Message Authentication Code(MAC)

2019. 4. 9

Contents

Message Integrity and Authentication

- Message Integrity
  - Verify that the message is not modified in any way, i.e., the received message is the same as the original message.
- Message Authentication
  - Verify that the message is sent by the claimed sender, i.e., verify the identity of the sender of the message.
  - Sometimes message integrity and authentication are used interchangeably, but I distinguish two terms.
  - And message authentication is different from entity(or user) authentication which has nothing to do with

How about encryption?

- Symmetric key encryption can guarantee message authentication, but not integrity.
- Digital signature can guarantee message integrity as well as message authentication.

Disclaimer

- Here, we try to find the crypt solutions which provide the message integrity and authentication, regardless of confidentiality.

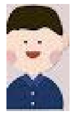
Message Authentication Code (MAC)

- Some hash values which are extracted from the image of a message are often called message digests.
- The message digest to be used for message integrity and authentication is called message authentication code (MAC).

MAC

- Alice computes MAC based on the message, and send MAC and the message to Bob.
- Bob checks the message integrity and authenticity by using MAC.

 Alice

Bob 

message M
$h = MAC(M)$

(M, h) →

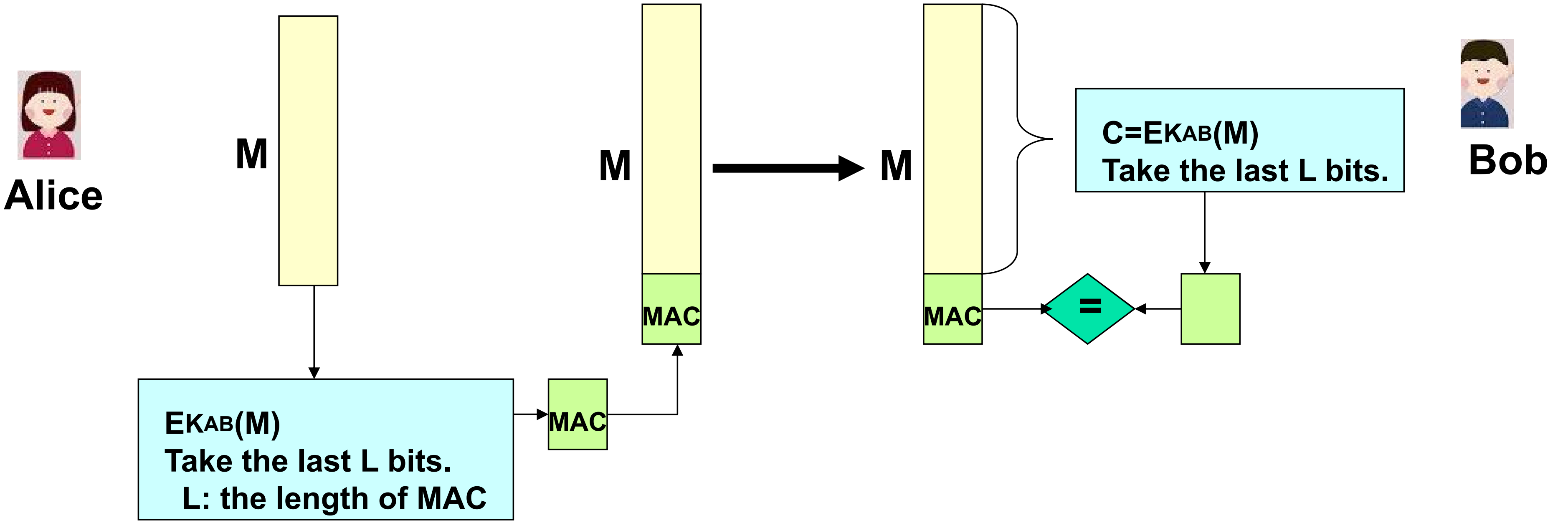
$h' = MAC(M)$
If $h = h'$, valid

MAC properties

- Arbitrary input length
- Fixed output length
- Message integrity
- Message authenticity
  - Depending on how MAC is generated
- Non-repudiation is not provided

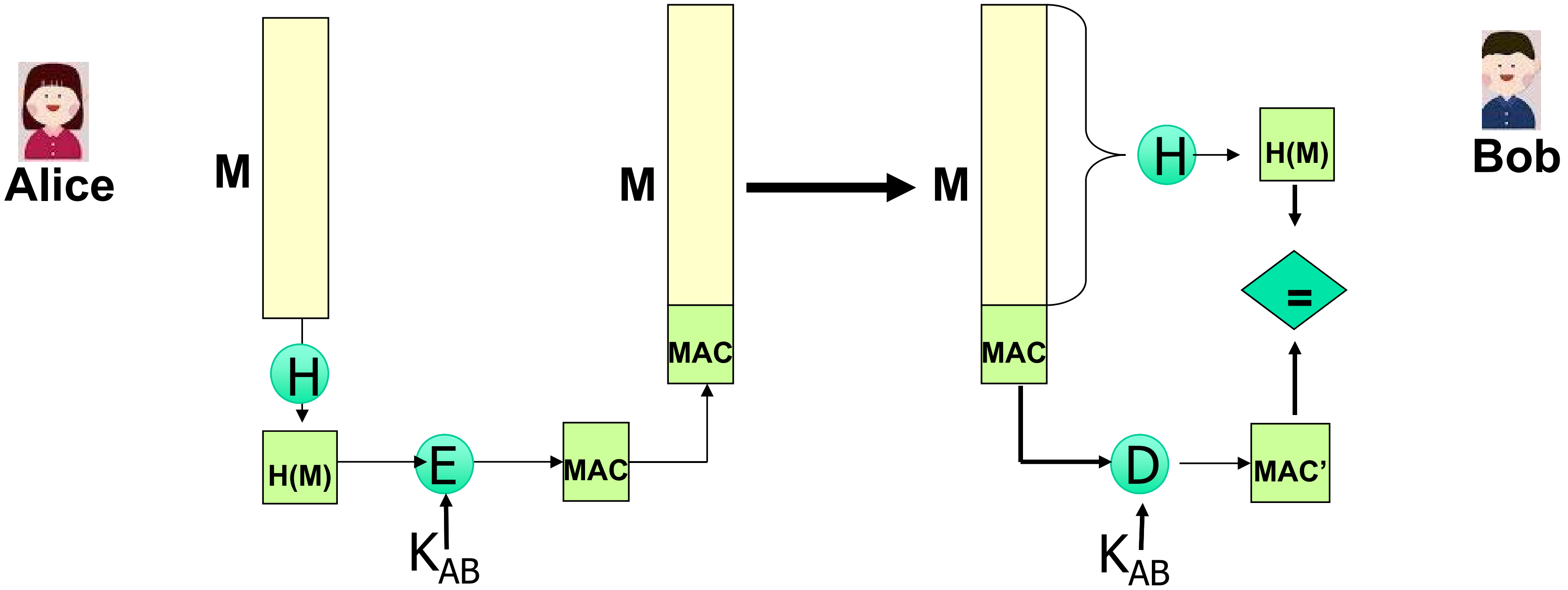
1^{st} Approach: Computing MAC by symmetric-key

- Using a symmetric-key, a sender generates a small block of data of an encrypted message as MAC and appends it to the message.



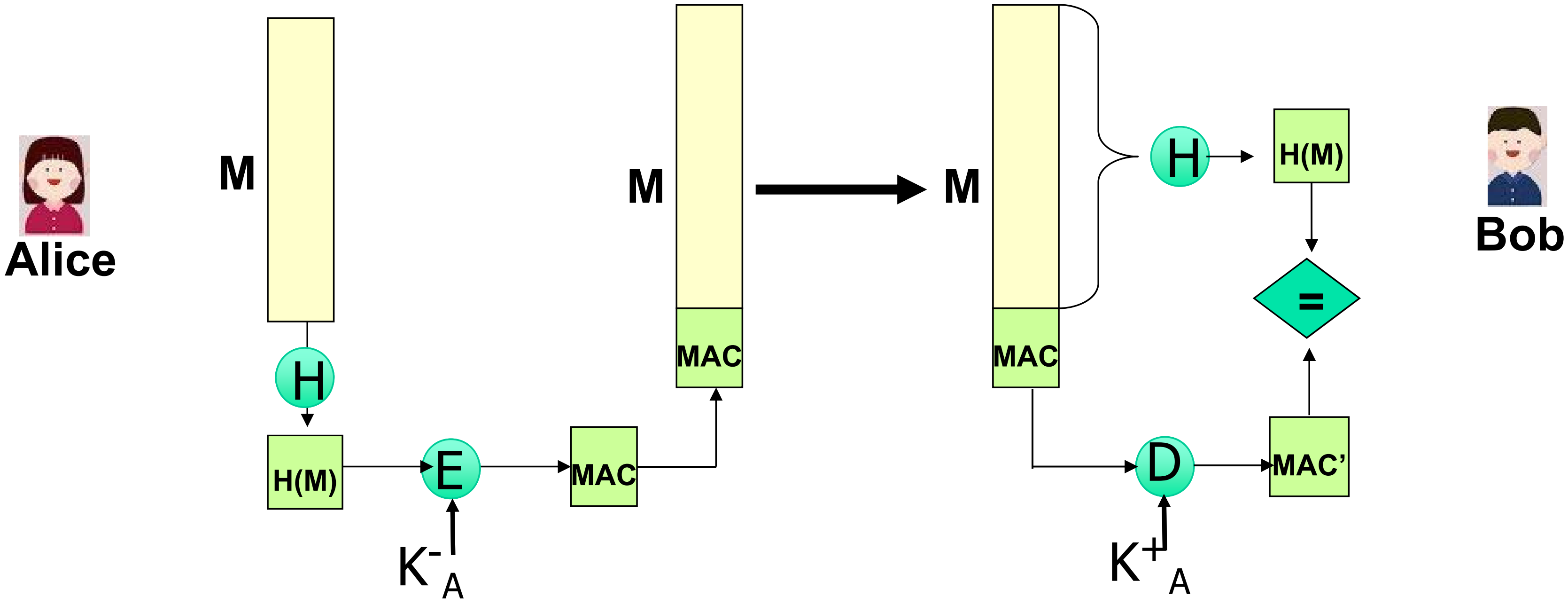
2st Approach: MAC by hash + symmetric-key

- Using a symmetric-key, a sender generates a small block of data which is the encrypted hashed value of a message as a MAC, and appends it to the message.



3rd Approach: MAC by hash + asymmetric-key

- Using a symmetric-key, a sender generates a small block of data which is the encrypted hash value of a message using her private key as a MAC, and appends it to the message.



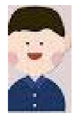
Motivation: only hash function for MAC

- Can we achieve the message integrity and authentication without resorting to encryption?
  - Encryption software is slow even though the message size is small.
  - Encryption hardware is not negligible and it is optimized for large message size.
- Are there any techniques which can avoid any encryption and only use the hash function?

Computing MAC by hash function

- One of the main applications of the hash function is to generate a small block of message tag which is called MAC.
- It can generate MAC with less computation because it doesn't require any encryption algorithm.
- Of course, the only hashed value of a message can not provide the message integrity.


Alice

Bob


message x
h = Hash(x)

(x, h) →

h' = Hash(x)
If h=h', valid

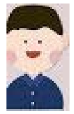
MAC with shared secret key

- Hash functions such as SHA-1 does not rely on a secret key.
- Can we also achieve message authenticity using MAC with hash function?
  - Answer is a keyed hash.
- A keyed hash incorporates a secret key into existing hash function algorithm.
- In a keyed hash, a hash function is treated as a "black box," which means any available hash function can be used.

A keyed hash for MAC

 Alice

shared secret key: K

Bob 

shared secret key: K

message M

h = Hash(M,K)

(M, h)

h' = Hash(M,K)

If h=h', then valid

HMAC

- HMAC is the keyed hash algorithm which is most widely accepted and used in real life applications.
- Proposed in 1996
- Widely used in practice, eg, IPsec, SSL/TSL
- RFC 2104
- double hashes
  - H(K|| H(K||M)) where K is secret key and M is message

How to construct HMAC

- How do we mix the shared secret key K and message M?
- Two options
  - $m = h(K||M)$
  - $m = h(M||K)$
- Which is better?

H: hash function

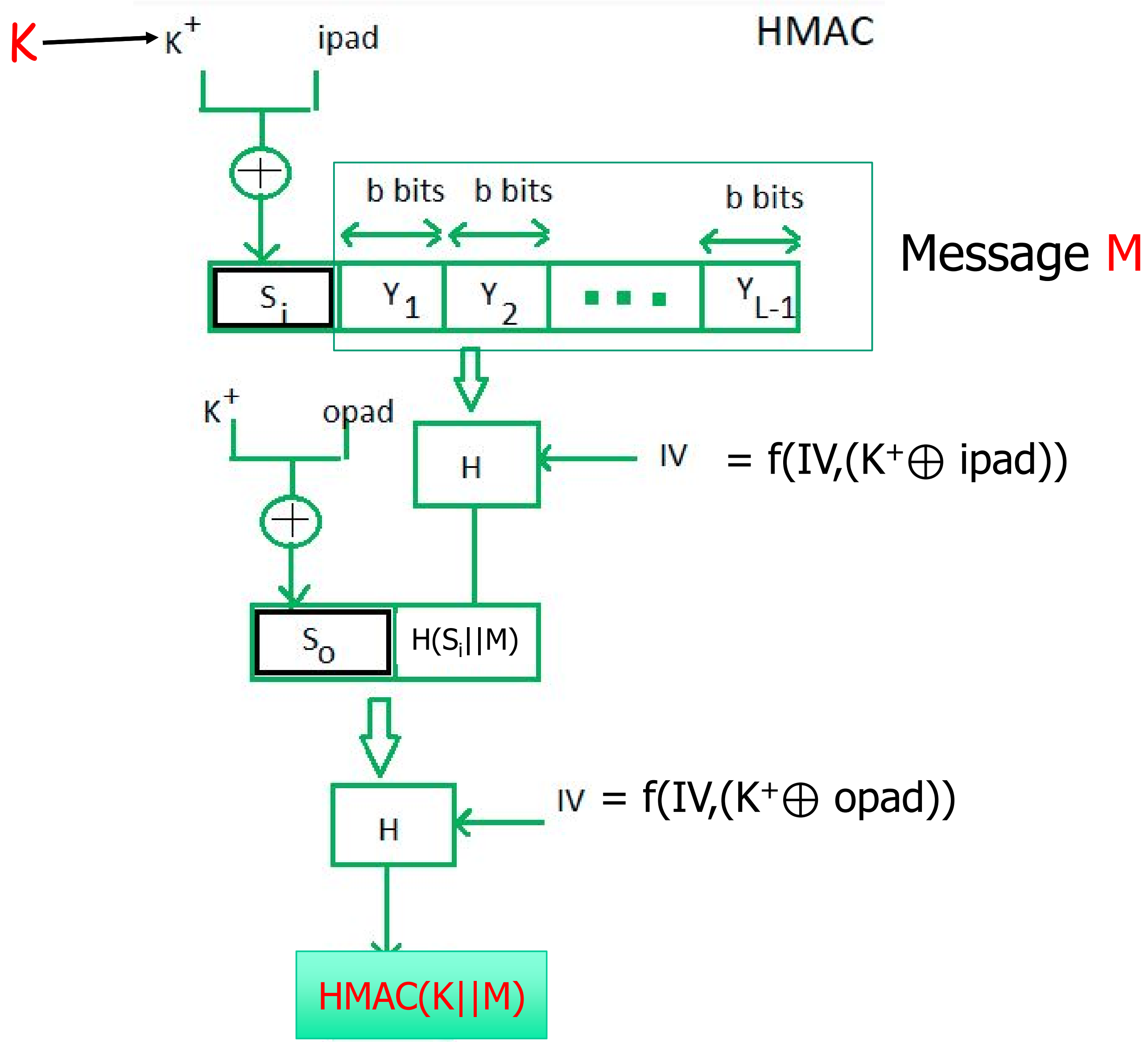
$$MAC(K, M) = H(K^+ \oplus opad \,||\, H(K^+ \oplus ipad \,||\, M))$$

K^+: extended secrete key (if K is less than 512 bits)

$$K^+ = \underbrace{00\ldots0||K}_{\text{hash input length, eg, 512 bits}}$$

$$ipad = \underbrace{00110110\ldots\ldots00110110}_{} = 3636..36,$$

$$opad = \underbrace{01011100\ldots\ldots01011100}_{\text{hash input length, eg, 512 bits}} = 5C5C..5C$$


K
K+
ipad
HMAC
b bits
b bits
b bits
Message M
Si
Y1
Y2
YL-1
K+
opad
H
IV
= f(IV,(K+⊕ ipad))
So
H(Si||M)
H
IV = f(IV,(K+⊕ opad))
HMAC(K||M)

HMAC computation

- HMAC approximately takes the same time to execute as the embedded hash function for long messages.
- IV can be precomputed and computed every time the key (K) changes.

HMAC security

- Any hash function can be chosen and used without affecting HMAC operation.
- HMAC often uses MD5 as a hash function. Then is HMAC not secure?
  - For attacking MD5, the attacker can choose any set of messages to find out the message to match the hash values.
    - Takes $2^{n/2}$
  - But for HMAC, he has to find out the pair (K, M) to match the hash value, especially online, which is infeasible even though the hash function is MD5.