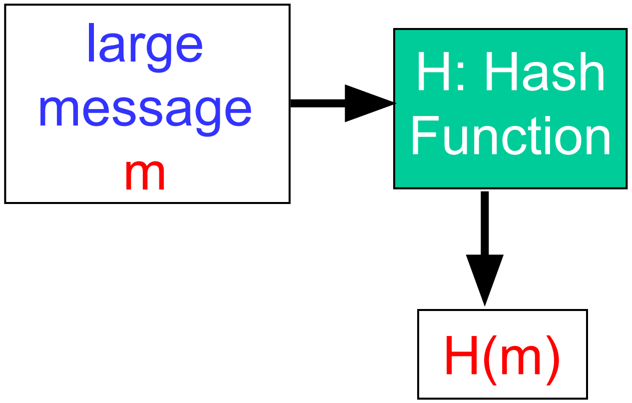
F6: Message authentication code (MAC)

tkkwon@snu.ac.kr

MESSAGE AUTHENTICATION CODE (MAC)

Hash function review

- Function H( ) that takes as input an arbitrary length message and outputs a fixed-length string: "message signature"
- Note that H( ) can be a many-to-1 function
- H( ) is "hash function"
  - MD5, SHA-1



- Desirable properties:
  - Easy to calculate
  - Irreversibility: Can't determine m from H(m)
  - Collision resistance: Computationally difficult to produce m and m' such that H(m) = H(m')
  - Seemingly random output

Source: Kurose at UMass

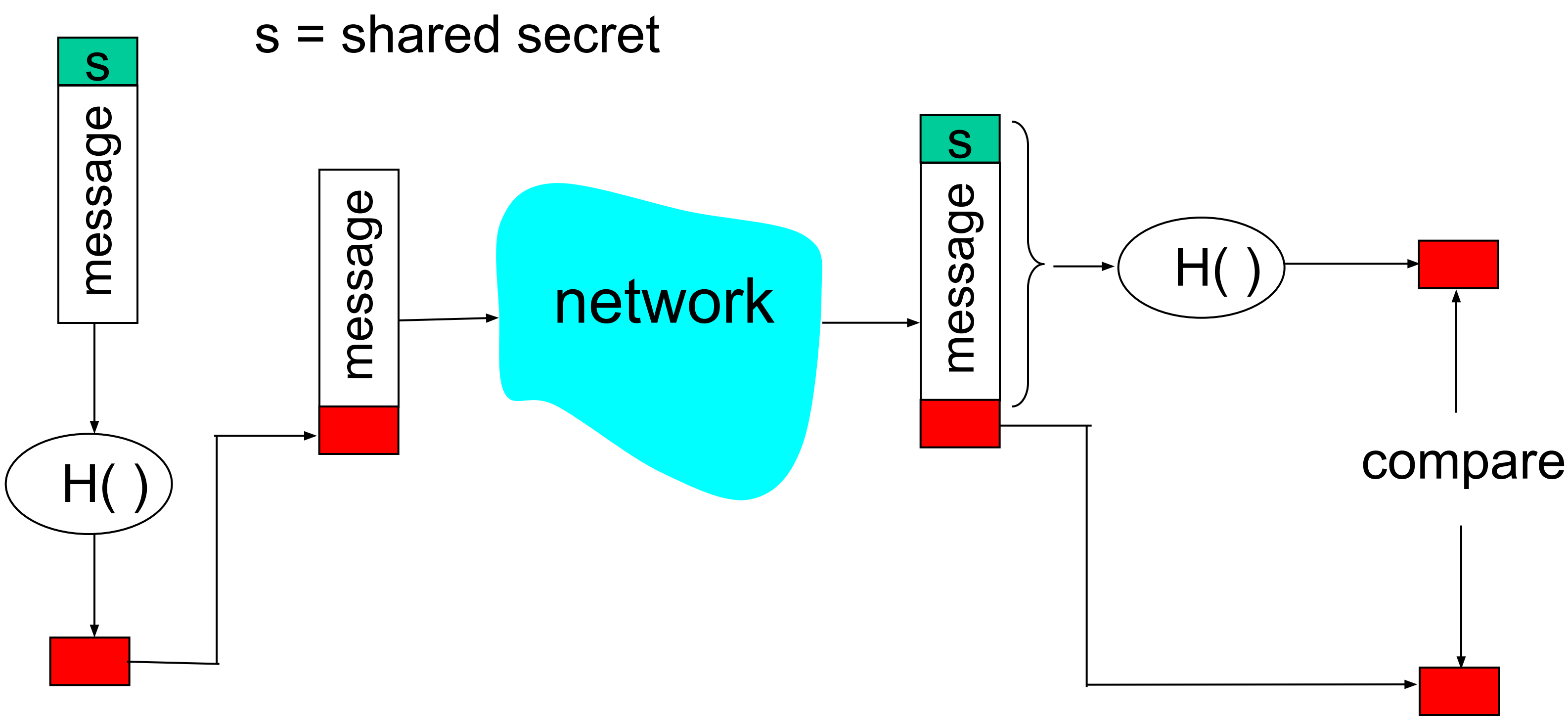
MAC

- verifies that the sender has the key
  - Authentication
  - Shared key vs. private key
- Verifies that message is not altered
  - Integrity
  - Message integrity code (MIC)
- Hash function or block cipher is usually used

MAC properties

- Cryptographic checksum
  - A MAC is a cryptographically secure authentication tag for given message.
- Symmetric
  - MACs are based on secret symmetric keys. The generating and verifying parties must share a secret key.
- Arbitrary message size
  - MACs accept messages of arbitrary length.
- Fixed output length
  - MACs generate fixed-size authentication tags.
- Message integrity
  - MACs provide message integrity: Any manipulations of a message during transit will be detected by the receiver.
- Message authentication
  - The receiving party is assured of the origin of the message.
- No non-repudiation
  - Since MACs are based on symmetric principles, they do not provide non-repudiation.

Message Authentication Code (MAC)



- No encryption!
- Also called “keyed hash”
- Notation: MAC(s,m) = H(s||m) ; send m || H(s||m)

- How about MAC(s,m) is generated by H(m||s) ?

Source: Kurose at UMass

Message Authentication Code

- A MAC scheme is a hash family, used for message authentication
- $MAC(K,M) = H_K(M)$
- The sender and the receiver share secret K
- The sender sends $(M, H_k(M))$
- The receiver receives (X,Y) and verifies that $H_K(X)=Y$, if so, then accepts the message as from the sender
- To be secure, an adversary shouldn't be able to come up with (X',Y') such that $H_K(X')=Y'$

how to construct a MAC with a key?

- secret prefix method
  - $H(K \mid\mid m)$
  - length extension attack
- secret suffix method
  - $H(m \mid\mid K)$
  - partial message collision attack
- envelope method
  - $H(K_1 \mid\mid m \mid\mid K_2)$
  - key discovery attack
- nested approach
  - $H(K_1 \mid\mid H(K_2 \mid\mid m))$

Two Weaknesses of Merkle-Dämgard construction

- Length extension attack
- Partial-message collision attack

Weakness1: Length extension attack

- Consider a block-based hash like SHA-1, with input blocks $m=(m_1, m_2, \ldots, m_k)$, and hash $H(m)$
- A new message $m' = (m_1, m_2, \ldots, m_k, m_{k+1})$, will have hash $H(m')=h(H(m),m_{k+1})$
  - h is a compression function
- In systems such as authentication applications, where we can calculate H(m||x), Eve can append extra text x to m and also update the hash

Constructing MAC from Hash Functions

- Let H be a one-way hash function

- MAC(K,M) = H(K || M)
  - Because of the Merkle-Damgard construction for hash functions, given M and t=H(K || M), an adversary can compute both M'=M||X and t', such that H(K||M') = t'
  - Then the receiver will conclude that MAC is fine

  - Thus, secret prefix method (secret || message) is insecure

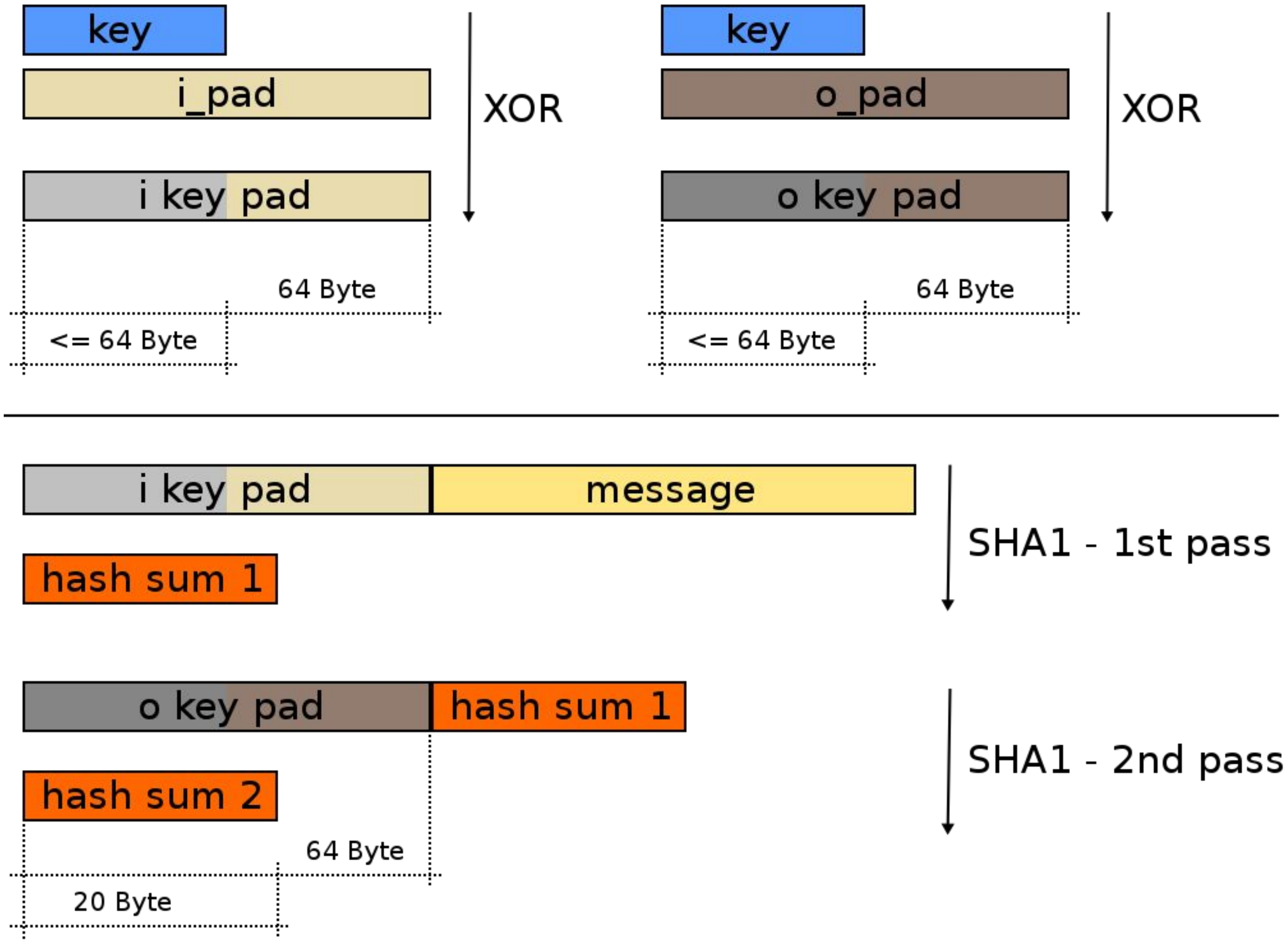
Weakness 2: partial message collision attack

- First, the attacker has to find 2 strings m and m' that lead to a collision when hashed by h(), i.e., the birthday attack.
  - h(m) = h(m')
- Then he obtains the MAC h(m||K) for message m
- now the attacker can forge a text-MAC pair (m',h(m||K))

- Secret suffix method (message || secret) is not secure

So we need a complicated MAC

- HMAC



Source: wikipedia

HMAC: Constructing MAC from Cryptographic Hash Functions

At high level: $HMAC_K[M] = H(K \,||\, H(K \,||\, M)$
$)$

$HMAC_K[M] = Hash[(K^+ \oplus opad) \,||\, Hash[(K^+ \oplus ipad)||M)]]$

- K^+ is the key padded (with 0) to B bytes, the input block size of the hash function
- ipad = the byte 0x36 repeated B times
- opad = the byte 0x5C repeated B times.

The choice of ipad and opad values

~: negation

- Goals
  - simple representation
  - “High” hamming distance between two values
- If ipad = opad, there might be vulnerability
  - Hamming distance 0
  - Same key is used twice
- If ipad = ~opad, there might be another vulnerability
  - Hamming distance 8
  - DES complement property, $c = E(k,m) \rightarrow \sim c = E(\sim k, \sim m)$
- Thus the hamming distance is chosen to be 4 between 0x36 and 0x5C

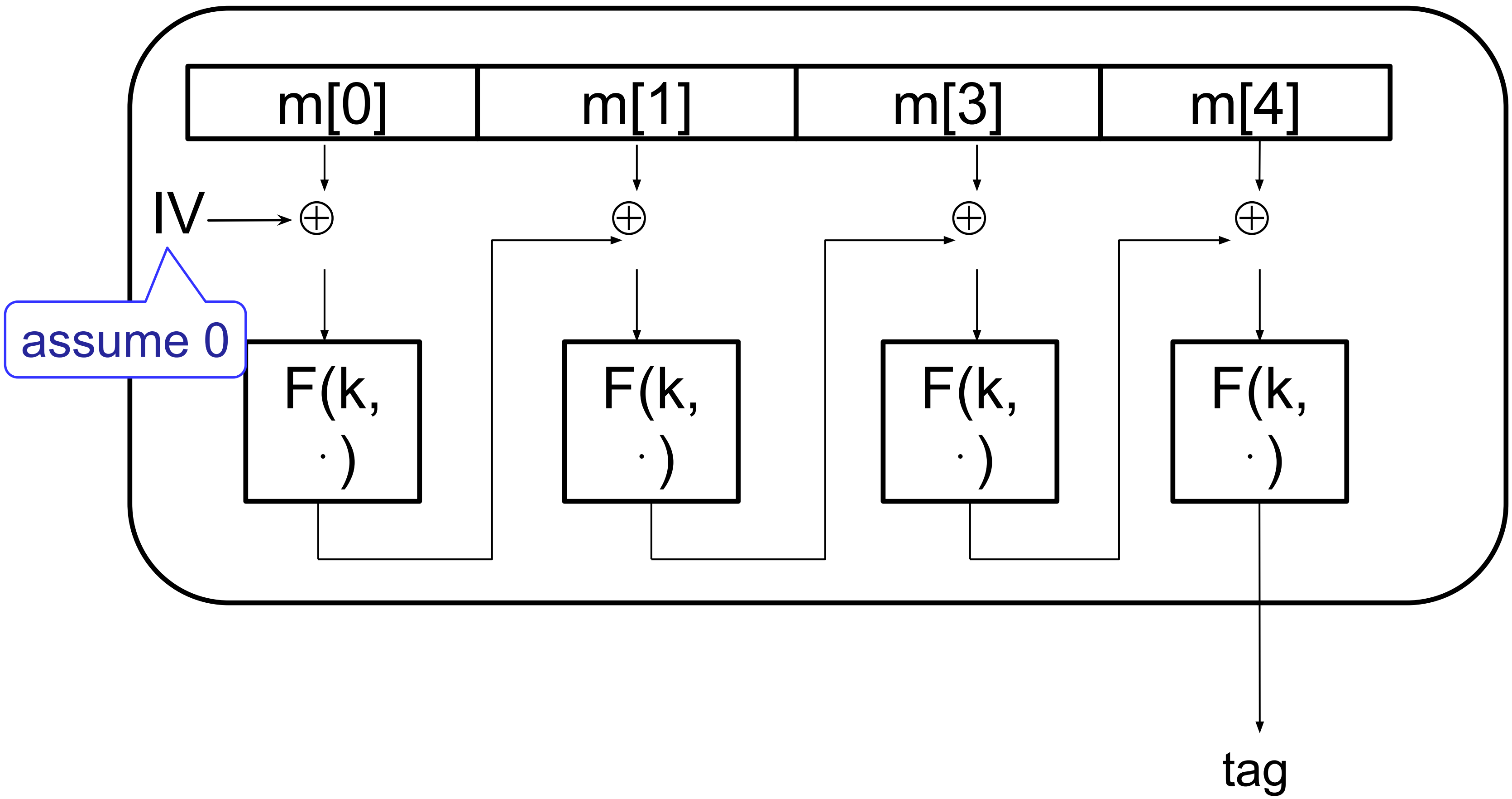
HMAC Security

- If used with a secure hash function (e.g., SHA-256) and according to the specification (key size, and use correct output), no known practical attacks against HMAC

CBC-MAC

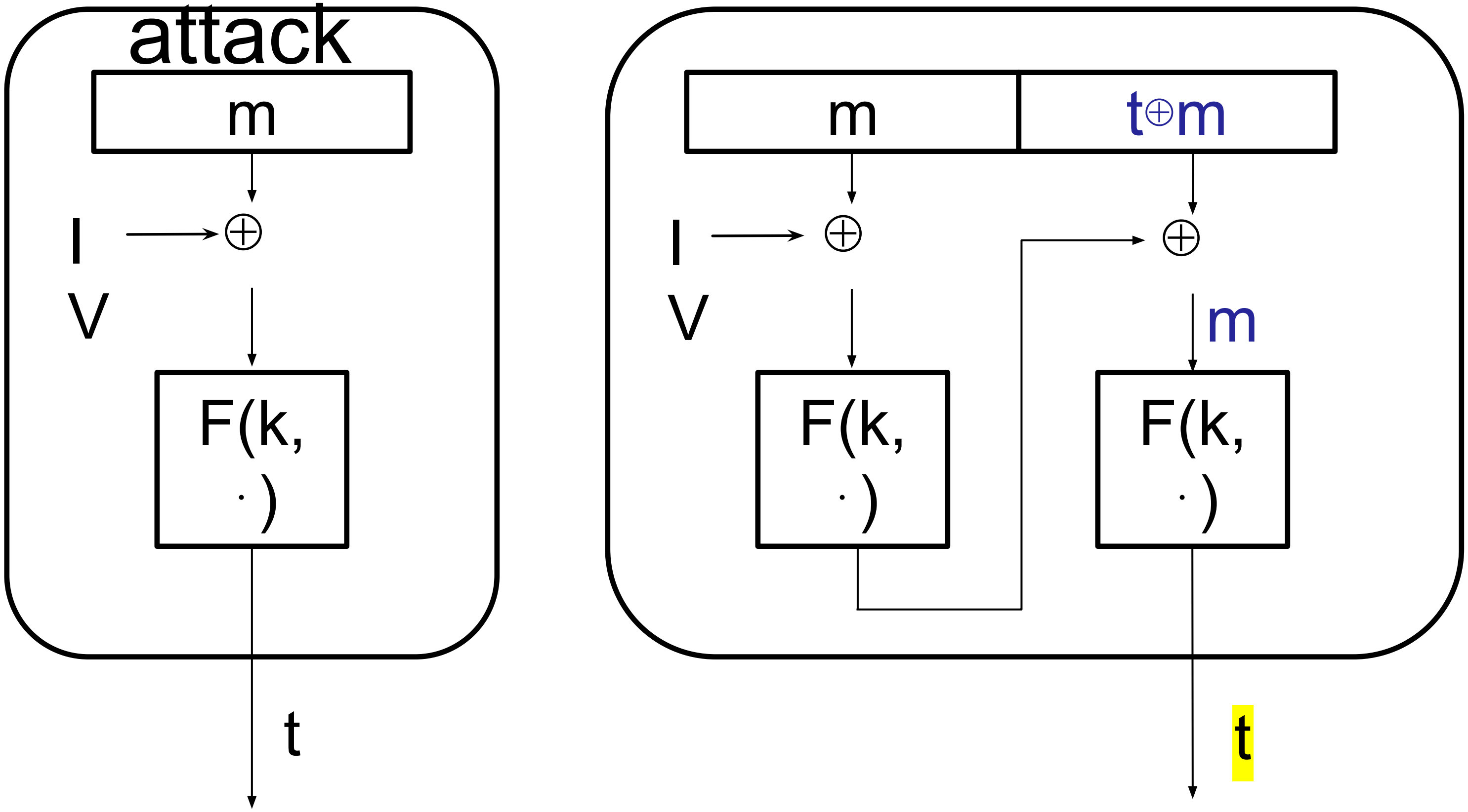
- We can rely on a block cipher to generate a MAC

<u>*original CBC*</u>



Attack on CBC-MAC

Break in 1-chosen message attack



Given CBC(k,m) = t, then CBC(k, m || t⊕m) = F(k, F(k,m)⊕(t⊕m))
= F(k, t⊕(t⊕m)) = F(k,m) = t

ECBC-MAC

- Encrypt-last-block CBC-MAC (ECBC-MAC)