UNIT III

Cryptography Hash Functions

Information Security | Authentication | PKI | Kerberos | X.509

Unit III — Topics Overview

01
Hash Functions
Features, Properties & Design

02
SHA & MD Algorithms
Popular hash function families

03
SHA-512 Deep Dive
Input formatting to output

04
Message Authentication
MAC, HMAC, Requirements

05
Digital Signatures & PKI
Kerberos, X.509, Key Management

What is a Hash Function?

A hash function is a mathematical function that converts a numerical input value into another compressed numerical value.

Input

Arbitrary length data — any size message or file

HASH FUNCTION

Output

Fixed length hash value (160–512 bits)

- Output is called message digest or hash value
- Hash functions also called compression functions
- n-bit hash function → produces n-bit output
- Popular functions: 160 to 512 bits output

Features of Hash Functions

Fixed Length Output

Hash function converts data of arbitrary length to a fixed length. This process is called hashing the data. Since hash is a smaller representation, it is also referred to as a digest.

Efficiency of Operation

For any hash function h with input x, computation of h(x) is a fast operation. Hash functions are computationally much faster than symmetric encryption.

Key Takeaways

- Hash functions reduce large data to compact fingerprints
- The same input always produces the same hash output
- Even a tiny change in input drastically changes the hash

Properties: Pre-Image Resistance

Pre-Image Resistance

It should be computationally hard to reverse a hash function.

- If a hash function h produced hash value z, it should be a difficult process to find any input x that hashes to z
- This property protects against an attacker who only has a hash value and is trying to find the input
- Also known as One-Way Property — easy to compute h(x), very hard to find x given h(x)

Properties: Second Pre-Image Resistance

Second Pre-Image Resistance

Given an input and its hash, it should be hard to find a different input with the same hash.

- For hash function h with input x producing h(x), it should be difficult to find any input y such that h(y) = h(x)
- Protects against an attacker who has an input value and its hash
- The attacker wants to substitute a different value in place of the original input
- Also known as Weak Collision Resistance

Properties: Collision Resistance

Collision Resistance

It should be hard to find two different inputs of any length that result in the same hash.

- For hash h, it is hard to find any two different inputs x and y such that h(x) = h(y)
- Since hash function is compressing, collisions cannot be eliminated — but should be hard to find

Important Relationship:

If a hash function is collision-resistant, then it is also second pre-image resistant.

Design of Hashing Algorithms

- At the heart of hashing is a mathematical function that operates on two fixed-size blocks of data
- The hash function forms part of the hashing algorithm — block sizes typically 128 to 512 bits
- Each round takes a fixed-size input: combination of the most recent message block and the output of the last round
- The process is repeated for as many rounds as required to hash the entire message

Avalanche Effect

The hash value of each message block becomes input to the next hash operation, creating an avalanche effect — resulting in substantially different hash values for messages that differ by even a single bit.

Popular Hash Functions: MD Family

Message Digest (MD)

MD2

Older, 128-bit hash function

MD4

Faster version, 128-bit

MD5

Most popular, 128-bit, RFC 1321

MD6

Latest variant in family

MD5 was widely used to provide assurance about integrity of transferred files. File servers provided pre-computed MD5 checksums.

Warning: In 2004, collisions were found in MD5. An analytical attack succeeded in an hour using a computer cluster. MD5 is no longer recommended for use.

Popular Hash Functions: SHA Family

Secure Hash Function (SHA)

SHA-0	**160-bit**	1993, NIST. Had weaknesses, not popular.
SHA-1	**160-bit**	1995, corrects SHA-0. Widely used in SSL. Collision found in 2005.
SHA-2	**224–512-bit**	Variants: SHA-224, SHA-256, SHA-384, SHA-512. No successful attacks.
SHA-3	**Variable**	Keccak algorithm, selected by NIST in Oct 2012. Efficient & attack-resistant.

SHA-512: Overview

SHA-512 processes input in stages:

1. **Input Formatting**
   Prepare message with padding & size info
2. **Hash Buffer Init**
   Initialize default values for first block
3. **Message Processing**
   Process each 1024-bit block iteratively
4. **Output**
   Final 512-bit hash value produced

SHA-512: Input Formatting

- SHA-512 has an input size limit — the entire formatted message must be a multiple of 1024 bits
- The formatted message has three parts: Original message + Padding bits + Size of original message
- Messages are processed as 1024-bit blocks

Padding Bits

- Padding bits are appended to get the message to the desired length
- Padding uses '0' bits with a leading '1' (format: 100000...000)
- Padding must always be done — even if only a single '1' bit is added

SHA-512: Buffer Init & Processing

Hash Buffer Initialization

The algorithm processes each 1024-bit block using the result from the previous block. A default value is used for the first block — this becomes the initial hash buffer.

Message Processing

Processing takes each 1024-bit block along with the result of previous processing. This produces an intermediate hash that feeds into the next block.

Output

After every 1024-bit block is processed, each intermediate result feeds the next block. When the final block finishes processing, we have the final 512-bit hash value for the original message.

Applications of Hash Functions

Password Storage

Instead of storing passwords in clear, login processes store hash values. The password file stores pairs (user id, h(P)). Intruders see only hashes — cannot login or reverse-engineer the password.

Data Integrity Check

Checksums generated on data files assure correctness. Users detect any changes made to the original file. Note: does not prove originality — an attacker could replace entire file and recompute hash.

Login Process using Hash:

- User enters password → system hashes it → compares with stored hash
- Pre-image resistance ensures attackers cannot recover original password from hash
- Each user has a unique stored hash value in the password file

Message Authentication Code (MAC)

MAC is a symmetric key cryptographic technique to provide message authentication.

How MAC Works:

- Sender and receiver share a symmetric key K
- Sender inputs message + key K into MAC algorithm → produces MAC value
- MAC compresses arbitrary input to fixed length, using secret key during compression
- Sender transmits message along with MAC value
- Receiver recomputes MAC from received message + shared key K
- If computed MAC matches received MAC → message is authentic

Key Difference: Unlike hash functions, MAC uses a secret key during compression.

Limitations of MAC

Establishment of Shared Secret

MAC can only provide authentication among users who have pre-shared a key. Establishing shared secret prior to MAC use is required, creating a key distribution challenge.

Inability to Provide Non-Repudiation

MAC cannot prove message was sent by the sender. Either party could have computed the MAC, so the sender can deny having sent a message — the receiver cannot prove otherwise.

Solution

Both these limitations can be overcome by using public key based digital signatures, which provide non-repudiation and don't require pre-shared secret keys between parties.

Message Authentication Requirements

Types of Threats & Attacks:

- Revelation — releasing message content to unauthorized parties without appropriate key
- Traffic Analysis — determining patterns and frequency of connections between parties
- Deception — adding fraudulent messages into communication network
- Content Modification — inserting, deleting, or changing message content
- Sequence Modification — changing the order/insertion/deletion of messages
- Timing Modification — replay and delay of messages, disrupting session tracking
- Source Refusal — source denies being the originator of a message
- Destination Refusal — receiver denies receiving the message

Message Authentication Functions

Two Functional Levels:

Lower Level

A function that produces an authenticator — the value that will further help in authenticating a message.

Higher Level

Uses the lower level function to help receivers verify the authenticity of messages.

Three Classes of Functions:

- Message Encryption — converts data to ciphertext (symmetric or public key)
- Message Authentication Code (MAC) — security code verifying integrity and authenticity
- Hash Function — mathematical function converting input to fixed-length digest

HMAC — Hash-based MAC

HMAC: Hashed Message Authentication Code

HMAC is a result of work on developing a MAC derived from cryptographic hash functions. It uses hashing twice, providing great resistance to cryptanalysis attacks.

HMAC Objectives:

- One-way function — easy to generate output from input but complex in reverse
- Less affected by collisions than regular hash functions
- Reuses algorithms like MD5 and SHA-1; allows replacing hash functions with more secure ones
- Handles keys in a simpler manner
- Standardized in RFC 2104; mandated in IP security (IPsec); covered by FIPS 198 NIST standard

HMAC Algorithm

- Takes message M containing blocks of length b bits
- An input signature (ipad) is padded to the left of the message
- Entire input fed into hash function → temporary message digest MD'
- MD' is appended to an output signature (opad)
- The combined value is hashed again → final message digest MD

$$HMAC(K, M) = H((K \oplus opad) \,||\, H((K \oplus ipad) \,||\, M))$$

Two hashing passes provide extra security layer compared to single hash MAC

Digital Signatures

What is a Digital Signature?

A cryptographic mechanism that provides authentication, non-repudiation, and integrity. Created using sender's private key — only they could have generated it.

Advantages Over MAC

Overcomes MAC limitations: no shared secret required, provides non-repudiation — sender cannot deny signing, and third parties can verify the signature.

Digital Signatures solve message authentication threats:

- Content, sequence, and timing of messages can be monitored
- Prevents source denial — sender cannot claim they didn't send it
- Combination with protocols handles destination denial

Public Key Infrastructure (PKI)

PKI provides assurance and identification of public keys and their distribution.

Components of PKI:

Digital Certificate	Public key certificate (X.509)
Private Key Tokens	Secure storage for private keys
Certification Authority	Issues and signs certificates
Registration Authority	Verifies identity before cert issuance
Certificate Management	Publishes, renews, revokes certificates

Public Key Cryptosystem

- Asymmetric algorithms use one key for encryption and a different but related key for decryption
- Computationally infeasible to determine the decryption key from only the encryption key
- Keys are typically large: 512, 1024, 2048 bits or more
- Keys are stored in secure devices such as USB tokens or hardware security modules

Ingredients of Public Key Encryption:

- Plaintext — original readable message
- Encryption Algorithm — performs conversions on plaintext
- Public and Private Keys — one for encryption, other for decryption
- Ciphertext — scrambled output based on plaintext and key
- Decryption Algorithm — recovers original plaintext from ciphertext

Key Distribution: Confidentiality & Auth

A and B have exchanged public keys. The following protocol ensures both confidentiality and authentication:

Step 1	A encrypts message with B's public key containing A's identifier (IDA) and nonce N1
Step 2	B sends back N1 + new nonce N2 encrypted with A's public key. Presence of N1 assures A that correspondent is B
Step 3	A returns N2 encrypted with B's public key — assures B that correspondent is A
Step 4	A sends secret key Ks: E(PUb, E(PRa, Ks)) — ensures only B can read it, only A could send it
Step 5	B computes D(PUa, D(PRb, M)) to recover secret key Ks

Hybrid Key Distribution Scheme

A hybrid approach used on IBM mainframes combines symmetric and public-key techniques:

Performance Advantage

Session keys change frequently. Public-key distribution would degrade performance due to high computational load. Public key is used only occasionally to update master keys.

Backward Compatibility

The hybrid scheme is easily overlaid on an existing Key Distribution Center (KDC) scheme with minimal disruption or software changes.

Three-Level Hierarchy:

- Level 1: Public-key encryption to distribute master keys (rare, high security)
- Level 2: KDC distributes session keys encrypted with master key
- Level 3: Session keys used for actual data encryption

Kerberos Authentication

Kerberos provides a centralized authentication server to authenticate users to servers and servers to users. It runs as a trusted third-party server — the Key Distribution Center (KDC).

Core Components:

Authentication Server (AS)

Performs initial authentication and issues Ticket Granting Tickets (TGT)

Database

Stores access rights and credentials. AS verifies users against this database.

Ticket Granting Server (TGS)

Issues service tickets that allow users to access specific servers.

Each user and service on the network is a principal in the Kerberos realm.

Kerberos: Authentication Steps

1. User logs in and requests services → sends request for ticket-granting service
2. AS verifies user in database → sends TGT and session key (encrypted with user's password)
3. User decrypts with password → sends TGT + authenticator (username, network address) to TGS
4. TGS decrypts ticket, verifies authenticator → creates service ticket for requested server
5. User sends service ticket + authenticator to the target server
6. Server verifies ticket and authenticator → grants access to the service

Kerberos: Limitations

- Each network service must be modified individually for use with Kerberos
- Does not work well in a timeshare environment
- Requires a secure, always-on Kerberos server — single point of failure
- Stores all passwords encrypted with a single key — compromise has wide impact
- Assumes workstations are secure
- May result in cascading loss of trust
- Scalability issues in very large environments

Despite limitations, Kerberos remains the best access security protocol available. The protocol is flexible enough to employ stronger encryption algorithms to combat new threats.

Kerberos: Advantages

Access Control

Single point for tracking all logins and enforcing protection policies

Mutual Authentication

Both client and server verify each other's authenticity during all steps

Limited Ticket Lifetime

Each ticket has timestamps and lifetime — authentication period managed by admins

Reusable Authentication

Each user is verified once per session — no repeated authentication

Security

Multiple secret keys, third-party authorization, and cryptography. Passwords never sent over network

Performance

Keeps client info after verification; outperforms NTLM on large farms

X.509 Authentication Service

X.509 is a digital certificate standard built on the ITU X.509 standard, defining the format of PKI certificates. Used for secure transaction processing and private information.

How It Works:

- Core of X.509 is the public key certificate associated with each user
- Certificates are produced by a trusted Certification Authority (CA)
- Directory servers provide an accessible location for users to acquire certificates
- X.509 uses ASN.1 (Abstract Syntax Notation) for its format
- Once issued, the certificate is attached to the user like an identity card
- The certificate is presented like an identity card at the resource requiring authentication

X.509 Certificate Format

Version Number:	Defines the X.509 version of the certificate
Serial Number:	Unique number issued by the Certification Authority
Signature Algorithm ID:	Algorithm used for signing the certificate
Issuer Name:	X.500 name of the CA that signed the certificate
Period of Validity:	Duration for which the certificate remains valid
Subject Name:	Name of the user to whom certificate is issued
Subject's Public Key:	Public key and the algorithm for which it should be used
Extension Block:	Additional standard information fields
Signature:	Hash of all other fields, encrypted with CA's private key

X.509 Applications

Document Signing

Digital signatures on documents using certificate-bound private key

Web Security (TLS/SSL)

HTTPS encryption for web servers — most common use case

Email Certificates

Signing and encrypting email communications (S/MIME)

Code Signing

Verifying software publisher identity and code integrity

SSH Keys

Secure Shell Protocol key-based authentication

Digital Identities

Online identity verification and authentication systems

Many protocols depend on X.509, making it the foundation of internet security.

Key Management

The security of any cryptosystem depends on how securely its keys are managed. Cryptographic schemes are rarely compromised through design weaknesses — they are often compromised through poor key management.

Two Key Requirements for Public Key Cryptography:

Secrecy of Private Keys

Throughout the key lifecycle, secret keys must remain secret from all parties except the owner and those authorized to use them.

Assurance of Public Keys

Public keys are open domain. Key management must focus on ensuring: the key is correct, who it is associated with, and what it can be used for.

The most crucial requirement — assurance of public key — is achieved through PKI.

Certification Authority (CA)

CA takes responsibility for identifying client identity and issuing digital certificates, which it digitally signs.

Key Functions of CA:

- Generating Key Pairs — CA may generate independently or jointly with the client
- Issuing Digital Certificates — verifies identity, then signs the certificate
- Publishing Certificates — via electronic directory or direct distribution
- Verifying Certificates — makes public key available to assist verification of signatures
- Revocation of Certificates — revokes certificates if private key is compromised; maintains Certificate Revocation List (CRL)

Classes of Certificates: Class 1 (email only) → Class 2 (personal info) → Class 3 (identity checks) → Class 4 (government/financial)

RA & Certificate Management

Registration Authority (RA)

CA may delegate identity verification to a third-party RA. The RA appears to clients as a CA but does not actually sign certificates. Performs necessary checks on requestors.

Certificate Management System (CMS)

System through which certificates are published, suspended, renewed, or revoked. Certificates are not deleted — they may be needed as proof for legal reasons. CA and RA run CMS together.

Private Key Tokens:

- Storing private key on personal computer is not recommended — attacker gaining access exposes private key
- Private keys stored on secure removable storage tokens protected by password
- Common formats: Entrust uses .epf; Verisign, GlobalSign, Baltimore use .p12 format

Digital Certificate Deep Dive

A digital certificate is the electronic equivalent of an identity card — issued to people, computers, software, or any entity needing to prove identity.

What a Digital Certificate Contains:

- Client's public key (stored by the Certification Authority)
- Client information (name, organization, email)
- Certificate expiration date
- Usage details — what the key can be used for
- Issuer (CA) information
- CA's digital signature covering all above fields

Based on ITU standard X.509 — hence also called X.509 certificates.

Comparison: Hash vs MAC vs Digital Signature

Comparison

Feature	Hash Function	MAC	Digital Signature
Key Used	None	Symmetric (shared)	Asymmetric (private)
Confidentiality	No	No	Optional
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Algorithm Speed	Very Fast	Fast	Slower
Key Distribution	Not needed	Pre-shared required	Public key infrastructure

Summary: Hash Functions & Algorithms

Hash Functions

- Convert arbitrary input to fixed-length output
- Key properties: pre-image, 2nd pre-image, collision resistance
- Avalanche effect ensures sensitivity to input changes

SHA Family

- SHA-0 → SHA-1 → SHA-2 → SHA-3
- SHA-512: 1024-bit blocks, padding, 512-bit output
- SHA-3 (Keccak) — current standard since 2012

MAC & HMAC

- MAC: symmetric key, encrypted checksum
- HMAC: double-hashing, RFC 2104, FIPS 198
- Limitations: shared secret + no non-repudiation

Authentication

- 8 types of authentication threats
- 3 classes: encryption, MAC, hash
- Digital signatures overcome MAC limitations

Summary: PKI, Kerberos & X.509

Kerberos

- Centralized KDC with AS + TGS
- 6-step authentication process using tickets
- Advantages: mutual auth, reusable, time-limited tickets
- Limitations: single point of failure, workstation trust

X.509

- ITU standard for PKI certificate format
- Contains version, serial, issuer, validity, public key
- Applications: TLS, email, code signing, SSH
- Foundation of internet security

PKI & Key Management

- PKI components: Digital Cert, Private Key Tokens, CA, RA, CMS
- CA functions: generate, issue, publish, verify, revoke certificates
- 4 certificate classes; RA performs identity checks
- Key lifecycle management: secrecy of private keys + assurance of public keys

UNIT III

Key Takeaways

- Hash functions provide data integrity through one-way compression
- MAC adds authentication; HMAC adds double-hash security
- Digital signatures solve non-repudiation that MAC cannot
- Kerberos centralizes authentication using tickets and a trusted KDC
- X.509 certificates form the backbone of PKI and internet trust
- Key management is as important as the cryptographic algorithm itself