Message integrity

Message Auth. Codes

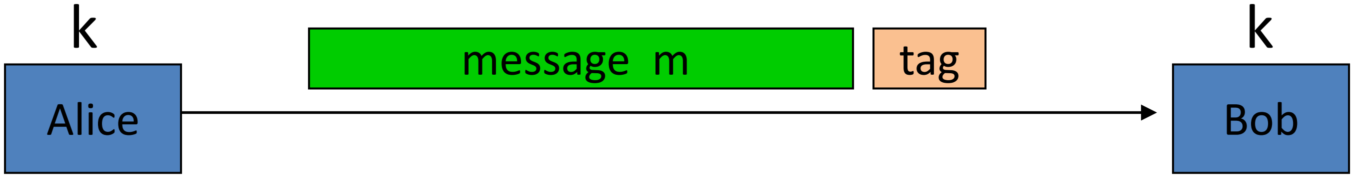
Message Integrity

Goal: **integrity**, no confidentiality.

Examples:

- Protecting public binaries on disk.
- Protecting banner ads on web pages.

Message integrity: MACs



Generate tag:

tag ← S(k, m)

Verify tag:

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

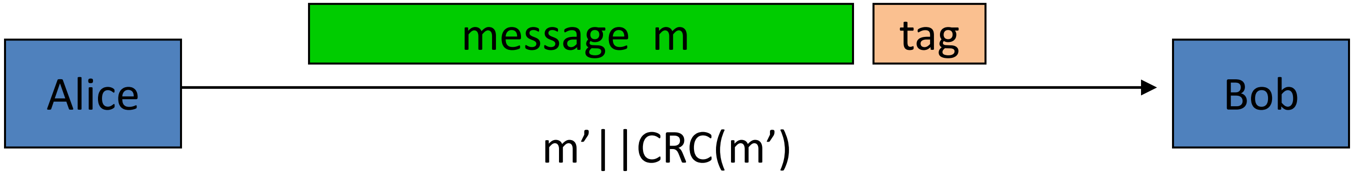
Def: **MAC** I = (S,V) defined over (K,M,T) is a pair of algs:

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

$$\forall k \in \mathcal{K}, \forall m \in \mathcal{M}$$

$$V\big(k, m, S(k,m)\big) = \text{"}yes\text{"}$$

Integrity requires a secret key



Generate tag:

tag $\leftarrow$ CRC(m)

Verify tag:

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

- Attacker can easily modify message m and re-compute CRC.
- CRC designed to detect **random**, not malicious errors.

Secure MACs

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

- for $m_1,m_2,\ldots,m_q$ 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).

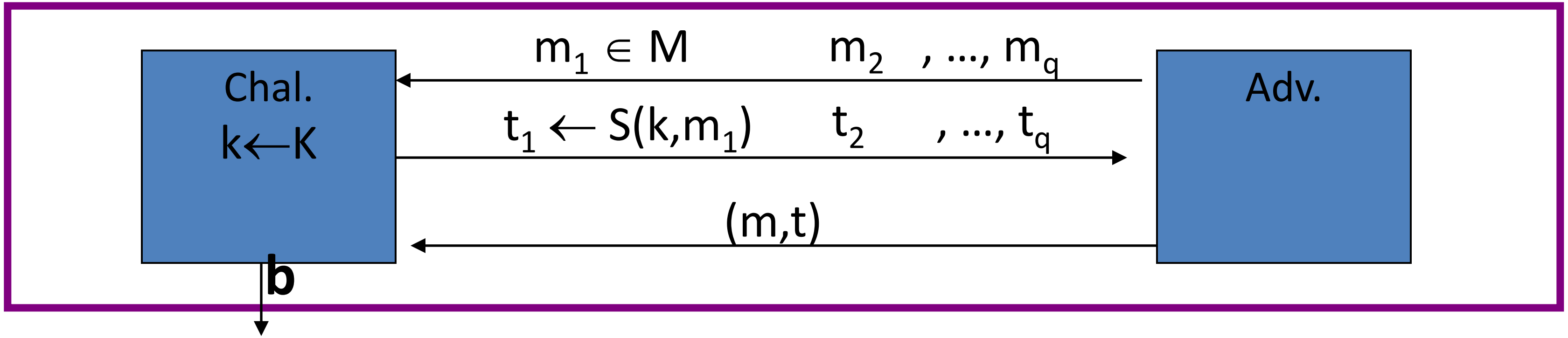
$$(m,t) \notin \{ (m_1,t_1) , \ldots , (m_q,t_q) \}$$

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

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

Secure MACs

- 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[Chal. outputs 1] is "negligible."

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

Suppose an attacker is able to find $m_0 \neq m_1$ such that

$S(k, m_0) = S(k, m_1)$ for ½ of the keys k in K

Can this MAC be secure?

Yes, the attacker cannot generate a valid tag for m_0 or m_1

No, this MAC can be broken using a chosen msg attack

It depends on the details of the MAC

(m_0, t) → output (m_1, t) valid existing Forgery

$$\boldsymbol{Adv[A, I] = 1/2}$$

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

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

$T = \{0,1\}^5$

tag len = 64, 96, 128 bits

TLS

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

Output: chose rand. $T \leftarrow \{0,1\}^5$

output (0,t)

$Adv[A, I] = 1/32$

Example: protecting system files

Suppose at install time the system computes:

filename F_1	filename F_2	$\cdots$	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

Message Integrity

MACs based on PRFs

Review: Secure MACs

MAC: signing alg. **$S(k,m) \longrightarrow t$** and verification alg. **$V(k,m,t) \longrightarrow 0,1$**

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

- for $m_1, m_2, \ldots, m_q$ 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).

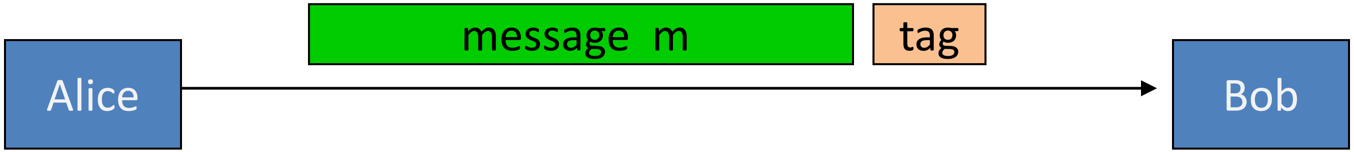
$$(m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \}$$

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

Secure PRF $\Rightarrow$ Secure MAC

For a PRF $\mathbf{F: K \times X \longrightarrow Y}$ 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.



tag $\leftarrow$ F(k,m)

accept msg if

tag = F(k,m)

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

$Adv[A, I_F] = 1/2^{10} = 1/1024$ → non-negligible

Security

<u>Thm</u>: If **F: K×X⟶Y** is a secure PRF and $1/|Y|$ is negligible (i.e. $|Y|$ is large) then I_F is a secure MAC.

In particular, for every eff. MAC adversary A attacking I_F there exists an eff. PRF adversary B attacking F s.t.:

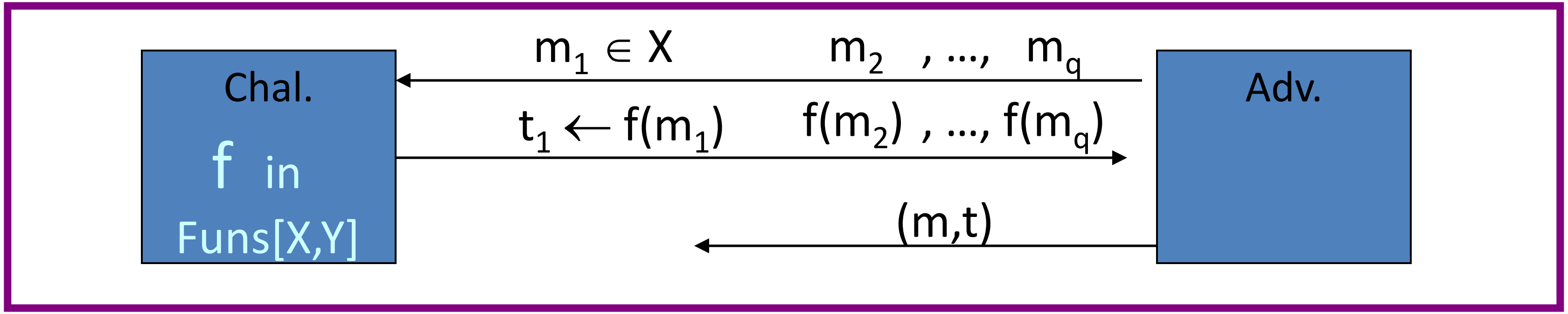
$$\mathrm{Adv}_{\mathrm{MAC}}[A, I_F] \leq \mathrm{Adv}_{\mathrm{PRF}}[B, F] + \mathbf{1/|Y|}$$

$\Rightarrow$ I_F is secure as long as $|Y|$ is large, say $|Y| = 2^{80}$.

Proof Sketch

Suppose $f: X \longrightarrow Y$ is a truly random function

Then MAC adversary A must win the following game:



$\Rightarrow$ f(m) is independent of $f(m_1), \ldots, f(m_q)$

A wins if $t = f(m)$ and $m \notin \{ m_1, \ldots, m_q \}$

The best the adversary can do is only guessing

$\Rightarrow$ $\Pr[A \text{ wins}] = 1/|Y|$ same must hold for the secure PRF: F(k,x)

Examples

- AES (is a secure PRF): A MAC for 16-byte/128-bit messages.

- Main question: How to convert Small-MAC into a Big-MAC ?

- Two main constructions used in practice:
  - **CBC-MAC** (banking – ANSI X9.9, X9.19, FIPS 186-3)
  - **HMAC** (Internet protocols: SSL, IPsec, SSH, …)

- Both convert a small-PRF into a big-PRF.

Truncating MACs based on PRFs

Easy lemma: suppose $F: K \times X \longrightarrow \{0,1\}^n$ is a secure PRF.

Then so is $F_t(k,m) = \underbrace{F(k,m)[1\ldots t]}_{\textit{first } t-\textit{bit of output}}$ for all $1 \le t \le n$

$\Rightarrow$ if (S,V) is a MAC is based on a secure PRF outputting n-bit tags

the truncated MAC outputting w bits is secure

… as long as $1/2^w$ is still negligible (say $w \ge 64$)

Message Integrity

CBC-MAC and NMAC

MACs and PRFs

Recall: secure PRF **F** $\Rightarrow$ secure MAC, as long as $|Y|$ is large

$$S(k, m) = F(k, m)$$

Our goal:

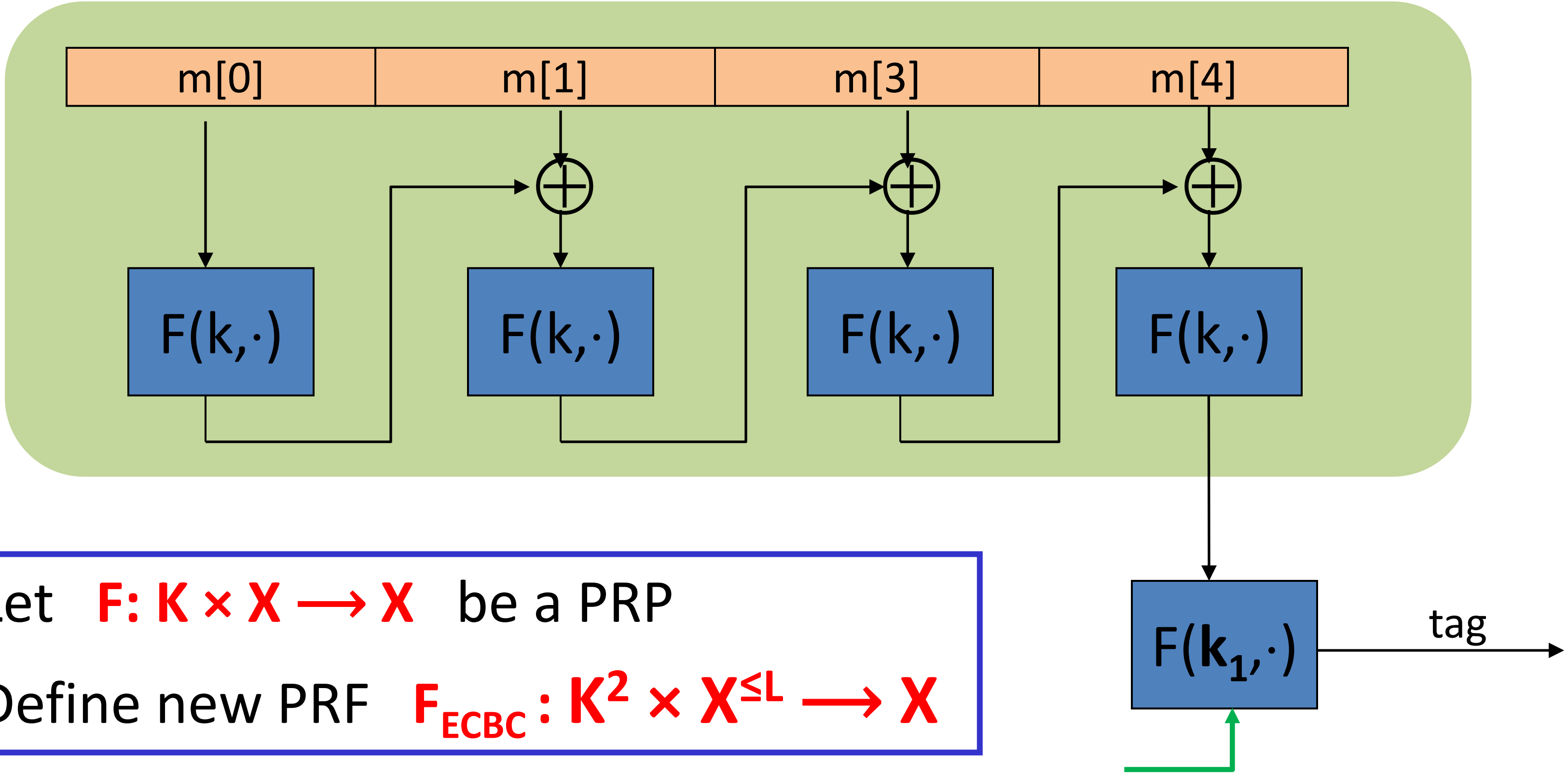
given a PRF for short messages (AES)

construct a PRF for long messages

From here on let $X = \{0,1\}^n$ (e.g. n=128, in case of AES)

Construction 1: encrypted CBC-MAC (ECBC)

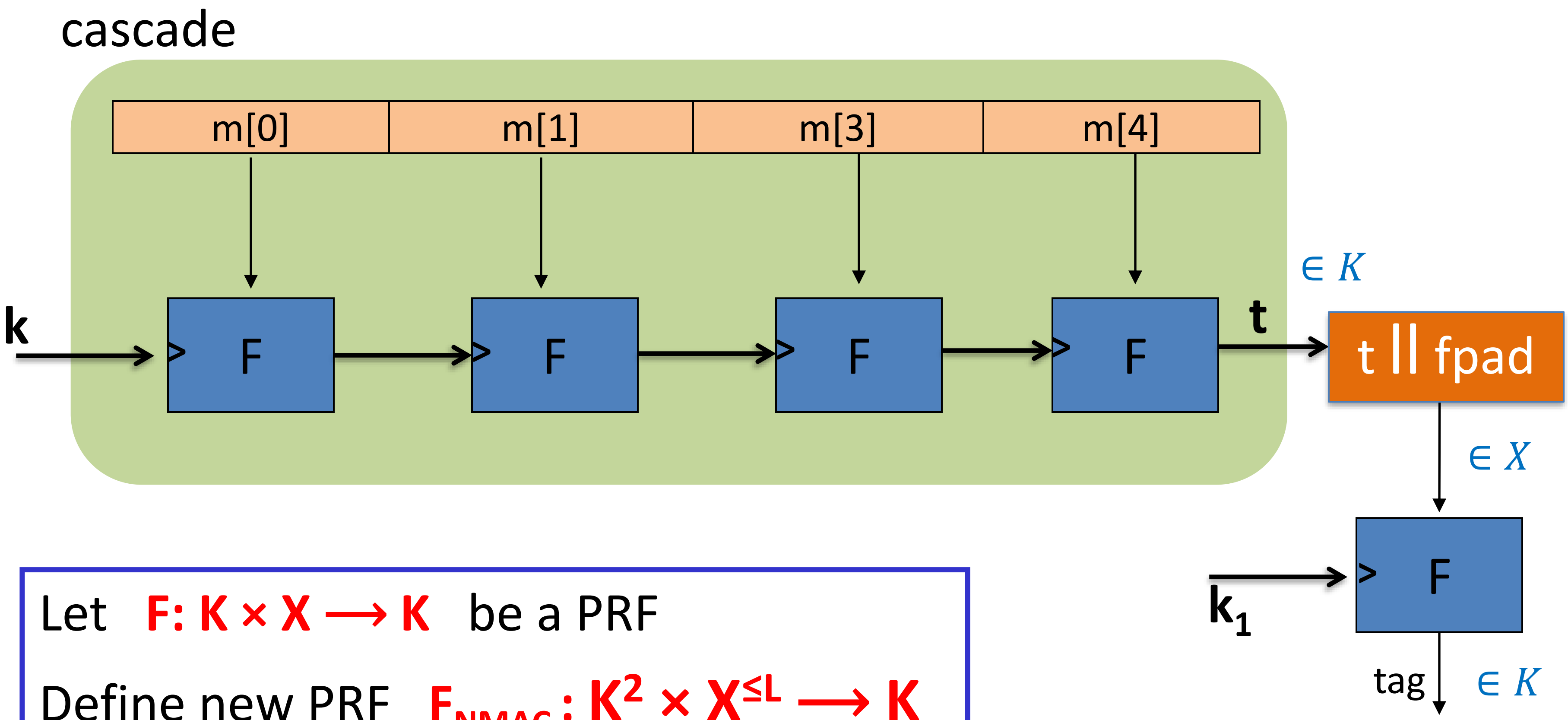
raw CBC



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

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

Construction 2: NMAC (nested MAC)



Let $\mathbf{F: K \times X \longrightarrow K}$ be a PRF

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

Why the last encryption step in ECBC-MAC and NMAC?

NMAC: suppose we define a MAC $I = (S,V)$ where

$$S(k,m) = \text{cascade}(k, m)$$

This MAC is secure

This MAC can be forged without any chosen msg queries

This MAC can be forged with one chosen msg query

This MAC can be forged, but only with two msg queries

cascade (k, m) ==> cascade(k, m||w) for any w

Extension attack

Why the last encryption step in ECBC-MAC?

Suppose we define a MAC $I_{RAW} = (S,V)$ where

$$S(k,m) = \text{rawCBC}(k,m)$$

Then I_{RAW} is easily broken using a 1-chosen msg attack.

Adversary works as follows:

- Choose an arbitrary one-block message $m \in X$
- Request tag for m. Get $t = F(k,m)$
- Output t as MAC forgery for the 2-block message $m' = (m, t \oplus m)$

Indeed: $\text{rawCBC}\big(k, (m, t \oplus m)\big) = F\big(k, F(k,m) \oplus (t \oplus m)\big) = F\big(k, t \oplus (t \oplus m)\big) = t$

The security bounds are tight: an attack

After signing $|X|^{1/2}$ messages with ECBC-MAC or
$|K|^{1/2}$ messages with NMAC
the MACs become insecure

Suppose the underlying PRF F is a PRP (e.g. AES)

- Then both PRFs (ECBC and NMAC) have the following extension property:

$$\forall x,y,w: \quad F_{BIG}(k, x) = F_{BIG}(k, y) \quad \Rightarrow \quad F_{BIG}(k, \mathbf{x \| w}) = F_{BIG}(k, \mathbf{y \| w})$$

The security bounds are tight: an attack

Let F_{BIG}: $\mathbf{K \times X \longrightarrow Y}$ be a PRF that has the extension property

$$F_{BIG}(k, x) = F_{BIG}(k, y) \quad \Rightarrow \quad F_{BIG}(k, \mathbf{x \| w}) = F_{BIG}(k, \mathbf{y \| w})$$

Generic attack on the derived MAC:

step 1: issue $|Y|^{1/2}$ message queries for rand. messages in X.

obtain (m_i, t_i) for $i = 1, \ldots, |Y|^{1/2}$

step 2: find a collision $t_u = t_v$ for $u \neq v$ (one exists w.h.p by b-day paradox)

step 3: choose some w and query for $t := F_{BIG}(k, \mathbf{m_u \| w})$

step 4: output forgery $(\mathbf{m_v \| w}, \mathbf{t})$. Indeed $t := F_{BIG}(k, \mathbf{m_v \| w})$

Comparison

ECBC-MAC is commonly used as an AES-based MAC

- CCM encryption mode (used in 802.11i)
- NIST standard called CMAC

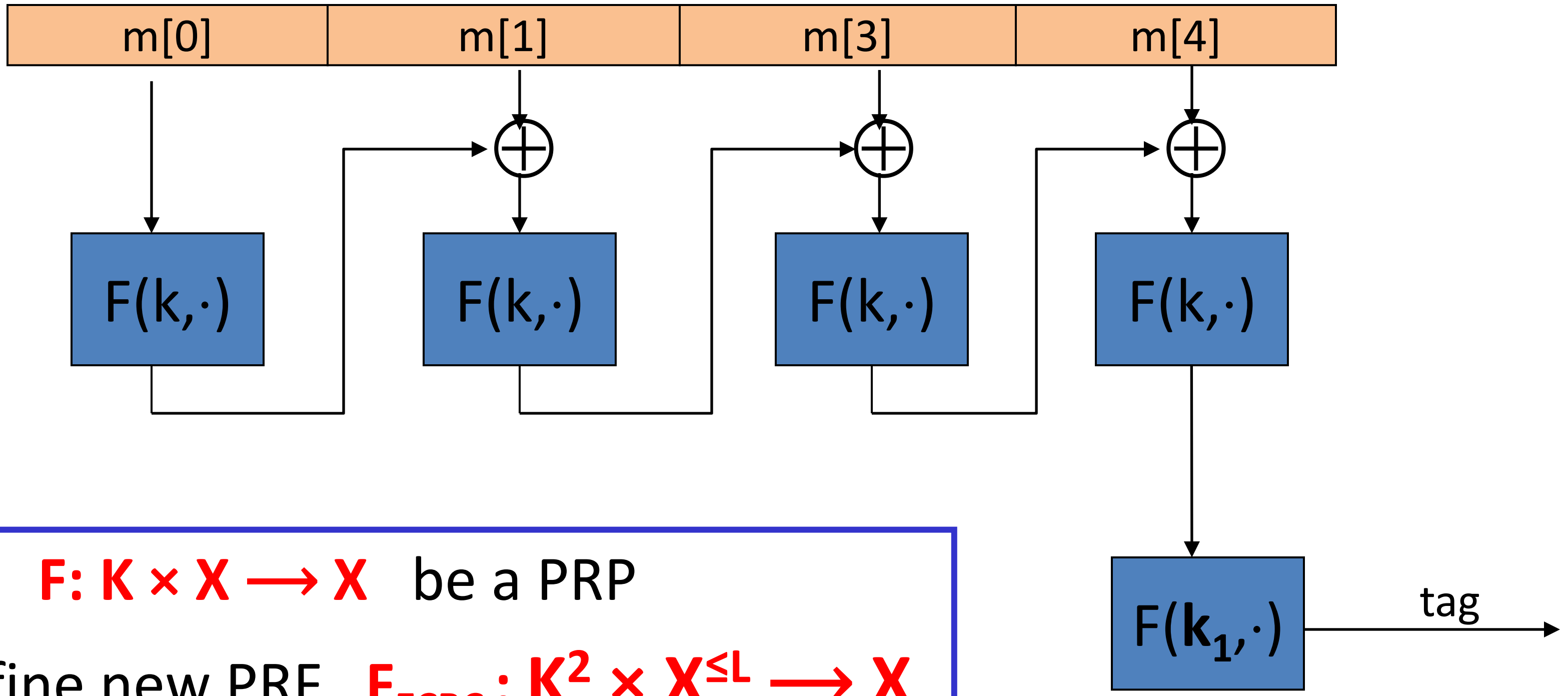
NMAC not usually used with AES or 3DES

- Main reason: need to change AES key on every block
  requires re-computing AES key expansion
- But NMAC is the basis for a popular MAC called HMAC (next)

Message Integrity

MAC padding

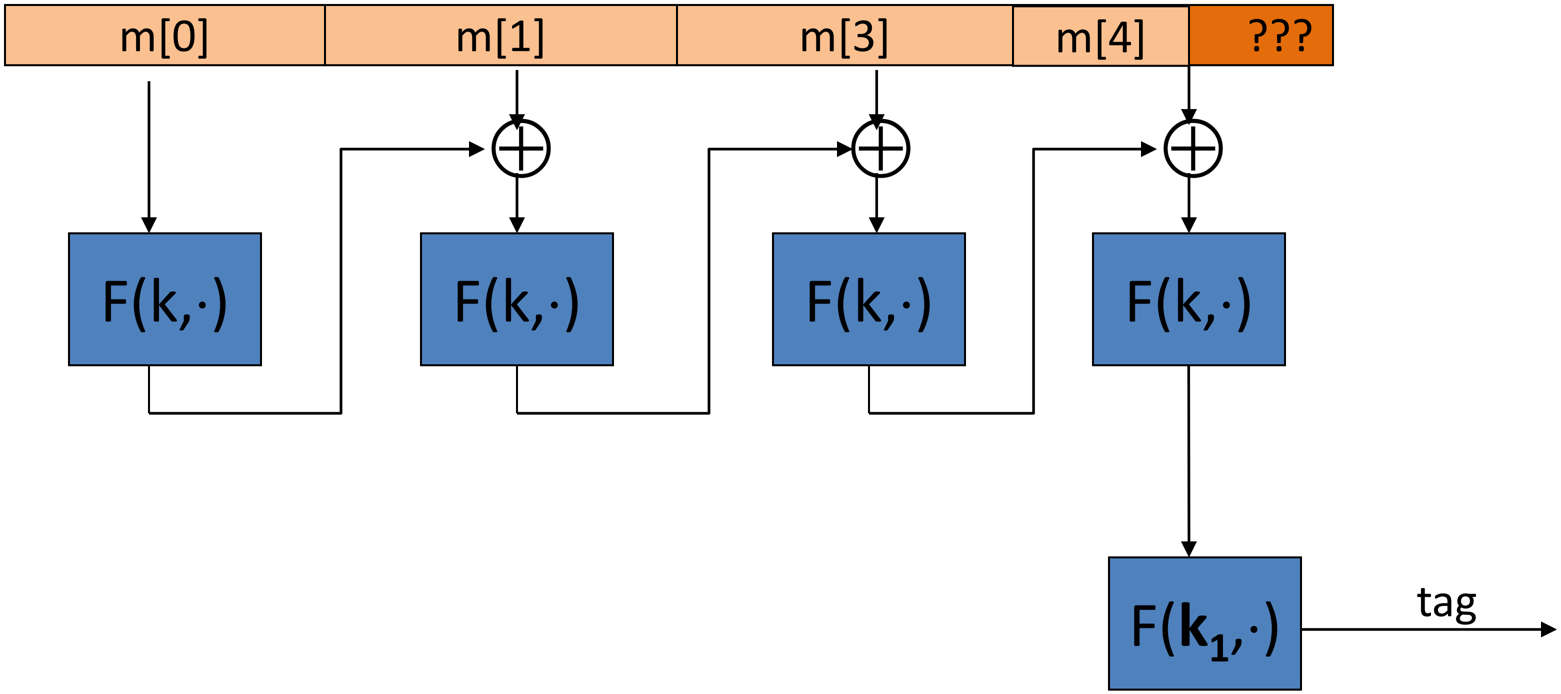
Recall: ECBC-MAC



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

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

What if msg. len. is not multiple of block-size?



CBC MAC padding

Bad idea: pad m with 0's

Is the resulting MAC secure?

Yes, the MAC is secure

It depends on the underlying MAC

No, given tag on msg **m** attacker obtains tag on **m||0**

Problem: pad(m) = pad(m||0)

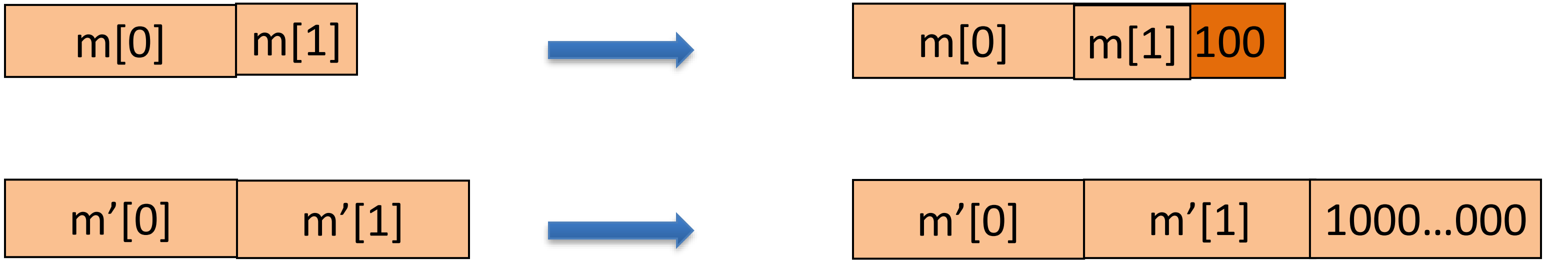
CBC MAC padding

For security, padding must be invertible !

$$m_0 \neq m_1 \quad \Rightarrow \quad pad(m_0) \neq pad(m_1)$$

<u>ISO</u>: pad with "1000…00". Add new dummy block if needed.

- The "1" indicates beginning of pad.

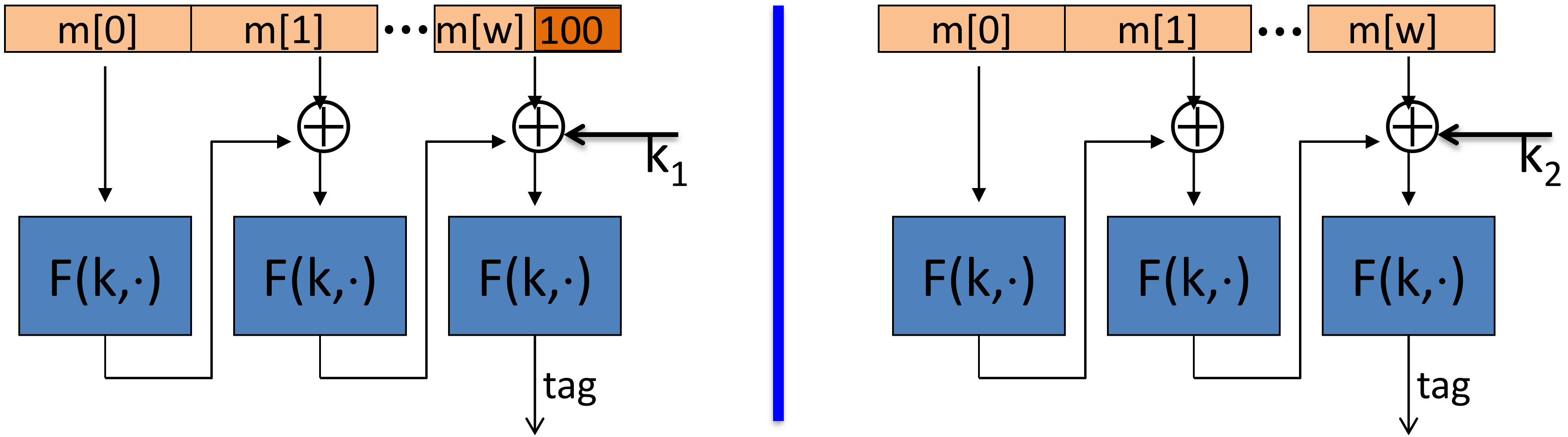


CMAC (NIST standard)

Variant of CBC-MAC where key = (k, k_1, k_2)

(K_1, k_2) derived from K

- No final encryption step (extension attack thwarted by last keyed xor)
- No dummy block (ambiguity resolved by use of k_1 or k_2)



Message Integrity

PMAC and Carter-Wegman MAC

- ECBC and NMAC are sequential.

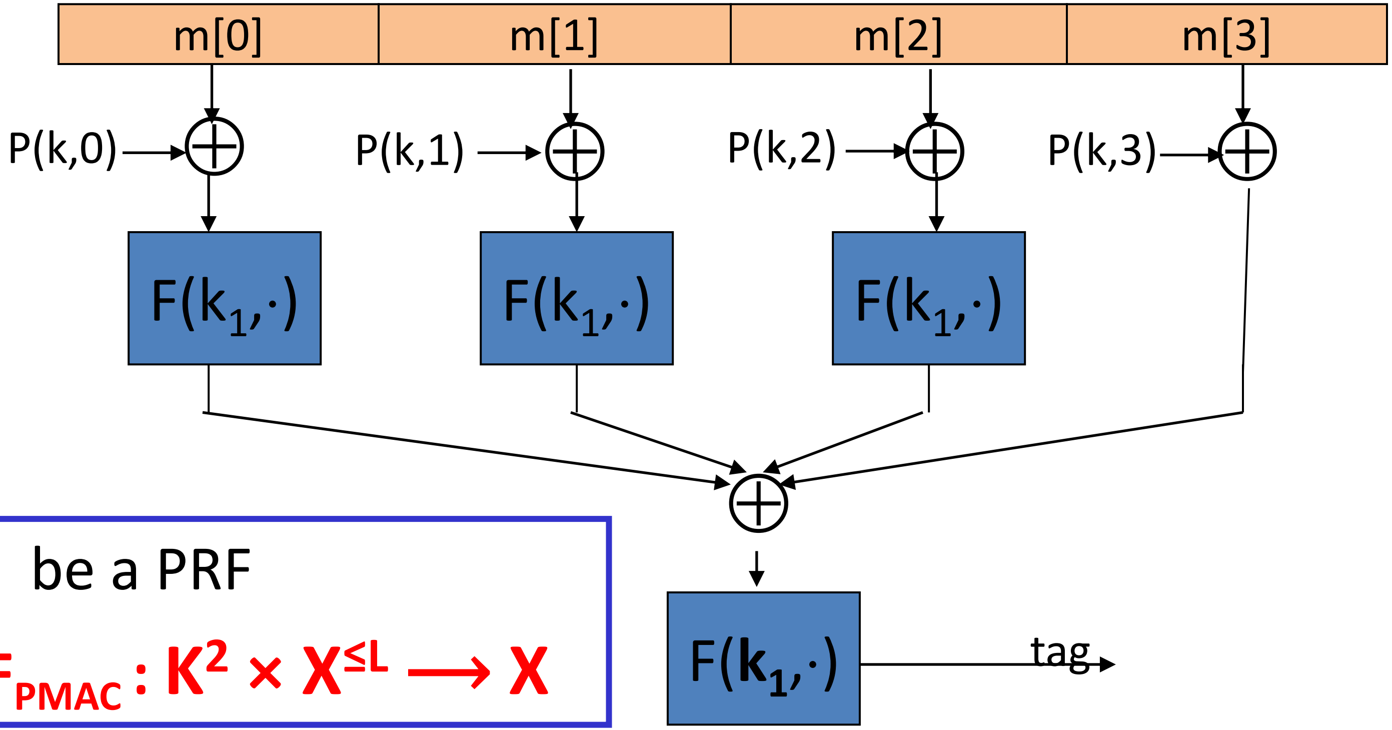
- Can we build a parallel MAC from a small PRF ??

Construction 3: PMAC – parallel MAC

P(k, i): an easy to compute function

key = (k, k_1)

Padding similar to CMAC



Let $\mathbf{F}: \mathbf{K} \times \mathbf{X} \longrightarrow \mathbf{X}$ be a PRF

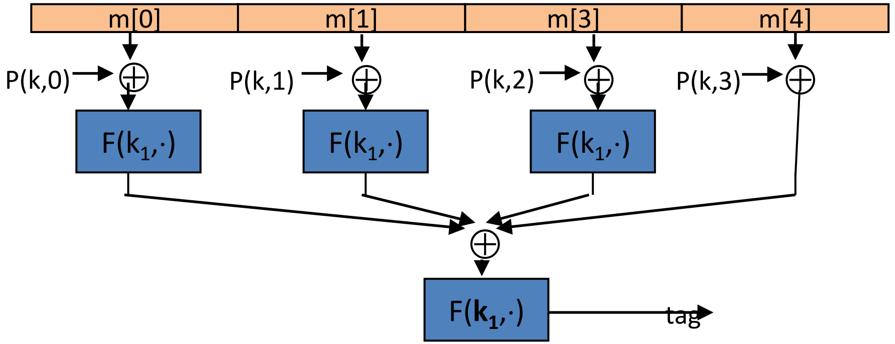
Define new PRF $\mathbf{F}_{\mathbf{PMAC}} : \mathbf{K}^2 \times \mathbf{X}^{\leq L} \longrightarrow \mathbf{X}$

PMAC is incremental

Suppose F is a PRP.

When $m[1] \longrightarrow m'[1]$

can we quickly update tag?



no, it can't be done

do $F^{-1}(k_1,tag) \oplus F\big(k_1, m'[1] \oplus P(k,1)\big)$

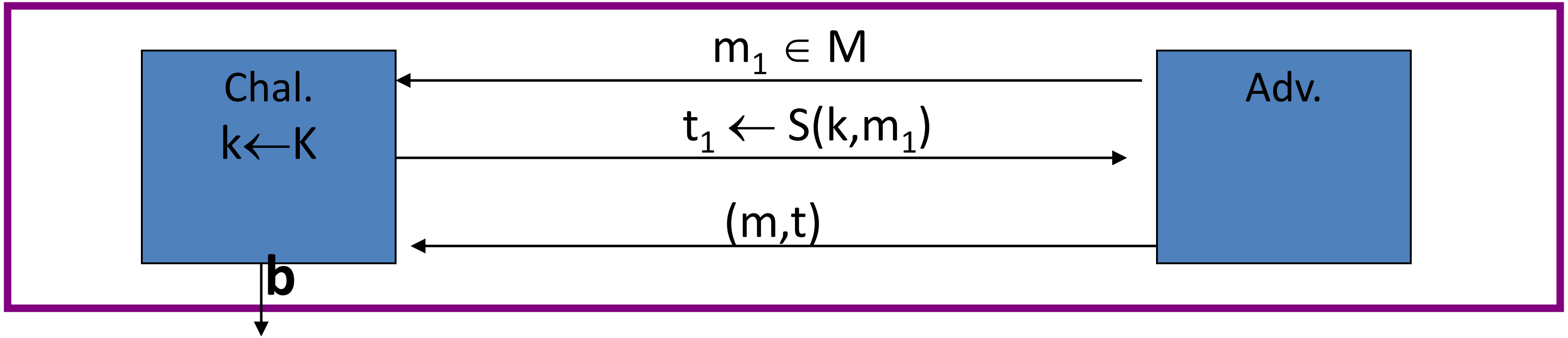
do $F^{-1}(k_1,tag) \oplus F\big(k_1, m[1] \oplus P(k,1)\big) \oplus F\big(k_1, m'[1] \oplus P(k,1)\big)$

do $tag \oplus F\big(k_1, m[1] \oplus P(k,1)\big) \oplus F\big(k_1, m'[1] \oplus P(k,1)\big)$

Then apply $F(k_1, \cdot)$

One time MAC (analog of one time pad)

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



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

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

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

One-time MAC: an example

Can be secure against **all** adversaries and faster than PRF-based MACs

Let q be a large prime (e.g. $\mathbf{q = 2^{128}+51}$)

key = (a, b) ∈ $\{1,\ldots,q\}^2$ (two random ints. in [1,q])

msg = (m[1], …, m[L]) where each block is 128 bit int.

$$\mathbf{S(\text{ key, msg }) = P_{msg}(a) + b \quad (\text{mod } q)}$$

where $P_{msg}(x) = x^{L+1} + m[L]\cdot x^{L} + \ldots + m[1]\cdot x$ is a poly. of deg L+1

We show: given **S(key, msg$_1$)** adv. has no info about **S(key, msg$_2$)**

One-time MAC $\Rightarrow$ Many-time MAC

Let (S,V) be a secure one-time MAC over $(K_I, M, \{0,1\}^n)$.

Let $F: K_F \times \{0,1\}^n \longrightarrow \{0,1\}^n$ be a secure PRF.

Carter-Wegman MAC: $CW\big((k_1,k_2), m\big) = \big(r,\ \underbrace{F(k_1,r)}_{\text{slow but short input}} \oplus \underbrace{S(k_2,m)}_{\text{fast long input}}\big)$

For random $r \longleftarrow \{0,1\}^n$. If you compute tag for same msg twice, each time you choose different r and get different tags (for the same msg).

Thm: If (S,V) is a secure **one-time** MAC and F a secure PRF then CW is a (many-time) secure MAC outputting tags in $\{0,1\}^{2n}$.

Note: PRF is only used for short messages and yet we get a MAC for long messages

$$CW((k_1,k_2), m) = (r, \underbrace{F(k_1,r) \oplus S(k_2,m)}_{t})$$

How would you verify a CW tag **(r, t)** on message **m** ?

Recall that $V(k_2,m,.)$ is the verification alg. for the one time MAC.

Run $V(k_2, m, F(k_1, t) \oplus r)$

Run $V(k_2, m, r)$

Run $V(k_2, m, t)$

✓ Run $V(k_2, m, \underbrace{F(k_1, r) \oplus t}_{S(k_2,m)})$

Construction 4:   HMAC   (Hash-MAC)

Most widely used MAC on the Internet.

… but, we first we need to discuss hash function.