

# MAS 433: Cryptography

## Lecture 12 Message Authentication Code

Wu Hongjun

# Lecture Outline

- Classical ciphers
- Symmetric key encryption
- Hash function and Message Authentication Code
  - Birthday attack
  - Hash function
  - **Message Authentication Code**
    - **CBC-MAC, CMAC**
    - **HMAC**
- Public key encryption
- Digital signature
- Key establishment and management
- Introduction to other cryptographic topics

# Recommended Reading

- CTP Section 4.4
- HAC Section 9.5
- NIST documents
  - CMAC: [http://csrc.nist.gov/publications/nistpubs/800-38B/SP\\_800-38B.pdf](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf)
  - HMAC: <http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf>
- Wikipedia
  - Message Authentication Code  
[http://en.wikipedia.org/wiki/Message\\_authentication\\_code](http://en.wikipedia.org/wiki/Message_authentication_code)
  - CBC-MAC  
<http://en.wikipedia.org/wiki/CBC-MAC>
  - CMAC  
<http://en.wikipedia.org/wiki/CMAC>
  - HMAC  
<http://en.wikipedia.org/wiki/HMAC>

# Message Authentication

- Whether the received message is sent from the sender
  - Message authentication implies data integrity
    - Data integrity: whether there is unauthorized modification to the message

# Message Authentication

- Inconvenient to use hash function itself for message authentication
  - A message digest represents only one message (computationally, for strong hash function)
  - But how to authenticate the message digest?

# Message Authentication

- Two approaches to authenticate message
  - Symmetric key approach:  
**Message Authentication Code**
    - Compress a message together with a secret key
    - The person who knows the secret key can verify the authenticity of the message
  - Public key approach: **digital signature**

# Message Authentication Code

- Message Authentication Code
  - An algorithm that compresses a **secret key** and a **message** with arbitrary length into a fixed length output (we call this output as message authentication tag, or MAC)
  - Verification: After receiving a message together with the attached authentication tag, repeat the above process to generate an authentication tag, compare it with the tag attached to the message, and check whether the message is sent from the person who posses the same secret key

Note: another frequently used MAC refers to Media Access Control address (MAC address). For example, each network card in a computer has a 48-bit unique identifier (MAC address).

# Message Authentication Code

- Two security parameters
  - The size of the secret key
    - Should be large enough to resist brute force attack
  - The size of the message authentication tag (MAC)
    - For  $n$ -bit MAC, the probability of modifying (or forging) a message without being detected is at least  $2^{-n}$ .
      - The size of MAC should not be too small
      - Normally, the MAC size should better be larger than 64 bits

# Message Authentication Code

- Security requirements on MAC algorithm
  - Attacker knows messages and their MACs
  - The key should remain secret even if it has been used to generate many MACs
    - Large key size
    - The MAC algorithm should resist other key recovery attacks
  - Without knowing the secret key, any forged or modified message can pass verification with negligible probability
    - An attacker may obtain the MACs of many messages, then try to forge only one message (generate the MAC for a forged message without knowing the secret key)

# Message Authentication Code

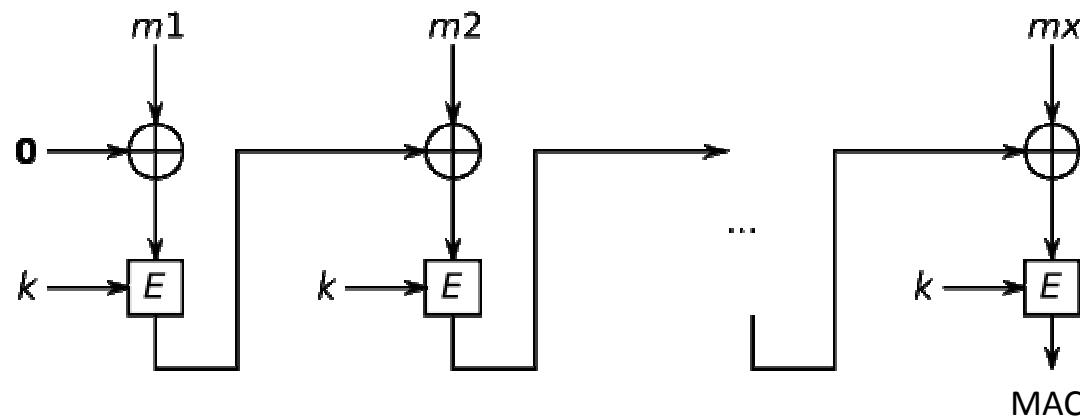
- In general, it is required to use different keys for encryption & authentication
  - Reason: the security of encryption & authentication can be separated
  - It is not advised to use the same key for different purposes
- If both symmetric key encryption & authentication are required
  - It is recommended to encrypt the message first, then authenticate the ciphertext
    - So that the security of authentication would not affect the security of encryption

# MAC Constructions

- MAC algorithm based on block cipher
  - CBC-MAC
  - CMAC (NIST recommendation)
- MAC algorithm based on hash function
  - HMAC (NIST standard)
- Dedicated MAC algorithm

# CBC-MAC

- Construct MAC algorithm from block cipher in CBC mode



- Differences between CBC-MAC and CBC encryption
  - CBC-MAC
    - Authentication, fixed public IV (0), output is the MAC
  - CBC encryption:
    - Encryption, random public IV, output is the ciphertext

# CBC-MAC

- CBC-MAC is insecure
  - Attack 1. Use the MACs of two messages, the MAC of a new message can be generated without knowing the secret key
    - Example:

$$M = m_0 \parallel m_1$$

$$\text{CBC-MAC}(M) = t$$

$$M' = m'_0 \parallel m'_1 \parallel m'_2$$

$$\text{CBC-MAC}(M') = t'$$

The new message:  $m_0 \parallel m_1 \parallel (m'_0 \oplus t) \parallel m'_1 \parallel m'_2$

The forged tag:  $t'$

# CBC-MAC

- CBC-MAC is insecure
  - Attack 2. With message length being appended to message, message can still be forged
    - Example:

Let  $M'_1$  represents  $M_1$  with length padding

$M'_2$  represents  $M_2$  with length padding

Let  $M_3 = M'_1 || a_0 || a_1 || a_2$ , where each  $a_i$  represents one message block.

Suppose now an attacker knows that the MACs of  $M_1, M_2, M_3$  are  $t_1, t_2, t_3$ , then an attacker can generate the MAC for the following message without knowing the secret key :

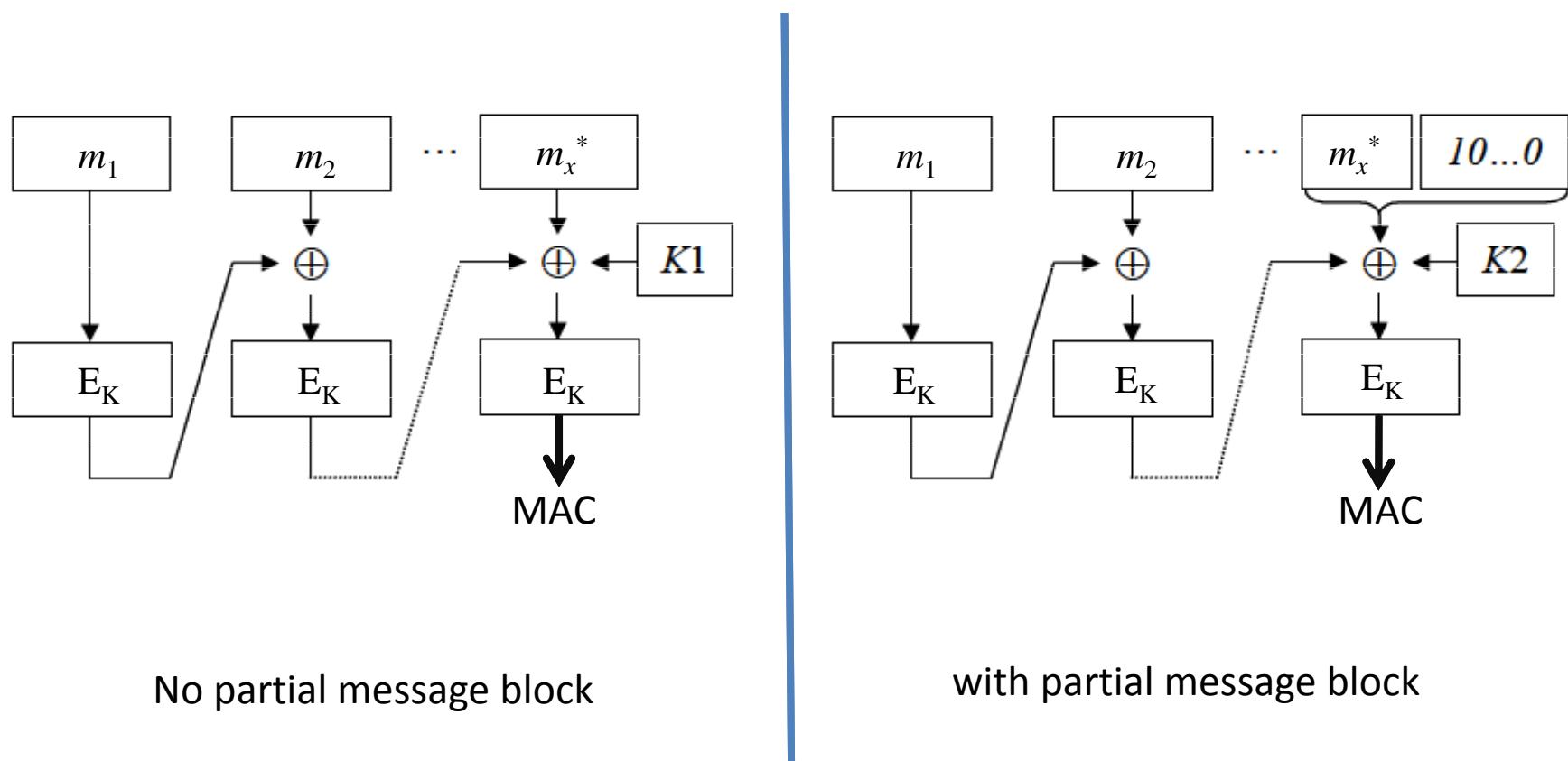
$$M_4 = M'_2 || (a_0 \oplus t_1 \oplus t_2) || a_1 || a_2, \text{ and the MAC is } t_3$$

# CMAC (Cipher-based Message Authentication Code)

- NIST recommendation
- Strengthen the CBC-MAC
  - Use **an additional key ( $K_1$  or  $K_2$ ) for the last message block** to thwart the attacks on CBC-MAC
  - The last message block (to eliminate the ambiguity in the last block: full or partial, how ‘partial’)
    - $K_1$  is used if it is full block
    - $K_2$  is used if it is partial block
      - Pad the partial block by bit ‘1’ followed by some zero bits
  - Derive  $K_1$  and  $K_2$  from the secret key  $K$
  - If the MAC size is less than the block size of block cipher, truncate some less significant bits

# CMAC

- CMAC without considering truncation



# CMAC

- Derive  $K_1$  and  $K_2$  from  $K$ 
    - Let  $W = E_K(0)$
    - Represent  $W$  as polynomial over GF(2)
      - If the block size of block cipher is 128 bits:
        - $K_1 = W \cdot x \pmod{(x^{128} + x^7 + x^2 + x + 1)}$
        - $K_2 = K_1 \cdot x \pmod{(x^{128} + x^7 + x^2 + x + 1)}$
      - If the block size of block cipher is 64 bits:
        - $K_1 = W \cdot x \pmod{(x^{64} + x^4 + x^3 + x + 1)}$
        - $K_2 = K_1 \cdot x \pmod{(x^{64} + x^4 + x^3 + x + 1)}$
- K<sub>1</sub> should not  
be set as W!
- Not a good idea to let K<sub>1</sub>  
and K<sub>2</sub> linearly dependent

# CMAC

- Security of CMAC
  - With  $2^{n/2}$  MACs being generated from the same key, if the messages are with certain pattern, an attacker can generate the MAC of a new message without knowing the secret key
    - Example:

Find two messages  $M = m_0 \parallel m_1 \parallel m_2 \parallel m_3 \parallel m_4$

$$M' = m'_0 \parallel m'_1 \parallel m_2 \parallel m_3 \parallel m_4$$

with the same MAC  $t$  (birthday attack). Then

$$E_K(m_0) \oplus m_1 = E_K(m'_0) \oplus m'_1$$

Now an attacker requests for the MAC of

$$M'' = m_0 \parallel (m_1 \oplus x) \parallel y \quad (\text{for any } x \text{ and } y)$$

the attacker can forge the new message

$$M''' = m'_0 \parallel (m'_1 \oplus x) \parallel y$$

(the MAC of  $M'''$  is the same as that of  $M''$ )

# HMAC

- The Keyed-Hash Message Authentication Code (HMAC)
  - NIST standard (2002)
  - Construct a secure MAC from a strong hash function

# HMAC

- How to construct a secure MAC algorithm from a strong hash function?
  - Method 1: key-prefix (**insecure**)
    - The key is prepended to the message, then hash
$$\text{MAC}_K(M) = \text{Hash}(K \parallel M)$$
(‘ $\parallel$ ’ indicates concatenation)
    - Insecure
      - An attacker can extend the message and generate new MAC for the extended message (if there is no finalization stage in the Merkle-Damgard construction, such as SHA-1, SHA-2).
      - Details: Suppose that  $(K \parallel M)$  after padding becomes  $(K \parallel M \parallel p)$ , and the attacker knows the MAC value  $\text{Hash}(K \parallel M \parallel p)$ . Then an attacker can compute  $\text{Hash}(K \parallel M \parallel p \parallel x)$  for any  $x$  due to the iterated nature of has function

# HMAC

- How to construct a secure MAC algorithm from a strong hash function?
  - Method 2: Key-suffix (**insecure**)
    - A secret key is appended to the end of message, then hash
$$\text{MAC}_K(M) = \text{Hash}(M \parallel K)$$
    - Not strong
      - A MAC can be forged with  $2^{n/2}$  hash function computations
      - Details:
        - » An attacker can apply birthday attack to find two messages  $M$  and  $M'$  (with the same length) satisfying  $\text{Hash}(M) = \text{Hash}(M')$
        - » Then the attacker requests for the MAC of  $M$
        - » The attacker immediately knows the MAC of  $M'$   
(The forgery is successful, since the people who know the secret key  $K$  has never generated MAC for  $M'$ )

# HMAC

- How to construct a secure MAC algorithm from a strong hash function?
  - Method 3: Key-prefix + key-suffix (envelope)  
**(not strong)**
    - A secret key is used as both prefix and suffix, then hash
$$\text{MAC}_K(M) = \text{Hash}(K \parallel M \parallel K)$$
    - Not strong(suppose  $n$ -bit key, and hash function with  $n$ -bit message digest)
      - The attacker requests the MACs of more than  $2^{n/2}$  messages, then recovers the  $n$ -bit secret key
        - » Reason: depending on the message length, a secret key may be separated into two parts, and those two parts appear in two compression functions

# HMAC

- How to construct a secure MAC algorithm from a strong hash function?
  - Method 3: key-prefix + key-suffix (envelope)
    - Example: \*recovering 32 key bits
      - Suppose that the messages lengths are chosen so that after padding, the last block contains only  $n-32$  key bits, i.e., the previous block contains the first 32 bits of the key
      - An attacker requests for the MACs of about  $2^{n/2+0.5}$  messages, with difference in the last message block (but the last padded block is the same).
        - » The attacker obtains two collisions  $\text{MAC}(x) = \text{MAC}(x')$ :
          - one collision happens before compressing the last padded block  
i.e.,  $\text{hash}(K \parallel x \parallel K_{32}) = \text{hash}(K \parallel x' \parallel K_{32})$

# HMAC

- How to construct a secure MAC algorithm from a strong hash function?
  - Method 3: key-prefix + key-suffix (envelope)
    - Example: recovering 32 key bits (contd.)
      - Then an attacker requests for MACs of  $\text{MAC}(x \parallel r \parallel y)$  and  $\text{MAC}(x' \parallel r \parallel y)$ , where  $r$  is a 32-bit number, and  $y$  is a fixed arbitrary message.
        - » For all the  $2^{32}$  values of  $r$ , if any particular  $r$  satisfies that  $\text{MAC}_k(x \parallel r \parallel y) = \text{MAC}_k(x' \parallel r \parallel y)$ , then we know that this  $r$  is equal to the first 32 bits of the key with very high chance.
        - Reason: if  $\text{hash}(K \parallel x \parallel K_{32}) = \text{hash}(K \parallel x' \parallel K_{32})$ , and if some  $r = K_{32}$ , then  $\text{hash}(K \parallel x \parallel r) = \text{hash}(K \parallel x' \parallel r)$   
 $\Rightarrow \text{hash}(K \parallel x \parallel r \parallel y \parallel K) = \text{hash}(K \parallel x' \parallel r \parallel y \parallel K)$ ,  
i.e.,  $\text{MAC}_k(x \parallel r \parallel y) = \text{MAC}_k(x' \parallel r \parallel y)$

# HMAC

- How to construct a secure MAC algorithm from a strong function?
  - Method 4: **HMAC** (NIST standard, FIPS 198a)
    - Suppose: key size less than or equal to message block size
    - Append ‘0’s to the end of key  $K$  so that the resulting  **$K'$**  is with one message block length
      - Example: 128-bit key, and 512-bit message block size, then 384 ‘0’ bits are appended to the key
    - Let opad and ipad be two constants with one block size
      - opad = 0x5c5c5c5c5c5c .....
      - ipad = 0x363636363636 .....
    - Compute the MAC as

$$\text{MAC}_K(M) = \text{Hash}((K' \oplus \text{opad}) \parallel \text{Hash}((K' \oplus \text{ipad}) \parallel M))$$

---

# HMAC

- Security of HMAC
  - For a hash function with  $n$ -bit internal state (the size of  $H_i$  is  $n$ -bit), if a key is used to generate the MACs of  $2^{n/2}$  messages with certain pattern, message can be forged successfully (birthday attack)
    - Similar to (but somehow different from) the security of CMAC

# Message Authentication Code

- Repeated trials
  - For  $n$ -bit MAC, a forged message & its MAC can pass verification with probability at least  $2^{-n}$
  - However, if there is no mechanism that stops repeated trials, the chance to forge a message successfully can become higher
    - Example: 1024 trials, the probability that one of the forged messages & their MAC can pass the verification with probability at least  $2^{-n+10}$

# Summary

- Message Authentication Code
  - Compresses a secret key and a message into an authentication tag with fixed length
  - MAC based on block cipher
    - CBC-MAC
    - CMAC (NIST recommendation)
  - MAC based on hash function
    - HMAC (NIST standard)