CSE 4743 Computer Security
Mid Term

Course Content

1	Computer Security Concepts, Security Attacks, Security Services, Security Mechanism, A Model for Network Security.
2	Symmetric Cipher Model, Substitution Techniques, Transposition Techniques, Steganography, Block Ciphers and the Data Encryption.
3	Euclid's Algorithm, Placement of Encryption Function, Traffic Confidentiality, key distribution.

Segment 1

1. What is computer security? Explain confidentiality, integrity and availability with suitable example.
2. Describe some challenges of computer security.
3. Describe OSI Architecture
4. Active attack vs Passive attack
5. Describe security mechanism
6. Describe Security Services with example.

Segment-2

1. Symmetric Cipher and Asymmetric cipher definition and working principle, difference and Examples.
2. One time Pad + Vigenère Cipher
3. Cryptanalysis and Brute force attack definitions
4. Caesar Cipher/Shift cipher
5. Transpositional Cipher
6. Stream cipher vs Block cipher

Segment (Segment 3)

1. Describe key distribution Scenario
2. Find GCD and flow chart.
3. AES design

Computer Security Study Guide - Segment 1

1. Computer Security Concepts

What is Computer Security?

Computer security is about protecting automated information systems. The main goal is to keep the system's resources safe. These resources include the physical parts (hardware), programs (software and firmware), and the information or data itself, as well as how they communicate (telecommunications). It means guarding these things from anyone using them or seeing them without permission, or from being damaged or destroyed.

Key Goals of Computer Security (The CIA Triad)

There are three main goals in computer security, often called the CIA triad:

- **Confidentiality:** This means keeping secrets secret.
  - **Data confidentiality:** Making sure private information is not shown or given to people who are not allowed to see it.
  - **Privacy:** Allowing individuals to control what information is collected about them, who stores it, and who it is shared with.
- **Integrity:** This is about making sure information and systems are correct and have not been improperly changed.
  - **Data integrity:** Making sure information and programs are changed only in ways that are planned and allowed.
  - **System integrity:** Making sure the system works the way it is supposed to, without being wrongly changed on purpose or by accident.

- **Availability:** This means that the systems and information are there and ready to be used when authorized people need them. The more important a system or service is, the more available it needs to be.

Other Important Concepts
Besides the CIA triad, two other important security ideas are:
- **Authenticity:** Making sure something is real and can be trusted.
- **Accountability:** Making sure that if someone does something, we can clearly find out who did it.

2. Challenges in Computer Security

Computer security can be difficult because:
- It's not as easy as it looks at first.
- When creating security features, you have to think about how attackers might try to break them.
- Because of this, the ways we secure things can sometimes seem strange or not what you would expect.
- After creating security tools, you need to figure out the best places to use them.
- Security often involves more than just a single technique; it requires having some secret information.
- Computer and network security is like a competition between attackers trying to find weak points and defenders trying to fix them. Attackers only need to find one weakness, but defenders must eliminate all of them for perfect security.
- Sometimes, security is only thought about after a system is already designed, instead of being a core part of the design from the beginning.

3. OSI Architecture of Security

The Open Systems Interconnection (OSI) Security Architecture, defined by ITU-T Recommendation X.800, provides a systematic framework for understanding and implementing security in network communications. It aligns security considerations with the seven layers of the OSI model, emphasizing that security is not a single solution but rather a multifaceted effort requiring attention at various stages of data transmission and processing.

The architecture centers around three key concepts:
- **Security Attacks:** Actions that compromise the security of information. These can be passive (monitoring communications, such as traffic analysis) or active (modifying or disrupting communications, such as denial-of-service or masquerading).
- **Security Mechanisms:** Processes or tools designed to detect, prevent, or recover from security attacks. Examples include encryption, digital signatures, and access control.
- **Security Services:** Processing or communication services that enhance the security of data processing systems and information transfers. These services utilize one or more security mechanisms to counter security attacks. Examples include authentication, confidentiality, and integrity.

The strength of the OSI Security Architecture lies in its layered approach, recognizing that vulnerabilities and appropriate security measures differ at each level of the network stack. While security services can be requested by or provided to any layer, the mechanisms to implement them are often specific to particular layers.

A **Threat** is a possible danger that could use a weakness to cause harm. It's a potential reason for security to be violated. An **Attack** is a purposeful action by someone trying to break security.

4. Security Attacks

Security attacks are actions that put an organization's safety at risk. They are divided into two main types:
- **Passive Attacks:** These are like secretly listening in on communications without changing them. The attacker just wants to get information. The sender and receiver usually don't know an attack is happening.

    - **Release of Message Contents:** The attacker listens to get the secret information in a message, like in a phone call or email.
    - **Traffic Analysis:** Even if the message content is hidden (like with encryption), the attacker can still watch the communication patterns, like who is talking to whom, how often, and how much data is being sent. This information might help them guess what the communication is about.
- **Active Attacks:** These attacks involve the attacker actively changing or interrupting the communication or system. The goal is usually to cause damage or disruption. The message doesn't stay in its original form.
    - **Modification of Message:** The attacker changes the message being sent, which can make it look wrong or disrupt the communication.
    - **Denial of Service (DoS):** The attacker floods a system or network with too much traffic to make it unavailable to normal users.
    - **Masquerade:** Someone pretends to be someone else to get unauthorized access to a system.
    - **Replay:** The attacker captures a message and sends it again later to cause a bad effect.

5. Security Mechanism

Security mechanisms are used to protect data and systems. They are divided into two types:

- **Specific Security Mechanisms:** These are used for particular layers or protocols in a network.
    - **Encipherment:** Using algorithms to scramble data so only someone with the right key can read it. This is like locking data in a box.
    - **Digital Signature:** Adding extra data or changing the original data so the receiver can be sure who sent it and that it hasn't been changed. It's like a virtual signature.
    - **Access Control:** Rules that decide who is allowed to access which resources.
    - **Data Integrity:** Ways to make sure data doesn't change during sending.
    - **Authentication Exchange:** Checking the identity of a user or system by exchanging information.
    - **Traffic Padding:** Adding extra data to hide the real message size and make traffic analysis harder.
    - **Routing Control:** Choosing secure paths for data and changing them if there are security threats.
    - **Notarization:** Using a trusted third party to confirm details about a data exchange.
- **Pervasive Security Mechanisms:** These work across different layers of the network and are not tied to just one security service.
    - **Trusted Functionality:** Parts of the system that are relied upon to work securely according to rules.
    - **Security Label:** A tag on data that shows its security level (like "secret" or "public").
    - **Event Detection:** Watching for things that are related to security, like someone trying to access something without permission.
    - **Security Audit Trail:** Recording information about security activities to check later that the system was working securely.
    - **Security Recovery:** Handling what happens after a security problem to get the system working normally again.

6. Security Services

Security services are provided to give specific protection to system resources. They help to counter security attacks. Here are some key security services:

- **Authentication:** Making sure that a communication is truly from the person or system it claims to be from. This stops people from pretending to be someone else.
    - **Peer Entity Authentication:** Checks the identity of both parties talking, especially when a connection starts or during the conversation. *Example: When you log in to a website, the website authenticates you, and sometimes your computer also authenticates to the website.*
    - **Data Origin Authentication:** Confirms that a message you received really came from the claimed sender. *Example: Receiving a digitally signed email that verifies the sender.*
- **Access Control:** Controlling who is allowed to use certain systems or applications over a network. The system first checks your identity (authentication) and then decides what you can do based on your permissions. Example: Only letting registered students access the online course materials.

- **Data Confidentiality:** Keeping data secret from attackers who might try to listen in (passive attacks). This can protect all data between two users or just parts of a message. It also includes keeping traffic patterns secret. Example: Encrypting your WhatsApp messages so only you and the recipient can read them.
- **Data Integrity:** Making sure that the information sent arrives exactly as it was sent, without any unwanted changes, errors, or loss.
  - **Connection-Oriented Integrity:** For continuous communication, this ensures messages are not duplicated, changed, reordered, or lost. It can also help against denial-of-service attacks. Example: Downloading a file and verifying its checksum to make sure it wasn't corrupted during download.
  - **Connectionless Integrity:** For single messages, this mainly checks that the message hasn't been changed. *Example: Receiving a single message and checking its digital signature to confirm it hasn't been tampered with.*
- **Nonrepudiation:** Making sure that neither the sender nor the receiver can falsely deny sending or receiving a message. The receiver can prove who sent the message, and the sender can prove the receiver got it. This creates trust and accountability. *Example: Online transactions often use nonrepudiation so neither the buyer nor seller can deny making the transaction.*

7. A Model for Network Security **(NOT in suggestion)**

Imagine a message being sent between two people over the internet. For this to be secure, the people sending and receiving need to work together using rules (protocols). Security is needed to protect the message from attackers who might try to steal or change it. There are two main parts to making a message secure:

1. **Transforming the message:** Changing the message so attackers can't understand it, for example, by using encryption to scramble it.
2. **Sharing secret information:** The two people need to share a secret, like a key for encryption, that the attacker doesn't know.

Sometimes, a trusted third party is also needed to help, like managing the secret information or sorting out disagreements about if a message is real. When designing a security service, there are four basic steps:

1. Create a method (algorithm) to do the security change on the information. This method should be strong enough that an attacker cannot easily defeat it.
2. Create the secret information (like a key) that will be used with the method.
3. Figure out how to share this secret information between the people who need it.
4. Define the rules (protocol) that the two people will follow to use the security method and the secret information to achieve the security goal.

Unwanted access and threats can include:

- ✓ **Information Access Threats:** Attacks where data is intercepted or changed to get unauthorized access.
- ✓ **Service Threats:** Attacks that use system weaknesses to stop or disrupt services for normal users, like viruses or worms.

Security mechanisms to counter these can include:

- **Gatekeeper Functions:** Things like asking for passwords or checking for viruses to stop unauthorized people.
- **Internal Controls:** Checking and finding unauthorized access or problems inside the system.

Computer Security Study Guide - Segment 2

1. What is Cryptography?

Definition of Cryptography

Cryptography is a method of protecting information and communications using codes. This makes sure that only the people who are supposed to receive the information can understand it. The word "cryptography" comes from "crypt" meaning "hidden" and "graphy" meaning "writing".

Features of Cryptography

Cryptography helps achieve several security features:

- **Confidentiality:** Only the intended person can access the information.
- **Integrity:** Information cannot be changed without being noticed.
- **Non-repudiation:** The sender cannot deny sending the information later.
- **Authentication:** The identities of the sender and receiver are confirmed.
- **Interoperability:** It allows secure communication between different systems.
- **Adaptability:** It changes over time to stay ahead of threats.

Types of Cryptography

There are different types of cryptography based on how keys are used:

- **Symmetric Key Cryptography:**
  - In this system, the sender and receiver use the *same* secret key to encrypt and decrypt messages.
  - It is usually faster and simpler than asymmetric cryptography.
  - However, the main challenge is safely sharing the secret key between the sender and receiver.
  - Examples include Data Encryption Standard (DES) and Advanced Encryption Standard (AES).
- **Asymmetric Key Cryptography:**
  - This system uses a *pair* of different keys for encryption and decryption.
  - There is a *public key* used for encryption and a *private key* used for decryption.
  - The public key can be known by anyone, but only the intended receiver has the private key to decrypt the message. The public and private keys are different.
  - The most popular example is the RSA algorithm.
- **Hash Functions:**
  - This type of algorithm does *not* use any key.
  - It calculates a fixed-length value (hash value) from the original text.
  - It's impossible to get the original text back from the hash value.
  - Hash functions are often used to encrypt passwords in operating systems.

2. Symmetric Cipher Model

- **Definition:** In Symmetric Key Cryptography, the sender and receiver use the *same* secret key to encrypt and decrypt messages.
- **Working Principle:**
  - A symmetric encryption system involves five main parts:
    - **Plaintext:** The original, understandable message.
    - **Encryption algorithm:** A set of rules that changes the plaintext into ciphertext using the secret key.
    - **Secret key:** A secret value known only to the sender and receiver, which the algorithm uses to make changes to the plaintext. A different key will result in a different outcome.
    - **Ciphertext:** The scrambled message produced by the encryption algorithm, which looks random and is not understandable on its own.
    - **Decryption algorithm:** The reverse of the encryption algorithm, which takes the ciphertext and the secret key and turns it back into the original plaintext.
  - For secure symmetric encryption, two things are necessary:
    - An attacker should not be able to decrypt the ciphertext or find the key, even if they have many examples of plaintext and their corresponding ciphertexts.
    - The sender and receiver must obtain and keep the secret key safe. If the key is discovered, anyone who knows the algorithm can read all communications using that key.
  - A key feature that makes symmetric encryption practical is that only the key needs to be kept secret, not the algorithm.

Difference from Asymmetric Cipher

- The main difference lies in the number of keys used. Symmetric Key Cryptography uses the *same* secret key for both encryption and decryption, while Asymmetric Key Cryptography uses a *pair* of different keys (a public key for encryption and a private key for decryption).

Examples

- Examples of Symmetric Key Cryptography include Data Encryption Standard (DES) and Advanced Encryption Standard (AES).

3. Asymmetric Cipher Model

- **Definition:** This system uses a pair of *different* keys for encryption and decryption. There is a public key used for encryption and a private key used for decryption.
- **Working Principle:**
  - The public key can be known by anyone, but only the intended receiver has the private key to decrypt the message.
  - The public and private keys are different.
  - To encrypt a message, the sender uses the receiver's *public* key.
  - To decrypt the message, the receiver uses their corresponding *private* key.

Difference from Symmetric Cipher

- The main difference is that Symmetric Key Cryptography uses the *same* secret key for both encryption and decryption, while Asymmetric Key Cryptography uses a *pair* of different keys (a public key for encryption and a private key for decryption).

Examples

- The most popular example of Asymmetric Key Cryptography is the RSA algorithm.

4. Attacks on Symmetric Ciphers

An attacker who sees the ciphertext but does not have the secret key or the original message might try to find the original message or the secret key (or both). Attackers usually know the encryption and decryption algorithms.

There are two main types of attacks on symmetric ciphers:

- **Cryptanalysis:** This attack uses the details of the encryption algorithm and possibly some knowledge about the original message (like common words or patterns) or pairs of original messages and their encrypted versions. The goal is to figure out the original message or the key being used.
- **Brute-force attack:** The attacker tries every possible key until they find the one that turns the ciphertext back into a readable message. On average, they need to try half of all possible keys to succeed.

An encryption method is called **unconditionally secure** if the encrypted message does not give enough information to figure out the original message, even if the attacker has a lot of encrypted text. The goal is usually to achieve **computationally secure** encryption, meaning it is so hard to break that the cost or time needed is too great compared to the value or useful life of the information. What makes brute-force attacks impractical is using an algorithm with a very large number of possible keys.

5. Substitution Techniques

Substitution techniques are one of the basic ways to encrypt. They work by replacing letters or symbols in the original text with other letters, numbers, or symbols.

- **Caesar Cipher:** The Caesar Cipher is the oldest and simplest way to encrypt text by substituting letters. Here's how it works:

- ✓ You pick a fixed number (this is your "shift" or "key").
- ✓ You replace each letter in your original message with the letter that is that many positions down the alphabet.
- ✓ For example, with a shift of 3, 'A' becomes 'D', 'B' becomes 'E', and so on. If you go past 'Z', you wrap back around to 'A'.

Why it's easy to break:

- ✓ This cipher isn't very secure because there are only 25 possible numbers you can use for the shift (since a shift of 26 brings you back to the start).
- ✓ An attacker can simply try every possible shift until they find the one that makes the message readable (a brute-force attack).

The formulas for this cipher are:

- ✓ Encryption: $C=(p+k)(mod26)$ (where C is the ciphertext letter, p is the plaintext letter, and k is the shift amount)
- ✓ Decryption: $p=(C-k)(mod26)$

- **Vigenere Cipher:** The Vigenère Cipher is a more complex way to substitute letters. Instead of shifting every letter by the same amount, it uses a repeating keyword. Each letter in your message is shifted by a different amount, based on the letter in the keyword that matches its position. Even though it's better than the simple Caesar cipher, you can still crack it by looking at how often letters appear. For instance, if your keyword is "cat" and your message starts with "hello", you'd shift 'h' by the amount for 'c', 'e' by the amount for 'a', 'l' by the amount for 't', and then repeat the keyword, shifting the next 'l' by the amount for 'c' again. While it's an improvement over the Caesar cipher, it can still be broken by analyzing the patterns of letters. The formulas used are $Ci=(pi+ki)(mod26)$ for encrypting and $pi=(Ci-ki)(mod26)$ for decrypting.

- **Vernam Cipher:** This system works with binary data (bits) instead of letters. It combines the plaintext bit (pi) with the key bit (ki) using the exclusive-or (XOR) operation to get the ciphertext bit (ci): $ci=pi \oplus ki$. Decryption uses the same operation: $pi=ci \oplus ki$.

- **One-Time Pad:** The One-Time Pad is a very secure encryption method, considered an improvement over the Vernam cipher.
  Here's how it works and why it's special:
  - ✓ It uses a key that is completely random.
  - ✓ This key is as long as the message you want to encrypt.
  - ✓ Crucially, the key is used only *one time* for a single message and is then thrown away.
  - ✓ It's considered unbreakable because the encrypted message looks totally random and doesn't give any clues about the original message.

  However, it has practical challenges:
  - ✓ It's difficult to create truly large amounts of random keys.
  - ✓ A big problem is safely getting a unique key (that's as long as the message) to both the sender and the receiver for every single message.

6. Transposition Techniques

Transposition ciphers are another basic way to encrypt. Instead of replacing letters, they change the *position* or order of the letters in the original text to create the encrypted text. The actual letters remain the same, just their arrangement is different.

- **Rail Fence Cipher:** This is a simple transposition cipher. The plaintext is written in a zig-zag pattern (like a fence) and then read row by row to get the ciphertext. For example, "meet me after the toga party" with 2 "rails" becomes:

m e t m a t r h t g p r y
e e f e t o a a t

Reading row by row gives: MEMATRHTGPRYETEFETEOAAT. This is very easy to break.

- **Columnar Transposition Cipher:** In this technique, the plaintext is written into rows in a grid, usually based on a keyword to determine the number of columns. The ciphertext is then read out column by column, in an order determined by the keyword. A pure transposition cipher can be recognized because the frequency of letters is the same as the original text. Cryptanalysis involves rearranging the columns.
- **Double Columnar Transposition Cipher:** This makes the columnar transposition more secure by applying the technique twice. The output from the first transposition becomes the input for the second transposition, using either the same or a different key. This results in a much more complex rearrangement that is harder to cryptanalyze.
- **Rotor Machine:** These machines implemented a complex, changing substitution cipher using multiple rotating cylinders. Each cylinder performs a substitution, and as they rotate with each keystroke, the overall substitution changes. Famous examples include the German Enigma and Japanese Purple machines used in World War II.

7. Cryptanalysis and Brute force attack definitions

These are two main ways attackers try to break symmetric ciphers.

Cryptanalysis:

a. This involves using details about the encryption algorithm, maybe some knowledge of the original message, or pairs of original and encrypted messages.
b. The goal is to figure out the original message or the secret key.

Brute-force attack:

a. The attacker tries *every single possible key* until they find the one that decrypts the message into something readable.
b. On average, they'll find the key after trying about half of all possibilities.

To make brute-force attacks impractical, encryption algorithms use a very large number of possible keys.

8. Steganography **(NOT in Suggestion)**

Definition of Steganography

Steganography is different from cryptography. Cryptography makes a message unreadable to outsiders by changing the text. Steganography, however, **hides the very existence** of the message. The hidden message looks like something else, not gibberish.

Methods of Steganography

Historically, different methods have been used:

- **Character marking:** Lightly marking certain letters in a printed text.
- **Invisible ink:** Writing with substances that only appear when heated or treated with chemicals.
- **Pin punctures:** Making tiny holes in selected letters.
- **Typewriter correction ribbon:** Typing between lines with this ribbon, visible only under strong light.

In the modern world, **digital steganography** is used. This involves hiding files, messages, images, or videos inside other digital files. Often, this is done by hiding the bits of the secret message within the least important bits of a "cover" file, like an image or sound file.

Disadvantages and Advantages

Compared to encryption, steganography has drawbacks:

- It often requires a lot of extra data ("overhead") to hide a small amount of information.
- Once the method is discovered, it becomes almost useless. This can be helped if the hiding method uses a key.

An advantage of steganography is that it can be used by people who need to hide the fact that they are communicating secretly. Encryption signals that the communication is important or secret, which might attract unwanted attention. A message can also be first encrypted and then hidden using steganography for greater security.

9. Stream Cipher vs Block Cipher

These are two main ways that symmetric encryption algorithms work to scramble data:

- **Stream Cipher:** Imagine encrypting a stream of data bit by bit or byte by byte as it flows through. That's what a stream cipher does. It processes the data in small pieces, one at a time.
- **Block Cipher:** Instead of processing data bit by bit, a block cipher takes a fixed-size chunk of data, called a "block" (like 64 or 128 bits), and encrypts that entire block all at once to produce a scrambled block of the same size. Block ciphers are generally used more widely for different types of applications compared to stream ciphers.

Think of it like this:

- **Stream Cipher:** Encrypting a conversation word by word as you speak it.
- **Block Cipher:** Encrypting a whole paragraph at once after you've written it.

Block Ciphers and Data Encryption Standard (DES)

Since creating an "ideal" block cipher that works perfectly for very large blocks isn't practical, a structure called the **Feistel cipher** is often used as a basis for designing block ciphers.

- **Feistel Cipher Structure:** This structure is like a recipe for building a block cipher. It works by repeatedly applying two basic steps in alternating rounds:
  - **Substitution:** Replacing parts of the data with other parts.
  - **Permutation (Transposition):** Rearranging the order of the data.
  - The Feistel cipher divides the block of data into two halves and mixes them up in each round using a function and a subkey.
- **Data Encryption Standard (DES):** This is an older example of a symmetric-key block cipher that uses the Feistel structure.
  - **How it works:** It takes a 64-bit block of plaintext and processes it through 16 rounds of substitutions and permutations based on a key.
  - **Key Size:** DES uses a 56-bit key.
  - **Security:** DES is *not* considered secure anymore because its key size (56 bits) is too small. Attackers can easily try every possible key (brute-force attack) to break it.
- **Double DES (2DES):** This was an attempt to make DES more secure by applying the DES algorithm twice using two different keys (K1 and K2). The idea was to increase the key size to 112 bits. However, it has a weakness called a "meet-in-the-middle" attack, which reduces its effective security significantly (only slightly better than single DES, around 57 bits). Because of this, 2DES is also not secure enough for today's needs.
  - **Encryption:** C=EK2(EK1(P))
  - **Decryption:** P=DK1(DK2(C))
- **Triple DES (3DES):** To provide better security than 2DES, 3DES applies the DES algorithm three times. This makes it much harder to break with brute force or the meet-in-the-middle attack, and it is still used in some applications today, though it is slower than newer algorithms like AES.

Computer Security Study Guide - Segment 3

1. Euclid's Algorithm (Finding GCD)

- **What it is:** Euclid's algorithm is a simple mathematical method to find the **Greatest Common Divisor (GCD)** of two positive whole numbers. The GCD is the largest number that can divide both of them exactly. For example, the GCD of 12 and 18 is 6.

- **Relatively Prime:** Two numbers are called "relatively prime" or "coprime" if their only common positive factor is 1. This means their GCD is 1. For example, 8 and 15 are relatively prime because their GCD is 1.
- **How it Works:** The algorithm uses a simple trick: the GCD of two numbers (a and b) is the same as the GCD of the smaller number (b) and the remainder you get when you divide the larger number (a) by the smaller one (b).
  - You keep dividing the last divisor by the last remainder until you get a remainder of 0.
  - The last remainder that was *not* zero is the GCD of your original two numbers.
- **Steps to find GCD of a and b ($a \geq b > 0$):**
  - ✓ Start with your two numbers, a and b.
  - ✓ Divide a by b and find the remainder, let's call it r1. (So, $a = q1 \times b + r1$)
  - ✓ If r1 is 0, then b is the GCD.
  - ✓ If r1 is not 0, now your new numbers are b and r1. Go back to step 2 and repeat the division.
  - ✓ Keep doing this ($b = q2 \times r1 + r2$, then $r1 = q3 \times r2 + r3$, and so on) until the remainder is 0. The GCD is the last remainder you got before the 0.
- **Example: Find gcd(710, 310):**
  - ✓ $710 = 2 \times 310 + 90$ (Remainder is 90)
  - ✓ $310 = 3 \times 90 + 40$ (Remainder is 40)
  - ✓ $90 = 2 \times 40 + 10$ (Remainder is 10)
  - ✓ $40 = 4 \times 10 + 0$ (Remainder is 0)
  - ✓ The last non-zero remainder was 10. So, gcd(710, 310) = 10.

Flow Chart for GCD (Euclid's Algorithm):
While the text doesn't provide a visual flowchart, the steps described can be easily represented in one:

1. Start
2. Input two numbers, a and b.
3. Make sure $a \geq b$ and $b > 0$. If not, swap a and b or handle the case where b=0.
4. Calculate the remainder r when a is divided by b ($r = a \pmod{b}$).
5. Is r=0?
   - If Yes, the GCD is b. Go to step 7.
   - If No, set a=b and b=r. Go back to step 4.
6. Output GCD (b).
7. End

2. Placement of Encryption Function (NOT in Suggestion)

When you encrypt data to protect its secrecy, you have to decide *where* in the network the encryption and decryption should happen. Attacks can happen at many points, especially in large networks you don't fully control.
There are two main places to put the encryption:

- **Link Encryption:**
  - Encryption is done at the beginning and end of *each link* (or connection) in the communication path.
  - This means all data traveling over each link is encrypted.
  - **Problem:** The message has to be decrypted at every network switch (like routers) to read the address information (in the header) so the switch knows where to send it next. This makes the message vulnerable for a moment at each switch.
  - If you're using a public network, you can't trust the security of all those switches.
  - Every pair of devices connected by a link needs a unique secret key, which can mean managing a lot of keys in a big network.
- **End-to-End Encryption:**
  - Encryption happens only at the very beginning (the sender's computer) and the very end (the final receiver's computer).

- The encrypted data travels through the network without being decrypted by the switches in between. Only the final receiver has the key to decrypt it.
- **Benefit:** This protects the data from attackers on the network links or at the switches. The end users don't have to worry about the security of the network path.
- **Problem:** Usually, only the actual data part of the message is encrypted. The header, which contains routing information needed by the switches, is left unencrypted. This means while the message content is secret, someone watching the network traffic can still see who is talking to whom, how often, and the message sizes (this is called **traffic analysis**).
- **Benefit:** End-to-end encryption can also help with **authentication** – confirming that the message really came from the expected sender because they are the only ones with the correct shared key. Link encryption doesn't provide this authentication from the original sender to the final receiver.

- **For Greater Security:** To get the best protection, you need to use both link and end-to-end encryption together.
  1. The sender encrypts the data part of the message using a key shared with the final receiver (end-to-end encryption).
  2. Then, the *entire* message (including the encrypted data and the unencrypted header) is encrypted *again* using a key shared with the next network switch (link encryption).
  3. As the message travels, each switch decrypts it using the link key for that specific link to read the header, then re-encrypts the whole message with the link key for the *next* link before sending it on.
  4. The message is secure everywhere except for the brief moment it's being processed inside a switch, where the header is temporarily readable.

3. Traffic Confidentiality **(NOT in Suggestion)**

- Even when you use end-to-end encryption, the message headers (like source and destination addresses) are usually not encrypted so that the network can route the message.
- This means an attacker can still watch the network traffic and see:
  - Who is communicating with whom?
  - How often they are communicating.
  - The size of the messages.
- This information about the *pattern* of communication can sometimes give attackers clues about what's being discussed, even if they can't read the message content.
- **Traffic confidentiality** is about keeping these traffic patterns secret. Link encryption helps protect traffic patterns on individual links, but as mentioned, the headers are exposed at the switches.

4. Key Distribution

- **What it is:** For symmetric encryption (where both sides use the same secret key) to work, the two people communicating *must* have the exact same secret key, and this key must be kept secret from everyone else. It's also good practice to change keys often. **Key distribution** is the process of safely getting that shared secret key to the two parties who need it, without anyone else getting a copy. It's a crucial part of symmetric encryption security.
- **Ways to Distribute Keys (between parties A and B):**
  - Party A physically carries or sends the key to Party B.
  - A trusted third party physically carries or sends the key to both A and B.
  - If A and B already shared a key before, one of them can encrypt the *new* key using the *old* key and send it. (This relies on one of the first two methods being used initially).
  - Using a **Trusted Third Party** (often called a **Key Distribution Center or KDC**): If both A and B have a secure way to talk to a trusted third party (C), C can send the key to both A and B using these secure connections.
- **Using a Key Distribution Center (KDC):** This is the most practical method when many people in a network need to communicate securely. It uses two types of keys:

- **Session Keys:** These are temporary keys used just for one specific conversation or connection between two end systems (like A and B). They are used for a short time and then discarded. You get these from the KDC.
- **Master Keys:** Each user or system (like A and B) has a unique, long-term secret key that they share *only* with the KDC. These master keys are used to encrypt the session keys when the KDC sends them out to the users.
- The KDC only needs to manage a master key for each user, which is much fewer keys than needing a unique key for every possible pair of users who might want to talk. Master keys can be set up initially using methods like physical delivery.

- **A Typical Key Distribution Scenario with a KDC:**
  a. User A wants to talk securely to User B. Both A and B have their own secret master keys (Ka and Kb) shared only with the KDC.
  b. A sends a message to the KDC asking for a session key (Ks) to talk to B. This message includes A's identity, B's identity, and a unique number (a "nonce") for this request to make sure it's not a repeated old request.
  c. The KDC gets the request, checks it, and creates a new session key (Ks) for A and B.
  d. The KDC sends a message back to A. This message is encrypted using A's master key (Ka) so only A can read it and know it's from the KDC. This message contains:
     ✓ The session key (Ks).
     ✓ A copy of A's original request (with the unique number) so A can verify the KDC got it right.
     ✓ Information for B (including the session key Ks and A's identity IDA), all encrypted using B's master key (Kb).
  e. A receives the message from the KDC, decrypts it with Ka, gets the session key Ks, and the encrypted part for B. A saves Ks and sends the encrypted part for B (E(Kb,[Ks||IDA])) to B.
  f. B receives the message from A, decrypts it with Kb, and gets the session key Ks and A's identity IDA. B now has the key, knows it's from A, and trusts it because it came securely from the KDC.
  g. Now A and B both have the session key (Ks) and can start their secure conversation.

- **Session Key Lifetime:** How long you use a session key before getting a new one is a trade-off.
  - Using keys for shorter times is more secure because attackers have less data encrypted with the same key to try and analyze.
  - However, getting new keys takes time and network resources. Security managers have to decide the right balance.

5. AES

What is AES?
The Advanced Encryption Standard (AES) is a widely used symmetric block cipher algorithm. It was adopted by the U.S. National Institute of Standards and Technology (NIST) in 2001 to replace the older Data Encryption Standard (DES).[1] As a symmetric cipher, it uses the same secret key for both encrypting plaintext into ciphertext and decrypting ciphertext back into plaintext.

Key Features of AES
- **Block Size:** AES operates on fixed-size blocks of 128 bits (16 bytes). This 128-bit block is often treated as a 4x4 array (matrix) of bytes during processing.
- **Key Sizes and Rounds:** AES supports three different key lengths: 128, 192, and 256 bits. The number of processing rounds in the algorithm depends on the key size used:
  - 128-bit key: 10 rounds
  - 192-bit key: 12 rounds
  - 256-bit key: 14 rounds Longer key sizes and more rounds provide a higher level of security.

- **Design Principle:** Unlike DES, which uses a Feistel network, AES is based on a **substitution-permutation network (SPN)**. This involves a series of linked operations that include replacing inputs with specific outputs (substitutions) and shuffling bits around (permutations).
- **Byte-Oriented Operations:** AES performs most of its computations on bytes rather than individual bits.

AES Encryption Process
The AES encryption process involves several steps that transform the 128-bit plaintext block into a 128-bit ciphertext block. The overall process includes an initial round key addition, a number of main rounds (depending on the key size), and a final round. Each main round consists of four different transformations, except for the final round which omits one of the steps.

The steps in each round (except the last one) are:
a. **SubBytes:** This is a non-linear substitution step where each byte in the 4x4 state matrix is replaced with another byte based on a fixed lookup table called an S-box. This introduces confusion.
b. **ShiftRows:** This is a transposition step where the rows of the state matrix are shifted cyclically to the left by a certain number of steps. The first row is not shifted, the second row is shifted one byte, the third row two bytes, and the fourth row three bytes. This step provides diffusion by moving bytes out of their original columns.
c. **MixColumns:** This is a linear mixing operation that operates on the columns of the state matrix. Each column is transformed using matrix multiplication in a specific finite field. This step further diffuses the data. This step is *not* performed in the final round.
d. **AddRoundKey:** In this step, each byte of the state matrix is combined with a byte from the round key using a bitwise XOR operation. A unique round key is used for each round, derived from the original cipher key through a process called the Key Expansion algorithm.

The initial step before the rounds is an `AddRoundKey` operation using the first round key. The final round consists of SubBytes, ShiftRows, and AddRoundKey.

AES Decryption Process
The decryption process in AES is similar to the encryption process but the steps are performed in reverse order, and inverse transformations are used (e.g., Inverse ShiftRows, Inverse SubBytes, Inverse MixColumns). The AddRoundKey step is its own inverse. Unlike a Feistel cipher, the encryption and decryption algorithms need to be implemented separately, although they are closely related.

Security of AES
AES is considered a highly secure encryption algorithm when implemented correctly. Its strength comes from its large key sizes, the number of rounds, and the use of substitution and permutation in each round to achieve diffusion and confusion. With current computational power, brute-force attacks are considered computationally infeasible for AES with 128, 192, or 256-bit keys.

Spring – 2023

1. Question:

a) Explain Confidentiality, Integrity and Availability with suitable example.

Confidentiality, Integrity, and Availability (CIA) are three fundamental goals of computer and information security. They are often called the CIA Triad.

a. **Confidentiality:**
   - ✓ **Explanation:** Confidentiality means ensuring that sensitive information is kept secret or private. It ensures that data is accessed only by authorized individuals and is protected from unauthorized viewing or disclosure.
   - ✓ **Example:** Student grade information in a university system should be confidential. Only the student, authorized university staff (like teachers and administrators), and perhaps parents (with permission) should be able to see the grades. Preventing others from accessing this information maintains confidentiality.

b. **Integrity:**
   - ✓ **Explanation:** Integrity means maintaining the accuracy, consistency, and trustworthiness of data. It ensures that information is not altered or tampered with in an unauthorized way, either accidentally or maliciously, during storage, processing, or transmission. Any modification should be done only by authorized users and through legitimate means.
   - ✓ **Example:** In an online banking system, the integrity of your account balance is crucial. The balance must accurately reflect all deposits and withdrawals. If someone unauthorized could change the balance figure, the integrity of the data would be lost, leading to incorrect financial information. Using techniques like hashing can help verify data integrity.

c. **Availability:**
   - ✓ **Explanation:** Availability means ensuring that systems, services, and data are accessible and usable by authorized users whenever they need them. It ensures that information and resources are readily available without undue delay.
   - ✓ **Example:** A hospital's patient record system must be available 24/7. Doctors and nurses need continuous access to patient history, allergies, and treatment plans to provide timely and effective care. If the system crashes or is inaccessible due to a cyber-attack (like a Denial-of-Service attack), it impacts availability and can compromise patient safety.

b) (OR)(a) Explain Access Control, Selective Field Data Integrity and Non-repudiation.

These are important security services used to protect information systems:

a. **Access Control:**
   - ✓ **Explanation:** Access Control is a security mechanism used to control who can access specific resources (like files, data, or system functionalities) and what actions they are allowed to perform (like read, write, or execute). It first verifies the identity of the user (authentication) and then checks their permissions (authorization) before granting access.
   - ✓ **Example:** In a university's online portal, a student can access their own grades and course materials, but they cannot access the grades of other students or administrative functions. This restriction is managed by access control rules.

b. **Selective Field Data Integrity:**
   - ✓ **Explanation:** Data Integrity ensures that data is accurate and has not been altered without authorization. Selective Field Data Integrity is a specific type of integrity protection where only certain chosen parts or 'fields' within a data message or record are protected against modification, instead of the entire data. This focuses protection on the most critical pieces of information.

- ✓ **Example:** When submitting an online payment form, selective field data integrity might be applied only to the 'Credit Card Number' and 'Amount Payable' fields to ensure these specific details are not tampered with during transmission, while other less critical fields might not have the same level of integrity check.

c. **Non-repudiation:**
   - ✓ **Explanation:** Non-repudiation is a service that provides proof that a particular action was taken by a specific party, preventing them from denying it later. It ensures that the sender of a message cannot deny sending it (proof of origin), and the receiver cannot deny receiving it (proof of delivery).
   - ✓ **Example:** When you send an important official email using a digital signature, the recipient has proof that the email genuinely came from you, and you cannot later deny having sent it. Similarly, if a recipient acknowledges receipt, the sender has proof it was delivered. This is crucial for contracts and financial transactions.

c) Describe some security mechanism that are included in ITU X.800 recommendation.

The ITU X.800 standard defines various security mechanisms used to protect data and systems during communication. These mechanisms are tools or processes designed to detect, prevent, or recover from security attacks.

X.800 divides these security mechanisms into two main categories:

a. **Specific Security Mechanisms:** These are applied to particular parts of the communication system, like specific protocols or layers. Examples mentioned in the documents include:
   i. **Encipherment:** This mechanism uses cryptographic algorithms to transform data into an unreadable format (ciphertext), ensuring confidentiality. Only those with the correct key can decipher the information.
   ii. **Digital Signature:** This mechanism uses cryptographic processes (often involving the sender's private key) to append data or transform existing data, allowing the receiver to verify the origin (authenticity) and integrity of the message, and protect against sender repudiation.
   iii. **Access Control Mechanisms:** These mechanisms enforce rules and policies to control who can access specific resources (like files, databases, or network services) and what actions they are permitted to perform (e.g., read, write, execute).
   iv. **Data Integrity Mechanisms:** These mechanisms are used to ensure that data has not been altered or corrupted during transmission or storage in an unauthorized manner. Techniques like checksums or cryptographic hash functions are often employed.
   v. **Authentication Exchange:** These mechanisms involve protocols used by communicating entities to prove their identities to each other. Examples include using passwords, challenge-response protocols, or digital certificates.
   vi. **Traffic Padding:** This involves adding dummy data to communication traffic to obscure actual data flow patterns, making traffic analysis by eavesdroppers more difficult.
   vii. **Routing Control:** This mechanism selects specific network routes dynamically or statically to ensure data travels through secure paths and avoids insecure or compromised network segments.
   viii. **Notarization:** This involves using a trusted third party to provide assurance regarding the properties of communicated data, such as its origin, integrity, or time of creation/delivery.

b. **Pervasive Security Mechanisms:** These mechanisms work across different parts and layers of the system and are not tied to a single security service. Examples include:
   i. **Trusted Functionality:** Parts of the system that are relied upon to act securely based on set rules.
   ii. **Security Label:** A tag on data indicating its security level (like "confidential").
   iii. **Event Detection:** Monitoring for events related to security, such as attempts to access things without permission.
   iv. **Security Audit Trail (Pervasive Mechanism):** This involves creating and maintaining logs of security-relevant events, which can be analyzed later to detect security breaches, investigate incidents, and ensure accountability.

v. **Security Recovery:** Handling the process of returning the system to normal operation after a security incident.

2. Question:

a) Explain the components of a Symmetric cipher model?

A Symmetric Cipher Model, also known as symmetric encryption or conventional encryption, uses a single shared key for both encrypting and decrypting information. It consists of the following five essential components:

a. **Plaintext:** This is the original, understandable message or data that needs to be protected. It is the input to the encryption process.
b. **Encryption Algorithm:** This is the mathematical procedure or set of rules used to transform the plaintext into ciphertext. The algorithm performs various substitutions and transformations on the plaintext. Examples include AES (Advanced Encryption Standard) and DES (Data Encryption Standard).
c. **Secret Key:** This is a piece of information (e.g., a string of bits) shared between the sender and the receiver and kept secret from others. It is a crucial input to both the encryption and decryption algorithms. The security of the entire process depends heavily on the secrecy of this key. The same key is used for both encryption and decryption.
d. **Ciphertext:** This is the scrambled, unreadable message produced as the output of the encryption algorithm. It is generated by applying the encryption algorithm and the secret key to the plaintext. Ciphertext can be transmitted over insecure channels without the original information being easily understood.
e. **Decryption Algorithm:** This is the reverse process of the encryption algorithm. It takes the ciphertext and the same secret key as input and transforms the ciphertext back into the original plaintext.

In summary, the sender uses the encryption algorithm with the shared secret key to convert plaintext into ciphertext. The ciphertext is then sent to the receiver, who uses the decryption algorithm with the *same* secret key to recover the original plaintext. The main challenge in this model is the secure distribution and management of the shared secret key.

b) Analyze the differences between cryptanalysis of Transposition cipher and Caesar cipher.

Transposition Cipher and **Caesar Cipher** are both classical encryption techniques, but they differ in how they manipulate plaintext. Here's an analysis comparing their cryptanalysis:

Feature	**Transposition Cipher Cryptanalysis**	**Caesar Cipher Cryptanalysis**
Cipher Type Attacked	Rearrangement (Permutation) of letters	Substitution of letters (fixed shift)
What is Analyzed?	The *position* and *order* of letters. Trying to find the pattern of mixing.	The *substitution* rule applied to each letter (the shift value).
Frequency Analysis (Single Letter)	Frequencies of letters in ciphertext are similar to original language (e.g., English). Does NOT directly reveal the key/pattern, but confirms it's likely a transposition.	Frequencies of letters are the *same* as original language, but *shifted*. Highly effective for finding the key (shift value).
Primary Attack Method	Guessing the transposition pattern (e.g., number of columns, key length), rearranging ciphertext, and looking for readable text/common letter groups. Anagramming approach.	Trying all possible shift values (Brute Force) or using frequency analysis to deduce the shift.
Key Space for Attacker	Depends on the method; can be related to permutations of columns (larger than Caesar, but still small by modern standards).	Very small (only 25 possible shifts in English alphabet).

Ease of Manual Breaking	Requires some structural guessing and rearranging; slightly more effort than Caesar.	Very easy and quick; can often be done by inspection or simple frequency count.
Overall Security (Classic)	More secure than Caesar, but still considered very weak.	Extremely weak; easily broken.
Goal of Cryptanalysis	To figure out the original order/pattern of the letters.	To figure out the amount of shift applied to the alphabet.

Example:

- **Caesar Cipher** with shift = 3:
  Plaintext: HELLO → Ciphertext: KHOOR
- **Transposition Cipher** (columnar, key = 3):
  Plaintext: HELLO → Rearranged: LEHOL (depends on key and pattern)

In simple terms, a **Caesar cipher changes *what* the letters are**, while a **Transposition cipher changes *where* the letters are**. Cryptanalysis of Caesar cipher is easier due to its limited key space, while transposition cipher requires analysis of character position and is slightly harder to break without knowing the key.

c) (OR) (b) i. Given the initial permutation "86574231" find the inverse initial permutation, show with a bit string example. ii. Given a 64-bit key, how would you derive the round key in DES for the 3rd round?

i. Given the initial permutation "86574231" find the inverse initial permutation, show with a bit string example.

The initial permutation (IP) "86574231" means that the bit that was originally at position 8 moves to position 1 in the permuted output, the bit from original position 6 moves to new position 2, the bit from original position 5 moves to new position 3, and so on. The last number, 1, means the bit from original position 1 moves to new position 8.

To find the *inverse* permutation (IP−1), we need to figure out the mapping that puts the bits back into their original spots. We can do this by asking: "Where did the bit at each *new* position originally come from?"

Let's list the mapping from Original Position -> New Position based on "86574231":

- Original 1 goes to New 8
- Original 2 goes to New 6
- Original 3 goes to New 5
- Original 4 goes to New 7
- Original 5 goes to New 4
- Original 6 goes to New 2
- Original 7 goes to New 3
- Original 8 goes to New 1

Now, let's reverse this to find the inverse mapping (New Position -> Original Position):

- Bit at New Position 1 came from Original Position 8.
- Bit at New Position 2 came from Original Position 6.
- Bit at New Position 3 came from Original Position 7.
- Bit at New Position 4 came from Original Position 5.
- Bit at New Position 5 came from Original Position 3.
- Bit at New Position 6 came from Original Position 2.
- Bit at New Position 7 came from Original Position 4.
- Bit at New Position 8 came from Original Position 1.

So, the inverse initial permutation is the sequence formed by these original positions: **86753241**.

Bit String Example:
Let's use an 8-bit string, for example: 10110010

1. **Apply the original permutation (86574231) to** 10110010**:**
   - New position 1 gets the bit from original position 8 (which is 0).
   - New position 2 gets the bit from original position 6 (which is 0).
   - New position 3 gets the bit from original position 5 (which is 0).
   - New position 4 gets the bit from original position 7 (which is 1).
   - New position 5 gets the bit from original position 4 (which is 1).
   - New position 6 gets the bit from original position 2 (which is 0).
   - New position 7 gets the bit from original position 3 (which is 1).
   - New position 8 gets the bit from original position 1 (which is 1).
   - The permuted string is: 00011011

2. **Apply the inverse permutation (86753241) to the permuted string (**00011011**):**
   - New position 1 gets the bit from the *current* position 8 (which is 1).
   - New position 2 gets the bit from the *current* position 6 (which is 0).
   - New position 3 gets the bit from the *current* position 7 (which is 1).
   - New position 4 gets the bit from the *current* position 5 (which is 1).
   - New position 5 gets the bit from the *current* position 3 (which is 0).
   - New position 6 gets the bit from the *current* position 2 (which is 0).
   - New position 7 gets the bit from the *current* position 4 (which is 1).
   - New position 8 gets the bit from the *current* position 1 (which is 0). The resulting string is: 10110010

This matches our original string, confirming that 86753241 is the correct inverse.

ii. Given a 64 bit key, how would you derive the round key in DES for the 3rd round?

Deriving the round keys in DES involves a series of steps applied to the initial 64-bit key. This is called the key schedule. Here's how you get the 3rd round key (K3):

a. **Start with the 64-bit Key:** This is the input to the key schedule.
b. **Permuted Choice 1 (PC-1):** The 64-bit key is processed by PC-1. This is a permutation table that also discards 8 bits (typically the 8th, 16th, ..., 64th bits, which were often used for parity). The output of PC-1 is a 56-bit key.
c. **Split the 56-bit Key:** The 56-bit key from PC-1 is split into two equal halves of 28 bits each. Let's call these C0 (the left 28 bits) and D0 (the right 28 bits).
d. **Iterative Left Circular Shifts:** For each round (from 1 to 16), the C and D halves from the *previous* step are subjected to a left circular shift. The number of bits shifted depends on the round number according to a fixed schedule in DES:
   - Rounds 1, 2, 9, and 16: Shift by 1 bit.
   - All other rounds (3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15): Shift by 2 bits.

   To get to the 3rd round key, we need to perform the shifts for rounds 1, 2, and 3:
   - C1, D1 are obtained by shifting C0, D0 left by 1 bit (for Round 1).
   - C2, D2 are obtained by shifting C1, D1 left by 1 bit (for Round 2).
   - C3, D3 are obtained by shifting C2, D2 left by **2 bits** (for Round 3).
e. **Concatenation:** The two halves C3 and D3 (each 28 bits) are concatenated together to form a 56-bit block.
f. **Permuted Choice 2 (PC-2):** This 56-bit block (C3||D3) is then processed by PC-2. This is another permutation table that selects 48 bits out of the 56 and permutes them. The 48-bit output of PC-2 is the round key K3 that will be used in the 3rd round of the DES encryption/decryption process.

So, starting from the 64-bit key, you first use PC-1, split, perform two 1-bit left shifts (for rounds 1 and 2) on the respective halves, then perform a 2-bit left shift (for round 3) on the halves, concatenate the results, and finally apply PC-2 to get the 48-bit round key K3.

3. Question:

a) Why do we need both End to end and Link encryption? Explain with example.

We need both End-to-end and Link encryption because they protect data at different points in its journey across a network, giving us stronger overall security. Each type covers weaknesses of the other.

a. **End-to-End Encryption (E2EE):**
   - **What it is:** Encrypts data right where it starts (the sender's device) and decrypts it only at the final destination (the receiver's device).
   - **What it protects:** The *content* of your message or data. Nobody in the middle, even if they intercept the data, can read what you sent because they don't have the key.
   - **What it *doesn't* hide:** The information about *who* is sending to *whom* (source and destination addresses in the packet header). Routers need this info to send the data to the right place.

b. **Link Encryption:**
   - **What it is:** Encrypts data as it travels over *each single link* (or "hop") between two network devices (like between your computer and the first router, or between two routers).
   - **What it protects:** The *entire packet* (including both the header and the data, which might already be end-to-end encrypted) while it is on that specific cable or wireless connection. It protects against someone "sniffing" the data directly off the wire or air on that link.
   - **What it *doesn't* protect:** The data when it is inside intermediate devices (like routers). At each hop, the data must be decrypted, the header read (to know where to send it next), and then the *entire packet* (with original headers and data) is re-encrypted using the key for the *next* link. This means intermediate devices can potentially see the data (if it's not also E2EE) and always see the headers.

Why Both are Needed:

- If you only use **Link Encryption**, your data is decrypted at every router or switch along the path. If any of these intermediate devices are compromised (hacked), your data can be read there. Also, link encryption often doesn't hide the header info from the devices themselves, only from outside eavesdroppers on the link.
- If you only use **End-to-End Encryption**, the content is safe, but the packet headers are visible to everyone along the path. This reveals who is communicating with whom, when, and how often, which can be sensitive information (traffic analysis).

Using Both:

When you use both, you get layered security:

- **E2EE** keeps your data *content* secret from everyone except the final recipient, even if intermediate devices are malicious.
- **Link Encryption** hides the *entire packet* (including the E2EE data and headers) from eavesdroppers on each physical link. This adds protection against traffic analysis on individual links and protects headers briefly while in transit between devices.

Example:

Imagine sending a confidential email from your laptop (Computer A) to a friend's laptop (Computer B), and the email goes through three routers (R1, R2, R3) on the internet.

a. Your email program (on Computer A) encrypts the email message itself using End-to-End encryption, so only your friend's email program (on Computer B) can read it. This encrypted message is put inside a network packet with header information (source=A, destination=B). The header is *not* encrypted by E2EE.
b. As the packet leaves your laptop to go to R1, your network card or a VPN client might apply Link Encryption. The *entire packet* (header + E2EE message) is encrypted for the link between A and R1.

c. R1 receives the packet, decrypts it using the Link A-R1 key. R1 can read the header (source A, destination B) to know where to send it next. The *original email content* is still encrypted because of the E2EE.
d. R1 then encrypts the *entire packet again* using a Link Encryption key for the next link (between R1 and R2) and sends it.
e. This process repeats: R2 decrypts (Link R1-R2), reads header, encrypts (Link R2-R3), sends. R3 decrypts (Link R2-R3), reads header, encrypts (Link R3-B), sends.
f. Finally, Computer B receives the packet. Link encryption (Link R3-B) is removed. The packet now has its original headers (source A, destination B) and the E2EE encrypted email message.
g. Your friend's email program on Computer B receives the packet and uses the End-to-End key to decrypt the email message, making it readable.

In this example, the email's content was protected by E2EE all the way. The packet (including the E2EE content and headers) was protected by Link Encryption on each segment of the journey, preventing eavesdropping on the links. The header was visible inside each router only long enough for routing decisions. This layered approach gives much better security.

b) Find the GCD of (450,120) with Euclidean algorithm.?

We will use the Euclidean algorithm to find the greatest common divisor (GCD) of 450 and 120. The algorithm involves repeatedly dividing the larger number by the smaller number and replacing the larger number with the1 smaller number and the smaller number with the remainder until the remainder is 0. The last non-zero remainder is the2 GCD.

Here are the steps to find gcd(450,120):

a. Divide 450 by 120:

$$450=3\times120+90$$

(The remainder is 90)

b. Now, we find the GCD of the previous divisor (120) and the remainder (90). Divide 120 by 90:

$$120=1\times90+30$$

(The remainder is 30)

c. Next, we find the GCD of the previous divisor (90) and the remainder (30). Divide 90 by 30:

$$90=3\times30+0$$

(The remainder is 0)

Since the remainder is now 0, the process stops. The last non-zero remainder was 30.
Therefore, the GCD of 450 and 120 is 30.

$$\gcd(450,120)=30.$$

c) Is Zq a Galois field? If yes, why is it? Show the additive inverse and multiplicative inverse for modulo 7 arithmetic.

Is Zq a Galois field? If yes, why is it?

Zq represents the set of integers $\{0,1,2,\ldots,q-1\}$ under addition and multiplication modulo q. Zq is a Galois field **IF AND ONLY IF q is a prime number**.

- If q is a prime number (like 2, 3, 5, 7, 11, etc.), then Zq is indeed a Galois field. We often write it as GF(q) in this case.
- If q is a composite number (like 4, 6, 8, 9, 10, etc.), then Zq is *not* a field.

Why is it a Galois field when q is prime?

A set with addition and multiplication operations forms a field if it satisfies certain properties (like being a commutative ring) AND, very importantly, **every non-zero element must have a multiplicative inverse**.

In Zq, a non-zero element a (where $1 \leq a < q$) has a multiplicative inverse modulo q if there exists an element b in Zq such that $a \times b \equiv 1 \pmod{q}$. This happens if and only if the greatest common divisor (GCD) of a and q is 1, i.e., $\gcd(a,q)=1$.

If q is a prime number, then for any number a from 1 to $q-1$, a is not a multiple of q. Since q has only two positive divisors (1 and q), the $\gcd(a,q)$ must be 1 for all $a \in \{1,2,\ldots,q-1\}$.

So, when q is prime, every non-zero element in Zq has a multiplicative inverse, fulfilling the crucial requirement for being a field. Since Zq is also a finite set, it is called a finite field or a Galois field.

If q is composite (e.g., q=4), consider the element 2 in $Z4=\{0,1,2,3\}$. $\gcd(2,4)=2$, which is not 1. There is no number b in Z4 such that $2 \times b \equiv 1 \pmod{4}$. (Check: $2\times0=0$, $2\times1=2$, $2\times2=0$, $2\times3=2$ modulo 4 - none are 1). Since a non-zero element (2) doesn't have a multiplicative inverse, Z4 is not a field.

Additive Inverse and Multiplicative Inverse for modulo 7 arithmetic (Z7)
$Z7=\{0,1,2,3,4,5,6\}$. Since 7 is a prime number, Z7 is a Galois field, GF(7).

Additive Inverse:
The additive inverse of an element a in Z7 is the element b such that $a+b \equiv 0 \pmod{7}$.

- Additive inverse of 0 is 0 (because $0+0 \equiv 0 \pmod{7}$)
- Additive inverse of 1 is 6 (because $1+6=7 \equiv 0 \pmod{7}$)
- Additive inverse of 2 is 5 (because $2+5=7 \equiv 0 \pmod{7}$)
- Additive inverse of 3 is 4 (because $3+4=7 \equiv 0 \pmod{7}$)
- Additive inverse of 4 is 3 (because $4+3=7 \equiv 0 \pmod{7}$)
- Additive inverse of 5 is 2 (because $5+2=7 \equiv 0 \pmod{7}$)
- Additive inverse of 6 is 1 (because $6+1=7 \equiv 0 \pmod{7}$)

Multiplicative Inverse:
The multiplicative inverse of a non-zero element a in Z7 is the element b such that $a \times b \equiv 1 \pmod{7}$.

- Multiplicative inverse of 1 is 1 (because $1\times1=1 \pmod{7}$)
- Multiplicative inverse of 2 is 4 (because $2\times4=8 \equiv 1 \pmod{7}$)
- Multiplicative inverse of 3 is 5 (because $3\times5=15 \equiv 1 \pmod{7}$)
- Multiplicative inverse of 4 is 2 (because $4\times2=8 \equiv 1 \pmod{7}$)
- Multiplicative inverse of 5 is 3 (because $5\times3=15 \equiv 1 \pmod{7}$)
- Multiplicative inverse of 6 is 6 (because $6\times6=36=5\times7+1 \equiv 1 \pmod{7}$)

Autmn – 2022

1. Question:

a) Explain how passwords are stored nowadays. Give examples of tradeoffs between usability and security considering "Authentication".

Answer: Nowadays, passwords are not stored directly in plain text due to security risks. Instead, they are stored using a secure method involving **Hashing** and **Salting**.

1. **Hashing:** When a user sets a password, it is processed by a one-way mathematical function called a hash function (like SHA-256, bcrypt). This function converts the password into a fixed-length string of characters called a hash. This hash is stored, not the original password. It's computationally very hard to get the original password back from the hash.
2. **Salting:** To make hashing even more secure and prevent attacks like using rainbow tables, a unique, random string called a "salt" is generated for *each* password. This salt is combined with the password before hashing (Salt + Password). The resulting hash, along with the unique salt, is stored. This ensures that even if two users have the same password, their stored hashes will be different because their salts are different.

When a user tries to log in, the system takes the password they enter, combines it with the stored salt for their account, hashes the combination, and compares the result to the stored hash. If they match, the password is correct.

Tradeoffs between Usability and Security in Authentication:
Authentication is the process of verifying a user's identity. When using passwords for this, there is a constant conflict between making the system easy for users (**Usability**) and making it resistant to attacks (**Security**).

- **Usability:** Refers to how convenient and easy the authentication method is for legitimate users.
  - *Example:* Allowing short, simple passwords (like "1234") is very easy for users to remember and type, offering high usability.
- **Security:** Refers to how well the authentication method protects against unauthorized access by attackers.
  - *Example:* Requiring long, complex passwords (e.g., 12+ characters with mixed cases, numbers, and symbols) is much harder for attackers to guess or crack, offering high security.

The Tradeoff:
Often, increasing security measures makes the system less usable:
- Requiring complex passwords makes them hard to remember, potentially leading users to write them down insecurely.
- Forcing frequent password changes can lead users to choose predictable patterns.
- Adding multi-factor authentication (like an OTP after password) increases login steps, reducing convenience.

Conversely, prioritizing usability can reduce security:
- Simple passwords are easy for users but are highly vulnerable to brute-force attacks.
- Not implementing account lockout after failed attempts allows attackers unlimited guesses.

Therefore, system designers must find a balance to ensure authentication is secure enough to protect assets while remaining practical and manageable for users.

b) "Software and system security is all about managing risk." Do you agree with this statement? Why? Calculate Annualized Loss Expectancy (ALE) for each loss type. Calculate the probability of small ATM fraud is five times higher than that of large ATM fraud.

Answer: Yes, I strongly agree with the statement that software and system security is fundamentally about managing risk.

Why?
Security isn't just about stopping every single attack, which is often impossible. Instead, it's about protecting valuable assets (like data, systems, services) from potential threats by addressing vulnerabilities. This process is essentially risk management:

1. **Identify Risks:** We identify potential threats (e.g., hackers, malware, natural disasters) and vulnerabilities (e.g., software bugs, weak passwords, misconfigurations) that could harm our assets. A risk is the potential for loss or damage when a threat exploits a vulnerability.
2. **Assess Risks:** We analyze the likelihood (how probable is the threat exploiting the vulnerability?) and impact (how much loss or damage would occur?) of each identified risk.
3. **Treat Risks:** Based on the assessment, we decide how to handle the risk. This could involve:
   - **Mitigation:** Implementing security controls (like firewalls, encryption, access controls, security awareness training) to reduce the likelihood or impact. This is what most security work involves.
   - **Acceptance:** Deciding the risk is low enough or too costly to mitigate, and accepting the potential loss.
   - **Transfer:** Shifting the risk to another party, e.g., through insurance.

- **Avoidance:** Not engaging in the risky activity.

4. **Monitor and Review:** Risks change over time, so we continuously monitor the environment and review our security measures.

So, all the activities we do in security – patching systems, setting policies, using anti-virus, designing secure software – are ultimately actions taken to manage the risks to our systems and data, trying to reduce potential losses to an acceptable level.

Calculating Annualized Loss Expectancy (ALE):
Annualized Loss Expectancy (ALE) is a way to quantify risk in financial terms over a year. The formula is:
ALE = Single Loss Expectancy (SLE) × Annualized Rate of Occurrence (ARO)

Where:

- **SLE (Single Loss Expectancy):** The expected monetary loss from a *single* occurrence of a risk event. SLE is often calculated as: SLE = Asset Value × Exposure Factor (EF)
  - *Asset Value:* The financial value of the asset being harmed.
  - *Exposure Factor (EF):* The percentage of loss expected from the asset due to the risk event (e.g., 0.5 for 50% loss).
- **ARO (Annualized Rate of Occurrence):** The expected number of times the risk event will occur in one year. This is often derived from historical data or predictions (e.g., an event occurring once every 5 years has an ARO of 1/5 = 0.2).

To calculate ALE for *each* loss type mentioned in the question's implicit table (e.g., small ATM fraud, large ATM fraud, etc.), we would need the specific values for:

1. The **Asset Value** affected by that loss type.
2. The **Exposure Factor** or the direct **Amount** of loss for *one* incident of that type (which might be given directly as SLE).
3. The **Incidence** or **Annualized Rate of Occurrence (ARO)** for that specific loss type per year.

Since the actual table with "Loss type, Amount, Incidence" values is not provided in the question, we cannot perform the specific numerical calculation of ALE for each loss type. However, the method would be to determine or be given the SLE (or calculate it from Asset Value and Exposure Factor) and the ARO for each loss type, and then multiply SLE by ARO for each one.

Calculating the Probability of Small ATM Fraud:
The question states that the probability of small ATM fraud is five times higher than that of large ATM fraud.
Let:

- P(small fraud) be the probability of small ATM fraud.
- P(large fraud) be the probability of large ATM fraud.

The given relationship is:

$$\mathbf{P(\text{small fraud}) = 5 \times P(\text{large fraud})}$$

To calculate the actual numerical probability of small ATM fraud, we need to know the numerical value of P(large fraud). For example, if we were given that the probability of large ATM fraud is 0.02 (or 2% chance) per year, then:

$$\mathbf{P(\text{small fraud}) = 5 \times 0.02 = 0.10}$$

So, the probability of small ATM fraud would be 0.10 (or 10%).

Without the specific probability value for large ATM fraud from the problem statement or the implicit table data, we can only state the relationship and the method of calculation.

2. Question:

a) Explain the CIA triad with necessary examples. What is its significance?

Answer: The CIA triad is a fundamental model in information security. It represents the three main goals or pillars that security efforts aim to achieve to protect information and information systems. CIA stands for Confidentiality, Integrity, and Availability.

a. **Confidentiality:**
   - **Explanation:** This means ensuring that information is kept secret and is only accessible to people who are authorized to see it. It prevents unauthorized disclosure of data.
   - **Example:** Protecting student records so that only the student and relevant university staff can view their grades and personal information. Using passwords and access controls helps maintain confidentiality.

b. **Integrity:**
   - **Explanation:** This means ensuring that information is accurate, complete, and has not been modified or tampered with in an unauthorized way. It also means ensuring systems function correctly.
   - **Example:** In a hospital, a patient's blood type or allergy information must be accurate and must not be changeable by just anyone. Security measures ensure that only authorized medical personnel can update this critical data, and that the data remains correct.

c. **Availability:**
   - **Explanation:** This means ensuring that authorized users can access the information and systems when they need them. Systems and data must be operational and accessible.
   - **Example:** An online banking website needs to be available 24/7 so customers can check their balance or make transactions anytime. A Denial-of-Service (DoS) attack, which prevents users from accessing the service, is a threat to availability.

Significance of the CIA Triad:

The significance of the CIA triad is that it provides a simple, core framework for understanding the objectives of information security. Almost all security threats and controls can be mapped back to one or more aspects of the CIA triad.

- It helps in identifying what needs to be protected.
- It guides the design and implementation of security policies and technologies.
- It provides a basis for evaluating the effectiveness of security measures – are they protecting confidentiality, integrity, and availability?

By focusing on these three pillars, organizations can develop a comprehensive approach to protecting their valuable information assets from various threats. It's the widely accepted foundation of information security principles.

b) Write down an algorithm (i.e. pseudocode) to cryptanalyze a transposition cipher. You can consider a suitable ciphertext example if required.

Answer:

c) (OR)(b) Analyze and explain the strength of a onetime pad with any suitable key, plaintext, and ciphertext example.

Answer: The One-Time Pad (OTP) is considered the only cryptosystem that provides **perfect secrecy** and is theoretically **unbreakable**. Its extreme strength comes from three critical conditions:

1. **The key must be truly random.** It should be generated by a truly unpredictable process.
2. **The key must be at least as long as the message.**
3. **The key must be used only once (hence "one-time").**

Strength of the One-Time Pad:
The strength of OTP lies in the fact that the ciphertext produced is completely random and bears no statistical relationship to the original plaintext. Encryption is typically done by combining the plaintext with the key using the XOR operation (or addition modulo 2 for binary data, or addition modulo 26 for alphabets).

Because the key is truly random and used only once, for any given ciphertext, *every possible plaintext of the same length is equally likely* to have produced that ciphertext with some possible key. An attacker looking at the ciphertext gains absolutely no information about the original plaintext. There is no way to distinguish the "correct" plaintext from any other possible message of the same length.

This property is called **perfect secrecy**. It means that even with unlimited computing power, an attacker cannot break the cipher.

Example:
Let's use the example idea where a single ciphertext could correspond to multiple plaintexts. This demonstrates perfect secrecy because the ciphertext doesn't uniquely determine the plaintext.

Suppose we have the following ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

If an attacker intercepts this ciphertext, they might try different keys.
- Using **Key 1**: pxlmvmsydofuyrvzwctnlebnecvgdupahfzzlmnyih When this key is combined with the ciphertext using the OTP operation (e.g., XOR or modulo addition depending on the data representation), it results in **Plaintext 1**: mr mustard with the candlestick in the hall (This is a plausible sentence).
- Using **Key 2**: pftgpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt If the same ciphertext is combined with **Key 2**, it results in **Plaintext 2**: miss scarlet with the knife in the library (This is also a plausible sentence).

For an attacker who doesn't know the original key, they would see the ciphertext and could potentially find multiple different keys that decrypt the ciphertext into completely different, but equally believable, plaintexts. The ciphertext itself provides no information to tell the attacker which of these plaintexts was the original message. This inability to gain any information about the plaintext from the ciphertext is the core of OTP's perfect secrecy and makes it theoretically unbreakable, provided the three strict conditions are met.

3. Question:
a) Describe how communication is secured using end-to-end encryption.

Answer: End-to-end encryption is a method where the protection of data happens at the two end systems involved in the communication.

Here's how it secures communication:
- The sender, which is one end system (like a host or terminal), encrypts the data.
- The data, now in an encrypted form, is sent across the network.
- This encrypted data travels through the network without being changed by any intermediate devices like switches or routers.
- The final receiving system, the other end system, shares a secret key with the sender.
- Using this shared secret key, the receiving system is able to decrypt the data back into its original, understandable form.

This process secures the transmission of the data against attacks that might happen on the network links or at the intermediate switches. It means that even if someone intercepts the data while it's traveling through the network, they won't be able to understand the content because it's encrypted and they don't have the secret key. The end user doesn't need to worry about the security level of the network or links supporting the communication for the data content itself.

b) (OR)(A) Explain congruency with an example. Show additive and multiplicative inverses for modulo 7 arithmetic.

Answer:

c) (B) Briefly describe the logical and physical key retrieval methods.

Answer: Key retrieval (or distribution) methods are ways to get the secret key to the parties who need it for secure communication. These methods can be broadly categorized as physical or logical.

- **Physical Key Retrieval Methods:** These involve physically transferring the secret key from one party to another. This means someone or something physically carries the key data.
  - *Examples from documents:* One party selecting a key and physically delivering it to the other party. A trusted third party selecting the key and physically delivering it to both parties. The documents mention that physical delivery is the simplest method but is most applicable when the people involved can meet in person.
- **Logical Key Retrieval Methods:** These involve transmitting the secret key electronically over a communication channel. This usually requires the channel itself to be secure or that the key is encrypted during transmission using another key.
  - *Examples from documents:* One party transmitting a new key to another, encrypted using a previously and recently used key. Using a trusted third party (like a Key Distribution Center or KDC) that delivers a new key to both parties over encrypted links that they share with the KDC. This involves sending the key data as part of a message through the network.

THE END