1 –Introduction to information Security

- **Information security**

Information security is the practice of protecting information by mitigating information risks. It involves the protection of information systems and the information processed, stored and transmitted by these systems from unauthorized access, use, disclosure, disruption, modification or destruction. This includes the protection of personal information, financial information, and sensitive or confidential information stored in both digital and physical forms. Effective information security requires a comprehensive and multi-disciplinary approach, involving people, processes, and technology.

Information Security is not only about securing information from unauthorized access. Information Security is basically the practice of preventing unauthorized access, use, disclosure, disruption, modification, inspection, recording or destruction of information. Information can be a physical or electronic one. Information can be anything like Your details or we can say your profile on social media, your data on mobile phone, your biometrics etc. Thus Information Security spans so many research areas like Cryptography, Mobile Computing, Cyber Forensics, Online Social Media, etc.

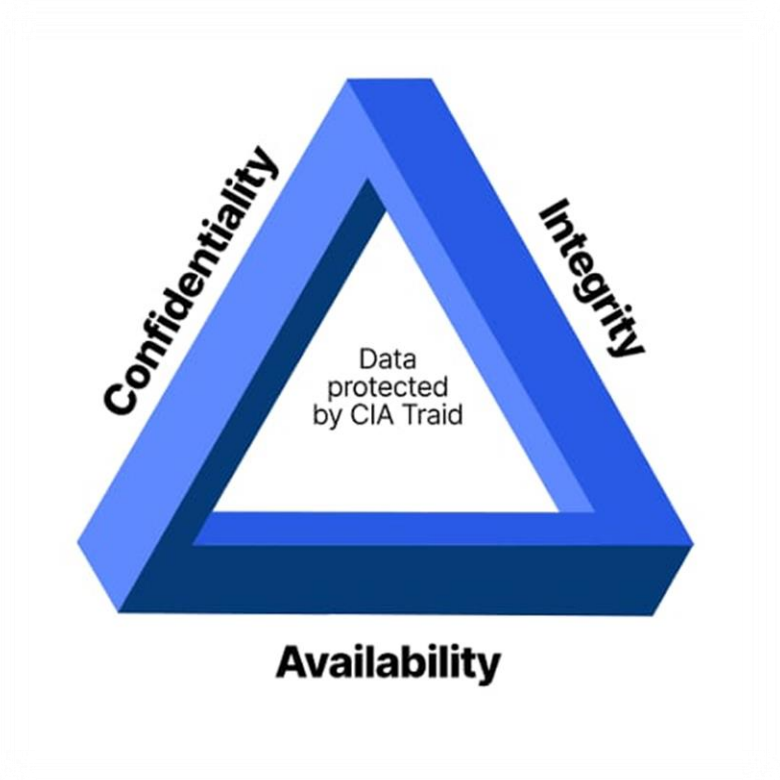
Why we use Information Security?

We use information security to protect valuable information assets from a wide range of threats, including theft, espionage, and cybercrime. Information security is necessary to ensure the confidentiality, integrity, and availability of information, whether it is stored digitally or in other forms such as paper documents. Here are some key reasons why information security is important:

- **Protecting sensitive information:** Information security helps protect sensitive information from being accessed, disclosed, or modified by unauthorized individuals. This includes personal information, financial data, and trade secrets, as well as confidential government and military information.
- **Mitigating risk:** By implementing information security measures, organizations can mitigate the risks associated with cyber threats and other security incidents. This includes minimizing the risk of data breaches, denial-of-service attacks, and other malicious activities.
- **Compliance with regulations**: Many industries and jurisdictions have specific regulations governing the protection of sensitive information. Information security measures help ensure compliance with these regulations, reducing the risk of fines and legal liability.
- **Protecting reputation:** Security breaches can damage an organization's reputation and lead to lost business. Effective information security can help protect an organization's reputation by minimizing the risk of security incidents.
- **Ensuring business continuity**: Information security helps ensure that critical business functions can continue even in the event of a

security incident. This includes maintaining access to key systems and data, and minimizing the impact of any disruptions.

Information Security programs are build around 3 objectives, commonly known as CIA – Confidentiality, Integrity, Availability.



- **Confidentiality** – means information is not disclosed to unauthorized individuals, entities and process. For example if we say I have a password for my Gmail account but someone saw while I was doing a login into Gmail account. In that case my password has been compromised and Confidentiality has been breached.
- **Integrity** – means maintaining accuracy and completeness of data. This means data cannot be edited in an unauthorized way. For

example if an employee leaves an organisation then in that case data for that employee in all departments like accounts, should be updated to reflect status to JOB LEFT so that data is complete and accurate and in addition to this only authorized person should be allowed to edit employee data.

- **Availability** – means information must be available when needed. For example if one needs to access information of a particular employee to check whether employee has outstanded the number of leaves, in that case it requires collaboration from different organizational teams like network operations, development operations, incident response and policy/change management. Denial of service attack is one of the factor that can hamper the availability of information.

Uses of Information Security :

Information security has many uses, including:

- **Confidentiality:** Keeping sensitive information confidential and protected from unauthorized access.
- **Integrity**: Maintaining the accuracy and consistency of data, even in the presence of malicious attacks.
- **Availability**: Ensuring that authorized users have access to the information they need, when they need it.
- **Compliance:** Meeting regulatory and legal requirements, such as those related to data privacy and protection.
- **Risk management**: Identifying and mitigating potential security threats to prevent harm to the organization.

- **Disaster recovery**: Developing and implementing a plan to quickly recover from data loss or system failures.
- **Authentication**: Verifying the identity of users accessing information systems.
- **Encryption:** Protecting sensitive information from unauthorized access by encoding it into a secure format.
- **Network security**: Protecting computer networks from unauthorized access, theft, and other types of attacks.
- **Physical security:** Protecting information systems and the information they store from theft, damage, or destruction by securing the physical facilities that house these systems

Issues of Information Security :

Information security faces many challenges and issues, including:

- **Cyber threats:** The increasing sophistication of cyber attacks, including malware, phishing, and ransomware, makes it difficult to protect information systems and the information they store.
- **Human error:** People can inadvertently put information at risk through actions such as losing laptops or smartphones, clicking on malicious links, or using weak passwords.
- **Insider threats**: Employees with access to sensitive information can pose a risk if they intentionally or unintentionally cause harm to the organization.

- **Legacy systems:** Older information systems may not have the security features of newer systems, making them more vulnerable to attack.
- **Complexity:** The increasing complexity of information systems and the information they store makes it difficult to secure them effectively.
- **Mobile and IoT devices:** The growing number of mobile devices and internet of things (IoT) devices creates new security challenges as they can be easily lost or stolen, and may have weak security controls.
- **Integration with third-party systems:** Integrating information systems with third-party systems can introduce new security risks, as the third-party systems may have security vulnerabilities.
- **Data privacy**: Protecting personal and sensitive information from unauthorized access, use, or disclosure is becoming increasingly important as data privacy regulations become more strict.
- **Globalization:** The increasing globalization of business makes it more difficult to secure information, as data may be stored, processed, and transmitted across multiple countries with different security requirements.

2 –Security Models and Policies

- **Security Policy**

A security policy could be a high-level document or set of documents that describes, in detail, the safety controls to implement in order to protect the corporate. It maintains confidentiality, availability, integrity, and asset values. A security policy also protects the corporate from threats like unauthorized access, theft, fraud, vandalism, fire, natural

disasters, technical failures, and accidental damage. Additionally, it protects against cyber-attack, malicious threats, international criminal activity foreign intelligence activities, and terrorism**.**

The following are the goals of security policies:

- To maintain an outline for the management and administration of network security
- To protect an organization’s computing resources
- To eliminate legal liabilities arising from workers or third parties
- To prevent wastage of company’s computing resources
- To prevent unauthorized modifications of the data
- To scale back risks caused by illegal use of the system resource
- To differentiate the user’s access rights
- To protect confidential, proprietary data from theft, misuse, and unauthorized disclosure.

1. Access Control Policy:

This policy outlines the procedures for controlling access to an organization's information systems and data. It specifies who is authorized to access certain data or systems and under what conditions. For example, a company may restrict access to certain confidential information to only authorized employees, who must use a two-factor authentication system to gain access.

2. Password Policy:

A password policy defines the requirements for creating and using passwords. It may specify password length, complexity, and expiration requirements. For example, a password policy may require passwords to be at least eight characters long, with a mix of uppercase and lowercase letters, numbers, and symbols, and to be changed every 90 days.

3. Acceptable Use Policy:

An acceptable use policy defines the appropriate use of an organization's information systems and data by employees, contractors, and other authorized users. It outlines the consequences of violating the policy, such as disciplinary action or termination. For example, a company may prohibit employees from using company-owned devices for personal use or accessing inappropriate websites.

4. Data Classification Policy:

A data classification policy categorizes an organization's data based on its sensitivity and sets appropriate levels of protection and handling procedures for each classification. For example, a company may classify its data as public, internal, confidential, and highly confidential, with corresponding security controls and access restrictions for each classification.

5. Incident Response Policy:

An incident response policy outlines the procedures for detecting, reporting, and responding to security incidents, such as data breaches or cyber attacks. It includes guidelines for containing and mitigating the damage, notifying affected parties, and preserving evidence for investigation. For example, a company may have a dedicated incident response team that is trained to respond to security incidents and has a well-defined plan for addressing them.

6. Physical Security Policy:

A physical security policy defines the measures for protecting an organization's physical assets, such as buildings, servers, and other equipment. It includes guidelines for securing facilities, controlling access to sensitive areas, and monitoring security cameras. For example,

a company may require employees to wear ID badges to access certain areas of the building and use biometric authentication to enter highly sensitive areas.

Access Control

Access control is a data security process that enables organizations to manage who is authorized to access corporate data and resources. Secure access control uses policies that verify users are who they claim to be and ensures appropriate control access levels are granted to users.

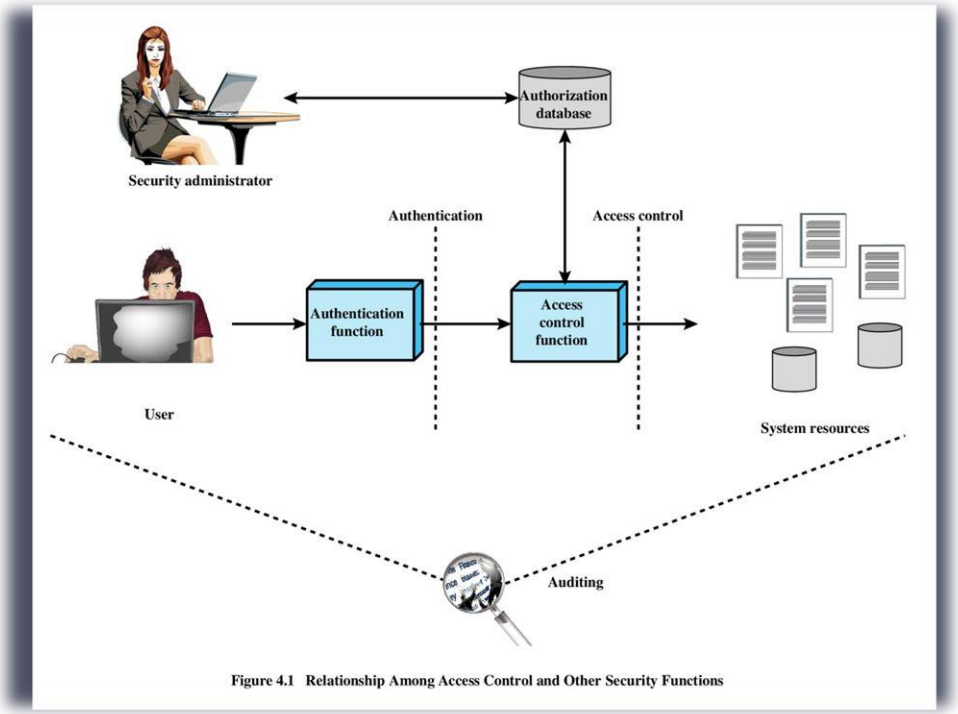


Figure 4.1 Relationship Among Access Control and Other Security Functions

What Are the Components of Access Control?

Access control is managed through several components:

1. Authentication

Authentication is the initial process of establishing the identity of a user. For example, when a user signs in to their email service or online banking account with a username and password combination, their identity has been authenticated. However, authentication alone is not sufficient to protect organizations' data.

2. Authorization

Authorization adds an extra layer of security to the authentication process. It specifies access rights and privileges to resources to determine whether the user should be granted access to data or make a specific transaction.

For example, an email service or online bank account can require users to provide two-factor authentication (2FA), which is typically a combination of something they know (such as a password), something they possess (such as a token), or something they are (like a biometric verification). This information can also be verified through a 2FA mobile app or a thumbprint scan on a smartphone.

3. Discretionary Access Control (DAC):

In DAC, access to a resource is controlled at the discretion of the resource owner. The resource owner can determine who can access the resource and what level of access they have. DAC is often used in environments where resources are shared among a small group of users, such as a home computer or a small office network. Examples of DAC mechanisms include file system permissions, access control lists (ACLs), and user groups.

4. Mandatory Access Control (MAC):

In MAC, access to a resource is controlled by a system-level security policy. The security policy specifies which users or processes are

authorized to access the resource, based on predefined security labels or classifications. MAC is often used in environments where the sensitivity of the data or resource is high, such as in government or military systems. Examples of MAC mechanisms include role-based access control (RBAC), attribute-based access control (ABAC), and label-based access control (LBAC).

Security Model

In information security, a security model is a conceptual framework that defines the principles and mechanisms used to protect a system or resource from unauthorized access, use, disclosure, disruption, modification, or destruction. A security model provides a structured approach to security that enables security professionals to design, implement, and manage effective security controls that align with the organization's security objectives.

There are several types of security models used in information security, including:

- **Bell-LaPadula**
- **Biba**
- **Clarke Wilson Security Model**
- **Brewer and Nash Model**

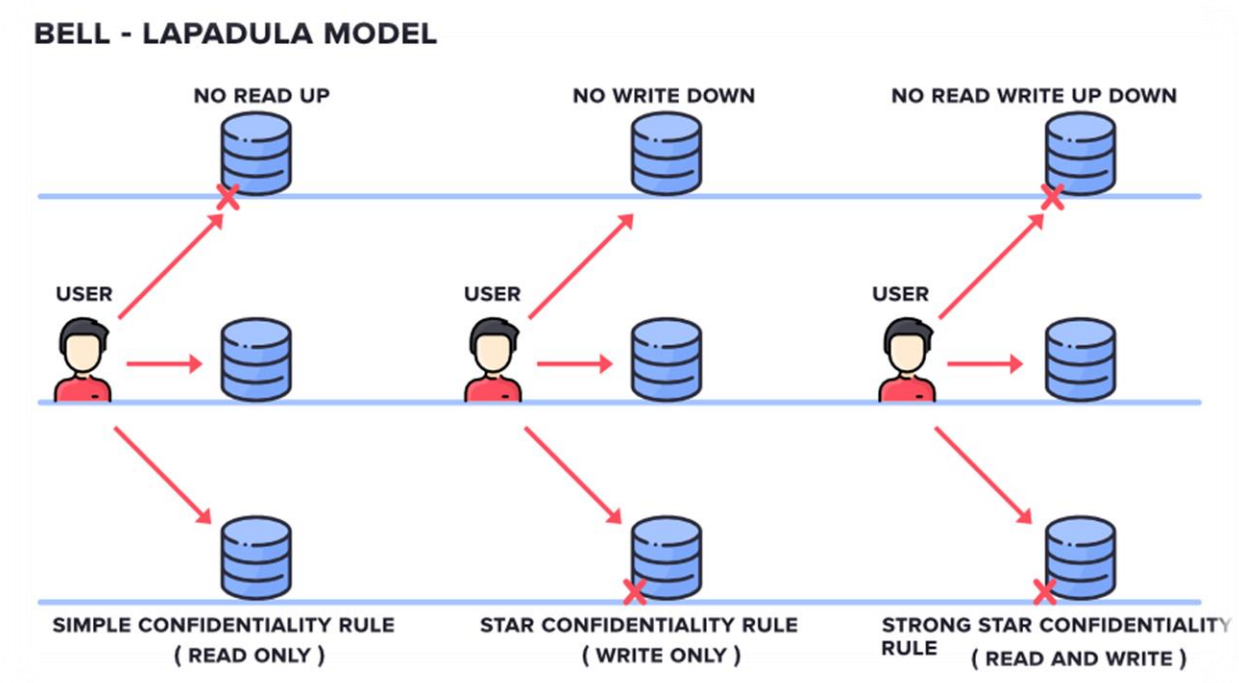
1. Bell-LaPadula

The ModelModel was created in the 1950s by Scientists David Elliot Bell and Leonard .J. LaPadula.Thus the ModelModel is known as Bell-

LaPadula Model. Bell-LaPadula Model. This ModelModel is used to protect the security of confidentiality. In this case, the classifications used to classify Subjects(Users) and Objects(Files) are arranged in a non-discretionary manner and about various layers of secret.

It has three primary rules:

- **SIMPLE CONFIDENTIALITY RULE**: The Simple Confidentiality Rule says that the Subject can read the files on the same Layer of Secrecy and the Lower Layer of Secrecy but not the Higher Layer of Secrecy because of this, this rule is known as No-Read-UP.
- **Star Confidentiality Rule 2**: This rule stated that the Subject is only able to write the document on the same layer of secrecy but not able to write in the lower Layer of Secrecy, and that is why we called this rule a No Write-down
- **The STRONG STAR CONFIDENTIALITY Rule**: Strong Star Confidentiality Rule is highly secure and robust that states that the Subject can read and write documents on the same Layer of Secrecy only, and not on the upper Layer of Secrecy or the Lower Layer of Secrecy This is why this rule is referred to as NO READ WRITE and DOWN**.**



2. Biba

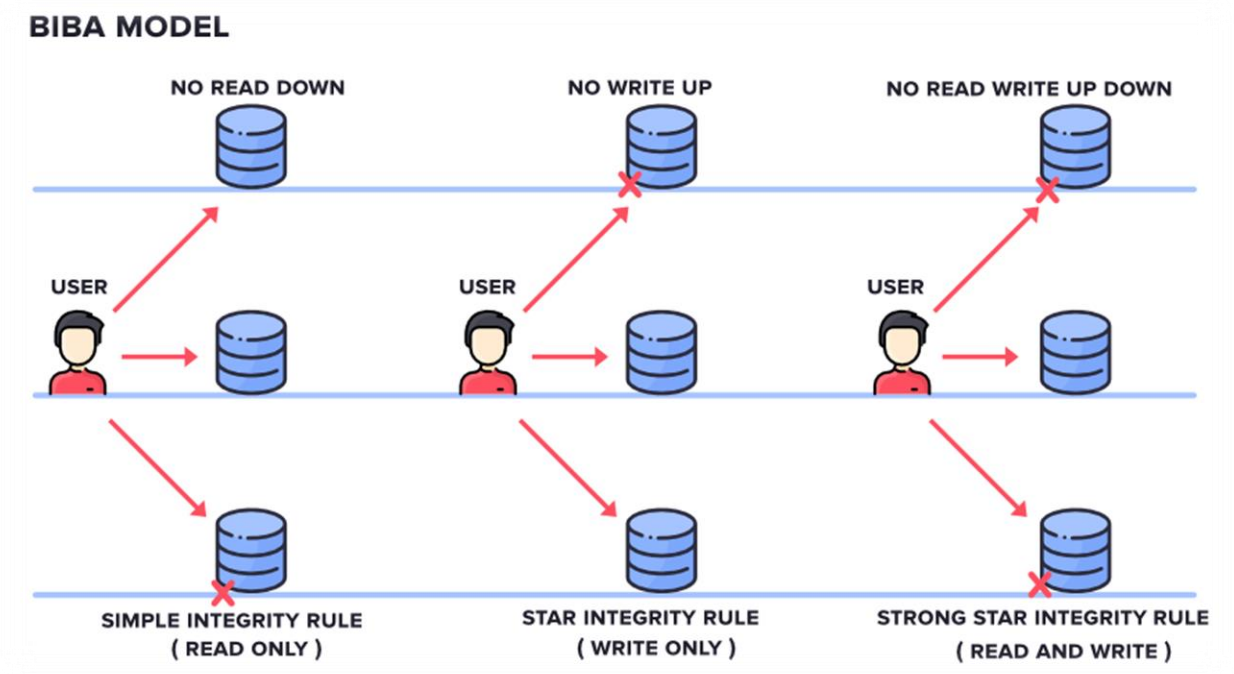
This Model was developed in the work of Scientist Kenneth .J. Biba. Therefore, this Model is known as Biba Model. This Model is used to safeguard security by ensuring Integrity in Security. The classifications used to classify Subjects(Users) and Objects(Files) are arranged in a non-discretionary way about various secret layers. This is the exact opposite to that of the Bell-LaPadula Model.

It is comprised of 3 Rules:

- **SIMPLE Integrity RULING:** According to this rule, the subject can read the file on the same and upper layer of secrecy but cannot

read the lower Layer of Secrecy; because of this, it is also known as NO-READ-Down.

- **Star Integrity Rule:** This rule states that the Subject can only write files that are on the same and the Lower Layer of Secrecy but cannot write on the upper Layer of Secrecy because of this, we called this rule a NO WRITE-UP
- STRONG STAR INTEGRITY RULE



3. Clarke Wilson Security Model

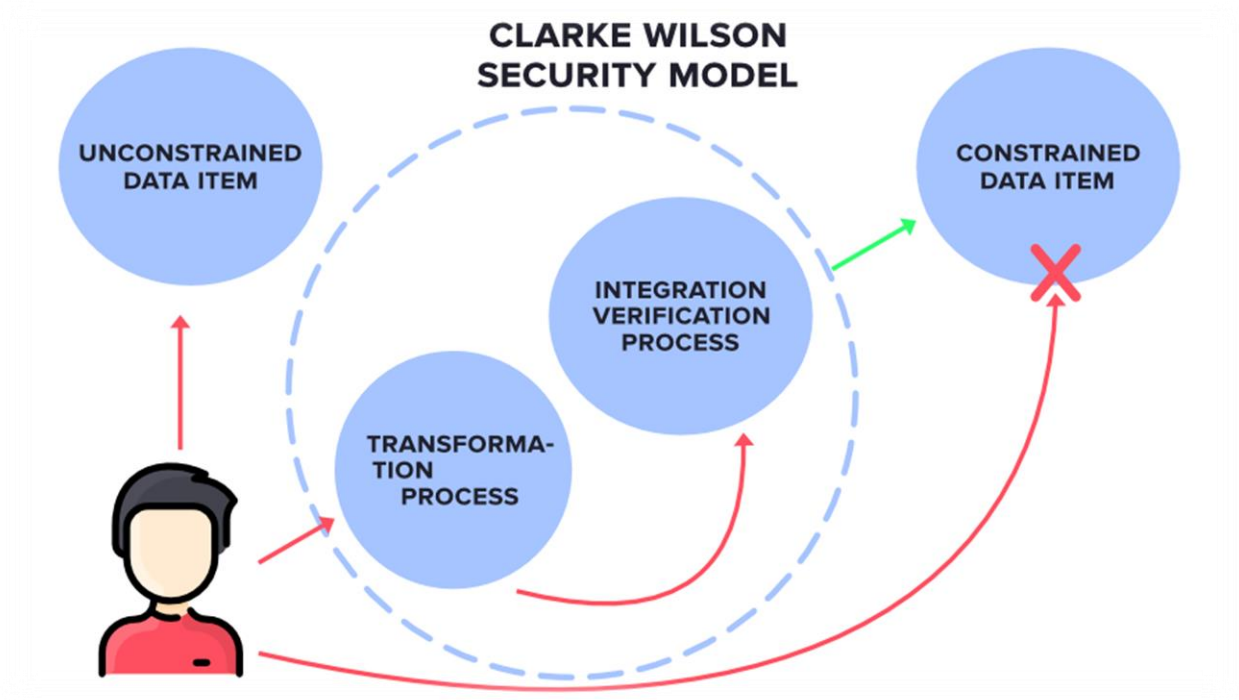
Model is a highly secure model that is highly secured. It includes the following elements.

- **SUBJECT:** It's any user who requests Data Items.
- **CONSTRAINED DATA ITEMS**: They are not accessible directly from the user. These must be accessible through the Clarke Wilson Security Model.

- **Unconstrained DATA ITEMS:** They can be directly accessed via the Subject.

The Components of Clarke Wilson Security Model

- **TRANSFORMATION PROCESS:** This is where the Subject's request to gain access to the constrained Data Items is processed via the Transform process, which transforms it into permissions and forwards it to the Integration Verification Process
- **Integration VERIFICATION Process**: Integration Verification Process will perform authentication and authorization. The Subject will be granted access to the restricted data items if the process succeeds.



4. Brewer and Nash Model

The ModelModel is also referred to as “the Chinese wall model. It can eliminate conflict of interest by preventing individuals, like consultants, from signing onto more than one COI, i.e., rows of interests categories. The modification of access control policies is based on the behavior of users. This means that if a user who has access to the data is on the other side, they cannot access data from the other side or are unavailable to the same user.

The Chinese Wall policy

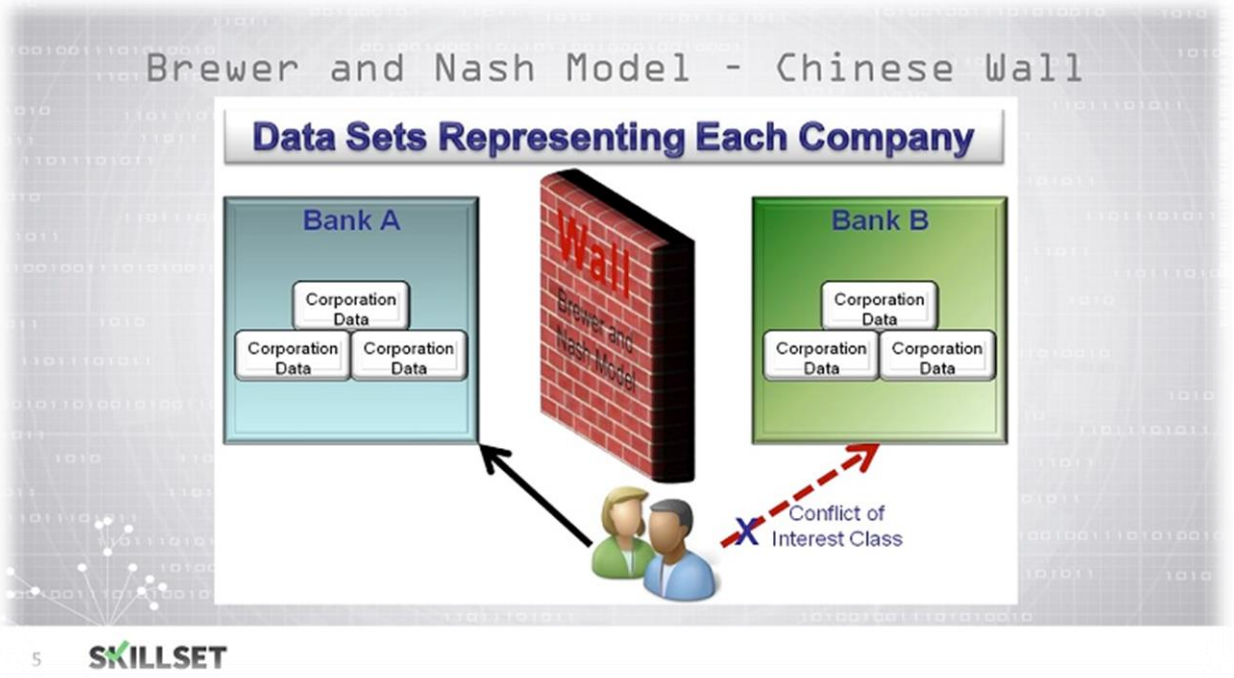
The Chinese wall policy builds on three level of abstraction :

- **Objects:** contain information about only one company (for example : files)
- **Company groups:** collect all objects concerning a particular company
- **Conflict classes:** cluster the groups of objects for competing

The essentials elements are

- **Subjects:** Active entities accessing protected objects
- **Objects:** Data organized according to 3 levels (Information; DataSet; Conflict-of Interest (Col) classes)
- Access Rules: Read rule, Write rule
- **Read Rule:** Subject S can read object O only if O is from the same company information as some object read by S, or O belongs to a COl class within which S has not read any object.

- **Write Rule**: Subject S can write object O only if S can read O by the Brewer-Nash Read rule, and no object can be read which is in different company dataset to the one for which write access is requested company.



3 –Program Security

Program security is a critical component of information security that involves protecting computer programs from unauthorized access, modification, or exploitation. It encompasses various practices and techniques aimed at ensuring that software applications are free from vulnerabilities and defects that can be exploited by attackers to compromise the confidentiality, integrity, or availability of data**.**

Fault tolerance terminology:

- **Fault :** It is a condition that causes the software to fail to perform its required function. • Faults - seen by „insiders" (e.g., programmers
- **Error** : Refers to difference between Actual Output and Expected output. Error/fault/failure example:
- **Failure** : It is the inability of a system or component to perform required function according to its specification. Failures - seen by „outsiders" (e.g., independent testers, users)

Judging S/w Security by Fixing Faults

- An approach to judge s/w security: penetrate and patch
- Red Team / Tiger Team tries to crack s/w
- If you withstand the attack => security is good Is this true? Rarely.
- Too often developers try to quick-fix problems discovered by Tiger Team

  Quick patches often introduce new faults due to:

- Pressure – causing narrow focus on fault, not context
- Non-obvious side effects
- System performance requirements not allowing for security overhead

Types of Program Flaws

- **Malicious**

- **Non Malicious**

Non Malicious program flaws, also known as vulnerabilities, are weaknesses or mistakes in software code that can be exploited by attackers to gain unauthorized access, manipulate data, or disrupt the normal functioning of a program or system. These flaws can exist in any type of software, including operating systems, applications, and web services. Here are some common types of program flaws:

- **Buffer Overflow**: This occurs when a program writes more data into a buffer than it can handle, causing the excess data to overflow into adjacent memory locations. Attackers can exploit this flaw by injecting malicious code into the overflowed buffer, potentially gaining control of the program or system.

- **Incomplete Mediation**

Incomplete mediation flaw — often inadvertent (=> nonmalicious) but with serious security consequences

Sensitive data are in exposed, uncontrolled condition □□Example

- URL to be generated by client's browser to access server, e.g.:

http://www.things.com/order/final&custID=101&part=555A&qy=20 &price=10&ship=boat&shipcost=5&total=205

- Instead, user edits URL directly, changing price and total cost as follows: http://www.things.com/order/final&custID=101&part=555A&qy=20 &price=1&ship=boat&shipcost=5&total=25

- User uses forged URL to access server □□The server takes 25 as the total cost
- Unchecked data are a serious vulnerability!
- Possible solution: anticipate problems
- Don't let client return a sensitive result (like total) that can be easily recomputed by server
- Use drop-down boxes / choice lists for data input □□Prevent user from editing input directly
- Check validity

Time-of-check to Time-of-use Errors

Time-of-check to time-of-use flaw — often inadvertent (=>

nonmalicious) but with serious security consequences

A.k.a. synchronization flaw / serialization flaw

TOCTTOU — mediation with "bait and switch" in the middle Non-computing example:

Swindler shows buyer real Rolex watch (bait)

After buyer pays, switches real Rolex to a forged one In computing:

Change of a resource (e.g., data) between time access checked and time access used

: Any examples of TOCTTOU problems from computing?

TOCTTOU — mediation with “bait and switch” in the middle □□...

- Q: Any examples of TOCTTOU problems from computing?

- A: E.g., DBMS/OS: serialization problem: pgm1 reads value of X = 10

pgm1 adds X = X+ 5

pgm2 reads X = 10, adds 3 to X, writes X = 13 pgm1 writes X = 15

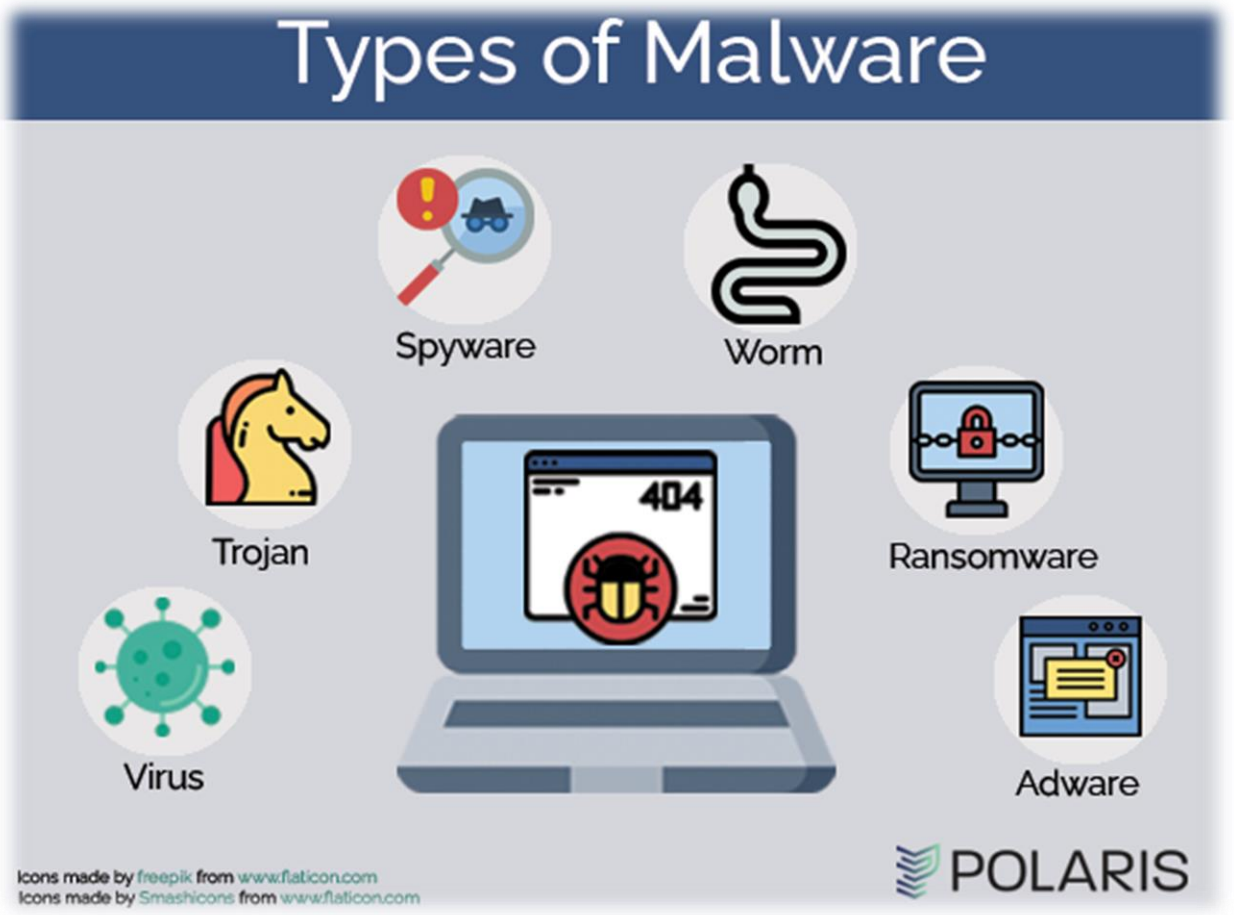
X ends up with value 15 – should be X = 18

Prevention of TOCTTOU errors

- Be aware of time lags
- Use digital signatures and certificates to „lock” data values after checking them
- So nobody can modify them after check & before use

4 –Malicious Software

Malicious software, commonly known as malware, refers to any software or code designed to infiltrate, damage, or gain unauthorized access to computer systems or networks. Malware is created with malicious intent and can cause various types of harm, including stealing sensitive information, disrupting system operations, or compromising security.



Here are some common types of malware and their characteristics:

1. Viruses: Viruses are self-replicating programs that infect other files or programs by inserting their own code. They can spread rapidly and cause damage to files, software, or the entire system. For example, the infamous "ILOVEYOU" virus in 2000 spread through email and overwrote or destroyed files on infected machines.

2. Worms: Worms are similar to viruses but don't require a host program to replicate. They exploit network vulnerabilities and spread

autonomously, infecting multiple systems. The "Conficker" worm, discovered in 2008, targeted Windows operating systems and created a botnet for various malicious activities.

3. Trojans: Trojans, or Trojan horses, masquerade as legitimate software or files to deceive users into executing them. Once activated, they can perform various malicious actions, such as stealing sensitive information, providing unauthorized access to the attacker, or creating backdoors for future attacks. The "Zeus" Trojan, first identified in 2007, aimed at banking credentials and financial fraud.

4. Ransomware: Ransomware encrypts a victim's files, making them inaccessible until a ransom is paid to the attacker. It often spreads through malicious email attachments, compromised websites, or exploit kits. Notable ransomware attacks include "WannaCry" in 2017 and "Ryuk" in 2019, which affected organizations worldwide.

5. Spyware: Spyware collects user information without their knowledge or consent. It can monitor online activities, capture keystrokes, record passwords, or track browsing habits. Spyware is often bundled with seemingly legitimate software or embedded in malicious websites. One example is the "FinFisher" spyware, which has been used for surveillance purposes.

6. Adware: Adware displays unwanted advertisements or redirects users to advertising websites. It can be annoying and disrupt user experience,

but it's generally considered less harmful than other malware types. "Superfish" adware, pre-installed on certain Lenovo laptops in 2014, caused privacy concerns by intercepting and modifying users' web traffic.

7. Botnets: Botnets are networks of infected computers or "bots" that can be remotely controlled by an attacker, typically through a command-and-control (C&C) server. They are commonly used for launching distributed denial-of-service (DDoS) attacks, sending spam emails, or conducting large-scale cybercrime operations. The "Mirai" botnet, discovered in 2016, targeted Internet of Things (IoT) devices.

How Viruses Work

- Pgm containing virus must be executed to spread virus or infect other pgms
- Even one pgm execution suffices to spread virus widely

Virus actions: spread / infect

Spreading – Example 1: Virus in a pgm on installation CD □ □User activates pgm contaning virus when she runs

INSTALL or SETUP

- Virus installs itself in any/all executing pgms present in memory
- Virus installs itself in pgms on hard disk
- From now on virus spreads whenever any of the infected pgms (from memory or hard disk) executes

How to protect against malware

- In no particular order, here's our tips on protecting against malware.
- 1. Pay attention to the domain and be wary if the site isn't a top-level domain, i.e., com, mil, net, org, edu, or biz, to name a few.
- 2. Use strong passwords with multi-factor authentication. A password manager can be a big help here.
- 3. Avoid clicking on pop-up ads while browsing the Internet.
- 4. Avoid opening email attachments from unknown senders.
- 5. Do not click on strange, unverified links in emails, texts, and social media messages.
- 6. Don't download software from untrustworthy websites or peer-to-peer file transfer networks.
- 7. Stick to official apps from Google Play and Apple's App Store on Android, OSX, and iOS (and don't jailbreak your phone). PC users should check the ratings and reviews before installing any software.
- 8. Make sure your operating system, browsers, and plugins are patched and up to date.
- 9. Delete any programs you don't use anymore.
- 10. Back up your data regularly. If your files become damaged, encrypted, or otherwise inaccessible, you'll be covered.
- 11. Download and install a cybersecurity program that actively scans and blocks threats from getting on your device. Malwarebytes, for example, offers proactive cybersecurity programs for Windows, Mac, Android, and Chromebook. Plus, our latest offering, Malwarebytes Browser Guard. It's free and it's the only browser extension that can stop tech support scams along with any other unsafe and unwanted content that comes at you through your browser.

How to remove malware

Follow these three easy steps to remove malware from your device.

1. Download and install a good cybersecurity program. As it happens, Malwarebytes has programs for every platform we've discussed in this article: Windows, Mac, Android, and Chromebook.

2. Run a scan using your new program. Even if you don't opt for Malwarebytes Premium, the free version of Malwarebytes is still great at removing malware. The free version, however, does not proactively stop threats from getting on your system in the first place.

3. Change all your passwords. Now that you know you're not being snooped on by some form of malware, you need to reset your passwords—not only for your PC or mobile device, but also your email, your social media accounts, your favorite shopping sites, and your online banking and billing centers. This may sound paranoid, but with spyware, banking Trojans and the like, you just don't know for sure what data was captured before you stopped the infection. As always, use some form of multi-factor authentication (at least two-factor) and don't think you need to memorize all your passwords. Use a password manager instead.

If your iPhone or iPad is infected with malware (as improbable as that may be). Things are a little trickier. Apple does not permit scans of either the device's system or other files, though Malwarebytes for iOS, for example, will screen and block scam calls and texts. Your only option is to wipe your phone with a factory reset, then restore it from your backup in iCloud or iTunes. If you didn't backup your phone, then you're starting over from scratch.

5 -Operating System Security

Every computer system and software design must handle all security risks and implement the necessary measures to enforce security policies. At the same time, it's critical to strike a balance because strong security measures might increase costs while also limiting the system's usability, utility, and smooth operation. As a result, system designers must assure efficient performance without compromising security.

What is Operating System Security?

The process of ensuring OS availability, confidentiality, integrity is known as operating system security. OS security refers to the processes or measures taken to protect the operating system from dangers, including viruses, worms, malware, and remote hacker intrusions. Operating system security comprises all preventive-control procedures that protect any system assets that could be stolen, modified, or deleted if OS security is breached.

Security refers to providing safety for computer system resources like software, CPU, memory, disks, etc. It can protect against all threats, including viruses and unauthorized access. It can be enforced by assuring the operating system's integrity, confidentiality, and availability. If an illegal user runs a computer application, the computer or data stored may be seriously damaged.

System security may be threatened through two violations, and these are as follows:

1. Threat

A program that has the potential to harm the system seriously.

2. Attack

A breach of security that allows unauthorized access to a resource.

There are two types of security breaches that can harm the system: malicious and accidental. Malicious threats are a type of destructive computer code or web script that is designed to cause system vulnerabilities that lead to back doors and security breaches. On the other hand, Accidental Threats are comparatively easier to protect against.

Security may be compromised through the breaches. Some of the breaches are as follows:

1.Breach of integrity

This violation has unauthorized data modification.

2.Theft of service

It involves the unauthorized use of resources.

3.Breach of confidentiality

It involves the unauthorized reading of data.

4.Breach of availability

It involves the unauthorized destruction of data.

5.Denial of service

It includes preventing legitimate use of the system. Some attacks may be accidental.

The goal of Security System

1. Integrity

Unauthorized users must not be allowed to access the system's objects, and users with insufficient rights should not modify the system's critical files and resources.

2. Secrecy

The system's objects must only be available to a small number of authorized users. The system files should not be accessible to everyone.

3. Availability

All system resources must be accessible to all authorized users, i.e., no single user/process should be able to consume all system resources. If such a situation arises, service denial may occur. In this case, malware may restrict system resources and preventing legitimate processes from accessing them.

How to ensure Operating System Security?

There are various ways to ensure operating system security. These are as follows:

Authentication

The process of identifying every system user and associating the programs executing with those users is known as authentication. The operating system is responsible for implementing a security system that ensures the authenticity of a user who is executing a specific program. In general, operating systems identify and authenticate users in three ways.

1. Username/Password

Every user contains a unique username and password that should be input correctly before accessing a system.

2. User Attribution

These techniques usually include biometric verification, such as fingerprints, retina scans, etc. This authentication is based on user uniqueness and is compared to database samples already in the system. Users can only allow access if there is a match.

3. User card and Key

To login into the system, the user must punch a card into a card slot or enter a key produced by a key generator into an option provided by the operating system.

One Time passwords

Along with standard authentication, one-time passwords give an extra layer of security. Every time a user attempts to log into the One-Time Password system, a unique password is needed. Once a one-time password has been used, it cannot be reused. One-time passwords may be implemented in several ways.

1. Secret Key

The user is given a hardware device that can generate a secret id that is linked to the user's id. The system prompts for such a secret id, which must be generated each time you log in.

2. Random numbers

Users are given cards that have alphabets and numbers printed on them. The system requests numbers that correspond to a few alphabets chosen at random.

3. Network password

Some commercial applications issue one-time passwords to registered mobile/email addresses, which must be input before logging in.

Firewalls

Firewalls are essential for monitoring all incoming and outgoing traffic. It imposes local security, defining the traffic that may travel through it. Firewalls are an efficient way of protecting network systems or local systems from any network-based security threat.

Physical Security

The most important method of maintaining operating system security is physical security. An attacker with physical access to a system may edit, remove, or steal important files since operating system code and configuration files are stored on the hard drive.

Components of Operating System

An operating system is a large and complex system that can only be created by partitioning into small parts. These pieces should be a well

defined part of the system, carefully defining inputs, outputs, and functions.

Although Windows, Mac, UNIX, Linux, and other OS do not have the same structure, most operating systems share similar OS system components, such as file, memory, process, I/O device management.

The components of an operating system play a key role to make a variety of computer system parts work together. There are the following components of an operating system, such as:

- Process Management
- File Management
- Network Management
- Main Memory Management
- Secondary Storage Management
- I/O Device Management
- Security Management
- Command Interpreter System

Operating system components help you get the correct computing by detecting CPU and memory hardware errors.

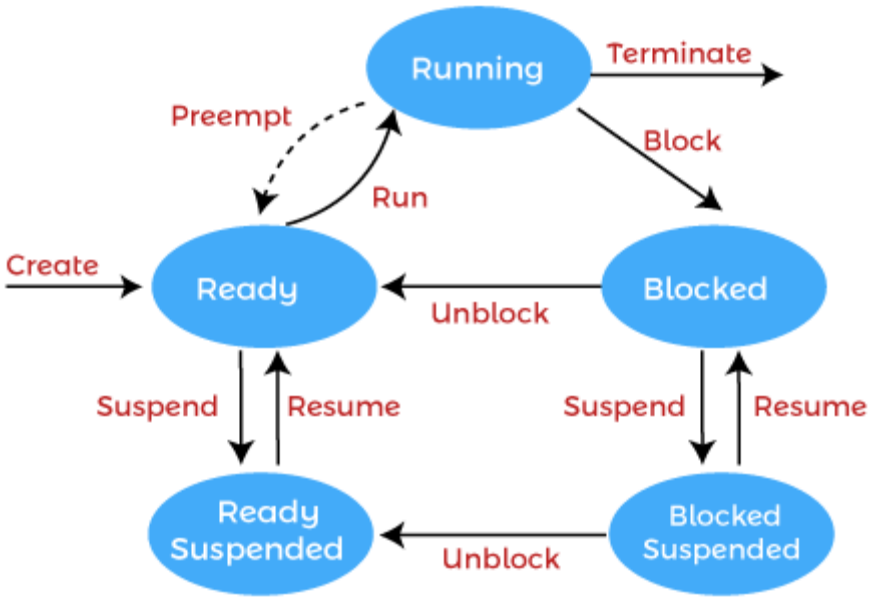
Process Management

The process management component is a procedure for managing many processes running simultaneously on the operating system. Every running software application program has one or more processes associated with them.

For example, when you use a search engine like Chrome, there is a process running for that browser program.

Process management keeps processes running efficiently. It also uses memory allocated to them and shutting them down when needed.

The execution of a process must be sequential so, at least one instruction should be executed on behalf of the process.



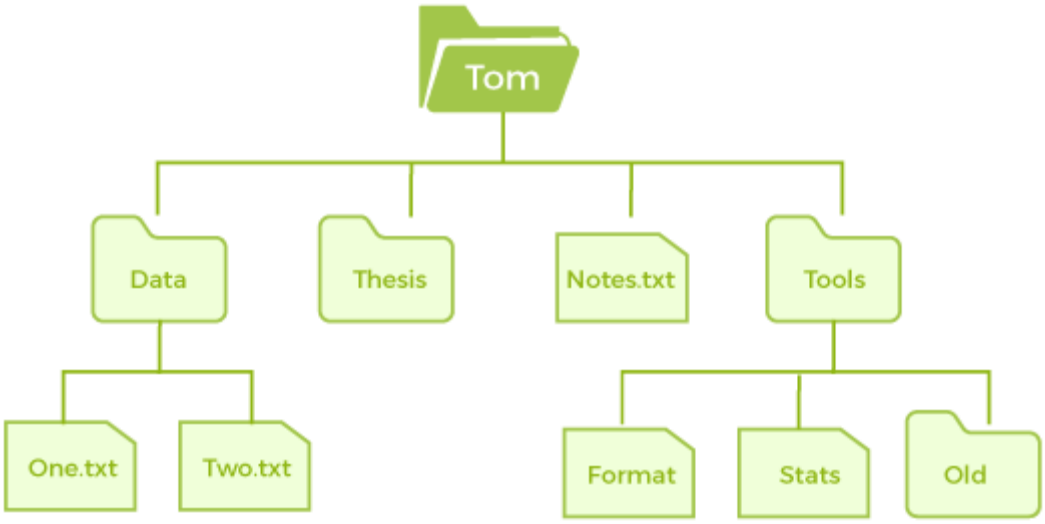
Functions of process management

Here are the following functions of process management in the operating system, such as:

- Process creation and deletion.
- Suspension and resumption.
- Synchronization process
- Communication process

File Management

A file is a set of related information defined by its creator. It commonly represents programs (both source and object forms) and data. Data files can be alphabetic, numeric, or alphanumeric.



Function of file management

The operating system has the following important activities in connection with file management:

- File and directory creation and deletion.
- For manipulating files and directories.
- Mapping files onto secondary storage.
- Backup files on stable storage media.

Network Management

Network management is the process of administering and managing computer networks. It includes performance management, provisioning of networks, fault analysis, and maintaining the quality of service.

A distributed system is a collection of computers or processors that never share their memory and clock. In this type of system, all the processors have their local memory, and the processors communicate with each other using different communication cables, such as fibre optics or telephone lines.

The computers in the network are connected through a communication network, which can configure in many different ways. The network can fully or partially connect in network management, which helps users design routing and connection strategies that overcome connection and security issues.

Functions of Network management

Network management provides the following functions, such as:

- Distributed systems help you to various computing resources in size and function. They may involve minicomputers, microprocessors, and many general-purpose computer systems.
- A distributed system also offers the user access to the various resources the network shares.

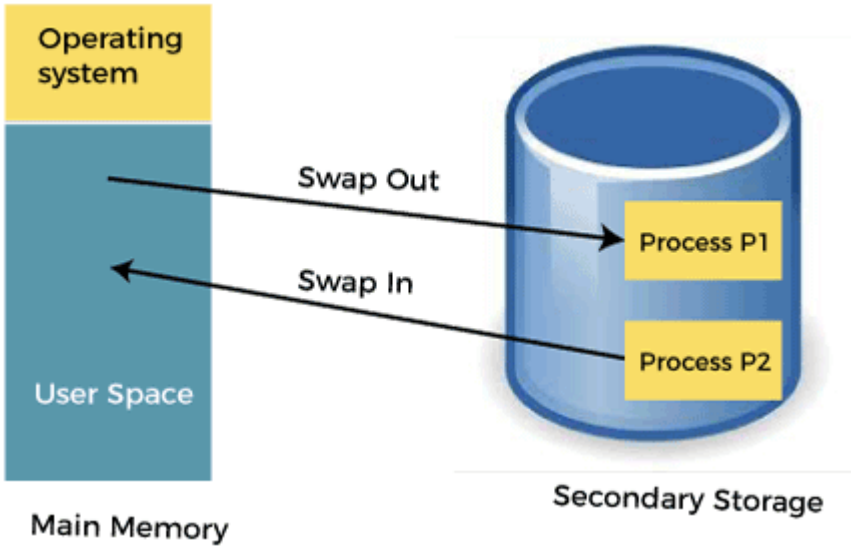
- It helps to access shared resources that help computation to speed up or offers data availability and reliability.

Main Memory management

Main memory is a large array of storage or bytes, which has an address. The memory management process is conducted by using a sequence of reads or writes of specific memory addresses.

It should be mapped to absolute addresses and loaded inside the memory to execute a program. The selection of a memory management method depends on several factors.

However, it is mainly based on the hardware design of the system. Each algorithm requires corresponding hardware support. Main memory offers fast storage that can be accessed directly by the CPU. It is costly and hence has a lower storage capacity. However, for a program to be executed, it must be in the main memory.



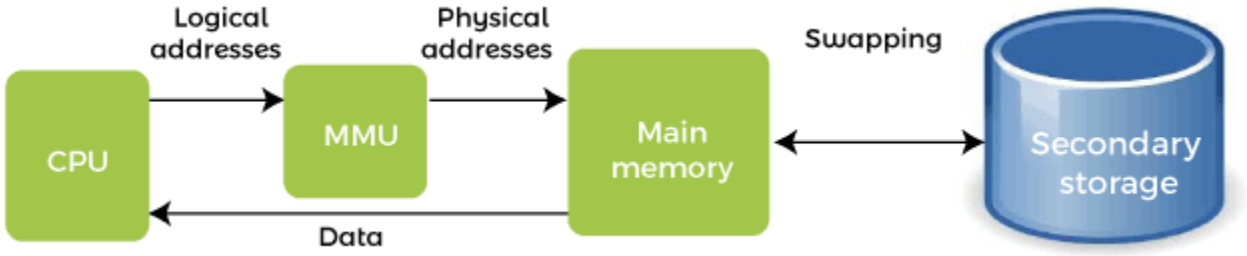
Functions of Memory management

An Operating System performs the following functions for Memory Management in the operating system:

- It helps you to keep track of primary memory.
- Determine what part of it are in use by whom, what part is not in use.
- In a multiprogramming system, the OS decides which process will get memory and how much.
- Allocates the memory when a process requests.
- It also de-allocates the memory when a process no longer requires or has been terminated.

Secondary-Storage Management

The most important task of a computer system is to execute programs. These programs help you to access the data from the main memory during execution. This memory of the computer is very small to store all data and programs permanently. The computer system offers secondary storage to back up the main memory.



Today modern computers use hard drives/SSD as the primary storage of both programs and data. However, the secondary storage management also works with storage devices, such as USB flash drives and CD/DVD drives. Programs like assemblers and compilers are stored on the disk

until it is loaded into memory, and then use the disk is used as a source and destination for processing.

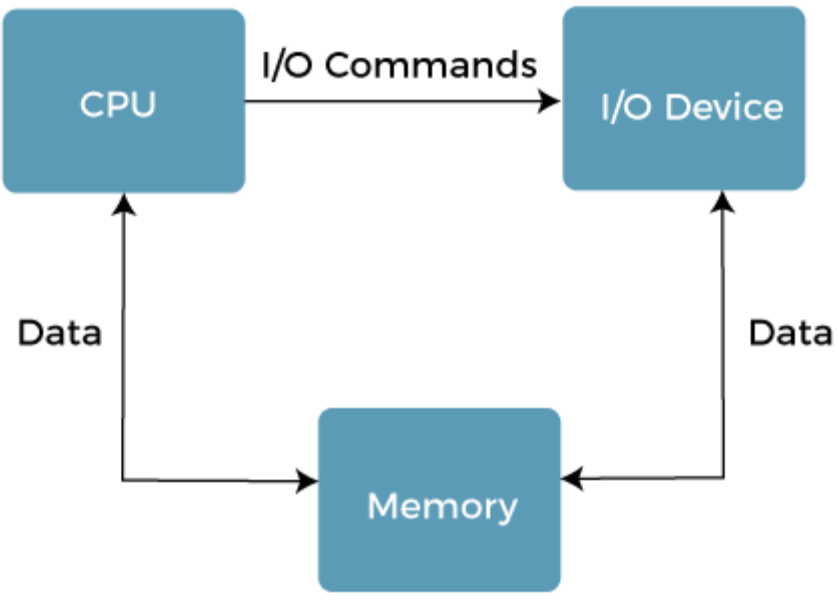
Functions of Secondary storage management

Here are some major functions of secondary storage management in the operating system:

- Storage allocation
- Free space management
- Disk scheduling

I/O Device Management

One of the important use of an operating system that helps to hide the variations of specific hardware devices from the user.



Functions of I/O management

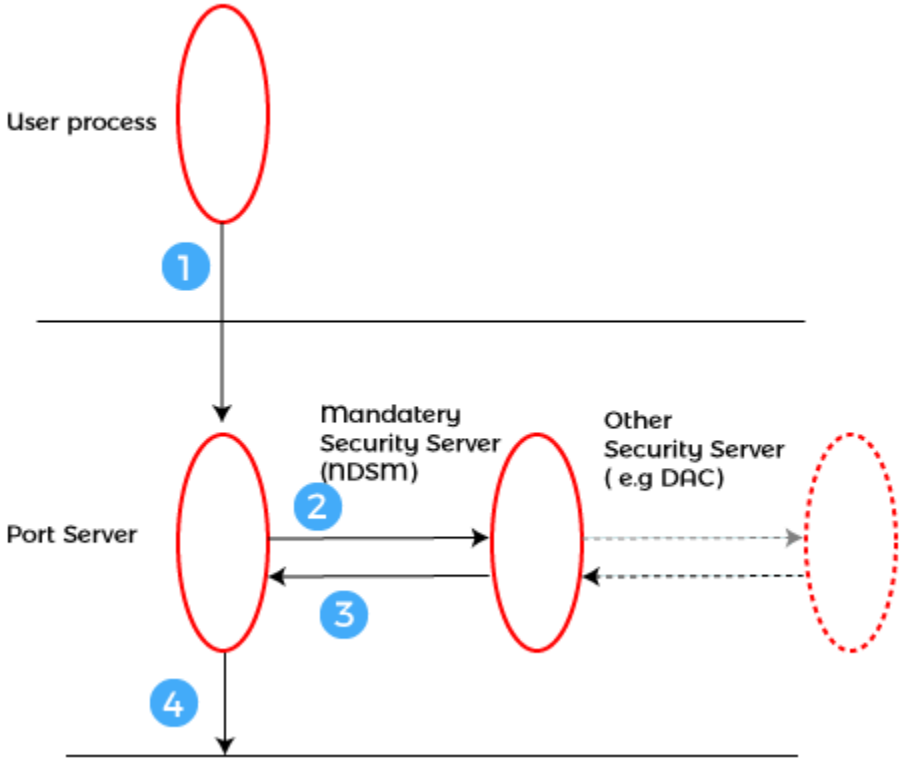
The I/O management system offers the following functions, such as:

- It offers a buffer caching system
- It provides general device driver code
- It provides drivers for particular hardware devices.
- I/O helps you to know the individualities of a specific device.

Security Management

The various processes in an operating system need to be secured from other activities. Therefore, various mechanisms can ensure those processes that want to operate files, memory CPU, and other hardware resources should have proper authorization from the operating system.

Security refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by computer controls to be imposed, together with some means of enforcement.

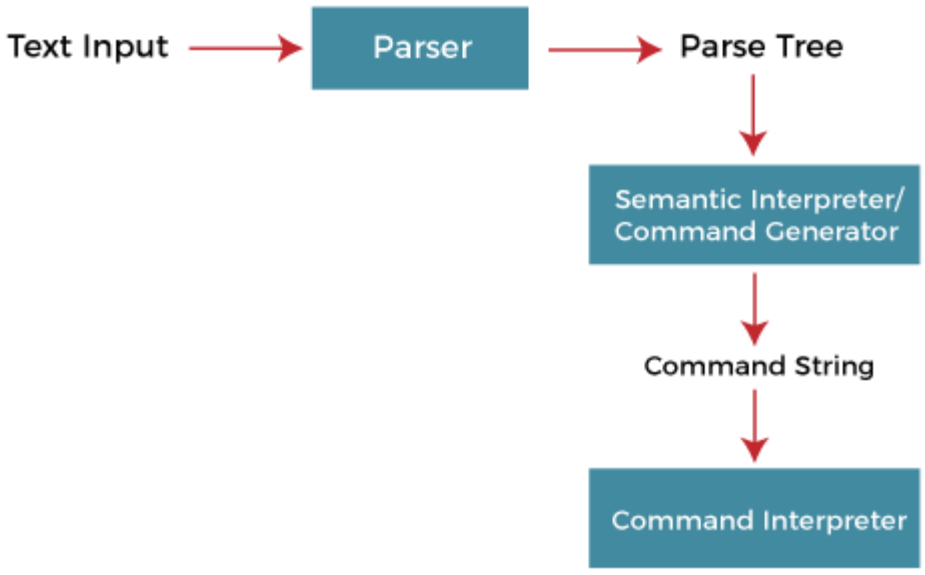


For example, memory addressing hardware helps to confirm that a process can be executed within its own address space. The time ensures that no process has control of the CPU without renouncing it. Lastly, no process is allowed to do its own I/O to protect, which helps you to keep the integrity of the various peripheral devices.

Security can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can prevent the foulness of a healthy subsystem by a malfunctioning subsystem. An unprotected resource cannot misuse by an unauthorized or incompetent user.

Command Interpreter System

One of the most important components of an operating system is its command interpreter. The command interpreter is the primary interface between the user and the rest of the system.



Many commands are given to the operating system by control statements. A program that reads and interprets control statements is

automatically executed when a new job is started in a batch system or a user logs in to a time-shared system. This program is variously called.

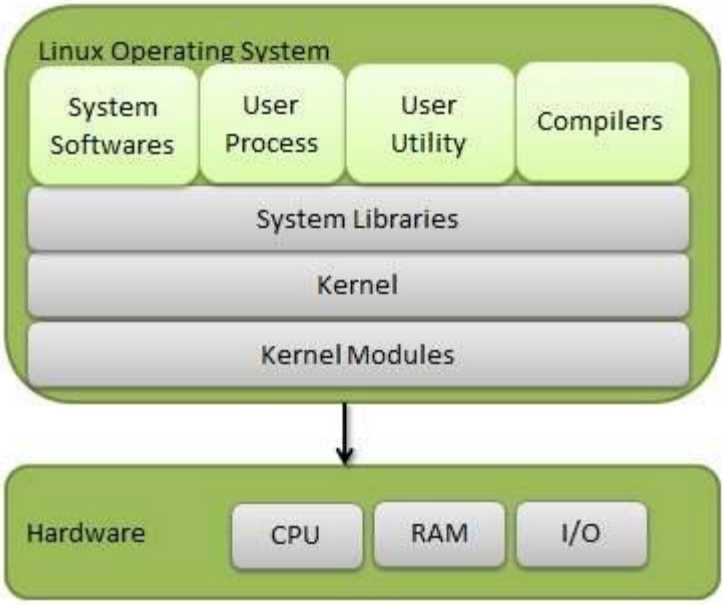
- The control card interpreter,
- The command-line interpreter,
- The shell (in UNIX), and so on.

Its function is quite simple, get the next command statement, and execute it. The command statements deal with process management, I/O handling, secondary storage management, main memory management, file system access, protection, and networking.

Components of Linux System

Linux Operating System has primarily three components

- **Kernel** − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.
- **System Library** − System libraries are special functions or programs using which application programs or system utilities accesses Kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module's code access rights.
- **System Utility** − System Utility programs are responsible to do specialized, individual level tasks.



Kernel Mode vs User Mode

Kernel component code executes in a special privileged mode called **kernel mode** with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

Support code which is not required to run in kernel mode is in System Library. User programs and other system programs works in **User Mode** which has no access to system hardware and kernel code. User programs/ utilities use System libraries to access Kernel functions to get system's low level tasks.

Basic Features

Following are some of the important features of Linux Operating System.

- **Portable** − Portability means software can works on different types of hardware in same way. Linux kernel and application programs supports their installation on any kind of hardware platform.
- **Open Source** − Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of Linux operating system and it is continuously evolving.
- **Multi-User** − Linux is a multiuser system means multiple users can access system resources like memory/ ram/ application programs at same time.
- **Multiprogramming** − Linux is a multiprogramming system means multiple applications can run at same time.
- **Hierarchical File System** − Linux provides a standard file structure in which system files/ user files are arranged.
- **Shell** − Linux provides a special interpreter program which can be used to execute commands of the operating system. It can be used to do various types of operations, call application programs. etc.
- **Security** − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

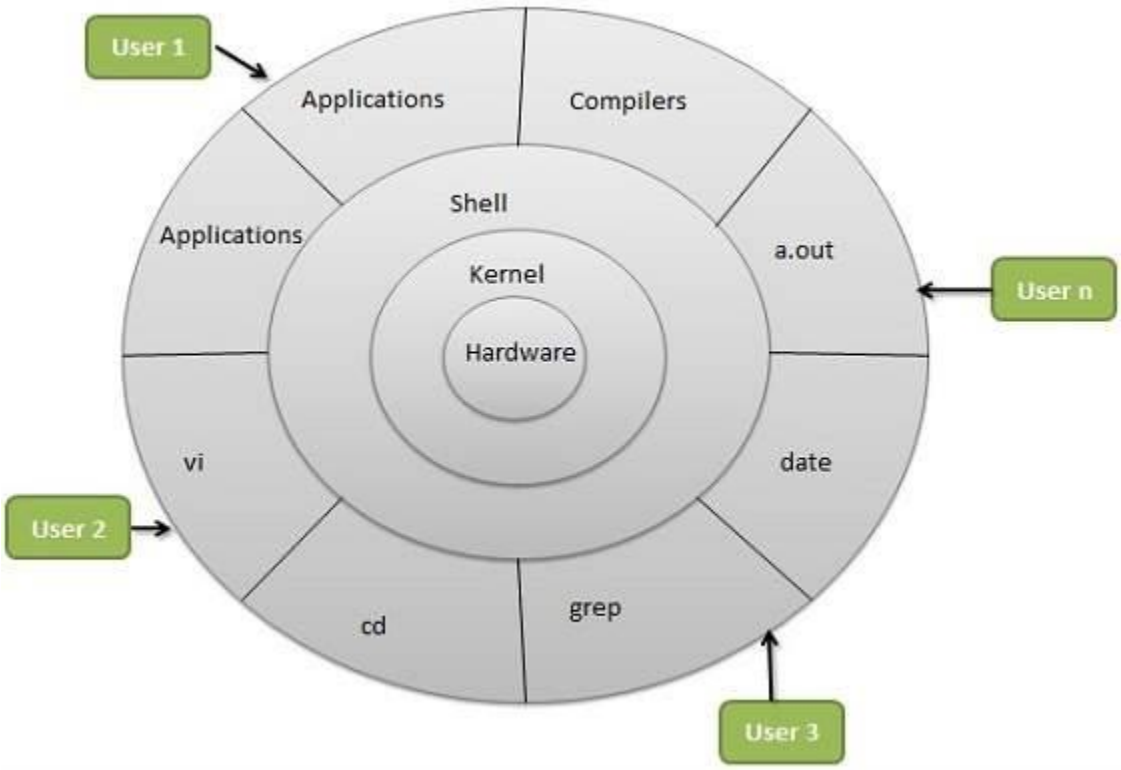
Architecture

The following illustration shows the architecture of a Linux system −

The architecture of a Linux System consists of the following layers −

- **Hardware layer** − Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc).
- **Kernel** − It is the core component of Operating System, interacts directly with hardware, provides low level services to upper layer components.
- **Shell** − An interface to kernel, hiding complexity of kernel's functions from users. The shell takes commands from the user and executes kernel's functions.

- **Utilities** − Utility programs that provide the user most of the functionalities of an operating systems.



6 –Privacy and Privacy Enhancement Tools

Privacy is a fundamental right that individuals have to control their personal information and protect it from unauthorized access or misuse. Privacy-enhancing tools are technologies or strategies that aim to enhance privacy and give users more control over their data.

4 categories of PETs:

- Encryption Tools (e.g., SSL)
- Policy Tools (e.g., P3P, TRUSTe)
- Filtering Tools (e.g., Cookie Cutters, Spyware) Anonymity Tools (e.g., Anonymizer, iPrivacy)

Encryption tools

- Examples: SSL, PGP, Encryptionizer
- Thought of as a security tool to prevent unauthorized access
- Users don't see the need
- Necessary for privacy protection but not sufficient by themselves

Pros:

- Inexpensive (free) / Easily accessible

Cons:

- Encryption Software isn't used unless it is built-in to the software
- Both parties need to use the same software

Conclusions:

- Easy access
- All parties need to use the same tool
- Good start but not sufficient enough

P3P (Platform for Privacy Preferences)

- Developed by World Wide Web Consortium
- Users declare their privacy policy on their browsers

- Websites register their policy with Security agencies
- The website policy is compared with user policy and the browser makes automated decisions

Benefits:

- Might help uncover privacy gaps for websites
- Can block cookies or prevent access to some sites
- Built into IE 6.0 and Netscape 7 as of July

TRUSTe

- Non-profit organization which ensures that websites are following their privacy policy
- Promotes fair information practices
- BBBonline (Better Business Bureau)

Conclusions

- Users are unaware of Privacy Policies
- Not all websites have Policy tools
- Need automated checks to see if websites their privacy policy

Filtering Tools

Some Types

1) SPAM filtering

2) Cookie Cutters

3) Spyware killers

1) SPAM Filters

Problems:

- Spammers use new technologies to defeat filters
- Legitimate E-mailers send SPAM resembling E-mail

Possible Solution:

- E-Mail postage scheme
- Have to pay a bit for each e-mail => too costly to spam

Infeasible solution

- Tough to impose worldwide
- Need homogenous technology for all parties
- Policy responsibility is unclear (Who will police it?)

2) Cookie Cutters

- Programs that prevent browsers from exchanging cookies
- Cookie /Pop-ups
- http headers that reveal sensitive info
- Banner ads / Animated graphics

3) Spyware Killers:

- To deal with spyware
- Spyware programs gather info and send it to websites

- Downloaded without user knowledge

Anonymity Tools

- Enable users to communicate anonymously Mask the IP address and personal info
- Some use 3rd party proxy servers
- Strip off user info and forward the rest to websites
- Not helpful for online transactions
- Expensive

Types of anonymity tools

1) Autonomy Enhancing (Anonymizer)

2) Seclusion Enhancing (iPrivacy)

3) Property Managing (.NET Passport)

1) Autonomy Enhancing Technology (1)

Examples:

- Anonymizer, Freedom by Zero Knowledge
- No user information is stored by anybody but its “owner”
- User has complete control

Anonymizer:

- One of the first PETs

- Not concerned with transaction security

Provides anonymity by:

- Routing through a proxy server
- Software to manage security at the “owner’s” PC
- Erases cookies and log files, pop-up blocker, kills Spyware, unlisted IP

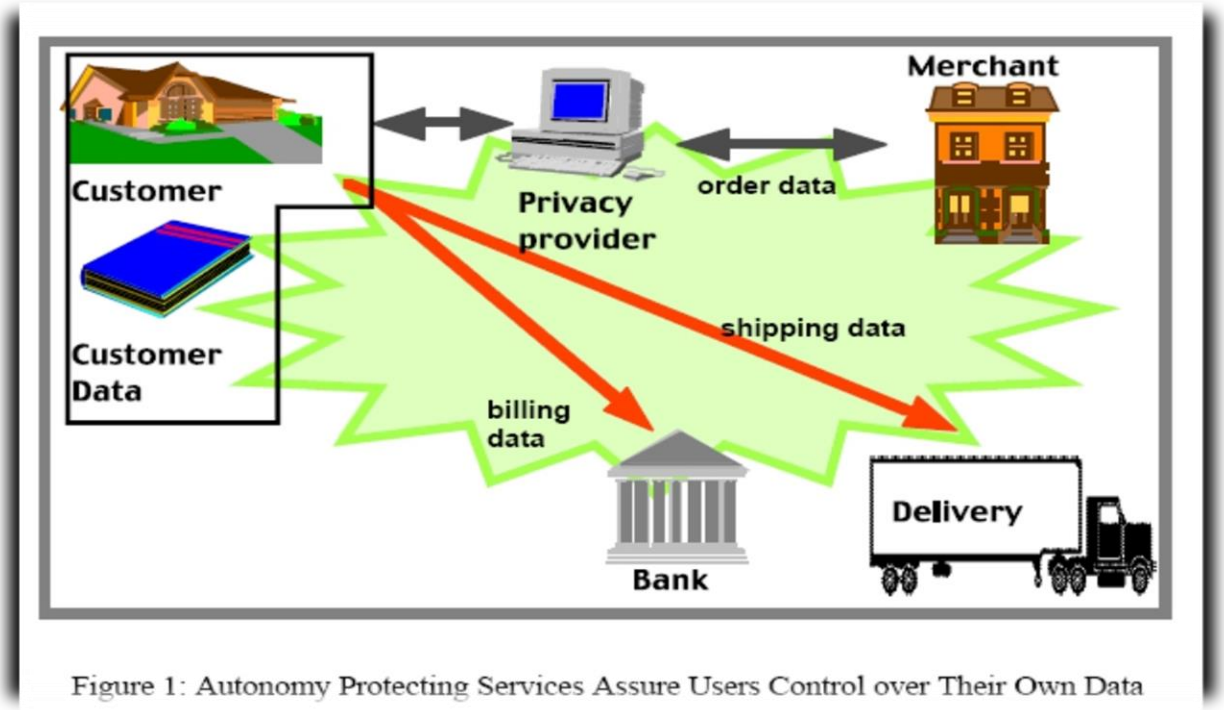


Figure 1: Autonomy Protecting Services Assure Users Control over Their Own Data

2) Seclusion Enhancing Technologies

Examples:

- iPrivacy, Incogno SafeZone
- Target Transaction processing companies
- Trusted third party (TTP) who promises not to contact the customer
- Consumer remains the decision maker

- TTP keeps limited data (dispute resolution)
- Transaction by transaction basis
- Customers can choose to not give any data to merchants

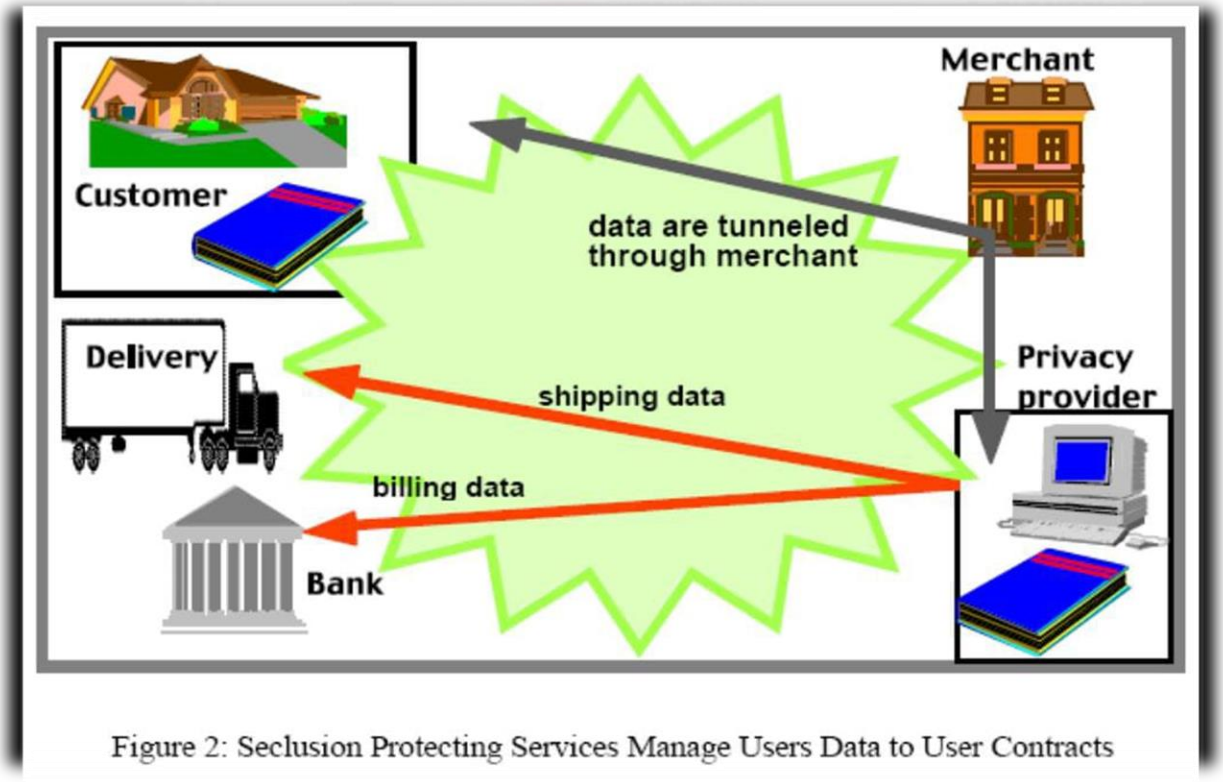


Figure 2: Seclusion Protecting Services Manage Users Data to User Contracts

iPrivacy

- Intermediary for users and companies
- Doesn’t have the ability to look at all user data
- Cannot map transactions to user info
- Each transaction needs to have personal info filled out
- Customer downloads software
- Client-side software for shipping and Credit Card companies
- Licensed to Credit Card and Shipping Companies
- Avoids replay attacks for CC companies
- Allows users to end associations with merchants

(3) Property Managing Technology

Example:

- .NET Passpor
- All user data is kept by the “privacy provider”
- Like a lawyer protecting privacy of a client
- Consumer doesn’t directly communicate with the merchant
- Consumer’s control rights are surrendered for service
- Potential for misuse of data
- User gives agency rights to the provider
- No direct contact with merchant

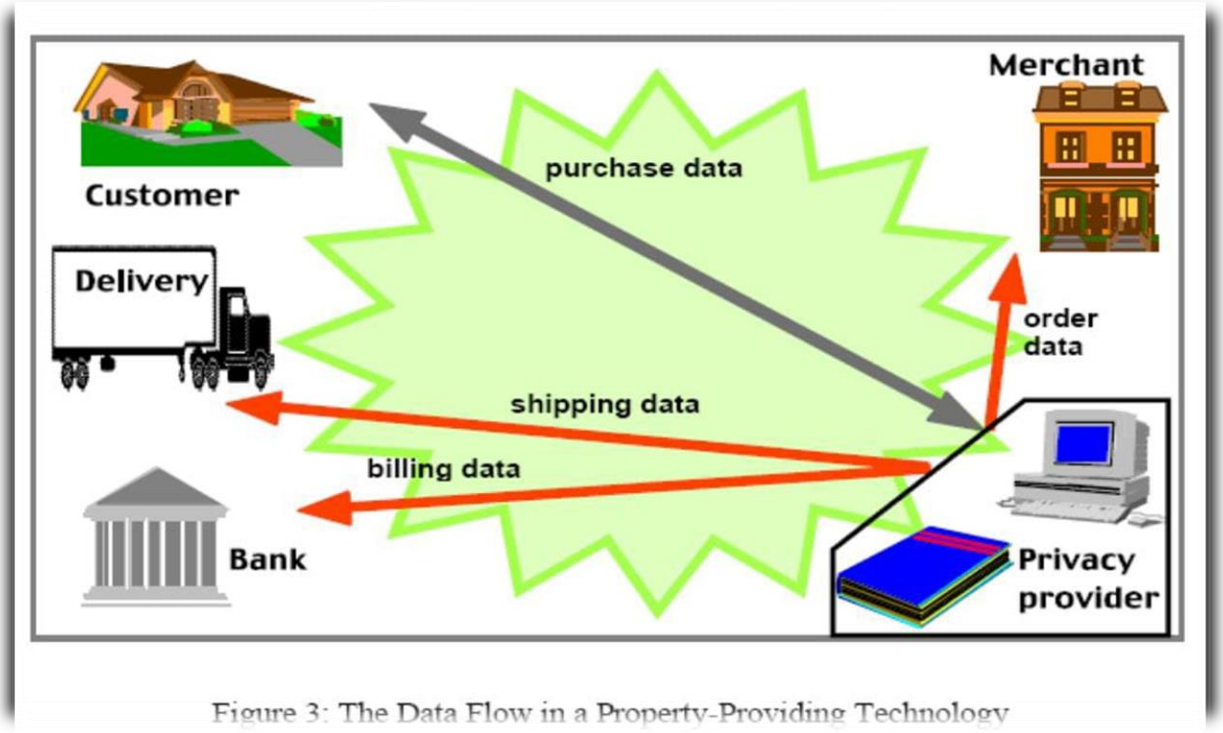


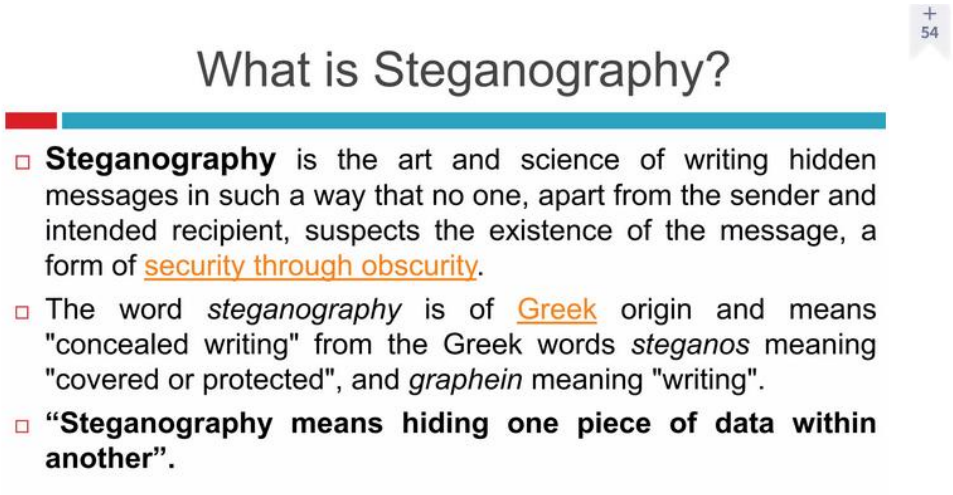
Figure 3: The Data Flow in a Property-Providing Technology

.NET Passport

- Single login service
- Customer’s personal info is contained in the Passport profile.

- Name, E-mail, state, country, zip, gender, b-day, occupation, telephone #
- Controls and logs all transactions
- Participating sites can provide personalized services □ □Merchants only get a Unique ID
- Participants:
- Ebay, MSN, Expedia, NASDAQ, Ubid.com

7 –Steganography



Example

Since everyone can read, encoding text in neutral sentences is doubtfully effective

***S**ince **E**veryone **C**an **R**ead, **E**ncoding **T**ext*
***I**n **N**eutral **S**entences **I**s **D**oubtfully **E**ffective*

'Secret inside'

History Of Steganography

- The first recorded uses of steganography can be traced back to 440 BC when Herodotus mentions two examples of steganography in his *Histories*.
- Demaratus sent a warning about a forthcoming attack to Greece by writing it directly on the wooden backing of a wax tablet before applying its beeswax surface.
- Wax tablets were in common use then as reusable writing surfaces, sometimes used for shorthand.
- Ancient Chinese wrote messages on fine silk, which was then crunched into a tiny ball and covered in wax. The messenger then swallowed the ball of wax.
- Special "inks" were important steganographic tools even during Second World War.
- During Second World War a technique was developed to shrink photographically a page of text into a dot less than one millimeter in diameter, and then hide this microdot in an apparently innocuous letter. (The first microdot has been spotted

Physical Techniques

- Physical Techniques
  - Hidden messages within wax tablets
  - Hidden messages on messenger's body
  - Hidden messages on paper written in secret inks
  - Messages written in Morse code on knitting yarn and then knitted into a piece of clothing worn by a courier
  - Messages written on envelopes in the area covered by postage stamps.

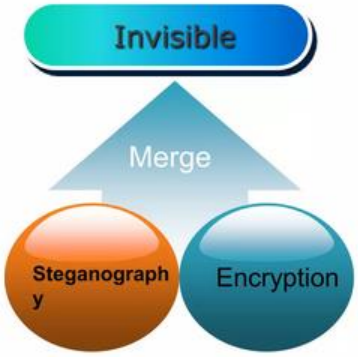
Digital Techniques

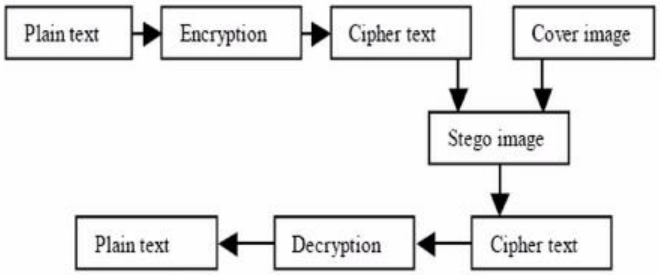
- Digital Techniques
  - Concealing messages within the lowest bits of noisy images or sound files.
  - Chaffing and winnowing.
  - Modifying the echo of a sound file (Echo Steganography)
  - Including data in ignored sections of a file, such as after the logical end of the carrier file.

Comparison Of Secret Communication Techniques

Secret Communication Techniques	Confidentiality	Integrity	Availability
Encryption	Yes	No	Yes
Digital Signatures	No	Yes	No
Steganography	Yes/No	Yes/No	Yes

Combined Crypto-Steganography

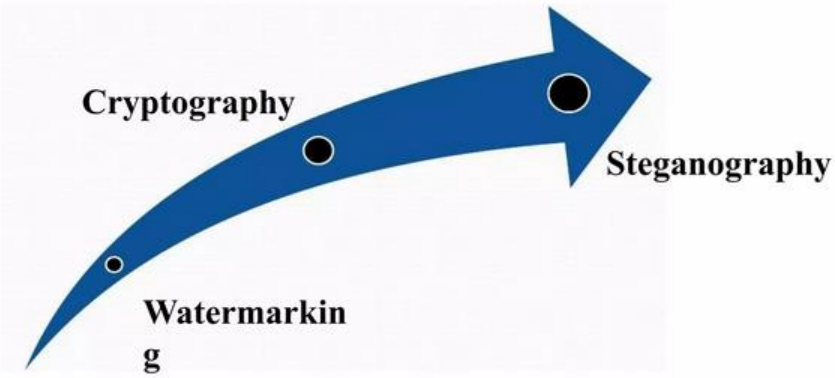




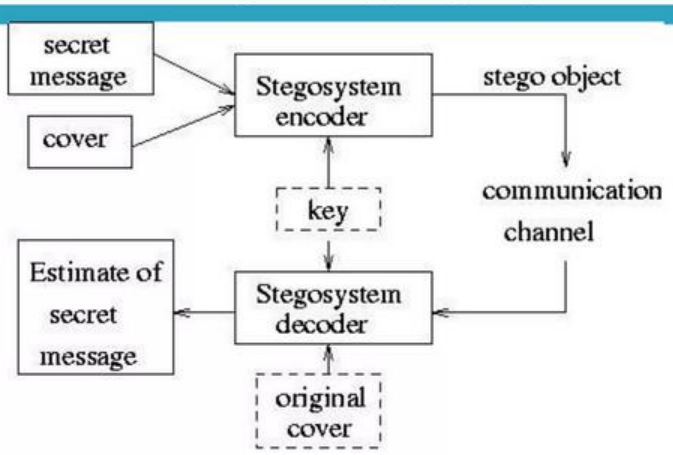
Steganography V/s Cryptography

Steganography	Cryptography
Unknown message passing	Known message passing
Steganography prevents discovery of the very existence of communication	Encryption prevents an unauthorized party from discovering the contents of a communication
Little known technology	Common technology
Technology still being develop for certain formats	Most of algorithm known by all
Once detected message is known	Strong current algorithm are resistant to attacks ,larger expensive computing power is required for cracking
Steganography does not alter the structure of the secret message	Cryptography alter the structure of the secret message

Evolution



Basic Steganography Model



Steganography Terms

- **Carrier or Cover File** - A Original message or a file in which hidden information will be stored inside of it .
- **Stego-Medium** - The medium in which the information is hidden.
- **Embedded or Payload** - The information which is to be hidden or concealed.
- **Steganalysis** - The process of detecting hidden information inside a file.

Types Of Stegosystems

- **There are three basic types of stegosystems**
  - **Pure stegosystems - no key is used.**
  - **Secret-key stegosystems - secret key is used.**
  - **Public-key stegosystems - public key is used**

Text Steganography

- Text steganography can be applied in the digital makeup format such as PDF, digital watermark or information hiding
- It is more difficult to realize the information hiding based on text. The simplest method of information hiding is to select the cover first, adopt given rules to add the phraseological or spelling mistakes, or replace with synonymy words.
- E.g 1] Textto setups some sentence structure in advance, fills in the empty location by arranged words,

  and then the text doesn't have phraseological mistakes, but have some word changes or morphology mistakes.

  2] TextHide hides the information in the manner of text overwriting and words' selection.

Text Steganography Methods

- Text Steganography in Markup Languages[HTML]
- Text Steganography in Specific characters in words
- Line shifting Method
- Word shifting
- Open spaces
- Semantic methods
- Character Encoding

Examples of Text Steganography

- An example of a message containing cipher text by German Spy in World War II:

  "A**p**parently *n**e**utral's p**r**otest i**s** t**h**oroughly d**i**scounted And i**g**nored. I**s**man h**a**rd h**i**t. B**l**ockade i**s**sue a**f**fects Pretext f**o**r e**m**bargo o**n** b**y** p**r**oducts, e**j**ecting s**u**ets a**n**d Veg**e**table o**i**ls.*"

- Taking the second letter in each word the following message emerges:

  Pershing sails from NY June 1.

Examples of Text Steganography

- Minor changes to shapes of characters

Examples of Text Steganography

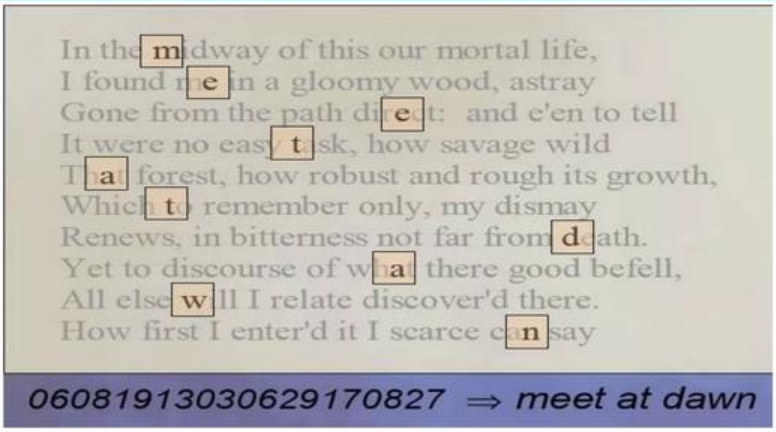


Image Steganography

- Using image files as hosts for steganographic messages takes advantage of the limited capabilities of the human visual system
- Some of the more common method for embedding messages in image files can be categorized into two main groups, image domain methods and transform domain methods

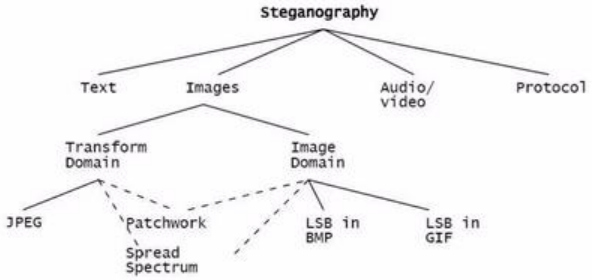


Image And Transform Domain

- Image – also known as spatial – domain techniques embed messages in the intensity of the pixels directly, while for transform – also known as frequency – domain, images are first transformed and then the message is embedded in the image
- Image domain techniques encompass bit-wise methods that apply bit insertion and noise manipulation and are sometimes characterised as “simple systems”
- Steganography in the transform domain involves the manipulation of algorithms and image transforms

LSB [Least Significant bit] Method

- Least significant bit (LSB) insertion is a common, simple approach to embedding information in a cover image
- The least significant bit (in other words, the 8th bit) of some or all of the bytes inside an image is changed to a bit of the secret message
- When using a 24-bit image, a bit of each of the red, green and blue colour components can be used, since they are each represented by a byte. In other words, one can store 3 bits in each pixel. An 800 × 600 pixel image, can thus store a total amount of 1,440,000 bits or 180,000 bytes of embedded data
- In its simplest form, LSB makes use of BMP images, since they use lossless compression

Example Of LSB Method

- A grid for 3 pixels of a 24-bit image can be as follows:

(00101101 00011100 11011100)
(10100110 11000100 00001100)
(11010010 10101101 01100011)

- When the number 200, which binary representation is 11001000, is embedded into the least significant bits of this part of the image, the resulting grid is as follows:

(00101101 00011101 11011100)
(10100110 11000101 00001100)
(11010010 10101100 01100011)

Example Of Image Steganography

Original Image	Stego Image	Original Image	Stego Image
			

Audio Steganography

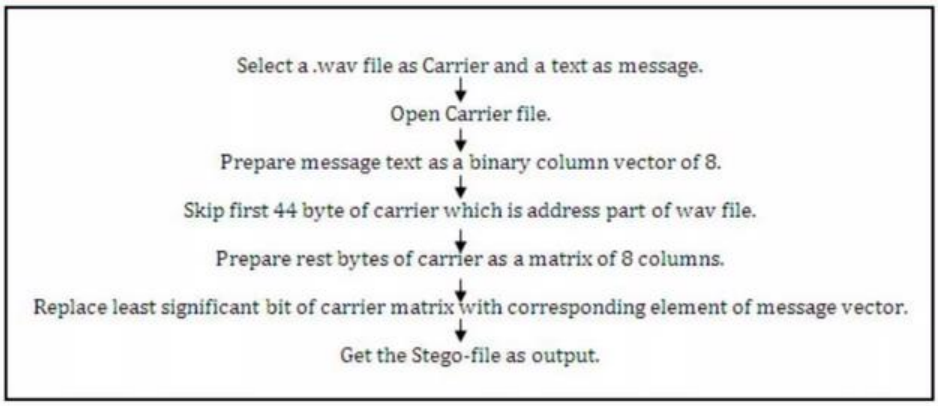
- Embedding secret messages into digital sound is known as

  audio Steganography.
- Audio Steganography methods can embed messages in WAV, AU, and even MP3 sound files.
- The properties of the human auditory system (HAS) are exploited in the process of audio Steganography

Audio Steganography

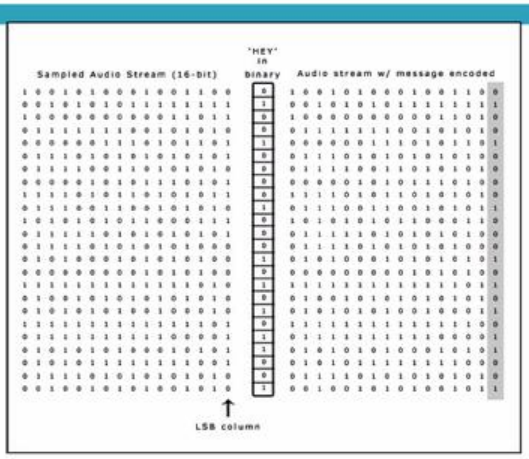
- To embed data secretly onto digital audio file there are few techniques introduced :
  - LSB Coding
  - Phase Coding
  - Parity Coding
  - Spread Spectrum

Flowchart Of Audio Steganography



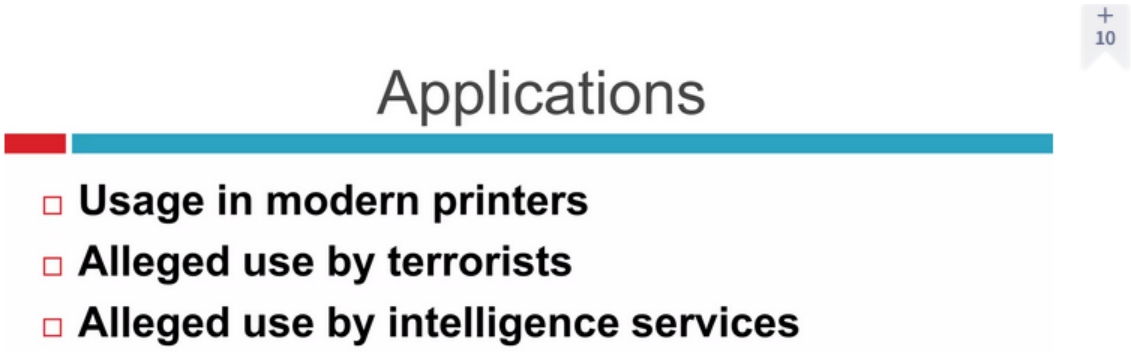
Example of LSB Method

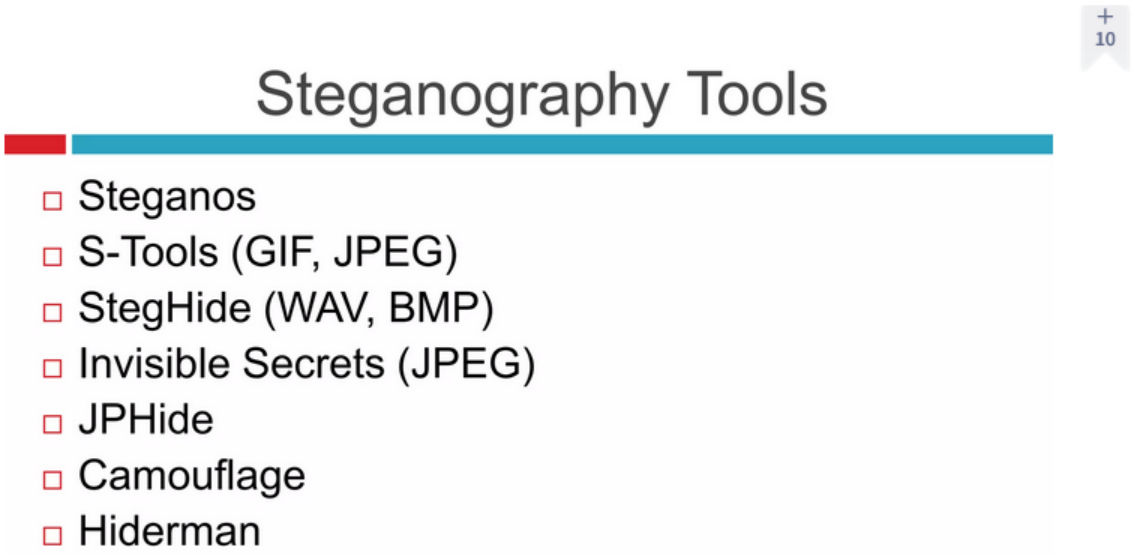
- The message 'HEY' is encoded in a 16-bit CD quality sample using the LSB method.
- Here the secret information is 'HEY' and the cover file is audio file. HEY is to be embedded inside the audio file. First the secret information 'HEY' and the audio file are converted into bit stream.
- The least significant column of the audio file is replaced by the bit stream of sectet information 'HEY'. The resulting file after embedding secret information 'HEY' is called Stego-file.



Applications

- **Confidential communication and secret data storing**
  - Steganography provides us with:
    - Potential capability to hide the existence of confidential data
    - Hardness of detecting the hidden (i.e., embedded) data
    - Strengthening of the secrecy of the encrypted data
- **Protection of data alteration**
- **Access control system for digital content distribution**

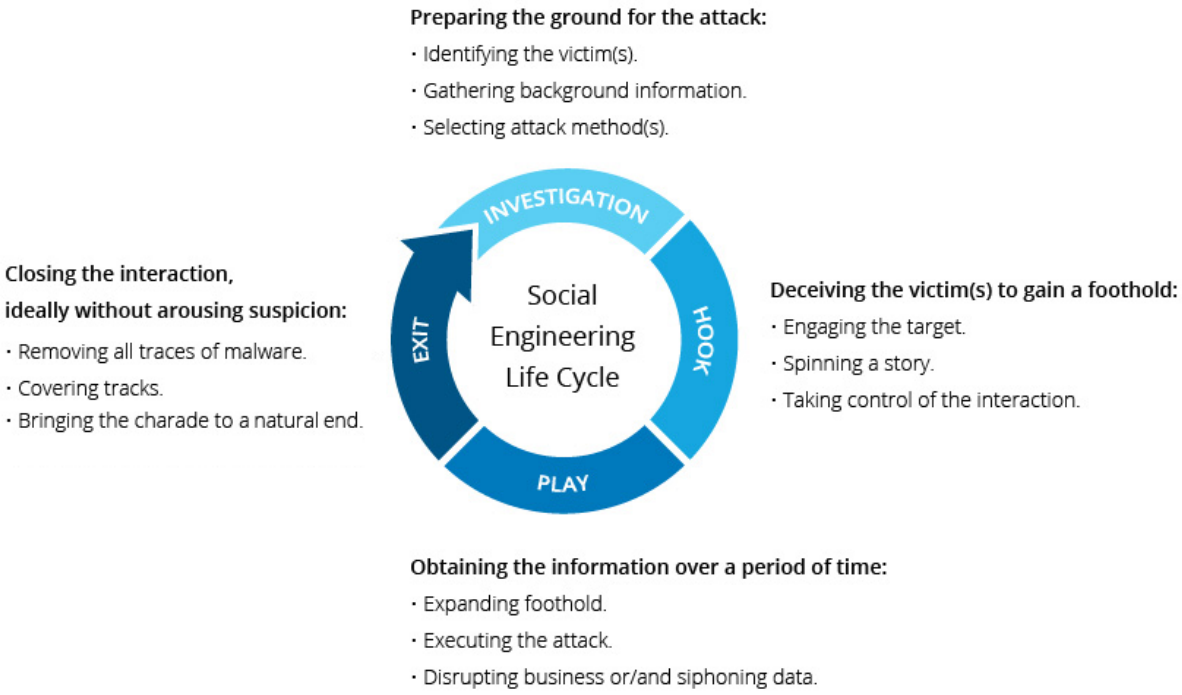




8 -Social engineering

Social engineering is the term used for a broad range of malicious activities accomplished through human interactions. It uses psychological manipulation to trick users into making security mistakes or giving away sensitive information.

Social engineering attacks happen in one or more steps. A perpetrator first investigates the intended victim to gather necessary background information, such as potential points of entry and weak security protocols, needed to proceed with the attack. Then, the attacker moves to gain the victim's trust and provide stimuli for subsequent actions that break security practices, such as revealing sensitive information or granting access to critical resources.



Social engineering attack techniques

Social engineering attacks come in many different forms and can be performed anywhere where human interaction is involved. The following are the five most common forms of digital social engineering assaults.

Baiting

As its name implies, baiting attacks use a false promise to pique a victim's greed or curiosity. They lure users into a trap that steals their personal information or inflicts their systems with malware.

The most reviled form of baiting uses physical media to disperse malware. For example, attackers leave the bait—typically malware-infected flash drives—in conspicuous areas where potential victims are certain to see them (e.g., bathrooms, elevators, the parking lot of a targeted company). The bait has an authentic look to it, such as a label presenting it as the company's payroll list.

Victims pick up the bait out of curiosity and insert it into a work or home computer, resulting in automatic malware installation on the system.

Baiting scams don't necessarily have to be carried out in the physical world. Online forms of baiting consist of enticing ads that lead to malicious sites or that encourage users to download a malware-infected application.

Scareware

Scareware involves victims being bombarded with false alarms and fictitious threats. Users are deceived to think their system is infected with malware, prompting them to install software that has no real benefit (other than for the perpetrator) or is malware itself. Scareware is also referred to as deception software, rogue scanner software and fraudware.

A common scareware example is the legitimate-looking popup banners appearing in your browser while surfing the web, displaying such text such as, "Your computer may be infected with harmful spyware programs." It either offers to install the tool (often malware-infected) for

you, or will direct you to a malicious site where your computer becomes infected.

Scareware is also distributed via spam email that doles out bogus warnings, or makes offers for users to buy worthless/harmful services.

Pretexting

Here an attacker obtains information through a series of cleverly crafted lies. The scam is often initiated by a perpetrator pretending to need sensitive information from a victim so as to perform a critical task.

The attacker usually starts by establishing trust with their victim by impersonating co-workers, police, bank and tax officials, or other persons who have right-to-know authority. The pretexter asks questions that are ostensibly required to confirm the victim's identity, through which they gather important personal data.

All sorts of pertinent information and records is gathered using this scam, such as social security numbers, personal addresses and phone numbers, phone records, staff vacation dates, bank records and even security information related to a physical plant.

Phishing

As one of the most popular social engineering attack types, phishing scams are email and text message campaigns aimed at creating a sense of urgency, curiosity or fear in victims. It then prods them into revealing sensitive information, clicking on links to malicious websites, or opening attachments that contain malware.

An example is an email sent to users of an online service that alerts them of a policy violation requiring immediate action on their part, such as a required password change. It includes a link to an illegitimate website—nearly identical in appearance to its legitimate version—prompting the unsuspecting user to enter their current credentials and new password. Upon form submittal the information is sent to the attacker.

Given that identical, or near-identical, messages are sent to all users in phishing campaigns, detecting and blocking them are much easier for mail servers having access to threat sharing platforms.

Social engineering prevention

Social engineers manipulate human feelings, such as curiosity or fear, to carry out schemes and draw victims into their traps. Therefore, be wary whenever you feel alarmed by an email, attracted to an offer displayed on a website, or when you come across stray digital media lying about. Being alert can help you protect yourself against most social engineering attacks taking place in the digital realm.

Moreover, the following tips can help improve your vigilance in relation to social engineering hacks.

- **Don’t open emails and attachments from suspicious sources** – If you don’t know the sender in question, you don’t need to answer an email. Even if you do know them and are suspicious about their message, cross-check and confirm the news from other sources, such as via telephone or directly from a service provider’s site.

Remember that email addresses are spoofed all of the time; even an email purportedly coming from a trusted source may have actually been initiated by an attacker.

- **Use multifactor authentication** – One of the most valuable pieces of information attackers seek are user credentials. Using multifactor authentication helps ensure your account's protection in the event of system compromise. Imperva Login Protect is an easy-to-deploy 2FA solution that can increase account security for your applications.
- **Be wary of tempting offers** – If an offer sounds too enticing, think twice before accepting it as fact. Googling the topic can help you quickly determine whether you're dealing with a legitimate offer or a trap.
- **Keep your antivirus/antimalware software updated** – Make sure automatic updates are engaged, or make it a habit to download the latest signatures first thing each day. Periodically check to make sure that the updates have been applied, and scan your system for possible infections.

9 –Security Threats on Social Networks

What is a Social Media Threat?

Social media is a virtual community that people use to share information, social network, and create and exchange information. Some characteristics of social media are that it allows people to share information, is internet-based, and can be both personal and business-focused. Social media sites like Facebook and Instagram can be great tools for those who wish to keep in touch with friends, family, and associates. Many people use these sites to post important life updates

like pictures of their children or updates on a new job. Another social media site many people use is YouTube, where they can watch music or informational videos to learn a new skill. LinkedIn is a social media site that many uses to find new job opportunities or to showcase their skills and talents.

While social media may have benefits when used correctly, there are also potential negative consequences of social media. Social media threats refer to things that weaken the security of a social media account. These threats are common, and those with social media accounts should know and understand what to look for so that they can avoid attacks.

How Social Media Threats Occur

The risks of social media that are associated with social media threats can endanger many people. These risks can affect different aspects of work and life. Some possible scenarios in which social media threats may occur are listed below:

Job-Related Information: Many social media threats arise from employees who share information on the internet. Hackers may be able to access critical company-related information because of an employee who left their account unprotected or did not keep their information private. This can lead to the company having a data breach, and all their information may be compromised.

Social Media Site Security: There have been many instances where people have had their pictures and information stolen by someone on a social media site. For example, depending on one's privacy settings, Facebook users can see and save pictures and information from

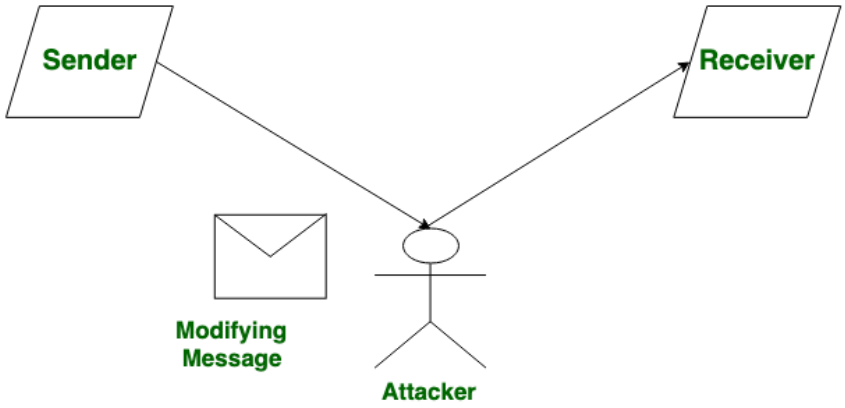
someone's profile. This is a common way that social media threats happen as people impersonate someone else and use their information as their own.

Malware Infection: This type of social media threat occurs as someone disguises malware as something legitimate, like a website where one can buy a popular product. They make the site look real, and users think nothing is wrong. When they click certain things on the site, malware is downloaded onto their device.

Two types of Attacks

- **Active attacks**
- **Passive attacks**

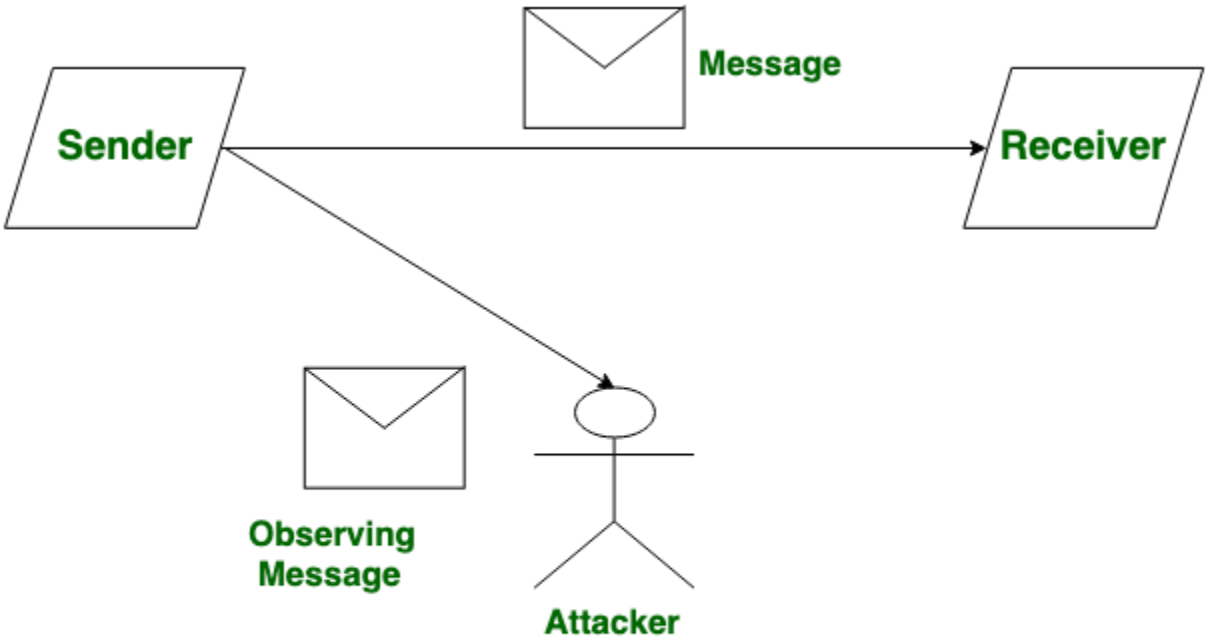
Active Attacks:
Active attacks are the type of attacks in which, The attacker efforts to change or modify the content of messages. Active Attack is dangerous to Integrity as well as availability. Due to active attack system is always damaged and System resources can be changed. The most important thing is that, In an active attack, Victim gets informed about the attack.



Active Attack

Passive Attacks:

Passive Attacks are the type of attacks in which, The attacker observes the content of messages or copies the content of messages. Passive Attack is a danger to Confidentiality. Due to passive attack, there is no harm to the system. The most important thing is that In a passive attack, Victim does not get informed about the attack.



Passive Attack

Difference between Active Attack and Passive Attack:

Active Attack	Passive Attack
In an active attack, Modification in information takes place.	While in a passive attack, Modification in the information does not take place.
Active Attack is a danger to **Integrity** as well as **availability**.	Passive Attack is a danger to **Confidentiality**.
In an active attack, attention is on prevention.	While in passive attack attention is on detection.
Due to active attacks, the execution system is always damaged.	While due to passive attack, there is no harm to the system.
In an active attack, Victim gets informed about the attack.	While in a passive attack, Victim does not get informed about the attack.
In an active attack, System resources can be changed.	While in passive attack, System resources are not changing.
Active attack influences the services of the system.	While in a passive attack, information and messages in the system or network are acquired.

Active Attack	Passive Attack
In an active attack, information collected through passive attacks is used during execution.	While passive attacks are performed by collecting information such as passwords, and messages by themselves.
An active attack is tough to restrict from entering systems or networks.	Passive Attack is easy to prohibit in comparison to active attack.
Can be easily detected.	Very difficult to detect.
The purpose of an active attack is to harm the ecosystem.	The purpose of a passive attack is to learn about the ecosystem.
In an active attack, the original information is modified.	In passive attack original information is Unaffected.
The duration of an active attack is short.	The duration of a passive attack is long.
The prevention possibility of active attack is High	The prevention possibility of passive attack is low.
Complexity is High	Complexity is low.

Social networks have become an integral part of our lives, but they also come with security threats that can compromise users' personal information and online safety. Here are some common security threats on social networks, along with examples, types, and prevention measures:

1. Phishing Attacks:

- **Example**: A user receives a message or email from a fake social network asking them to click on a link to verify their account, but the link leads to a malicious website.

- **Type**: Social engineering attack.

- **Prevention**: Be cautious of unsolicited messages or emails. Avoid clicking on suspicious links or providing personal information unless you verify the authenticity of the request. Enable two-factor authentication (2FA) for an added layer of security.

2. Identity Theft:

- **Example**: An attacker steals personal information from a user's social media profile and uses it for fraudulent activities.

- **Type**: Personal data breach.

- **Prevention:** Limit the amount of personal information you share on social networks. Use strong, unique passwords and enable 2FA. Regularly review your privacy settings and avoid accepting friend requests or connections from unknown individuals.

3. Malware Distribution:

- **Example:** A user clicks on a link shared on a social network that appears to be harmless but leads to a website hosting malware, which infects the user's device.

- **Type**: Malicious link or file.

- **Prevention**: Exercise caution when clicking on links or downloading files from unknown sources. Keep your device's antivirus software up to date and regularly scan for malware. Avoid downloading apps from unofficial app stores.

4**. Account Hijacking**:

- Example: An attacker gains unauthorized access to a user's social media account by guessing their password or using a leaked password from another breach.

- **Type**: Unauthorized access.

- **Prevention:** Use strong, unique passwords for your social media accounts and avoid reusing passwords across different platforms. Enable 2FA to provide an extra layer of security. Regularly monitor your account activity and report any suspicious activity immediately.

5. Social Engineering Attacks:

- **Example**: An attacker impersonates a friend or trusted entity on a social network to manipulate the user into revealing sensitive information or performing actions they wouldn't normally do.

- **Type**: Psychological manipulation.

- **Prevention**: Be cautious of requests for sensitive information or financial transactions, especially from unfamiliar sources. Verify the identity of individuals or organizations before providing any information. Educate yourself about social engineering tactics to recognize and avoid them.

6. Privacy Violations:

- Example: A social network's privacy settings are configured improperly, allowing unauthorized users or applications to access **personal information.**

- **Type:** Configuration or platform vulnerability.

- **Prevention**: Regularly review and adjust your privacy settings on social networks. Limit the amount of personal information you share publicly. Be mindful of the permissions you grant to third-party applications and avoid granting access to unnecessary data.

7. Cyberbullying and Harassment:

- **Example:** A user experiences persistent online harassment, bullying, or stalking through a social network.

- **Type**: Abusive behavior.

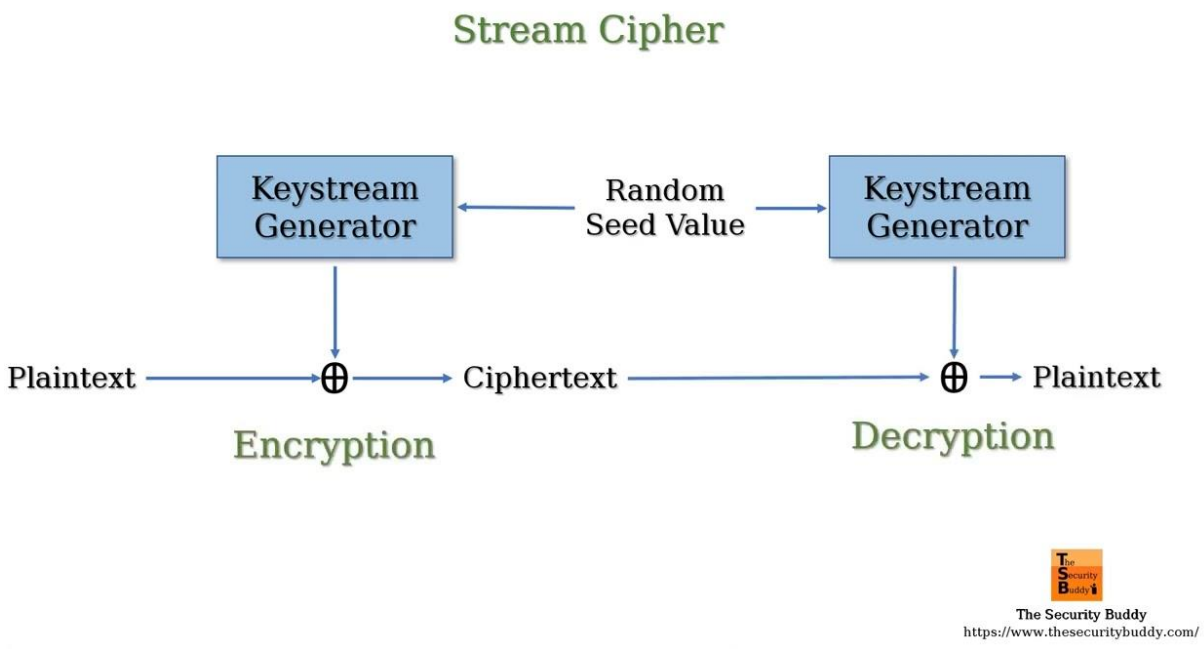
- **Prevention**: Be cautious about accepting friend requests or connections from unknown individuals. Report and block users engaging in abusive behavior. Adjust your privacy settings to limit interactions from unknown users. If you're a victim of cyberbullying, document the incidents and report them to the social network's support team.

Prevention measures for social network security threats include maintaining strong passwords, enabling 2FA, being cautious of sus…

Stream Cipher

A stream cipher is an encryption technique that works byte by byte to transform plain text into code that's unreadable to anyone without the proper key.

Stream ciphers are linear, so the same key both encrypts and decrypts messages. And while cracking them can be difficult, hackers have managed to do it.



How do stream ciphers work?

All cryptographic methods aim to scramble data to hide it from outsiders. But unlike their counterparts, stream ciphers work on each bit

of data in a message rather than chunking the message into groups and encrypting them in blocks.

Stream ciphers rely on:

- **Plaintext.** You must have a message you'd like to encode.
- **Keystreams**. A set of random characters replaces those in the plaintext**.** They could be numbers, letters, or symbols.
- **Ciphertext.** This is the encoded message.

Generating a key is a complicated mathematical process. Even so, most computers can push through each step in seconds.

Bits of plaintext enter the stream cipher, and the cipher manipulates each bit with the mathematical formula. The resulting text is completely scrambled, and the recipient can't read it without the proper key.

With the right key, a recipient can push the ciphertext back through the stream cipher and transform the garbled data back to plaintext.

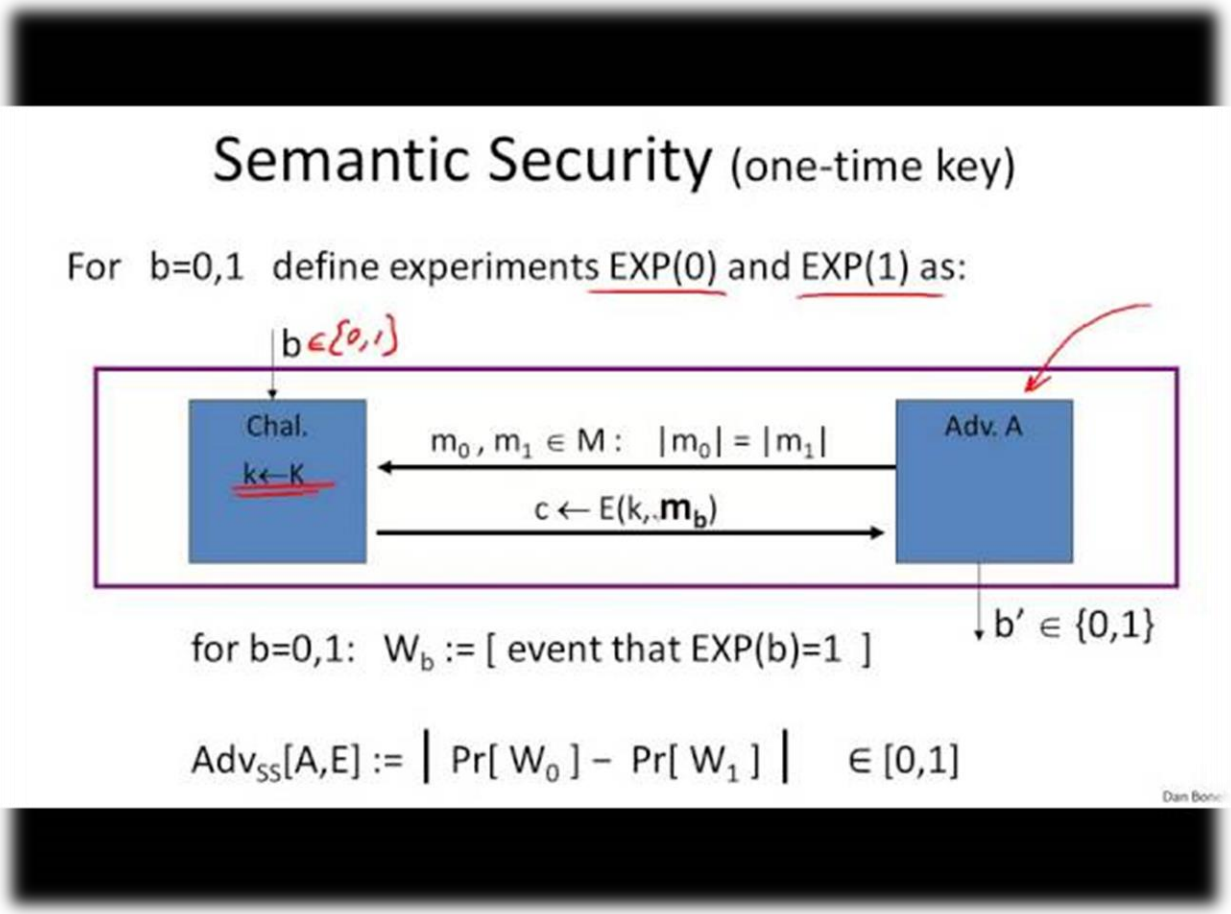
There are two main types of stream ciphers, and they each work slightly differently.

- **Synchronous stream ciphers**: A secret key generates keystreams, and they're made independently of both the plaintext and the ciphertext.
- **Self-synchronizing stream ciphers:** They use a secret key, but they include another form of randomization to make hacking harder.

Stream ciphers aren't the only tools at your disposal. You could also use block ciphers. Block ciphers break messages down into pieces, and then each piece moves through an encryption algorithm.

10 -Semantic Security

Semantic security is a concept in cryptography that ensures the confidentiality of encrypted data even when an adversary has significant knowledge about the plaintext and the encryption algorithm being used. It aims to prevent any meaningful information from being learned from the ciphertext alone.



To understand semantic security, let's consider an example using a symmetric encryption scheme called the Advanced Encryption Standard (AES).

Suppose Alice wants to send a confidential message to Bob. She encrypts the message using AES and sends the ciphertext to Bob. Eve, an eavesdropper, intercepts the ciphertext and attempts to decrypt it to gain knowledge of the original message.

In a semantically secure encryption scheme, even if Eve has access to multiple ciphertexts and knows the corresponding plaintext for some of them, she should not be able to determine the plaintext for any new ciphertext without a significant amount of effort.

Let's illustrate this with an example:

1. Alice has a plaintext message: "HELLO" and wants to send it to Bob.
2. Alice encrypts the message using AES, and let's say the resulting ciphertext is "YXZAB."
3. Eve, the eavesdropper, intercepts the ciphertext "YXZAB."

In a semantically secure encryption scheme, Eve cannot obtain any meaningful information about the original plaintext "HELLO" solely from the ciphertext "YXZAB." Even if she knows that "HELLO" corresponds to "YXZAB" in this particular case, she shouldn't be able to

use this information to decrypt any new ciphertext without an impractical amount of computational effort.

Semantic security provides a strong level of confidentiality by ensuring that the encryption scheme hides the underlying message's meaning, even if an adversary has access to some ciphertexts and has partial knowledge about the plaintexts.

It's important to note that semantic security does not guarantee protection against other forms of attacks or security threats, such as traffic analysis or side-channel attacks. It specifically focuses on protecting the confidentiality of the encrypted data.

11 –Block Ciphers and Pseudorandom Functions

What is a block cipher?

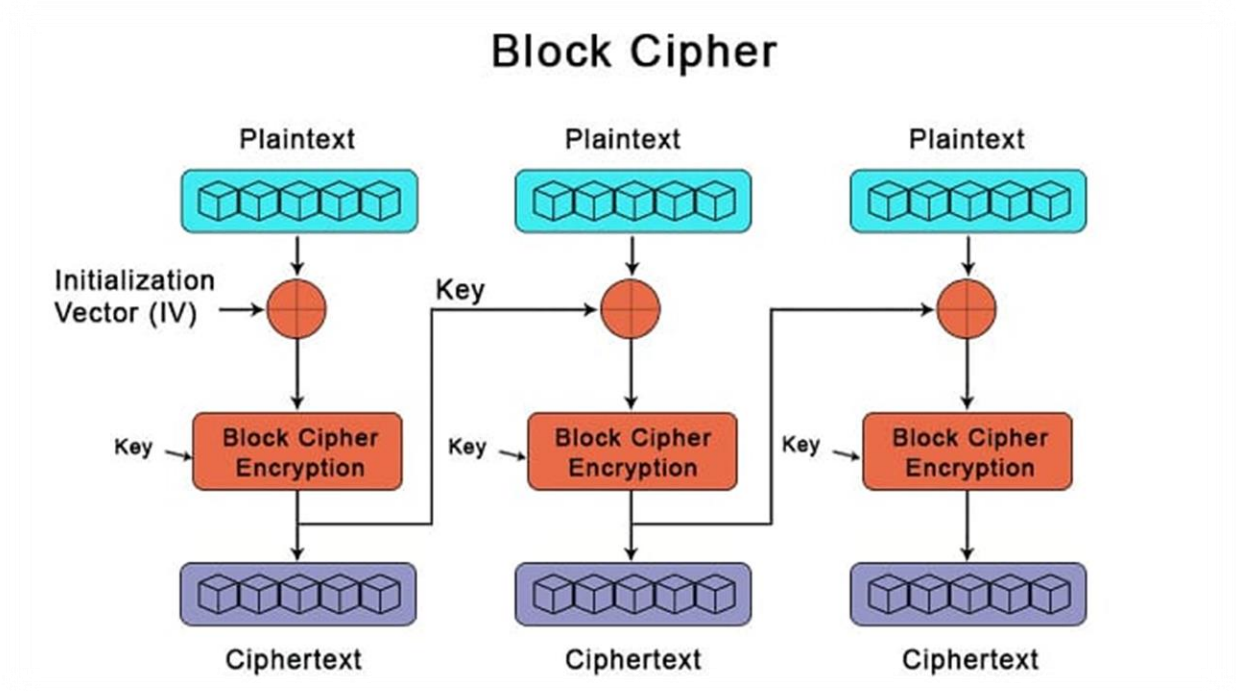
A block cipher is a method of encrypting data in blocks to produce ciphertext using a cryptographic key and algorithm. The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time. Most modern block ciphers are designed to encrypt data in fixed-size blocks of either 64 or 128 bits.

How does a block cipher work?

A block cipher uses a symmetric key and algorithm to encrypt and decrypt a block of data. A block cipher requires an initialization vector (IV) that is added to the input plaintext in order to increase the keyspace of the cipher and make it more difficult to use brute force to break the key. The IV is derived from a random number generator, which is combined with text in the first block and the key to ensure all subsequent blocks result in ciphertext that does not match that of the first encryption bloc

The block size of a block cipher refers to the number of bits that are processed together. Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are both symmetric block ciphers.

The DES block cipher was originally designed by IBM in 1975 and consisted of 64-bit blocks and a 56-bit key. This cipher is not considered secure anymore, due to the short key size, and was replaced in 1998 by AES. AES uses a 128-bit block size and a 128-, 192- or 256-bit key size.



What are the different modes of operation in block cipher?

Block ciphers only encrypt messages that are the same size as their block length, so each block of plaintext with more or less blocks needs to be encrypted separately. The following block cipher modes of operation define how these blocks are encrypted:

- **Electronic codebook (ECB) mode**. ECB mode is used to electronically code messages as their plaintext form. It is the simplest of all block cipher modes of operation. It does not add any randomness to the key stream, and it is the only mode that can be used to encrypt a single-bit stream. This means that each plaintext symbol, such as a character from the plaintext alphabet, is converted into a ciphertext symbol using the cipher's key and a substitution alphabet. Each plaintext block is encrypted independently of all the other blocks. If a plaintext block is only 8 bytes, only 8 bytes of the key are used; if a plaintext block is 100 bytes, all 100 bytes of the key are used.
- **Cipher block chaining (CBC) mode**. CBC mode is a method of encrypting data that ensures that each block of plaintext is combined with the previous ciphertext block before being encrypted. The symmetric key algorithm creates a ciphertext that depends on all plaintext blocks processed before it in a data stream. This is done to ensure that each block of the ciphertext is dependent on all of the previous blocks. Each plaintext block is XORed (exclusive OR) with the previous ciphertext block before being encrypted with the cipher algorithm. CBC mode is used in a variety of security applications. For example, Secure Sockets Layer/Transport Layer Security uses CBC mode to encrypt data that is transferred over the internet.

- **Ciphertext feedback (CFB) mode**. In contrast to CBC mode, which encrypts a set number of bits of plaintext at a time, it is sometimes necessary to encrypt and transfer plaintext values instantly, one at a time. Like CBC, CFB also uses an IV. CFB uses a block cipher as a component of a random number generator. In CFB mode, the previous ciphertext block is encrypted, and the output is XORed with the current plaintext block to create the current ciphertext block. The XOR operation conceals plaintext patterns.
- **Output feedback (OFB) mode.** OFB mode can be used with any block cipher and is similar in some respects to CBC mode. It uses a feedback mechanism, but instead of XORing the previous block of ciphertext with the plaintext before encryption, in OFB mode, the previous block of ciphertext is XORed with the plaintext after it is encrypted.
- **Counter (CTR) mode**. CTR mode uses a block chaining mode of encryption as a building block. The process of encrypting data is performed by XORing the plaintext with a sequence of pseudorandom values, each of which is generated from the ciphertext using a feedback function. The CTR encryption process can be visualized as a series of XORs between blocks of plaintext and corresponding blocks of ciphertext.

Pseudorandom Function

A pseudorandom function (PRF) is a cryptographic construction that takes an input and produces an output that appears random but is actually deterministic. PRFs are commonly used in cryptography for various purposes, such as generating secret keys, providing integrity checks, and ensuring data confidentiality.

The basic idea behind a PRF is to mimic the behavior of a truly random function while being computationally efficient and deterministic. It takes an input, called the key, along with additional data, and produces an output that is indistinguishable from random for any practical purpose. However, given the same input, a PRF will always produce the same output, which is the key property that distinguishes it from a truly random function.

The security of a PRF depends on the underlying cryptographic algorithms and the strength of the key used. The key is typically a random value chosen from a large keyspace. The more bits in the key, the larger the keyspace, and the harder it becomes for an adversary to guess the key through brute force or other attacks.

A commonly used PRF is the HMAC (Hash-based Message Authentication Code), which combines a cryptographic hash function (such as SHA-256) with a secret key. The HMAC algorithm applies a specific computation to the input and the key, resulting in a fixed-size output that is considered pseudorandom and secure if the underlying hash function is secure.

To use a PRF, you typically provide it with a key and input data. The PRF applies its internal algorithm to the key and input, producing an output that can be used in various cryptographic protocols. The output appears random to an observer who does not know the key, but someone with the key can reproduce the output for the same input.

It's important to note that while a PRF provides pseudorandomness, it is not truly random. If the underlying key or algorithm is compromised, the output of the PRF may be predictable or vulnerable to attacks. Therefore, the security of a PRF relies on using strong keys and ensuring the confidentiality of the key material.

Overall, a pseudorandom function is a cryptographic construction that emulates the behavior of a random function while being deterministic and computationally efficient. It is widely used in cryptography for various purposes, providing security and confidentiality in many applications.

12 –Chosen Plain Text Security and modes of operation

Chosen plaintext security (CPA security) is a property of cryptographic systems, particularly symmetric encryption algorithms, that ensures the security of the system even if an adversary can obtain the ciphertexts of chosen plaintexts. In other words, an adversary has the ability to choose arbitrary plaintext messages and obtain their corresponding ciphertexts from the encryption oracle.

CPA security is important because it reflects real-world scenarios where an adversary may have some control over the plaintexts they submit for encryption. It guarantees that even with such knowledge, the adversary

cannot gain any advantage or obtain information about the encryption key or the plaintexts of other messages.

To achieve CPA security, encryption algorithms are typically used in combination with a mode of operation. A mode of operation determines how the encryption algorithm is applied to multiple blocks of data, as many encryption algorithms operate on fixed-size blocks (e.g., 128 bits).

Here are some commonly used modes of operation for symmetric encryption algorithms:

1. Electronic Codebook (ECB): This is the simplest mode of operation. Each block of plaintext is encrypted independently with the same key, resulting in separate ciphertext blocks. However, ECB can be insecure for certain types of data, as identical plaintext blocks will result in identical ciphertext blocks.

2. Cipher Block Chaining (CBC): In CBC mode, each plaintext block is XORed with the previous ciphertext block before encryption. This introduces feedback and ensures that each ciphertext block depends on all the previous plaintext blocks. A random initialization vector (IV) is used as the first input to the XOR operation.

3. Counter (CTR): CTR mode converts the encryption algorithm into a stream cipher. Instead of encrypting blocks sequentially, it encrypts a

unique counter value for each block, generating a keystream. The keystream is XORed with the plaintext to produce the ciphertext.

4. Galois/Counter Mode (GCM): GCM combines the counter mode (CTR) with an authentication mechanism called Galois Message Authentication Code (GMAC). It provides both confidentiality and integrity/authentication of the ciphertext. GCM is widely used in secure communication protocols, such as TLS.

These are just a few examples of modes of operation. There are other modes, such as CFB (Cipher Feedback Mode), OFB (Output Feedback Mode), and XTS (XEX-based Tweaked CodeBook Mode), each designed to address specific requirements or constraints.

When choosing a mode of operation, it is important to consider factors such as security guarantees, performance, parallelizability, and resistance against known attacks. Additionally, it is crucial to properly handle the initialization vectors (IVs) and ensure they are unique and unpredictable to maintain the security of the encryption process.

13 –The DES and AES block ciphers

DES

Data encryption standard (DES) has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found

slightly on the decline. DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text go as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits.

The basic idea is shown in the figure:

We have mentioned that DES uses a 56-bit key. Actually, The initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.

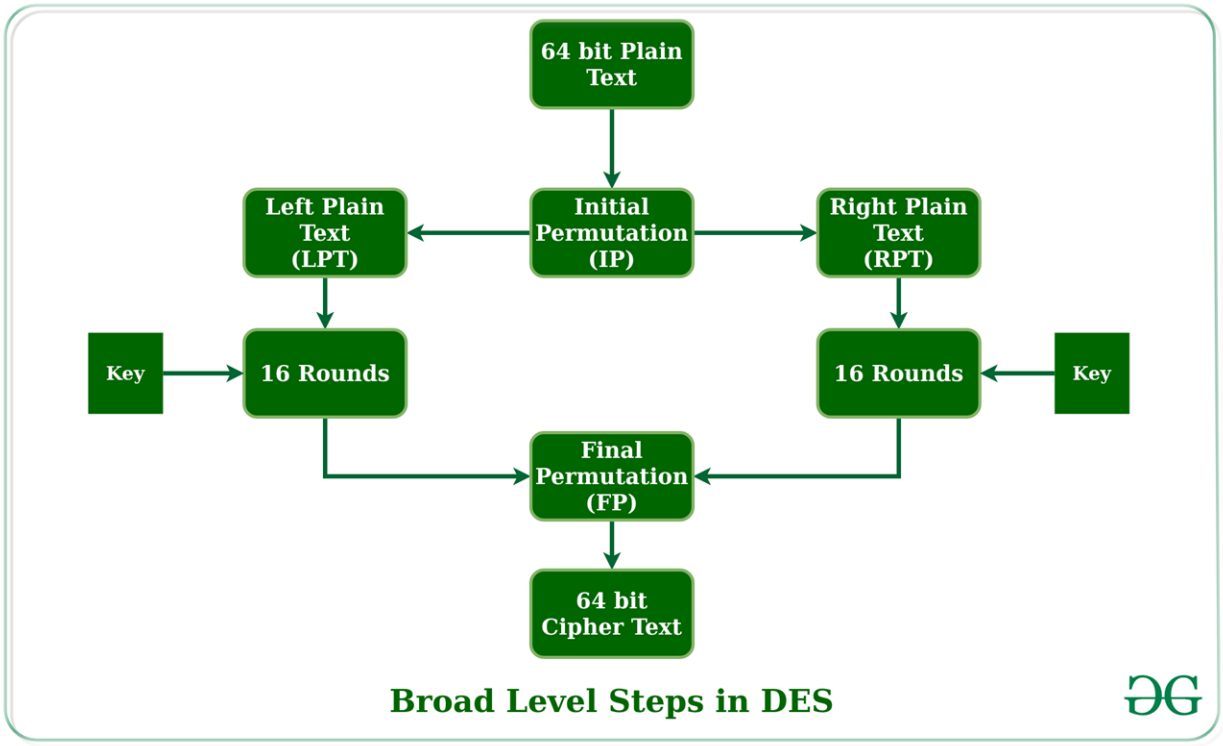
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Figure - discording of every 8th bit of original key

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key.

DES is based on the two fundamental attributes of cryptography: substitution (also called confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

- In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
- The initial permutation is performed on plain text.
- Next, the initial permutation (IP) produces two halves of the permuted block; saying Left Plain Text (LPT) and Right Plain Text (RPT).
- Now each LPT and RPT go through 16 rounds of the encryption process.
- In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
- The result of this process produces 64-bit ciphertext.



Broad Level Steps in DES

Initial Permutation (IP):

As we have noted, the initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in the figure. For example, it says that the IP

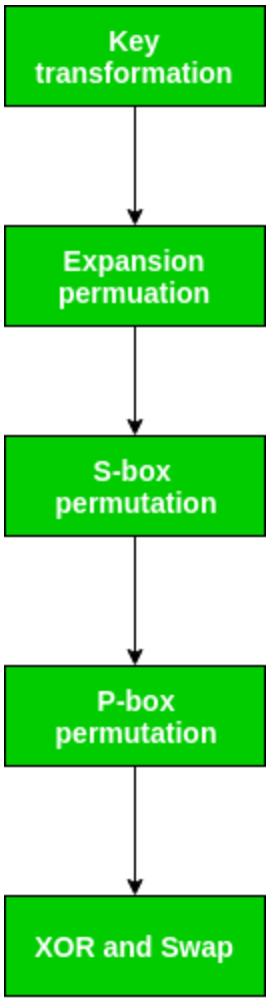
replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block, and so on.

This is nothing but jugglery of bit positions of the original plain text block. the same rule applies to all the other bit positions shown in the figure.

58	50	42	34	26	18	10	2	60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6	64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1	59	51	43	35	27	19	11	3
61	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure - Initial permutation table

As we have noted after IP is done, the resulting 64-bit permuted text block is divided into two half blocks. Each half-block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad-level steps outlined in the figure.



Step-1: Key transformation:

We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called key transformation. For this, the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

For example: if the round numbers 1, 2, 9, or 16 the shift is done by only one position for other rounds, the circular shift is done by two

positions. The number of key bits shifted per round is shown in the figure.

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
#key bits shifted	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Figure - number of key bits shifted per round

After an appropriate shift, 48 of the 56 bits are selected.From the 48 we might obtain 64 or 56 bits based on requirement which helps us to recognize that this model is very versatile and can handle any range of requirements needed or provided. for selecting 48 of the 56 bits the table is shown in the figure given below. For instance, after the shift, bit number 14 moves to the first position, bit number 17 moves to the second position, and so on. If we observe the table , we will realize that it contains only 48-bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as a selection of a 48-bit subset of the original 56-bit key it is called Compression Permutation.

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Figure - compression permutation

Because of this compression permutation technique, a different subset of key bits is used in each round. That makes DES not easy to crack.

Step-2: Expansion Permutation:

Recall that after the initial permutation, we had two 32-bit plain text areas called Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called expansion permutation. This happens as the 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4-bit block of the previous step is then expanded to a corresponding 6-bit block, i.e., per 4-bit block, 2 more bits are added.

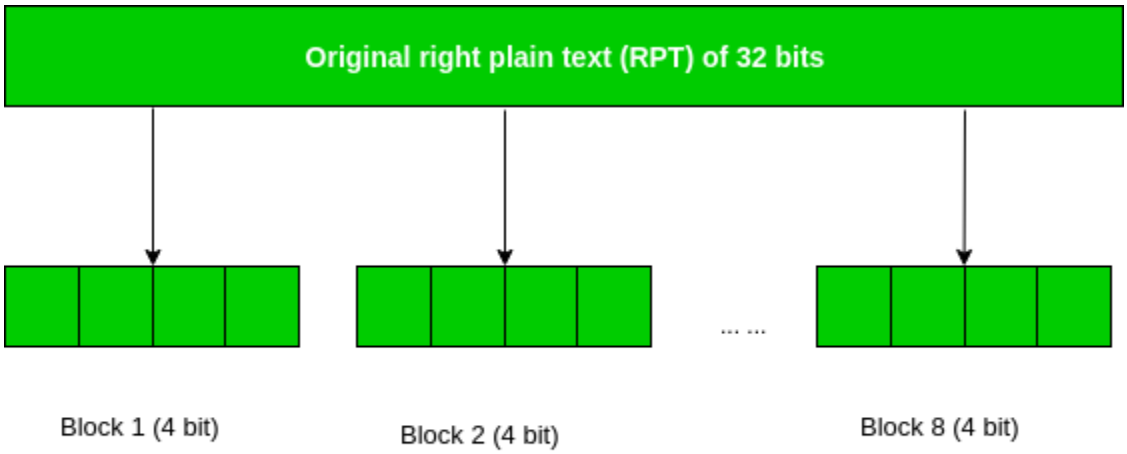


Figure - division of 32 bit RPT into 8 bit blocks

This process results in expansion as well as a permutation of the input bit while creating output. The key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the 32-bit RPT to 48-bits. Now the 48-bit key is XOR with 48-bit RPT and the resulting output is given to the next step, which is the S-Box substitution.

AES

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Points to remember

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

Working of the cipher :

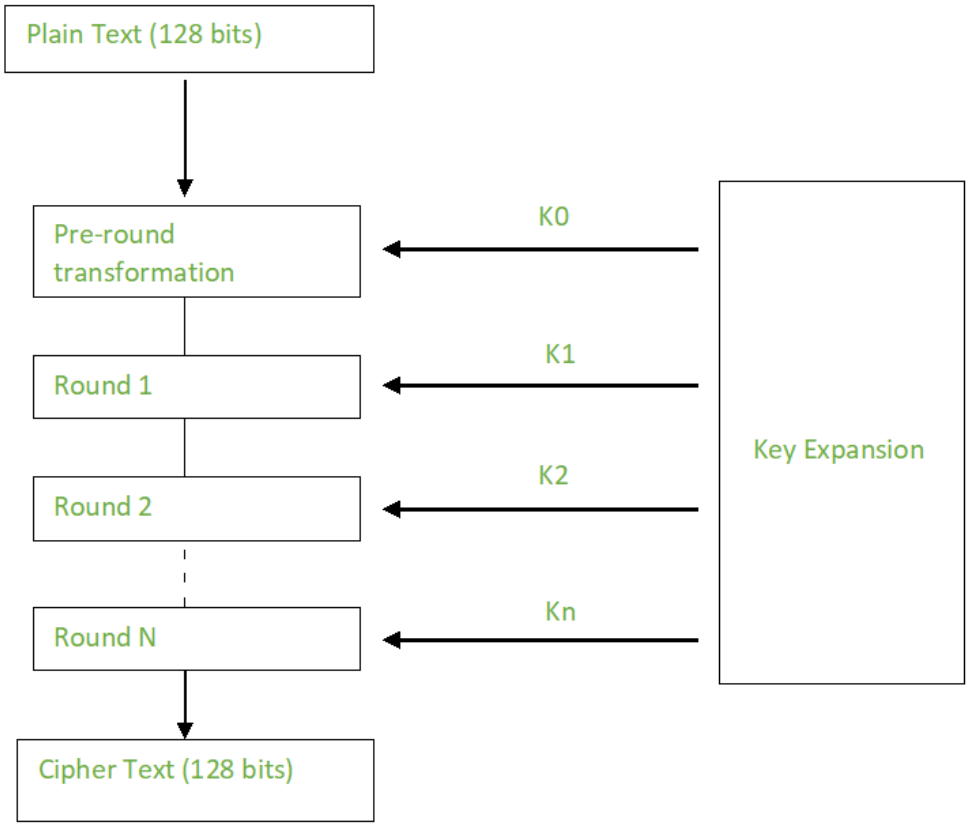
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

- 128 bit key – 10 rounds
- 192 bit key – 12 rounds
- 256 bit key – 14 rounds

Creation of Round keys :

A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.



Encryption :

AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.

[b0 | b4 | b8 | b12 |

| b1 | b5 | b9 | b13 |

| b2 | b6 | b10| b14 |

| b3 | b7 | b11| b15]

Each round comprises of 4 steps :

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

The last round doesn't have the MixColumns round.

The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.

SubBytes :

This step implements the substitution.

In this step each byte is substituted by another byte. Its performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16 byte (4 x 4) matrix like before.

The next two steps implement the permutation.

ShiftRows :

This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

(A left circular shift is performed.)

[b0	b1	b2	b3]		[b0	b1	b2	b3]
| b4	b5	b6	b7 |	->	| b5	b6	b7	b4 |
| b8	b9	b10	b11 |		| b10	b11	b8	b9 |
[b12	b13	b14	b15]		[b15	b12	b13	b14]

MixColumns :

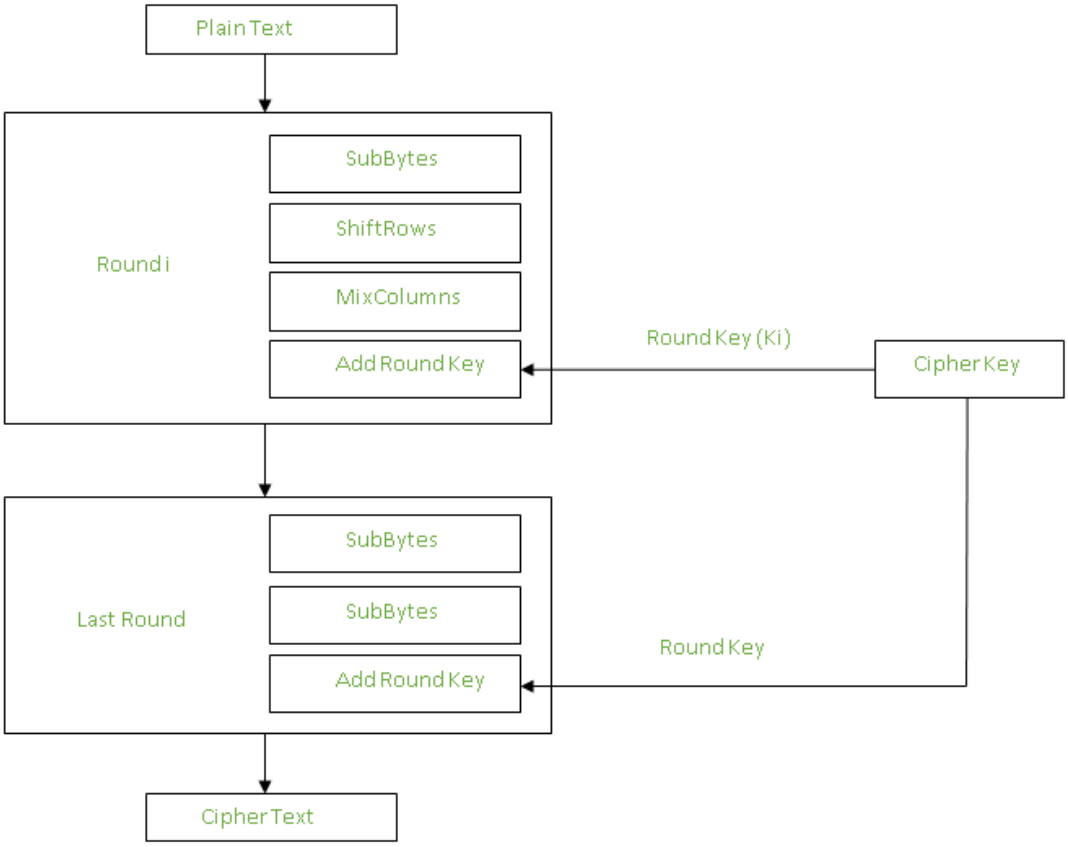
This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.

This step is skipped in the last round.

$$\begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix}$$

Add Round Keys :

Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.



After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

Decryption :

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes.Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption is as follows :

- Add round key
- Inverse MixColumns
- ShiftRows
- Inverse SubByte

The decryption process is the encryption process done in reverse so i will explain the steps with notable differences

Inverse MixColumns :

This step is similar to the MixColumns step in encryption, but differs in the matrix used to carry out the operation.

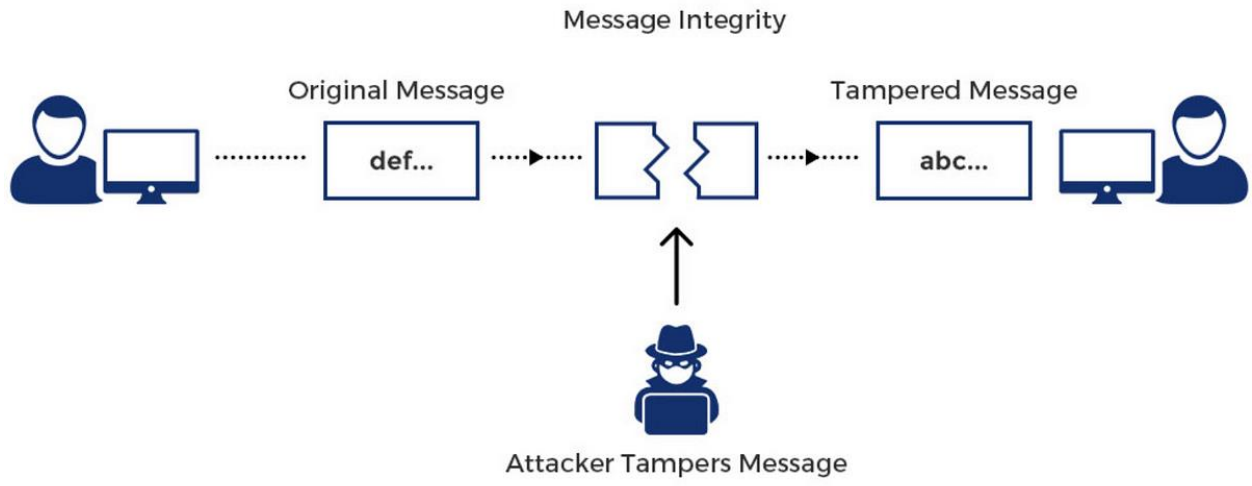
$$\begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

Inverse SubBytes : Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption

14 – Message Integrity

Message integrity refers to the property of ensuring that a message remains unaltered and intact during transit or storage. It ensures that the message has not been modified, tampered with, or corrupted in any way by an unauthorized entity.

Maintaining message integrity is crucial in various scenarios, such as secure communication protocols, digital signatures, data storage, and financial transactions. It ensures that the receiver can trust the authenticity and integrity of the received message and can detect any unauthorized modifications.



TYPES

1. Hash Functions: Hash functions are mathematical algorithms that take an input (message) and produce a fixed-size output called a hash value or message digest. Hash functions are designed to be fast and deterministic, meaning the same input always produces the same output. Any modification to the input will result in a different hash value. By comparing the hash values of the sent and received messages, one can verify the message integrity. Commonly used hash functions include SHA-256, SHA-3, and MD5 (though it is considered weak for security purposes).

2. Message Authentication Codes (MAC): MACs are cryptographic constructs that provide both message integrity and authentication. They use a secret key to generate a tag, which is appended to the message.

The receiver can verify the integrity by recalculating the MAC using the same key and comparing it with the received MAC. HMAC (Hash-based Message Authentication Code) is a widely used MAC algorithm that combines a cryptographic hash function with a secret key.

3**. Digital Signatures:** Digital signatures provide integrity, authenticity, and non-repudiation. They involve the use of public-key cryptography, where the sender signs the message with their private key, and the receiver verifies the signature using the corresponding public key. The signature is calculated using a hash function and the sender's private key, ensuring that any alteration to the message will invalidate the signature. Common digital signature algorithms include RSA and ECDSA (Elliptic Curve Digital Signature Algorithm).

4. Checksums: Checksums are simple error-detection mechanisms that add up the values of all the bytes in a message and include the result as a checksum value. The receiver recalculates the checksum and compares it with the received value. If they match, the message is considered intact. However, checksums do not provide the same level of security as cryptographic hash functions or MACs and are more susceptible to intentional tampering.

Working

The working of message integrity involves the use of cryptographic mechanisms to ensure that a message remains unaltered and intact during transit or storage. Here's a step-by-step explanation of how message integrity is achieved:

1. Sender computes the integrity check value:

The sender generates an integrity check value for the message using a cryptographic mechanism such as a Message Authentication Code (MAC) or a cryptographic hash function. The integrity check value is computed based on the content of the message and a secret key known only to the sender and the intended receiver.

2. Sender appends the integrity check value to the message:

The sender combines the integrity check value with the original message, typically by appending it to the message. This creates a single entity that includes both the message content and the integrity check value.

3. Sender transmits the message:

The sender sends the complete message, including the original message content and the appended integrity check value, to the receiver. The transmission can occur over various communication channels, such as networks, the internet, or physical media.

4. Receiver receives the message:

The receiver receives the message sent by the sender.

5. Receiver recomputes the integrity check value:

The receiver independently recomputes the integrity check value for the received message using the same cryptographic mechanism and the shared secret key. This ensures that the receiver has a reference value to compare with the integrity check value received from the sender.

6. Receiver compares the integrity check values:

The receiver compares the integrity check value computed in the previous step with the integrity check value received from the sender. If the two values match, it indicates that the message has remained unaltered during transit, and its integrity is intact. If they do not match, it signifies that the message has been modified or corrupted.

By comparing the integrity check values, the receiver can verify the authenticity and integrity of the message. If the values match, it assures the receiver that the message has not been tampered with and can be trusted. If the values do not match, it indicates potential tampering or corruption of the message, and appropriate actions can be taken, such as discarding the message or requesting a retransmission.

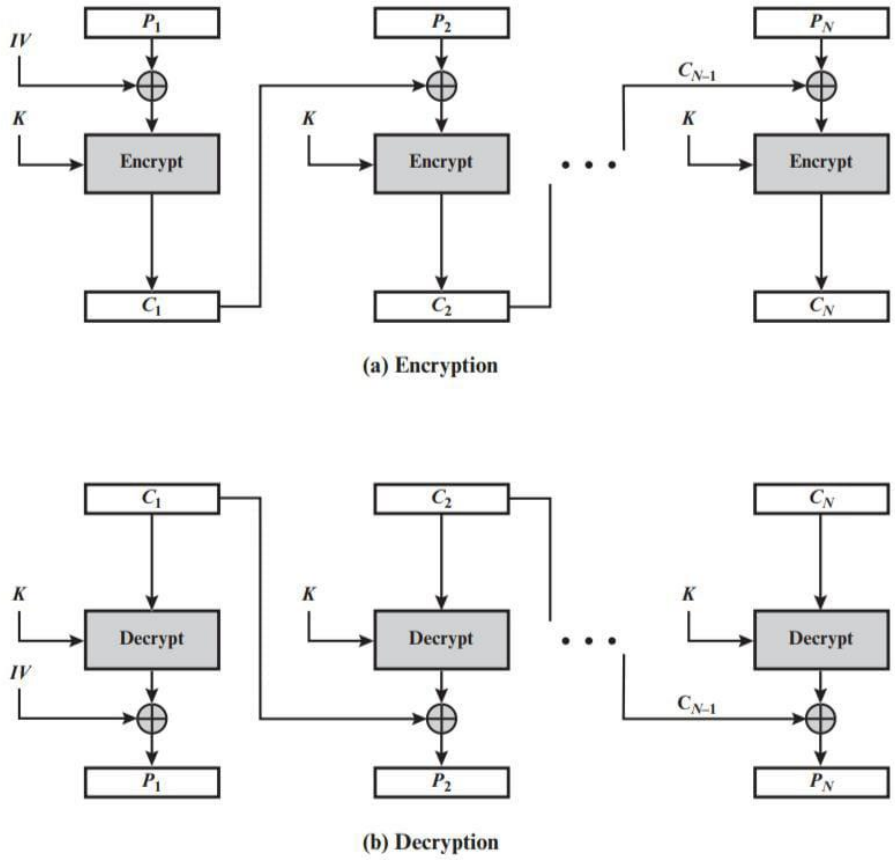
The use of cryptographic mechanisms like MACs or cryptographic hash functions ensures that even a slight modification to the message will result in a significantly different integrity check value. This property allows the receiver to detect any unauthorized modifications to the message content.

It's worth noting that message integrity mechanisms alone do not provide confidentiality. They primarily focus on detecting modifications to the message rather than protecting its confidentiality. For secure communication, encryption mechanisms should be used in combination with message integrity mechanisms to ensure both confidentiality and integrity.

15 –CBC-MAC, HMAC ,PMAC and CW-MAC

CBC

CBC (Cipher Block Chaining) is a mode of operation for symmetric encryption algorithms, such as the Advanced Encryption Standard (AES). CBC is designed to provide confidentiality and protect the privacy of data by introducing an element of feedback and interdependence between the blocks of plaintext and ciphertext.



Here's how CBC mode works:

- **Block Encryption**: CBC operates on fixed-size blocks of data, typically 128 bits (16 bytes) for AES. The plaintext message is divided into blocks, and each block is encrypted independently using the chosen encryption algorithm (e.g., AES) with a secret encryption key.

- **Initialization Vector (IV)**: CBC requires the use of an Initialization Vector, which is a random value of the same block size as the encryption algorithm (e.g., 128 bits for AES). The IV is XORed with the first plaintext block before encryption.

- **Chaining**: For subsequent blocks, prior to encryption, CBC XORs the plaintext block with the previous ciphertext block. This introduces a feedback mechanism where each ciphertext block depends on all previous plaintext blocks. The result of the XOR operation is then encrypted using the encryption algorithm.

- **Encryption Process**: The encryption process continues for all blocks of the plaintext, each depending on the previous ciphertext block through the XOR operation. The resulting ciphertext blocks are generated.

To decrypt the ciphertext and recover the original plaintext, the reverse process is followed:

- **Block Decryption:** Each ciphertext block is decrypted independently using the decryption algorithm (e.g., AES) and the same secret encryption key.

- **Chaining**: Similar to encryption, for each decrypted ciphertext block (except the first one), CBC XORs it with the previous ciphertext block, which effectively reverses the XOR operation performed during encryption.

- **Initialization Vector (IV**): In CBC decryption, the IV is XORed with the first decrypted ciphertext block to recover the first block of the plaintext.

- **Decryption Process**: The decryption process continues for all ciphertext blocks, each block being decrypted and XORed with the previous ciphertext block to recover the original plaintext blocks.

CBC mode provides several security benefits:

Confidentiality: By XORing each plaintext block with the previous ciphertext block, CBC introduces a level of diffusion, making it difficult for an attacker to extract meaningful information from the ciphertext.

Error Propagation: Any errors or alterations in the ciphertext will propagate to subsequent blocks, making it easier to detect tampering.

Random Initialization Vector (IV): A unique and random IV for each encryption operation ensures that even if the same plaintext is encrypted multiple times, the resulting ciphertext will be different.

However, it's important to note that CBC mode does not provide authentication or integrity checking. To achieve both confidentiality and integrity, additional mechanisms such as Message Authentication Codes (MACs) or digital signatures should be used in conjunction with CBC.

In summary, CBC is a widely used mode of operation for symmetric encryption algorithms like AES, offering confidentiality and error propagation. By introducing feedback and chaining between blocks, CBC enhances the security of data during encryption and decryption processes.

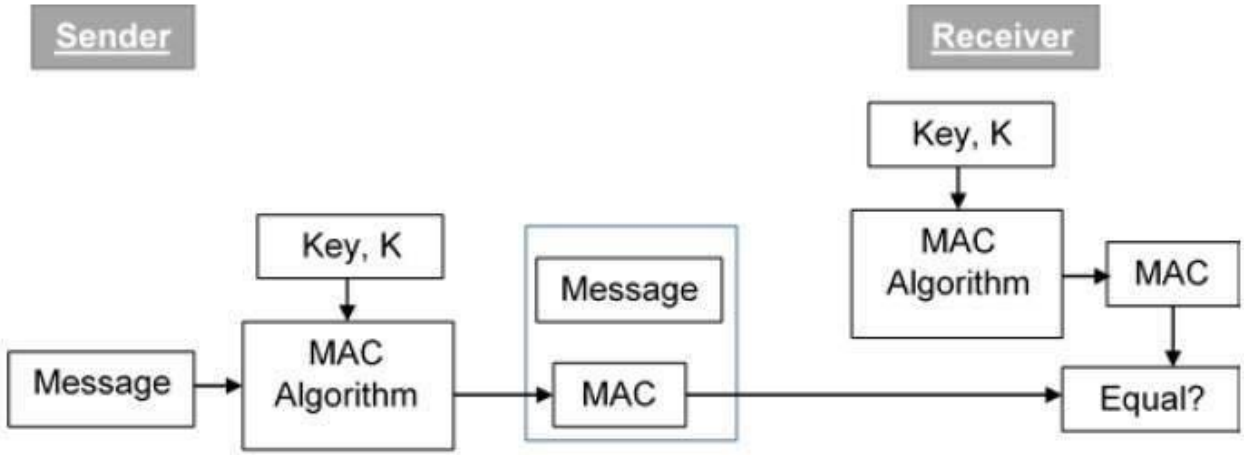
MAC

Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.

Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.

The process of using MAC for authentication is depicted in the following illustration –



Let us now try to understand the entire process in detail −

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.

- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.

- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.

- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.

- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.

- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

CBC-MAC

CBC-MAC, short for Cipher Block Chaining Message Authentication Code, is a method used to provide message integrity and authenticity in cryptography. It is a construction that combines a symmetric encryption algorithm with a message authentication code (MAC).

The CBC-MAC algorithm operates on fixed-length blocks of data, typically using a block cipher such as AES (Advanced Encryption Standard). It takes two inputs: a secret key and the data to be authenticated. The data is divided into blocks, and each block is XORed with the previous ciphertext block before encryption.

Here's a step-by-step explanation of the CBC-MAC algorithm:

- **Dividing the Data**: The data to be authenticated is divided into fixed-length blocks. Padding may be applied if the data length is not a multiple of the block size.

- **Initialization**: A random initialization vector (IV) is generated.

- **Encryption**: The first data block is XORed with the IV. The resulting value is then encrypted using the block cipher algorithm with the secret key. The ciphertext is stored.

- **Chaining:** For each subsequent block, the previous ciphertext block is XORed with the current data block. The result is then encrypted using the block cipher with the secret key, and the ciphertext is stored again.

- **Finalization:** The last ciphertext block obtained from the chaining process is considered the MAC value. It represents the integrity and authenticity of the entire message.

To verify the authenticity of a message using CBC-MAC, the receiver applies the same algorithm to the received message and compares the resulting MAC with the one transmitted alongside the message. If they match, the message is considered valid and has not been tampered with.

It's important to note that CBC-MAC is designed for fixed-length messages and does not provide protection against replay attacks or guarantee non-repudiation. For variable-length messages, additional techniques like length extension and padding are required.

HMAC

HMAC (Hash-based Message Authentication Code) is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key. Like any of the MAC, it is used for both data integrity and authentication. Checking data integrity is necessary for the parties involved in communication. HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5, SHA-1, or SHA-256. Digital signatures are nearly similar to HMACs i.e they both employ a hash function and a shared key. The difference lies in the keys i.e HMACs use symmetric key(same copy) while Signatures use asymmetric (two different keys).

Applications

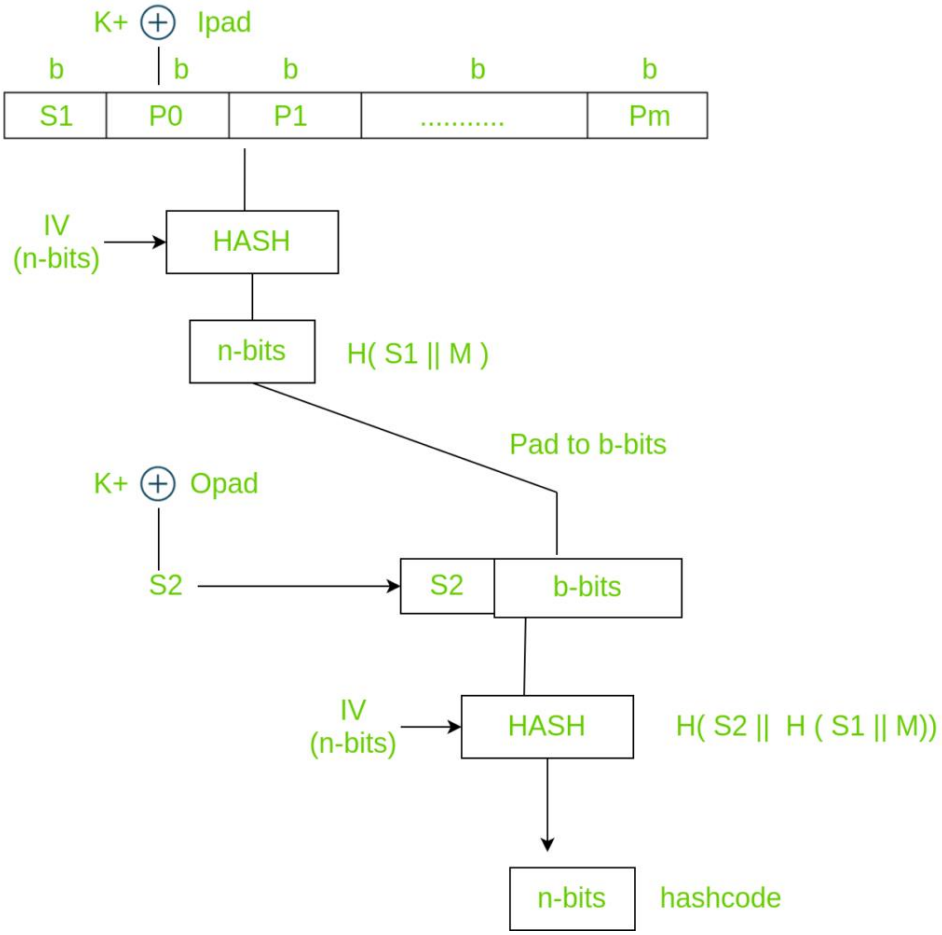
- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest. That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

Working of HMAC

HMACs provides client and server with a shared private key that is known only to them. The client makes a unique hash (HMAC) for every request. When the client requests the server, it hashes the requested data with a private key and sends it as a part of the request. Both the message and key are hashed in separate steps making it secure. When the server receives the request, it makes its own HMAC. Both the HMACS are compared and if both are equal, the client is considered legitimate.

The formula for HMAC:

HMAC = hashFunc(secret key + message)



Using key K (0 < K < b), K+ is generated by padding O's on left side of key K until length becomes b bits. The reason why it's not padded on right is change(increase) in the length of key. b bits because it is the block size of plain text. There are two predefined padding bits called ipad and opad. All this is done before applying hash function to the plain text message.

ipad - 00110110

opad – 01011100

Now we have to calculate S bits

K+ is EXORed with ipad and the result is S1 bits which is equivalent to b bits since both K+ and ipad are b bits. We have to append S1 with plain text messages. Let P be the plain text message.

S1, p0, p1 upto Pm each is b bits. m is the number of plain text blocks. P0 is plain text block and b is plain text block size. After appending S1 to Plain text we have to apply HASH algorithm (any variant). Simultaneously we have to apply initialization vector (IV) which is a buffer of size n-bits. The result produced is therefore n-bit hashcode i.e H(S1 || M).

Similarly, n-bits are padded to b-bits And K+ is EXORed with opad producing output S2 bits. S2 is appended to the b-bits and once again hash function is applied with IV to the block. This further results into n-bit hashcode which is H(S2 || H(S1 || M)).

Summary:

- Select K.
- If K < b, pad 0’s on left until k=b. K is between 0 and b (0 < K < b)
- EXOR K+ with ipad equivalent to b bits producing S1 bits.
- Append S1 with plain text M
- Apply SHA-512 on (S1 || M)
- Pad n-bits until length is equal to b-bits
- EXOR K+ with opad equivalent to b bits producing S2 bits.
- Append S2 with output of step 5.
- Apply SHA-512 on step 7 to output n-bit hashcode.

PMAC

PMAC (Parallelizable Message Authentication Code) is a cryptographic message authentication code that provides integrity and authenticity for messages. It is a fast and secure MAC algorithm suitable for a wide range of applications. Here's an explanation of how PMAC works and an example:

Working of PMAC:

1. Key Generation: PMAC uses a secret key to generate the authentication code. The key should be unique and known only to the sender and receiver.

2. Initialization: PMAC requires an initialization vector (IV), which should be unique for each message. The IV should be randomly generated or derived from a secure source.

3. Message Processing:

a. Divide the message into fixed-size blocks.

b. For each block:

- Encrypt the block using a block cipher (usually AES) with the key and an incremented counter value derived from the IV.

- XOR the resulting ciphertext with a polynomial evaluation based on the key and the corresponding counter value.

- Update the authentication code with the XORed value.

4. Finalization: Once all blocks are processed, the authentication code is computed by encrypting a final block with the key and a special end-of-message counter value, and then XORing the ciphertext with a polynomial evaluation based on the key and the counter value. This final value serves as the authentication code.

5. Authentication: The sender includes the authentication code along with the message. The receiver independently computes the authentication code for the received message using the same key and IV. If the computed code matches the received code, the message is considered authentic and unaltered.

Example:

Suppose Alice wants to send a message to Bob using PMAC with a secret key and a unique IV. Here's how the process would work:

1**. Key Generation:** Alice and Bob agree on a shared secret key.

2. Initialization: Alice generates a unique IV for the message.

3. Message Processing:

- Alice divides the message into fixed-size blocks and processes each block using the PMAC algorithm.

- For each block, Alice encrypts the block using a block cipher (e.g., AES) with the key and an incremented counter value derived from the IV.

- Alice XORs the resulting ciphertext with a polynomial evaluation based on the key and the counter value, updating the authentication code.

4. Finalization: Once all blocks are processed, Alice computes the final authentication code by encrypting a special end-of-message block with the key and the end-of-message counter value. She XORs the ciphertext with a polynomial evaluation based on the key and the counter value to obtain the authentication code.

5**. Authentication:** Alice sends the message along with the authentication code to Bob. Bob independently computes the authentication code for the received message using the same key and IV. If the computed code matches the received code, Bob can be confident that the message is authentic and unaltered.

PMAC provides strong integrity and authenticity guarantees for the message, ensuring that any modifications to the message or the authentication code will be detected. It is a lightweight and efficient MAC algorithm suitable for various applications in information security.

CW-MAC

CW-MAC (Cartwright and Waters MAC) is a construction for building Message Authentication Codes (MACs) that provides security against chosen-message attacks. It is designed to be secure in the presence of key-dependent initialization vectors, making it suitable for various cryptographic applications. Here's an explanation of how CW-MAC works and an example:

Working of CW-MAC:

1. Key Generation: A secret key is generated using a secure random number generator. The key is used to generate the initialization vector (IV) and other parameters for the CW-MAC construction.

2. Initialization: The CW-MAC construction involves initializing the internal state based on the key and IV. This initialization step is performed once per key and IV pair.

3. MAC Generation:

a. Message Processing: The input message is divided into fixed-length blocks. Each block is processed iteratively by the CW-MAC construction.

b. Internal State Update: The internal state of CW-MAC is updated using the processed blocks and the key-dependent IV. This step ensures that the MAC output is dependent on the key and the entire message.

4. Finalization: After processing all message blocks, a finalization step is performed to generate the MAC output. The finalization process incorporates the current state of CW-MAC, the key, and the IV to produce the MAC tag.

Example: CW-MAC Construction

Let's consider an example of generating a CW-MAC for a message using a specific instantiation of the construction:

1. Key Generation: Generate a secret key (K) using a secure random number generator. Generate the initialization vector (IV) and other parameters.

2. Initialization: Initialize the internal state of CW-MAC based on the key and IV.

3. MAC Generation:

- Input Message: "Hello, World!"
- Block Size: 8 characters (for simplicity)
- Process Blocks:
  - Block 1: "Hello, W"
  - Block 2: "orld!"
- Internal State Update: Update the internal state of CW-MAC using the processed blocks and the key-dependent IV.

4. Finalization: Perform a finalization step to generate the MAC output (tag) based on the current state, the key, and the IV.

The resulting MAC tag is a fixed-length value that represents the integrity and authenticity of the input message. To verify the integrity of the message, the recipient of the message computes the CW-MAC using the same key, IV, and construction parameters. The generated tag is

compared to the received tag, and if they match, the message is considered authentic and unaltered.

CW-MAC provides security against chosen-message attacks, meaning that even if an attacker has access to multiple MACs for different messages, they cannot forge a valid MAC for a new message without knowledge of the key. The key-dependent IV ensures that the MAC output is dependent on both the key and the specific message being authenticated, providing additional security guarantees.

16 –Collision Resistant Hashing

Collision-resistant hashing is a property of cryptographic hash functions. A hash function takes an input (message) of arbitrary length and produces a fixed-size output (hash value or digest). Collision resistance means that it is computationally infeasible to find two different inputs that produce the same hash value.

The working of collision-resistant hashing can be summarized as follows:

1. Input: The hash function takes an input message, which can be of any length.

2. Compression: The message is divided into blocks of a fixed size (depending on the hash function), and the hash function applies a compression function to each block. The compression function takes the

current hash value and the current block as inputs and produces an intermediate hash value.

3. Chaining: The intermediate hash values are chained together, so the output of each compression step becomes the input for the next compression step. This ensures that the final hash value depends on the entire input message.

4. Output: After processing all the blocks, the final hash value, also known as the hash digest, is produced. The digest is typically a fixed-length string of bits or bytes.

Collision resistance is achieved by designing hash functions with certain properties, such as:

a. Pre-image resistance: Given a hash value, it should be computationally infeasible to find an input message that produces that hash value.

b. Second pre-image resistance: Given an input message, it should be computationally infeasible to find a second message that produces the same hash value.

c. Collision resistance: It should be computationally infeasible to find any two different input messages that produce the same hash value.

To ensure collision resistance, cryptographic hash functions employ various techniques, including complex mathematical operations, mixing functions, and non-linearity. Well-known hash functions that are widely used and considered collision-resistant include SHA-256 (part of the SHA-2 family) and SHA-3 hash functions.

Collision-resistant hashing is fundamental in many security applications, such as password hashing, digital signatures, message authentication codes (MACs), and various forms of data integrity verification. It provides a crucial mechanism for ensuring the integrity and authenticity of data while protecting against malicious tampering or forgery.

Let's walk through a **simple example of collision-resistant hashing** using the SHA-256 (Secure Hash Algorithm 256-bit) as an illustration. Keep in mind that this is a simplified example for demonstration purposes.

Let's say we have two messages:

Message 1: "Hello, world!"

Message 2: "Goodbye, world!"

We'll apply SHA-256 hashing to both messages and compare the resulting hash values. SHA-256 will produce a fixed-length 256-bit hash value regardless of the input size.

1. Applying SHA-256 to Message 1 ("Hello, world!"):

   - SHA-256 Hash: 0x2ef7bde608ce5404e97d5f042f95f89f1c232871

2. Applying SHA-256 to Message 2 ("Goodbye, world!"):

   - SHA-256 Hash: 0x45f7ab2fb43e63e87a7d20027a2eeae0b4ef5f16

As you can see, the hash values for the two messages are distinct. This demonstrates the uniqueness property of cryptographic hash functions.

Now, let's create a scenario to showcase the collision resistance property. We'll modify Message 2 slightly to see if we can generate the same hash value as Message 1.

Modified Message 2: "Goodbye, world!!"

1. Applying SHA-256 to Message 1 ("Hello, world!"):

   - SHA-256 Hash: 0x2ef7bde608ce5404e97d5f042f95f89f1c232871

2. Applying SHA-256 to Modified Message 2 ("Goodbye, world!!"):

   - SHA-256 Hash: 0x2ef7bde608ce5404e97d5f042f95f89f1c232871

In this case, you can observe that the hash values of both Message 1 and the modified Message 2 are the same. However, note that generating such a collision is extremely difficult in practice due to the cryptographic properties of the SHA-256 algorithm.

Collision resistance means that it should be computationally infeasible to find two different inputs that produce the same hash value (collision) using a cryptographic hash function. While it's theoretically possible to find collisions, the time and computational resources required to do so are so vast that it's considered practically impossible.

Cryptographic hash functions like SHA-256 are designed to provide a high level of security and are widely used in various applications, including digital signatures, password storage, data integrity checks, and more**.**

17 –Authenticated Encryption

Authenticated encryption is a cryptographic process that combines both confidentiality and integrity protections for data. It ensures that the data remains confidential and cannot be read or modified by unauthorized parties, while also verifying the integrity of the data to ensure that it has not been tampered with.

In traditional encryption schemes, confidentiality is achieved through encryption algorithms such as AES (Advanced Encryption Standard),

while integrity is ensured through separate mechanisms like digital signatures or message authentication codes (MACs). Authenticated encryption, on the other hand, provides both confidentiality and integrity in a single operation, making it more efficient and secure.

There are different modes of operation for authenticated encryption, such as Galois/Counter Mode (GCM), Counter with Cipher Block Chaining-MAC (CCM), and EAX (Encrypt-then-Authenticate-then-eXtend). These modes combine encryption algorithms with integrity protection mechanisms, typically using symmetric encryption algorithms like AES.

When using authenticated encryption, the sender encrypts the data using a secret key and includes an authentication tag, which is a short piece of data that acts as a proof of integrity. The receiver then decrypts the data using the same secret key and verifies the authenticity of the data by checking the authentication tag. If the tag matches the decrypted data, it confirms that the data has not been tampered with.

Authenticated encryption provides several benefits, including:

1. Confidentiality: The data remains confidential and cannot be read by unauthorized parties.

2. Integrity: The data's integrity is verified, ensuring that it has not been modified in transit.

3. Efficiency: Authenticated encryption combines both confidentiality and integrity protections in a single operation, reducing computational overhead.

4. Simplicity: Using authenticated encryption simplifies the cryptographic operations required to protect data.

5. Security: Authenticated encryption schemes have undergone rigorous analysis and are designed to resist various cryptographic attacks.

It is important to choose a well-vetted and secure authenticated encryption scheme and use it correctly to ensure the confidentiality and integrity of sensitive data.

Here's an example of how authenticated encryption works:

1. Encryption:

- Alice wants to send a message to Bob securely. She first encrypts the message using a symmetric encryption algorithm like AES (Advanced Encryption Standard).

- Alice generates a unique encryption key (KE) and encrypts the message using this key. The encrypted message is denoted as Ciphertext.

2. Authentication:

- Alice also generates a Message Authentication Code (MAC) using a symmetric authentication algorithm like HMAC (Hash-based Message Authentication Code).

- Alice uses a unique authentication key (KA) and applies the HMAC algorithm to the plaintext message, resulting in a MAC value.

3. **Combining the ciphertext and MAC:**

- Alice combines the Ciphertext and MAC, resulting in a single string.

- She sends this combined string to Bob.

4. Decryption and verification:

- Bob receives the combined string from Alice.

- He separates the Ciphertext and MAC.

5**. Authentication:**

- Bob uses the received authentication key (KA) and the decrypted message to compute a MAC value.

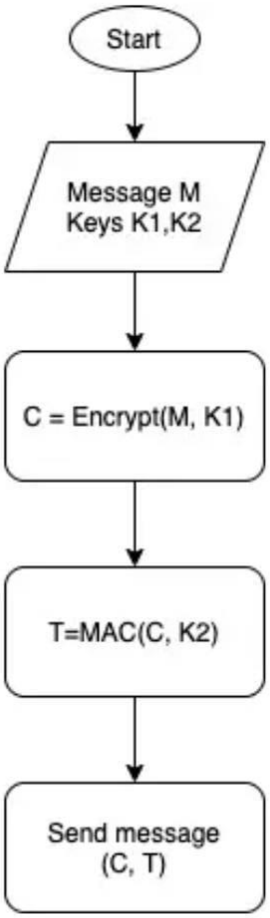
- He compares this MAC value with the received MAC value to verify if the message has been tampered with. If they match, the message is considered authentic.

6. Decryption: - If the MAC verification is successful, Bob uses the received encryption key (KE) to decrypt the Ciphertext and obtain the original plaintext message.

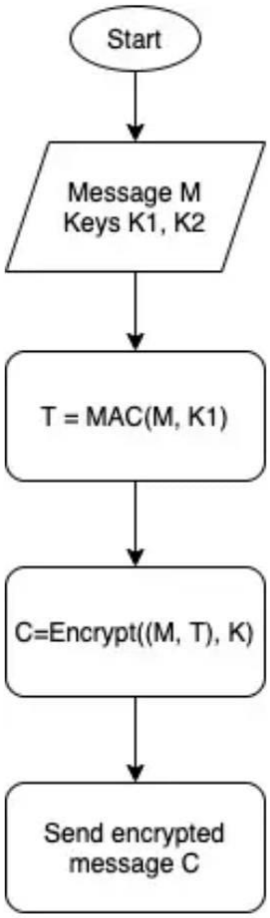
By combining encryption and authentication, authenticated encryption provides a secure and reliable means of protecting data confidentiality and integrity. It ensures that the encrypted message cannot be decrypted by unauthorized entities and that the message remains unaltered during transit or storage

Approach	**Description**	**Example of applications**
Encrypt, then authenticate	– Generate two keys – Encrypt the message with the first key – Authenticate the encrypted message with the second key.	IPSec
Authenticate, then encrypt	– Generate two keys – Calculate the message authentication code using the first key – Encrypt the message plus the message authentication code using the second key	SSL and TLS
Hash, then encrypt	– Calculate the cryptographic hash of the message. – Encrypt the	Wired Equivalent Privacy (WEP)

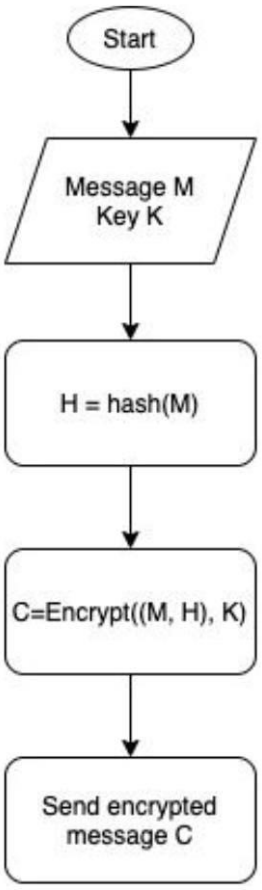
	message plus the hash function	
Encrypt and authenticate	– Generate two keys – Encrypt the message with the first key – Authenticate the message (plain text) with the second key	SSH



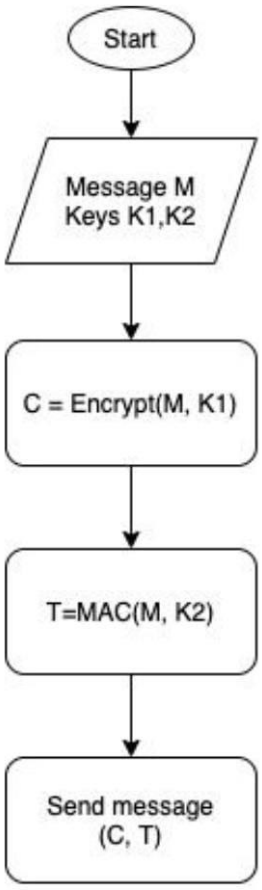
Encrypt, then authenticate



Authenticate, then encrypt



Hash, then encrypt



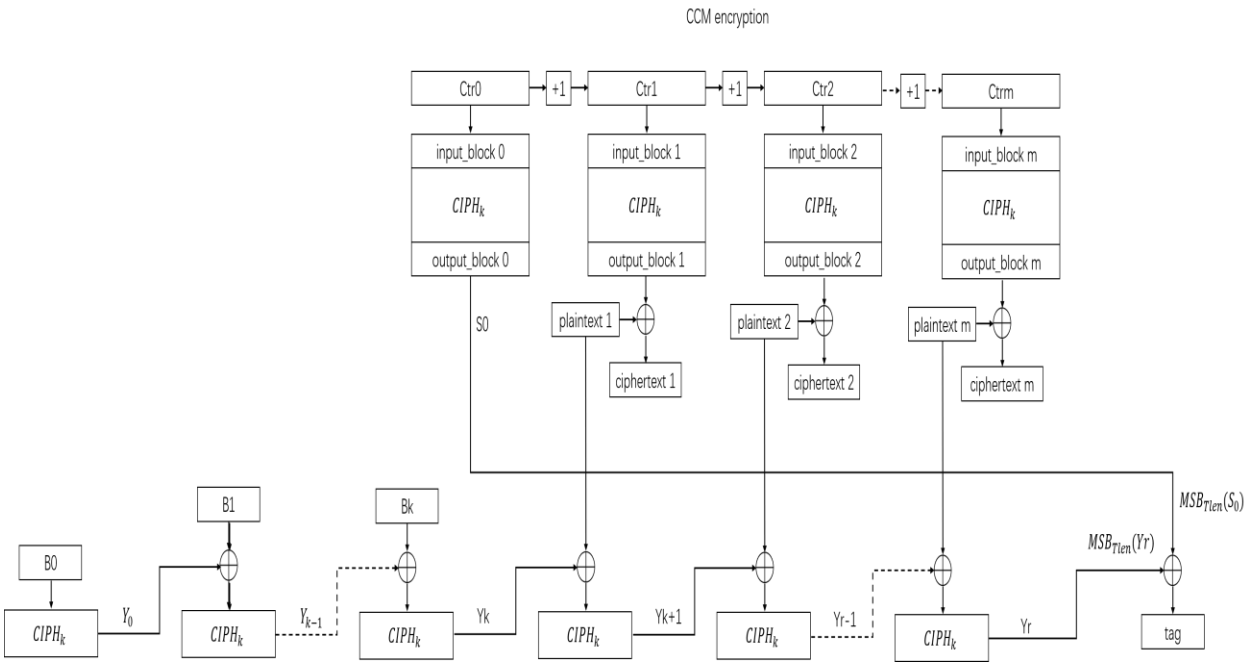
Encrypt and authenticate

18 –CCM, GCM, TLS and IPsec

CCM

CCM stands for Counter with CBC-MAC (CCM) mode, which is a mode of operation for block ciphers that provides authenticated encryption. It is commonly used in information security to ensure the confidentiality, integrity, and authenticity of data.

CCM mode combines the Counter (CTR) mode for encryption and the Cipher-Block Chaining Message Authentication Code (CBC-MAC) for authentication. It is suitable for resource-constrained devices like IoT devices as it requires minimal computational resources.



Here's how CCM mode works:

1. Dividing the data:

- The input data is divided into two parts: plaintext and associated data (AAD). The plaintext contains the actual message, while the AAD can include additional information that is not encrypted but is authenticated.

2. Encryption:

- The plaintext is encrypted using the Counter (CTR) mode of operation, which generates a stream of ciphertext blocks.

- The Counter mode uses a unique counter value for each block, combined with a nonce and a key.

3. Authentication:

- The CBC-MAC algorithm is applied to the plaintext and the AAD separately.

- The MAC values generated for both the plaintext and the AAD are concatenated.

4. Combining the ciphertext and MAC:

- The ciphertext and the MAC value are concatenated to form the authenticated encrypted output.

5**. Decryption and verification:**

- On the receiving end, the ciphertext is decrypted using the same Counter mode.

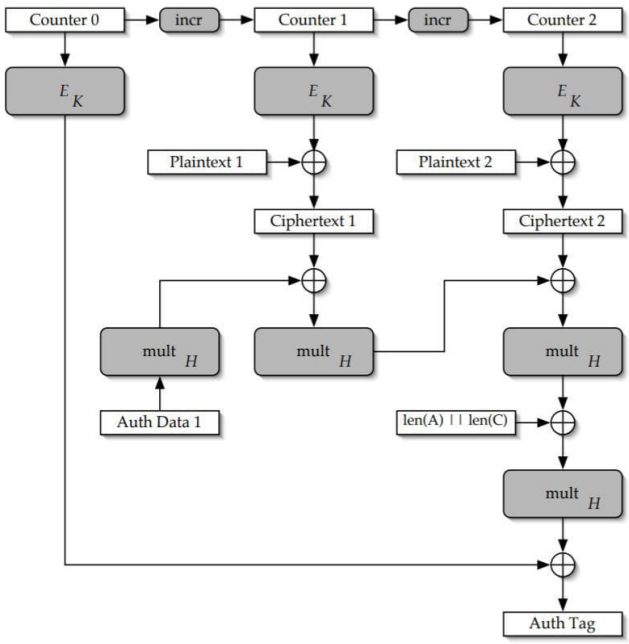
- The CBC-MAC algorithm is applied to the decrypted plaintext and the received MAC separately.

- The calculated MAC values are compared with the received MAC value. If they match, the data is considered authentic.

CCM mode provides confidentiality through encryption and integrity through the CBC-MAC algorithm. It also offers authenticity as the MAC value ensures that the data has not been tampered with. This mode of operation is widely used in applications where data integrity and confidentiality are critical, such as wireless communication protocols like Wi-Fi Protected Access (WPA2) and Bluetooth Low Energy (BLE).

GCM

GCM (Galois/Counter Mode) is a widely used encryption mode in information security. It combines the counter mode of encryption (CTR) with the Galois mode of authentication (GMAC). GCM provides both confidentiality (encryption) and integrity (authentication) of data, making it suitable for secure communication and storage.



Here's a brief overview of how GCM works:

1. Key Generation: A secret key is generated using a secure key generation algorithm, typically with a suitable key size (128 bits, 192 bits, or 256 bits) depending on the desired security level.

2. Initialization: A unique initialization vector (IV) is generated for each encryption operation. The IV is typically a nonce (number used once) and should be unpredictable to ensure the security of the encryption.

3. Encryption: GCM uses the counter mode (CTR) for encryption, where a counter is encrypted and then XORed with the plaintext to produce the ciphertext. The counter value is derived from the IV and an initial counter value. This process is performed for each block of data.

4. Authentication: GCM utilizes the Galois mode of authentication (GMAC) to provide data integrity. GMAC generates a tag (also known as a MAC, message authentication code) that represents the authentication of the ciphertext. It uses a polynomial multiplication algorithm in a Galois field to compute the authentication tag.

5. Finalization: The encryption process is completed by appending the authentication tag to the ciphertext.

To decrypt and verify the integrity of the data encrypted with GCM, the recipient uses the same secret key, IV, and the ciphertext. The recipient performs the decryption process using the counter mode (CTR) and then computes the authentication tag using GMAC. If the computed authentication tag matches the one received with the ciphertext, the data is considered authentic and has not been tampered with.

GCM is widely adopted because it provides strong security, efficient encryption/decryption operations, and the ability to process data in parallel. It is used in various protocols and applications, including network security protocols like TLS (Transport Layer Security), IPsec (Internet Protocol Security), and Wi-Fi Protected Access (WPA2 and WPA3).

TLS

Transport Layer Security, or TLS, is a widely adopted security protocol designed to facilitate privacy and data security for communications over the Internet. A primary use case of TLS is encrypting the communication between web applications and servers, such as web browsers loading a website. TLS can also be used to encrypt other communications such as email, messaging, and voice over IP (VoIP). In this article we will focus on the role of TLS in web application security.

TLS was proposed by the Internet Engineering Task Force (IETF), an international standards organization, and the first version of the protocol was published in 1999. The most recent version is TLS 1.3, which was published in 2018

What is the difference between TLS and HTTPS?

HTTPS is an implementation of TLS encryption on top of the HTTP protocol, which is used by all websites as well as some other web services. Any website that uses HTTPS is therefore employing TLS encryption.

Why should businesses and web applications use the TLS protocol?

TLS encryption can help protect web applications from data breaches and other attacks. Today, TLS-protected HTTPS is a standard practice for websites. The Google Chrome browser gradually cracked down on non-HTTPS sites, and other browsers have followed suit. Everyday Internet users are more wary of websites that do not feature the HTTPS padlock icon.

What does TLS do?

There are three main components to what the TLS protocol accomplishes: Encryption, Authentication, and Integrity.

- **Encryption:** hides the data being transferred from third parties.
- **Authentication**: ensures that the parties exchanging information are who they claim to be.
- **Integrity**: verifies that the data has not been forged or tampered with

How does TLS work?

A TLS connection is initiated using a sequence known as the TLS handshake. When a user navigates to a website that uses TLS, the TLS handshake begins between the user's device (also known as the client device) and the web server.

During the TLS handshake, the user's device and the web server:

Specify which version of TLS (TLS 1.0, 1.2, 1.3, etc.) they will use

Decide on which cipher suites (see below) they will use

Authenticate the identity of the server using the server's TLS certificate

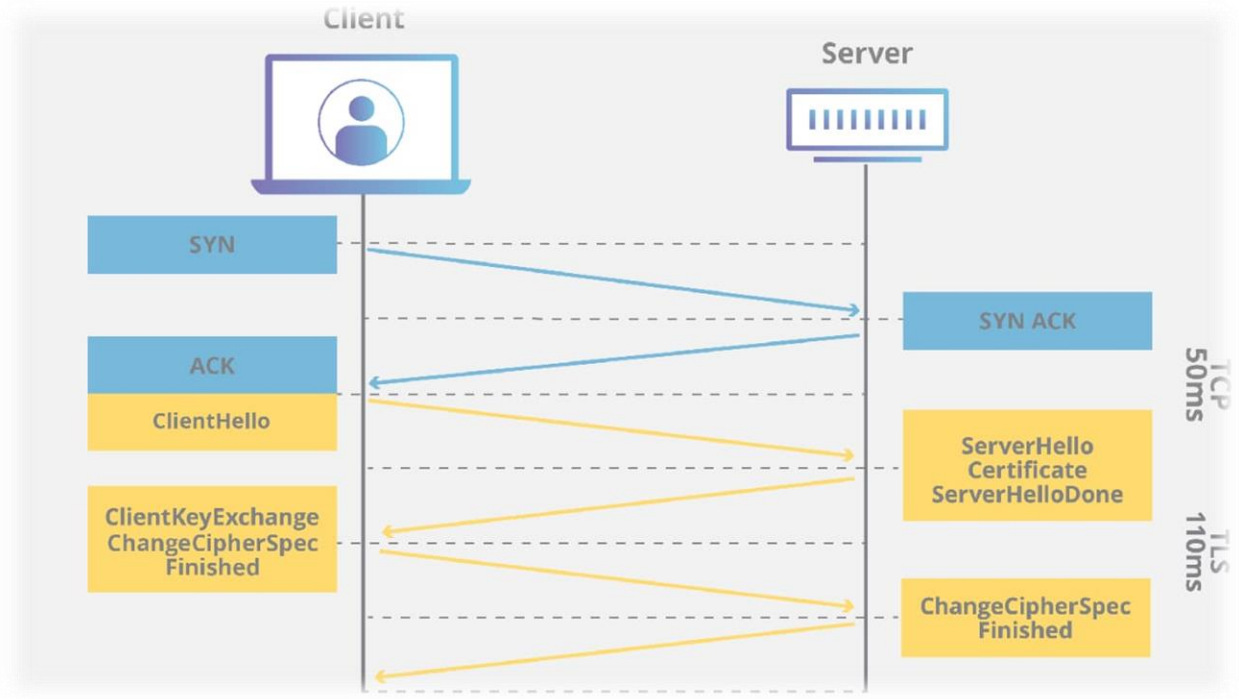
Generate session keys for encrypting messages between them after the handshake is complete

The TLS handshake establishes a cipher suite for each communication session. The cipher suite is a set of algorithms that specifies details such as which shared encryption keys, or session keys, will be used for that particular session. TLS is able to set the matching session keys over an unencrypted channel thanks to a technology known as public key cryptography.

The handshake also handles authentication, which usually consists of the server proving its identity to the client. This is done using public keys. Public keys are encryption keys that use one-way encryption, meaning that anyone with the public key can unscramble the data encrypted with the server's private key to ensure its authenticity, but only the original

sender can encrypt data with the private key. The server's public key is part of its TLS certificate.

Once data is encrypted and authenticated, it is then signed with a message authentication code (MAC). The recipient can then verify the MAC to ensure the integrity of the data. This is kind of like the tamper-proof foil found on a bottle of aspirin; the consumer knows no one has tampered with their medicine because the foil is intact when they purchase it.



IPsec

IP Sec (Internet Protocol Security) is an Internet Engineering Task Force (IETF) standard suite of protocols between two communication points across the IP network that provide data authentication, integrity, and

confidentiality. It also defines the encrypted, decrypted, and authenticated packets. The protocols needed for secure key exchange and key management are defined in it.

Uses of IP Security

IPsec can be used to do the following things:

- To encrypt application layer data.
- To provide security for routers sending routing data across the public internet.
- To provide authentication without encryption, like to authenticate that the data originates from a known sender.
- To protect network data by setting up circuits using IPsec tunneling in which all data being sent between the two endpoints is encrypted, as with a Virtual Private Network(VPN) connection

Components of IP Security

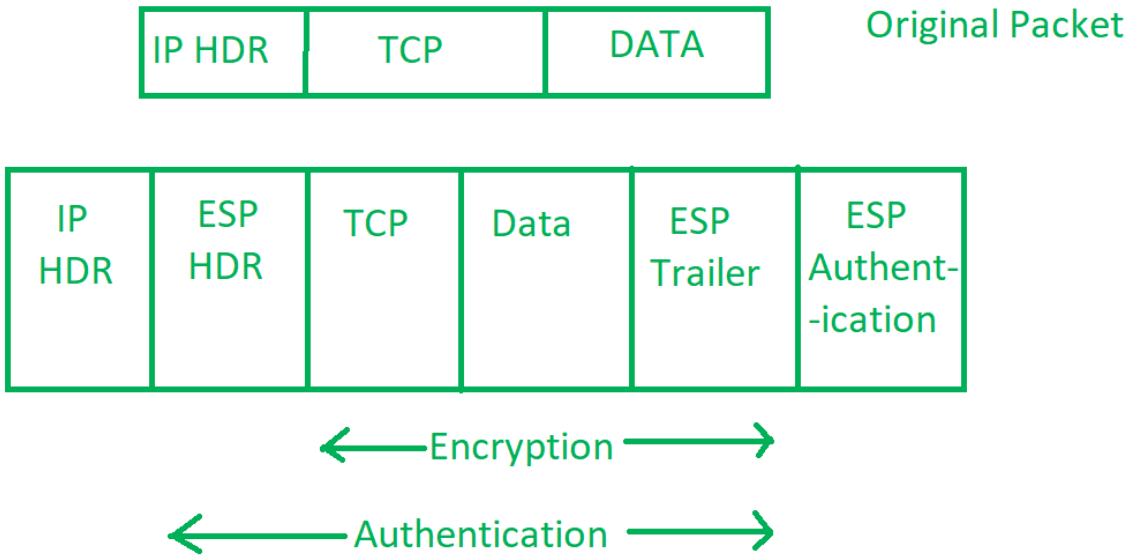
It has the following components:

- Encapsulating Security Payload (ESP)
- Authentication Header (AH)
- Internet Key Exchange (IKE)

1. Encapsulating Security Payload (ESP): It provides data integrity, encryption, authentication, and anti-replay. It also provides authentication for payload.

2. Authentication Header (AH): It also provides data integrity, authentication, and anti-replay and it does not provide encryption. The anti-replay protection protects against the unauthorized transmission of packets. It does not protect data confidentiality.

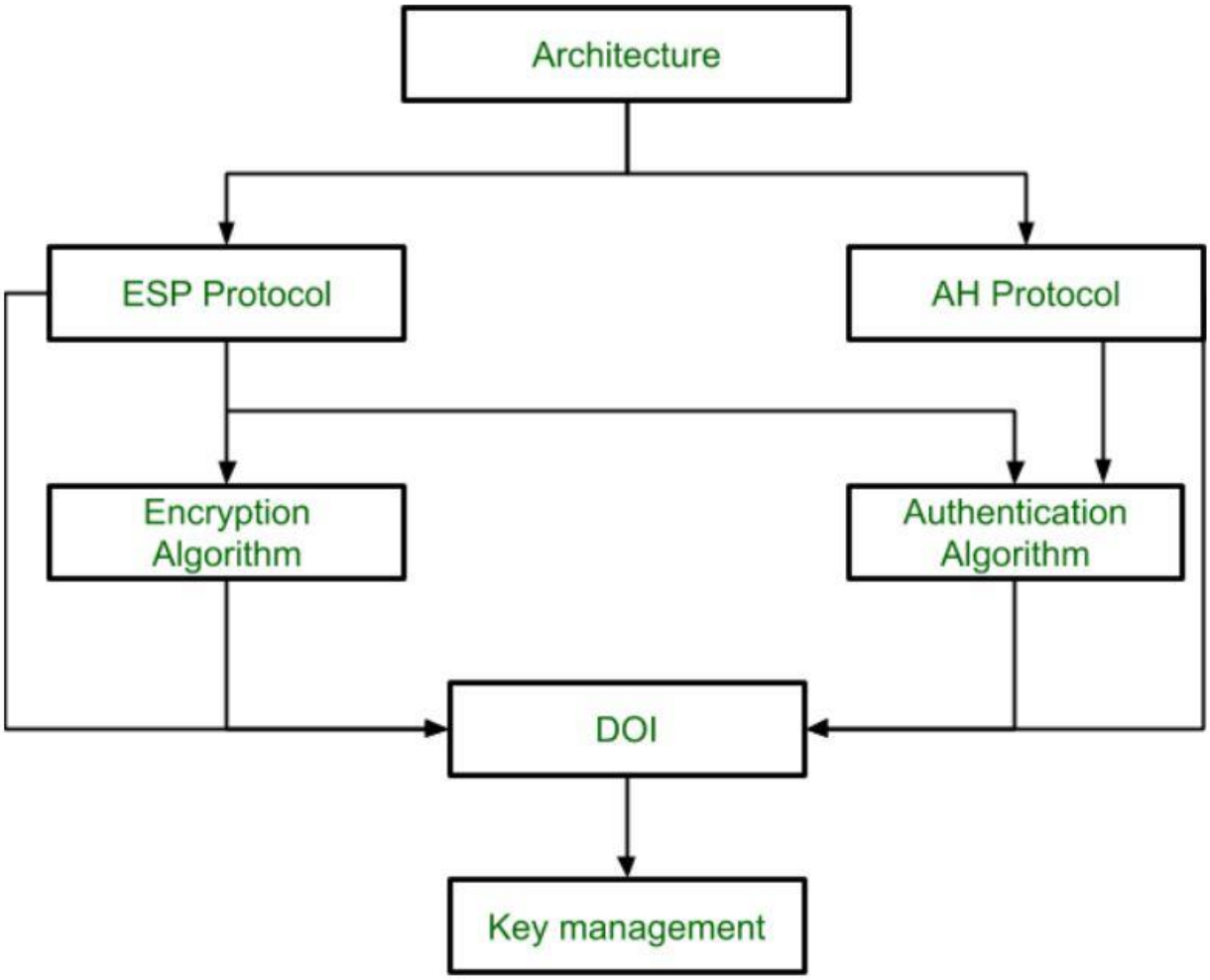
3. Internet Key Exchange (IKE): It is a network security protocol designed to dynamically exchange encryption keys and find a way over Security Association (SA) between 2 devices. The Security Association (SA) establishes shared security attributes between 2 network entities to support secure communication. The Key Management Protocol (ISAKMP) and Internet Security Association provides a framework for authentication and key exchange. ISAKMP tells how the setup of the Security Associations (SAs) and how direct connections between two hosts are using IPsec. Internet Key Exchange (IKE) provides message content protection and also an open frame for implementing standard algorithms such as SHA and MD5. The algorithm's IP sec users produce a unique identifier for each packet. This identifier then allows a device to determine whether a packet has been correct or not. Packets that are not authorized are discarded and not given to the receiver.



IP Security Architecture

IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture includes protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- **Confidentiality**
- **Authenticity**
- **Integrity**



Working on IP Security

- The host checks if the packet should be transmitted using IPsec or not. This packet traffic triggers the security policy for itself. This is done when the system sending the packet applies appropriate encryption. The incoming packets are also checked by the host that they are encrypted properly or not.
- Then IKE Phase 1 starts in which the 2 hosts(using IPsec) authenticate themselves to each other to start a secure channel. It has 2 modes. The Main mode provides greater security and the Aggressive mode which enables the host to establish an IPsec circuit more quickly.

- The channel created in the last step is then used to securely negotiate the way the IP circuit will encrypt data across the IP circuit.
- Now, the IKE Phase 2 is conducted over the secure channel in which the two hosts negotiate the type of cryptographic algorithms to use on the session and agree on secret keying material to be used with those algorithms.
- Then the data is exchanged across the newly created IPsec encrypted tunnel. These packets are encrypted and decrypted by the hosts using IPsec SAs.
- When the communication between the hosts is completed or the session times out then the IPsec tunnel is terminated by discarding the keys by both hosts

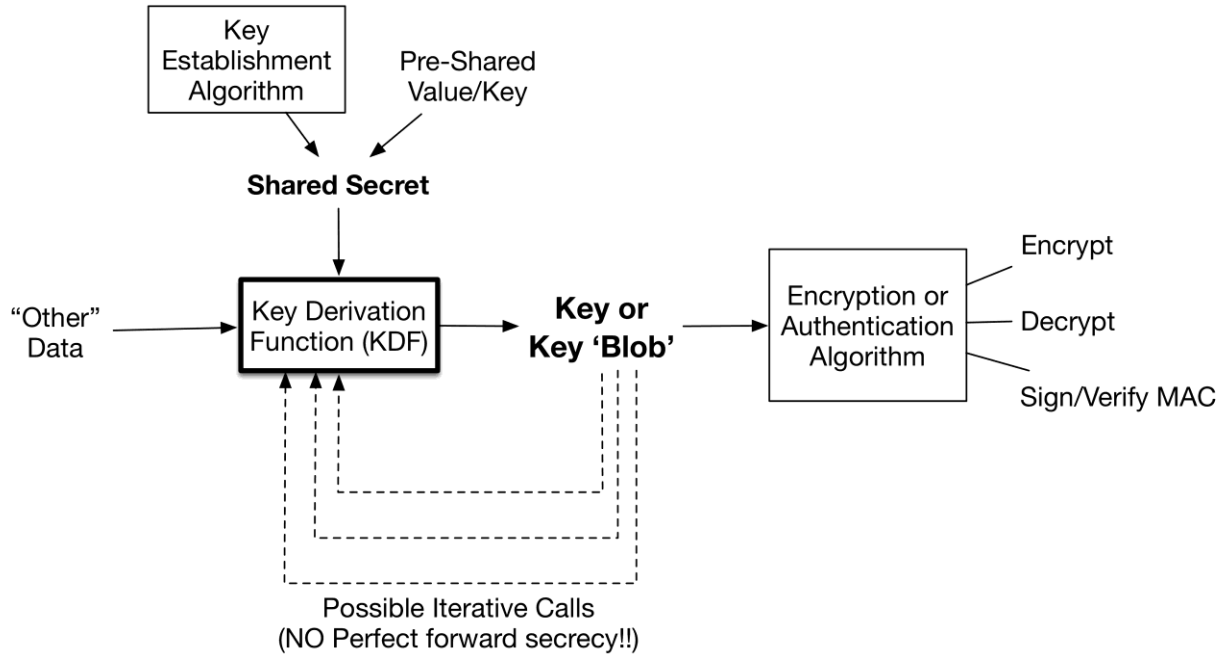
Features of IPSec

- Authentication: IPSec provides authentication of IP packets using digital signatures or shared secrets. This helps ensure that the packets are not tampered with or forged.
- Confidentiality: IPSec provides confidentiality by encrypting IP packets, preventing eavesdropping on the network traffic.
- Integrity: IPSec provides integrity by ensuring that IP packets have not been modified or corrupted during transmission.
- Key management: IPSec provides key management services, including key exchange and key revocation, to ensure that cryptographic keys are securely managed.
- Tunneling: IPSec supports tunneling, allowing IP packets to be encapsulated within another protocol, such as GRE (Generic Routing Encapsulation) or L2TP (Layer 2 Tunneling Protocol).

- Flexibility: IPSec can be configured to provide security for a wide range of network topologies, including point-to-point, site-to-site, and remote access connections.
- Interoperability: IPSec is an open standard protocol, which means that it is supported by a wide range of vendors and can be used in heterogeneous environments.

19 -Key Derivation Functions

A key derivation function (KDF) is a cryptographic algorithm used to derive one or more secret keys from a master key or other input material. KDFs are commonly employed in information security to generate keys for various purposes, such as encryption, authentication, and key agreement. They ensure that the derived keys have desired properties, such as randomness, uniqueness, and security.



The properties of keys

If our important data is secured by a secret key, we need to make sure that hackers cannot figure out the key. If they can find the key, they can decrypt the data just as easily as we can.

In addition to us needing to keep the key a secret, the key must also meet certain properties in order for us to be confident that an attacker will not be able to figure it out:

Key length

The length of the key is an important factor in the overall security of a given cryptosystem. Key length essentially refers to the size of the key,

with larger keys being generally more secure (there are exceptions to this). However, there is always a trade-off between security and speed.

The underlying concept is relatively simple: If someone is thinking of a number between one and ten, you will be able to guess it in just a few tries. If they are thinking of a number between one and a million you will probably get bored before you figure it out.

Keys are basically just big numbers, so the larger the field of numbers to choose from, the harder it will be for an attacker to figure out your key. Keys need to be certain lengths or greater in order to be secure for a given purpose. This depends on the situation, but as an example, AES has three key lengths:

128 bits

192 bits

256 bits

Other algorithms, like RSA, use much larger keys. It's not uncommon for RSA to use 4,096 bit keys. However, these large RSA keys do not necessarily grant more security to a cryptosystem than the smaller AES keys. This is due to some complexities of the math that underlie these two algorithms, but it's out of the scope of this article.

Randomness

A key needs to be random, or close enough to random that even sophisticated techniques can't detect any patterns. If there are any patterns or restrictions, it can make it much easier for a hacker to figure out the key.

Let’s explain this further by returning to our example from above: If your friend chooses a truly random number between one and a million, then there is an equal chance that it could be any single number within that range. To figure it out, you would have to guess every single number in a systematic fashion until you got lucky and came across the right one. On average, it would take you 500,000 guesses to get it right. This would take you an exceptionally long time.

Let’s switch up the example and say that you’ve seen your friend play this game with others before. You noticed that the numbers they choose aren’t actually random. Whenever they play, the number always ends in 837. You aren’t sure why, but for some reason your friend has an obsession with this number. You have noticed a pattern, and you can use it to make your job a whole lot easier.

Instead of having to work your way through every single number, you could speed things up dramatically by only guessing numbers that end in 837:

837, 1837, 2837, 3837, 4837, etc..

Because you figured out the pattern, it should only take you an average of 500 guesses to figure out the number that your friend is thinking of.

The security of keys works exactly the same way. If all of the keys from a given system follow a predictable pattern, it is much easier to figure out what the key actually is.

Uniform distribution

It’s not just patterns that we have to worry about. We also have to worry about whether keys are from a uniform distribution. For example, let’s say you have a key generator that is supposed to give you random 128-bit keys. We will use decimal numbers to explain this, to avoid complicating things any further than necessary.

In decimal, the largest 128-bit number is:

340,282,366,920,938,463,463,374,607,431,768,211,456

If the key generator was truly random and had a uniform distribution, you would expect it to be equally likely for the key to be any single number between 0 and 340,282,366,920,938,463,463,374,607,431,768,211,456.

But let’s say there is a bug in the key generator. You run it a few times, and these are the keys that it outputs:

8722

9345

0497

1438

7506

4379

3984

What’s happened? These numbers are seemingly random, but they certainly aren’t uniform and evenly spread out between 0 and 340,282,366,920,938,463,463,374,607,431,768,211,456. They are all clustering under 10,000 because of the bug.

If an attacker figured out that the key generator used in the cryptosystem was faulty and only output keys in such a limited range, it would make their job much easier. Due to the limited distribution of the keys, the 128-bit key would be trivial to guess.

It would take an average of only 5,000 guesses, instead of the trillions and trillions that a truly random and uniform 128-bit key would require. This is why uniformity is another important aspect of key generation

Working of a Key Derivation Function:

1. Input: A KDF takes one or more inputs, including a master key, a salt (optional but recommended for added security), and additional parameters specific to the KDF algorithm.

2. Key Expansion: The KDF applies a series of steps to expand or derive multiple secret keys from the given inputs. These steps typically involve repeated cryptographic operations and transformations.

3. Key Extraction: The final step of the KDF extracts the derived keys from the expanded key material. The extracted keys can be of different lengths and used for specific cryptographic purposes.

Example: HKDF (HMAC-based Key Derivation Function)

HKDF is a widely used key derivation function based on HMAC (Hash-based Message Authentication Code). It provides a secure and efficient way to derive cryptographic keys. Here's an example of how HKDF works:

1. Input:

- Master Key (MK): A high-entropy secret key used as the input to HKDF.

- Salt: An optional random value (recommended for added security) to strengthen the derived keys.

- Info: Additional context-specific information that can be used to derive different keys for different purposes.

2. Key Extraction:

a. Extract: HKDF uses an HMAC function with a secure hash algorithm (e.g., SHA-256) to derive a pseudorandom key (PRK) from the master key and the salt. The HMAC is computed as follows:

PRK = HMAC-Hash(Salt, MK)

b. Expand: HKDF then expands the PRK into multiple derived keys of required lengths. This expansion is achieved using the HMAC function and the Info parameter. The derived keys are generated using the following formula:

Derived Key = HMAC-Hash(PRK, Previous Key || Info || Counter)

The counter is incremented for each derived key, and the previous key is the previously generated derived key (or PRK for the first iteration). The Info parameter ensures that different keys can be derived for different purposes even if the master key and salt remain the same.

3. Output:

HKDF provides the derived keys generated in the expansion step as the output. These derived keys can be used for encryption, authentication, or other cryptographic operations.

HKDF ensures that the derived keys have desired properties, such as randomness, security, and independence. By incorporating a salt and using an HMAC-based approach, HKDF offers robust key derivation suitable for a variety of security applications.

20 –odds and ends:Deterministic encryption

Deterministic encryption is a form of encryption that allows for deterministic encryption and decryption operations, meaning that the same plaintext will always encrypt to the same ciphertext. It is useful in scenarios where searchability and equality comparisons on encrypted data are required. Here's an explanation of how deterministic encryption works and an example:

Working of Deterministic Encryption:

1. Key Generation: A deterministic encryption scheme involves generating a secret encryption key that will be used for both encryption and decryption operations. The key generation process typically follows the same principles as symmetric encryption schemes.

2. Encryption: To encrypt a plaintext message using deterministic encryption, the encryption algorithm takes the encryption key and the plaintext as inputs and produces a fixed-length ciphertext. The encryption process ensures that the same plaintext always results in the same ciphertext.

3. Decryption: The decryption process in deterministic encryption is the reverse of the encryption process. It takes the encryption key and the ciphertext as inputs and produces the original plaintext message.

Example: Format Preserving Encryption (FPE)

Format Preserving Encryption (FPE) is an example of deterministic encryption that preserves the format and length of the original plaintext. It is often used to encrypt sensitive data, such as credit card numbers or Social Security numbers, while maintaining their original structure for compatibility with existing systems.

Let's consider an example of encrypting a 16-digit credit card number using FPE:

1. Key Generation: A secret key is generated using a secure random number generator.

2. Encryption: The encryption algorithm takes the encryption key and the plaintext credit card number as inputs. It performs encryption operations while ensuring that the format and length of the plaintext are preserved. The same plaintext will always result in the same ciphertext.

3. Decryption: The decryption algorithm takes the encryption key and the ciphertext as inputs. It reverses the encryption process to produce the original plaintext credit card number.

For example, if we have the credit card number "1234 5678 9012 3456," the deterministic encryption algorithm will encrypt it to a

ciphertext such as "XWYZ PQRS TUVW ABCD." Decrypting the ciphertext using the encryption key will yield the original credit card number.

Deterministic encryption allows for search operations and comparisons on the encrypted data without the need for decryption. For example, in a database, one can search for records with a particular encrypted credit card number or perform equality checks without exposing the actual values.

It's important to note that deterministic encryption does not provide semantic security, as the same plaintext always results in the same ciphertext. Thus, it may not be suitable for scenarios where randomization and probabilistic encryption properties are required

21 – Non-expanding encryption and Format Preserving encryption

Non-expanding encryption and format-preserving encryption (FPE) are two techniques used in information security to protect data while preserving its format or structure. Let's explore each of these concepts and provide examples for better understanding:

1. Non-expanding Encryption:

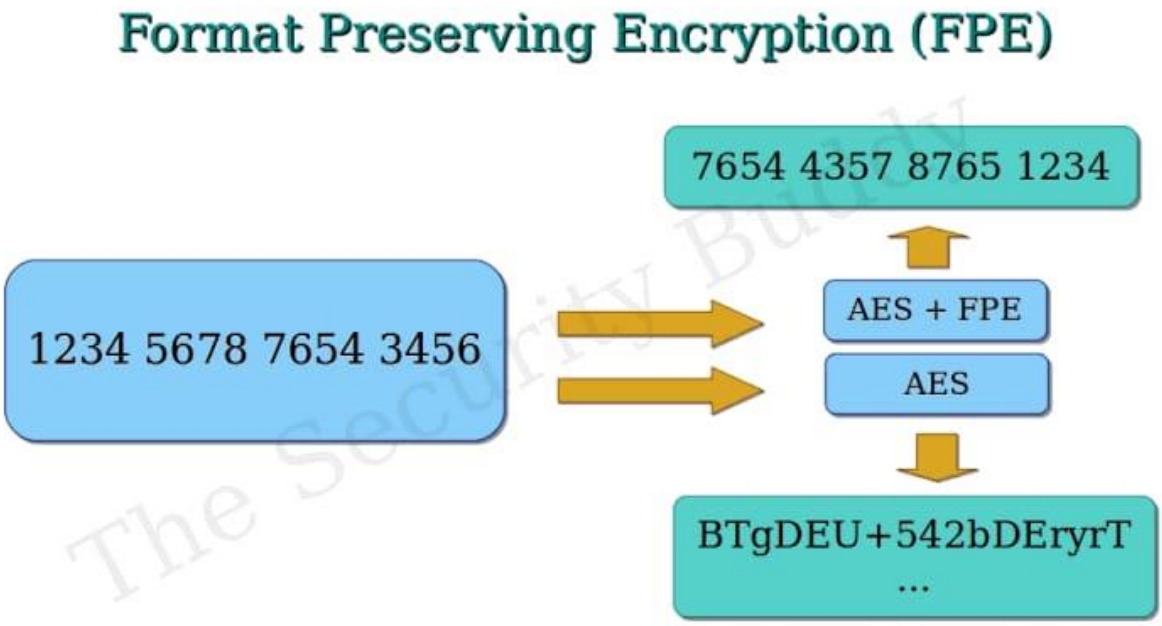
Non-expanding encryption refers to a type of encryption where the length or size of the encrypted data remains the same as the original data. In other words, the encrypted output does not grow or expand in size. This property can be desirable in situations where the size of the data needs to be preserved, such as when encrypting data fields in databases or storing encrypted data in fixed-length data structures.

For example, consider a database storing credit card numbers. If non-expanding encryption is used, the encrypted credit card numbers will have the same length as the original credit card numbers. This allows the encrypted data to be stored in the same field size without requiring any modifications to the database schema or data structures.

2. Format Preserving Encryption (FPE):

Format-preserving encryption (FPE) is a type of encryption that not only protects the confidentiality of data but also preserves its original format or structure. FPE allows encrypted data to maintain the same format as the original data, such as preserving the length, character set, and other structural properties.

For instance, let's say you have a 16-digit credit card number and you want to encrypt it using FPE. The encrypted output will also be a 16-digit number, ensuring that the format is preserved. This can be useful in scenarios where encrypted data needs to be processed by systems or applications that expect data in a specific format without requiring any changes to their logic or algorithms.



An example of FPE algorithm is the FF1 mode of the Format-Preserving Encryption Standard (FIPS-FF1). It is defined in the NIST Special Publication 800-38G and provides a way to encrypt and decrypt data while preserving its format.

In summary, non-expanding encryption ensures that the size of the encrypted data remains the same as the original data, while format-preserving encryption allows the encrypted data to retain the same format or structure as the original data. These techniques are commonly used in various information security applications to protect sensitive data while maintaining compatibility with existing systems and data structures.

22 – Basic key exchange :Diffie-Hellman

Diffie-Hellman algorithm:

The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.

P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Step-by-Step explanation is as follows:

Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated $x = G^a mod P$	$y = G^b mod P$ Key generated

Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a mod P$	Generated Secret Key $k_b = x^b mod P$

Algebraically, it can be shown that

$$k_a = k_b$$

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice: x =(9^4 mod 23) = (6561 mod 23) = 6

Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
Bob receives public key x = 6

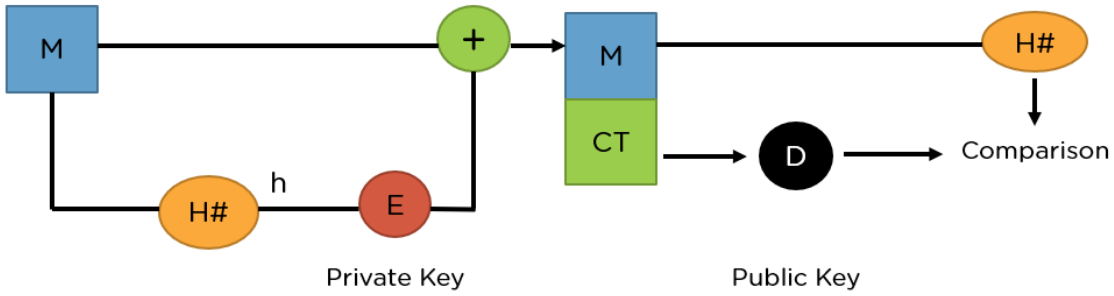
Step 6: Alice and Bob compute symmetric keys
Alice: ka = y^a mod p = 65536 mod 23 = 9
Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

23 –RSA and Merkle puzzles

RSA

The RSA algorithm is a public-key signature algorithm developed by Ron Rivest, Adi Shamir, and Leonard Adleman. Their paper was first published in 1977, and the algorithm uses logarithmic functions to keep the working complex enough to withstand brute force and streamlined enough to be fast post-deployment. The image below shows it verifies the digital signatures using RSA methodology.



M - Plaintext

H - Hash function

h - Hash digest

‘+’ - Bundle both plaintext and digest

E - Encryption

D - Decryption

RSA can also encrypt and decrypt general information to securely exchange data along with handling digital signature verification. The image above shows the entire procedure of the RSA algorithm. You will understand more about it in the next section.

RSA in Data Encryption

When using RSA for encryption and decryption of general data, it reverses the key set usage. Unlike signature verification, it uses the

receiver's public key to encrypt the data, and it uses the receiver's private key in decrypting the data. Thus, there is no need to exchange any keys in this scenario.

There are two broad components when it comes to RSA cryptography, they are:

- **Key Generation**: Generating the keys to be used for encrypting and decrypting the data to be exchanged.
- **Encryption/Decryption Function**: The steps that need to be run when scrambling and recovering the data.

Key Generation

You need to generate public and private keys before running the functions to generate your ciphertext and plaintext. They use certain variables and parameters, all of which are explained below:

- Choose two large prime numbers (p and q)
- Calculate n = p*q and z = (p-1)(q-1)
- Choose a number e where 1 < e < z
- Calculate d = e-1mod(p-1)(q-1)
- You can bundle private key pair as (n,d)
- You can bundle public key pair as (n,e)

Encryption/Decryption Function

Once you generate the keys, you pass the parameters to the functions that calculate your ciphertext and plaintext using the respective key.

- If the plaintext is m, ciphertext = me mod n.
- If the ciphertext is c, plaintext = cd mod n

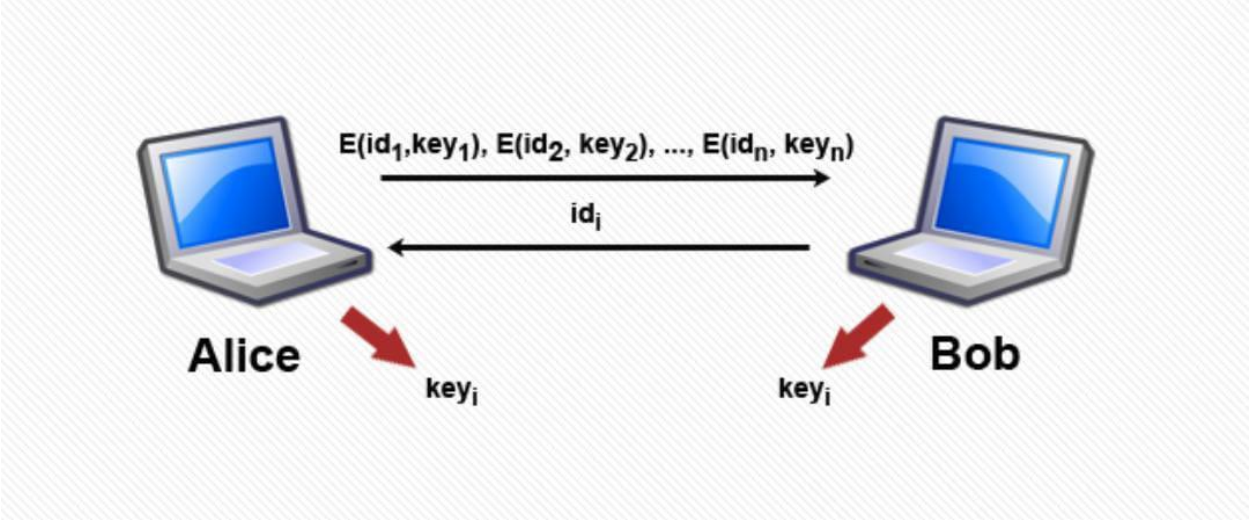
To understand the above steps better, you can take an example where p = 17 and q=13. Value of e can be 5 as it satisfies the condition $1 < e < (p-1)(q-1)$.

- N = p * q = 221
- D = e-1mod(p-1)(q-1) = 29
- Public Key pair = (221,5)
- Private Key pair = (221,29)
- If the plaintext(m) value is 10, you can encrypt it using the formula me mod n = 82.

- To decrypt this ciphertext(c) back to original data, you must use the formula cd mod n = 29.

Merkle Puzzles

The Merkle puzzle is a cryptographic construction used in information security to securely establish a shared secret key between two parties over an insecure communication channel. It was introduced by Ralph Merkle in 1974 and provides a solution to the key exchange problem.



The working principle of the Merkle puzzle can be explained as follows:

1. Puzzle Generation: The sender (Alice) generates a large number of encrypted puzzle pieces. Each puzzle piece is encrypted with a different candidate key.

2. Puzzle Transmission: Alice sends the encrypted puzzle pieces to the receiver (Bob) over the insecure channel.

3. Puzzle Solving: Bob selects a puzzle piece from the received set and attempts to decrypt it using a candidate key. He repeats this process with different puzzle pieces until he successfully decrypts one of them.

4. Key Exchange: Once Bob successfully decrypts a puzzle piece, he informs Alice by sending the index or identifier of the puzzle piece.

5. Key Confirmation: Alice checks the index or identifier received from Bob and reveals the corresponding candidate key used to encrypt the puzzle piece.

6. **Shared Key**: Both Alice and Bob now possess the shared secret key used for secure communication.

Here's an example to illustrate the Merkle puzzle:

1. Puzzle Generation: Alice generates a set of 1,000 encrypted puzzle pieces, each encrypted with a different candidate key.

2. Puzzle Transmission: Alice sends the encrypted puzzle pieces to Bob over an insecure channel.

3. Puzzle Solving: Bob randomly selects a puzzle piece, let's say the 567th piece, and tries to decrypt it using a candidate key.

4. Key Exchange: Bob successfully decrypts the puzzle piece and informs Alice that he solved the 567th puzzle piece.

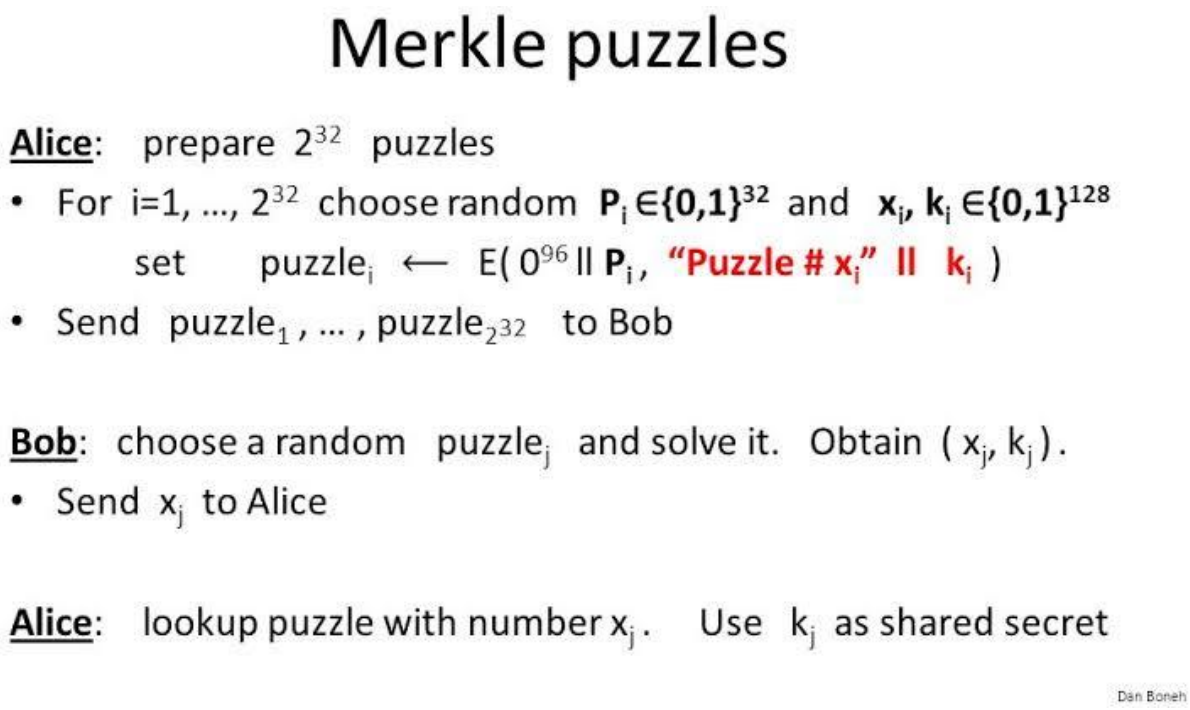
5. Key Confirmation: Alice verifies the information and reveals the corresponding candidate key used to encrypt the 567th puzzle piece.

6. Shared Key: Both Alice and Bob now possess the shared secret key derived from the decrypted puzzle piece, which they can use for secure communication.

The Merkle puzzle provides security by making it computationally infeasible for an eavesdropper to discover the shared secret key by decrypting multiple puzzle pieces. Additionally, the puzzle complexity can be adjusted by varying the number of puzzle pieces generated, making it resistant to brute-force attacks.

It's worth noting that the Merkle puzzle is a conceptual construction and may not be the most practical solution for key exchange in modern cryptographic protocols. It serves as an important milestone in the development of cryptographic primitives and has paved the way for

more efficient and secure key exchange mechanisms like Diffie-Hellman key exchange and public-key cryptography.



24 –Computational number theory

Computational number theory plays a crucial role in information security, particularly in areas such as encryption, key exchange protocols, and digital signatures. It provides the mathematical foundations for secure communication and data protection. Here's a brief overview of how computational number theory is applied in information security, along with an example:

1. Modular Arithmetic: Modular arithmetic is a fundamental concept in computational number theory. It involves performing arithmetic

operations on integers modulo a given number, often denoted as "mod n." Modular arithmetic is used in various cryptographic algorithms to achieve properties like confidentiality, integrity, and authenticity.

2. Prime Numbers: Prime numbers are essential in information security. Large prime numbers are used in various cryptographic algorithms, such as RSA (Rivest-Shamir-Adleman), to generate secure keys. The security of these algorithms relies on the difficulty of factoring large composite numbers into their prime factors.

3. Discrete Logarithm Problem: The discrete logarithm problem (DLP) is another significant concept in computational number theory. It states that given a group, a generator element, and a result of raising the generator to a power, it is computationally difficult to determine the exponent. This problem forms the basis of various cryptographic schemes, including Diffie-Hellman key exchange and elliptic curve cryptography.

Example: Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange is a widely used key exchange protocol that relies on the computational difficulty of the discrete logarithm problem. Here's a simplified example of how it works:

- Alice and Bob agree on a large prime number, p, and a primitive root modulo p, g. These parameters are public.
- Alice chooses a random secret number, a, and computes A = g^a mod p.
- Bob chooses a random secret number, b, and computes B = g^b mod p.
- Alice and Bob exchange their computed values, A and B, respectively.
- Alice calculates the shared secret key as s = B^a mod p.
- Bob calculates the shared secret key as s = A^b mod p.

Now, Alice and Bob both have the same shared secret key, which they can use for symmetric encryption or other purposes. An eavesdropper who intercepts A and B but does not know a or b would need to solve the discrete logarithm problem to compute the shared secret key, which is computationally infeasible for large prime numbers.

This example demonstrates how computational number theory is applied in the Diffie-Hellman key exchange, providing a secure way for two parties to establish a shared secret key over an insecure channel.

25 –Number theoretic hardness assumptions

Number theoretic hardness assumptions form the basis of many cryptographic algorithms and protocols in information security. These assumptions involve computational problems that are believed to be

difficult to solve efficiently. Here's an explanation of number theoretic hardness assumptions and an example:

1. Integer Factorization Assumption: The integer factorization assumption is based on the belief that factoring a large composite number into its prime factors is computationally hard. This assumption underlies the security of various encryption schemes, including the widely used RSA algorithm. The security of RSA relies on the difficulty of factoring large numbers, making it impractical for an attacker to determine the private key given the public key.

Example: In RSA encryption, Bob generates a public key by choosing two large prime numbers, p and q, and computing their product n = p * q. The factors p and q are kept secret. The public key consists of the modulus n and an encryption exponent e. Alice can encrypt a message using Bob's public key and send it to him. Bob, who knows the factors p and q, can efficiently compute the corresponding private key to decrypt the message. The security of this encryption scheme is based on the assumption that factoring large numbers is difficult.

2. Discrete Logarithm Assumption: The discrete logarithm assumption (DLP) is based on the belief that computing the discrete logarithm in a finite group is computationally hard. This assumption forms the basis of various cryptographic schemes, such as Diffie-Hellman key exchange and elliptic curve cryptography.

Example: In the Diffie-Hellman key exchange protocol, Alice and Bob agree on a large prime number p and a primitive root modulo p, g. Alice chooses a secret exponent a, calculates A = g^a mod p, and sends it to Bob. Bob chooses a secret exponent b, calculates B = g^b mod p, and sends it to Alice. They both independently compute the shared secret key as s = A^b mod p = B^a mod p. The security of this protocol relies on the assumption that it is computationally hard to calculate the secret exponent a or b given g, p, and A or B.

3. Elliptic Curve Discrete Logarithm Assumption: The elliptic curve discrete logarithm assumption (ECDLP) is similar to the discrete logarithm assumption but applied to elliptic curve groups. It assumes that computing the discrete logarithm in an elliptic curve group is computationally hard. This assumption is the foundation for elliptic curve cryptography (ECC), which offers strong security with shorter key sizes compared to traditional cryptographic algorithms.

Example: In elliptic curve cryptography, instead of working with modular arithmetic, computations are performed on points defined on an elliptic curve. The security of ECC relies on the assumption that it is computationally hard to compute the discrete logarithm in the elliptic curve group. This assumption forms the basis for various ECC-based algorithms, such as elliptic curve Diffie-Hellman (ECDH) key exchange and elliptic curve digital signature algorithms (ECDSA).

These number theoretic hardness assumptions provide the underlying security foundations for many cryptographic algorithms and protocols in

information security. While they are widely believed to be true, their security depends on the current state of mathematical knowledge and the absence of efficient algorithms for solving these problems.

26 –Public Key Encryption

When the two parties communicate to each other to transfer the intelligible or sensible message, referred to as plaintext, is converted into apparently random nonsense for security purpose referred to as ciphertext.

Encryption:

The process of changing the plaintext into the ciphertext is referred to as encryption.

The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext.

The security of conventional encryption depends on the major two factors:

- **The Encryption algorithm**
- **Secrecy of the key**

Once the ciphertext is produced, it may be transmitted. The Encryption algorithm will produce a different output depending on the specific key being used at the time. Changing the key changes the output of the algorithm.

Once the ciphertext is produced, it may be transmitted. Upon reception, the ciphertext can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption.

Decryption:

The process of changing the ciphertext to the plaintext that process is known as decryption.

Public Key Encryption :

Asymmetric is a form of Cryptosystem in which encryption and decryption are performed using different keys-Public key (known to everyone) and Private key (Secret key). This is known as Public Key Encryption.

basis	Encryption	Public- Key Encryption
Required for Work:	• Same algorithm with the same key is used for encryption and decryption. • The sender and receiver must share the algorithm and key.	• One algorithm is used for encryption and a related algorithm decryption with pair of keys, one for encryption and other for decryption. • Receiver and Sender must each have one of the matched pair of keys (not identical) .
Required for Security:	• Key must be kept secret. • If the key is secret, it is very impossible to decipher message. • Knowledge of the algorithm plus samples of	• One of the two keys must be kept secret. • If one of the key is kept secret, it is very impossible to decipher message.

	ciphertext must be impractical to determine the key.	• Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be impractical to determine the other key.

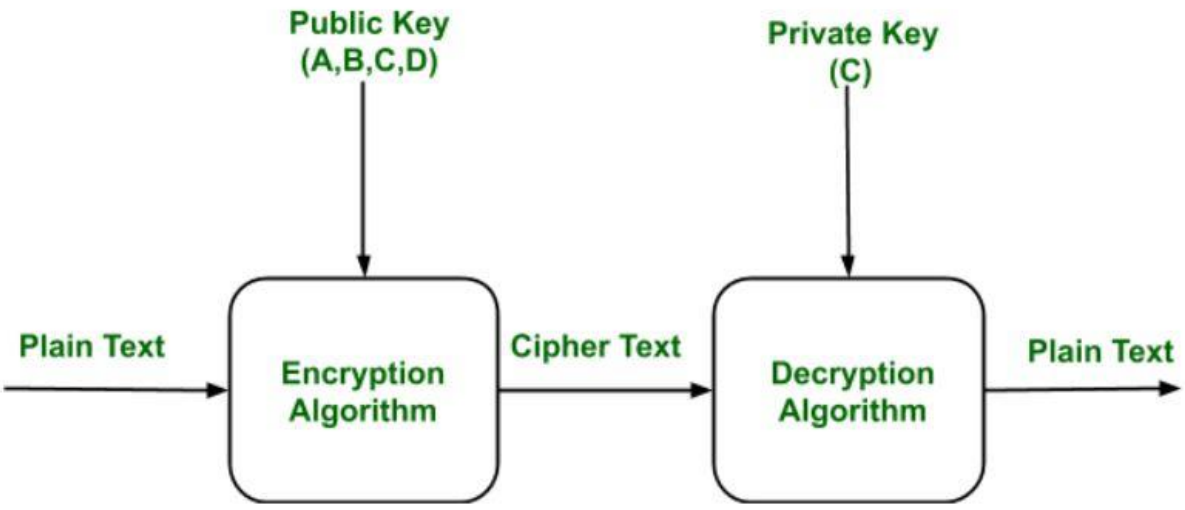
Characteristics of Public Encryption key:

- Public key Encryption is important because it is infeasible to determine the decryption key given only the knowledge of the cryptographic algorithm and encryption key.
- Either of the two keys (Public and Private key) can be used for encryption with other key used for decryption.
- Due to Public key cryptosystem, public keys can be freely shared, allowing users an easy and convenient method for encrypting content and verifying digital signatures, and private keys can be kept secret, ensuring only the owners of the private keys can decrypt content and create digital signatures.
- The most widely used public-key cryptosystem is RSA (Rivest–Shamir–Adleman). The difficulty of finding the prime factors of a composite number is the backbone of RSA.

Example:

Public keys of every user are present in the Public key Register. If B wants to send a confidential message to C, then B encrypt the message using C Public key. When C receives the message from B then C can

decrypt it using its own Private key. No other recipient other than C can decrypt the message because only C know C's private key.



Components of Public Key Encryption:

Plain Text:

This is the message which is readable or understandable. This message is given to the Encryption algorithm as an input.

Cipher Text:

The cipher text is produced as an output of Encryption algorithm. We cannot simply understand this message.

Encryption Algorithm:

The encryption algorithm is used to convert plain text into cipher text.

Decryption Algorithm:

It accepts the cipher text as input and the matching key (Private Key or Public key) and produces the original plain text

Public and Private Key:

One key either Private key (Secret key) or Public Key (known to everyone) is used for encryption and other is used for decryption.

27 –Trapdoor Permutations and RSA

A trapdoor permutation is a cryptographic primitive that allows for efficient computation in one direction, while making the inverse computation computationally difficult without specific knowledge called the "trapdoor." It forms the basis for various cryptographic schemes, including public key encryption and digital signatures. Here's an explanation of how trapdoor permutations work and an example:

1. **Definition:** A trapdoor permutation is a function that is easy to compute in one direction but hard to invert without knowledge of additional information (the trapdoor). Given the output of the function, it should be computationally difficult to determine the input without the trapdoor information.

2. Key Generation: To create a trapdoor permutation, a specific algorithm is used to generate a pair of keys: a public key and a private key. The public key is made available to everyone, while the private key is kept secret.

3. Encryption: Given a message M, the trapdoor permutation can be applied with the public key to generate a ciphertext C. The encryption process is efficient and can be performed by anyone who has access to the public key.

4. Decryption: The decryption process is where the trapdoor comes into play. With the private key (which includes the trapdoor information), the recipient can efficiently compute the inverse of the trapdoor permutation and obtain the original message M from the ciphertext C.

Example: RSA Encryption Scheme

The RSA encryption scheme is an example of a trapdoor permutation. It is based on the computational difficulty of factoring large composite numbers into their prime factors. Here's how RSA works:

1. Key Generation: Bob generates a public key (e, n) and a private key (d, n) as follows:

- Bob chooses two large prime numbers, p and q.

- Bob computes n = p * q, which serves as the modulus for the RSA scheme.

- Bob computes the totient of n as $\varphi(n) = (p - 1) * (q - 1)$.

- Bob chooses an encryption exponent e that is relatively prime to φ(n) (i.e., gcd(e, φ(n)) = 1).

- Bob calculates the decryption exponent d such that (e * d) mod φ(n) = 1.

Bob publishes the public key (e, n) and keeps the private key (d, n) secret.

2. Encryption: Alice wants to send a message M to Bob. She converts the message into a numerical representation (plaintext) and encrypts it using Bob's public key:

- Alice obtains Bob's public key (e, n).

- Alice computes the ciphertext C as C = M^e mod n.

Alice sends the ciphertext C to Bob.

3. Decryption: Bob receives the ciphertext C and decrypts it using his private key:

- Bob applies the trapdoor permutation by computing M = C^d mod n.

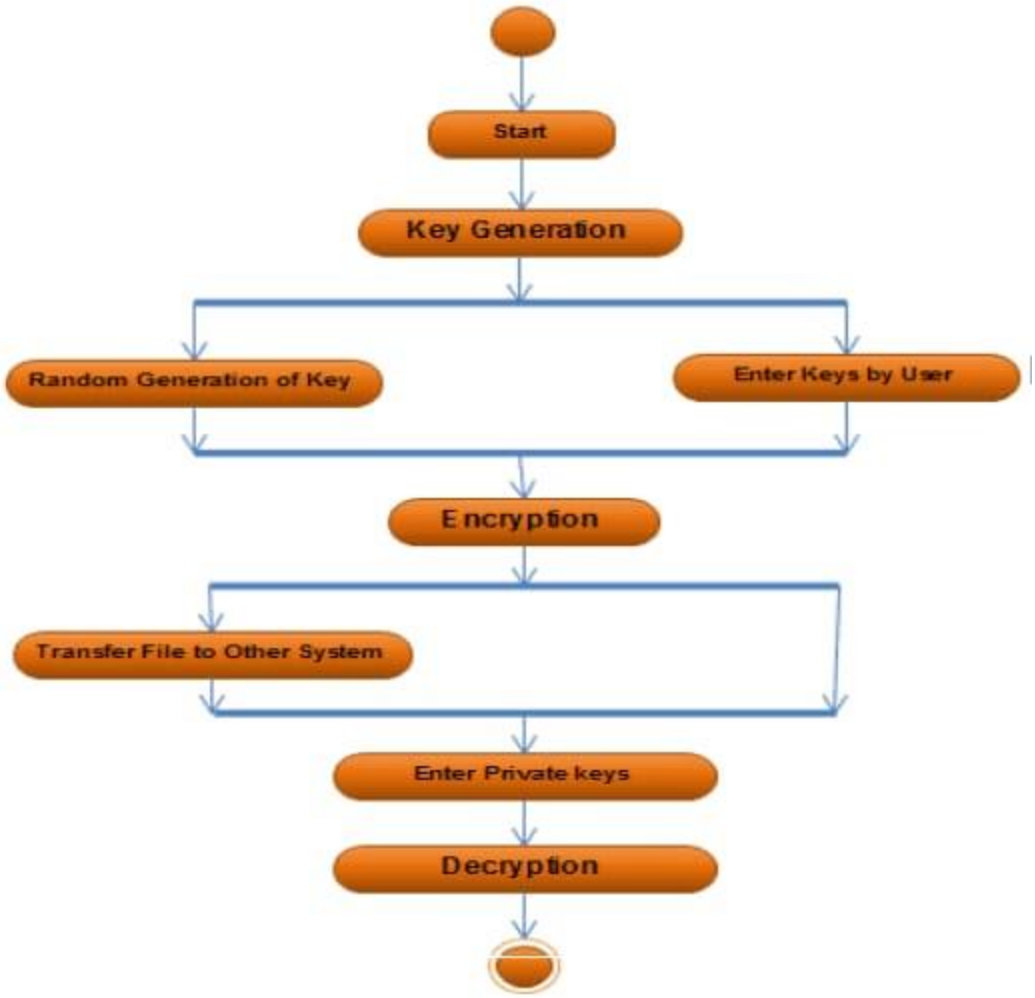
Bob obtains the original message M.

In this example, the trapdoor permutation property of RSA is based on the difficulty of factoring large numbers, making it computationally infeasible for an attacker to determine the private key and decrypt the ciphertext without knowledge of the prime factors of n.

Trapdoor permutations provide a powerful tool for achieving secure communication and data protection in information security, as they enable efficient encryption and decryption operations while maintaining the confidentiality of the private key.

28 – The Elgamal system and variants

The ElGamal encryption system is a public-key encryption scheme based on the computational difficulty of the discrete logarithm problem. It provides a secure way to encrypt messages and is widely used in information security. Variants of the ElGamal system have been developed to address specific requirements or enhance its security. Here's an explanation of how the ElGamal system works, along with examples of its variants:



1. ElGamal Encryption System:

Key Generation:

- Bob chooses a large prime number, p, and a generator g of the multiplicative group modulo p.

- Bob selects a private key, a, as a random integer between 1 and p-1.

- Bob computes his public key, A, as A = g^a mod p.

- Bob publishes the public key (A, g, p) and keeps the private key a secret.

Encryption:

- Alice wants to send a message M to Bob.
- Alice randomly chooses a secret number, k, between 1 and p-1.
- Alice computes the ciphertext as follows:
  - c1 = g^k mod p.
  - c2 = M * (A^k) mod p.
- Alice sends the ciphertext (c1, c2) to Bob.

Decryption:

- Bob receives the ciphertext (c1, c2).
- Bob computes the shared secret key as s = (c1^a)^-1 mod p.
- Bob decrypts the message by computing M = c2 * s mod p.

The security of the ElGamal system is based on the difficulty of computing the discrete logarithm, specifically the difficulty of determining k given c1 and c2.

2. ElGamal Variant: ElGamal Digital Signature

The ElGamal encryption system can be modified to create an ElGamal digital signature scheme. Instead of encrypting a message, the sender

signs a message to provide authenticity and integrity. Here's an overview of the ElGamal digital signature scheme:

Key Generation:

- Bob chooses a large prime number, p, and a generator g of the multiplicative group modulo p.

- Bob selects a private key, a, as a random integer between 1 and p-1.

- Bob computes his public key, A, as A = g^a mod p.

- Bob publishes the public key (A, g, p) and keeps the private key a secret.

Signature Generation:

- Alice wants to sign a message M.

- Alice randomly chooses a secret number, k, between 1 and p-1.

- Alice computes the signature as follows:

  - r = g^k mod p.

  - s = (M - a*r) * k^-1 mod (p-1).

- Alice sends the signature (r, s) along with the message M.

Signature Verification:

- Bob receives the message M and the signature (r, s).

- Bob computes the verification value as follows:

- v1 = (A^r * r^s) mod p.

- v2 = g^M mod p.

- If v1 = v2, Bob accepts the signature as valid; otherwise, it is rejected.

The ElGamal digital signature scheme provides a way for the recipient to verify the authenticity and integrity of the received message.

3. **ElGamal Variant: Elliptic Curve ElGamal**

To improve efficiency and provide stronger security, the ElGamal encryption system can be adapted to use elliptic curve cryptography (ECC). The Elliptic Curve ElGamal scheme operates similarly to the original ElGamal system but utilizes points on an elliptic curve instead of modular arithmetic. The computations are performed using elliptic curve group operations, such as point addition and scalar multiplication, making it more efficient.

Thank You

END

(CR)Kashif Malik

(GR)Misbah Mustafa