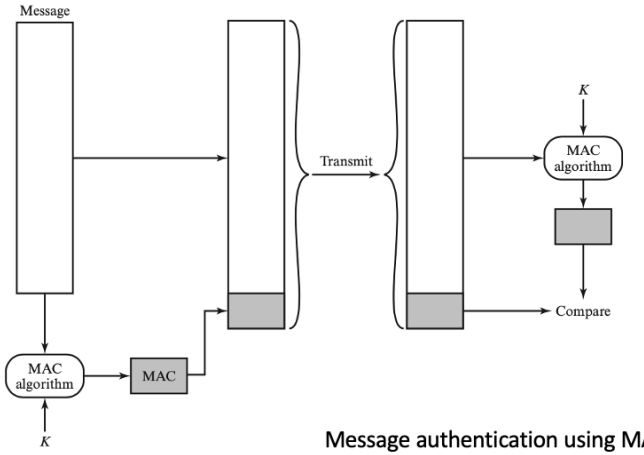
Message Authentication

Message authentication

- message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- then three alternative functions used:
  - message encryption – symmetric
  - message authentication code (MAC)
  - digital signature



Message authentication using MAC

Message encryption

- Symmetric message encryption by itself also provides a measure of authentication
- if symmetric encryption is used then:
  - receiver knows sender must have created it
  - since only sender and receiver know key used
  - know content cannot be altered

Message encryption

- if public-key encryption is used:
  - encryption provides no confidence of sender
  - since anyone potentially knows public-key
  - so, need to recognize corrupted messages
  - however, if
    - sender **signs** message using their private-key
    - then encrypts with recipients' public key
    - have both secrecy and authentication
  - but at cost of two public-key uses on message

Reasons to avoid encryption authentication

- Encryption software is quite slow
- Encryption hardware costs are nonnegligible
- Encryption hardware is optimized toward large data sizes
- An encryption algorithm may be protected by a patent

(1) MAC (Hash)

(2) Digital Signature

RC4

Hash Function

Hash functions

- Hash function: h = H(M)
  - M can be of any size
  - h is always of fixed size
  - Typically, h << size(M)

fast. h small

smaller bandwidth

MAC.

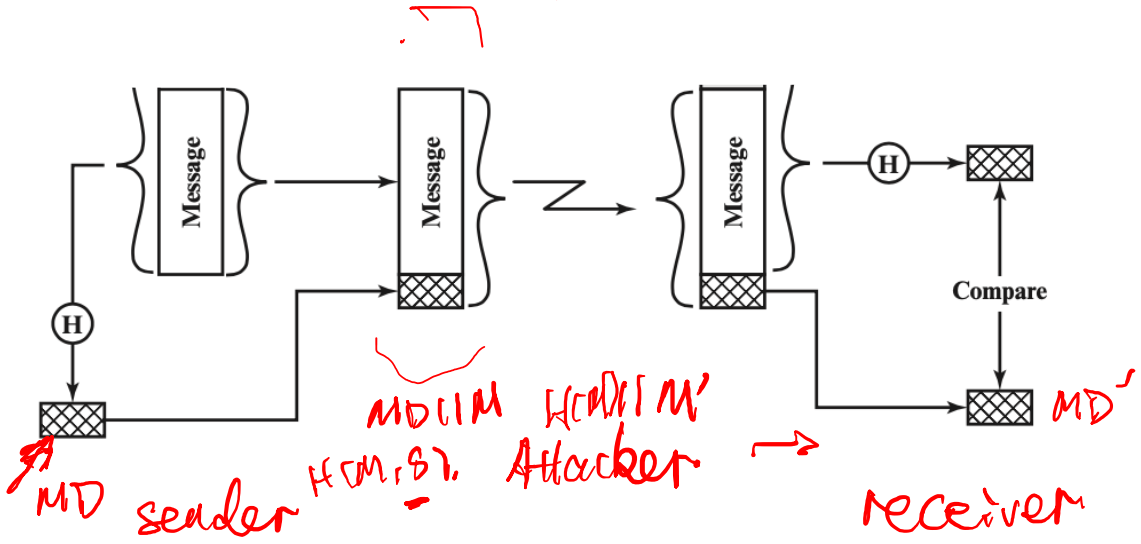
many-to-one.

collisions!

compression function — lossy / non-lossy

LLM.

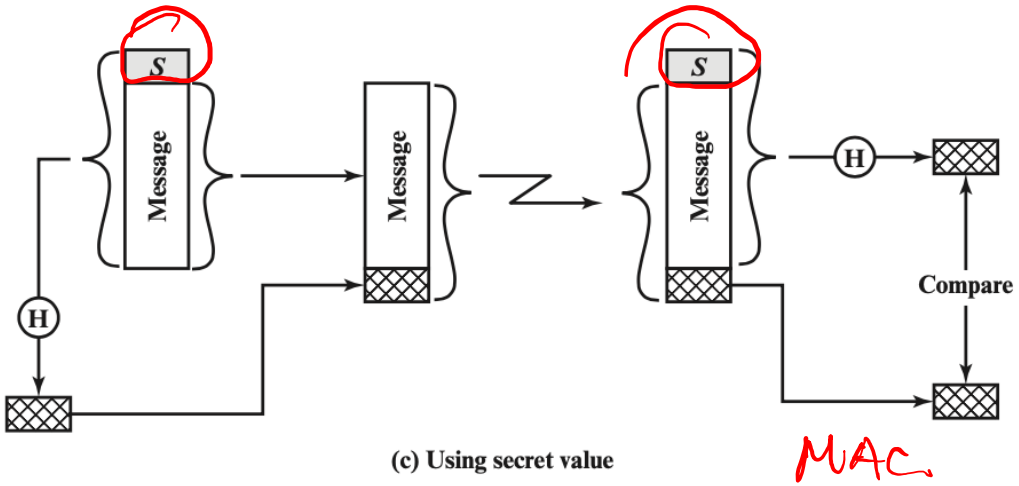
One use case - using hash function



- Message, M
- A calculates $MD_M = H(M)$
- B recalculates MD'_M, and check
- $MD'_M = MD_M$

Version	Operating System	Description	MD5 Sum	File Size	Sigstore	SBOM	GPG
Gzipped source tarball	Source release		138c2e19c835ead10499571e0d4cf189	28.0 MB	.sigstore	SPDX	SIG
XZ compressed source tarball	Source release		256cdb3bbf45cdce7499e52ba6c36ea3	21.7 MB	.sigstore	SPDX	SIG

One use case - using hash function



(c) Using secret value

MAC

- Initialization: A and B share a common secret, S_{AB}
- Message, M
- A calculates $MD_M = H(S_{AB} || M)$
- B recalculates MD'_M, and check
- $MD'_M = MD_M$

Requirements for secure hash functions

- 1. can be applied to any sized message `M`
- 2. produces fixed-length output `h`
- 3. is easy to compute `h=H(M)` for any message `M`
- 4. given `h` is infeasible to find `x` s.t. `H(x)=h`
  - one-way property or preimage resistance
- 5. given `x` is infeasible to find `x'` s.t. `H(x')=H(x)`
  - weak collision resistance or second pre-image resistant
- 6. infeasible to find any pair of `x, x'` s.t. `H(x')=H(x)`
  - strong collision resistance

Hash Function: Collision Resistance

- **Collision**: Two different inputs with the same output
  - $x \neq x'$ and $H(x) = H(x')$
  - Can we design a hash function with no collisions?
    - No, because there are more inputs than outputs (pigeonhole principle)
  - However, we want to make finding collisions *infeasible* for an attacker
- **Collision resistance**: It is infeasible to (i.e. no polynomial time attacker can) find any pair of inputs $x' \neq x$ such that $H(x) = H(x')$

Secure hash function

- A hash function that satisfies the first five properties is referred to as a weak hash function
- **Security:** random/unpredictability, no predictable patterns for how changing the input affects the output → diffusion,
  - Changing 1 bit in the input causes the output to be completely different
  - Also called "random oracle" assumption

Hash { 1. Output uniform
2. One-way
3. Computationally difficult to find x'

Secure hash function

- A hash function that satisfies the first five properties is referred to as a weak hash function
- **Security:** random/unpredictability, no predictable patterns for how changing the input affects the output
  - Changing 1 bit in the input causes the output to be completely different
  - Also called “random oracle” assumption
- A message digest
  - output of a cryptographic hash function containing a string of digits created by a one-way hashing formula
  - provides data integrity
- Examples: SHA-1 (Secure Hash Algorithm 1), SHA-2, SHA-3, MD5

Hash Function: Examples

- MD5
  - Output: 128 bits
  - Security: Completely broken
- SHA-1
  - Output: 160 bits
  - Security: Completely broken in 2017
  - Was known to be weak before 2017, but still used sometimes
- SHA-2
  - Output: 256, 384, or 512 bits (sometimes labeled SHA-256, SHA-384, SHA-512)
  - Not currently broken, but some variants are vulnerable to a length extension attack
  - Current standard
- SHA-3 (Keccak)
  - Output: 256, 384, or 512 bits
  - Current standard (not meant to replace SHA-2, just a different construction)