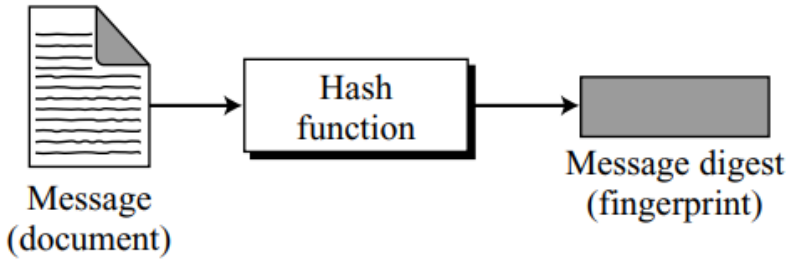
MESSAGE INTEGRITY:

- ✓ AES, RSA or others cryptographic algorithms can provide secrecy, or confidentiality,
- ✓ but not integrity.
- ✓ However, there are occasions where we may not even need secrecy but instead must have integrity.
- ✓ For example: Alice writes a will to distribute her estate after her death, which does not need encryption since it is publicly accessible, but its integrity must be preserved to prevent any changes to its contents.
- ✓ One way to preserve the integrity of a document is through the use of a fingerprint.

- ❑ (Document, fingerprint) equivalent to (message, digest pair).
- ✓ To preserve message integrity, it is processed through a cryptographic hash function, which generates a compressed representation, similar to a fingerprint.



- ✓ The message digest needs to be safe from change.

Checking Integrity: To verify message integrity, we re-run the cryptographic hash function and compare the new digest with the original; if they match, the message remains unchanged.

Preimage Resistance: A cryptographic hash function must be preimage resistant. Given a hash function h and y = h(M), it must be extremely difficult for Eve to find any message, M', such that y = h(M').

Preimage Attack

Given: y = h(M) **Find: M' such that y = h(M')**

Suppose you have a hash function H that produces a hash of "Hello" resulting in a hash value of 5d414. Preimage resistance means that if someone sees this hash, they cannot easily find an input string that hashes to this value (i.e., they can't easily determine that the original input was "Hello")

Second Preimage Resistance:

Second Preimage Attack

Given: M and h(M) **Find: $M' \neq M$ such that h(M) = h(M')**

If you have an input x1 (e.g., "Hello") with a hash value of 5d414, second preimage resistance means that it should be hard to find a different input 🆔 (like "World") such that ("Hello")=H("World"). If you can find such a different input, the hash function fails this property.

If you have an input x1 (e.g., "Hello") with a hash value of H(x1)=5d414, second preimage resistance means that it should be difficult to find a different input x2 (e.g., "World") such that H(x1)=H(x2). In this case, H("Hello")=H("World") would violate the second preimage resistance property. If someone can find such a different input x2 that produces the same hash as x1 , the hash function fails this property. Therefore, a secure hash function should make it computationally infeasible to find any second input that generates the same hash as the first.

Attack: Eve intercepts (has access to) a message M and its digest h(M). She creates another message M'≠ M, but h(M) = h(M'). Eve sends the M' and h(M') to Bob. Eve has forged the message.

Key Difference

- **Preimage Resistance** deals with finding an input from its hash.
- **Second Preimage Resistance** deals with finding a different input that produces the same hash as a given input.

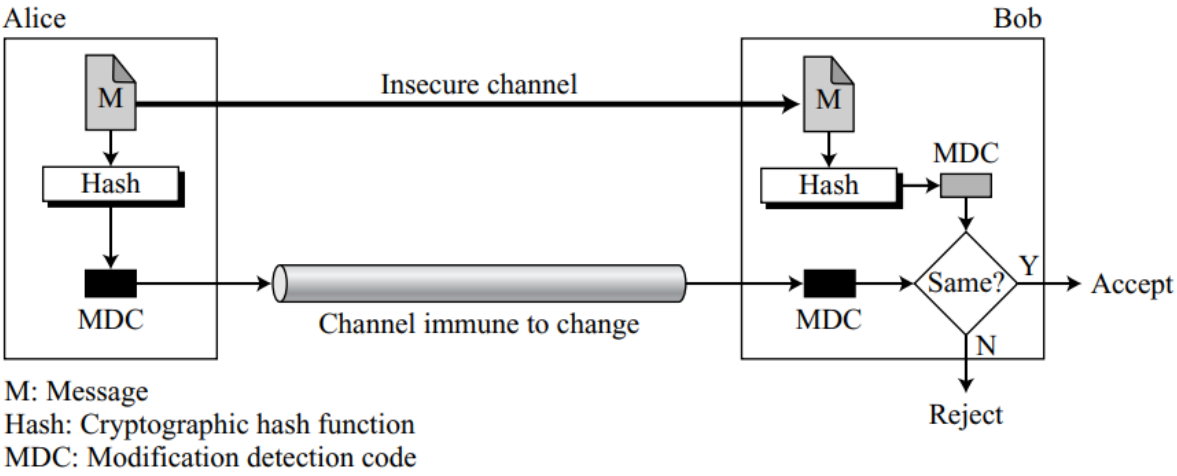
Collision Resistance

Definition: Collision resistance means that it is hard to find any two different inputs x1 and x2 such that H(x1)=H(x2). This property ensures that no two distinct inputs should produce the same hash value.

MESSAGE AUTHENTICATION:

- ✓ A message digest ensures the integrity of a message by confirming it hasn't been altered, but it does not authenticate the sender.
- ✓ When Alice sends a message to Bob, he needs proof that it is indeed Alice who sent it, as a message digest alone cannot provide this proof.
- ✓ The digest generated by a cryptographic hash function is known as a modification detection code (MDC), which can detect changes in the message.
- ✓ For message authentication, a message authentication code (MAC) is required to verify the sender's identity.

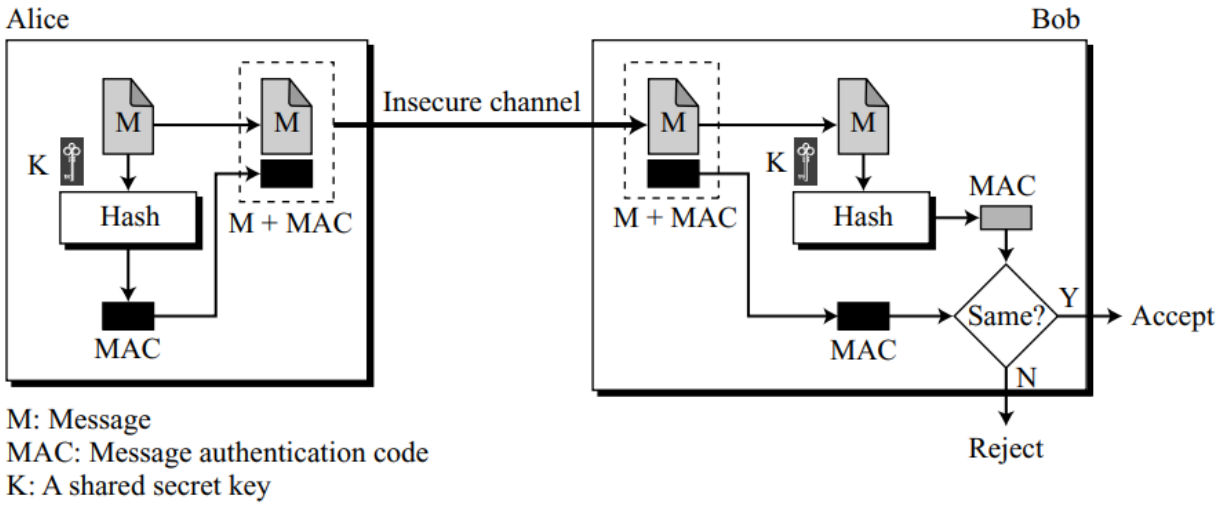
MDC:



Attack: If both the message and the MDC are sent through the insecure channel, Eve can intercept the message, change it, create a new MDC from the message, and send both to Bob. Bob never knows that the message has come from Eve. Note that the term safe can mean a trusted party; the term channel can mean the passage of time.

MAC:

Message Authentication Code (MAC)



A **Message Authentication Code (MAC)** is a cryptographic checksum that ensures both the integrity and authenticity of a message. It is generated using a secret key and a hash function (or a block cipher), providing a way to verify that the message was sent by a legitimate sender and has not been altered in transit.

- **Integrity**: A MAC ensures that the message has not been modified. If even a single bit of the message changes, the MAC will also change, indicating potential tampering.
- **Authentication**: A MAC verifies the sender's identity. Only someone with the secret key can generate a valid MAC for a given message, allowing the recipient to confirm that the message originated from the expected sender.

If Alice wants to send a message to Bob securely, she creates a MAC using her message and a secret key. She sends both the message and the MAC to Bob. Upon receiving them, Bob uses the same secret key to generate a MAC from the received message. If his MAC matches the one Alice sent, he can be confident that the message is authentic and unchanged.