

目标是cca
但还未达到
要先引入工具

Message Integrity and CCA Security

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

哈希函数在密码学里叫做“杂凑函数”

Hash Functions

- A hash function (**algorithm**) is denoted by

$$h: \{0, 1\}^* \rightarrow \{0, 1\}^n$$

where n is a security parameter.

输入任意长，输出定长
一般输出是256/512

哈希按地址寻址，查找速度最快 $n \log n$
冲突以后可以加链

- Let x be some message. $h(x)$ is called the message digest.
- x can be of arbitrary length while $h(x)$ has a fixed length
- Given x , it is easy to compute $h(x)$.

Hash Functions

- A hash function (**algorithm**) is denoted by

$$h: \{0, 1\}^* \rightarrow \{0, 1\}^n$$

where n is a security parameter.

$h(x)$ cannot be too short.

- Recommended message digest lengths (in bits):
- 128 (MD5), 已经被攻破
- 160 (SHA-1), 危险
- 224/256/384/512 (SHA-2), 老的
- 224/256/384/512 (SHA-3) 最主流

Hash Functions (Security Definitions)

- A hash function (algorithm) is denoted by $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$

Adversary's Target: Given a hash function h , find $x \neq x'$ such that $h(x) = h(x')$.

Hash要做密码学需要的三个特性: 1.抗碰撞; 2.输入不相等 输出不能等 3.单向性

1. Collision Resistance:

Given a hash function $h: \mathcal{X} \rightarrow \mathcal{Y}$, there is no efficient mechanism (P.P.T. adversary) to find $x, x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

第一个有多个解, 第二个只有一个解

2. Second Pre-Image Resistance:

Given a hash function $h: \mathcal{X} \rightarrow \mathcal{Y}$ and $x \in \mathcal{X}$, there is no efficient mechanism (P.P.T. adversary) to find $x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

Hash Functions (Security Definitions)

1. Collision Resistance:

Given a hash function $h: \mathcal{X} \rightarrow \mathcal{Y}$, there is no efficient mechanism (P.P.T. adversary) to find $x, x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

2. Second Pre-Image Resistance:

Given a hash function $h: \mathcal{X} \rightarrow \mathcal{Y}$ and $x \in \mathcal{X}$, there is no efficient mechanism (P.P.T. adversary) to find $x' \in \mathcal{X}$ such that $x' \neq x$ and $h(x') = h(x)$.

NOTE: $n/2$

No P.P.T. adversary can do

=

Each adversary can only successfully do in polynomial time with negligible probability.

Hash Functions (Security Definitions)

- A hash function (**algorithm**) is denoted by $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$

Adversary's Target: Given a hash function h , find $x \neq x'$ such that $h(x) = h(x')$.

3. Pre-Image Resistance:

Given a hash function $h: \mathcal{X} \rightarrow \mathcal{Y}$ **and $y \in \mathcal{Y}$** , there is no efficient mechanism (P.P.T. adversary) to find $x \in \mathcal{X}$ such that $y = h(x)$.

$n/2$

Hash Functions (Relations)

Collision Resistance **Implies** Second Pre-Image Resistance

Collision Resistance **Doesn't Imply** Pre-Image Resistance

A security implies B security:

If a cryptosystem is secure with A property, then it must also have the B property.

A security doesn't imply B security:

It is possible that a cryptosystem is A secure but not B secure.

Collision \rightarrow Second Pre-image

Theorem: Collision *resistance* implies second pre-image *resistance*.

Proof (sketch):

Collision \rightarrow Second Pre-image

Theorem: Collision *resistance* implies second pre-image *resistance*.

Proof (sketch): **Otherwise**

Suppose we have a function h that has collision resistance but it is **not** second pre-image resistance.

Consider a fixed input x . If h does not have second pre-image resistance, then we can find a distinct x' such that $h(x) = h(x')$.

We have that (x, x') is a pair of distinct inputs hashing to the same output, contradicting the assumption.

Collision Resistance

- The Problem: Given a hash function $h: \mathcal{X} \rightarrow \mathcal{Y}$ find $x' \neq x$ and $h(x') = h(x)$.
- The (ε, q) -Algorithm:
Choose $\mathcal{X}_0 \subseteq_R \mathcal{X}: |\mathcal{X}_0| = q$
For each $x_i \in \mathcal{X}_0$ (Go through the elements in order.)
 $y_i \leftarrow h(x_i)$
 If $y_i = y_{i'}$ for $i' < i$
 return $(x_i, x_{i'})$
return “Failure”

- What chance do we have?

$$\varepsilon \leq 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{|\mathcal{Y}|} \right)$$

Cal: 23 people in one room
50% expectations to birthday collision.

- This is also related to the [birthday paradox](#).
 - How many people do you need in a room before the probability of any two sharing a birthday is at least $\frac{1}{2}$.
(We only need about 23 people)
 - In the above, we can roughly have $q = |\mathcal{Y}|^{\frac{1}{2}}$ for $\varepsilon = 0.5$
 - $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$. We only need about $q=2^{\{64\}}$ for $n=128$

抗碰撞的大小为 $2^{2/n}$

What are hash functions used for?

- For storing password. Server stores $h(\text{pw})$ instead of pw . (For confidentiality)
- For integrity (e.g. HMAC). (introduced later)
- In digital signatures, we sign on $H(m)$ instead of m . (For efficiency)
- For proof-of-work. (In Bit-coin and blockchain)

Proof of Work 工作量证明

Given a data D, a person will be rewarded if he/she can find r such that

1. $h(D, r) = \text{Output}$
2. $\text{Output} = \text{00000000000000000000} \text{*****}$

(If Alice is using a super-computer while Bob is using a laptop, Alice will mostly win against Bob)

- A hash function (e.g., SHA-256) takes a data block as input, and produces a “random” fixed size output.
- Proof of work:
 - A variable salt value is included in the block
 - Goal: to find a hash with N zeros at the beginning of the output

The Random-Oracle Model ****

- By assuming a hash function is a black box random function
- One can only have *oracle* access to the hash function, meaning they do not have a formula or algorithm for determining the hash value by themselves.
- Random oracle (black box) responds every query with a random response.
 - The response must be consistent
- The Random Oracle Model is a model of an **ideal** hash function used in security proofs.

Considering the ciphertext $CT=(x, H(x) \oplus m)$. What should the adversary do in order to decrypt the ciphertext?

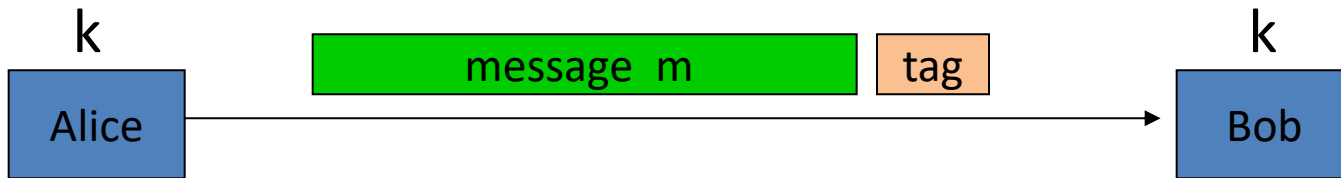
Message Integrity

Goal: **integrity**, no confidentiality.

Examples:

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

Message integrity: MACs



Generate tag:

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

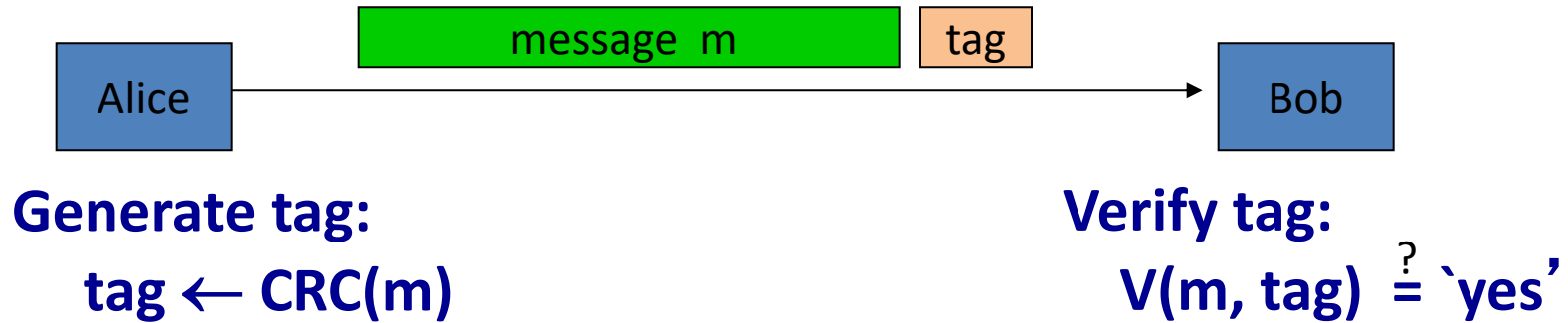
Verify tag:

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

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



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

Attacker's goal: **existential forgery**

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

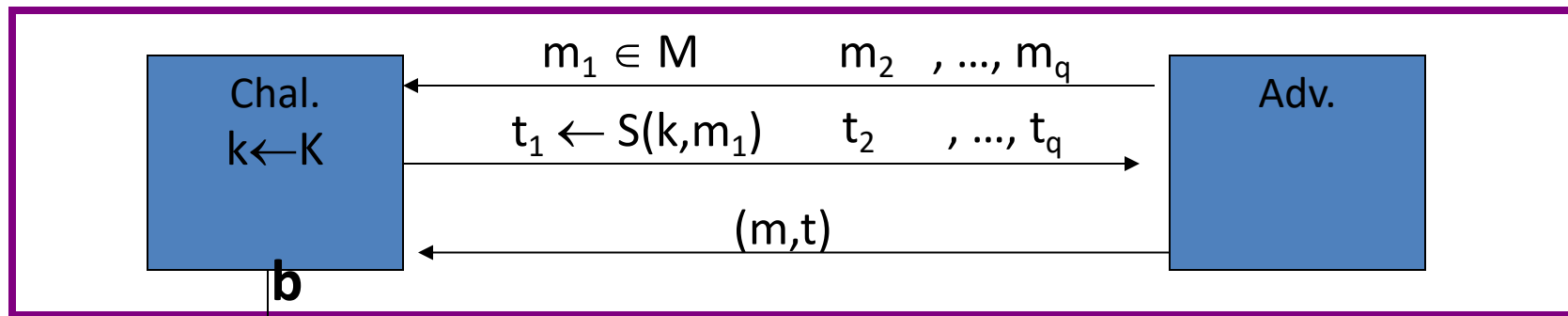
$$(m, t) \notin \{ (m_1, t_1), \dots, (m_q, t_q) \}$$

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

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

Secure MACs

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



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

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

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

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

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

$$S(k, m_0) = S(k, m_1) \quad \text{for } \frac{1}{2} \text{ of the keys } k \text{ in } K$$

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.

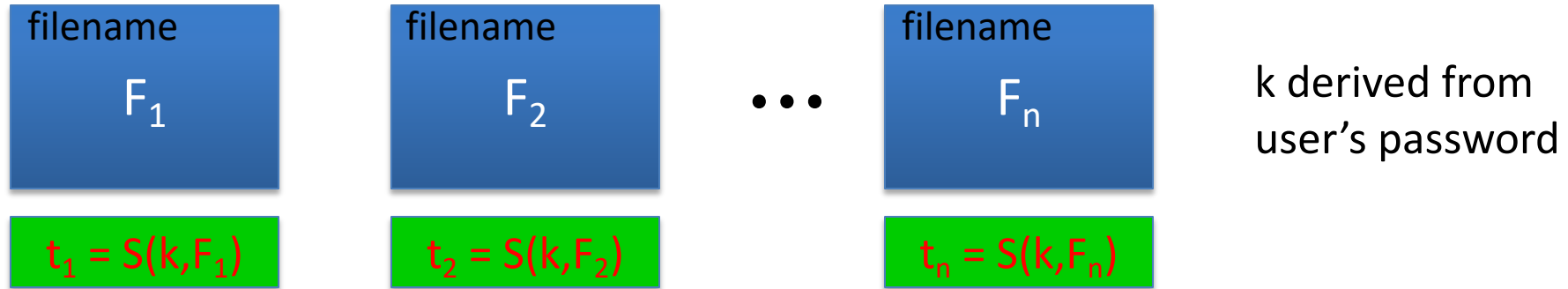
Suppose $S(k,m)$ is always 5 bits long

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:



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. $S(k,m) \rightarrow t$ and verification alg. $V(k,m,t) \rightarrow 0,1$

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 new valid message/tag pair (m, t) .

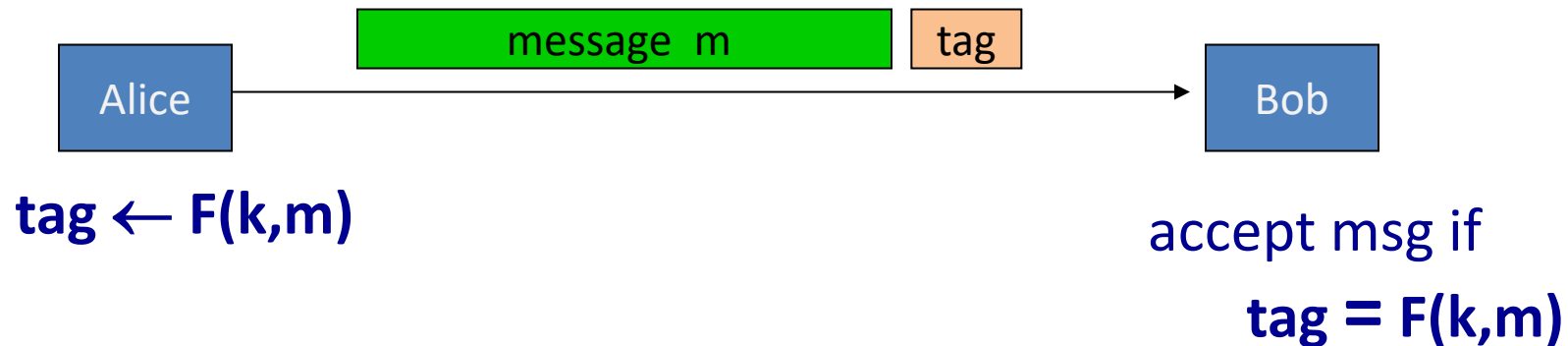
$$(m, t) \notin \{ (m_1, t_1), \dots, (m_q, t_q) \}$$

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

Secure PRF \Rightarrow Secure MAC

For a PRF $F: K \times X \rightarrow Y$ 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.



A bad example

Suppose $F: K \times X \rightarrow Y$ 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
- ☐

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

Security

Thm: If $F: K \times X \rightarrow Y$ 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.:

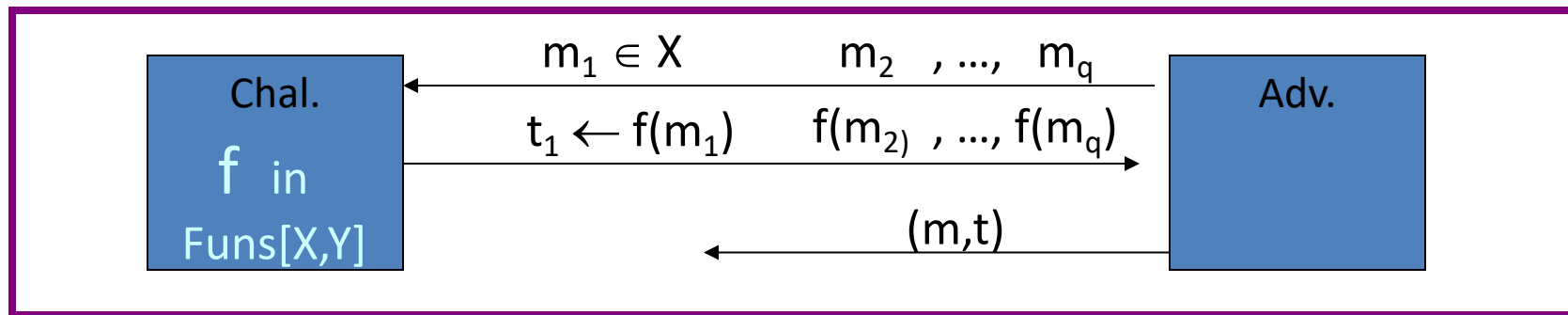
$$\text{Adv}_{\text{MAC}}[A, I_F] \leq \text{Adv}_{\text{PRF}}[B, F] + 1/|Y|$$

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

Proof Sketch

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

Then MAC adversary A must win the following game:



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

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


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 $F: K \times X \rightarrow \{0,1\}^n$ is a secure PRF.

Then so is $F_t(k,m) = F(k,m)[1...t]$ for all $1 \leq t \leq n$


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 $w \geq 64$)

Collision resistance

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

Collision Resistance

Let $H: M \rightarrow T$ be a hash function $(|M| \gg |T|)$

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 :

$$\text{Adv}_{\text{CR}}[A, H] = \Pr[A \text{ outputs collision for } H]$$

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 $H: M^{\text{big}} \rightarrow M$

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

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

Thm: If I is a secure MAC and H is collision resistant
then I^{big} is a secure MAC.

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

MACs from Collision Resistance

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

Collision resistance is necessary for security:

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

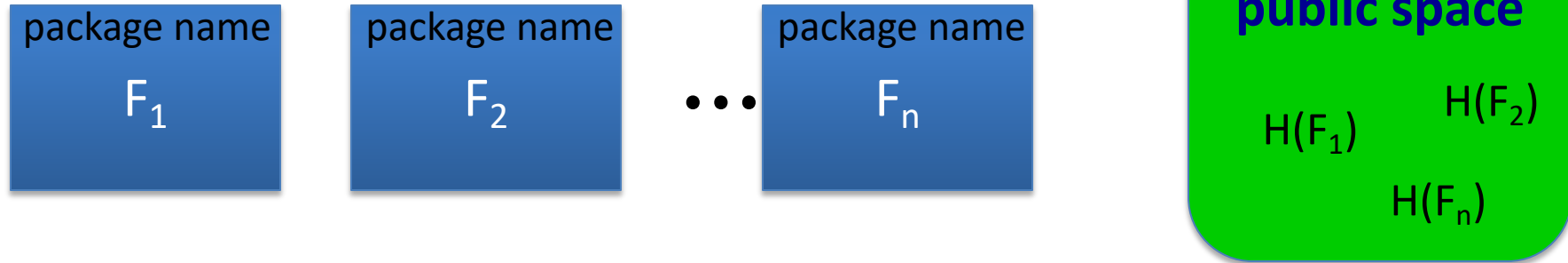
Then: S^{big} is insecure under a 1-chosen msg attack

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

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

The Merkle-Damgard Paradigm

Collision resistance: review

Let $H: M \rightarrow T$ be a hash function ($|M| \gg |T|$)

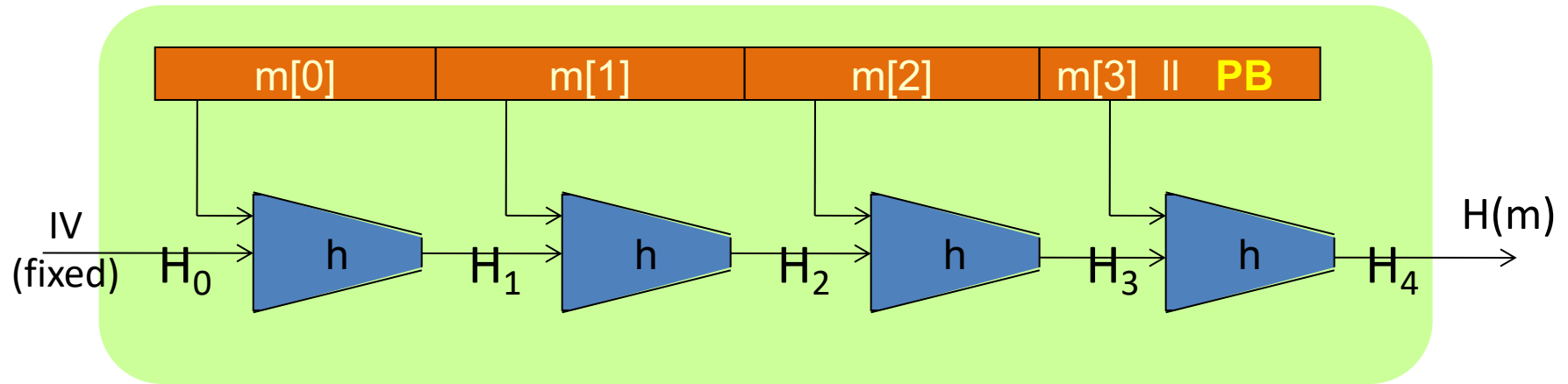
A **collision** for H is a pair $m_0, m_1 \in M$ such that:

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

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-Damgård iterated construction



Given $\mathbf{h: T \times X \rightarrow T}$ (compression function)

we obtain $\mathbf{H: X^{\leq L} \rightarrow T}$. 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 .

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

$$IV = H'_0, H'_1, \dots, H'_r, H'_{r+1} = H(M')$$

If $[H_t \neq H'_r \text{ or } M_t \neq M'_r \text{ or } PB \neq PB'] \Rightarrow$
We have a collision on h .
Stop.

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

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



$t=r$

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

If $[H_t \neq H'_{t-1} \text{ or } M_t \neq M'_{t-1}]$ 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) $\forall i: M_i = M'_i \implies M = M'$ (Cannot happen because M, M' are collision on H .)

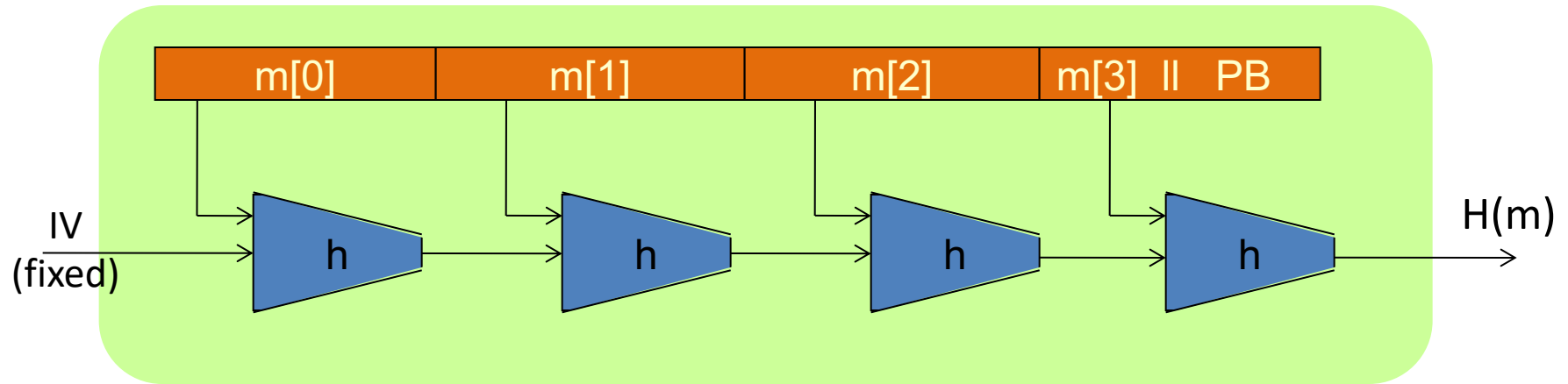
⇒ To construct C.R. function,
suffices to construct compression function

End of Segment

Collision resistance

HMAC: a MAC from SHA-256

The Merkle-Damgård iterated construction



Thm: h collision resistant $\Rightarrow H$ collision resistant

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

MAC from a Merkle-Damgard Hash Function

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

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

This MAC is insecure because:

Given $H(k \parallel m)$ can compute $H(w \parallel k \parallel m \parallel \text{PB})$ for any w .

Given $H(k \parallel m)$ can compute $H(k \parallel m \parallel w)$ for any w .

√ Given $H(k \parallel m)$ can compute $H(k \parallel m \parallel \text{PB} \parallel w)$ for any w .

Anyone can compute $H(k \parallel m)$ for any m .

Standardized method: HMAC (Hash-MAC)

Most widely used MAC on the Internet.

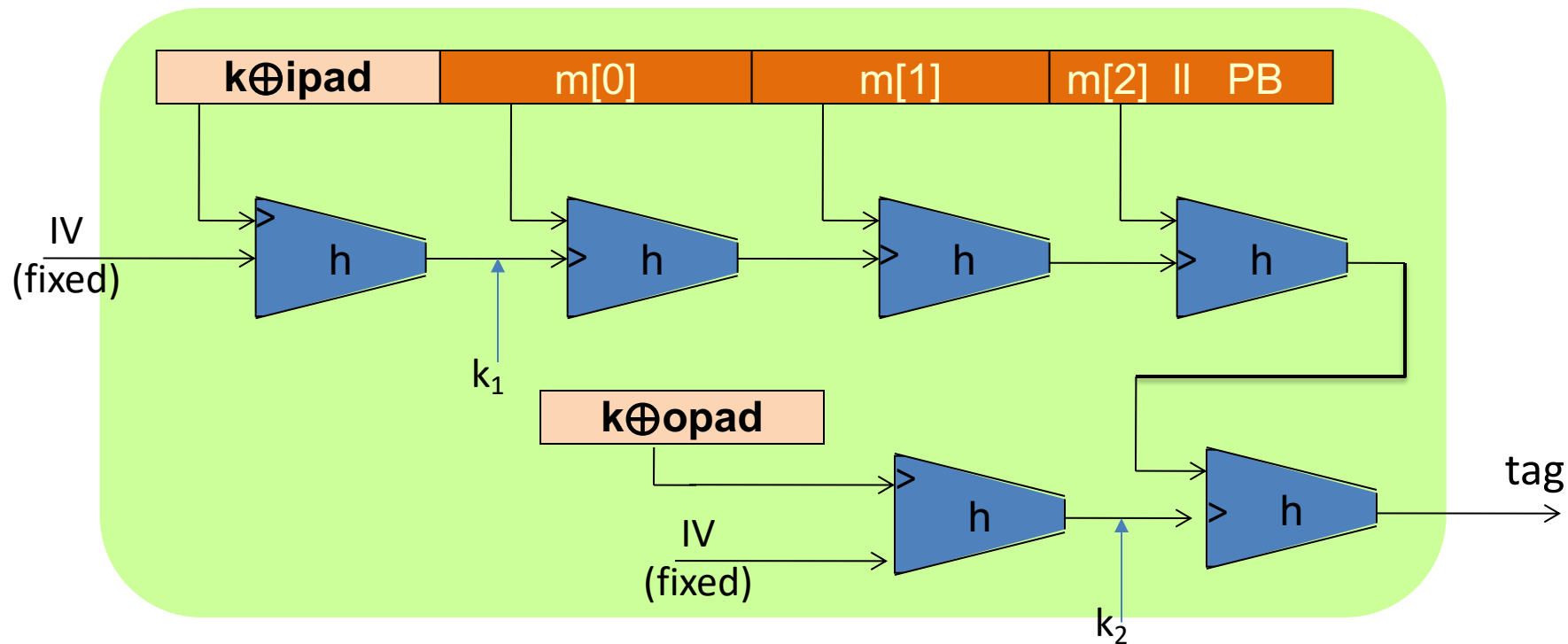
H: hash function.

example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

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

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

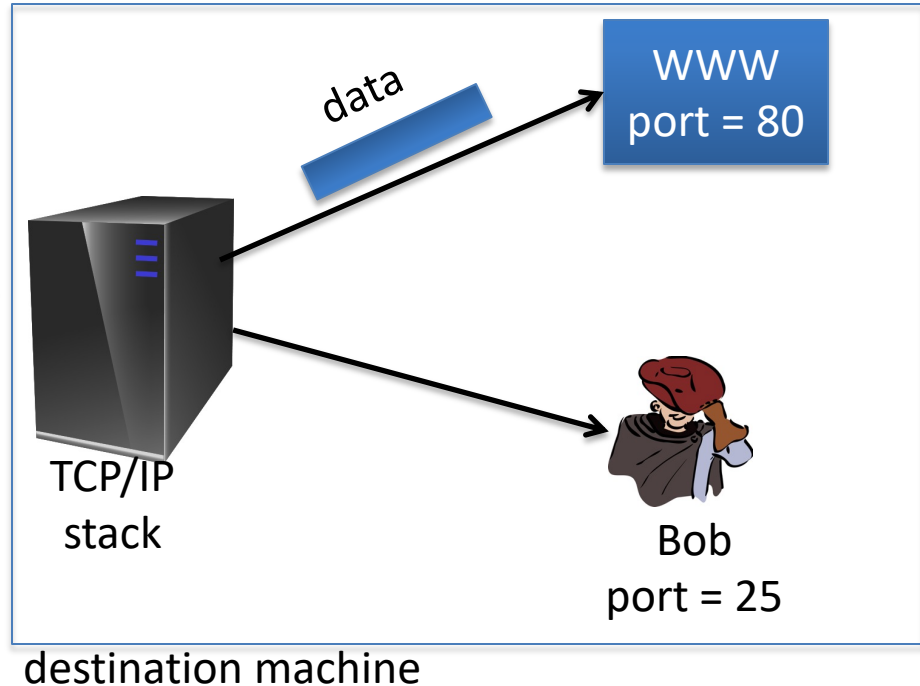
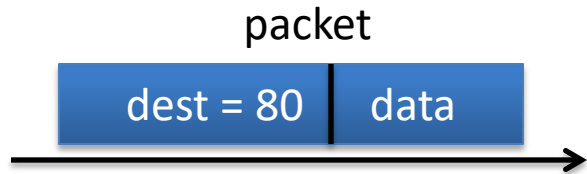
Authenticated Encryption

Sample tampering attacks

TCP/IP: (highly abstracted)

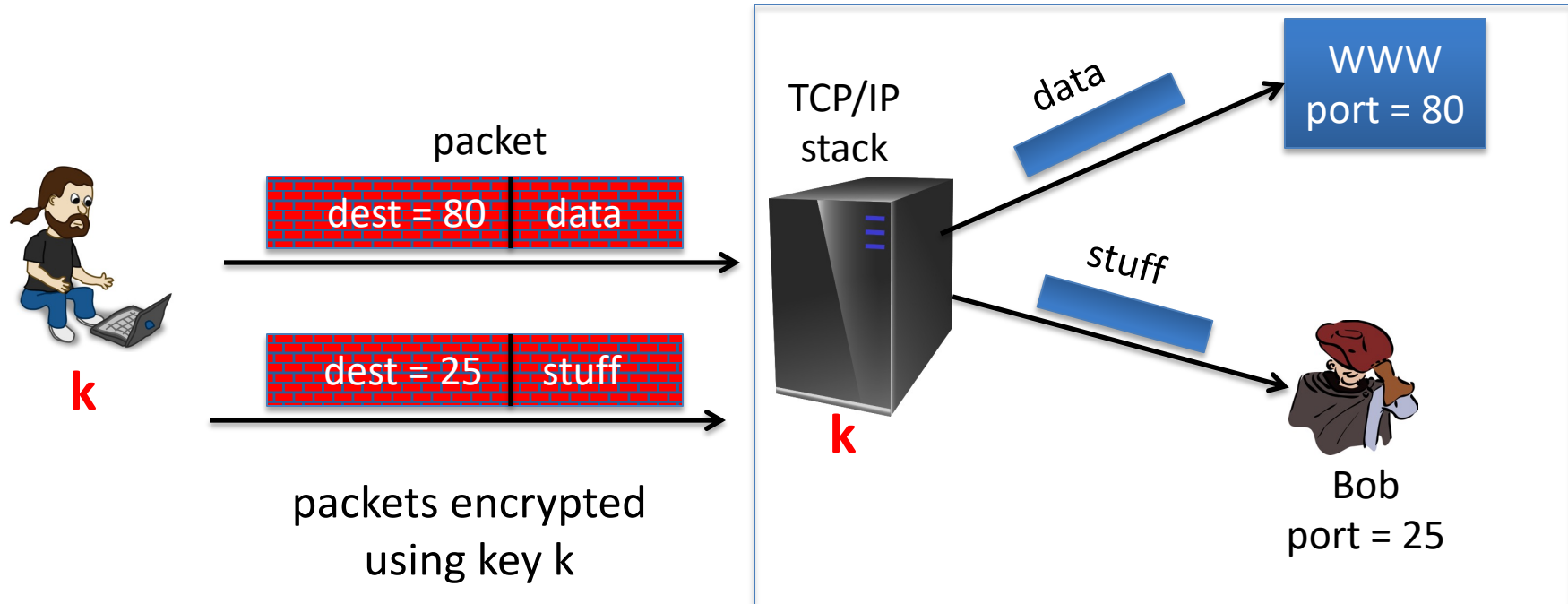


source machine



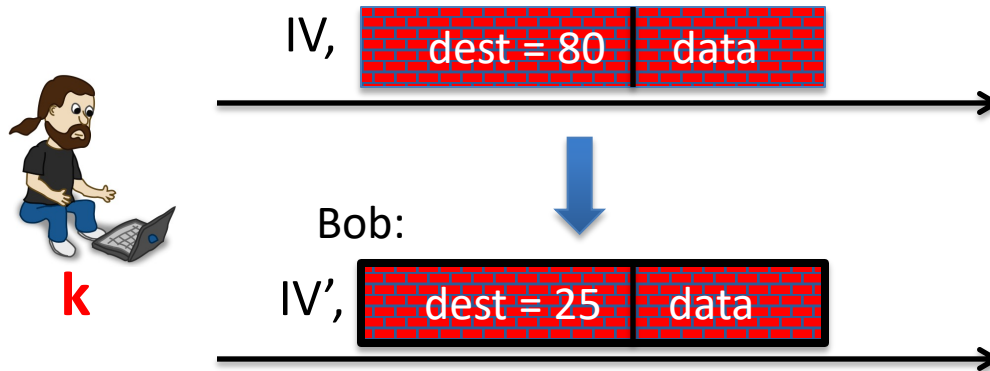
Sample tampering attacks

IPsec: (highly abstracted)

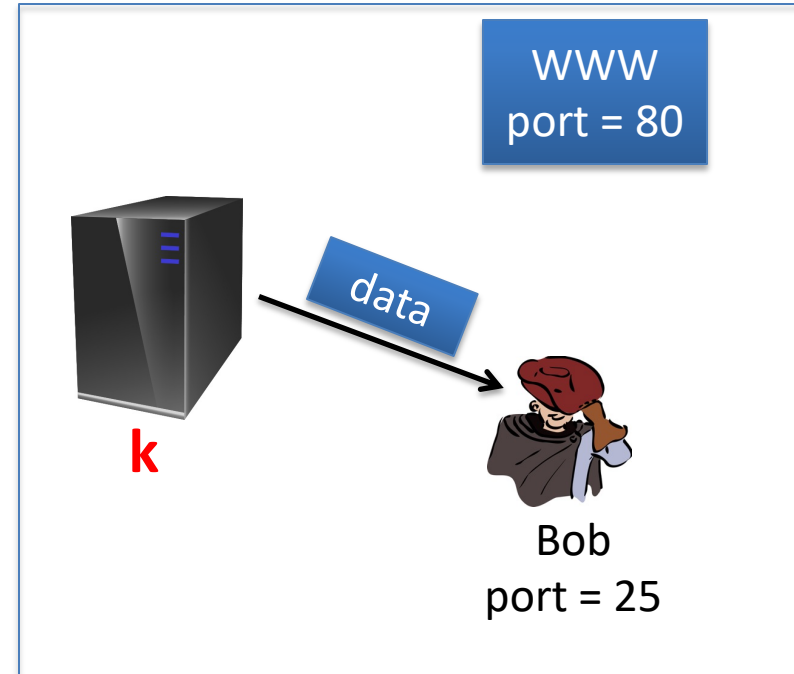


Reading someone else's data

Note: attacker obtains decryption of any ciphertext beginning with “dest=25”



Easy to do for CBC with rand. IV
(only IV is changed)





Encryption is done with CBC with a random IV.

What should IV' be? $m[0] = D(k, c[0]) \oplus IV = \text{"dest=80..."}$

$$IV' = IV \oplus (...25...)$$

$$IV' = IV \oplus (...80...)$$

$$IV' = IV \oplus (...80...) \oplus (...25...)$$

It can't be done

The lesson

CPA security cannot guarantee secrecy under active attacks.

Only use one of two modes:

- If message needs integrity but no confidentiality:
use a **MAC**
- If message needs both integrity and confidentiality:
use **authenticated encryption** modes (this module)

End of Segment

Authenticated Encryption

Definitions

Goals


An **authenticated encryption** system (E,D) is a cipher where

As usual: $E: K \times M \times N \rightarrow C$

but $D: K \times C \times N \rightarrow M \cup \{\perp\}$

Security: the system must provide

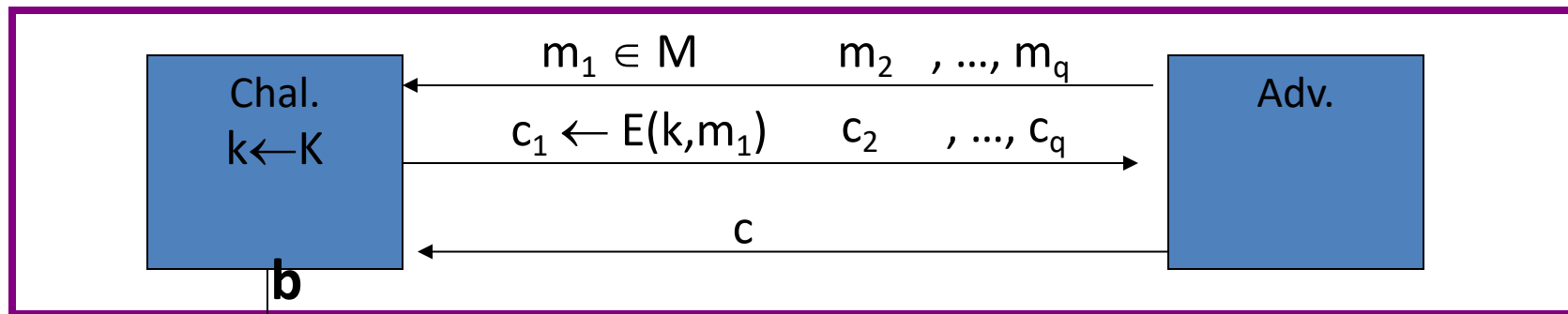
- sem. security under a CPA attack, and
- **ciphertext integrity**:
attacker cannot create new ciphertexts that decrypt properly



ciphertext
is rejected

Ciphertext integrity

Let (E,D) be a cipher with message space M .



$$\begin{cases} b=1 & \text{if } D(k,c) \neq \perp \text{ and } c \notin \{c_1, \dots, c_q\} \\ b=0 & \text{otherwise} \end{cases}$$

Def: (E,D) has **ciphertext integrity** if for all “efficient” A :

$$\text{Adv}_{\text{CI}}[A,E] = \Pr[\text{Chal. outputs 1}] \text{ is “negligible.”}$$

Authenticated encryption

Def: cipher (E,D) provides authenticated encryption (**AE**) if it is

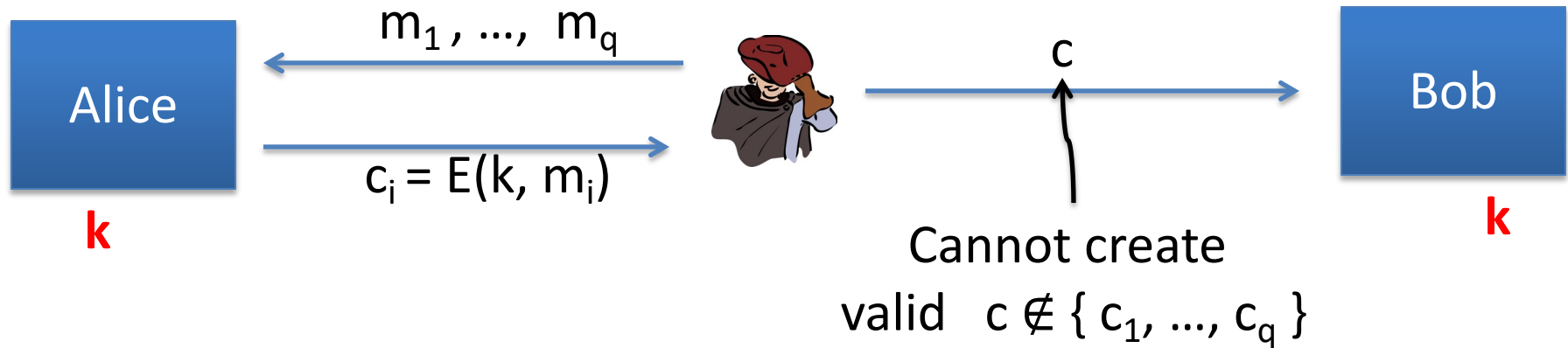
- (1) semantically secure under CPA, and
- (2) has ciphertext integrity

Bad example: CBC with rand. IV does not provide AE

- $D(k,\cdot)$ never outputs \perp , hence adv. easily wins CI game

Implication 1: authenticity

Attacker cannot fool Bob into thinking a message was sent from Alice



\Rightarrow if $D(k, c) \neq \perp$ Bob knows message is from someone who knows k
(but message could be a replay)

Implication 2

Authenticated encryption \Rightarrow

Security against **chosen ciphertext attacks**
(next segment)

End of Segment

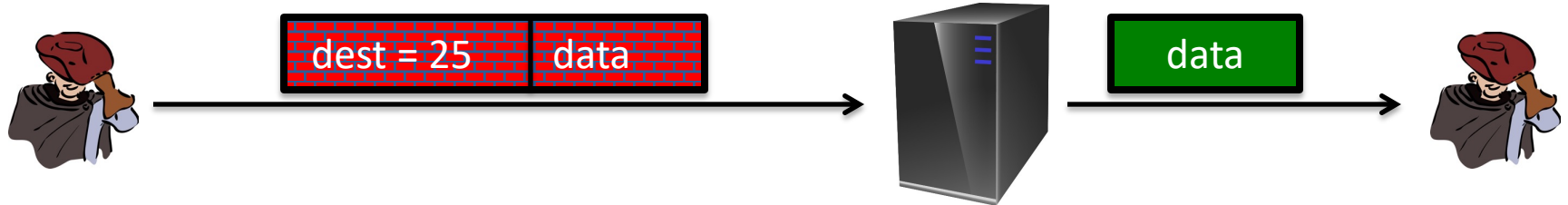
Authenticated Encryption

Chosen ciphertext attacks

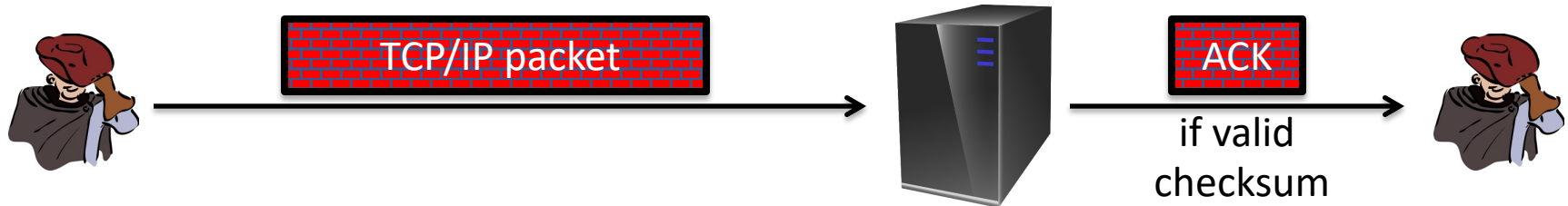
Example chosen ciphertext attacks

Adversary has ciphertext c that it wants to decrypt

- Often, adv. can fool server into decrypting **certain** ciphertexts (not c)



- Often, adversary can learn partial information about plaintext



Chosen ciphertext security

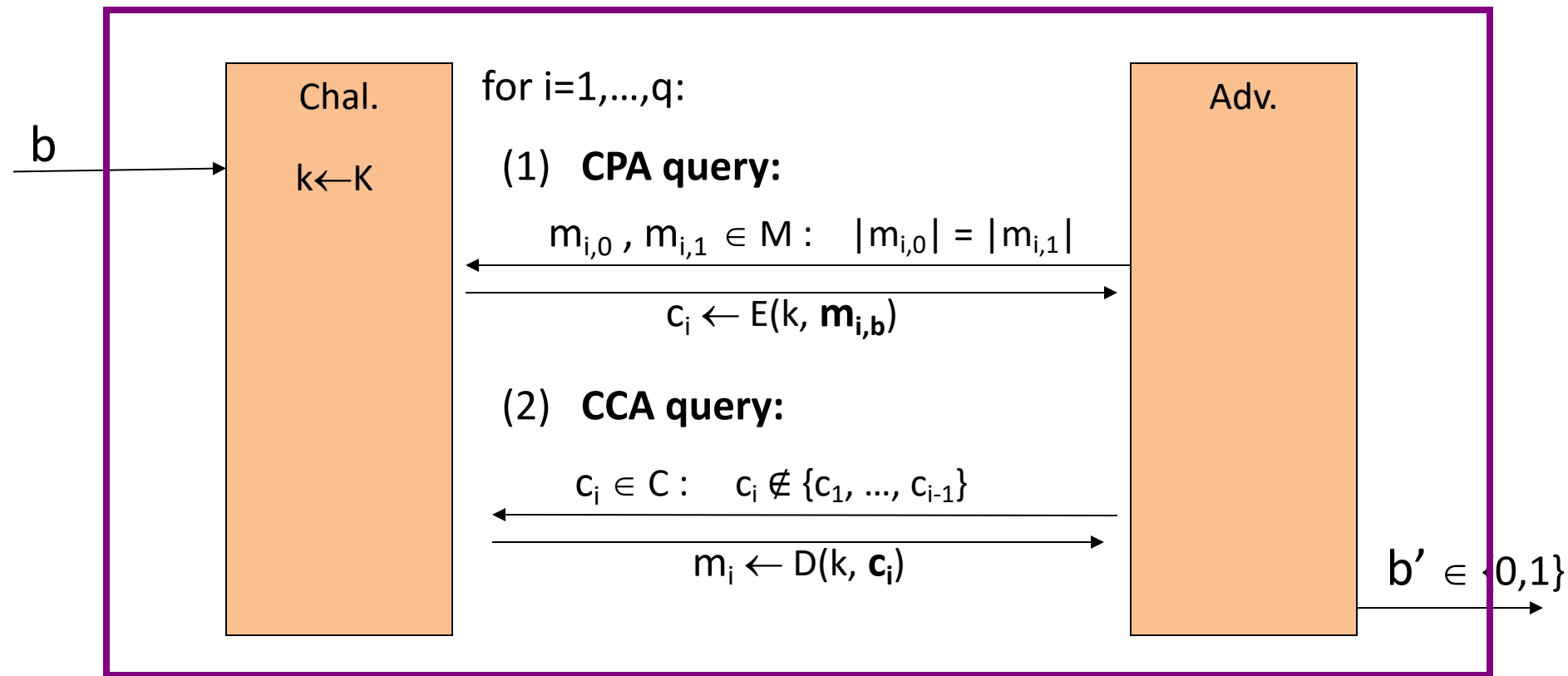
Adversary's power: both CPA and CCA

- Can obtain the encryption of arbitrary messages of his choice
- Can decrypt any ciphertext of his choice, other than challenge
(conservative modeling of real life)

Adversary's goal: Break semantic security

Chosen ciphertext security: definition

$\mathbb{E} = (E, D)$ cipher defined over (K, M, C) . For $b=0,1$ define $\text{EXP}(b)$:

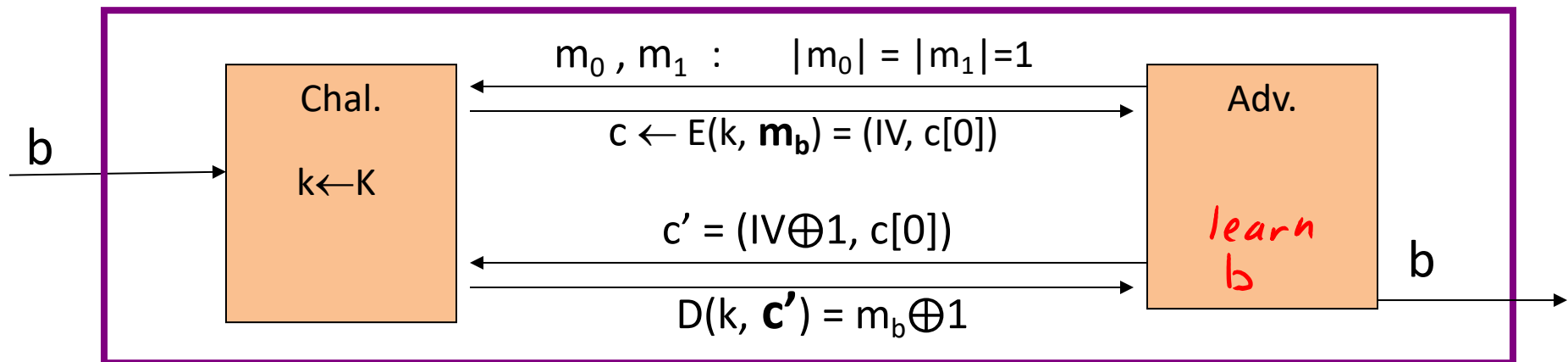


Chosen ciphertext security: definition

\mathbb{E} is CCA secure if for all “efficient” A :

$$\text{Adv}_{\text{CCA}}[A, \mathbb{E}] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right| \text{ is “negligible.”}$$

Example: CBC with rand. IV is not CCA-secure



Authenticated enc. \Rightarrow CCA security

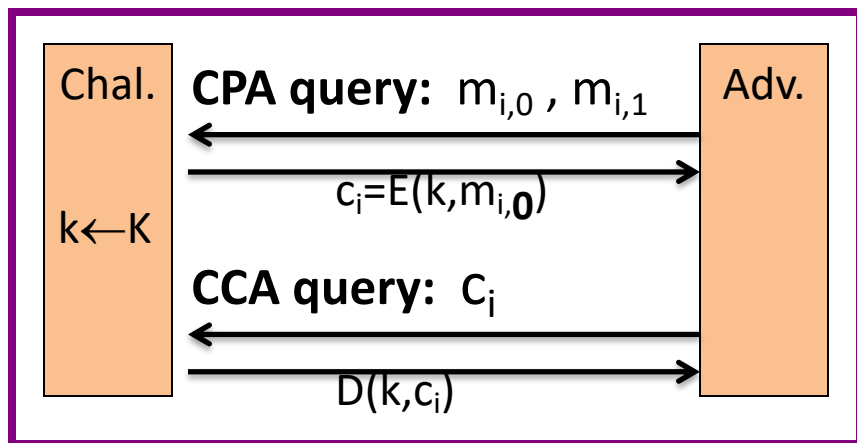
Thm: Let (E,D) be a cipher that provides AE.

Then (E,D) is CCA secure !

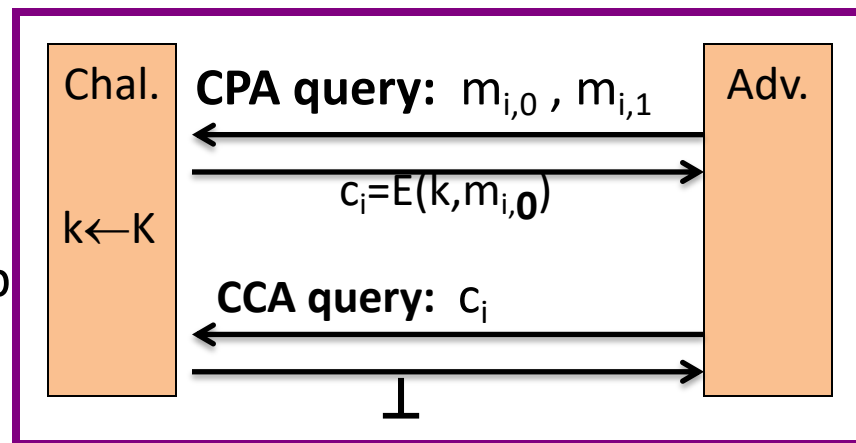
In particular, for any q-query eff. A there exist eff. B_1, B_2 s.t.

$$\text{Adv}_{\text{CCA}}[A,E] \leq 2q \cdot \text{Adv}_{\text{CI}}[B_1,E] + \text{Adv}_{\text{CPA}}[B_2,E]$$

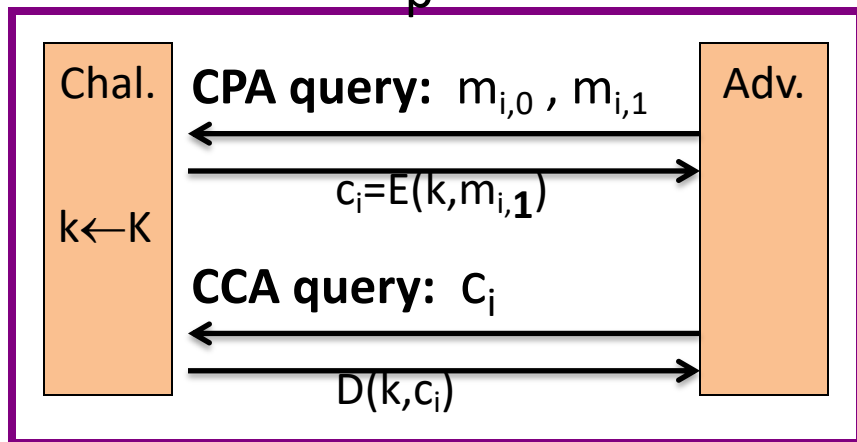
Proof by pictures



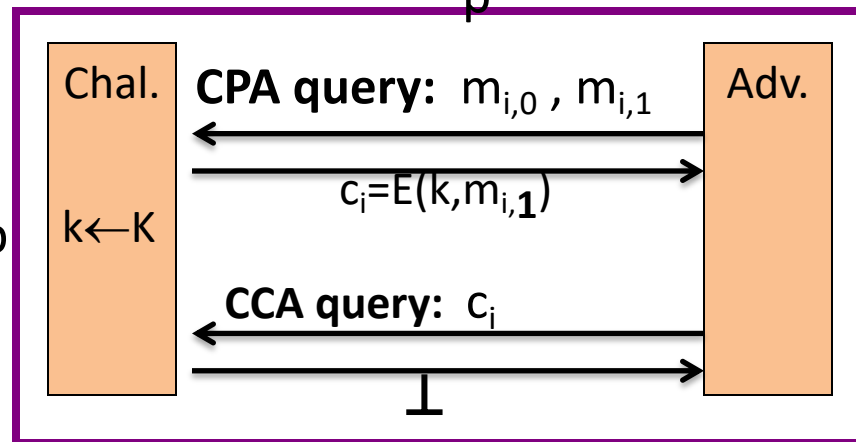
\approx_p



\approx_p



\approx_p



So what?

Authenticated encryption:

- ensures confidentiality against an active adversary that can decrypt some ciphertexts

Limitations:

- does not prevent replay attacks
- does not account for side channels (timing)

End of Segment

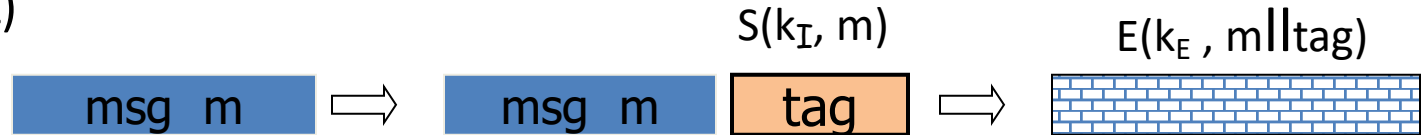
Authenticated Encryption

Constructions from
ciphers and MACs

Combining MAC and ENC (CCA)

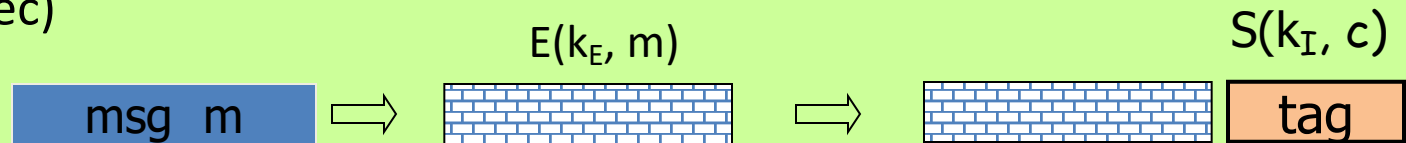
Encryption key k_E . MAC key = k_I

Option 1: (SSL)

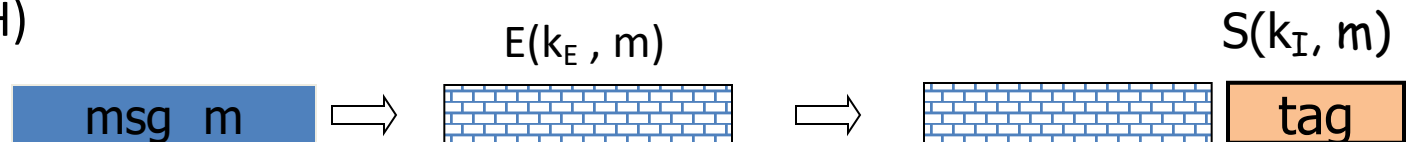


Option 2: (IPsec)

**always
correct**



Option 3: (SSH)



A.E. Theorems

Let (E,D) be CPA secure cipher and (S,V) secure MAC. Then:

1. **Encrypt-then-MAC:** always provides A.E.

2. **MAC-then-encrypt:** may be insecure against CCA attacks

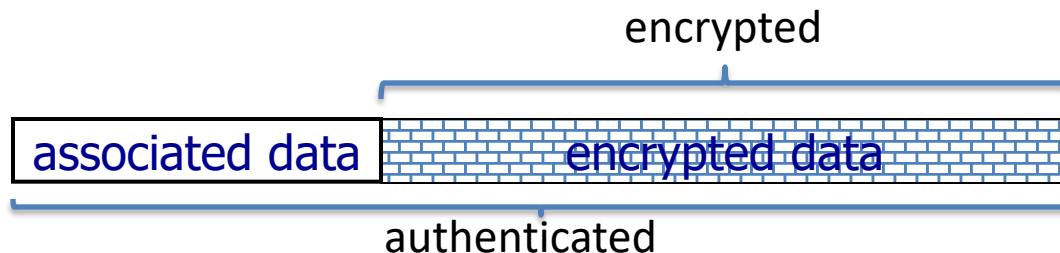
however: when (E,D) is rand-CTR mode or rand-CBC
M-then-E provides A.E.

for rand-CTR mode, one-time MAC is sufficient

Standards (at a high level)

- **GCM:** CTR mode encryption then CW-MAC
(accelerated via Intel's PCLMULQDQ instruction)
- **CCM:** CBC-MAC then CTR mode encryption (802.11i)
- **EAX:** CTR mode encryption then CMAC

All support AEAD: (auth. enc. with associated data). All are nonce-based.



An example API (OpenSSL)

```
int AES_GCM_Init(AES_GCM_CTX *ain,  
    unsigned char *nonce, unsigned long noncelen,  
    unsigned char *key, unsigned int klen )
```

```
int AES_GCM_EncryptUpdate(AES_GCM_CTX *a,  
    unsigned char *aad, unsigned long aadlen,  
    unsigned char *data, unsigned long datalen,  
    unsigned char *out, unsigned long *outlen)
```

Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

| <u>Cipher</u> | <u>code size</u> | <u>Speed (MB/sec)</u> | | |
|---------------|------------------|-----------------------|-----------|-----|
| AES/GCM | large** | 108 | AES/CTR | 139 |
| AES/CCM | smaller | 61 | AES/CBC | 109 |
| AES/EAX | smaller | 61 | | |
| | | | AES/CMAC | 109 |
| AES/OCB | | 129* | HMAC/SHA1 | 147 |

* extrapolated from Ted Kravitz's results

** non-Intel machines

End of Segment

Key Derivation

Deriving many keys from one

Typical scenario. a single source key (SK) is sampled from:

- Hardware random number generator
- A key exchange protocol (discussed later)

Need many keys to secure session:

- unidirectional keys; multiple keys for nonce-based CBC.

Goal: generate many keys from this one source key



When source key is uniform

F : a PRF with key space K and outputs in $\{0,1\}^n$

Suppose source key SK is uniform in K

- Define Key Derivation Function (KDF) as:

KDF(SK , CTX , L) :=

$F(SK, (CTX \parallel 0)) \parallel F(SK, (CTX \parallel 1)) \parallel \dots \parallel F(SK, (CTX \parallel L))$

CTX: a string that uniquely identifies the application

KDF(SK, CTX, L) :=

$F(\text{SK}, (\text{CTX} \parallel 0)) \parallel F(\text{SK}, (\text{CTX} \parallel 1)) \parallel \dots \parallel F(\text{SK}, (\text{CTX} \parallel L))$

What is the purpose of CTX?

Even if two apps sample same SK they get indep. keys

It's good practice to label strings with the app. name

It serves no purpose

What if source key is not uniform?

Recall: PRFs are pseudo random only when key is uniform in K

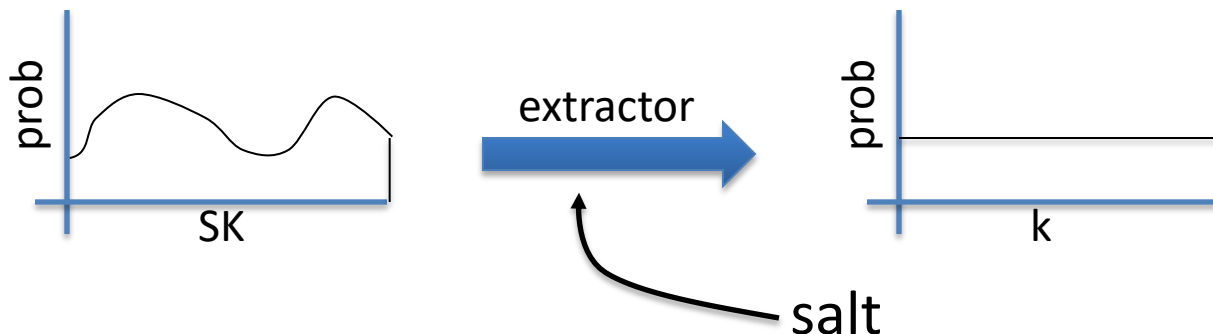
- SK not uniform \Rightarrow PRF output may not look random

Source key often not uniformly random:

- Key exchange protocol: key uniform in some subset of K
- Hardware RNG: may produce biased output

Extract-then-Expand paradigm

Step 1: **extract** pseudo-random key k from source key SK



salt: a fixed non-secret string chosen at random

step 2: **expand** k by using it as a PRF key as before

HKDF: a KDF from HMAC

Implements the extract-then-expand paradigm:

- extract: use $k \leftarrow \text{HMAC}(\text{salt}, SK)$
- Then expand using HMAC as a PRF with key k

Password-Based KDF (PBKDF)

Deriving keys from passwords:

- Do not use HKDF: passwords have insufficient entropy
- Derived keys will be vulnerable to dictionary attacks

PBKDF defenses: **salt** and a **slow hash function**

Standard approach: **PKCS#5** (PBKDF1)

$H^{(c)}(\text{pwd} \parallel \text{salt})$: iterate hash function c times

End of Segment

Random Oracle Model

Motivation

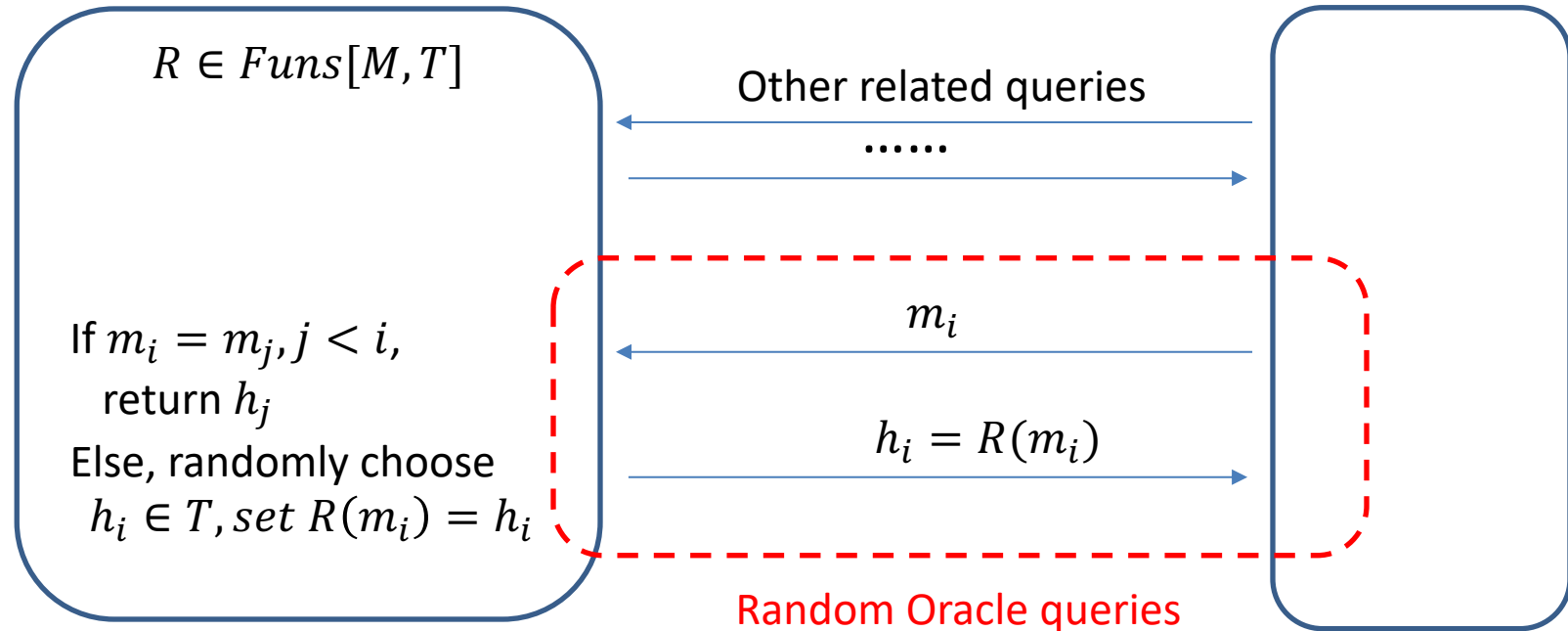
- Hash function is heavily used in many cryptographic applications
- It would be useful to analyze the security if we can simplify the model of hash function H as if it were a truly random function R .
- If H maps from space M to space T , the function R is chosen uniformly at random from the set $\text{Funs}[M, T]$.
- In security proof, we can translate any attack game into its random oracle version: the challenger uses R in place of H for all its computations
 - The adversary is allowed to obtain the value of R at any input points of his choosing.

Security proof in RO model

Scheme S involves computing a hash function H . If scheme S evaluates H at arbitrary points of its choice, but does not look at the internal implementation of H , we say S use H as an oracle.

Challenger

adversary



Challenger maintains a table of all queried m_i and the corresponding h_i

Discussion

- Security proved in RO model only rules out “generic attacks” on systems that would work if the hash function were a random oracle.
- They do not guarantee any security for systems built with any specific hash function.
- Still useful especially in practice since schemes proved secure in RO model are usually more efficient than ones proved in standard model (without RO model).