Unit- III:

Message Authentication, Digital Signature, Key Management, Key Exchange, Hash Function. Universal Hashing, Cryptographic Hash Function, MD, Secure Hash Algorithm (SHA), Digital Signature Standard (DSS), Cryptanalysis: Time-Memory Trade-off Attack, Differential Cryptanalysis. Secure channel and authentication system like Kerberos.

• Message authentication ensures that the message has been sent by a genuine identity and not by an imposter.

• The service used to provide message authentication is a Message Authentication Code (MAC).

• A MAC uses a keyed hash function that includes the symmetric key between the sender and receiver when creating the digest.

• Figure shows how a sender A uses a keyed hash function to authenticate his message and how the receiver B can verify the authenticity of the message.

• This system makes use of a symmetric key shared by A and B.

• A, using this symmetric key and a keyed hash function, generates a MAC.

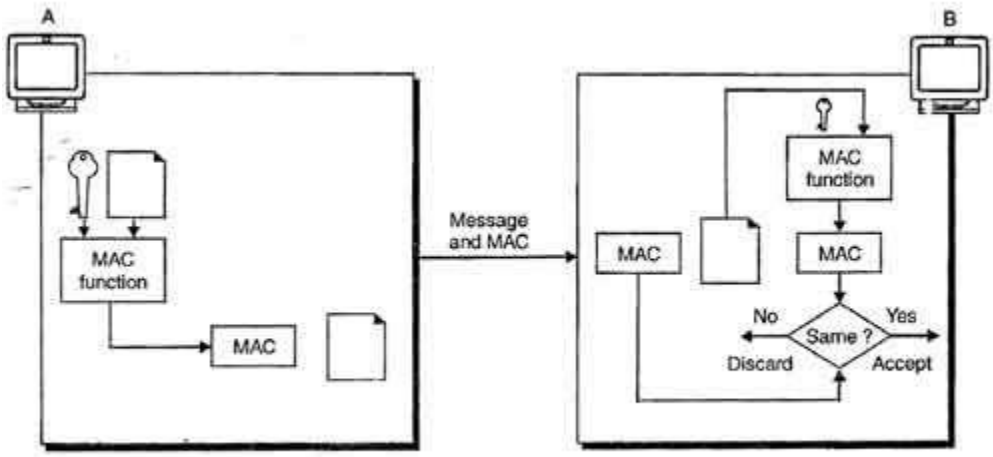
• A then sends this MAC along with the original message to B.

• B receives the message and the MAC and separates the message from the MAC.

• B then applies the same keyed hash function to the message using the same symmetric key to get a fresh MAC.

• B then compares the MAC sent by A with the newly generated MAC.

• If the two MACs are identical, it shows that the message has not been modified and the sender of the message is definitely A.

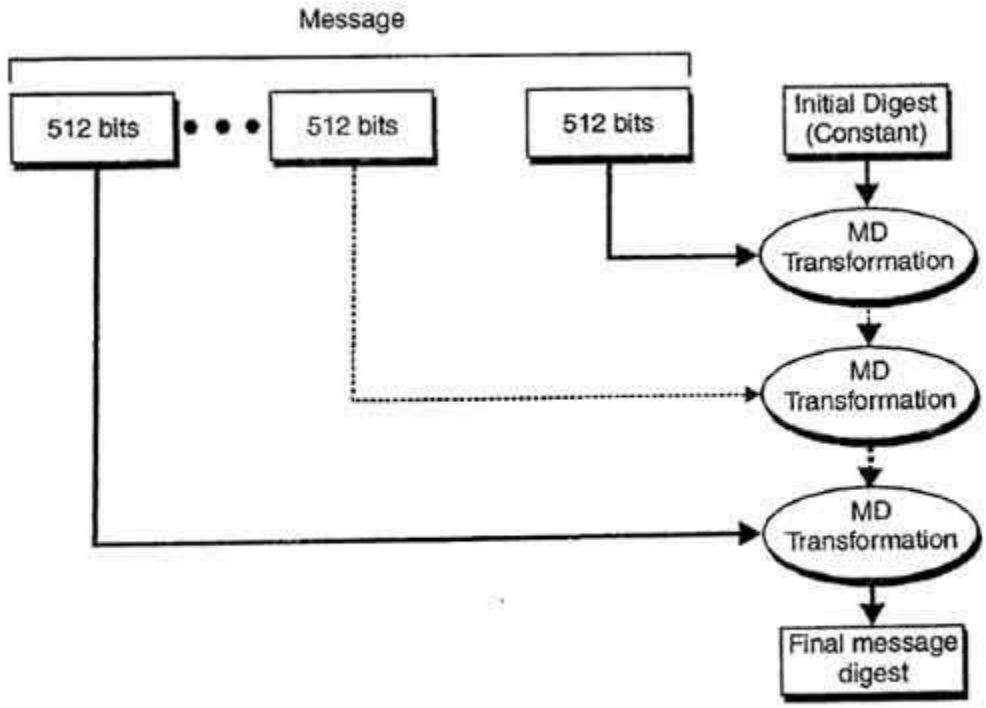


MAC created by A and checked by B.

• There are a number of popular message digest algorithms known as MDn for various values of n.

• MD5 is the most popular and is fifth in a series of message digests designed by Ronald Rivest.

• The basic operation of MD5 is shown in fig.



Generation of message digest using MD5.

• This algorithm operates on message 512 bits at a time.

• Messages not multiple of 512 bits are padded with:

1. A string consisting of 1 followed by zeroes, and

2. 64-bit integer that indicates the length of original message, to make the length of the composite message multiples of 512 bits.

• The message digest calculation begins with a digest value initialized to a constant.

• This value is combined with the first 512 bits of the message to produce a new value for the digest.

• The new value is than combined with the next 512 bits of the message using the same transformation.

• This process is repeated on each 512-bit block till the final value of digest is obtained from the last block of the message.

• The digest is 128-bit long for any message length.

Threats to Data Integrity

When sensitive information is exchanged, the receiver must have the assurance that the message has come intact from the intended sender and is not modified inadvertently or otherwise. There are two different types of data integrity threats, namely **passive** and **active**.

Passive Threats

This type of threats exists due to accidental changes in data.

- These data errors are likely to occur due to noise in a communication channel. Also, the data may get corrupted while the file is stored on a disk.
- Error-correcting codes and simple checksums like Cyclic Redundancy Checks (CRCs) are used to detect the loss of data integrity. In these techniques, a digest of data is computed mathematically and appended to the data.

  11101 -> 11001

Active Threats

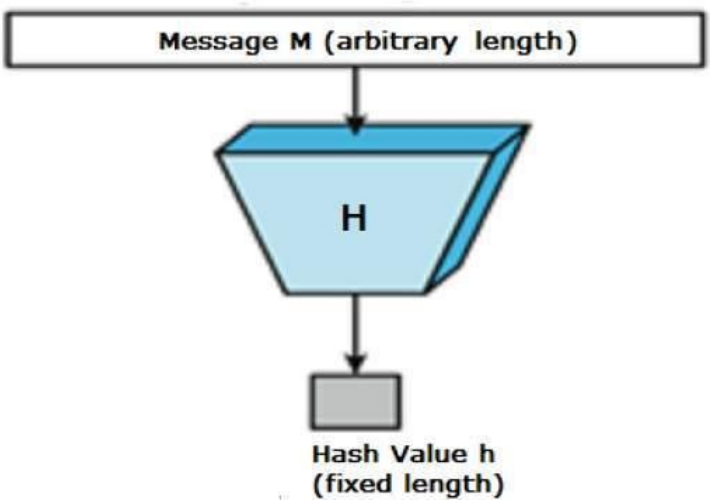
In this type of threats, an attacker can manipulate the data with malicious intent.

- At simplest level, if data is without digest, it can be modified without detection. The system can use techniques of appending CRC to data for detecting any active modification.
- At higher level of threat, attacker may modify data and try to derive new digest for modified data from exiting digest. This is possible if the digest is computed using simple mechanisms such as CRC.
- Security mechanism such as Hash functions are used to tackle the active modification threats.

Hash functions are extremely useful and appear in almost all information security applications.

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.

Values returned by a hash function are called **message digest** or simply **hash values**. The following picture illustrated hash function −



Features of Hash Functions

The typical features of hash functions are −

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

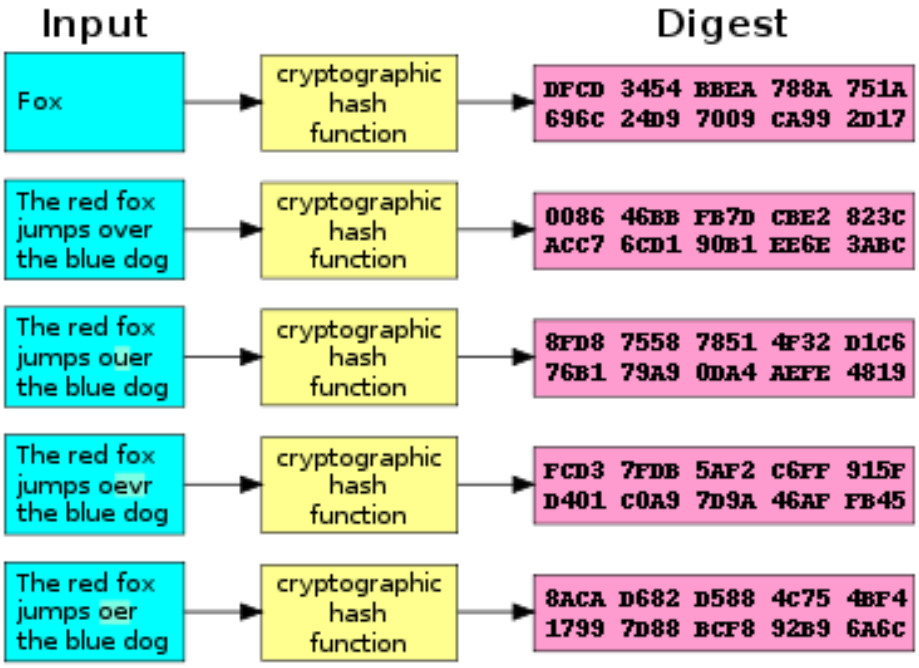
Properties of Hash Functions

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

- **Pre-Image Resistance**
    - This property means that it should be computationally hard to reverse a hash function.
    - In other words, if a hash function h produced a hash value z, then it should be a difficult process to find any input value x that hashes to z.

- This property protects against an attacker who only has a hash value and is trying to find the input.

- **Second Pre-Image Resistance**
  - This property means given an input and its hash, it should be hard to find a different input with the same hash.



  - In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find any other input value y such that h(y) = h(x).
  - This property of hash function protects against an attacker who has an input value and its hash, and wants to substitute different value as legitimate value in place of original input value.

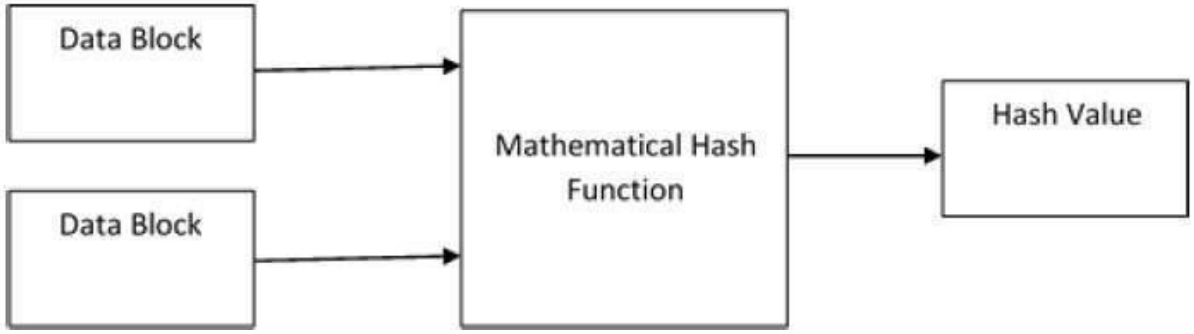
- **Collision Resistance**
  - This property means it should be hard to find two different inputs of any length that result in the same hash. This property is also referred to as collision free hash function.
  - In other words, for a hash function h, it is hard to find any two different inputs x and y such that h(x) = h(y).
  - Since, hash function is compressing function with fixed hash length, it is impossible for a hash function not to have collisions. This property of collision free only confirms that these collisions should be hard to find.
  - This property makes it very difficult for an attacker to find two input values with the same hash.

- Also, if a hash function is collision-resistant **then it is second pre-image resistant.**

Design of Hashing Algorithms

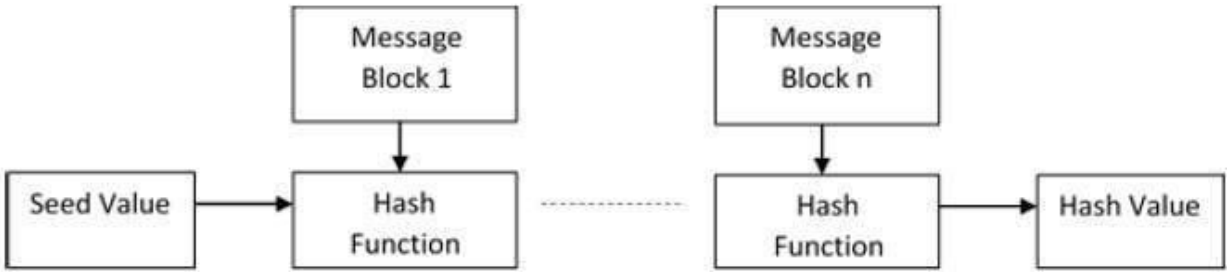
At the heart of a hashing is a mathematical function that operates on two fixed-size blocks of data to create a hash code. This hash function forms the part of the hashing algorithm.

The size of each data block varies depending on the algorithm. Typically the block sizes are from 128 bits to 512 bits. The following illustration demonstrates hash function −



Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for as many rounds as are required to hash the entire message. Schematic of hashing algorithm is depicted in the following illustration −



Since, the hash value of first message block becomes an input to the second hash operation, output of which alters the result of the third operation, and so on. This effect, known as an **avalanche** effect of hashing. Avalanche Effect: This means that every minor change in the message results in a major change in the hash value.

Avalanche effect results in substantially different hash values for two messages that differ by even a single bit of data.

Understand the difference between hash function and algorithm correctly. The hash function generates a hash code by operating on two blocks of fixed-length binary data.

Hashing algorithm is a process for using the hash functions, specifying how the message will be broken up and how the results from previous message blocks are chained together.

Popular Hash Functions

Let us briefly see some popular hash functions −

Message Digest (MD)

MD5 was most popular and widely used hash function for quite some years.

- The MD family comprises of hash functions MD2, MD4, MD5 and MD6. It was adopted as Internet Standard RFC 1321. It is a 128-bit hash function.
- MD5 digests have been widely used in the software world to provide assurance about integrity of transferred file. For example, file servers often provide a pre-computed MD5 checksum for the files, so that a user can compare the checksum of the downloaded file to it.
- In 2004, collisions were found in MD5. An analytical attack was reported to be successful only in an hour by using computer cluster. This collision attack resulted in compromised MD5 and hence it is no longer recommended for use.

Secure Hash Function (SHA)

Family of SHA comprise of four SHA algorithms; SHA-0, SHA-1, SHA-2, and SHA-3. Though from same family, there are structurally different.

- The original version is SHA-0, a 160-bit hash function, was published by the National Institute of Standards and Technology (NIST) in 1993. It had few weaknesses and did not become very popular. Later in 1995, SHA-1 was designed to correct alleged weaknesses of SHA-0.
- SHA-1 is the most widely used of the existing SHA hash functions. It is employed in several widely used applications and protocols including Secure Socket Layer (SSL) security.
- In 2005, a method was found for uncovering collisions for SHA-1 within practical time frame making long-term employability of SHA-1 doubtful.
- SHA-2 family has four further SHA variants, SHA-224, SHA-256, SHA-384, and SHA-512 depending up on number of bits in their hash value. No successful attacks have yet been reported on SHA-2 hash function.
- Though SHA-2 is a strong hash function. Though significantly different, its basic design is still follows design of SHA-1. Hence, NIST called for new competitive hash function designs.
- In October 2012, the NIST chose the Keccak algorithm as the new SHA-3 standard. Keccak offers many benefits, such as efficient performance and good resistance for attacks.

RIPEMD

The RIPEMD is an acronym for RACE Integrity Primitives Evaluation Message Digest. This set of hash functions was designed by open research community and generally known as a family of European hash functions.

- The set includes RIPEMD, RIPEMD-128, and RIPEMD-160. There also exist 256, and 320-bit versions of this algorithm.
- Original RIPEMD (128 bit) is based upon the design principles used in MD4 and found to provide questionable security. RIPEMD 128-bit version came as a quick fix replacement to overcome vulnerabilities on the original RIPEMD.
- RIPEMD-160 is an improved version and the most widely used version in the family. The 256 and 320-bit versions reduce the chance of accidental collision, but do not have higher levels of security as compared to RIPEMD-128 and RIPEMD-160 respectively.

Whirlpool

This is a 512-bit hash function.

- It is derived from the modified version of Advanced Encryption Standard (AES). One of the designers was Vincent Rijmen, a co-creator of the AES.
- Three versions of Whirlpool have been released; namely WHIRLPOOL-0, WHIRLPOOL-T, and WHIRLPOOL.

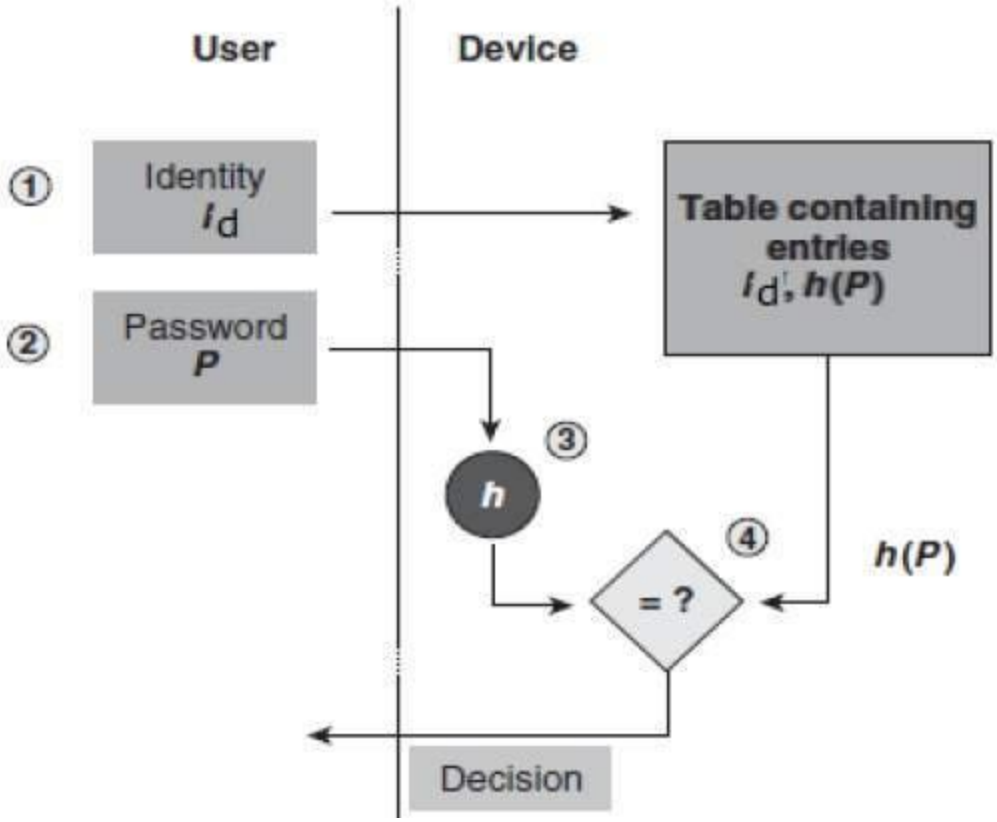
Applications of Hash Functions

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

Password Storage

Hash functions provide protection to password storage.

- Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.
- The Password file consists of a table of pairs which are in the form (user id, h(P)).
- The process of logon is depicted in the following illustration −

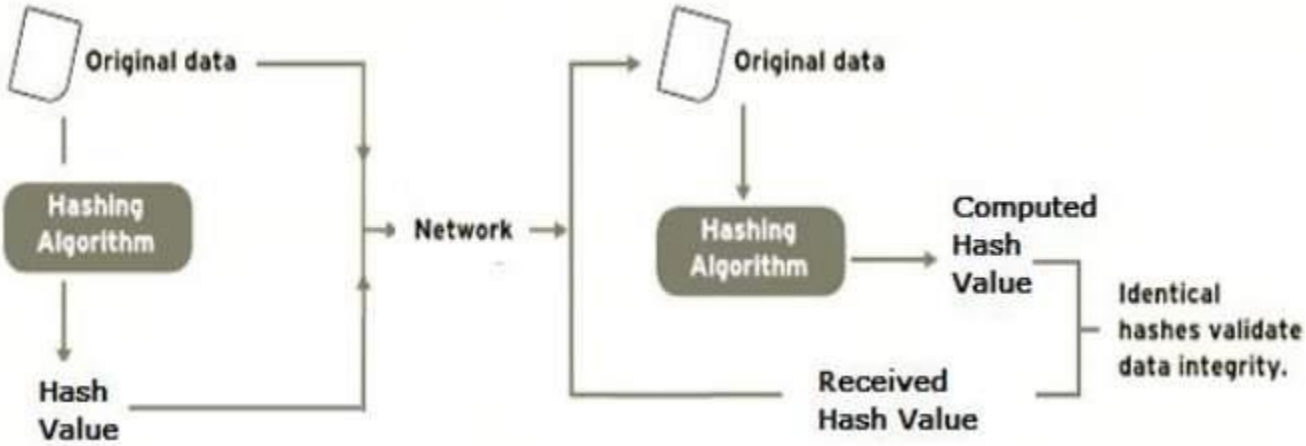


- An intruder can only see the hashes of passwords, even if he accessed the password. He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

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.

The process is depicted in the following illustration −



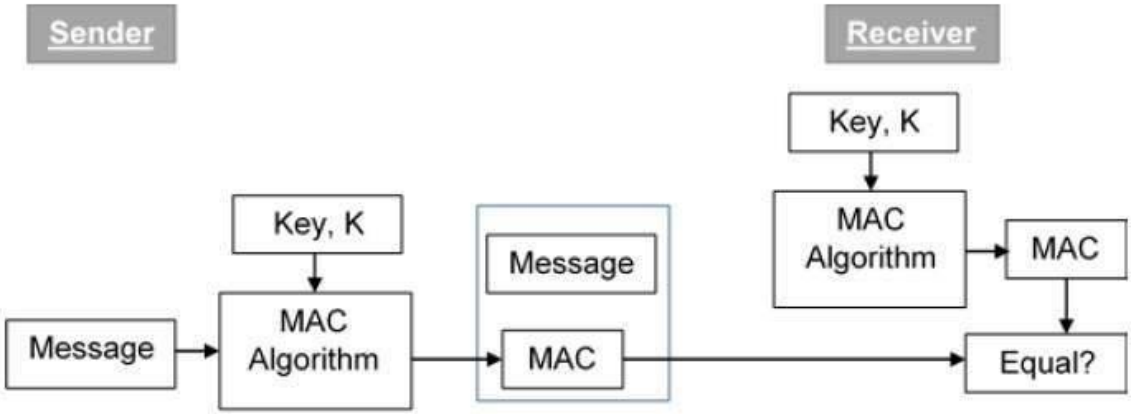
The integrity check helps the user to detect any changes made to original file. It however, does not provide any assurance about originality. The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver. This integrity check application is useful only if the user is sure about the originality of file.

Message Authentication Code (MAC)

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

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

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



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

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.
- The sender forwards the message along with the MAC. Here, we assume that the message is sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.
- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Limitations of MAC

There are two major limitations of MAC, both due to its symmetric nature of operation −

- **Establishment of Shared Secret.**
    - It can provide message authentication among pre-decided legitimate users who have shared key.
    - This requires establishment of shared secret prior to use of MAC.
- **Inability to Provide Non-Repudiation**
    - Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
    - MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.
    - Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

Both these limitations can be overcome by using the public key based digital signatures discussed in following section.

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software or digital document. It's the digital equivalent of a handwritten signature or stamped seal, but it offers far more inherent security. A digital signature is intended to solve the problem of tampering and impersonation in digital communications.

Digital signatures can provide evidence of origin, identity and status of electronic documents, transactions or digital messages. Signers can also use them to acknowledge informed consent.

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

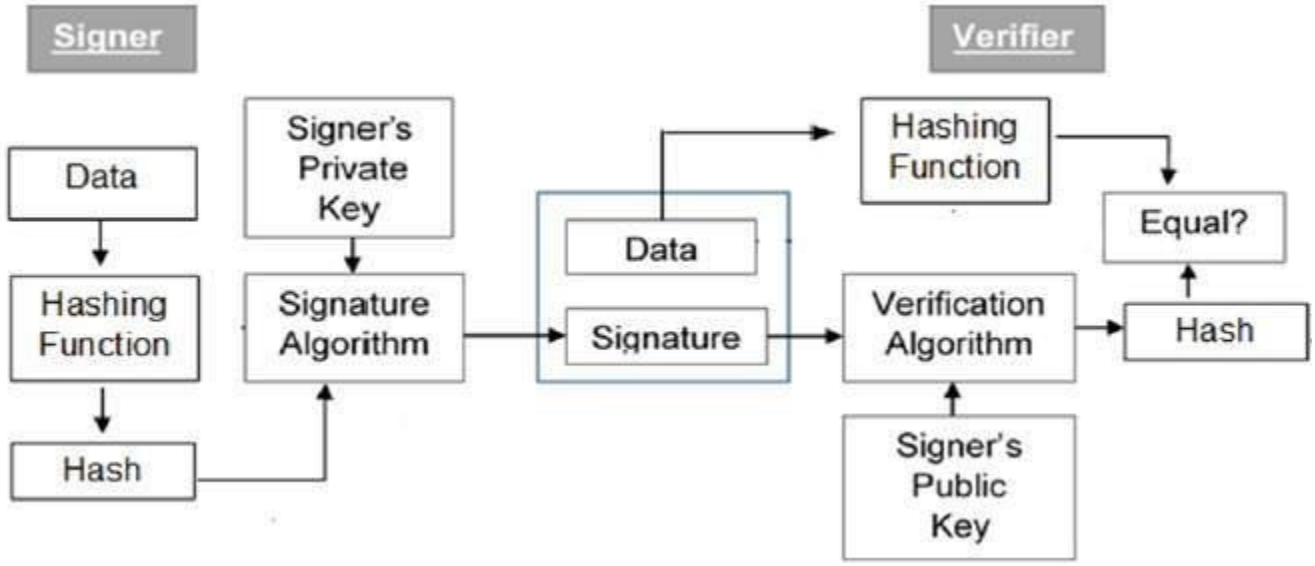
Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party.

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

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

Model of Digital Signature

The digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration −



The following points explain the entire process in detail −

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

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

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

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

Importance of Digital Signature

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

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

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

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

Encryption with Digital Signature

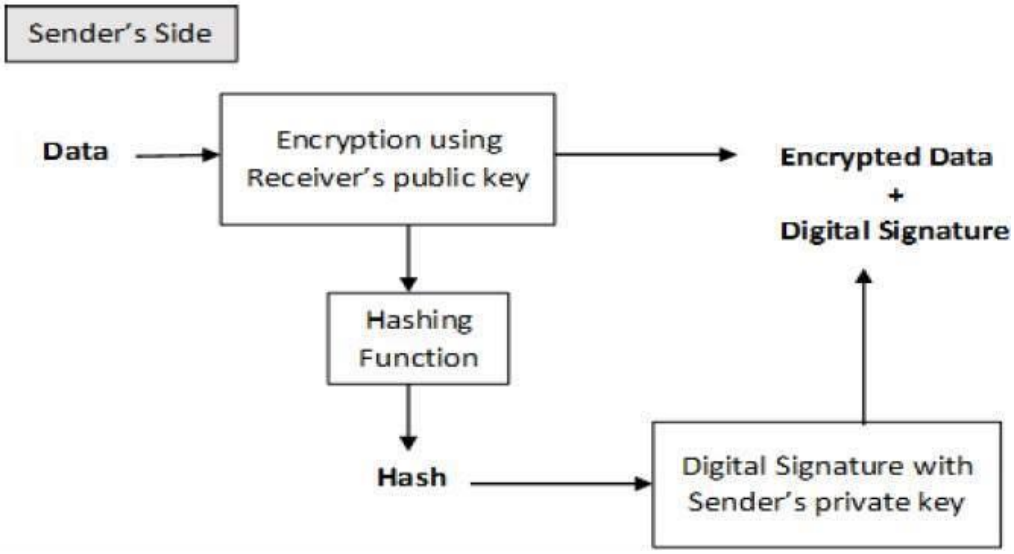
In many digital communications, it is desirable to exchange an encrypted messages than plaintext to achieve confidentiality. In public key encryption scheme, a public (encryption) key of sender is available in open domain, and hence anyone can spoof his identity and send any encrypted message to the receiver.

This makes it essential for users employing PKC for encryption to seek digital signatures along with encrypted data to be assured of message authentication and non-repudiation.

This can archived by combining digital signatures with encryption scheme. Let us briefly discuss how to achieve this requirement. There are **two possibilities, sign-then-encrypt** and **encrypt-then-sign**.

However, the crypto system based on sign-then-encrypt can be exploited by receiver to spoof identity of sender and sent that data to third party. Hence, this method is not preferred. The

process of encrypt-then-sign is more reliable and widely adopted. This is depicted in the following illustration −



The receiver after receiving the encrypted data and signature on it, first verifies the signature using sender's public key. After ensuring the validity of the signature, he then retrieves the data through decryption using his private key.

The most distinct feature of Public Key Infrastructure (PKI) is that it uses a pair of keys to achieve the underlying security service. The key pair comprises of private key and public key.

Since the public keys are in open domain, they are likely to be abused. It is, thus, necessary to establish and maintain some kind of trusted infrastructure to manage these keys.

What are the benefits of digital signatures?

Security is the main benefit of digital signatures. Security capabilities embedded in digital signatures ensure a document is not altered and signatures are legitimate. Security features and methods used in digital signatures include the following:

Personal identification numbers (PINs), passwords and codes. Used to authenticate and verify a signer's identity and approve their signature. Email, username and password are the most common methods used.

Asymmetric cryptography. Employs a public key algorithm that includes private and public key encryption and authentication.

Checksum. A long string of letters and numbers that represents the sum of the correct digits in a piece of digital data, against which comparisons can be made to detect errors or changes. A checksum acts as a data fingerprint.

Cyclic redundancy check (CRC). An error-detecting code and verification feature used in digital networks and storage devices to detect changes to raw data.

Certificate authority (CA) validation. CAs issue digital signatures and act as trusted third parties by accepting, authenticating, issuing and maintaining digital certificates. The use of CAs helps avoid the creation of fake digital certificates.

Trust service provider (TSP) validation. A TSP is a person or legal entity that performs validation of a digital signature on a company's behalf and offers signature validation reports.

Other benefits to using digital signatures include the following:

Timestamping. By providing the data and time of a digital signature, timestamping is useful when timing is critical, such as for stock trades, lottery ticket issuance and legal proceedings.

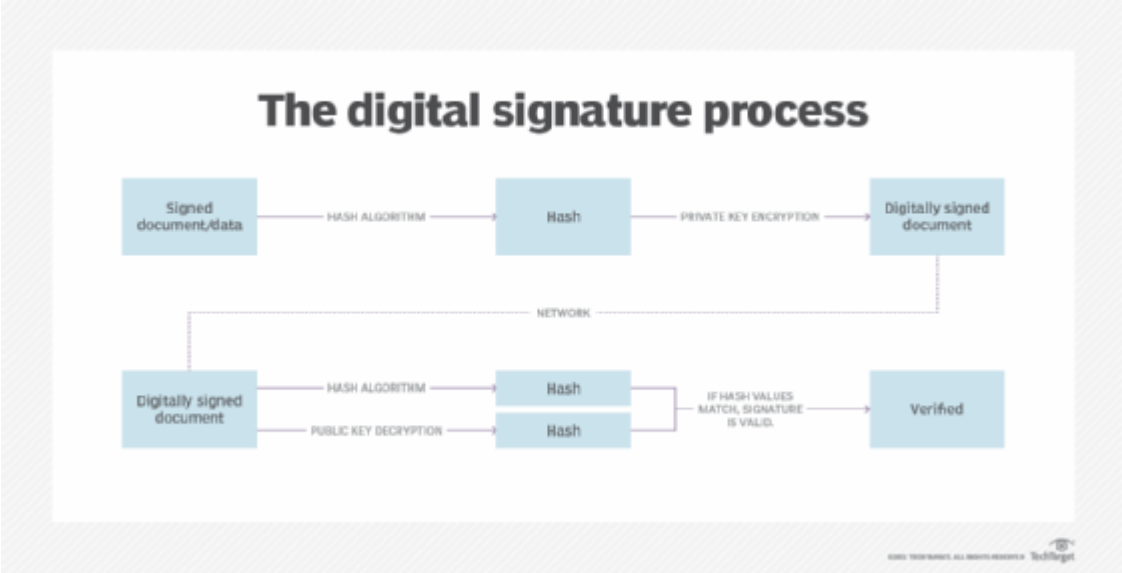
Globally accepted and legally compliant. The public key infrastructure (PKI) standard ensures vendor-generated keys are made and stored securely. Because of the international standard, a growing number of countries are accepting digital signatures as legally binding.

Time savings. Digital signatures simplify the time-consuming processes of physical document signing, storage and exchange, enabling businesses to quickly access and sign documents.

Cost savings. Organizations can go paperless and save money previously spent on the physical resources and on the time, personnel and office space used to manage and transport them.

Positive environmental impact. Reducing paper use also cuts down on the physical waste generated by paper and the negative environmental impact of transporting paper documents.

Traceability. Digital signatures create an audit trail that makes internal record-keeping easier for business. With everything recorded and stored digitally, there are fewer opportunities for a manual signee or record-keeper to make a mistake or misplace something.



Classes and types of digital signatures

There are three different classes of digital signature certificates (DSCs):

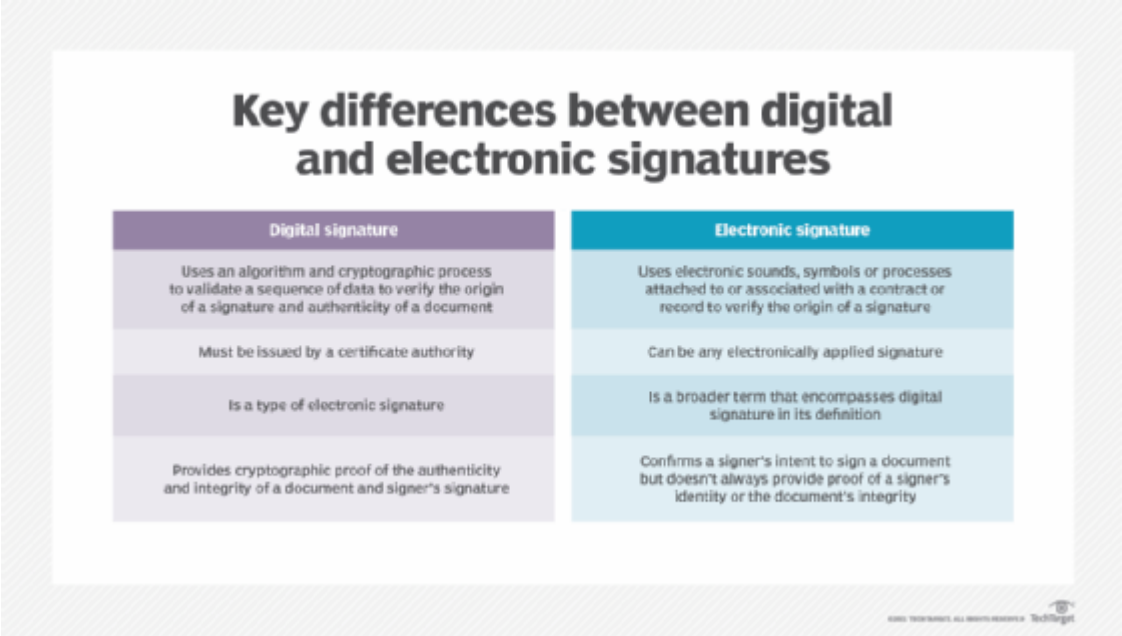
Class 1. Cannot be used for legal business documents as they are validated based only on an email ID and username. Class 1 signatures provide a basic level of security and are used in environments with a low risk of data compromise.

Class 2. Often used for electronic filing (e-filing) of tax documents, including income tax returns and goods and services tax (GST) returns. Class 2 digital signatures authenticate a signer's identity against a pre-verified database. Class 2 digital signatures are used in environments where the risks and consequences of data compromise are moderate.

Class 3. The highest level of digital signatures, Class 3 signatures require a person or organization to present in front of a certifying authority to prove their identity before signing. Class 3 digital signatures are used for e-auctions, e-tendering, e-ticketing, court filings and in other environments where threats to data or the consequences of a security failure are high.

What's the difference between a digital signature and an electronic signature?

Digital signature is a technical term, defining the result of a cryptographic process or mathematical algorithm that can be used to authenticate a sequence of data. The term electronic signature -- or e-signature -- is a legal term that is defined legislatively.



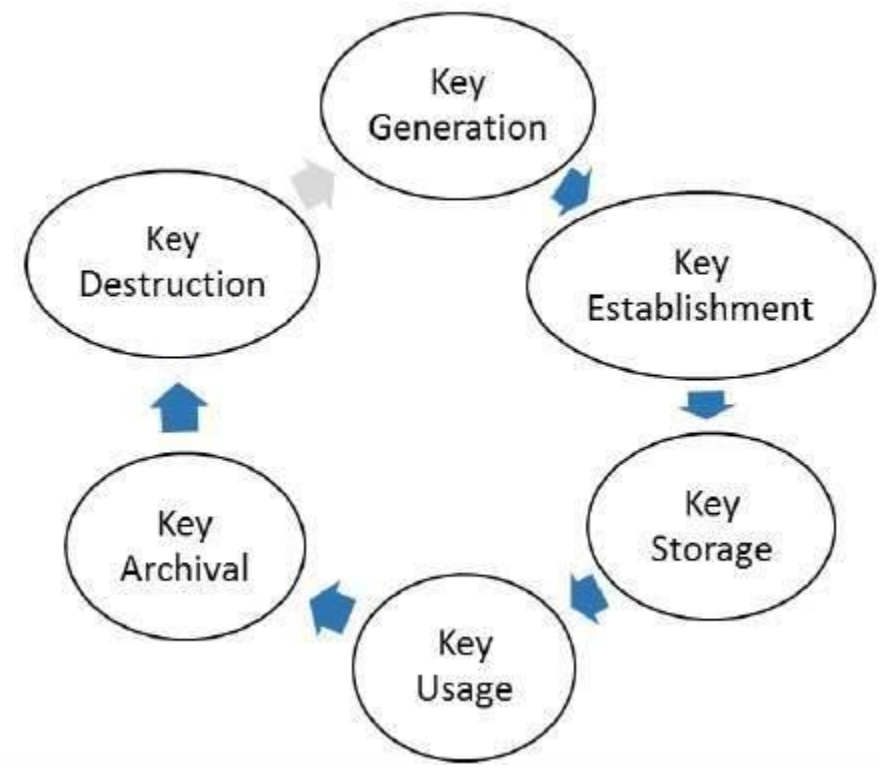
Key Management

It goes without saying that the security of any cryptosystem depends upon how securely its keys are managed. Without secure procedures for the handling of cryptographic keys, the benefits of the use of strong cryptographic schemes are potentially lost.

It is observed that cryptographic schemes are rarely compromised through weaknesses in their design. However, they are often compromised through poor key management.

There are some important aspects of key management which are as follows −

- Cryptographic keys are nothing but special pieces of data. Key management refers to the secure administration of cryptographic keys.
- Key management deals with entire key lifecycle as depicted in the following illustration –



- There are two specific requirements of key management for public key cryptography.
    - **Secrecy of private keys.** Throughout the key lifecycle, secret keys must remain secret from all parties except those who are owner and are authorized to use them.
    - **Assurance of public keys.** In public key cryptography, the public keys are in open domain and seen as public pieces of data. By default there are no assurances of whether a public key is correct, with whom it can be associated, or what it can be used for. Thus key management of public keys needs to focus much more explicitly on assurance of purpose of public keys.

The most crucial requirement of 'assurance of public key' can be achieved through the public-key infrastructure (PKI), a key management systems for supporting public-key cryptography.

Public Key Infrastructure (PKI)

PKI provides assurance of public key. It provides the identification of public keys and their distribution. An anatomy of PKI comprises of the following components.

- Public Key Certificate, commonly referred to as 'digital certificate'.

- Private Key tokens.
- Certification Authority.
- Registration Authority.
- Certificate Management System.

Digital Certificate

For analogy, a certificate can be considered as the ID card issued to the person. People use ID cards such as a driver's license, passport to prove their identity. A digital certificate does the same basic thing in the electronic world, but with one difference.

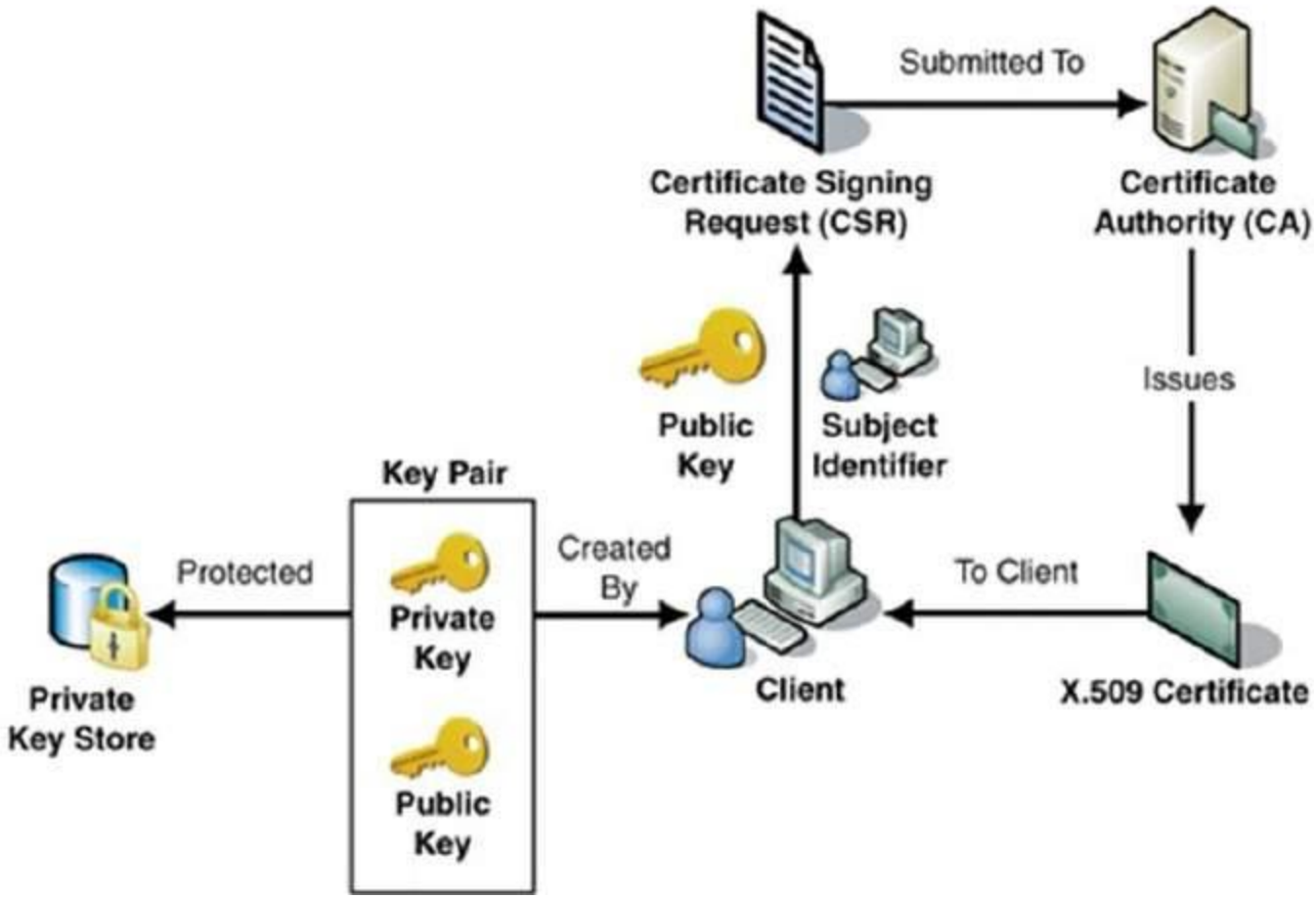
Digital Certificates are not only issued to people but they can be issued to computers, software packages or anything else that need to prove the identity in the electronic world.

- Digital certificates are based on the ITU standard X.509 which defines a standard certificate format for public key certificates and certification validation. Hence digital certificates are sometimes also referred to as X.509 certificates.

  Public key pertaining to the user client is stored in digital certificates by The Certification Authority (CA) along with other relevant information such as client information, expiration date, usage, issuer etc.

- CA digitally signs this entire information and includes digital signature in the certificate.
- Anyone who needs the assurance about the public key and associated information of client, he carries out the signature validation process using CA’s public key. Successful validation assures that the public key given in the certificate belongs to the person whose details are given in the certificate.

The process of obtaining Digital Certificate by a person/entity is depicted in the following illustration.



As shown in the illustration, the CA accepts the application from a client to certify his public key. The CA, after duly verifying identity of client, issues a digital certificate to that client.

Certifying Authority (CA)

As discussed above, the CA issues certificate to a client and assist other users to verify the certificate. The CA takes responsibility for identifying correctly the identity of the client asking for a certificate to be issued, and ensures that the information contained within the certificate is correct and digitally signs it.

Key Functions of CA

The key functions of a CA are as follows −

- **Generating key pairs** − The CA may generate a key pair independently or jointly with the client.
- **Issuing digital certificates** − The CA could be thought of as the PKI equivalent of a passport agency − the CA issues a certificate after client provides the credentials to confirm his identity. The CA then signs the certificate to prevent modification of the details contained in the certificate.

- **Publishing Certificates** − The CA need to publish certificates so that users can find them. There are two ways of achieving this. One is to publish certificates in the equivalent of an electronic telephone directory. The other is to send your certificate out to those people you think might need it by one means or another.
- **Verifying Certificates** − The CA makes its public key available in environment to assist verification of his signature on clients' digital certificate.
- **Revocation of Certificates** − At times, CA revokes the certificate issued due to some reason such as compromise of private key by user or loss of trust in the client. After revocation, CA maintains the list of all revoked certificate that is available to the environment.

Classes of Certificates

There are four typical classes of certificate −

- **Class 1** − These certificates can be easily acquired by supplying an email address.
- **Class 2** − These certificates require additional personal information to be supplied.
- **Class 3** − These certificates can only be purchased after checks have been made about the requestor's identity.
- **Class 4** − They may be used by governments and financial organizations needing very high levels of trust.

Registration Authority (RA)

CA may use a third-party Registration Authority (RA) to perform the necessary checks on the person or company requesting the certificate to confirm their identity. The RA may appear to the client as a CA, but they do not actually sign the certificate that is issued.

Certificate Management System (CMS)

It is the management system through which certificates are published, temporarily or permanently suspended, renewed, or revoked. Certificate management systems do not normally delete certificates because it may be necessary to prove their status at a point in time, perhaps for legal reasons. A CA along with associated RA runs certificate management systems to be able to track their responsibilities and liabilities.

Private Key Tokens

While the public key of a client is stored on the certificate, the associated secret private key can be stored on the key owner's computer. This method is generally not adopted. If an attacker gains access to the computer, he can easily gain access to private key. For this reason, a private key is stored on secure removable storage token access to which is protected through a password.

Different vendors often use different and sometimes proprietary storage formats for storing keys. For example, Entrust uses the proprietary .epf format, while Verisign, GlobalSign, and Baltimore use the standard .p12 format.

Hierarchy of CA

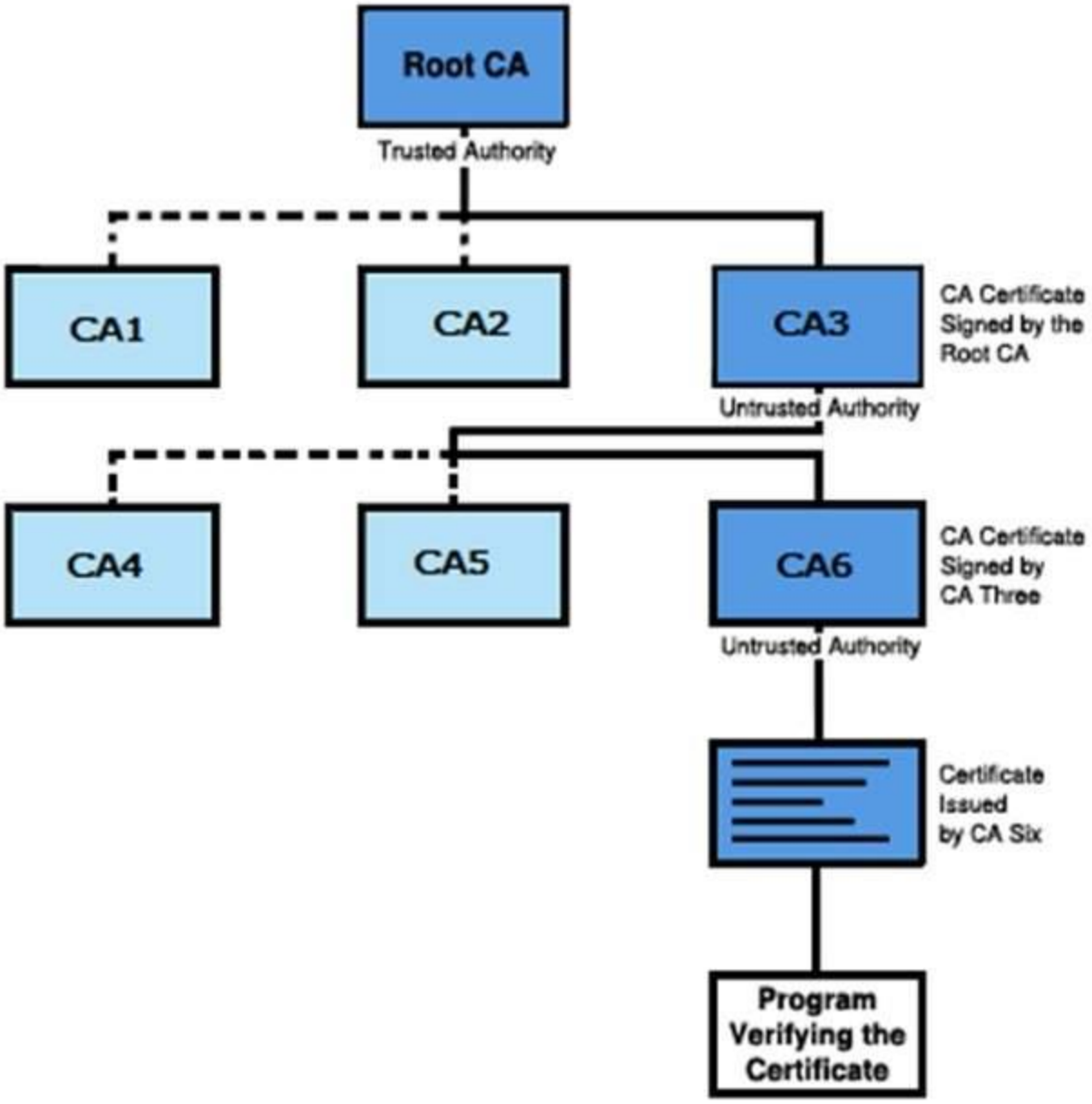
With vast networks and requirements of global communications, it is practically not feasible to have only one trusted CA from whom all users obtain their certificates. Secondly, availability of only one CA may lead to difficulties if CA is compromised.

In such case, the hierarchical certification model is of interest since it allows public key certificates to be used in environments where two communicating parties do not have trust relationships with the same CA.

- The root CA is at the top of the CA hierarchy and the root CA's certificate is a self-signed certificate.
- The CAs, which are directly subordinate to the root CA (For example, CA1 and CA2) have CA certificates that are signed by the root CA.
- The CAs under the subordinate CAs in the hierarchy (For example, CA5 and CA6) have their CA certificates signed by the higher-level subordinate CAs.

Certificate authority (CA) hierarchies are reflected in certificate chains. A certificate chain traces a path of certificates from a branch in the hierarchy to the root of the hierarchy.

The following illustration shows a CA hierarchy with a certificate chain leading from an entity certificate through two subordinate CA certificates (CA6 and CA3) to the CA certificate for the root CA.



Verifying a certificate chain is the process of ensuring that a specific certificate chain is valid, correctly signed, and trustworthy. The following procedure verifies a certificate chain, beginning with the certificate that is presented for authentication −

- A client whose authenticity is being verified supplies his certificate, generally along with the chain of certificates up to Root CA.
- Verifier takes the certificate and validates by using public key of issuer. The issuer's public key is found in the issuer's certificate which is in the chain next to client's certificate.
- Now if the higher CA who has signed the issuer's certificate, is trusted by the verifier, verification is successful and stops here.

- Else, the issuer's certificate is verified in a similar manner as done for client in above steps. This process continues till either trusted CA is found in between or else it continues till Root CA.