UNIT I-COMPUTER SECURITY CONCEPTS

Computer security encompasses various concepts to protect systems and data. Some key concepts include:

1. **Authentication: Verifying the identity of users or systems to ensure they have the right access privileges.

2. **Authorization: Granting or restricting access to resources based on authenticated identities and their permissions.

3. **Encryption: Protecting sensitive information by converting it into a secure, unreadable format that can only be deciphered with the appropriate key.

4. **Firewalls: Implementing barriers to control incoming and outgoing network traffic, preventing unauthorized access and potential security threats.

5. **Antivirus and Anti-malware: Using software to detect, prevent, and remove malicious software that could compromise the system.

6. **Patch Management: Regularly updating and applying patches to software and systems to fix vulnerabilities and improve security.

7. **Intrusion Detection and Prevention Systems (IDPS): monitoring network or system activities to detect and respond to potential security threats.

8. **Security Policies: Establishing rules and guidelines to govern the organization's approach to security, including user behavior, data handling, and system configurations.

9. **Incident Response: Developing procedures to effectively respond to and recover from security incidents or breaches.

10. **Security Awareness: Educating users and employees about security risks and best practices to reduce the likelihood of human-related security issues.

11. **Physical Security: Protecting hardware, servers, and data centers from physical threats, such as theft, vandalism, or natural disasters.

12. **Secure Software Development: Integrating security measures throughout the software development lifecycle to create robust and secure applications.

13. **Network Security: Safeguarding the integrity and confidentiality of data during transmission across networks, often through protocols like HTTPS.

14. **Two-Factor Authentication (2FA) and Multi-Factor Authentication (MFA):** Adding an extra layer of security by requiring more than one method of authentication, such as a password and a code sent to a mobile device.

15. **Security Audits and Assessments: Regularly evaluating and testing systems for vulnerabilities to identify and address potential security risks.

Implementing a comprehensive approach that combines these concepts helps create a strong defense against various cyber security threats.
In the context of cryptography, the OSI Security Architecture addresses security concerns at various layers of the OSI model. Here's how cryptography is typically applied to each layer:

1. **Physical Layer (Layer 1):
   - Cryptography at this layer is less common, as it primarily deals with the transmission of raw bits over a physical medium. However, encryption may be used in certain cases to secure physical connections.

2. **Data Link Layer (Layer 2):
   - **Security Focus:** MAC address filtering and cryptographic techniques like MAC sec (Media Access Control Security) to ensure the integrity and confidentiality of data at this layer.

3. **Network Layer (Layer 3):
   - **Security Focus: IPsec (Internet Protocol Security) is commonly used at this layer for securing communication between network devices. It provides protocols for authentication and encryption at the network layer.

4. **Transport Layer (Layer 4):
   - **Security Focus:** TLS (Transport Layer Security) and its predecessor SSL (Secure Sockets Layer) operate at this layer, providing encryption and authentication for end-to-end communication between applications.

5. **Session Layer (Layer 5):
   - **Security Focus:** Cryptographic techniques may be applied for secure session establishment, maintenance, and termination. This could involve the use of session keys for encryption.

6. **Presentation Layer (Layer 6):
   - **Security Focus:** Encryption and encoding formats are managed at this layer, ensuring secure data exchange by converting data into a format that can be transmitted securely.

7. **Application Layer (Layer 7):
   - **Security Focus:** Cryptographic protocols such as HTTPS (HTTP Secure) are used to secure communication between applications. This involves encrypting data exchanged between web browsers and servers.

Cryptography plays a crucial role in securing communications at different layers of the OSI model. It provides confidentiality, integrity, and authentication mechanisms to protect data as it traverses a network. The choice of cryptographic techniques depends on the specific requirements and security considerations at each layer.

Computer and network security

Cryptography is the science of keeping messages private. A cryptographic system uses encryption keys between two trusted communication partners. These keys encrypt and decrypt information so that the information is known only to those who have the keys.

There are two kinds of cryptographic systems: symmetric-key and asymmetric-key. Symmetric-key (or secret-key) systems use the same secret key to encrypt and decrypt a message. Asymmetric-key (or public-key) systems use one key (public) to encrypt a message and a different key (private) to decrypt it. Symmetric-key systems are simpler and faster, but two parties must somehow exchange the key in a secure way because if the secret key is discovered by outside parties, security is compromised. Asymmetric-key systems, commonly known as public-key systems, avoid this problem because the public key may be freely distributed, but the private key is never transmitted.

Cryptography provides information security as follows:

Authentication: verifies that the entity on the other end of a communications link is the intended recipient of a transmission.
Non-repudiation provides undeniable proof of origin of transmitted data.

Data integrity ensures that information is not altered during transmission.

Data confidentiality ensures that data remains private during transmission

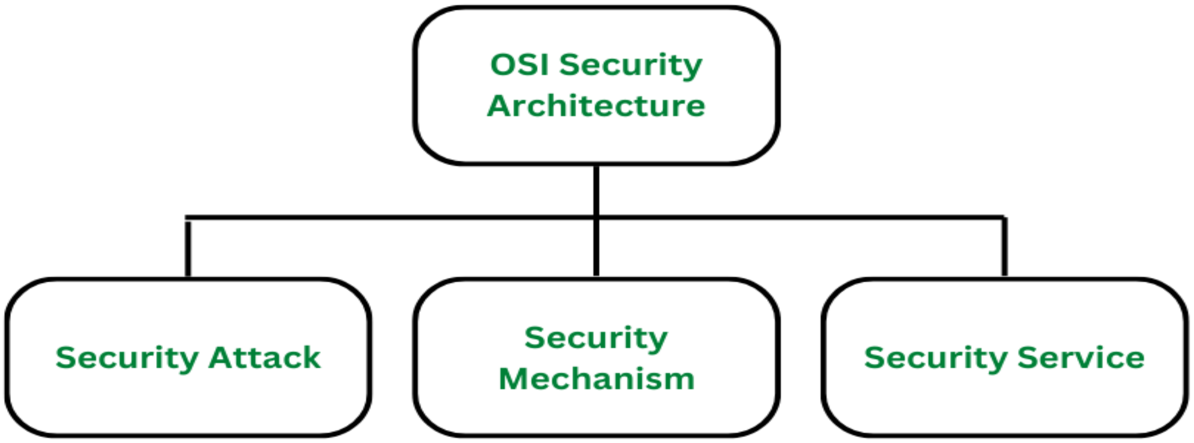
OSI Security Architecture

The security of an organization is the greatest concern of the people working at the organization. Safety and security are the pillars of cyber technology. It is hard to imagine the cyber world without thinking about security. The architecture of security is thus a very important aspect of the organization. The OSI (Open Systems Interconnection) Security Architecture defines a systematic approach to providing security at each layer. It defines security services and security mechanisms that can be used at each of the seven layers of the OSI model to provide security for data transmitted over a network. These security services and mechanisms help to ensure the confidentiality, integrity, and availability of the data.

OSI architecture is internationally acceptable as it lays the flow of providing safety in an organization.

OSI Security Architecture focuses on these concepts:

Security Attack:
Security mechanism: A security mechanism is a means of protecting a system, network, or device against unauthorized access, tampering, or other security threats.
Security Service:



Classification of OSI Security Architecture

OSI Security Architecture is categorized into three broad categories namely Security Attacks, Security mechanisms, and Security Services. We will discuss each in detail

1. Security Attacks:

A security attack is an attempt by a person or entity to gain unauthorized access to disrupt or compromise the security of a system, network, or device. These are defined as the actions that put at risk an organization's safety. They are further classified into 2 sub-categories:

A. Passive Attack:
Attacks in which a third-party intruder tries to access the message/ content/ data being shared by the sender and receiver by keeping a close watch on the transmission or eave-dropping the transmission is called Passive Attacks. These types of attacks involve the attacker observing or monitoring system, network, or device activity without actively disrupting or altering it. Passive attacks are typically focused on gathering information or intelligence, rather than causing damage or disruption.

Here, both the sender and receiver have no clue that their message/ data are accessible to some third-party intruder. The message/ data transmitted remains in its usual form without any deviation from its usual behavior.

This makes passive attacks very risky as there is no information provided about the attack happening in the communication process. One way to prevent passive attacks is to encrypt the message/data that needs to be transmitted; this will prevent third-party intruders to use the information though it would be accessible to them.

Passive attacks are further divided into two parts based on their behavior:

- Eavesdropping: This involves the attacker intercepting and listening to communications between two or more parties without their knowledge or consent. Eavesdropping can be performed using a variety of techniques, such as packet sniffing, or man-in-the-middle attacks.
- Traffic analysis: This involves the attacker analyzing network traffic patterns and metadata to gather information about the system, network, or device. Here the intruder can't read the message but only understands the pattern and length of encryption. Traffic analysis can be performed using a variety of techniques, such as network flow analysis, or protocol analysis

B. Active Attacks: Active attacks refer to types of attacks that involve the attacker actively disrupting or altering system, network, or device activity. Active attacks are typically focused on causing damage or disruption, rather than gathering information or intelligence. Here, both the sender and receiver have no clue that their message/ data are modified by some third-party intruder. The message/ data transmitted don't remain in its usual form and shows deviation from its usual behavior. This makes active attacks dangerous as there is no information provided of the attack happening in the communication process and the receiver is not aware that the data/ message received is not from the sender.

Active attacks are further divided into four parts based on their behavior:

- Masquerade is a type of attack in which the attacker pretends to be an authentic sender in order to gain unauthorized access to a system. This type of attack can involve the attacker using stolen or forged credentials, or manipulating authentication or authorization controls in some other way.

- Replay is a type of active attack in which the attacker intercepts a transmitted message through a passive channel and then maliciously or fraudulently replays or delays it at a later time.
- Modification of Message involves the attacker modifying the transmitted message and making the final message received by the receiver look like it's not safe or non-meaningful. This type of attack can be used to manipulate the content of the message or to disrupt the communication process.
- Denial of service (DoS) attacks involves the attacker sending a large volume of traffic to a system, network, or device in an attempt to overwhelm it and make it unavailable to legitimate users.

2. Security Mechanism

The mechanism that is built to identify any breach of security or attack on the organization is called a security mechanism. Security Mechanisms are also responsible for protecting a system, network, or device against unauthorized access, tampering, or other security threats. Security mechanisms can be implemented at various levels within a system or network and can be used to provide different types of security, such as confidentiality, integrity, or availability.

Some examples of security mechanisms include:

- Encipherment (Encryption) involves the use of algorithms to transform data into a form that can only be read by someone with the appropriate decryption key. Encryption can be used to protect data it is transmitted over a network, or to protect data when it is stored on a device.
- Digital signature is a security mechanism that involves the use of cryptographic techniques to create a unique, verifiable identifier for a digital document or message, which can be used to ensure the authenticity and integrity of the document or message.
- Traffic padding is a technique used to add extra data to a network traffic stream in an attempt to obscure the true content of the traffic and make it more difficult to analyze.
- Routing control allows the selection of specific physically secure routes for specific data transmission and enables routing changes, particularly when a gap in security is suspected.

3. Security Services:

Security services refer to the different services available for maintaining the security and safety of an organization. They help in preventing any potential risks to security. Security services are divided into 5 types:

- Authentication is the process of verifying the identity of a user or device in order to grant or deny access to a system or device.
- Access control involves the use of policies and procedures to determine who is allowed to access specific resources within a system.
- Data Confidentiality is responsible for the protection of information from being accessed or disclosed to unauthorized parties.
- Data integrity is a security mechanism that involves the use of techniques to ensure that data has not been tampered with or altered in any way during transmission or storage.
- Non- repudiation involves the use of techniques to create a verifiable record of the origin and transmission of a message, which can be used to prevent the sender from denying that they sent the message.

Benefits of OSI Architecture:

Below listed are the benefits of OSI Architecture in an organization:

1. Providing Security:

- OSI Architecture in an organization provides the needed security and safety, preventing potential threats and risks.
- Managers can easily take care of the security and there is hassle-free security maintenance done through OSI Architecture.

2. Organizing Task:

- The OSI architecture makes it easy for managers to build a security model for the organization based on strong security principles.
- Managers get the opportunity to organize tasks in an organization effectively.

3. Meets International Standards:

- Security services are defined and recognized internationally meeting international standards.
- The standard definition of requirements defined using OSI Architecture is globally accepted.

Active and Passive attacks in Information Security

It’s important that the distinction between active and passive attacks can be blurry, and some attacks may involve elements of both. Additionally, not all attacks are technical in nature; social engineering attacks, where an attacker manipulates or deceives users in order to gain access to sensitive information, are also a common form of attack.

Active attacks:

Active attacks are a type of cyber security attack in which an attacker attempts to alter, destroy, or disrupt the normal operation of a system or network. Active attacks involve the attacker taking direct action against the target system or network, and can be more dangerous than passive attacks, which involve simply monitoring or eavesdropping on a system or network.

Types of active attacks are as follows:

Masquerade

Modification of messages

Repudiation

Replay

Denial of Service

Masquerade: Masquerade is a type of cyber security attack in which an attacker pretends to be someone else in order to gain access to systems or data. This can involve impersonating a legitimate user or system to trick other users or systems into providing sensitive information or granting access to restricted areas.

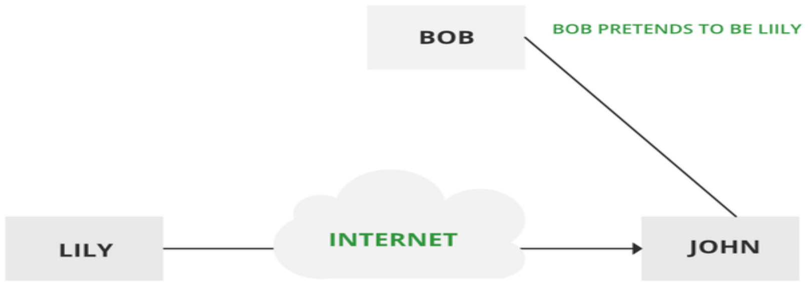
There are several types of masquerade attacks, including:

Username and password masquerade: In a username and password masquerade attack, an attacker uses stolen or forged credentials to log into a system or application as a legitimate user.

IP address masquerade: In an IP address masquerade attack, an attacker spoofs or forges their IP address to make it appear as though they are accessing a system or application from a trusted source.

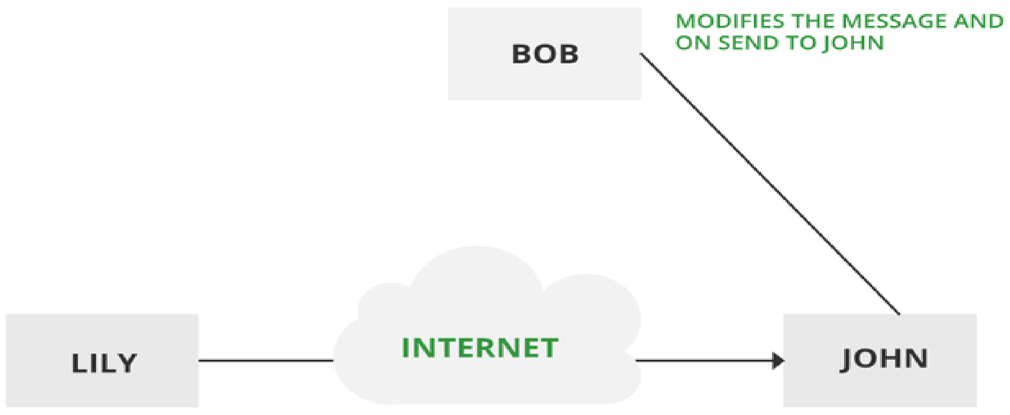
Website masquerade: In a website masquerade attack, an attacker creates a fake website that appears to be legitimate in order to trick users into providing sensitive information or downloading malware.

Email masquerade: In an email masquerade attack, an attacker sends an email that appears to be from a trusted source, such as a bank or government agency, in order to trick the recipient into providing sensitive information or downloading malware.



Modification of messages –

It means that some portion of a message is altered or that message is delayed or reordered to produce an unauthorized effect. Modification is an attack on the integrity of the original data. It basically means that unauthorized parties not only gain access to data but also spoof the data by triggering denial-of-service attacks, such as altering transmitted data packets or flooding the network with fake data. Manufacturing is an attack on authentication. For example, a message meaning “Allow JOHN to read confidential file X” is modified as “Allow Smith to read confidential file X”.



Repudiation attacks are a type of cyber security attack in which an attacker attempts to deny or repudiate actions that they have taken, such as making a transaction or sending a message. These attacks can be a serious problem because they can make it difficult to track down the source of the attack or determine who is responsible for a particular action.

There are several types of repudiation attacks, including:

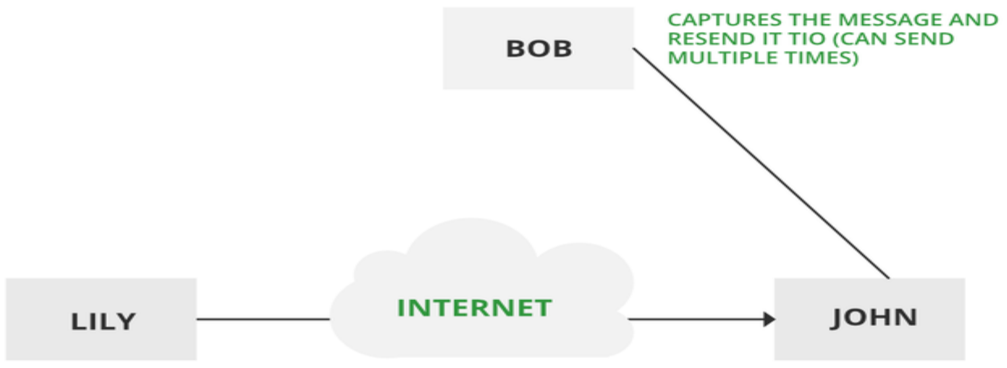
Message repudiation attacks: In a message repudiation attack, an attacker sends a message and then later denies having sent it. This can be done by using spoofed or falsified headers or by exploiting vulnerabilities in the messaging system.

Transaction repudiation attacks: In a transaction repudiation attack, an attacker makes a transaction, such as a financial transaction, and then later denies having made it. This can be done by exploiting vulnerabilities in the transaction processing system or by using stolen or falsified credentials.

Data repudiation attacks: In a data repudiation attack, an attacker modifies or deletes data and then later denies having done so. This can be done by exploiting vulnerabilities in the data storage system or by using stolen or falsified credentials.

Replay –

It involves the passive capture of a message and its subsequent transmission to produce an authorized effect. In this attack, the basic aim of the attacker is to save a copy of the data originally present on that particular network and later on use this data for personal uses. Once the data is corrupted or leaked it is insecure and unsafe for the users.


BOB
CAPTURES THE MESSAGE AND RESEND IT TIO (CAN SEND MULTIPLE TIMES)
LILY
INTERNET
JOHN

Denial of Service (DoS) is a type of cyber security attack that is designed to make a system or network unavailable to its intended users by overwhelming it with traffic or requests.

In a DoS attack, an attacker floods a target system or network with traffic or requests in order to consume its resources, such as bandwidth, CPU cycles, or memory, and prevent legitimate users from accessing it.

There are several types of DoS attacks, including:

Flood attacks: In a flood attack, an attacker sends a large number of packets or requests to a target system or network in order to overwhelm its resources.

Amplification attacks: In an amplification attack, an attacker uses a third-party system or network to amplify their attack traffic and direct it towards the target system or network, making the attack more effective.

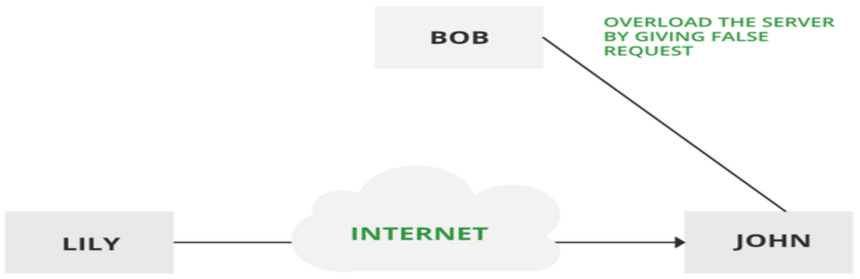
To prevent DoS attacks, organizations can implement several measures, such as:

1. Using firewalls and intrusion detection systems to monitor network traffic and block suspicious activity.

2. Limiting the number of requests or connections that can be made to a system or network.

3. Using load balancers and distributed systems to distribute traffic across multiple servers or networks.

4. Implementing network segmentation and access controls to limit the impact of a DoS attack.



Passive attacks: A Passive attack attempts to learn or make use of information from the system but does not affect system resources. Passive Attacks are in the nature of eavesdropping on or monitoring transmission. The goal of the opponent is to obtain information that is being transmitted. Passive attacks involve an attacker passively monitoring or collecting data without altering or destroying it.

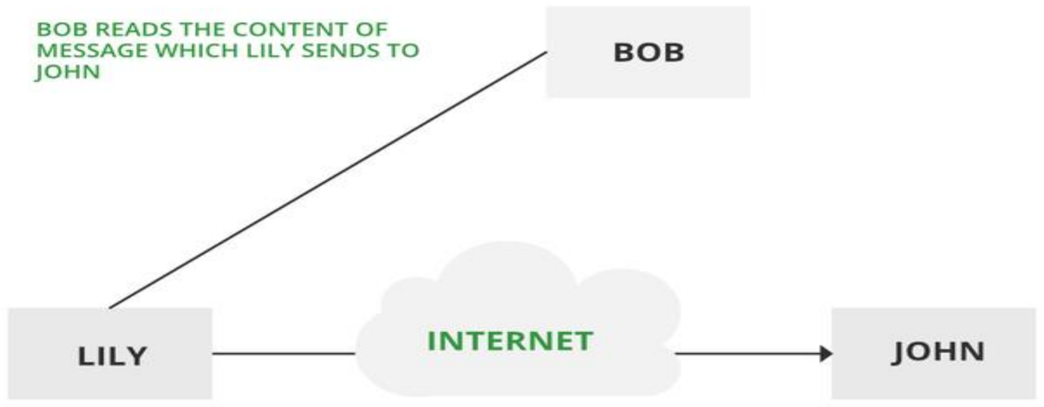
Examples of passive attacks include eavesdropping, where an attacker listens in on network traffic to collect sensitive information, and sniffing, where an attacker captures and analyzes data packets to steal sensitive information.

Types of Passive attacks are as follows:

- The release of message content
- Traffic analysis

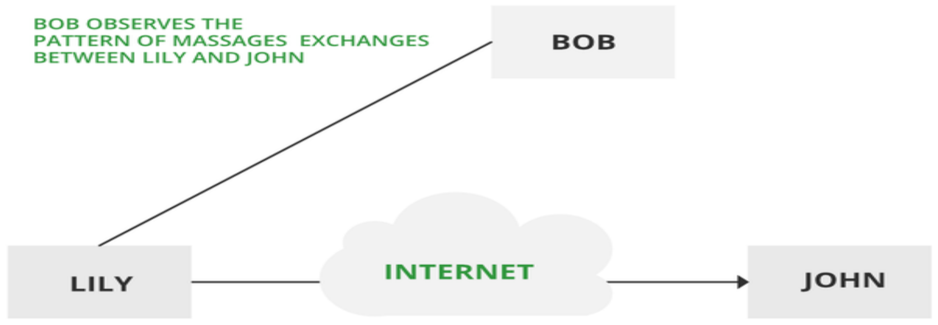
The release of message content –

Telephonic conversation, an electronic mail message, or a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.



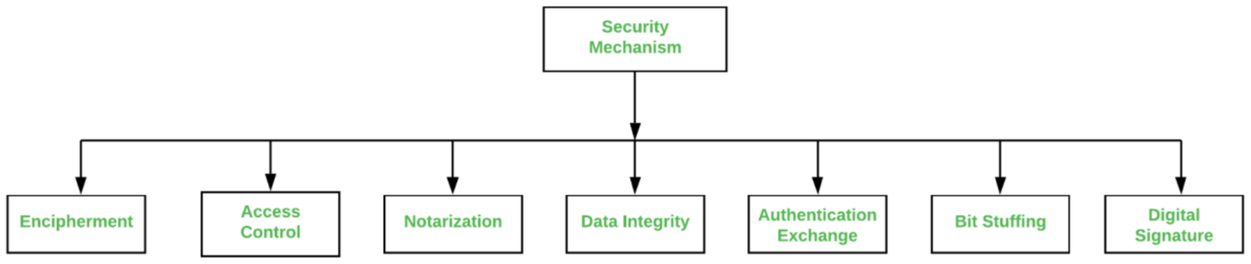
Traffic analysis –

Suppose that we had a way of masking (encryption) information, so that the attacker, even if captured the message, could not extract any information from the message. The opponent could determine the location and identity of the communicating host and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place. The most useful protection against traffic analysis is encryption of SIP traffic. To do this, an attacker would have to access the SIP proxy (or its call log) to determine who made the call.


BOB OBSERVES THE PATTERN OF MASSAGES EXCHANGES BETWEEN LILY AND JOHN
BOB
LILY
INTERNET
JOHN

SECURITY MECHANISM:

Network Security is a field in computer technology that deals with ensuring security of computer network infrastructure. As the network is very necessary for sharing of information whether it is at hardware level such as printer, scanner, or at software level. Therefore security mechanisms can also be termed as a set of processes that deal with recovery from security attacks. Various mechanisms are designed to recover from these specific attacks at various protocol layers.



Types of Security Mechanism are :

1. Decipherment:

   This security mechanism deals with hiding and covering of data which helps data to become confidential. It is achieved by applying mathematical calculations or algorithms which reconstruct information into non-readable form. It is achieved by two famous techniques named Cryptography and Decipherment. Level of data encryption is dependent on the algorithm used for decipherment.

2. Access Control:

This mechanism is used to stop unattended access to data which you are sending. It can be achieved by various techniques such as applying passwords, using a firewall, or just by adding PIN to data.

2. Notarization:

   This security mechanism involves use of trusted third parties in communication. It acts as a mediator between sender and receiver so that any chance of conflict is reduced. This mediator keeps record of requests made by sender to receiver for later denied.

3. Data Integrity:

   This security mechanism is used by appending value to data to which is created by data itself. It is similar to sending packet of information known to both sending and receiving parties and checked before and after data is received. When this packet or data which is appended is checked and is the same while sending and receiving data integrity is maintained.

4. Authentication exchange:

   This security mechanism deals with identity to be known in communication. This is achieved at the TCP/IP layer where two-way handshaking mechanism is used to ensure data is sent or not

5. Bit stuffing:

   This security mechanism is used to add some extra bits into data which is being transmitted. It helps data to be checked at the receiving end and is achieved by even parity or Odd Parity.

6. Digital Signature:

   This security mechanism is achieved by adding digital data that is not visible to eyes. It is a form of electronic signature which is added by sender which is checked by receiver electronically. This mechanism is used to preserve data which is not more confidential but the sender's identity is to be notified.

A Model for Network Security

When we send our data from the source side to the destination side we have to use some transfer method like the internet or any other communication channel by which we are able to send our message. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. When the transfer of data happens from one source to another source some logical information channel is established between them by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals. When we use the protocol for this logical information channel the main aspect of security has come. who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

1. A security-related transformation on the information to be sent.
2. Some secret information is shared by the two principals and, it is hoped, unknown to the opponent.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission. This model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation.
2. Generate the secret information to be used with the algorithm.

3. Develop methods for the distribution and sharing of secret information.
4. Specify a protocol to be used by the two principles that make use of the security algorithm and the secret information to achieve a particular security service.

e-PGPathshala

Subject : Computer Science

Paper: Cryptography and Network Security

Module: Data Encryption Standard

Module No: CS/CNS/17

Quadrant 1 – e-text

Cryptography and Network Security

Module 17- Data Encryption Standard

Learning Objectives

- To know the design of DES.
- To learn about encryption and decryption in DES.
- To learn about strength of DES.
- To understand differential and linear cryptanalysis.

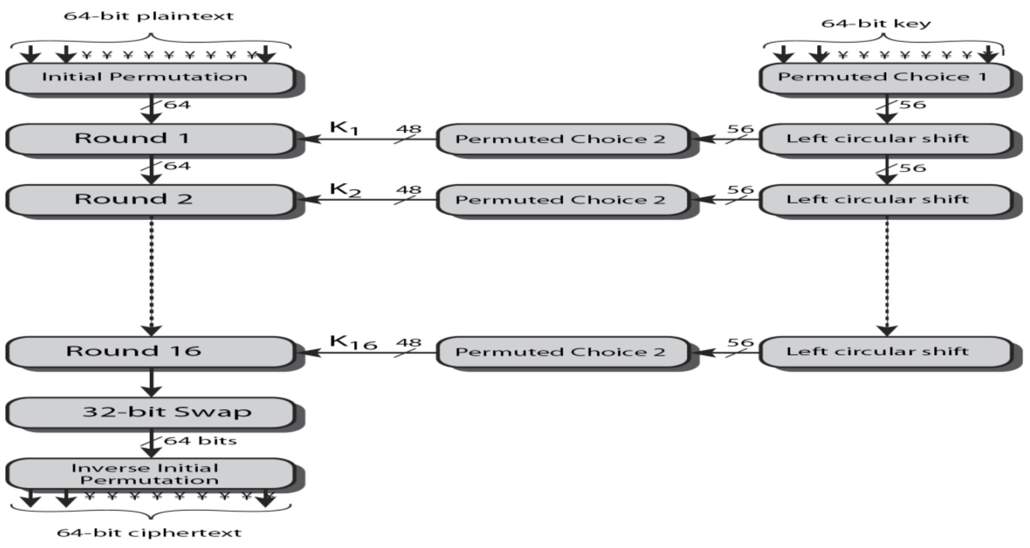
1. INTRODUCTION

The Data Encryption Standard (DES) was jointly developed in 1974 by IBM and the U.S. government to set a standard that everyone could use to securely communicate with each other. It operates on blocks of 64 bits using a secret key that is 56 bits long. The original proposal used a secret key that was 64 bits long. It is widely believed that the removal of these 8 bits from the key was done to make it possible for U.S. government agencies to secretly crack messages.

DES started out as the "Lucifer" algorithm developed by IBM. The US National Security Agency (NSA) made several modifications, after which it was adopted as Federal Information Processing Standard (FIPS) standard 46-3 and ANSI standard X3.92.

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a feistel network). As with most encryption schemes, DES expects two inputs - the plaintext to be encrypted and the secret key. The manner in which the plaintext is accepted, and the key arrangement used for encryption and decryption, both determine the type of cipher it is. DES is therefore a symmetric, 64 bit block cipher as it uses the same key for both encryption and decryption and only operates on 64 bit blocks of data at a time5 (be they plaintext or ciphertext). The key size used is 56 bits, however a 64 bit (or eight-byte) key is actually input. The least significant bit of each byte is either used for parity (odd for DES) or set arbitrarily and does not increase the security in any way. All blocks are numbered from left to right which makes the eight bit of each byte the parity bit.

1.1 DES Encryption Overview



The above figure shows the work flow during a DES encryption. An initial permutation on 64 bit plaintext block is performed at the first step in sequence of flow. Then plaintext block is spilted into two 32 bit sub blocks. After that, blocks undergo 16 rounds of identical operations. The 32 bit left half and 32 bit right half are swapped at the end of 16 th round.Then it is permuted using a inverse initial permutation function. The output of this final permutation is 64 bit ciphertext.

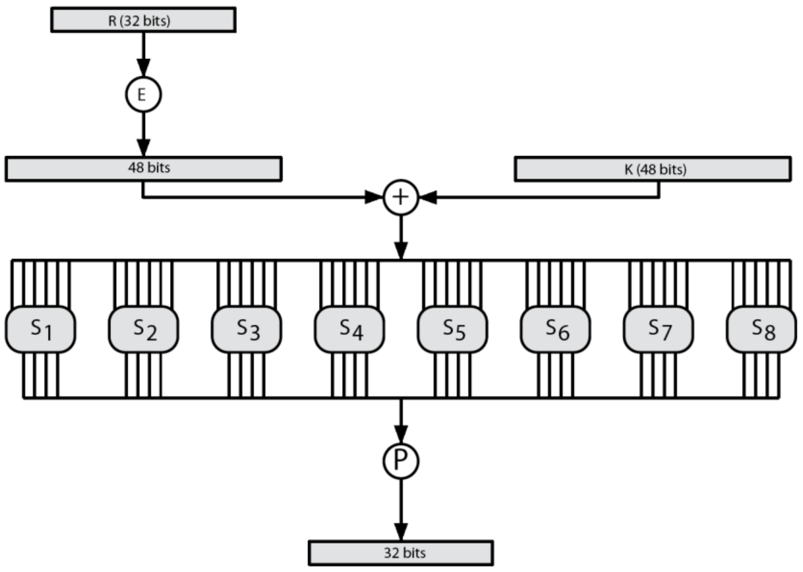
1.1.1 Initial Permutation IP

IP is first step of the data computation .IP reorders the input data bits even bits to left half(LH), odd bits to 32 bit right half (RH). It has quite regular in structure so that easy to implement in h/w.

For example:

IP(675a6967 5e5a6b5a) = (ffb2194d 004df6fb)

1.1.2 DES Round Structure



Let’s see what is happening in each round.32 bit RH bits are given to expansion function E. The output of E is 48 bit. It is XORed with 48 bit key and output is given to 8 S boxes. Here Each S box has 6 bit input and it produces 4 bit output. Then outputs from all S boxes are combined and given to input of permutation P Box. Hence 32 bit permuted bits are the output of each round.

1.1.3 Substitution Boxes S

DES uses eight S-boxes which map 6 to 4 bits.Each S-box is actually 4 little 4 bit boxes .Outer bits 1 & 6 (**row** bits) select one row of 4.Inner bits 2-5 (**col** bits) are substituted .Result is 8 lots of 4 bits, or 32 bits.Row selection depends on both data & key. Feature known as autoclaving (autokeying) is applied here.

For example:

S(18 09 12 3d 11 17 38 39) = 5fd25e03

2. DES Key Schedule

Subkeys should be computed in each round of DES. Initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves .

16 tages consisting of (i)Rotating **each half** separately either 1 or 2 places depending on the **key rotation schedule** K (ii) Selecting 24-bits from each half & permuting them by PC2 for use in round function F

3. DES Decryption

Decrypt must unwind steps of data computation .With Feistel design, do encryption steps again using subkeys in reverse order (SK16 … SK1).IP undoes final FP step of encryption First round with SK16 undoes 16th encrypt round till 16th round with SK1 undoes 1st encrypt round then final FP undoes initial encryption IP thus recovering original data value

4. Avalanche Effect

One of the key desirable properties of encryption algorithm is Avalance effect, where a change of one input or key bit results in changing approx half output bits. Making attempts to “home-in” by guessing keys impossible in DES.DES exhibits strong avalanche.

5. Strength of DES

5.1 Key Size

DES uses 56-bit keys, which have $2^{56} = 7.2 \times 10^{16}$ values. Even though it is hard to do brute force search ,recent advances have shown is possible. They are listed below:

- in 1997 on Internet in a few months
- in 1998 on dedicated h/w (EFF) in a few days
- in 1999 above combined in 22hrs!

So alternatives to DES has to be considered.

5.2 Analytic Attacks

Several analytic attacks on DES are possible. These analytic attacks utilise some deep structure of the cipher .IT can be done by gathering information about encryptions or can eventually recover some/all of the sub-key bits or if necessary then exhaustively search for the rest .Generally these are statistical attacks include differential cryptanalysis ,linear cryptanalysis and related key attacks

5.3 Timing Attacks

Timing attacks done at actual implementation of cipher. It uses knowledge of consequences of implementation to derive information about some/all subkey bits, specifically use fact that calculations can take varying times depending on the value of the inputs to it. It is particularly problematic on smartcards.

using a large number of trial encryptions . The effectiveness given by: $|p-{}^{1}/_{2}|$

6 DES Design Criteria

DES design criteria is reported bt Coppersmith. There are 7 criteria for S-boxes provide for non-linearity, resistance to differential cryptanalysis and good confusion and 3 criteria for permutation P provide for increased diffusion.

SUMMARY

We have considered Feistel cipher design & structure DES details to strength the Differential & Linear Cryptanalysis.

e-PGPathshala

Subject : Computer Science

Paper: Cryptography and Network Security

Module: Advanced Encryption Standard(part1),

Module No: CS/CNS/19

Quadrant 1 – e-text

AES (Part 1)

Availability of advanced computing systems make ease to break the ciphers that we have discussed so far. A replacement of DES was needed because of small key size. Triple DES is too slow because we have to run 48 rounds effectively. So it is not a good solution. The Advanced Encryption Standard (AES) is a successor of many algorithms which are later proved to be vulnerable. The AES algorithm is a symmetric encryption algorithm which uses a single key for both encryption and decryption process. AES is also the official encryption used by the government of United States of America and Canada. It is used for both encryption of data in transit (data uploading and downloading) and for data at rest (Data in Hard Drive and other storage devices). Though they will use a 256 bit encryption key to encrypt the data.

In 1997, National Institute for Standards and Technology send out for an open call for ciphers. Private key symmetric block cipher ,128-bit data, 128/192/256-bit keys ,Stronger & faster than Triple-DES ,Provide full specification & design details ,Both C and Java implementations were NIST's requirements for the AES candidate submissions. In fact, two set of criteria evolved. When NIST issued its original request for candidate algorithm nominations in 1997, the request stated that candidate algorithms would be compared based on the factors shown in Stallings Table5.1, which were used to evaluate field of 15 candidates to select shortlist of 5. These had categories of security, cost, and algorithm & implementation characteristics. The final criteria evolved during the evaluation process, and were used to select Rijndael from that short-list and different categories of: general security, ease of software & hardware implementation, implementation attacks, & flexibility (in en/decrypt, keying, other factors).

The AES shortlist of 5 ciphers as:

- ❑ MARS (IBM) - complex, fast, high security margin
- ❑ RC6 (USA) - v. simple, v. fast, low security margin
- ❑ Rijndael (Belgium) - clean, fast, good security margin
- ❑ Serpent (Euro) - slow, clean, v. high security margin
- ❑ Twofish (USA) - complex, v. fast, high security margin

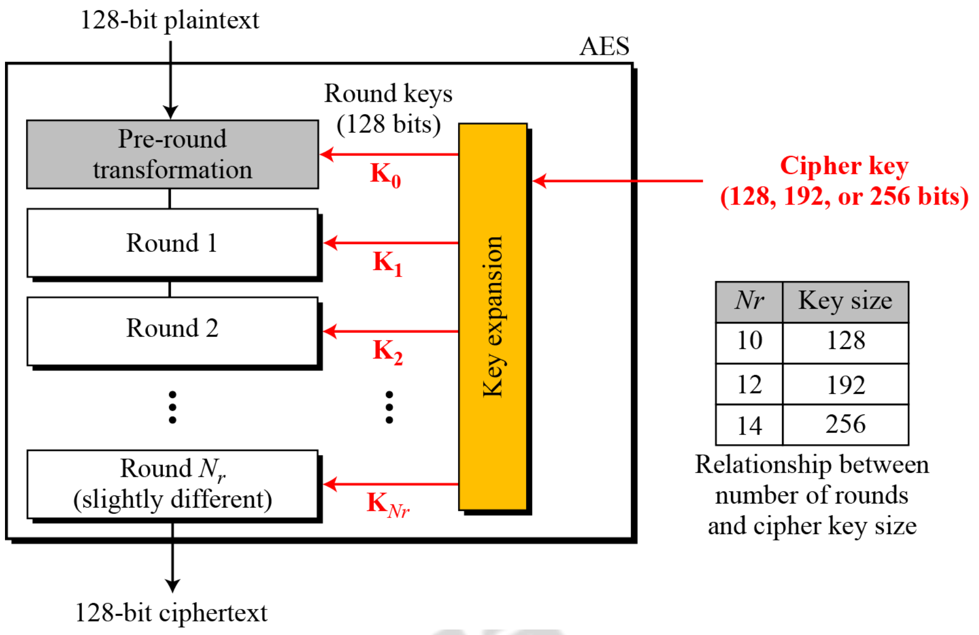
Note mix of commercial (MARS, RC6, Twofish) verses academic (Rijndael, Serpent) proposals, sourced from various countries.

All were thought to be good – it came down to the best balance of attributes to meet criteria, in particular the balance between speed, security & flexibility.

Rijndael was selected as the AES in Oct-2000. It was designed by Vincent Rijmen and Joan Daemen in Belgium and issued as FIPS PUB 197 standard in Nov-2001 .AES isaAn **iterative** rather than **Feistel** cipher.ie, processes data as block of 4 columns of 4 bytes (128 bits) and operates on entire data block in every round .

Rounds in AES

- Rounds are (almost) identical
    - ❑ First and last round are a little different



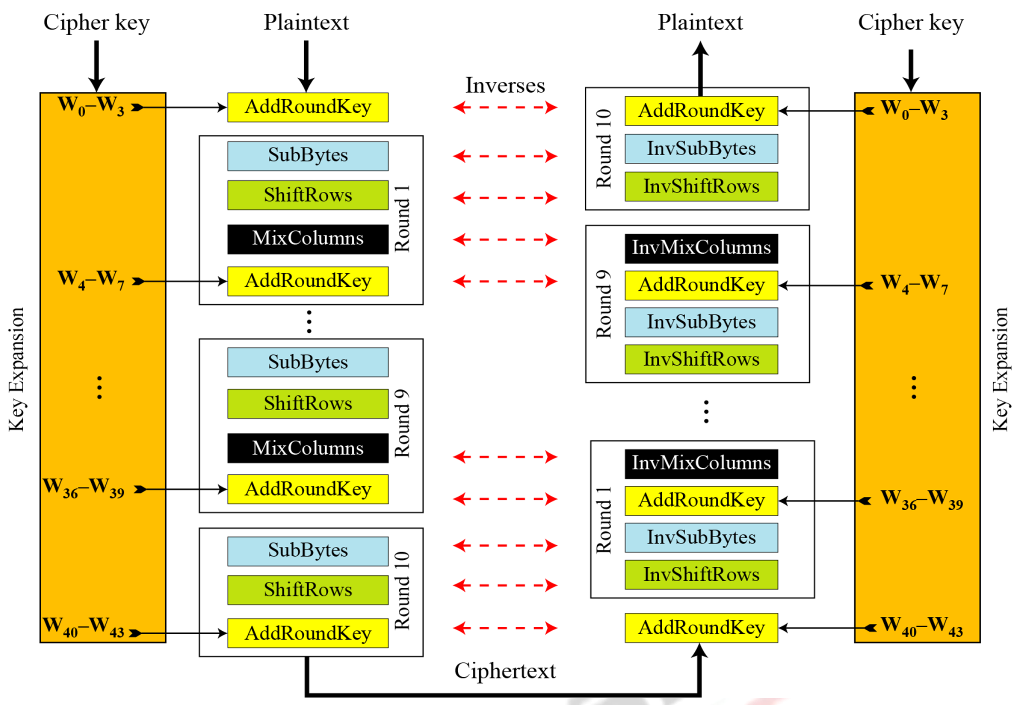
Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

The input to the AES encryption and decryption algorithms is a single 128-bit block, depicted in FIPS PUB 197, as a square matrix of bytes .This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output.

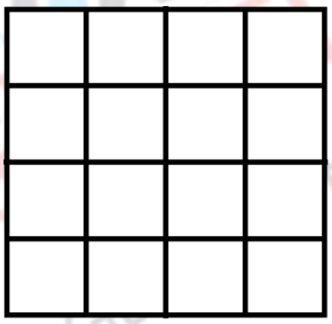
The key is expanded into 44/52/60 lots of 32-bit words (see later), with 4 used in each round.

The data computation then consists of an "add round key" step, then 9/11/13 rounds with all 4 steps, and a final 10th/12th/14th step of byte subs + mix cols + add round key. This can be viewed as alternating XOR key & scramble data bytes operations. All of the steps are easily reversed, and can be efficiently implemented using XOR's & table lookups.



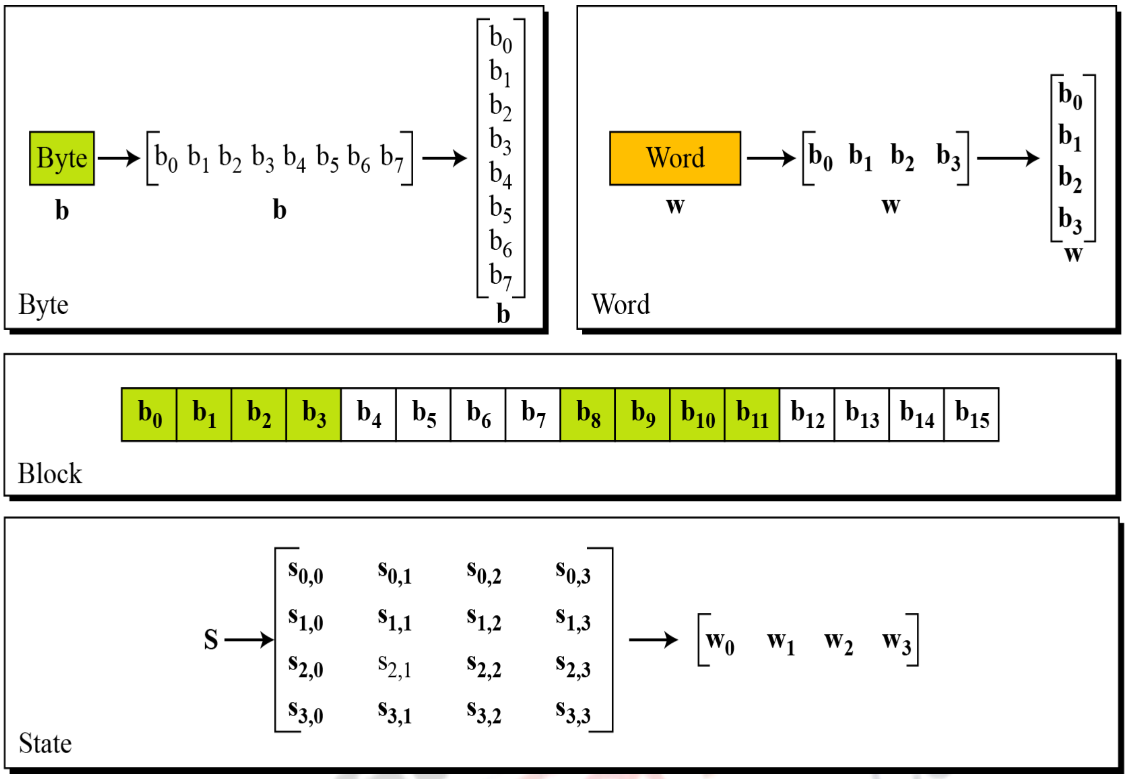
The above figure shows the overall structure of AES.

128-bit values

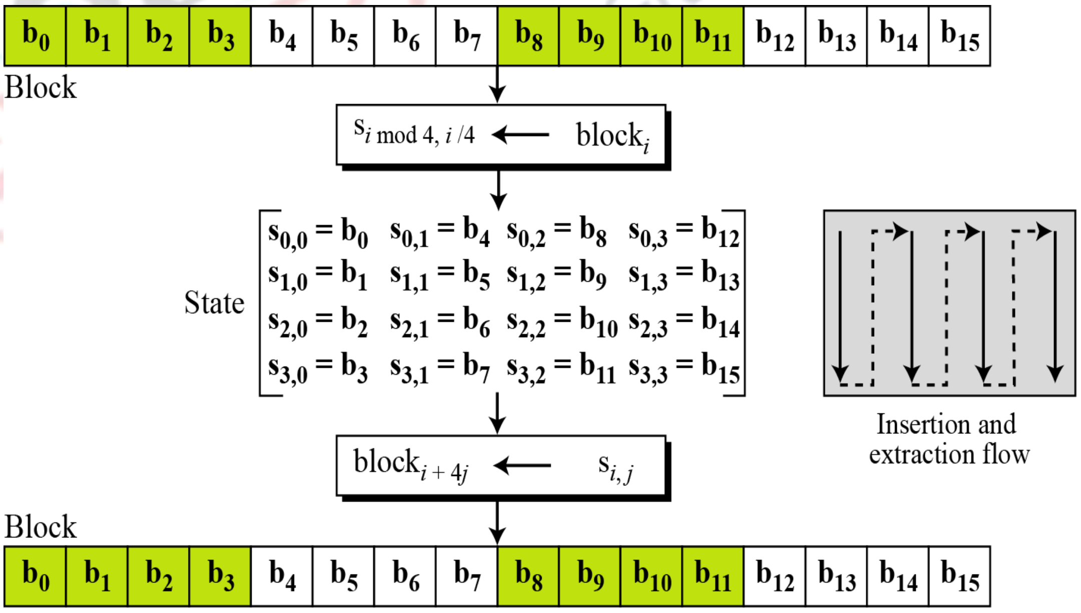


Data block viewed as 4-by-4 table of bytes and it is represented as 4 by 4 matrix of 8-bit bytes. Key is expanded to array of 32 bits words

- Data Unit



The above figure shows the data unit. Block to state transformation is done as shown in figure below.


Block
$s_{i \bmod 4,\, i/4} \leftarrow \text{block}_i$
State
$\text{block}_{i+4j} \leftarrow s_{i,j}$
Block
Insertion and extraction flow

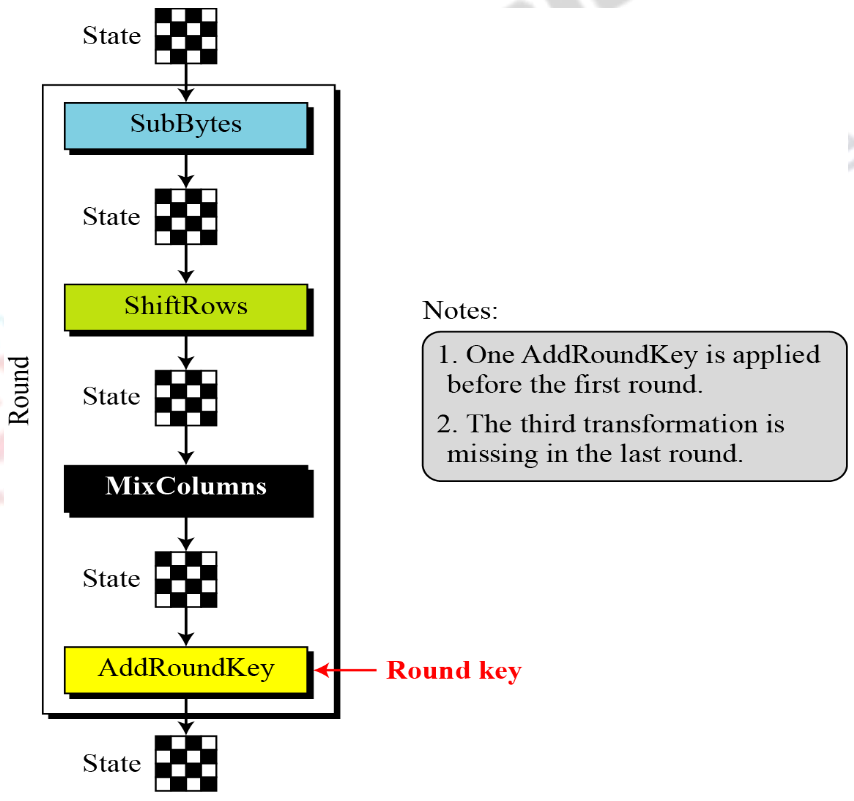
Now we are going see how the plaintext is converted to state.

Text	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Hexadecimal	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \text{State}$$

- Details of Each Round.

Each round consists of four operations namely SubBytes, ShiftRows , MixColumns, Add Round key as shown in figure.



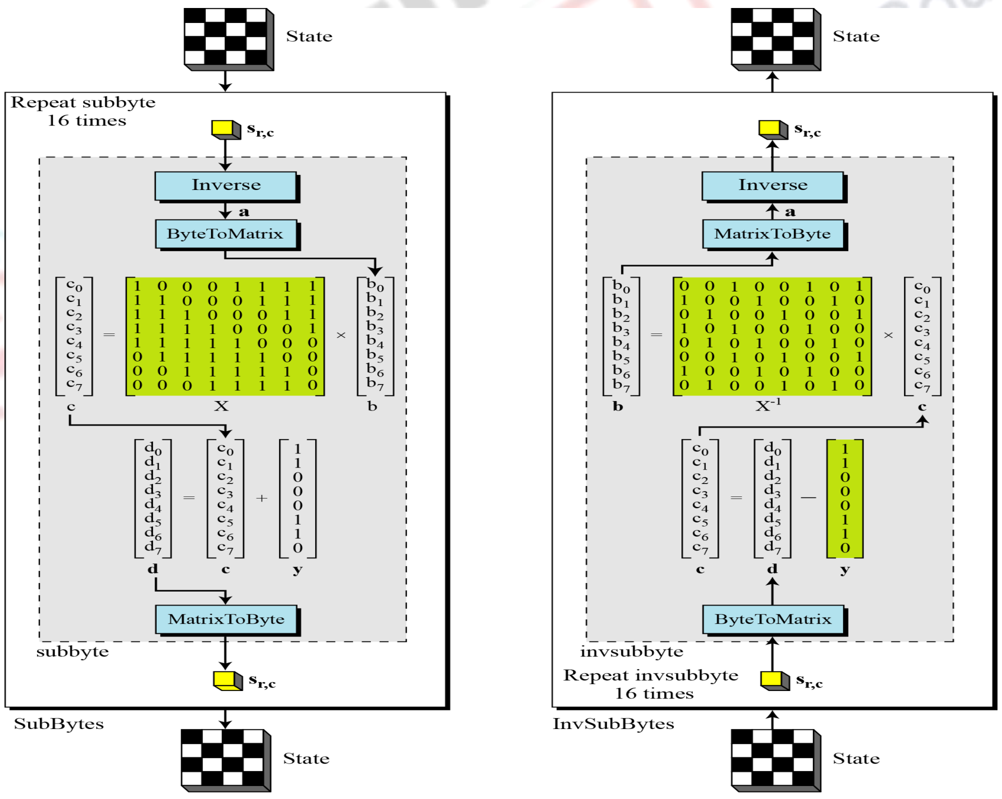
1. One AddRoundKey is applied before the first round.
2. The third transformation is missing in the last round.

Now discuss each of the four stages used in AES. The Substitute bytes stage uses an S-box to perform a byte-by-byte substitution of the block. There is a single 8-bit wide S-box used on every byte. This S-box is a permutation of all 256 8-bit values, constructed using a transformation which treats the values as polynomials in GF(2^8) – however it is fixed, so really only need to know the table when implementing. Decryption requires the inverse of the table. These tables are given in Stallings Table 4.5.

The table was designed to be resistant to known cryptanalytic attacks. Specifically, the Rijndael developers sought a design that has a low correlation between input bits and output bits, with the property that the output cannot be described as a simple mathematical function of the input, with no fixed points and no “opposite fixed points”.

- **SubBytes: Byte Substitution**

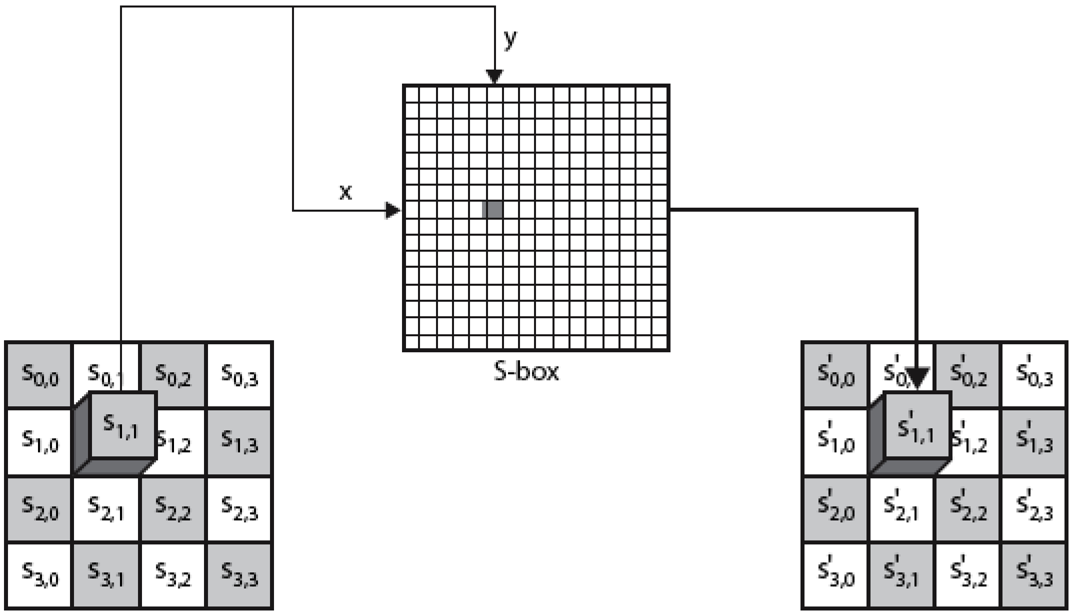
The SubBytes and InvSubBytes transformations are inverses of each other.



A simple substitution of each byte provide a confusion. Uses one S-box of 16x16 bytes containing a permutation of all 256 8-bit values. Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits).

For eg. byte {95} is replaced by byte in row 9 column 5 which has value {2A}

The SubBytes operation involves 16 independent byte-to-byte transformations.

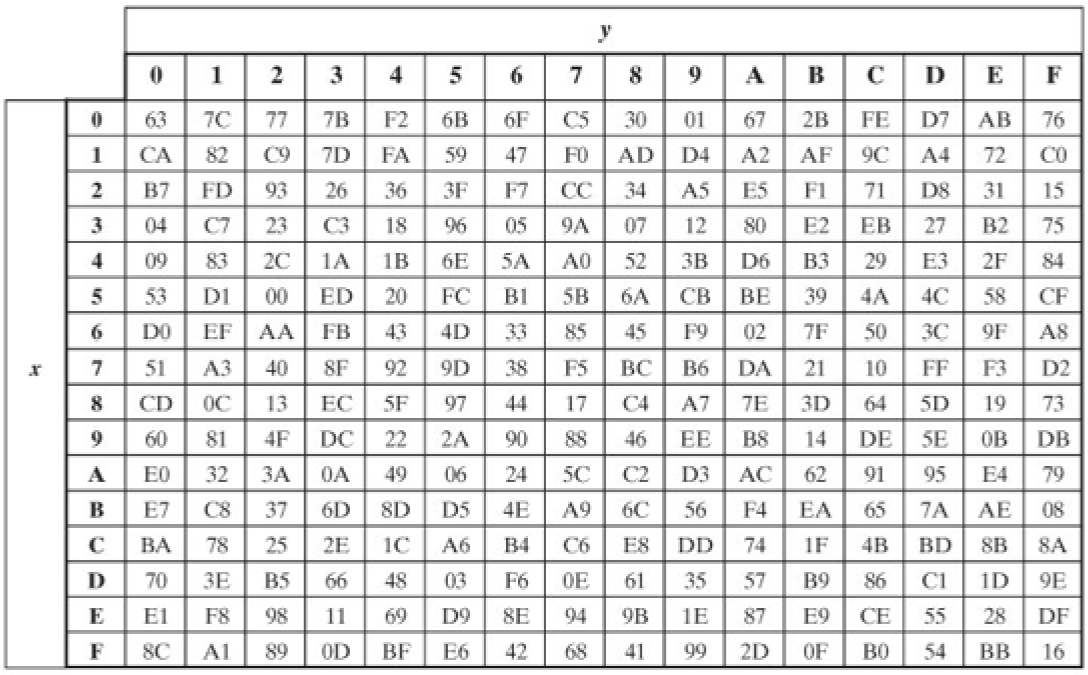


y
x
S-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Interpret the byte as two hexadecimal digits *xy* .Software implementation, use row (*x*) and column (*y*) as lookup pointer.

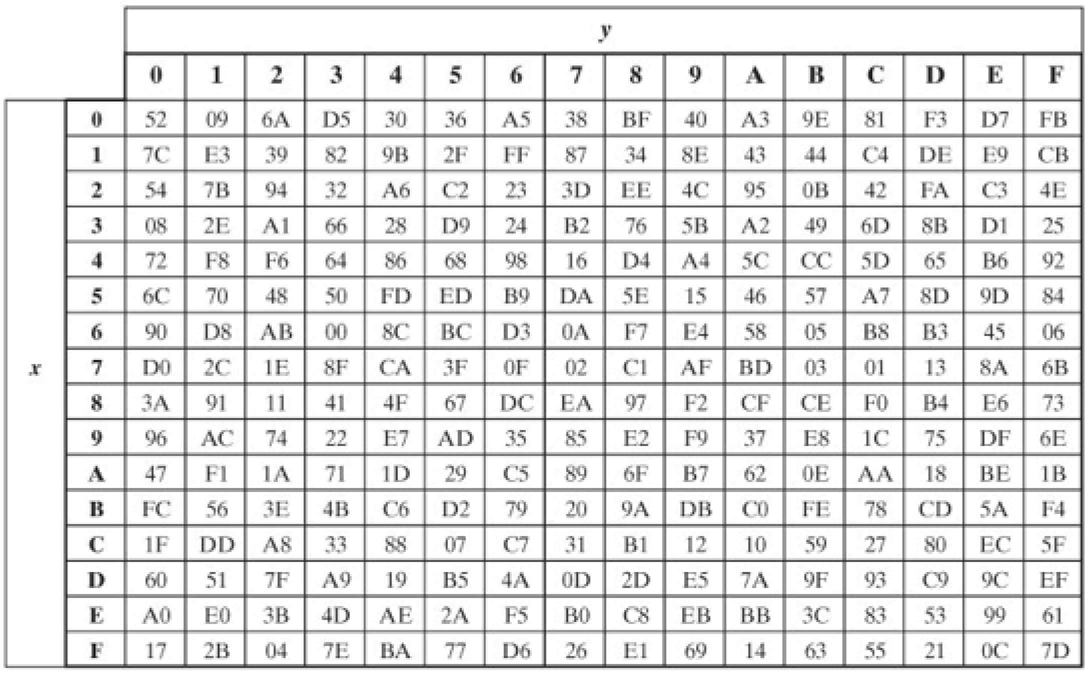
Ie, $S_{1,1} = xy_{16}$

SubByte table is implements by table lookup as shown below.



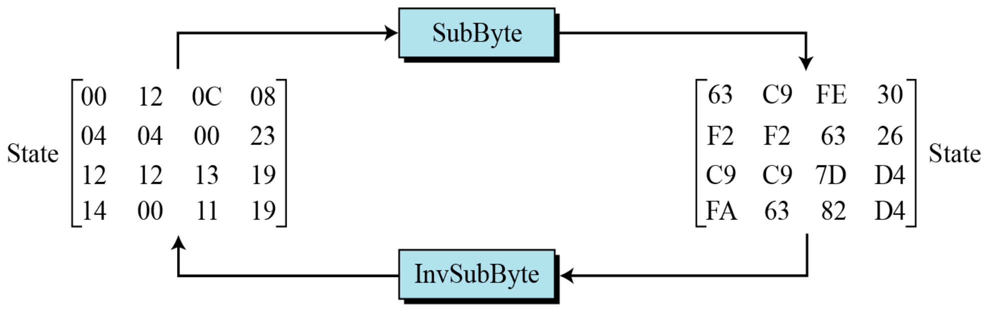
		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
x	**7**	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

The InvSubByte table is:



		y															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
x	**7**	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

The following gives a sample of SubByte and InvSubByte operations.



Summary

We studied:

- the AES selection process
- the details of Rijndael – the AES cipher
- looked at the steps in each round
- the key expansion
- implementation aspects

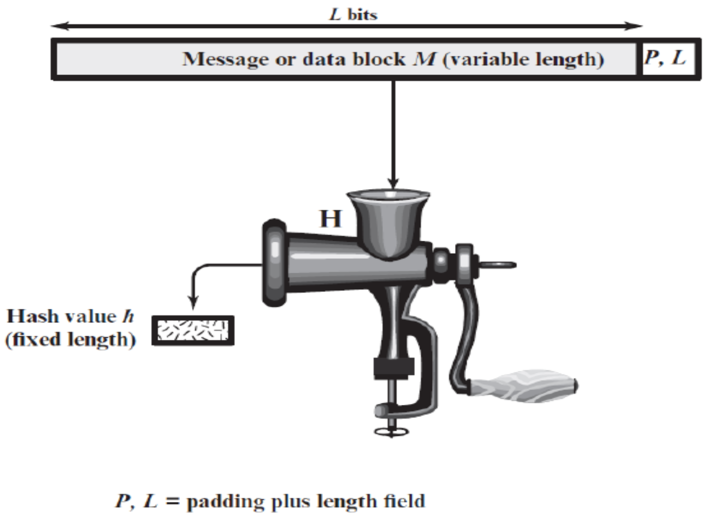
CRYPTOGRAPHY

UNIT 4

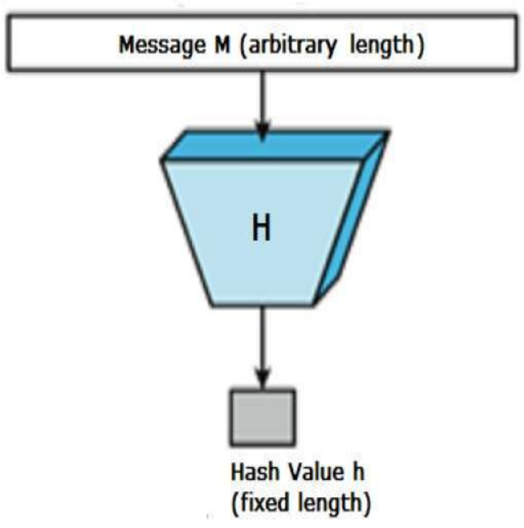
HASH FUNCTIONS AND DIGITAL SIGNATURES: Cryptographic Hash Functions – Application of Hash Functions – Two Simple Hash Functions – Secure Hash Algorithm(SHA) –Message Authentication Codes – Authentication requirement – Authentication function – MAC – HMAC – CMAC – Digital signature and authentication protocols – Digital Signature Standards –Digital Signatures Schemes– Digital Certificate – Key Management and Distribution.

4.1 Cryptographic Hash Functions:

- ✓ A **hash function** H accepts a variable-length block of data *M* as input and produces a fixed-size hash value *h* = H(*M*).
- ✓ A “good” hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in *M* results, with high probability, in a change to the hash value.
- ✓ The kind of hash function needed for security applications is referred to as a **cryptographic hash function**.
- ✓ A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either
  - o A data object that maps to a pre-specified hash result (the one-way property) or
  - o Two data objects that map to the same hash result (the collision-free property).
- ✓ Because of these characteristics, hash functions are often used to determine whether or not data has changed.


L bits
Message or data block M (variable length)
P, L
H
Hash value h (fixed length)
P, L = padding plus length field

Cryptographic Hash function h=H(M)



The above figure depicts the general operation of a cryptographic hash function.

- ❖ Typically, the input is padded out to an integer multiple of some fixed length (e.g., 1024 bits), and the padding includes the value of the length of the original message in bits.

- The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.

Features of Hash Functions

The typical features of hash functions are −

- **Fixed Length Output (Hash Value)**
  - Hash function coverts data of arbitrary length to a fixed length. This process is often referred to as **hashing the data**.
  - In general, the hash is much smaller than the input data, hence hash functions are sometimes called **compression functions**.
  - Since a hash is a smaller representation of a larger data, it is also referred to as a **digest**.
  - Hash function with n bit output is referred to as an **n-bit hash function**. Popular hash functions generate values between 160 and 512 bits.
- **Efficiency of Operation**
  - Generally for any hash function h with input x, computation of h(x) is a fast operation.
  - Computationally hash functions are much faster than a symmetric encryption.

Properties of Hash Functions

In order to be an effective cryptographic tool, the hash function is desired to possess following properties −

- **Pre-Image Resistance**
  - This property means that it should be computationally hard to reverse a hash function.
  - In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.
  - This property protects against an attacker who only has a hash value and is trying to find the input.
- **Second Pre-Image Resistance**
  - This property means given an input and its hash, it should be hard to find a different input with the same hash.
  - In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.
- **Collision Resistance**
  - This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
  - In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.
  - Also, if a hash function is collision-resistant **then it is second pre-image resistant.**

4.2 Application of Hash Functions

The most versatile cryptographic algorithm is the cryptographic hash function. It is used in a wide variety of security applications and Internet protocols. The following are various applications where it is employed.

4.2.1 Message Authentication:

Message authentication is a mechanism or service used to verify the integrity of a message.

Message authentication assures that data received are exactly as sent (i.e., there is no modification, insertion, deletion, or replay).

When a hash function is used to provide message authentication, the hash function value is often referred to as a **message digest**.

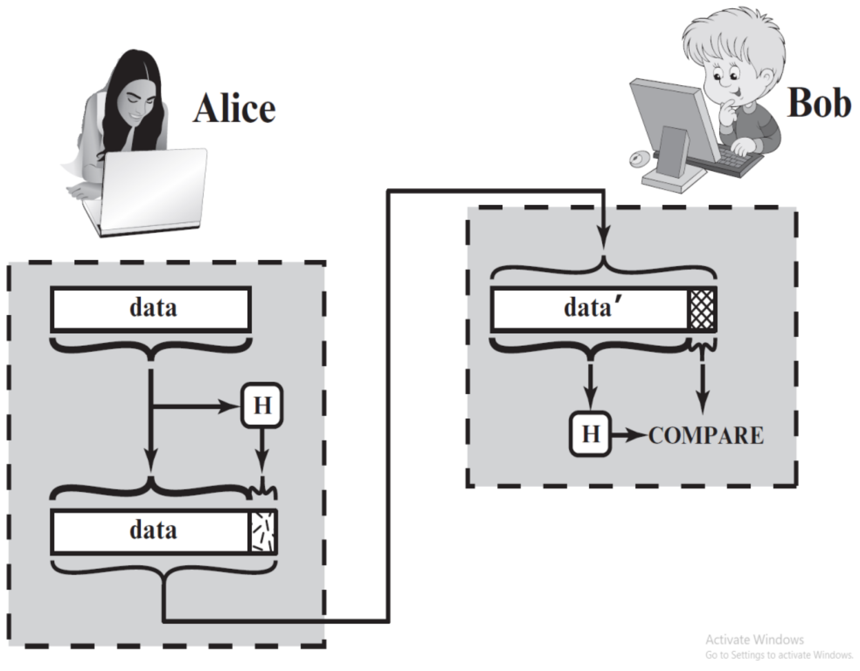
- ❖ The essence of the use of a hash function for message integrity is as follows.

o The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message.

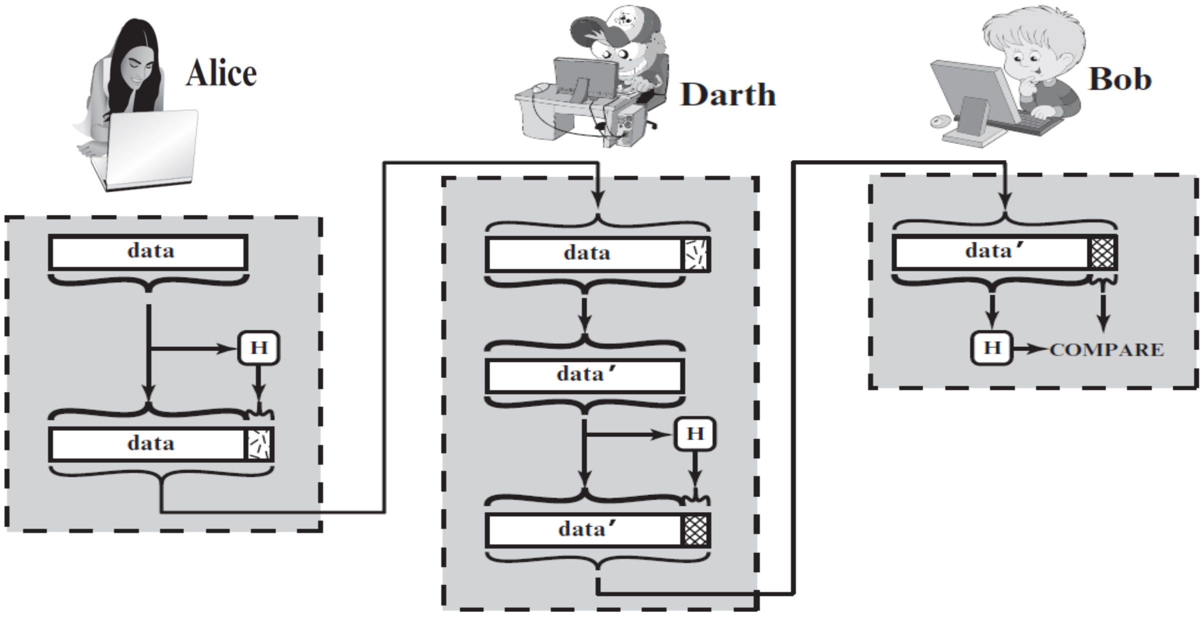
o The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value.

o If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered (Figure a).

o The hash value must be transmitted in a secure fashion. That is, the hash value must be protected so that if an adversary alters or replaces the message, it is not feasible for adversary to also alter the hash value to fool the receiver. This type of attack is shown in Figure b.



(a) Use of hash function to check data integrity



(b) Man-in-the-middle attack

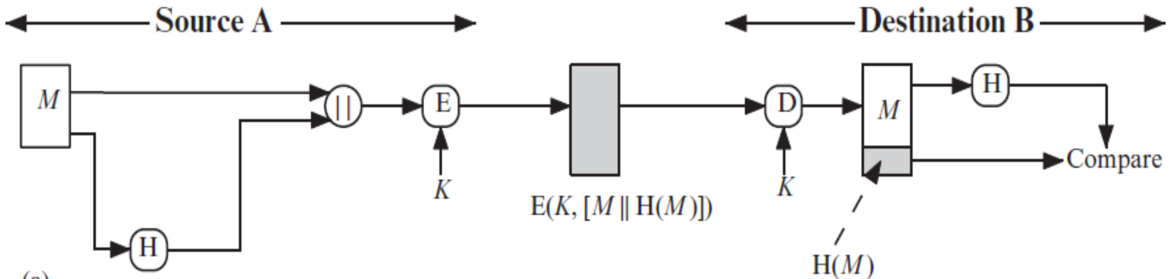
The following are a variety of ways in which a hash code can be used to provide message authentication.

a. The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

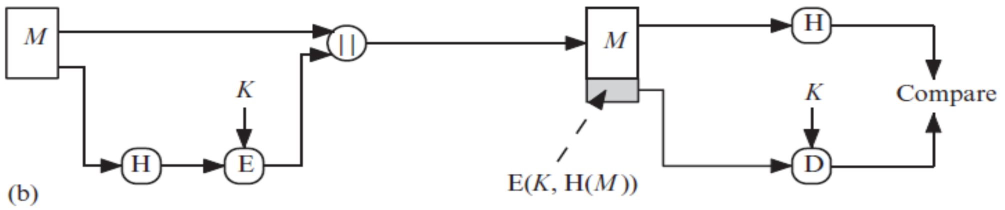
b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value *S*. A computes the hash value over the concatenation of *M* and *S* and appends the resulting hash value to *M*. Because B possesses *S*, it can re-compute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

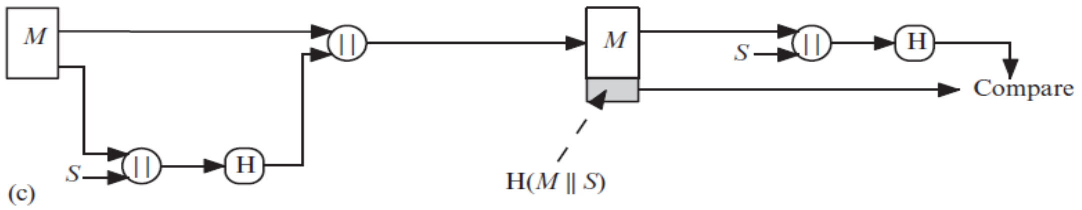
d. Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.



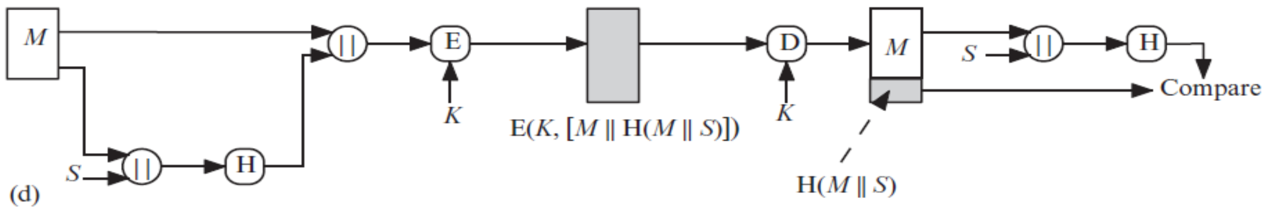
(a)



(b)



(c)



(d)

- More commonly, message authentication is achieved using a **message authentication code (MAC)**, also known as a **keyed hash function**.
- Typically, MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.
- A MAC function takes as input a secret key and a data block and produces a hash value, referred to as the MAC, which is associated with the protected message.
- If the integrity of the message needs to be checked, the MAC function can be applied to the message and the result compared with the associated MAC value.
- An attacker who alters the message will be unable to alter the associated MAC value without knowledge of the secret key.

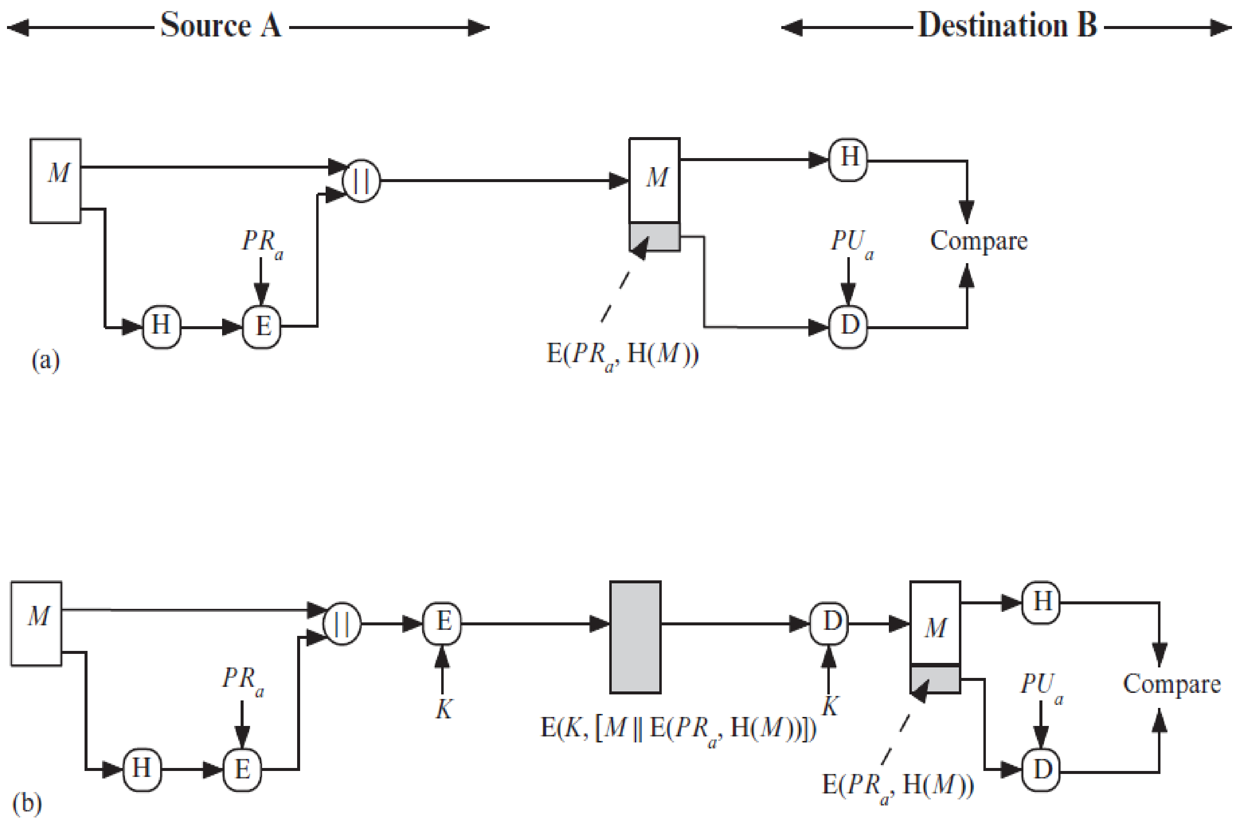
4.2.2 Digital Signatures:

- ✓ Another important application, which is similar to the message authentication application, is the **digital signature**.
- ✓ The operation of the digital signature is similar to that of the MAC.
- ✓ In the case of the digital signature, the hash value of a message is encrypted with a user's private key.
- ✓ Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.
- ✓ In this case, an attacker who wishes to alter the message would need to know the user's private key.

- ✓ Following figures illustrates, in a simplified fashion, how a hash code is used to provide a digital signature.

a. The hash code is encrypted, using public-key encryption with the sender's private key. As with Figure b, this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

b. If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.



4.2.3 Other Applications:

- ❖ Hash functions are commonly used to create a **one-way password file**.
- ❖ Hash functions can be used for **intrusion detection** and **virus detection**.
- ❖ A cryptographic hash function can be used to construct a **pseudorandom function (PRF)** or a **pseudorandom number generator (PRNG)**.

4.3 Two-Simple Hash Functions:

- ❖ To get the understanding of security considerations involved in cryptographic hash functions, we present two simple, insecure hash functions in this section.
- ❖ All hash functions operate using the following general principles.
  - ✓ The input (message, file, etc.) is viewed as a sequence of *n* -bit blocks.
  - ✓ The input is processed one block at a time in an iterative fashion to produce an *n*-bit hash function.
- ❖ One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as:

$$C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

where

C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

This operation produces a simple parity bit for each bit position and is known as a longitudinal redundancy check.

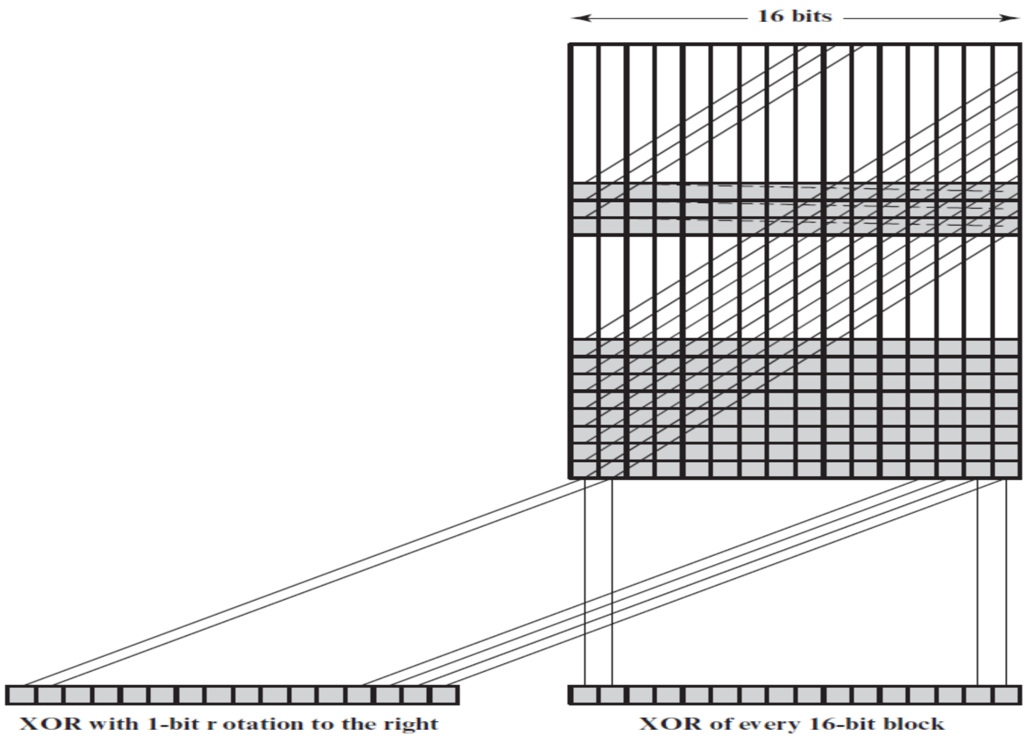
- It is reasonably effective for random data as a data integrity check. Each *n*-bit hash value is equally likely.
- Thus, the probability that a data error will result in an unchanged hash value is 2-*n*.
- With more predictably formatted data, the function is less effective.
- For example, in most normal text files, the high-order bit of each octet is always zero.
- So if a 128-bit hash value is used, instead of an effectiveness of 2-128, the hash function on this type of data has an effectiveness of 2-112.
- A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows.

  1. Initially set the *n*-bit hash value to zero.

  2. Process each successive *n*-bit block of data as follows:

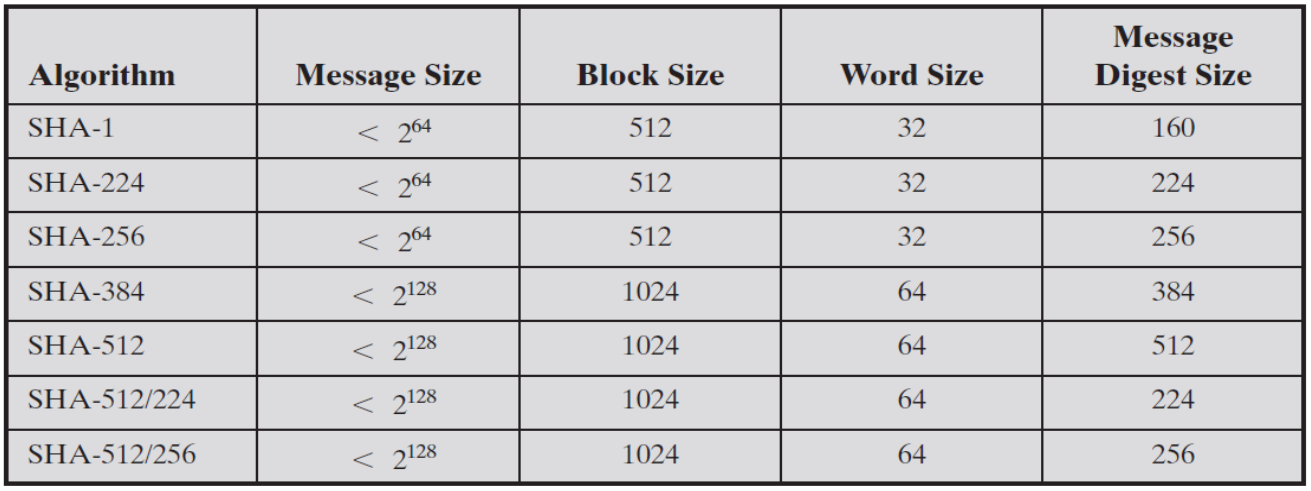
  a. Rotate the current hash value to the left by one bit.

  b. XOR the block into the hash value.
- This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input.
- Although the second procedure provides a good measure of data integrity, it is virtually useless for data security when an encrypted hash code is used with a plaintext message.
- Although a simple XOR or rotated XOR (RXOR) is insufficient if only the hash code is encrypted, you may still feel that such a simple function could be useful when the message together with the hash code is encryp


16 bits
XOR with 1-bit r otation to the right
XOR of every 16-bit block

4.4 Secure Hash Algorithm(SHA)

In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA). Indeed, because virtually every other widely used hash function had been found to have substantial cryptanalytic weaknesses, SHA was more or less the last remaining standardized hash algorithm by 2005. SHA was developed



Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

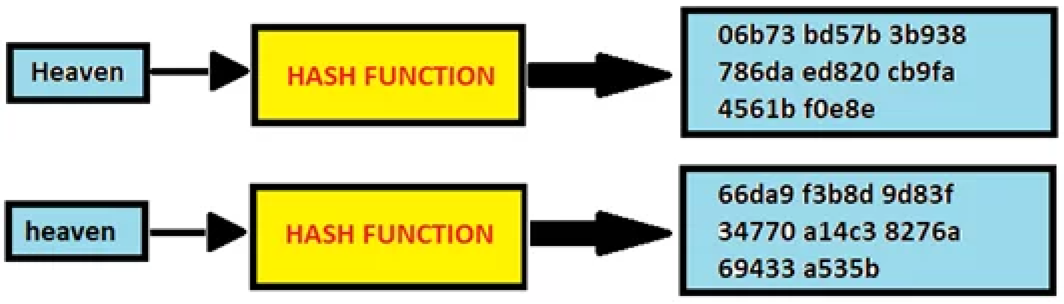
by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993. When weaknesses were discovered in SHA, now known as **SHA-0**, a revised version was issued as FIPS 180-1 in 1995 and is referred to as **SHA-1**. The actual standards document is entitled "Secure Hash Standard." SHA is based on the hash function MD4, and its design closely models MD4.

SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively. Collectively, these hash algorithms are known as **SHA-2**. These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. A revised document was issued as FIP PUB 180-3 in 2008, which added a 224-bit version.

In 2015, NIST issued FIPS 180-4, which added two additional algorithms: SHA-512/224 and SHA-512/256. SHA-1 and SHA-2 are also specified in RFC 6234, which essentially duplicates the material in FIPS 180-3 but adds a C code implementation.

In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on SHA-2 by 2010. Shortly thereafter, a research team described an attack in which two separate messages could be found that deliver the same SHA-1 hash using 269 operations, far fewer than the 280 operations previously thought needed to find a collision with an SHA-1 hash. This result should hasten the transition to SHA-2. In this section, we provide a description of SHA-512. The other versions are quite similar.

- ✓ SHA is a modified version of MD5 and used for hashing data and certificates. A hashing algorithm shortens the input data into a smaller form that cannot be understood by using bitwise operations, modular additions, and compression functions. You may be wondering, can hashing be cracked or decrypted? Hashing is similar to encryption, the only difference between hashing and encryption is that hashing is one-way, meaning once the data is hashed, the resulting hash digest cannot be cracked, unless a brute force attack is used. See the image below for the working of SHA algorithm. SHA works in such a way even if a single character of the message changed, then it will generate a different hash. For example, hashing of two similar, but different messages i.e., Heaven and heaven is different. However, there is only a difference of a capital and small letter.



The initial message is hashed with SHA-1, resulting in the hash digest

“06b73bd57b3b938786daed820cb9fa4561bf0e8e”. If the second, similar, message is hashed with SHA-1, the hash digest will look like “66da9f3b8d9d83f34770a14c38276a69433a535b”. This is referred to as the avalanche effect. This effect is important in cryptography, as it means even the slightest change in the input message completely changes the output. This will stop attackers from being able to understand what the hash digest originally said and telling the receiver of the message whether or not the message has been changed while in transit.

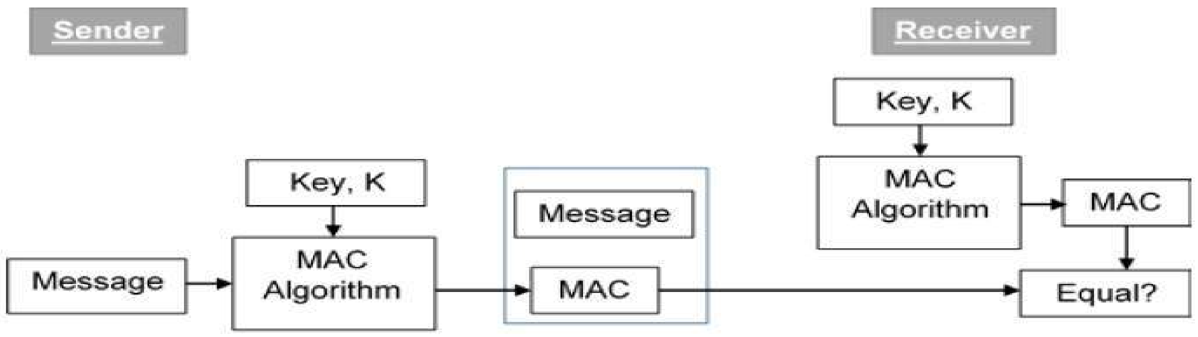
SHAs also assist in revealing if an original message was changed in any way. By referencing the original hash digest, a user can tell if even a single letter has been changed, as the hash digests will be completely different. One of the most important parts of SHAs are that they are deterministic. This means that as long as the hash function used is known, any computer or user can recreate the hash digest. The determinism of SHAs is one of reasons every SSL certificate on the Internet is required to have been hashed with a SHA-2 function.

4.5 Message Authentication Codes:

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.

4.6 Message Authentication Requirements:

In the context of communications across a network, the following attacks can be identified.

1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3. Masquerade: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or no receipt by someone other than the message recipient.

4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. Timing modification: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.

7. Source repudiation: Denial of transmission of message by source.

8. Destination repudiation: Denial of receipt of message by destination.

4.7 Message Authentication Functions:

- ✓ Any message authentication or digital signature mechanism has two levels of functionality.
- ✓ At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message.
- ✓ This lower-level function is then used as a primitive in a higher-level authentication protocol that enables
- ✓ a receiver to verify the authenticity of a message.

- ✓ We are concerned with the types of functions that may be used to produce an authenticator. These may be grouped into three classes.
- ✓
  - o **Hash function:** A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator
  - o **Message encryption:** The ciphertext of the entire message serves as its authenticator
  - o **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

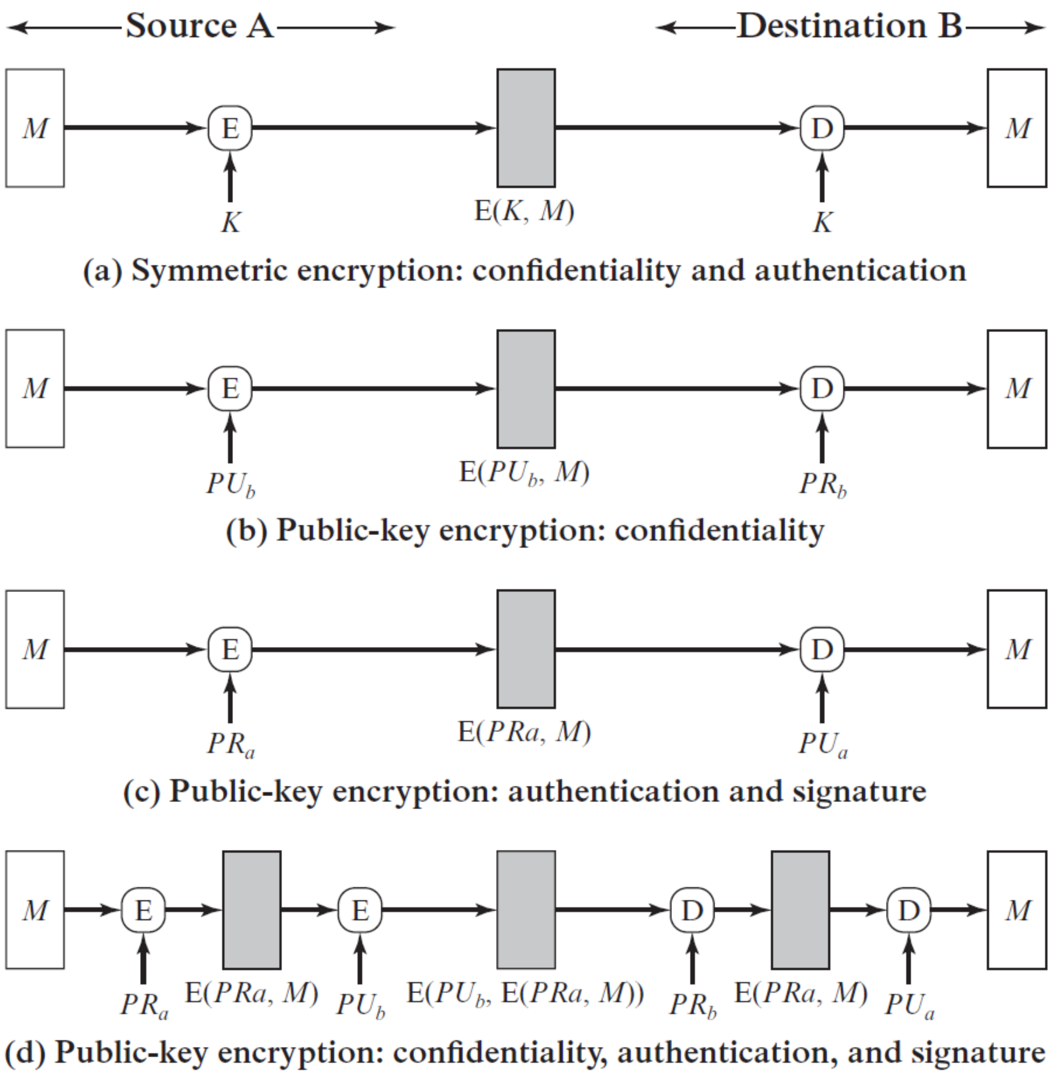
Message Encryption:

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

Symmetric Encryption:

Consider the straightforward use of symmetric encryption (Figure a).

- ❖ A message *M* transmitted from source A to destination B is encrypted using a secret key *K* shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.



(a) Symmetric encryption: confidentiality and authentication

(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature

(d) Public-key encryption: confidentiality, authentication, and signature

Public-Key Encryption:

- The straightforward use of public-key encryption (Figure b) provides confidentiality but not authentication.
- The source (A) uses the public key PU_b of the destination (B) to encrypt M. Because only B has the corresponding private key PR_b, only B can decrypt the message. This scheme provides no authentication, because any opponent could also use B's public key to encrypt a message and claim to be A.
- To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt (Figure c). This provides authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses PR_a and therefore the only party with the information necessary to construct ciphertext that can be decrypted with PU_a.
- There must be some internal structure to the plaintext so that the receiver can distinguish between
- well-formed plaintext and random bits.
- Assuming there is such structure, then the scheme of Figure c does provide authentication. It also provides what is known as digital signature.
- Only A could have constructed the ciphertext because only A possesses PR_a. Not even B, the recipient, could have constructed the ciphertext. Therefore, if B is in possession of the ciphertext, B has the means to prove that the message must have come from A.
- In effect, A has "signed" the message by using its private key to encrypt.
- Note that this scheme does not provide confidentiality. Anyone in possession of A's public key can decrypt the ciphertext.
- To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality (Figure d).
- The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

4.8 HMAC (Hash based MACs)

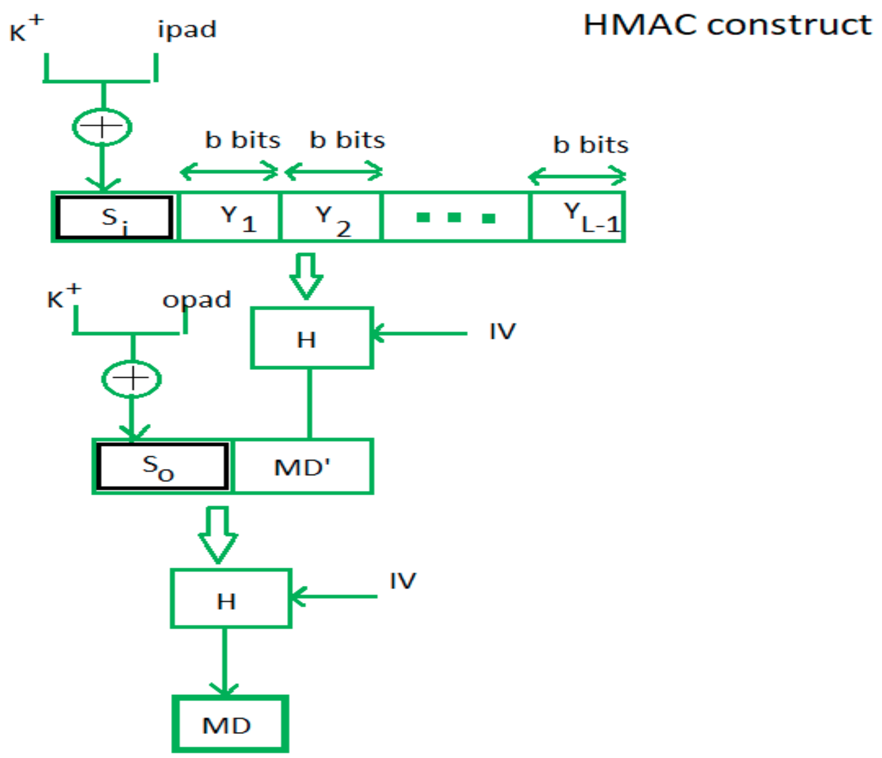
HMAC algorithm stands for Hashed or Hash-based Message Authentication Code. It is a result of work done on developing a MAC derived from cryptographic hash functions. HMAC is a great resistance towards cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC and thus is more secure than any other authentication code. RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security. The FIPS 198 NIST standard has also issued HMAC.

Objectives –

- As the Hash Function, HMAC is also aimed to be one way, i.e, easy to generate output from input but complex the other way round.
- It aims at being less affected by collisions than the hash functions.
- HMAC reuses the algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- HMAC tries to handle the Keys in a more simple manner.

HMAC algorithm: The working of HMAC starts with taking a message M containing blocks of length b bits. An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a temporary message-digest MD'. MD' again is appended to an output signature and the whole is applied a hash function again, the result is our final message digest MD.

Here is a simple structure of HMAC:



Here, H stands for Hashing function,
M is the original message
Si and So are input and output signatures respectively,
Yi is the ith block in original message M, where I ranges from [1, L)
L = the count of blocks in M
K is the secret key used for hashing
IV is an initial vector (some constant)
The generation of input signature and output signature Si and So respectively.

$S_i = K^+ \oplus \text{ipad}$ where K^+ is nothing but K padded with zeros on the left so that the result is b bits in length

$S_o = K^+ \oplus \text{opad}$ where ipad and opad are 00110110 and 01011100 respectively taken b/8 times repeatedly.

$MD' = H(S_i \,||\, M)$

$MD = H(S_o \,||\, MD')$ or $MD = H(S_o \,||\, H(S_i \,||\, M))$

To a normal hash function, HMAC adds a compression instance to the processing. This structural implementation holds efficiency for shorter MAC values.

4.9 Cipher-Based Message Authentication Code -CMAC

- It is a MAC that is based on the use of a block cipher mode of operations for use with AES and triple DES.
- The CMAC overcomes the limitations of the Data Authentication Algorithm (DAA) which is based on DES.

The operation of the CMAC can be defined as follows:

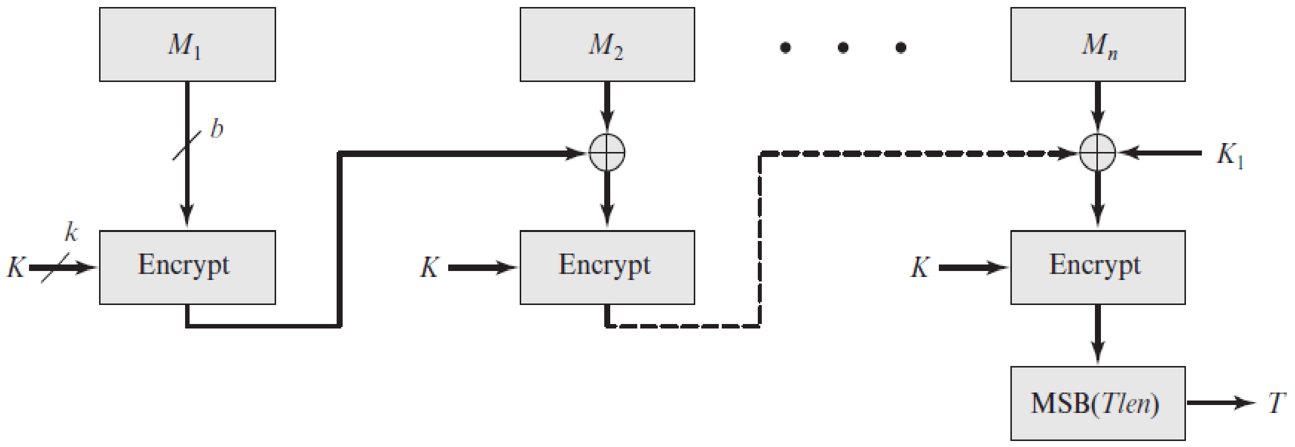
When the message is an integer multiple n of the cipher block length b. For AES, b = 128, and for triple DES, b = 64. The message is divided into n blocks (M1, M2,…, Mn). The algorithm makes use of a k-bit encryption key K and a b-bit constant, K1. For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows

$$\begin{aligned}
C_1 &= E(K, M_1) \\
C_2 &= E(K, [M_2 \oplus C_1]) \\
C_3 &= E(K, [M_3 \oplus C_2]) \\
&\cdot \\
&\cdot \\
&\cdot \\
C_n &= E(K, [M_n \oplus C_{n-1} \oplus K_1]) \\
T &= \text{MSB}_{Tlen}(C_n)
\end{aligned}$$

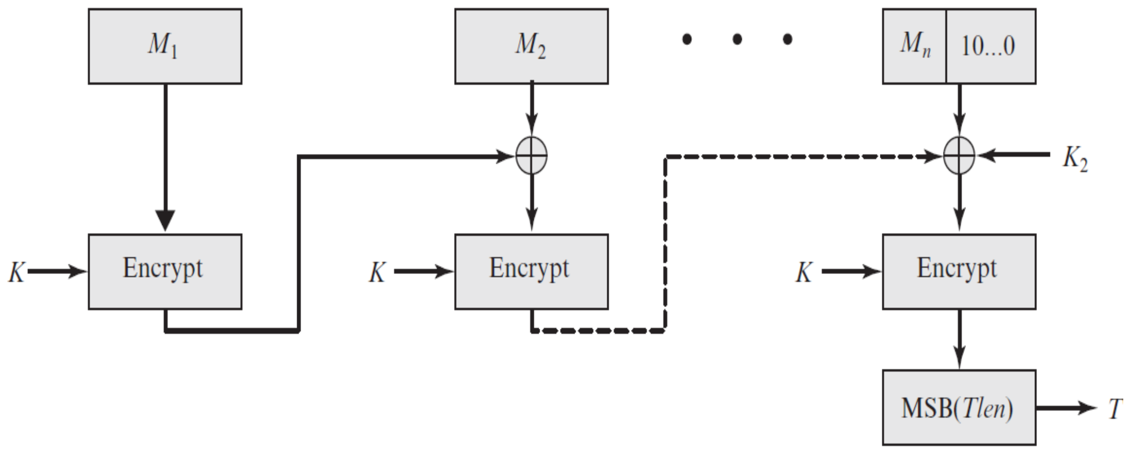
where

T	= message authentication code, also referred to as the tag
$Tlen$	= bit length of T
$\text{MSB}_s(X)$	= the s leftmost bits of the bit string X

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b. The CMAC operation then precedes as before, except that a different b-bit key K2 is used instead of K1.



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

The two b-bit keys are derived from the k-bit encryption key as follows.

$$\begin{aligned} L &= E(K, 0^b) \\ K_1 &= L \cdot x \\ K_2 &= L \cdot x^2 = (L \cdot x) \cdot x \end{aligned}$$

where multiplication (.) is done in the finite field GF(2^b) and x and x^2 are first- and second-order polynomials that are elements of GF(2^b). Thus, the binary representation of x consists of b - 2 zeros followed by 10; the binary representation of x^2 consists of b - 3 zeros followed by 100. The finite field is defined with respect to an irreducible polynomial that is lexicographically first among all such polynomials with the minimum possible number of nonzero terms. For the two approved block sizes, the polynomials are $x^{64} + x^4 + x^3 + x + 1$ and $x^{128} + x^7 + x^2 + x + 1$. To generate K_1 and K_2, the block cipher is applied to the block that consists entirely of 0 bits. The first sub key is derived from the resulting cipher text by a left shift of one bit and, conditionally, by XOR ing a constant that depends on the block size. The second sub key is derived in the same manner from the first subkey.

4.10 Digital signature and authentication protocols

Digital signatures are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message.

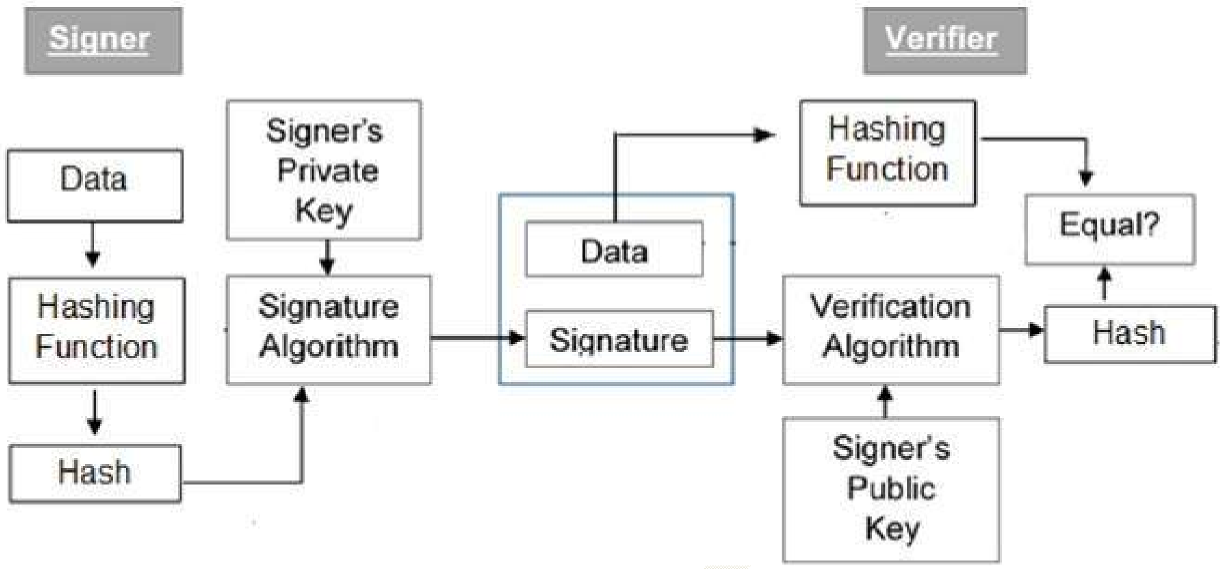
Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

Model of Digital Signature

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration



The following points explain the entire process in detail −

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier.
- Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence **signing a hash is more efficient than signing the entire data**.

Importance of Digital Signature

Out of all cryptographic primitives, the digital signature using public key cryptography is considered as very important and useful tool to achieve information security.

Apart from ability to provide non-repudiation of message, the digital signature also provides message authentication and data integrity. Let us briefly see how this is achieved by the digital signature −

- **Message authentication** − When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- **Data Integrity** − In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The hash of modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- **Non-repudiation** − Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

By adding public-key encryption to digital signature scheme, we can create a cryptosystem that can provide the four essential elements of security namely − Privacy, Authentication, Integrity, and Non-repudiation.

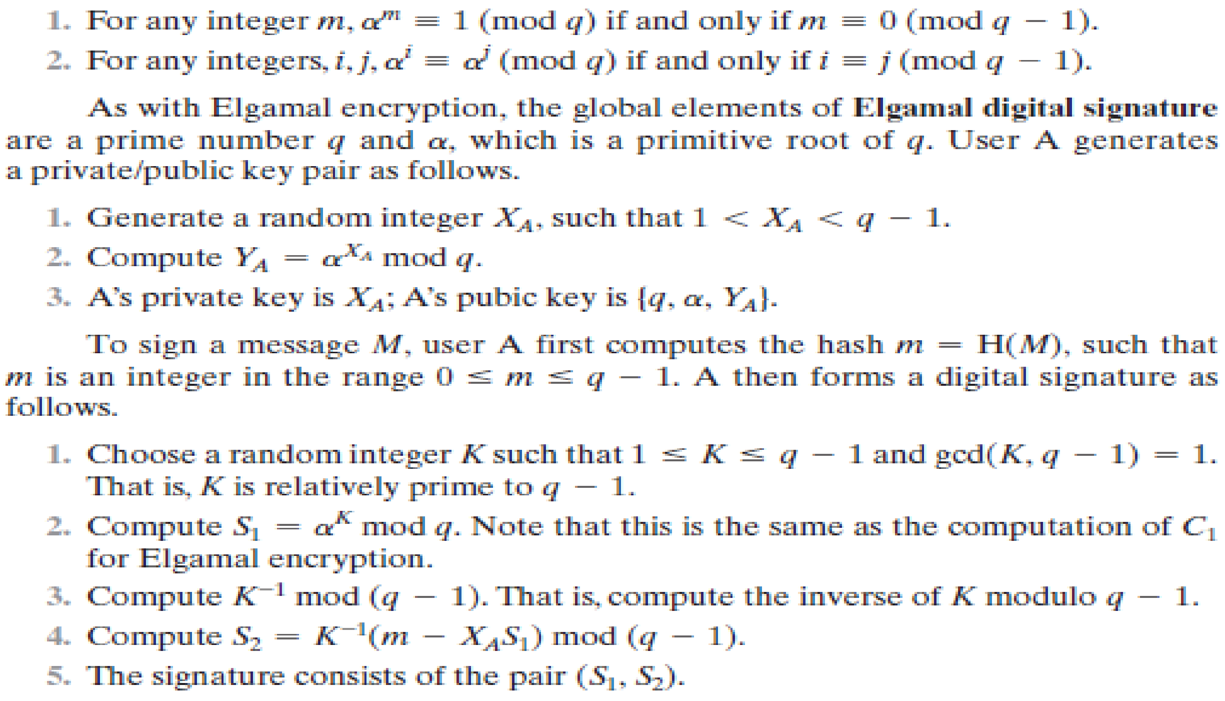
4.11 Digital Signatures Schemes

4.11.1 ELGAMAL DIGITAL SIGNATURE SCHEME

Before examining the NIST Digital Signature standard, it will be helpful to under- stand the ElGamal and Schnorr signature schemes. Recall that the ElGamal encryption scheme is designed to enable encryption by a user's public key with decryption by the user's private key. The ElGamal signature scheme involves the use of the private key for encryption and the public key for decryption.

Before proceeding, we need a result from number theory. That for a prime number q, if a is a primitive root of q, then

$\alpha, \alpha^2, \ldots, \alpha^{q-1}$ are distinct (mod q). It can be shown that, if a is a primitive root of q, then



1. For any integer m, $\alpha^m \equiv 1 \pmod{q}$ if and only if $m \equiv 0 \pmod{q-1}$.
2. For any integers, i, j, $\alpha^i \equiv \alpha^j \pmod{q}$ if and only if $i \equiv j \pmod{q-1}$.

As with Elgamal encryption, the global elements of **Elgamal digital signature** are a prime number q and α, which is a primitive root of q. User A generates a private/public key pair as follows.

1. Generate a random integer X_A, such that $1 < X_A < q - 1$.
2. Compute $Y_A = \alpha^{X_A} \bmod q$.
3. A's private key is X_A; A's pubic key is $\{q, \alpha, Y_A\}$.

To sign a message M, user A first computes the hash $m = \mathrm{H}(M)$, such that m is an integer in the range $0 \le m \le q - 1$. A then forms a digital signature as follows.

1. Choose a random integer K such that $1 \le K \le q - 1$ and $\gcd(K, q - 1) = 1$. That is, K is relatively prime to $q - 1$.
2. Compute $S_1 = \alpha^K \bmod q$. Note that this is the same as the computation of C_1 for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of K modulo $q - 1$.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair (S_1, S_2).

Any user B can verify the signature as follows.

1. Compute $V_1 = \alpha^m \bmod q$.
2. Compute $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$.

The signature is valid if $V_1 = V_2$. Let us demonstrate that this is so. Assume that the equality is true. Then we have

$\alpha^m \bmod q = (Y_A)^{S_1}(S_1)^{S_2} \bmod q$	assume $V_1 = V_2$
$\alpha^m \bmod q = \alpha^{X_A S_1}\alpha^{KS_2} \bmod q$	substituting for Y_A and S_1
$\alpha^{m - X_A S_1} \bmod q = \alpha^{KS_2} \bmod q$	rearranging terms
$m - X_A S_1 \equiv KS_2 \bmod (q - 1)$	property of primitive roots
$m - X_A S_1 \equiv KK^{-1}(m - X_A S_1) \bmod (q - 1)$	substituting for S_2

For example, let us start with the prime field GF(19); that is, $q = 19$. It has primitive roots {2, 3, 10, 13, 14, 15}, as shown in Table 2.7. We choose $\alpha = 10$.

Alice generates a key pair as follows:

1. Alice chooses $X_A = 16$.
2. Then $Y_A = \alpha^{X_A} \bmod q = \alpha^{16} \bmod 19 = 4$.
3. Alice's private key is 16; Alice's pubic key is $\{q, \alpha, Y_A\} = \{19, 10, 4\}$.

Suppose Alice wants to sign a message with hash value $m = 14$.

1. Alice chooses $K = 5$, which is relatively prime to $q - 1 = 18$.
2. $S_1 = \alpha^K \bmod q = 10^5 \bmod 19 = 3$ (see Table 2.7).
3. $K^{-1} \bmod (q - 1) = 5^{-1} \bmod 18 = 11$.
4. $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1) = 11\,(14 - (16)(3)) \bmod 18 = -374 \bmod 18 = 4$.

Bob can verify the signature as follows.

1. $V_1 = \alpha^m \bmod q = 10^{14} \bmod 19 = 16$.
2. $V_2 = (Y_A)^{S_1}(S_1)^{S_2} \bmod q = (4^3)(3^4) \bmod 19 = 5184 \bmod 19 = 16$.

Thus, the signature is valid because $V_1 = V_2$.

4.11.2 SCHNORR DIGITAL SIGNATURE SCHEME

The Schnorr signature scheme is based on discrete logarithms The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature. The main work for signature generation does not depend on the message and can be done during the idle time of the processor. The message-dependent part of the signature generation requires multiplying a 2*n*-bit integer with an *n*-bit integer. The scheme is based on using a prime modulus *p*, with *p* - 1 having a prime factor *q* of appropriate size; that is, *p* - 1 K 0 (mod *q*). Typically, we use *p* ≈ 21024 and *q* ≈ 2160. Thus, *p* is a 1024-bit number, and *q* is a 160-bit number, which is also the length of the SHA-1 hash value.

The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.

1. Choose primes p and q, such that q is a prime factor of $p - 1$.
2. Choose an integer a, such that $a^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

A user with private key s and public key v generates a signature as follows.

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$$e = \mathrm{H}(M \| x)$$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Any other user can verify the signature as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = \mathrm{H}(M \| x')$.

To see that the verification works, observe that

$$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod{p}$$

Hence, $\mathrm{H}(M \| x') = \mathrm{H}(M \| x)$.

4.12 Digital Certificate

What is a digital certificate?

A digital certificate is a form of electronic credential that can prove the authenticity of a user, device, server, or website. It uses PKI to help exchange communications and data securely over the internet.

This form of authentication is a type of cryptography that requires the use of public and private keys to validate users.

Public key certificates are issued by trusted third parties, a CA, who signs the certificate, thus verifying the identity of the device or user that is requesting access. To ensure validity, the public key will be matched with a corresponding private key that only the recipient has knowledge of. Digital certificates have a specific key pair that they are associated with: one public and one private.

A digital certificate contains the following identifiable information:

- User’s name
- Company or department of user
- IP (internet protocol) address or serial number of device
- Copy of the public key from a certificate holder
- Duration of time the certificate is valid for
- Domain certificate is authorized to represent

Benefits of digital certification

Digital certification can offer a level of security that is increasingly important in this digital age. In fact, cybersecurity has been named one of the top priorities of the U.S. Government by the Department of Homeland Security (DHS). Cybercrime is a major threat to businesses and individuals.

Digital certificates can provide the following benefits:

- **Security:** Digital certificates can keep internal and external communications confidential and protect the integrity of the data. It can also provide access control, ensuring only the intended recipient receives and can access the data.
- **Authentication:** With a digital certificate, users can be sure that the entity or person they are communicating with is who they say they are and makes sure that communications reach only the intended recipient.
- **Scalability:** Digital certificates can be used across a variety of platforms for individuals and large and small businesses alike. They can be issued, renewed, and revoked in a matter of seconds. They can be used to secure a range of user devices and be managed through one centralized system.
- **Reliability:** A digital certificate can only be issued by a publicly trusted and rigorously vetted CA, meaning that they cannot be easily tricked or faked.
- **Public trust:** The use of a digital certificate proves authenticity of a website, documents, or emails. It can assure users and clients that the company or individual is genuine and respects privacy and values security.

Different types of digital certification

There are three main types of public key certificates: TLS/SSL (Transport Layer Security/Secure Sockets Layer) certificates, client certificates, and code signing certificates. There are also variations within each type of certificate.

- **TLS/SSL certificates:** The TLS/SSL certificate is used to secure communications between a computer and the server, and it is hosted by the server. When a client computer seeks to access the server, the server will present the digital certificate to prove that it is authentic and the desired destination.

The HTTPS (Hypertext Transfer Protocol Secure) designation at the beginning of a web address or URL (Uniform Resource Locator) indicates the presence of a digital certificate.

When a client computer is presented with the digital certificate from the server, it will then run a certification path validation to ensure that the subject of the certificate matches the host name. Within the subject field of the certificate, a primary host name, or Common Name, must be identified. There can be multiple host names in the case of Subject Alternative Name (SAN) certificates and Unified Communications Certificates (UCCs).

Public web servers, or internet-facing servers, are required to have a digital certificate signed by a trusted CA. The TLS/SSL certificates can be domain validated, which is used for websites, or organization validated, which is used for light business authentication.

The extended validation provides full business authentication. It can offer the highest amount of security, trust, and authentication.

- **Client certificates:** This is a form of a digital ID that can identify one machine to another — a specific user to another user. This can be used to allow a user to access a protected and secure database and also for email.

With email, often the S/MIME (Secure/Multipurpose Internet Mail Extensions) protocol is used, which works for communications within an organization. Both parties will need to have copies of the digital certificate before communicating.

Email messages can be both encrypted and message integrity validated through use of a client certificate. Each user will need to send a digitally signed message and import the sender's certificate ahead of time.

- **Code signing certificates:** This type of digital certificate involves software or files. The publisher or developer of software will sign it to validate its authenticity to users downloading it.

This can be highly beneficial when software is downloaded through a third-party, ensuring that it is what it should be and has not been tampered with by malicious actors. This can confirm that files or software downloaded from the internet are valid and authentic.

Where digital certificates are used

Public certificate authorities are required to adhere to a set of baseline requirements. Most web browsers are set up to trust a pre-selected list of CAs, which are set by the browser itself or the operating system of the device. The verification of a digital certificate often happens behind the scenes and quickly, without a user even being aware of the process.

Websites use digital certificates to create the HTTPS connection, authenticating their validity by being signed by a trusted CA. This can help a browser to know it is visiting the real website it is seeking and not a fake or fraudulent one.

Digital certificates are also used in e-commerce to protect sensitive, identification, and financial information. Online shopping, stock trading, banking, and gaming all use digital certificates. Digital certificates can be used for electronic credit card holders and merchants to protect the financial transaction.

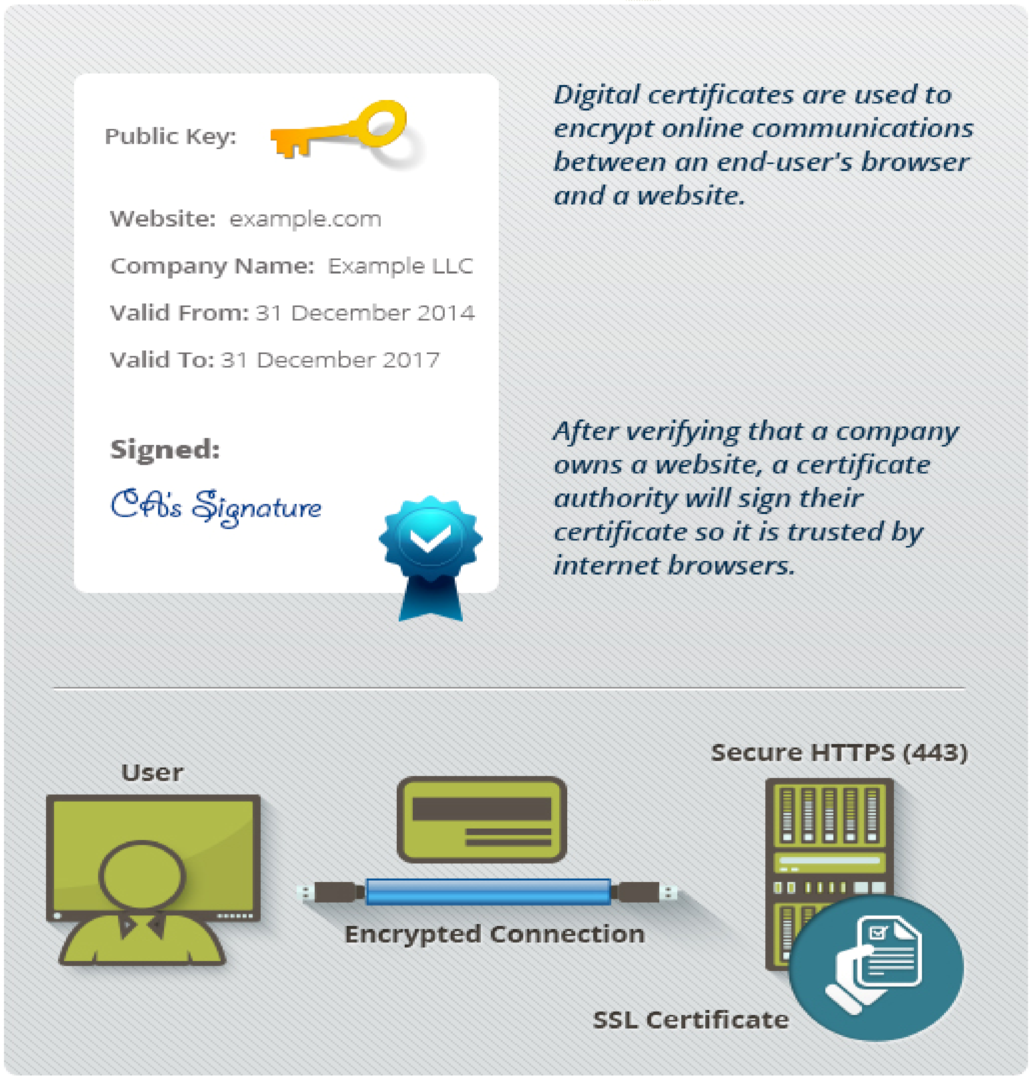
Another common use for digital certificates is for email communication. Email can also frequently contain a digital signature, which sends encrypted messages using a hashing approach.

Criticisms of digital certificates

While digital certificates are designed to invoke public trust and prove security and validity, they are not infallible. Digital certificates do have potential weaknesses that bad actors have exploited.

Organizations can be breached, for example, and cybercriminals can steal certifications and private key information, allowing them to then distribute malware. An illegitimate certificate can configure an infected system to trust it, opening the door to attack.

The MITM (man-in-the-middle) attack has also been known to intercept SSL/TLS traffic to gain access to sensitive information by either creating a fake root CA certificate or installing a rogue certificate that can then bypass security protocols. Overall, however, the use of digital certificates to secure websites is considered to be more secure than not using them.


Public Key:
Website: example.com
Company Name: Example LLC
Valid From: 31 December 2014
Valid To: 31 December 2017
Signed:
CA's Signature
Digital certificates are used to encrypt online communications between an end-user's browser and a website.
After verifying that a company owns a website, a certificate authority will sign their certificate so it is trusted by internet browsers.
User
Encrypted Connection
Secure HTTPS (443)
SSL Certificate

4.13 Key Management and Distribution.

4.13.1 Symmetric Key Distribution Using Symmetric Encryption

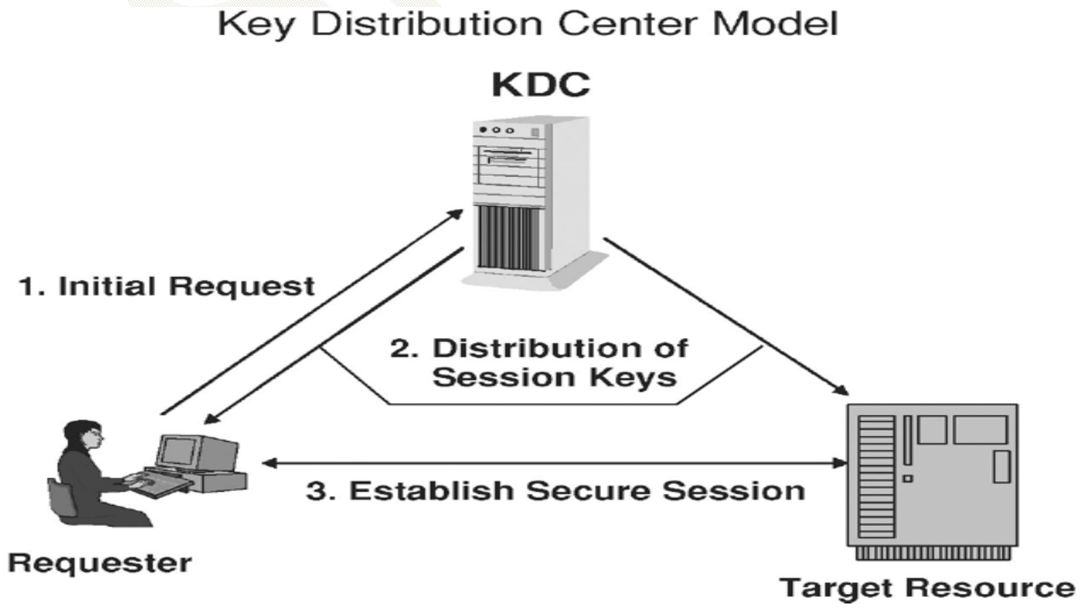
For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the *key distribution technique*, a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

4.13.1.1 A Key Distribution Scenario

The key distribution concept can be deployed in a number of ways. A typical scenario is illustrated in below Figure. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.


Key Distribution Center Model
KDC
1. Initial Request
2. Distribution of Session Keys
3. Establish Secure Session
Requester
Target Resource

A has a master key, *Ka*, known only to itself and the KDC; similarly, B shares the master key *Kb* with the KDC. The following steps occur.

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, *N*1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.
2. The KDC responds with a message encrypted using *Ka*. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC.

The message includes two items intended for A:
- The one-time session key, *Ks*, to be used for the session
- The original request message, including the nonce, to enable A to match this response with the appropriate request. Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request. In addition, the message includes two items intended for B:

- The one-time session key, *Ks*, to be used for the session
- An identifier of A (e.g., its network address), *IDA* These last two items are encrypted with *Kb* (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

4.13.1.2 Hierarchical Key Control

It is not necessary to limit the key distribution function to a single KDC. Indeed, for very large networks, it may not be practical to do so. As an alternative, a hierarchy of KDCs can be established. For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. In this case, any one of the three KDCs involved can actually select the key. The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork. A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities. Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

4.13.1.3 Session Key Lifetime

The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerations in determining the lifetime of a particular session key. For connection-oriented protocols, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session. If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles. For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination. Thus, it is not obvious how often one needs to change the session key. The most secure approach is to use a new session key for each exchange. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction. A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

4.13.1.4 Decentralized Key Control

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized. Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.

A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution. Thus, there may need to be as many as $[n(n - 1)]/2$ master keys for a configuration with n end systems.

A session key may be established with the following sequence of steps.
1. A issues a request to B for a session key and includes a nonce, $N1$.
2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value $f(N1)$, and another nonce, $N2$.
3. Using the new session key, A returns $f(N2)$ to B.

Thus, although each node must maintain at most $(n - 1)$ master keys, as many session keys as required may be generated and used. Because the messages transferred using the master key are short, cryptanalysis is difficult. As before, session keys are used for only a limited time to protect them.

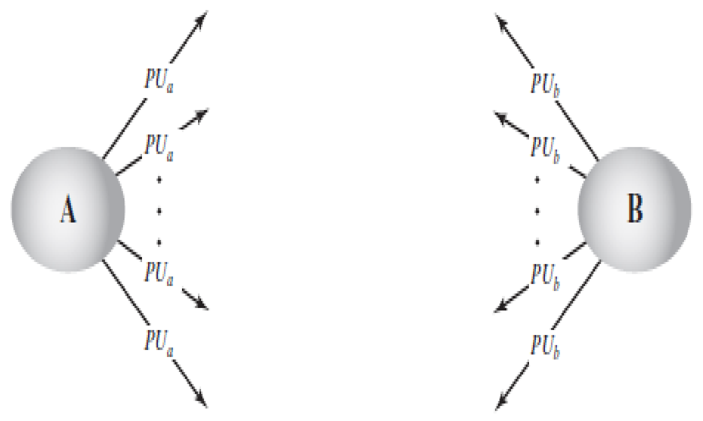
4.13.2 DISTRIBUTION OF PUBLIC KEYS

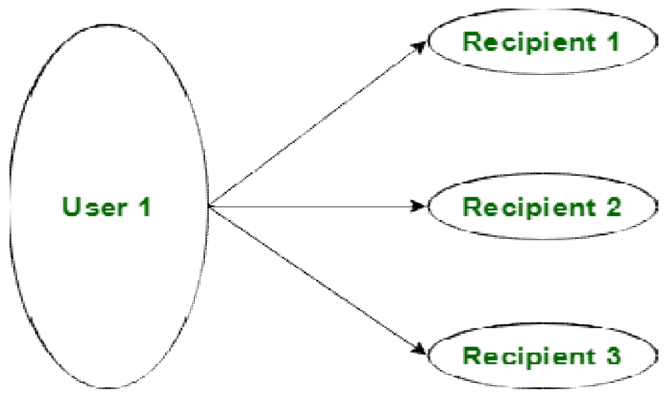
In cryptography, it is a very tedious task to distribute the public and private keys between sender and receiver. If the key is known to the third party (forger/eavesdropper) then the whole security mechanism becomes worthless. So, there comes the need to secure the exchange of keys.

The public key can be distributed in four ways:

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates.

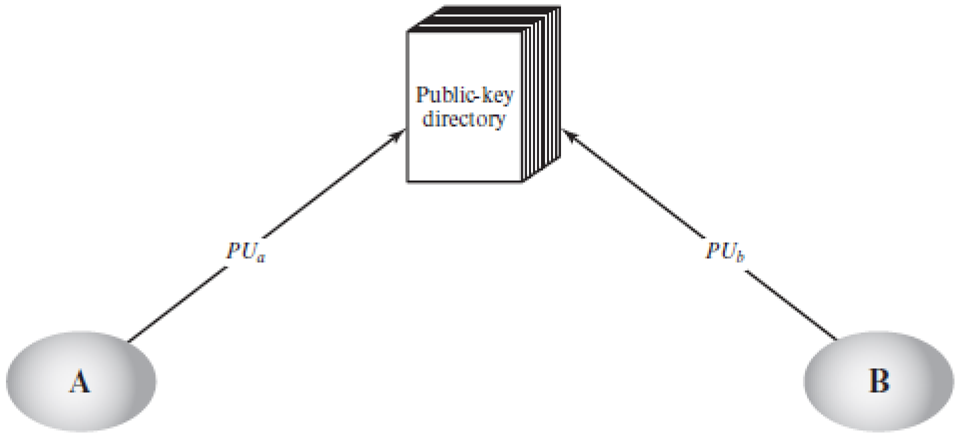
a. Public Announcement: Here the public key is broadcasted to everyone. The major weakness of this method is a forgery. Anyone can create a key claiming to be someone else and broadcast it. Until forgery is discovered can masquerade as claimed user.



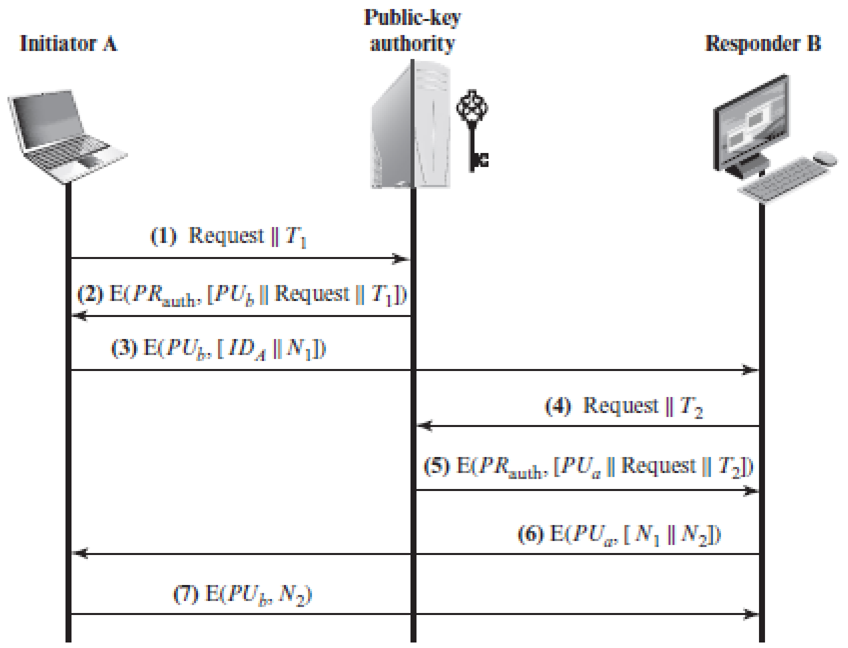


Public Key Announcement

b. Publicly Available Directory: In this type, the public key is stored in a public directory. Directories are trusted here, with properties like Participant Registration, access and allow to modify values at any time, contains entries like {name, public-key}. Directories can be accessed electronically still vulnerable to forgery or tampering.

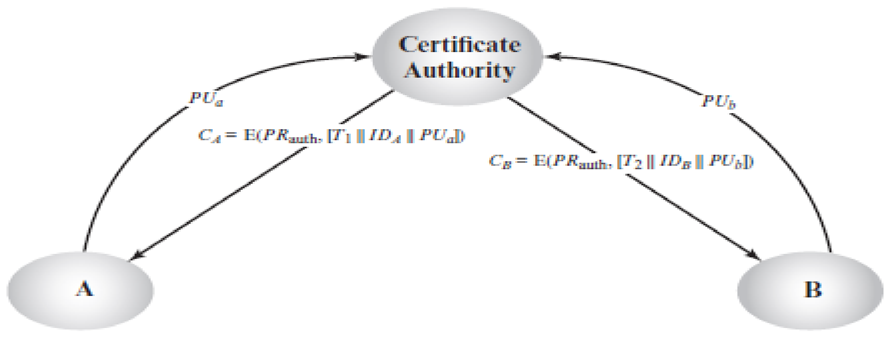


c. Public Key Authority: It is similar to the directory but, improves security by tightening control over the distribution of keys from the directory. It requires users to know the public key for the directory. Whenever the keys are needed, real-time access to the directory is made by the user to obtain any desired public key securely.


Initiator A
Public-key authority
Responder B
(1) Request || T_1
(2) E(PR_{auth}, [PU_b || Request || T_1])
(3) E(PU_b, [ID_A || N_1])
(4) Request || T_2
(5) E(PR_{auth}, [PU_a || Request || T_2])
(6) E(PU_a, [N_1 || N_2])
(7) E(PU_b, N_2)

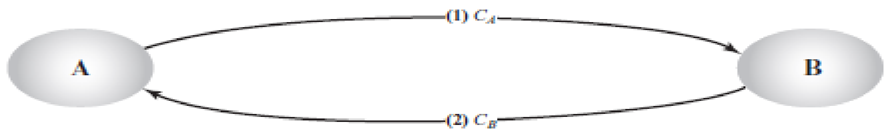
d. Public Certification: This time authority provides a certificate (which binds an identity to the public key) to allow key exchange without real-time access to the public authority each time. The certificate is accompanied by some other info such as period of validity, rights of use, etc. All of this content is signed by the private key of the certificate authority and it can be verified by anyone possessing the authority's public key.

First sender and receiver both request CA for a certificate which contains a public key and other information and then they can exchange these certificates and can start communication.



Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
A
B

(a) Obtaining certificates from CA



(b) Exchanging certificates