# Module 3
# Cryptographic Hashes,MSG Digest and Digital Certificates

## Message Authentication

Data is prone to various attacks. One of these attacks includes message authentication. This threat arises when the user does not have any information about the originator of the message. Message authentication can be achieved using cryptographic methods which further make use of keys.

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by either source or destination.

**Authentication Requirements:**

- **Revelation:** It means releasing the content of the message to someone who does not have an appropriate cryptographic key.
- **Analysis of Traffic:** Determination of the pattern of traffic through the duration of connection and frequency of connections between different parties.
- **Deception:** Adding out of context messages from a fraudulent source into a communication network. This will lead to mistrust between the parties communicating and may also cause loss of critical data.
- **Modification in the Content:** Changing the content of a message. This includes inserting new information or deleting/changing the existing one.
- **Modification in the sequence:** Changing the order of messages between parties. This includes insertion, deletion, and reordering of messages.
- **Modification in the Timings:** This includes replay and delay of messages sent between different parties. This way session tracking is also disrupted.
- **Source Refusal:** When the source denies being the originator of a message.

- **Destination refusal:** When the receiver of the message denies the reception.

**Message Authentication Functions:**

All message authentication and digital signature mechanisms are based on two functionality levels:

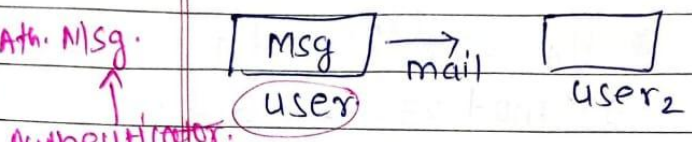
- **Lower level:** At this level, there is a need for a function that produces an authenticator, which is the value that will further help in the authentication of a message.
- **Higher-level:** The lower level function is used here in order to help receivers verify the authenticity of messages.

These message authentication functions are divided into three classes:

# CHAPTER-3.

(2) Authentication function. (A.F)

with A.F produce the authenticator.
& Authecator produce msg.




Auth. Msg.
Authenticator.
A.F

(1) Msg. Encryptn. - cipher text act as authtor

(2) MAC (msg authentication code)

we have authent.F & applay them on plain text along with key which produce fixed length code called MAC.

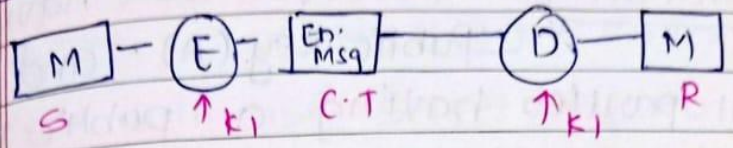
C (M, K) → Fixed length code (MAC)
msg
key.
(A.F)
act as Authecticator

(3) Hash funcⁿ. (H) → indepent of key.

H. value generated: H (M) = Fixed length code (H code)

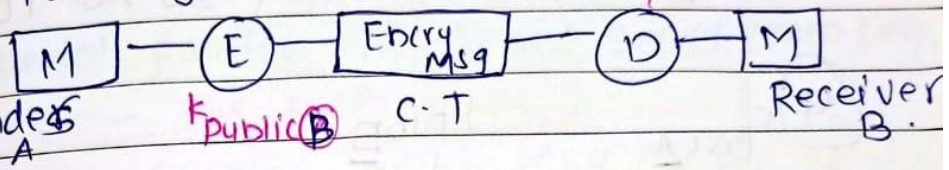
# 1- MSG ENCRYPTION—-(CIPHERTEXT IS AUTHENTICATOR)

(I) Msg Encryptn - Ciphertext is authenti' cation.

(1) 


Both share same key K1

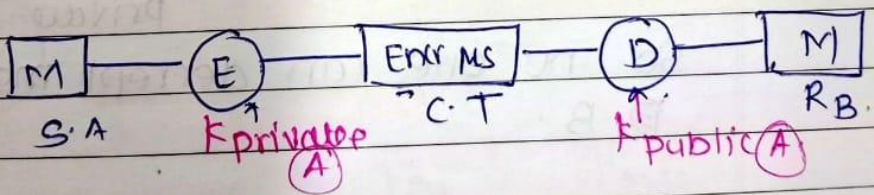
(2) For Assymetric.




[pub/priv] → same key should use ⇒ CO2 It's in pair.

✗ Authection - In this A. Auth not happ. CO2. Receiver B (privte k) dont know who send the encryption msg.

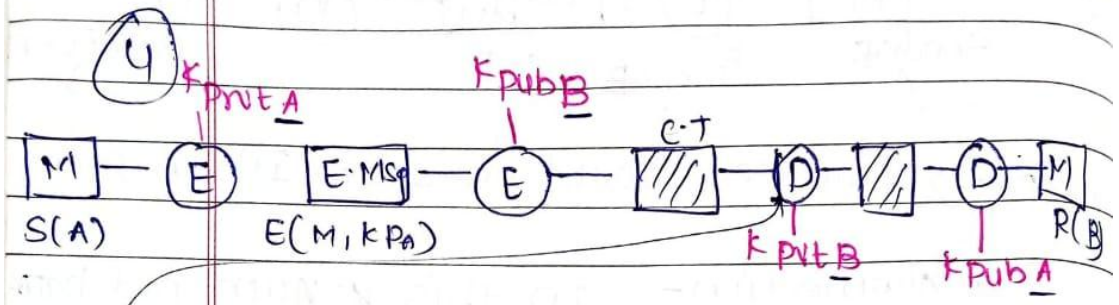
✓ confidentiality - we have to decrept from B. so B have private key
∴ confidentiality is there.

(3) 


✓ Auth ⇒ IF a 3rd another person want to decrypt so they get to know that from which where msg comes.
∴ public (A) ⇒ All user know msg comes from (A.)
so authenticatn is to happen

X Confidentiality – Receiver can't having a Public key (A). and Pub key All parties having a public key. So confidentiality is not their.

* If 100 people use their all can decrept the msg so no confidentiality.




✓ ~~confidect~~ Authentication.

R. having Pub(A) and R. will know if i have Pub(A) so (A) is the sender who have pvt key.

✓ Confidentiality. ⇒ B having only private key So no one can decrept msg accept B.

So we can acheive msg by these 3 way.

## 2- MAC

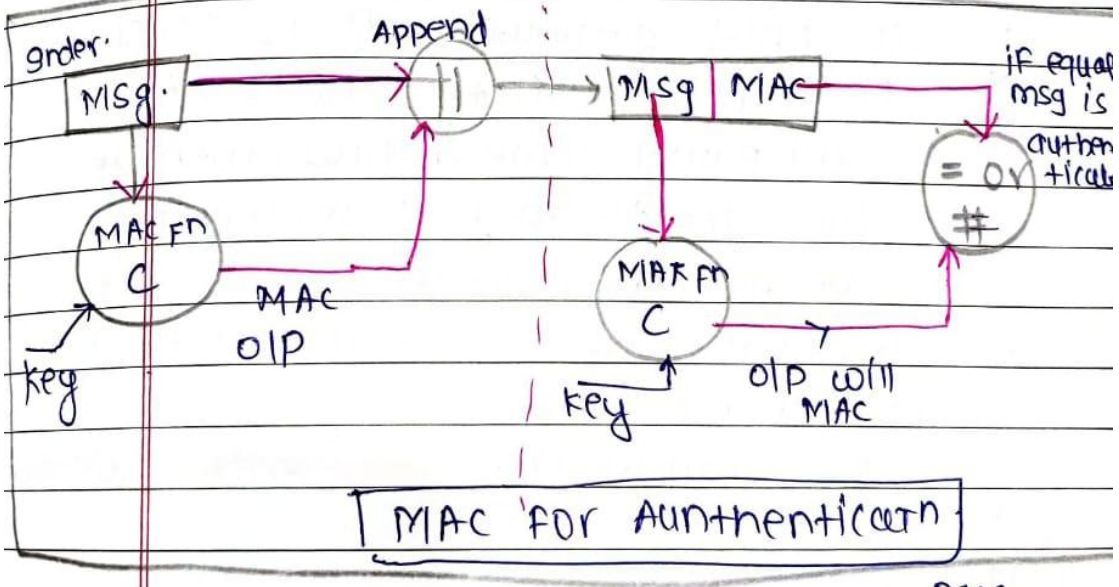
MAC - compress the msg & give out (fixed) (2 MB | MAC - 2 kb).

- we will use secret key to generate a small fixed size of block data called MAC.
- It's then append (join) with msg.
- comm$^n$ parties will share secret comm key.

(I) MAC - FOR Authentication:

* When sender send Msg to B. it calculate MAC as a funct$^n$ of the msg. and the key. MAC = C(K, M)

M = i/p msg
K = shared secret key
C = MAC funct$^n$.




MAC for Authentication

(1) A sender having msg & sender pass a key through MAC funct$^n$ so get MAC o/p

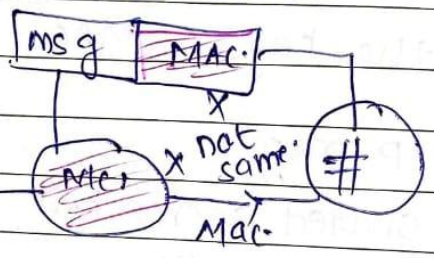
(2) then append Msg & MAC o/p together. and we get Msg & MAC as o/p

(3) Both S & R having cmn key.

(3) Again msg Receiver will pass msg with key thorugh MAC Fun(th) it will get O/P (MAC)

(4) i's MAC & MAC are same it means msg o's same it is not exchange in betn. called it's authentic

⇒ when not authenticate.

if 

both X same then.

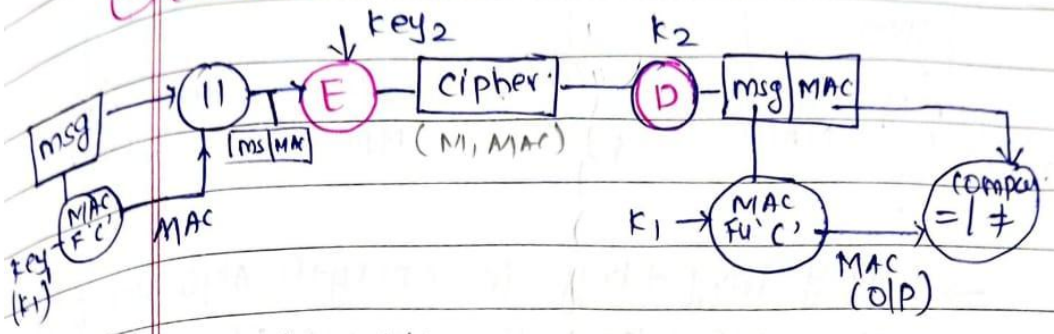
(1) so only authenti can is acheive.

(2) NO confidentiality beco2 if 3rd party come in blw, then he can get the msg. ∴ no security.
(we not providing en/dec we only pass msg throug MAC) so chance is 3rd party involve so no authenti cate

II

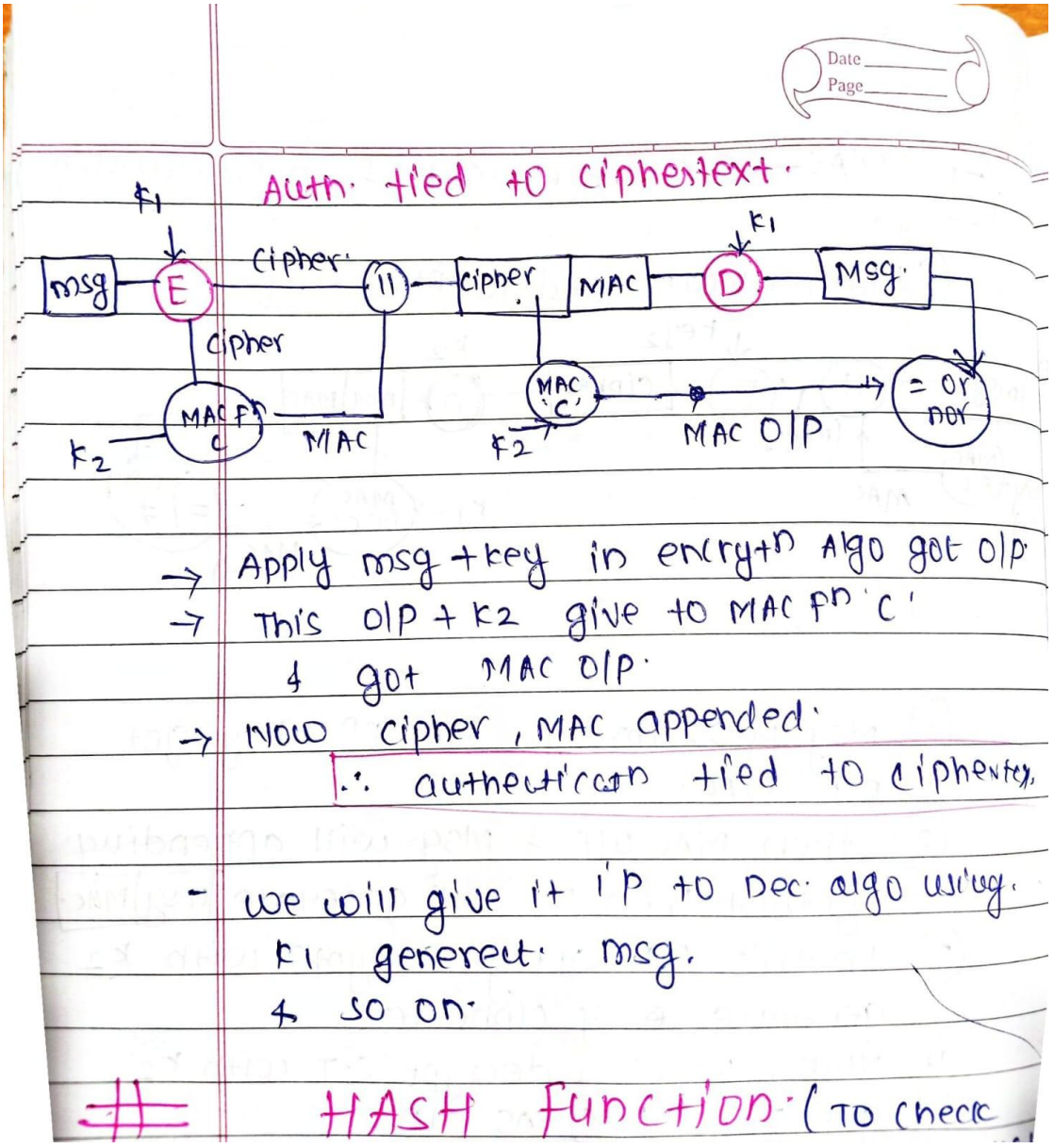
(II) MAC – For Authentication & confidentiality.

(a) authentication tied to plaintext.




(1) Msg pass through MAC Fn & key got o|p MAC.
(2) then MAC o|p & Msg will appending together then o|p will generate msg | MAC.
(3) In this o|p apply Encrypth with $k_2$ generate ◎ cy ciphertext.
(4) Now, Receiver decrypt C.T with $k_2$ got Msg | MAC as o|p.

(5) Receiver pass key ◎ $k_1$ through MAC Fn'C get o|p MAC. → compare both MAC
(6) If both MAC same then it' mean's msg is authticate.
(7) Here




**3- HASH FUNCTION**

De/En= confidentiality.

OP => fixed length.

charect:

(1) Non Reversible – Psw will not Revers.
fn (Pooja) = KMPSW # WSPMS.

(2) Fix length O/P => If our inpt is 1000 word still it give fixed length (capacity)

(3) weak collision. / NO collision.
if fn (Pooja) = KMPSN

Obvious it will not match with other Psd
if Fn (Poojap) = SPPW...
if only 1 charch change whole o/p will change.

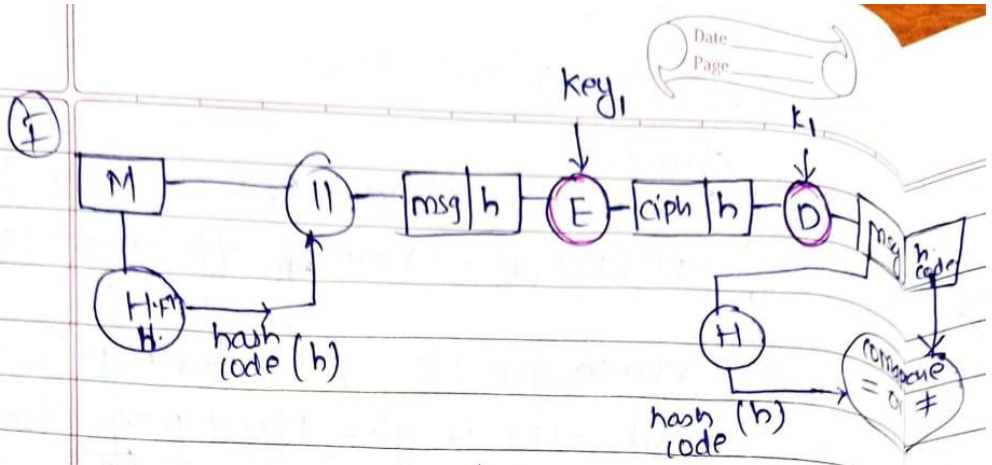
4. (1) Same as MAC but in H.F key are not their.

(2) Take variable size Msg & produce fixed O/P. (called Hash code / H.value.) msg digest)

(3) only i/p (msg) H.F (Hash fun)
h.V (has value)

H.f => called compression Fn.
H.F= fixed length code 'h'
original msg < opt msg.

(4) Their are diff. Method to provide authentication in diff. situation.




= msg is authenticate.
≠ msg is not authenticate.

Authenticate:- Sender / Receive uses same key. (=)
Confident- maintain Encrypth & decrpth to save (secure)

2 Method

Feature
① Fixed length op, ② compression pn
③ Digest (smaller represntn of larger data)

(MD5) - message digest 5.

working.

Padding ⇒ Add pdd → original msg.

Total length = $512 * i - 64$

512 = 1x2x3,
i = 1, 2, 3, 4...

T.L is 64 bit less than exact multiple of 512

let's L = 1000

| | |
|---|---|
| $1000 = 512 \times 2 - 64$ | $1000 = 512 \times 3 - 64$ |
| $1000 = 1024 - 64$ | $1000 = 1536 - 64$ |
| | $= 1472$ |

~~L = 1000 + 472~~

$= 1000 + 472 = 1472$

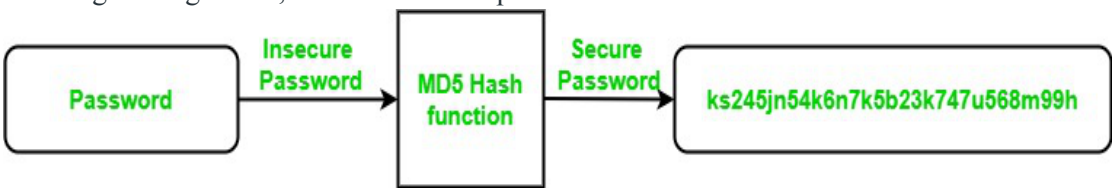
64 bit less than of multiple of 512

# What is the MD5 Algorithm?

**MD5** is a cryptographic hash function algorithm that takes the message as input of any length and changes it into a fixed-length message of 16 bytes. MD5 algorithm stands for the **message-digest algorithm**. MD5 was developed as an improvement of MD4, with advanced security purposes. The output of MD5 (Digest size) is always **128 bits. MD5** was developed in 1991 by **Ronald Rivest.**

**Use Of MD5 Algorithm:**

- It is used for file authentication.
- In a web application, it is used for security purposes. e.g. Secure password of users etc.
- Using this algorithm, We can store our password in 128 bits format.




*MD5 Algorithm*

**Working of the MD5 Algorithm:**

MD5 algorithm follows the following steps

1. **Append Padding Bits:** In the first step, we add padding bits in the original message in such a way that the total length of the message is 64 bits less than the exact multiple of 512.

Suppose we are given a message of 1000 bits. Now we have to add padding bits to the original message. Here we will add 472 padding bits to the original message. After adding the padding bits the size of the original message/output of the first step will be 1472 i.e. 64 bits less than an exact multiple of 512 (i.e. 512*3 = 1536).

**Length(original message + padding bits) = 512 * i – 64** where i = 1,2,3 . . .

**2. Append Length Bits:** In this step, we add the length bit in the output of the first step in such a way that the total number of the bits is the perfect multiple of 512. Simply, here we add the 64-bit as a length bit in the output of the first step.
i.e. output of first step = 512 * n – 64
length bits = 64.

After adding both we will get **512 * n** i.e. the exact multiple of 512.

**3. Initialize MD buffer:** Here, we use the 4 buffers i.e. J, K, L, and M. The size of each buffer is 32 bits.

- J = 0x67425301
- K = 0xEDFCBA45
- L = 0x98CBADFE
- M = 0x13DCE476

**4. Process Each 512-bit Block:** This is the most important step of the MD5 algorithm. Here, a total of 64 operations are performed in 4 rounds. In the 1st round, 16 operations will be performed, 2nd round 16 operations will be performed, 3rd round 16 operations will be performed, and in the 4th round, 16 operations will be performed. We apply a different function on each round i.e. for the 1st round we apply the F function, for the 2nd G function,

3rd for the H function, and 4th for the I function.
We perform OR, AND, XOR, and NOT (basically these are logic gates) for calculating functions. We use 3 buffers for each function i.e. K, L, M.

- F(K,L,M) = (K AND L) OR (NOT K AND M)
- G(K,L,M) = (K AND L) OR (L AND NOT M)
- H(K,L,M) = K XOR L XOR M
- I(K,L,M) = L XOR (K OR NOT M)

After applying the function now we perform an operation on each block. For performing operations we need

- add modulo $2_{32}$
- M[i] – 32 bit message.
- K[i] – 32-bit constant.
- <<<n – Left shift by n bits.

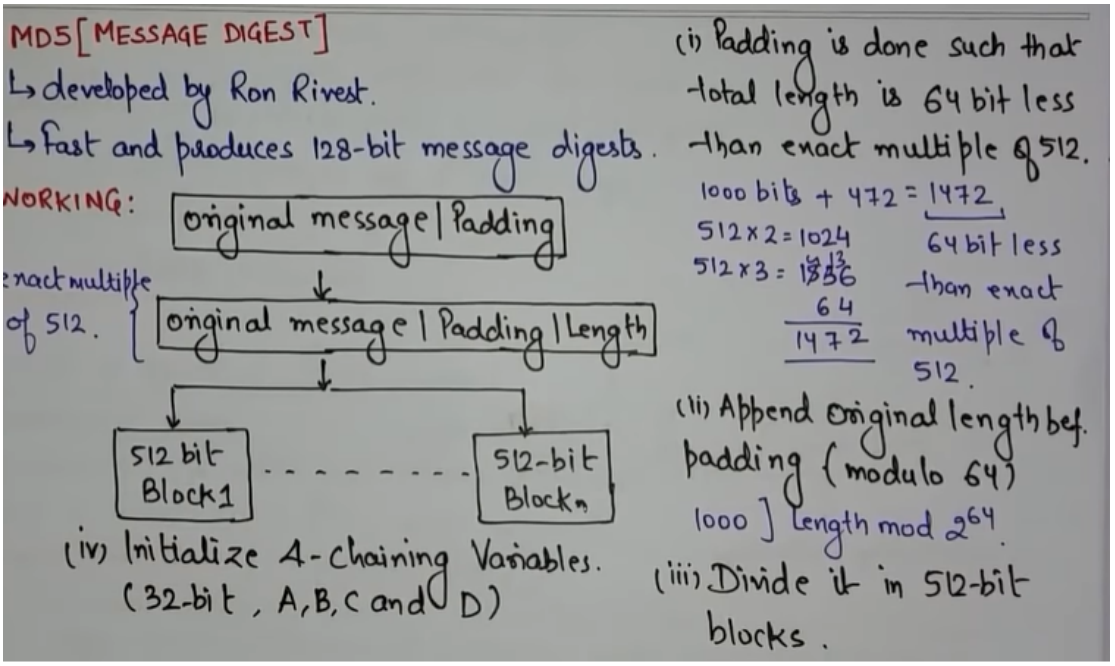
Now take input as initialize MD buffer i.e. J, K, L, M. Output of K will be fed in L, L will be fed into M, and M will be fed into J. After doing this now we perform some operations to find the output for J.

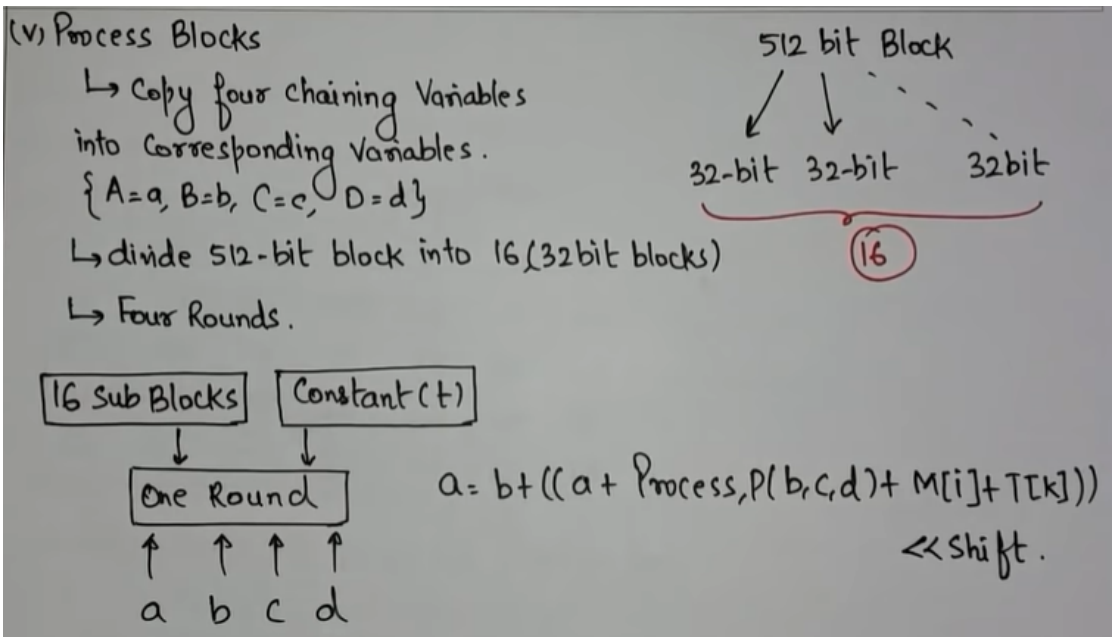
- In the first step, Outputs of K, L, and M are taken and then the function F is applied to them. We will add modulo $2_{32}$ bits for the output of this with J.
- In the second step, we add the M[i] bit message with the output of the first step.
- Then add 32 bits constant i.e. K[i] to the output of the second step.
- At last, we do left shift operation by n (can be any value of n) and addition modulo by $2_{32}$.

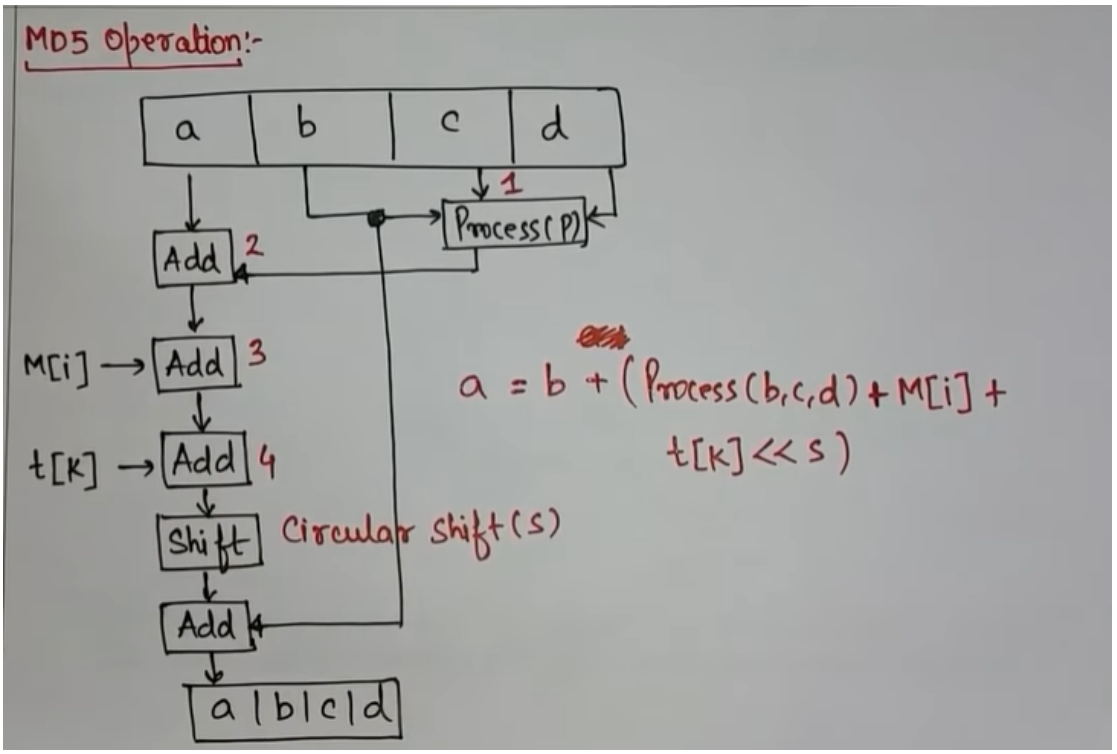
After all steps, the result of J will be fed into K. Now same steps will be used for all functions G, H, and I. After performing all 64 operations we will get our message digest.

**Output:**

After all, rounds have been performed, the buffer J, K, L, and M contains the MD5 output starting with the lower bit J and ending with Higher bits M.






## HMAC & CMAC

# HMAC(Hash based Message Authentication Code)

**HMAC** (Hash-based Message Authentication Code) is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is)

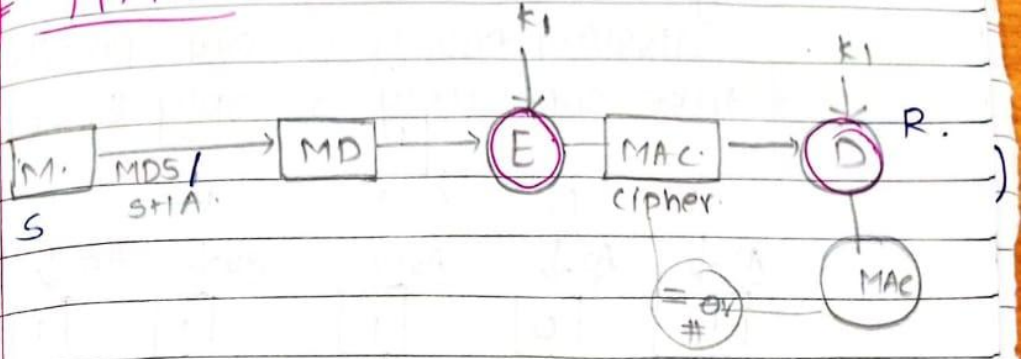
to be authenticated and a secret shared key. Like any of the MAC, it is used for both data integrity and authentication. Checking data integrity is necessary for the parties involved in communication. HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5,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

**Applications**

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

In MAC → direct MAC is generated
In HMAC — MAC is generated through MD

# HMAC (both confidentiality, Authentication)




(1) Sender send original message through the hash function ~~or with~~ through MD5 or SHA algorithm
(2) It generate message digest
(3) Apply encry algo with private key on MD it get MAC as ciphertext.
(4) Now R. decrypt with same public key get out as MAC if both are equal then no msg is not changed by thired party.

# CMAC (cipher Based MAC.)

(1) Msg has size limit.
(2) It's Algo. based on BLOCK cipher.
(3) Given msg is divided into equal no. of block (pair) & each block is encrypted seperately.

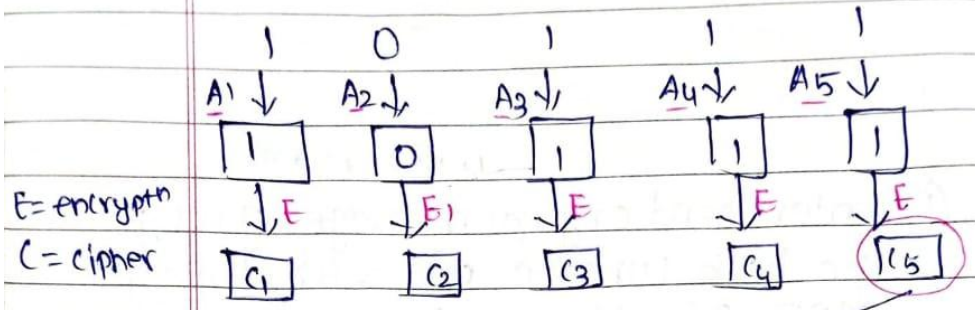
e.g pooja — n = 5. (divide equal)
po, oj, (ja) ✗ (not equal)
poo, (ja) ✗ not equal.

take single single

e.g. msg = 10111

dividing equaly is not possible

take seperately & aply E seperately.




E = encryptn

C = cipher

last cipher act as MAC. (i.e C5)

In HASH And MIAC we have only 1 MAC. but here we have no. of MAC (i.e cipher) so select last cipher as MAC.

$C_1 = E(K, A_1)$

$C_2 = E[K, (A_2 \oplus C_1)]$ — original key

$C_3 = E[K, (A_3 \oplus C_2)]$ — K; msg (original msg + privis cipher)

key will same for all block

Encryptn (key + msg)

$C_n = E[K, (A_n \oplus C_{n-1})]$

Here, Cn Act as MAC.

## Message Authentication Code (MAC)

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

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

## 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.

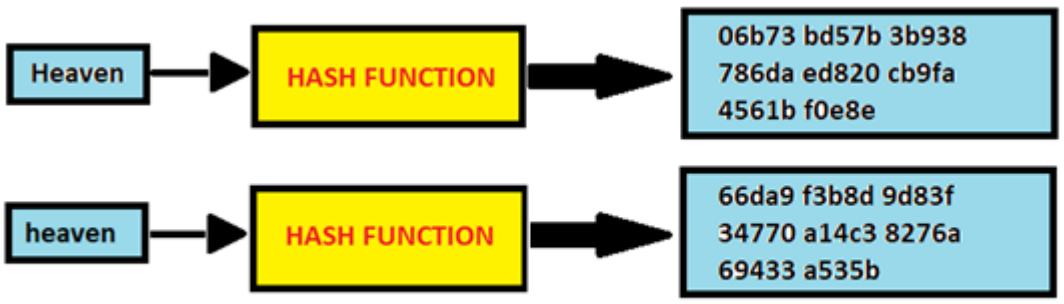
- Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

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

## SHA

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.

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.




This effect is important in cryptography, as it means even the slightest change in the input message completely changes the output. This will stop attackers from being able to understand what the hash digest originally said and telling the receiver of the message whether or not the message has been changed while in transit.

## SECURE HASH ALGORITHM (SHA):

(NIST) Process Block → Copy of C.V
→ 512 → 16 Sub Block (32)
→ four Round (20 steps)

SHA is a modified version of MD5.

↳ o/p is a message digest of 160 bits in Length.

### SHA Properties:-

(i) Generating original message from digest
(ii) finding two messages generating same digest
} Infeasible.

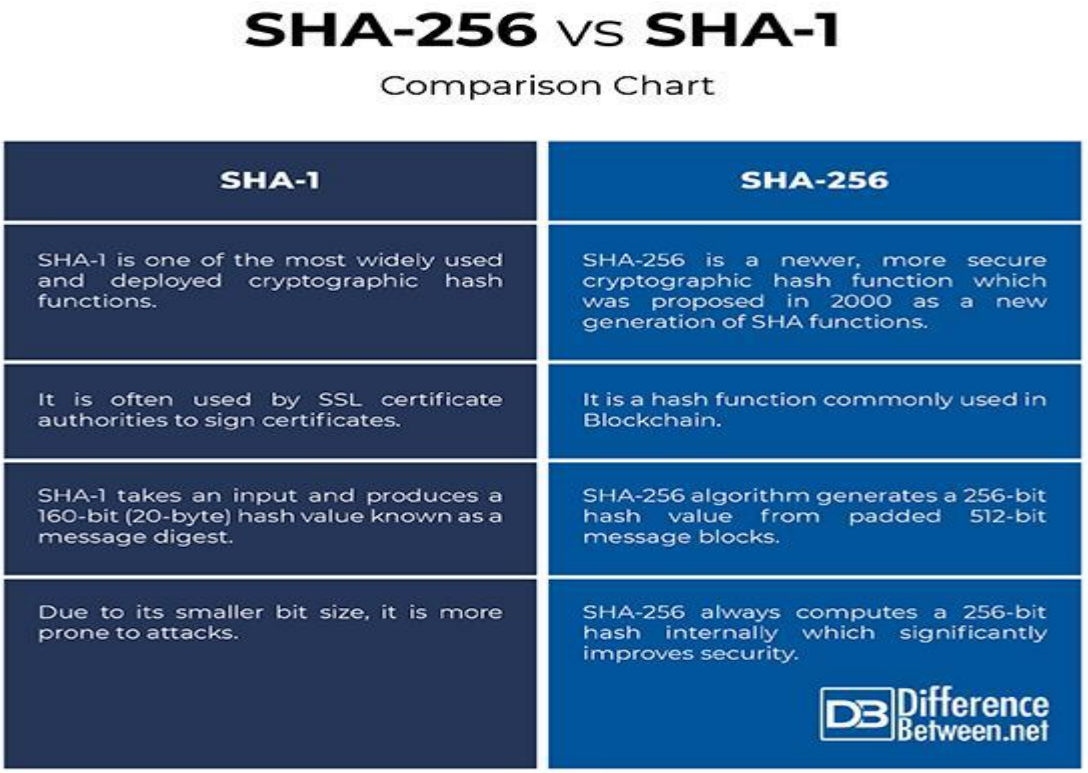
### Working of SHA:

(i) Padding [64 bit less than exact multiple of 512]
(ii) Append Length
(iii) divide the i/p into 512-bit blocks
~~(iv)~~ Five chaining Variables (A, B, C, D, E)
(v) Process Blocks.

} exactly same as MD5

$$abcde = (e + \text{Process P} + s^5(a) + W[t] + K[t]), a, s^{30}(b), c, d.$$

| Keys For Comparison | MD5 | SHA |
|---|---|---|
| Security | Less Secure than SHA | High Secure than MD5 |
| Message Digest Length | 128 Bits | 160 Bits |
| Attacks required to find out original Message | $2^{128}$ bit operations required to break | $2^{160}$ bit operations required to break |
| Attacks to try and find two messages producing the same MD | $2^{64}$ bit operations required to break | $2^{80}$ bit operations required to break |
| Speed | Faster, only 64 iterations | Slower than MD5, Required 80 iterations |
| Successful attacks so far | Attacks reported to some extents | No such attach report yet |




Examples of SHA names used are SHA-1, SHA-2, SHA-256, SHA-512, SHA-224, and SHA-384, but in actuality there are only two types: SHA-1 and SHA-2. The other larger numbers, like SHA-256, are just versions of SHA-2 that note the bit lengths of the SHA-2.

# HMAC(Hash based Message Authentication Code)

**HMAC** (Hash-based Message Authentication Code) is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key. Like any of the MAC, it is used for both data

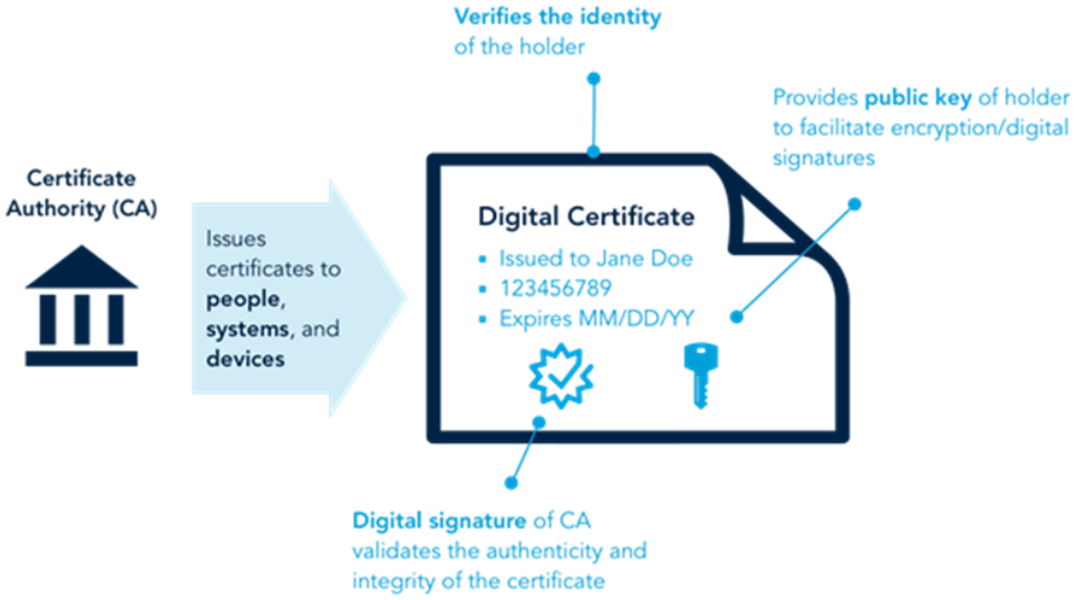
integrity and authentication. Checking data integrity is necessary for the parties involved in communication. HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5, SHA-1, or SHA-256. Digital signatures are nearly similar to HMACs i.e they both employ a hash function and a shared key. The difference lies in the keys i.e HMACs use symmetric key(same copy) while Signatures use asymmetric (two different keys).

**Applications**

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

# Digital Certificate

A digital certificate is a digital document that includes the public key related to an individual, organization, or a computer. Certificates are issued by CA - certificate authorities. They have documented policies for determining owner identity and distributing certificates.




# X.509 Authentication

X.509 digital certificate-based authentication is a standard-based security framework that is used to secure private information and transaction processing. Certificates are

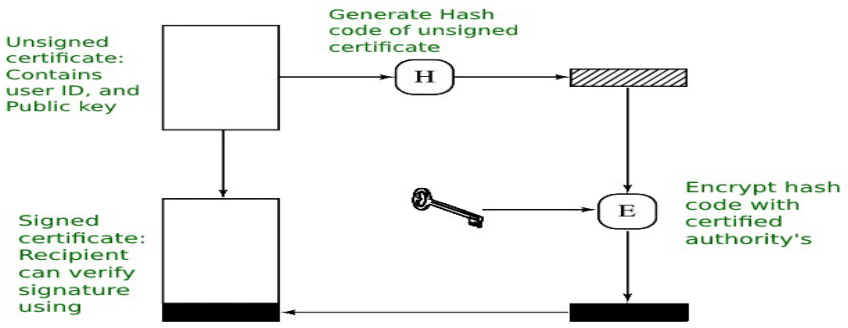
exchanged in a way to ensure that the one who presents a certificate possesses the private key associated with the public-key which is contained in the certificate.

An X.509 certificate is a digital certificate that uses the globally accepted international X.509 public key infrastructure (PKI) standard to verify that a specific public key belongs to the user, service identity or computer contained within the certificate. The format of the public key certificate is defined by the X.509 standard. X.509 certificates are utilized in many Internet protocols, for instance, TLS/SSL, which is the basis for HTTPS that is certainly a secure protocol for browsing the web. Offline applications, like electronic signatures, also use X.509 certificates

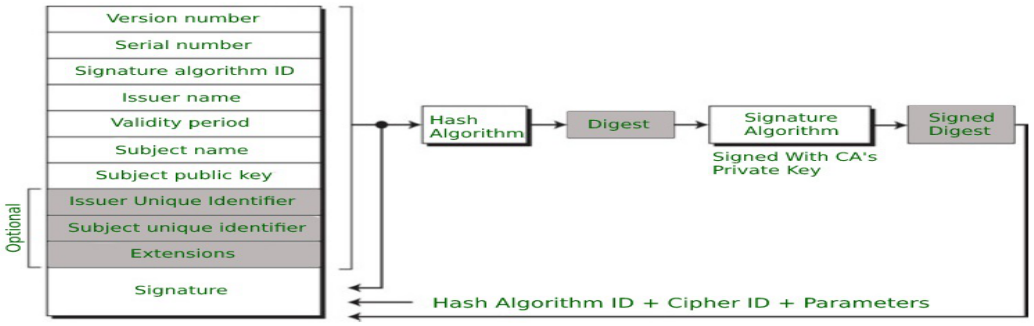
## Working of X.509 Authentication Service Certificate:

The core of the X.509 authentication service is the public key certificate connected to each user. These user certificates are assumed to be produced by some trusted certification authority and positioned in the directory by the user or the certified authority. These directory servers are only used for providing an effortless reachable location for all users so that they can acquire certificates. X.509 standard is built on an IDL known as ASN.1. With the help of Abstract Syntax Notation, the X.509 certificate format uses an associated public and private key pair for encrypting and decrypting a message.

Once an X.509 certificate is provided to a user by the certified authority, that certificate is attached to it like an identity card. The chances of someone stealing it or losing it are less, unlike other unsecured passwords. With the help of this analogy, it is easier to imagine how this authentication works: the certificate is basically presented like an identity at the resource that requires authentication.




## Format of X.509 Authentication Service Certificate:




Generally, the certificate includes the elements given below:

- **Version number:** It defines the X.509 version that concerns the certificate.
- **Serial number:** It is the unique number that the certified authority issues.
- **Signature Algorithm Identifier:** This is the algorithm that is used for signing the certificate.
- **Issuer name:** Tells about the X.500 name of the certified authority which signed and created the certificate.
- **Period of Validity:** It defines the period for which the certificate is valid.
- **Subject Name:** Tells about the name of the user to whom this certificate has been issued.
- **Subject's public key information:** It defines the subject's public key along with an identifier of the algorithm for which this key is supposed to be used.
- **Extension block:** This field contains additional standard information.
- **Signature:** This field contains the hash code of all other fields which is encrypted by the certified authority private key.

**Applications of X.509 Authentication Service Certificate:**

Many protocols depend on X.509 and it has many applications, some of them are given below:

- Document signing and Digital signature
- Web server security with the help of Transport Layer Security (TLS)/Secure Sockets Layer (SSL) certificates
- Email certificates
- Code signing
- Secure Shell Protocol (SSH) keys
- Digital Identities

## WHAT IS PKI?

Public Key Infrastructure (PKI) is a system of processes, technologies, and policies that allows you to encrypt and sign data. You can issue digital certificates that authenticate the identity of users, devices, or services. These certificates create a secure connection for both public web pages and private systems—such as your virtual private network (VPN), internal Wi-Fi, wiki pages, and other services that support MFA. Have questions?

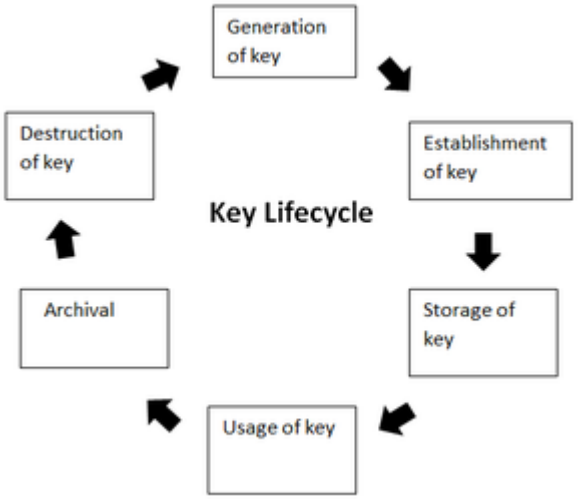
It helps to protect confidential data and gives unique identities to users and systems. Thus, it ensures security in communications.

The public key infrastructure uses a pair of keys: the public key and the private key to achieve security. The public keys are prone to attacks and thus an intact infrastructure is needed to maintain them.

## Managing Keys in the Cryptosystem:

The 3 main areas of key management are as follows:

- A cryptographic key is a piece of data that must be managed by secure administration.
- It involves managing the key life cycle which is as follows:




- Public key management further requires:
    - **Keeping the private key secret:** Only the owner of a private key is authorized to use a private key. It should thus remain out of reach of any other person.
    - **Assuring the public key:** Public keys are in the open domain and can be publicly accessed. When this extent of public accessibility, it becomes hard to know if a key is correct and what it will be used for. The purpose of a public key must be explicitly defined.

PKI or public key infrastructure aims at achieving the assurance of public key.

## Public Key Infrastructure:

Public key infrastructure affirms the usage of a public key. PKI identifies a public key along with its purpose. It usually consists of the following components:

- A digital certificate also called a public key certificate
- Private Key tokens
- Registration authority
- Certification authority
- CMS or Certification management system