Definition of Security in Computing 1

Computer security is the protection of computer systems and networks from the theft of or damage to their hardware, software, or electronic data & information, as well as from the disruption of the services they provide. It is basically the protection of computer systems and information from harm, theft, and unauthorized use. It is the process of preventing and detecting unauthorized use of your computer system.

Security Goals 1

When we talk about computer security, we mean that we are addressing three important aspects of any computer-related system: confidentiality, integrity, and availability.
Confidentiality ensures that computer-related assets are accessed only by authorized parties. That is, only those who should have access to something will actually get that access. By "access", we mean not only reading but also viewing, printing, or simply knowing that a particular asset exists. Confidentiality is sometimes called secrecy or privacy.
Integrity means that assets can be modified only by authorized parties or only in authorized ways. In this context, modification includes writing, changing, changing status, deleting, and creating.
Availability means that assets are accessible to authorized parties at appropriate times. In other words, if some person or system has legitimate access to a particular set of objects, that access should not be prevented. For this reason, availability is sometimes known by its opposite, denial of service.
Security in computing addresses these three goals. One of the challenges in building a secure system is finding the right balance among the goals, which often conflict. For example, it is easy to preserve a particular object's confidentiality in a secure system simply by preventing everyone from reading that object. However, this system is not secure, because it does not meet the requirement of availability for proper access. That is, there must be a balance between confidentiality and availability.

Other Security Components (Authentication, Authorization, Non-Repudiation) 1

Authentication: The process of verifying the identity of a user or process
Authorization: Authorization is the function of specifying access rights/privileges to resources
Non-Repudiation: Provides assurance of the origin or delivery of data in order to protect the sender against false denial by the recipient that the data has been received

Threats in Computing Systems 1

A computer system threat is anything that leads to loss or corruption of data or physical damage to the hardware and/or infrastructure.
In computer security, a threat is a potential negative action or event facilitated by a vulnerability that results in an unwanted impact to a computer system or application.

Kinds of Threats 1

An **interception** means that some unauthorized party has gained access to an asset. The outside party can be a person, a program, or a computing system. Examples of this type of failure are illicit copying of program or data files, or wiretapping to obtain data in a network.

Although a loss may be discovered fairly quickly, a silent interceptor may leave no traces by which the interception can be readily detected.
In an **interruption**, an asset of the system becomes lost, unavailable, or unusable. An example is malicious destruction of a hardware device, erasure of a program or data file, or malfunction of an operating system file manager so that it cannot find a particular disk file.
If an unauthorized party not only accesses but tampers with an asset, the threat is a **modification**. For example, someone might change the values in a database, alter a program so that it performs an additional computation, or modify data being transmitted electronically. It is even possible to modify hardware. Some cases of modification can be detected with simple measures, but other, more subtle, changes may be almost impossible to detect.
Finally, an unauthorized party might create a **fabrication** of counterfeit objects on a computing system. The intruder may insert spurious transactions to a network communication system or add records to an existing database. Sometimes these additions can be detected as forgeries, but if skillfully done, they are virtually indistinguishable from the real thing.
These four classes of threats interception, interruption, modification, and fabrication describe the kinds of problems we might encounter.

Vulnerabilities (Hardware & Software) 1

A vulnerability is a weakness in the security system, for example, in procedures, design, or implementation, that might be exploited to cause loss or harm. For instance, a particular system may be vulnerable to unauthorized data manipulation because the system does not verify a user's identity before allowing data access.

Hardware Vulnerabilities

Hardware is more visible than software, largely because it is composed of physical objects. Because we can see what devices are hooked to the system, it is rather simple to attack by adding devices, changing them, removing them, intercepting the traffic to them, or flooding them with traffic until they can no longer function. However, designers can usually put safeguards in place.

Software Vulnerabilities

Computing equipment is of little use without the software (operating system, controllers, utility programs, and application programs) that users expect. Software can be replaced, changed, or destroyed maliciously, or it can be modified, deleted, or misplaced accidentally. Whether intentional or not, these attacks exploit the software's vulnerabilities.

Sometimes, the attacks are obvious, as when the software no longer runs. More subtle are attacks in which the software has been altered but seems to run normally. Whereas physical equipment usually shows some mark of inflicted injury when its boundary has been breached, the loss of a line of source or object code may not leave an obvious mark in a program. Furthermore, it is possible to change a program so that it does all it did before, and then some. That is, a malicious intruder can "enhance" the software to enable it to perform functions you may not find desirable. In this case, it may be very hard to detect that the software has been changed, let alone to determine the extent of the change.

A classic example of exploiting software vulnerability is the case in which a bank worker realized that software truncates the fractional interest on each account. In other words, if the monthly interest on an account is calculated to be $14.5467, the software credits only $14.54 and ignores the $.0067. The worker amended the software so that the throw-away interest (the $.0067) was placed into his own account. Since the accounting practices ensured only that all accounts balanced, he built up a large amount of money from the thousands of account throw-aways without detection. It was only when he bragged to a colleague of his cleverness that the scheme was discovered.

Software Deletion

Software is surprisingly easy to delete. Each of us has, at some point in our careers, accidentally erased a file or saved a bad copy of a program, destroying a good previous copy. Because of software's high value to a commercial computing center, access to software is usually carefully controlled through a process called **configuration management** so that software cannot be deleted, destroyed, or replaced accidentally. Configuration management uses several techniques to ensure that each version or release retains its integrity. When configuration management is used, an old version or release can be replaced with a newer version only when it has been thoroughly tested to verify that the improvements work correctly without degrading the functionality and performance of other functions and services.

Software Modification

Software is vulnerable to modifications that either cause it to fail or cause it to perform an unintended task. Indeed, because software is so susceptible to "off by one" errors, it is quite easy to modify. Changing a bit or two can convert a working program into a failing one. Depending on which bit was changed, the program may crash when it begins or it may execute for some time before it falters.

With a little more work, the change can be much more subtle: The program works well most of the time but fails in specialized circumstances. For instance, the program may be maliciously modified to fail when certain conditions are met or when a certain date or time is reached. Because of this delayed effect, such a program is known as a **logic bomb.** For example, a disgruntled employee may modify a crucial program so that it accesses the system date and halts abruptly after July 1. The employee might quit on May l and plan to be at a new job miles away by July.

Another type of change can extend the functioning of a program so that an innocuous program has a hidden side effect. For example, a program that ostensibly structures a listing of files belonging to a user may also modify the protection of all those files to permit access by another user.

Other categories of software modification include

- **Trojan horse:** a program that overtly does one thing while covertly doing another
- **virus:** a specific type of Trojan horse that can be used to spread its "infection" from one computer to another
- **trapdoor:** a program that has a secret entry point
- **information leaks** in a program: code that makes information accessible to unauthorized people or programs

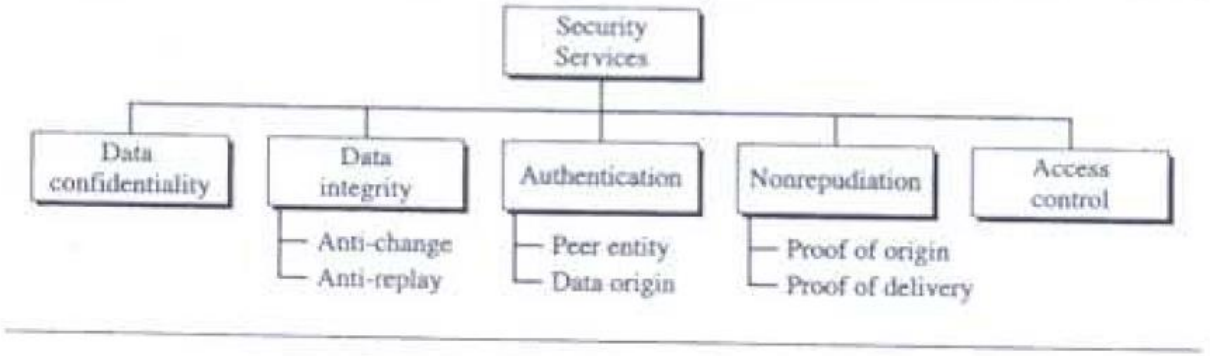
Methods of Defense 1

Harm occurs when a threat is realized against vulnerability. To protect against harm, then, we can neutralize the threat, close the vulnerability, or both. The possibility for harm to occur is called risk. We can deal with harm in several ways. We can seek to

- **prevent it**, by blocking the attack or closing the vulnerability
- **deter it**, by making the attack harder but not impossible
- **deflect it**, by making another target more attractive (or this one less so)
- **detect it**, either as it happens or sometime after the fact
- **recover** from its effects

Security Mechanisms & Services 1

Figure 1.3 *Security services*



Data Confidentiality

Data confidentiality is designed to protect data from disclosure attack. The service as defined by X.800 is very broad and encompasses confidentiality of the whole message or part of a message and also protection against traffic analysis. That is, it is designed to prevent snooping and traffic analysis attack.

Data Integrity

Data integrity is designed to protect data from modification, insertion, deletion, and replaying by an adversary. It may protect the whole message or part of the message.

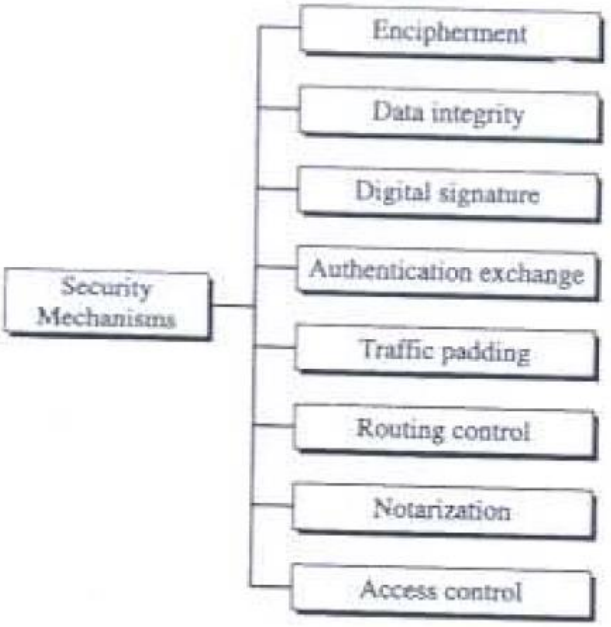
Authentication

This service provides the **authentication** of the party at the other end of the line. In connection-oriented communication, it provides authentication of the sender or receiver

Security Mechanisms

ITU-T (X.800) also recommends some **security mechanisms** to provide the security services defined in the previous section. Figure 1.4 gives the taxonomy of these mechanisms.

Figure 1.4 *Security mechanisms*



during the connection establishment (peer entity authentication). In connectionless communication, it authenticates the source of the data (data origin authentication).

Nonrepudiation

Nonrepudiation service protects against repudiation by either the sender or the receiver of the data. In nonrepudiation with proof of the origin, the receiver of the data can later prove the identity of the sender if denied. In nonrepudiation with proof of delivery, the sender of data can later prove that data were delivered to the intended recipient.

Access Control

Access control provides protection against unauthorized access to data. The term *access* in this definition is very broad and can involve reading, writing, modifying, executing programs, and so on.

Encipherment: Hiding or covering data

Data Integrity

The **data integrity** mechanism appends to the data a short checkvalue that has been created by a specific process from the data itself. The receiver receives the data and the checkvalue. He creates a new checkvalue from the received data and compares the newly created checkvalue with the one received. If the two checkvalues are the same, the integrity of data has been preserved.

Digital Signature

A **digital signature** is a means by which the sender can electronically sign the data and the receiver can electronically verify the signature. The sender uses a process that involves showing that she owns a private key related to the public key that she has announced publicly. The receiver uses the sender's public key to prove that the message is indeed signed by the sender who claims to have sent the message.

Authentication Exchange

In **authentication exchange,** two entities exchange some messages to prove their identity to each other. For example, one entity can prove that she knows a secret that only she is supposed to know.

Traffic Padding

Traffic padding means inserting some bogus data into the data traffic to thwart the adversary's attempt to use the traffic analysis.

Routing Control

Routing control means selecting and continuously changing different available routes between the sender and the receiver to prevent the opponent from eavesdropping on a particular route.

Notarization

Notarization means selecting a third trusted party to control the communication between two entities. This can be done, for example, to prevent repudiation. The receiver can involve a trusted party to store the sender request in order to prevent the sender from later denying that she has made such a request.

Access Control

Access control uses methods to prove that a user has access right to the data or resources owned by a system. Examples of proofs are passwords and PINs.

Basics of cryptography 1

Cryptography is the study of secret (crypto-) writing (-graphy). It is concerned with developing algorithms which may be used to:
It conceals the context of some message from all except the sender and recipient (privacy or secrecy). It verifies the correctness of a message to the recipient (authentication or integrity). It is the basis of many technological solutions to computer and communications security problems.

Classical Cryptosystems 1

In the classical cryptography the original data i.e., the plain text is transformed into the encoded format i.e. cipher text so that we can transmit this data through insecure communication channels. A data string which known as key is used to control the transformation of the data from plain text to cipher text. This arrangement helps to keep data safe as it required the key for extracting the original information from the cipher text. Without the key no one can read the data. In this technique it is assumed that the only authorized receiver has the key.
Classical Cryptography has two types of techniques:
Symmetric Cryptography:
In the symmetric cryptography a single key is used for encrypting and decryption the data. This encryption key is private key. This is the limitation of this encryption technique that this private key must be distributed only among the authorized sender and receiver.

Asymmetric Cryptography:
In the asymmetric cryptography a pair of key, i.e., public key and private key is used for encryption and

decryption. A sender can use its public key to encrypt the data and on receiver end receiver can decrypt the data by using its private key. This technique overcomes the problem of key distribution.

Mono & Polyalphabetic Ciphers 1

Monoalphabetic Ciphers

We first discuss a group of substitution ciphers called the **monoalphabetic ciphers.** In monoalphabetic substitution, a character (or a symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position in the text. For example, if the algorithm says that letter A in the plaintext is changed to letter D, every letter A is changed to letter D. In other words, the relationship between letters in the plaintext and the ciphertext is one-to-one.

Polyalphabetic Ciphers

In **polyalphabetic substitution,** each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is one-to-many. For example, "a" could be enciphered as "D" in the beginning of the text, but as "N" at the middle. Polyalphabetic ciphers have the advantage of hiding the letter frequency of the underlying language. Eve cannot use single-letter frequency statistic to break the ciphertext.

To create a polyalphabetic cipher, we need to make each ciphertext character dependent on both the corresponding plaintext character and the position of the plaintext character in the message. This implies that our key should be a stream of subkeys, in which each subkey depends somehow on the position of the plaintext character that uses that subkey for encipherment. In other words, we need to have a key stream $k = (k_1, k_2, k_3, \ldots)$ in which k_i is used to encipher the ith character in the plaintext to create the ith character in the ciphertext.

Working principles of Substitution & Transposition Ciphers

A **substitution cipher** replaces one symbol with another. If the symbols in the plaintext are alphabetic characters, we replace one character with another. For example, we can replace letter A with letter D, and letter T with letter Z. If the symbols are digits (0 to 9), we can replace 3 with 7, and 2 with 6. Substitution ciphers can be categorized as either monoalphabetic ciphers or polyalphabetic ciphers.

A **transposition cipher** does not substitute one symbol for another, instead it changes the location of the symbols. A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eighth position in the plaintext

Block Ciphers Vs Stream Ciphers 2

Block Cipher Converts the plain text into cipher text by taking plain text's block at a time.	Stream Cipher Converts the plain text into cipher text by taking 1 byte of plain text at a time.

Types of P-Boxes 2

P-Boxes

A **P-box** (permutation box) parallels the traditional transposition cipher for characters. It transposes bits. We can find three types of P-boxes in modern block ciphers: straight P-boxes, expansion P-boxes, and compression P-boxes, as shown in Figure 5.4.

Figure 5.4 *Three types of P-boxes*

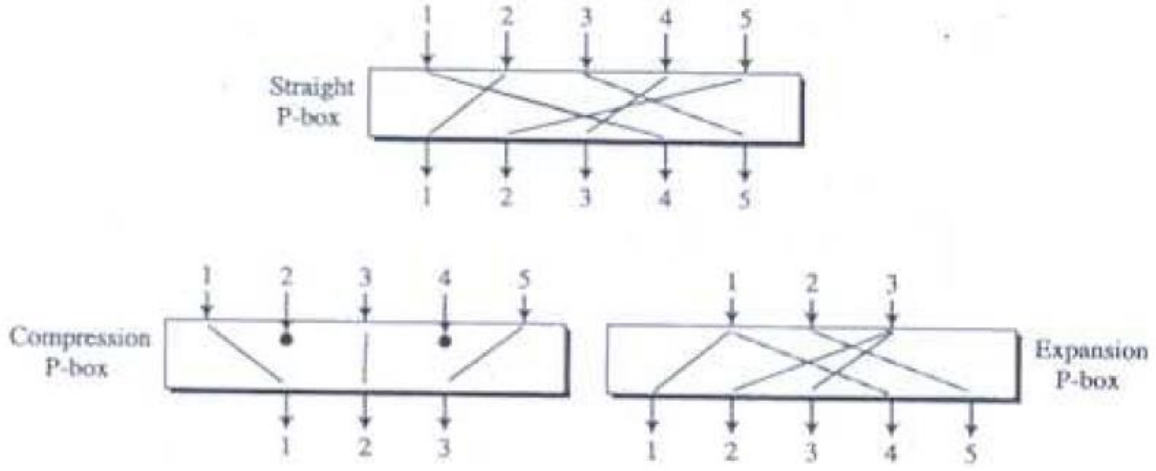


Figure 5.4 shows a 5 × 5 straight P-box, a 5 × 3 compression P-box, and a 3 × 5 expansion P-box. We will discuss each of them in more detail.

RSA Algorithm 2

The **Rivest Shamir Adelman (RSA)** cryptosystem is a public key system. Based on an underlying hard problem and named after its three inventors, this algorithm was introduced in 1978 and to date remains secure. RSA has been the subject of extensive cryptanalysis, and no serious flaws have yet been found. Although the amount of analysis is no guarantee of a method's security, our confidence in the method grows as time passes without discovery of a flaw.
RSA relies on an area of mathematics known as number theory, in which mathematicians study properties of numbers such as their prime factors. The RSA encryption algorithm combines results from number theory with the degree of difficulty in determining the prime factors of a given number. The RSA algorithm also operates with arithmetic mod n.

The two keys used in RSA, d and e, are used for decryption and encryption. For simplicity, we call the encryption key e and the decryption key d.

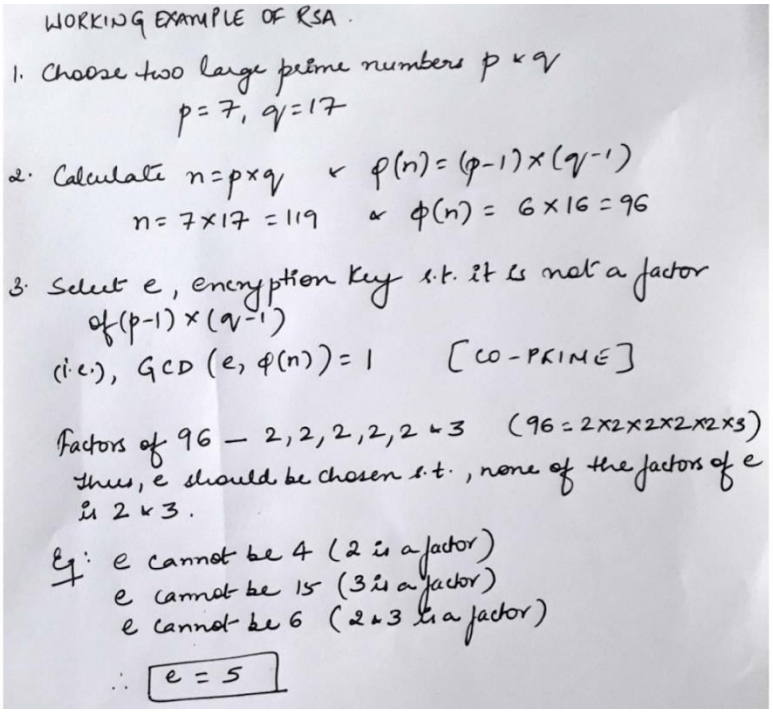
RSA Algorithm

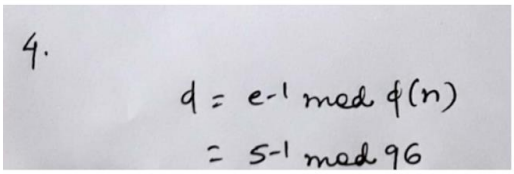
Public key pair {e,n}

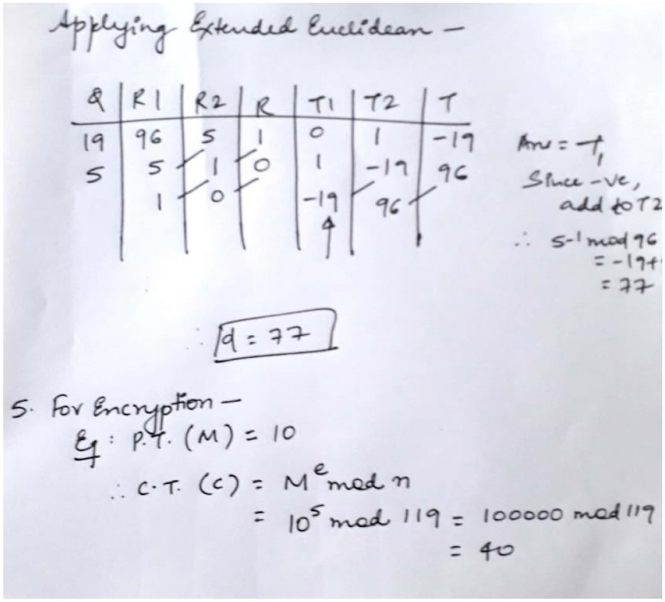
Private key pair {d,n}

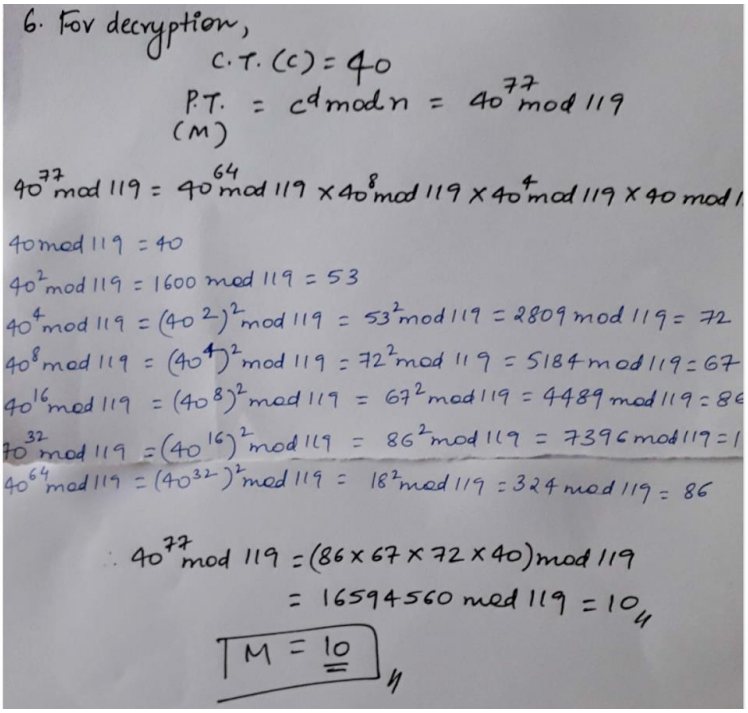
Encryption: $C = M^e \bmod n$

Decryption: $M = C^d \bmod n$









Hash functions (Defination, Properties, Applications) 3

Hash function: A hash function is a mathematical function that converts a numerical input value into another compressed numerical value. The input to the hash function is of arbitrary length but output is always of fixed length

Properties of Hash functions:

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).

There are two direct applications of hash function based on its cryptographic properties.

One is the Password Storage (Hash functions provide protection to password storage)

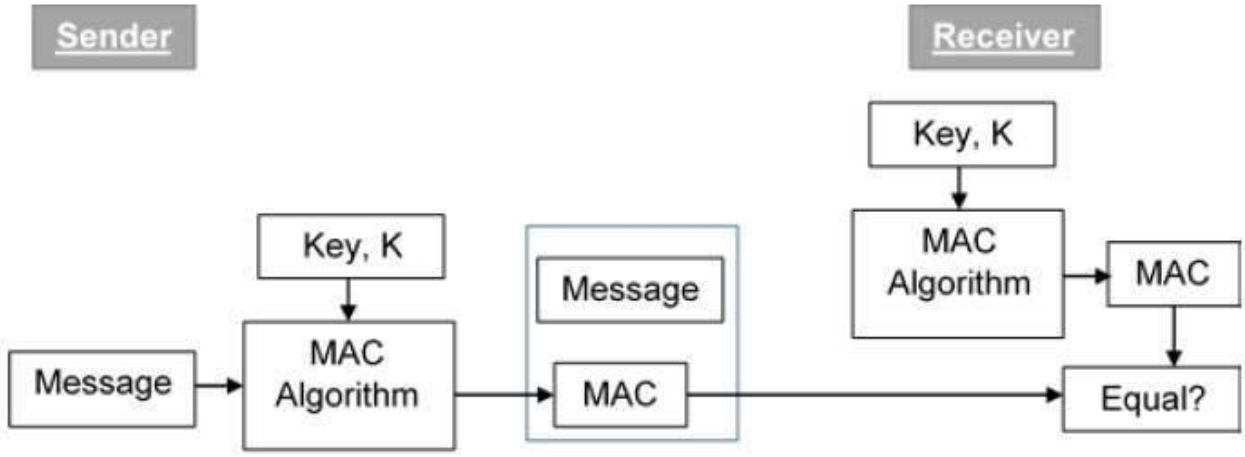
Other is Data Integrity Check (Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files. This application provides assurance to the user about correctness of the data)

MDC Message Detection Code 3

The **Message Digest** guarantees the integrity of a message that not been changed. However, message digest does not authenticate the sender of the message. To provide message authentication, sender needs to provide proof that he/she sending the message & not an imposter. The digest created by a cryptographic hash function is called a **Modification Detection Code (MDC)** to detect any modification in the message.

MAC Message Authentication Code 3

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 −



Nested MAC 3

It Used to improve the security of MAC, **nested MACs** are used. In nested MAC the hashing is done in two steps:

1. The key is concatenated with the message and is hashed to create an intermediate digest.
2. The key is concatenated with the intermediate digest to create the final digest

Hashed MAC 3

- It is the standard version of nested MAC. The implementation of HMAC is much more complex than the simplified nested MAC.

- The message is divided into N blocks, each of b bits.
- The secret key is left padded with 0's to create a b-bit key.
- The result of step 2 is XORed with a constant called **ipad** (input pad) to create a b-bit block. The value of ipad is 36 in hexadecimal.
- The resulting block is prepended to the N-block message. The result is N+1 blocks.
- The result of step 4 is hashed to create an n-bit digest. The digest is called the intermediate HMAC.
- The intermediate n-bit HMAC is padded with 0's to make a b-bit block.
- Steps 2 and 3 are repeated by a different constant **opad** (output pad). The value of opad is 5C in hexadecimal.
- The result of step 7 is prepended to the block of step 6.
- The result of step 8 is hashed with the same hashing algorithm to create the final n-bit HMAC.

CMAC Cipher-based Message Authentication Code 3

- The message is divided into N blocks, each m bits long.
- The size of the CMAC is n bits.
- If the last block is not m bits, it is padded with a 1 bit followed by enough 0 bits to make it m bits.
- The first block of the message is encrypted with the symmetric key to create a new m bit block of encrypted data.

- This block is XORed with the next block and the result is encrypted again to create a new m bit block.
- The process continues until the last block and the result is encrypted.
- The n leftmost bit from the last block is the CMAC.
- In addition to symmetric key K, CMAC also uses another key k, which is applied only at the last step.

Key Generation in CMAC

- The key is derived from the encryption algorithm with plaintext of m 0 bits using the cipher key, K.

The result is then multiplied by x if no padding is applied and multiplied by x2 if padding is applied.

Digital Signatures 3

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.

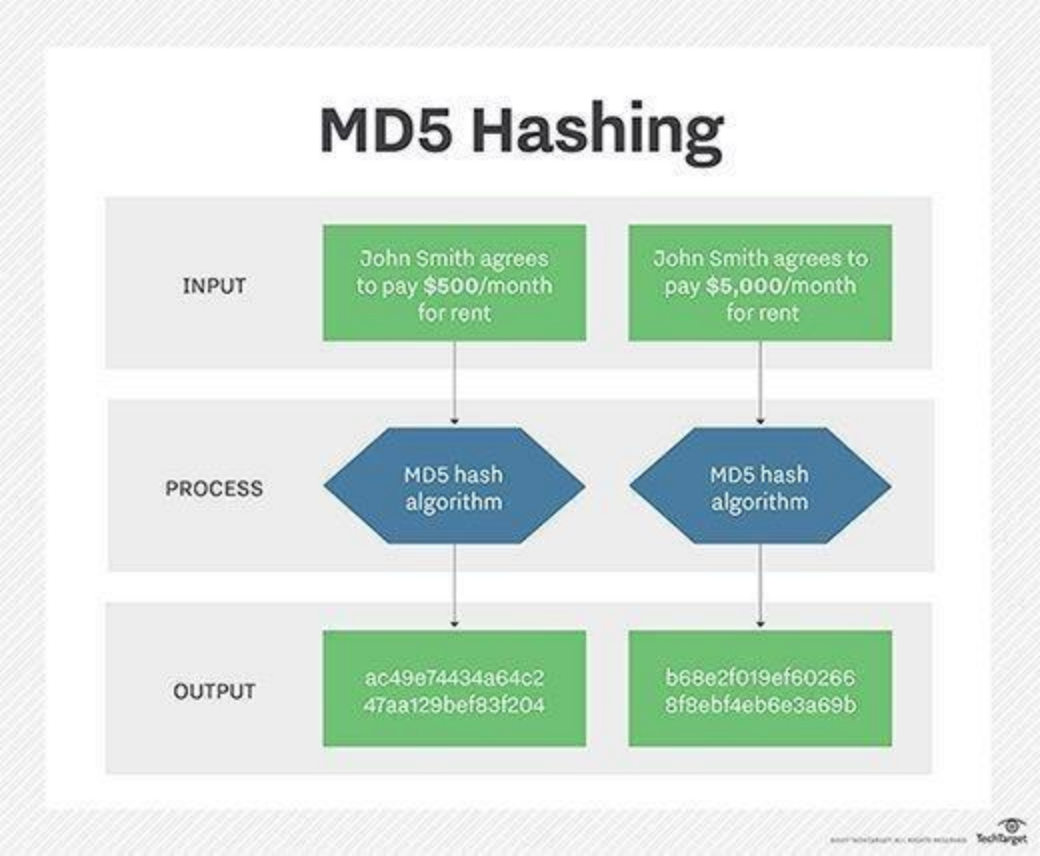
Digital Certificates 3

Digital Certificates are a standard of security for establishing an encrypted link between a server and a client. Generally, between a mail server or a webserver, which protects data in transitions by encrypting them. A Digital Certificate is also a Digital ID or a passport which is issued by a Third Party Authority which verifies the identity of the server's owner and not claiming a false identity.

Asymmetric Key Cryptography 3

MD5 3

MD5
The MD5 (message-digest algorithm) hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value to be used for authenticating the original message.
The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating digital signatures. But MD5 has been deprecated for uses other than as a noncryptographic checksum to verify data integrity and detect unintentional data corruption.

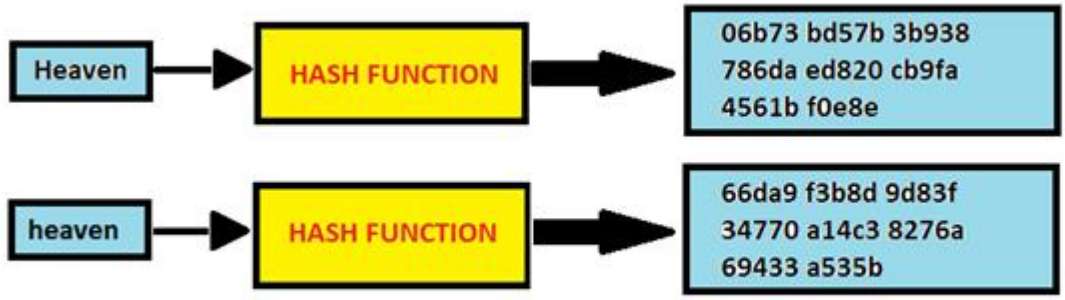


Steps

1. Pad message so its length is 448 mod 512 (Message is 64 bit less than the multiple of 512)
2. Append a 64-bit length value to message (1 is appended followed by many zeros)
3. Initialise 4-word (128-bit) MD buffer (A,B,C,D) each of 32bits
4. Process message in 16-word (512-bit) blocks:
    - using **4 rounds** of 16 bit operations on message block & buffer
    - add output to buffer input to form new buffer value
5. Output hash value is the final buffer value

SHA 3

SHA stands for secure hashing algorithm. 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.

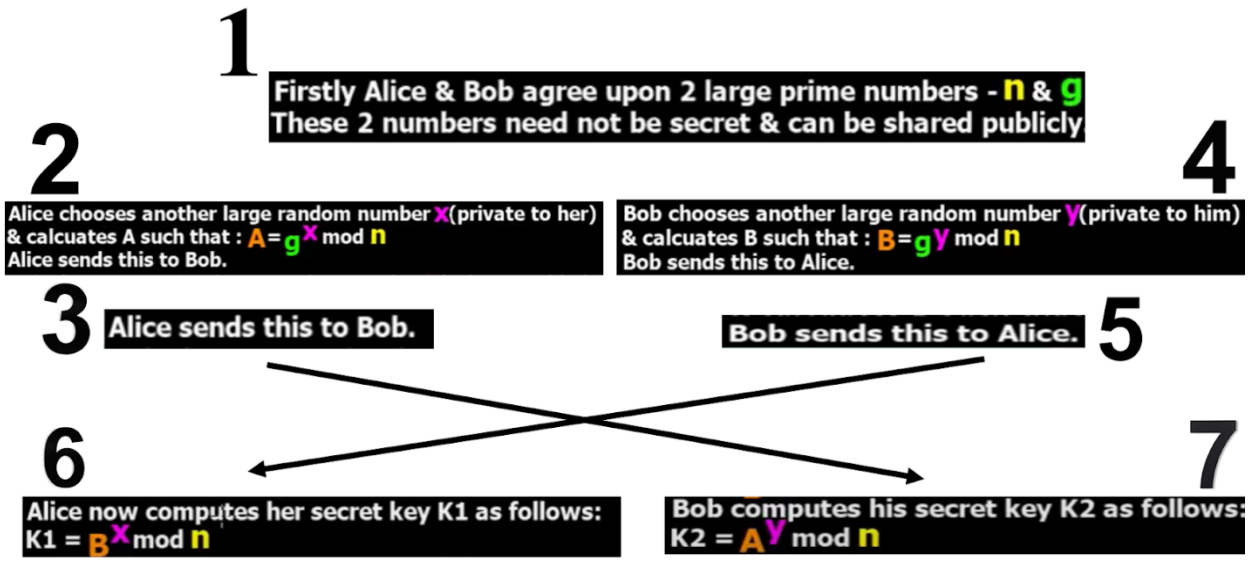


Secure Hash Algorithm 1 (SHA-1).

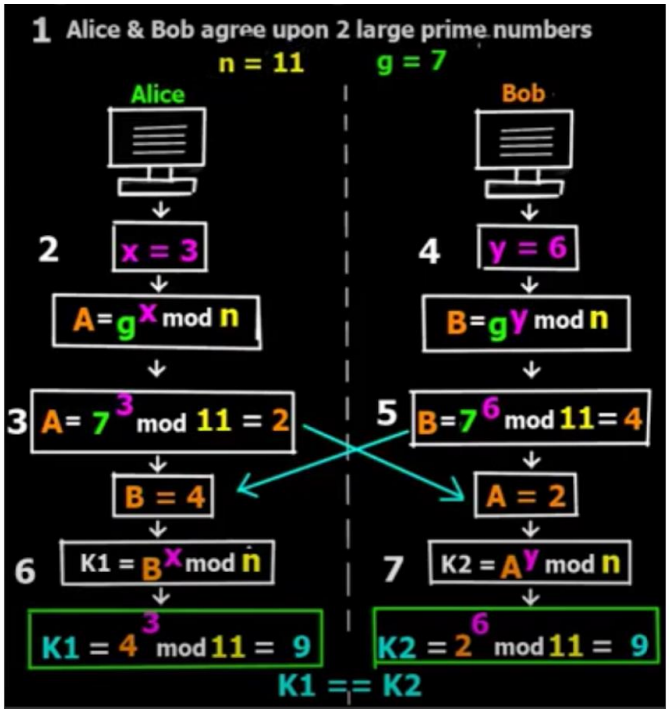
Secure Hash Algorithm 1 (SHA-1). Developed by the U.S. government in the 1990s, SHA-1 used techniques like those of MD5 in the design of message-digest algorithms. But SHA-1 generated more secure 160-bit values when compared to MD5's 128-bit hash value lengths. Despite this, SHA-1 had some weaknesses and did not prove to be the ultimate algorithmic methodology for encryption, either. Security concerns began to emerge, prompting companies like Microsoft to discontinue support for SHA-1 in its software.

Diffie Hellman Key Exchange Algorithm 2

The scheme says that the two parties, who want to communicate securely, can agree on a symmetric key using this technique. Diffie-Hellman key exchange algorithm can be used only for key agreement, but not for encryption/decryption. Once both parties agree on the key to be used, they need to use this for actual encryption/decryption of messages further.



K1 = K2 (key exchange complete)



DOS/DDOS Attacks 5

DOS Stands for Denial of service attack.	DDOS Stands for Distributed Denial of service attack.
In Dos attack single system targets the victims system.	In DDos multiple system attacks the victims system..

Victim PC is loaded from the packet of data sent from a single location.	Victim PC is loaded from the packet of data sent from Multiple location.
Dos attack is slower as compared to ddos.	DDos attack is faster than Dos Attack.
Can be blocked easily as only one system is used.	It is difficult to block this attack as multiple devices are sending packets and attacking from multiple locations.

Firewalls, IDS 5

IDS and firewall both are related to network security but an IDS differs from a firewall as a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls restrict access between networks to prevent intrusion and if an attack is from inside the network it doesn't signal. An IDS describes a suspected intrusion once it has happened and then signals an alarm.