Message Integrity

This slide is made based the online course of Cryptography by Dan Boneh

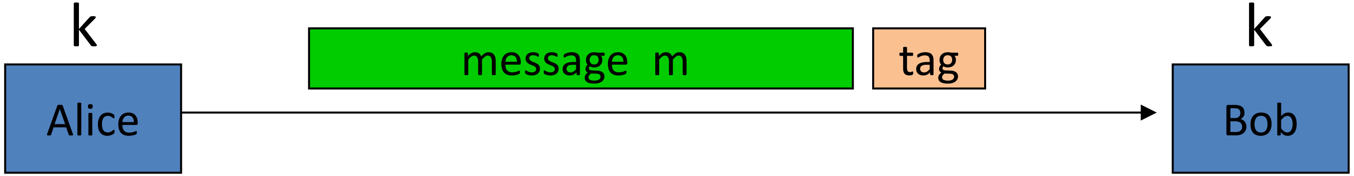
Message Integrity

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:

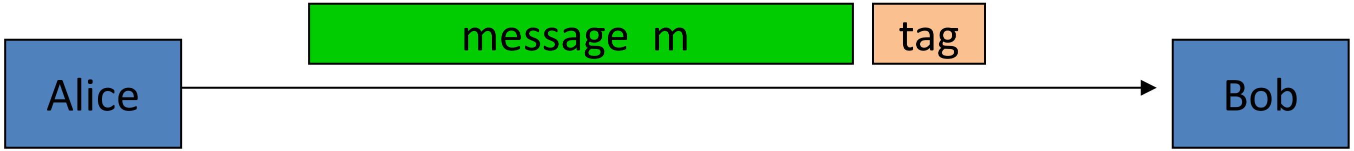
V(k, m, tag) $\stackrel{?}{=}$ `yes'

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'

Integrity requires a secret key



Generate tag:

tag ← 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).

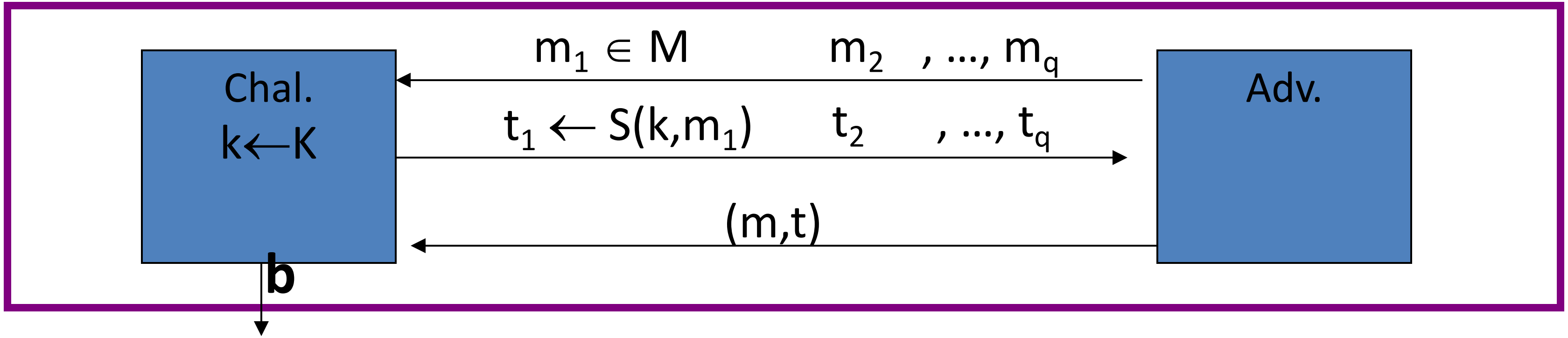
$$(m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \}$$

$\Rightarrow$ 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:

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

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

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

$$S(k, m_0) = S(k, m_1) \quad \text{for } ½ \text{ of the keys } k \text{ in } K$$

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

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
-

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

End of Segment

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).

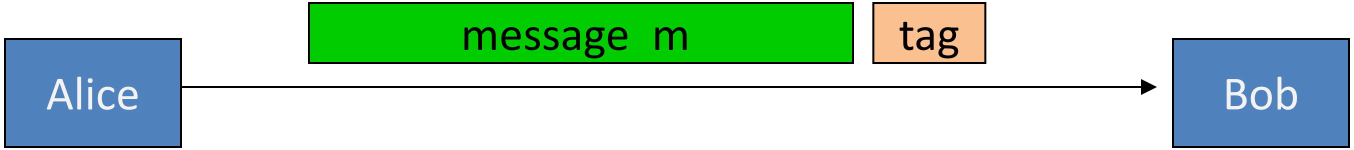
$$(m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \}$$

$\Rightarrow$ 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
- ○

$\mathrm{Adv}[A, I_F] = 1/1024$

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.:

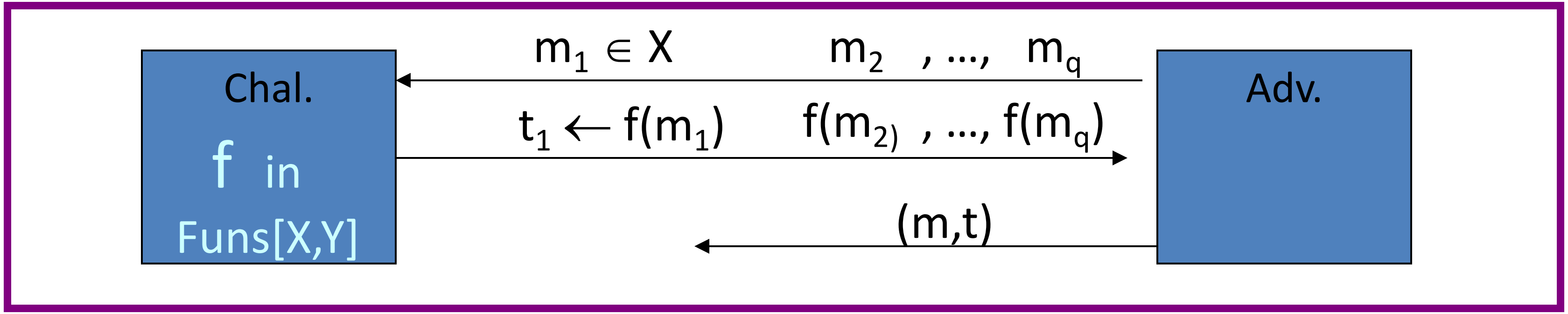
$$\mathrm{Adv}_{\mathrm{MAC}}[A, I_F] \leq \mathrm{Adv}_{\mathrm{PRF}}[B, F] + 1/|Y|$$

$\Rightarrow$ 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:



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

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

Dan Boneh

Examples

- AES: a MAC for 16-byte 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 $F_t(k,m) = F(k,m)[1...t]$ for all $1 \le t \le n$

First t-bit of output

$\Rightarrow$ 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 $w \ge 64$)

End of Segment

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:

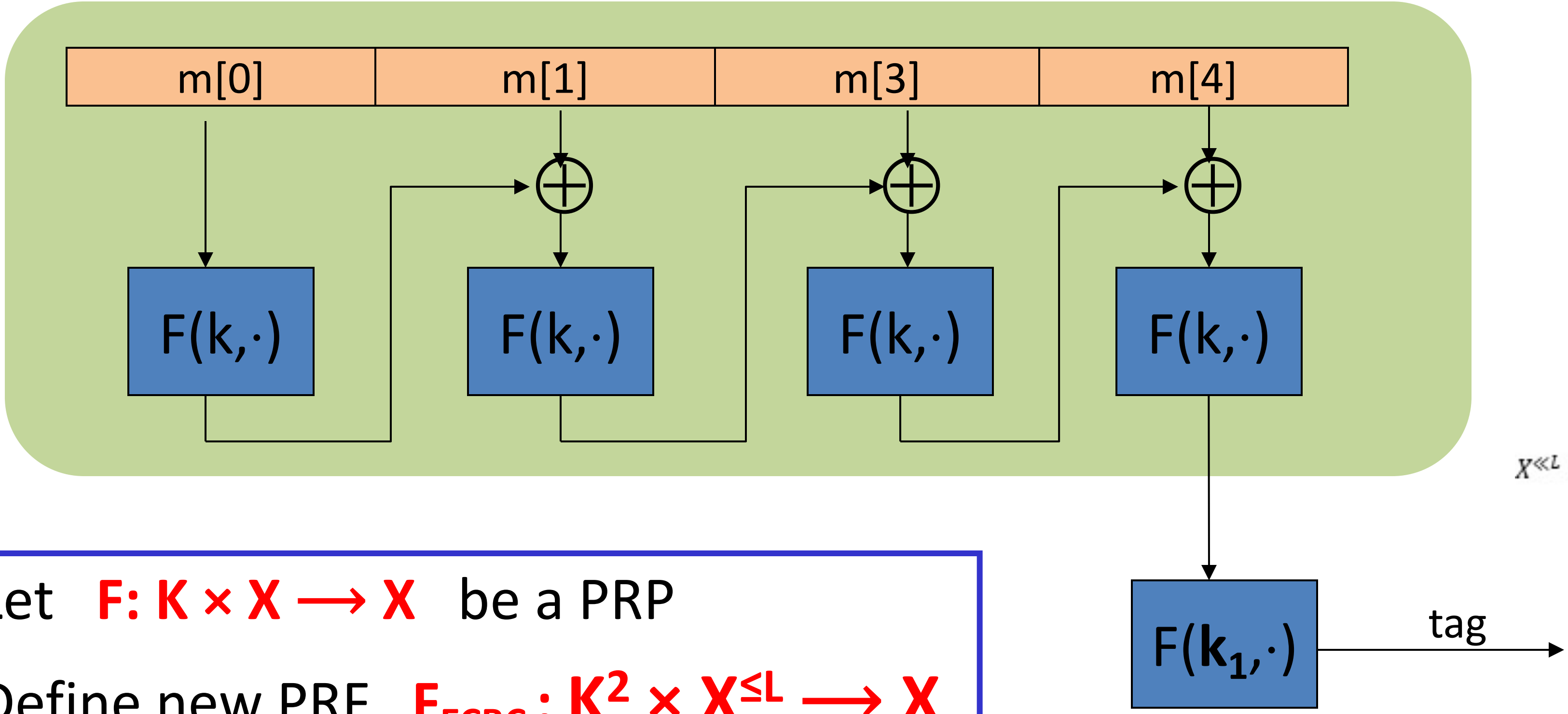
given a PRF for short messages (AES)

construct a PRF for long messages

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

Construction 1: encrypted CBC-MAC

raw CBC

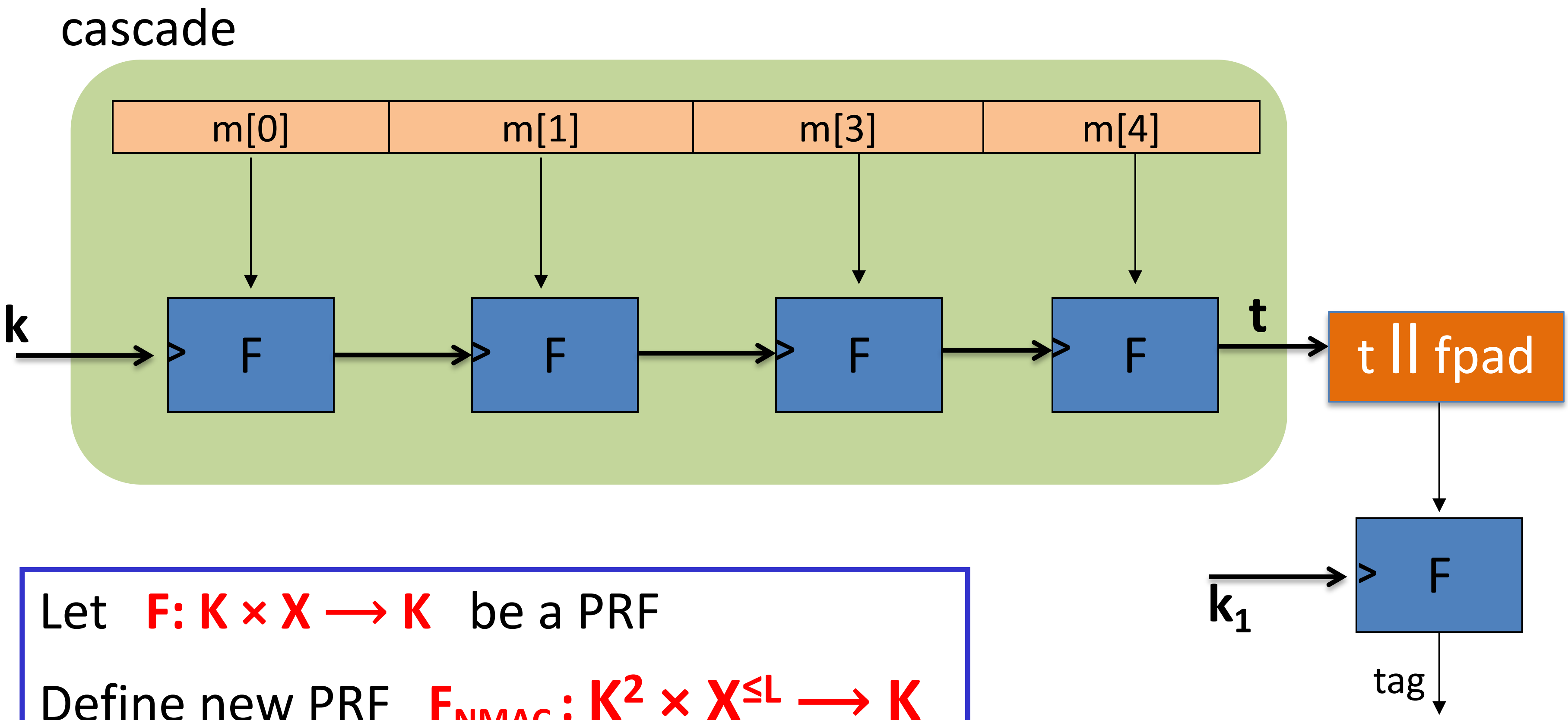


$$X^{\leq L} = \bigcup_{i=1}^{L} X^i$$

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

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

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

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 $(m, t \oplus m)$

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

Collision resistance

This slide is made based the online course of Cryptography by Dan Boneh

Collision Resistance

Let $H: M \to T$ be a hash function ($|M| >> |T|$)

A **collision** for H is a pair $m_0, m_1 \in M$ such that:

$$H(m_0) = H(m_1) \quad \text{and} \quad m_0 \neq m_1$$

A function H is **collision resistant** if for all (explicit) "eff" algs. A:

$$\mathbf{Adv_{CR}[A,H] = Pr[\ A\ outputs\ collision\ for\ H]}$$

is "neg".

Example: SHA-256 (outputs 256 bits)

MACs from Collision Resistance

Let I = (S,V) be a MAC for short messages over (K,M,T) (e.g. AES)

Let H: $M^{big} \to M$

Def: $I^{big} = (S^{big}, V^{big})$ over (K, M^{big}, T) as:

$$S^{big}(k,m) = S(k,H(m)) \quad ; \quad V^{big}(k,m,t) = V(k,H(m),t)$$

Thm: If I is a secure MAC and H is collision resistant

then I^{big} is a secure MAC.

Example: $S(k,m) = AES_{2\text{-block-cbc}}(k,\ \text{SHA-256}(m))$ is a secure MAC.

MACs from Collision Resistance

$S^{big}(k, m) = S(k, H(m))$; $V^{big}(k, m, t) = V(k, H(m), t)$

Collision resistance is necessary for security:

Suppose adversary can find $m_0 \neq m_1$ s.t. $H(m_0) = H(m_1)$.

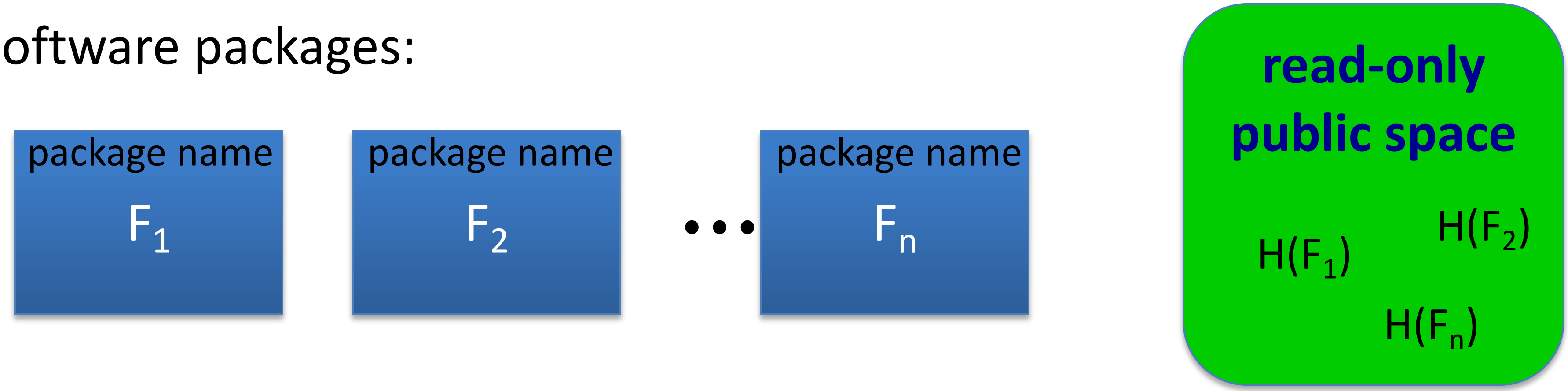
Then: S^{big} is insecure under a 1-chosen msg attack

step 1: adversary asks for $t \longleftarrow S(k, m_0)$

step 2: output (m_1, t) as forgery

Protecting file integrity using C.R. hash

Software packages:



When user downloads package, can verify that contents are valid

H collision resistant $\Rightarrow$

attacker cannot modify package without detection

no key needed (public verifiability), but requires read-only space

End of Segment

Collision resistance

Generic birthday attack

Generic attack on C.R. functions

Let $H: M \to \{0,1\}^n$ be a hash function ($|M| >> 2^n$)

Generic alg. to find a collision **in time** $\mathbf{O(2^{n/2})}$ hashes

Algorithm:

1. Choose $\mathbf{2^{n/2}}$ random messages in M: $m_1, \ldots, m_{2^{n/2}}$ (distinct w.h.p)
2. For i = 1, …, $2^{n/2}$ compute $t_i = H(m_i) \in \{0,1\}^n$
3. Look for a collision ($t_i = t_j$). If not found, got back to step 1.

How well will this work?

The birthday paradox

Let $r_1, \ldots, r_n \in \{1,\ldots,B\}$ be indep. identically distributed integers.

Thm: when **n**= 1.2 × **B**$^{1/2}$ then $\Pr[\exists i \neq j:\ r_i = r_j] \geq ½$

Proof: (for uniform indep. $r_1, \ldots, r_n$)

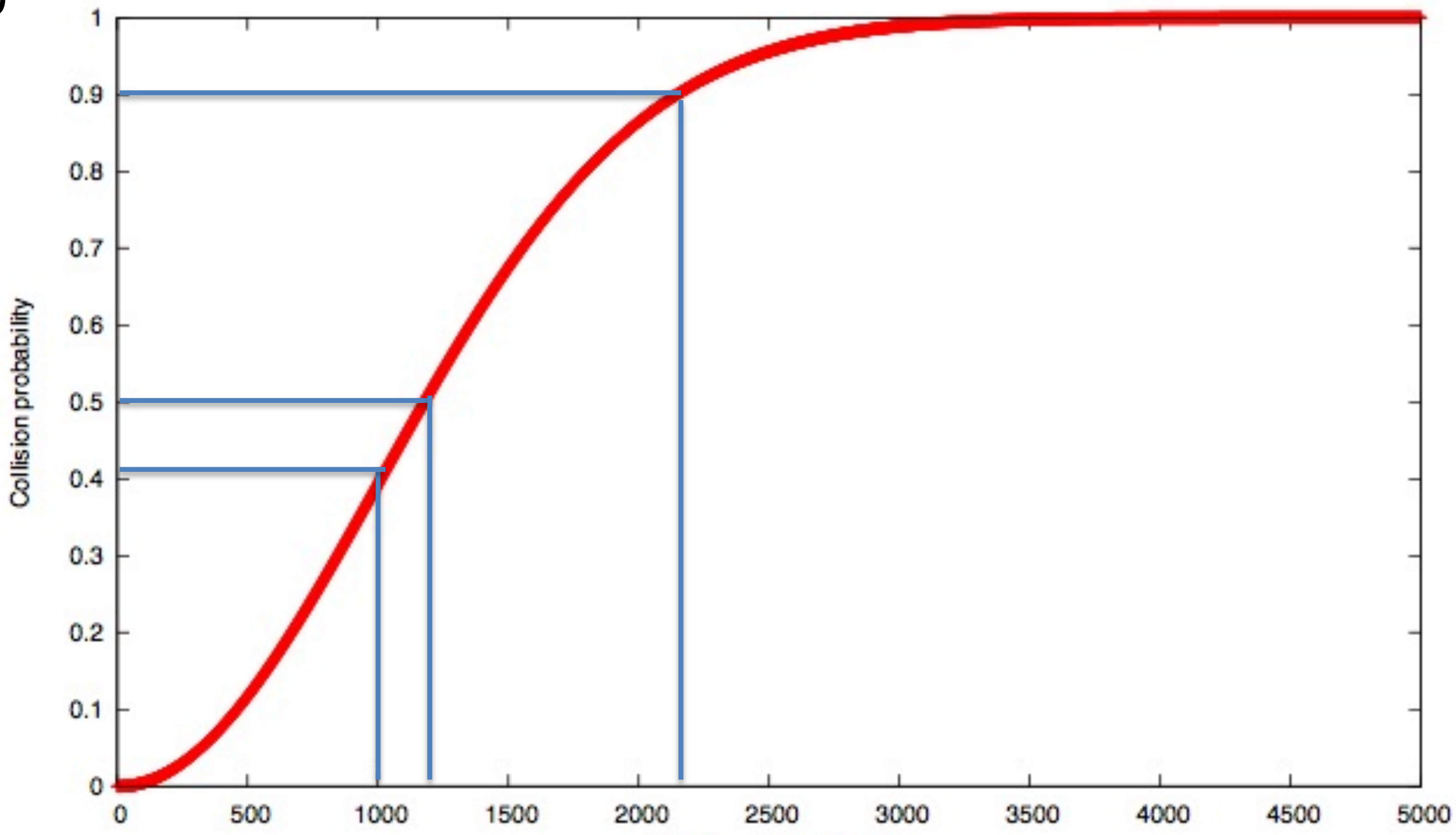
$$\Pr[\exists\, i \neq j: r_i = r_j] = 1 - \Pr[\forall i \neq j: r_i \neq r_j]$$

$$= 1 - \left(\frac{B-1}{B}\right)\left(\frac{B-2}{B}\right)\cdots\left(\frac{B-n+1}{B}\right) = 1 - \prod_{i=1}^{n-1}\left(1 - \frac{i}{B}\right) \geq 1 - \prod_{i=1}^{n-1} e^{-\frac{i}{B}}$$

$$= 1 - e^{-\frac{1}{B}\sum_{i=1}^{n-1} i} \geq 1 - e^{-\frac{n^2}{2B}} \geq 1 - e^{-0.72} = 0.53 > 1/2$$

$$1 - x \leq e^{-x},\ \frac{n^2}{2B} = 0.72$$

$B=10^6$



samples n

Generic attack

$H: M \to \{0,1\}^n$.   Collision finding algorithm:

1. Choose $\mathbf{2^{n/2}}$ random elements in M: $m_1, \ldots, m_{2^{n/2}}$
2. For i = 1, …, $2^{n/2}$ compute $t_i = H(m_i)$ $\in \{0,1\}^n$
3. Look for a collision ($t_i = t_j$).   If not found, got back to step 1.

Expected number of iteration ≈ 2

Running time: $\mathbf{O(2^{n/2})}$ (space $O(2^{n/2})$)

Sample C.R. hash functions:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	function	digest size (bits)	Speed (MB/sec)	generic attack time
NIST standards	SHA-1	160	153	2^{80}
	SHA-256	256	111	2^{128}
	SHA-512	512	99	2^{256}
	Whirlpool	512	57	2^{256}

* best known collision finder for SHA-1 requires 2^{51} hash evaluations

Quantum Collision Finder

	Classical algorithms	**Quantum algorithms**
Block cipher **$E: K \times X \longrightarrow X$** exhaustive search	$O(\|K\|)$	$O(\|K\|^{1/2})$
Hash function **$H: M \longrightarrow T$** collision finder	$O(\|T\|^{1/2})$	$O(\|T\|^{1/3})$

End of Segment

Collision resistance

The Merkle-Damgard Paradigm

Collision resistance: review

Let $H: M \to T$ be a hash function $\quad (|M| >> |T|)$

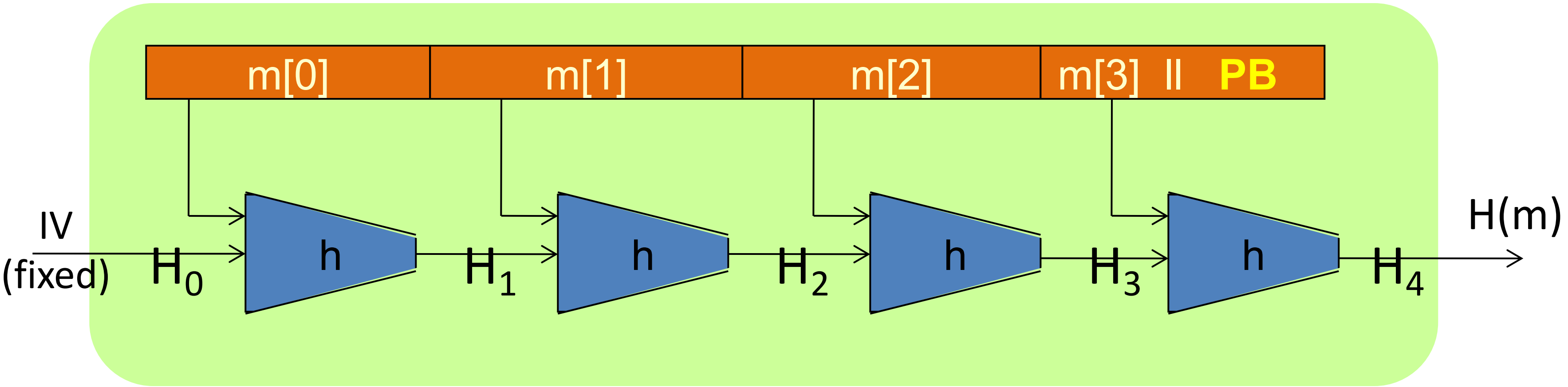
A **collision** for H is a pair $m_0 , m_1 \in M$ such that:

$$\mathbf{H(m_0) = H(m_1) \quad and \quad m_0 \neq m_1}$$

Goal: collision resistant (C.R.) hash functions

Step 1: given C.R. function for **short** messages,
construct C.R. function for **long** messages

The Merkle-Damgard iterated construction



Given **h: T × X ⟶ T** (compression function)

we obtain **H: $X^{\leq L}$ ⟶ T** . H_i - chaining variables

PB: padding block

If no space for PB
add another block

MD collision resistance

Thm: if h is collision resistant then so is H.

Proof: collision on H $\Rightarrow$ collision on h

Suppose H(M) = H(M'). We build collision for h.

$$IV = H_0, \quad H_1, \; \dots, \; H_t, \quad H_{t+1} = H(M)$$

$$IV = H_0', \quad H_1', \; \dots, \; H'_r, \quad H'_{r+1} = H(M')$$

$h(H_t, M_t \parallel PB) = H_{t+1} = H'_{r+1} = h(H'_r, M'_r \parallel PB')$

If $[H_t \neq H'_r \; or \; M_t \neq M'_r \; or \; PB \neq PB'] \Rightarrow$
We have a collission on h.
Stop.

Otherwise, Suppose $H_t = H'_r$ and $M_t = M'_r$ and $PB = PB'$

$\Downarrow$ t=r

Then: $h(H_{t-1}, M_{t-1}) = H_t = H'_t = h(H'_{t-1}, M'_{t-1})$

If $[H_t \neq H'_{t-1}\ or M_t \neq M'_{t-1}]$ then we have a collision on h. Stop.

Otherwise, $H_t \neq H'_{t-1}\ and\ M_t \neq M'_t$ and $M_{t-1} = M'_{t-1}$

Iterate all the way to beginning and either :

(1)find collision on h,or

(2) $\forall\, i: M_i = M'_i \Longrightarrow M = M'$ (Cannot happen because M, M' are collision on H.)

$\Rightarrow$ To construct C.R. function,

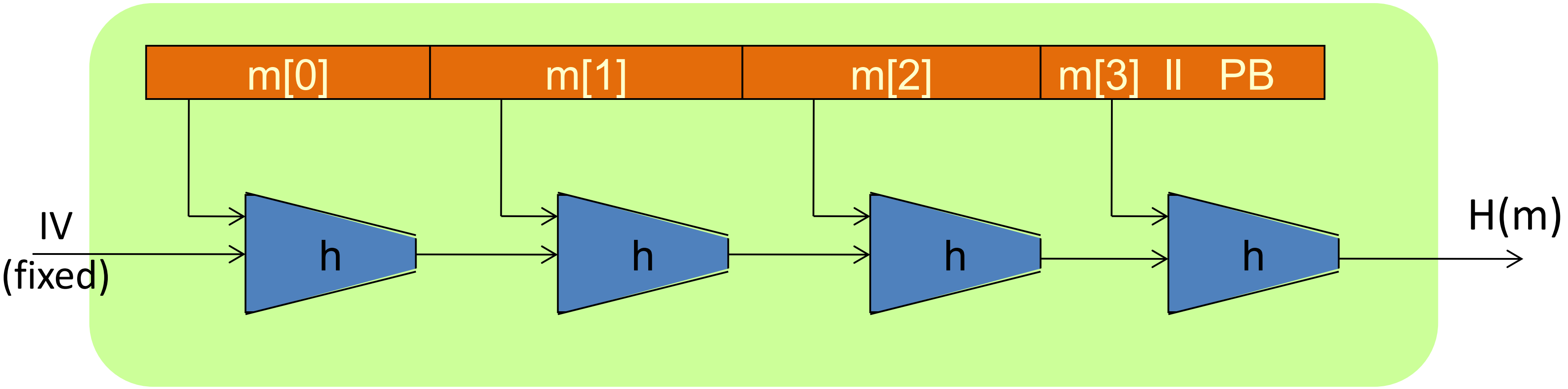
suffices to construct compression function

End of Segment

Collision resistance

HMAC: a MAC from SHA-256

The Merkle-Damgard iterated construction



Thm: h collision resistant ⇒ H collision resistant

Can we use H(.) to directly build a MAC?

MAC from a Merkle-Damgard Hash Function

$\mathbf{H: X^{\leq L} \longrightarrow T}$ a C.R. Merkle-Damgard Hash Function

Attempt #1: $\mathbf{S(k, m) = H(\, k \parallel m)}$

This MAC is insecure because:

Given H(k ll m) can compute H(w ll k ll m ll PB) for any w.

Given H(k ll m) can compute H(k ll m ll w) for any w.

√ Given H(k ll m) can compute H(k ll m ll PB ll w) for any w.

Anyone can compute H(k ll m) for any m.

Standardized method: HMAC (Hash-MAC)

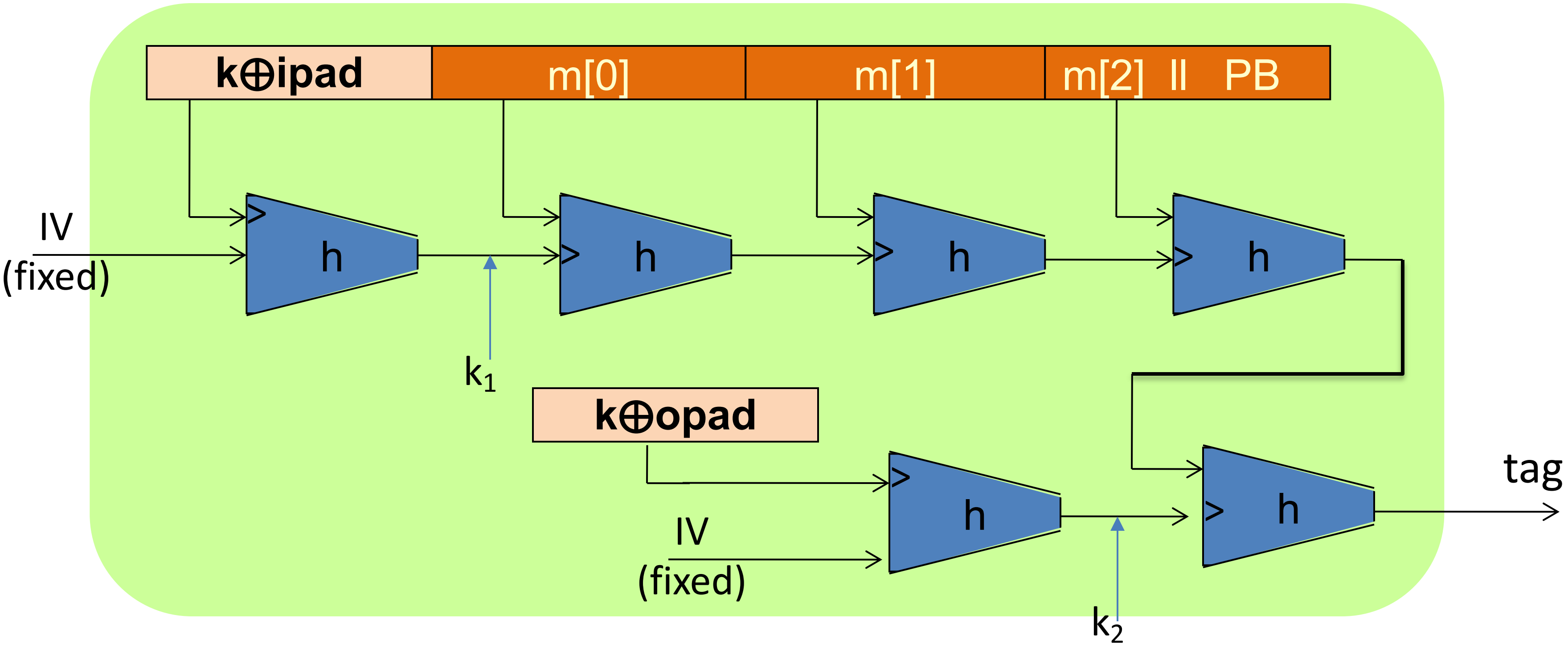
Most widely used MAC on the Internet.

H: hash function.
example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

HMAC: $S(k, m) = H\big(k \oplus opad \parallel \mathbf{H(k \oplus ipad \parallel m)} \big)$

HMAC in pictures



Similar to the NMAC PRF.

main difference: the two keys k_1, k_2 are dependent

HMAC properties

Built from a black-box implementation of SHA-256.

HMAC is assumed to be a secure PRF

- Can be proven under certain PRF assumptions about h(.,.)
- Security bounds similar to NMAC
  - Need $q^2/|T|$ to be negligible ($q \ll |T|^{1/2}$)

In TLS: must support HMAC-SHA1-96

End of Segment