Message Authentication Codes

Chanathip Namprempre

Computer Science
Reed College

Agenda: Message Authentication Codes

1. Motivation: why integrity protection
2. Encryption doesn't provide authenticity.
3. MAC schemes: syntax and security definitions
4. Block cipher based MAC schemes: π_1, π_2, CBC MAC, XCBC
5. Example attacks and secure MAC schemes
6. Hash-based MAC scheme

Motivation

Message Authentication

We often want to protect integrity of messages.

Private-key setting : MACs

Public-key setting : Digital Signatures

Motivation

Integrity is important in many applications.

Forger	$\xrightarrow{\quad M \quad}$	Receiver
	fake deposits	[bank's computer]
	fake commands	[data server]
	fake financial data	[server of an investment company]

Receiver thinks M comes from some legitimate sender, but in fact M comes from Forger.

Points to remember

- We do not assume that M is to be kept secret.
- F (forger) controls the channel, i.e. F can drop, inject, modify, repeat packets.

Encryption does NOT provide anthenticity.

F wants to change deposit from 100 to 900.

Some may think "it is hard for F to change ciphertext so that it decrypts to 900 without knowing the key K."

WRONG! This is easy!

Example : suppose encryption is OTP.

Example :One-time Pad

- Suppose $M = 0001$ [i.e. 1 in decimal].
- F wants to change M to $M' = 1001$ [i.e. 9 in decimal].
- Suppose $K = 1010$.
- So $C = M \oplus K = 1011$.

$$\begin{array}{ll} F: & \text{take } C = 1011 \\ & \text{let } \Delta = M \oplus M' = 0001 \oplus 1001 = 1000 \\ & \text{compute } C' = C \oplus \Delta = 1011 \oplus 1000 = 0011 \end{array}$$

- When receiver decrypts, they get $0011 \oplus 1010 = 1001 = M'$!
- F did not need to know K.

False conclusions:

- “Don’t use OTP.”
  OTP is for secrecy! It does its job, not something else. (Don’t use a car if you want to fly.)
- “Example is contrived.”
  OTP is for real. CTR is pretty much OTP. CBC doesn’t work much better either.
- “Should add redundancy”
  Adding pads won’t help here.

Correct conclusions

Encryption gives you privacy, not authenticity.

[In fact, with most encryption schemes, any ciphertxt will decrypt to something.]

Bottom line:

Good cryptographic design is goal-oriented.

Must understand goal before designing schemes.

A good designer uses the right tool for the desired goal.

Message Authentication Codes

Syntax and security definitions

Syntax (MAC : scheme & tag)

Definition

A MAC scheme consists of 3 algorithms : $\pi = (\mathcal{K}, MAC, VF)$

$\mathcal{K}$: randomized key generation algorithm

$$K \xleftarrow{\$} \mathcal{K}$$

$[Keys(\pi) = \{K | K \text{ has non-zero probability of being output by } \mathcal{K}\}]$

MAC : MAC-generation algorithm
randomized or stateful

$$Tag \xleftarrow{\$} MAC_K(M)$$

$[M \in \{0,1\}^*, K \in Keys(\pi), Tag \in \{0,1\}^* \cup \{\perp\}]$

Syntax

VF : MAC-verification algorithm
deterministic

$$d \leftarrow VF_K(M, Tag)$$

$[M \in \{0,1\}^*, K \in Keys(\pi), Tag \in \{0,1\}^*, d \in \{0,1\}]$

Correctness

$\forall K \in Keys(\pi), M \in \{0,1\}^*,$

$$\Pr\left[\, Tag = \perp \text{ OR } VF_K(M, Tag) = 1 \;:\; Tag \stackrel{\$}{\leftarrow} MAC_K(M) \,\right] = 1\ .$$

Points to remember about syntax

- Just syntax. No security yet.
- VF is deterministic. Hard to require stateful receiver with consistent states.
- MAC-generation could be deterministic & stateless. When this happens, $VF_K(\cdot, \cdot)$ just recomputes Tag and compare, i.e.

  $VF_K(M, Tag)$
  $\quad Tag' \leftarrow MAC_K(M)$
  $\quad$ If $(Tag = Tag' \text{and} Tag' \neq \perp)$ then 1 else 0

  So for deterministic MAC, we can say $\pi = (\mathcal{K}, MAC)$.

Security definition

Issues

Want: hard for adversary F to forge valid tags.

- Want this for any messages, not just "meaningful" ones. (Want security guarantee for all applications)
- Want more than hardness of key recovery. (Maybe can forge without knowing key. That'd be bad.)
- Let F see valid message-tag pairs *before* forging? (No-message attacks? Chosen-message attacks?)
- Replay? (F sees a (M, Tag) pair and repeats it.) (Don't allow for now.)
- What if F can forge many pairs of valid (M, Tag)'s and is content if any of the pair is valid? (Let F submits many verification queries & wins if at least one is valid)

Security definition (cont.)

- "signing" query
  (# of queries $= q_s$; # of bits $= \mu_s$)
- verification query
  (# of queries $= q_v$; # of bits of M's $= \mu_v$)
- F win if $VF_K(\cdot)$ ever returns 1 on a pair (M, Tag) not previously returned by $MAC_K(\cdot)$.

WUF-CMA

Subroutine *Initialize*

$K \xleftarrow{\$} \mathsf{KG}\,;\ S \leftarrow \emptyset\,;\ win \leftarrow \text{false}$

Subroutine $\mathtt{Tag}(M)$

$S \leftarrow S \cup \{M\}\,;\ \text{Return } \mathsf{Tag}(K, M)$

Subroutine $\mathtt{Vf}(M, T)$

$v \leftarrow \mathsf{Vf}(M, T)$

If $v = 1$ and $M \notin S$ then $win \leftarrow$ true

Return v

Subroutine *Finalize*

Return *win*

Experiment $\mathbf{Exp}_{\mathsf{MA}}^{\mathrm{wuf\text{-}cma}}(A)$

Initialize

$A^{\mathtt{Tag},\mathtt{Vf}}$

Return *Finalize*

wuf-cma advantage

The **wuf-cma advantage** of an adversary A mounting a chosen-message attack against MA is

$$\mathbf{Adv}_{\mathsf{MA}}^{\mathrm{wuf\text{-}cma}}(A) = \Pr\left[\,\mathbf{Exp}_{\mathsf{MA}}^{\mathrm{wuf\text{-}cma}}(A) \Rightarrow \text{true}\,\right] .$$

SUF-CMA

Subroutine *Initialize*

$K \xleftarrow{\$} \mathrm{KG}\,;\; S \leftarrow \emptyset\,;\; \mathit{win} \leftarrow \text{false}$

Subroutine $\mathtt{Tag}(M)$

$T \xleftarrow{\$} \mathrm{Tag}(K, M)\,;\; S \leftarrow S \cup \{(M, T)\}$

Return $\mathrm{Tag}(K, M)$

Subroutine $\mathtt{Vf}(M, T)$

$v \leftarrow \mathrm{Vf}(M, T)$

If $v = 1$ and $(M, T) \notin S$ then $\mathit{win} \leftarrow$ true

Return v

Subroutine *Finalize*

Return *win*

Experiment $\mathbf{Exp}_{\mathrm{MA}}^{\mathrm{suf\text{-}cma}}$

Initialize

$A^{\mathtt{Tag},\mathtt{Vf}}$

Return *Finalize*

suf-cma advantage

The **suf-cma advantage** of an adversary A mounting a chosen-message attack against MA is

$$\mathbf{Adv}_{\mathrm{MA}}^{\mathrm{suf\text{-}cma}}(A) = \Pr\left[\mathbf{Exp}_{\mathrm{MA}}^{\mathrm{suf\text{-}cma}}(A) \Rightarrow \text{true}\right].$$

Block cipher based MAC schemes

Example: MAC scheme π_0

MAC scheme π_0

Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^L$ be a family of functions.

$MAC_K(M)$
    if $(|M| \neq n)$ then return $\perp$
    Return $F_K(M)$

Theorem: PRF $\Longrightarrow$ MAC

If $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^L$ is a secure PRF and 2^L is large, then π_0 is a secure MAC.

For every efficient adversary A against π_0, there exists an efficient adversary B against F such that

$$\mathbf{Adv}_{\pi_0}^{\mathrm{uf\text{-}cma}}(A) \leq \mathbf{Adv}_F^{\mathrm{prf}}(B) + \frac{1}{2^L} .$$

Example: MAC scheme π'_0

MAC scheme π'_0

Let $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^L$ and $F' : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^t$ be families of functions. Let F' be the same as F except the output is truncated to t bits.

$MAC_K(M)$
    if $(|M| \neq n)$ then return $\perp$
    Return $F'_K(M)$

Fact: Let t and L be integers and let $t < L$. Then,
if $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^L$ is a secure PRF, then
$F' : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^t$ is also a secure PRF.

So, if F is a secure PRF, then π'_0 is also a secure MAC.

MAC for large inputs?

What if the input messages are longer than one block?

Example: MAC scheme π_1

π_1

Let $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^L$ be a family of functions.

$MAC_K(M)$
- if $(|M| \bmod n \neq 0$ or $|M| = 0)$ then return $\perp$
- Break M into n-bit blocks $M = M[1] \ldots M[s]$
- for $i = 1$ to s do $y_i \leftarrow F_K(M[i])$
- $Tag \leftarrow y_1 \oplus \ldots \oplus y_s$
- Return Tag

Example : MAC scheme π_2

What if we modify π_1 to prevent the previous attack by enciphering the block number along with the data block?

π_2

Let m be an integer such that $1 \leq m \leq n - 1$.
Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^L$ be a family of functions.

$MAC_K(M)$

$t \leftarrow n - m$

if $(|M| \bmod t \neq 0$ or $|M| = 0$ or $|M|/t \geq 2^m)$ then return $\perp$

Break M into t-bit blocks $M = M[1] \ldots M[s]$

for $i = 1$ to s do $y_i \leftarrow F_K([i]_m \| M[i])$

$Tag \leftarrow y_1 \oplus \ldots \oplus y_s$

Return Tag

Secure?

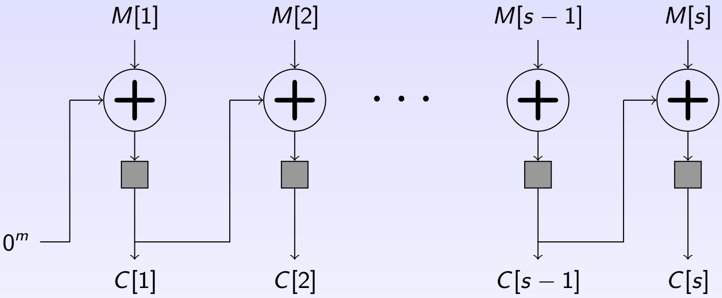
CBC MAC

CBC MAC is like CBC encryption but with $IV = 0^n$ and the tag is the last ciphertext block.

If fixed-length input, then SECURE.

Otherwise, INSECURE!

CBC MAC: pictorially



- The tagging algorithm for CBC MAC. The gray boxes denote the permutation E_K where E is the underlying block cipher and K is the shared secret key.
- It is assumed here that the message length is a *fixed* multiple of the block size.

ECBC MAC

- CBC MAC is vulnerable to a length-extension attack.
- How to fix CBC MAC for variable-length messages?
  - Encipher the last block of output with another key.
  - The additional encipherment prevents the length-extension attack.
  - This solution is called ECBC MAC.

Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^L$ be a block cipher.

Theorem: ECBC MAC security

For every efficient q-query PRF adversary A attacking ECBC, there exists an efficient adversary B attacking F such that

$$\mathbf{Adv}_{ECBC}^{\text{uf-cma}}(A) \le \mathbf{Adv}_{F}^{\text{prp-cpa}}(B) + \frac{2q^2}{2^n} .$$

ECBC MAC security: interpretation

$$\mathbf{Adv}^{\text{uf-cma}}_{ECBC}(A) \leq \mathbf{Adv}^{\text{prp-cpa}}_{F}(B) + \frac{2q^2}{2^n} \ .$$

Suppose q is the number of message MACed with a secret key.

- ECBC MAC is secure as long as $q << 2^{n/2}$
- Suppose we want $\mathbf{Adv}^{\text{uf-cma}}_{ECBC}(A) \leq 1/2^{32}$
  - Then, $q^2/2^n < 1/2^{32}$
  - With AES, $n = 128$. So, $q \leq 2^{48}$. Acceptable.
  - With 3DES, $n = 64$. So, $q \leq 2^{16}$. Unacceptable.
- Once $q = 2^{n/2}$, we can attack ECBC using birthday attack + length-extension.
  - Find $M_1 \neq M_2$ that get MACed to the same tag t, then query for tag of $M_1 \| w$, and forge with $(M_2 \| w, t)$.

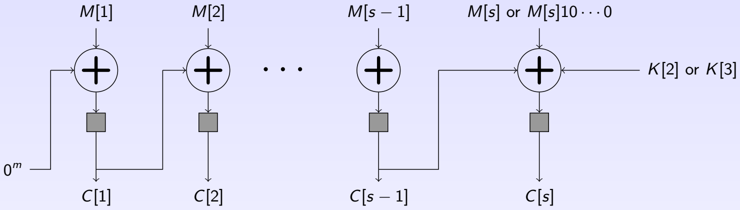
Padding input messages

What if input messages aren't of length multiple of n?

- pad with 0's?
- pad with 100...0?
- pad with 100...0 with dummy block?
- CMAC (like CBC MAC) but
  - pad with 100...0 if necessary and don't pad if unnecessary
  - then xor the last block with another key K_1 if there's a pad or with K_2 if there's no pad.
  - No dummy block.
  - No additional encipherment like ECBC MAC.

XCBC MAC

XCBC MAC is like CBC MAC but uses two extra keys so that it can deal with variable-length input messages.



- The tagging algorithm for CBC MAC. The gray boxes denote the permutation E_K where E is the underlying block cipher and K is the shared secret key.
- If the last block $M[s]$ is too short, it is padded by $10\cdots0$ until its length equals the block size.
- If the last block has not been padded, then $K[2]$ is additionally exclusive-ored with the message. Otherwise, $K[3]$ is used instead.

CMAC (aka OMAC for One-key MAC)

- CMAC is XCBC MAC but with $K[2]$ and $K[3]$ derived from $E_K(0^n)$.
- CMAC is recommended by NIST.

Example attacks against MACs

Example: Attack against π_1

Adverary $A_1^{MAC_K(\cdot), VF_K(\cdot,\cdot)}$

$M \leftarrow 0^n || 0^n$

$Tag \leftarrow 0^L$

$d \leftarrow VF_K(M, Tag)$

$$\mathbf{Adv}_{\pi_1}^{\text{uf-cma}}(A_1) = 1$$

Resources: $t = O(n + L), q_s = 0, q_v = 1, \mu_s = 0, \mu_v = 2n$

Example: Attack against π_2

idea:

To forge on $M = b_1 b_2$, we want to know what

$$F_K([1]_m b_1) \oplus F_K([2]_m b_2)$$

looks like. So we ask a query $b_1 a_2$ & $a_1 b_2$ then *XOR* out the extras, i.e. $F_K([1]_m a_1)$ & $F_K([2]_m a_2)$.

$$\begin{array}{lll} \text{We ask for} & Tag_1 \text{ of} & M_1 = a_1 a_2 \\ & Tag_2 \text{ of} & M_2 = a_1 b_2 \\ & Tag_3 \text{ of} & M_3 = b_1 a_2 \end{array}$$

Forge on $M = b_1 b_2$ with tag value $tag_1 \oplus tag_2 \oplus tag3$.

$$\begin{array}{lll} M_1 : & tag_1 & = F_K([1]_m a_1) \oplus F_K([2]_m a_2) \\ M_2 : & tag_2 & = F_K([1]_m a_1) \oplus F_K([2]_m b_2) \\ M_3 : & tag_3 & = F_K([1]_m b_1) \oplus F_K([2]_m a_2) \\ M : & tag & = F_K([1]_m b_1) \oplus F_K([2]_m b_2) \end{array}$$

Example: Attack against π_2

Adverary $A_2^{MAC_K(\cdot),VF_K(\cdot,\cdot)}$

$$
\begin{array}{l}
\text{Let } a_1, b_1 \text{ be distinct } l-m\text{-bit strings} \\
\text{Let } a_2, b_2 \text{ be distinct } l-m\text{-bit strings} \\
Tag_1 \leftarrow MAC_K(a_1 a_2) \\
Tag_2 \leftarrow MAC_K(a_1 b_2) \\
Tag_3 \leftarrow MAC_K(b_1 a_2) \\
Tag \leftarrow Tag_1 \oplus Tag_2 \oplus Tag_3 \\
d \leftarrow VF_K(b_1 b_2, Tag)
\end{array}
$$

$$\mathbf{Adv}_{\pi_1}^{\text{uf-cma}}(A_1) = 1$$

Resources: ??

Attack against variable-length input version of CBC MAC

$$\mathbf{Adv}_{\pi}^{\mathrm{uf\text{-}cma}}(A) = 1$$

Resources: $q_v = 1, q_s = 1, \mu_v = 2n, \mu_s = n, t = O(n)$

This attack doesn't work if input length is fixed.

In fact, CBC MAC with fixed-length inputs is secure.

Making π_2 secure

π_2 with i being a counter is secure.

MAC algorithm is **stateful**: ctr starts at zero and gets incremented across messages.

stateful π_2

Let $F : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^L$ be a family of functions.

$MAC_K(M)$:

$\quad$ static $ctr \leftarrow 0$

$\quad$ $t \leftarrow n - m$

$\quad$ If $(|M| \bmod t \neq 0$ or $|M| = 0$ or $ctr + \frac{|M|}{t} \geq 2^m)$ then return $\perp$

$\quad$ Break M into t-bits blocks $M = M[1] || \cdots || M[s]$

$\quad$ For $i = 1, \cdots, s$ do $y_i \leftarrow F_K([ctr + i]_m || M[i])$

$\quad$ $Tag \leftarrow y_1 \oplus \cdots \oplus y_s$

$\quad$ $ctr \leftarrow ctr + s$

$\quad$ Return Tag

Stateful version of π_2 is a secure MAC.

Result (informal)

π is a secure *MAC* assuming that F is a PRF.

To prove this, we need to show that, $\forall A$ attacking π, we can construct B attacking F.

Idea

- B runs A using g in place of F_K to compute *Tag*
- If A ever forges successfully, B outputs 1. Otherwise, it outputs 0.

MAC based on hash function

HMAC

$MAC_K(M) = H(K \oplus \mathrm{opad} \| H(K \oplus \mathrm{ipad} \| M))$