Message Authentication Codes

Reading: Chapter 4 of Katz & Lindell

Message authentication

- Bob receives a message m from Alice, he wants to know
  - (Data origin authentication) whether the message was really sent by Alice.
  - (Data integrity) whether the message has been modified.
- Two solutions:
  - Alice attaches a message authentication code (MAC) to the message (using a symmetric key to compute the MAC).
  - Or she attaches a signature to the message (using an asymmetric key to compute the signature).

Basic idea of MAC

- Message authentication protocol:
  1. Alice and Bob share a secret key k.
  2. To send a message m, Alice computes a tag $t := MAC_k(m)$ and sends (m, t) to Bob.
  3. On receiving (m', t'), Bob checks whether $t' = MAC_k(m')$. If so, he accepts the message; otherwise, he rejects it.
- The tag t is called a message authentication code (MAC).
- Security requirement: computationally infeasible to forge a valid pair $(x, MAC_k(x))$ without knowing the key k.

MAC Scheme (formal definition)

- A MAC scheme is a triple $(Gen,\ Mac,\ Vrfy)$:
  - Key generation algorithm: On input 1^n, outputs a key $k \leftarrow_u \{0,1\}^n$.
  - Tag generation algorithm Mac: On input a key k and a message $m \in M$, Mac outputs a tag t. We write $t \leftarrow Mac_k(m)$. (M is the message space. Assume $M = \{0,1\}^{l(n)}$ or $M = \{0,1\}^*$.)
  - Verification algorithm $Vrfy$: On input a key k, a message m, and a tag t, algorithm $Vrfy$ outputs 1 (meaning valid) or 0 (invalid). $Vrfy_k(m,t) = 0$ or 1.
  - Gen, Mac are probabilistic algorithms. $Vrfy$ is deterministic.
- Correctness requirement: for every $k \in K$ and $m \in M$,

$$Vrfy_k\left(m, Mac_k(m)\right) = 1.$$

- Canonical verification (used when *Mac* is deterministic):

$$Vrfy_k(m,t) = \begin{cases} 1 & \text{if } Mac_k(m) = t \\ 0 & \text{otherwise} \end{cases}$$

- If $M = \{0,1\}^{l(n)}$, the scheme is said to be a fixed-length MAC scheme for messages of length $l(n)$.

- Fixed-length MAC schemes are easier to construct.

- General MAC schemes for $M = \{0,1\}^*$ can be constructed from fixed-length schemes.

Chosen-Message Attacks on MAC Schemes

Experiment $\text{MAC-Forge}_{A,\Pi}(n)$:

1. A key $k \leftarrow G(1^n)$ is generated.
2. The adversary A is given input 1^n and oracle access to $Mac_k(\cdot)$.
   A may ask the oracle to compute tags for messages of its choice.
   Let Q be the set of all queries A has made to the oracle.
3. A eventually outputs a pair (m, t).
   (A tries to forge a valid pair of message and tag.)
4. $\text{MAC-Forge}_{A,\Pi}(n) = 1$ (A succeeds) if $m \notin Q$ and $Vrfy_k(m,t) = 1$.

- Remarks:
  - Adversary: an adaptive chosen-message attacker.
  - Forgery: an existential forgery.

MAC security: existential unforgeability under an adaptive chosen-message attack

Definition: A MAC scheme $(Gen, Mac, Vrfy)$ is existentially unforgeable under an adaptive chosen-message attack (or simply secure) if for all polynomial-time adversaries A, there exists a negligible function $negl$ such that

$$\Pr\left[\text{MAC-Forge}_{A,\Pi}(n)=1\right]\le negl(n)$$

or

$$\Pr\left[Vrfy_k\left(A^{Mac_k(\cdot)}\left(1^n\right)\right)=1:\ k\leftarrow_u \{0,1\}^n\right]\le negl(n)$$

where the output of A, (m,t), satisfies $m\notin Q$.

Strong MAC security

- If a MAC scheme is secure, the probability is negligible that A can forge a valid (m,t) with $m \notin Q$.
- However, it may be possible for A to forge a different valid tag $t' \neq t$ for some message $m \in Q$, where t is the tag returned by the oracle on m.
- If no adversary is able to do so, the MAC scheme is strongly secure.
- To formally define strong security, modify the experiment as follows:
  - Let $Q' = \{(m,t):\ m \in Q,\ t \text{ is the tag returned by the oracle on } m\}$.
  - A succeeds if an only if it outputs a valid pair $(m,t) \notin Q'$.
- If MAC is deterministic, then "secure" $\Leftrightarrow$ "strongly secure".

Constructing secure MAC schemes

- Let F be a pseudorandom function.
- We will use F to construct secure MAC schemes in several steps.
  - Secure fixed-length MAC schemes for messages of length n
  - Secure fixed-length MAC schemes for messages of length $n \cdot l(n)$
  - Secure MAC schemes for arbitrary-length messages
- For simplicity, assume message length is a multiple of n.

  (We can always do padding to make this assumption true.)

Secure MAC schemes for $M = \{0,1\}^n$

- Let F be a pseudorandom function.
- Fixed-length MAC scheme for messages of length n:
  - Key generation: On input 1^n, $k \leftarrow_u \{0,1\}^n$.
  - Tag generation: On input $k \in \{0,1\}^n$ and message $m \in \{0,1\}^n$, output the tag $t := F_k(m)$.
  - Verification: On input (m,t), $Vrfy_k(m,t) := \begin{cases} 1 & \text{if } F_k(m) = t \\ 0 & \text{otherwise} \end{cases}$
- Theorem: Such a MAC scheme is secure.

Basic CBC-MAC

- Let F be a pseudorandom function.
- Basic CBC-MAC works as follows:
  - Key generation: $k \leftarrow_u \{0,1\}^n$.
  - Tag generation: For key $k \in \{0,1\}^n$ and message $m \in \{0,1\}^{n \cdot q}$,
    - parse m as $m = (m_1, \ldots, m_q)$ // q blocks //
    - apply CBC to m with $IV = 0^n$, i.e., let

$$t_0 := 0^n \text{ and } t_i := F_k(m_i \oplus t_{i-1}) \text{ for } 1 \le i \le q$$

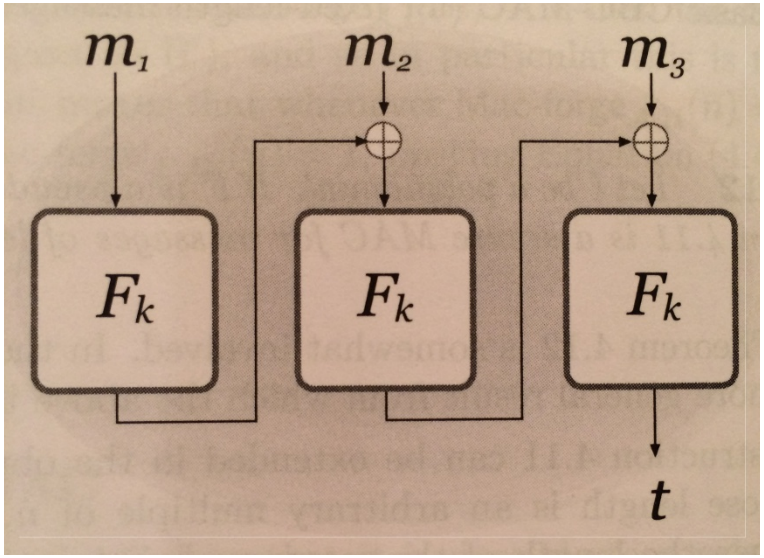
    - output t_q as the tag
  - Verification: canonical

- Theorem: For any fixed length function l, basic CBC-MAC is secure for messages of length $n \cdot l(n)$.

Remarks

- It is important that t_0 (IV) is fixed, or the scheme would be insecure.
- Also, the scheme would be insecure if message length is variable.
  - Suppose $t := Mac_k(m_1 \| m_2 \| m_3)$ and $t' := Mac_k(m_4)$.
  - Let m_4' be such that $t \oplus m_4' = m_4$.
  - Then $Mac_k(m_1 \| m_2 \| m_3 \| m_4') = t'$.

$$IV = 0^n$$

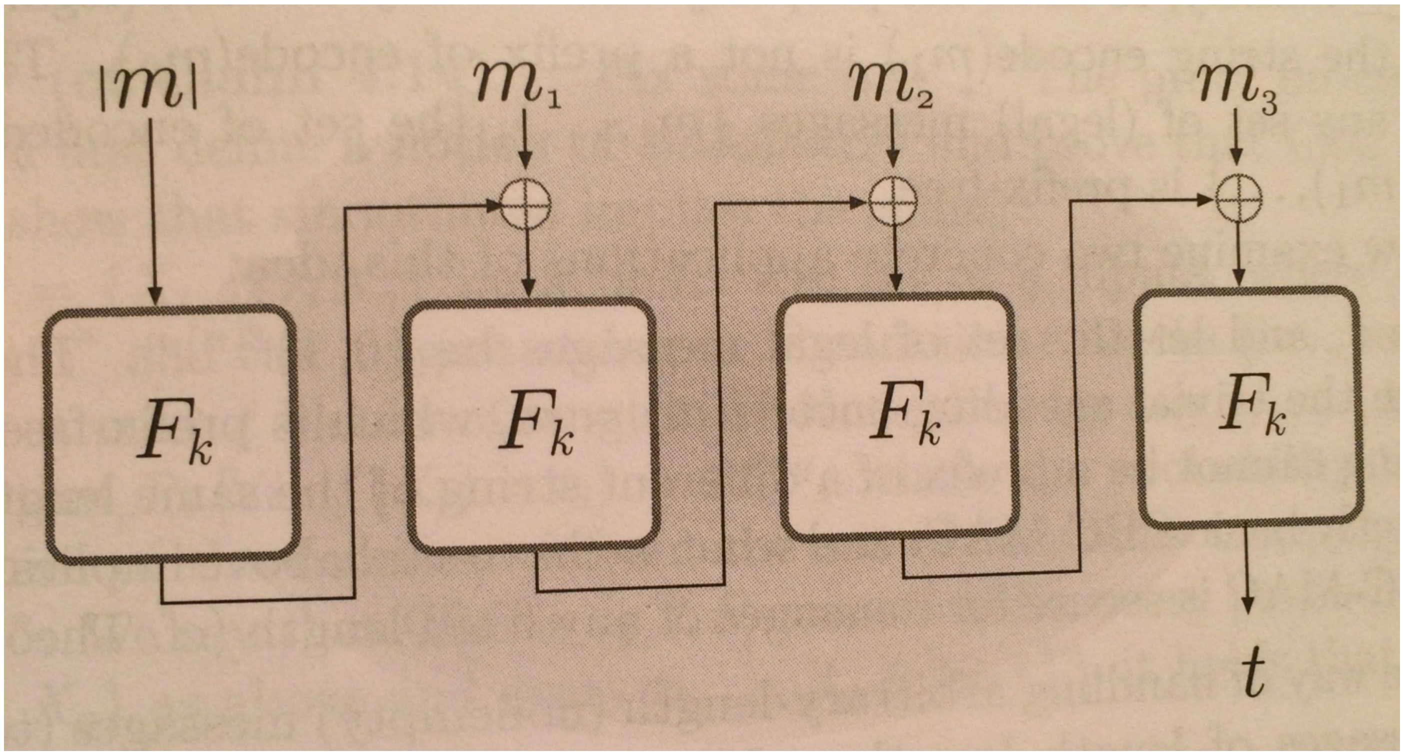


CBC-MAC for arbitrary-length messages

- A FIPS and ISO standard.
- There are several variants of CBC-MAC.

One variant of CBC-MAC:

- Prepend the message m with its length $|m|$ (as an n-bit string) and then compute basic CBC-MAC on the result.
- Remarks:
  - There is a limitation on $|m|$.
  - It would be insecure if $|m|$ is appended to the end of m.


|m|
m1
m2
m3
Fk
Fk
Fk
Fk
t

Another variant of CBC-MAC

- Generate two keys $k_1,\ k_2 \leftarrow_u \{0,1\}^n$.
- To authenticate a message m, let the tag be

$$t := F_{k_2}\left(\text{basic-CBC-MAC}_{k_1}(m)\right).$$

- One may use only one key k and generate $k_1,\ k_2$ from k:

$$k_1 := F_k(1) \quad \text{and} \quad k_2 := F_k(2)$$

Security of CBC-MAC (for arbitrary length)

Theorem: CBC-MAC is secure if F is a pseudorandom function.

- In practice, block ciphers (such as DES, AES) are used.

Authenticated Encryption

To ensure both secrecy and integrity

Unforgeable encryption

- Experiment $\text{Enc-Forge}_{A,\Pi}(n)$:
  - Run $Gen(1^n)$ to obtain a key k.
  - The adversary A is given 1^n and access to oracle $Enc_k(\cdot)$, and outputs a ciphertext c.
  - Let $m := Dec_k(c)$. Let Q be the set of all messages that A has asked the oracle for encryption.
  - The output of the experiment is 1 (A succeeds) if and only if m is a valid message ($m \in M$) and $m \notin Q$.

- Definition: An encryption scheme $\Pi = (Gen, Enc, Dec)$ is unforgeable if for every A, $\Pr\left[\text{Enc-Forge}_{A,\Pi}(n) = 1\right] \le negl(n)$.

Authenticated encryption scheme

- Definition: A symmetric-key encryption scheme is an authenticated encryption scheme if it is CCA-secure and unforgeable.
- We will construct an authenticated encryption scheme from a CPA-secure encryption scheme and a strongly secure MAC scheme.
- Three natural ways:
  - Encrypt and authenticate (insecure)
  - Authenticate then encrypt (insecure)
  - Encrypt then authenticate (secure)
- In the following slides, let m be a message, k_E an encryption key, and k_M a MAC key.

Encrypt and Authenticate

- Sender: encrypt and authenticate m independently. That is, the ciphertext is $\langle c, t \rangle$ where

$$c \leftarrow Enc_{k_E}(m), \qquad t \leftarrow Mac_{k_M}(m)$$

- Receiver: given ciphertext $\langle c, t \rangle$, do

$$m \leftarrow Dec_{k_E}(c), \text{ and then check if } Vrfy_{k_M}(m, t) = 1.$$

- Security:
  - Not necessarily EAV-secure, since t might leak info about m.
  - If Mac is deterministic (e.g., CBC-MAC), then the scheme is not CPA-secure.

Authenticate then Encrypt

- Sender: authenticate m first and then encrypt m and the tag. Thus, the ciphertext is c computed as:

$$t \leftarrow Mac_{k_M}(m), \qquad c \leftarrow Enc_{k_E}(m \| t)$$

- Receiver: given ciphertext c, do

$$m \| t \leftarrow Dec_{k_E}(c) \text{ and then check if } Vrfy_{k_M}(m,t) = 1.$$

- A potential attack:
  - Suppose PKCS#5 padding is used. Suppose the receiver does:

    if the padding is incorrect then return a "bad padding" error

    elseif the tag is incorrect then return a "bad mac" error.
  - The padding attack can be conducted to recover the entire $m \| t$.

Encrypt then Authenticate

- Sender: encrypt m first and then authenticate the result. Thus, the ciphertext is $\langle c, t \rangle$ where

$$c \leftarrow Enc_{k_E}(m), \qquad t \leftarrow Mac_{k_M}(c)$$

- Receiver: on receiving $\langle c, t \rangle$, if $Vrfy_{k_M}(c,t) = 1$ then $m \leftarrow Dec_{k_E}(c)$.

- Theorem: If the encryption scheme is CPA-secure and the MAC scheme is strongly secure, then the encryption-then-authenticate construction yields an authenticated encryption scheme.

CPA-secure encryption + strongly secure MAC

$\Rightarrow$ CCA-secure and unforgeable encryption