**A Definition of Computer Security**

The NIST *Computer Security Handbook* [NIST95] defines the term *computer*

*Security* as

**Computer Security:** The protection afforded to an automated information system

in order to attain the applicable objectives of preserving the integrity, availability,

and confidentiality of information system resources (includes hardware,

software, firmware, information/data, and telecommunications).

This definition introduces three key objectives that are at the heart of

Computer security.

**Confidentiality:** This term covers two related concepts:

**Data**2 **confidentiality:** Assures that private or confidential information is

not made available or disclosed to unauthorized individuals.

**Privacy:** Assures that individuals control or influence what information

Related to them may be collected and stored and by whom and to whom

that information may be disclosed.

**Integrity:** This term covers two related concepts:

**Data integrity:** Assures that data (both stored and in transmitted packets)

and programs are changed only in a specified and authorized manner.

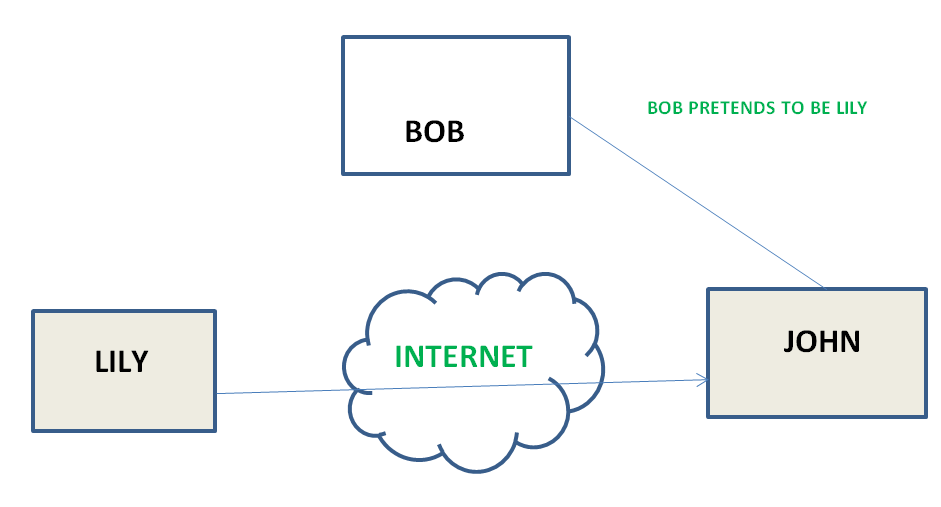
**System integrity:** Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

**Availability:** Assures that systems work promptly and service is not denied to authorized users.

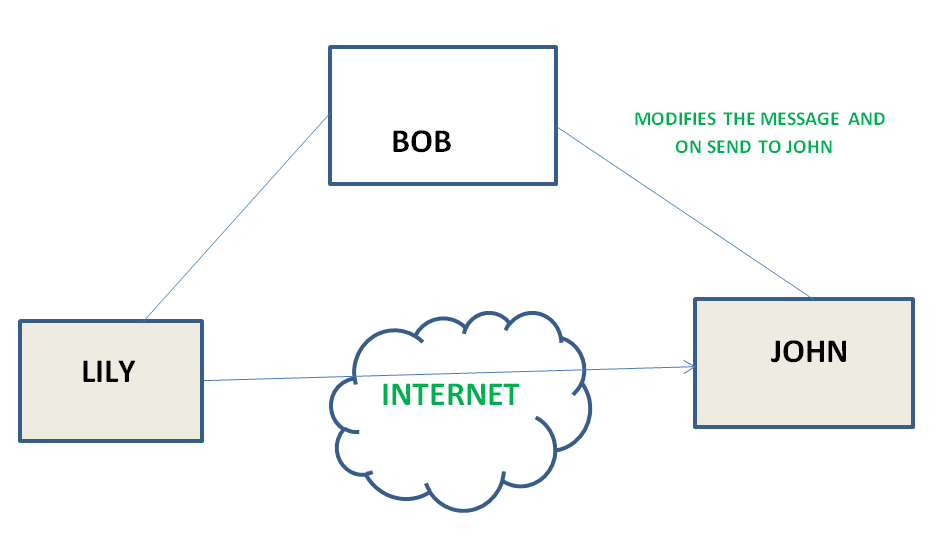
“Security attacks”

**Active attacks:** An Active attack attempts to alter system resources or effect their operations. Active attack involve some modification of the data stream or creation of false statement. Types of active attacks are as following:

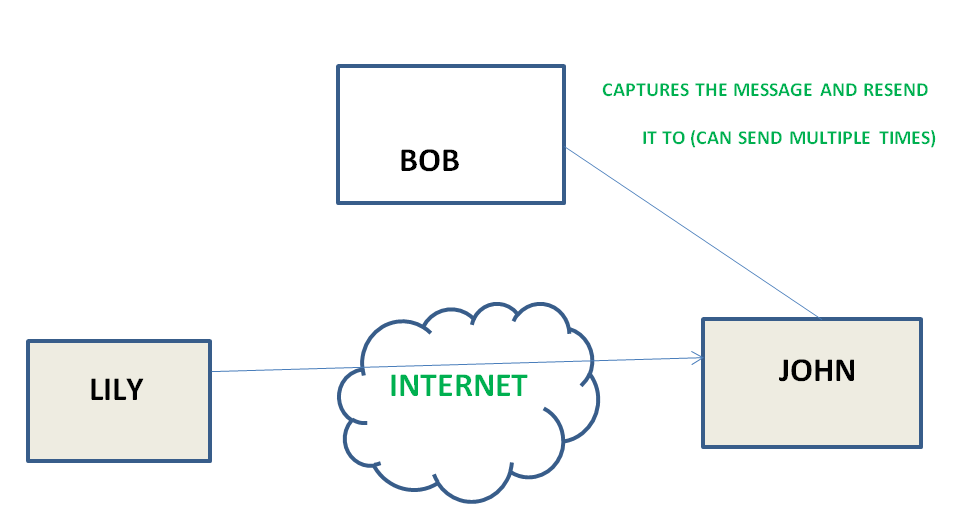
1. **Masquerade –**  
   Masquerade attack takes place when one entity pretends to be different entity. A Masquerade attack involves one of the other form of active attacks.



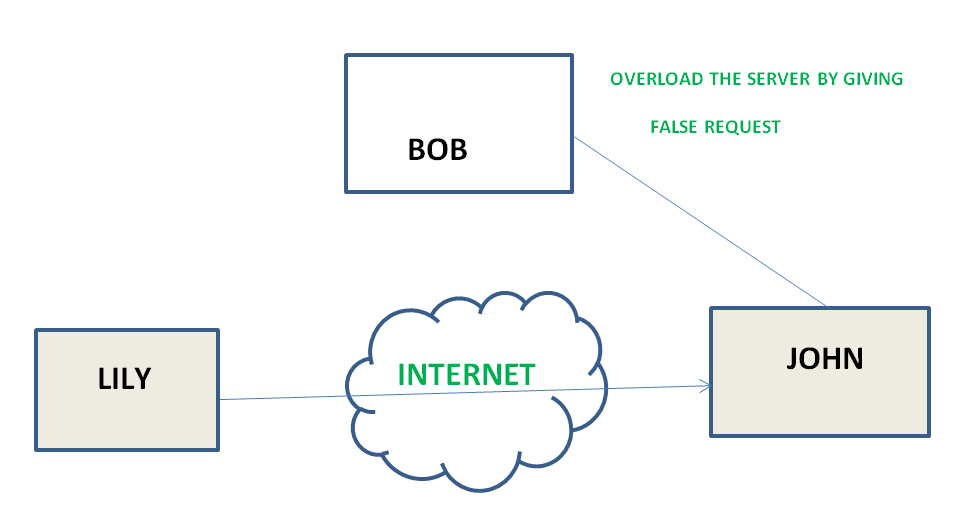
1. **Modification of messages –**  
   It means that some portion of a message is altered or that message is delayed or reordered to produce an unauthorised effect. For example, a message meaning “Allow JOHN to read confidential file X” is modified as “Allow Smith to read confidential file X”.



1. **Repudiation –**  
   This attack is done by either sender or receiver. The sender or receiver can deny later that he/she has send or receive a message. For example, customer ask his Bank “To transfer an amount to someone” and later on the sender(customer) deny that he had made such a request. This is repudiation.
2. **Replay –**  
   It involves the passive capture of a message and its subsequent the transmission to produce an authorized effect.

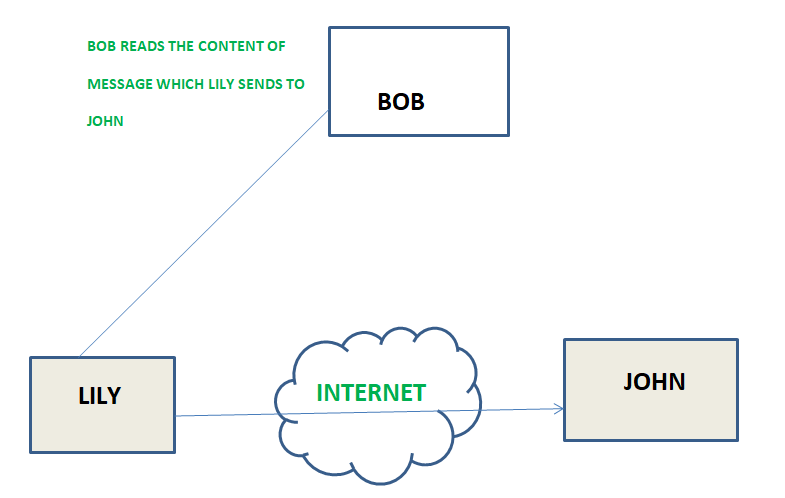


1. **Denial of Service –**  
   It prevents normal use of communication facilities. This attack may have a specific target. For example, an entity may suppress all messages directed to a particular destination. Another form of service denial is the disruption of an entire network wither by disabling the network or by overloading it by messages so as to degrade performance.

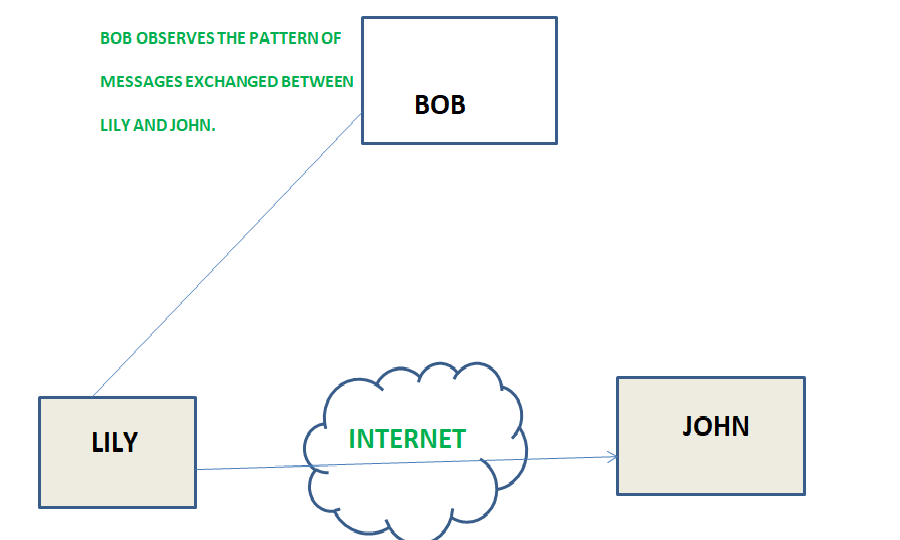


**Passive attacks:** A Passive attack attempts to learn or make use of information from the system but does not affect system resources. Passive Attacks are in the nature of eavesdropping on or monitoring of transmission. The goal of the opponent is to obtain information is being transmitted. Types of Passive attacks are as following:

1. **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.



1. **Traffic analysis –**  
   Suppose that we had a way of masking (encryption) of 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 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.



“Security services”

**Authentication**

The assurance that the communicating entity is the one that it claims to be.

**Peer Entity Authentication**

Used in association with a logical connection to provide confidence in the identity of the entitiesconnected.

**Data-Origin Authentication**

In a connectionless transfer, provides assurance thatthe source of received data is as claimed.

**Access Control**

The prevention of unauthorized use of a resource(i.e., this service controls who can have access to are source, under what conditions access can occur, and what those accessing the resource are allowed to do).

**Data Confidentiality**

The protection of data from unauthorized disclosure.

**Connection Confidentiality**

The protection of all user data on a connection.

**Connectionless Confidentiality**

The protection of all user data in a single data block.

**Selective-Field Confidentiality**

The confidentiality of selected fields within the user data on a connection or in a single data block.

**Traffic-Flow Confidentiality**

The protection of the information that might be derived from observation of traffic flows.

**Data Integrity**

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

**Connection Integrity with Recovery**

Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

**Connection Integrity without Recovery**

As above, but provides only detection without recovery.

**Selective-Field Connection Integrity**

Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

**Connectionless Integrity**

Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

**Selective-Field Connectionless Integrity**

Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

**Non repudiation** Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

**Non repudiation, Origin**

Proof that the message was sent by the specified party.

**Non repudiation, Destination**

Proof that the message was received by the specified party.

“Security mechanism”

**Specific Security Mechanisms**

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

**Encipherment**

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

**Digital Signature**

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

**Access Control**

A variety of mechanisms that enforce access rights to resources.

**Data Integrity**

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

**Authentication Exchange**

A mechanism intended to ensure the identity of an entity by means of information exchange.

**Traffic Padding**

The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

**Routing Control**

Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

**Notarization**

The use of a trusted third party to assure certain properties of a data exchange.

**Pervasive Security Mechanisms**

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

**Trusted Functionality**

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

**Security Label**

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

**Event Detection**

Detection of security-relevant events.

**Security Audit Trail**

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

**Security Recovery**

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

“Symmetric encryption principles ”

A **symmetric encryption** scheme has five ingredients (Figure 2.1):

■■ **Plaintext:** This is the original message or data that is fed into the algorithm

as input.

■■ **Encryption algorithm:** The **encryption** algorithm performs various substitutions

and transformations on the plaintext.

■■ **Secret key:** The secret key is also input to the algorithm. The exact substitutions

and transformations performed by the algorithm depend on the key.

■■ **Ciphertext:** This is the scrambled message produced as output. It depends on

the plaintext and the secret key. For a given message, two different keys will

produce two different ciphertexts.

■■ **Decryption algorithm:** This is essentially the encryption algorithm run in reverse.

It takes the ciphertext and the same secret key and produces the original plaintext.

There are two requirements for secure use of symmetric encryption:

**1.** We need a strong encryption algorithm. At a minimum, we would like the algorithm

to be such that an opponent who knows the algorithm and has access to

one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

**2.** Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

**Cryptography**

Cryptographic systems are generically classified along three independent

dimensions:

**1. The type of operations used for transforming plaintext to ciphertext.** All

encryption

algorithms are based on two general principles: substitution, in

which each element in the plaintext (bit, letter, group of bits or letters) is

mapped into another element; and transposition, in which elements in the

plaintext are rearranged. The fundamental requirement is that no information

be lost (i.e., that all operations be reversible). Most systems, referred to

as product systems, involve multiple stages of substitutions and transpositions.

**2. The number of keys used.** If both sender and receiver use the same key, the

system is referred to as symmetric, single-key, secret-key, or conventional

encryption. If the sender and receiver each use a different key, the system is referred to as asymmetric, two-key, or public-key encryption.

**3. The way in which the plaintext is processed.** A **block cipher** processes the input one block of elements at a time, producing an output block for each input block. A **stream cipher** processes the input elements continuously, producing output one element at a time, as it goes along.

**Cryptanalysis**

The process of attempting to discover the plaintext or key is known as **cryptanalysis**.

**Type of Attack Known to Cryptanalyst**

Ciphertext only ■ Encryption algorithm ■ Ciphertext to be decoded

Known plaintext ■ Encryption algorithm■ Ciphertext to be decoded■ One or more plaintext–ciphertext pairs formed with the secret key

Chosen plaintext ■ Encryption algorithm■ Ciphertext to be decoded

■ Plaintext message chosen by cryptanalyst, together with its corresponding

Ciphertext generated with the secret key

Chosen ciphertext ■ Encryption algorithm

■ Ciphertext to be decoded

■ Purported ciphertext chosen by cryptanalyst, together with its corresponding

decrypted plaintext generated with the secret key

Chosen text ■ Encryption algorithm

■ Ciphertext to be decoded

■ Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext

generated with the secret key

■ Purported ciphertext chosen by cryptanalyst, together with its corresponding

decrypted plaintext generated with the secret key

“Symmetric block algorithm”

The most commonly used symmetric encryption algorithms are block ciphers. A **block cipher** processes the plaintext input in fixed-sized blocks and produces a block of ciphertext of equal size for each plaintext block.

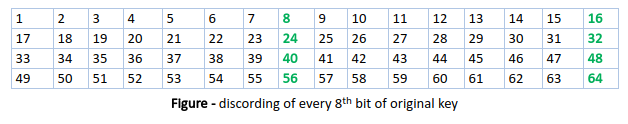
Data encryption standard (DES) | Set 1

Last Updated: 03-04-2020

**Data encryption standard (DES)** has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on decline.

DES is a block cipher, and encrypts data in blocks of size of 64 bit each, means 64 bits of plain text goes as the input to DES, which produces 64 bits of cipher text. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is show in figure.

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



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

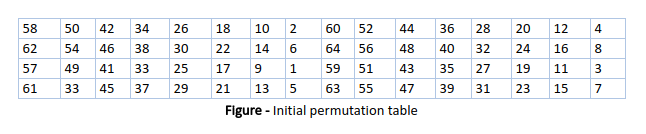
DES is based on the two fundamental attributes of cryptography: substitution (also called as confusion) and transposition (also called as diffusion). DES consists of 16 steps, each of which is called as a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level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

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

**Initial Permutation (IP) –**  
As we have noted, the Initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as show in figure.

For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block and so on.

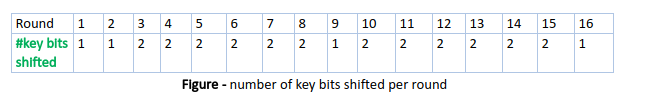
This is nothing but jugglery of bit positions of the original plain text block. the same rule applies for all the other bit positions which shows in the figure.



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

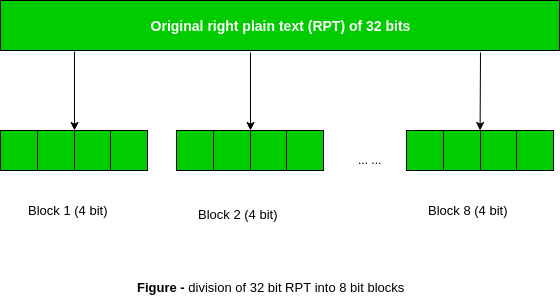
**Step-1: Key transformation –**  
We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called as key transformation. For this the 56 bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round.

For example, if the round number 1, 2, 9 or 16 the shift is done by only position for other rounds, the circular shift is done by two positions. The number of key bits shifted per round is show in figure.



After an appropriate shift, 48 of the 56 bit are selected. for selecting 48 of the 56 bits the table show in figure given below. For instance, after the shift, bit number 14 moves on the first position, bit number 17 moves on the second position and so on. If we observe the table carefully, we will realize that it contains only 48 bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as selection of a 48-bit sub set of the original 56-bit key it is called Compression Permutation.

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



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

**Triple DES**

As early as 1979, IBM realized that the DES key length was too short and devised

a way to effectively increase it, using triple encryption (Tuchman, 1979).

The method chosen, which has since been incorporated in International Standard

8732, is illustrated in Fig. 8-8. Here, two keys and three stages are used. In the

first stage, the plaintext is encrypted using DES in the usual way with *K*1. In the

second stage, DES is run in decryption mode, using *K*2 as the key. Finally, another

DES encryption is done with *K*1.

This design immediately gives rise to two questions. First, why are only two

keys used, instead of three? Second, why is **EDE** (**Encrypt Decrypt Encrypt**)

used, instead of **EEE** (**Encrypt Encrypt Encrypt**)? The reason that two keys are

used is that even the most paranoid of cryptographers believe that 112 bits is

adequate for routine commercial applications for the time being. (And among

cryptographers, paranoia is considered a feature, not a bug.) Going to 168 bits

would just add the unnecessary overhead of managing and transporting another

key for little real gain.

The reason for encrypting, decrypting, and then encrypting again is backward

compatibility with existing single-key DES systems. Both the encryption and decryption

functions are mappings between sets of 64-bit numbers. From a cryptographic

point of view, the two mappings are equally strong. By using EDE, however,

instead of EEE, a computer using triple encryption can speak to one using

single encryption by just setting *K*1 *K*2. This property allows triple encryption

to be phased in gradually, something of no concern to academic cryptographers

but of considerable importance to IBM and its customers.

“AES”

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The features of AES are as follows −

* Symmetric key symmetric block cipher
* 128-bit data, 128/192/256-bit keys
* Stronger and faster than Triple-DES
* Provide full specification and design details
* Software implementable in C and Java

## Operation of AES

AES is an iterative rather than Feistel cipher. It is based on ‘substitution–permutation network’. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −

Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

The schematic of AES structure is given in the following illustration −



## Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below −



### Byte Substitution (SubBytes)

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

### Shiftrows

Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row. Shift is carried out as follows −

* First row is not shifted.
* Second row is shifted one (byte) position to the left.
* Third row is shifted two positions to the left.
* Fourth row is shifted three positions to the left.
* The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

### MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

### Addroundkey

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

“Stream cipher”

Stream Ciphers

Last Updated: 16-10-2020

In stream cipher, one byte is encrypted at a time while in block cipher ~128 bits are encrypted at a time.

Initially, a key(k) will be supplied as input to pseudorandom bit generator and then it produces a random 8-bit output which is treated as keystream.

The resulted keystream will be of size 1 byte, i.e., 8 bits.

1. Stream Cipher follows the sequence of pseudorandom number stream.
2. One of the benefits of following stream cipher is to make cryptanalysis more difficult, so the number of bits chosen in the Keystream must be long in order to make cryptanalysis more difficult.
3. By making the key more longer it is also safe against brute force attacks.
4. The longer the key the stronger security is achieved, preventing any attack.
5. Keystream can be designed more efficiently by including more number of 1s and 0s, for making cryptanalysis more difficult.
6. Considerable benefit of a stream cipher is, it requires few lines of code compared to block cipher.

**Encryption :**  
For Encryption,

* Plain Text and Keystream produces Cipher Text (Same keystream will be used for decryption.).
* The Plaintext will undergo XOR operation with keystream bit-by-bit and produces the Cipher Text.

**Example –**

Plain Text : 10011001

Keystream : 11000011

`````````````````````

Cipher Text : 01011010

**Decryption :**  
For Decryption,

* Cipher Text and Keystream gives the original Plain Text (Same keystream will be used for encryption.).
* The Ciphertext will undergo XOR operation with keystream bit-by-bit and produces the actual Plain Text.

**Example –**

Cipher Text : 01011010

Keystream : 11000011

``````````````````````

Plain Text : 10011001

Decryption is just the reverse process of Encryption i.e. performing XOR with Cipher Text.

“Approaches to message authentication”

**10. Message Authentication**

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication. A message, file, document, or other collection of data is said to be authentic when it is genuine and came from its alleged source. Message authentication is a procedure that allows communicating parties to verify that received message is authentic. The two important aspects are to verify that the contents of the message have not been altered and that the source is authentic. We may also wish to verify a message’s timeliness (it has been artificially delayed and replayed) and sequence relative to other messages flowing between two parties.

10.**1. Authentication Using Conventional Encryption**

It is possible to perform authentication simply by the use of conventional encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit. 10.2. Message Authentication without Message Encryption

10.2. **Message Authentication without Message Encryption**

We examine several approaches to message authentication that do not rely on encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

10. 2. 1 **Message Authentication Code**

One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key KAB. When A has a message to send to B, it calculates the message authentication code as a function of the message and the key: MACM = F (KAB, M). The message plus code are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new message authentication code. The received code is compared to the calculated code. If we assume that only the receiver and the sender know the identity of the key, and if the received code matches the calculate code, then

1. The receiver is assured that the message has not been altered.

2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper code.

3. If the message includes a sequence number, then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

A number of algorithms could be used to generate the code. The national Bureau of Standards, in its publication DES Modes of Operation, recommends the use of Data Encryption Algorithm (DEA).

**10. 3. Hash Functions**

In general, a hash function is an efficiently evaluated function that takes an input string (usually binary) of arbitrary length and produces an output string of some fixed length, called a hash value or message digest. Value provides a way of checking whether the message has been manipulated or corrupted in transit or storage

The hash function H must be satisfied:

1.It should be one-way: For a given hash value v=H(x) it should be infeasible for an

opponent to find a message x such that x=H-1(v).

2. It should at least be weakly collision resistant: Given a hash value v=H(x) and the• message x from which it was computed, it should be computationally infeasible for an opponent to find another message y different from x such that v =H(y).

3.It might be strongly collision resistant: It is computationally infeasible for an• opponent to find a pair of distinct messages x and y such that H(x)=H(y).

“Secure hash algorithms”

**Secure Hash Algorithms**, also known as SHA, are a family of [cryptographic](https://brilliant.org/wiki/cryptography/) functions designed to keep data secured. It works by transforming the data using a [hash function](https://brilliant.org/wiki/hash-function/): an algorithm that consists of [bitwise operations](https://brilliant.org/wiki/bitwise-operations/), [modular additions](https://brilliant.org/wiki/modular-additions/?wiki_title=modular%20additions), and [compression functions](https://brilliant.org/wiki/compression-functions/). The hash function then produces a fixed-size string that looks nothing like the original. These algorithms are designed to be [one-way functions](https://brilliant.org/wiki/one-way-functions/?wiki_title=one-way%20functions), meaning that once they’re transformed into their respective hash values, it’s virtually impossible to transform them back into the original data. A few algorithms of interest are SHA-1, SHA-2, and SHA-3, each of which was successively designed with increasingly stronger encryption in response to hacker attacks. SHA-0, for instance, is now obsolete due to the widely exposed vulnerabilities.

A common application of SHA is to encrypting passwords, as the server side only needs to keep track of a specific user’s hash value, rather than the actual password. This is helpful in case an attacker hacks the database, as they will only find the hashed functions and not the actual passwords, so if they were to input the hashed value as a password, the hash function will convert it into another string and subsequently deny access. Additionally, SHAs exhibit the avalanche effect, where the modification of very few letters being encrypted causes a big change in output; or conversely, drastically different strings produce similar hash values. This effect causes hash values to not give any information regarding the input string, such as its original length. In addition, SHAs are also used to detect the tampering of data by attackers, where if a text file is slightly changed and barely noticeable, the modified file’s hash value will be different than the original file’s hash value, and the tampering will be rather noticeable.

# What is HMAC(Hash based Message Authentication Code)?

Last Updated: 26-05-2020

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



#### History

Processes and decisions pertinent to business are greatly dependent on integrity. If attackers tamper this data, it may affect the processes and business decisions. So while working online over the internet, care must be taken to ensure integrity or least know if the data is changed. That is when HMAC comes into use.

#### Applications

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

#### Working of HMAC

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

**Formula for HMAC:**

HMAC = hashFunc(secret key + message)

**“Public key encryption”**

The RSA method is based on some principles from number theory. We will

now summarize how to use the method; for details, consult the paper.

1. Choose two large primes, *p* and *q* (typically 1024 bits).

2. Compute *n* *p* *q* and *z* (*p* −1) (*q* −1)*.*

3. Choose a number relatively prime to *z* and call it *d*.

4. Find *e* such that *e* *d* 1 *mod z.*

With these parameters computed in advance, we are ready to begin encryption.

Divide the plaintext (regarded as a bit string) into blocks, so that each plaintext

message, *P*, falls in the interval 0 ≤*P < n.* Do that by grouping the plaintext into

blocks of *k* bits, where *k* is the largest integer for which 2*k < n* is true.

To encrypt a message, *P*, compute *C* *Pe* (mod *n*). To decrypt *C*, compute

*P* *Cd* (mod *n*). It can be proven that for all *P* in the specified range, the encryption

and decryption functions are inverses. To perform the encryption, you

need *e* and *n*. To perform the decryption, you need *d* and *n*. Therefore, the public

key consists of the pair (*e*, *n*) and the private key consists of (*d*, *n*)*.*

The security of the method is based on the difficulty of factoring large numbers.

If the cryptanalyst could factor the (publicly known) *n*, he could then find *p*

and *q*, and from these *z*. Equipped with knowledge of *z* and *e*, *d* can be found

using Euclid’s algorithm. Fortunately, mathematicians have been trying to factor

large numbers for at least 300 years, and the accumulated evidence suggests that it

is an exceedingly difficult problem

# Digital Signature

A digital signature is a mathematical technique which validates the authenticity and integrity of a message, software or digital documents. It allows us to verify the author name, date and time of signatures, and authenticate the message contents. The digital signature offers far more inherent security and intended to solve the problem of tampering and impersonation (Intentionally copy another person's characteristics) in digital communications.

The computer-based business information authentication interrelates both technology and the law. It also calls for cooperation between the people of different professional backgrounds and areas of expertise. The digital signatures are different from other electronic signatures not only in terms of process and result, but also it makes digital signatures more serviceable for legal purposes. Some electronic signatures that legally recognizable as signatures may not be secure as digital signatures and may lead to uncertainty and disputes.

## Application of Digital Signature

The important reason to implement digital signature to communication is:

* Authentication
* Non-repudiation
* Integrity

## Authentication

Authentication is a process which verifies the identity of a user who wants to access the system. In the digital signature, authentication helps to authenticate the sources of messages.

## Non-repudiation

Non-repudiation means assurance of something that cannot be denied. It ensures that someone to a contract or communication cannot later deny the authenticity of their signature on a document or in a file or the sending of a message that they originated.

## Integrity

Integrity ensures that the message is real, accurate and safeguards from unauthorized user modification during the transmission.

## Algorithms in Digital Signature

A digital signature consists of three algorithms:

**1. Key generation algorithm**

The key generation algorithm selects private key randomly from a set of possible private keys. This algorithm provides the private key and its corresponding public key.

**2. Signing algorithm**

A signing algorithm produces a signature for the document.

**3. Signature verifying algorithm**

A signature verifying algorithm either accepts or rejects the document's authenticity.

## How digital signatures work

Digital signatures are created and verified by using public key cryptography, also known as asymmetric cryptography. By the use of a public key algorithm, such as RSA, one can generate two keys that are mathematically linked- one is a private key, and another is a public key.

The user who is creating the digital signature uses their own private key to encrypt the signature-related document. There is only one way to decrypt that document is with the use of signer's public key.

This technology requires all the parties to trust that the individual who creates the signature has been able to keep their private key secret. If someone has access the signer's private key, there is a possibility that they could create fraudulent signatures in the name of the private key holder.

The steps which are followed in creating a digital signature are:

1. Select a file to be digitally signed.
2. The hash value of the message or file content is calculated. This message or file content is encrypted by using a private key of a sender to form the digital signature.
3. Now, the original message or file content along with the digital signature is transmitted.
4. The receiver decrypts the digital signature by using a public key of a sender.
5. The receiver now has the message or file content and can compute it.
6. Comparing these computed message or file content with the original computed message. The comparison needs to be the same for ensuring integrity.

**Key Management:**  
In cryptography it is a very tedious task to distribute the public and private key between sender and receiver. If 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.

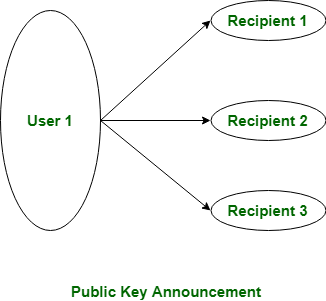
There are 2 aspects for Key Management:

1. Distribution of public keys.
2. Use of public-key encryption to distribute secret.

**Distribution of Public Key:**

Public key can be distributed in 4 ways: Public announcement, Publicly available directory, Public-key authority, and Public-key certificates. These are explained as following below.

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



1. **Publicly Available Directory:**  
   In this type, the public key is stored at 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.

1. **Public Key Authority:**  
   It is similar to the directory but, improve security by tightening control over distribution of keys from directory. It requires users to know public key for the directory. Whenever the keys are needed, a real-time access to directory is made by the user to obtain any desired public key securely.
2. **Public Certification:**  
   This time authority provides a certificate (which binds identity to the public key) to allow key exchange without real-time access to the public authority each time. The certificate is accompanied with some other info such as period of validity, rights of use etc. All of this content is signed by the trusted Public-Key or Certificate Authority (CA) and it can be verified by anyone possessing the authority’s public-key

**“Firewall”**

**8.6.2 Firewalls**

The ability to connect any computer, anywhere, to any other computer, anywhere,

is a mixed blessing. For individuals at home, wandering around the Internet

is lots of fun. For corporate security managers, it is a nightmare. Most companies

have large amounts of confidential information online—trade secrets, product

development plans, marketing strategies, financial analyses, etc. Disclosure of

this information to a competitor could have dire consequences.

In addition to the danger of information leaking out, there is also a danger of

information leaking in. In particular, viruses, worms, and other digital pests can

breach security, destroy valuable data, and waste large amounts of administrators’

time trying to clean up the mess they leave. Often they are imported by careless

employees who want to play some nifty new game.

Consequently, mechanisms are needed to keep ‘‘good’’ bits in and ‘‘bad’’ bits

out. One method is to use IPsec. This approach protects data in transit between

secure sites. However, IPsec does nothing to keep digital pests and intruders from

getting onto the company LAN. To see how to accomplish this goal, we need to

look at firewalls.

**Firewalls** are just a modern adaptation of that old medieval security standby:

digging a deep moat around your castle. This design forced everyone entering or

leaving the castle to pass over a single drawbridge, where they could be inspected

by the I/O police. With networks, the same trick is possible: a company can have

many LANs connected in arbitrary ways, but all traffic to or from the company is

forced through an electronic drawbridge (firewall), as shown in Fig. 8-29. No

other route exists.

The firewall acts as a **packet filter**. It inspects each and every incoming and

outgoing packet. Packets meeting some criterion described in rules formulated by

the network administrator are forwarded normally. Those that fail the test are

uncermoniously dropped.

The filtering criterion is typically given as rules or tables that list sources and

destinations that are acceptable, sources and destinations that are blocked, and default

rules about what to do with packets coming from or going to other machines.

In the common case of a TCP/IP setting, a source or destination might consist of

an IP address and a port. Ports indicate which service is desired. For example,

TCP port 25 is for mail, and TCP port 80 is for HTTP. Some ports can simply be

blocked. For example, a company could block incoming packets for all IP addresses

combined with TCP port 79. It was once popular for the Finger service to

look up people’s email addresses but is little used today.

Other ports are not so easily blocked. The difficulty is that network administrators

want security but cannot cut off communication with the outside world.

That arrangement would be much simpler and better for security, but there would

be no end to user complaints about it. This is where arrangements such as the

**DMZ** (**DeMilitarized Zone**) shown in Fig. 8-29 come in handy. The DMZ is the

part of the company network that lies outside of the security perimeter. Anything

goes here. By placing a machine such as a Web server in the DMZ, computers on

the Internet can contact it to browse the company Web site. Now the firewall can

be configured to block incoming TCP traffic to port 80 so that computers on the

Internet cannot use this port to attack computers on the internal network.

### Packet-Filtering Firewalls



As the most “basic” and oldest type of firewall architecture, packet-filtering firewalls basically create a checkpoint at a traffic router or switch. The firewall performs a simple check of the data packets coming through the router—inspecting information such as the destination and origination IP address, packet type, port number, and other surface-level information without opening up the packet to inspect its contents.

If the information packet doesn’t pass the inspection, it is dropped.

The good thing about these firewalls is that they aren’t very resource-intensive. This means they don’t have a huge impact on system performance and are relatively simple. However, they’re also relatively easy to bypass compared to firewalls with more robust inspection capabilities.

### Circuit-Level Gateways

As another simplistic firewall type that is meant to quickly and easily approve or deny traffic without consuming significant computing resources, circuit-level gateways work by verifying the transmission control protocol (TCP) handshake. This TCP handshake check is designed to make sure that the session the packet is from is legitimate.

While extremely resource-efficient, these firewalls do not check the packet itself. So, if a packet held malware, but had the right TCP handshake, it would pass right through. This is why circuit-level gateways are not enough to protect your business by themselves.

### Stateful Inspection Firewalls

These firewalls combine both packet inspection technology and TCP handshake verification to create a level of protection greater than either of the previous two architectures could provide alone.

However, these firewalls do put more of a strain on computing resources as well. This may slow down the transfer of legitimate packets compared to the other solutions.

### Proxy Firewalls (Application-Level Gateways/Cloud Firewalls)

Proxy firewalls operate at the application layer to filter incoming traffic between your network and the traffic source—hence, the name “application-level gateway.” These firewalls are delivered via a cloud-based solution or another proxy device. Rather than letting traffic connect directly, the proxy firewall first establishes a connection to the source of the traffic and inspects the incoming data packet.

This check is similar to the stateful inspection firewall in that it looks at both the packet and at the TCP handshake protocol. However, proxy firewalls may also perform deep-layer packet inspections, checking the actual contents of the information packet to verify that it contains no malware.

Once the check is complete, and the packet is approved to connect to the destination, the proxy sends it off. This creates an extra layer of separation between the “client” (the system where the packet originated) and the individual devices on your network—obscuring them to create additional anonymity and protection for your network.

If there’s one drawback to proxy firewalls, it’s that they can create significant slowdown because of the extra steps in the data packet transferal process.

### Next-Generation Firewalls

Many of the most recently-released firewall products are being touted as “next-generation” architectures. However, there is not as much consensus on what makes a firewall truly next-gen.

Some common features of next-generation firewall architectures include deep-packet inspection (checking the actual contents of the data packet), TCP handshake checks, and surface-level packet inspection. Next-generation firewalls may include other technologies as well, such as intrusion prevention systems (IPSs) that work to automatically stop attacks against your network.

The issue is that there is no one definition of a next-generation firewall, so it’s important to verify what specific capabilities such firewalls have before investing in one.

“Password management”

Password Management

1. Password Protection 2. The Vulnerability of Passwords 3. Access Control

**PASSWORD MANAGEMENT**

**1. Password Protection**

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

       The ID determines whether the user is authorized to gain access to a system. 



        The ID determines the privileges accorded to the user. 

        The ID is used in ,what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user. 

**2. The Vulnerability of Passwo111rds**

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, the following procedure is employed.

Each user selects a password of up to eight printable characters in length. 



This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. 



        The encryption routine, known as crypt(3), is based on DES. The DES algorithm is modified using a 12-bit "salt" value.

Typically, this value is related to the time at which the password is assigned to the user. 



The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros.



The output of the algorithm then serves as input for a second encryption. 



This process is repeated for a total of 25 encryptions. 



The resulting 64-bit output is then translated into an 11-character sequence. 



The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID. 



This method has been shown to be secure against a variety of cryptanalytic attacks 

**The salt serves three purposes:**

        It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ. 



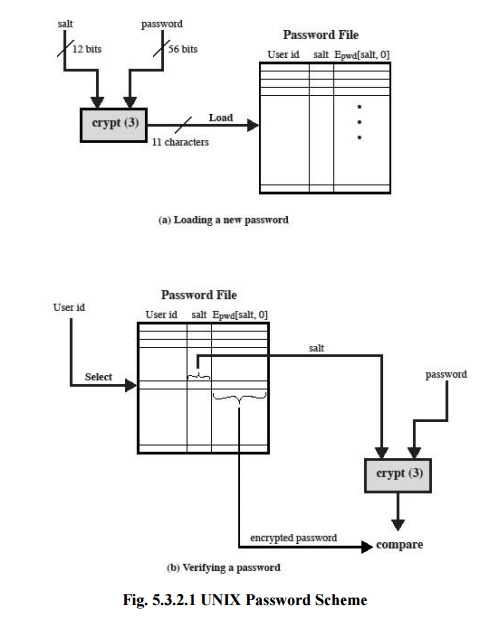
        It effectively increases the length of the password without requiring the user to remember two additional characters. 



        It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack. 

When a user attempts to log on to a UNIX system, the user provides an ID and a password. The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password. The salt and user-supplied password are used as input to the encryption routine. If the result matches the stored value, the password is accepted.The encryption routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25.

Thus, there are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine.



**3. Access Control**

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user.

**Password Selection Strategies**

Four basic techniques are in use:

  User education 



  Computer-generated passwords 



 Reactive password checking 

 Proactive password checking 

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines

**Computer-generated passwords**also have problems. If the passwords are quite random innature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down

**A reactive password**checking strategy is one in which the system periodically runs its ownpassword cracker to find guessable passwords.

The most promising approach to improved password security is a **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

## What is Common Criteria?

Common Criteria (CC) is an international set of specifications and guidelines designed to evaluate information security products and systems. Common Criteria, officially known as the Common Criteria for Information Technology Security Evaluation, was developed to certify that products and systems meet a pre-defined security standard for government deployments. Security products that have undergone successful testing and evaluation are awarded Common Criteria certification.

#### A Brief History of Common Criteria

The standard was developed by the governments of the U.S., Canada, Germany, France, the UK and the Netherlands in 1994. Common Criteria is the result of combining the CTCPEC (Canada), the TCSEC (U.S.), and the ITSEC (European) standards. The unification of security evaluation criteria would help to avoid the re-evaluation of products and systems addressing international markets.

When reviewing Common Criteria documentation or certifications, there are several key concepts to consider.

**Get a Common Criteria Certified NGFW**

[**Learn How**](https://www.forcepoint.com/product/ngfw-next-generation-firewall)

## Common Criteria Key Concepts

**Target of Evaluation** – The device or system to be reviewed for CC certification.

**Protection Profile (PP)** – Template used to define a standard set of security requirements for a particular class of related products. A protection profile serves as a reusable template of security requirements. Depending on the Target of Evaluation, multiple profiles may be used at once.

**Security Target (ST)** – Explicitly stated set of requirements specific to the capabilities of the product under evaluation.

**Security Functional Requirements (SFRs)** – Security requirements that refer to unique security functions provided by a product.

**Evaluation Assurance Levels (EAL)** – Used to define the way the product is tested and how thoroughly. These levels are scaled from 1 to 7, with 7 being the highest level and 1 the lowest. A higher number does not necessarily mean that the product went through more rigorous testing.

## How Are Products Tested?

If a vendor has a product that they would like to be evaluated under the Common Criteria standards, they must complete a Security Target (ST) description. This will include an overview of the product's security features and an evaluation of any potential security risks. The vendor will also need to complete a self-assessment that details how the product complies with the relevant Evaluation Assurance Level and Protection Profile the vendor wants their product to be tested against.

Tests are usually carried out under laboratory conditions to validate the product's security features and to evaluate how well the product meets the requirements defined in the Protection Profile. If the results are successful, the product will usually be awarded CC certification. The objective of CC certification is to assure customers that they can trust the products they are investing in to support the vendor's claims and most importantly, offer the best protection for their network environment