Cryptography & Network Security(BCS072)

Unit – 3 Message Authentication Codes

EduShine CLASSES By- ARMAN ALI

III	Message Authentication Codes: Authentication requirements, authentication functions, message authentication code, hash functions, birthday attacks, security of hash functions, Secure hash algorithm (SHA) Digital Signatures: Digital Signatures, Elgamal Digital Signature Techniques, Digital signature standards (DSS), proof of digital signature algorithm,

EduShine Classes – Arman Ali

Download Notes : https://rzp.io/rzp/ENyaHHe

Cryptography & Network Security(BCS072)

EduShine CLASSES By- ARMAN ALI

🔐 What is Authentication?

Authentication means **verifying who someone is**.

It checks whether the person (or computer/system) really is **who they claim to be** before giving access.

💡 Simple Example:

When you log in to your Gmail account —

You enter your **username and password** → Google checks if it's really you.

✅ If correct → You get access.

❌ If wrong → Access denied.

That process is **authentication**.

1.Single-Factor Authentication (SFA)

You use **only one method** to prove your identity.

✅ Example:

You just enter your **password** to log in.

⚫ *Not very secure* — if someone knows your password, they can log in.

Cryptography & Network Security(BCS072)

EduShine CLASSES By- ARMAN ALI

2. Two-Factor Authentication (2FA)

You use **two different methods** for login.
Usually, it combines:

- **Something you know** (password)
- **Something you have** (OTP, phone, or email code)

✅ **Example:**

You enter your **password**,
then enter the **OTP** you receive on your phone.

Much safer than single-factor

● *Takes a few more seconds*

3. Multi-Factor Authentication (MFA)

You use **two or more** authentication methods together.
It can combine:

- Password (something you know)
- OTP or smart card (something you have)
- Fingerprint or face (something you are)

✅ **Example:**

To unlock a secure app →
you enter **password + OTP + fingerprint**.

Highly secure

● *Can be complex or need special devices*

Cryptography & Network Security(BCS072)

EduShine CLASSES
By- ARMAN ALI

What is Message Authentication Code (MAC)?

A Message Authentication Code (MAC) is a small piece of information (a code) used to check both authenticity and integrity of a message.

It ensures that:

- The message really came from the correct sender (authenticity).
- The message was not changed or tampered during transmission (integrity).

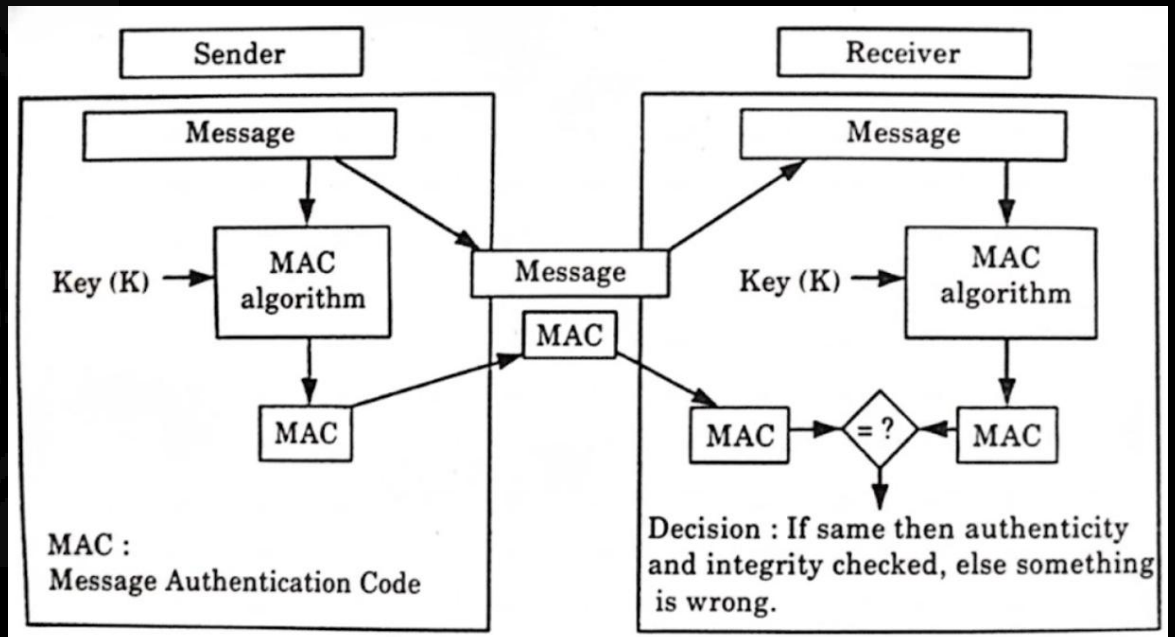


Fig. 3.1.1. Message Authentication Code.

❖ How it Works (from the diagram)

➤ Step 1: At Sender Side

- Sender has a message and a secret key (K).
- Both message and key are given to a MAC algorithm.
- The algorithm produces a MAC value (code).
- The sender sends message + MAC to the receiver.

➤ Step 2: At Receiver Side

- Receiver gets the message + MAC.
- Receiver also uses the same secret key (K) with the same MAC algorithm to create a new MAC.
- The receiver then compares the received MAC with the newly generated MAC.

➤ Step 3: Decision

- If both MACs are same → Message is authentic and not changed. ✓
- If MACs are different → Message has been modified or fake. ✗

🛠 Authentication Requirements in MAC

To make MAC work properly, it must satisfy these requirements:

1. **Authentication of origin:**
   The receiver should be sure that the message really came from the sender who knows the secret key.
2. **Data integrity:**
   It should detect if **any bit** in the message is changed during transmission.
3. **Confidentiality (optional):**
   Sometimes the message may also be encrypted for extra security.
4. **Resistance to forgery:**
   No attacker should be able to **create a valid MAC** without knowing the secret key.

❖ Different types of models of Message Authentication Code (MAC) :

💡 What is a MAC model?

A **MAC model** means the way in which a system **creates and checks** the Message Authentication Code (MAC).

There are **three main models** of MAC generation:

1. **MAC without encryption**
2. **MAC with internal error code**
3. **MAC with external error code**

1. MAC without Encryption

In this model, the **MAC is created directly from the message** and the **secret key**, without doing any encryption of the message.

👉 How it works:

- The sender uses a **MAC algorithm** (like hash + key) to create a MAC.
- The message is sent as it is (not encrypted).
- The receiver uses the same key to check the MAC.

👉 **Example:**

Imagine you send a text message and a small security code made using a shared secret key. The message is not hidden (anyone can read it), but no one can **change** it without being caught.

👉 **Use:**

When **confidentiality is not required**, but **integrity** and **authenticity** are important.

2. MAC with Internal Error Code

Here, the **message is encrypted first**, and then a **MAC is generated from the encrypted data**.

👉 **How it works:**

Message → **Encrypted using key**

The **encrypted message** is given to the **MAC algorithm**

MAC is produced and sent along with the message

👉 **Benefit:**

Even if someone tries to change the message, both encryption and MAC will detect it.

👉 **Use:**

When we need both **confidentiality** and **integrity** together.

👉 **Example:**

You send a locked (encrypted) file + a code created from that locked file.

If someone tries to open or modify it, the code will not match.

3. MAC with External Error Code

In this model, the **MAC is generated from the plain message first**, and then **the message is encrypted**.

👉 **How it works:**

Message → MAC generated → Then **message and MAC** are both **encrypted together**.

👉 **Benefit:**

Provides **strong security** because both the message and its MAC are hidden.

👉 **Use:**

Used in systems where **high-level security** is required — for example, banking or military communication.

Cryptography & Network Security(BCS072)

👉 Example:

You create a message, attach its security code, and then lock both together in a box (encrypt). Anyone who doesn't have the key cannot even see the message or MAC.

Model	Description	Security Provided
1. MAC without encryption	MAC is generated directly from message and key. Message is not encrypted.	Provides integrity & authenticity only.
2. MAC with internal error code	Message is first encrypted, then MAC is generated from encrypted data.	Provides confidentiality, integrity & authenticity.
3. MAC with external error code	MAC is generated first, then both message and MAC are encrypted together.	Provides very strong security (confidentiality + integrity + authenticity).

Cryptography & Network Security(BCS072)

EduShine CLASSES By- ARMAN ALI

💡 What is HMAC and CMAC?

Both **HMAC** and **CMAC** are special types of **Message Authentication Codes (MACs)** used to ensure:

- The message is **not changed** (integrity)
- The message is from a **real sender** (authenticity)
- They differ only in **how they create the MAC**.

🔐 1. HMAC — Hash-based Message Authentication Code

👉 **Full form:** HMAC = **Hash-based Message Authentication Code**

HMAC uses a **hash function** (like SHA-256 or MD5) and a **secret key** to make a special code (MAC) for the message.

It combines **hashing** and **keyed security**.

Example :

Let’s say you want to send a message "HELLO" securely.

1.You and your friend share a **secret key (K)**.

2.You apply a **hash function** like SHA-256 on the combination of the key and message:

HMAC = Hash(Key + Message)

3. You send both the **message** and the **HMAC**.
4. Your friend uses the same key and does the same hash process.

- If the HMAC matches, message is **authentic and not changed**.

✅ **Advantages of HMAC:**

- Very fast and easy to implement.
- Uses **existing hash functions** (like SHA-256).
- Protects message integrity and authenticity.

❌ **Disadvantages:**

- Does **not provide encryption** (message is still visible).
- Depends on the strength of the hash function used.

🔒 2. CMAC — Cipher-based Message Authentication Code

☞ **Full form:** CMAC = **Cipher-based Message Authentication Code**

CMAC uses a **block cipher** (like AES or DES) instead of a hash function to create a MAC.
So, instead of "hashing" the message, it "encrypts" it in a special way to produce the authentication code.

Simple steps (easy example):

1. You and your friend share a **secret key (K)**.
2. You use **AES algorithm** on the message with that key to generate a code (MAC).
3. You send the **message + MAC**.
4. Receiver uses the same AES process with the same key to verify the MAC.

If both match → ✓ message is genuine.
If not → ✗ message was changed.

Cryptography & Network Security(BCS072)

✓ Advantages of CMAC:

Stronger than HMAC because it uses **block cipher (AES)**.
Provides **high security** and resistance to attacks.
Perfect for systems already using AES.

✗ Disadvantages:

Slower than HMAC because encryption is heavier.
Needs block cipher implementation (more complex).

Cryptography & Network Security(BCS072)

EduShine CLASSES By- ARMAN ALI

Feature	HMAC	CMAC
Full form	Hash-based Message Authentication Code	Cipher-based Message Authentication Code
Algorithm used	Uses hash functions (like SHA-256, MD5)	Uses block ciphers (like AES, DES)
Speed	Faster	Slower
Security	Depends on hash function strength	Stronger (uses encryption)
Used in	Internet messages, APIs, web services	Banking, smart cards, secure hardware
Main purpose	Integrity and authenticity	Integrity and authenticity

Cryptography & Network Security(BCS072)

EduShine CLASSES By- ARMAN ALI

💡 What is a Hash Function?

A **hash function** is a method that takes any size of data (message, file, text, etc.) and converts it into a **fixed-length code** called a **hash value** or **digest**.

✅ Example :

Let's say you have a message:

👉 "HELLO WORLD"

A **hash function** (like SHA-512) will convert it into something like this:

👉 2CF24DBA5FB0A... (a 512-bit long code)

Even if you change a single letter (e.g., "HELLO WORLd"), the hash value will change completely!

🎯 Purpose of Hash Functions

Hash functions are mainly used for:

- ✓ **Integrity check** – to confirm that data is not changed.
- ✓ **Digital signatures** – to verify sender authenticity.
- ✓ **Password protection** – storing hashed passwords.
- ✓ **Message Authentication Codes (HMAC)** – as seen earlier.

Cryptography & Network Security(BCS072)

EduShine CLASSES
By- ARMAN ALI

🔐 What is SHA-512?

👉 **SHA** = Secure Hash Algorithm

👉 **512** = Output hash is **512 bits long**

SHA-512 is part of the SHA-2 family developed by **NIST (National Institute of Standards and Technology)**.

It is a **very strong and secure hash function** used in modern cryptography.

Block diagram of SHA-512 :

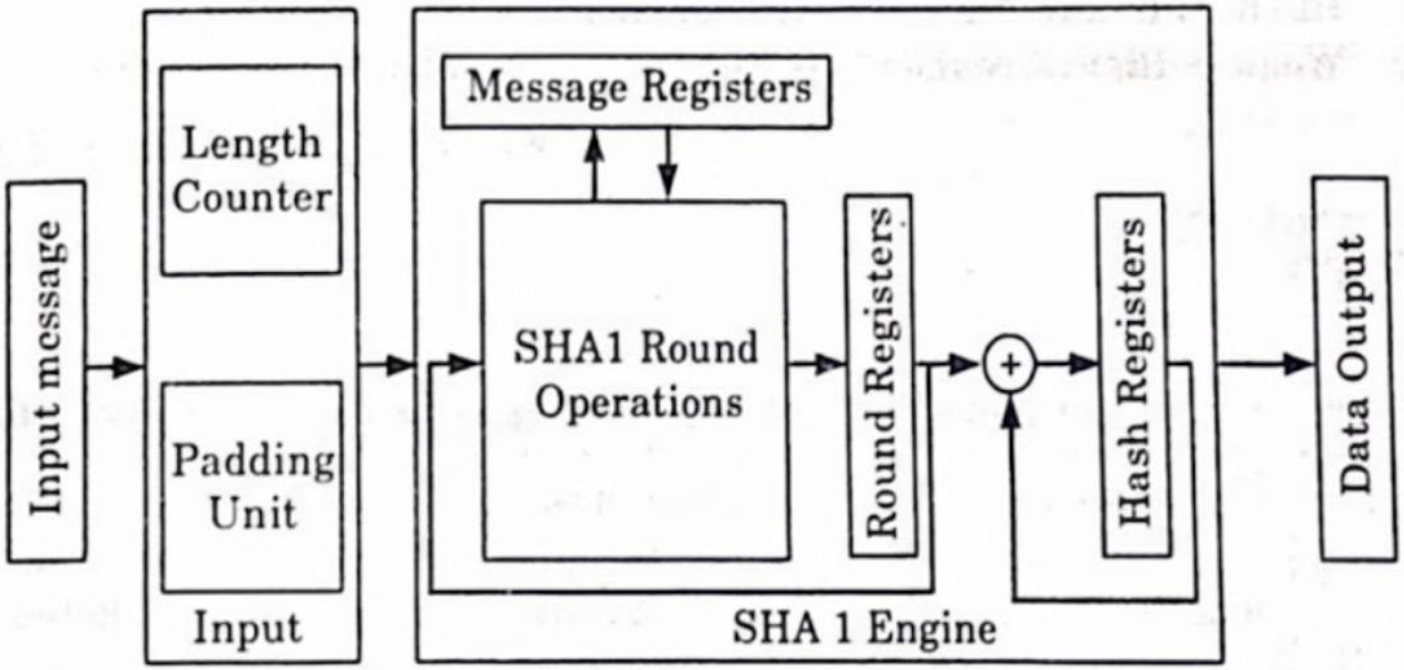


Fig. 3.9.1.

❖ Working Steps of SHA-512 :

Let's go step-by-step using your diagram 👇

Step 1: Input Message

This is the message you want to hash (e.g., your text, file, or password).

Example:

"HELLO"

Step 2: Padding Unit

The message is not always of the right length, so:

- Extra bits (0s and 1s) are **added (padded)** at the end
- Padding makes the total message length a **multiple of 1024 bits**

This step ensures that the message fits perfectly for processing.

Step 3: Length Counter

Here the **length of the original message** is recorded (in 128 bits).

This helps SHA know how long the message was before padding.

Step 4: Divide Message into Blocks

Now, the padded message is **split into blocks** of **1024 bits** each.

Each block will be processed one by one through the SHA-512 engine.

Step 5: SHA-1 Engine

Even though it says “SHA1 Engine” in diagram, it means the **core processing unit** that performs multiple rounds of mathematical operations.

Each block goes through **80 rounds of processing**.

Inside the Engine:

Let’s understand the **round operations** (in simple terms):

❑ **Components inside the engine:**

- **Message Registers** – Hold message block bits temporarily.
- **Round Registers** – Used to store intermediate results.
- **Hash Registers** – Hold the final hash values.

Step 6: Round Operations

Each round performs several logical and arithmetic functions:

1. **Bitwise Operations** – like AND, OR, XOR
2. **Shifts and Rotations** – move bits left/right
3. **Additions** – modular additions (mod 2^64)
4. **Compression Function** – mixes the data with constants and previous hash

Each round makes the hash more random and irreversible.

Step 7: Add Round Result to Hash Registers

After every 80 rounds, the result is **added to previous hash registers** (cumulative effect).
This means each block affects the next block — so even a small change in input affects the entire output hash.

Step 8: Output (Final Hash Value)

After processing all blocks, the final **512-bit hash value** is produced.
That's the **unique fingerprint** of your message.

💡 Security of Hash Function :

How safe the hash function is against attackers trying to **break** or **cheat** it.

A **secure hash function** must make it extremely hard to:

- find two different inputs that give the same hash,
- guess the input from the hash,
- or modify data without changing its hash.

🔒 Three Main Security Properties

1. Preimage Resistance (One-way property)

Given a hash value **h**, it should be **hard to find any message M** such that hash(M) = h

Example:

You know the hash “A12BC3”, but can’t guess the original message “HELLO”.

If you can’t reverse it, the hash is one-way.

✓ **Protects against:** attackers trying to find the original message.

2. Second Preimage Resistance

Given one message M_1, it should be **hard to find another message M_2** such that hash(M_1) = hash(M_2)

Example:

If a file has hash X, no one should find another file with the same hash X.

✓ **Protects against:** data replacement attacks.

3. Collision Resistance

It should be **hard to find any two different messages** M_1 and M_2 that produce the same hash.

Example:

If two different inputs “Cat” and “Dog” give the same hash, that’s a **collision**.

A secure hash makes that almost impossible.

✓ **Protects against:** fake document or signature attacks.

✓ **Common Secure Hash Functions**

SHA-256, SHA-384, SHA-512, SHA-3 → Secure

Cryptography & Network Security(BCS072)

🎂 What is a Birthday Attack?

A **birthday attack** is a type of **cryptographic attack** that tries to **find two different messages** that produce the **same hash value**.

This is called a **collision**.

It is based on a simple idea from **the birthday paradox** in probability.

🎉 Step-by-step Easy Explanation:

1. Hash function creates a fixed-size output (called a hash) from a message.

Example:

H("Hello") → 2cf24dba5fb0a...

H("Hello!") → 9a0b8c4e2d3f...

2. Normally, we expect that **different messages** should give **different hashes**.

3. But sometimes, by chance, **two different messages** may give the **same hash**.

That’s called a **collision**.

4. A **birthday attack** uses **mathematical probability** to find such a **collision** faster than normal guessing.

🎈 Why it's called a "Birthday" attack?

Because it's based on the **Birthday Paradox** —
In a group of just **23 people**, there is a **50% chance** that **two people have the same birthday**,
even though there are **365 days** in a year!

👉 Similarly, for hash functions, you don't need to try *all* possible values — you can find two messages with the same hash much faster.

🔒 How to Prevent Birthday Attacks

✓ Use **strong hash functions** (like SHA-256 or SHA-512).
✓ Avoid weak ones like **MD5** or **SHA-1**.
✓ Use **longer hash outputs** — longer hashes mean fewer collisions.

Cryptography & Network Security(BCS072)

✍ What is a Digital Signature?

A **Digital Signature** is like an **electronic version of your handwritten signature**, but it is **much more secure** because it uses **mathematics and cryptography**.

It helps to:

- ✓ **Prove the identity** of the sender (Authentication)
- ✓ **Ensure message is not changed** (Integrity)
- ✓ **Prevent denial** (Non-repudiation — sender can't deny later)

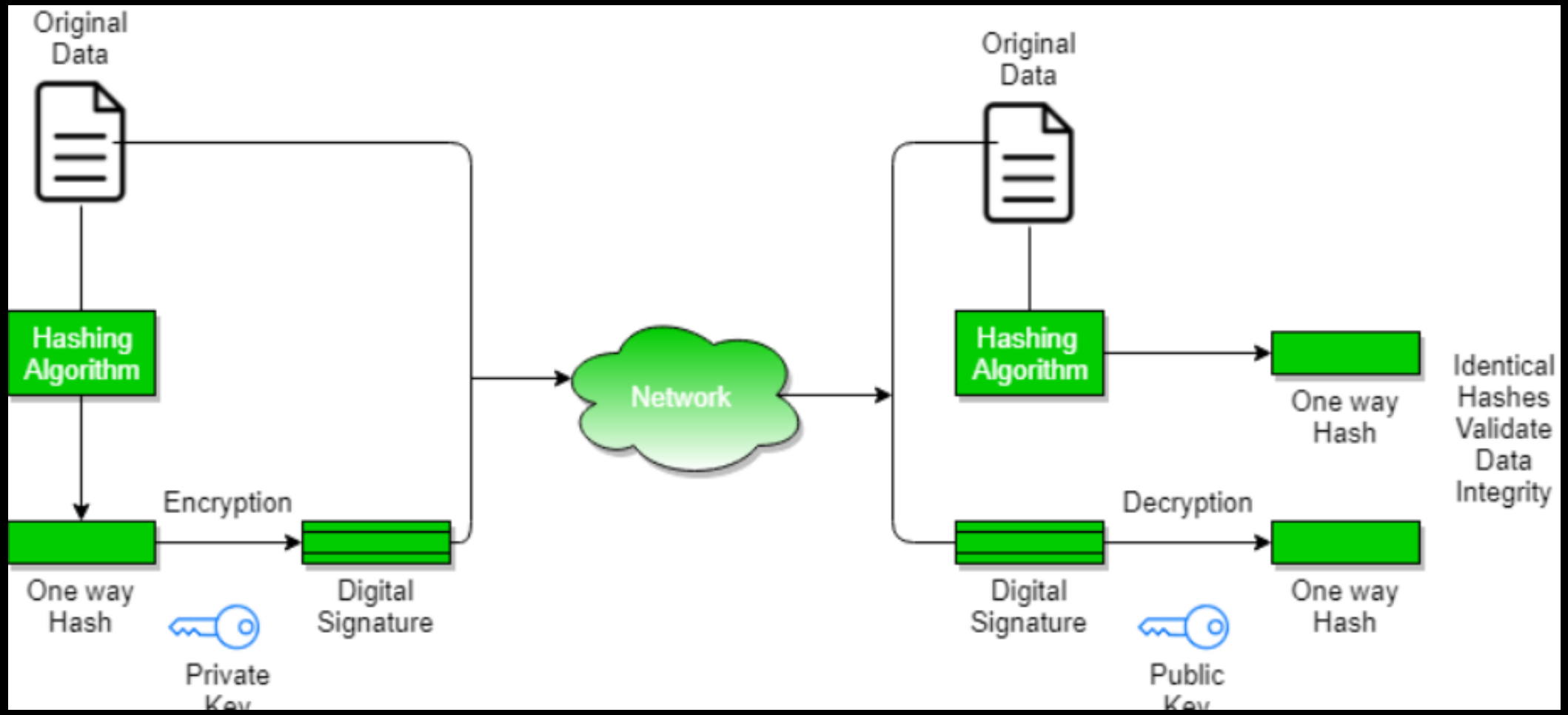
💡 Simple Example

Imagine you send an important message — "Pay ₹5000 to Arman."

If you send it directly over the internet, someone could **change it** to "Pay ₹50,000 to Arman." 😂

So, you use a **Digital Signature** to protect it.

❖ How Digital Signature Works?



Cryptography & Network Security(BCS072)

EduShine CLASSES By- ARMAN ALI

1. **Message creation:**
   You write your message (e.g., “Pay ₹5000 to Arman”).
2. **Hashing:**
   The message is passed through a **hash function** (like SHA-256) to create a small fixed-size code called a **hash value**.

Example: H("Pay ₹5000 to Arman") = 5A3C7F...

3. Signing with Private Key:

The sender uses their **private key** to **encrypt** this hash value.

This encrypted hash is called the **Digital Signature**.

4. Sending Message + Signature:

The sender sends both the **original message** and the **digital signature**.

5. Verification:

The receiver uses the **sender’s public key** to **decrypt** the signature and get the hash value.

Then, the receiver creates a new hash of the received message and compares both.

1. If both match → ✓ Message is real and not changed.

2. If not → ✗ Message is fake or tampered.

Key Idea

1. **Private Key** → Used to *sign* the message
2. **Public Key** → Used to *verify* the message

❖ Digital Signature Algorithm (DSA)

DSA stands for **Digital Signature Algorithm**.

It is one of the most commonly used algorithms to create and verify digital signatures.

It was developed by **NSA (National Security Agency)** in the USA and approved by **NIST**.

⚙ Steps in DSA

1. Key Generation:

The sender generates two keys:

🔒 **Private Key** – kept secret

🔑 **Public Key** – shared with others

2. Signing:

The sender takes the **message**, creates a **hash** of it, and uses the **private key** to make a

digital signature.

3. Verification:

The receiver uses the **public key** to verify the **signature** and make sure the message is original and unaltered.

🎁 Example

Let's say Arman sends a file to his teacher.

- Arman creates a hash of the file.
- He signs it using his **private key** (DSA algorithm).
- The teacher uses Arman's **public key** to verify it.

If the hash values match, the teacher knows:

- The file is **really from Arman**
- It has **not been modified**

Cryptography & Network Security(BCS072)

✓ Advantages of Digital Signatures

Advantage	Description
Authentication	Confirms who sent the message
Integrity	Ensures message not changed
Non-repudiation	Sender cannot deny later
Security	Very hard to fake due to cryptography

⚠ Limitations

Limitation	Description
Key management	Private key must be stored safely
Complex math	Needs strong algorithms and computing power
Revocation	If private key is lost, signature becomes invalid

Cryptography & Network Security(BCS072)

❖ ElGamal Digital Signature

Now, **ElGamal Digital Signature** is **one type** of digital signature algorithm based on **modular arithmetic and discrete logarithm** (just like in cryptography).

❖ Main Idea of ElGamal Digital Signature

ElGamal signature is based on a simple idea:

- You generate a pair of keys – **Private Key** (secret) and **Public Key** (shared with everyone). Then you use the **private key to sign** a message, and others can **verify it using your public key**.

◆ Steps :

Let's break it step by step 👇

Step 1: Key Generation

- Choose a large **prime number p**.
- Choose a **generator g** (any number smaller than p).
- Choose a **private key x** (a secret number).
- Compute the **public key y = g^x mod p**.

Now you have:

- **Public key** → (p, g, y)
- **Private key** → x

📝 Step 2: Signature Generation

Suppose you want to sign a message **M**.

1. Choose a random number **k** such that **k** and **p–1** are coprime.
2. Compute **r = g^k mod p**.
3. Compute **s = (H(M) – x·r) × k^{-1} mod (p–1)**

- Here **H(M)** = hash of message M
- **k^{-1}** = modular inverse of k

Your **digital signature** is the pair **(r, s)**.

✅ Step 3: Signature Verification

To check if the signature is valid, the receive does:

1.Compute **v1 = y^r × r^s mod p**

2.Compute **v2 = g^{H(M)} mod p**

If **v1 = v2**, then the signature is **valid** — the message is from the real sender and not changed.

Example

Let's take small numbers (for understanding only):

- p = 23
- g = 5
- x = 6 → private key
- y = 5^6 mod 23 = 8 → public key

Now, you want to sign message **M = 10**

1. Choose random **k = 7**
2. r = 5^7 mod 23 = 17
3. s = (10 – 6×17) × 7^{-1} mod 22
   (you calculate and get some s value)

Your signature = (r, s)

Now receiver uses your public key to check if **v1 = v2**.

If yes → signature valid ✓

Key Points to Remember

- It ensures **authenticity** (sender is real).
- It ensures **integrity** (message not changed).
- Based on **discrete logarithm problem** (very hard to break).
- Used in **digital security systems, authentication, and blockchain**.

Cryptography & Network Security(BCS072)

🌟 What is Digital Signature Standard (DSS)?

👉 The **Digital Signature Standard (DSS)** is a **set of rules** made by the **U.S. government (NIST)** for creating **digital signatures**.

It defines **how to generate and verify** a digital signature safely and correctly.

DSS ensures that your **digital signature is secure, authentic, and trusted** — just like a handwritten signature but in digital form.

❖ Main Idea

DSS is **not one algorithm** — it's a **standard** that allows **different algorithms** for digital signing.

The **main algorithm** used in DSS is the **DSA (Digital Signature Algorithm)**.

So basically:

- **DSS = Standard (rules)**
- **DSA = Algorithm (used inside DSS)**

❖ Components of DSS

DSS defines **three main parts** of a digital signature system:

- **Key Generation** – How to create public and private keys
- **Signature Generation** – How to sign the message
- **Signature Verification** – How to check if the signature is valid

🔑 Working (in Simple Words)

Step 1: Key Generation

- The sender creates a **public key** and a **private key**.
- The **private key** is secret (used for signing).
- The **public key** is shared (used for verification).

Step 2: Signature Generation

- The message is **hashed** (using SHA algorithm).
- The sender uses their **private key** to sign that hash.
- The result is a **digital signature**.

Step 3: Signature Verification

- The receiver also creates the hash of the received message.
- Then uses the **sender's public key** to check if the signature is valid.
- If the two hashes match, ✓ message is authentic and unchanged.

📒 **Example (Easy)**

Imagine you (Arman) are sending a project file to your teacher.

- You sign it digitally using your **private key**.
- The teacher uses your **public key** to verify.

  If valid → teacher knows the file really came from you and was not changed. ✓

This process follows the **Digital Signature Standard (DSS)**.

Thank You....