Length Extension Attacks

$H(x||m)||x||m$

$k||x$

$x||k$

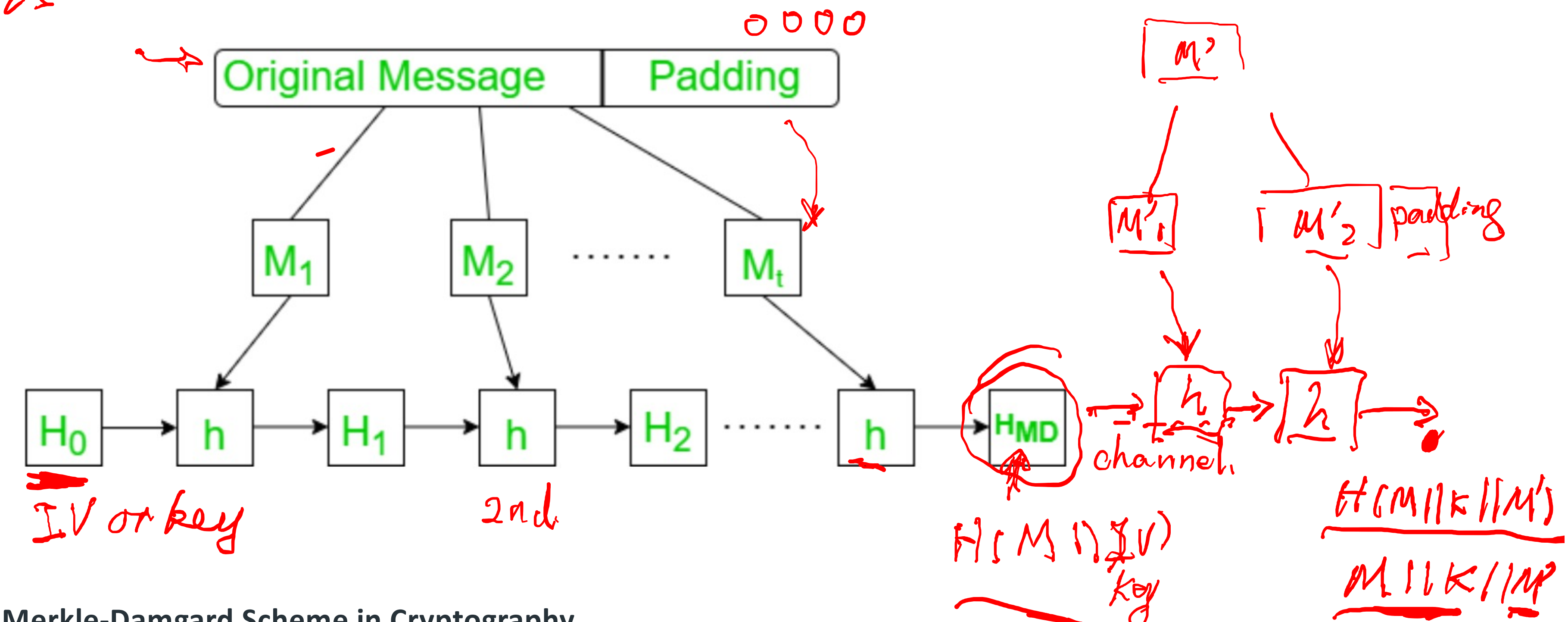
- **Length extension attack**: Given $H(x)$ and the length of x, but not x, an attacker can create $H(x \mid\mid m)$ for any m of the attacker's choosing
  - Length extension attack - Wikipedia
- SHA-256 (256-bit version of SHA-2) is vulnerable
- SHA-3 is not vulnerable

* implementation of hash function matters

Merkle-Damgard Scheme

SHA-256

Attack,



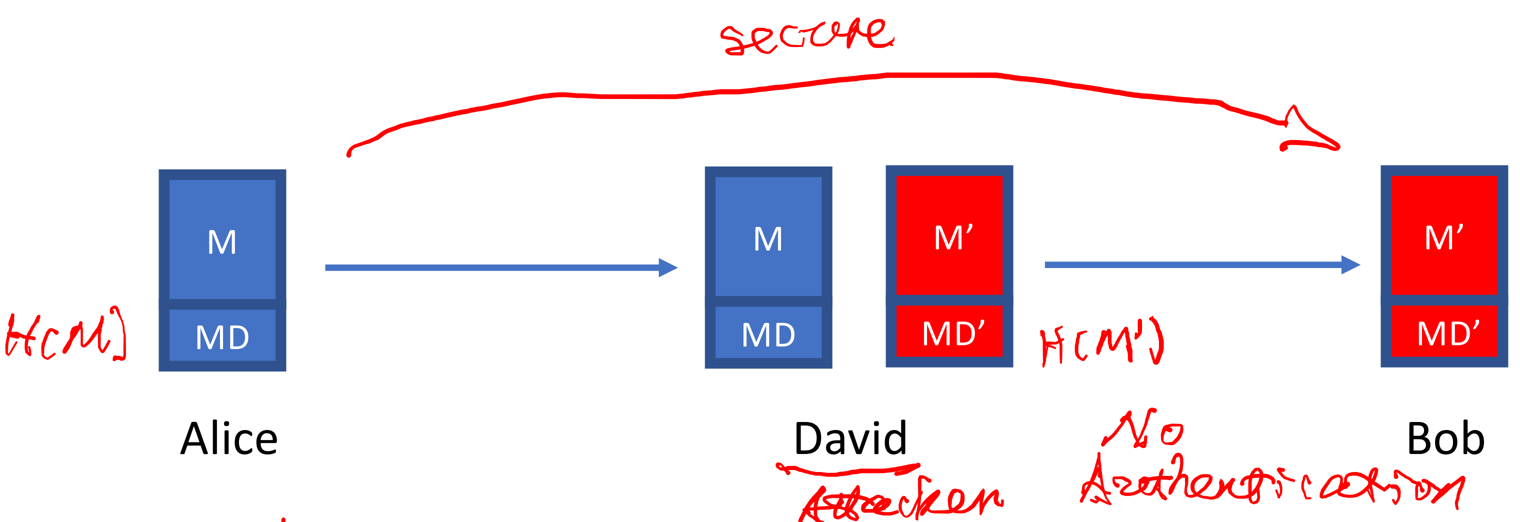
Merkle-Damgard Scheme in Cryptography

https://www.geeksforgeeks.org/computer-networks/merkle-damgard-scheme-in-cryptography/

Do hashes provide integrity?

- It depends on your threat model
- Scenario
  - Alice and Bob want to communicate over an insecure channel
  - David might tamper with messages
- Idea: Use cryptographic hashes
  - Alice sends her message with a cryptographic hash over the channel
  - Bob receives the message and computes a hash on the message
  - Bob checks that the hash he computed matches the hash sent by Alice
- Threat model: David can modify the message *and the hash*
  - No integrity!

Man-in-the-middle attack



Solutions: 1. MAC

2. Digital Signature

Do hashes provide integrity?

- It depends on your threat model
- If the attacker can modify the hash, hashes don't provide integrity
- Main issue: Hashes are *unkeyed* functions
  - There is no secret key being used as input, so any attacker can compute a hash on any value

Solutions

- A message digest created using a secret symmetric key is known as a Message Authentication Code (MAC), because it can provide assurance that the message has not been modified
- The sender can also generate a message digest and then sign the digest using the private key of an asymmetric key pair, forming a digital signature. The signature must then be verified by the receiver through comparing it with a locally generated digest $H(M)$

Hashes: Summary

- Map arbitrary large input to fixed-length output
- Output is deterministic
- Security properties
  - One way: Given an output y, it is infeasible to find any input x such that $H(x) = y$.
  - Second preimage resistant: Given an input x, it is infeasible to find another input $x' \neq x$ such that $H(x) = H(x')$. weak ~~to~~ collision 5th   find x'
  - Collision resistant: It is infeasible to find any pair of inputs $x' \neq x$ such that $H(x) = H(x')$. strong collision 6th
  - Randomized output
- Some hashes are vulnerable to length extension attacks implementation
- Hashes don't provide integrity (unless you can publish the hash securely)

Message Authentication Code

Message authentication code (MAC)

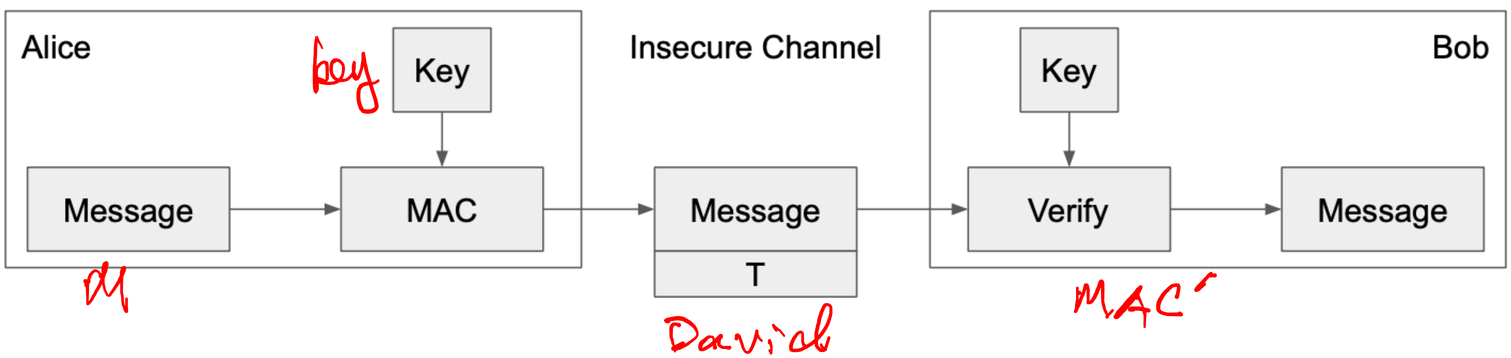
- generated by an algorithm that creates a small fixed-sized block
  - depending on both message and some key ← preshared secret.
    symmetric.
  - not be reversible → h( )
  - $MAC_M = F(K_{AB}, M)$
- appended to message as a signature
- receiver performs same computation on message and checks it matches the MAC
  key → sender & receiver
- provides assurance that message is unaltered and comes from sender

Authentication — integrity — validate ID of origin / non-repudiation of origin

MACs: Usage

- Alice wants to send *M* to Bob, but doesn't want David to tamper with it
- Alice sends *M* and *T* = MAC(*K*, *M*) to Bob
- Bob receives *M* and *T*
- Bob computes MAC(*K*, *M*) and checks that it matches *T*
- If the MACs match, Bob is confident the message has not been tampered with (integrity)

pre-shared



MACs: Definition

timestamp || session ID ← TLS

- Two parts: PRNG( )
  - KeyGen() → *K*: Generate a key *K* session
  - MAC(*K*, *M*) → *T*: Generate a tag *T* for the message *M* using key *K*
    - hash( ) Inputs: A secret key and an arbitrary-length message
    - HMAC( ) Output: A fixed-length **tag** on the message
- Properties
  - **Correctness**: Determinism
    - Note: Some more complicated MAC schemes have an additional Verify(*K*, *M*, *T*) function that don't require determinism, but this is out of scope
  - **Efficiency**: Computing a MAC should be efficient → fast → cost effective
  - **Security**: existentially unforgeable under chosen plaintext attack

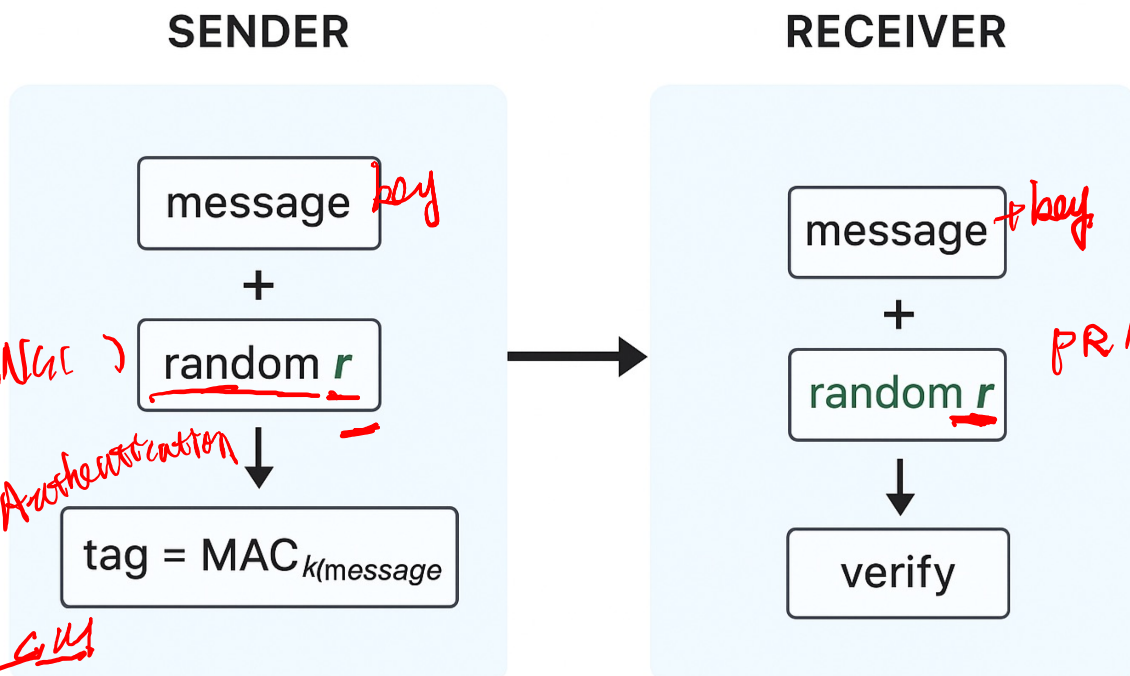
M encryption

C

Attacker M, C

good to derive

Randomized MAC (Non-Deterministic)



key

+key

PRNG( )

PRNG( )
RC4

3rd Encryption & Authentication
AES-GCM