2=C, and so on.
9. Using this mapping, the encrypted message becomes: **Cipher Text= R W D D C M C P D X**

NOTE: Affine Decryption can be computed by the following formula:

Plain Text = (**a^-1** * (C - b)) mod 26, so for a=5, the **a^-1** =21

HOMEWORK:

- FIND PLAINTEXT FOR ABOVE CIPHERTEXT?
- WRITE A PROGRAM TO FIND **a^-1** TO ANY VALUE OF **a**?
- FIND CIPHERTEXT OF MESSAGE "CYBERSECURITY" WHERE **a=3 AND b=5?**

TRANSPOSTION ENCRYPTION:

SDES (Simple Data Encryption Standard) is a symmetric-key encryption algorithm that was

SDES (SIMPLE DATA ENCRYPTION STANDARD):

SDES (Simple Data Encryption Standard) is a symmetric-key encryption algorithm that was designed to be a simpler and more efficient version of the widely used Data Encryption Standard (DES) algorithm. SDES works by taking a plaintext message and transforming it into a ciphertext message using a secret key.

The SDES algorithm operates on:

- **8-bit** blocks of **plaintext** and
- Uses a **10-bit key,** the key is used to perform **two rounds of substitution and permutation** operations on the plaintext,
- Resulting in a ciphertext that is then sent securely over the network.

Let's say we want to encrypt the 8-bit message "11010010" using the 10-bit key "1010011010". Here are the steps we would follow:

First, we need to expand the 8-bit message to 10 bits using the SDES expansion function. The expansion function takes the input bits and produces an output with more bits using a fixed permutation table. The table for SDES is:

4 1 2 3 2 3 4 1

So, to expand our 8-bit message, we would copy the second bit to the first position, the fourth bit to the second position, and so on, resulting in a 10-bit value of "0110100010".

Next, we need to perform the key generation process to create two 8-bit subkeys. The key generation process involves permuting and shifting the 10-bit key in a specific way. The exact process is:

Apply the P10 permutation to the 10-bit key. This rearranges the bits in a fixed order.
Split the permuted key into two halves of 5 bits each: C0 and D0.
Apply two different left shifts to each of the halves, resulting in C1 and D1.
Combine C1 and D1 into a 10-bit value and apply the P8 permutation to generate the first subkey K1.
Apply another left shift to each of C1 and D1 to get C2 and D2, and then combine them and apply P8 to get the second subkey K2.
Using our example key "1010011010", the key generation process would produce the subkeys:

K1 = 10110010
K2 = 01011010

Now we can begin the encryption process. We start by dividing the 10-bit expanded message into two halves of 5 bits each: L0 and R0. In our example, this gives:

L0 = 01101
R0 = 00010

We then perform two rounds of a function called the F function. The F function takes an 8-bit input and a 8-bit subkey, and produces an 8-bit output. The exact process for the F function is:

Expand the 8-bit input to 10 bits using the same expansion function as before.
XOR the expanded input with the 8-bit subkey.
Divide the result into two halves of 4 bits each: S1 and S2.
Look up the values of S1 and S2 in two fixed substitution tables. Each substitution table takes 4 bits as input and produces 2 bits as output.
Combine the outputs of the substitution tables into an 8-bit value using another fixed permutation table.
Using our example message and subkey, the first round of the F function would be:

Expand R0 to get "0010000001"
XOR the expansion with K1 to get "1001001011"
Divide the result into S1 = 1001 and S2 = 0011
Look up the values of S1 and S2 in the substitution tables. This gives us 9 and 10, which we combine to get "10001010" using the permutation table.
The second round of the F function is similar, but instead of using K1 as the subkey, we use K2