Digital Signature is an encrypted Hash Code

Question? What is the orginal data of the hash code?

A a file
B an electronic signature
C contract message

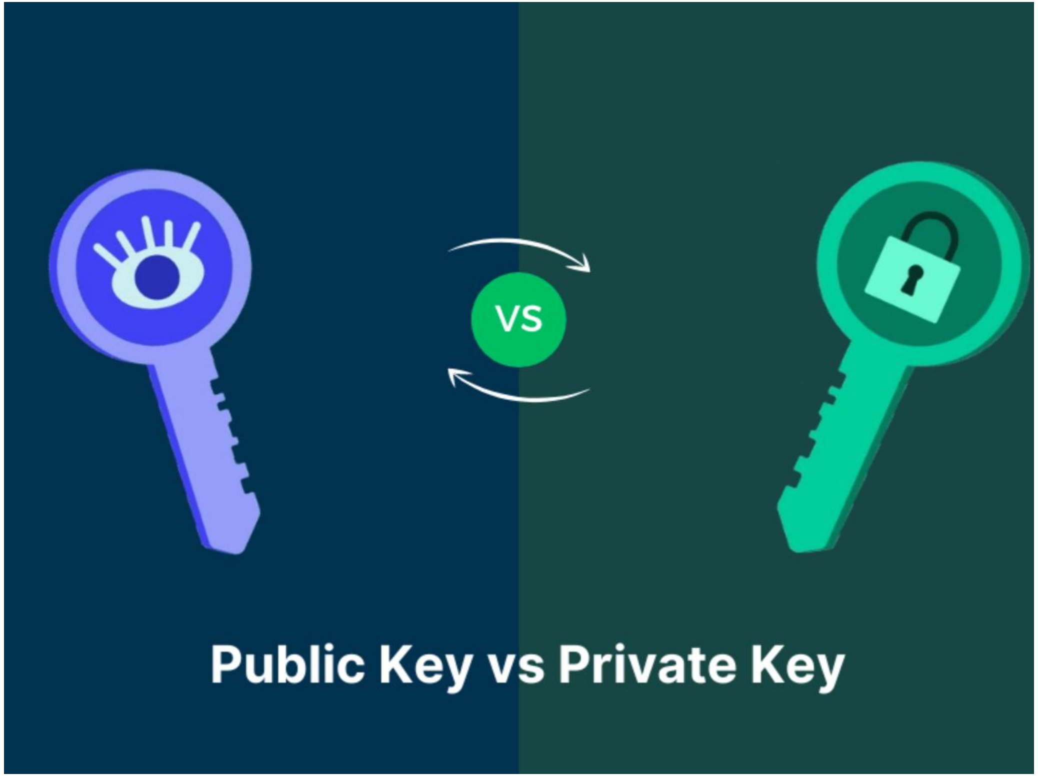
CSC116 Certificate

Scenario 1:

1. You send your public key to me.
2. But I don't trust it.
3. I send you a random message encrypted with your public key.
4. You try to decrypt it with your private key to prove to me the public key is yours.

You **own the private key** corresponding to this public key.
It is high possibility that this public key is yours.
✅

But also maybe the attacker steals the key pairs



"If someone gives me a public key and says it belongs to a bank, how can I be sure it really belongs to the bank and not an attacker?"

Problem: **A public key needs identity proof.**

Solution: **Certificates.**

Limitation of Current System: you need to authenticate this public key is your public key

General | Details

Issued To

Common Name (CN)	welcome9.miami.edu
Organization (O)	University of Miami
Organizational Unit (OU)	<Not Part Of Certificate>

server (welcome9.miami.edu) holds the **private key**, and only the server can use it.

Issued By

Common Name (CN)	InCommon RSA Server CA 2
Organization (O)	Internet2
Organizational Unit (OU)	<Not Part Of Certificate>

Validity Period

Issued On	Thursday, August 21, 2025 at 8:00:00 PM
Expires On	Tuesday, September 22, 2026 at 7:59:59 PM

SHA-256 Fingerprints

Certificate	a5ff16599e0e9a71d4e911a0111c50d8504830e667e0b904fe9df4ed73e3695f
Public Key	0d58e46752606d8eb143bf6d9d2b141245b2f5aa40f48fedbe1274b7c13465cb

The **public key** is distributed to anyone who connects, so they can:

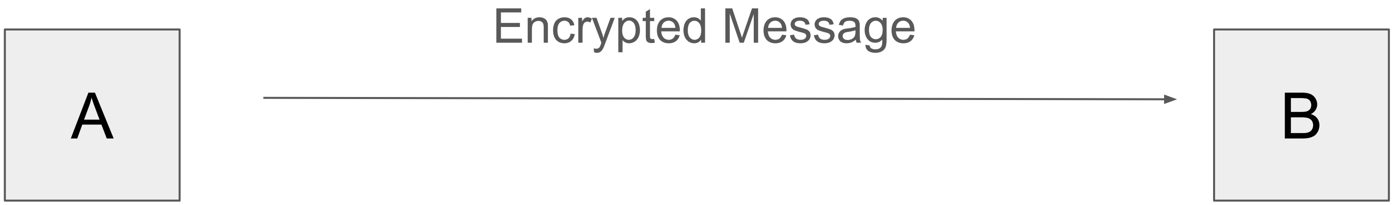
- Verify the server's **digital signatures**.

CSC116 Message Authention Codes (MACs)

Question: “If you send a message to your friend over the internet, how can you be sure it hasn’t been tampered with? And how do you know it really came from your friend?”

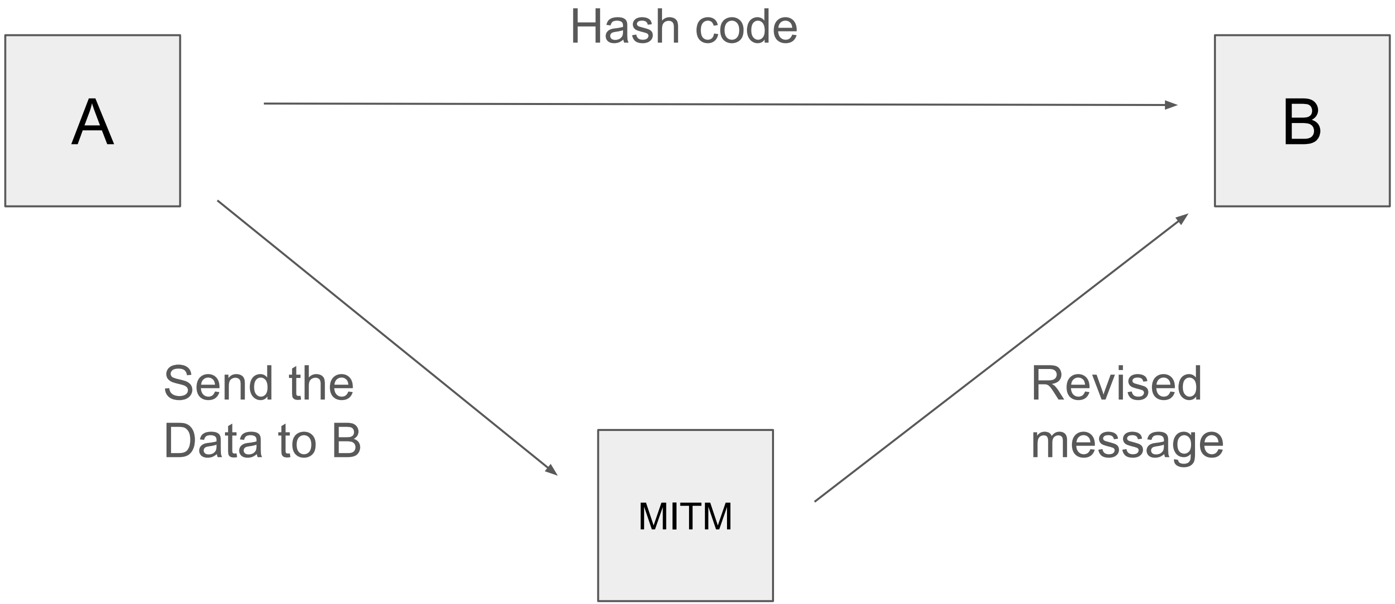
A send a encryption to —> B
It doesn’t mean that the message is originally from A. It may from attacker.

Question 2 ?



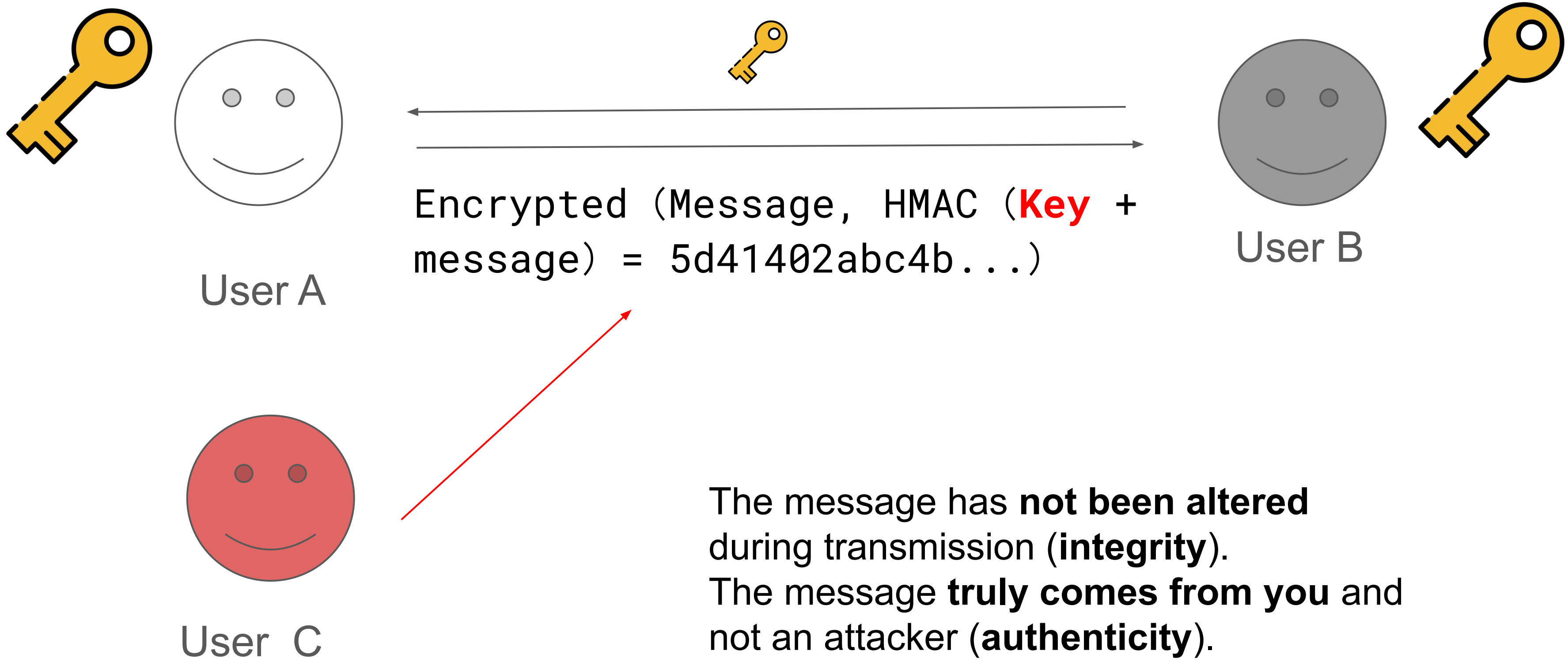
Confidentiality is ensured

Question 3 ?



Message Authentication Code (MAC): A cryptographic code that ensures a message hasn't been altered and verifies the identity of the sender.

- Integrity: Ensures the message wasn't tampered with during transmission.
- Authenticity: Confirms the message came from a trusted sender.



The message has **not been altered** during transmission (**integrity**).
The message **truly comes from you** and not an attacker (**authenticity**).

Step 1: Generating the MAC

1. The sender has a message `M = "Hello"` and a shared secret key `K`.
2. Using a cryptographic algorithm (e.g., HMAC), the sender generates the MAC:
   MAC = HMAC(K, M)
3. The sender transmits the message along with the MAC:
   `("Hello", MAC)`

Step 2: Verifying the MAC

1. The receiver receives the message M and the MAC.
2. The receiver uses the same secret key K and algorithm to generate their own MAC:
   MAC' = HMAC(K, M)
3. They compare the two MACs:
   - If **MAC == MAC'**, the message is authentic and hasn't been tampered with.
   - If they don't match, the message may have been altered or isn't from a trusted source.

Hash function
Vs.
HMAC function

Hash Function (e.g., SHA-256):

1. Input message: "Hello"
2. Apply hash function: SHA256("Hello")
3. Output: A fixed-size hash, like 2cf24dba5fb0...

- Key Point: Anyone with "Hello" can compute the same hash.

HMAC Function (e.g., HMAC-SHA256):

1. Input message: "Hello"
2. Secret key: "symmtric key"
3. Apply HMAC: HMAC_SHA256("symmtric key", "Hello")
4. Output: A unique HMAC, like 5d41402abc4b...

- Key Point: Only those with "symmtric" can generate or verify this HMAC.