Basic Crypto Tools

Dr. Chen Zhang

Department of Computer Science
The Hang Seng University of Hong Kong

Slides partially adapted from lecture notes by M. Goodrich&R. Tamassia, W. Stallings&L. Brown, and Dan Boneh.

Cryptographic Hash Functions

Hash Functions

- A hash function h maps a plaintext (pre-image) P to a fixed-length value x = h(P) called hash value or digest of P
  - A collision is a pair of plaintexts P and Q that map to the same hash value, h(P) = h(Q)
  - Collisions are **unavoidable**
  - For efficiency, the computation of the hash function should take time proportional to the length of the input plaintext

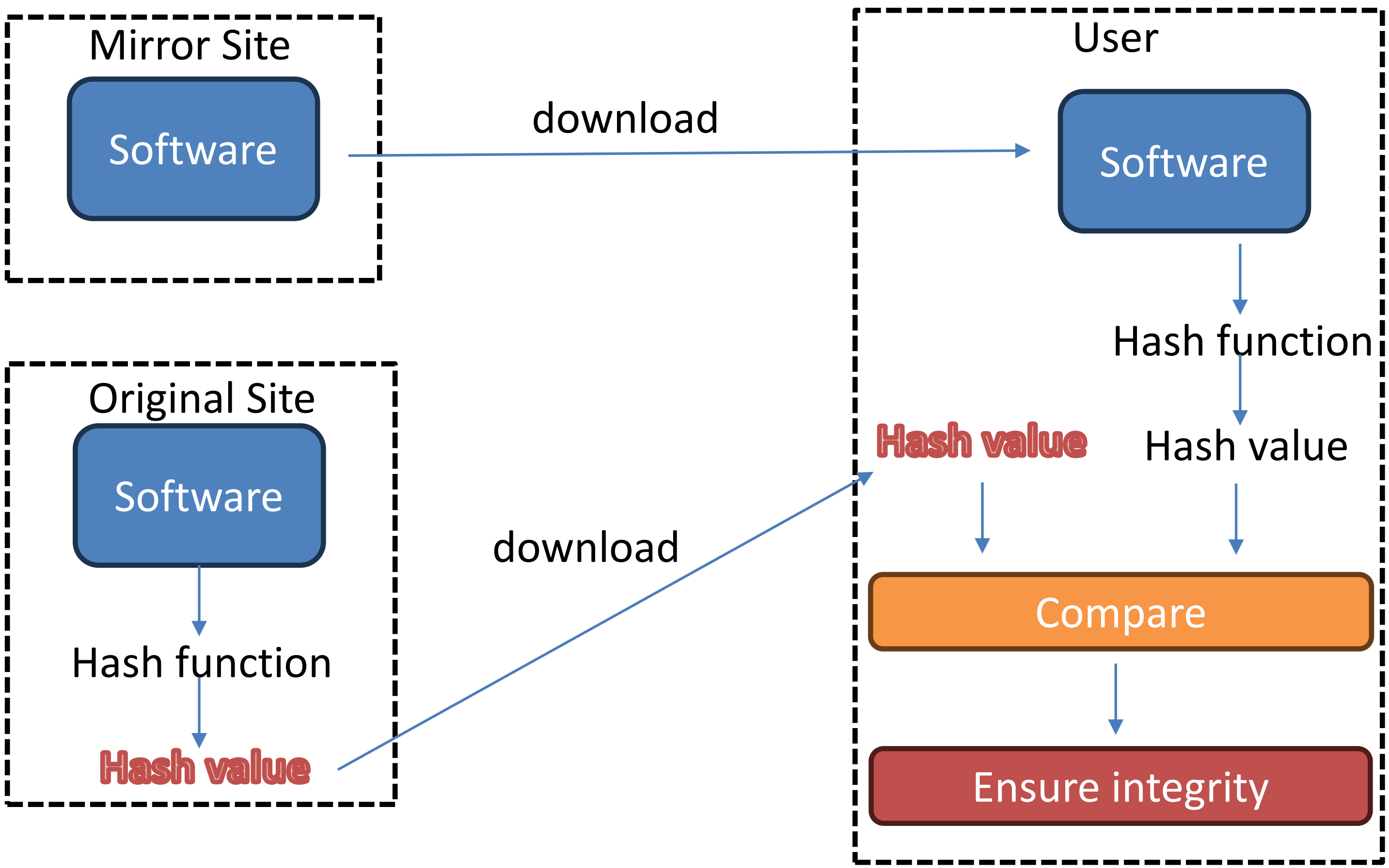
Cryptographic Hash Functions

- A cryptographic hash function satisfies additional properties

  - Preimage resistance (aka one-way)
    - Given a hash value x, it is hard to find a plaintext P such that h(P) = x
  - Second preimage resistance (aka **weak** collision resistance)
    - Given a plaintext P, it is hard to find a plaintext Q such that h(Q) = h(P)
  - Collision resistance (aka **strong** collision resistance)
    - It is hard to find a pair of plaintexts P and Q such that h(Q) = h(P)

An Example

- Download the software in Mirror site.



Message-Digest Algorithm 5 (MD5)

- Developed by Ron Rivest in 1991
- Uses 128-bit hash values
- Still widely used in legacy applications although considered insecure
- Various severe vulnerabilities discovered

Secure Hash Algorithm (SHA)

- Developed by NSA and approved as a federal standard by NIST
- SHA-0 and SHA-1 (1993)
  - 160-bits
  - Considered insecure
  - Still found in legacy applications
  - Vulnerabilities less severe than those of MD5
- SHA-2 family (2002)
  - 256 bits (SHA-256) or 512 bits (SHA-512)
  - Still considered secure despite published attack techniques
- SHA-3 released in 2015
  - Novel construction: **Keccak (sponge structure with absorbing phase and squeezing phase.)**
  - Internally different from the MD5-like structure of SHA-1 and SHA-2

Data Integrity: Applications of Cryptographic Hash Functions

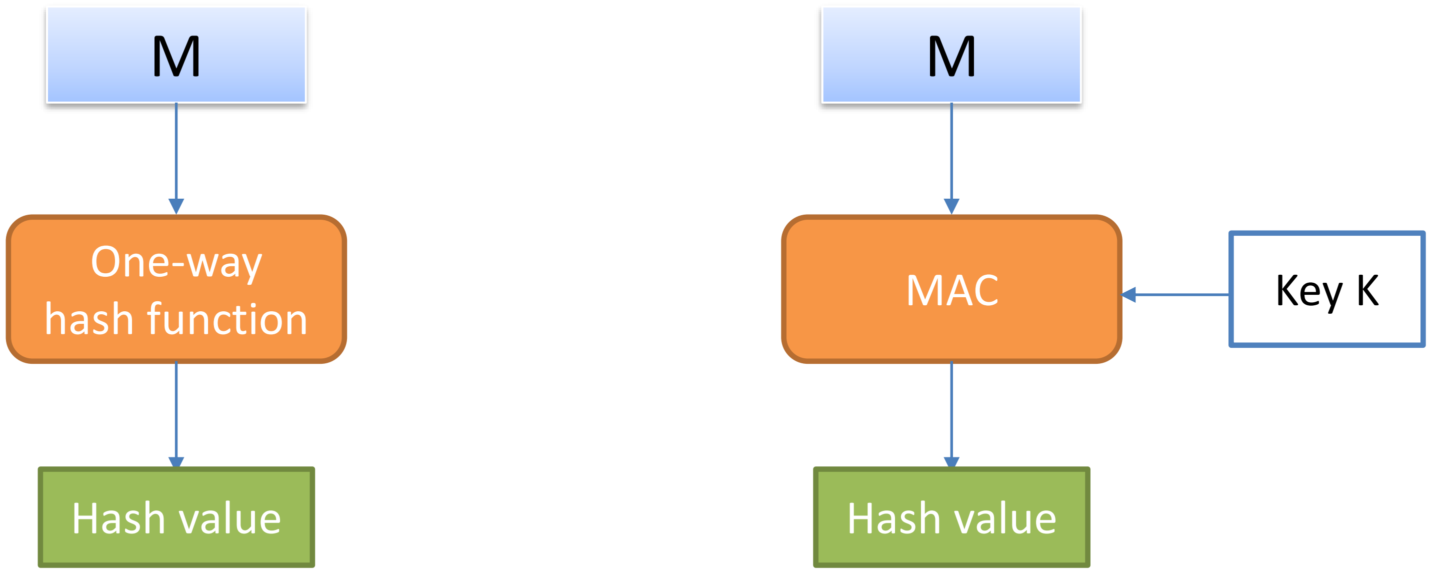
A Scenario

- There are two banks Alice and Bob.
- Bob receives a remittance request with content:
  - Transfer 200 from Bob to Alice
- For Bob, he needs to check:
  - Has the remittance request been tampered with? - **Integrity**
  - Is the remittance request really sent from Alice? (not sent by someone else pretending to be Alice) – **Authentication**

Problem: How to ensure **Integrity** and **Authentication** at the same time?

Message Authentication Code (MAC)

- MAC is a technique for **checking data integrity** and **performing authentication**.
- MAC: cryptographic hash function H(K,M) with two inputs:
  - Secret key K
  - Message M

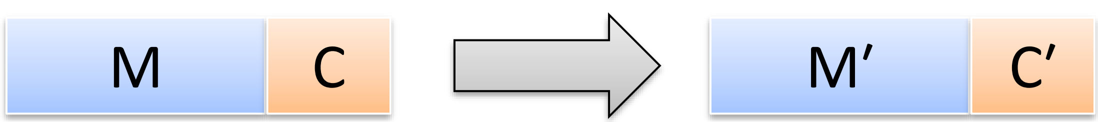


The comparison between on-way hash function and MAC

Message Authentication Code (MAC)

- Message integrity with MAC
  - Sequence of messages transmitted over insecure channel
  - Secret key K shared by sender and recipient
  - Sender computes MAC C = H(K,M) and transmits it along with message M
  - Receiver recomputes MAC from received message and compares it with received MAC
  - Attacker cannot compute correct MAC for a forged message
  - More efficient than signing each message

Compute
C = H(K,M)

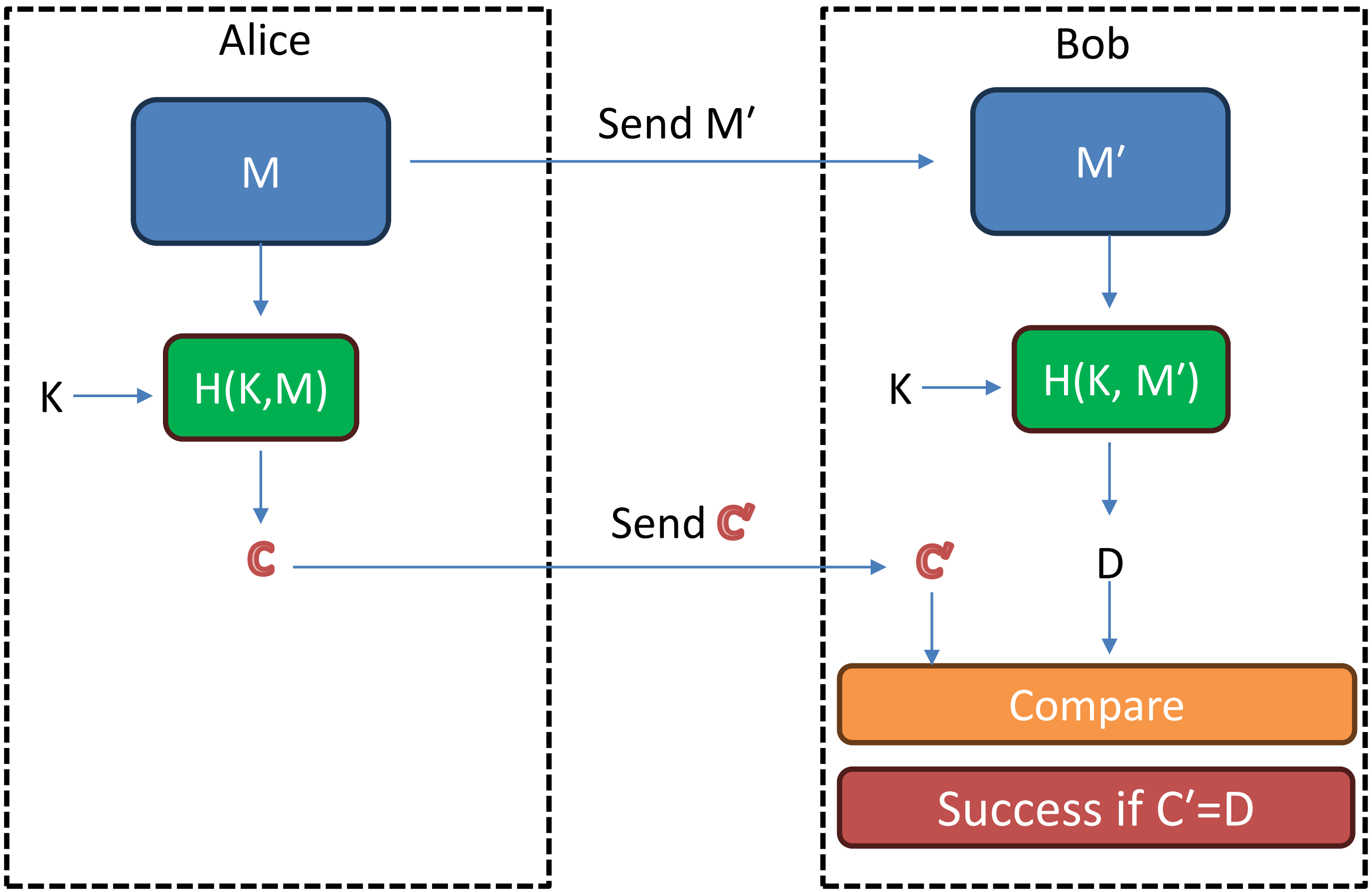


sent message

received message

Compute
D = H(K,M′)
Accept if D = C′

Message Authentication Code (MAC)



Construction: HMAC (Hash-MAC)

- Building a MAC from a cryptographic hash function is not immediate.
- Because of the iterative construction of standard hash functions, the following MAC constructions are proved to be **insecure**:
  - $H(K \| M)$
  - $H(M \| K)$
  - $H(K \| M \| K)$

Construction: HMAC (Hash-MAC)

Most widely used MAC on the Internet, e.g., IPSEC

HMAC security is the same as that of the underlying cryptographic hash function

H: hash function.

example: SHA-256 ; output is 256 bits

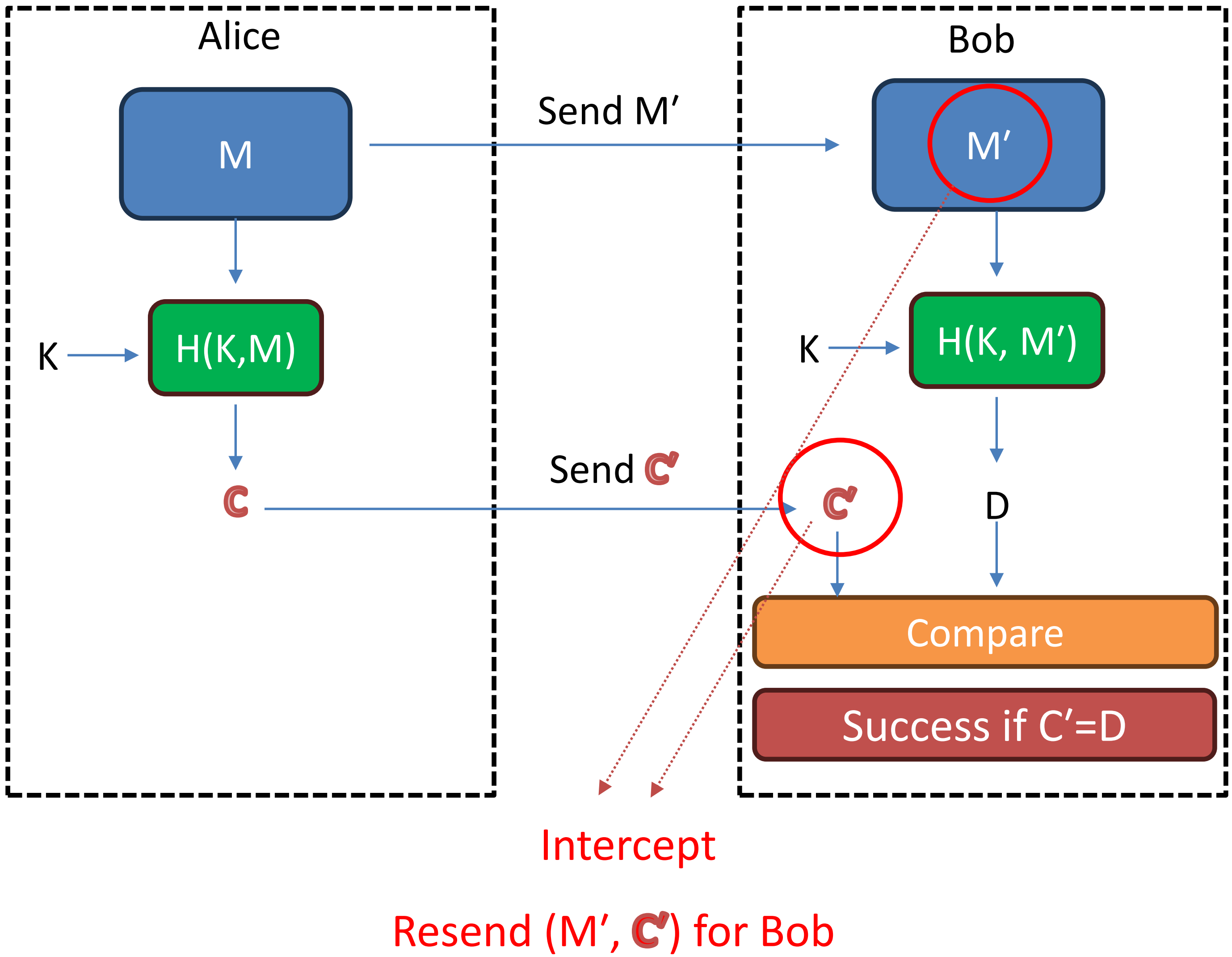
Building a MAC out of a hash function:

Standardized method: HMAC

$$H(k, m) = H(k \oplus opad \,\|\, \mathbf{H(k \oplus ipad \,\|\, m)})$$

opad, and ipad are specified padding constants

Further Protect Data Against Replay Attacks



Further Protect Data Against Replay Attacks

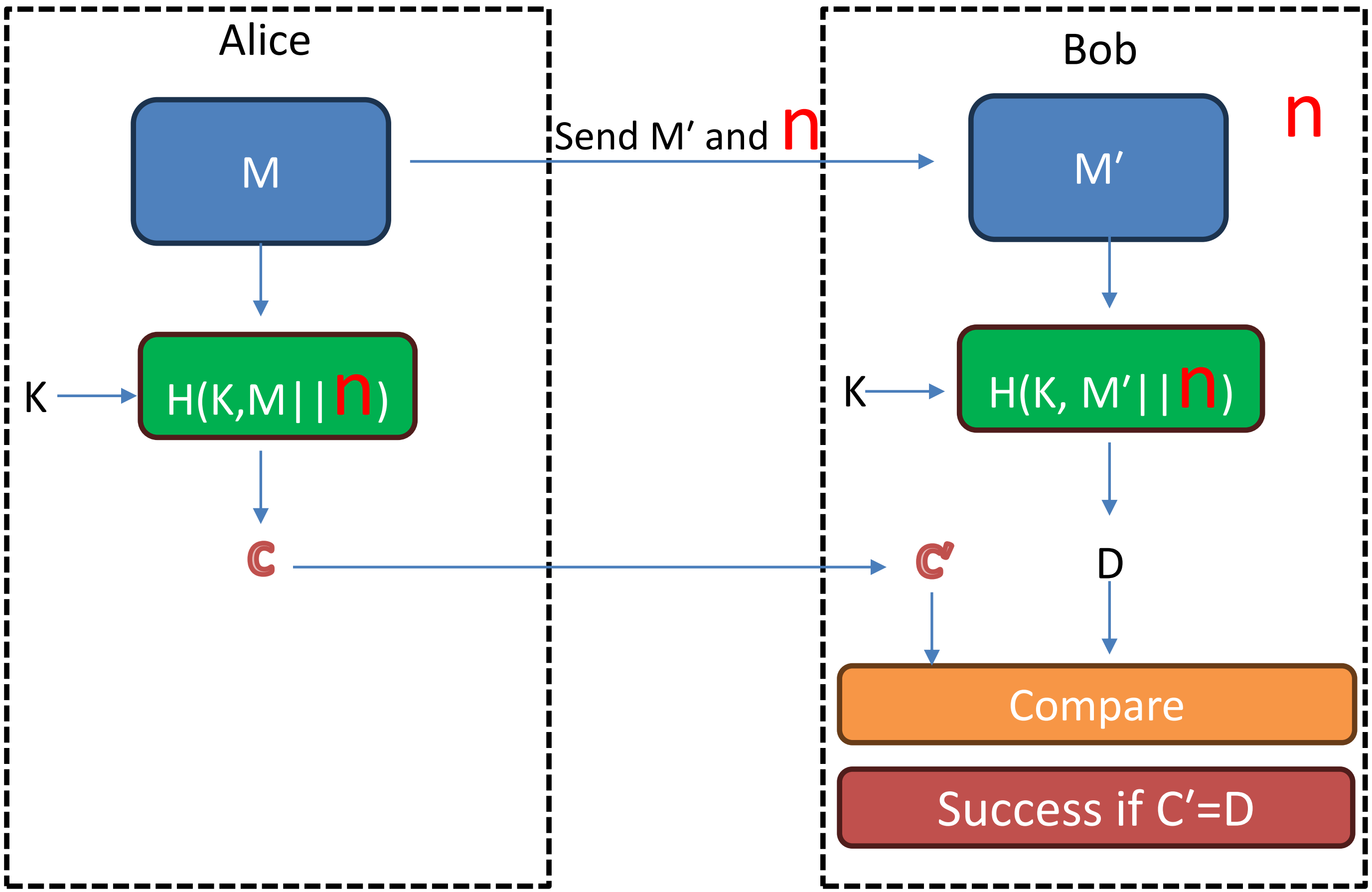
- The replay attack occurs when a cybercriminal eavesdrops on a secure network communication, intercepts it, and then fraudulently resends it to misdirect the receiver into doing what the hacker wants.
- The added danger of replay attacks is that a hacker doesn't even need advanced skills to decrypt a message after capturing it from the network. The attack could be successful simply by **resending** the whole thing.

Further Protect Data Against Replay Attacks

- Methods to against replay attacks:
  - Send message with a session key (nonce).

  When the message is sent, the session key is randomly generated. This way, only the sender and receiver have access to the communication. Any subsequent sessions require a different session key.
  - Send message with timestamps (include the time and date of data transmission).
  - One-Time Passwords (OTPs)

Further Protect Data Against Replay Attacks



Use different nonce **n** each time

Authenticated Encryption:
Encryption + MAC

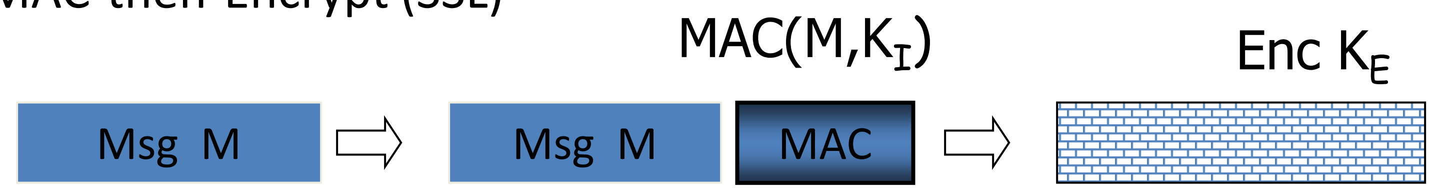
Securing a Communication Channel

- Authentication encryption: gain the benefits of encryption with MAC (ensure confidentiality, integrity, and authentication simultaneously)
- Three operations of combining MAC and encryption
  - MAC-then-Encrypt: first compute the MAC of plaintext, and then encrypt both MAC and plaintext.
  - Encrypt-then-MAC: first use key to encrypt plaintext, then compute the MAC of ciphertext.
  - Encrypt-and-MAC: use key to encrypt plaintext and compute the MAC of plaintext.

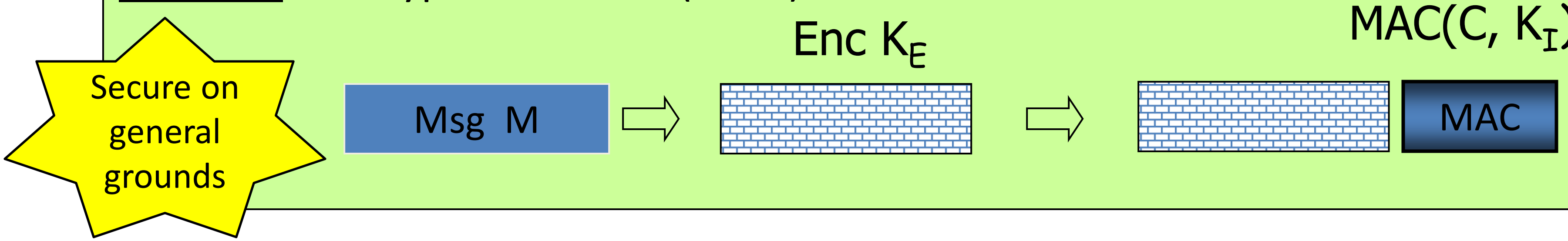
Options of Combining MAC and ENC (CCA)

Encryption key K_E MAC key = K_I

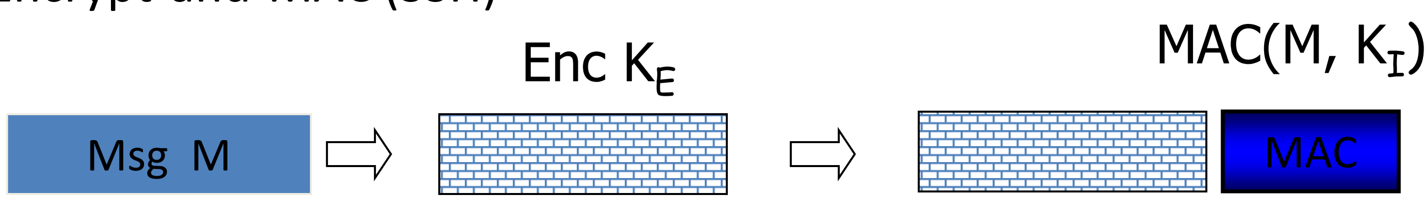
Option 1: MAC-then-Encrypt (SSL)



Option 2: Encrypt-then-MAC (IPsec)



Option 3: Encrypt-and-MAC (SSH)



Summary

- MAC
- Replay attack
- Authenticated encryption: encrypt-then-MAC

The Limitation of MAC

Suppose: Bob received the message m sent from Alice.

1) Bob cannot prove to others that message m was sent by Alice.
2) MAC cannot prevent nonrepudiation