3

Module 3

Cryptographic Hashes, Message Digests and Digital Certificates

Syllabus

At the end of this unit, you should be able to understand and comprehend the following syllabus topics:

- Message Authentication requirements
- Message Authentication functions
- Types of Authentication
  - Hash function
    - SHA
    - MD5
  - MAC
  - HMAC
  - CMAC
- Security of hash function and MAC
- X.509 - Digital Certificate
- Public Key Infrastructure (PKI)

3.1 Message Authentication Requirements

So far, in the earlier chapters, you learnt about information secrecy – which was all about keeping the message secret. The focus of this chapter is information accuracy which is about ensuring the message integrity or message authentication.

✍ ***Definition :*** *Message authentication is a process to ensure that the received message is exactly the same as it was sent.*

What does this mean? This means that :

- The message has not been altered in anyway
  - No addition
  - No modification
  - No deletion
- The message is actually sent by the sender (proof of sender's identity)
- The order or the sequence of the messages is not changed
- The messages are sent and received within an expected time frame

Let's consider a day to day scenario. You go to a shop to buy bread and in-exchange you give the shopkeeper a Rupees 100 currency note. The shopkeeper happily takes the note from you, examines it briefly, keeps it in the drawer and return you some change. You take the change, examine the returned currency briefly, slide it down in your wallet and move.

What just happened? Why did the shopkeeper take the note you gave without any hesitation? Why did you take the change without any hesitation? What did the shopkeeper examine when you gave the note and why? What did you examine when you received the change and why? Are you telling me, *come on*, this is child simple? Are you trying to explain me the following?

- There were two parties to the transaction – you and shopkeeper
- Shopkeeper wanted to accurately determine that the note you gave was genuine and not fake
- Shopkeeper looked at some of the known properties of the note (that were attached to the note itself) and verified it accurately
- You wanted to accurately determine that the change currency that you got was genuine and not fake
- You looked at some of the known properties of the note that you got and verified the note accurately

So, you understand that in this world, where the faith is based on the principle of "trust but verify", you need a mechanism to accurately determine the accuracy of the information that you get.

Have you ever trusted a fake news or video clip and later laughed at yourself or felt bad about trusting the information without checking it genuinely? Don't you feel that there should be *someone* who could just tell you if the information you got was accurate or not? Did you really receive the email that you were sent, or someone modified it on its way ? Did someone modify your file? If all these problems worry you, I request you not to worry, you have a way out. Please heartily welcome Cryptographic Hash functions!

3.2 Message Authentication Functions

At a high level, the message authentication can be performed using three mechanisms :

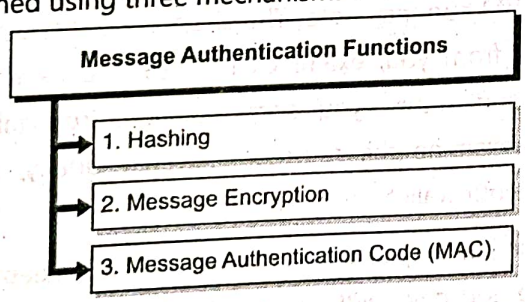

Fig. 3.2.1 : Message authentication function

1. **Hashing :** It deals with producing a unique hash value of the message that can be computed at the sender's and the receiver's end. If the hashes match at both the ends, the message is verified.
2. **Message encryption :** In this, the ciphertext of the entire message can be used to serve as its authenticator. The message is encrypted at the sender's and the receiver's end. If they both get the same ciphertext using the same key, it verifies the message. Note here that the focus is not to encrypt the message but to get the resulting ciphertext to serve as a way to verify the authenticity of the message.
3. **Message Authentication Code (MAC) :** MAC is very similar to hash. It uses a key to calculate the hash value of the message. The MAC is calculated at both the ends (sender's and receiver's) using the same key. If the MAC value matches at both the ends, the message is verified. You will learn about it in detail in the subsequent sections.

3.3 Cryptographic Hash Functions

As encryption provides message confidentiality, hashing provides message integrity and authentication.

3.3.1 Introduction

✍ ***Definition :*** *Hashing is the process of taking any length of input information and finding a unique fixed length representation of that input information.*

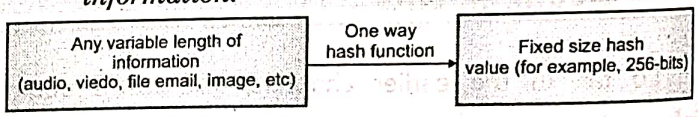

Fig. 3.3.1 : Example of hash function

Hashing is the process of finding a unique message digest (or hash value) that corresponds to the input information. The length of input could be just one character or a huge video file. Hashing always produces a fixed size representation of the information. Note here that hashing does not make the information unreadable. It is not same as encryption. Hashing is just a way to attach some additional information to the original information that could later prove that the information is not modified. Recall the discussion from concept building Section 3.1. The shopkeeper can see Rupees 100 written clearly but verifies the currency note using other properties "attached" to it that proves its originality and authenticity.

3.3.2 How does this Work?

At the source of the information (it could be sender, website, company or anything else where the information is created) the hash value is calculated. This hash value along with the original information is sent to the receiver as shown in Fig. 3.3.2.

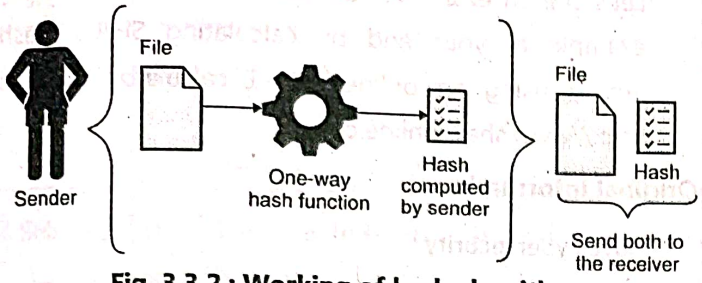

Fig. 3.3.2 : Working of hash algorithms

At the destination of the information (it could be a receiver, website, email program, or anything else where the information is received to process further), the hash value is calculated again and matched with the hash value that came with the information from source. If the two hash values (at source and at destination) match, the information is determined to be unmodified and is consumed. But, if the two hash values do not match, it proves that the information is altered and is often rejected as shown in Fig. 3.3.3.

I want to again stress on the point that hashing is only for integrity checking of the information. The original information may or may not be encrypted. If the information is to be encrypted, following could be one of the sequences for integrity checking.

1. Create information
2. Calculate its hash value
3. Encrypt information
4. Send encrypted information and hash value
5. Receive encrypted information and hash value
6. Decrypt information
7. Calculate its hash value at the receiving end
8. Match source and destination hash
9. If matched, process information
10. If not matched, reject information

Now, you might be thinking, yeah, this sounds good but what if someone captures the message and the hash, alters the message and creates and attaches the new hash to match the altered message and sends it to the receiver. How would receiver ever know? Great question, I must say. I would answer that later on. Let's park it at the moment.

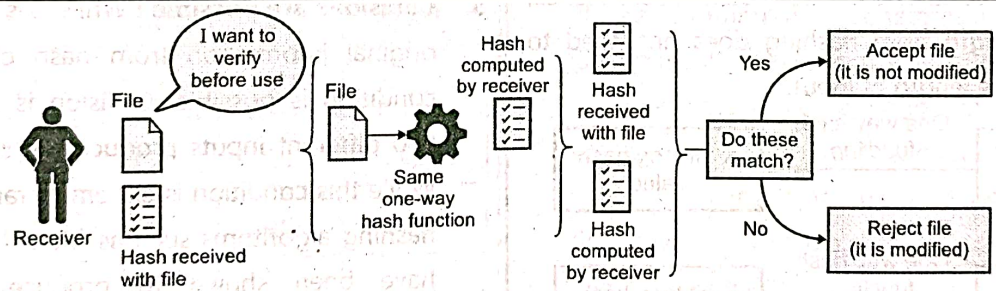

Fig. 3.3.3 : Hashing used for proving data integrity

3.3.3 Characteristics of Hash Functions

Unlike encryption, hash functions have some unique properties are shown in Fig. 3.3.4. Let's learn about them.

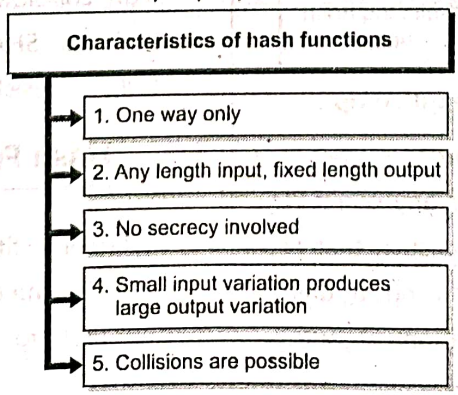

Fig. 3.3.4 : Characteristics of hash function

1. **One-way only :** This means that it is easy to calculate hash value from information and nearly impossible to convert the hash value back to the original information. Understand it like this. It is easy to convert milk into cheese, but can you convert cheese back to the original milk? Sounds weird and impossible right? That's how hash functions are One-way only.

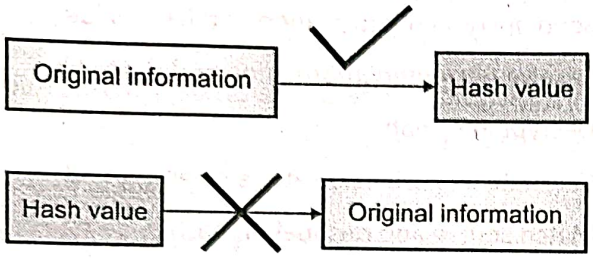

Fig. 3.3.5 : One-way only

2. **Any length input, fixed length output :** Hash functions can work on any length of input. It could work on a single character or a huge video file. It would always produce the same length output. Unlike encryption where the output size is more or less same as the input size, output from hashing process is only a unique representation of the original information and does not contain the information itself. Hence, the output length from hashing does not need to change with the length of input.

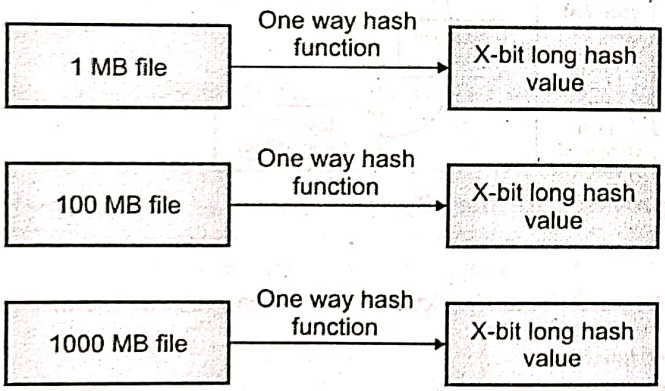

Fig. 3.3.6 : Any length input, fixed length output

3. **No secrecy involved :** Hashing process does not require a key and neither it requires any secret. The hashing algorithms are implemented such that they are only a one-way process producing a unique representation of the input information. Algorithms are publicly known as well.

4. **Small variation in input produces large variation in output :** It is nearly impossible to establish any relation between input and output in the hashing process. A small variation in the input totally changes the hash output. This is also called avalanche effect. Let's see an example. You can also try the following example at your end by calculating SHA-1 hash output using an online SHA-1 calculator such as http://www.sha1-online.com/.

Original information :

I love cybersecurity

SHA-1 Hash value :

653da04906493b11d0735020db1f94360cac64f0

Original information :

U love cybersecurity

SHA-1 Hash value :

8b9a7e5b4e5c8306c6ba678454e485356cb0aa24

Note that it does not matter which online calculator you use. You would get the same SHA-1 output for the given input.

5. **Collisions are possible :** While it is impossible to find original information from hash output, a collision condition is possible. Collision is a situation where two different inputs produce the same hash output. While this condition is extremely rare, some historical hashing algorithms such as MD2, MD4, MD5, SHA-1 have been shown to produce collision. These algorithms are no more used in the industry today. A hashing algorithm is considered strong if it provides high collision resistance. Newer hashing algorithms such as SHA-256 and SHA3-256 provide strong collision resistance.

3.4 Hash Functions (Algorithms)

National Institute of Standards and Technology (NIST) has specified the family of hashing algorithms that may or may not be fit for current use in the industry.

Table 3.4.1 : Family of hashing algorithms

Hash Family Name	List of hashing algorithms	Approved for use?
SHA-1	SHA-1	No
SHA-2	SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256	Yes
SHA-3	SHA3-224, SHA3-256, SHA3-384, SHA3-512	Yes

Table 3.4.2 : Hashing Algorithms

Hashing Algorithm	Output size in bits	Block size in bits	Rounds	First published
SHA-1	160	512	80	1995
SHA-224	224	512	64	2004
SHA-256	256	512	64	2001
SHA-384	384	1024	80	2001
SHA-512	512	1024	80	2001
SHA-512/224	224	1024	80	2012
SHA-512/256	256	1024	80	2012
SHA3-224	224	1152	24	2015
SHA3-256	256	1088	24	2015
SHA3-384	384	832	24	2015
SHA3-512	512	576	24	2015

3.4.1 SHA-1

✍ ***Definition :*** *Secure Hash Algorithm 1 is a cryptographic hash function that produces 160-bit long hash value from a given input.*

Collisions have been found in the algorithm and hence it is avoided in the industry wherever possible.

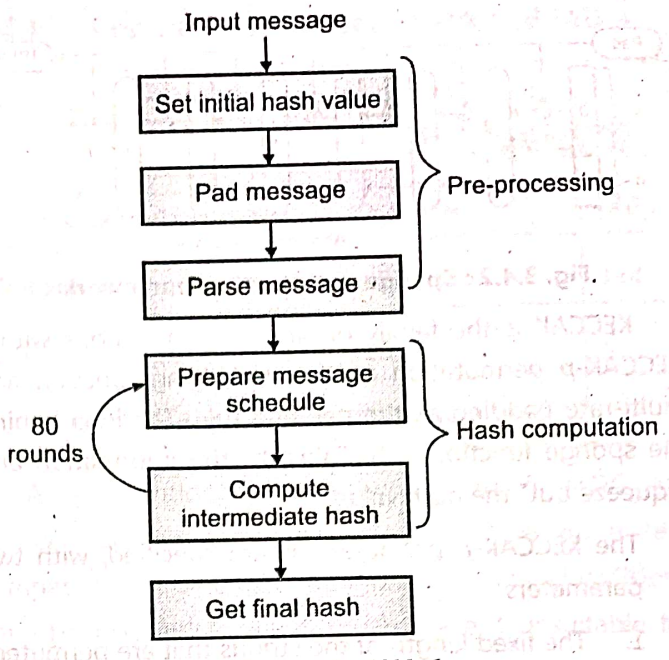

Fig. 3.4.1 : SHA-1

The algorithm has two stages are shown in Fig. 3.4.1

1. Pre-processing

Pre-processing involves padding a message, parsing the padded message into equal sized blocks and then setting initial values to be used for hash computation.

2. Hash computation

The hash computation generates a message schedule (sequence of processing) from the padded message and uses that schedule along with various functions, constants and mathematical operations to generate a series of intermediate hash values per round. The final hash is value is computed after the last round and is 160-bit long.

3.4.2 SHA-3

✍ ***Definition :*** *Secure Hash Algorithm 3 is the newest addition to the hash function family. It is comprised of various hash functions that compute secure hash values.*

It is considered to be most secure against collision and is highly recommended to be used in the industry. SHA-3 is structurally quite different from earlier versions : SHA-1 and SHA-2. SHA-3 is a subset of the broader cryptographic primitive family KECCAK.

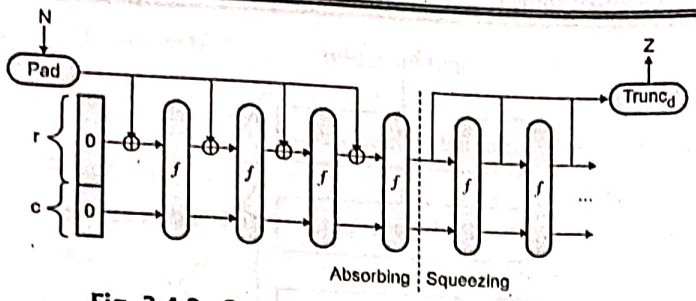

Fig. 3.4.2 : Sponge construction framework

KECCAK is the family of all sponge functions with a KECCAK-*p* permutation as the underlying function and multi-rate padding as the padding rule. The idea behind the sponge function is to "absorb" the information and "squeeze out" the hash value.

- The KECCAK-*p* permutations are specified, with two parameters :
  1. The fixed length of the strings that are permuted
  2. The number of rounds
- The sponge construction is a framework for specifying functions on binary data. It has three components :
  1. Function f that works on fixed-length strings
  2. Parameter r that determines the rate of operation
  3. A padding rule denoted by pad
- The sponge function can be denoted as SPONGE[f, pad, r].

A sponge function takes two inputs:

1. A bit string that is denoted by N
2. The bit length that is denoted by d

- So, the overall function is denoted as SPONGE[f, pad, r](N, d).
- In the sponge function, Absorb Phase involves :
  1. XORing of message blocks into a subset of the state
  2. Using the permutation function for transferring the whole state
- In the sponge function, Squeeze Phase involves :
  1. Reading output blocks from the same subset of state
  2. Replacing with state transformation function

3.4.3 MD5

✍ ***Definition :*** *MD5 is a hashing algorithm.*

MD5 was designed by Ronald Rivest in 1991.

3.4.3(A) Major Attributes of MD5

- Block size – 512 bits
- Output size (hash size) – 128 bits
- Rounds – 4
- It is broken (collisions have been shown) and obsolete and is no more considered fit for cryptographic use
- It is replaced by newer algorithms such as SHA-1 and SHA-2

3.4.3(B) MD5 Algorithm Details

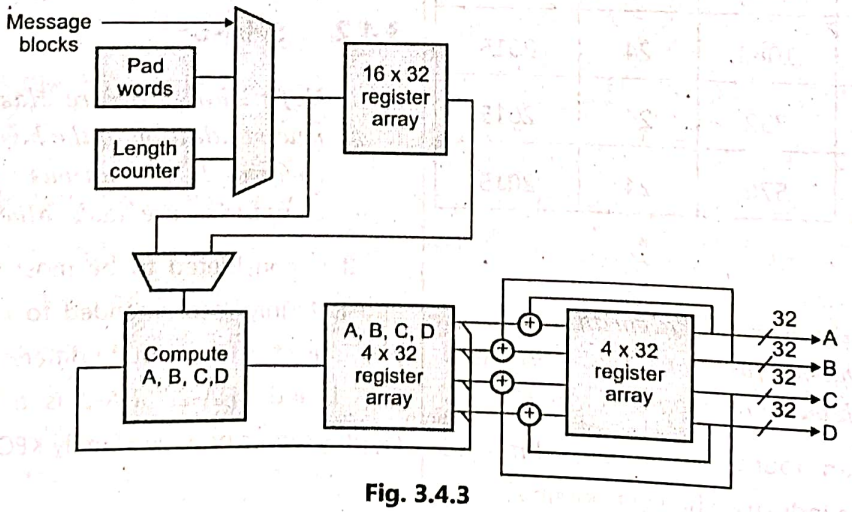

Fig. 3.4.3

MD5 algorithm specifies the following 5 steps for computing the message digest (or the hash value).

1. **Append Padding Bits :** The message is "padded" (extended) so that its length (in bits) is congruent to 448, modulo 512. That is, the message is extended so that it is just 64 bits shorter of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already congruent to 448, modulo 512. Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, at least one bit and at most 512 bits are appended.
2. **Append Length :** A 64-bit representation of the length, of the message before the padding bits were added, is appended to the result of the previous step. After this step, the resulting message length (after padding with bits and with length specification) is an exact multiple of 512 bits.
3. **Initialize MD Buffer :** A four-word buffer (A,B,C,D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized.
4. **Process Message in 16-Word Blocks :** At this stage, the four auxiliary functions take as input three 32-bit words and produce as output one 32-bit word. The functions are

   (a) F(X,Y,Z) = X *AND* Y *OR* *NOT*(X) *AND* Z

   (b) G(X,Y,Z) = X *AND* Z *OR* Y *AND NOT*(Z)

   (c) H(X,Y,Z) = X *XOR* Y *XOR* Z

   (d) I(X,Y,Z) = Y *XOR* (X *OR NOT*(Z))
5. **Output :** From these respective four functions, a message digest is produced and stored in the respective registers – A, B, C, D. The final output (message digest or the hash value) is given by concatenating (bringing together) these values.

3.4.4 Comparison between SHA and MD5

Q. SHA provides better security than MD. Justify. **MU - Dec. 18, 5 Marks**

Comparison Attribute	SHA	MD5
Output bits	160 – 512	128
Number of rounds	24 – 80	4
Collision Found	No (for SHA-2 above)	Yes

As you understand, the output bits of MD5 and number of rounds in the algorithm are quite less compared to SHA family of algorithms. Several collisions have been found for MD5 which make it unsuitable for modern data integrity requirements.

3.5 Message Authentication Code (MAC)

Hash values provide integrity. But what if someone alters the message and adds a new hash? How do you know that you received the original message that the sender had intended to send? Hash values are computed on the original message and usually sent along with the message. Creating hash values does not require any keys. Hence, hash values can only provide integrity but cannot provide any additional assurance that the message is authentic.

✍ ***Definition** : A Message Authentication Code (MAC) is a piece of information that can be used to authenticate a message.*

MAC confirms that the message came from the stated sender and has not been changed. The MAC value protects both a message's data integrity as well as its authenticity. MAC is sometimes also called as ***keyed* hash.**

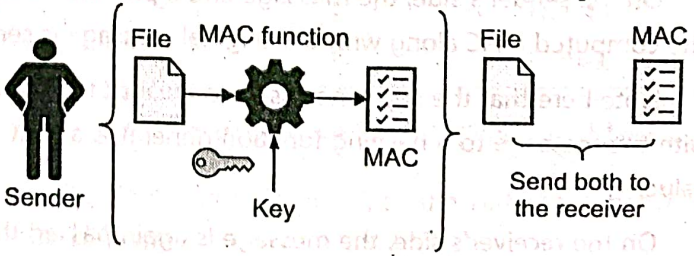

Fig. 3.5.1

The sender intends to send a file (or a message). He computes the MAC value for the file using the pre-shared key. He then forwards the message as well as the corresponding MAC value to the receiver.

At the receiver's end, the MAC value is again computed using the same pre-shared key. This computed MAC value is then compared with the received MAC. If these two match, the file (or the message) is not altered and is authenticated.

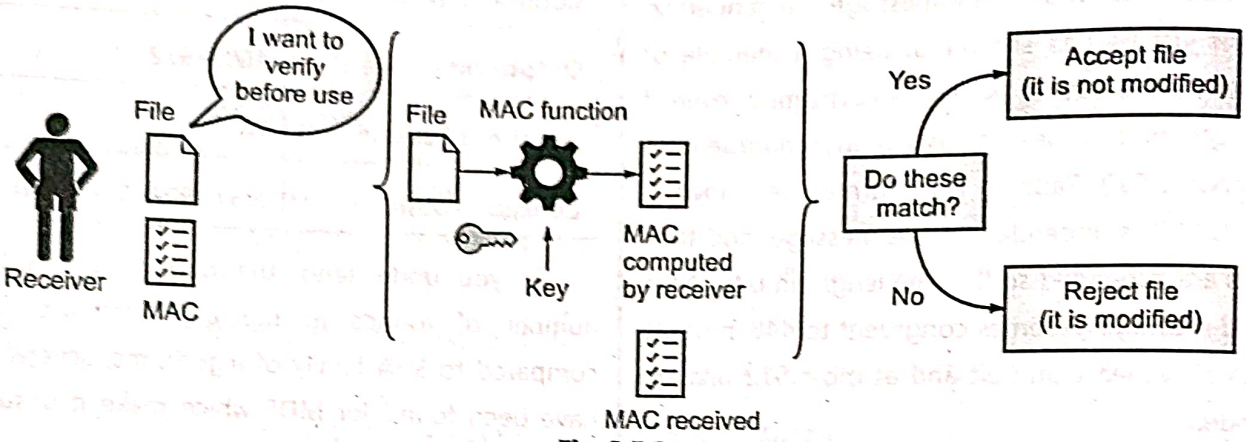

Fig. 3.5.2

There are three types of MAC :

1. HMAC (Hash MAC)
2. CBC-MAC
3. CMAC (Cipher-based MAC)

Let's learn about each of them.

3.5.1 HMAC

✍ ***Definition :*** *In HMAC, a hashing function is used as a MAC function to calculate the MAC value.*

The hashing function could be general hash functions such as MD5, SHA-1, or SHA-2.

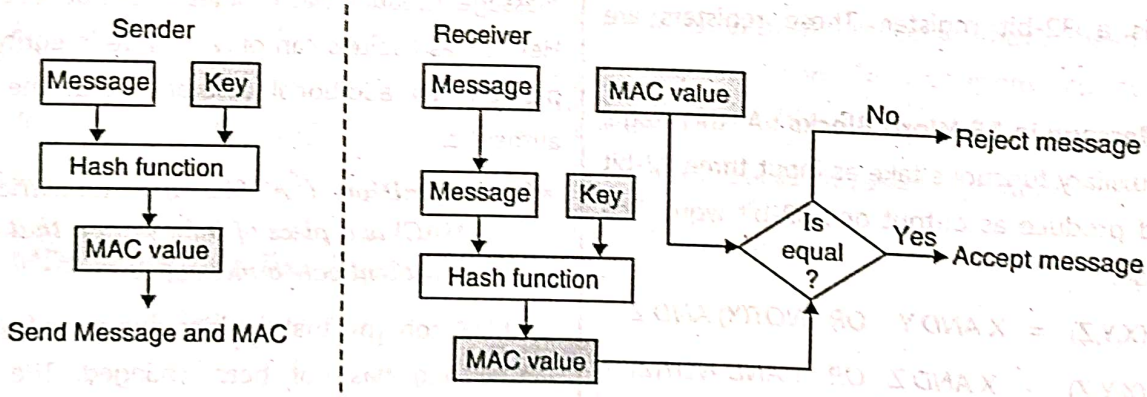

Fig. 3.5.3

On the sender's side, the message and a pre-shared key is passed together into a hash function to compute a MAC. The computed MAC along with the original message is sent to the receiver. The key is not sent.

Note here that the key is not used to encrypt the message. The key just provides a random value that when passed with the message to a hashing function generates a MAC value. The key has no other role except to provide a random value.

On the receiver's side, the message is again passed through the same hashing function using the same pre-shared key. A MAC value is computed at the receiver's side and is matched with the MAC value that came from the sender. If the two MAC values match, the message is accepted else the message is rejected.

3.5.2 CBC-MAC

✍ ***Definition :*** *In CBC-MAC, a symmetric block cipher encryption function in the CBC mode is used as the MAC function to calculate the MAC value.*

The message is encrypted with a symmetric block cipher in the CBC mode, and the output of the final block of ciphertext is used as the MAC. The sender does not send the encrypted version of the message, but instead sends the plaintext version and the computed MAC.

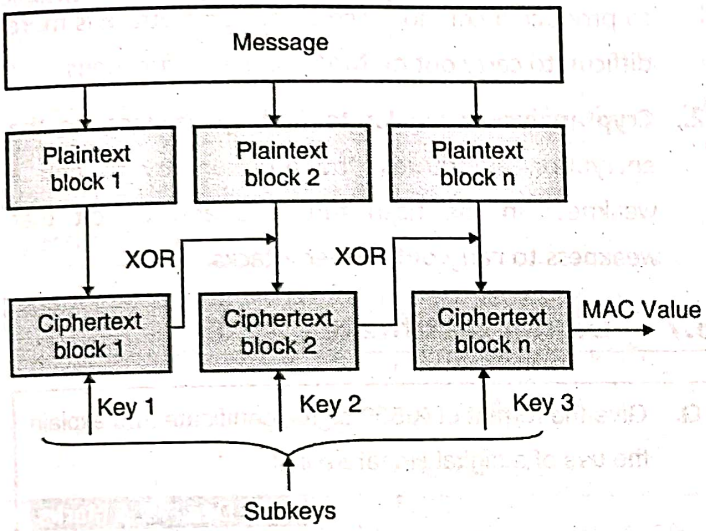

Fig. 3.5.4

The receiver receives the plaintext message and encrypts it with the same symmetric block cipher in CBC mode and calculates a MAC value at her end. If the two MAC values (one received and one computed) match, the message is accepted else it is rejected. This method does not use a hashing algorithm as in HMAC.

3.5.3 CMAC

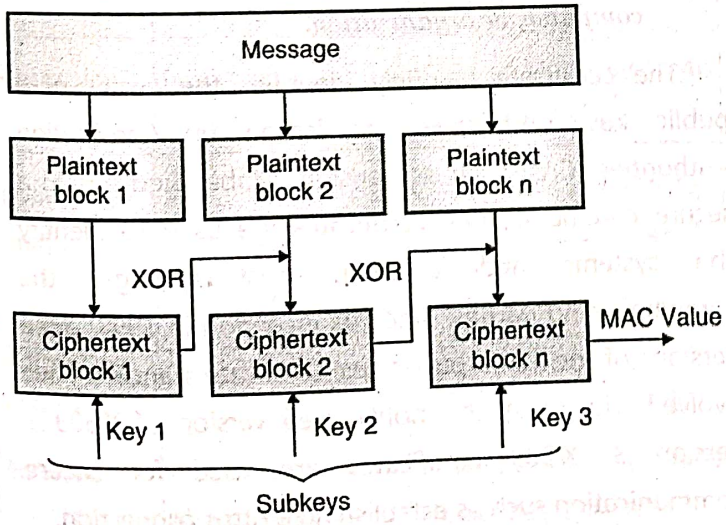

Fig. 3.5.5 : CMAC

CMAC (Cipher-based MAC) is very similar to CBC-MAC.

✍ ***Definition :*** *In CMAC, a symmetric block cipher encryption function is used as the MAC function to calculate the MAC value.*

Note here that like CBC-MAC, a symmetric block cipher encryption function is used here as well but not in CBC mode. That's the major difference between CBC-MAC and CMAC. CMAC is typically calculated using AES-128 algorithm and provides strongest form of message authentication and integrity. It is also called as One-Key MAC or OMAC.

3.5.4 Comparison between HMAC, CBC-MAC and CMAC

The Table 3.5.2 summarises the key differences between HMAC, CBC-MAC and CMAC.

Table 3.5.2 : Comparison between HMAC, CBC-MAC and CMAC

Comparison Attribute	HMAC	CBC-MAC	CMAC
MAC generation function	Hash Function	Symmetric cipher in CBC mode	Symmetric cipher
Speed of MAC generation	Highest	Lowest	Medium
Strength of MAC	High	Very High	Very High
Number of Keys used	One	One	One key divided into multiple sub-keys

3.5.5 Comparison between Hash and MAC

The Table 3.5.1 summarizes the difference between Hash and MAC.

Table 3.5.1 : Comparison between Hash and MAC

Comparison Attribute	Hash	MAC
Integrity	Yes	Yes
Authentication	No	Yes
Non-repudiation	No	No
Keys Used	None	Symmetric Keys

3.6 Security of Hash Functions and MAC

Hash functions and MAC need to be secure by themselves to be useful. You cannot rely on these message authentication mechanisms if it is easy to "break" them. Let's understand some of the desired security properties of these mechanisms and possible attacks on them.

3.6.1 Security of Hash Functions

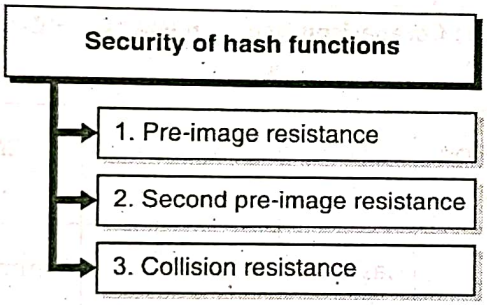

Fig. 3.6.1 : Security of Hash Functions

The three desired security properties of hash functions are as following.

1. **Pre-image resistance :** This property ensures the one-way only security criteria of hash functions. It states that for a given hash value h, it should be impossible to find any message m such that, h = hash (m). So, in a nutshell, you cannot find the original message from its hash value.
2. **Second pre-image resistance :** This property states that given a message m_1, it is hard to find another message m_2, such that hash(m_1) = hash(m_2). So, looking at a message it should be hard to find another message which could produce the same hash. This property is also called weak collision resistance.
3. **Collision resistance :** This property states that it should be hard to find any two messages pairs m_1 and m_2 such that hash(m_1) = hash(m_2). This property is also called strong collision resistance.

3.6.2 Attacks on Hash Functions and MAC

There are two possible ways to attack hash functions and MAC.

1. **Brute-force attack :** In this, the attacker repeatedly tries to find various combinations of messages so as to produce a collision. The brute-force attack is more difficult to carry out on MAC than hash functions.
2. **Cryptanalysis :** Similar to finding weakness in the encryption algorithms, the attackers try to find a weakness in the hash functions and exploit that weakness to carry out further attacks.

3.7 Digital Certificate - X.509

Q. Give the format of X.509 digital certificate and explain the use of a digital signature in it.

MU - Dec. 15, 5 Marks

Q. What is a digital certificate? How does it help to validate the authenticity of a user? Explain the X.509 certificate format. **MU- Dec. 17, 10 Marks**

Q. Write short notes on X.509. **MU - May 19, 4 Marks**

X.509 is a standard that defines the requirements for public key certificates.

✍ ***Definition :*** *A certificate is a signed data structure that binds a public key to a person, computer, or organization.*

The certificate uniquely identifies the owner of a public key. Certificates are issued by Certification Authorities (CAs) such as Verisign, Global Sign etc. In a secure communication, certificates are used to identify the systems and establish trust amongst the communicating parties. Since its inception in 1998, three versions of the X.509 public key certificate standard have evolved. The most commonly used version of X.509 is version 3. X.509 certificates are used for secure communication such as establishing a https connection.

Table 3.7.1 : Contents of a X.509 version 3 certificate

Fields	Purpose	Example
Version	Identifies the version of the certificate	V3
Serial Number	Unique number for the certificate	79ad16a14aa0a5ad4c7358f407132e65
Signature algorithm	Algorithm used to create digital signature for certificate	sha1RSA
Signature hash algorithm	Algorithm used to calculate hash of certificate	sha1
Issuer	Name of certificate issuer	CN = Microsoft Root Certificate Authority DC = Microsoft DC = com
Valid from	Date from which certificate is valid	Thursday, May 10, 2001 4:49:22 AM
Valid to	Date until which certificate is valid	Monday, May 10, 2021 4:58:13 AM
Subject	Name of certificate owner	CN = Microsoft Root Certificate Authority DC = Microsoft DC = com
Public key	Public key	a5 94 ef 15 14 89 fd 4b 73 ... (4096 bits)
Issuer unique ID	ID of issuing Certificate Authority (CA)	03475573948593hgfyuerwe327e7e52513fc2ae3
Subject unique ID	ID of subject	0eac826040562797e52513fc2ae10a539559e4a4
Extensions	Optional Information	Thumbprint, Friendly Name, Key Usage, etc.

Note : If you use Microsoft® Windows® OS, you can go to (Windows Key + R) Run and type certmgr.msc. It would open up certificate manager where you can browse various certificates stored on your system. Double-click on any certificate and examine the various fields it has.

3.8 Public Key Infrastructure (PKI)

Asymmetric cryptography has been widely adopted to provide various cryptographic services such as encryption, digital signature, authentication and non-repudiation. Asymmetric cryptography is critically dependent on the availability of an ecosystem of public private key-pairs. In this ecosystem, the key-pairs need to be securely generated, transported, distributed, verified, used and disposed. Public Key Infrastructure (PKI) provides such an ecosystem of operation.

✍ ***Definition :*** *Public Key Infrastructure (PKI) consists of programs, data formats, procedures, communication protocols, security policies, and public key cryptographic mechanisms working in a comprehensive manner to operationalize asymmetric cryptography based cryptographic services.*

It supports the distribution, revocation and verification of public keys used for public key encryption and enables linking of identities with public key certificates.

Note : Do not confuse between Public Key Cryptography and Public Key Infrastructure. Public Key Cryptography refers to the use of asymmetric keys for cryptographic services whereas Public Key Infrastructure is about managing the entire lifecycle of those keys. It is more about infrastructure management around those keys.

3.8.1 Components of PKI

PKI has several components that work together in a comprehensive manner to operationalize the ecosystem of public-private keys. The major components of PKI are as following.

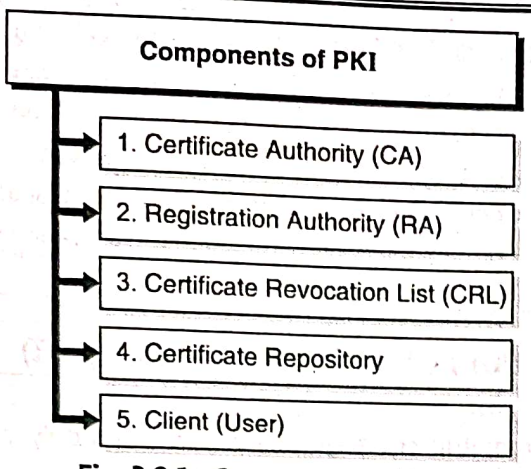

Fig. 3.8.1 : Components of PKI

1. Certificate Authority (CA)

✍ ***Definition :*** *The Certificate Authority (CA) is a trusted third party that provides the root of trust (highest level of trust) for all PKI certificates and provides services that can be used to authenticate the issued certificates.*

When the CA signs the certificate, it binds the certificate's identity to the public key, and the CA takes liability for the authenticity of that certificate. It is this trusted third party (the CA) that allows people who have never met to authenticate to each other and to communicate securely. A CA is a trusted organization (or server) that maintains and issues digital certificates. Some examples of CAs are Verisign, Comodo and Geotrust.

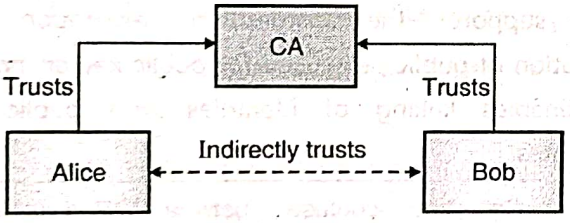

Fig. 3.8.2 : Certificate Authority (CA)

2. Registration Authority (RA)

✍ ***Definition :*** *The Registration Authority (RA) performs the certification registration duties. The RA establishes and confirms the identity of a certificate requester, initiates the certificate generation process with the CA, and manages the certificate life-cycle.*

The RA cannot issue certificates itself. It acts as a middle-man between the certificate requester and the associated CA. RA verifies all the necessary information before approving the request to get a certificate from the CA.

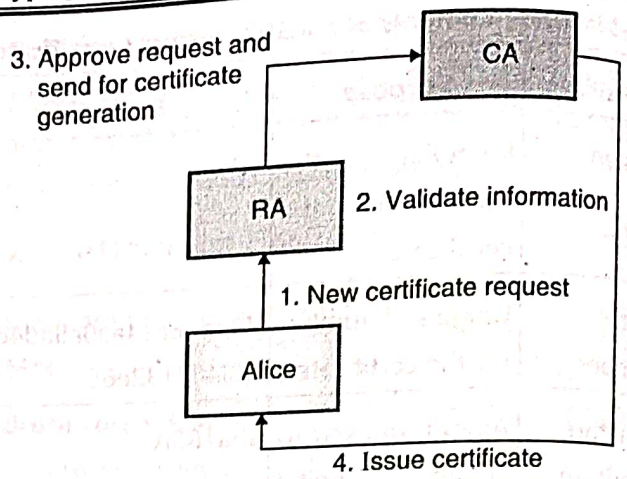

Fig. 3.8.3 : Registration Authority (RA)

3. Certificate Revocation List (CRL) : Certificates hold the public key information whereas the private key information should be kept secure with the user. In scenarios where the private key is compromised for any reason, the certificate cannot be trusted any further and hence it is crucial to let the world know that the certificate should not be trusted. Such a mechanism to notify the voidance of the certificate is called Certificate Revocation List (CRL).

✍ ***Definition :*** *A Certificate Revocation List (CRL) is a list of digital certificates that have been revoked by the issuing Certificate Authority (CA) before their scheduled expiration date and should no longer be trusted.*

CRLs provide the blacklist of certificates and are used by PKI clients to verify a certificate's validity and trustworthiness.

4. Certificate Repository : This is a generic database that stores both the valid as well as the invalid certificates. This database stores information about all the issued certificates. In addition to the certificate itself, the database includes validity period and status of each certificate. Certificate revocation is done by updating this database and marking the certificates as unfit for use.

5. Client : These are users, machines or other end point entities that use PKI infrastructure. They could request for new certificates, request certificates for other users (to get their public key) or carry out other certificate operations such as checking the CRL to confirm a certificate's validity.

Steps Involved in a Digital Certificate Generation

Following Fig. 3.8.4 provides the high-level steps involved in certificate generation.

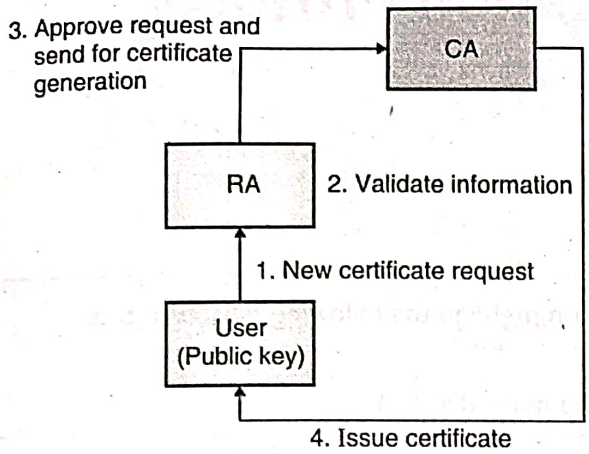

Fig. 3.8.4 : Steps involved in a Digital Certificate Generation

1. Any user, desiring to get a certificate from the CA, sends her information such as name, address, etc. along with her public key to the RA.
2. The RA verifies all the submitted information and approves the certificate request.
3. The RA forwards the information received from the user along with her public key to the CA.
4. The CA computes the hash value of the user's public key and encrypts the hash value with its own private key to generate a digital signature. It then sends all this information to the user in the form of a digital certificate.

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Message Authentication Requirements

Q. 1 What do you mean by message authentication requirements? Why is it important? **(4 Marks)**

Message Authentication Functions

Q. 2 Describe message authentication functions. **(4 Marks)**

Cryptographic Hash Function

Q. 3 What is Hashing? Describe its properties. **(6 Marks)**

Q. 4 With a block diagram, explain how can you use hashing for ensuring message integrity? **(8 Marks)**

Q. 5 Write a short note on avalanche effect in hashing. **(4 Marks)**

Q. 6 Write a short note on collision in hashing. Which hashing algorithm would you suggest using today and why? **(4 Marks)**

Q. 7 With a block diagram, describe SHA-1. **(8 Marks)**

Q. 8 With a block diagram, describe SHA-3. **(8 Marks)**

Q. 9 With a block diagram, describe MD5. **(8 Marks)**

MAC (Message Authentication Code)

Q. 10 What is MAC (Message Authentication Code)? Why is it used over hashing? **(6 Marks)**

Q. 11 What is MAC (Message Authentication Code)? How does it work? **(8 Marks)**

Q. 12 With a block diagram, explain HMAC. **(8 Marks)**

Q. 13 With a block diagram, explain CBC-MAC. **(8 Marks)**

Q. 14 With a block diagram, explain CMAC. **(8 Marks)**

Q. 15 Compare HMAC, CBC-MAC, and CMAC. **(4 Marks)**

Q. 16 Provide a comparison between Hash and MAC. **(4 Marks)**

Security of Hash Functions and MAC

Q. 17 Explain the desired security properties that hash functions should have. **(6 Marks)**

Q. 18 What are some attacks possible on hash functions and MAC? **(4 Marks)**

Digital Certificate - X.509

Q. 19 What is a X.509 Certificate? Why is it used? **(4 Marks)**

Q. 20 What is a X.509 Certificate? Describe its contents. **(8 Marks)**

Public Key Infrastructure (PKI)

Q. 21 What is PKI? List its components. **(8 Marks)**

Q. 22 What is a X.509 Certificate? Describe the steps involved in its generation. **(6 Marks)**

❑❑❑