The AES Process

1997: NIST publishes request for proposal

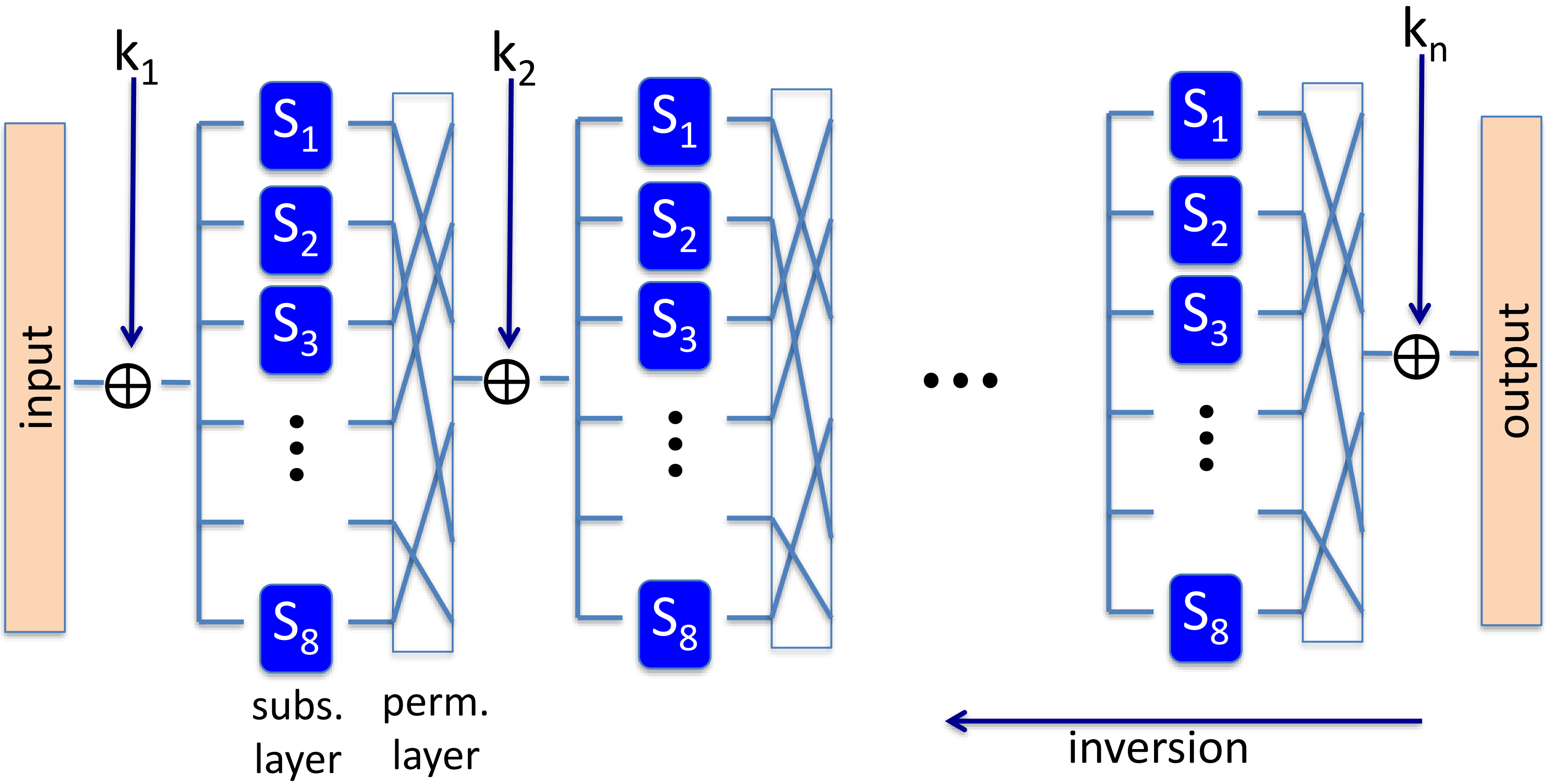
1998: 15 submissions. Five claimed attacks.

1999: NIST chooses 5 finalists

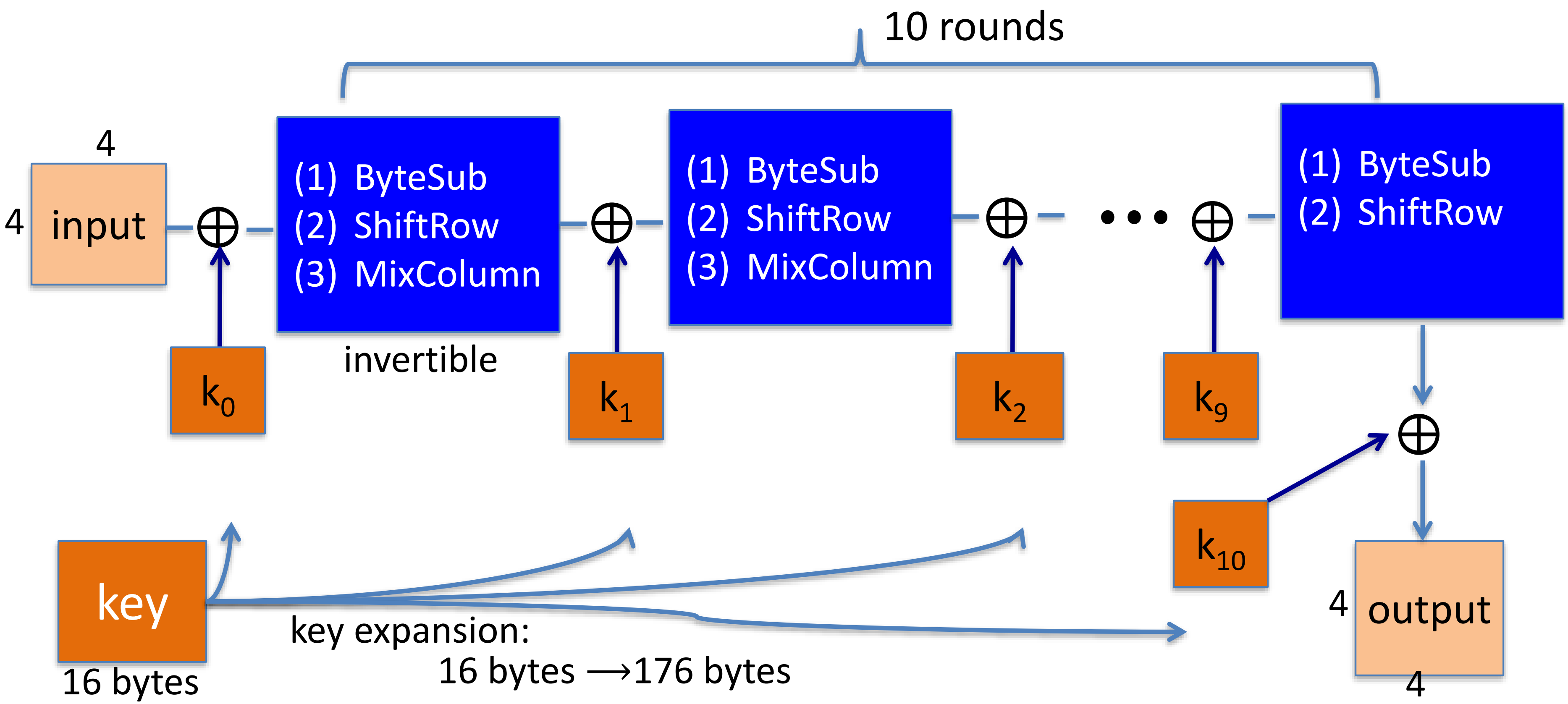
2000: NIST chooses Rijndael as AES (designed in Belgium)

Key sizes: 128, 192, 256 bits. Block size: 128 bits

AES is a Subs-Perm Network (not Feistel)



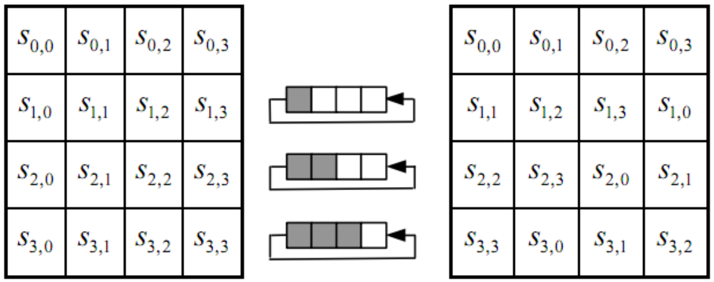
AES-128 Schematic



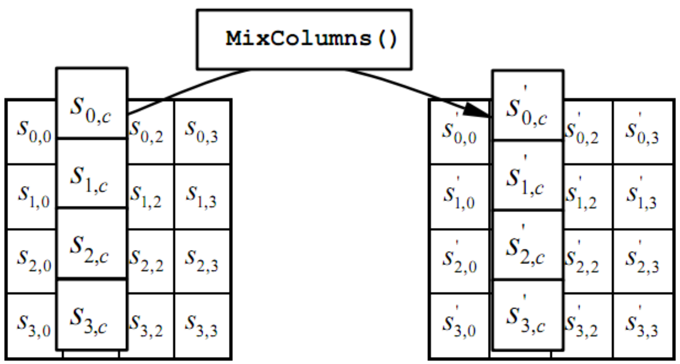
The Round Function

ByteSub: a 1 byte S-box. 256 byte table (easily computable)

ShiftRows:



MixColumns:



Code size/performance tradeoff

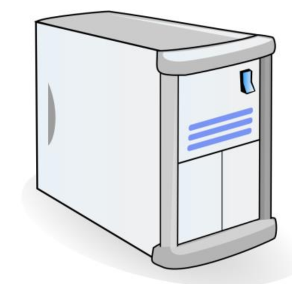
	Code size	Performance
Pre-compute round functions (24KB or 4KB)	largest	fastest: table lookups and xors
Pre-compute S-box only (256 bytes)	smaller	slower
No pre-computation	smallest	slowest

Example: Javascript AES

AES in the browser:

AES library (6.4KB)

no pre-computed tables



Prior to encryption:

pre-compute tables

Then encrypt using tables

Attacks

Best key recovery attack:

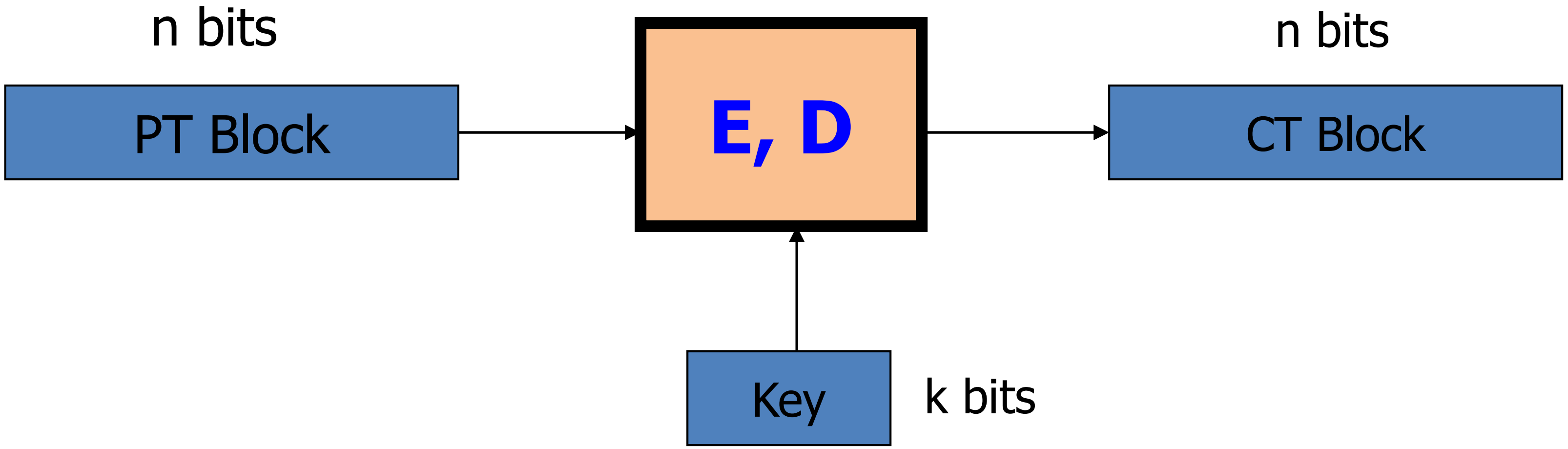
four times better than ex. search [BKR'11]

Related key attack on AES-256: [BK'09]

Given 2^{99} inp/out pairs from **four related keys** in AES-256

can recover keys in time $\approx 2^{99}$

Review PRFs and PRPs



Canonical examples:

1. 3DES: n= 64 bits, k = 168 bits
2. AES: n=128 bits, k = 128, 192, 256 bits

Abstractly: PRPs and PRFs

- Pseudo Random Function (**PRF**) defined over (K,X,Y):

  $$F: K \times X \to Y$$

  such that exists "efficient" algorithm to evaluate F(k,x)

- Pseudo Random Permutation (**PRP**) defined over (K,X):

  $$E: K \times X \to X$$

  such that:

  1. Exists "efficient" <u>deterministic</u> algorithm to evaluate E(k,x)
  2. The function $E(k, \cdot)$ is one-to-one
  3. Exists "efficient" inversion algorithm D(k,x)

Secure PRFs

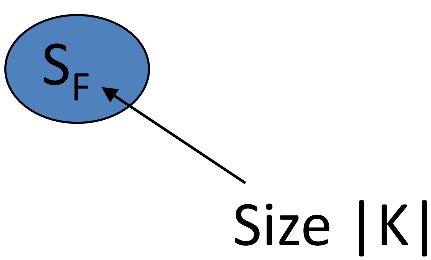
Let $F: K \times X \rightarrow Y$ be a PRF

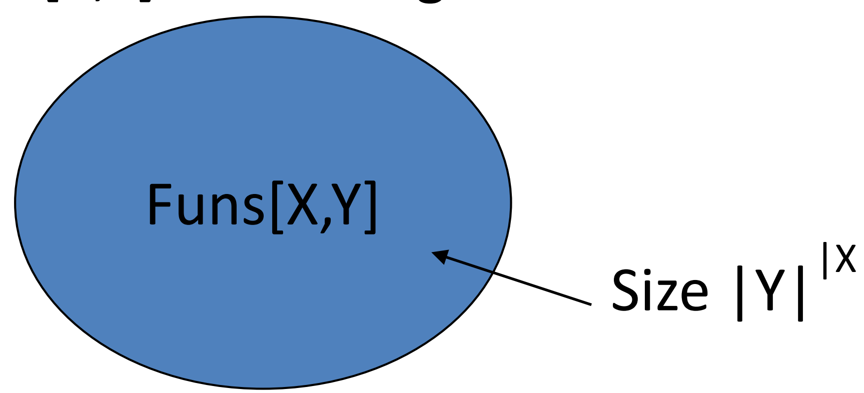
$$\begin{cases} \text{Funs[X,Y]:} & \text{the set of } \underline{\textbf{all}} \text{ functions from X to Y} \\ S_F = \{\ F(k,\cdot)\ \text{ s.t. }\ k \in K\ \} & \subseteq \quad \text{Funs[X,Y]} \end{cases}$$

<u>Intuition</u>: a PRF is **secure** if

a random function in Funs[X,Y] is indistinguishable from

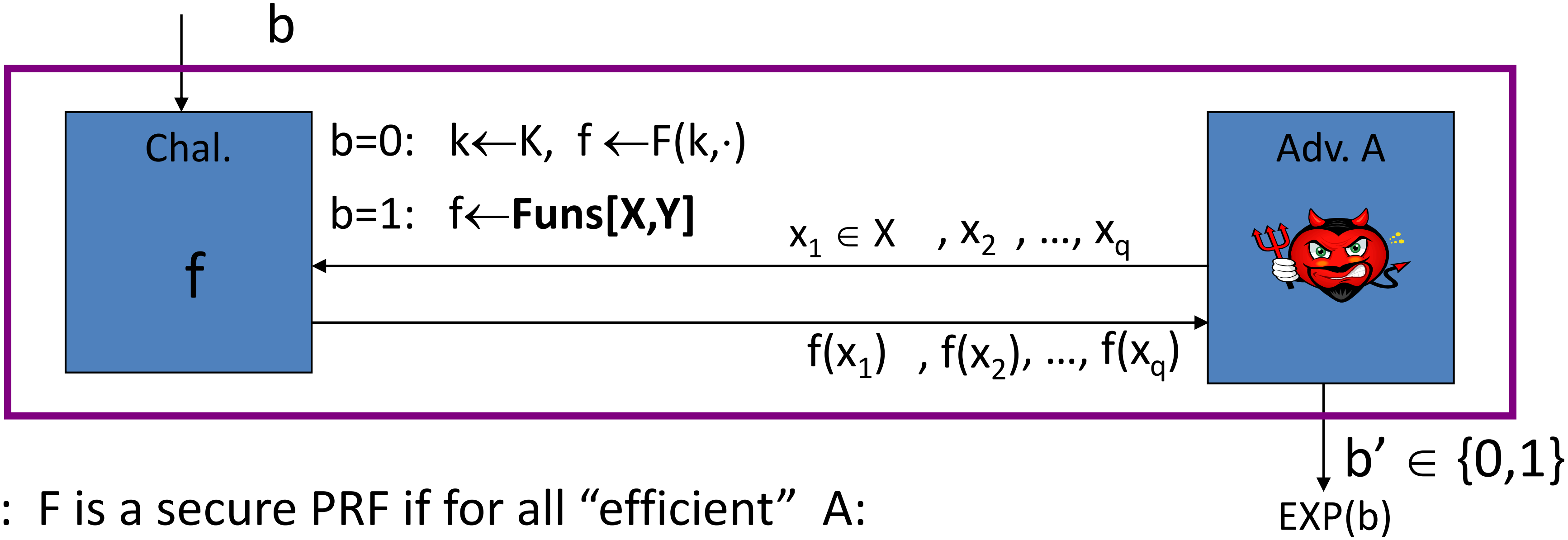
a random function in S_F





Secure PRF: definition

For $b=0,1$ define experiment $EXP(b)$ as:



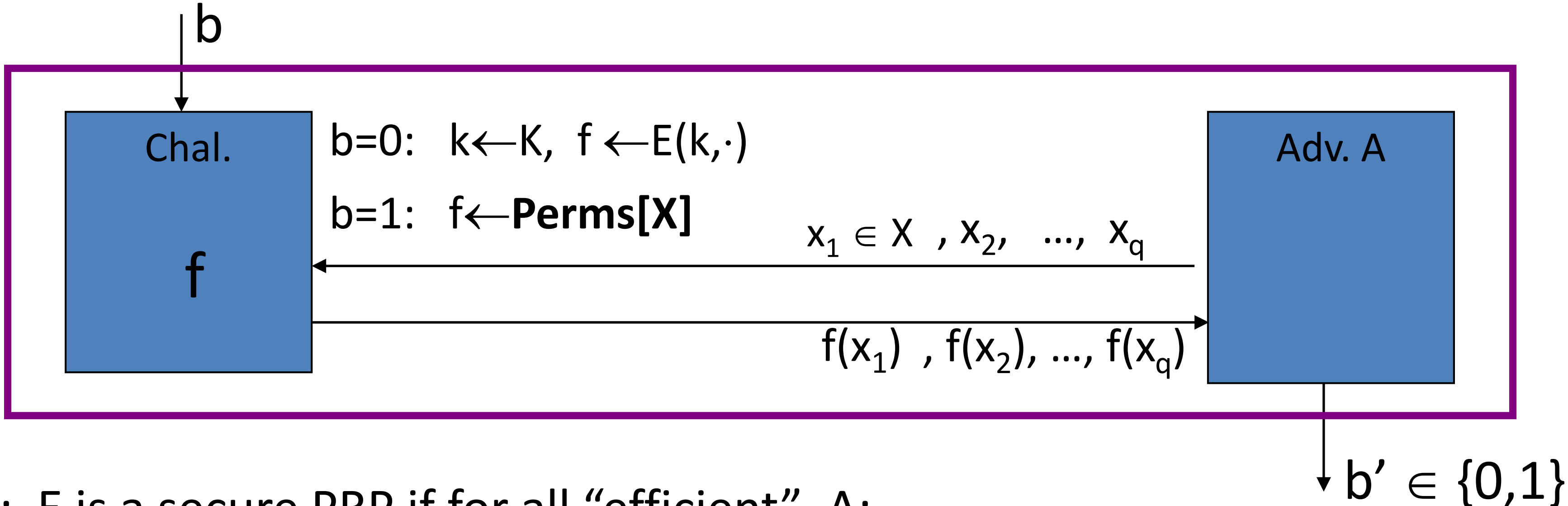
Def: F is a secure PRF if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{PRF}}[A,F] := \left|\Pr[EXP(0)=1] - \Pr[EXP(1)=1]\right|$$

is "negligible."

Secure PRPs (secure block cipher)

For b=0,1 define experiment EXP(b) as:



Def: E is a secure PRP if for all "efficient" A:

$$\mathrm{Adv}_{\mathrm{PRP}}[A,E] = \left|\Pr[\mathrm{EXP}(0)=1] - \Pr[\mathrm{EXP}(1)=1]\right|$$

is "negligible."

Example Secure PRPs

PRPs believed to be secure: 3DES, AES, …

AES-128: $K \times X \rightarrow X$ where $K = X = \{0,1\}^{128}$

An example concrete assumption about AES:

All 2^{80}–time algs. A have $\mathrm{Adv}_{\mathrm{PRP}}[A, \mathbf{AES}] < 2^{-40}$

Final Note

Suggestion:

- don’ t think about the inner-workings of AES and 3DES.

We assume both are secure PRPs and will see how to use them

Security for Many-Time Key

Example applications:

1. File systems: Same AES key used to encrypt many files.
2. IPsec: Same AES key used to encrypt many packets.

Key used more than once $\Rightarrow$ adv. sees many CTs with same key

Adversary's power: chosen-plaintext attack (CPA)

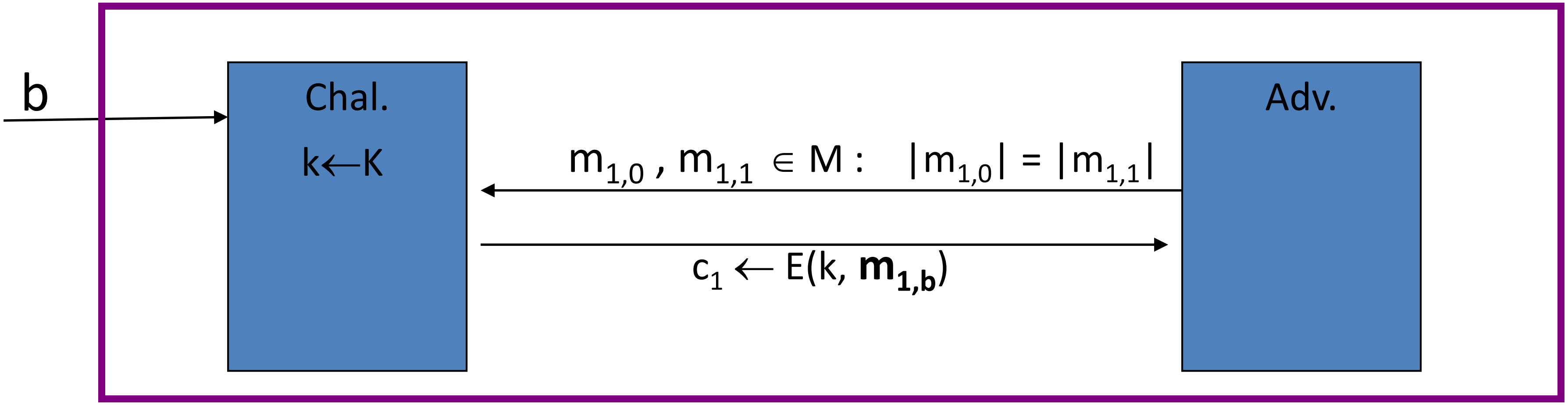
Can obtain the encryption of arbitrary messages of his choice

(conservative modeling of real life)

Adversary's goal: Break sematic security

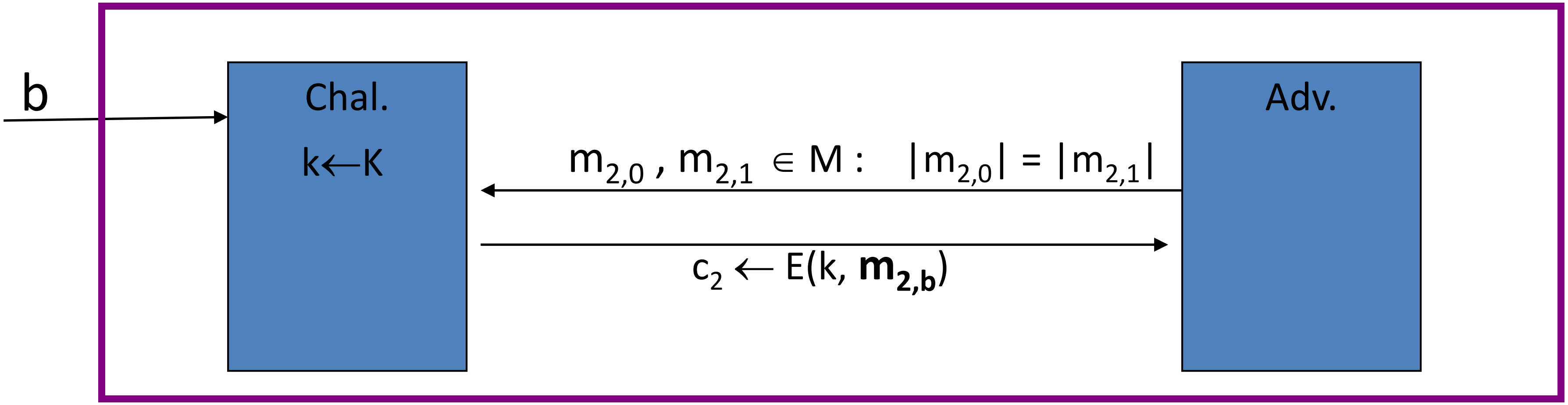
Semantic Security for Many-Time Key

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



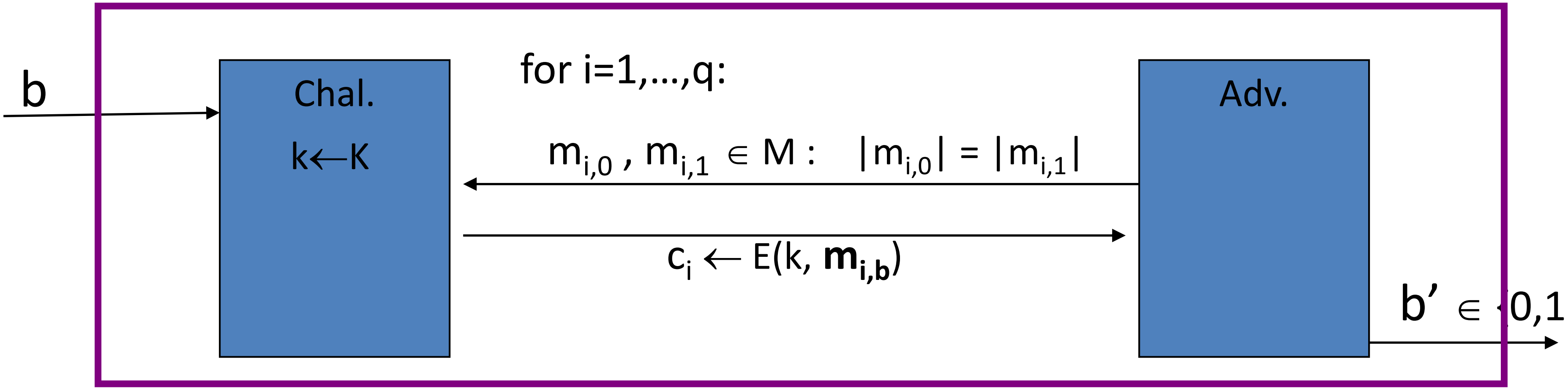
Semantic Security for Many-Time Key

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



Semantic Security for Many-Time Key (CPA security)

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



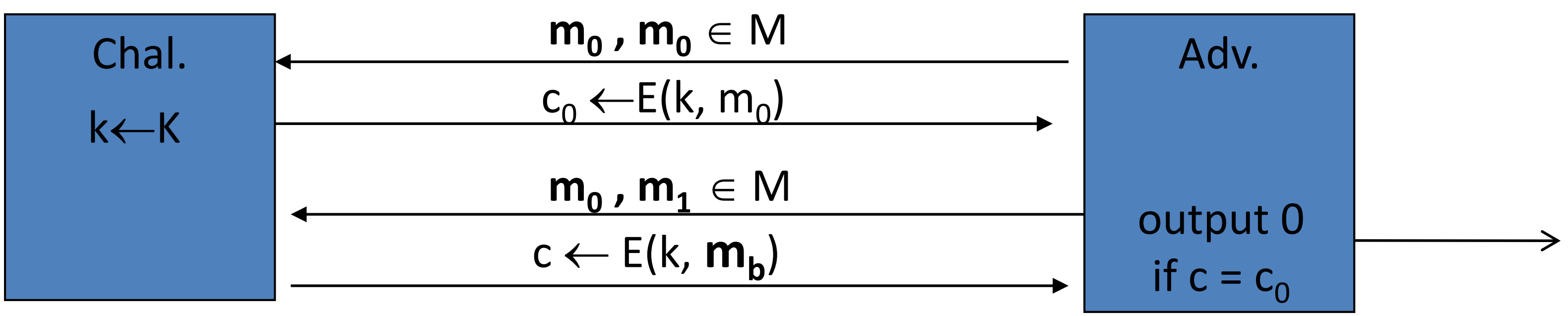
if adv. wants c = E(k, m) it queries with $m_{j,0} = m_{j,1} = m$

Def: $\mathbb{E}$ is sem. sec. under CPA if for all "efficient" A:

$\text{Adv}_{\text{CPA}}[A,\mathbb{E}] = \big|\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]\big|$ is "negligible."

Ciphers insecure under CPA

Suppose E(k,m) always outputs same ciphertext for msg m.   Then:

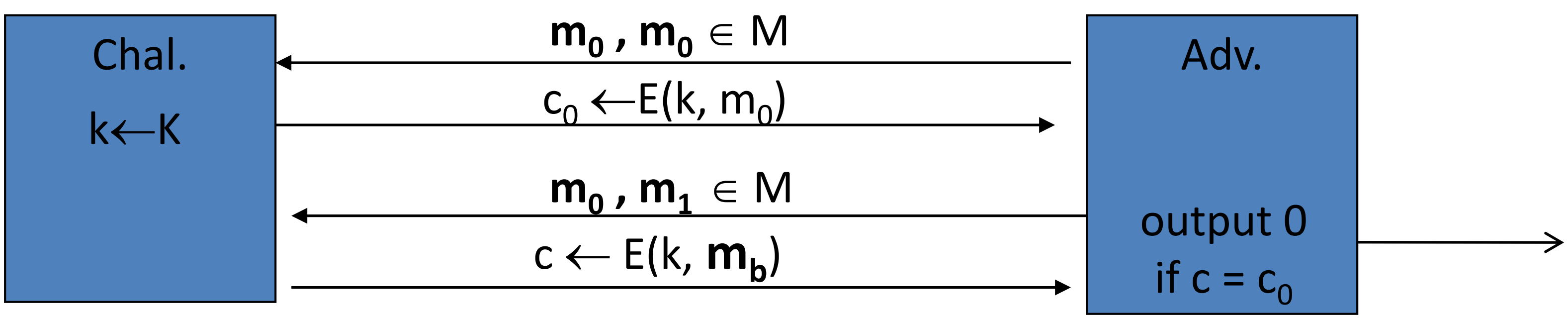


So what?   an attacker can learn that two encrypted files are the same, two encrypted packets are the same, etc.

Leads to significant attacks when message space M is small

Ciphers insecure under CPA

Suppose E(k,m) always outputs same ciphertext for msg m.   Then:

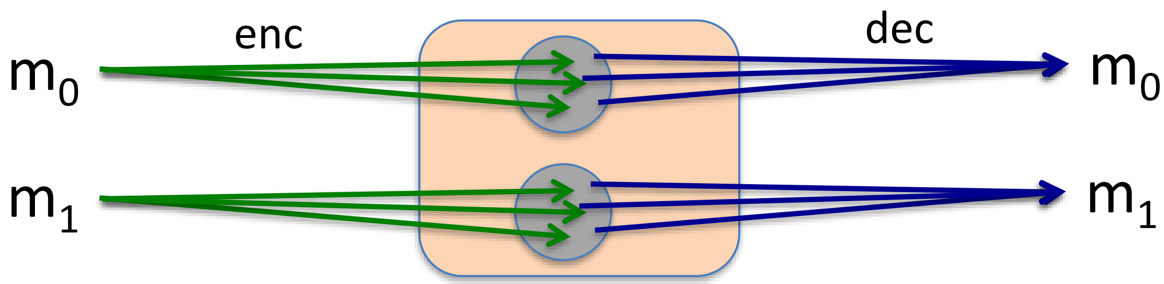


If secret key is to be used multiple times $\Rightarrow$

given the same plaintext message twice,
encryption must produce different outputs.

Solution 1: Randomized Encryption

E(k,m) is a randomized algorithm:



⇒ encrypting same msg twice gives different ciphertexts (w.h.p)

⇒ ciphertext must be longer than plaintext

Roughly speaking: CT-size = PT-size + "# random bits"

Solution 1: Randomized Encryption

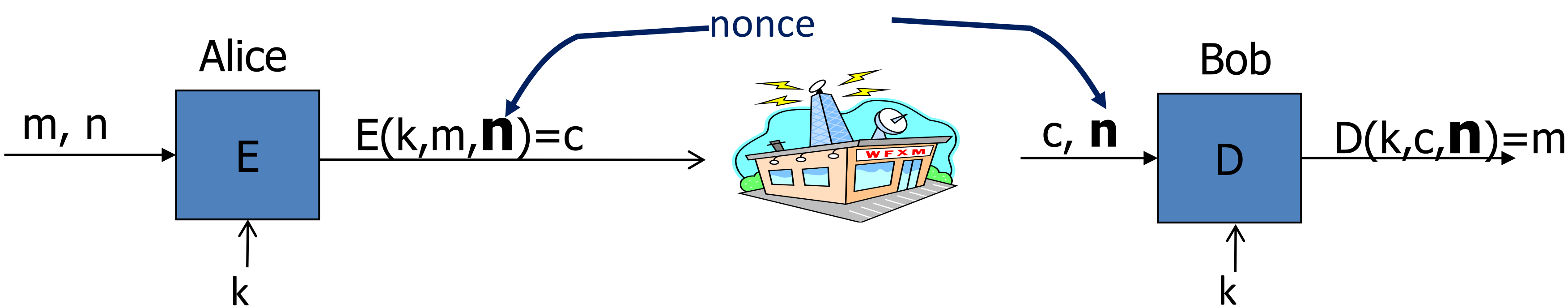
Let $F: K \times R \longrightarrow M$ be a secure PRF.

For $m \in M$ define $E(k,m) = [\ r \xleftarrow{R} R,\ \text{output}\ (r,\ F(k,r) \oplus m)\ ]$

Is E semantically secure under CPA?

- Yes, whenever F is a secure PRF
- No, there is always a CPA attack on this system
- → Yes, but only if R is large enough so r never repeats (w.h.p)
- It depends on what F is used

Solution 2: nonce-based Encryption



nonce n: a value that changes from msg to msg.

(k,n) pair <u>never</u> used more than once

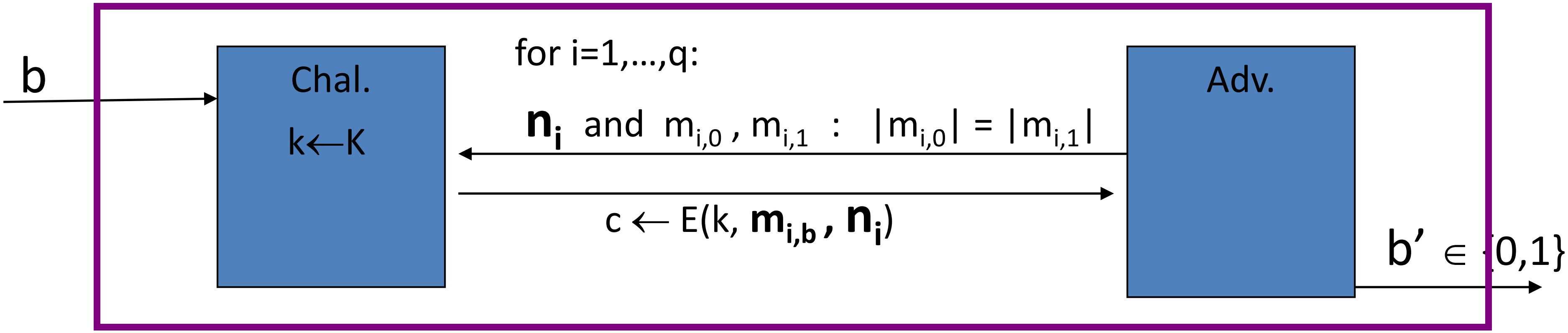
<u>method 1</u>: nonce is a **counter** (e.g. packet counter)

- used when encryptor keeps state from msg to msg
- if decryptor has same state, need not send nonce with CT

<u>method 2</u>: encryptor chooses a **random nonce**, $n \leftarrow \mathcal{N}$

Solution 2: nonce-based Encryption

System should be secure when nonces are chosen adversarially.



All nonces $\{n_1, \ldots, n_q\}$ must be distinct.

Def: nonce-based $\mathbb{E}$ is sem. sec. under CPA if for all "efficient" A:

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

Lecture 4.4: Message Integrity

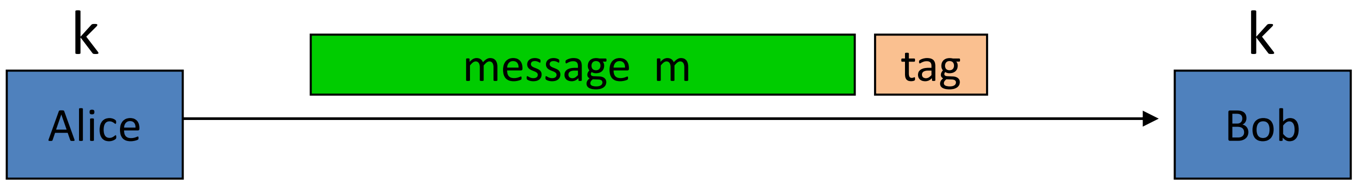
Message Integrity

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:

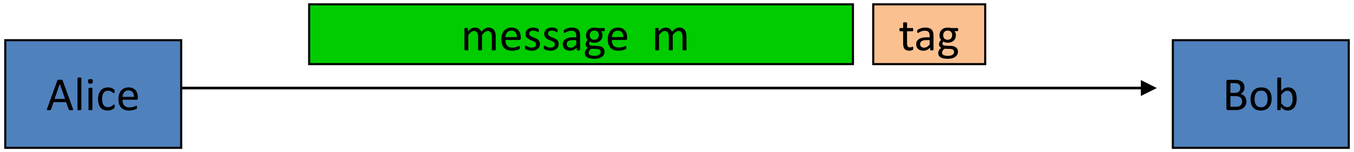
V(k, m, tag) $\stackrel{?}{=}$ `yes'

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 $\leftarrow$ 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).

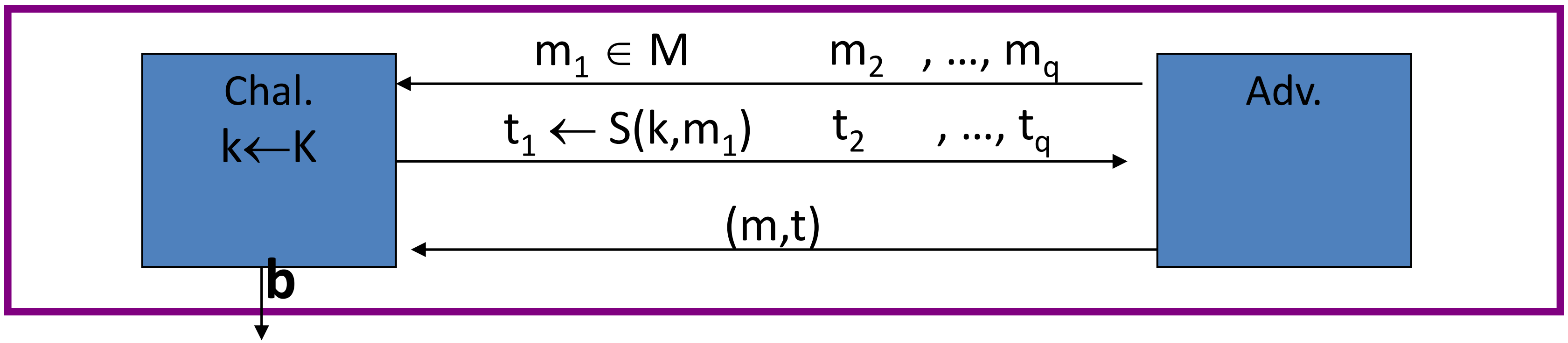
$$(m,t) \notin \{ (m_1,t_1), \ldots, (m_q,t_q) \}$$

$\Rightarrow$ 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."

Example

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

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

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

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

$Adv_{MAC}[A,I] = 1/2$

Example

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

$Adv_{MAC}[A,I] = 1/32$

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

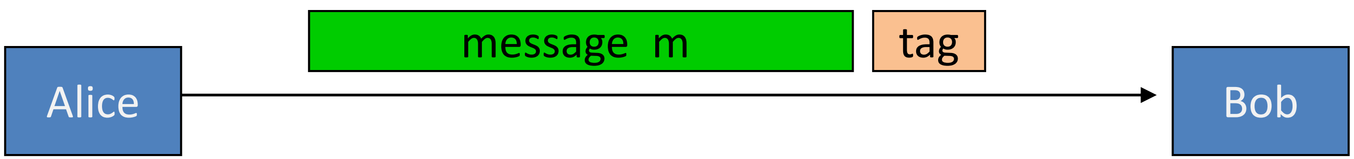
User reboots into clean OS and supplies his password

– Then: secure MAC $\Rightarrow$ all modified files will be detected

Secure PRF => 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 ← F(k,m)

accept msg if

tag = F(k,m)

A Bad Example

Suppose $\mathbf{F: K \times X \longrightarrow Y}$ is a secure PRF with $\mathbf{Y = \{0,1\}^{10}}$

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

$\mathrm{Adv}_{\mathrm{MAC}}[A,I] = 1/1024$

Security

Thm: If $\mathbf{F: K\times X \longrightarrow Y}$ 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.:

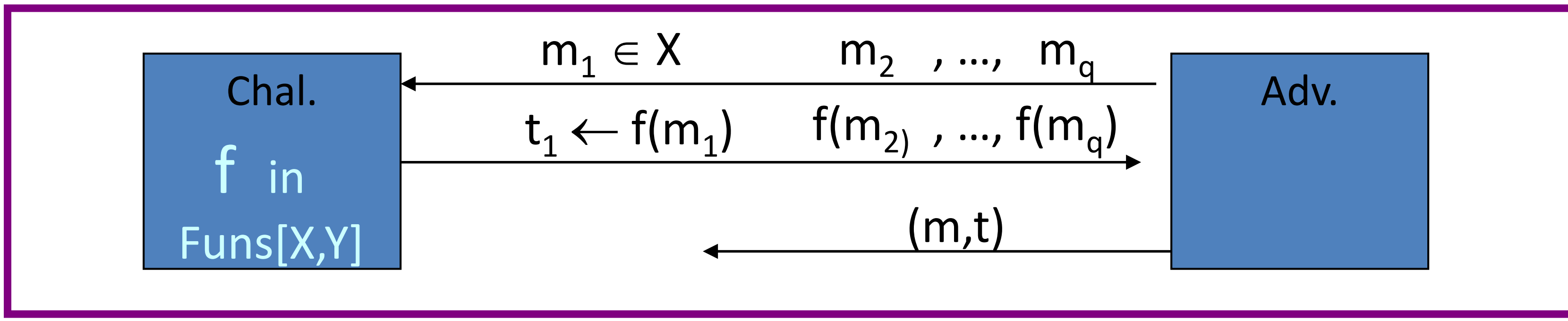
$$\mathrm{Adv}_{\mathrm{MAC}}[A, I_F] \leq \mathrm{Adv}_{\mathrm{PRF}}[B, F] + 1/|Y|$$

$\Rightarrow$ 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 $\mathbf{t = f(m)}$ and $\mathbf{m \notin \{ m_1 , \ldots , m_q \}}$

$\Rightarrow$ Pr[A 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 ?

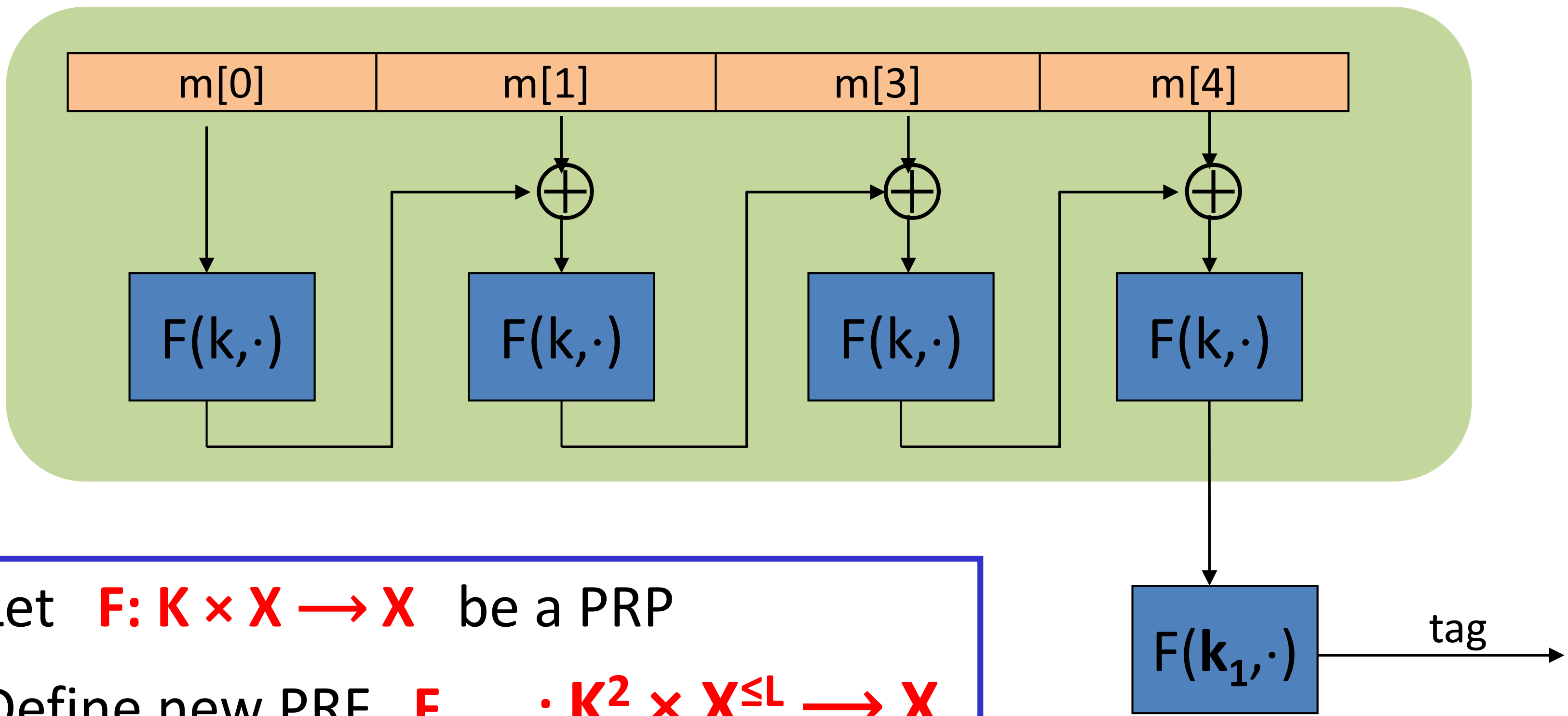
Two main constructions used in practice:

- **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.

Construction 1: encrypted CBC-MAC

raw CBC



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

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

Why the last encryption step in ECBC-MAC?

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

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

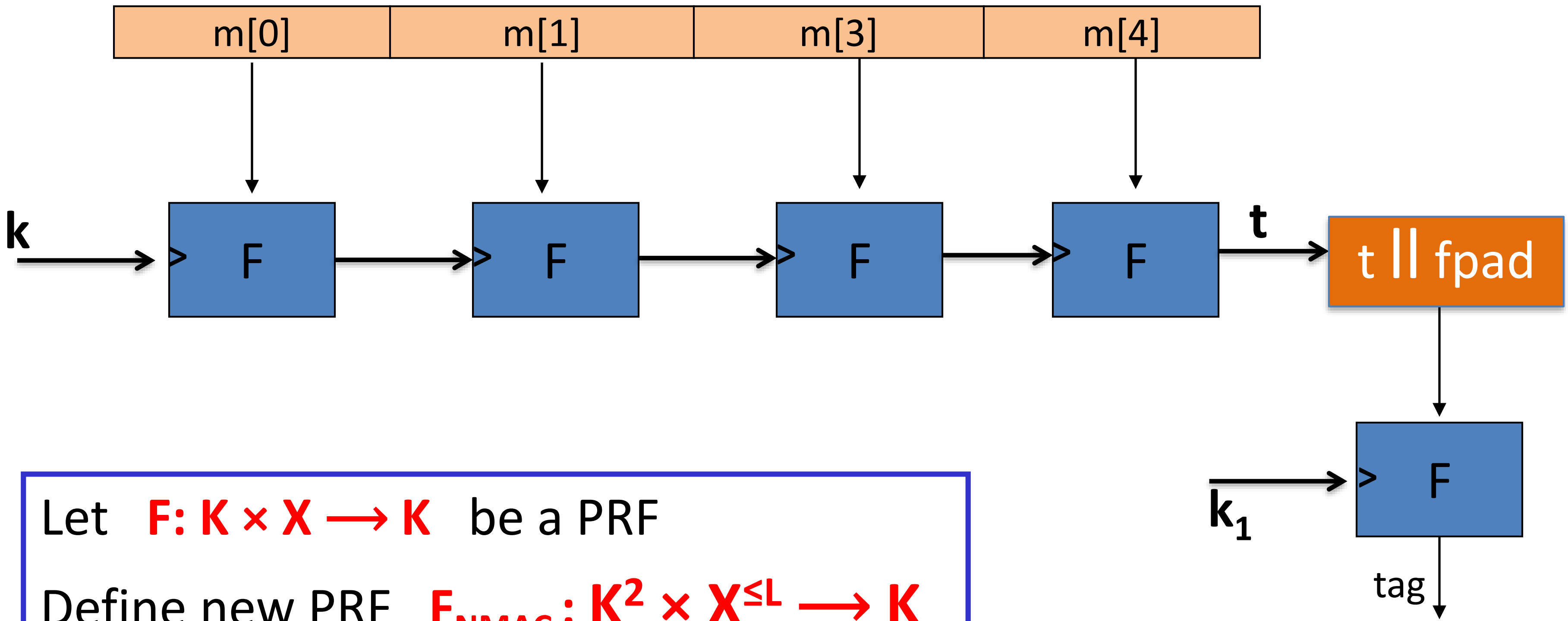
Then I_{RAW} is easily broken using a 1-chosen msg attack.

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

Construction 2: NMAC (nested MAC)



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

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

Truncating MACs based on PRFs

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

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

$\Rightarrow$ 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 $w \geq 64$)

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. $H(m_0) = H(m_1)$.

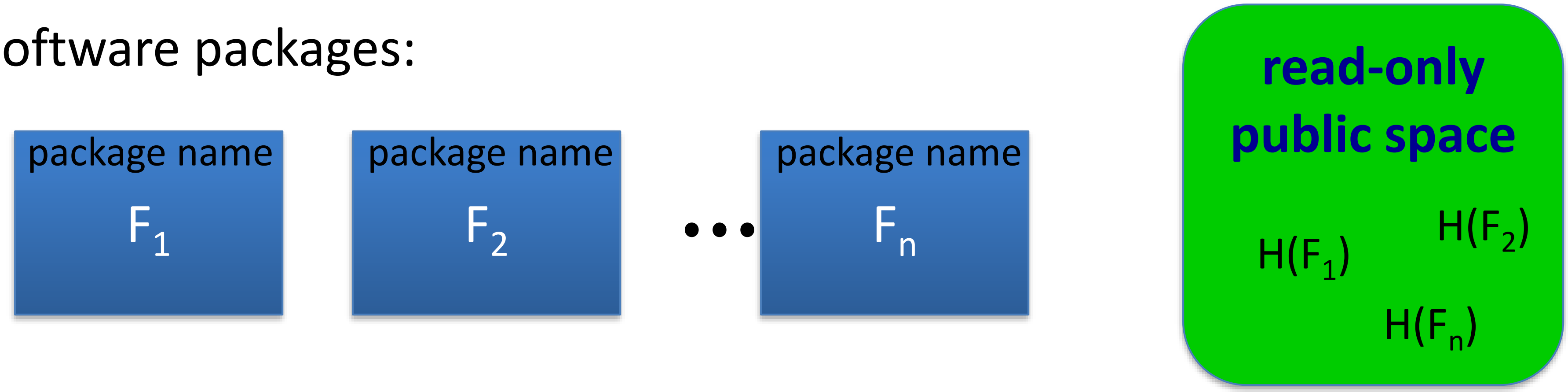
Then: S^{big} is insecure under a 1-chosen msg attack

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

step 2: output (m_1, t) as forgery

Protecting file integrity using Collision Resistance Hash

Software packages:



When user downloads package, can verify that contents are valid

H collision resistant $\Rightarrow$

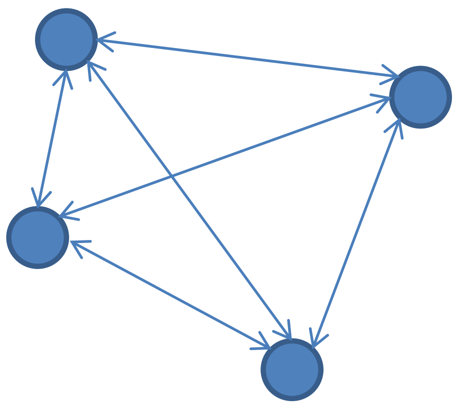
attacker cannot modify package without detection

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

Lecture 4.5: Basic Key Exachange

Key Management

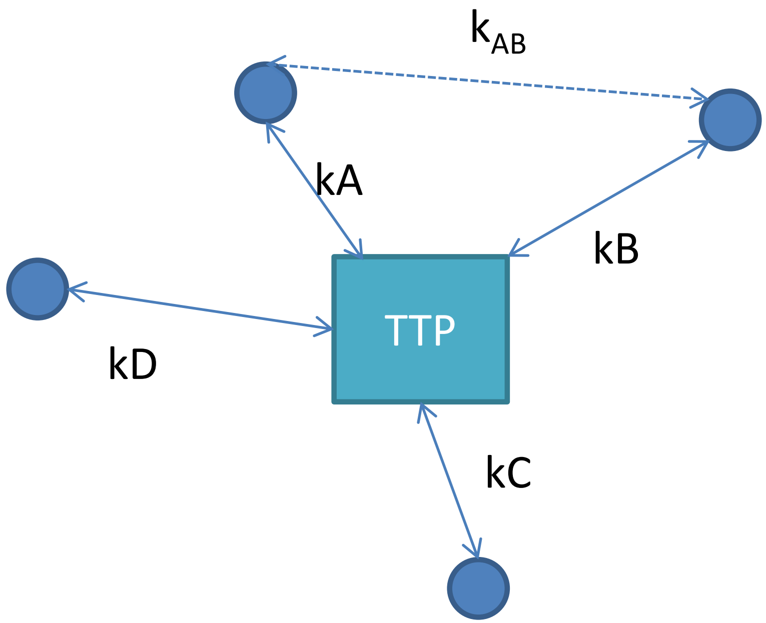
Problem: n users. Storing mutual secret keys is difficult



Total: O(n) keys per user

A Better Solution

Online Trusted 3rd Party (TTP)



Every use remembers one key

Generating Keys: A toy protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Bob **(k_B)**

Alice **(k_A)**

TTP

"Alice wants key with Bob"

choose random k_{AB}

$E(k_A, \text{"A,B"} \,||\, k_{AB})$

Ticket <- $E(k_B, \text{"A,B"} \,||\, k_{AB})$

ticket

k_{AB}

k_{AB}

(E,D) a CPA-secure cipher

Generating Keys: A Toy Protocol

Alice wants a shared key with Bob. Eavesdropping security only.

Eavesdropper sees: $E(k_A,\ \text{“A, B”} \parallel k_{AB})$; $E(k_B,\ \text{“A, B”} \parallel k_{AB})$

(E,D) is CPA-secure $\Rightarrow$

eavesdropper learns nothing about k_{AB}

Note: TTP needed for every key exchange, knows all session keys.

(basis of Kerberos system)

Key Question

Can we generate shared keys without an **online** trusted 3rd party?

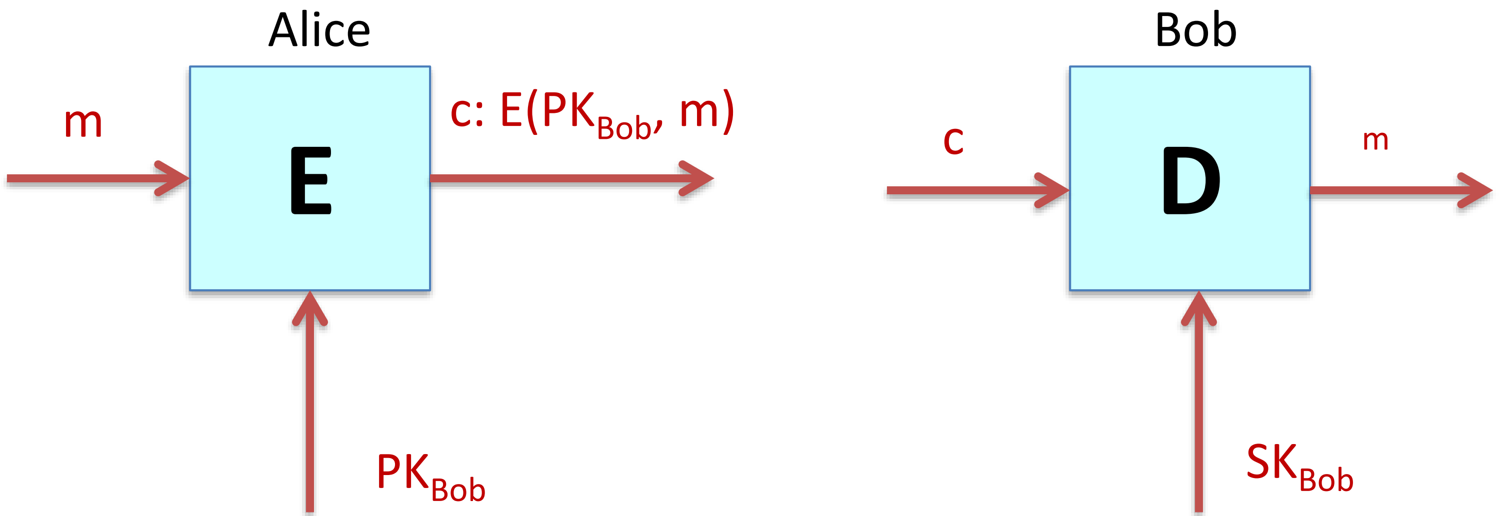
Answer: yes!

Starting point of public-key cryptography:

Merkle (1974), Diffie-Hellman (1976), RSA (1977)

More recently: ID-based enc. (BF 2001), Functional enc. (BSW 2011)

Public Key Encryption



PK: Public Key SK: Secret Key