CSIT 495/595 - Introduction to Cryptography
Message Authentication Codes

Bharath K. Samanthula
Department of Computer Science
Montclair State University

Outline

- MAC: Motivation and Definition
- Encrypted CBC-MAC
- Authenticated encryption and its types
- Secure communication sessions

Message Integrity

- Data Confidentiality: use encryption to prevent an adversary from learning anything about the content of the messages transmitted over an open communication channel
- Message Integrity (or Message Authentication): how a party receiving a message can be sure that it was sent by the claimed sender and was not modified in transit
- Example: suppose a user X sends a request to its bank over Internet to transfer \$ 1000 to one of his friends
  - Is the request authentic?
  - Are the transaction details modified by an adversary?
- Can we use standard error-correction codes to verify message integrity?

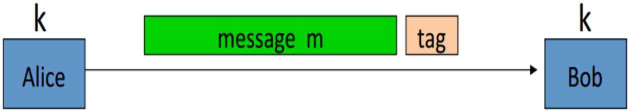
Data Confidentiality vs. Message Integrity

- Do applications always need both confidentiality and integrity? NO
  - protecting OS related files on hard disk
  - protecting banner ads on web pages
- Key observation: encryption schemes that ensure data confidentiality are not necessarily designed to guarantee message integrity
- Never assume that encryption by default solves the problem of message authentication
- Example 1 - Encryption using Stream Ciphers: Suppose $c = k \oplus m$. A single bit flip in c can yield an entirely different message ($\neq m$) upon decryption
- Example 2 - Encryption using Block Ciphers: changing a single bit affects one or more blocks

Message Authentication Code (MAC)

- In general, encryption does not solve the message integrity problem
- *Message Authentication Code (MAC)* enables the receiver to verify *authenticity* of the source and the *integrity* of the received message
- Key question: can we have a single encryption scheme that simultaneously achieves confidentiality and integrity?
  - YES!!.... Authenticated encryption (more details on this later)

MAC: Basic Idea



Generate tag:

$tag \leftarrow S(k, m)$

Verify tag:

$V(k, m, tag) \stackrel{?}{=}$ `yes'

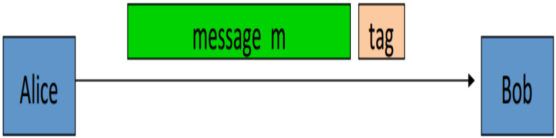
Def: **MAC** $I = (S,V)$ defined over (K,M,T) is a pair of algs:

- $S(k,m)$ outputs t in T
- $V(k,m,t)$ outputs `yes' or `no'

where S, V: MAC signing and verification algorithms
(K,M,T): key space, message space, and tag space

MAC requires Private Key

Can we use avoid private keys and use error-correcting codes such as CRC to ensure message integrity?



Generate tag:

$$tag \leftarrow CRC(m)$$

Verify tag:

$$V(m, tag) \stackrel{?}{=} \text{`yes'}$$

- Attacker can easily modify message m and recompute CRC
- For example, attacker can send (m’, t’) to Bob

MAC Security (1)

Attacker's power: **chosen message attack**

- for $m_1,m_2,\dots,m_q$ attacker is given $t_i \leftarrow S(k,m_i)$

Attacker's goal: **existential forgery**

- produce some <u>**new**</u> valid message/tag pair (m,t).

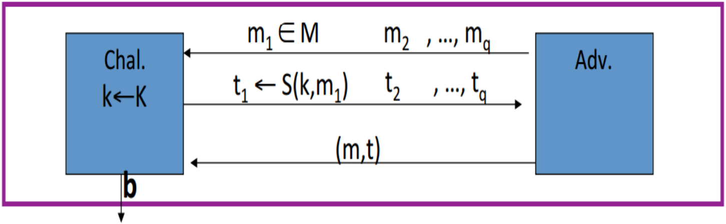
$$(m,t) \notin \{ (m_1,t_1), \dots, (m_q,t_q) \}$$

$\Rightarrow$ attacker cannot produce a valid tag for a new message

$\Rightarrow$ given (m,t) attacker cannot even produce (m,t') for $t' \neq t$

MAC Security (2)

- For a MAC $I=(S,V)$ and adv. A define a MAC game as:



$$\begin{cases} \mathbf{b}=1 & \text{if } V(k,m,t) = \text{`yes'} \text{ and } (m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \} \\ \mathbf{b}=0 & \text{otherwise} \end{cases}$$

Def: $I=(S,V)$ is a **secure MAC** if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{MAC}}[A,I] = \Pr[\text{Chal. outputs } 1] \quad \text{is "negligible."}$$

MAC - Sample Question

Let I = (S,V) be a MAC.

Suppose S(k,m) is always 5 bits long

Can this MAC be secure?

- No, an attacker can simply guess the tag for messages
- It depends on the details of the MAC
- Yes, the attacker cannot generate a valid tag for any message

MAC Example: Protecting System Files

Suppose at install time the system computes:

filename F_1	filename F_2	…	filename F_n
$t_1 = S(k,F_1)$	$t_2 = S(k,F_2)$		$t_n = S(k,F_n)$

k derived from user's password

Later a virus infects system and modifies system files

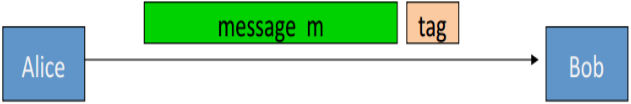
User reboots into clean OS and supplies his password

– Then: secure MAC $\Rightarrow$ all modified files will be detected

Secure PRF $\to$ Secure MAC

For a PRF $F: K \times X \to Y$ define a MAC $I_F = (S,V)$ as:

- $S(k,m) := F(k,m)$
- $V(k,m,t)$: output \`yes' if $t = F(k,m)$ and \`no' otherwise.



$\text{tag} \leftarrow F(k,m)$

accept msg if

$\text{tag} = F(k,m)$

Secure PRF $\rightarrow$ Secure MAC: A Bad Example

Suppose $F: K \times X \longrightarrow Y$ is a secure PRF with $Y = \{0,1\}^{10}$

Is the derived MAC I_F a secure MAC system?

- Yes, the MAC is secure because the PRF is secure
- No tags are too short: anyone can guess the tag for any msg
- It depends on the function F

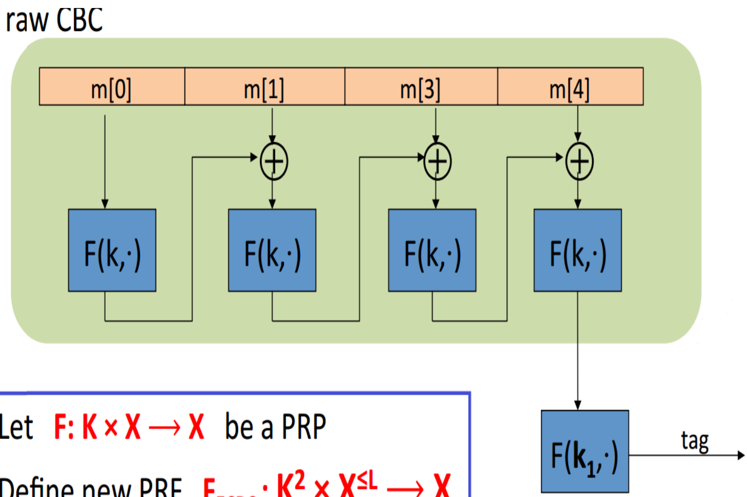
Two Well-known MACs

- Encrypted CBC-MAC
- HMAC (will be discussed in the next lecture)

CBC-MAC

- Similar to CBC-mode encryption, except with the following two differences.
  - IV is a random vector in CBC-mode encryption whereas there is no IV in CBC-MAC
  - In CBC-mode encryption, multiple ciphertexts (one for each block) are output whereas in CBC-MAC there is only one output from the final block

Encrypted CBC-MAC (ECBC-MAC)



Let $F: K \times X \longrightarrow X$ be a PRP

Define new PRF $F_{ECBC}: K^2 \times X^{\leq L} \longrightarrow X$

Encrypted CBC-MAC (ECBC-MAC)

- Key Questions:
  - Is the ECBC-MAC secure without the last encryption?
  - What happens if the message is not a multiple of block size?

Authenticated Encryption

- Goal: Can we ensure confidentiality and integrity simultaneously by default in the encryption scheme?
- A lack of integrity can sometimes lead to a breach in secrecy and vice versa
- There is no yet standard definition for authenticated encryption
- We will look at some generic approaches

Authenticated Encryption Constructions

- Let k_E and k_M denote encryption and message authentication keys
- Three natural Constructions:
  - Encrypt-and-authenticate
  - Authenticate-then-encrypt
  - Encrypt-then-authenticate

Encrypt-and-authenticate

- Given a message m, the sender transmits (c, t) to the receiver where

$$c \leftarrow Enc_{k_E}(m) \quad \text{and} \quad t \leftarrow S(k_M, m)$$

- The receiver decrypts c, and verifies the tag t
- Limitations:
  - tag $S(k_M, m)$ can leak information to eavesdropper
  - Example: For a MAC where the first bit of the tag is always equal to the first of the message
  - If a deterministic MAC like CBC-MAC is used, then the tag remains the same for a given message and key k_M

Authenticate-then-encrypt

- Given a message m, the sender computes the ciphertext c as follows

$$t \leftarrow S(k_M, m) \quad \text{and} \quad c \leftarrow Enc_{k_E}(m \| t)$$

- The receiver decrypts c to obtain $m \| t$ and verifies the tag t
- Limitations:
  - Two error messages possible: "bad padding" and "authentication failure"
  - If the attacker can distinguish between the two errors, he/she can recover the whole plaintext from a given ciphertext
  - A real-world attack: in configurations of IPsec

Encrypt-then-authenticate

- Given a message m, the sender computes (c, t) as follows

$$c \leftarrow Enc_{k_E}(m) \quad \text{and} \quad t \leftarrow S(k_M, c)$$

- Receiver first verifies t. If successful, then he decrypts c
- Observations:
  - This approach is sound, as long as the MAC is strongly secure
  - MAC is verified before decryption takes place, so MAC verification process cannot leak anything about the plaintext

Secure Communication Session (1)

- Often parties wish to communicate securely (that is achieving both secrecy and integrity) over the course of a communication session
- A naive way of encrypting the message using authenticated encryption may not work
- Potential Attacks
  - Re-ordering Attack: The attacker can swap the order of messages
  - Replay Attack: The attacker can send (replay) a valid ciphertext to Bob which was previously sent by Alice
  - Reflection Attack: An attacker can take a ciphertext c, which was earlier sent from Alice to Bob, and send it back to Alice

Secure Communication Session (2)

The above attacks can be easily prevented using counters and a directionality bit as follows

- Each party maintains two counter $ctr_{A,B}$ and $ctr_{B,A}$, both initialized to 0, to keep track of the number of messages sent from Alice to Bob and vice versa
- Directionality bits: $b_{A,B} = 0$ (message is from Alice to Bob) and $b_{B,A} = 1$ (message is from Bob to Alice)
- Alice sends $c \leftarrow Enc_k(b_{A,B} \| ctr_{A,B} \| m)$ to Bob and increments $ctr_{A,B}$
- Bob decrypts and parses $b \| ctr \| m$
- **If** $b = b_{A,B}$ and $ctr = ctr_{A,B}$, then
  Bob outputs m and increments $ctr_{A,B}$
  else
  Bob rejects the message

Summary

- MAC: Motivation and Definition
- Encrypted CBC-MAC
- Authenticated encryption and its types
- Secure communication sessions

Useful References

- Chapter 4, Introduction to Modern Cryptography by Jonathan Katz and Yehuda Lindell, 2nd Edition, CRC Press, 2015.
- `https://cseweb.ucsd.edu/~mihir/cse207/w-mac.pdf`