



Introduction

Course Overview

Welcome

Course objectives:

- Learn how crypto primitives work
- Learn how to use them correctly and reason about security

My recommendations:

- Take notes
- Pause video frequently to think about the material
- Answer the in-video questions

Cryptography is everywhere

Secure communication:

- web traffic: HTTPS
- wireless traffic: 802.11i WPA2 (and WEP), GSM, Bluetooth

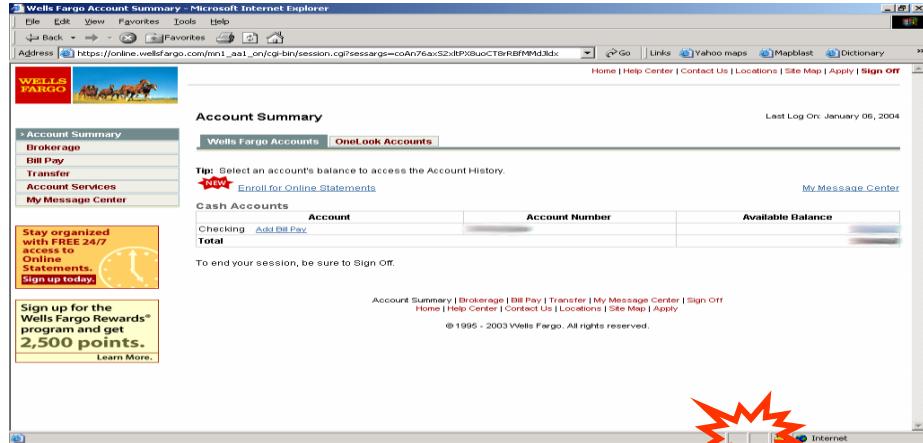
Encrypting files on disk: EFS, TrueCrypt

Content protection (e.g. DVD, Blu-ray): CSS, AACS

User authentication

... and much much more

Secure communication



HTTPS



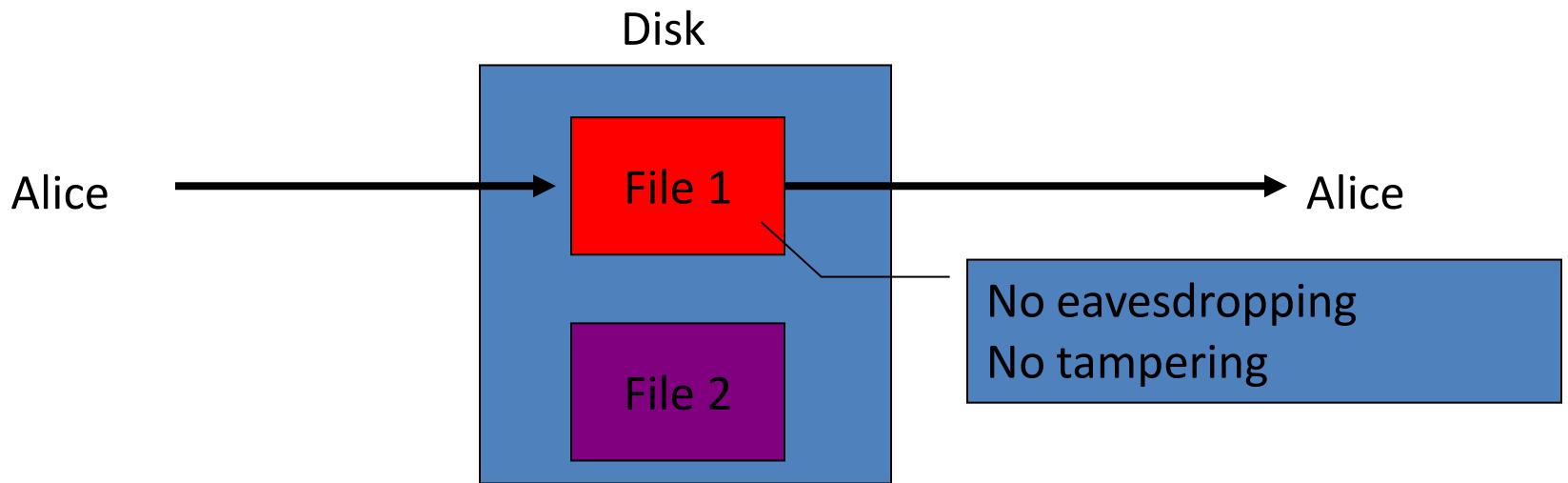
no eavesdropping
no tampering

Secure Sockets Layer / TLS

Two main parts

1. Handshake Protocol: **Establish shared secret key using public-key cryptography** (2nd part of course)
2. Record Layer: **Transmit data using shared secret key**
Ensure confidentiality and integrity (1st part of course)

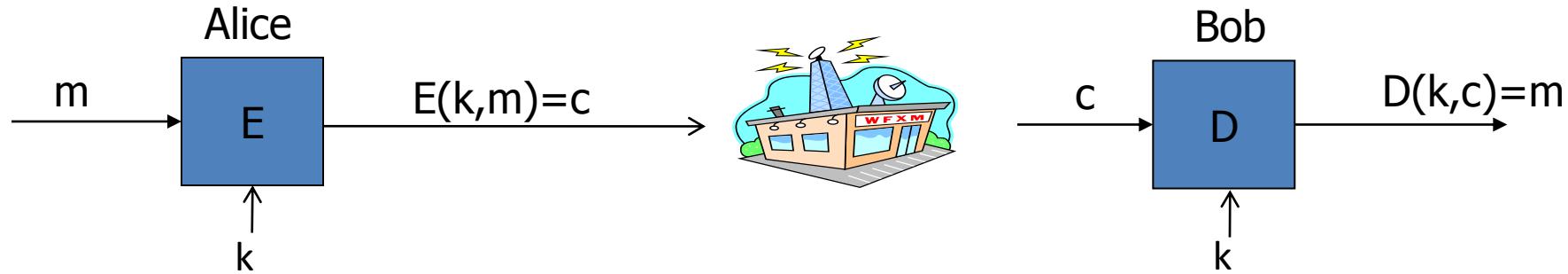
Protected files on disk



Analogous to secure communication:

Alice today sends a message to Alice tomorrow

Building block: sym. encryption



E, D : cipher k : secret key (e.g. 128 bits)

m, c : plaintext, ciphertext

Encryption algorithm is **publicly known**

- Never use a proprietary cipher

Use Cases

Single use key: (one time key)

- Key is only used to encrypt one message
 - encrypted email: new key generated for every email

Multi use key: (many time key)

- Key used to encrypt multiple messages
 - encrypted files: same key used to encrypt many files
- Need more machinery than for one-time key

Things to remember

Cryptography is:

- A tremendous tool
- The basis for many security mechanisms

Cryptography is not:

- The solution to all security problems
- Reliable unless implemented and used properly
- Something you should try to invent yourself
 - many many examples of broken ad-hoc designs

End of Segment

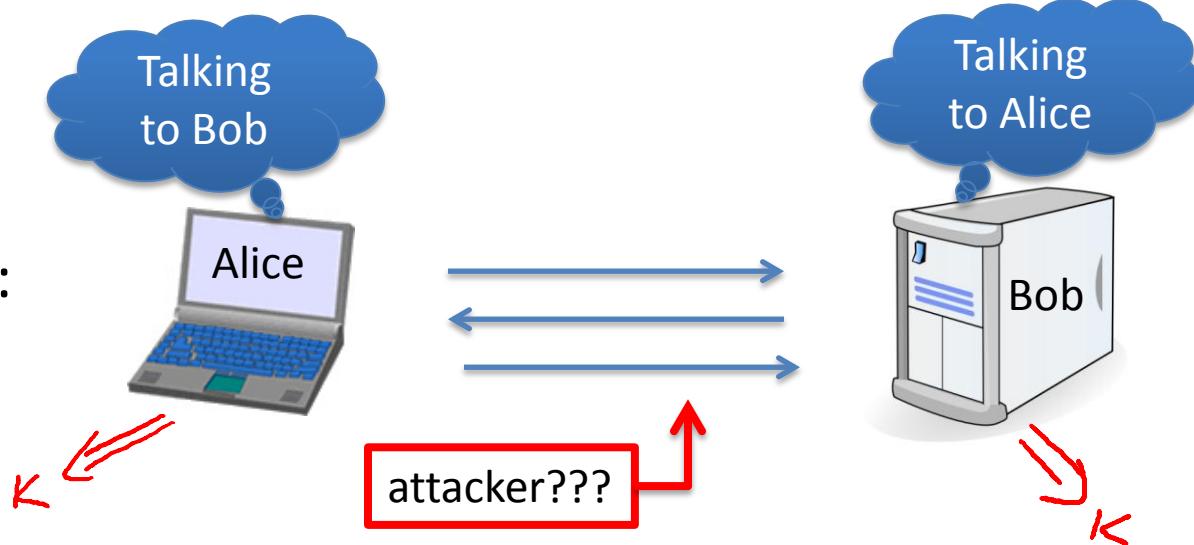


Introduction

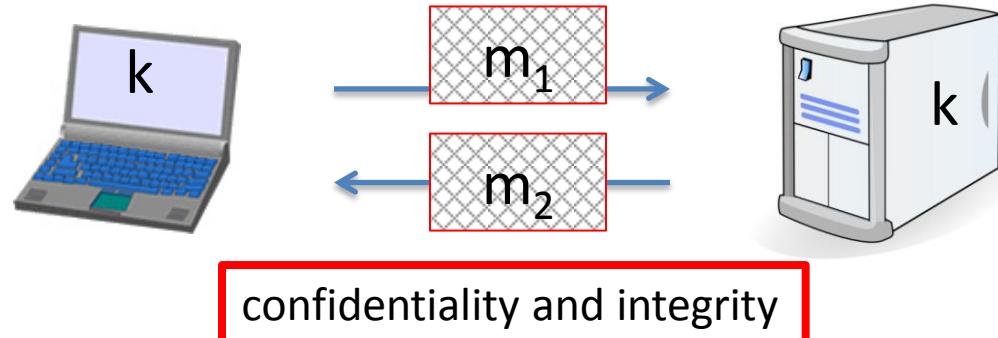
What is cryptography?

Crypto core

Secret key establishment:

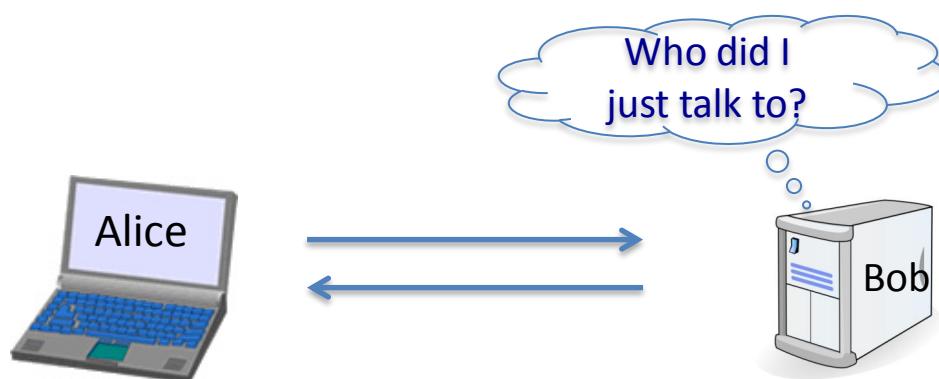


Secure communication:



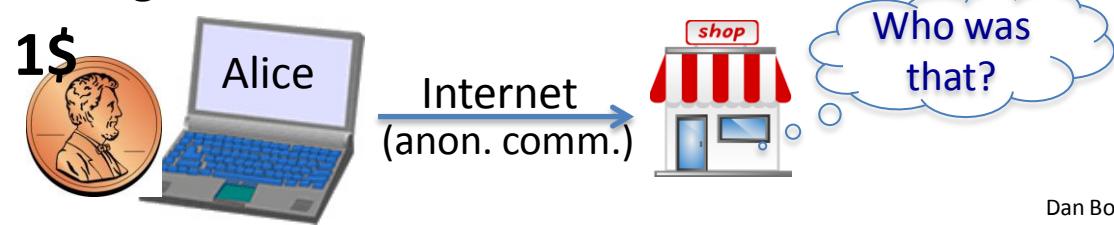
But crypto can do much more

- Digital signatures



But crypto can do much more

- Digital signatures
- Anonymous communication
- Anonymous **digital cash**
 - Can I spend a “digital coin” without anyone knowing who I am?
 - How to prevent double spending?

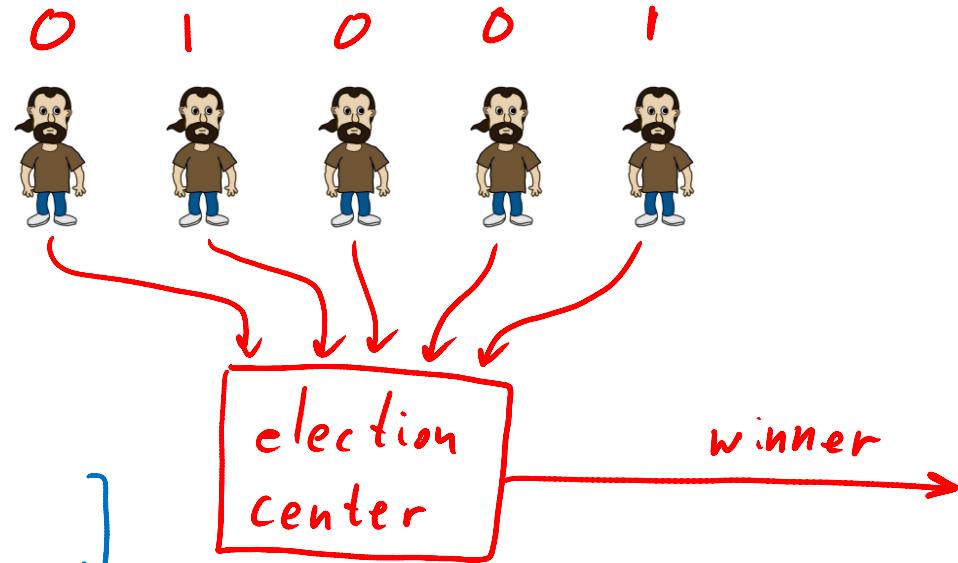


Protocols

- Elections
- Private auctions

winner = MAJ [votes]

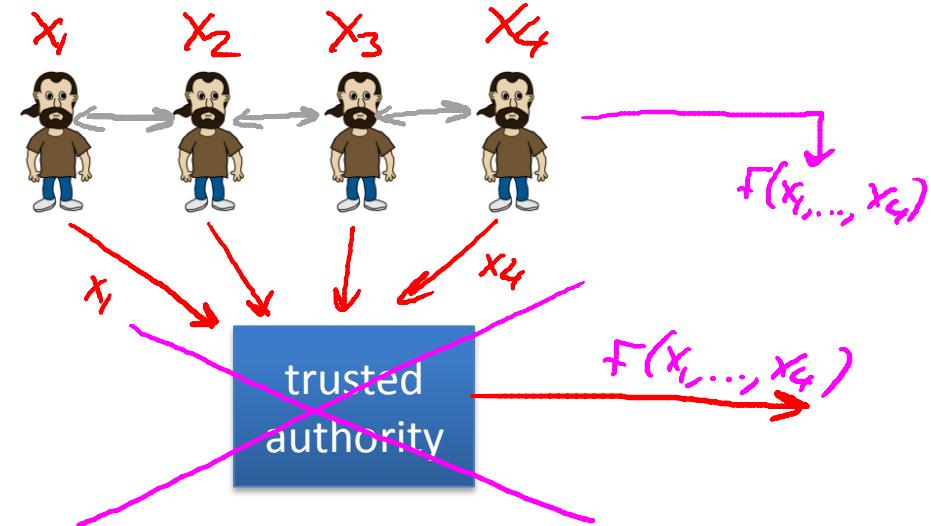
auction
winner = $\begin{bmatrix} \text{highest bidder,} \\ \text{pays 2nd highest bid} \end{bmatrix}$



Protocols

- Elections
- Private auctions

Goal: compute $f(x_1, x_2, x_3, x_4)$

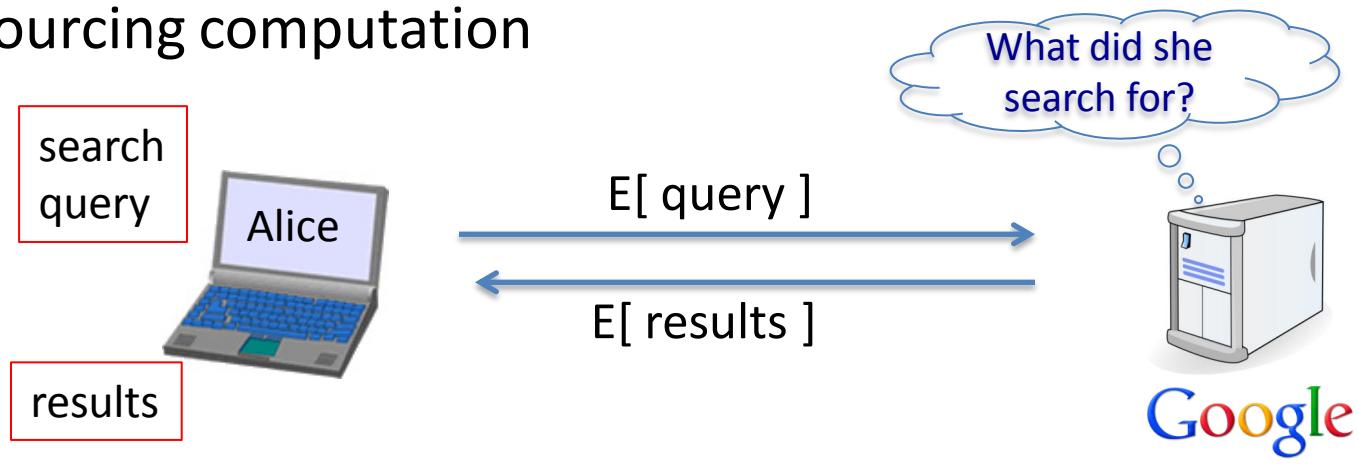


“Thm:” anything that can be done with trusted auth. can also be done without

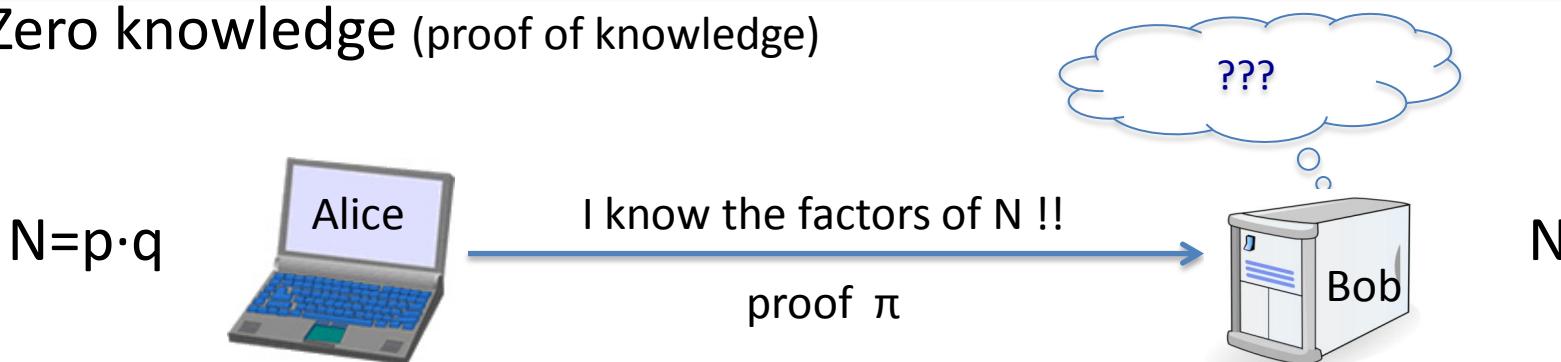
- Secure multi-party computation

Crypto magic

- Privately outsourcing computation



- Zero knowledge (proof of knowledge)



A rigorous science

The three steps in cryptography:

- Precisely specify threat model
- Propose a construction
- Prove that breaking construction under threat mode will solve an underlying hard problem

End of Segment

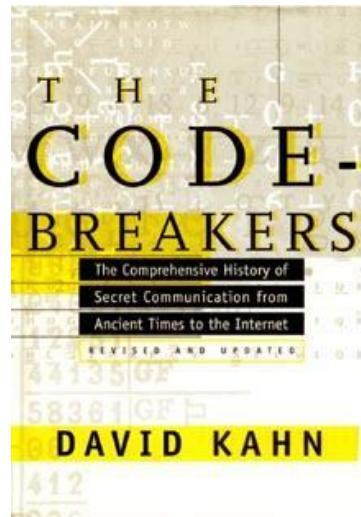


Introduction

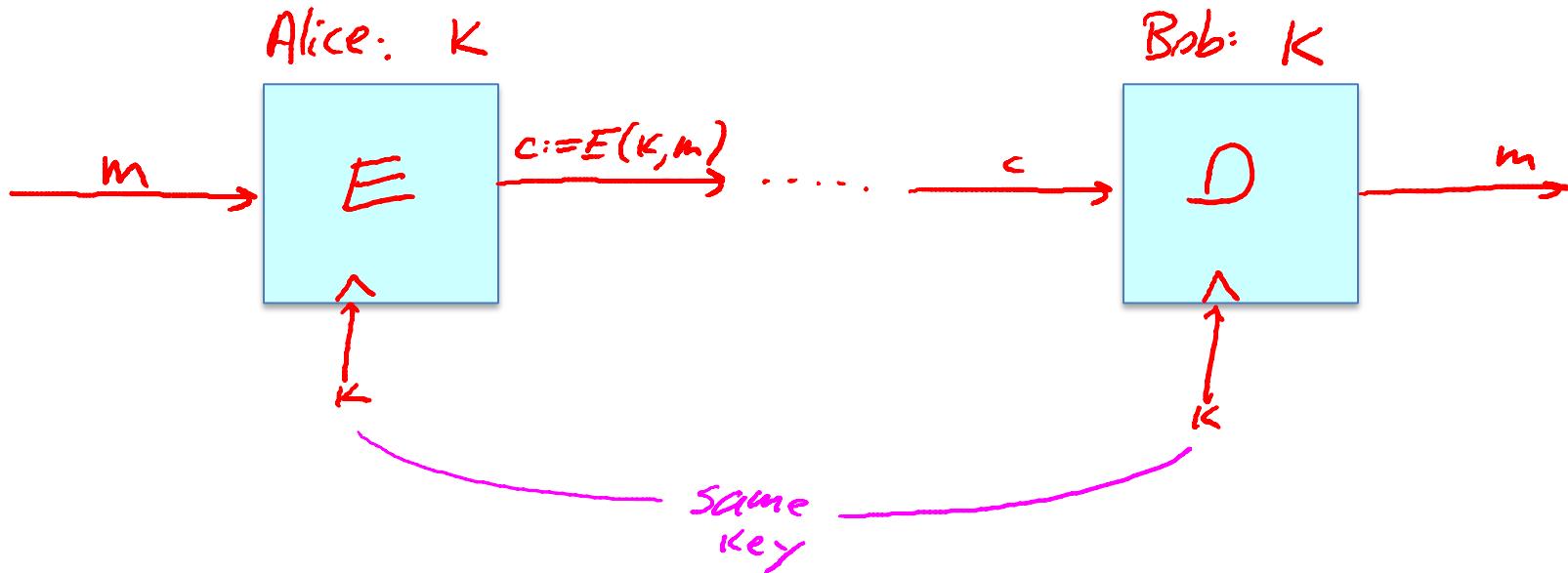
History

History

David Kahn, “The code breakers” (1996)



Symmetric Ciphers



Few Historic Examples (all badly broken)

1. Substitution cipher

$$c := E(k, "bcza") = "whac"$$

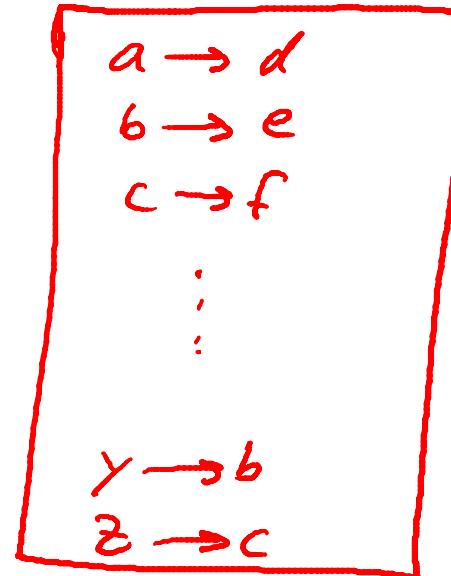
$$D(k, c) = "bcza"$$

$k :=$

$a \rightarrow c$
 $b \rightarrow w$
 $c \rightarrow n$
⋮
 $z \rightarrow a$

Caesar Cipher (no key)

shift by 3 :



What is the size of key space in the substitution cipher assuming 26 letters?

$$|\mathcal{K}| = 26$$

$$|\mathcal{K}| = 26! \quad (26 \text{ factorial})$$



$$|\mathcal{K}| = 2^{26}$$

$$26! \approx 2^{88}$$

$$|\mathcal{K}| = 26^2$$

How to break a substitution cipher?

What is the most common letter in English text?

“X”

“L”

“E” 

“H”

How to break a substitution cipher?

- (1) Use frequency of English letters

"e": 12.7% , "t": 9.1% , "a": 8.1%

- (2) Use frequency of pairs of letters (digrams)

"he", "an", "in", "th"

⇒ CT only attack !!

An Example

UKBYBIPOUZBCUFEEBORUKBYBHOBBRFESPVBWFOPERVNBCVBZPRUBOOPERVNBCVBPCYYFVUFO
FEIKNWFRFIKJNUPWRIFIPOUNVNIPUBRNCUKBEFWWF DNCHXCYBOHOPYXPUBNCUBOYNRVNIWN
CPOJIOFHOPZRVFZIXUBORJRUBZRBCHNCBBONCHRJZSFVNVRJRUBZRPCYZPUKBZPUNVPWPCYVF
ZIXUPUNFCPWRVNBCVBRPYYNUNFCPWWJUKBYBIPOUZBCUIPOUNVNIPUBRNCHOPYXPUBNCUB
OYNRVNIWCPOJIOFHOPZRNCRVNBCUNENVVFZIXUNCHPCYVFZIXUPUNFCPWZPUKBZPUNVR

B	36
N	34
U	33
P	32
C	26

→ E

→ T

→ A

NC	11
PU	10
UB	10
UN	9

digrams

→ IN

→ AT

UKB	6
RVN	6
FZI	4

trigrams

→ THE

2. Vigener cipher (16'th century, Rome)

k = **C R Y P T O C R Y P T O C R Y P T**
(+ mod 26)

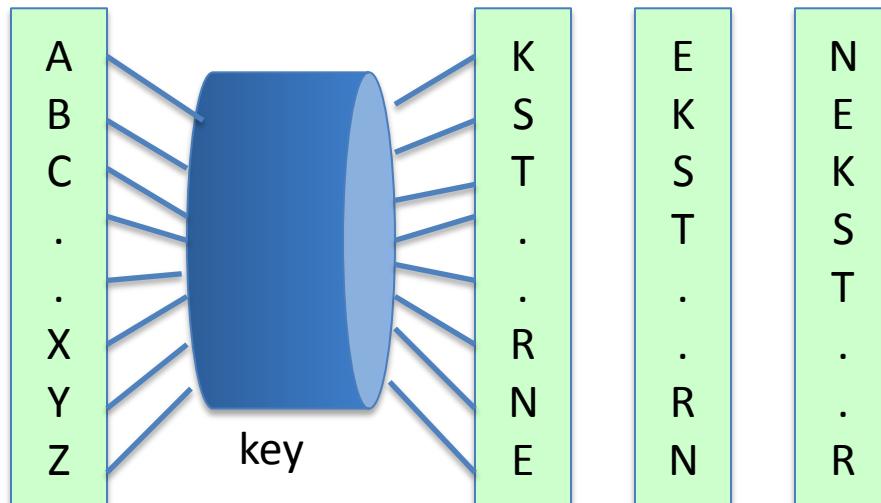
m = W H A T A N I C E D A Y T O D A Y

c = z z z j u c | l u d t u n | w g c q s
↑ ↑ ↑

suppose most common = "H" → first letter of key = "H" – "E" = "C"

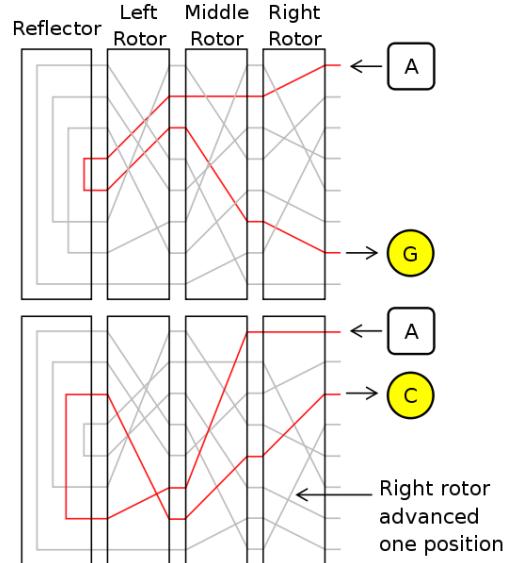
3. Rotor Machines (1870-1943)

Early example: the Hebern machine (single rotor)



Rotor Machines (cont.)

Most famous: the Enigma (3-5 rotors)



$$\# \text{ keys} = 26^4 = 2^{18} \quad (\text{actually } 2^{36} \text{ due to plugboard})$$

4. Data Encryption Standard (1974)

DES: # keys = 2^{56} , block size = 64 bits

Today: AES (2001), Salsa20 (2008) (and many others)

End of Segment

See also: http://en.wikibooks.org/High_School_Mathematics_Extensions/Discrete_Probability



Introduction

Discrete Probability (crash course, cont.)

U : finite set (e.g. $U = \{0,1\}^n$)

Def: **Probability distribution P** over U is a function $P: U \rightarrow [0,1]$

such that $\sum_{x \in U} P(x) = 1$

Examples:

1. Uniform distribution: for all $x \in U$: $P(x) = 1/|U|$
2. Point distribution at x_0 : $P(x_0) = 1$, $\forall x \neq x_0$: $P(x) = 0$

Distribution vector: $(P(000), P(001), P(010), \dots, P(111))$

Events

- For a set $A \subseteq U$: $\Pr[A] = \sum_{x \in A} P(x) \in [0,1]$ note: $\Pr[U]=1$
- The set A is called an **event**

Example: $U = \{0,1\}^8$

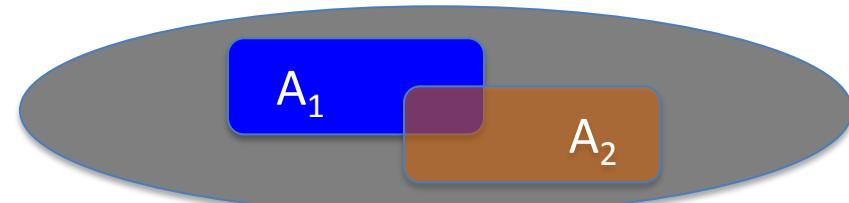
- $A = \{ \text{ all } x \text{ in } U \text{ such that } \text{lsb}_2(x)=11 \} \subseteq U$
for the uniform distribution on $\{0,1\}^8$: $\Pr[A] = 1/4$

The union bound

- For events A_1 and A_2

$$\Pr[A_1 \cup A_2] \leq \Pr[A_1] + \Pr[A_2]$$

$$A_1 \cap A_2 = \emptyset \Rightarrow \Pr[A_1 \cup A_2] = \Pr[A_1] + \Pr[A_2]$$



Example:

$$A_1 = \{ \text{ all } x \text{ in } \{0,1\}^n \text{ s.t. } \text{lsb}_2(x)=11 \} \quad ; \quad A_2 = \{ \text{ all } x \text{ in } \{0,1\}^n \text{ s.t. } \text{msb}_2(x)=11 \}$$

$$\Pr[\text{lsb}_2(x)=11 \text{ or } \text{msb}_2(x)=11] = \Pr[A_1 \cup A_2] \leq \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

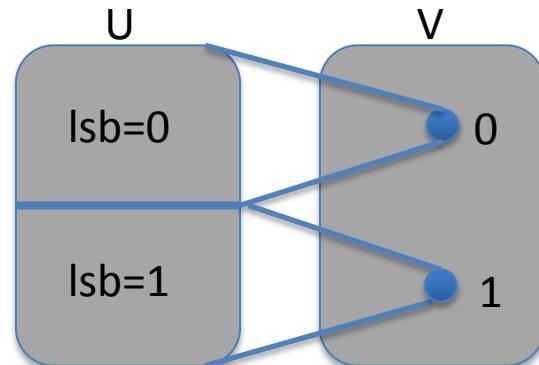
Random Variables

Def: a random variable X is a function $X:U \rightarrow V$

Example: $X: \{0,1\}^n \rightarrow \{0,1\}$; $X(y) = \text{lsb}(y) \in \{0,1\}$

For the uniform distribution on U :

$$\Pr[X=0] = 1/2, \quad \Pr[X=1] = 1/2$$



More generally:

rand. var. X induces a distribution on V : $\Pr[X=v] := \Pr[X^{-1}(v)]$

The uniform random variable

Let U be some set, e.g. $U = \{0,1\}^n$

We write $r \leftarrow^R U$ to denote a uniform random variable over U

for all $a \in U$: $\Pr[r = a] = 1/|U|$

(formally, r is the identity function: $r(x) = x$ for all $x \in U$)

Let r be a uniform random variable on $\{0,1\}^2$

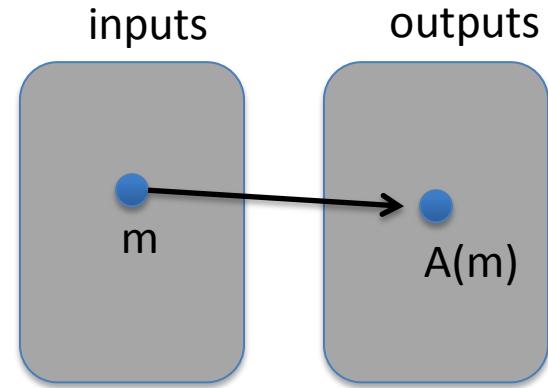
Define the random variable $X = r_1 + r_2$

Then $\Pr[X=2] = \frac{1}{4}$

Hint: $\Pr[X=2] = \Pr[r=11]$

Randomized algorithms

- Deterministic algorithm: $y \leftarrow A(m)$

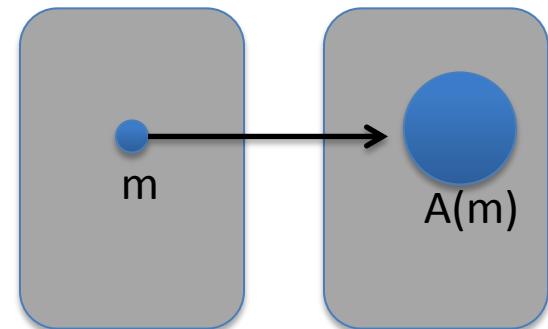


- Randomized algorithm

$$y \leftarrow A(m; r) \quad \text{where } r \xleftarrow{R} \{0,1\}^n$$

output is a random variable

$$y \xleftarrow{R} A(m)$$



Example: $A(m; k) = E(k, m)$, $y \xleftarrow{R} A(m)$

End of Segment

See also: http://en.wikibooks.org/High_School_Mathematics_Extensions/Discrete_Probability



Introduction

Discrete Probability (crash course, cont.)

Recap

U : finite set (e.g. $U = \{0,1\}^n$)

Prob. distr. P over U is a function $P: U \rightarrow [0,1]$ s.t. $\sum_{x \in U} P(x) = 1$

$A \subseteq U$ is called an **event** and $\Pr[A] = \sum_{x \in A} P(x) \in [0,1]$

A **random variable** is a function $X: U \rightarrow V$.

X takes values in V and defines a distribution on V

Independence

Def: events A and B are **independent** if $\Pr[A \text{ and } B] = \Pr[A] \cdot \Pr[B]$

random variables X, Y taking values in V are **independent** if

$$\forall a, b \in V: \Pr[X=a \text{ and } Y=b] = \Pr[X=a] \cdot \Pr[Y=b]$$

Example: $U = \{0,1\}^2 = \{00, 01, 10, 11\}$ and $r \xleftarrow{R} U$

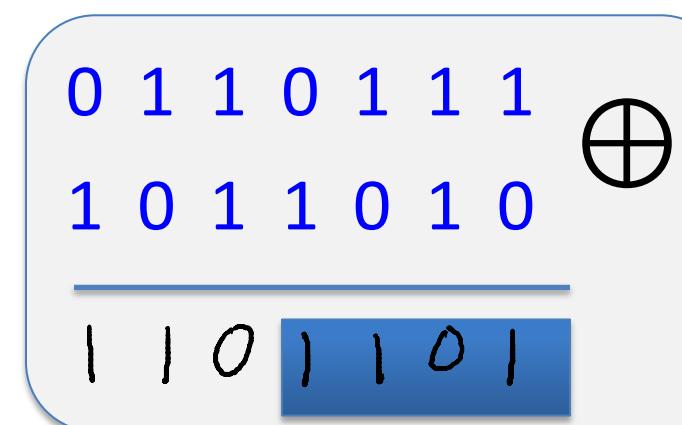
Define r.v. X and Y as: $X = \text{lsb}(r)$, $Y = \text{msb}(r)$

$$\Pr[X=0 \text{ and } Y=0] = \Pr[r=00] = \frac{1}{4} = \Pr[X=0] \cdot \Pr[Y=0]$$

Review: XOR

XOR of two strings in $\{0,1\}^n$ is their bit-wise addition mod 2

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0



An important property of XOR

Thm: Y a rand. var. over $\{0,1\}^n$, X an indep. uniform var. on $\{0,1\}^n$

Then $Z := Y \oplus X$ is uniform var. on $\{0,1\}^n$

Proof: (for $n=1$)

$$\Pr[Z=0] = \Pr[(x,y)=(0,0) \text{ or } (x,y)=(1,1)] =$$

$$= \Pr[(x,y)=(0,0)] + \Pr[(x,y)=(1,1)] =$$

$$= \frac{p_0}{2} + \frac{p_1}{2} = \frac{1}{2}$$

Y	\Pr
0	p_0
1	p_1

X	\Pr
0	$1/2$
1	$1/2$

x	y	\Pr
0	0	$p_0/2$
0	1	$p_1/2$
1	0	$p_0/2$
1	1	$p_1/2$



The birthday paradox

Let $r_1, \dots, r_n \in U$ be indep. identically distributed random vars.

Thm: when $n = 1.2 \times |U|^{1/2}$ then $\Pr[\exists i \neq j: r_i = r_j] \geq \frac{1}{2}$

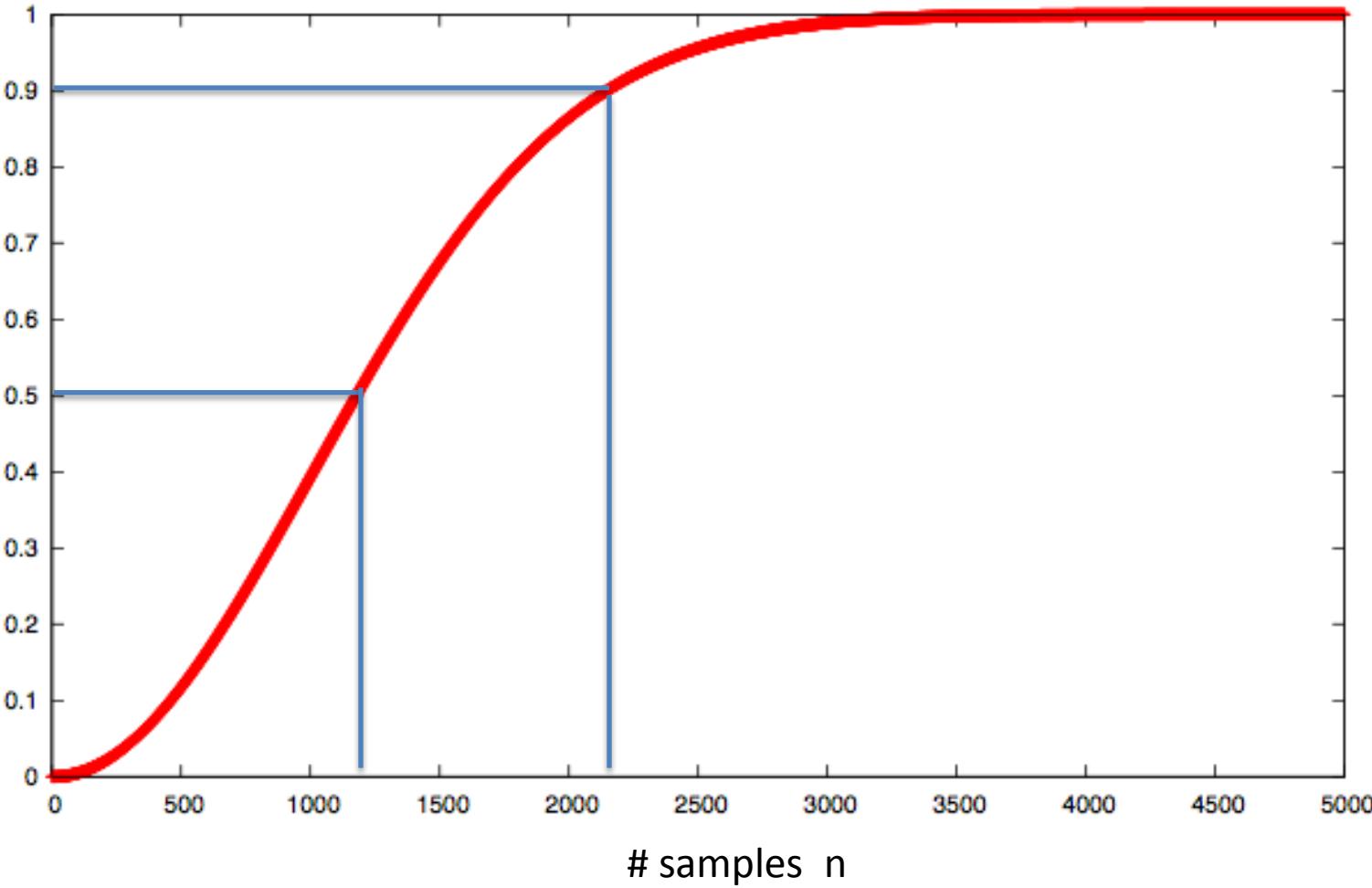
notation: $|U|$ is the size of U

Example: Let $U = \{0,1\}^{128}$

After sampling about 2^{64} random messages from U ,
some two sampled messages will likely be the same

$|U|=10^6$

collision probability



End of Segment



Stream ciphers

The One Time Pad

Symmetric Ciphers: definition

Def: a **cipher** defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$

is a pair of “efficient” algs (E, D) where

$$E: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C} \quad , \quad D: \mathcal{K} \times \mathcal{C} \rightarrow \mathcal{M}$$

$$\text{s.t. } \forall m \in \mathcal{M}, k \in \mathcal{K}: D(k, E(k, m)) = m$$

- E is often randomized. D is always deterministic.

The One Time Pad

(Vernam 1917)

First example of a “secure” cipher

$$\mathcal{M} = \mathcal{C} = \{0,1\}^n , \quad \mathcal{K} = \{0,1\}^n$$

key = (random bit string as long the message)

The One Time Pad

(Vernam 1917)

$$C := E(K, m) = K \oplus m$$

$$D(K, C) = K \oplus C$$

msg: 0 1 1 0 1 1 1

key: 1 0 1 1 0 1 0

CT:



Indeed:

$$D(K, E(K, m)) = D(K, K \oplus m) = K \oplus (K \oplus m) = (K \oplus K) \oplus m = 0 \oplus m = m$$

You are given a message (m) and its OTP encryption (c).

Can you compute the OTP key from m and c ?

No, I cannot compute the key.

Yes, the key is $k = m \oplus c$. 

I can only compute half the bits of the key.

Yes, the key is $k = m \oplus m$.

The One Time Pad

(Vernam 1917)

Very fast enc/dec !!

... but long keys (as long as plaintext)

Is the OTP secure? What is a secure cipher?

What is a secure cipher?

Attacker's abilities: **CT only attack** (for now)

Possible security requirements:

attempt #1: **attacker cannot recover secret key**

$$E(k, m) = m \quad \text{would be secure}$$

attempt #2: **attacker cannot recover all of plaintext**

$$E(k, m_0 \parallel m_1) = m_0 \parallel k \oplus m_1 \quad \text{would be secure}$$

Shannon's idea:

CT should reveal no “info” about PT

Information Theoretic Security

(Shannon 1949)

Def: A cipher (E, D) over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ has perfect secrecy if

$$\forall m_0, m_1 \in \mathcal{M} \quad (\text{len}(m_0) = \text{len}(m_1)) \quad \text{and} \quad \forall c \in \mathcal{C}$$

$$\Pr[E(k, m_0) = c] = \Pr[E(k, m_1) = c]$$

where k is uniform in \mathcal{K} ($k \leftarrow \mathcal{K}$)

Information Theoretic Security

Def: A cipher (E, D) over (K, M, C) has **perfect secrecy** if

$$\forall m_0, m_1 \in M \quad (|m_0| = |m_1|) \quad \text{and} \quad \forall c \in C$$

$$Pr[E(k, m_0) = c] = Pr[E(k, m_1) = c] \quad \text{where } k \leftarrow K$$

-
- ⇒ Given CT can't tell if msg is m_0 or m_1 (for all m_0, m_1)
 - ⇒ most powerful adv. learns nothing about PT from CT
 - ⇒ no CT only attack!! (but other attacks possible)

Lemma: OTP has perfect secrecy.

Proof:

$$\forall m, c: \Pr_{k \in \mathcal{K}} [E(k, m) = c] = \frac{\#\text{keys } k \in \mathcal{K} \text{ s.t. } E(k, m) = c}{|\mathcal{K}|}$$

So: if $\forall m, c: \#\{k \in \mathcal{K} : E(k, m) = c\} = \text{const.}$

\Rightarrow cipher has perfect secrecy

Let $m \in \mathcal{M}$ and $c \in \mathcal{C}$.

How many OTP keys map m to c ?

None

1 

2

Depends on m

Lemma: OTP has perfect secrecy.

Proof:

For OTP: $\forall m, c : \text{if } E(k, m) = c$

$$\Rightarrow k \oplus m = c \Rightarrow k = m \oplus c$$

$$\Rightarrow \boxed{\#\{k \in \mathbb{K} : E(k, m) = c\} = 1}$$

\Rightarrow OTP has perfect Secrecy 

The bad news ...

Thm: perfect secrecy $\Rightarrow |K| \geq |M|$

i.e. perfect secrecy \Rightarrow key-len \geq msg-len

\Rightarrow hard to use in practice !!

End of Segment



Stream ciphers

Pseudorandom
Generators

Review

Cipher over (K, M, C) : a pair of “efficient” algs (E, D) s.t.

$$\forall m \in M, k \in K: D(k, E(k, m)) = m$$

Weak ciphers: subs. cipher, Vigener, ...

A good cipher: **OTP** $M=C=K=\{0,1\}^n$

$$E(k, m) = k \oplus m , \quad D(k, c) = k \oplus c$$

Lemma: OTP has perfect secrecy (i.e. no CT only attacks)

Bad news: perfect-secrecy \Rightarrow key-len \geq msg-len

Stream Ciphers: making OTP practical

idea: replace “random” key by “pseudorandom” key

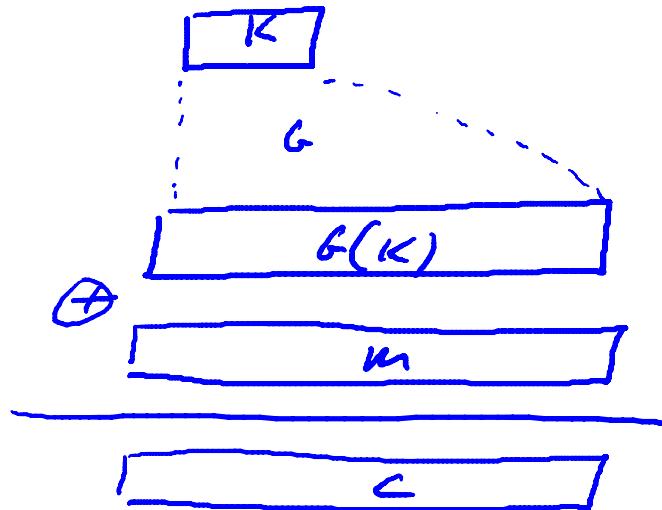
PRG is a function $g: \underbrace{\{0,1\}^s}_{\text{seed space}} \rightarrow \{0,1\}^n$ $n \gg s$

(eff. computable by a deterministic algorithm)

Stream Ciphers: making OTP practical

$$C := E(K, m) = m \oplus g(K)$$

$$D(K, c) = c \oplus g(K)$$



Can a stream cipher have perfect secrecy?

- Yes, if the PRG is really “secure”
- No, there are no ciphers with perfect secrecy
- Yes, every cipher has perfect secrecy
- No, since the key is shorter than the message 

Stream Ciphers: making OTP practical

Stream ciphers cannot have perfect secrecy !!

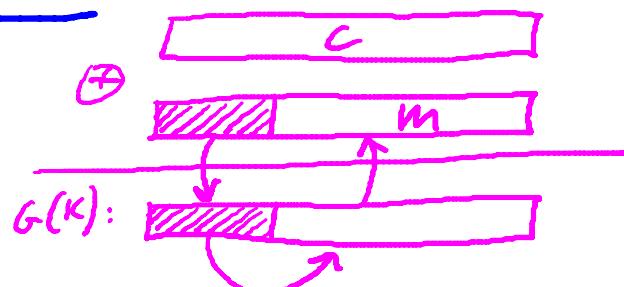
- Need a different definition of security
- Security will depend on specific PRG

PRG must be unpredictable

Suppose PRG is predictable :

$$\exists i: \left. g(k) \right|_{1, \dots, i} \xrightarrow{\text{alg}} \left. g(k) \right|_{i+1, \dots, n}$$

Then:



even $\left. g(k) \right|_{1, \dots, i} \rightarrow \left. g(k) \right|_{i+1}$
is a problem!

PRG must be unpredictable

We say that $G: K \rightarrow \{0,1\}^n$ is **predictable** if:

\exists "eff" alg. A and $\exists 0 \leq i \leq n-1$ s.t.

$$\Pr_{\substack{K \leftarrow g_K}} \left[A(G(K)) \Big|_{i, \dots, i} = G(K) \Big|_{i+1} \right] > \frac{1}{2} + \varepsilon$$

For non-negligible ε (e.g. $\varepsilon = 1/2^{30}$)

Def: PRG is **unpredictable** if it is not predictable

$\Rightarrow \forall i$: no "eff" adv. can predict bit $(i+1)$ for "non-neg" ε

Suppose $G:K \rightarrow \{0,1\}^n$ is such that for all k : $\text{XOR}(G(k)) = 1$

Is G predictable ??

Yes, given the first bit I can predict the second

No, G is unpredictable

Yes, given the first $(n-1)$ bits I can predict the n 'th bit 

It depends

Weak PRGs

(do not use for crypto)

Lin. Cong. generator with parameters a, b, p :

```
r[i] ← a · r[i-1] + b mod p  
output bits of r[i]  
i++
```

seed = r[0]

glibc random():

```
r[i] ← ( r[i-3] + r[i-31] ) % 232  
output r[i] >> 1
```

never use random()
for crypto !!
(e.g. Kerberos V4)

End of Segment



Stream ciphers

Negligible vs.
non-negligible

Negligible and non-negligible

- In practice: ϵ is a scalar and
 - ϵ non-neg: $\epsilon \geq 1/2^{30}$ (likely to happen over 1GB of data)
 - ϵ negligible: $\epsilon \leq 1/2^{80}$ (won't happen over life of key)
- In theory: ϵ is a function $\epsilon: \mathbb{Z}^{\geq 0} \rightarrow \mathbb{R}^{\geq 0}$ and
 - ϵ non-neg: $\exists d: \epsilon(\lambda) \geq 1/\lambda^d$ inf. often ($\epsilon \geq 1/\text{poly}$, for many λ)
 - ϵ negligible: $\forall d, \lambda \geq \lambda_d: \epsilon(\lambda) \leq 1/\lambda^d$ ($\epsilon \leq 1/\text{poly}$, for large λ)

Few Examples

$\varepsilon(\lambda) = 1/2^\lambda$: negligible

$\varepsilon(\lambda) = 1/\lambda^{1000}$: non-negligible

$$\varepsilon(\lambda) = \begin{cases} 1/2^\lambda & \text{for odd } \lambda \\ 1/\lambda^{1000} & \text{for even } \lambda \end{cases}$$

Negligible

Non-negligible



PRGs: the rigorous theory view

PRGs are “parameterized” by a security parameter λ

- PRG becomes “more secure” as λ increases

Seed lengths and output lengths grow with λ

For every $\lambda=1,2,3,\dots$ there is a different PRG G_λ :

$$G_\lambda : K_\lambda \rightarrow \{0,1\}^{n(\lambda)}$$

(in the lectures we will always ignore λ)

An example asymptotic definition

We say that $\mathbf{G}_\lambda : \mathcal{K}_\lambda \rightarrow \{0,1\}^{n(\lambda)}$ is predictable at position i if:

there exists a polynomial time (in λ) algorithm A s.t.

$$\Pr_{k \leftarrow \mathcal{K}_\lambda} [A(\lambda, \mathbf{G}_\lambda(k) \Big|_{1,\dots,i}) = \mathbf{G}_\lambda(k) \Big|_{i+1}] > 1/2 + \varepsilon(\lambda)$$

for some non-negligible function $\varepsilon(\lambda)$

End of Segment



Stream ciphers

Attacks on OTP and stream ciphers

Review

OTP: $E(k,m) = m \oplus k$, $D(k,c) = c \oplus k$

Making OTP practical using a PRG: $G: K \rightarrow \{0,1\}^n$

Stream cipher: $E(k,m) = m \oplus G(k)$, $D(k,c) = c \oplus G(k)$

Security: PRG must be unpredictable (better def in two segments)

Attack 1: two time pad is insecure !!

Never use stream cipher key more than once !!

$$C_1 \leftarrow m_1 \oplus \text{PRG}(k)$$

$$C_2 \leftarrow m_2 \oplus \text{PRG}(k)$$

Eavesdropper does:

$$C_1 \oplus C_2 \rightarrow$$



Enough redundancy in English and ASCII encoding that:

$$m_1 \oplus m_2 \rightarrow m_1, m_2$$

Real world examples

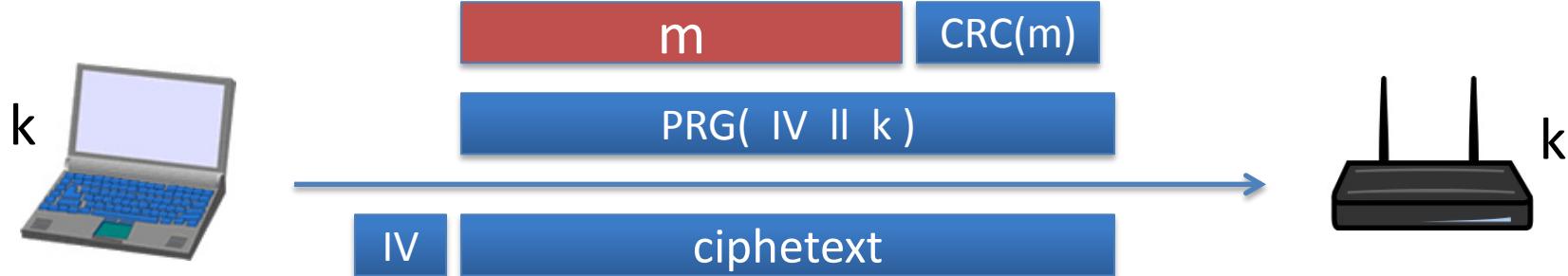
- Project Venona
- MS-PPTP (windows NT):



Need different keys for $C \rightarrow S$ and $S \rightarrow C$

Real world examples

802.11b WEP:

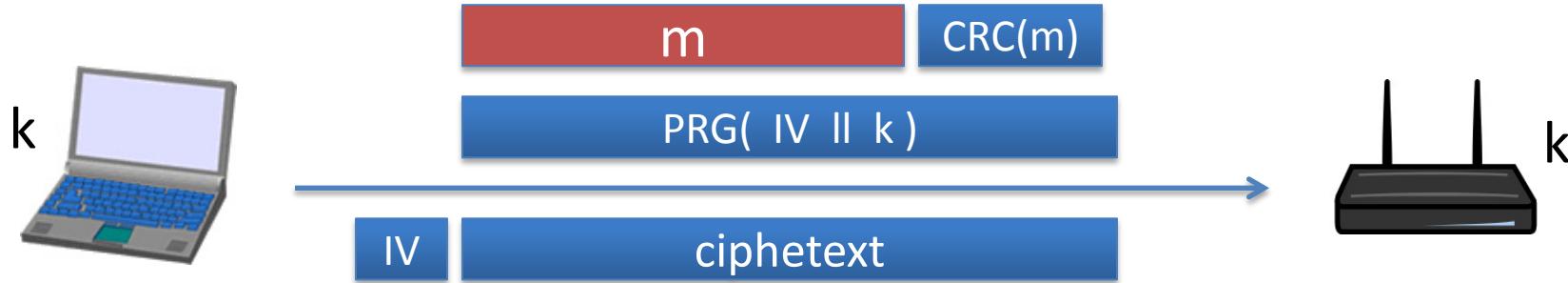


Length of IV: 24 bits

- Repeated IV after $2^{24} \approx 16M$ frames
- On some 802.11 cards: IV resets to 0 after power cycle

Avoid related keys

802.11b WEP:



key for frame #1: $(1 \parallel k)$

key for frame #2: $(2 \parallel k)$

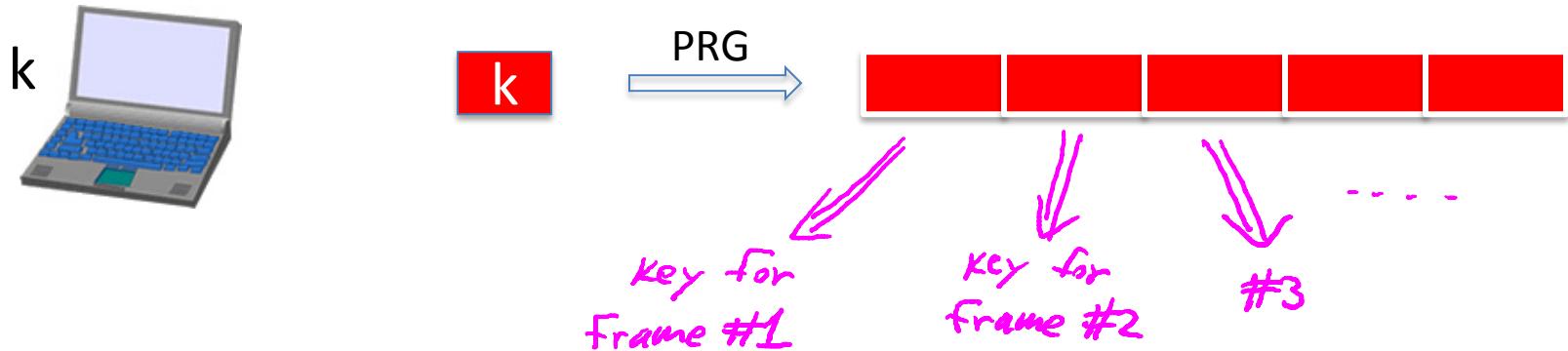
⋮ $\frac{24}{6}$ bits ↗ ↗ 1024 bits

For the RC4 PRG:

FMS2001 \Rightarrow can recover IC
after 10^6 frames

Recent attacks $\approx 40,000$ frames

A better construction

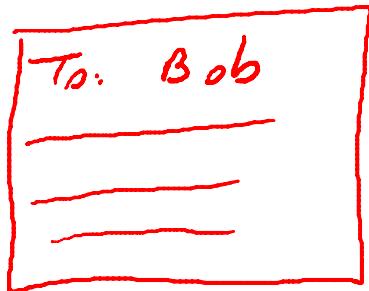


⇒ now each frame has a pseudorandom key

better solution: use stronger encryption method (as in WPA2)

Yet another example: disk encryption

(1)



enc. disk
→

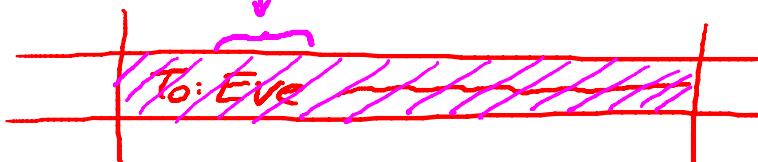


Later:

(2)



enc. disk
→

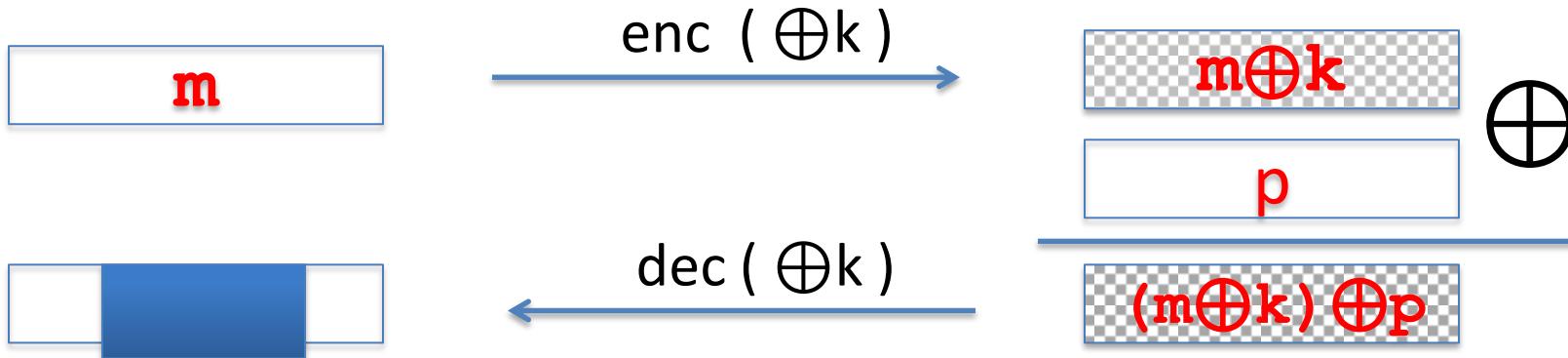


Two time pad: summary

Never use stream cipher key more than once !!

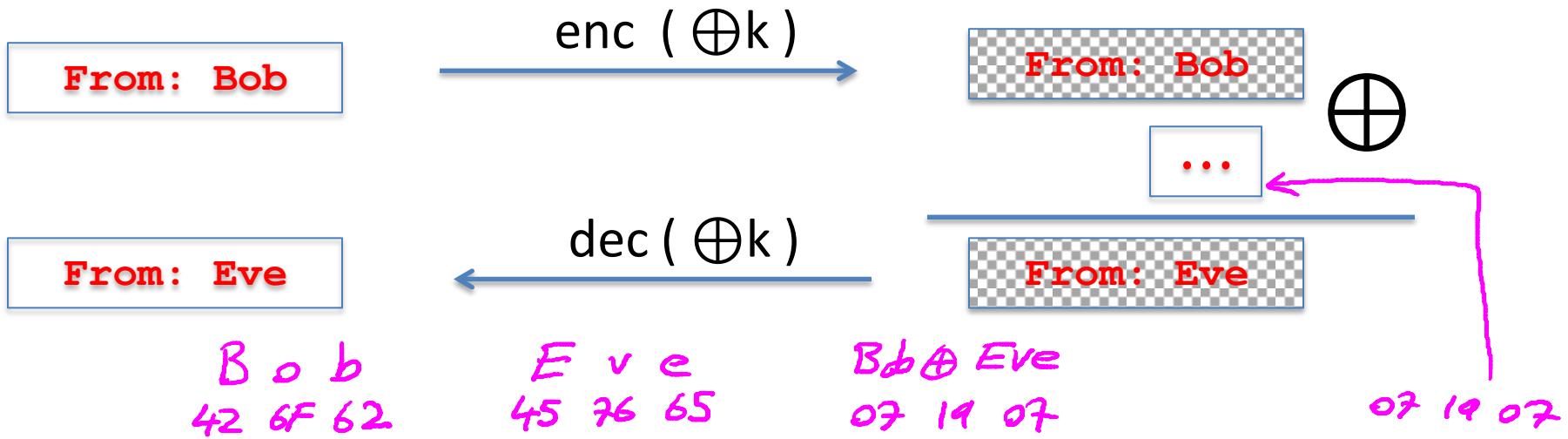
- Network traffic: negotiate new key for every session (e.g. TLS)
- Disk encryption: typically do not use a stream cipher

Attack 2: no integrity (OTP is malleable)



Modifications to ciphertext are undetected and have **predictable** impact on plaintext

Attack 2: no integrity (OTP is malleable)



Modifications to ciphertext are undetected and have predictable impact on plaintext

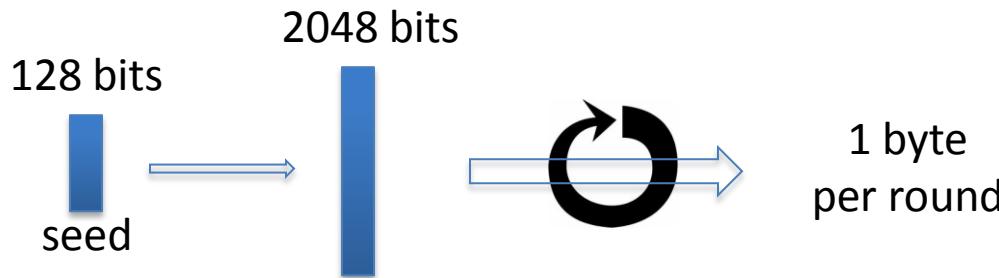
End of Segment



Stream ciphers

Real-world Stream Ciphers

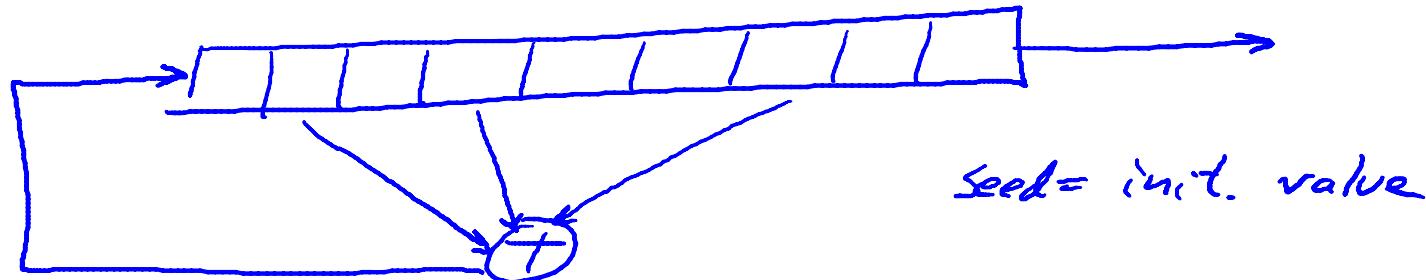
Old example (software): RC4 (1987)



- Used in HTTPS and WEP
- Weaknesses:
 1. Bias in initial output: $\Pr[\text{2}^{\text{nd}} \text{ byte} = 0] = 2/256$
 2. Prob. of (0,0) is $1/256^2 + 1/256^3$
 3. Related key attacks

Old example (hardware): CSS (badly broken)

Linear feedback shift register (LFSR):



DVD encryption (CSS): 2 LFSRs

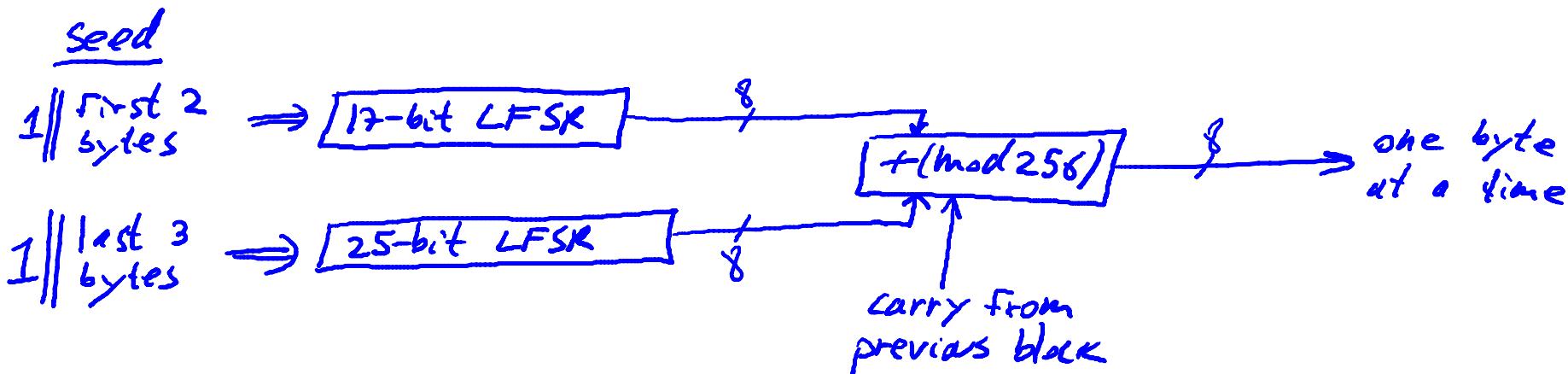
GSM encryption (A5/1,2): 3 LFSRs

Bluetooth (E0): 4 LFSRs

} all broken

Old example (hardware): CSS (badly broken)

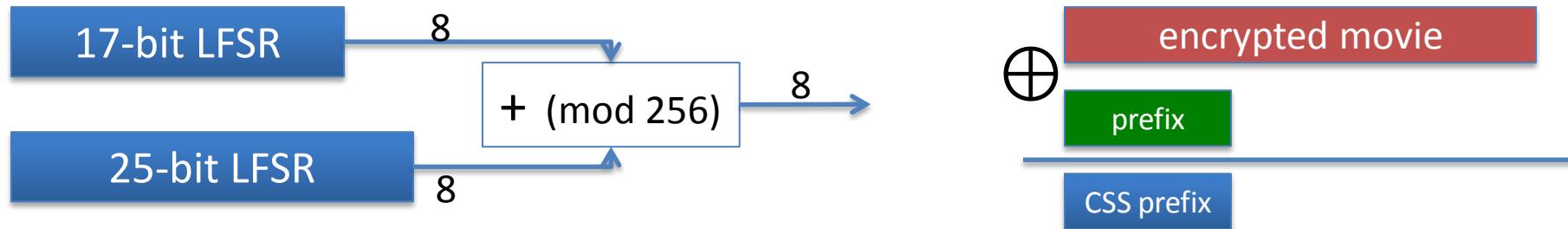
CSS: seed = 5 bytes = 40 bits



Easy to break in time $\approx 2^{17}$

Cryptanalysis of CSS

(2^{17} time attack)



For all possible initial settings of 17-bit LFSR do:

- Run 17-bit LFSR to get 20 bytes of output
- Subtract from CSS prefix \Rightarrow candidate 20 bytes output of 25-bit LFSR
- If consistent with 25-bit LFSR, found correct initial settings of both !!

Using key, generate entire CSS output

Modern stream ciphers: eStream

$$\text{PRG: } \underbrace{\{0,1\}^s}_{\text{seed}} \times R \rightarrow \{0,1\}^n$$


Nonce: a non-repeating value for a given key.

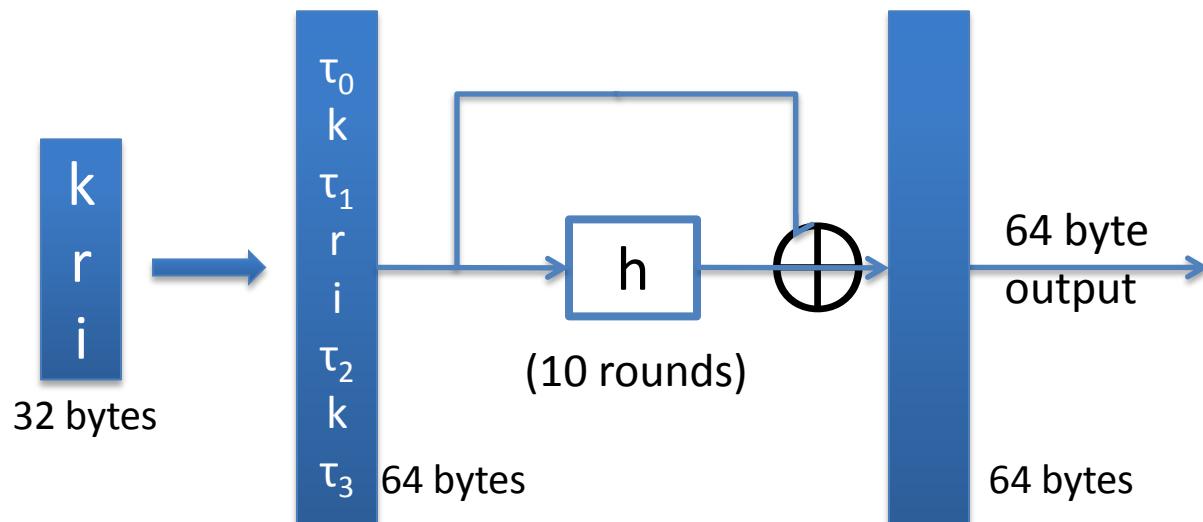
$$E(k, m ; r) = m \oplus \text{PRG}(k ; r)$$

The pair (k,r) is never used more than once.

eStream: Salsa 20 (SW+HW)

Salsa20: $\{0,1\}^{128 \text{ or } 256} \times \{0,1\}^{64} \xrightarrow{\text{nonce}} \{0,1\}^n$ (max n = 2^{73} bits)

$\text{Salsa20}(k; r) := H(k, (r, 0)) \parallel H(k, (r, 1)) \parallel \dots$



h : invertible function. designed to be fast on x86 (SSE2)

Is Salsa20 secure (unpredictable) ?

- Unknown: no known **provably** secure PRGs
- In reality: no known attacks better than exhaustive search

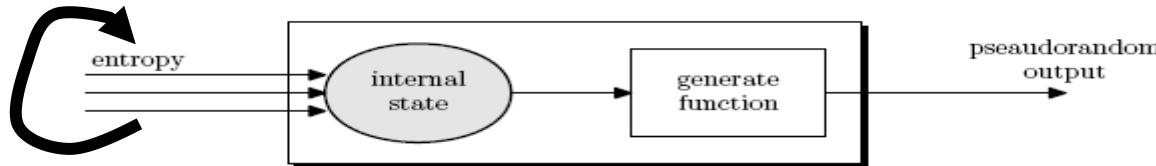
Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

<u>PRG</u>	<u>Speed (MB/sec)</u>
RC4	126
eStream	643
Salsa20/12	727
Sosemanuk	

Generating Randomness (e.g. keys, IV)



Pseudo random generators in practice: (e.g. /dev/random)

- Continuously add entropy to internal state
- Entropy sources:
 - Hardware RNG: Intel **RdRand** inst. (Ivy Bridge). 3Gb/sec.
 - Timing: hardware interrupts (keyboard, mouse)

NIST SP 800-90: NIST approved generators

End of Segment



Stream ciphers

PRG Security Defs

Let $G: K \rightarrow \{0,1\}^n$ be a PRG

Goal: define what it means that

$[K \xleftarrow{R} \mathcal{K}, \text{ output } G(K)]$



is “indistinguishable” from

$[r \xleftarrow{R} \{0,1\}^n, \text{ output } r]$

Statistical Tests

Statistical test on $\{0,1\}^n$:

an alg. A s.t. $A(x)$ outputs “0” or “1”

not random ↗ *random* ↘

Examples:

$$(1) \quad A(x)=1 \quad \text{iff} \quad | \#0(x) - \#1(x) | \leq 10 \cdot \sqrt{n}$$

$$(2) \quad A(x)=1 \quad \text{iff} \quad | \#00(x) - \frac{n}{4} | \leq 10 \cdot \sqrt{n}$$

Statistical Tests

More examples:

$$(3) A(x)=1 \text{ iff } \max\text{-run-of-}0(x) < 10 \cdot \log_2(n)$$

•
•
•
•

Advantage

Let $G:K \rightarrow \{0,1\}^n$ be a PRG and A a stat. test on $\{0,1\}^n$

Define:

$$\text{Adv}_{\text{PRG}}[A,G] := \left| \Pr_{K \in \mathcal{R}^K} [A(G(K))=1] - \Pr_{r \in \{0,1\}^n} [A(r)=1] \right| \in [0,1]$$

Adv close to 1 $\Rightarrow A$ can dist. G from random

Adv close to 0 $\Rightarrow A$ cannot

A silly example: $A(x) = 0 \Rightarrow \text{Adv}_{\text{PRG}}[A,G] =$



Suppose $G:K \rightarrow \{0,1\}^n$ satisfies $\text{msb}(G(k)) = 1$ for $2/3$ of keys in K

Define stat. test $A(x)$ as:

if [$\text{msb}(x)=1$] output “1” else output “0”

Then

$$\text{Adv}_{\text{PRG}}[A, G] = \left| \overbrace{\Pr[A(G(k))=1]}^{2/3} - \overbrace{\Pr[A(r)=1]}^{1/2} \right| =$$



Secure PRGs: crypto definition

Def: We say that $G:K \rightarrow \{0,1\}^n$ is a secure PRG if

\forall "eff" stat. tests A :

$Adv_{PRG}[A, G]$ is "negligible"

Are there provably secure PRGs?

but we have heuristic candidates.

Easy fact: a secure PRG is unpredictable

We show: PRG predictable \Rightarrow PRG is insecure

Suppose A is an efficient algorithm s.t.

$$\Pr_{\substack{K \leftarrow \mathcal{R} \\ g_K}} [A(g(K)|_{1, \dots, i}) = g(K)|_{i+1}] > \frac{1}{2} + \varepsilon$$

for non-negligible ε (e.g. $\varepsilon = 1/1000$)

Easy fact: a secure PRG is unpredictable

Define statistical test B as:

$$B(x) = \begin{cases} \text{if } A(x|_{1,\dots,i}) = x_{i+1} & \text{output 1} \\ \text{else} & \text{output 0} \end{cases}$$

$$r \xleftarrow{R} \{0,1\}^n : \Pr[B(r) = 1] = \frac{1}{2}$$

$$r \xleftarrow{g(k)} : \Pr[B(g(k)) = 1] > \frac{1}{2} + \varepsilon$$

$$\implies \text{Adv}_{\text{PRG}}[B, g] = \left| \Pr[B(r) = 1] - \Pr[B(g(k)) = 1] \right| > \varepsilon$$

Thm (Yao'82): an unpredictable PRG is secure

Let $G:K \rightarrow \{0,1\}^n$ be PRG

“Thm”: if $\forall i \in \{0, \dots, n-1\}$ PRG G is unpredictable at pos. i
then G is a secure PRG.

If next-bit predictors cannot distinguish G from random
then no statistical test can !!

Let $G:K \rightarrow \{0,1\}^n$ be a PRG such that
from the last $n/2$ bits of $G(k)$
it is easy to compute the first $n/2$ bits.

Is G predictable for some $i \in \{0, \dots, n-1\}$?

- Yes 
- No

More Generally

Let P_1 and P_2 be two distributions over $\{0,1\}^n$

Def: We say that P_1 and P_2 are

computationally indistinguishable (denoted $P_1 \approx_p P_2$)

if \forall "eff" stat. tests A

$$\left| \Pr_{x \leftarrow P_1} [A(x)=1] - \Pr_{x \leftarrow P_2} [A(x)=1] \right| < \text{negligible}$$

Example: a PRG is secure if $\{ k \xleftarrow{R} K : G(k) \} \approx_p \text{uniform}(\{0,1\}^n)$

End of Segment



Stream ciphers

Semantic security

Goal: secure PRG \Rightarrow “secure” stream cipher

What is a secure cipher?

Attacker's abilities: **obtains one ciphertext** (for now)

Possible security requirements:

attempt #1: **attacker cannot recover secret key**

$$E(k, m) = m$$

attempt #2: **attacker cannot recover all of plaintext**

$$E(k, m_0 \parallel m_1) = m_0 \parallel m_1 \oplus k$$

Recall Shannon's idea:

CT should reveal no “info” about PT

Recall Shannon's perfect secrecy

Let (E, D) be a cipher over (K, M, C)

(E, D) has perfect secrecy if $\forall m_0, m_1 \in M \quad (|m_0| = |m_1|)$

$$\{ E(k, m_0) \} = \{ E(k, m_1) \} \quad \text{where } k \leftarrow K$$

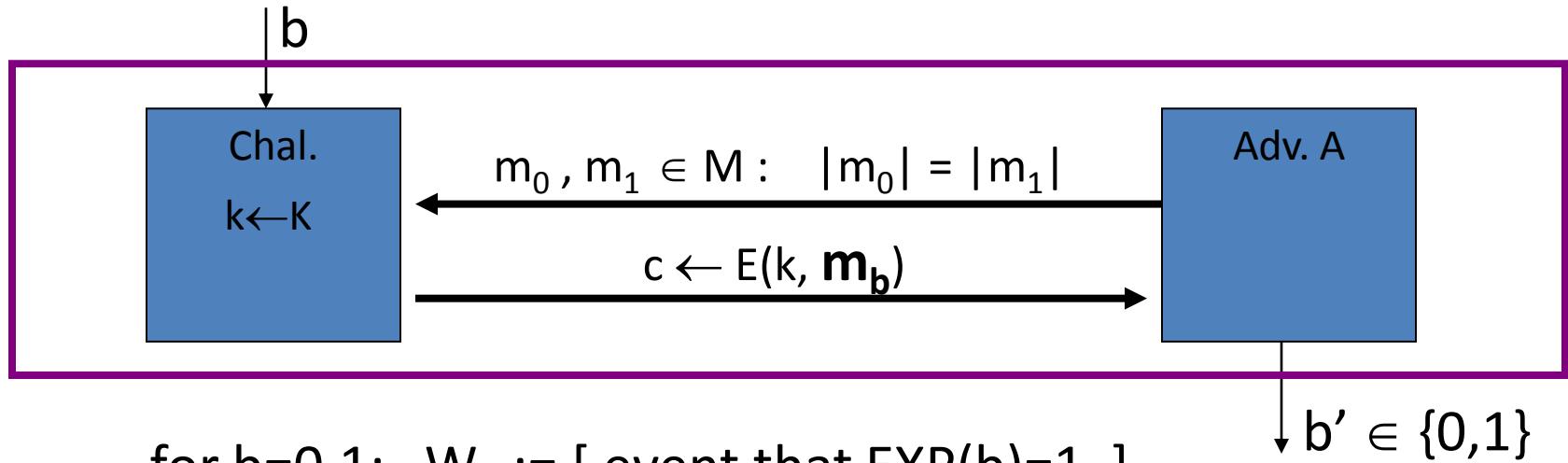
(E, D) has perfect secrecy if $\forall m_0, m_1 \in M \quad (|m_0| = |m_1|)$

$$\{ E(k, m_0) \} \approx_p \{ E(k, m_1) \} \quad \text{where } k \leftarrow K$$

... but also need adversary to exhibit $m_0, m_1 \in M$ explicitly

Semantic Security (one-time key)

For $b=0,1$ define experiments $\text{EXP}(0)$ and $\text{EXP}(1)$ as:



$$\text{Adv}_{\text{SS}}[A, E] := \left| \Pr[W_0] - \Pr[W_1] \right| \in [0,1]$$

Semantic Security (one-time key)

Def: \mathbb{E} is **semantically secure** if for all efficient A

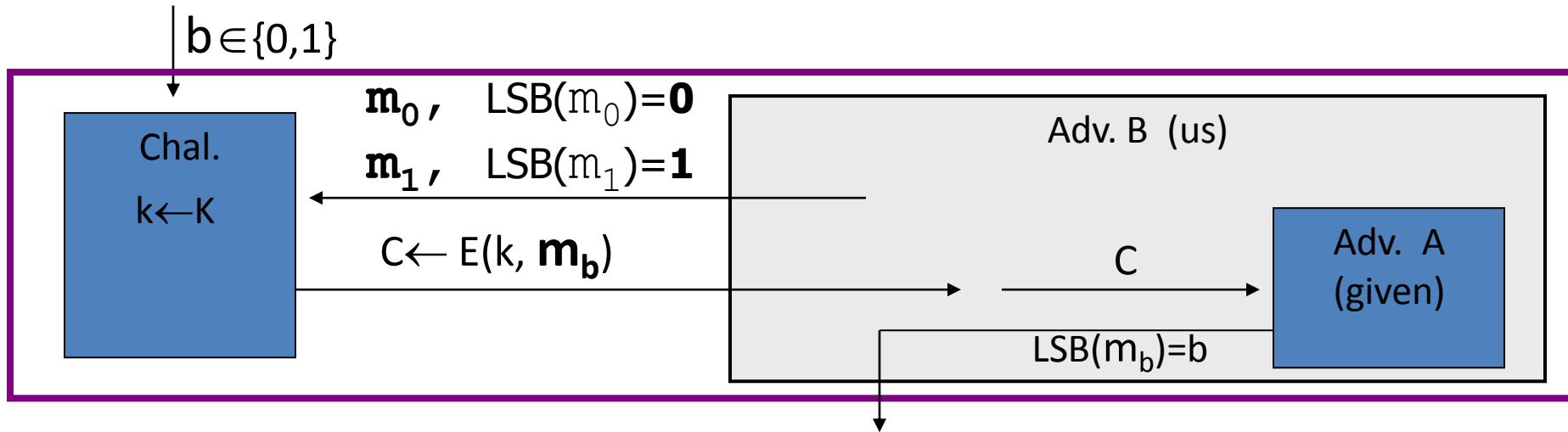
$\text{Adv}_{\text{SS}}[A, \mathbb{E}]$ is negligible.

\Rightarrow for all explicit $m_0, m_1 \in M$: $\{ E(k, m_0) \} \approx_p \{ E(k, m_1) \}$

Examples

Suppose efficient A can always deduce LSB of PT from CT.

⇒ $\mathbb{E} = (E, D)$ is not semantically secure.

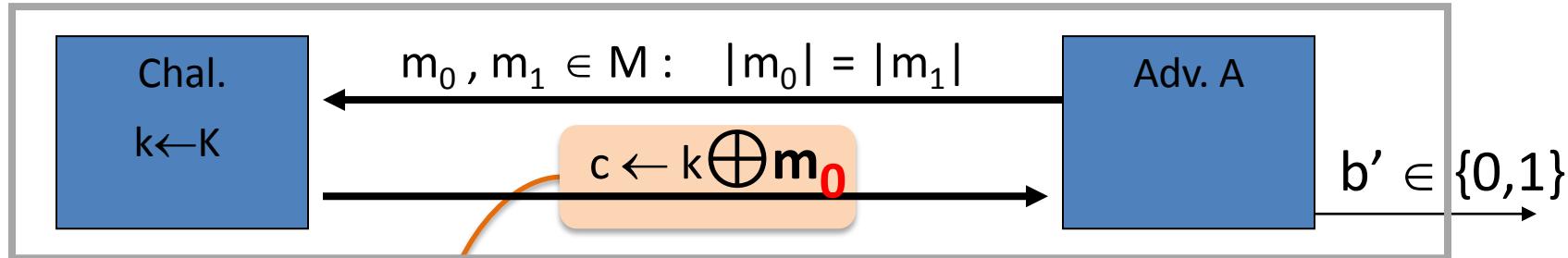


Then $\text{Adv}_{\text{SS}}[B, \mathbb{E}] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right| =$



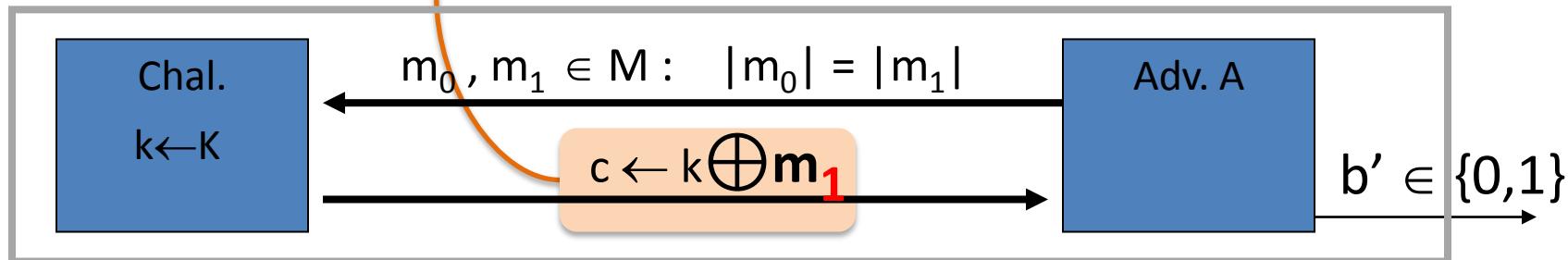
OTP is semantically secure

$\text{EXP}(0)$:



identical distributions

$\text{EXP}(1)$:



For all A: $\text{Adv}_{\text{SS}}[A, \text{OTP}] = \left| \Pr[A(k \oplus m_0) = 1] - \Pr[A(k \oplus m_1) = 1] \right|$



End of Segment



Stream ciphers

Stream ciphers are semantically secure

Goal: secure PRG \Rightarrow semantically secure stream cipher

Stream ciphers are semantically secure

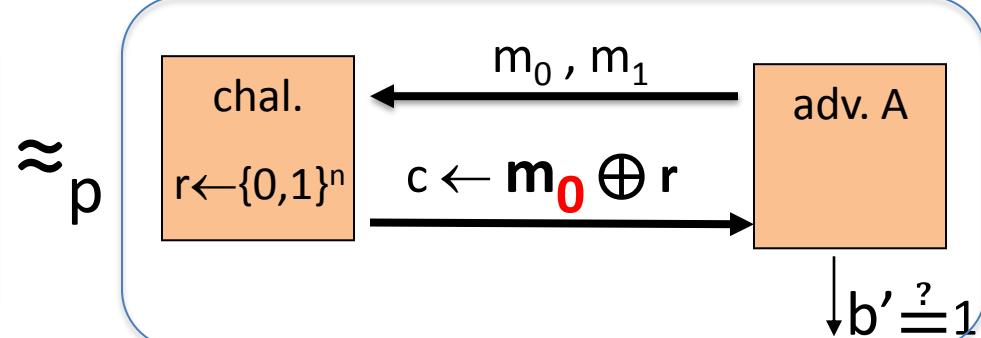
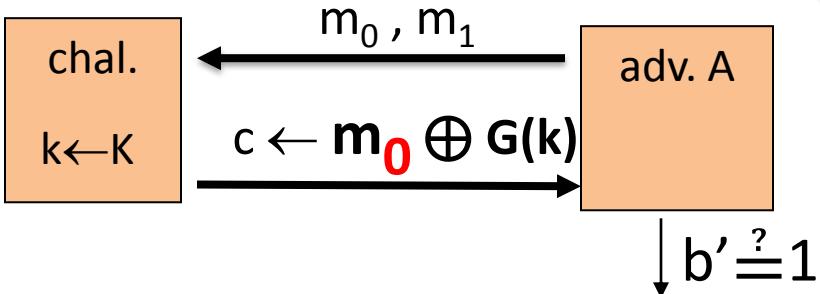
Thm: $G:K \rightarrow \{0,1\}^n$ is a secure PRG \Rightarrow

stream cipher E derived from G is sem. sec.

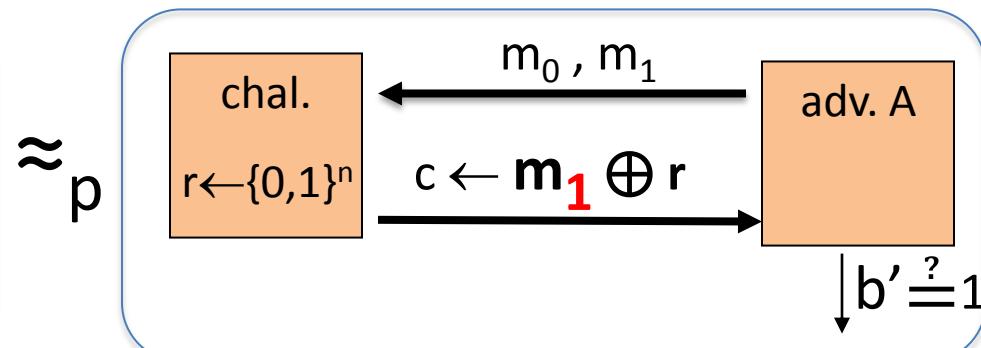
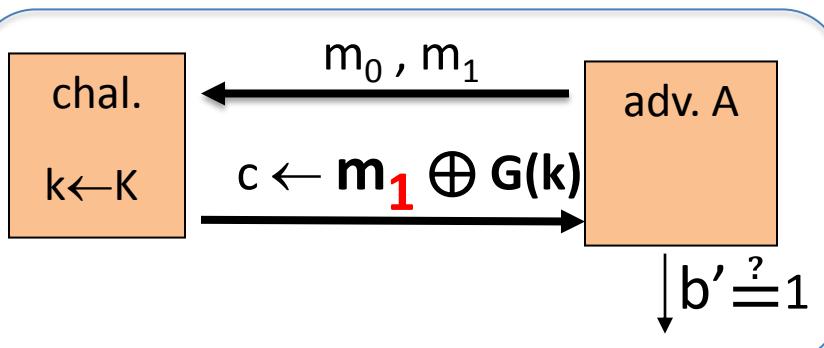
\forall sem. sec. adversary A , \exists a PRG adversary B s.t.

$$\text{Adv}_{\text{SS}}[A, E] \leq 2 \cdot \text{Adv}_{\text{PRG}}[B, G]$$

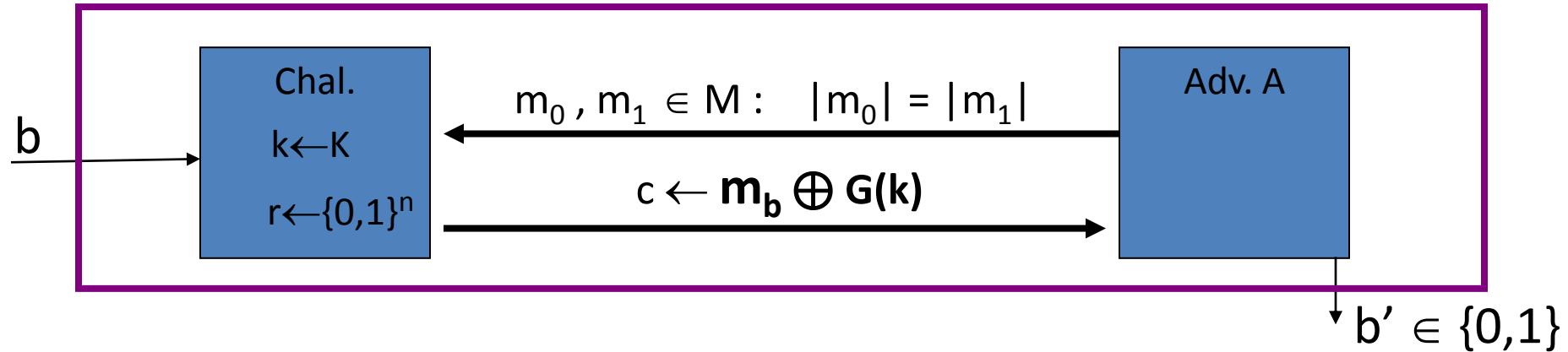
Proof: intuition



\approx_p



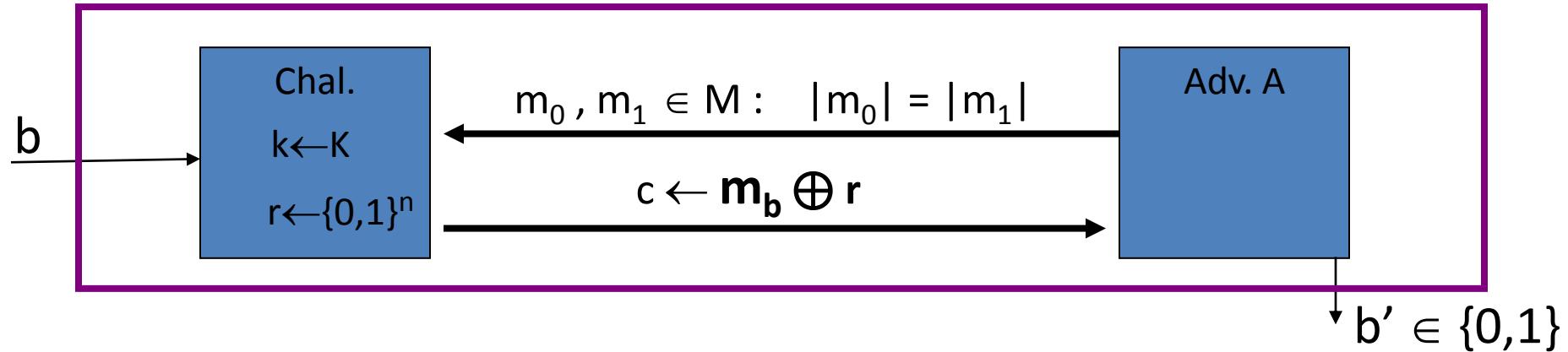
Proof: Let A be a sem. sec. adversary.



For $b=0,1$: $W_b := [\text{event that } b'=1]$.

$$\text{Adv}_{SS}[A, E] = | \Pr[W_0] - \Pr[W_1] |$$

Proof: Let A be a sem. sec. adversary.



For $b=0,1$: $W_b := [\text{event that } b'=1]$.

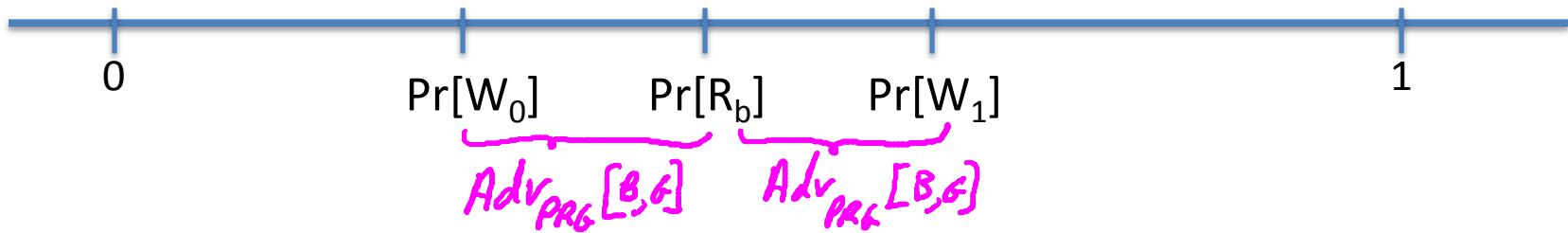
$$\text{Adv}_{SS}[A, E] = | \Pr[W_0] - \Pr[W_1] |$$

For $b=0,1$: $R_b := [\text{event that } b'=1]$

Proof: Let A be a sem. sec. adversary.

Claim 1: $|\Pr[R_0] - \Pr[R_1]| = \text{Adv}_{\text{SS}}[A, \text{OTP}] = 0$

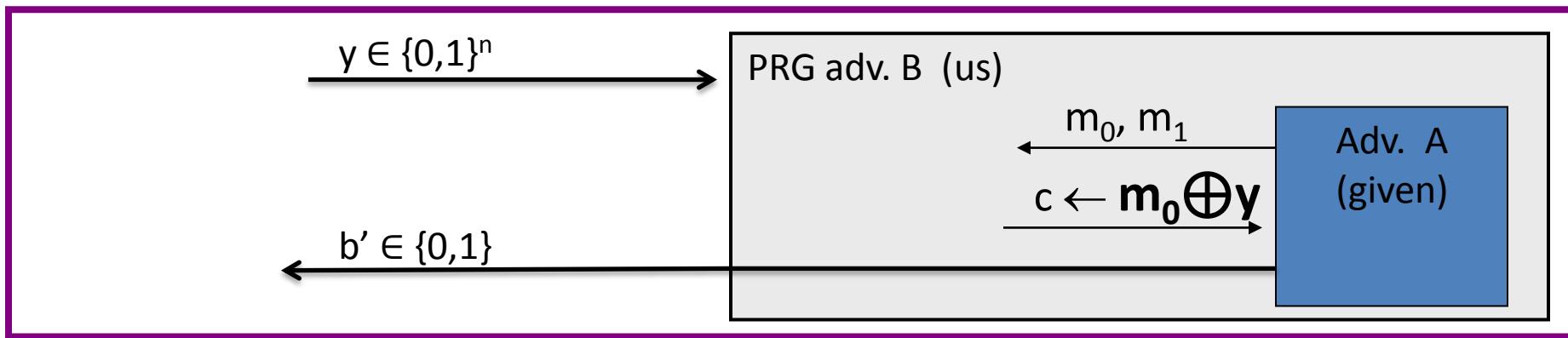
Claim 2: $\exists B: |\Pr[W_b] - \Pr[R_b]| = \text{Adv}_{\text{PRG}}[B, G]$ For $b=0, 1$



$$\Rightarrow \text{Adv}_{\text{SS}}[A, E] = |\Pr[W_0] - \Pr[W_1]| \leq 2 \cdot \text{Adv}_{\text{PRG}}[B, G]$$

Proof of claim 2: $\exists B: |\Pr[W_0] - \Pr[R_0]| = \text{Adv}_{\text{PRG}}[B, G]$

Algorithm B:



$$\text{Adv}_{\text{PRG}}[B, G] = \left| \Pr_{r \leftarrow \{0,1\}^n} [B(r) = 1] - \Pr_{k \leftarrow \mathcal{R}} [B(g(k)) = 1] \right| = \left| \Pr[R_0] - \Pr[W_0] \right|$$

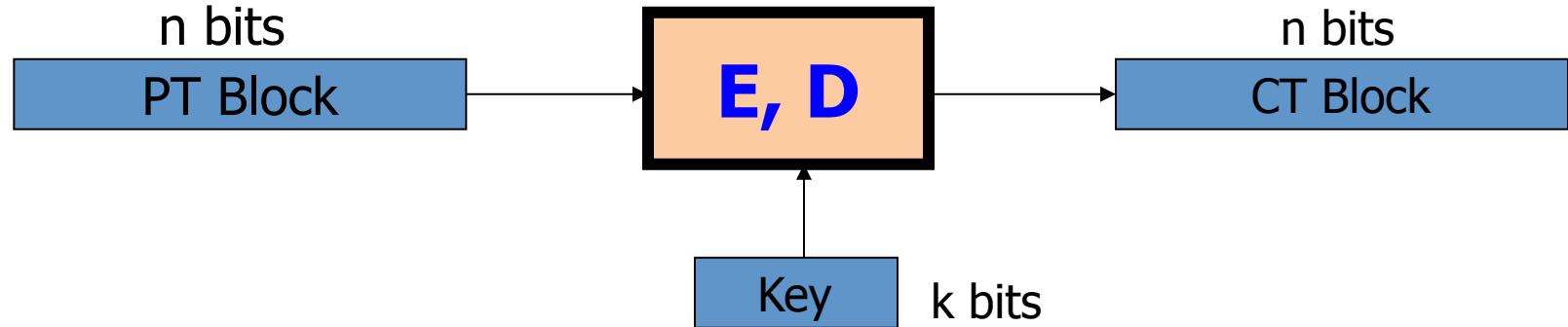
End of Segment



Block ciphers

What is a block cipher?

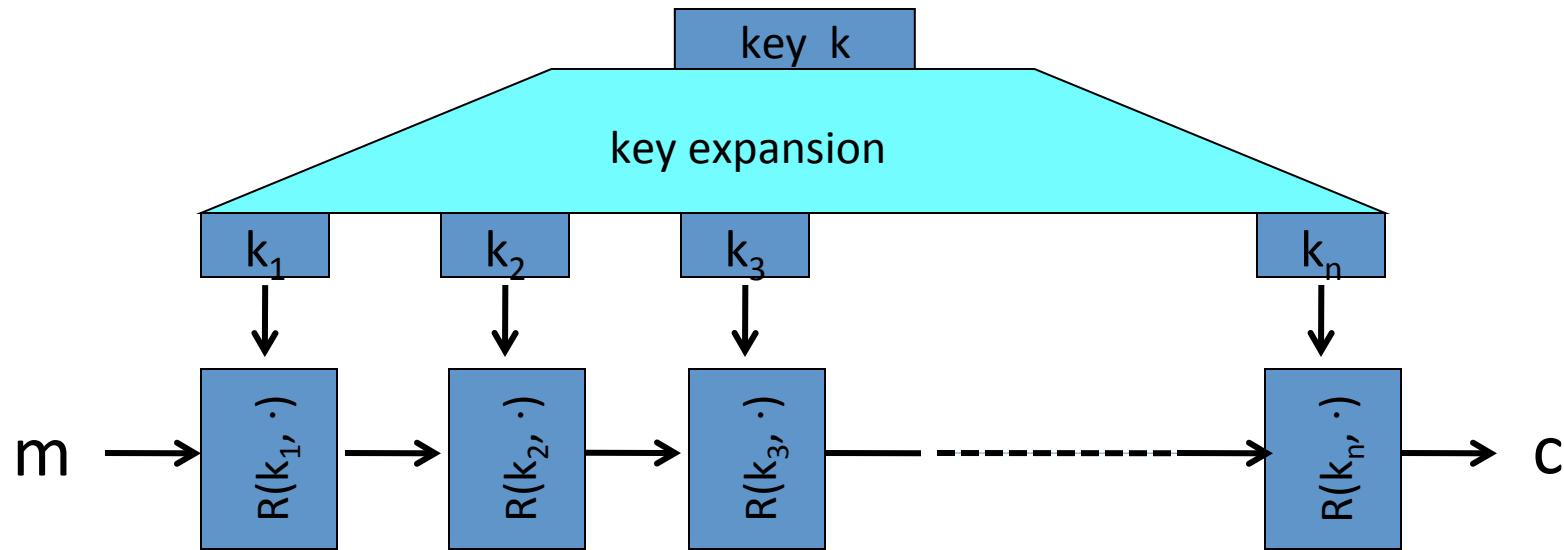
Block ciphers: crypto work horse



Canonical examples:

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

Block Ciphers Built by Iteration



$R(k, m)$ is called a round function

for 3DES ($n=48$), for AES-128 ($n=10$)

Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	<u>Cipher</u>	<u>Block/key size</u>	<u>Speed (MB/sec)</u>
stream	RC4		126
	Salsa20/12		643
	Sosemanuk		727
block	3DES	64/168	13
	AES-128	128/128	109

Abstractly: PRPs and PRFs

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

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

such that exists “efficient” algorithm to evaluate $F(k, x)$

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

$$E: K \times X \rightarrow X$$

such that:

1. Exists “efficient” deterministic algorithm to evaluate $E(k, x)$
2. The function $E(k, \cdot)$ is one-to-one
3. Exists “efficient” inversion algorithm $D(k, y)$

Running example

- Example PRPs: 3DES, AES, ...

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

3DES: $K \times X \rightarrow X$ where $X = \{0,1\}^{64}$, $K = \{0,1\}^{168}$

- Functionally, any PRP is also a PRF.
 - A PRP is a PRF where $X=Y$ and is efficiently invertible.

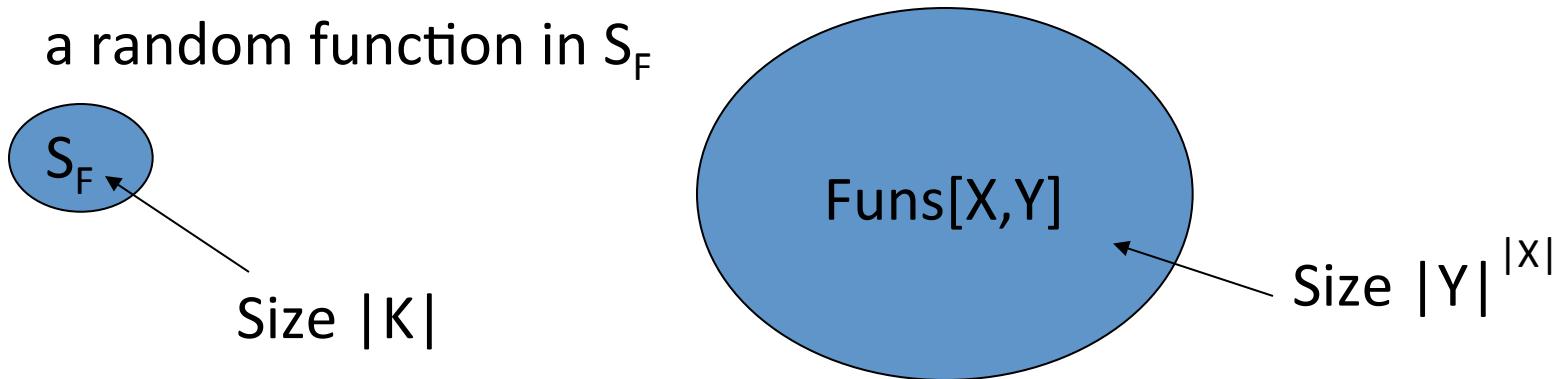
Secure PRFs

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

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

- Intuition: a PRF is **secure** if

a random function in $\text{Funs}[X,Y]$ is indistinguishable from
a random function in S_F

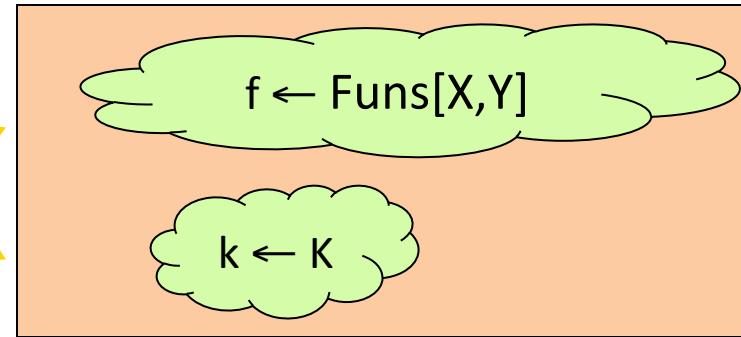
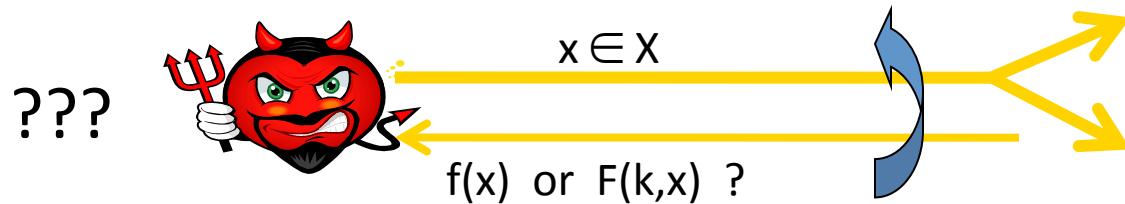


Secure PRFs

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

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

- Intuition: a PRF is **secure** if
a random function in $\text{Funs}[X,Y]$ is indistinguishable from
a random function in S_F



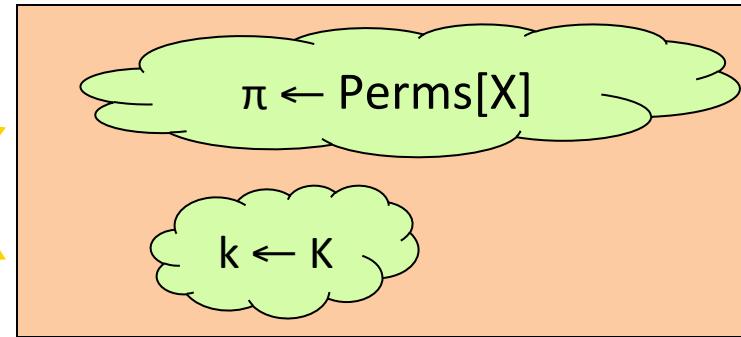
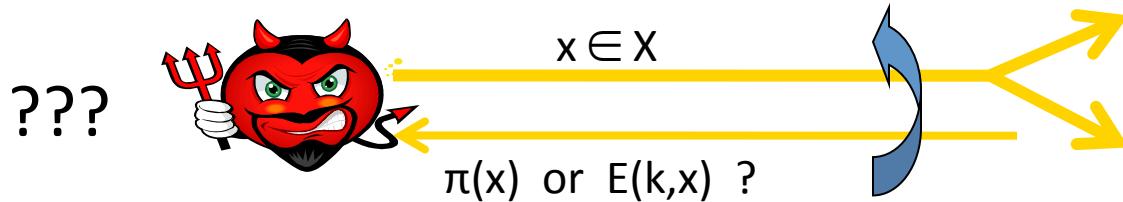
Secure PRPs

(secure block cipher)

- Let $E: K \times X \rightarrow Y$ be a PRP

$\left\{ \begin{array}{l} \text{Perms}[X]: \text{ the set of all } \underline{\text{one-to-one}} \text{ functions from } X \text{ to } Y \\ S_F = \{ E(k, \cdot) \text{ s.t. } k \in K \} \subseteq \text{Perms}[X, Y] \end{array} \right.$

- Intuition: a PRP is **secure** if
a random function in $\text{Perms}[X]$ is indistinguishable from
a random function in S_F



Let $F: K \times X \rightarrow \{0,1\}^{128}$ be a secure PRF.

Is the following G a secure PRF?

$$G(k, x) = \begin{cases} 0^{128} & \text{if } x=0 \\ F(k,x) & \text{otherwise} \end{cases}$$

- No, it is easy to distinguish G from a random function
- Yes, an attack on G would also break F
- It depends on F

An easy application: $\text{PRF} \Rightarrow \text{PRG}$

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

Then the following $G: K \rightarrow \{0,1\}^{nt}$ is a secure PRG:

$$G(k) = F(k,0) \parallel F(k,1) \parallel \dots \parallel F(k,t-1)$$

Key property: parallelizable

Security from PRF property: $F(k, \cdot)$ indist. from random function $f(\cdot)$

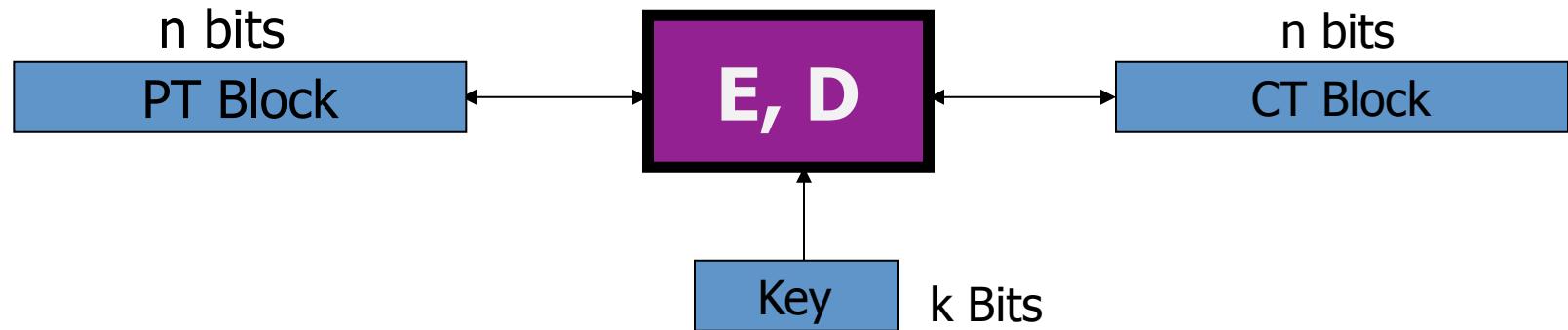
End of Segment



Block ciphers

The data encryption
standard (DES)

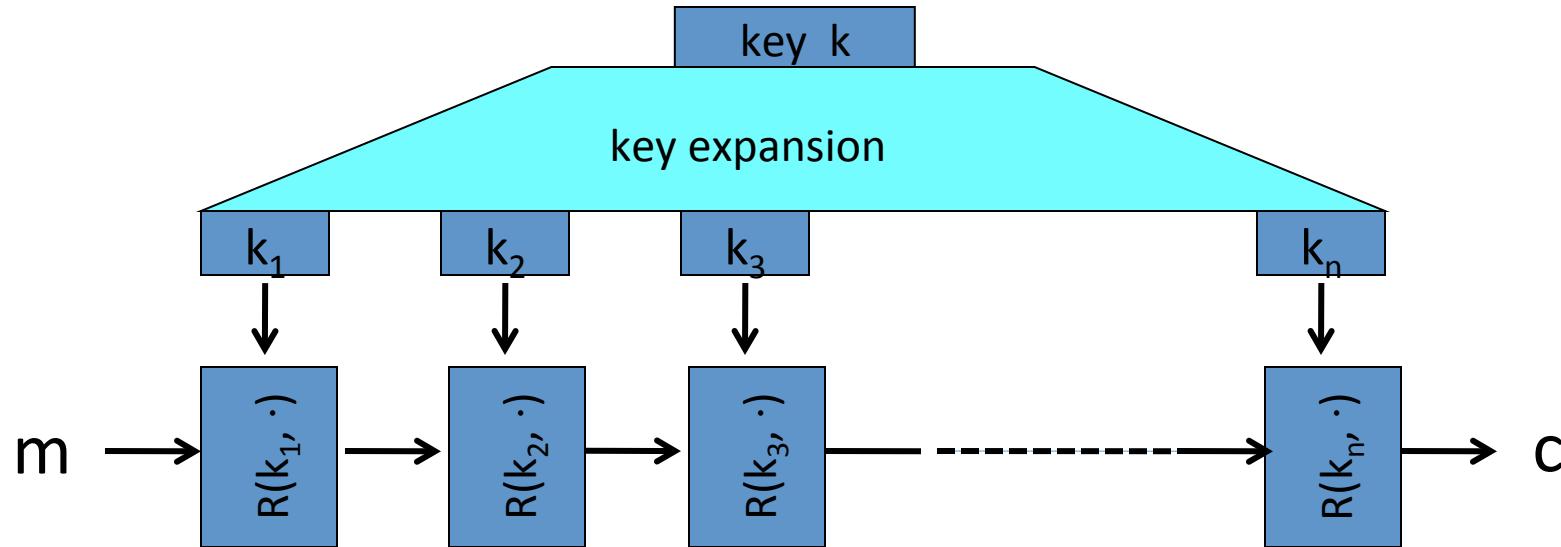
Block ciphers: crypto work horse



Canonical examples:

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

Block Ciphers Built by Iteration



$R(k, m)$ is called a round function

for 3DES ($n=48$), for AES-128 ($n=10$)

The Data Encryption Standard (DES)

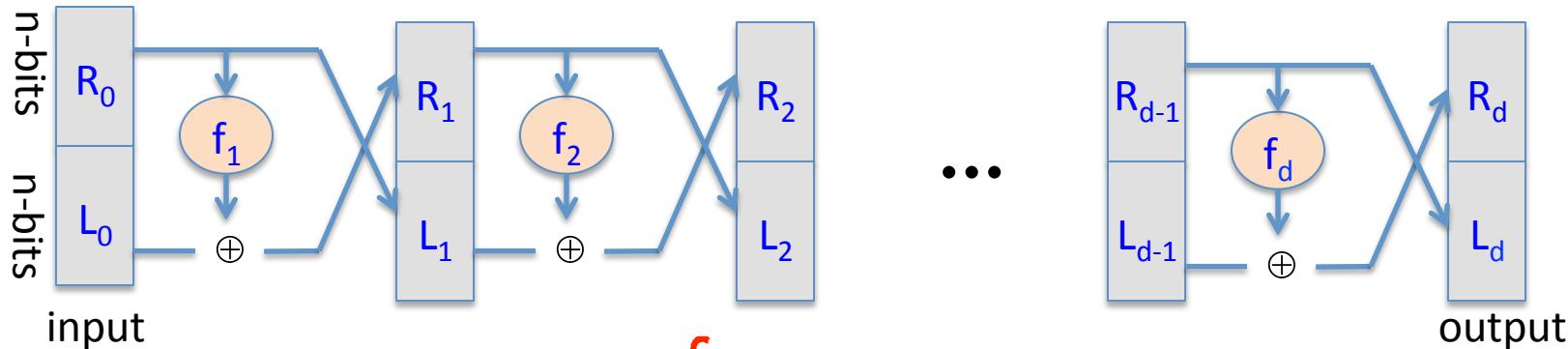
- Early 1970s: Horst Feistel designs Lucifer at IBM
key-len = 128 bits ; block-len = 128 bits
- 1973: NBS asks for block cipher proposals.
IBM submits variant of Lucifer.
- 1976: NBS adopts DES as a federal standard
key-len = 56 bits ; block-len = 64 bits
- 1997: DES broken by exhaustive search
- 2000: NIST adopts Rijndael as AES to replace DES

Widely deployed in banking (ACH) and commerce

DES: core idea – Feistel Network

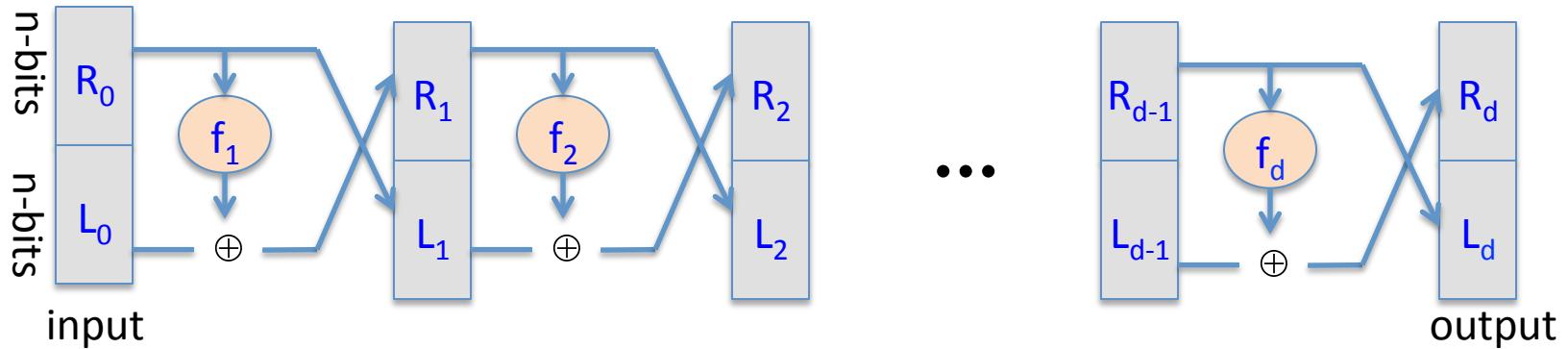
Given functions $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$

Goal: build invertible function $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$



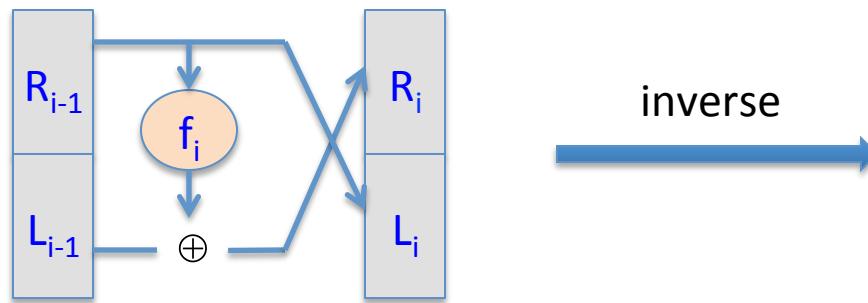
In symbols:

$$\begin{cases} R_i = f_i(R_{i-1}) \oplus L_{i-1}, \\ L_i = R_{i-1}, \end{cases}$$



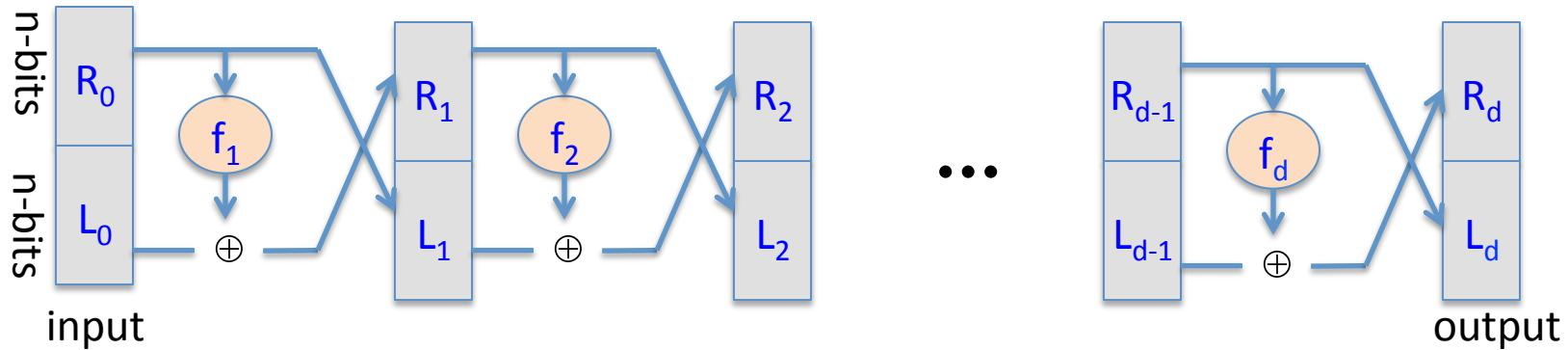
Claim: for all $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$
 Feistel network $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is invertible

Proof: construct inverse



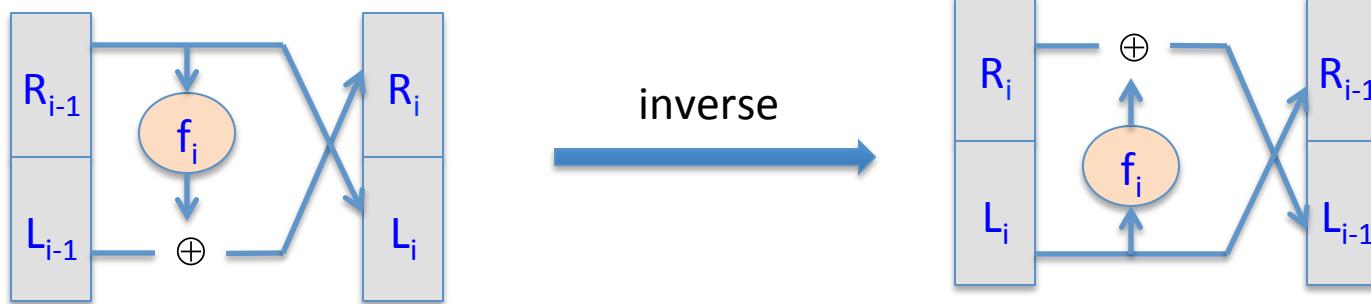
$$R_{i-1} = L_i$$

$$L_{i-1} = \boxed{\quad}$$

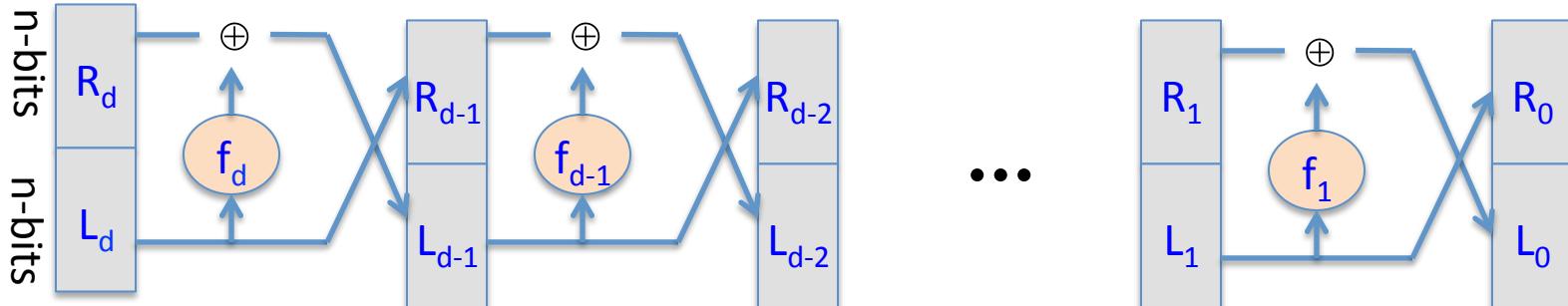


Claim: for all $f_1, \dots, f_d: \{0,1\}^n \rightarrow \{0,1\}^n$
 Feistel network $F: \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ is invertible

Proof: construct inverse



Decryption circuit

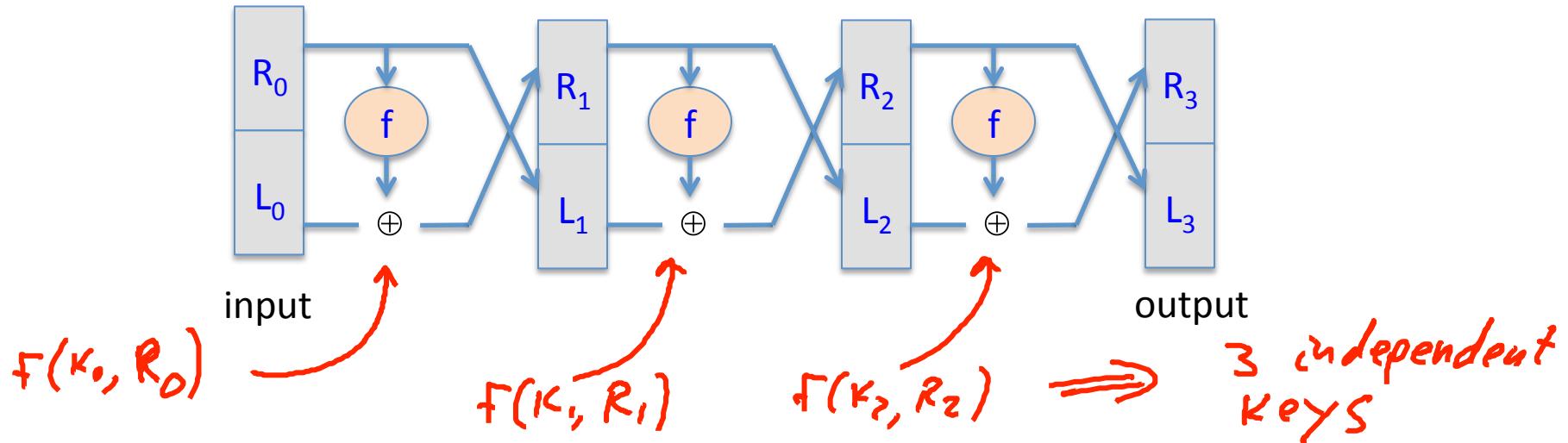


- Inversion is basically the same circuit, with f_1, \dots, f_d applied in reverse order
- General method for building invertible functions (block ciphers) from arbitrary functions.
- Used in many block ciphers ... but not AES

“Thm:” (Luby-Rackoff ‘85):

$f: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a secure PRF

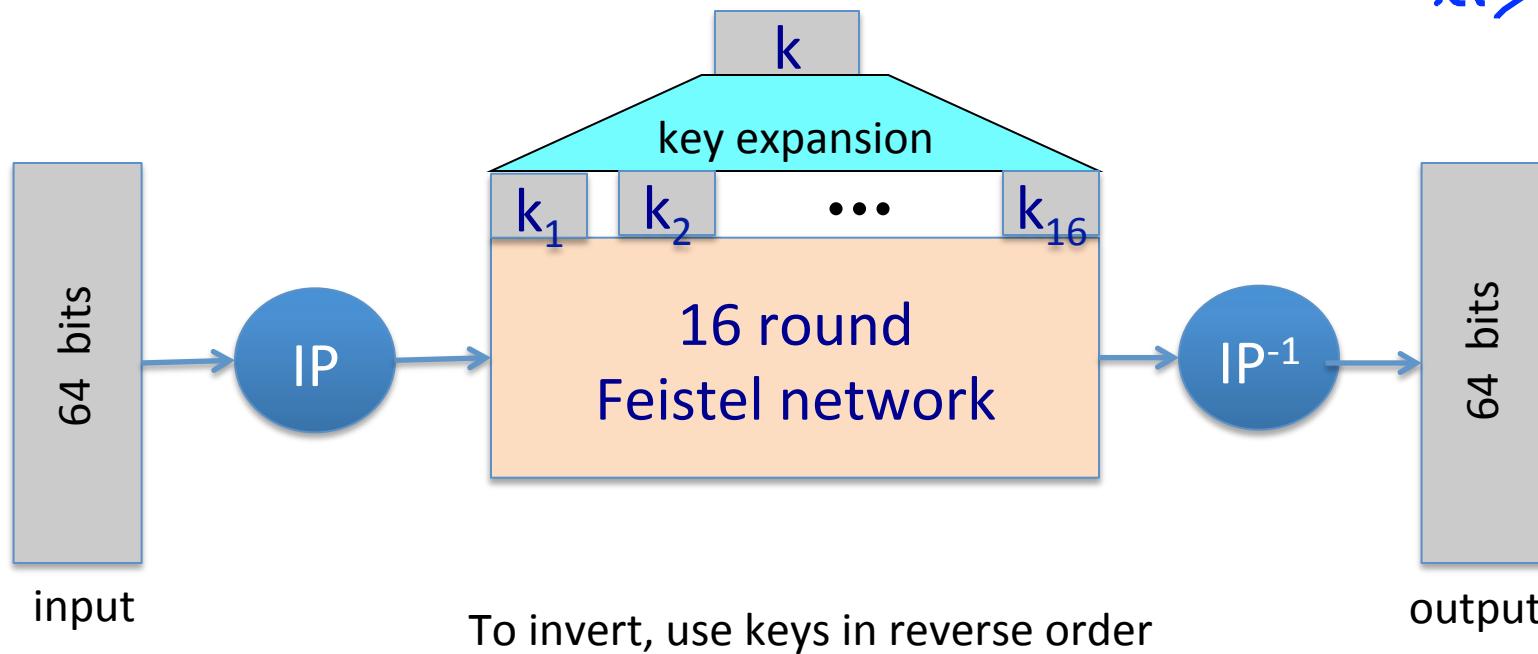
\Rightarrow 3-round Feistel $F: K^3 \times \{0,1\}^{2n} \rightarrow \{0,1\}^{2n}$ a secure PRP



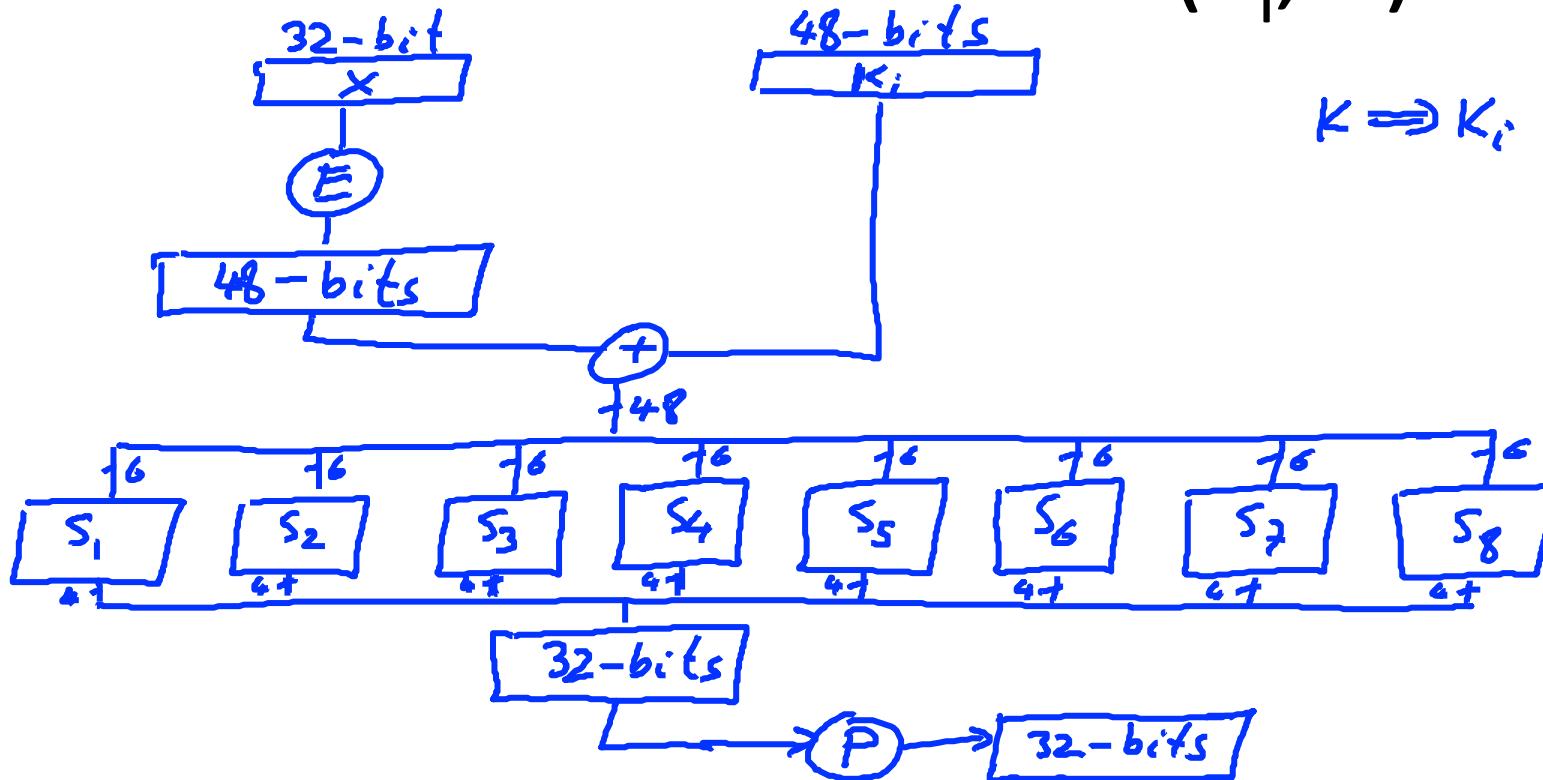
DES: 16 round Feistel network

$$f_1, \dots, f_{16}: \{0,1\}^{32} \rightarrow \{0,1\}^{32} , \quad f_i(x) = F(k_i, x)$$

↑
from
key k



The function $F(k_i, x)$



S-box: function $\{0,1\}^6 \rightarrow \{0,1\}^4$, implemented as look-up table.

The S-boxes

$$S_i: \{0,1\}^6 \rightarrow \{0,1\}^4$$

		Middle 4 bits of input															
		S ₅	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Example: a bad S-box choice

Suppose:

$$S_i(x_1, x_2, \dots, x_6) = (x_2 \oplus x_3, x_1 \oplus x_4 \oplus x_5, x_1 \oplus x_6, x_2 \oplus x_3 \oplus x_6)$$

or written equivalently: $S_i(\mathbf{x}) = A_i \cdot \mathbf{x} \pmod{2}$

$$\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{matrix} \cdot \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{matrix} = \begin{matrix} x_2 \oplus x_3 \\ x_1 \oplus x_4 \oplus x_5 \\ x_1 \oplus x_6 \\ x_2 \oplus x_3 \oplus x_6 \end{matrix}$$

We say that S_i is a linear function.

Example: a bad S-box choice

Then entire DES cipher would be linear: \exists fixed binary matrix B s.t.

$$\text{DES}(k, m) = \begin{matrix} & 64 \\ & \cdot \\ \text{DES}(k, m) = & \begin{matrix} 832 \\ B \end{matrix} \cdot \begin{matrix} m \\ k_1 \\ k_2 \\ \vdots \\ k_{16} \end{matrix} = \begin{matrix} c \end{matrix} \quad (\text{mod } 2) \end{matrix}$$

But then: $\text{DES}(k, m_1) \oplus \text{DES}(k, m_2) \oplus \text{DES}(k, m_3) = \text{DES}(k, m_1 \oplus m_2 \oplus m_3)$

$$k = \begin{pmatrix} k_1 \\ \vdots \\ k_{16} \end{pmatrix}$$

$$B \begin{pmatrix} m_1 \\ k \end{pmatrix} \oplus B \begin{pmatrix} m_2 \\ k \end{pmatrix} \oplus B \begin{pmatrix} m_3 \\ k \end{pmatrix} = B \begin{pmatrix} m_1 \oplus m_2 \oplus m_3 \\ k \oplus k \oplus k \end{pmatrix}$$

Choosing the S-boxes and P-box

Choosing the S-boxes and P-box at random would result in an insecure block cipher (key recovery after $\approx 2^{24}$ outputs) [BS'89]

Several rules used in choice of S and P boxes:

- No output bit should be close to a linear func. of the input bits
 - S-boxes are 4-to-1 maps
- ⋮

End of Segment



Block ciphers

Exhaustive Search Attacks

Exhaustive Search for block cipher key

Goal: given a few input output pairs $(m_i, c_i = E(k, m_i)) \quad i=1,\dots,3$
find key k .

Lemma: Suppose DES is an *ideal cipher*

(2^{56} random invertible functions $\pi_{1,\dots,16}: \{0,1\}^{64} \rightarrow \{0,1\}^{64}$)

Then $\forall m, c$ there is at most one key k s.t. $c = \text{DES}(k, m)$

Proof: $\Pr[\exists k' \neq k : c = \text{DES}(k, m) = \text{DES}(k', m)] \leq$ with prob. $\geq 1 - 1/256 \approx 99.5\%$

$$\leq \sum_{k' \in \{0,1\}^{56}} \Pr[\text{DES}(k, m) = \text{DES}(k', m)] \leq 2^{56} \cdot \frac{1}{2^{64}} = \frac{1}{2^8}$$

Exhaustive Search for block cipher key

For two DES pairs $(m_1, c_1 = \text{DES}(k, m_1)), (m_2, c_2 = \text{DES}(k, m_2))$
unicity prob. $\approx 1 - 1/2^{71}$

For AES-128: given two inp/out pairs, unicity prob. $\approx 1 - 1/2^{128}$

⇒ two input/output pairs are enough for exhaustive key search.

DES challenge

```
msg = "The unknown messages is: XXXX ... "
```

CT = $c_1 \quad c_2 \quad c_3 \quad c_4$

Goal: find $k \in \{0,1\}^{56}$ s.t. $\text{DES}(k, m_i) = c_i$ for $i=1,2,3$

1997: Internet search -- **3 months**

1998: EFF machine (deep crack) -- **3 days** (250K \$)

1999: combined search -- **22 hours**

2006: COPACOBANA (120 FPGAs) -- **7 days** (10K \$)

⇒ 56-bit ciphers should not be used !! (128-bit key ⇒ 2^{72} days)

Strengthening DES against ex. search

Method 1: Triple-DES

- Let $E : K \times M \rightarrow M$ be a block cipher
- Define $3E: K^3 \times M \rightarrow M$ as

$$3E(k_1, k_2, k_3, m) = E(k_1, D(k_2, E(k_3, m)))$$
$$k_1 = k_2 = k_3 \Rightarrow \text{single DES}$$

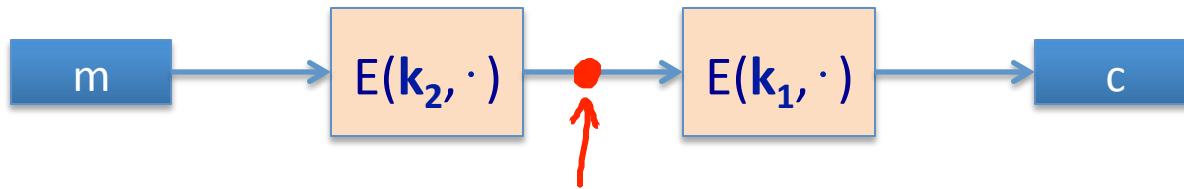
For 3DES: key-size = $3 \times 56 = 168$ bits. 3×slower than DES.

(simple attack in time $\approx 2^{118}$)

Why not double DES?

- Define $2E((k_1, k_2), m) = E(k_1, E(k_2, m))$

key-len = 112 bits for DES



Find (k_1, k_2) s.t.
 $E(k_1, E(k_2, M)) = C$
Equivalently:
 $E(k_2, M) = D(k_1, C)$

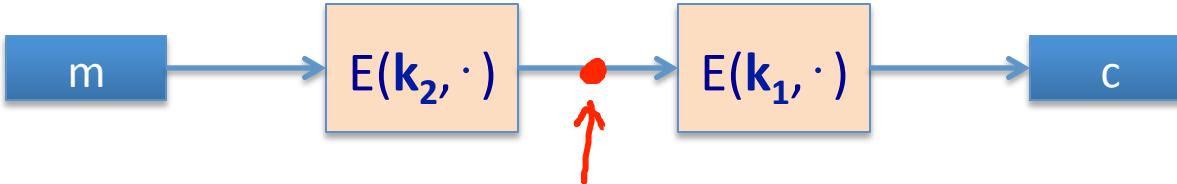
Attack: $M = (m_1, \dots, m_{10})$, $C = (c_1, \dots, c_{10})$.

- step 1: build table.
sort on 2nd column

$k^0 = 00\dots00$	$E(k^0, M)$
$k^1 = 00\dots01$	$E(k^1, M)$
$k^2 = 00\dots10$	$E(k^2, M)$
\vdots	\vdots
$k^N = 11\dots11$	$E(k^N, M)$

2^{56} entries

Meet in the middle attack



Attack: $M = (m_1, \dots, m_{10})$, $C = (c_1, \dots, c_{10})$

- step 1: build table.

$k^0 = 00\dots00$	$E(k^0, M)$
$k^1 = 00\dots01$	$E(k^1, M)$
$k^2 = 00\dots10$	$E(k^2, M)$
\vdots	\vdots
$k^N = 11\dots11$	$E(k^N, M)$

- Step 2: for all $k \in \{0,1\}^{56}$ do:

test if $D(k, C)$ is in 2nd column.

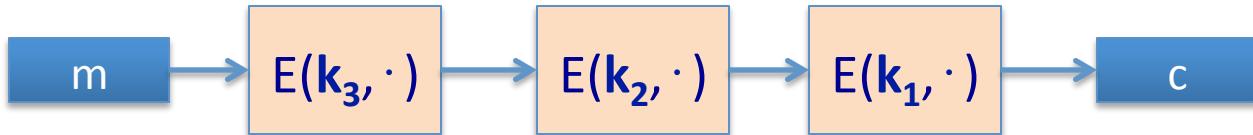
if so then $E(k^i, M) = D(k, C) \Rightarrow (k^i, k) = (k_2, k_1)$

Meet in the middle attack



Time = $\underbrace{2^{56}\log(2^{56})}_{\text{build + sort table}} + \underbrace{2^{56}\log(2^{56})}_{\text{search in table}} < 2^{63} \ll 2^{112}, \quad \text{space} \approx 2^{56}$

Same attack on 3DES: Time = 2^{118} , space $\approx 2^{56}$



Method 2: DESX

$E : K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a block cipher

Define EX as $EX(k_1, k_2, k_3, m) = k_1 \oplus E(k_2, m \oplus k_3)$

For DESX: key-len = $64+56+64 = 184$ bits

... but easy attack in time $2^{64+56} = 2^{120}$ (homework)

Note: $k_1 \oplus E(k_2, m)$ and $E(k_2, m \oplus k_1)$ does nothing !!

End of Segment



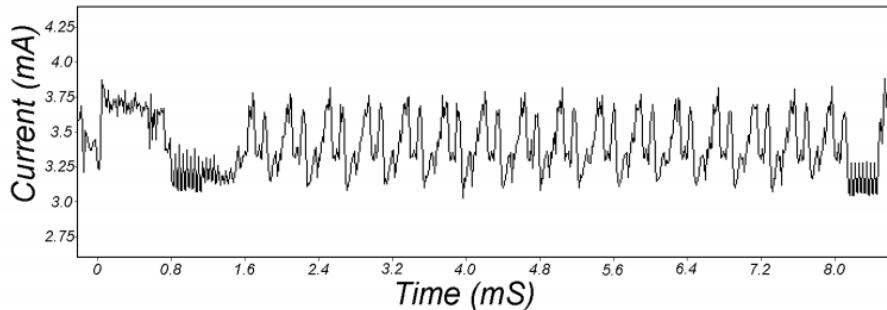
Block ciphers

More attacks on
block ciphers

Attacks on the implementation

1. Side channel attacks:

- Measure **time** to do enc/dec, measure **power** for enc/dec



[Kocher, Jaffe, Jun, 1998]

2. Fault attacks:

- Computing errors in the last round expose the secret key k

⇒ do not even implement crypto primitives yourself ...

Linear and differential attacks

[BS'89, M'93]

Given *many* inp/out pairs, can recover key in time less than 2^{56} .

Linear cryptanalysis (overview) : let $c = \text{DES}(k, m)$

Suppose for random k, m :

$$\Pr \left[\underbrace{m[i_1] \oplus \cdots \oplus m[i_r]}_{\text{subset of msg bits}} \oplus \underbrace{c[j_1] \oplus \cdots \oplus c[j_v]}_{\text{subset of ciphered bits}} = \underbrace{k[l_1] \oplus \cdots \oplus k[l_u]}_{\text{subset of key bits}} \right] = \frac{1}{2} + \varepsilon$$

For some ε . For DES, this exists with $\varepsilon = 1/2^{21} \approx 0.0000000477$

Linear attacks

$$\Pr \left[m[i_1] \oplus \cdots \oplus m[i_r] \oplus c[j_j] \oplus \cdots \oplus c[j_v] = k[l_1] \oplus \cdots \oplus k[l_u] \right] = \frac{1}{2} + \varepsilon$$

Thm: given $1/\varepsilon^2$ random $(m, c=DES(k, m))$ pairs then

$$k[l_1, \dots, l_u] = \text{MAJ} \left[m[i_1, \dots, i_r] \oplus c[j_j, \dots, j_v] \right]$$

with prob. $\geq 97.7\%$

\Rightarrow with $1/\varepsilon^2$ inp/out pairs can find $k[l_1, \dots, l_u]$ in time $\approx 1/\varepsilon^2$.

Linear attacks

For DES, $\epsilon = 1/2^{21} \Rightarrow$

with 2^{42} inp/out pairs can find $k[l_1, \dots, l_u]$ in time 2^{42}

Roughly speaking: can find 14 key “bits” this way in time 2^{42}

Brute force remaining $56 - 14 = 42$ bits in time 2^{42}

Total attack time $\approx 2^{43}$ ($<< 2^{56}$) with 2^{42} random inp/out pairs

Lesson

A tiny bit of linearity in S_5 lead to a 2^{42} time attack.

⇒ don't design ciphers yourself !!

Quantum attacks

Generic search problem:

Let $f: X \rightarrow \{0,1\}$ be a function.

Goal: find $x \in X$ s.t. $f(x)=1$.

Classical computer: best generic algorithm time = $O(|X|)$

Quantum computer [Grover '96] : time = $O(|X|^{1/2})$

Can quantum computers be built: unknown

Quantum exhaustive search

Given $m, c = E(k, m)$ define

$$f(k) = \begin{cases} 1 & \text{if } E(k, m) = c \\ 0 & \text{otherwise} \end{cases}$$

Grover \Rightarrow quantum computer can find k in time $O(|K|^{1/2})$

DES: time $\approx 2^{28}$, AES-128: time $\approx 2^{64}$

quantum computer \Rightarrow 256-bits key ciphers (e.g. AES-256)

End of Segment



Block ciphers

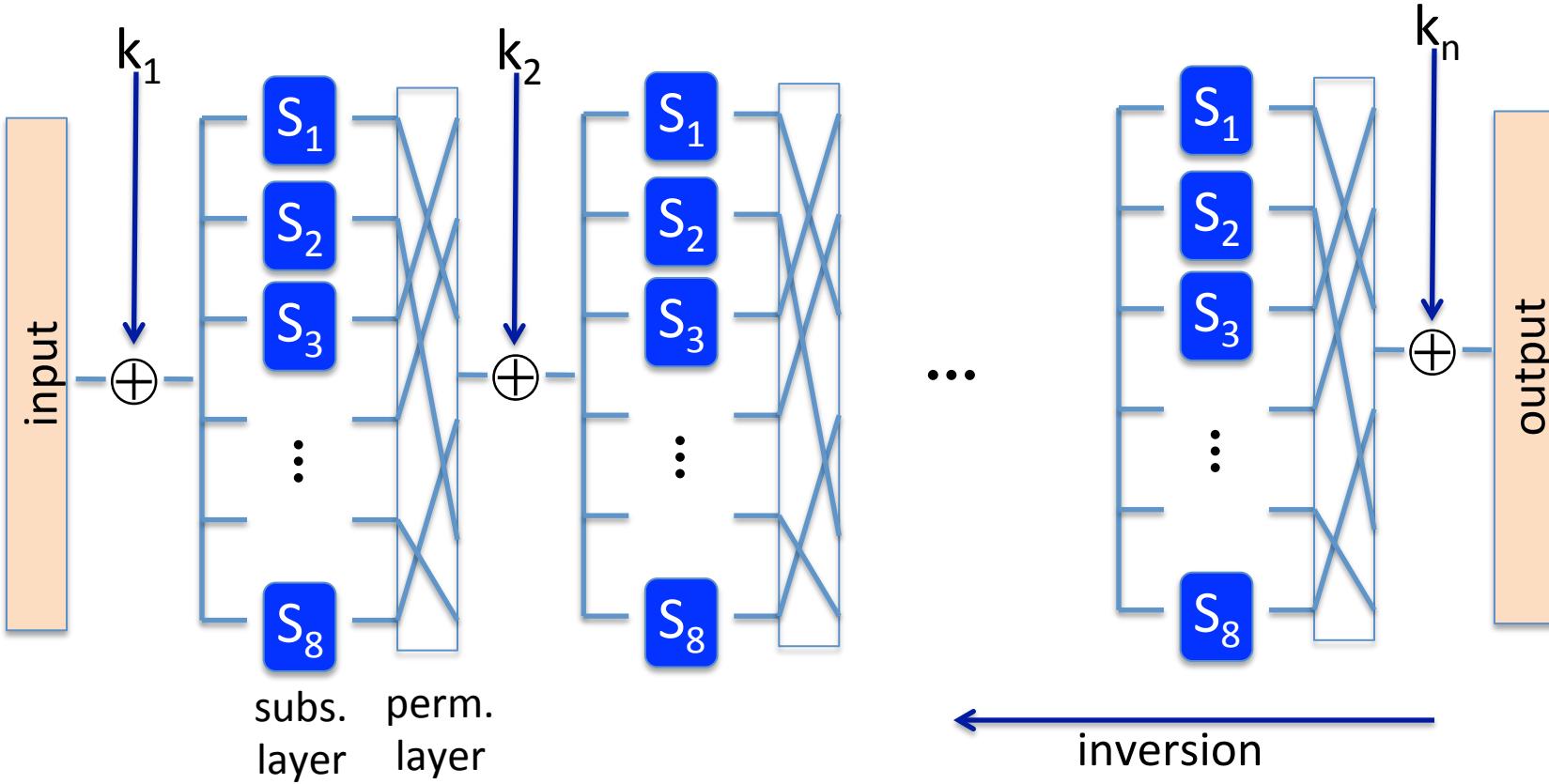
The AES block cipher

The AES process

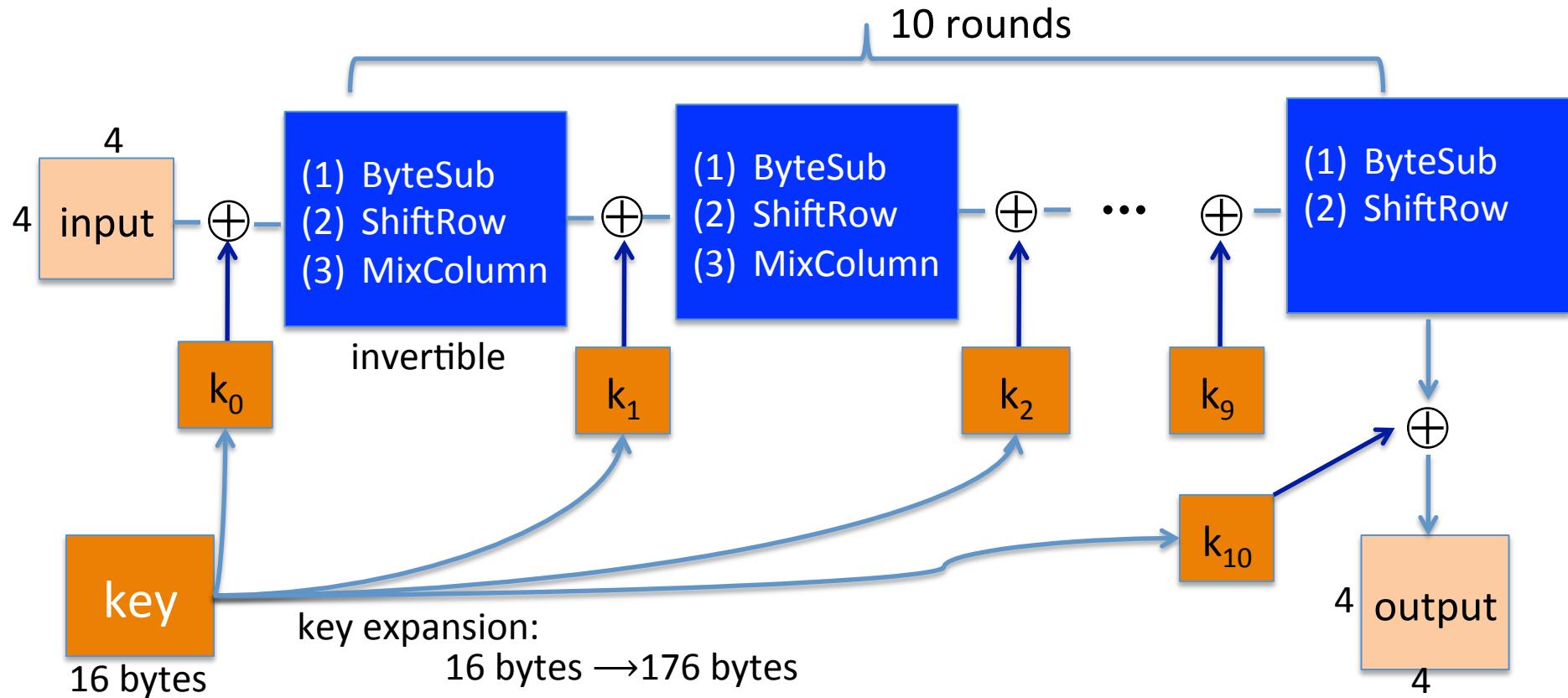
- 1997: NIST publishes request for proposal
- 1998: 15 submissions. Five claimed attacks.
- 1999: NIST chooses 5 finalists
- 2000: NIST chooses Rijndael as AES (designed in Belgium)

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

AES is a Subs-Perm network (not Feistel)



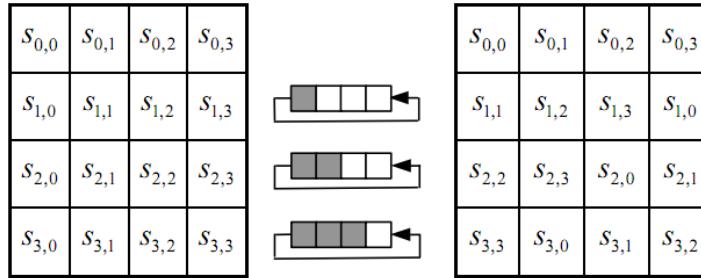
AES-128 schematic



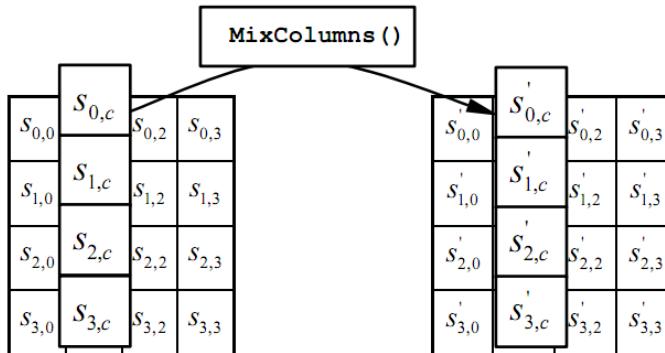
The round function

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

- **ShiftRows:**



- **MixColumns:**

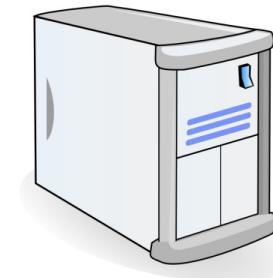
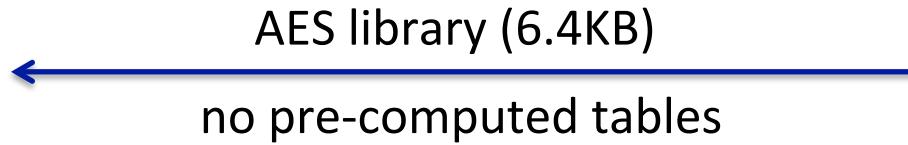


Code size/performance tradeoff

	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:



Prior to encryption:
pre-compute tables

Then encrypt using tables

<http://crypto.stanford.edu/sjcl/>

Dan Boneh

AES in hardware

AES instructions in Intel Westmere:

- **aesenc, aesenclast:** do one round of AES
128-bit registers: xmm1=state, xmm2=round key
aesenc xmm1, xmm2 ; puts result in xmm1
- **aeskeygenassist:** performs AES key expansion
- Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer

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 $\approx 2^{99}$

End of Segment



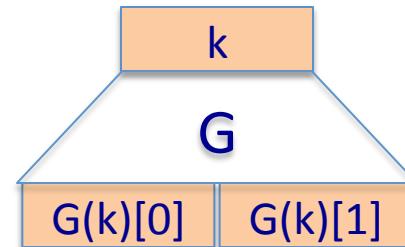
Block ciphers

Block ciphers from PRGs

Can we build a PRF from a PRG?

Let $G: K \rightarrow K^2$ be a secure PRG

Define 1-bit PRF $F: K \times \{0,1\} \rightarrow K$ as



$$F(k, x \in \{0,1\}) = G(k)[x]$$

Thm: If G is a secure PRG then F is a secure PRF

Can we build a PRF with a larger domain?

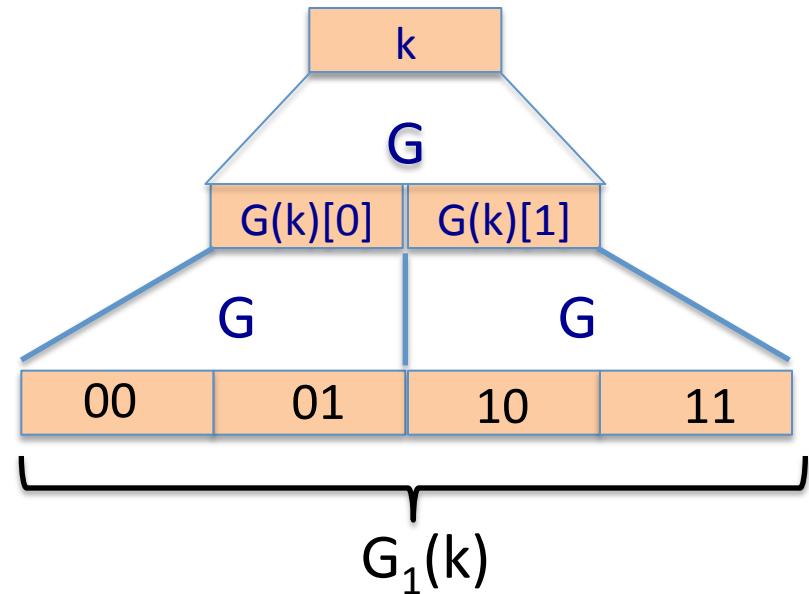
Extending a PRG

Let $G: K \rightarrow K^2$.

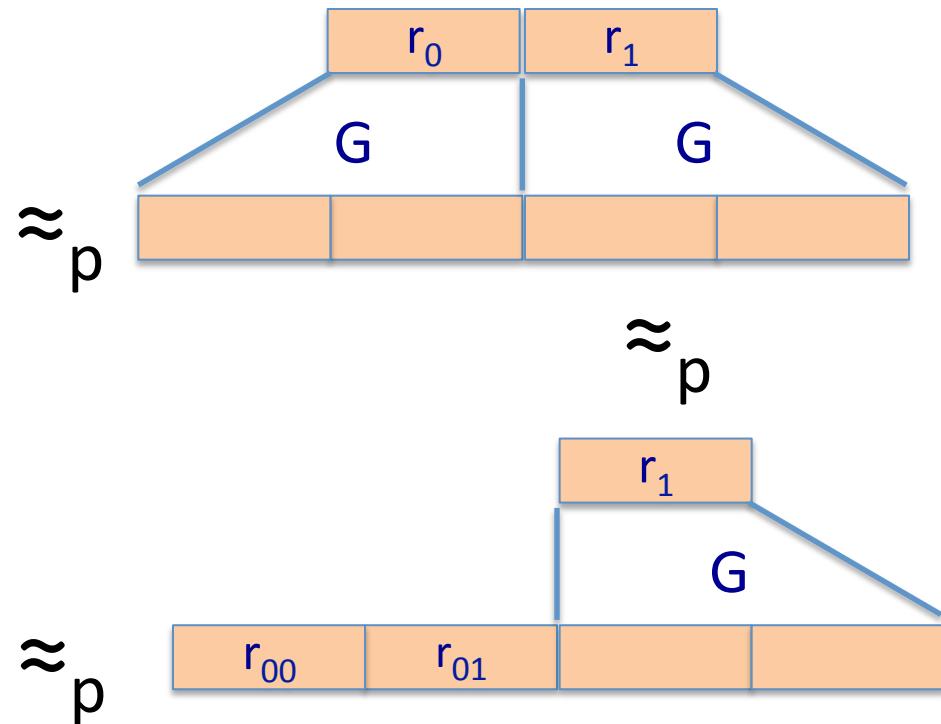
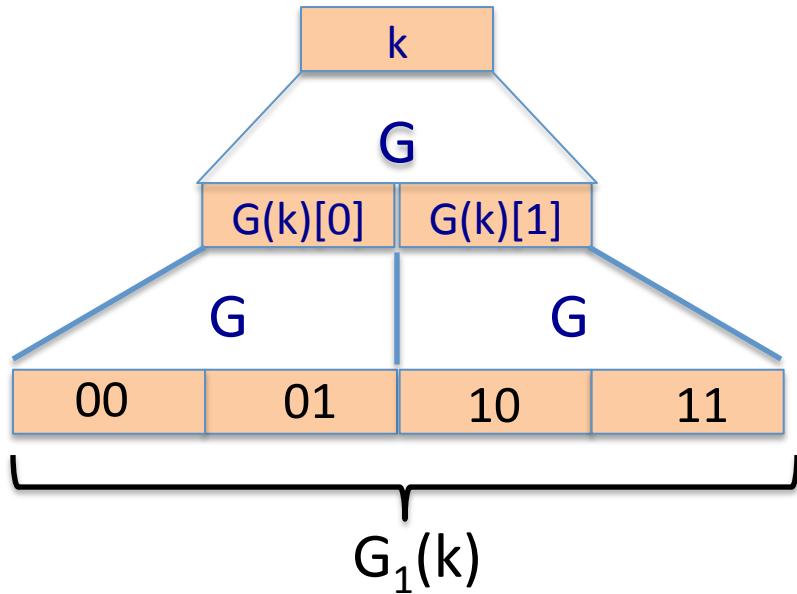
define $G_1: K \rightarrow K^4$ as $G_1(k) = G(G(k)[0]) \parallel G(G(k)[1])$

We get a 2-bit PRF:

$$F(k, x \in \{0,1\}^2) = G_1(k)[x]$$



G_1 is a secure PRG

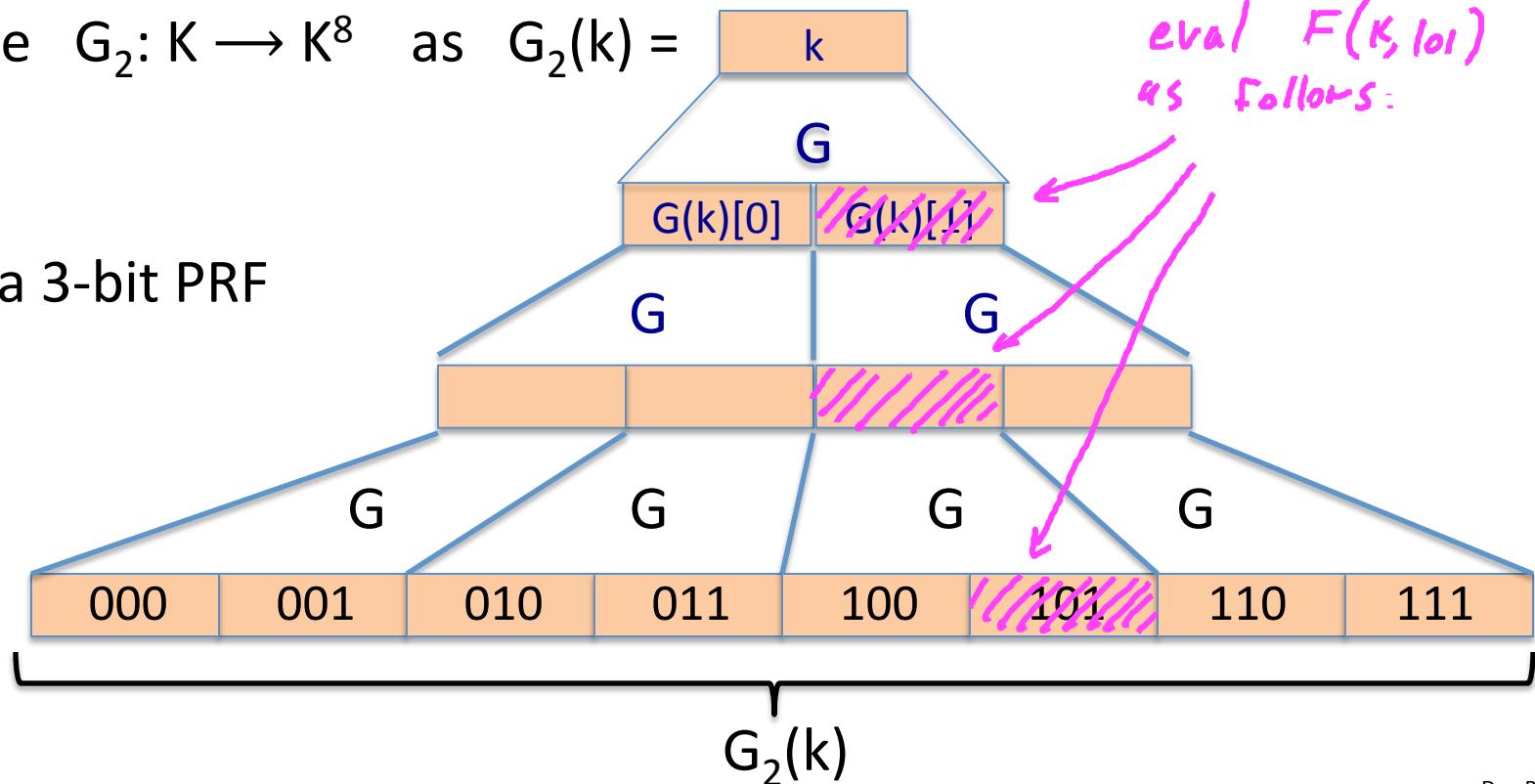


Extending more

Let $G: K \rightarrow K^2$.

define $G_2: K \rightarrow K^8$ as $G_2(k) =$

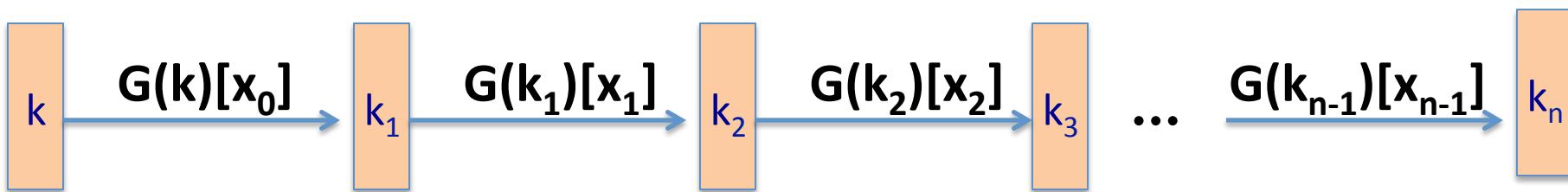
We get a 3-bit PRF



Extending even more: the GGM PRF

Let $G: K \rightarrow K^2$. define PRF $F: K \times \{0,1\}^n \rightarrow K$ as

For input $x = x_0 x_1 \dots x_{n-1} \in \{0,1\}^n$ do:



Security: G a secure PRG $\Rightarrow F$ is a secure PRF on $\{0,1\}^n$.

Not used in practice due to slow performance.

Secure block cipher from a PRG?

Can we build a secure PRP from a secure PRG?

- No, it cannot be done
-  Yes, just plug the GGM PRF into the Luby-Rackoff theorem
- It depends on the underlying PRG
-

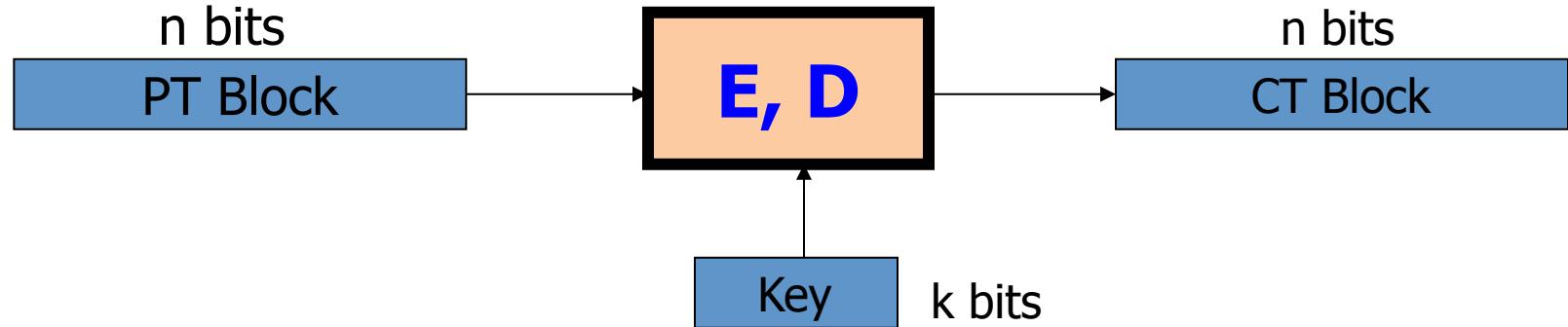
End of Segment



Using block ciphers

Review: PRPs and PRFs

Block ciphers: crypto work horse



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) :

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

such that exists “efficient” algorithm to evaluate $F(k, x)$

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

$$E: K \times X \rightarrow X$$

such that:

1. Exists “efficient” deterministic algorithm to evaluate $E(k, x)$
2. The function $E(k, \cdot)$ is one-to-one
3. Exists “efficient” inversion algorithm $D(k, x)$

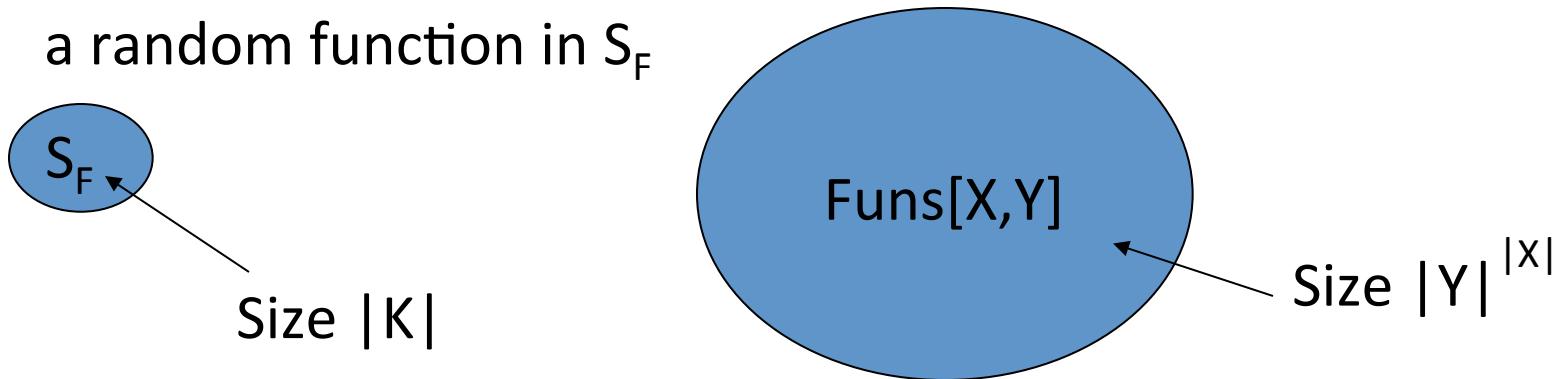
Secure PRFs

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

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

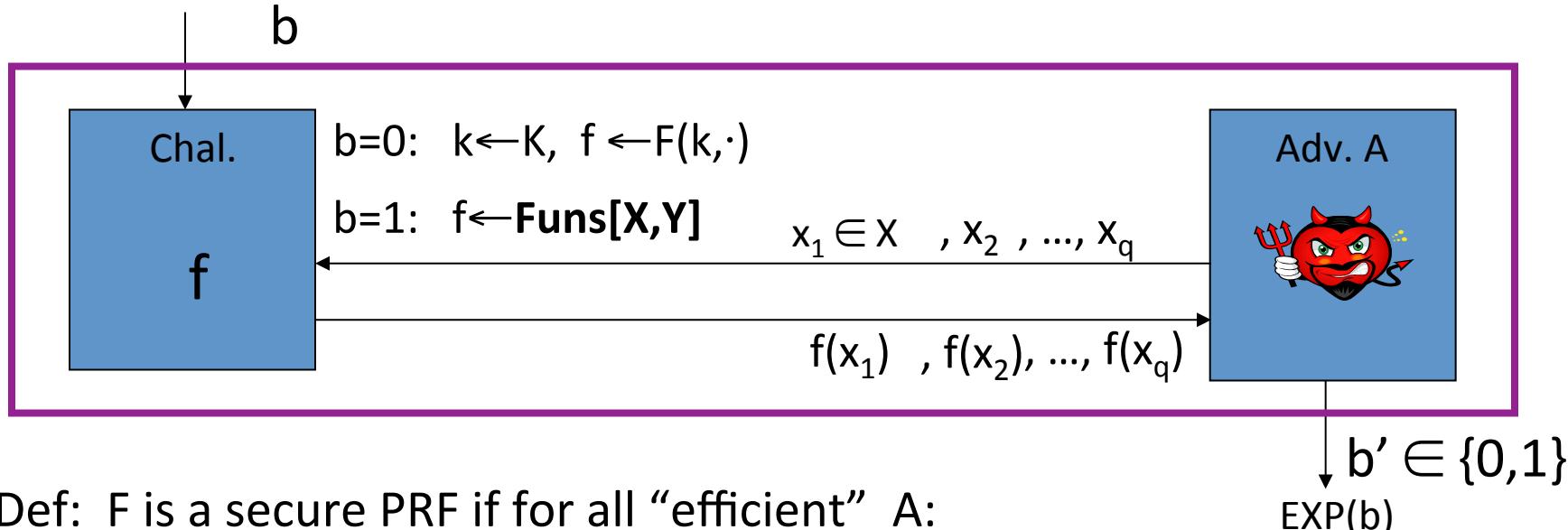
- Intuition: a PRF is **secure** if

a random function in $\text{Funs}[X,Y]$ is indistinguishable from
a random function in S_F



Secure PRF: definition

- For $b=0,1$ define experiment $\text{EXP}(b)$ as:



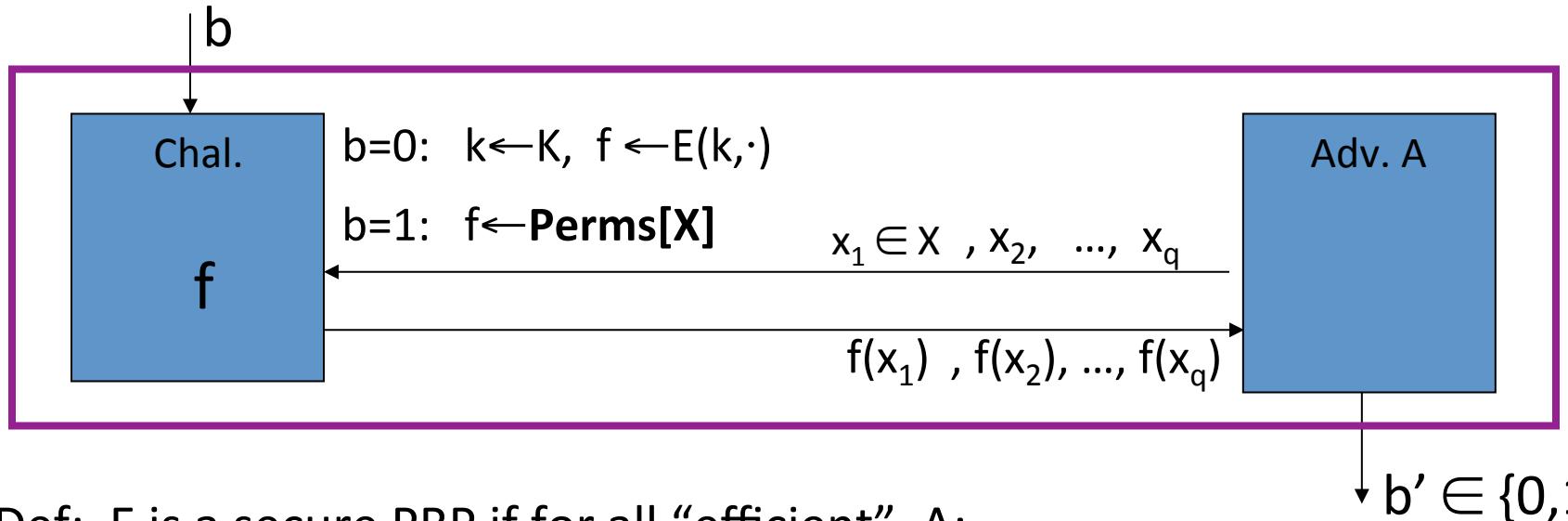
- Def: F is a secure PRF if for all “efficient” A :

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

is “negligible.”

Secure PRP (secure block cipher)

- For $b=0,1$ define experiment $\text{EXP}(b)$ as:



- Def: E is a secure PRP if for all “efficient” A :

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

is “negligible.”

Let $X = \{0,1\}$. $\text{Perms}[X]$ contains two functions

Consider the following PRP:

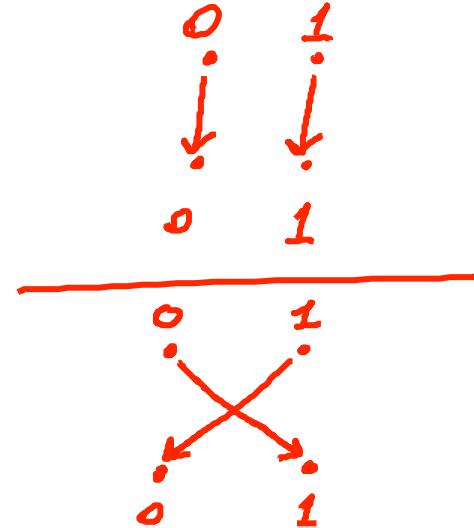
key space $K=\{0,1\}$, input space $X = \{0,1\}$,

PRP defined as:

$$E(k,x) = x \oplus k$$

Is this a secure PRP?

- Yes
- No
- It depends
-



Example secure PRPs

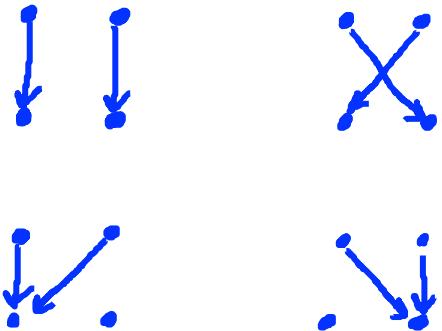
- PRPs believed to be secure: 3DES, AES, ...
AES-128: $K \times X \rightarrow X$ where $K = X = \{0,1\}^{128}$
- An example concrete assumption about AES:
All 2^{80} -time algs. A have $\text{Adv}_{\text{PRP}}[A, \text{AES}] < 2^{-40}$

Consider the 1-bit PRP from the previous question: $E(k,x) = x \oplus k$

Is it a secure PRF?

Note that $\text{Funs}[X,X]$ contains four functions

- Yes
- No
- It depends
-



Attacker A:

- (1) query $f(\cdot)$ at $x=0$ and $x=1$
- (2) if $f(0) = f(1)$ output “1”, else “0”

$$\text{Adv}_{\text{PRF}}[A,E] = |0 - \frac{1}{2}| = \frac{1}{2}$$

PRF Switching Lemma

Any secure PRP is also a secure PRF, if $|X|$ is sufficiently large.

Lemma: Let E be a PRP over (K, X)

Then for any q -query adversary A :

$$\left| \text{Adv}_{\text{PRF}}[A, E] - \text{Adv}_{\text{PRP}}[A, E] \right| < \frac{q^2 / 2|X|}{\text{neg.}}$$

\Rightarrow Suppose $|X|$ is large so that $q^2 / 2|X|$ is “negligible”

Then $\text{Adv}_{\text{PRP}}[A, E]$ “negligible” \Rightarrow $\text{Adv}_{\text{PRF}}[A, E]$ “negligible”

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

End of Segment



Using block ciphers

Modes of operation:
one time key

example: encrypted email, new key for every message.

Using PRPs and PRFs

Goal: build “secure” encryption from a secure PRP (e.g. AES).

This segment: **one-time keys**

1. Adversary’s power:

Adv sees only one ciphertext (one-time key)

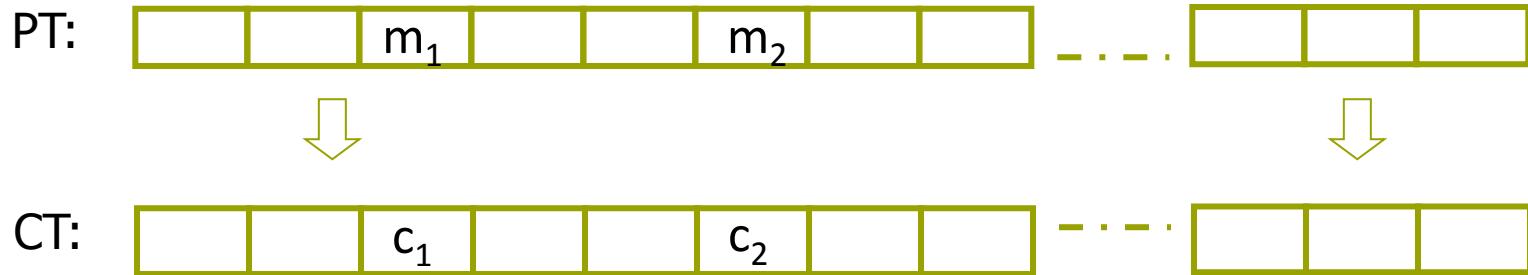
2. Adversary’s goal:

Learn info about PT from CT (semantic security)

Next segment: many-time keys (a.k.a chosen-plaintext security)

Incorrect use of a PRP

Electronic Code Book (ECB):



Problem:

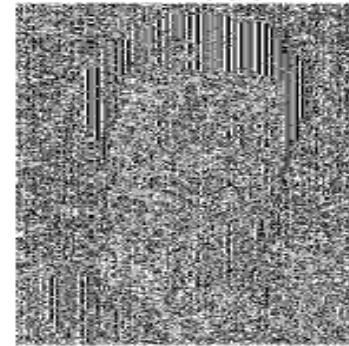
- if $m_1 = m_2$ then $c_1 = c_2$

In pictures

An example plaintext



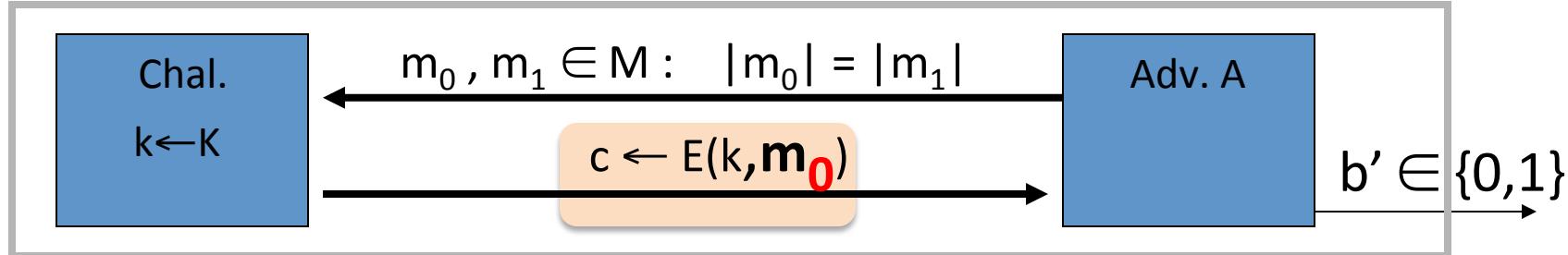
Encrypted with AES in ECB mode



(courtesy B. Preneel)

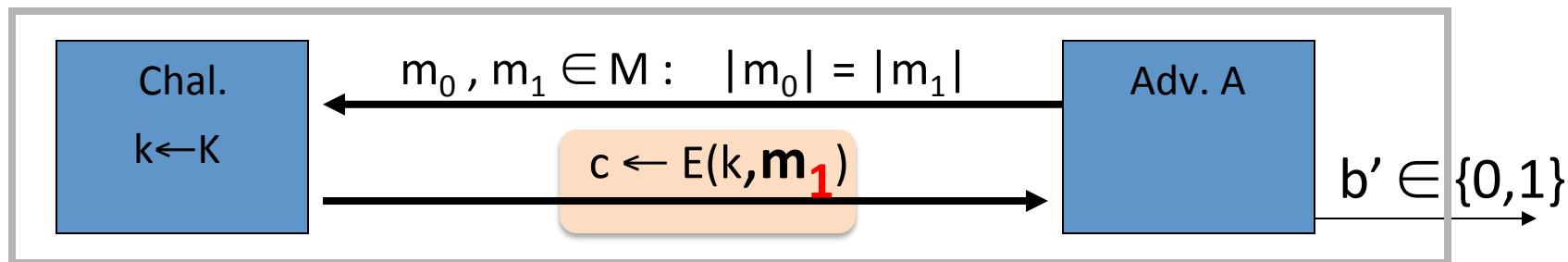
Semantic Security (one-time key)

$\text{EXP}(0):$



one time key \Rightarrow adversary sees only one ciphertext

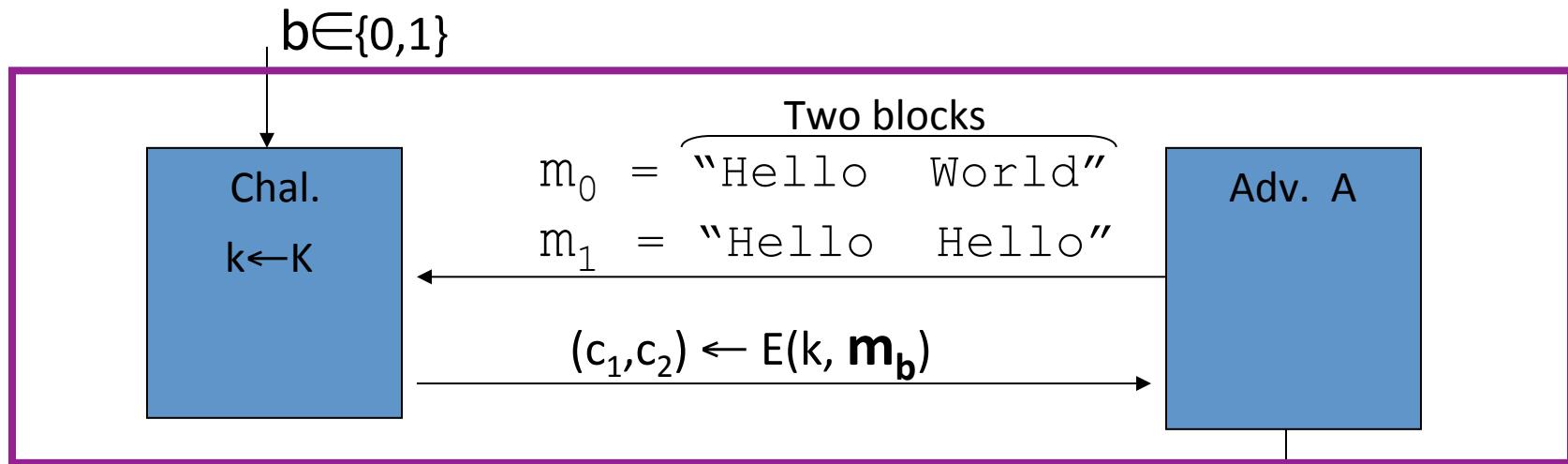
$\text{EXP}(1):$



$$\text{Adv}_{\text{SS}}[\text{A}, \text{OTP}] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right| \text{ should be "neg."}$$

ECB is not Semantically Secure

ECB is not semantically secure for messages that contain more than one block.



Then $\text{Adv}_{SS} [A, \text{ECB}] =$

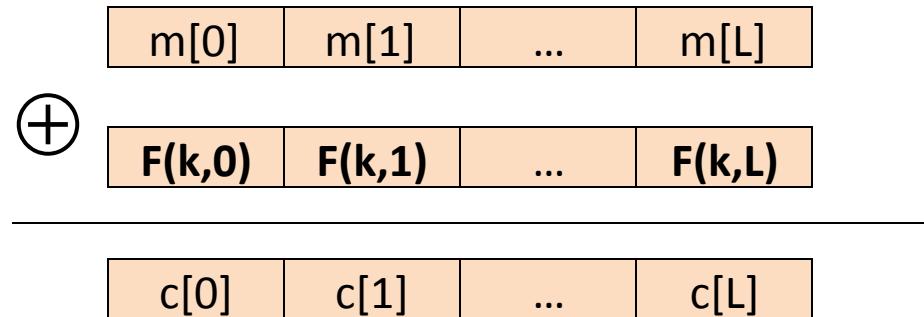
If $c_1=c_2$ output 0, else output 1

Secure Construction I

Deterministic counter mode from a PRF $F : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$

- $E_{DETCTR}(k, m) =$

(e.g. $n=128$)



⇒ Stream cipher built from a PRF (e.g. AES, 3DES)

Det. counter-mode security

Theorem: For any $L > 0$,

If F is a secure PRF over (K, X, X) then

E_{DETCTR} is sem. sec. cipher over (K, X^L, X^L) .

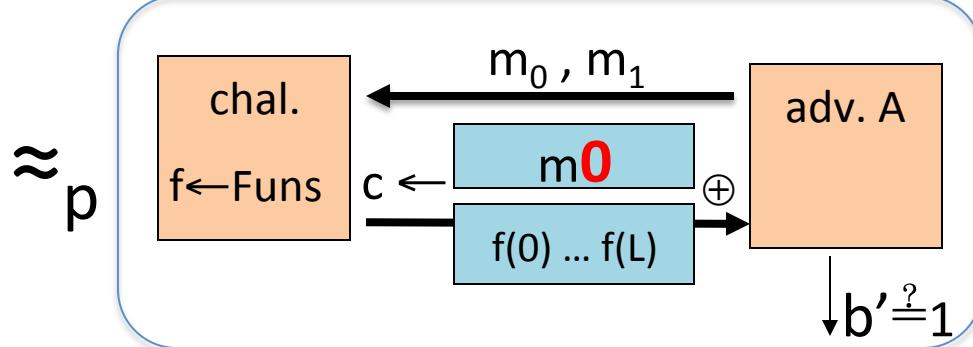
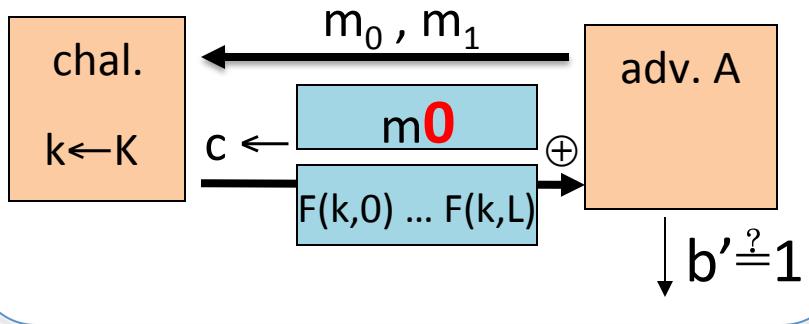
In particular, for any eff. adversary A attacking E_{DETCTR} there exists an eff. PRF adversary B s.t.:

$$\text{Adv}_{SS}[A, E_{DETCTR}] = 2 \cdot \text{Adv}_{\text{PRF}}[B, F]$$

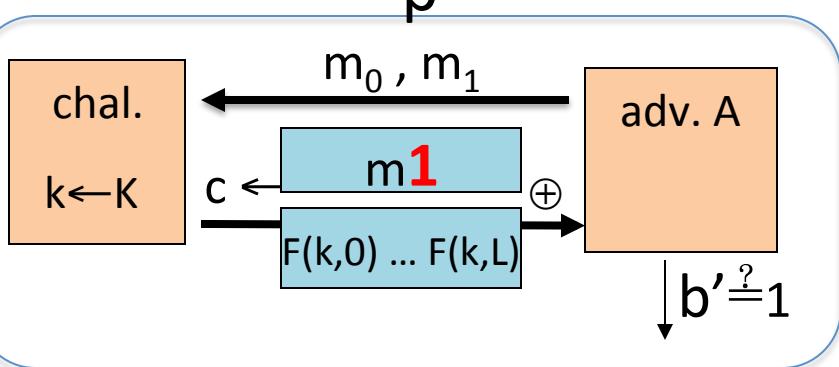
$\text{Adv}_{\text{PRF}}[B, F]$ is negligible (since F is a secure PRF)

Hence, $\text{Adv}_{SS}[A, E_{DETCTR}]$ must be negligible.

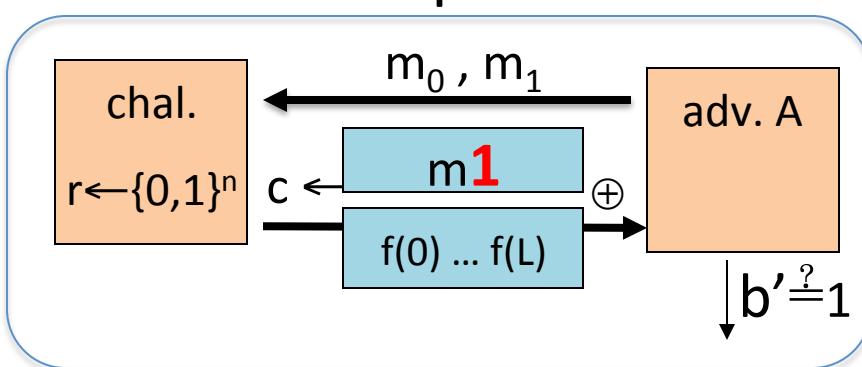
Proof



\approx_p



\approx_p



End of Segment



Using block ciphers

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.

Semantic Security for many-time key

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

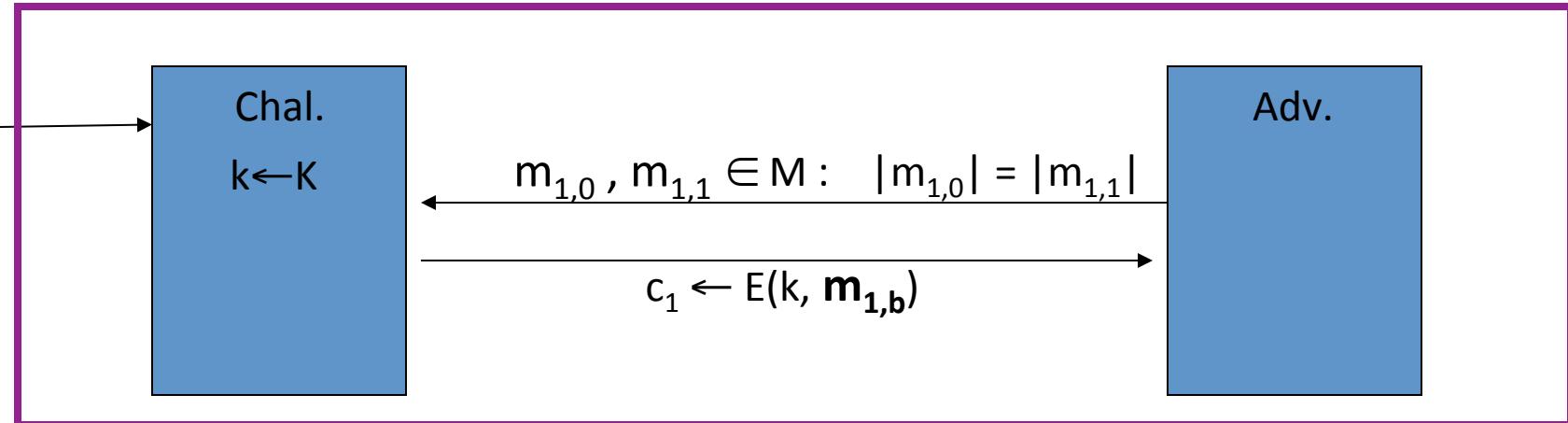
Adversary's power: chosen-plaintext attack (CPA)

- Can obtain the encryption of arbitrary messages of his choice
(conservative modeling of real life)

Adversary's goal: Break semantic security

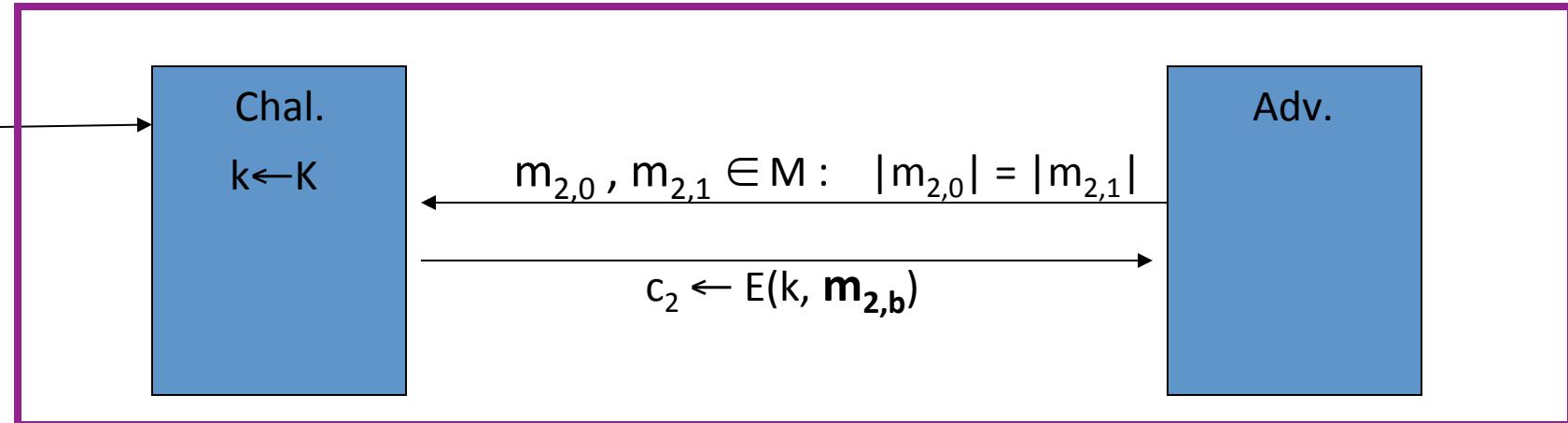
Semantic Security for many-time key

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



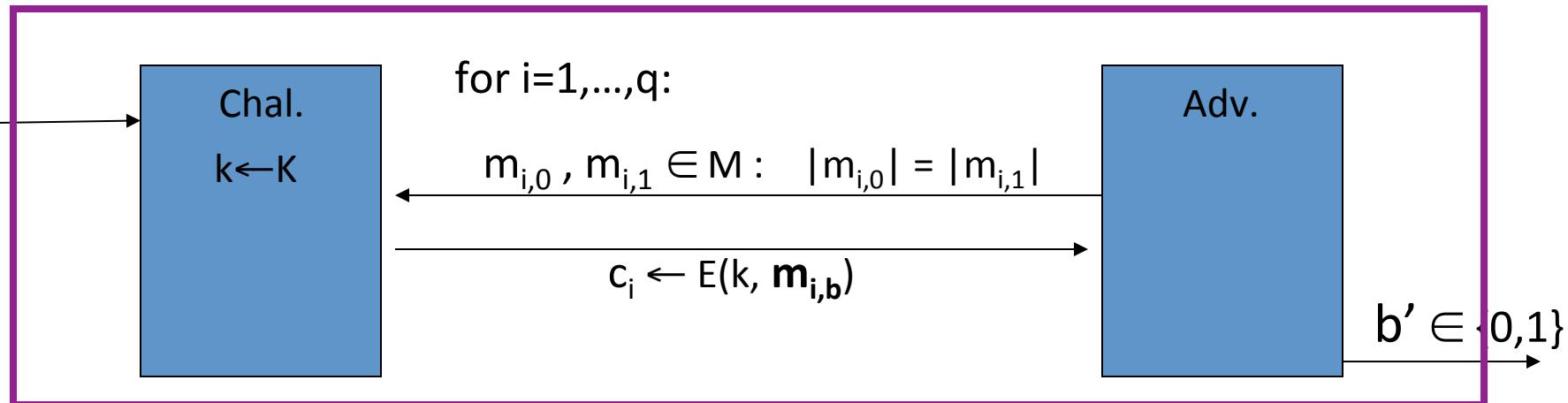
Semantic Security for many-time key

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



Semantic Security for many-time key (CPA security)

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



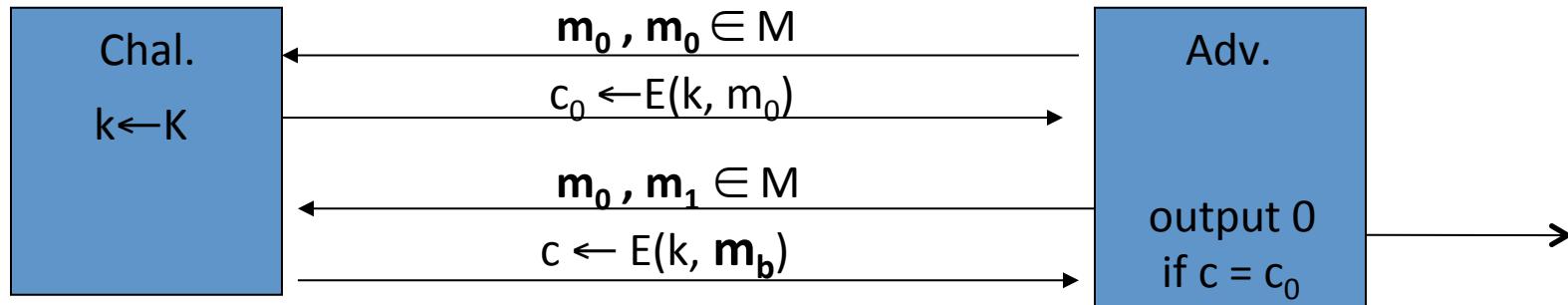
if adv. wants $c = E(k, m)$ it queries with $m_{j,0} = m_{j,1} = m$

Def: E is sem. sec. under CPA if for all “efficient” A :

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

Ciphers insecure under CPA

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



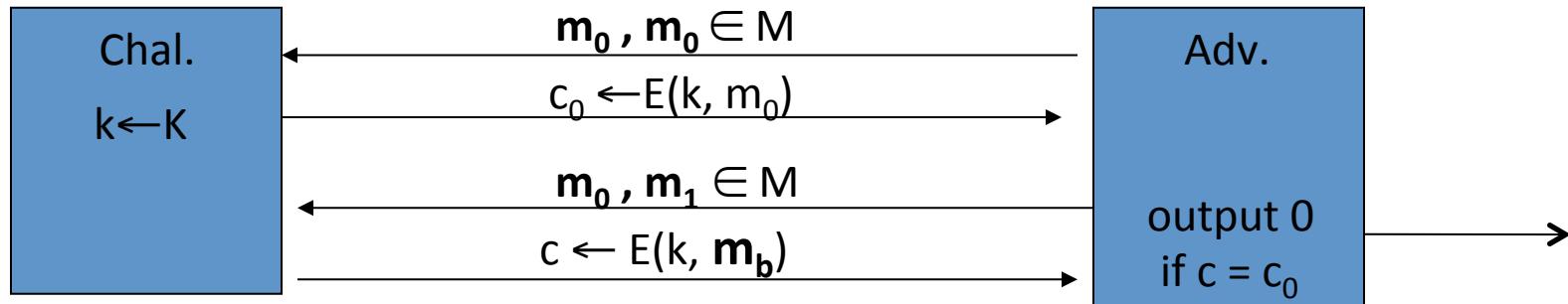
So what?

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:

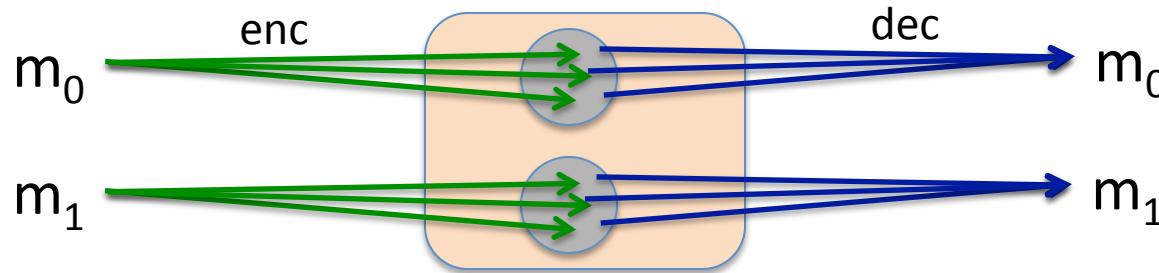


If secret key is to be used multiple times \Rightarrow

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"

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

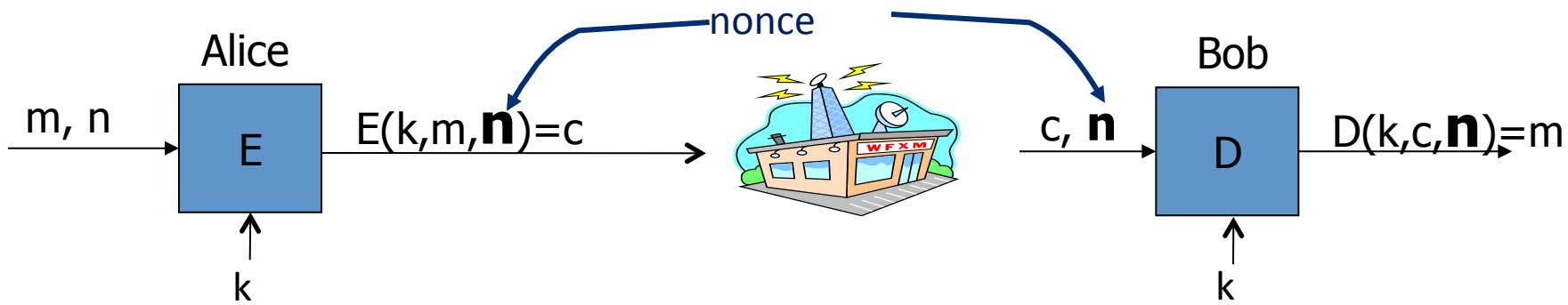
$$\text{Exp}(r, F(r) \oplus m)$$

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

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

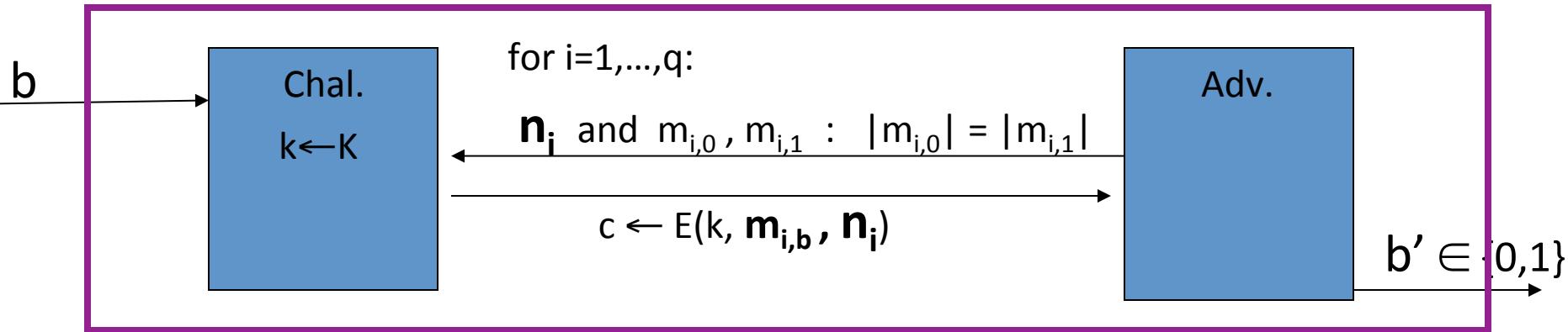
Solution 2: nonce-based Encryption



- nonce n : a value that changes from msg to msg.
 (k, n) pair never used more than once
- method 1: 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
- method 2: encryptor chooses a **random nonce**, $n \leftarrow \mathcal{N}$

CPA security for nonce-based encryption

System should be secure when nonces are chosen adversarially.



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

Def: nonce-based E is sem. sec. under CPA if for all “efficient” A :

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

Let $F: K \times R \rightarrow M$ be a secure PRF. Let $r = 0$ initially.

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

$\tilde{\tau}_p(r, f(r) \oplus m)$

Is E CPA secure nonce-based encryption?

- Yes, whenever F is a secure PRF
- No, there is always a nonce-based CPA attack on this system
- Yes, but only if R is large enough so r never repeats
- It depends on what F is used

End of Segment



Using block ciphers

Modes of operation:
many time key (CBC)

Example applications:

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

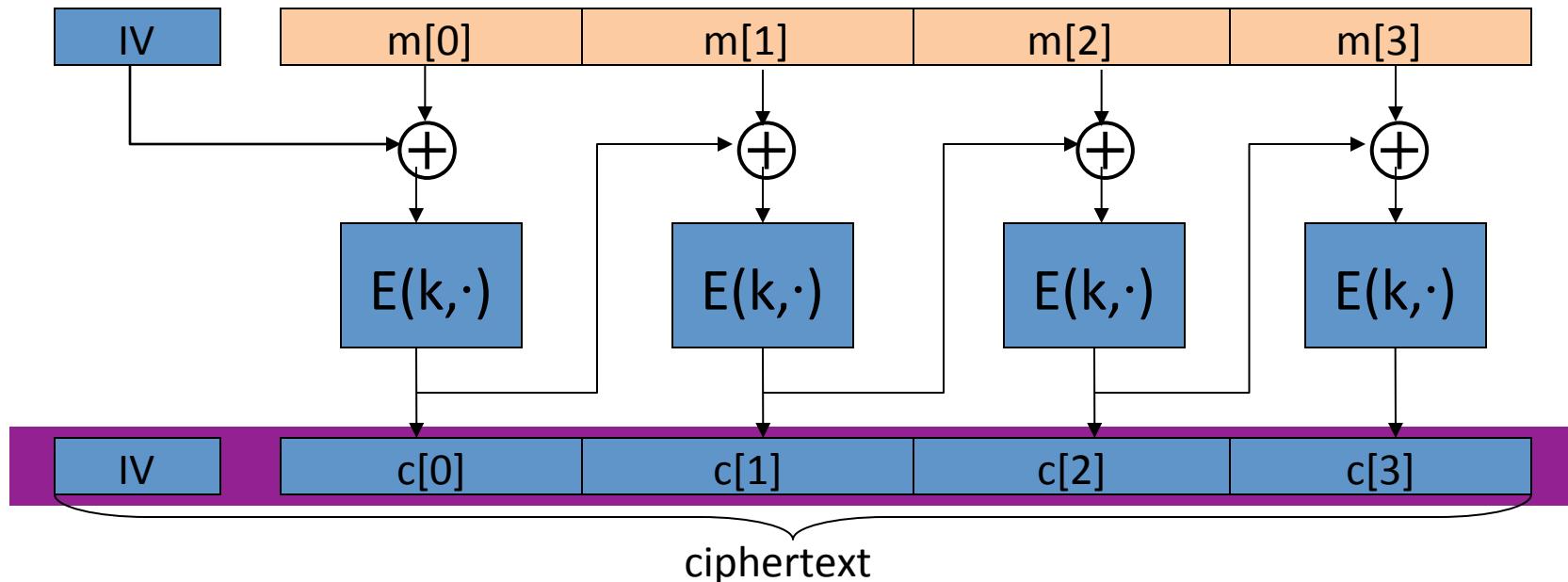
Construction 1: CBC with random IV

Let (E, D) be a PRP.

$$E : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$$

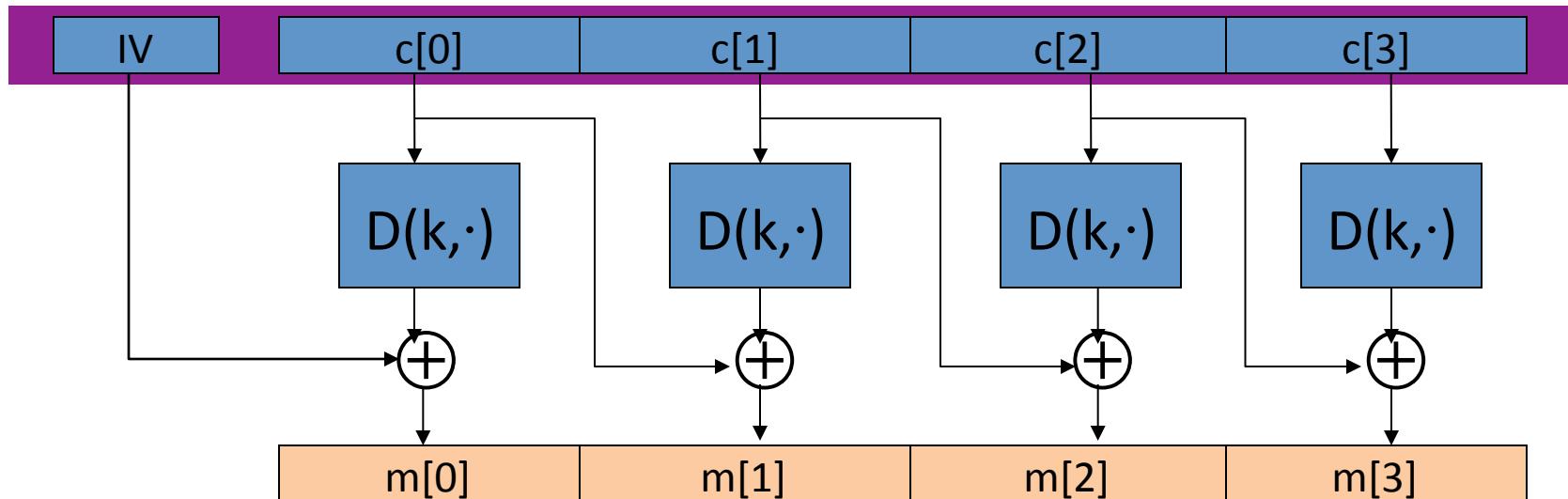
$E_{\text{CBC}}(k, m)$: choose random $\text{IV} \in X$ and do:

$$\text{IV} \in \{0,1\}^n$$



Decryption circuit

In symbols: $c[0] = E(k, IV \oplus m[0]) \Rightarrow m[0] =$



CBC: CPA Analysis

CBC Theorem: For any $L > 0$,

If E is a secure PRP over (K, X) then

E_{CBC} is a sem. sec. under CPA over (K, X^L, X^{L+1}) .

In particular, for a q -query adversary A attacking E_{CBC}
there exists a PRP adversary B s.t.:

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CBC}}] \leq 2 \cdot \text{Adv}_{\text{PRP}}[B, E] + 2 q^2 L^2 / |X|$$

Note: CBC is only secure as long as $q^2 L^2 \ll |X|$

An example

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CBC}}] \leq 2 \cdot \text{PRP Adv}[B, E] + \mathbf{2 q^2 L^2 / |X|}$$

$q = \# \text{ messages encrypted with } k$, $L = \text{length of max message}$

Suppose we want $\text{Adv}_{\text{CPA}}[A, E_{\text{CBC}}] \leq 1/2^{32} \iff q^2 L^2 / |X| < 1/2^{32}$

- AES: $|X| = 2^{128} \Rightarrow q L < 2^{48}$

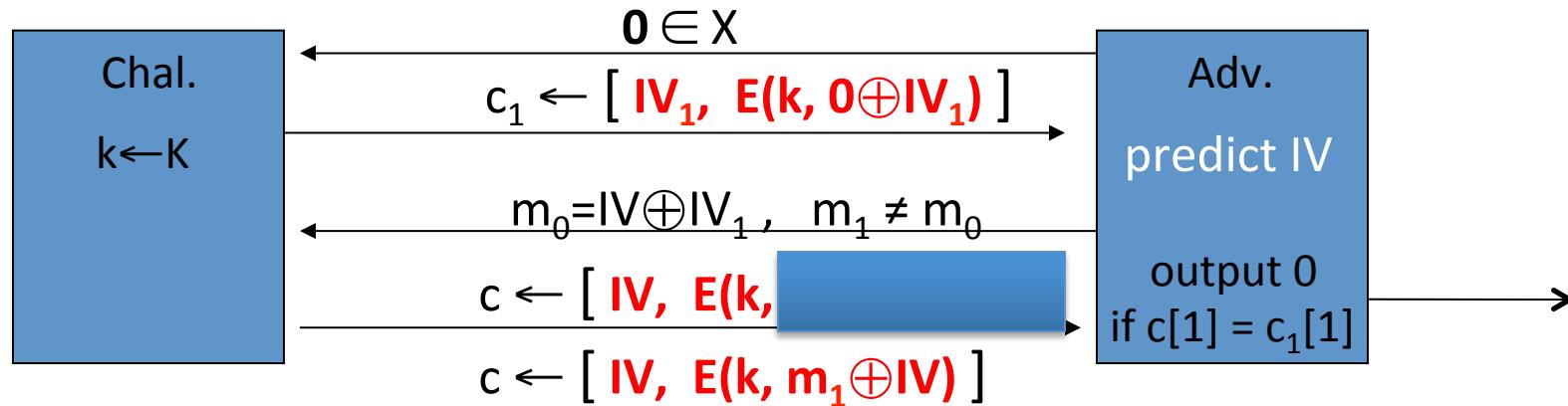
So, after 2^{48} AES blocks, must change key

- 3DES: $|X| = 2^{64} \Rightarrow q L < 2^{16}$

Warning: an attack on CBC with rand. IV

CBC where attacker can predict the IV is not CPA-secure !!

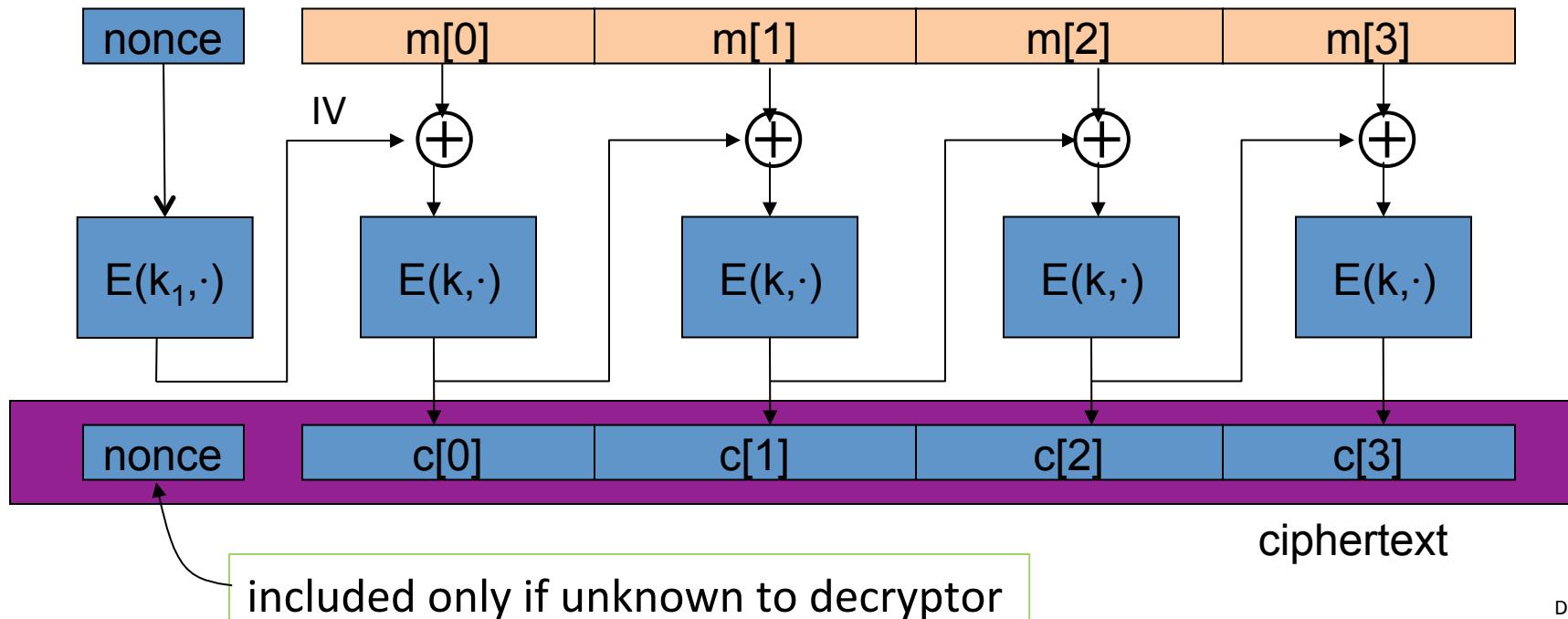
Suppose given $c \leftarrow E_{\text{CBC}}(k, m)$ can predict IV for next message



Bug in SSL/TLS 1.0: IV for record #i is last CT block of record #(i-1)

Construction 1': nonce-based CBC

- Cipher block chaining with unique nonce: key = (k, k_1)
unique nonce means: (key, n) pair is used for only one message



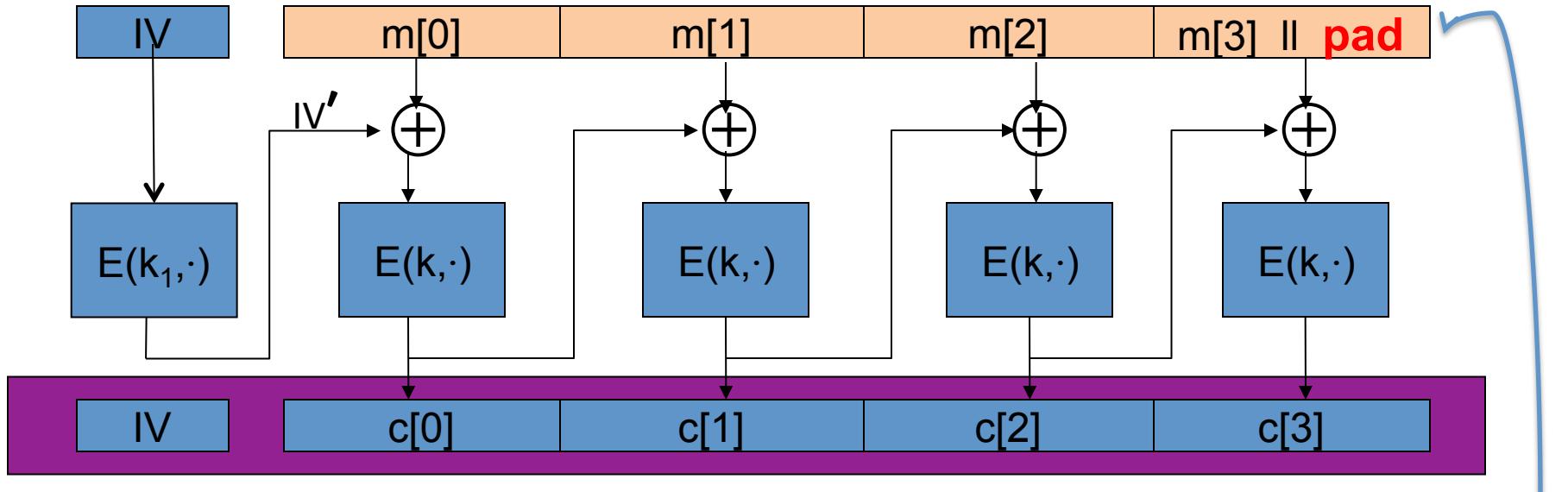
An example Crypto API (OpenSSL)

```
void AES_cbc_encrypt(  
    const unsigned char *in,  
    unsigned char *out,  
    size_t length,  
    const AES_KEY *key,  
    unsigned char *ivec,           ← user supplies IV  
    AES_ENCRYPT or AES_DECRYPT);
```

otherwise, no
CPA security

When nonce is non random need to encrypt it before use

A CBC technicality: padding



TLS: for $n > 0$, n byte pad is $\boxed{n \ n \ n \dots n}$

if no pad needed, add a dummy block

removed
during
decryption

End of Segment



Using block ciphers

Modes of operation:
many time key (CTR)

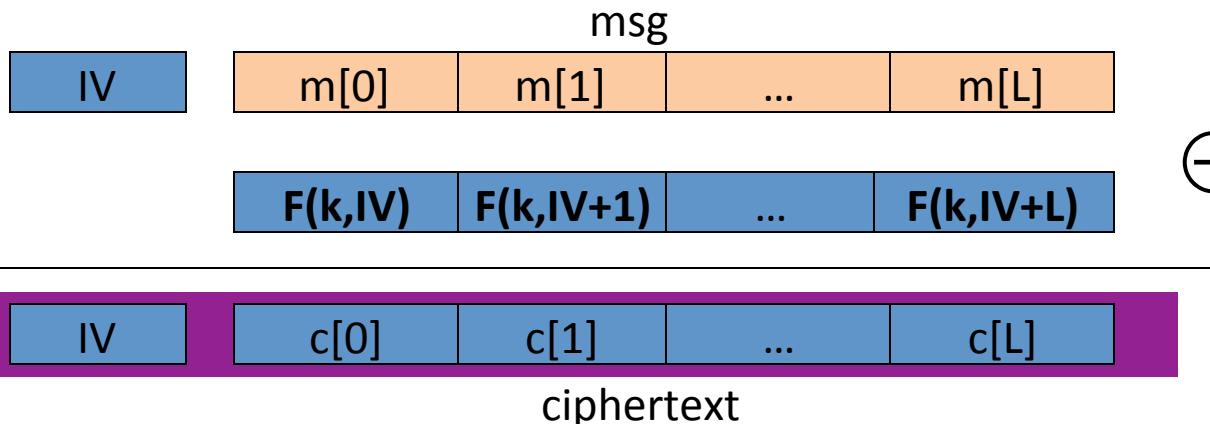
Example applications:

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

Construction 2: rand ctr-mode

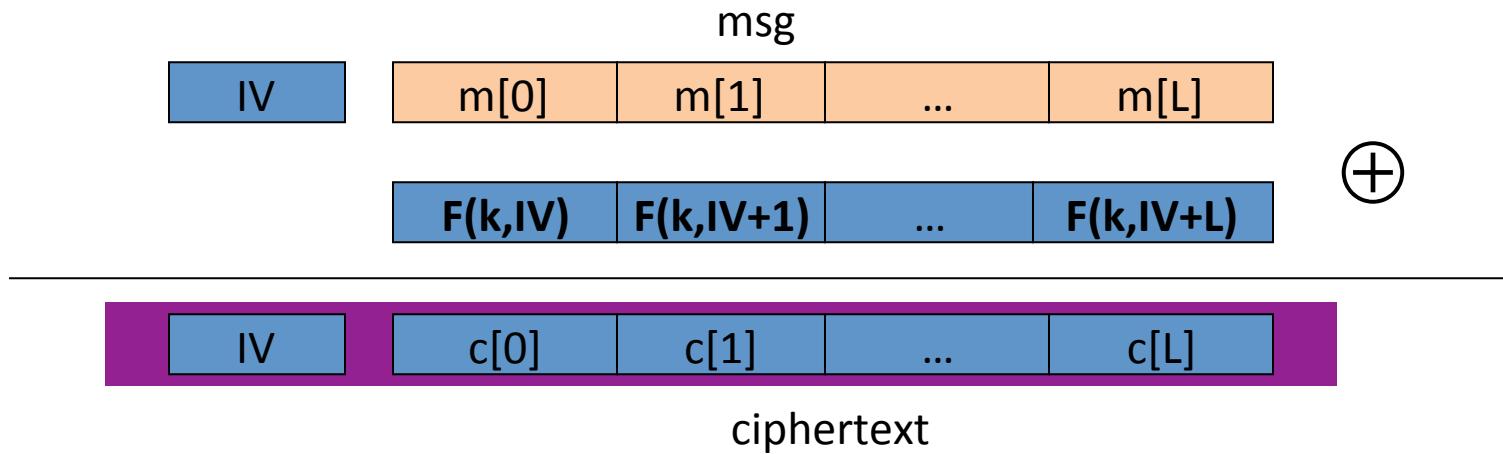
Let $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRF.

$E(k,m)$: choose a random $\text{IV} \in \{0,1\}^n$ and do:

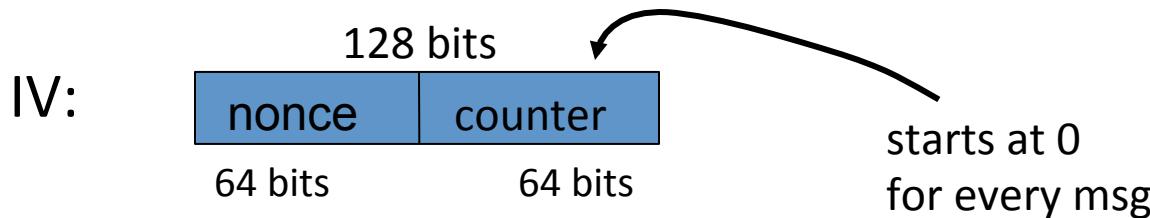


note: parallelizable (unlike CBC)

Construction 2': nonce ctr-mode



To ensure $F(k, x)$ is never used more than once, choose IV as:



rand ctr-mode (rand. IV): CPA analysis

- Counter-mode Theorem: For any $L > 0$,

If F is a secure PRF over (K, X, X) then

E_{CTR} is a sem. sec. under CPA over (K, X^L, X^{L+1}) .

In particular, for a q -query adversary A attacking E_{CTR} there exists a PRF adversary B s.t.:

$$\text{Adv}_{CPA}[A, E_{CTR}] \leq 2 \cdot \text{Adv}_{PRF}[B, F] + 2q^2 L / |X|$$

Note: ctr-mode only secure as long as $q^2 L \ll |X|$. Better than CBC !

An example

$$\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 2 \cdot \text{Adv}_{\text{PRF}}[B, E] + 2q^2 L / |X|$$

$q = \# \text{ messages encrypted with } k$, $L = \text{length of max message}$

Suppose we want $\text{Adv}_{\text{CPA}}[A, E_{\text{CTR}}] \leq 1/2^{32} \iff q^2 L / |X| < 1/2^{32}$

- AES: $|X| = 2^{128} \Rightarrow q L^{1/2} < 2^{48}$

So, after 2^{32} CTs each of len 2^{32} , must change key
(total of 2^{64} AES blocks)

Comparison: ctr vs. CBC

	CBC	ctr mode
uses	PRP	PRF
parallel processing	No	Yes
Security of rand. enc.	$q^2 L^2 \ll X $	$q^2 L \ll X $
dummy padding block	Yes	No
1 byte msgs (nonce-based)	16x expansion	no expansion

(for CBC, dummy padding block can be solved using ciphertext stealing)

Summary

- PRPs and PRFs: a useful abstraction of block ciphers.
- We examined two security notions: (security against eavesdropping)
 1. Semantic security against one-time CPA.
 2. Semantic security against many-time CPA.
- Note: neither mode ensures data integrity.
- Stated security results summarized in the following table:

Goal \ Power	one-time key	Many-time key (CPA)	CPA and integrity
Sem. Sec.	steam-ciphers det. ctr-mode	rand CBC rand ctr-mode	later
?	?	?	?

Further reading

- A concrete security treatment of symmetric encryption:
Analysis of the DES modes of operation,
M. Bellare, A. Desai, E. Jokipii and P. Rogaway, FOCS 1997
- Nonce-Based Symmetric Encryption, P. Rogaway, FSE 2004

End of Segment



Message integrity

Message Auth. Codes

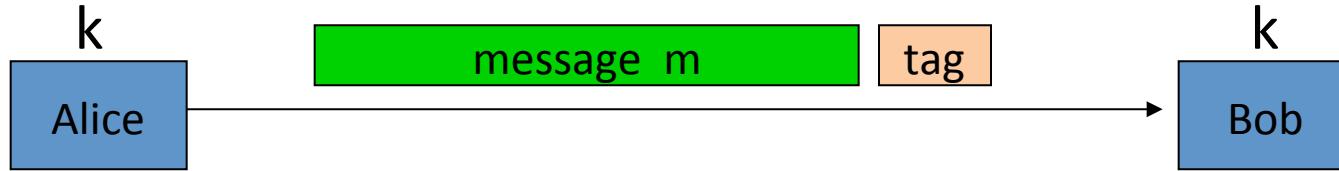
Message Integrity

Goal: **integrity**, no confidentiality.

Examples:

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

Message integrity: MACs



Generate tag:

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

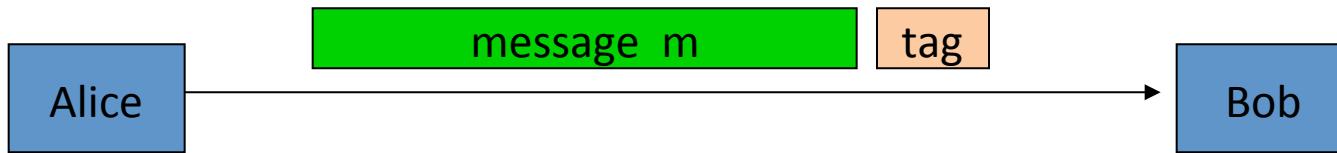
Verify tag:

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

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



Generate tag:

$$\text{tag} \leftarrow \text{CRC}(m)$$

Verify tag:

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

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

Attacker's goal: **existential forgery**

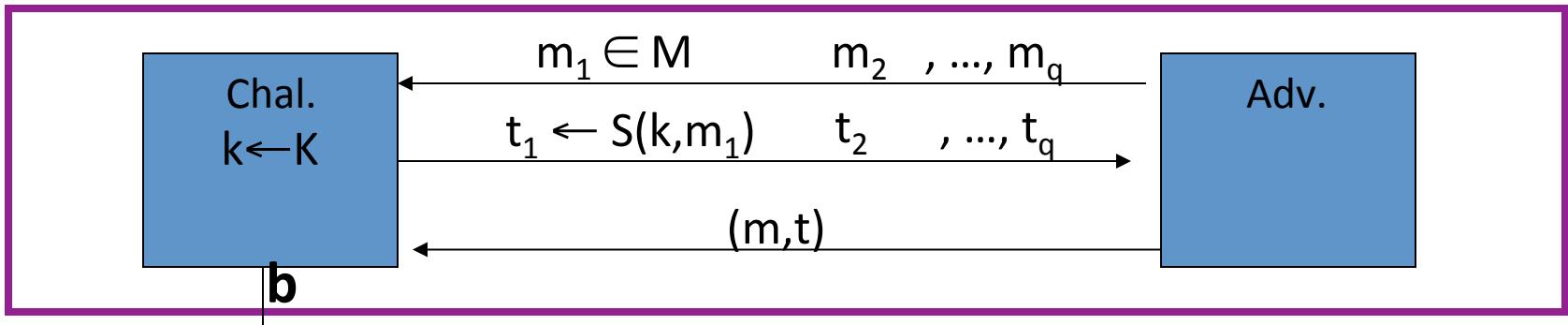
- produce some new valid message/tag pair (m, t) .

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

-
- ⇒ attacker cannot produce a valid tag for a new message
 - ⇒ 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 :

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

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

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

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

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
-

$$\text{Adv}[A, I] = \frac{1}{2}$$

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
-

$$A \in [A, I] = \mathbb{F}_{32}$$

Example: protecting system files

Suppose at install time the system computes:



\dots



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. $S(k,m) \rightarrow t$ 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) .

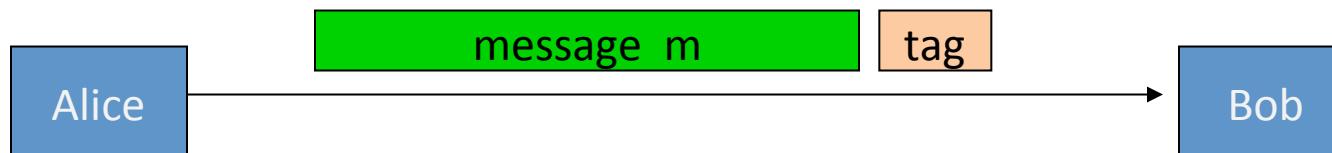
$$(m, t) \notin \{ (m_1, t_1), \dots, (m_q, t_q) \}$$

⇒ 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.



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

accept msg if

$\text{tag} = F(k, m)$

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
-

$$Adv[A, I_F] = 1/1024$$

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.:

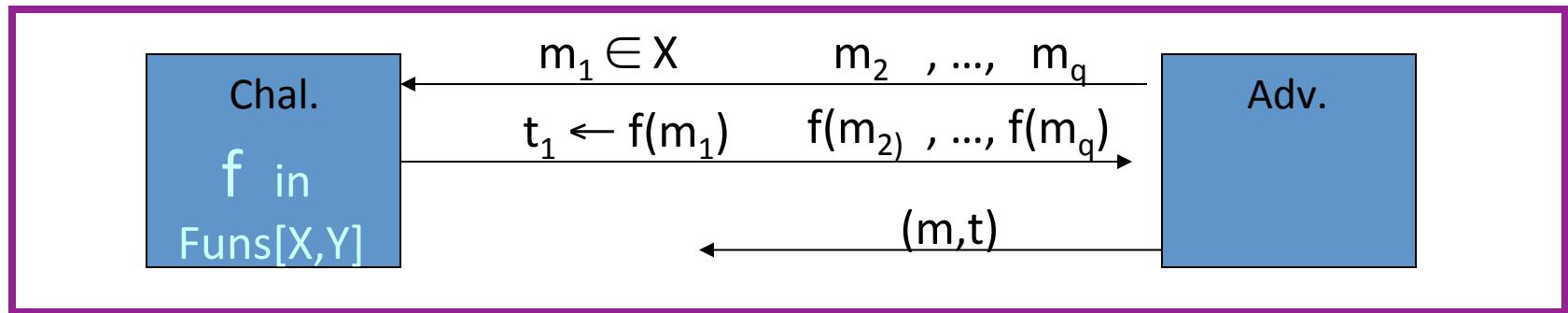
$$\text{Adv}_{\text{MAC}}[A, I_F] \leq \text{Adv}_{\text{PRF}}[B, F] + 1/|Y|$$

$\Rightarrow I_F$ 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 $F_t(k,m) = \underbrace{F(k,m)[1\dots t]}_{\text{first } t\text{-bit}\text{ of output}}$ for all $1 \leq t \leq n$

⇒ 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 $w \geq 64$)

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:

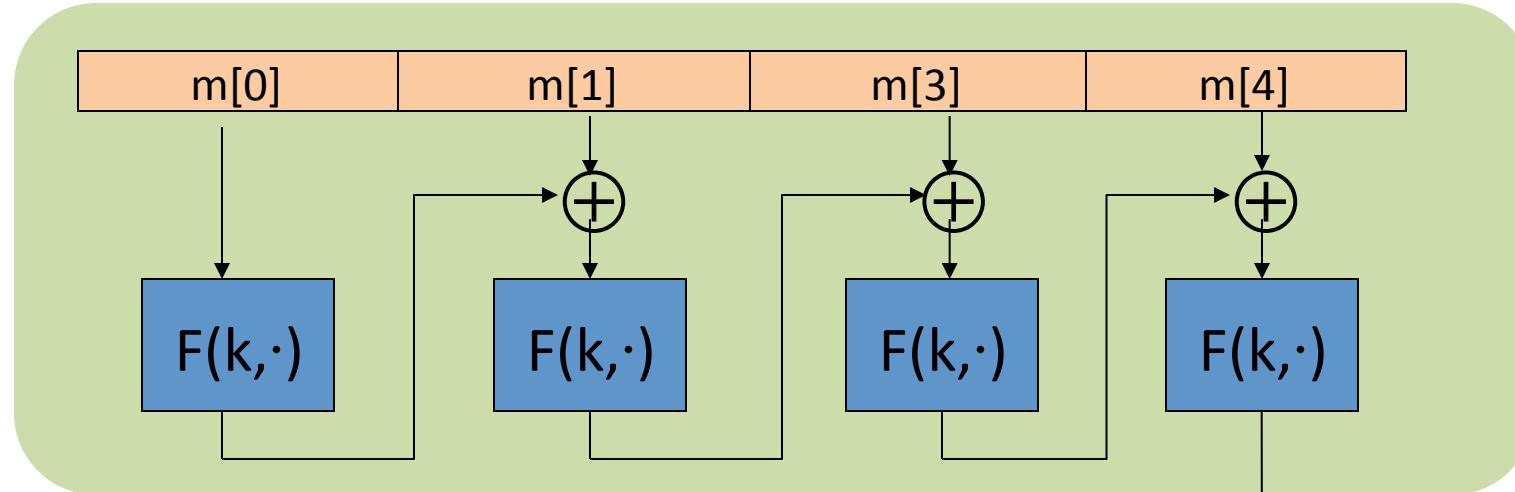
given a PRF for short messages (AES)

construct a PRF for long messages

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

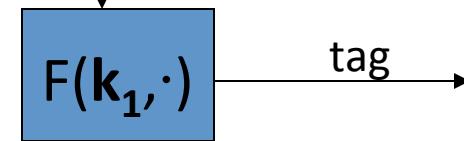
Construction 1: encrypted CBC-MAC

raw CBC



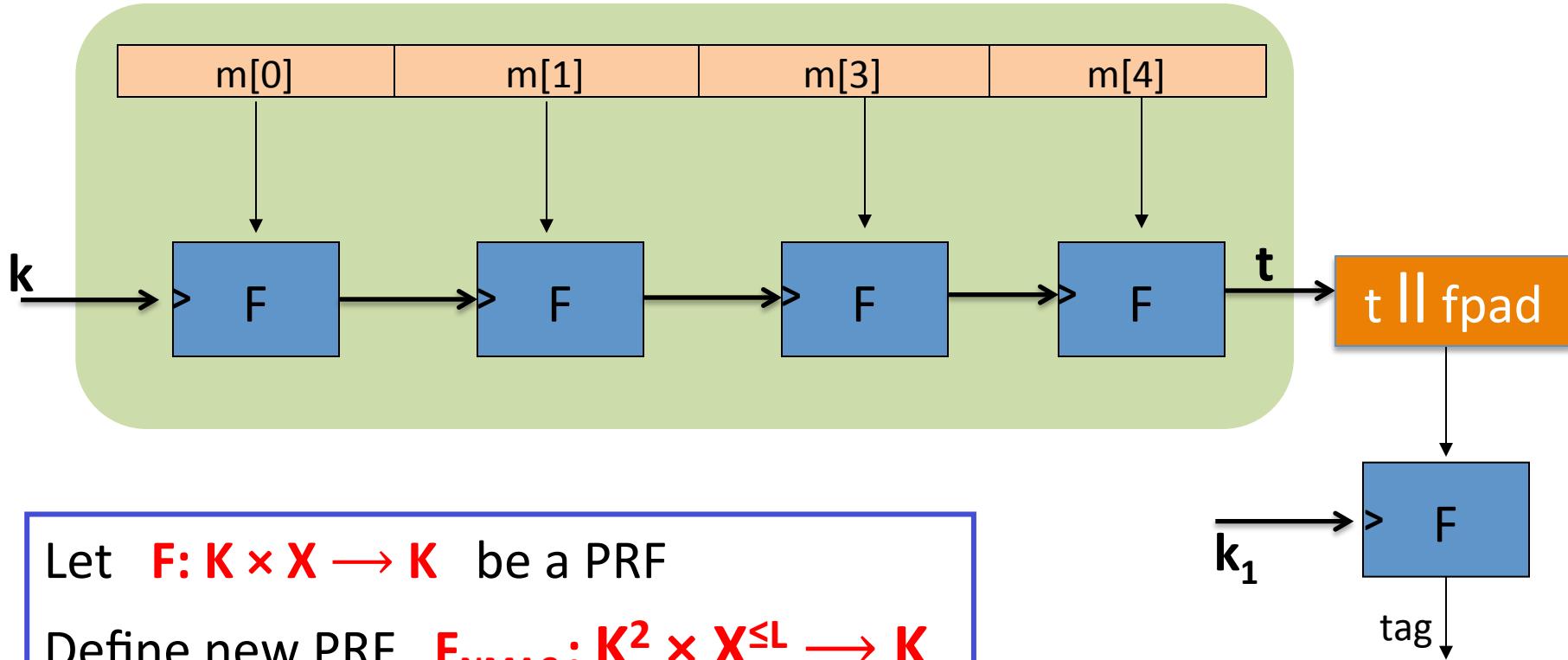
Let $F: K \times X \rightarrow X$ be a PRP

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



Construction 2: NMAC (nested MAC)

cascade



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) \Rightarrow cascade(k, m||v) For any w

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

ECBC-MAC and NMAC analysis

Theorem: For any $L > 0$,

For every eff. q -query PRF adv. A attacking F_{ECBC} or F_{NMAC}
there exists an eff. adversary B s.t.:

$$\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq \text{Adv}_{\text{PRP}}[B, F] + 2q^2 / |X|$$

$$\text{Adv}_{\text{PRF}}[A, F_{\text{NMAC}}] \leq q \cdot L \cdot \text{Adv}_{\text{PRF}}[B, F] + q^2 / 2|K|$$

CBC-MAC is secure as long as $q \ll |X|^{1/2}$

NMAC is secure as long as $q \ll |K|^{1/2}$ (2^{64} for AES-128)

An example

$$\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq \text{Adv}_{\text{PRP}}[B, F] + 2q^2 / |X|$$

$q = \# \text{ messages MAC-ed with } k$

Suppose we want $\text{Adv}_{\text{PRF}}[A, F_{\text{ECBC}}] \leq 1/2^{32} \iff q^2 / |X| < 1/2^{32}$

- AES: $|X| = 2^{128} \Rightarrow q < 2^{48}$

So, after 2^{48} messages must, must change key

- 3DES: $|X| = 2^{64} \Rightarrow q < 2^{16}$

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:

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

The security bounds are tight: an attack

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

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

Generic attack on the derived MAC:

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

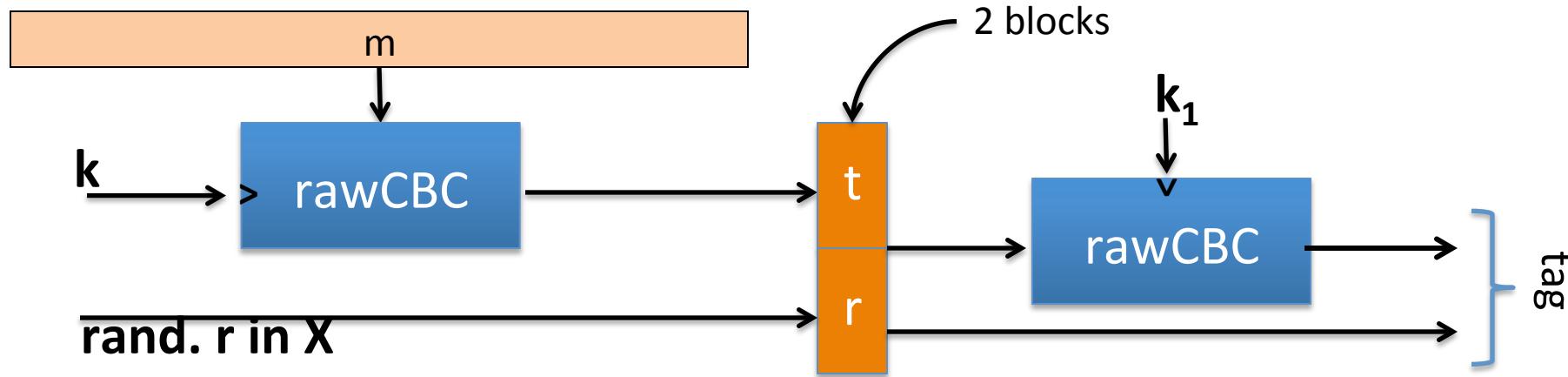
obtain (m_i, t_i) for $i = 1, \dots, |Y|^{1/2}$

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 $t := F_{\text{BIG}}(k, m_u||w)$

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

Better security: a rand. construction



Let $F: K \times X \rightarrow X$ be a PRF. Result: MAC with tags in X^2 .

Security: $\text{Adv}_{\text{MAC}}[A, I_{\text{RCBC}}] \leq \text{Adv}_{\text{PRP}}[B, F] \cdot (1 + 2 q^2 / |X|)$

⇒ For 3DES: can sign $q=2^{32}$ msgs with one key

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)

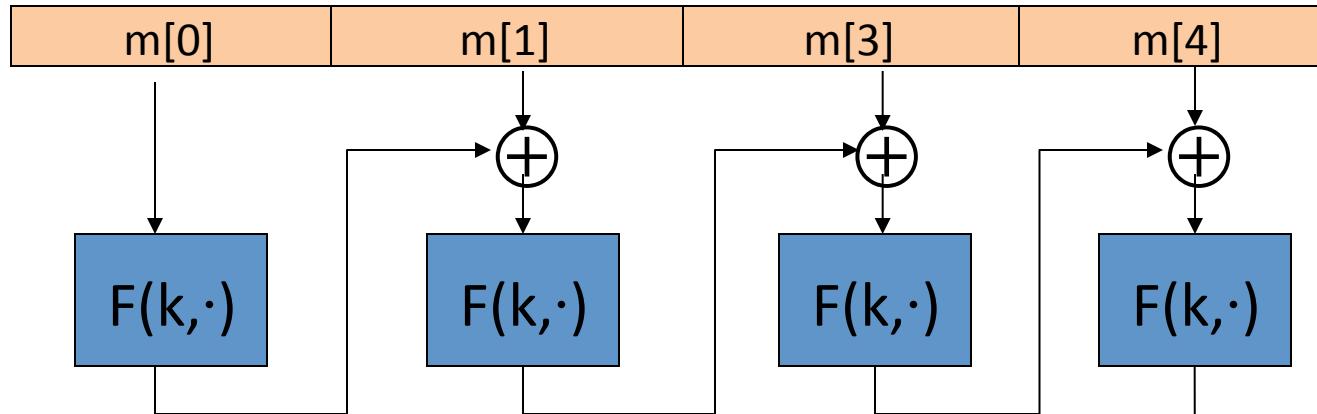
End of Segment



Message Integrity

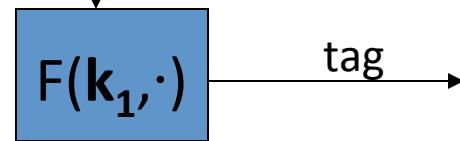
MAC padding

Recall: ECBC-MAC

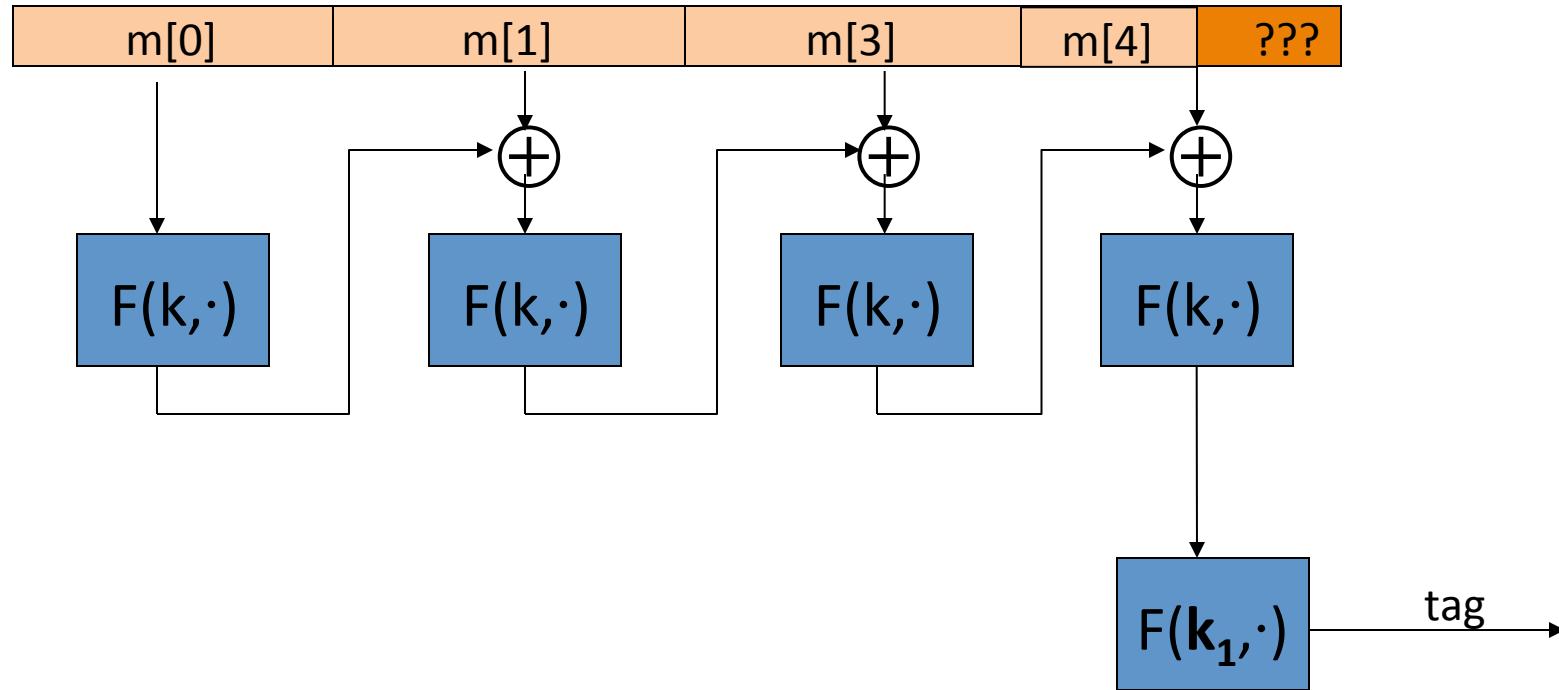


Let $F: K \times X \rightarrow X$ be a PRP

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



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: $\text{pad}(m) = \text{pad}(m||0)$

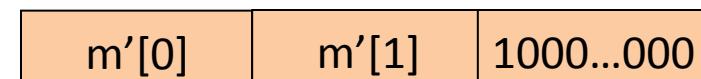
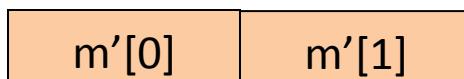
CBC MAC padding

For security, padding must be invertible !

$$m_0 \neq m_1 \Rightarrow \text{pad}(m_0) \neq \text{pad}(m_1)$$

ISO: pad with “1000...00”. Add new dummy block if needed.

- The “1” indicates beginning of pad.

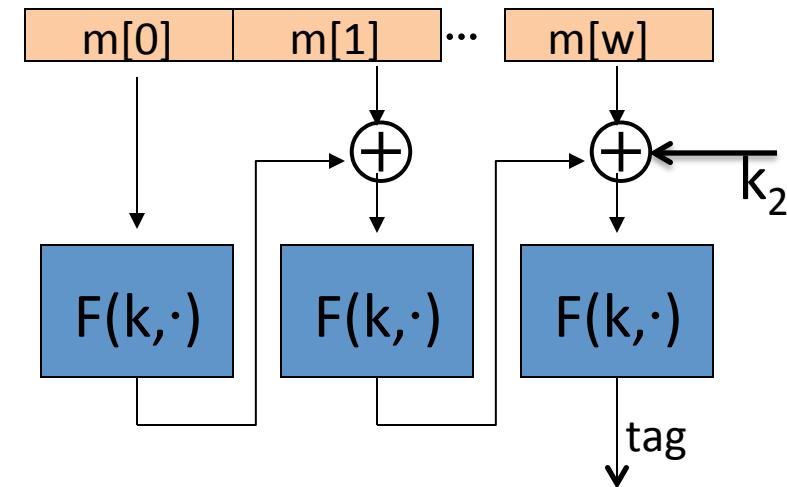
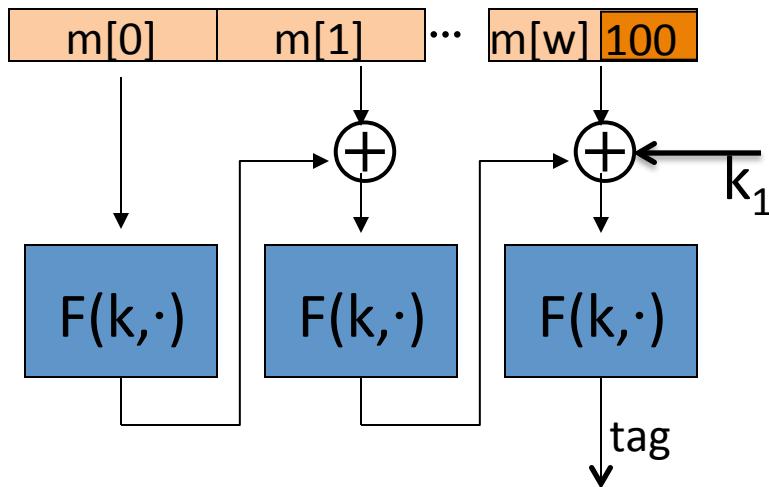


CMAC (NIST standard)

(k_1, k_2) derived
from K

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

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



End of Segment



Message Integrity

PMAC and
Carter-Wegman MAC

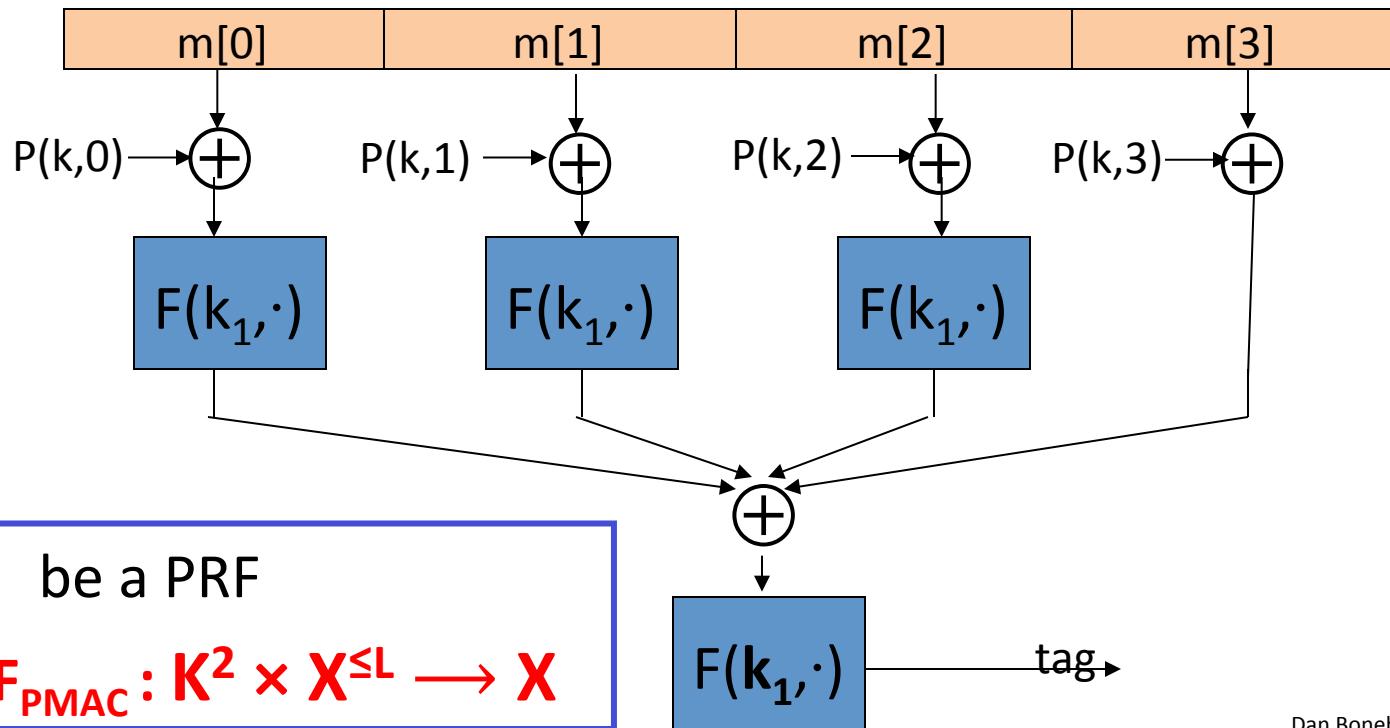
- ECBC and NMAC are sequential.
- Can we build a parallel MAC from a small PRF ??

Construction 3: PMAC – parallel MAC

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

key = (k, k_1)

Padding similar
to CMAC



PMAC: Analysis

PMAC Theorem: For any $L > 0$,

If F is a secure PRF over (K, X, X) then

F_{PMAC} is a secure PRF over $(K, X^{\leq L}, X)$.

For every eff. q -query PRF adv. A attacking F_{PMAC} there exists an eff. PRF adversary B s.t.:

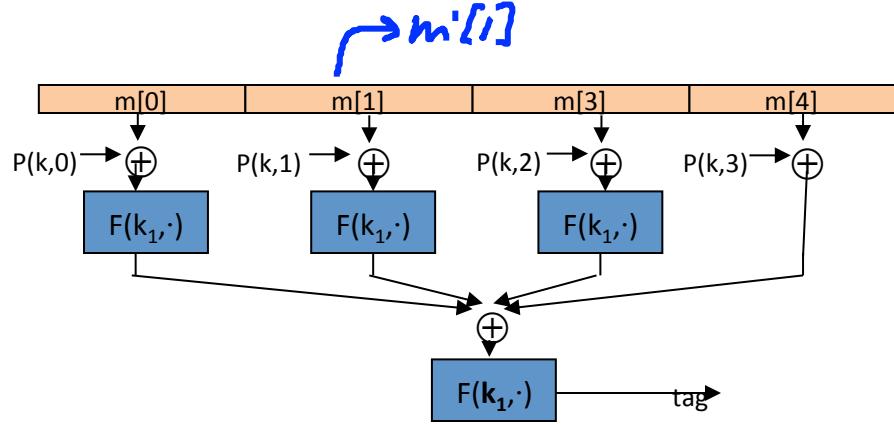
$$\text{Adv}_{\text{PRF}}[A, F_{\text{PMAC}}] \leq \text{Adv}_{\text{PRF}}[B, F] + 2 q^2 L^2 / |X|$$

PMAC is secure as long as $qL \ll |X|^{1/2}$

PMAC is incremental

Suppose F is a PRP.

When $m[1] \rightarrow m'[1]$
can we quickly update tag?

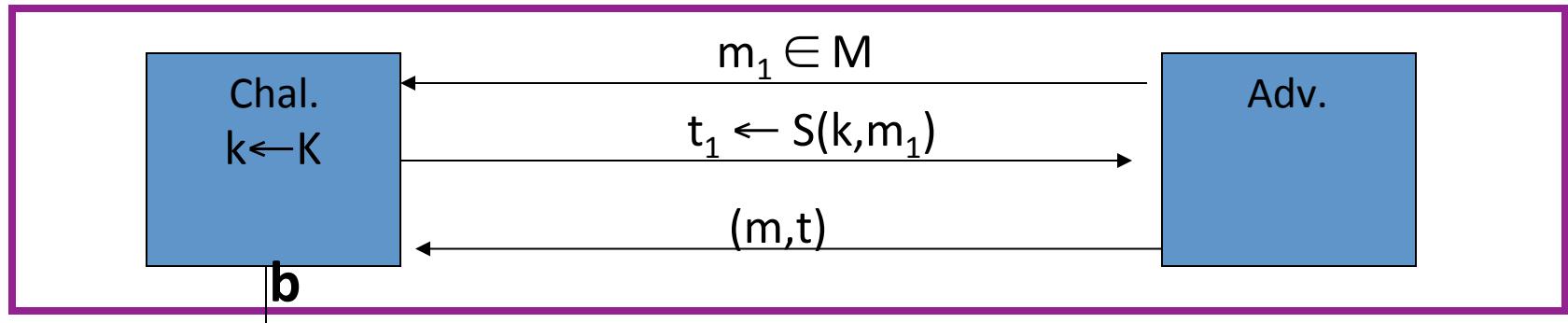


- no, it can't be done
- do $F^{-1}(k_1, \text{tag}) \oplus F(k_1, m'[1] \oplus P(k,1))$
- do $F^{-1}(k_1, \text{tag}) \oplus F(k_1, m[1] \oplus P(k,1)) \oplus F(k_1, m'[1] \oplus P(k,1))$
- do $\text{tag} \oplus F(k_1, m[1] \oplus P(k,1)) \oplus F(k_1, m'[1] \oplus P(k,1))$

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:



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

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

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. $q = 2^{128}+51$)

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

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

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

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

We show: given $S(\text{key}, \text{msg}_1)$ adv. has no info about $S(\text{key}, \text{msg}_2)$

One-time security (unconditional)

Thm: the one-time MAC on the previous slide satisfies (L=msg-len)

$$\forall m_1 \neq m_2, t_1, t_2: \Pr_{a,b} [s(a,b), m_1) = t_1 \mid s(a,b), m_2) = t_2] \leq L/q$$

Proof: $\forall m_1 \neq m_2, t_1, t_2:$

$$(1) \Pr_{a,b} [s(a,b), m_2) = t_2] = \Pr_{a,b} [P_{m_2}(a) + b = t_2] = 1/q$$

$$(2) \Pr_{a,b} [s(a,b), m_1) = t_1 \text{ and } s(a,b), m_2) = t_2] =$$

$$\Pr_{a,b} [P_{m_1}(a) - P_{m_2}(a) = t_1 - t_2 \text{ and } P_{m_2}(a) + b = t_2] \leq L/q^2 \blacksquare$$

\Rightarrow given valid (m_2, t_2) , adv. outputs (m_1, t_1) and is right with prob. $\leq L/q$

One-time MAC \Rightarrow Many-time MAC

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

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

Carter-Wegman MAC: $CW((k_1, k_2), m) = (r, F(k_1, r) \oplus S(k_2, m))$



for random $r \leftarrow \{0,1\}^n$.

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

$$CW((k_1, k_2), m) = (r, F(k_1, r) \oplus S(k_2, m))$$

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

Construction 4: HMAC (Hash-MAC)

Most widely used MAC on the Internet.

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

Further reading

- J. Black, P. Rogaway: CBC MACs for Arbitrary-Length Messages: The Three-Key Constructions. *J. Cryptology* 18(2): 111-131 (2005)
- K. Pietrzak: A Tight Bound for EMAC. *ICALP* (2) 2006: 168-179
- J. Black, P. Rogaway: A Block-Cipher Mode of Operation for Parallelizable Message Authentication. *EUROCRYPT* 2002: 384-397
- M. Bellare: New Proofs for NMAC and HMAC: Security Without Collision-Resistance. *CRYPTO* 2006: 602-619
- Y. Dodis, K. Pietrzak, P. Puniya: A New Mode of Operation for Block Ciphers and Length-Preserving MACs. *EUROCRYPT* 2008: 198-219

End of Segment



Collision resistance

Introduction

Recap: message integrity

So far, four MAC constructions:

- PRFs :
 - ECBC-MAC, CMAC** : commonly used with AES (e.g. 802.11i)
 - NMAC** : basis of HMAC (this segment)
 - PMAC**: a parallel MAC
- randomized MAC :
 - Carter-Wegman MAC**: built from a fast one-time MAC

This module: MACs from collision resistance.

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:

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

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

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

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

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

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

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

Example: $S(k, m) = \text{AES}_{\text{2-block-cbc}}(k, \text{SHA-256}(m))$ 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. $H(m_0) = H(m_1)$.

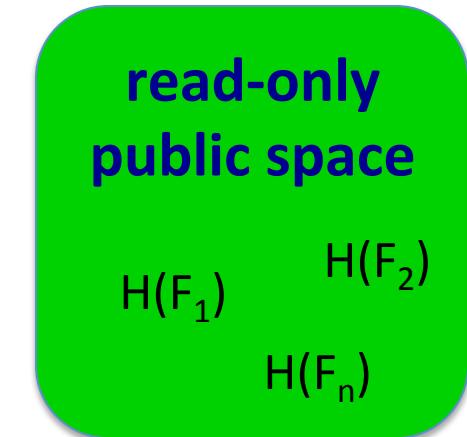
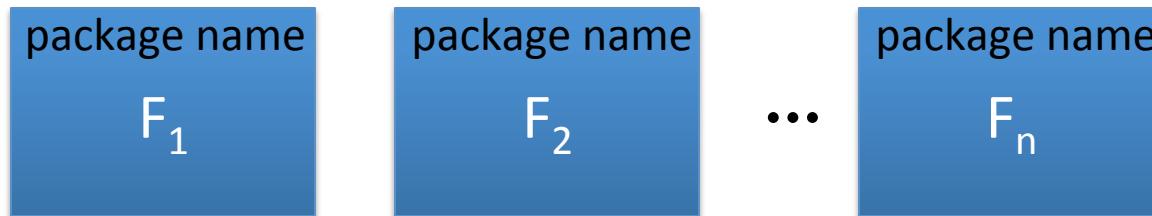
Then: S^{big} is insecure under a 1-chosen msg attack

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

Generic birthday attack

Generic attack on C.R. functions

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

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

Algorithm:

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

How well will this work?

The birthday paradox

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

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

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

$$\Pr\left[\exists i \neq j : r_i = r_j\right] = 1 - \Pr\left[\forall i \neq j : r_i \neq r_j\right] = 1 - \left(\frac{B-1}{B}\right)\left(\frac{B-2}{B}\right) \cdots \left(\frac{B-n+1}{B}\right) =$$

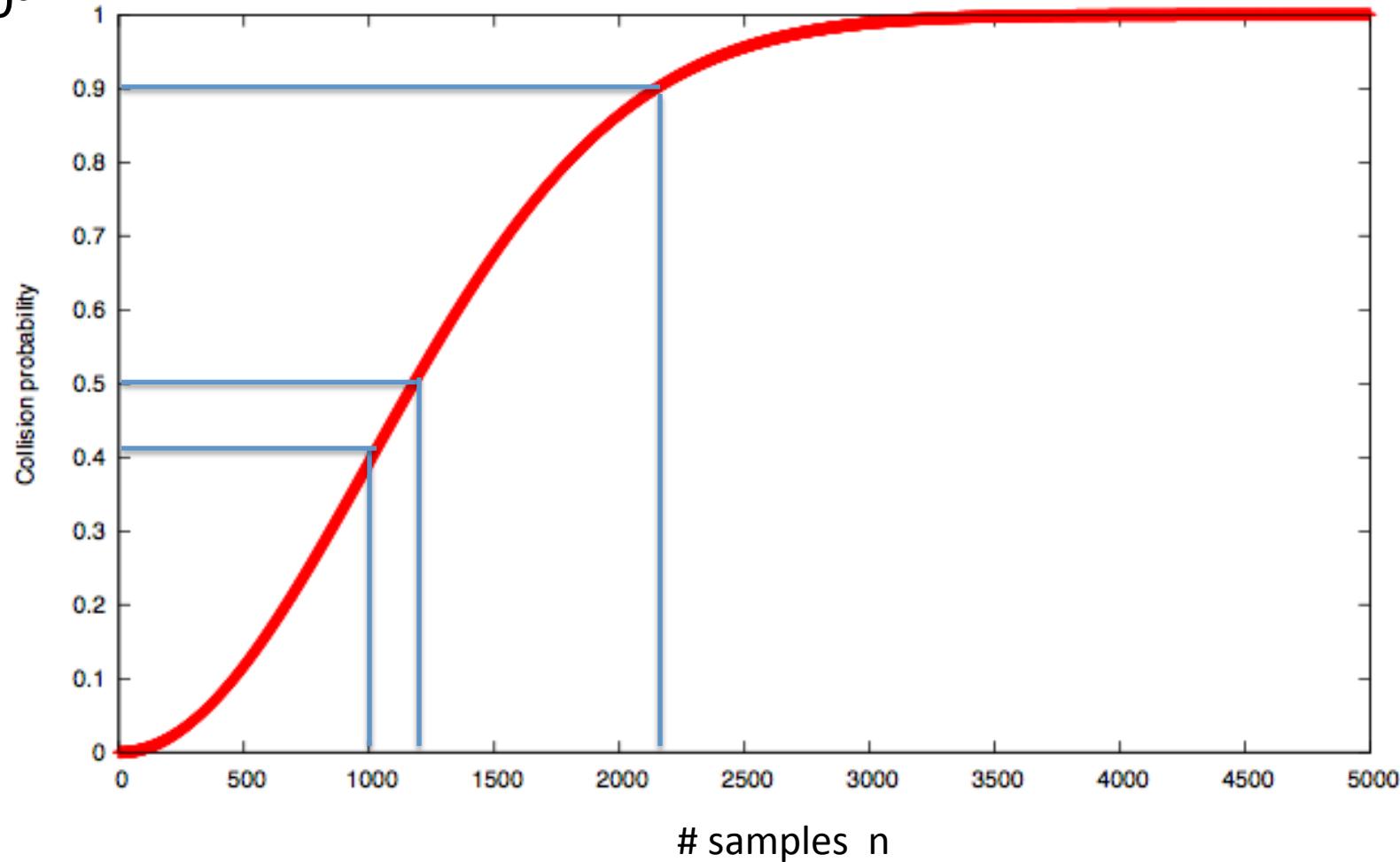
$$= 1 - \prod_{i=1}^{n-1} \left(1 - \frac{i}{B}\right) \geq 1 - \prod_{i=1}^{n-1} e^{-i/B} = 1 - e^{-\frac{1}{B} \sum_{i=1}^{n-1} i} \geq 1 - e^{-n^2/2B}$$

$1-x \leq e^{-x}$

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

■

$B=10^6$



Generic attack

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

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

Expected number of iteration ≈ 2

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

Sample C.R. hash functions:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

	<u>digest size (bits)</u>	<u>Speed (MB/sec)</u>	<u>generic attack time</u>
<u>function</u>			
NIST standards	SHA-1	160	2^{80}
	SHA-256	256	2^{128}
	SHA-512	512	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 $E: K \times X \rightarrow X$ exhaustive search	$O(K)$	$O(K ^{1/2})$
Hash function $H: M \rightarrow T$ collision finder	$O(T ^{1/2})$	$O(T ^{1/3})$

End of Segment



Collision resistance

The Merkle-Damgard Paradigm

Collision resistance: review

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

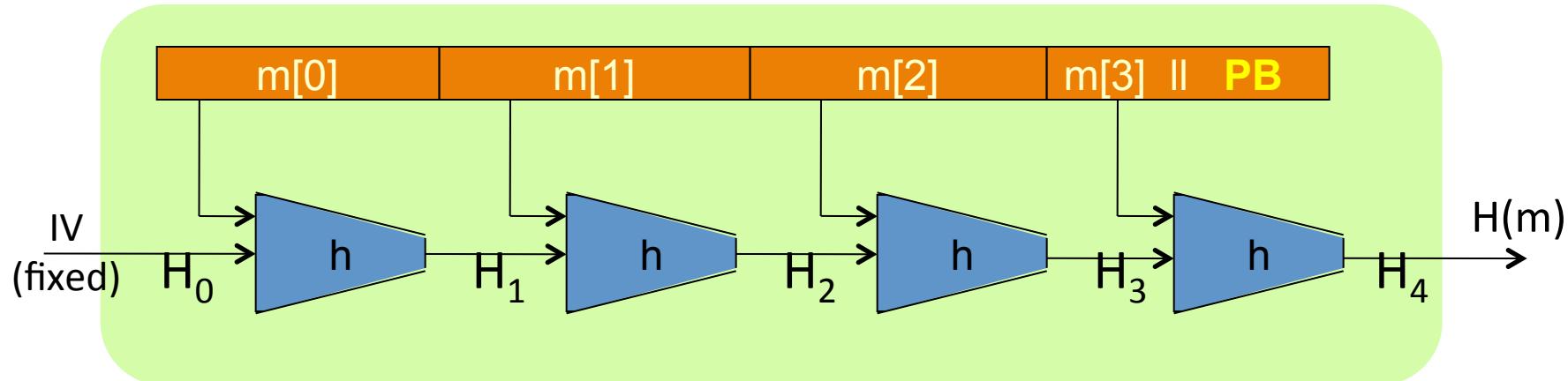
A collision for H is a pair $m_0, m_1 \in M$ such that:

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

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

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

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

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

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

IF $\begin{cases} H_t \neq H'_{r'} \text{ or} \\ M_t \neq M'_{r'} \text{ or} \\ PB \neq PB' \end{cases}$

\Rightarrow we have a collision on h .

STOP

Otherwise,

Suppose $\underline{H_t = H'_r}$ and $\underline{M_t = M'_r}$ and $PB = PB'$

$$\begin{array}{c} \curvearrowleft \\ t=r \end{array}$$

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

If $\begin{cases} H_{t-1} \neq H'_{t-1} \\ \text{or} \\ M_{t-1} \neq M'_{t-1} \end{cases}$ then we have a collision on h . STOP.

Otherwise, $H_{t-1} = H'_{t-1}$ and $M_t = M'_r$ and $M_{t-1} = M'_{t-1}$.

Iterate all the way to beginning and either:

(1) find collision on h , or

(2) $\forall i: M_i = M'_i \Rightarrow M = M'$ 

cannot happen
because M, M'
are collision
on H .



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

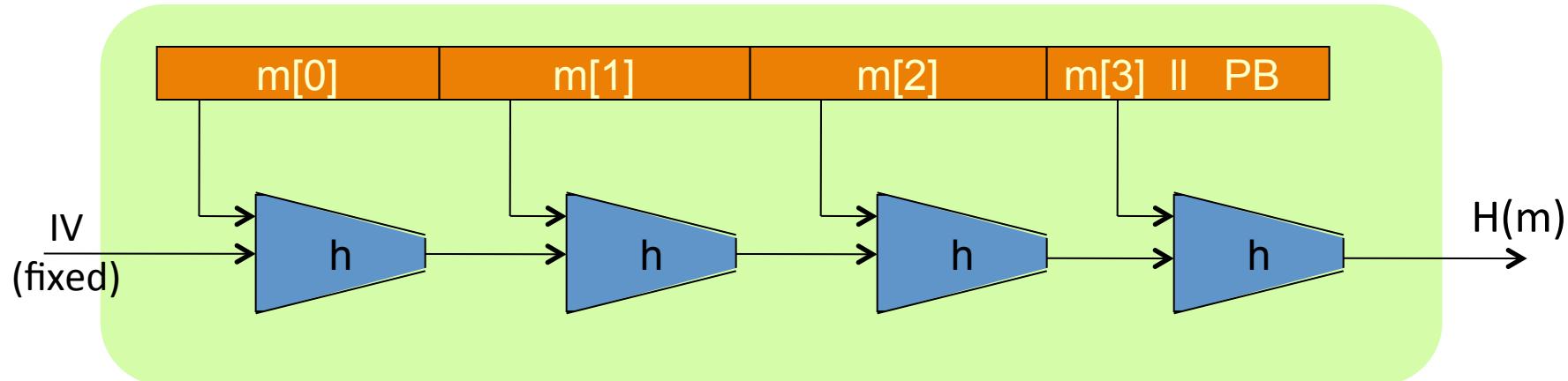
End of Segment



Collision resistance

Constructing Compression Functions

The Merkle-Damgard iterated construction



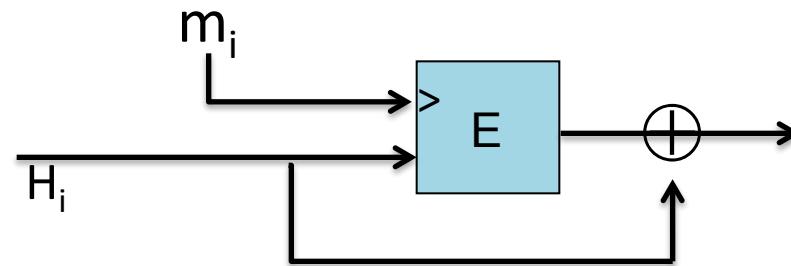
Thm: h collision resistant \Rightarrow H collision resistant

Goal: construct compression function $\mathbf{h: T \times X \rightarrow T}$

Compr. func. from a block cipher

$E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ a block cipher.

The **Davies-Meyer** compression function: $\mathbf{h(H, m) = E(m, H) \oplus H}$



Thm: Suppose E is an ideal cipher (collection of $|K|$ random perms.).

Finding a collision $\mathbf{h(H,m)=h(H',m')}$ takes $\mathbf{O(2^{n/2})}$ evaluations of (E,D) .

Best possible !!

Suppose we define $\mathbf{h(H, m) = E(m, H)}$

Then the resulting $h(.,.)$ is not collision resistant:

to build a collision (H,m) and (H',m')

choose random (H,m,m') and construct H' as follows:

- $H' = D(m', E(m, H)) \quad \leftarrow \quad E(m', H') = E(m, H)$
- $H' = E(m', D(m, H))$
- $H' = E(m', E(m, H))$
- $H' = D(m', D(m, H))$

Other block cipher constructions

Let $E: \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}^n$ for simplicity

Miyaguchi-Preneel: $h(H, m) = E(m, H) \oplus H \oplus m$ (Whirlpool)

$h(H, m) = E(H \oplus m, m) \oplus m$

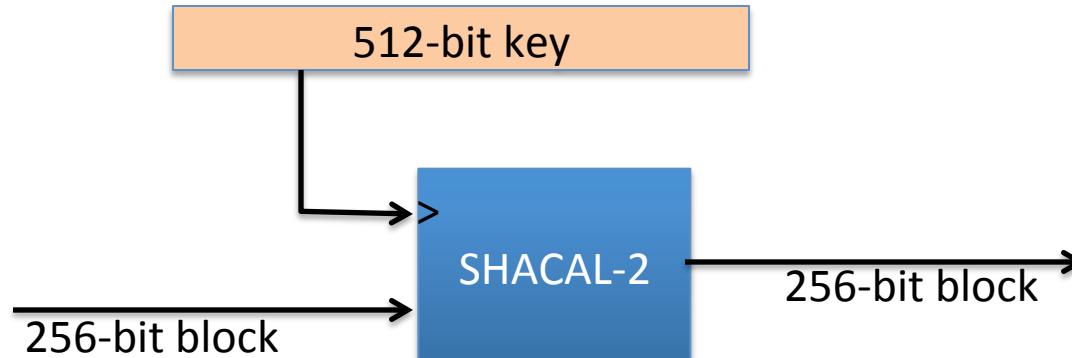
total of 12 variants like this

Other natural variants are insecure:

$h(H, m) = E(m, H) \oplus m$ (HW)

Case study: SHA-256

- Merkle-Damgård function
- Davies-Meyer compression function
- Block cipher: SHACAL-2



Provable compression functions

Choose a random 2000-bit prime p and random $1 \leq u, v \leq p$.

For $m, h \in \{0, \dots, p-1\}$ define $\boxed{h(H, m) = u^H \cdot v^m \pmod{p}}$

Fact: finding collision for $h(.,.)$ is as hard as solving “discrete-log” modulo p .

Problem: slow.

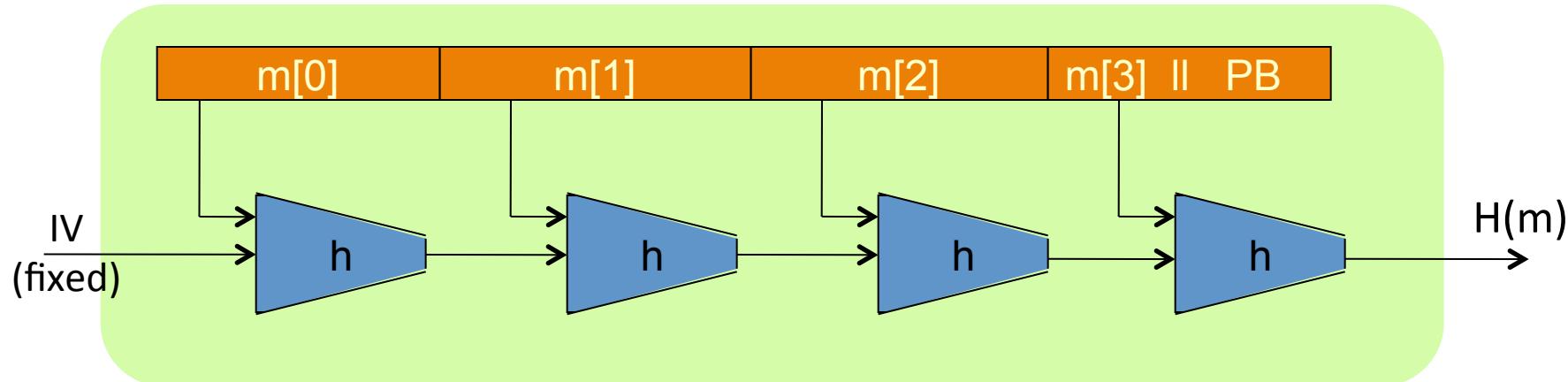
End of Segment



Collision resistance

HMAC:
a MAC from SHA-256

The Merkle-Damgard iterated construction



Thm: h collision resistant \Rightarrow H collision resistant

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

MAC from a Merkle-Damgard Hash Function

$H: X^{≤L} \rightarrow T$ 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 $H(w \parallel k \parallel m \parallel PB)$ 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 $H(k \parallel m \parallel PB \parallel w)$ 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.

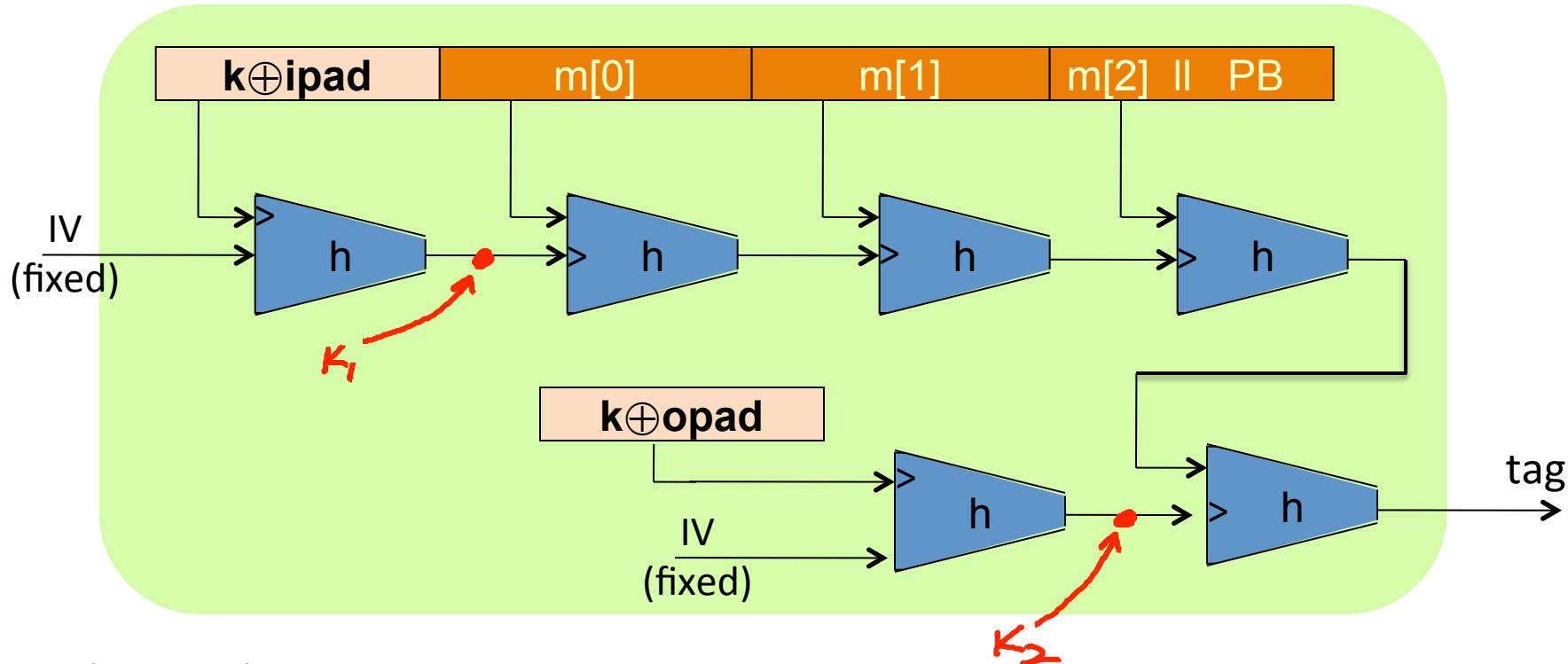
H: hash function.

example: SHA-256 ; output is 256 bits

Building a MAC out of a hash function:

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

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 ($q \ll |T|^{\frac{1}{2}}$)

In TLS: must support HMAC-SHA1-96

End of Segment



Collision resistance

Timing attacks on MAC verification

Warning: verification timing attacks [L'09]

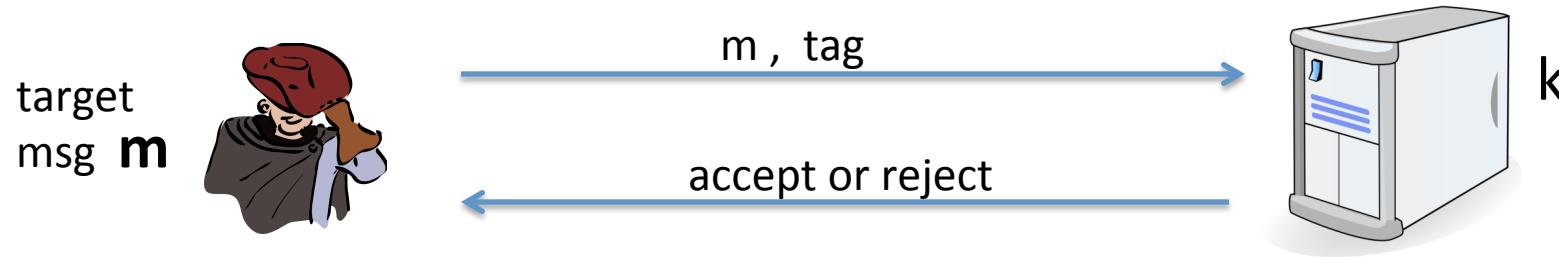
Example: Keyczar crypto library (Python) [simplified]

```
def Verify(key, msg, sig_bytes):  
    return HMAC(key, msg) == sig_bytes
```

The problem: ‘==’ implemented as a byte-by-byte comparison

- Comparator returns false when first inequality found

Warning: verification timing attacks [L'09]



Timing attack: to compute tag for target message m do:

Step 1: Query server with random tag

Step 2: Loop over all possible first bytes and query server.

stop when verification takes a little longer than in step 1

Step 3: repeat for all tag bytes until valid tag found



Defense #1

Make string comparator always take same time (Python) :

```
return false if sig_bytes has wrong length  
result = 0  
for x, y in zip( HMAC(key,msg) , sig_bytes):  
    result |= ord(x) ^ ord(y)  
return result == 0
```

Can be difficult to ensure due to optimizing compiler.

Defense #2

Make string comparator always take same time (Python) :

```
def Verify(key, msg, sig_bytes):  
    mac = HMAC(key, msg)  
    return HMAC(key, mac) == HMAC(key, sig_bytes)
```

Attacker doesn't know values being compared

Lesson

Don't implement crypto yourself !

End of Segment



Authenticated Encryption

Active attacks on
CPA-secure encryption

Recap: the story so far

Confidentiality: semantic security against a CPA attack

- Encryption secure against **eavesdropping only**

Integrity:

- Existential unforgeability under a chosen message attack
- CBC-MAC, HMAC, PMAC, CW-MAC

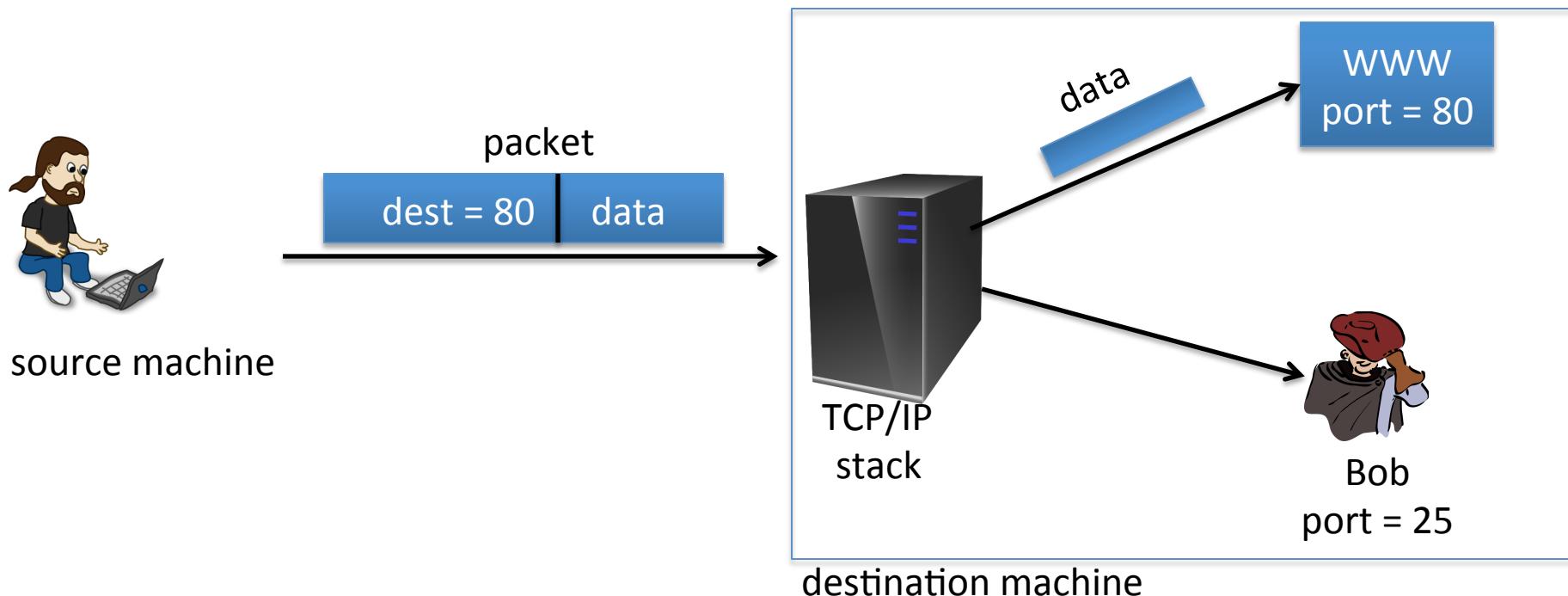
This module: encryption secure against **tampering**

(*active adversary*)

- Ensuring both confidentiality and integrity

Sample tampering attacks

TCP/IP: (highly abstracted)

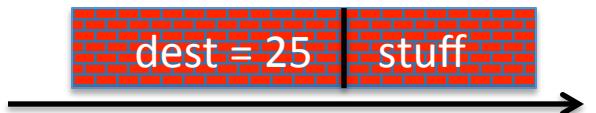
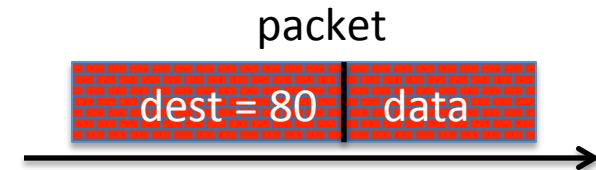


Sample tampering attacks

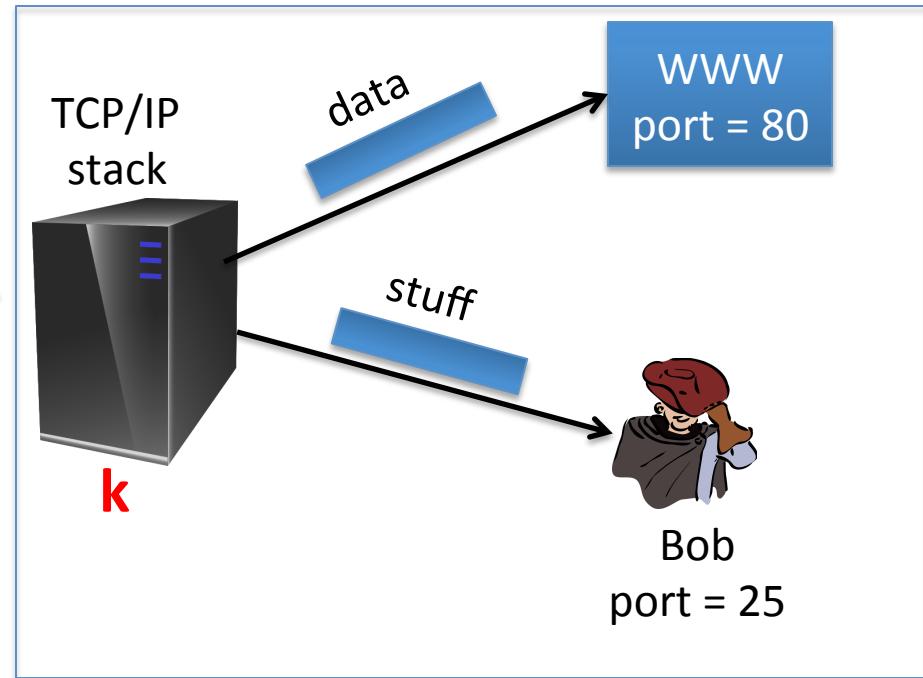
IPsec: (highly abstracted)



k

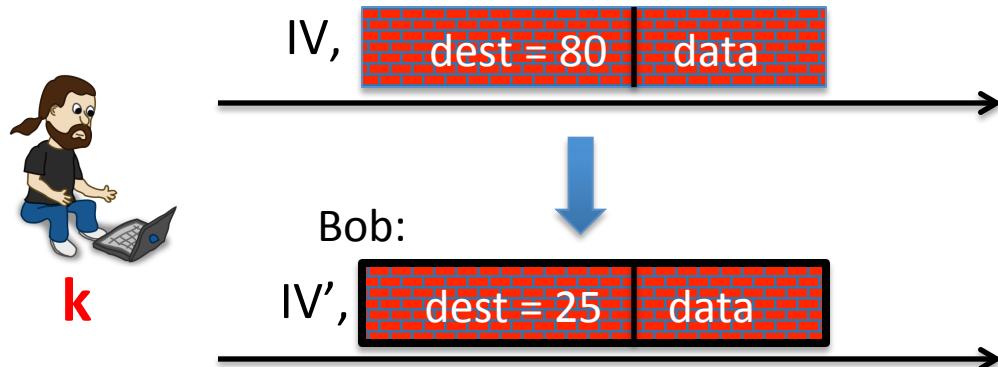


packets encrypted
using key k

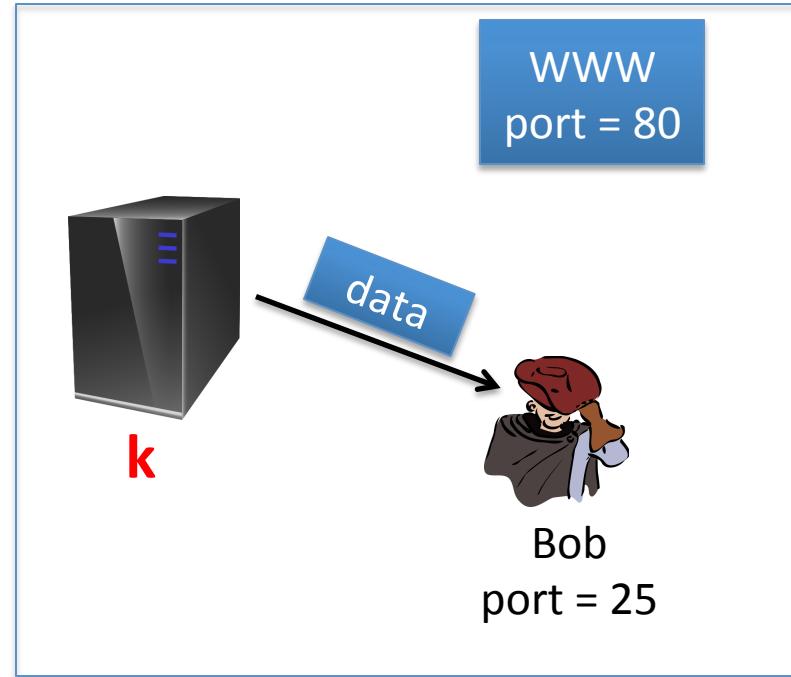


Reading someone else's data

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?

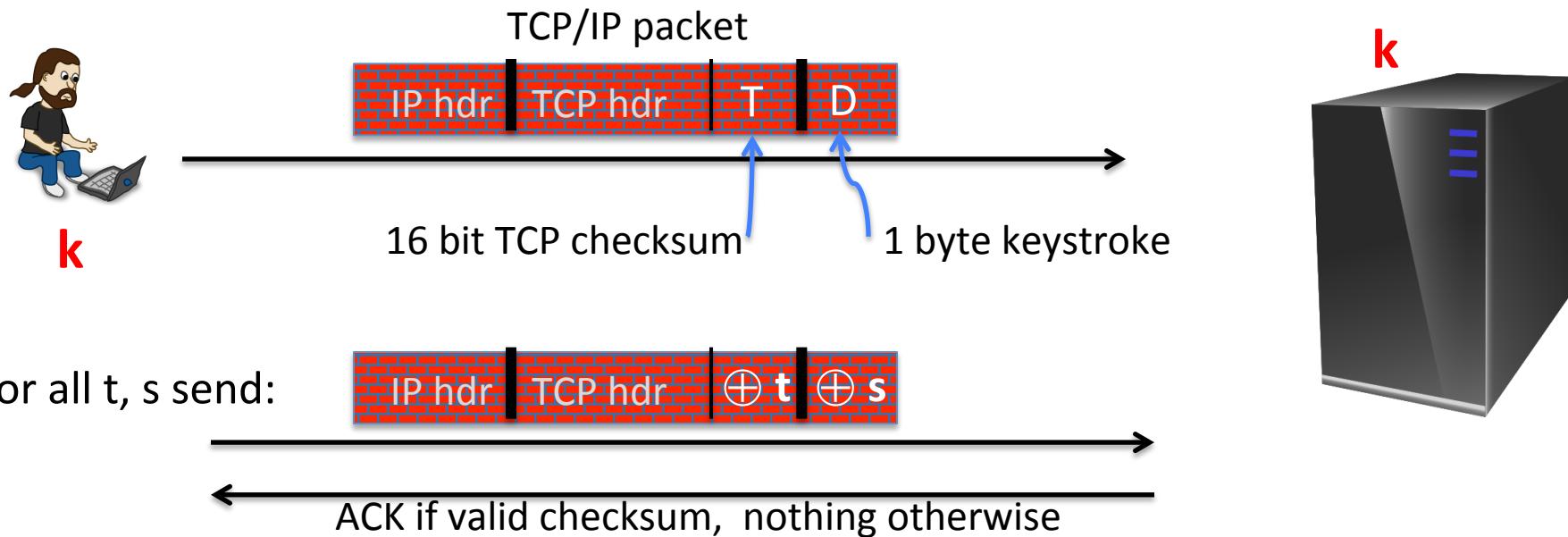
$$m[0] = D(k, c[0]) \oplus \text{IV} = \text{"dest=80..."}$$

- $\text{IV}' = \text{IV} \oplus (\dots 25\dots)$
- $\text{IV}' = \text{IV} \oplus (\dots 80\dots)$
- $\text{IV}' = \text{IV} \oplus (\dots 80\dots) \oplus (\dots 25\dots)$
- It can't be done

$$\begin{aligned}
 \underline{D(k, c[0]) \oplus \text{IV}'} &= \overbrace{D(k, c[0]) \oplus \text{IV}}^{\text{"dest=80..."}} \oplus \overbrace{25}^{\text{"dest=25..."}}
 \\ &= \dots 25\dots
 \end{aligned}$$

An attack using only network access

Remote terminal app.: each keystroke encrypted with CTR mode



$$\{ \text{checksum}(\text{hdr}, D) = t \oplus \text{checksum}(\text{hdr}, D \oplus s) \} \Rightarrow \text{can find } D$$

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

Security: the system must provide

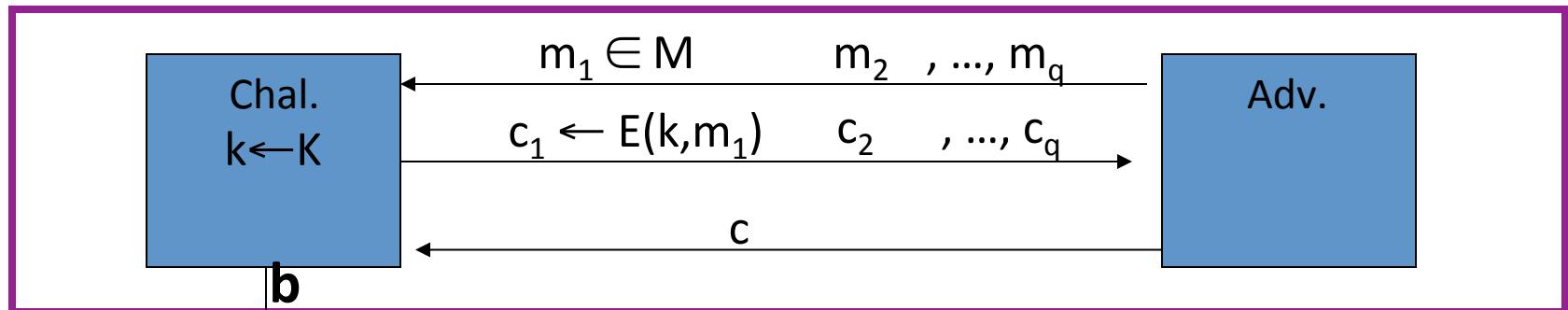
- sem. security under a CPA attack, and
- **ciphertext integrity**:
attacker cannot create new ciphertexts that decrypt properly



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:

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

Authenticated encryption

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

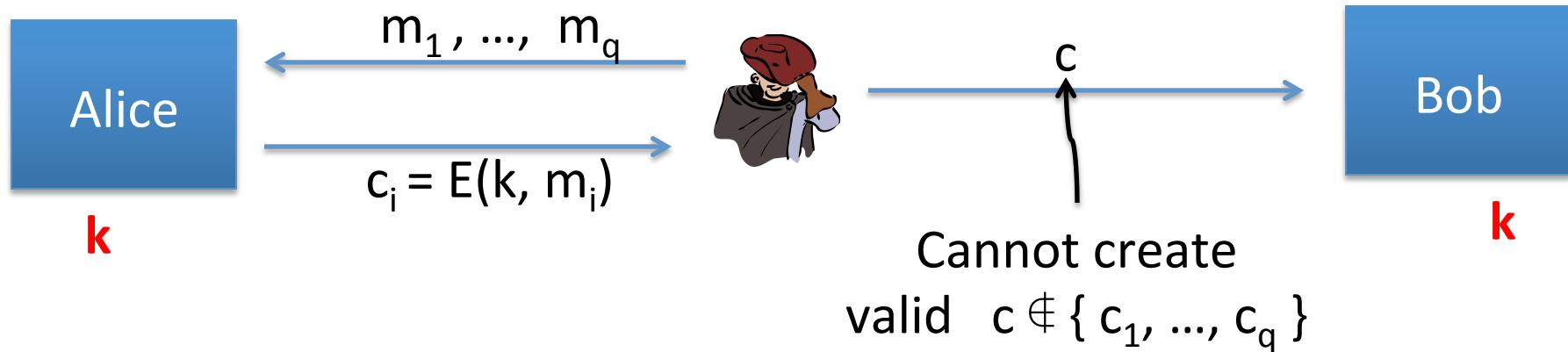
- (1) semantically secure under CPA, and
- (2) has ciphertext integrity

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



⇒ 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



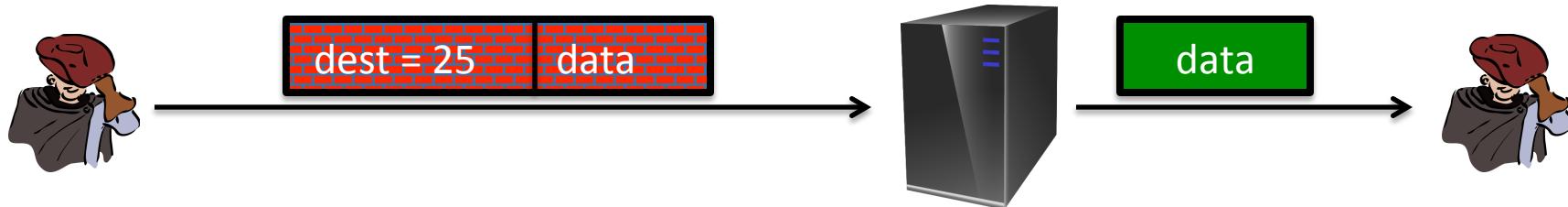
Authenticated Encryption

Chosen ciphertext
attacks

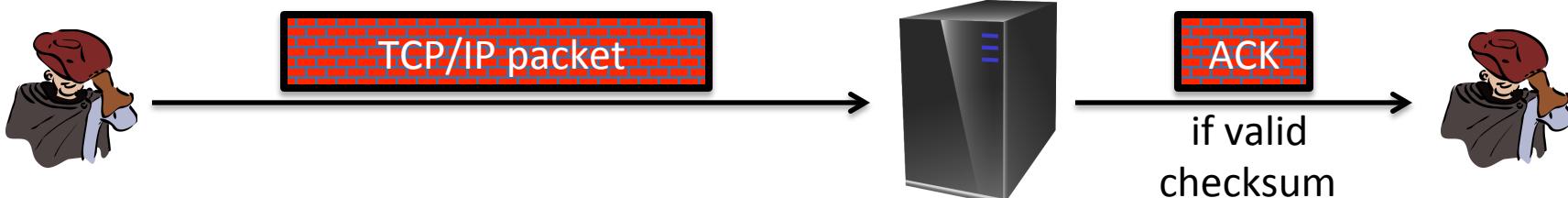
Example chosen ciphertext attacks

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

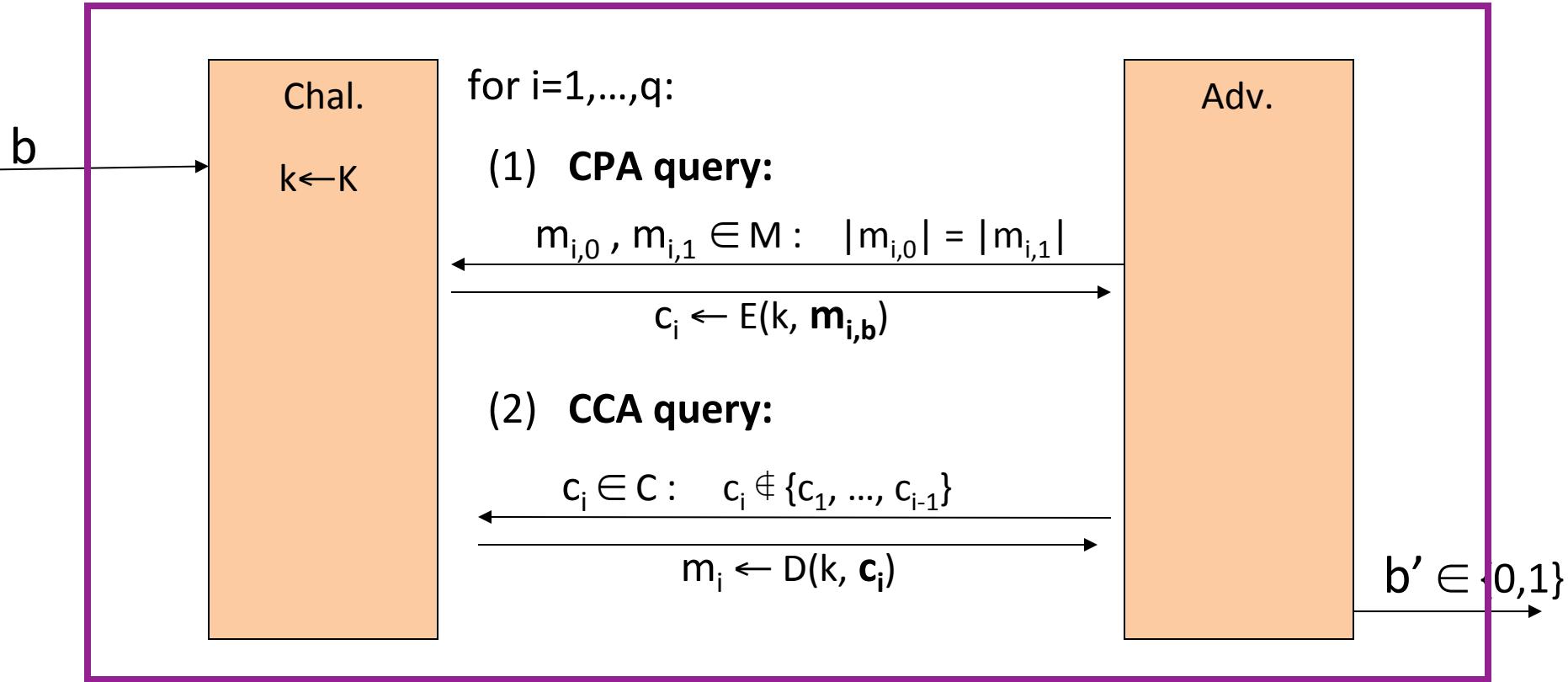
Adversary's power: both CPA and CCA

- 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

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

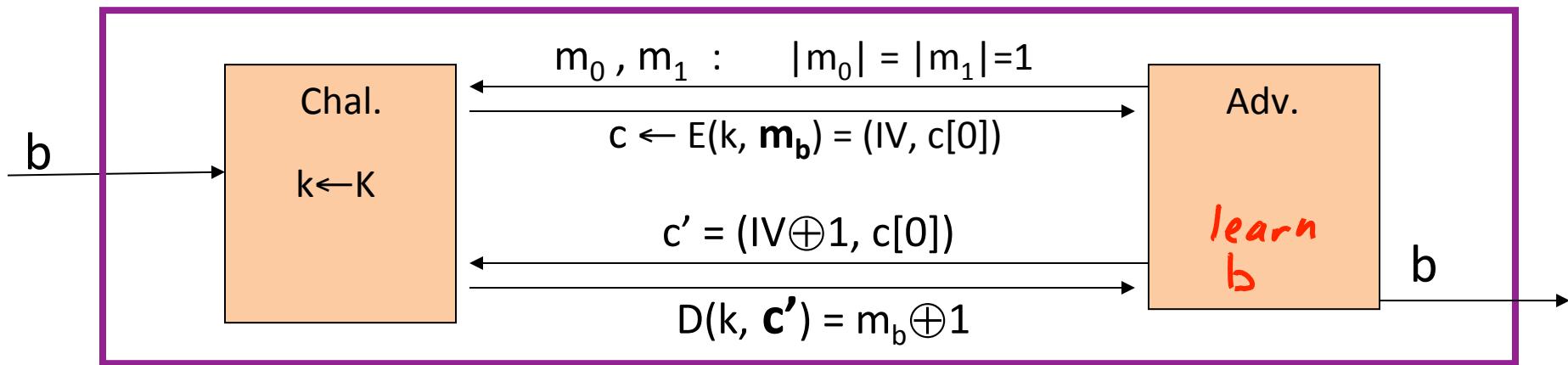


Chosen ciphertext security: definition

E is CCA secure if for all “efficient” A :

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

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



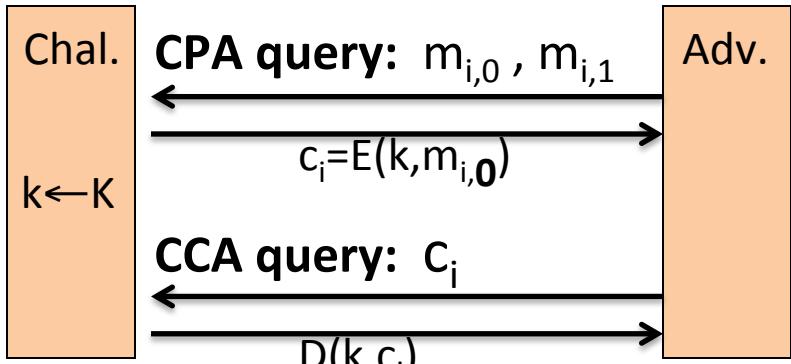
Authenticated enc. \Rightarrow CCA security

Thm: Let (E, D) be a cipher that provides AE.
Then (E, D) is CCA secure !

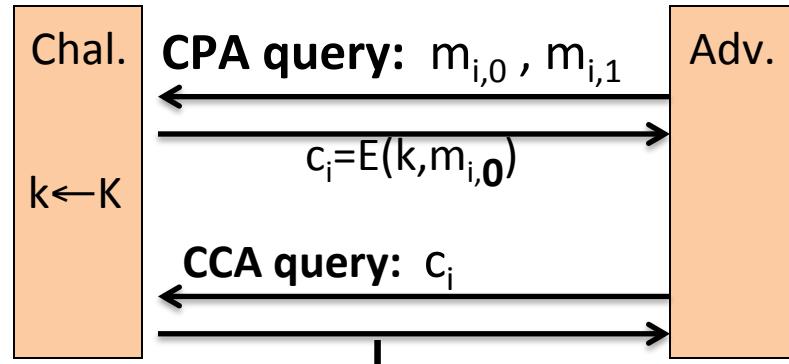
In particular, for any q -query eff. A there exist eff. B_1, B_2 s.t.

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

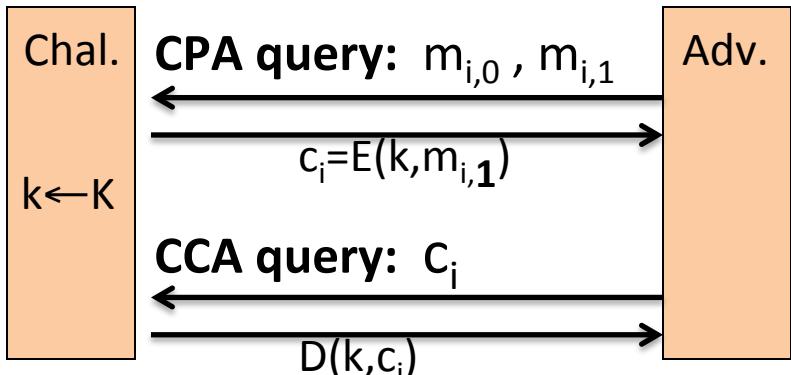
Proof by pictures



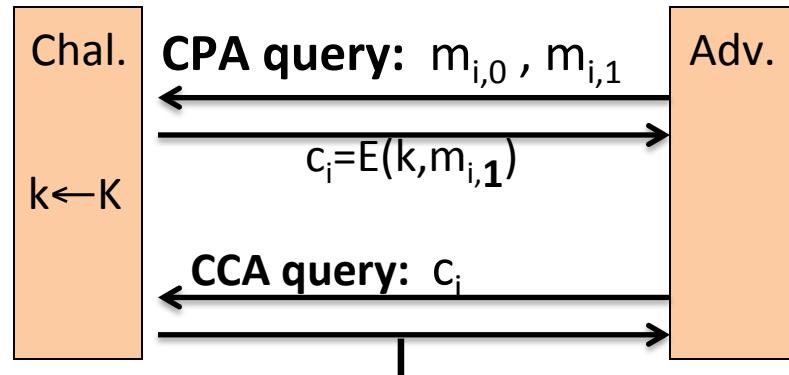
\approx p



p



≈ p



20

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

... but first, some history

Authenticated Encryption (AE): introduced in 2000 [KY'00, BN'00]

Crypto APIs before then: (e.g. MS-CAPI)  *crypto API*

- Provide API for CPA-secure encryption (e.g. CBC with rand. IV)
- Provide API for MAC (e.g. HMAC)

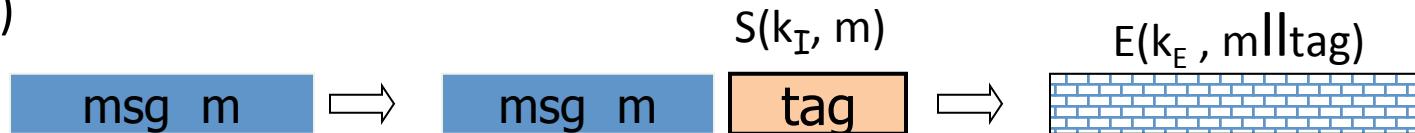
Every project had to combine the two itself without a well defined goal

- Not all combinations provide AE ...

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

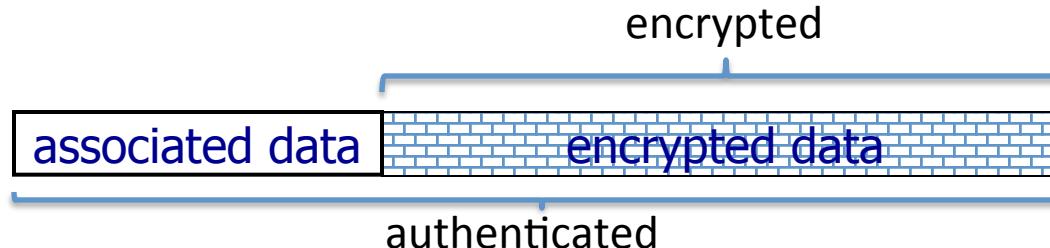
Let (E,D) be CPA secure cipher and (S,V) secure MAC. Then:

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 )
```

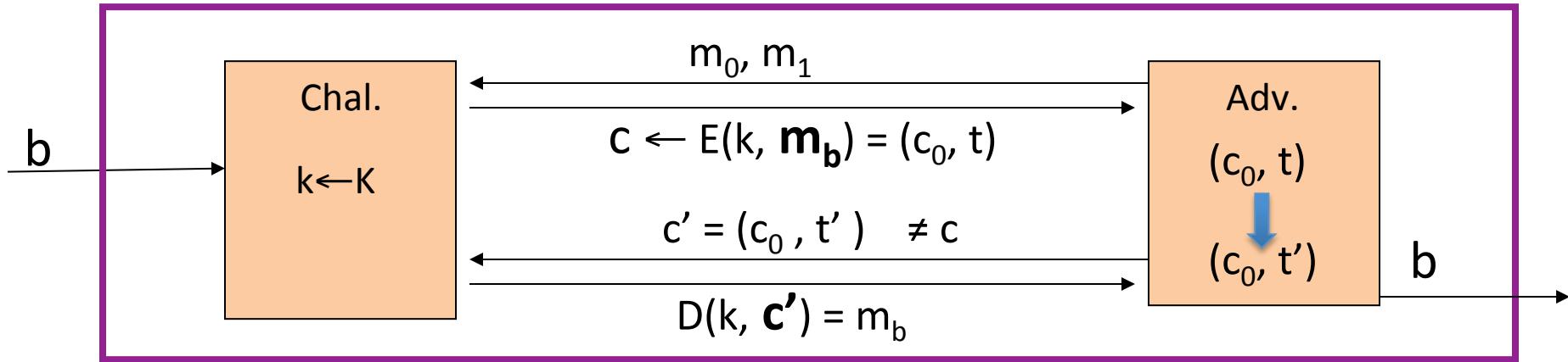
```
int AES_GCM_EncryptUpdate(AES_GCM_CTX *a,  
                           unsigned char *aad, unsigned long aadlen,  
                           unsigned char *data, unsigned long datalen,  
                           unsigned char *out, unsigned long *outlen)
```

MAC Security -- an explanation

Recall: MAC security implies $(m, t) \not\Rightarrow (m, t')$

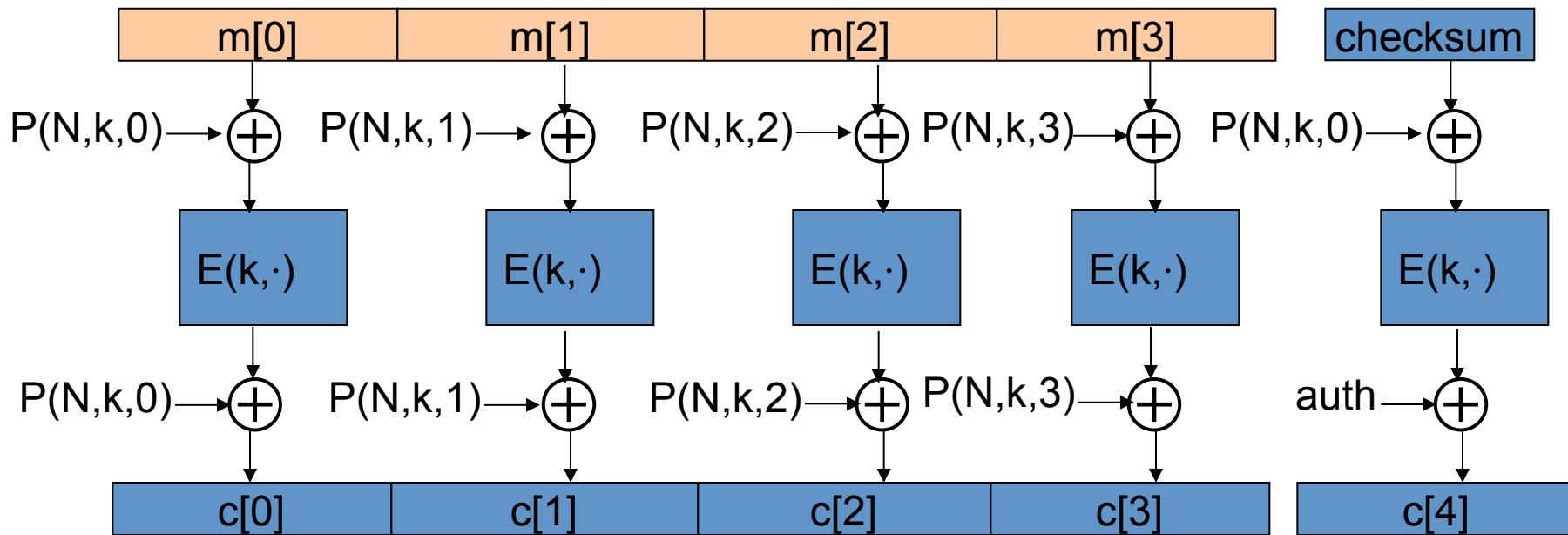
Why? Suppose not: $(m, t) \rightarrow (m, t')$

Then Encrypt-then-MAC would not have Ciphertext Integrity !!



OCB: a direct construction from a PRP

More efficient authenticated encryption: one $E()$ op. per block.



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



Authenticated Encryption

Case study: TLS

The TLS Record Protocol (TLS 1.2)



Unidirectional keys: $k_{b \rightarrow s}$ and $k_{s \rightarrow b}$

Stateful encryption:

- Each side maintains two 64-bit counters: $ctr_{b \rightarrow s}$, $ctr_{s \rightarrow b}$
- Init. to 0 when session started. $ctr++$ for every record.
- Purpose: replay defense

TLS record: encryption (CBC AES-128, HMAC-SHA1)

$$k_{b \rightarrow s} = (k_{mac}, k_{enc})$$



Browser side $\text{enc}(k_{b \rightarrow s}, \text{data}, \text{ctr}_{b \rightarrow s})$:

not transmitted in packet

step 1: $\text{tag} \leftarrow S(k_{mac}, [++\text{ctr}_{b \rightarrow s} \parallel \text{header} \parallel \text{data}])$

step 2: pad [header || data || tag] to AES block size

step 3: CBC encrypt with k_{enc} and new random IV

step 4: prepend header

TLS record: decryption (CBC AES-128, HMAC-SHA1)

Server side $\text{dec}(k_{b \rightarrow s}, \text{record}, \text{ctr}_{b \rightarrow s})$:

step 1: CBC decrypt record using k_{enc}

step 2: check pad format: send **bad_record_mac** if invalid

step 3: check tag on $[++\text{ctr}_{b \rightarrow s} \parallel \text{header} \parallel \text{data}]$
send **bad_record_mac** if invalid

Provides authenticated encryption
(provided no other info. is leaked during decryption)

Bugs in older versions (prior to TLS 1.1)

IV for CBC is predictable: (chained IV)

IV for next record is last ciphertext block of current record.

Not CPA secure. (a practical exploit: BEAST attack)

Padding oracle: during decryption

if pad is invalid send **decryption failed** alert

if mac is invalid send **bad_record_mac** alert

⇒ attacker learns info. about plaintext (attack in next segment)

Lesson: when decryption fails, do not explain why

Leaking the length

The TLS header leaks the length of TLS records

- Lengths can also be inferred by observing network traffic

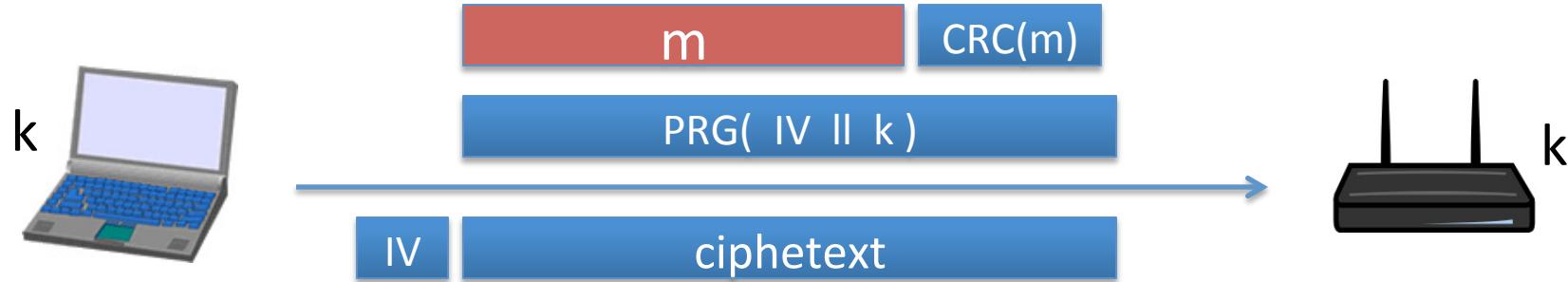
For many web applications, leaking lengths reveals sensitive info:

- In tax preparation sites, lengths indicate the type of return being filed which leaks information about the user's income
- In healthcare sites, lengths leak what page the user is viewing
- In Google maps, lengths leak the location being requested

No easy solution

802.11b WEP: how not to do it

802.11b WEP:



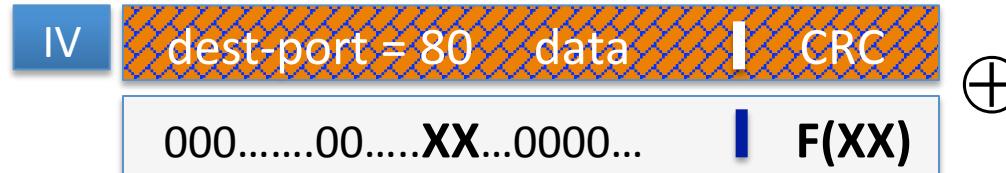
Previously discussed problems:

two time pad and related PRG seeds

Active attacks

Fact: CRC is linear, i.e. $\forall m,p: \text{CRC}(m \oplus p) = \text{CRC}(m) \oplus F(p)$

WEP ciphertext:



$$XX = 25 \oplus 80$$



Upon decryption: CRC is valid, but ciphertext is changed !!

End of Segment



Authenticated Encryption

CBC paddings attacks

Recap

Authenticated encryption: CPA security + ciphertext integrity

- Confidentiality in presence of **active** adversary
- Prevents chosen-ciphertext attacks

Limitation: cannot help bad implementations ... (this segment)

Authenticated encryption modes:

- Standards: GCM, CCM, EAX
- General construction: encrypt-then-MAC

The TLS record protocol (CBC encryption)

Decryption: $\text{dec}(k_{b \rightarrow s}, \text{record}, \text{ctr}_{b \rightarrow s})$:

step 1: CBC decrypt record using k_{enc}

step 2: check pad format: abort if invalid

step 3: check tag on $[++\text{ctr}_{b \rightarrow s} \parallel \text{header} \parallel \text{data}]$
abort if invalid

Two types of error:

- **padding error**
- **MAC error**



Padding oracle

Suppose attacker can differentiate the two errors
(pad error, MAC error):

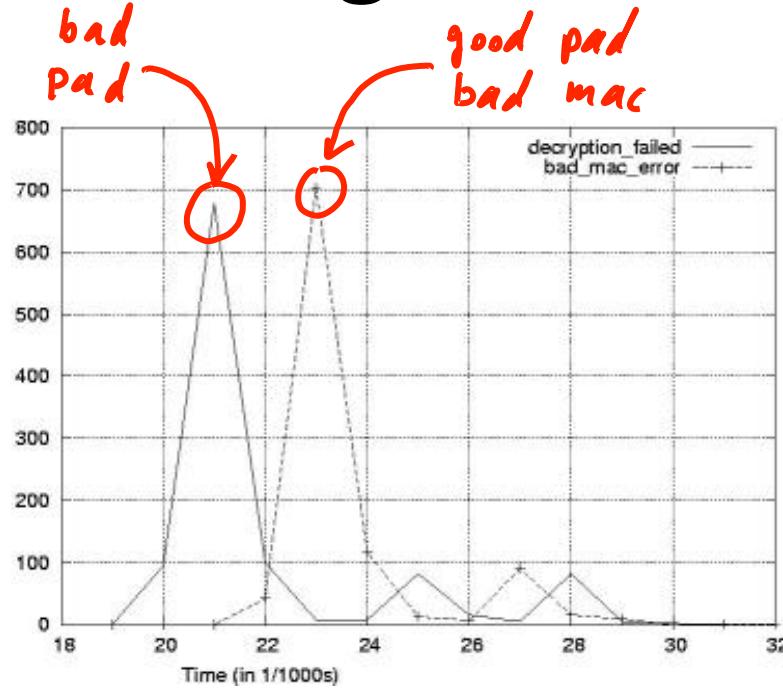
⇒ **Padding oracle**:

attacker submits ciphertext and learns if
last bytes of plaintext are a valid pad

Nice example of a
chosen ciphertext attack



Padding oracle via timing OpenSSL



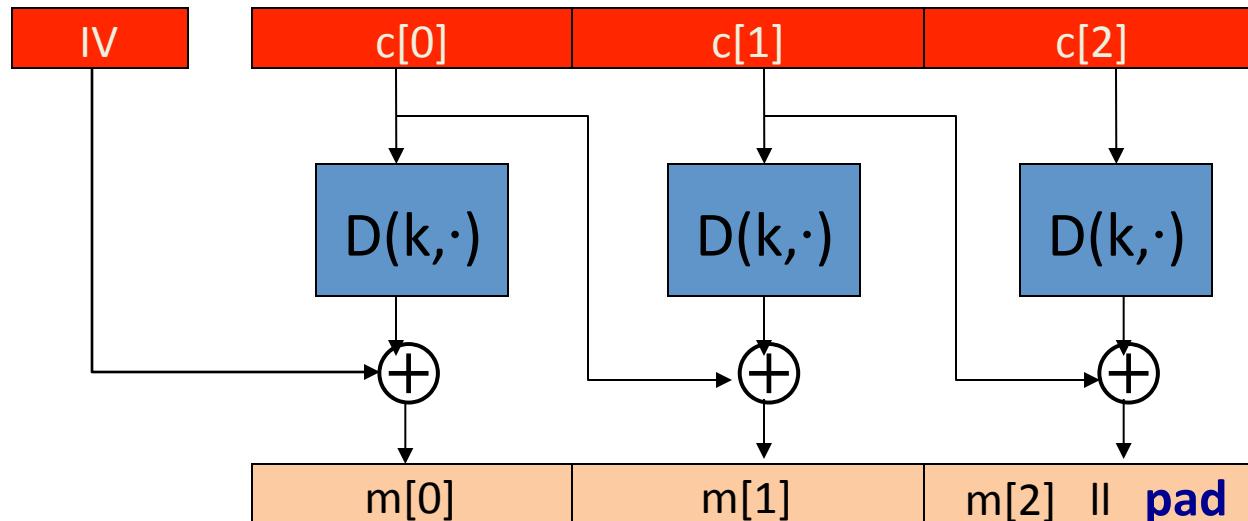
Credit: Brice Canvel

(fixed in OpenSSL 0.9.7a)

In older TLS 1.0: padding oracle due to different alert messages.

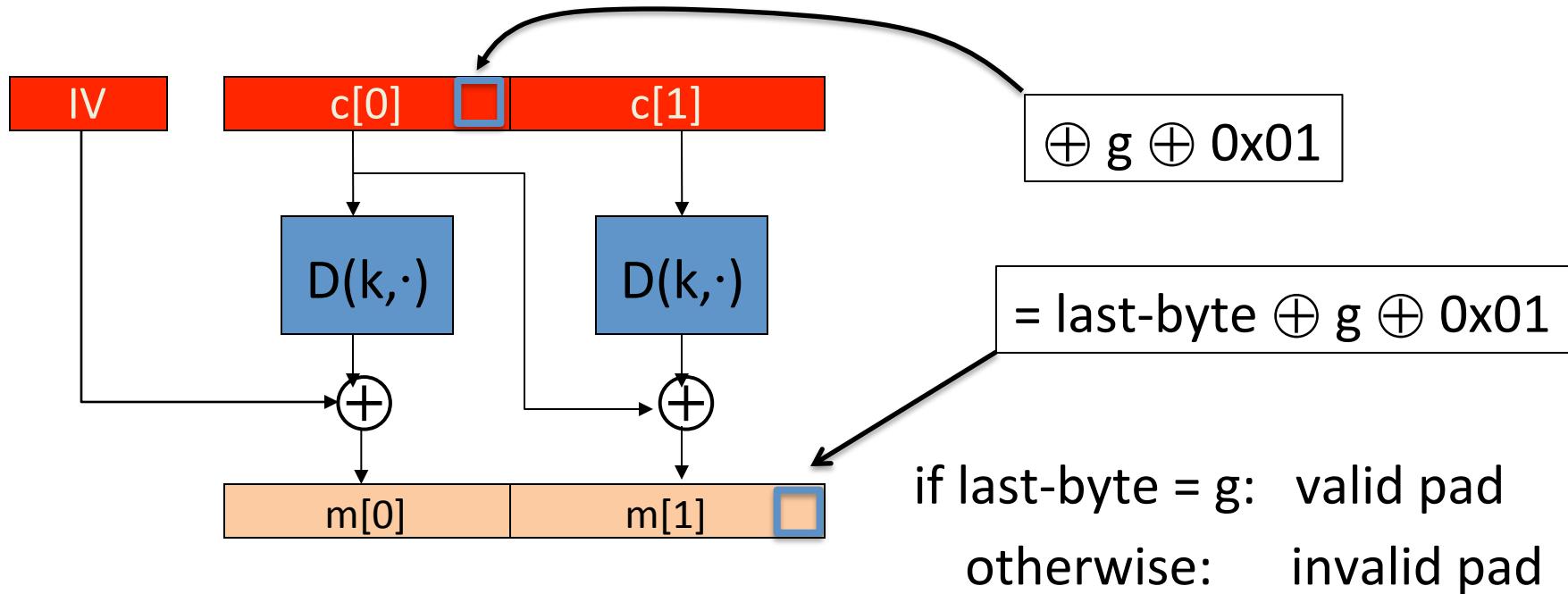
Using a padding oracle (CBC encryption)

Attacker has ciphertext $c = (c[0], c[1], c[2])$ and it wants $m[1]$



Using a padding oracle (CBC encryption)

step 1: let \mathbf{g} be a guess for the last byte of $m[1]$



Using a padding oracle (CBC encryption)

Attack: submit $(\text{IV}, \mathbf{c}'[0], \mathbf{c}[1])$ to padding oracle

⇒ attacker learns if last-byte = g

Repeat with $g = 0, 1, \dots, 255$ to learn last byte of $m[1]$

Then use a (02, 02) pad to learn the next byte and so on ...

IMAP over TLS

Problem: TLS renegotiates key when an invalid record is received

Enter IMAP over TLS: (protocol for reading email)

- Every five minutes client sends login message to server:
LOGIN "username" "password"
- Exact same attack works, despite new keys
 ⇒ recovers password in a few hours.

Lesson

1. Encrypt-then-MAC would completely avoid this problem:
MAC is checked first and ciphertext discarded if invalid
2. MAC-then-CBC provides A.E., but padding oracle destroys it

Will this attack work if TLS used counter mode instead of CBC?
(i.e. use MAC-then-CTR)

- Yes, padding oracles affect all encryption schemes
- It depends on what block cipher is used
- No, counter mode need not use padding 
-

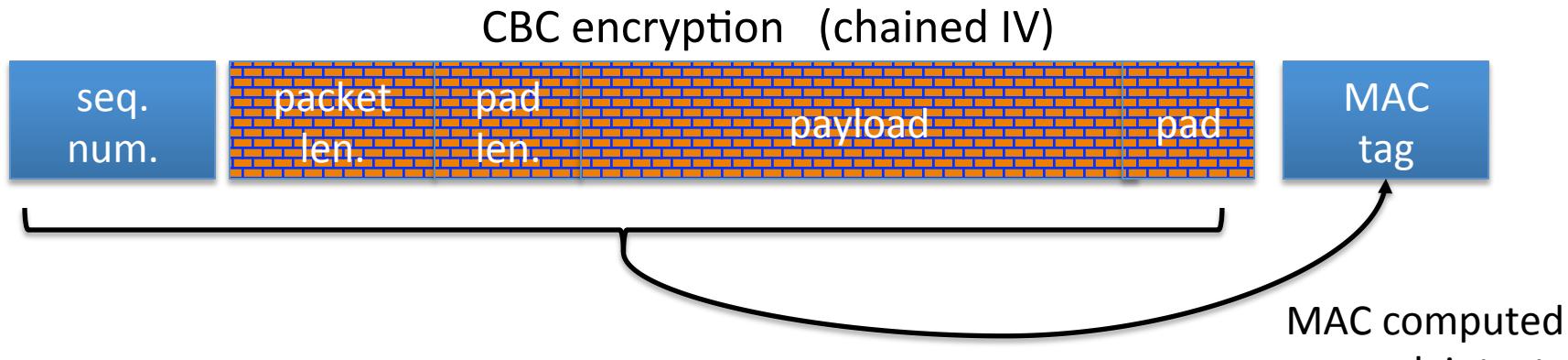
End of Segment



Authenticated Encryption

Attacking non-atomic
decryption

SSH Binary Packet Protocol

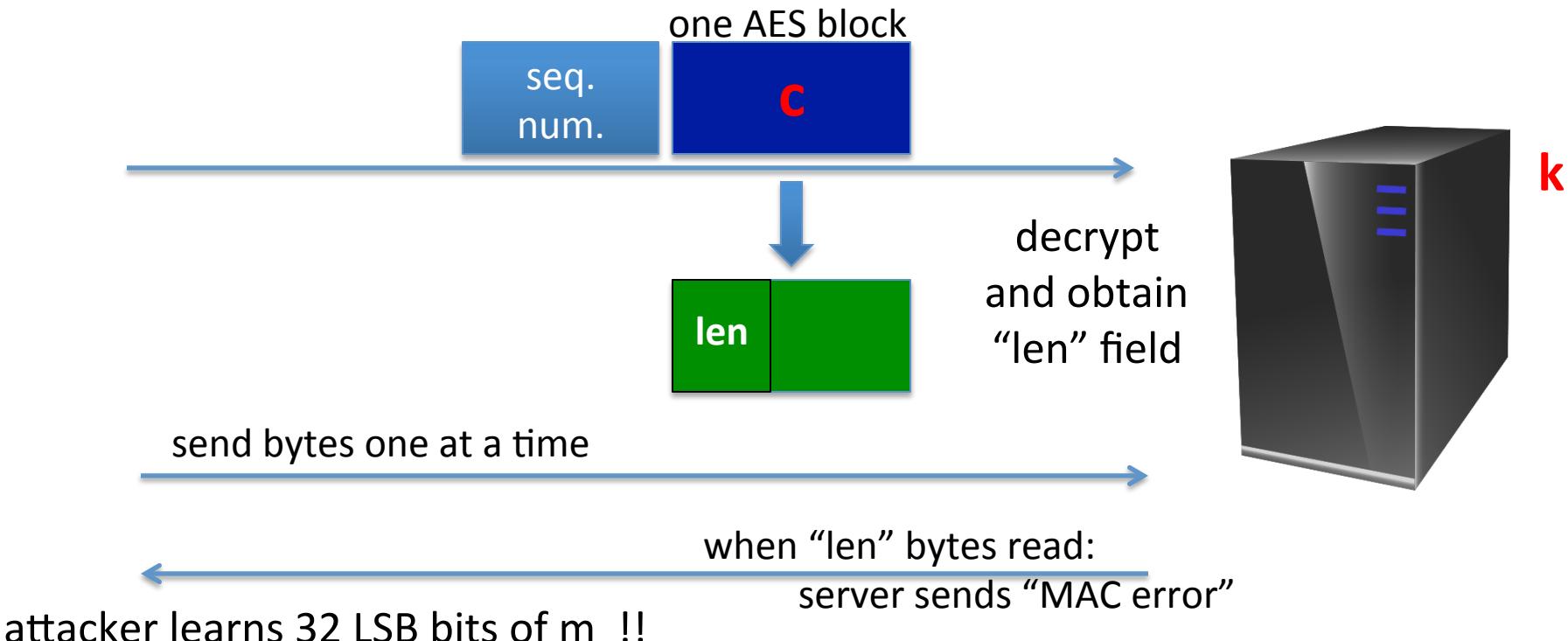


Decryption:

- step 1: decrypt packet length field only (!)
- step 2: read as many packets as length specifies
- step 3: decrypt remaining ciphertext blocks
- step 4: check MAC tag and send error response if invalid

An attack on the enc. length field (simplified)

Attacker has one ciphertext block $c = \text{AES}(k, m)$ and it wants m



Lesson

The problem: (1) non-atomic decrypt
(2) len field decrypted and used before it is authenticated

How would you redesign SSH to resist this attack?

- ➡ Send the length field unencrypted (but MAC-ed)
- Replace encrypt-and-MAC by encrypt-then-MAC
- ➡ Add a MAC of (seq-num, length) right after the len field
- Remove the length field and identify packet boundary by verifying the MAC after every received byte

Further reading

- The Order of Encryption and Authentication for Protecting Communications, H. Krawczyk, Crypto 2001.
- Authenticated-Encryption with Associated-Data, P. Rogaway, Proc. of CCS 2002.
- Password Interception in a SSL/TLS Channel, B. Canvel, A. Hiltgen, S. Vaudenay, M. Vuagnoux, Crypto 2003.
- Plaintext Recovery Attacks Against SSH, M. Albrecht, K. Paterson and G. Watson, IEEE S&P 2009
- Problem areas for the IP security protocols, S. Bellovin, Usenix Security 1996.

End of Segment



Odds and ends

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) :=

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

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

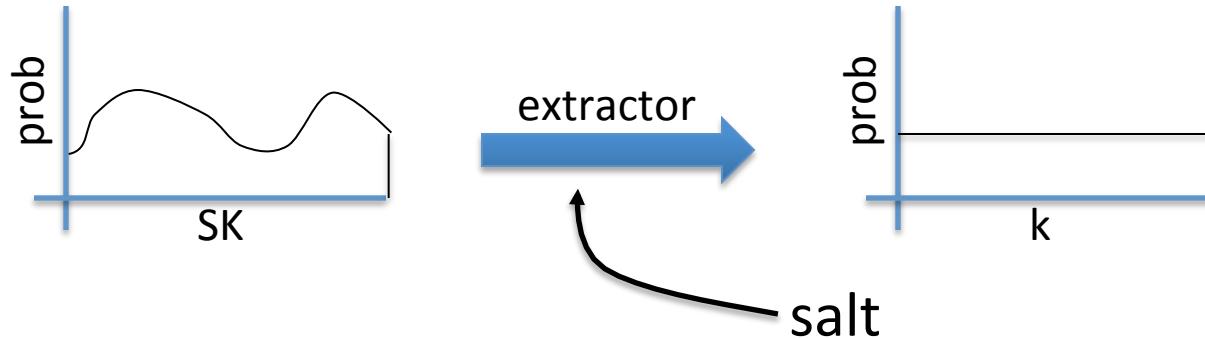
- SK not uniform \Rightarrow PRF output may not look random

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 $\mathbf{k} \leftarrow \text{HMAC(salt, SK)}$
- Then expand using HMAC as a PRF with key \mathbf{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
(more on this later)

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

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

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

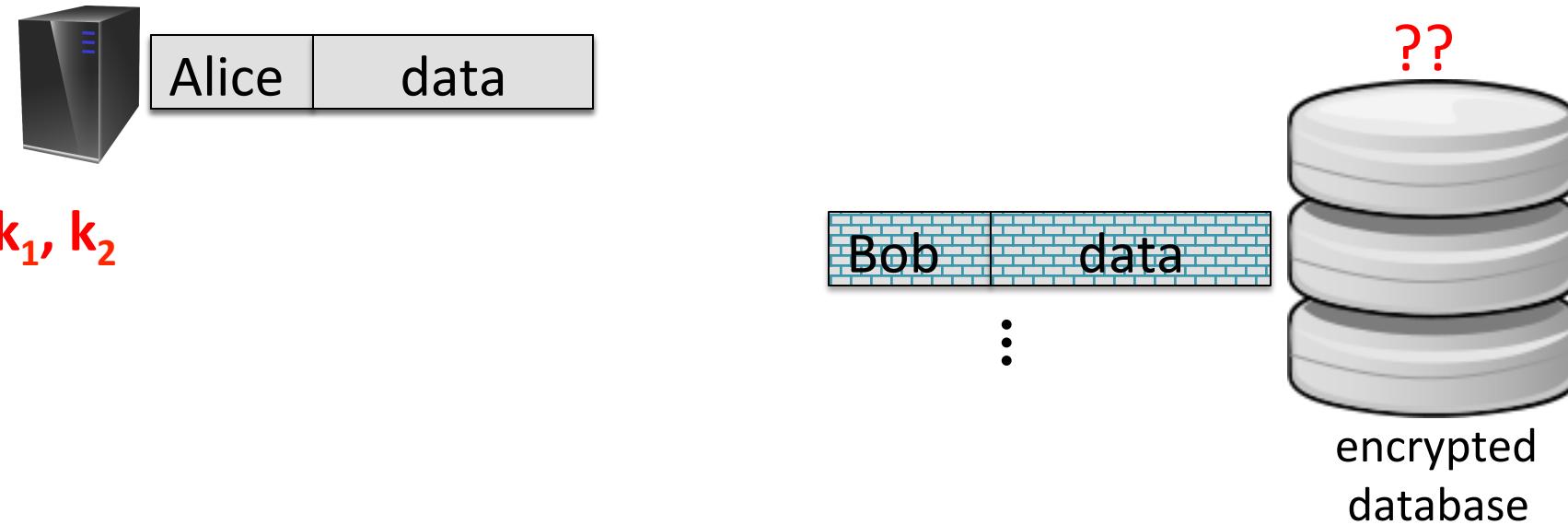
End of Segment



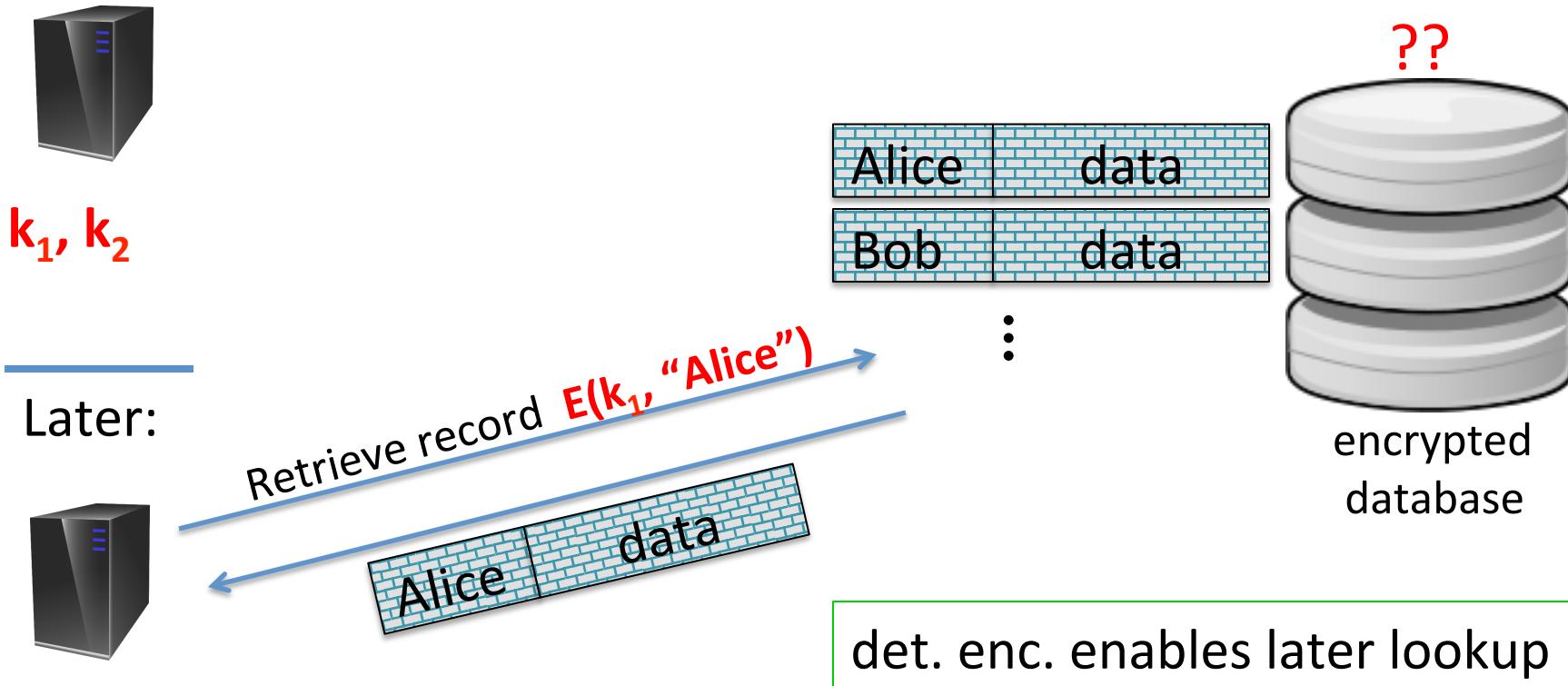
Odds and ends

Deterministic Encryption

The need for det. Encryption (no nonce)



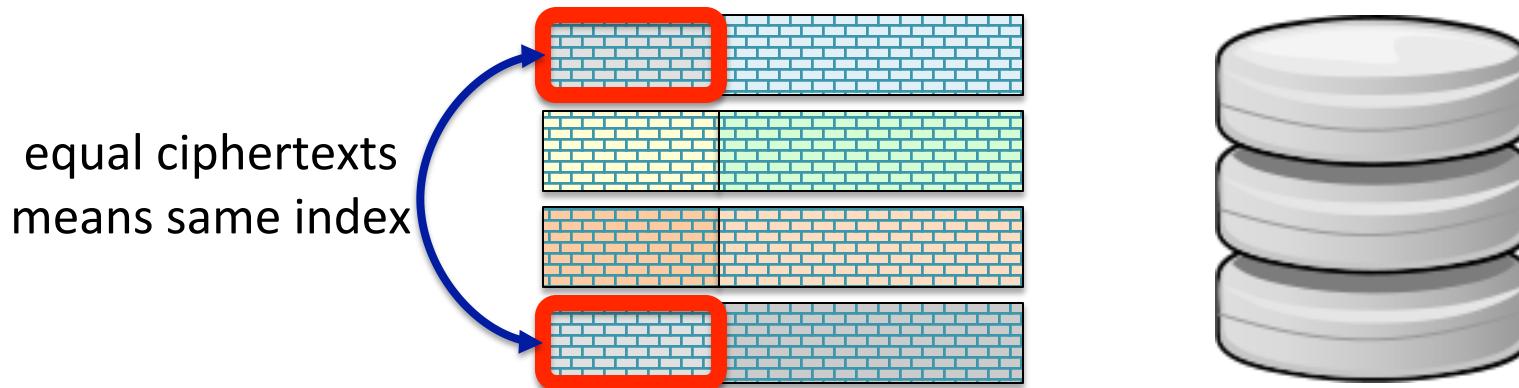
The need for det. Encryption (no nonce)



Problem: det. enc. cannot be CPA secure

The problem: attacker can tell when two ciphertexts encrypt the same message \Rightarrow leaks information

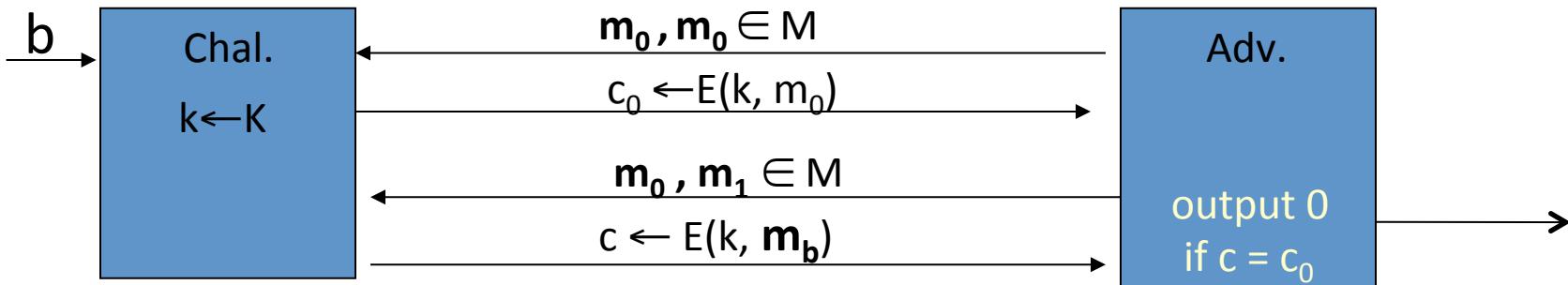
Leads to significant attacks when message space M is small.



Problem: det. enc. cannot be CPA secure

The problem: attacker can tell when two ciphertexts encrypt the same message \Rightarrow leaks information

Attacker wins CPA game:



A solution: the case of unique messages

Suppose encryptor never encrypts same message twice:

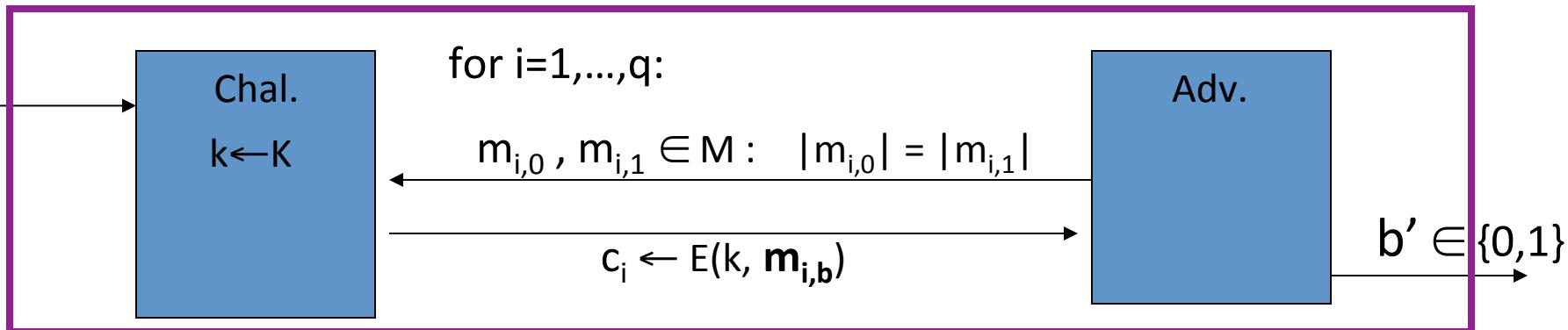
the pair (k, m) never repeats

This happens when encryptor:

- Chooses messages at random from a large msg space (e.g. keys)
- Message structure ensures uniqueness (e.g. unique user ID)

Deterministic CPA security

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



where $m_{1,0}, \dots, m_{q,0}$ are distinct and $m_{1,1}, \dots, m_{q,1}$ are distinct

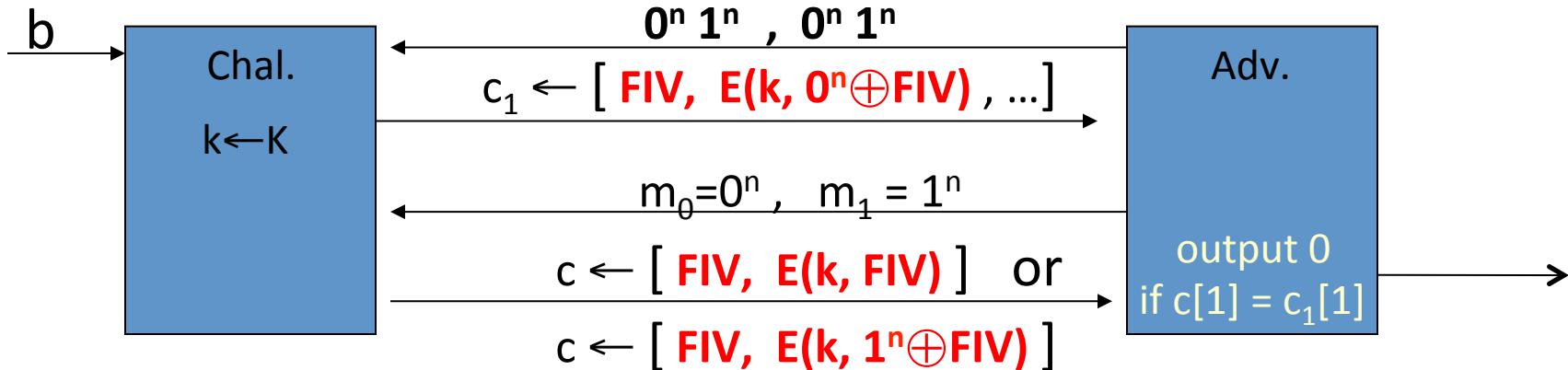
Def: E is sem. sec. under det. CPA if for all efficient A :

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

A Common Mistake

CBC with fixed IV is not det. CPA secure.

Let $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure PRP used in CBC

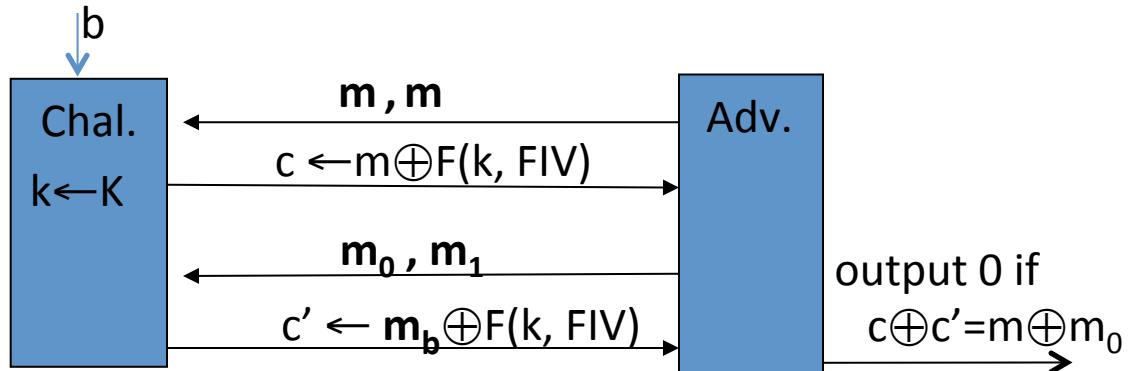


Leads to significant attacks in practice.

Is counter mode with a fixed IV det. CPA secure?



- Yes
- No
- It depends



End of Segment



Odds and ends

Deterministic Encryption
Constructions:
SIV and wide PRP

Deterministic encryption

Needed for maintaining an encrypted database index

- Lookup records by encrypted index

Deterministic CPA security:

- Security if never encrypt same message twice using same key:
the pair (key , msg) is unique

Formally: we defined deterministic CPA security game

Construction 1: Synthetic IV (SIV)

Let (E, D) be a CPA-secure encryption. $E(k, m ; r) \rightarrow c$

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

Define: $E_{\text{det}}(k_1, k_2, m) = \begin{cases} r \leftarrow F(k_1, m) \\ c \leftarrow E(k_2, m ; r) \\ \text{output } r \end{cases}$

Thm: E_{det} is sem. sec. under det. CPA .

Proof sketch: distinct msgs. \Rightarrow all r 's are indist. from random

Well suited for messages longer than one AES block (16 bytes)

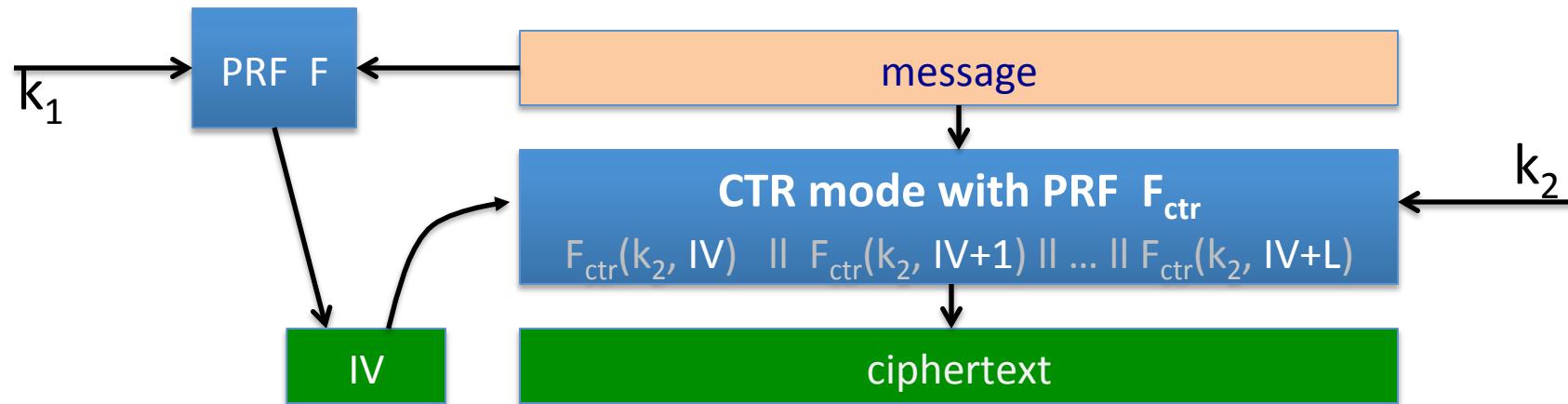
Ensuring ciphertext integrity

Goal: det. CPA security and ciphertext integrity

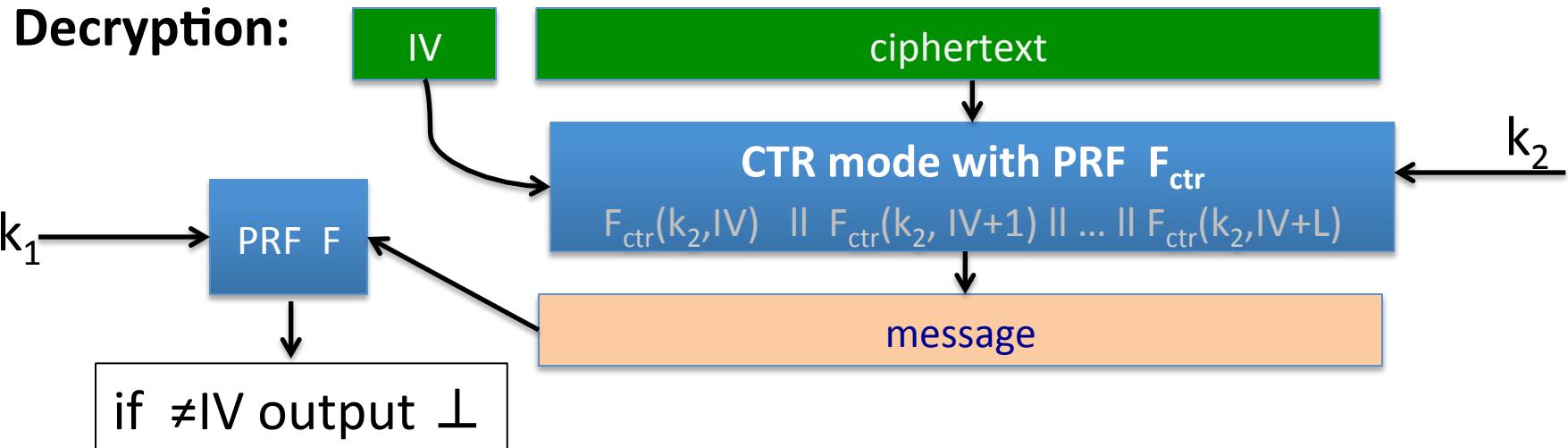
⇒ **DAE: deterministic authenticated encryption**

Consider a SIV special case: SIV-CTR

SIV where cipher is counter mode with rand. IV



Det. Auth. Enc. (DAE) for free



Thm: if F is a secure PRF and CTR from F_{ctr} is CPA-secure
then SIV-CTR from F, F_{ctr} provides DAE

Construction 2: just use a PRP

Let (E, D) be a secure PRP. $E: K \times X \rightarrow X$

Thm: (E, D) is sem. sec. under det. CPA .

Proof sketch: let $f: X \rightarrow X$ be a truly random invertible func.

in $\text{EXP}(0)$ adv. sees: $f(m_{1,0}), \dots, f(m_{q,0})$  q random values in X

in $\text{EXP}(1)$ adv. sees: $f(m_{1,1}), \dots, f(m_{q,1})$

Using AES: Det. CPA secure encryption for 16 byte messages.

Longer messages?? Need PRPs on larger msg spaces ...

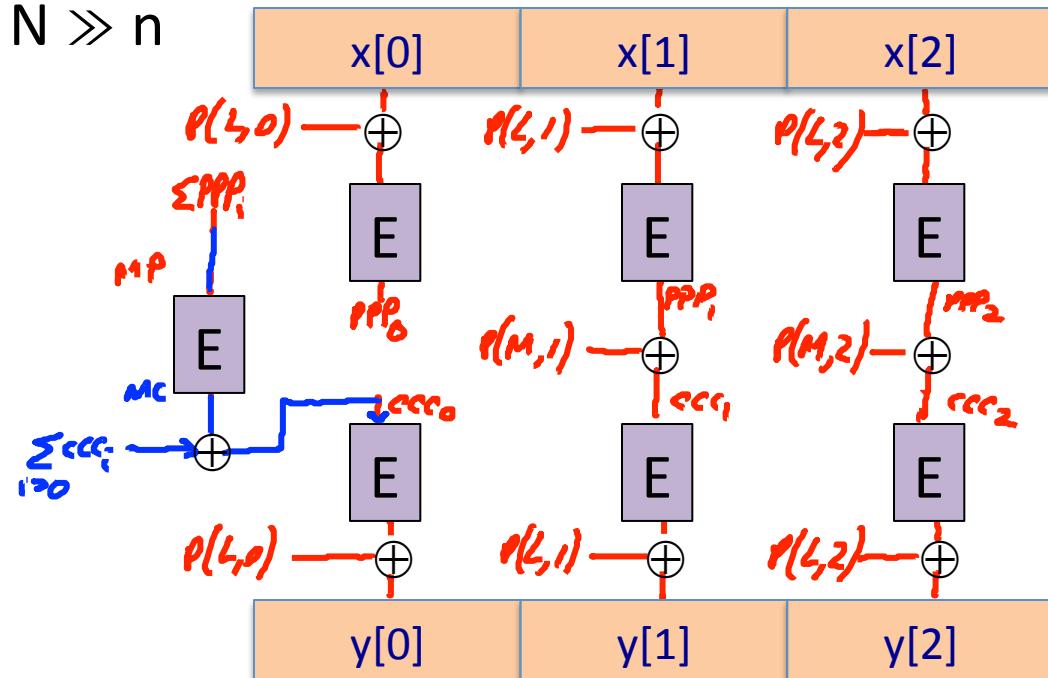
EME: constructing a wide block PRP

Let (E, D) be a secure PRP. $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$

EME: a PRP on $\{0,1\}^N$ for $N \gg n$

Key = (K, L)

$M \leftarrow MP \oplus MC$



Performance:

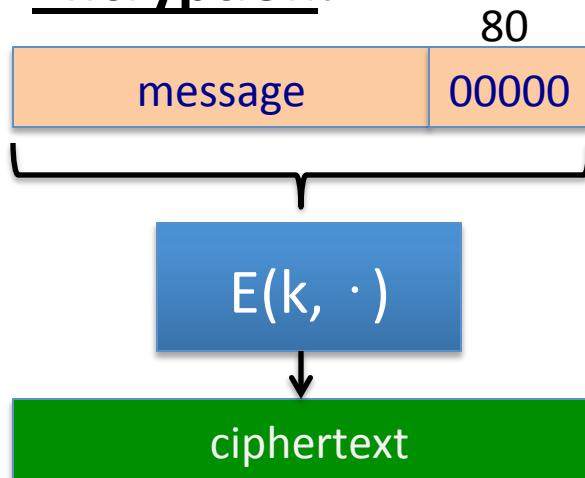
- can be 2x slower than SIV

PRP-based Det. Authenticated Enc.

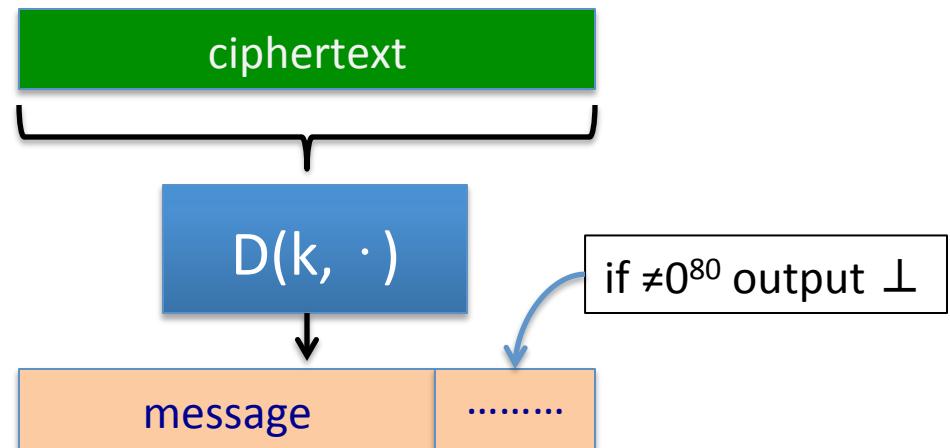
Goal: det. CPA security and ciphertext integrity

⇒ **DAE: deterministic authenticated encryption**

Encryption:



Decryption:

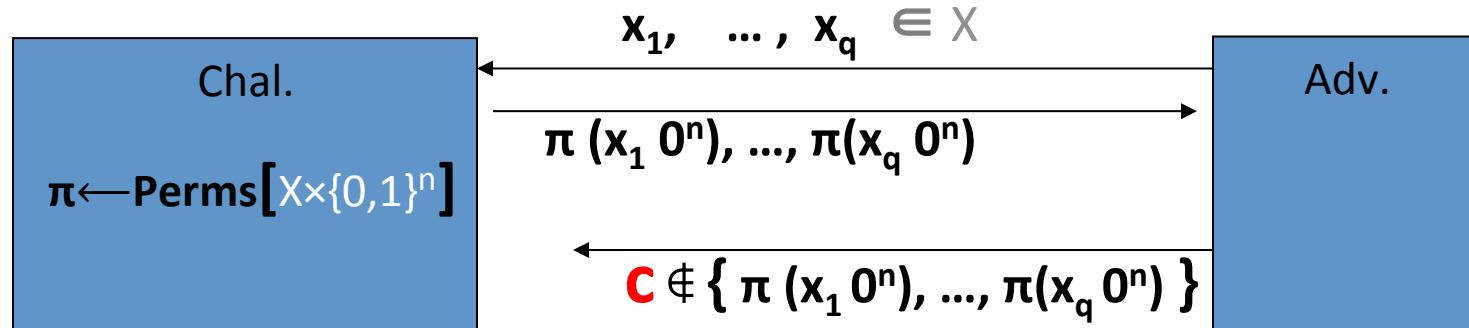


PRP-based Det. Authenticated Enc.

Let (E, D) be a secure PRP. $E: K \times (\mathbb{X} \times \{0,1\}^n) \rightarrow \mathbb{X} \times \{0,1\}^n$

Thm: $1/2^n$ is negligible \Rightarrow PRP-based enc. provides DAE

Proof sketch: suffices to prove ciphertext integrity



But then $\Pr[\text{LSB}_n(\pi^{-1}(C)) = 0^n] \leq 1/2^n$

End of Segment



Odds and ends

Tweakable encryption

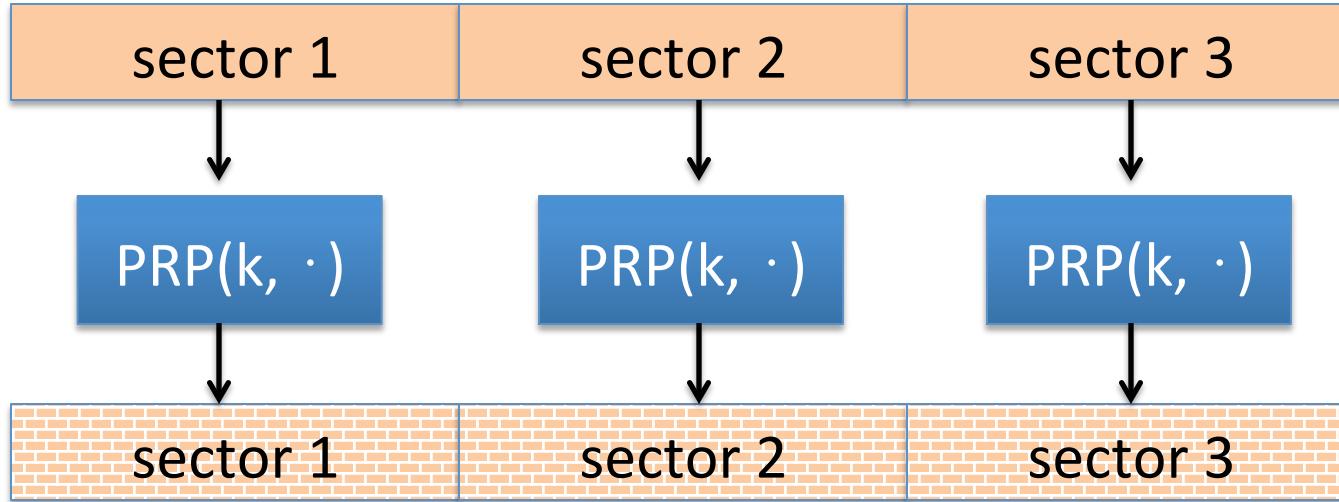
Disk encryption: no expansion

Sectors on disk are fixed size (e.g. 4KB)

- ⇒ encryption cannot expand plaintext (i.e. $M = C$)
- ⇒ must use deterministic encryption, no integrity

Lemma: if (E, D) is a det. CPA secure cipher with $M=C$
then (E, D) is a PRP.

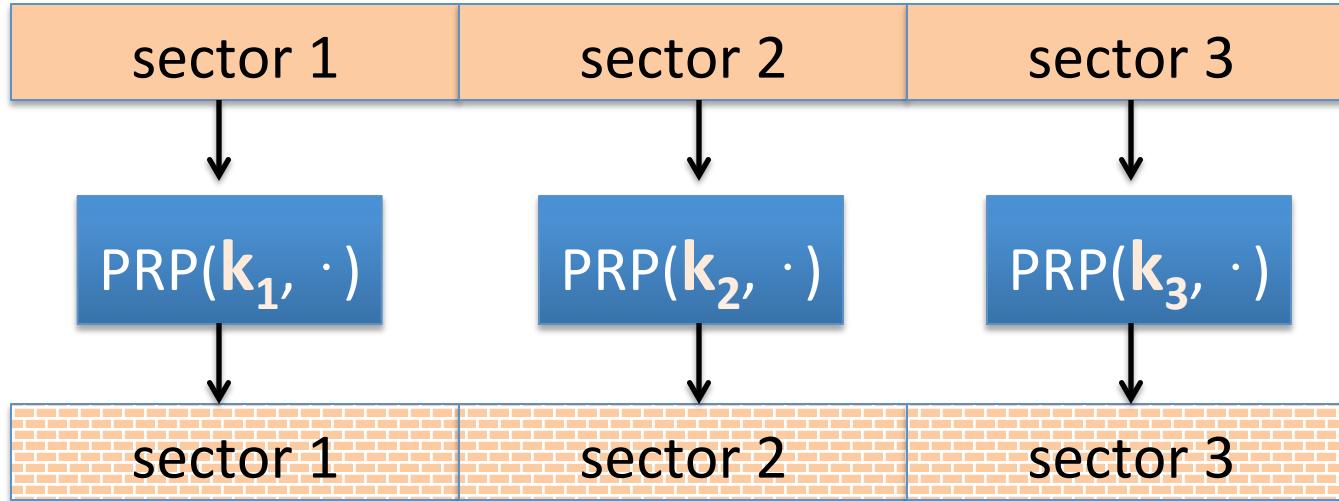
⇒ every sector will need to be encrypted with a PRP



Problem: sector 1 and sector 3 may have same content

- Leaks same information as ECB mode

Can we do better?



Avoids previous leakage problem

- ... but attacker can tell if a sector is changed and then reverted

Managing keys: the trivial construction $k_t = \text{PRF}(k, t)$, $t=1,\dots,L$

Can we do better?

Tweakable block ciphers

Goal: construct many PRPs from a key $k \in K$.

Syntax: $E, D: K \times T \times X \rightarrow X$

for every $t \in T$ and $k \leftarrow K$:

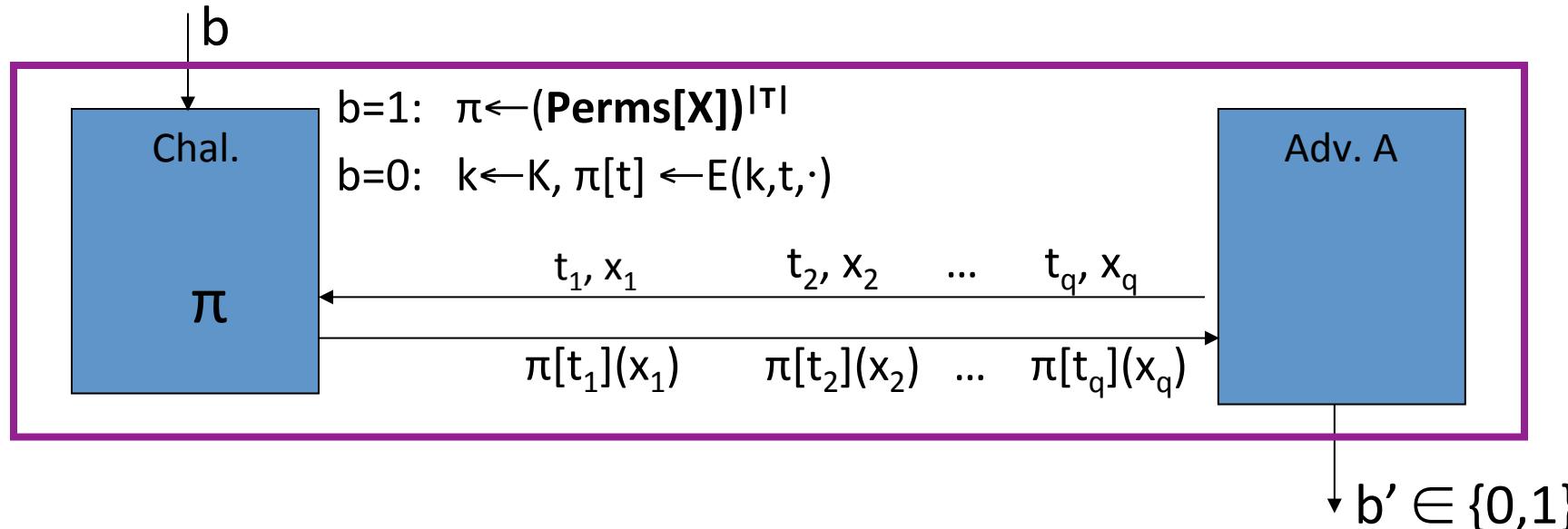
$E(k, t, \cdot)$ is an invertible func. on X , indist. from random

Application: use sector number as the tweak

\Rightarrow every sector gets its own independent PRP

Secure tweakable block ciphers

$E, D : K \times T \times X \rightarrow X$. For $b=0,1$ define experiment $\text{EXP}(b)$ as:



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

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

Example 1: the trivial construction

Let (E, D) be a secure PRP, $E: K \times X \rightarrow X$.

- The trivial tweakable construction: (suppose $K = X$)

$$E_{\text{tweak}}(k, t, x) = E(E(k, t), x)$$

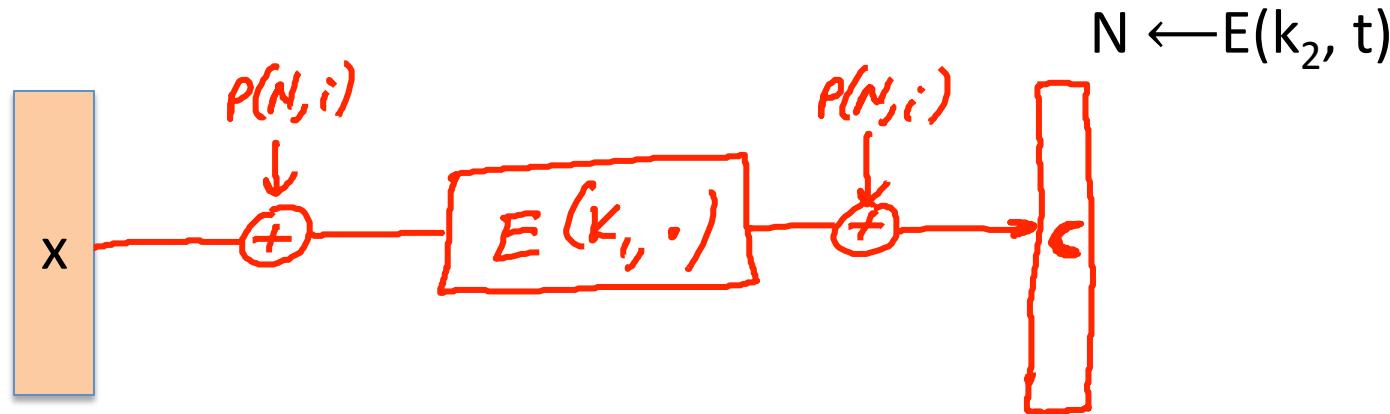
\Rightarrow to encrypt n blocks need $2n$ evals of $E(.,.)$

2. the XTS tweakable block cipher

[R'04]

Let (E, D) be a secure PRP, $E: K \times \{0,1\}^n \rightarrow \{0,1\}^n$.

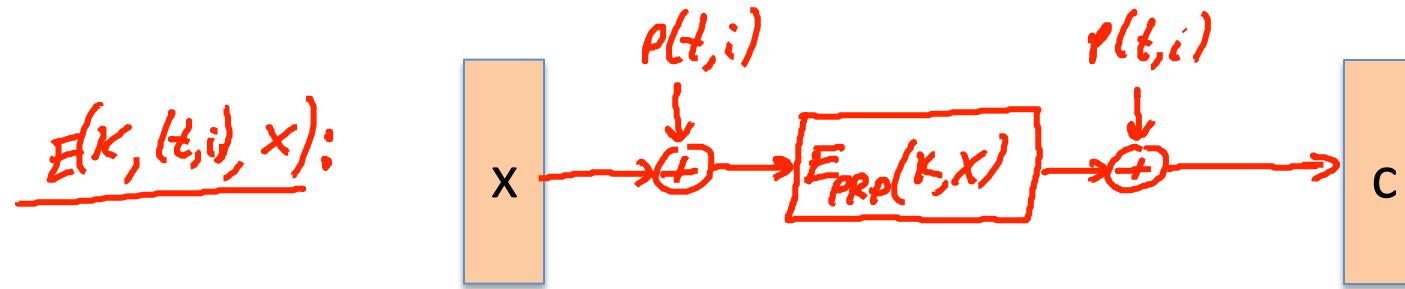
- XTS: $E_{\text{tweak}}((k_1, k_2), (t, i), x) =$



⇒ to encrypt n blocks need $n+1$ evals of $E(.,.)$

Is it necessary to encrypt the tweak before using it?

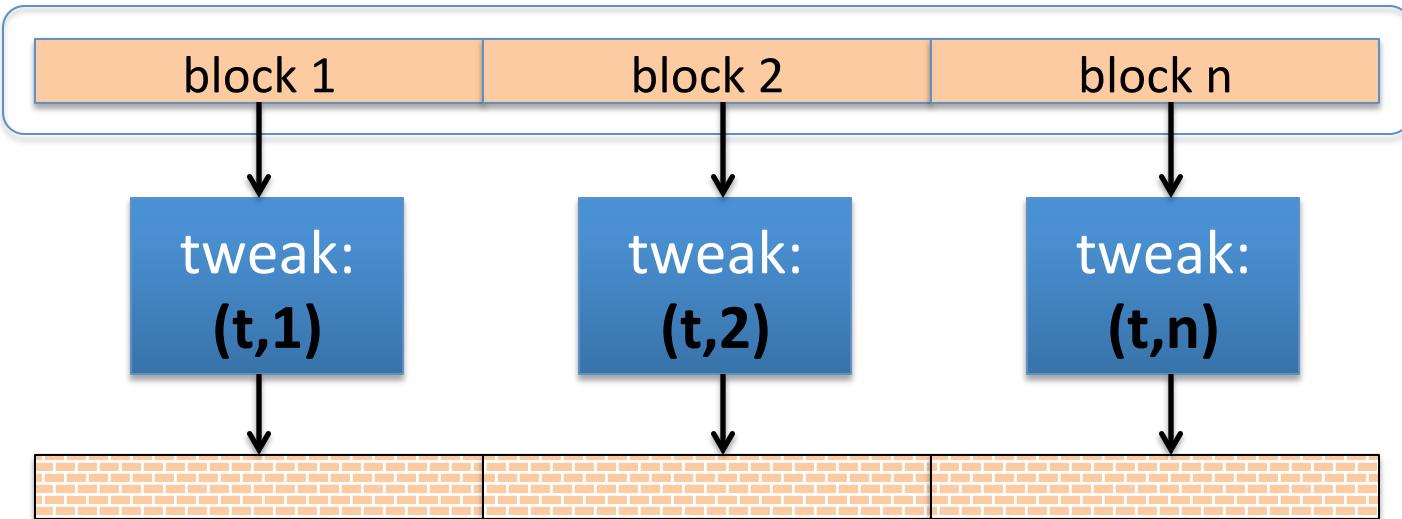
That is, is the following a secure tweakable PRP?



- Yes, it is secure
- No: $E(k, (t,1), P(t,2)) \oplus E(k, (t,2), P(t,1)) = P(t,1)$
- No: $E(k, (t,1), P(t,1)) \oplus E(k, (t,2), P(t,2)) = P(t,1) \oplus P(t,2)$
- No: $E(k, (t,1), P(t,1)) \oplus E(k, (t,2), P(t,2)) = 0$

Disk encryption using XTS

sector # t:



- note: block-level PRP, not sector-level PRP.
- Popular in disk encryption products:
Mac OS X-Lion, TrueCrypt, BestCrypt, ...

Summary

- Use tweakable encryption when you need many independent PRPs from one key
- XTS is more efficient than the trivial construction
 - Both are narrow block: 16 bytes for AES
- EME (previous segment) is a tweakable mode for wide block
 - 2x slower than XTS

End of Segment



Odds and ends

Format preserving
encryption

Encrypting credit card numbers

Credit card format: **bbbb bbnn nnnn nnnc** (≈ 42 bits)



Goal: end-to-end encryption

Intermediate processors expect to see a credit card number
⇒ encrypted credit card should look like a credit card

Format preserving encryption (FPE)

This segment: given $0 < s \leq 2^n$, build a PRP on $\{0, \dots, s-1\}$

from a secure PRF $F: K \times \{0,1\}^n \rightarrow \{0,1\}^n$ (e.g. AES)

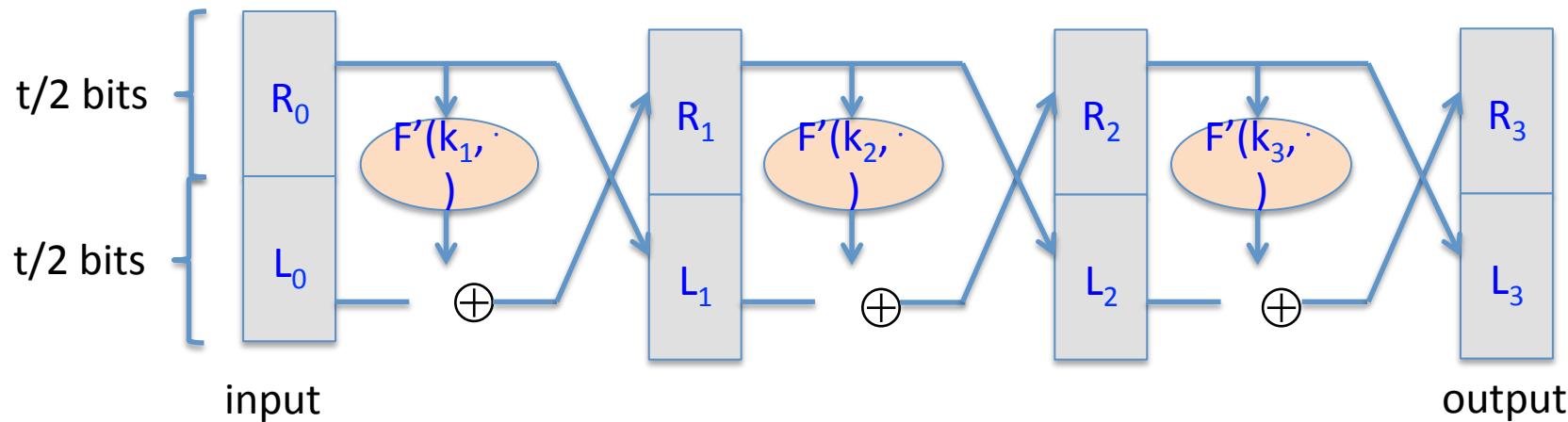
Then to encrypt a credit card number: ($s = \text{total } \# \text{ credit cards}$)

1. map given CC# to $\{0, \dots, s-1\}$
2. apply PRP to get an output in $\{0, \dots, s-1\}$
3. map output back a to CC#

Step 1: from $\{0,1\}^n$ to $\{0,1\}^t$ ($t < n$)

Want PRP on $\{0, \dots, s-1\}$. Let t be such that $2^{t-1} < s \leq 2^t$.

Method: Luby-Rackoff with $F': K \times \{0,1\}^{t/2} \rightarrow \{0,1\}^{t/2}$ (truncate F)



(better to use 7 rounds a la Patarin, Crypto'03)

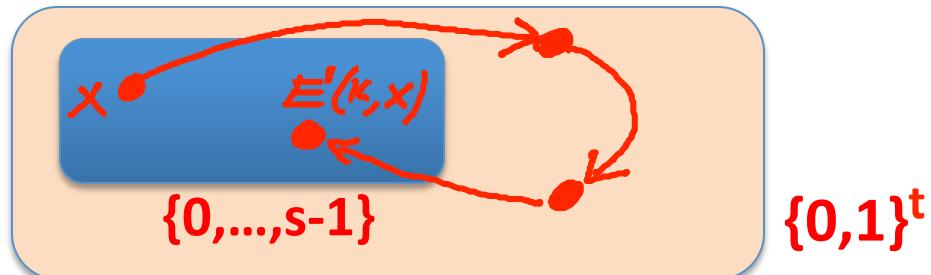
Step 2: from $\{0,1\}^t$ to $\{0,\dots,s-1\}$

Given PRP $(E,D): K \times \{0,1\}^t \rightarrow \{0,1\}^t$

we build $(E',D'): K \times \{0,\dots,s-1\} \rightarrow \{0,\dots,s-1\}$

$E'(k, x)$: on input $x \in \{0,\dots,s-1\}$ do:

$y \leftarrow x$; do { $y \leftarrow E(k, y)$ } until $y \in \{0,\dots,s-1\}$; output y



Expected # iterations: 2

Security

Step 2 is tight: $\forall A \exists B: \text{PRP}_{\text{adv}}[A, E] = \text{PRP}_{\text{adv}}[B, E']$

Intuition: \forall sets $Y \subseteq X$, applying the transformation to a random perm. $\pi: X \rightarrow X$
gives a random perm. $\pi': Y \rightarrow Y$

Step 1: same security as Luby-Rackoff construction
(actually using analysis of Patarin, Crypto'03)

note: no integrity

Further reading

- Cryptographic Extraction and Key Derivation: The HKDF Scheme.
H. Krawczyk, Crypto 2010
- Deterministic Authenticated-Encryption:
A Provable-Security Treatment of the Keywrap Problem.
P. Rogaway, T. Shrimpton, Eurocrypt 2006
- A Parallelizable Enciphering Mode. S. Halevi, P. Rogaway, CT-RSA 2004
- Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. P. Rogaway, Asiacrypt 2004
- How to Encipher Messages on a Small Domain:
Deterministic Encryption and the Thorp Shuffle.
B. Morris, P. Rogaway, T. Stegers, Crypto 2009

End of Segment

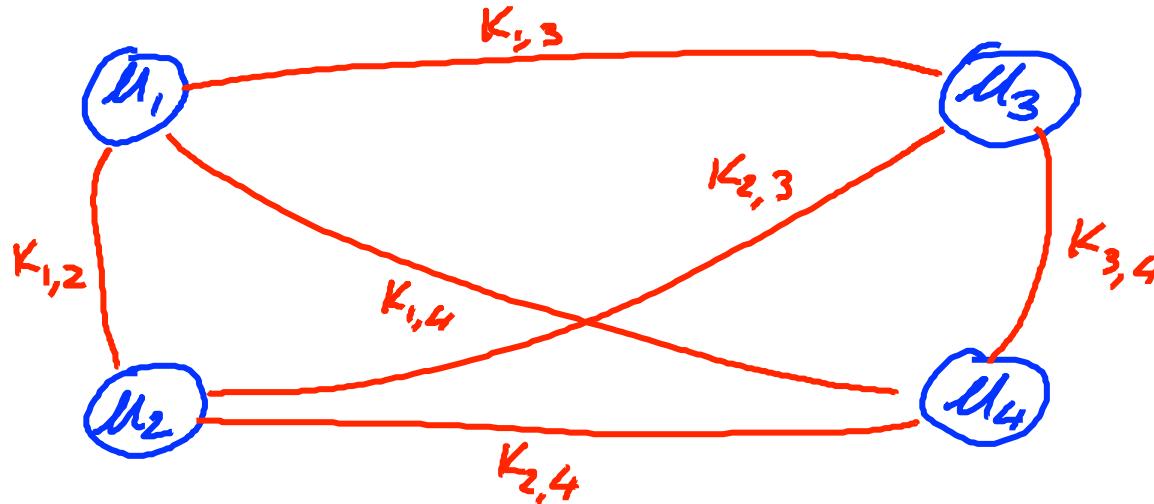


Basic key exchange

Trusted 3rd parties

Key management

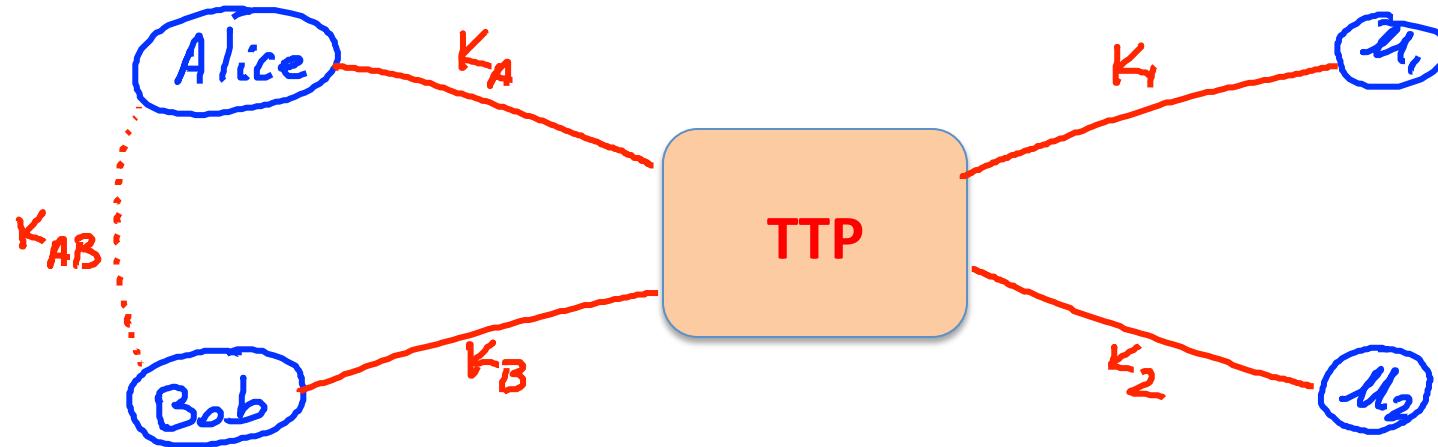
Problem: n users. Storing mutual secret keys is difficult



Total: $O(n)$ keys per user

A better solution

Online Trusted 3rd Party (TTP)



Every user only remembers one key.

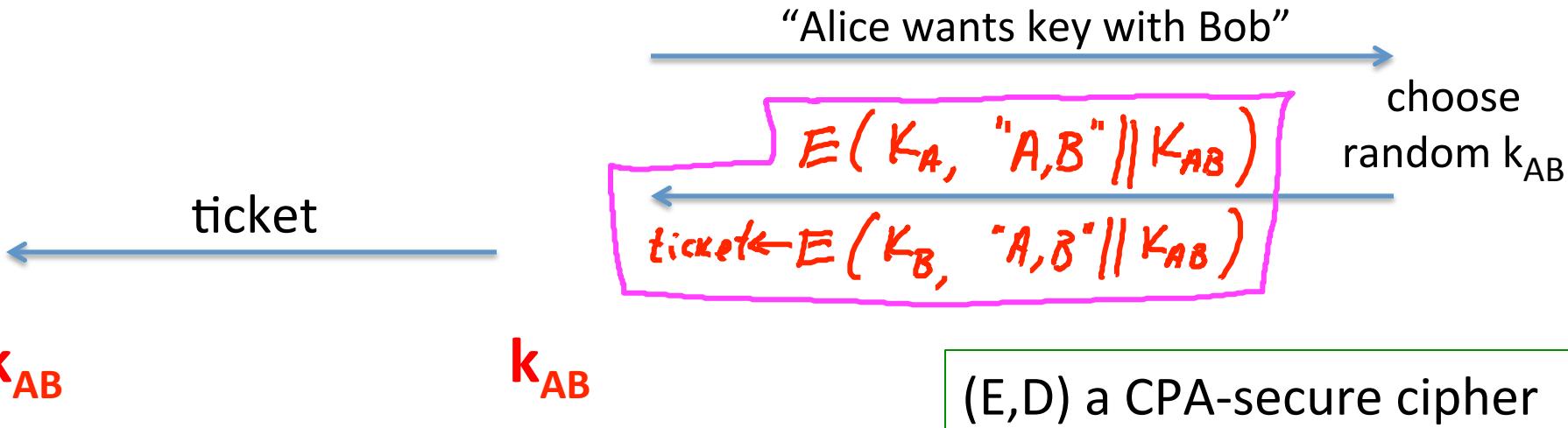
Generating keys: a toy protocol

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

Bob (k_B)

Alice (k_A)

TTP



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: $E(k_A, "A, B" \parallel k_{AB})$; $E(k_B, "A, B" \parallel k_{AB})$

(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)

Toy protocol: insecure against active attacks

Example: insecure against replay attacks

Attacker records session between Alice and merchant Bob

- For example a book order

Attacker replays session to Bob

- Bob thinks Alice is ordering another copy of book

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)

End of Segment



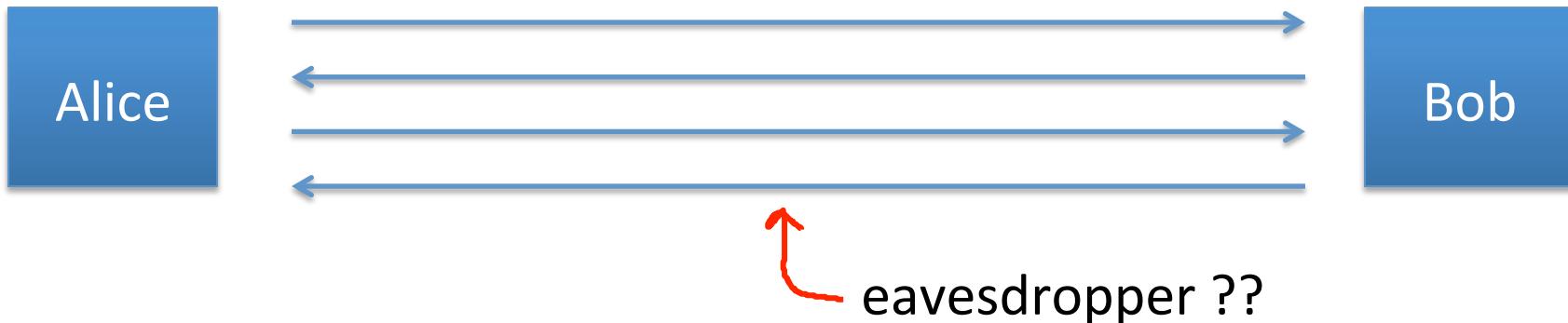
Basic key exchange

Merkle Puzzles

Key exchange without an online TTP?

Goal: Alice and Bob want shared key, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



Can this be done using generic symmetric crypto?

Merkle Puzzles (1974)

Answer: yes, but very inefficient

Main tool: puzzles

- Problems that can be solved with some effort
- Example: $E(k,m)$ a symmetric cipher with $k \in \{0,1\}^{128}$
 - $\text{puzzle}(P) = E(P, \text{"message"})$ where $P = 0^{96} \parallel b_1 \dots b_{32}$
 - Goal: find P by trying all 2^{32} possibilities

Merkle puzzles

Alice: prepare 2^{32} puzzles

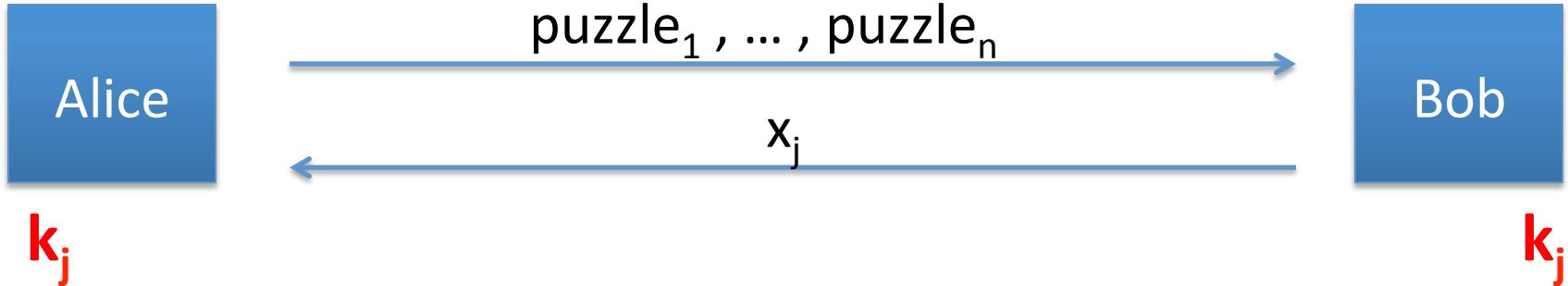
- For $i=1, \dots, 2^{32}$ choose random $P_i \in \{0,1\}^{32}$ and $x_i, k_i \in \{0,1\}^{128}$
set $\text{puzzle}_i \leftarrow E(0^{96} \parallel P_i, \text{"Puzzle \# } x_i \text{"} \parallel k_i)$
- Send $\text{puzzle}_1, \dots, \text{puzzle}_{2^{32}}$ to Bob

Bob: choose a random puzzle_j and solve it. Obtain (x_j, k_j) .

- Send x_j to Alice

Alice: lookup puzzle with number x_j . Use k_j as shared secret

In a figure



Alice's work: $O(n)$ (prepare n puzzles)

Bob's work: $O(n)$ (solve one puzzle)

Eavesdropper's work: $O(n^2)$ (e.g. 2^{64} time)

Impossibility Result

Can we achieve a better gap using a general symmetric cipher?

Answer: unknown

But: roughly speaking,

quadratic gap is best possible if we treat cipher as
a black box oracle [IR'89, BM'09]

End of Segment



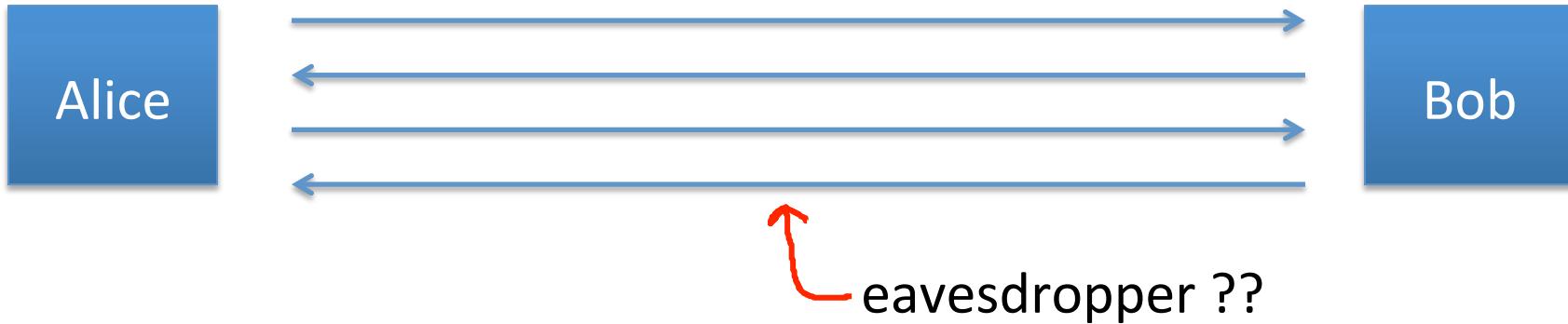
Basic key exchange

The Diffie-Hellman protocol

Key exchange without an online TTP?

Goal: Alice and Bob want shared secret, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



Can this be done with an exponential gap?

The Diffie-Hellman protocol (informally)

Fix a large prime p (e.g. 600 digits)

Fix an integer g in $\{1, \dots, p\}$

Alice

choose random a in $\{1, \dots, p-1\}$

Bob

choose random b in $\{1, \dots, p-1\}$

$$\text{"Alice"}, \quad A \leftarrow g^a \pmod{p}$$

$$\text{"Bob"}, \quad B \leftarrow g^b \pmod{p}$$

$$B^a \pmod{p} = (g^b)^a = k_{AB} = g^{ab} \pmod{p} = (g^a)^b = A^b \pmod{p}$$

Security (much more on this later)

Eavesdropper sees: $p, g, A=g^a \pmod{p}$, and $B=g^b \pmod{p}$

Can she compute $g^{ab} \pmod{p}$??

More generally: define $DH_g(g^a, g^b) = g^{ab} \pmod{p}$

How hard is the DH function mod p?

How hard is the DH function mod p?

Suppose prime p is n bits long.

Best known algorithm (GNFS): run time $\exp(\tilde{O}(\sqrt[3]{n}))$

<u>cipher key size</u>	<u>modulus size</u>	<u>Elliptic Curve size</u>
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	<u>15360</u> bits	512 bits

As a result: slow transition away from $(\text{mod } p)$ to elliptic curves



[www.google.com](#)

The identity of this website has been verified by Thawte SGC CA.

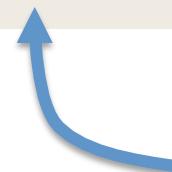
[Certificate Information](#)



Your connection to www.google.com is encrypted with 128-bit encryption.

The connection uses TLS 1.0.

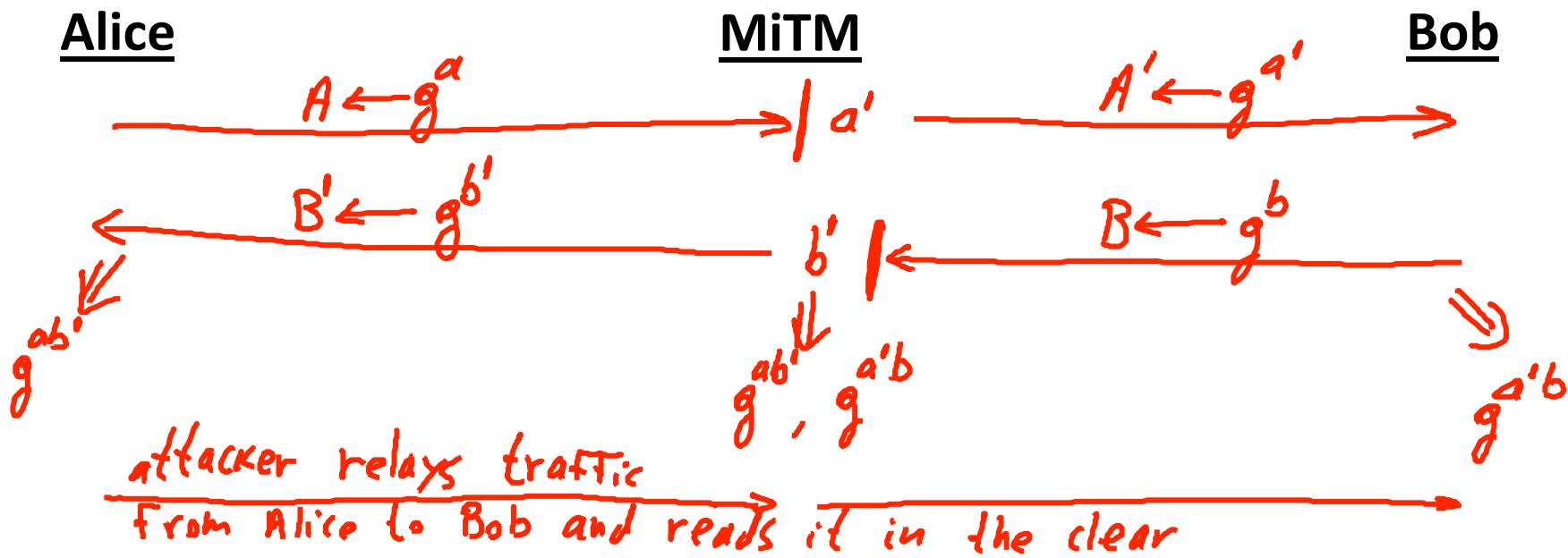
The connection is encrypted using RC4_128, with SHA1 for message authentication and ECDHE_RSA as the key exchange mechanism.



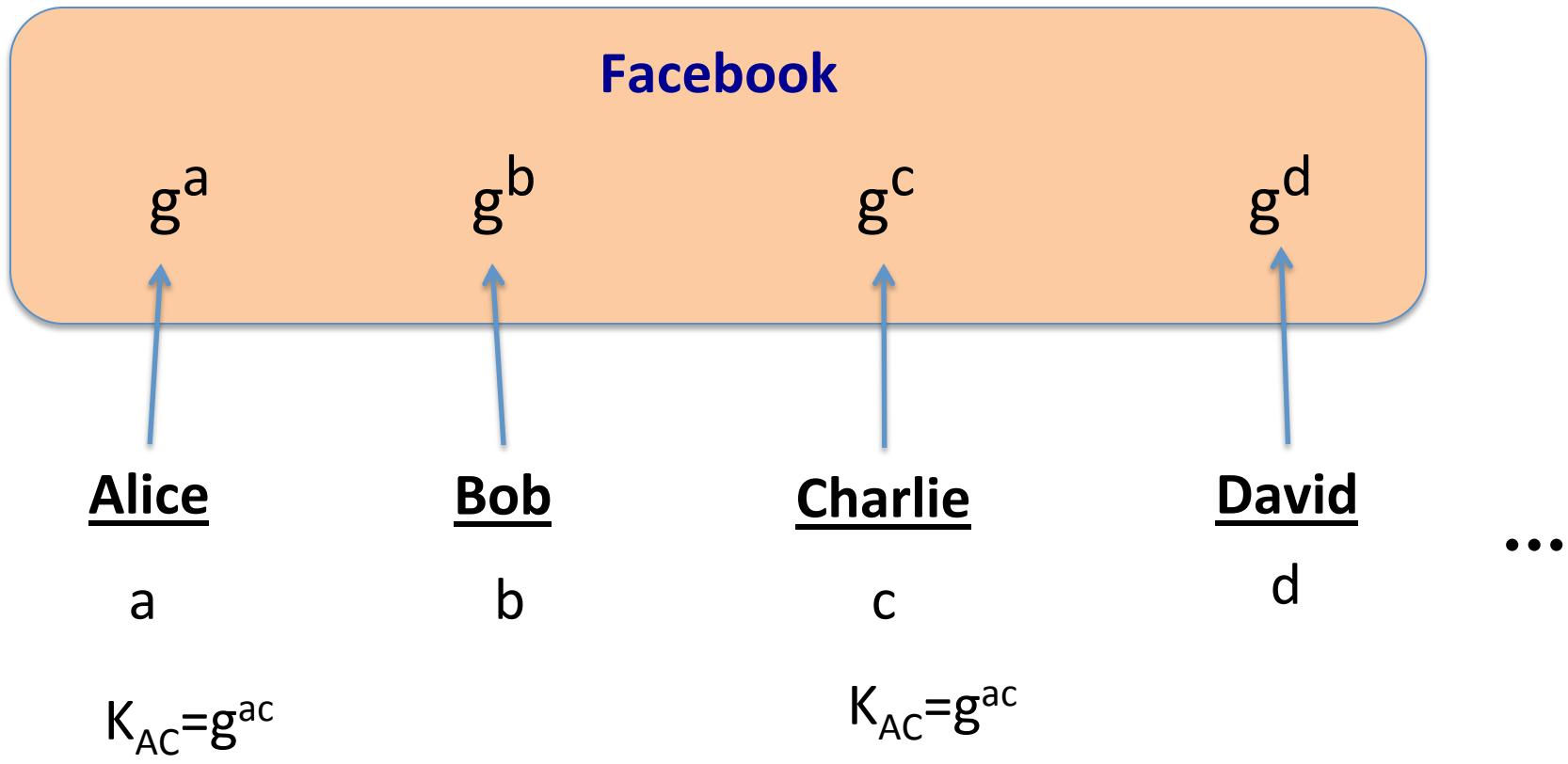
Elliptic curve
Diffie-Hellman

Insecure against man-in-the-middle

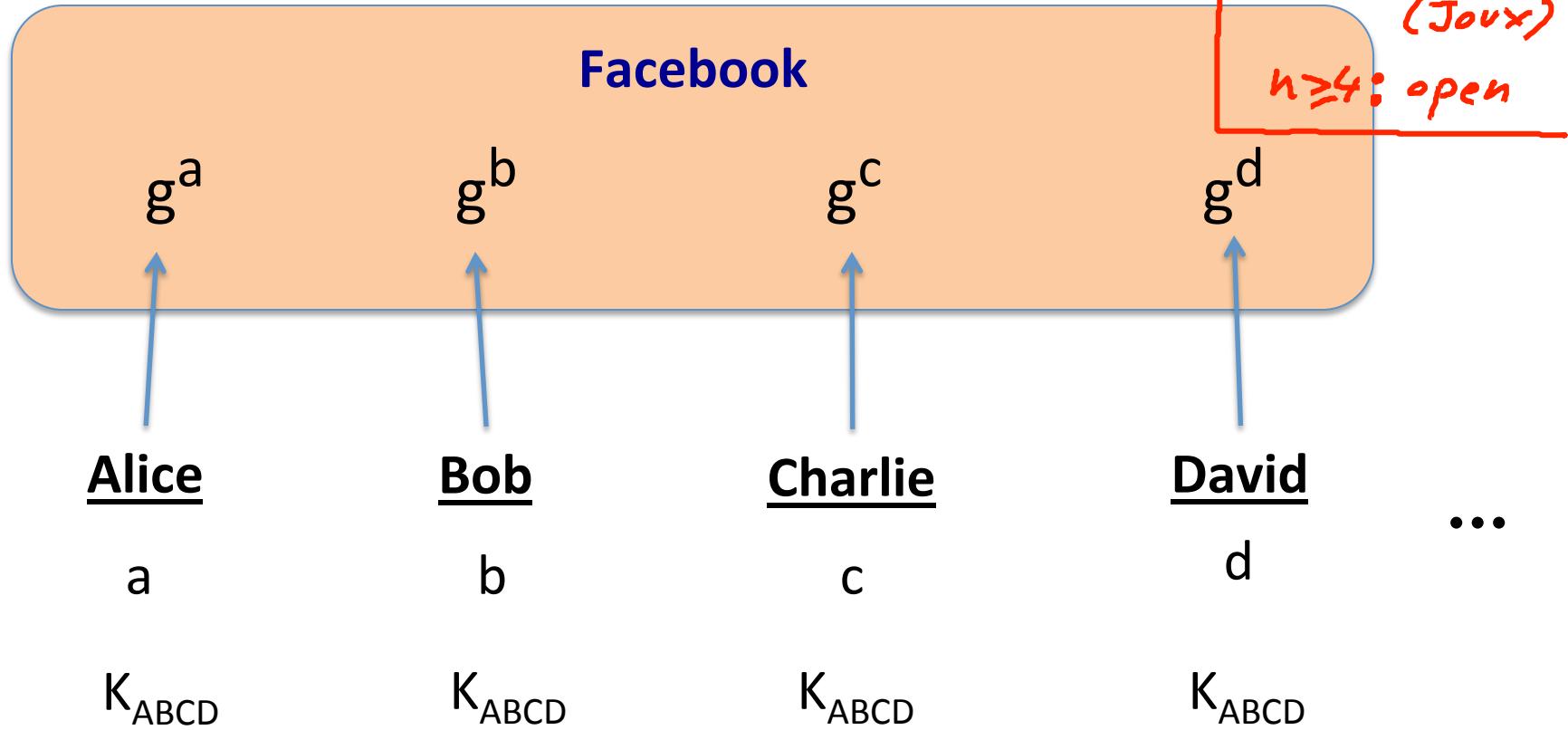
As described, the protocol is insecure against **active** attacks



Another look at DH



An open problem



End of Segment



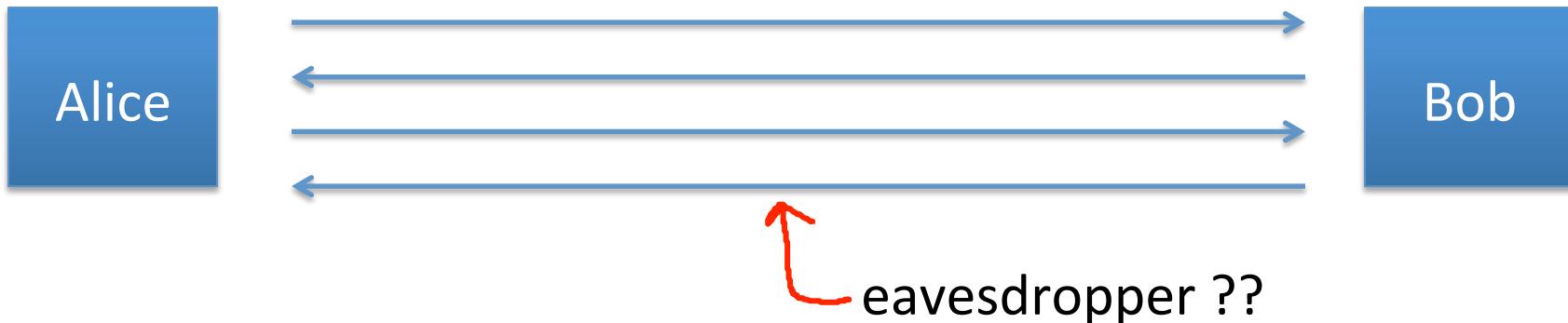
Basic key exchange

Public-key encryption

Establishing a shared secret

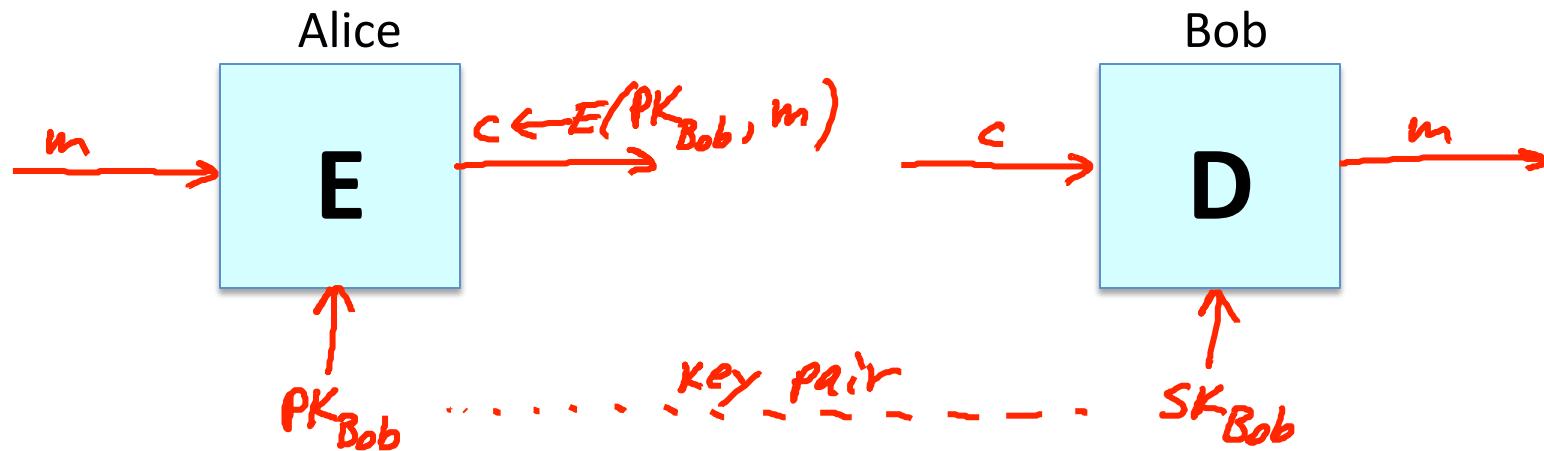
Goal: Alice and Bob want shared secret, unknown to eavesdropper

- For now: security against eavesdropping only (no tampering)



This segment: a different approach

Public key encryption



PK : public key , SK : secret key

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

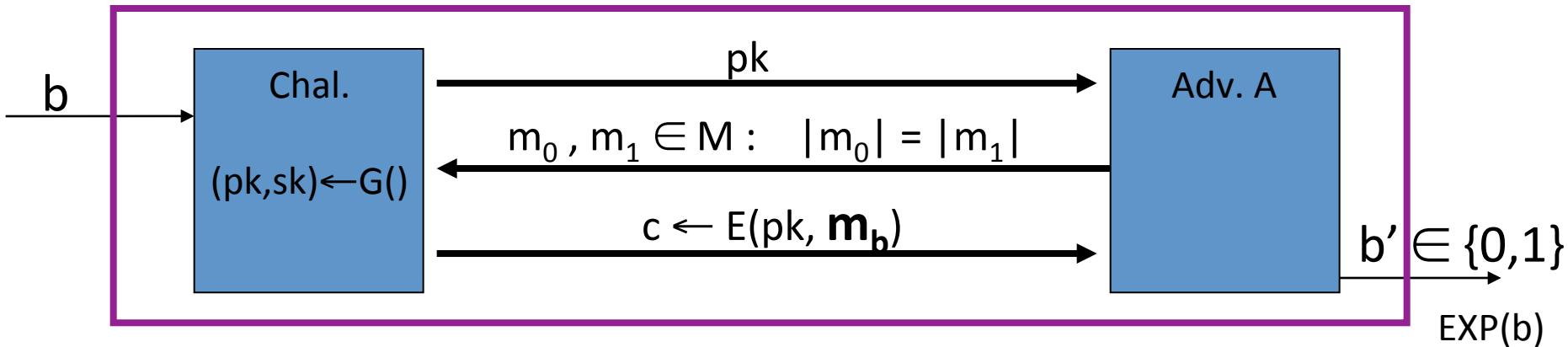
- $G()$: randomized alg. outputs a key pair (pk, sk)
- $E(pk, m)$: randomized alg. that takes $m \in M$ and outputs $c \in C$
- $D(sk, c)$: det. alg. that takes $c \in C$ and outputs $m \in M$ or \perp

Consistency: $\forall (pk, sk) \text{ output by } G :$

$$\forall m \in M : D(sk, E(pk, m)) = m$$

Semantic Security

For $b=0,1$ define experiments $\text{EXP}(0)$ and $\text{EXP}(1)$ as:



Def: $E = (G, E, D)$ is sem. secure (a.k.a IND-CPA) if for all efficient A :

$$\text{Adv}_{\text{SS}}[A, E] = |\Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1]| < \text{negligible}$$

Establishing a shared secret

Alice

$(pk, sk) \leftarrow G()$

Bob

choose random
 $x \in \{0,1\}^{128}$

“Alice”, pk

“Bob”, $c \leftarrow E(pk, x)$

$D(sk, c) \rightarrow x$

x : shared secret

Security (eavesdropping)

Adversary sees $\text{pk}, \ E(\text{pk}, x)$ and wants $x \in M$

Semantic security \Rightarrow

adversary cannot distinguish

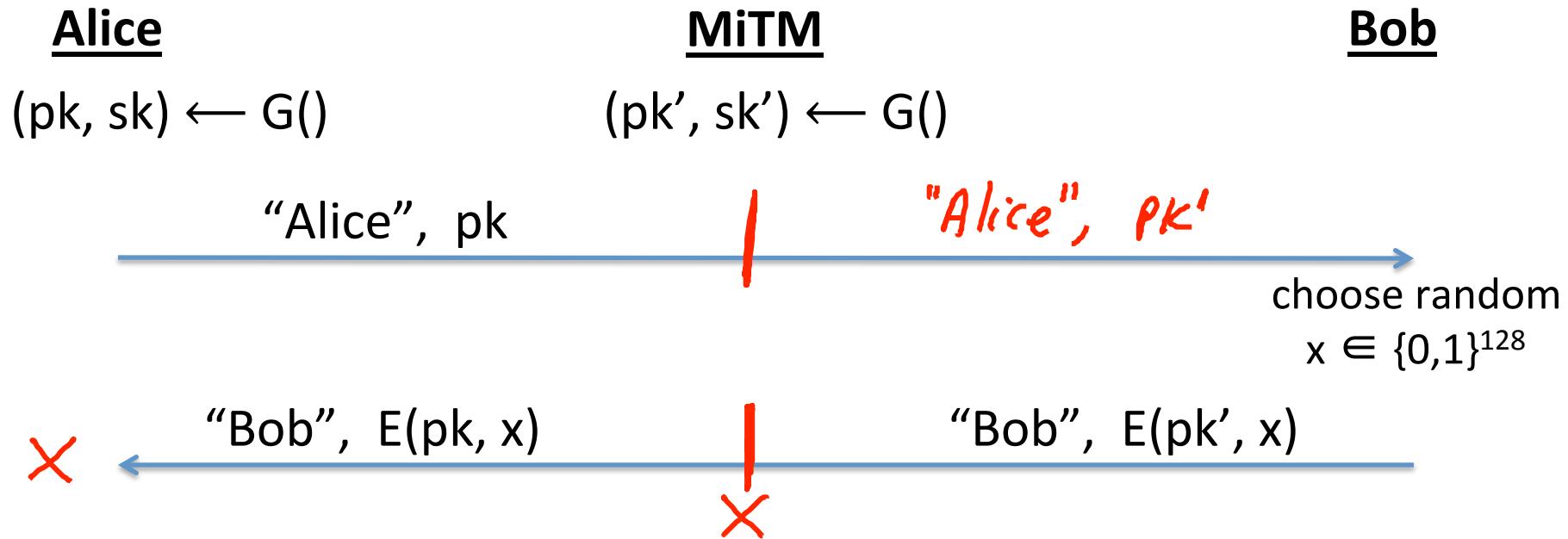
$\{ \text{pk}, \ E(\text{pk}, x), \ x \}$ from $\{ \text{pk}, \ E(\text{pk}, x), \ \text{rand} \in M \}$

\Rightarrow can derive session key from x .

Note: protocol is vulnerable to man-in-the-middle

Insecure against man in the middle

As described, the protocol is insecure against **active** attacks



Public key encryption: constructions

Constructions generally rely on hard problems from number theory and algebra

Next module:

- Brief detour to catch up on the relevant background

Further readings

- Merkle Puzzles are Optimal,
B. Barak, M. Mahmoody-Ghidary, Crypto '09
- On formal models of key exchange (sections 7-9)
V. Shoup, 1999

End of Segment



Intro. Number Theory

Notation

Background

We will use a bit of number theory to construct:

- Key exchange protocols
- Digital signatures
- Public-key encryption

This module: crash course on relevant concepts

More info: read parts of Shoup's book referenced
at end of module

Notation

From here on:

- N denotes a positive integer.
- p denote a prime.

Notation: $\mathbb{Z}_N = \{0, 1, 2, \dots, N-1\}$

Can do addition and multiplication modulo N

Modular arithmetic

Examples: let $N = 12$

$$9 + 8 = 5 \quad \text{in } \mathbb{Z}_{12}$$

$$5 \times 7 = 11 \quad \text{in } \mathbb{Z}_{12}$$

$$5 - 7 = 10 \quad \text{in } \mathbb{Z}_{12}$$

Arithmetic in \mathbb{Z}_N works as you expect, e.g. $x \cdot (y+z) = x \cdot y + x \cdot z$ in \mathbb{Z}_N

Greatest common divisor

Def: For ints. x, y : $\gcd(x, y)$ is the greatest common divisor of x, y

Example: $\gcd(12, 18) = 6$

$$\boxed{2} \times 12 - \boxed{-1} \times 18 = 6$$

Fact: for all ints. x, y there exist ints. a, b such that

$$a \cdot x + b \cdot y = \gcd(x, y)$$

a, b can be found efficiently using the extended Euclid alg.

If $\gcd(x, y) = 1$ we say that x and y are relatively prime

Modular inversion

Over the rationals, inverse of 2 is $\frac{1}{2}$. What about \mathbb{Z}_N ?

Def: The **inverse** of x in \mathbb{Z}_N is an element y in \mathbb{Z}_N s.t. $x \cdot y = 1$ in \mathbb{Z}_N

y is denoted x^{-1} .

Example: let N be an odd integer. The inverse of 2 in \mathbb{Z}_N is $\frac{N+1}{2}$

$$2 \cdot \left(\frac{N+1}{2}\right) = N+1 = 1 \text{ in } \mathbb{Z}_N$$

Modular inversion

Which elements have an inverse in \mathbb{Z}_N ?

Lemma: x in \mathbb{Z}_N has an inverse if and only if $\gcd(x, N) = 1$

Proof:

$$\begin{aligned}\gcd(x, N) = 1 \Rightarrow \exists a, b: a \cdot x + b \cdot N = 1 &\Rightarrow a \cdot x = 1 \text{ in } \mathbb{Z}_N \\ &\Rightarrow x^{-1} = a \text{ in } \mathbb{Z}_N\end{aligned}$$

$$\begin{aligned}\gcd(x, N) > 1 \Rightarrow \forall a: \gcd(a \cdot x, N) > 1 \Rightarrow a \cdot x \neq 1 \text{ in } \mathbb{Z}_N \\ \gcd(x, N) = 2 \Rightarrow \forall a: a \cdot x \text{ is even} \Rightarrow \frac{\text{even}}{a \cdot x} \neq \frac{\text{odd}}{b \cdot N + 1}\end{aligned}$$

More notation

Def: $\mathbb{Z}_N^* = (\text{set of invertible elements in } \mathbb{Z}_N) = \{ x \in \mathbb{Z}_N : \gcd(x, N) = 1 \}$

Examples:

1. for prime p , $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} = \{1, 2, \dots, p - 1\}$
2. $\mathbb{Z}_{12}^* = \{1, 5, 7, 11\}$

For x in \mathbb{Z}_N^* , can find x^{-1} using extended Euclid algorithm.

Solving modular linear equations

Solve: $a \cdot x + b = 0$ in \mathbb{Z}_N

Solution: $x = -b \cdot a^{-1}$ in \mathbb{Z}_N

Find a^{-1} in \mathbb{Z}_N using extended Euclid. Run time: $O(\log^2 N)$

What about modular quadratic equations?
next segments

End of Segment



Intro. Number Theory

Fermat and Euler

Review

N denotes an n -bit positive integer. p denotes a prime.

- $Z_N = \{ 0, 1, \dots, N-1 \}$
- $(Z_N)^* = (\text{set of invertible elements in } Z_N) = \{ x \in Z_N : \gcd(x, N) = 1 \}$

Can find inverses efficiently using Euclid alg.: time = $O(n^2)$

Fermat's theorem (1640)

Thm: Let p be a prime

$$\forall x \in (\mathbb{Z}_p)^*: x^{p-1} = 1 \text{ in } \mathbb{Z}_p$$

Example: $p=5.$ $3^4 = 81 = 1 \text{ in } \mathbb{Z}_5$

So: $x \in (\mathbb{Z}_p)^* \Rightarrow x \cdot x^{p-2} = 1 \Rightarrow x^{-1} = x^{p-2} \text{ in } \mathbb{Z}_p$

another way to compute inverses, but less efficient than Euclid

Application: generating random primes

Suppose we want to generate a large random prime

say, prime p of length 1024 bits (i.e. $p \approx 2^{1024}$)

Step 1: choose a random integer $p \in [2^{1024} , 2^{1025}-1]$

Step 2: test if $2^{p-1} = 1$ in \mathbb{Z}_p

If so, output p and stop. If not, goto step 1 .

Simple algorithm (not the best). $\text{Pr}[p \text{ not prime }] < 2^{-60}$

The structure of $(\mathbb{Z}_p)^*$

Thm (Euler): $(\mathbb{Z}_p)^*$ is a **cyclic group**, that is

$$\exists g \in (\mathbb{Z}_p)^* \text{ such that } \{1, g, g^2, g^3, \dots, g^{p-2}\} = (\mathbb{Z}_p)^*$$

g is called a **generator** of $(\mathbb{Z}_p)^*$

Example: $p=7.$ $\{1, 3, 3^2, 3^3, 3^4, 3^5\} = \{1, 3, 2, 6, 4, 5\} = (\mathbb{Z}_7)^*$

Not every elem. is a generator: $\{1, 2, 2^2, 2^3, 2^4, 2^5\} = \{1, 2, 4\}$

Order

For $g \in (\mathbb{Z}_p)^*$ the set $\{1, g, g^2, g^3, \dots\}$ is called
the **group generated by g** , denoted $\langle g \rangle$

Def: the **order** of $g \in (\mathbb{Z}_p)^*$ is the size of $\langle g \rangle$

$$\text{ord}_p(g) = |\langle g \rangle| = (\text{smallest } a > 0 \text{ s.t. } g^a = 1 \text{ in } \mathbb{Z}_p)$$

Examples: $\text{ord}_7(3) = 6$; $\text{ord}_7(2) = 3$; $\text{ord}_7(1) = 1$

Thm (Lagrange): $\forall g \in (\mathbb{Z}_p)^* : \text{ord}_p(g) \text{ divides } p-1$

Euler's generalization of Fermat (1736)

Def: For an integer N define $\varphi(N) = |(Z_N)^*|$ (Euler's φ func.)

Examples: $\varphi(12) = |\{1,5,7,11\}| = 4$; $\varphi(p) = p-1$

$$\text{For } N=p \cdot q: \quad \varphi(N) = N-p-q+1 = (p-1)(q-1)$$

Thm (Euler): $\forall x \in (Z_N)^*: x^{\varphi(N)} = 1 \text{ in } Z_N$

$$\text{Example: } 5^{\varphi(12)} = 5^4 = 625 = 1 \text{ in } Z_{12}$$

Generalization of Fermat. Basis of the RSA cryptosystem

End of Segment



Intro. Number Theory

Modular e'th roots

Modular e'th roots

We know how to solve modular linear equations:

$$a \cdot x + b = 0 \quad \text{in } Z_N$$

Solution: $x = -b \cdot a^{-1}$ in Z_N

What about higher degree polynomials?

Example: let p be a prime and $c \in Z_p$. Can we solve:

$$x^2 - c = 0 , \quad y^3 - c = 0 , \quad z^{37} - c = 0 \quad \text{in } Z_p$$

Modular e'th roots

Let p be a prime and $c \in \mathbb{Z}_p$.

Def: $x \in \mathbb{Z}_p$ s.t. $x^e = c$ in \mathbb{Z}_p is called an e'th root of c .

Examples: $7^{1/3} = 6$ in \mathbb{Z}_{11} $6^3 = 216 = 7$ in \mathbb{Z}_{11}

$$3^{1/2} = 5 \text{ in } \mathbb{Z}_{11}$$

$2^{1/2}$ does not exist in \mathbb{Z}_{11}

$$1^{1/3} = 1 \text{ in } \mathbb{Z}_{11}$$

The easy case

When does $c^{1/e}$ in \mathbb{Z}_p exist? Can we compute it efficiently?

The easy case: suppose $\gcd(e, p-1) = 1$

Then for all c in $(\mathbb{Z}_p)^*$: $c^{1/e}$ exists in \mathbb{Z}_p and is easy to find.

Proof: let $d = e^{-1}$ in \mathbb{Z}_{p-1} . Then

$$c^{1/e} = c^d \text{ in } \mathbb{Z}_p$$

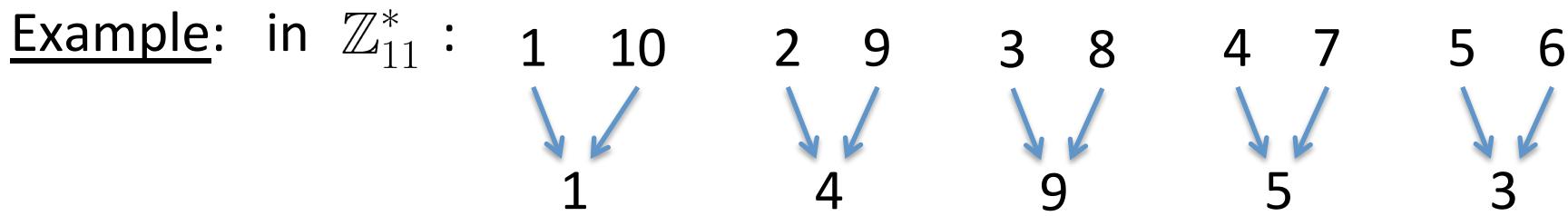
$$\begin{aligned} d \cdot e = 1 \text{ in } \mathbb{Z}_{p-1} &\Rightarrow \exists k \in \mathbb{Z}: d \cdot e = k \cdot (p-1) + 1 \Rightarrow \\ &\Rightarrow (c^d)^e = c^{d \cdot e} = c^{k \cdot (p-1) + 1} = [c^{p-1}]^k \cdot c = c \in \mathbb{Z}_p \end{aligned}$$

The case $e=2$: square roots

If p is an odd prime then $\gcd(2, p-1) \neq 1$

$$\begin{array}{ccc} x & -x \\ \downarrow & \downarrow \\ x^2 \end{array}$$

Fact: in \mathbb{Z}_p^* , $x \rightarrow x^2$ is a 2-to-1 function



Def: x in \mathbb{Z}_p is a **quadratic residue** (Q.R.) if it has a square root in \mathbb{Z}_p

p odd prime \Rightarrow the # of Q.R. in \mathbb{Z}_p is $(p-1)/2 + 1$

Euler's theorem

Thm: $x \in (\mathbb{Z}_p)^*$ is a Q.R. $\iff x^{(p-1)/2} = 1$ in \mathbb{Z}_p (p odd prime)

Example:

$$\begin{array}{l} \text{in } \mathbb{Z}_{11} : 1^5, 2^5, 3^5, 4^5, 5^5, 6^5, 7^5, 8^5, 9^5, 10^5 \\ = 1 \quad -1 \quad 1 \quad 1 \quad 1, \quad -1, \quad -1, \quad -1, \quad 1, \quad -1 \end{array}$$

Note: $x \neq 0 \Rightarrow x^{(p-1)/2} = (x^{p-1})^{1/2} = 1^{1/2} \in \{1, -1\}$ in \mathbb{Z}_p

Def: $x^{(p-1)/2}$ is called the Legendre Symbol of x over p (1798)

Computing square roots mod p

Suppose $p \equiv 3 \pmod{4}$

Lemma: if $c \in (\mathbb{Z}_p)^*$ is Q.R. then $\sqrt{c} = c^{(p+1)/4}$ in \mathbb{Z}_p

Proof:
$$\left[c^{\frac{p+1}{4}}\right]^2 = c^{\frac{p+1}{2}} = \underbrace{c^{\frac{p-1}{2}}}_{=1} \cdot c = c \quad \text{in } \mathbb{Z}_p$$

When $p \equiv 1 \pmod{4}$, can also be done efficiently, but a bit harder

run time $\approx O(\log^3 p)$

Solving quadratic equations mod p

Solve: $a \cdot x^2 + b \cdot x + c = 0$ in \mathbb{Z}_p

Solution: $x = \frac{(-b \pm \sqrt{b^2 - 4 \cdot a \cdot c})}{2a}$ in \mathbb{Z}_p

- Find $(2a)^{-1}$ in \mathbb{Z}_p using extended Euclid.
- Find square root of $b^2 - 4 \cdot a \cdot c$ in \mathbb{Z}_p (if one exists)
using a square root algorithm

Computing e'th roots mod N ??

Let N be a composite number and $e > 1$

When does $c^{1/e}$ in \mathbb{Z}_N exist? Can we compute it efficiently?

Answering these questions requires the factorization of N
(as far as we know)

End of Segment

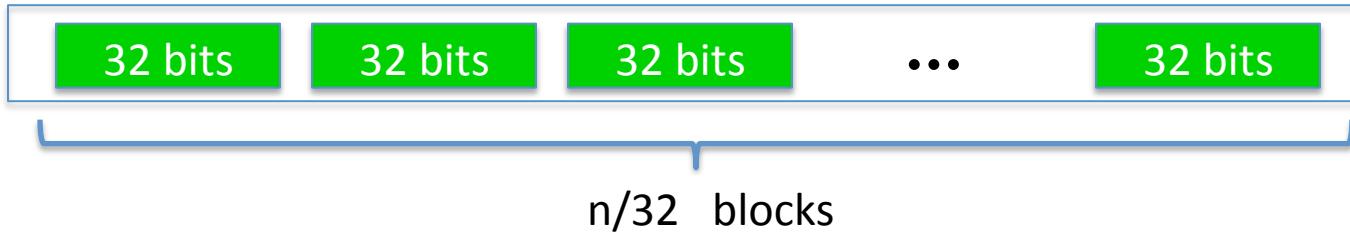


Intro. Number Theory

Arithmetic algorithms

Representing bignums

Representing an n-bit integer (e.g. n=2048) on a 64-bit machine



Note: some processors have 128-bit registers (or more)
and support multiplication on them

Arithmetic

Given: two n-bit integers

- **Addition and subtraction:** linear time $O(n)$

- **Multiplication:** naively $O(n^2)$. Karatsuba (1960): $O(n^{1.585})$

Basic idea: $(2^b x_2 + x_1) \times (2^b y_2 + y_1)$ with 3 mults.

Best (asymptotic) algorithm: about $O(n \cdot \log n)$.

- **Division with remainder:** $O(n^2)$.

$\log_2 3$

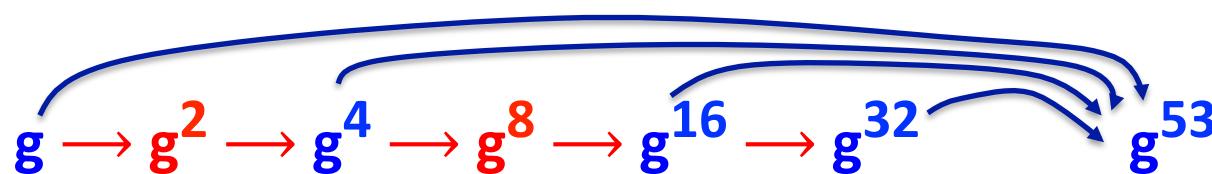
Exponentiation

Finite cyclic group G (for example $G = \mathbb{Z}_p^*$)

Goal: given g in G and x compute g^x

Example: suppose $x = 53 = (110101)_2 = 32+16+4+1$

Then: $g^{53} = g^{32+16+4+1} = g^{32} \cdot g^{16} \cdot g^4 \cdot g^1$



The repeated squaring alg.

Input: g in G and $x > 0$; **Output:** g^x

write $x = (x_n x_{n-1} \dots x_2 x_1 x_0)_2$

```
y ← g , z ← 1
```

```
for i = 0 to n do:
```

```
    if (x[i] == 1): z ← z · y
```

```
    y ← y2
```

```
output z
```

example: g^{53}

y	z
g^2	g
g^4	g
g^8	g^5
g^{16}	g^5
g^{32}	g^{21}
g^{64}	g^{53}

Running times

Given n-bit int. N:

- **Addition and subtraction in Z_N :** linear time $T_+ = O(n)$
- **Modular multiplication in Z_N :** naively $T_x = O(n^2)$
- **Modular exponentiation in Z_N (g^x):**

$$O((\log x) \cdot T_x) \leq O((\log x) \cdot n^2) \leq O(n^3)$$

End of Segment



Intro. Number Theory

Intractable problems

Easy problems

- Given composite N and x in \mathbb{Z}_N find x^{-1} in \mathbb{Z}_N
- Given prime p and polynomial $f(x)$ in $\mathbb{Z}_p[x]$
find x in \mathbb{Z}_p s.t. $f(x) = 0$ in \mathbb{Z}_p (if one exists)

Running time is linear in $\deg(f)$.

... but many problems are difficult

Intractable problems with primes

Fix a prime $p > 2$ and g in $(\mathbb{Z}_p)^*$ of order q .

Consider the function: $x \mapsto g^x$ in \mathbb{Z}_p

Now, consider the inverse function:

$\text{Dlog}_g(g^x) = x$ where x in $\{0, \dots, q-2\}$

Example: in \mathbb{Z}_{11} : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

$\text{Dlog}_2(\cdot)$: 0, 1, 8, 2, 4, 9, 7, 3, 6, 5

DLOG: more generally

Let \mathbf{G} be a finite cyclic group and \mathbf{g} a generator of \mathbf{G}

$$G = \{ 1, g, g^2, g^3, \dots, g^{q-1} \} \quad (q \text{ is called the order of } G)$$

Def: We say that **DLOG is hard in G** if for all efficient alg. A :

$$\Pr_{g \leftarrow G, x \leftarrow \mathbb{Z}_q} [A(G, q, g, g^x) = x] < \text{negligible}$$

Example candidates:

- (1) $(\mathbb{Z}_p)^*$ for large p ,
- (2) Elliptic curve groups mod p

Computing Dlog in $(\mathbb{Z}_p)^*$ (n-bit prime p)

Best known algorithm (GNFS): run time $\exp(\tilde{O}(\sqrt[3]{n}))$

<u>cipher key size</u>	<u>modulus size</u>	<u>Elliptic Curve group size</u>
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	<u>15360</u> bits	512 bits

As a result: slow transition away from $(\text{mod } p)$ to elliptic curves

An application: collision resistance

Choose a group G where Dlog is hard (e.g. $(\mathbb{Z}_p)^*$ for large p)

Let $q = |G|$ be a prime. Choose generators g, h of G

For $x, y \in \{1, \dots, q\}$ define $H(x, y) = g^x \cdot h^y$ in G

Lemma: finding collision for $H(.,.)$ is as hard as computing $\text{Dlog}_g(h)$

Proof: Suppose we are given a collision $H(x_0, y_0) = H(x_1, y_1)$

then $g^{x_0} \cdot h^{y_0} = g^{x_1} \cdot h^{y_1} \Rightarrow g^{x_0 - x_1} = h^{y_1 - y_0} \Rightarrow h = g^{x_0 - x_1 / y_1 - y_0}$

Intractable problems with composites

Consider the set of integers: (e.g. for n=1024)

$$\mathbb{Z}_{(2)}(n) := \{ N = p \cdot q \text{ where } p, q \text{ are } n\text{-bit primes} \}$$

Problem 1: Factor a random N in $\mathbb{Z}_{(2)}(n)$ (e.g. for n=1024)

Problem 2: Given a polynomial $f(x)$ where $\text{degree}(f) > 1$

and a random N in $\mathbb{Z}_{(2)}(n)$

find x in \mathbb{Z}_N s.t. $f(x) = 0$ in \mathbb{Z}_N

The factoring problem

Gauss (1805): *“The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic.”*

Best known alg. (NFS): run time $\exp(\tilde{O}(\sqrt[3]{n}))$ for n-bit integer

Current world record: **RSA-768** (232 digits)

- Work: two years on hundreds of machines
- Factoring a 1024-bit integer: about 1000 times harder
 ⇒ likely possible this decade

Further reading

- A Computational Introduction to Number Theory and Algebra,
V. Shoup, 2008 (V2), Chapter 1-4, 11, 12

Available at [/shoup.net/ntb/ntb-v2.pdf](http://shoup.net/ntb/ntb-v2.pdf)

End of Segment

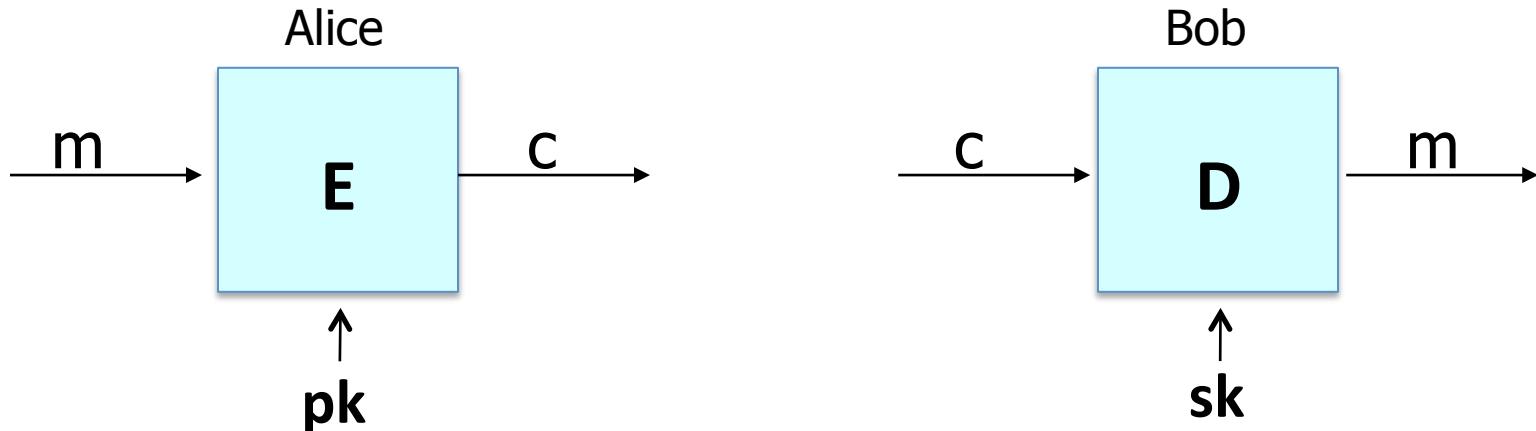


Public Key Encryption from trapdoor permutations

Public key encryption:
definitions and security

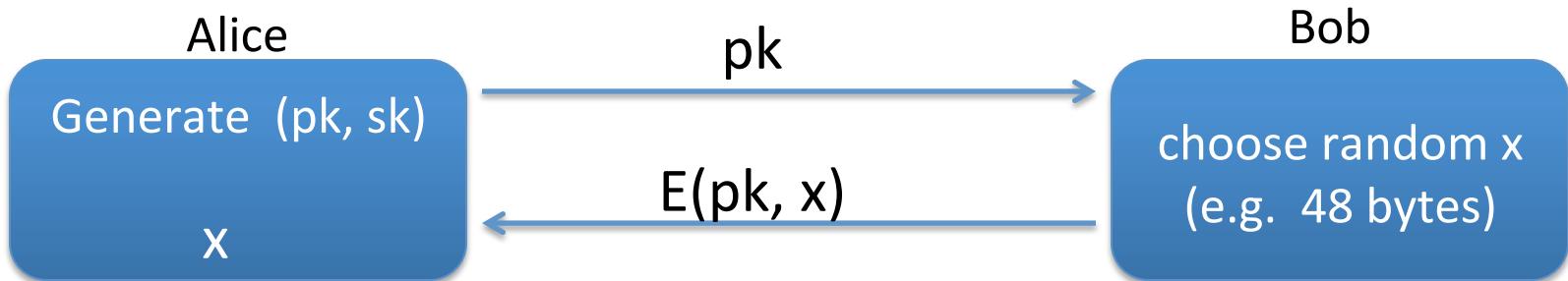
Public key encryption

Bob: generates (PK, SK) and gives PK to Alice



Applications

Session setup (for now, only eavesdropping security)



Non-interactive applications: (e.g. Email)

- Bob sends email to Alice encrypted using pk_{alice}
- Note: Bob needs pk_{alice} (public key management)

Public key encryption

Def: a public-key encryption system is a triple of algs. (G, E, D)

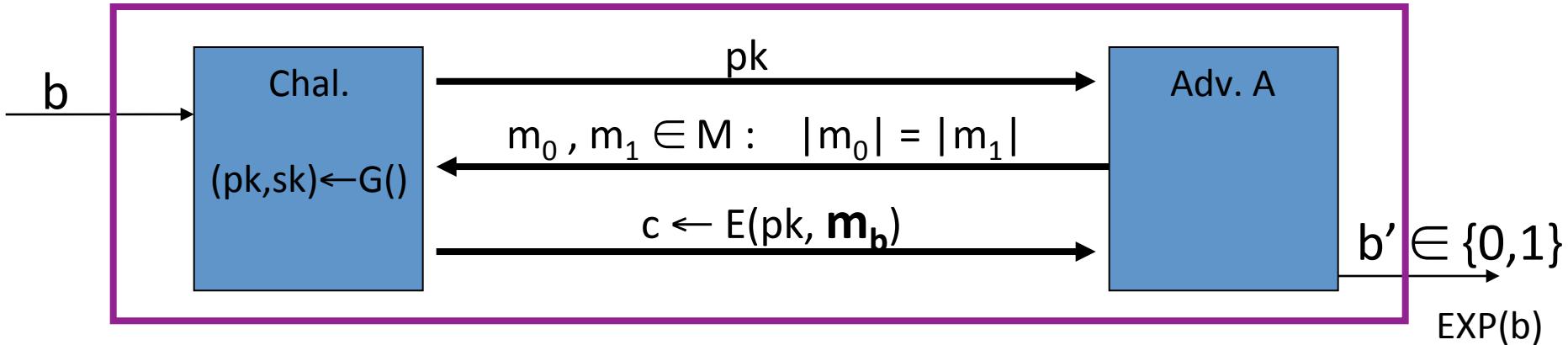
- $G()$: randomized alg. outputs a key pair (pk, sk)
- $E(pk, m)$: randomized alg. that takes $m \in M$ and outputs $c \in C$
- $D(sk, c)$: det. alg. that takes $c \in C$ and outputs $m \in M$ or \perp

Consistency: $\forall (pk, sk) \text{ output by } G :$

$$\forall m \in M : D(sk, E(pk, m)) = m$$

Security: eavesdropping

For $b=0,1$ define experiments $\text{EXP}(0)$ and $\text{EXP}(1)$ as:



Def: $E = (G, E, D)$ is sem. secure (a.k.a IND-CPA) if for all efficient A :

$$\text{Adv}_{\text{SS}}[A, E] = \left| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \right| < \text{negligible}$$

Relation to symmetric cipher security

Recall: for symmetric ciphers we had two security notions:

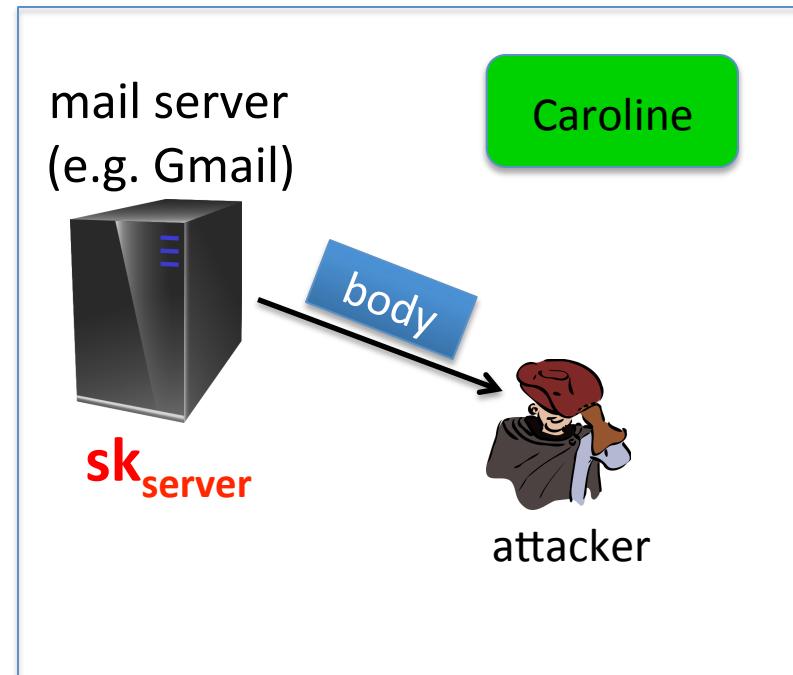
- One-time security and many-time security (CPA)
- We showed that one-time security $\not\Rightarrow$ many-time security

For public key encryption:

- One-time security \Rightarrow many-time security (CPA)
(follows from the fact that attacker can encrypt by himself)
- Public key encryption **must** be randomized

Security against active attacks

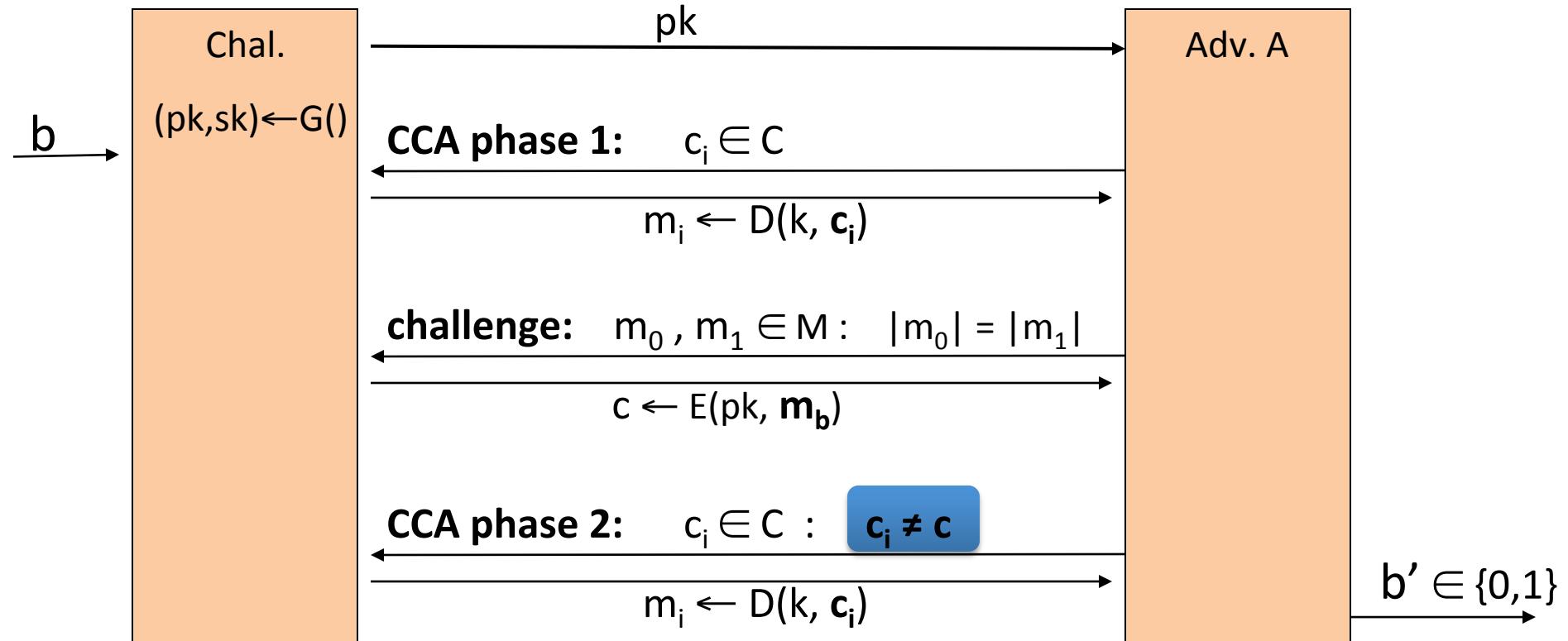
What if attacker can tamper with ciphertext?



Attacker is given decryption of msgs
that start with “**to: attacker**”

(pub-key) Chosen Ciphertext Security: definition

$E = (G, E, D)$ public-key enc. over (M, C) . For $b=0,1$ define $\text{EXP}(b)$:

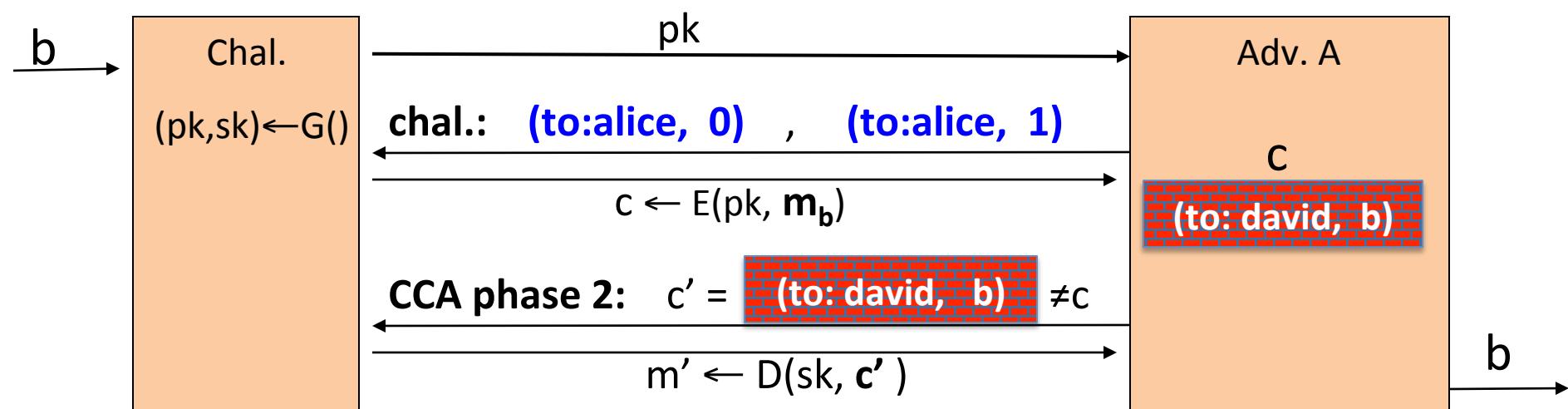


Chosen ciphertext security: definition

Def: E is CCA secure (a.k.a IND-CCA) if for all efficient A :

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

Example: Suppose



Active attacks: symmetric vs. pub-key

Recall: secure symmetric cipher provides **authenticated encryption**
[chosen plaintext security & ciphertext integrity]

- Roughly speaking: **attacker cannot create new ciphertexts**
- Implies security against chosen ciphertext attacks

In public-key settings:

- Attacker **can** create new ciphertexts using pk !!
- So instead: we directly require chosen ciphertext security

This and next module:

constructing CCA secure pub-key systems

End of Segment



Public Key Encryption from trapdoor permutations

Constructions

Goal: construct chosen-ciphertext secure public-key encryption

Trapdoor functions (TDF)

Def: a trapdoor func. $X \rightarrow Y$ is a triple of efficient algs. (G, F, F^{-1})

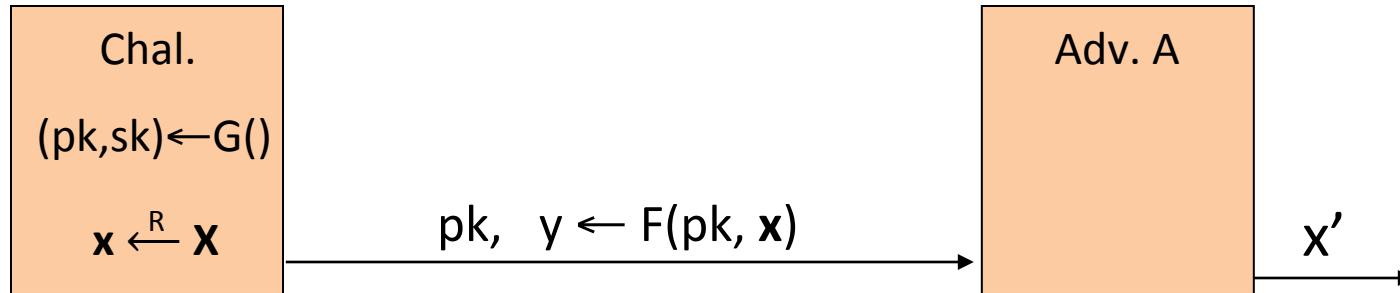
- $G()$: randomized alg. outputs a key pair (pk, sk)
- $F(pk, \cdot)$: det. alg. that defines a function $X \rightarrow Y$
- $F^{-1}(sk, \cdot)$: defines a function $Y \rightarrow X$ that inverts $F(pk, \cdot)$

More precisely: $\forall (pk, sk) \text{ output by } G$

$$\forall x \in X: F^{-1}(sk, F(pk, x)) = x$$

Secure Trapdoor Functions (TDFs)

(G, F, F^{-1}) is secure if $F(pk, \cdot)$ is a “one-way” function:
can be evaluated, but cannot be inverted without sk



Def: (G, F, F^{-1}) is a secure TDF if for all efficient A :

$$\text{Adv}_{\text{OW}}[A, F] = \Pr[\mathbf{x} = \mathbf{x}'] < \text{negligible}$$

Public-key encryption from TDFs

- (G, F, F^{-1}) : secure TDF $X \rightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K, M, C)
- $H: X \rightarrow K$ a hash function

We construct a pub-key enc. system (G, E, D) :

Key generation G : same as G for TDF

Public-key encryption from TDFs

- (G, F, F^{-1}) : secure TDF $X \rightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K, M, C)
- $H: X \rightarrow K$ a hash function

$E(pk, m)$:

$$x \xleftarrow{R} X, \quad y \leftarrow F(pk, x)$$

$$k \leftarrow H(x), \quad c \leftarrow E_s(k, m)$$

output (y, c)

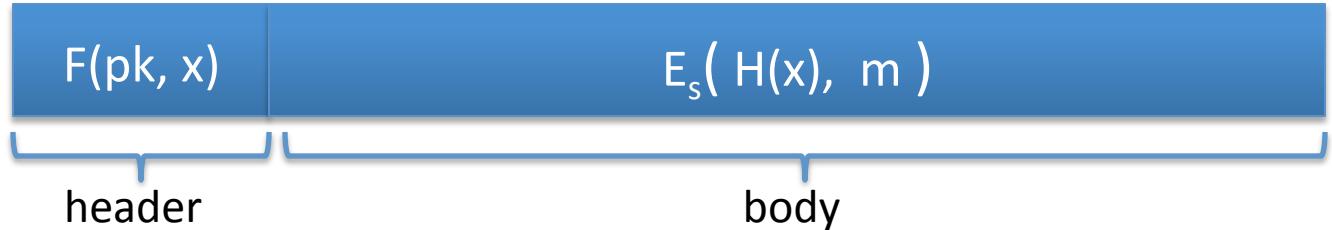
$D(sk, (y, c))$:

$$x \leftarrow F^{-1}(sk, y),$$

$$k \leftarrow H(x), \quad m \leftarrow D_s(k, c)$$

output m

In pictures:



Security Theorem:

If (G, F, F^{-1}) is a secure TDF, (E_s, D_s) provides auth. enc.
and $H: X \rightarrow K$ is a “random oracle”
then (G, E, D) is CCA^{ro} secure.

Incorrect use of a Trapdoor Function (TDF)

Never encrypt by applying F directly to plaintext:

E(pk, m) :

output $c \leftarrow F(pk, m)$

D(sk, c) :

output $F^{-1}(sk, c)$

Problems:

- Deterministic: cannot be semantically secure !!
- Many attacks exist (next segment)

Next step: construct a TDF

End of Segment



Public Key Encryption from trapdoor permutations

The RSA trapdoor
permutation

Review: trapdoor permutations

Three algorithms: (G, F, F^{-1})

- G : outputs pk, sk . pk defines a function $F(pk, \cdot) : X \rightarrow X$
- $F(pk, x)$: evaluates the function at x
- $F^{-1}(sk, y)$: inverts the function at y using sk

Secure trapdoor permutation:

The function $F(pk, \cdot)$ is one-way without the trapdoor sk

Review: arithmetic mod composites

Let $N = p \cdot q$ where p, q are prime

$$Z_N = \{0, 1, 2, \dots, N-1\} ; (Z_N)^* = \{\text{invertible elements in } Z_N\}$$

Facts: $x \in Z_N$ is invertible $\Leftrightarrow \gcd(x, N) = 1$

– Number of elements in $(Z_N)^*$ is $\varphi(N) = (p-1)(q-1) = N-p-q+1$

Euler's thm: $\forall x \in (Z_N)^* : x^{\varphi(N)} = 1$

The RSA trapdoor permutation

First published: Scientific American, Aug. 1977.

Very widely used:

- SSL/TLS: certificates and key-exchange
 - Secure e-mail and file systems
- ... many others

The RSA trapdoor permutation

G(): choose random primes $p, q \approx 1024$ bits. Set $N = pq$.
choose integers e, d s.t. $e \cdot d = 1 \pmod{\varphi(N)}$
output $pk = (N, e)$, $sk = (N, d)$

F(pk, x): $\mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$; $\text{RSA}(x) = x^e$ (in \mathbb{Z}_N)

$$F^{-1}(sk, y) = y^d ; \quad y^d = \text{RSA}(x)^d = x^{ed} = x^{k\varphi(N)+1} = (x^{\varphi(N)})^k \cdot x = x$$

The RSA assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

$$\Pr [A(N,e,y) = y^{1/e}] < \text{negligible}$$

where $p,q \xleftarrow{R} n\text{-bit primes}, N \leftarrow pq, y \xleftarrow{R} \mathbb{Z}_N^*$

Review: RSA pub-key encryption (ISO std)

(E_s, D_s) : symmetric enc. scheme providing auth. encryption.

$H: Z_N \rightarrow K$ where K is key space of (E_s, D_s)

- $G()$: generate RSA params: $pk = (N, e)$, $sk = (N, d)$
- $E(pk, m)$:
 - (1) choose random x in Z_N
 - (2) $y \leftarrow RSA(x) = x^e$, $k \leftarrow H(x)$
 - (3) output $(y, E_s(k, m))$
- $D(sk, (y, c))$: output $D_s(H(RSA^{-1}(y)), c)$

Textbook RSA is insecure

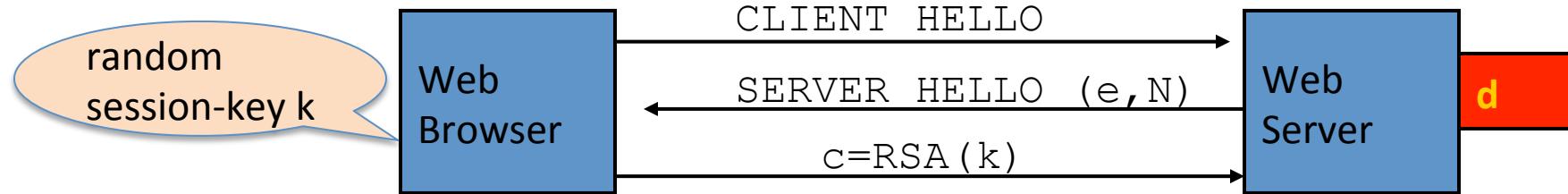
Textbook RSA encryption:

- public key: (N, e) Encrypt: $c \leftarrow m^e \pmod{N}$
- secret key: (N, d) Decrypt: $c^d \rightarrow m$

Insecure cryptosystem !!

- Is not semantically secure and many attacks exist
- ⇒ The RSA trapdoor permutation is not an encryption scheme !

A simple attack on textbook RSA



Suppose k is 64 bits: $k \in \{0, \dots, 2^{64}\}$. Eve sees: $c = k^e$ in \mathbb{Z}_N

If $k = k_1 \cdot k_2$ where $k_1, k_2 < 2^{34}$ (prob. $\approx 20\%$) then $c/k_1^e = k_2^e$ in \mathbb{Z}_N

Step 1: build table: $c/1^e, c/2^e, c/3^e, \dots, c/2^{34e}$. time: 2^{34}

Step 2: for $k_2 = 0, \dots, 2^{34}$ test if k_2^e is in table. time: 2^{34}

Output matching (k_1, k_2) .

Total attack time: $\approx 2^{40} \ll 2^{64}$

End of Segment



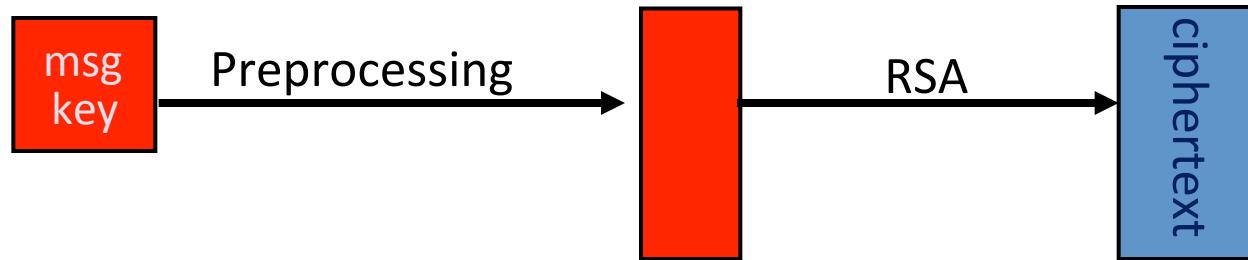
Public Key Encryption from trapdoor permutations

PKCS 1

RSA encryption in practice

Never use textbook RSA.

RSA in practice (since ISO standard is not often used) :

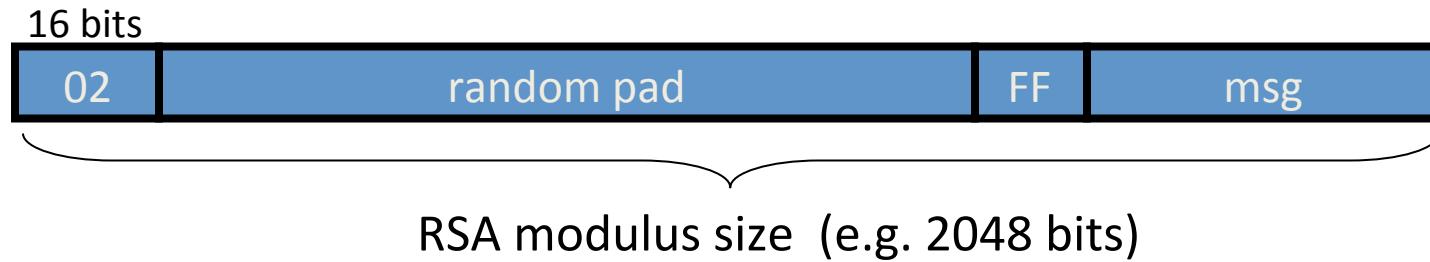


Main questions:

- How should the preprocessing be done?
- Can we argue about security of resulting system?

PKCS1 v1.5

PKCS1 mode 2: (encryption)

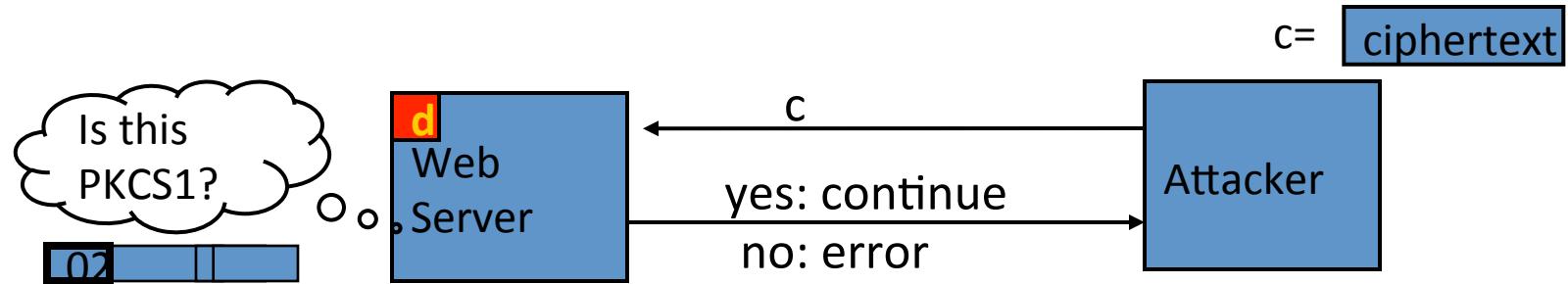


- Resulting value is RSA encrypted
- Widely deployed, e.g. in HTTPS

Attack on PKCS1 v1.5

(Bleichenbacher 1998)

PKCS1 used in HTTPS:

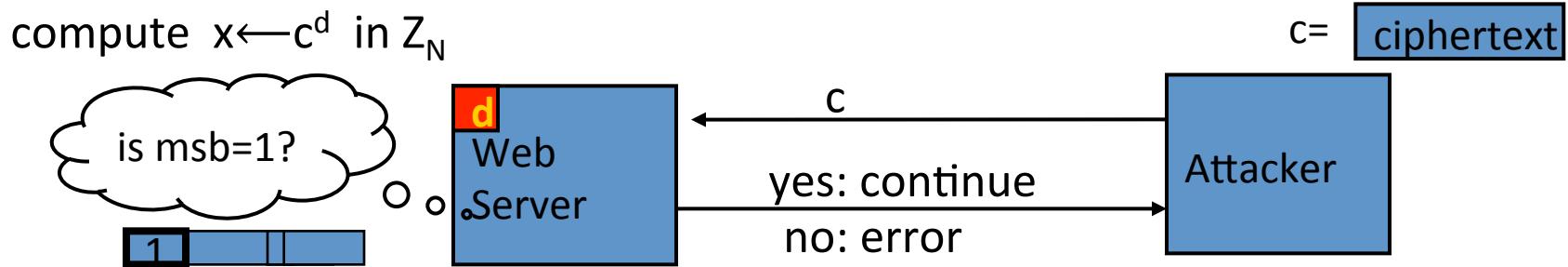


⇒ attacker can test if 16 MSBs of plaintext = '02'

Chosen-ciphertext attack: to decrypt a given ciphertext c do:

- Choose $r \in \mathbb{Z}_N$. Compute $c' \leftarrow r^e \cdot c = (r \cdot \text{PKCS1}(m))^e$
- Send c' to web server and use response

Baby Bleichenbacher



Suppose N is $N = 2^n$ (an invalid RSA modulus). Then:

- Sending c reveals $\text{msb}(x)$
- Sending $2^e \cdot c = (2x)^e$ in Z_N reveals $\text{msb}(2x \bmod N) = \text{msb}_2(x)$
- Sending $4^e \cdot c = (4x)^e$ in Z_N reveals $\text{msb}(4x \bmod N) = \text{msb}_3(x)$
- ... and so on to reveal all of x

HTTPS Defense

(RFC 5246)

Attacks discovered by Bleichenbacher and Klima et al. ... can be avoided by treating incorrectly formatted message blocks ... in a manner indistinguishable from correctly formatted RSA blocks.

In other words:

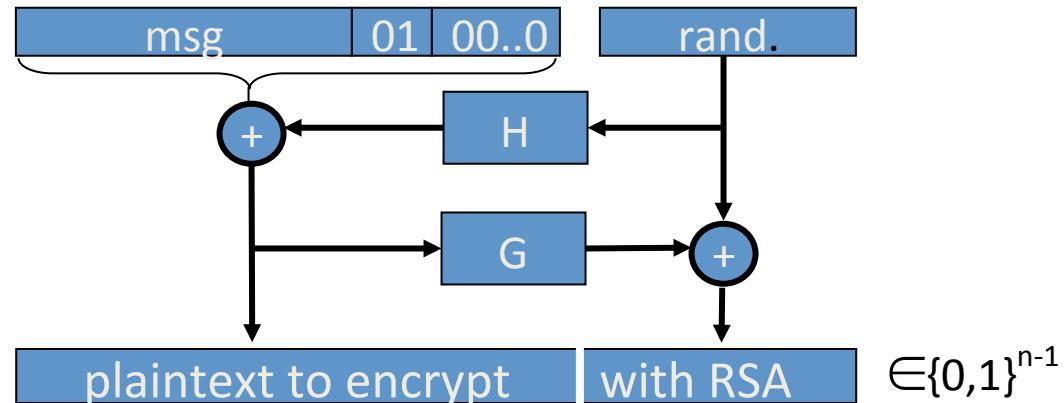
1. *Generate a string R of 46 random bytes*
2. *Decrypt the message to recover the plaintext M*
3. *If the PKCS#1 padding is not correct*

pre_master_secret = R

PKCS1 v2.0: OAEP

New preprocessing function: OAEP [BR94]

check pad
on decryption.
reject CT if invalid.



Thm [FOPS'01] : RSA is a trap-door permutation \Rightarrow
RSA-OAEP is CCA secure when H, G are *random oracles*

in practice: use SHA-256 for H and G

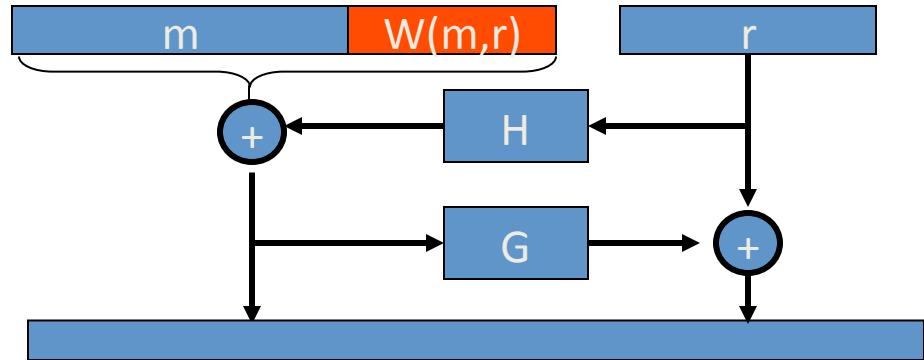
OAEP Improvements

OAEP+: [Shoup'01]

\forall trap-door permutation F

F-OAEP+ is CCA secure when
H,G,W are *random oracles*.

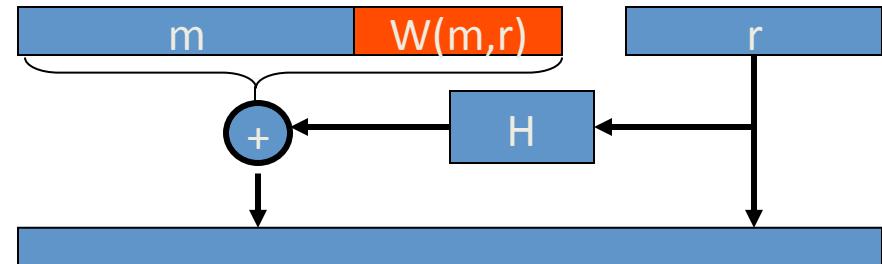
During decryption validate W(m,r) field.



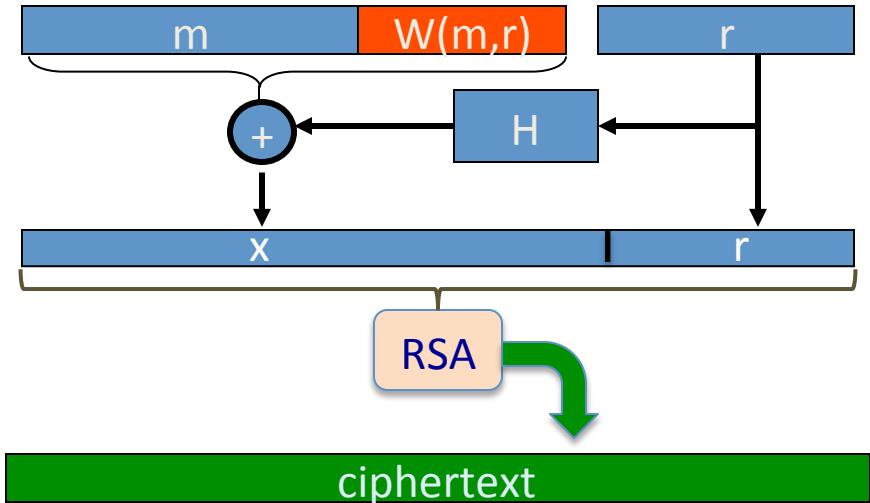
SAEP+: [B'01]

RSA ($e=3$) is a trap-door perm \Rightarrow

RSA-SAEP+ is CCA secure when
H,W are *random oracle*.



How would you decrypt
an SAEP ciphertext **ct** ?



-
- $(x,r) \leftarrow \text{RSA}^{-1}(\text{sk},\text{ct}) , (m,w) \leftarrow x \oplus H(r) , \text{ output } m \text{ if } w = W(m,r)$
 - $(x,r) \leftarrow \text{RSA}^{-1}(\text{sk},\text{ct}) , (m,w) \leftarrow r \oplus H(x) , \text{ output } m \text{ if } w = W(m,r)$
 - $(x,r) \leftarrow \text{RSA}^{-1}(\text{sk},\text{ct}) , (m,w) \leftarrow x \oplus H(r) , \text{ output } m \text{ if } r = W(m,x)$

Subtleties in implementing OAEP

[M '00]

```
OAEP-decrypt(ct):
```

```
    error = 0;
```

```
.....
```

```
if ( RSA-1(ct) > 2n-1 )
```

```
    { error = 1; goto exit; }
```

```
.....
```

```
if ( pad(OAEP-1(RSA-1(ct))) != "01000" )
```

```
    { error = 1; goto exit; }
```

Problem: timing information leaks type of error

⇒ Attacker can decrypt any ciphertext

Lesson: Don't implement RSA-OAEP yourself !

End of Segment



Public Key Encryption from trapdoor permutations

Is RSA a one-way
function?

Is RSA a one-way permutation?

To invert the RSA one-way func. (without d) attacker must compute:

$$x \text{ from } c = x^e \pmod{N}.$$

How hard is computing e 'th roots modulo N ??

Best known algorithm:

- Step 1: factor N (hard)
- Step 2: compute e 'th roots modulo p and q (easy)

Shortcuts?

Must one factor N in order to compute e'th roots?

To prove no shortcut exists show a reduction:

- Efficient algorithm for e'th roots mod N
 \Rightarrow efficient algorithm for factoring N.
- Oldest problem in public key cryptography.

Some evidence no reduction exists: (BV'98)

- “Algebraic” reduction \Rightarrow factoring is easy.

How **not** to improve RSA's performance

To speed up RSA decryption use small private key d ($d \approx 2^{128}$)

$$c^d = m \pmod{N}$$

Wiener'87: if $d < N^{0.25}$ then RSA is insecure.

BD'98: if $d < N^{0.292}$ then RSA is insecure (open: $d < N^{0.5}$)

Insecure: priv. key d can be found from (N, e)

Wiener's attack

Recall: $e \cdot d = 1 \pmod{\varphi(N)}$ $\Rightarrow \exists k \in \mathbb{Z} : e \cdot d = k \cdot \varphi(N) + 1$

$$\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| = \frac{1}{d \cdot \varphi(N)} \leq \frac{1}{\sqrt{N}}$$

$$\varphi(N) = N - p - q + 1 \Rightarrow |N - \varphi(N)| \leq p + q \leq 3\sqrt{N} \leq \gamma\sqrt{N}$$

$$d \leq N^{0.25}/3 \Rightarrow \left| \frac{e}{N} - \frac{k}{d} \right| \leq \underbrace{\left| \frac{e}{N} - \frac{e}{\varphi(N)} \right|}_{\leq \frac{3\sqrt{N}}{N} \cdot \frac{e}{\varphi(N)}} + \underbrace{\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right|}_{\leq \frac{1}{\sqrt{N}}} \leq \frac{1}{2d^2}$$

$$\frac{1}{2d^2} - \frac{1}{\sqrt{N}} \geq \frac{3}{\sqrt{N}}$$

→ Continued fraction expansion of e/N gives k/d .

$e \cdot d = 1 \pmod{k} \Rightarrow \gcd(d, k) = 1 \Rightarrow$ can find d from k/d

End of Segment



Public Key Encryption from trapdoor permutations

RSA in practice

RSA With Low public exponent

To speed up RSA encryption use a small e : $c = m^e \pmod{N}$

- Minimum value: $e=3$ $(\gcd(e, \varphi(N)) = 1)$
- Recommended value: $e=65537=2^{16}+1$

Encryption: 17 multiplications

Asymmetry of RSA: fast enc. / slow dec.

- ElGamal (next module): approx. same time for both.

Key lengths

Security of public key system should be comparable to security of symmetric cipher:

Cipher key-size

80 bits

128 bits

256 bits (AES)

RSA

Modulus size

1024 bits

3072 bits

15360 bits

Implementation attacks

Timing attack: [Kocher et al. 1997] , [BB'04]

The time it takes to compute $c^d \pmod{N}$ can expose d

Power attack: [Kocher et al. 1999)

The power consumption of a smartcard while it is computing $c^d \pmod{N}$ can expose d .

Faults attack: [BDL'97]

A computer error during $c^d \pmod{N}$ can expose d .

A common defense: check output. 10% slowdown.

An Example Fault Attack on RSA (CRT)

A common implementation of RSA decryption: $x = c^d$ in Z_N

decrypt mod p: $x_p = c^d$ in Z_p
decrypt mod q: $x_q = c^d$ in Z_q

} combine to get $x = c^d$ in Z_N

Suppose error occurs when computing x_q , but no error in x_p

Then: output is x' where $x' = c^d$ in Z_p but $x' \neq c^d$ in Z_q

$\Rightarrow (x')^e = c$ in Z_p but $(x')^e \neq c$ in $Z_q \Rightarrow \gcd((x')^e - c, N) = p$

RSA Key Generation Trouble [Heninger et al./Lenstra et al.]

OpenSSL RSA key generation (abstract):

```
prng.seed(seed)
p = prng.generate_random_prime()
prng.add_randomness(bits)
q = prng.generate_random_prime()
N = p*q
```

Suppose poor entropy at startup:

- Same p will be generated by multiple devices, but different q
- N_1, N_2 : RSA keys from different devices $\Rightarrow \gcd(N_1, N_2) = p$

RSA Key Generation Trouble [Heninger et al./Lenstra et al.]

Experiment: factors 0.4% of public HTTPS keys !!

Lesson:

- Make sure random number generator is properly seeded when generating keys

Further reading

- Why chosen ciphertext security matters, V. Shoup, 1998
- Twenty years of attacks on the RSA cryptosystem,
D. Boneh, Notices of the AMS, 1999
- OAEP reconsidered, V. Shoup, Crypto 2001
- Key lengths, A. Lenstra, 2004

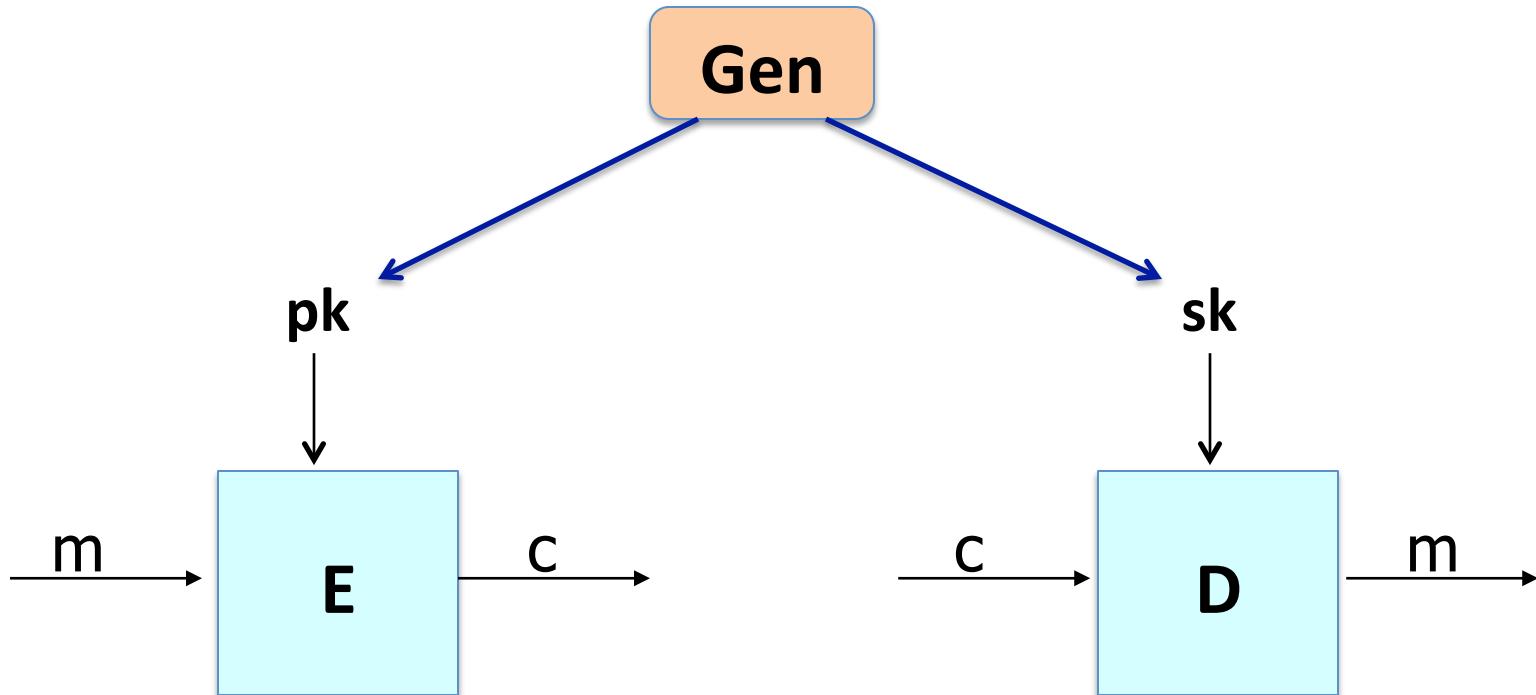
End of Segment



Public key encryption from Diffie-Hellman

The ElGamal Public-key System

Recap: public key encryption: (Gen, E, D)

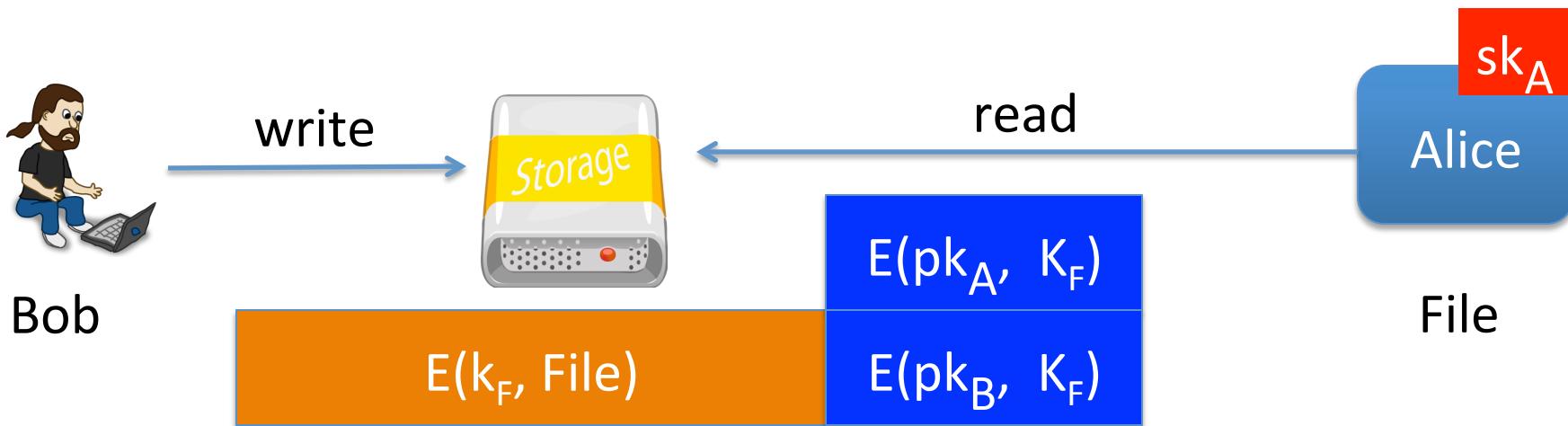


Recap: public-key encryption applications

Key exchange (e.g. in HTTPS)

Encryption in non-interactive settings:

- Secure Email: Bob has Alice's pub-key and sends her an email
- Encrypted File Systems

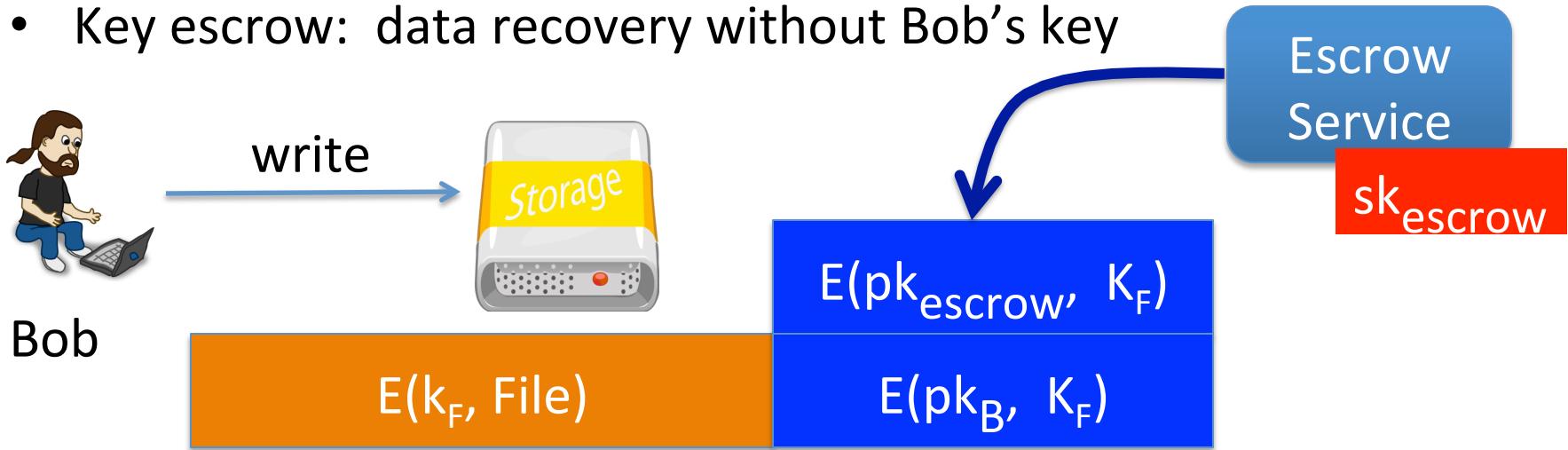


Recap: public-key encryption applications

Key exchange (e.g. in HTTPS)

Encryption in non-interactive settings:

- Secure Email: Bob has Alice's pub-key and sends her an email
- Encrypted File Systems
- Key escrow: data recovery without Bob's key



Constructions

This week: two families of public-key encryption schemes

- Previous lecture: based on trapdoor functions (such as RSA)
 - Schemes: ISO standard, OAEP+, ...
- This lecture: based on the Diffie-Hellman protocol
 - Schemes: ElGamal encryption and variants (e.g. used in GPG)

Security goals: chosen ciphertext security

Review: the Diffie-Hellman protocol (1977)

Fix a finite cyclic group G (e.g. $G = (\mathbb{Z}_p)^*$) of order n

Fix a generator g in G (i.e. $G = \{1, g, g^2, g^3, \dots, g^{n-1}\}$)

Alice

choose random a in $\{1, \dots, n\}$

$$A = g^a$$

Bob

choose random b in $\{1, \dots, n\}$

$$B = g^b$$

$$B^a = (g^b)^a = \boxed{k_{AB} = g^{ab}} = (g^a)^b = A^b$$

ElGamal: converting to pub-key enc. (1984)

Fix a finite cyclic group G (e.g. $G = (\mathbb{Z}_p)^*$) of order n

Fix a generator g in G (i.e. $G = \{1, g, g^2, g^3, \dots, g^{n-1}\}$)

Alice

choose random a in $\{1, \dots, n\}$

$$A = g^a$$

Treat as a
public key

Bob

choose random b in $\{1, \dots, n\}$

$$\text{compute } g^{ab} = A^b,$$

derive symmetric key k ,

$ct = [B = g^b, \text{ encrypt message } m \text{ with } k]$

ElGamal: converting to pub-key enc. (1984)

Fix a finite cyclic group G (e.g. $G = (\mathbb{Z}_p)^*$) of order n

Fix a generator g in G (i.e. $G = \{1, g, g^2, g^3, \dots, g^{n-1}\}$)

Alice

choose random a in $\{1, \dots, n\}$

$$A = g^a$$

Treat as a
public key

Bob

choose random b in $\{1, \dots, n\}$

compute $g^{ab} = A^b$,
derive symmetric key k ,
encrypt message m with k

To decrypt:

compute $g^{ab} = B^a$,

derive k , and decrypt

$$ct = [$$

$$B = g^b,$$

encrypt message m with k]

The ElGamal system (a modern view)

- G : finite cyclic group of order n
- (E_s, D_s) : symmetric auth. encryption defined over (K, M, C)
- $H: G^2 \rightarrow K$ a hash function

We construct a pub-key enc. system (Gen, E, D) :

- Key generation Gen :
 - choose random generator g in G and random a in Z_n
 - output $\text{sk} = a$, $\text{pk} = (g, h=g^a)$

The ElGamal system (a modern view)

- G : finite cyclic group of order n
- (E_s, D_s) : symmetric auth. encryption defined over (K, M, C)
- $H: G^2 \rightarrow K$ a hash function

$E(pk=(g,h), m) :$

$$b \xleftarrow{R} Z_n, u \leftarrow g^b, v \leftarrow h^b$$

$$k \leftarrow H(u, v), c \leftarrow E_s(k, m)$$

output (u, c)

$D(sk=a, (u,c)) :$

$$v \leftarrow u^a$$

$$k \leftarrow H(u, v), m \leftarrow D_s(k, c)$$

output m

ElGamal performance

E(pk=(g,h), m) :

$$b \leftarrow Z_n, u \leftarrow g^b, v \leftarrow h^b$$

D(sk=a, (u,c)) :

$$v \leftarrow u^a$$

Encryption: 2 exp. (fixed basis)

- Can pre-compute $[g^{(2^i)}, h^{(2^i)} \text{ for } i=1, \dots, \log_2 n]$
- 3x speed-up (or more)

Decryption: 1 exp. (variable basis)

Next step: why is this system chosen ciphertext secure?
under what assumptions?

End of Segment



Public key encryption from Diffie-Hellman

ElGamal Security

Computational Diffie-Hellman Assumption

G : finite cyclic group of order n

Comp. DH (CDH) assumption holds in G if: $g, g^a, g^b \not\Rightarrow g^{ab}$

for all efficient algs. A :

$$\Pr[A(g, g^a, g^b) = g^{ab}] < \text{negligible}$$

where $g \leftarrow \{\text{generators of } G\}, a, b \leftarrow \mathbb{Z}_n$

Hash Diffie-Hellman Assumption

G : finite cyclic group of order n , $H: G^2 \rightarrow K$ a hash function

Def: Hash-DH (HDH) assumption holds for (G, H) if:

$$(g, g^a, g^b, H(g^b, g^{ab})) \approx_p (g, g^a, g^b, R)$$

where $g \leftarrow \{\text{generators of } G\}$, $a, b \leftarrow \mathbb{Z}_n$, $R \leftarrow K$

H acts as an extractor: strange distribution on $G^2 \Rightarrow$ uniform on K

Suppose $K = \{0,1\}^{128}$ and

$H: G^2 \rightarrow K$ only outputs strings in K that begin with 0
(i.e. for all $x,y: \text{msb}(H(x,y))=0$)

Can Hash-DH hold for (G, H) ?

- Yes, for some groups G
-  No, Hash-DH is easy to break in this case
- Yes, Hash-DH is always true for such H

ElGamal is sem. secure under Hash-DH

KeyGen: $g \leftarrow \{\text{generators of } G\}$, $a \leftarrow \mathbb{Z}_n$

output $\text{pk} = (g, h=g^a)$, $\text{sk} = a$

E(pk=(g,h), m) : $b \leftarrow \mathbb{Z}_n$

$k \leftarrow H(g^b, h^b)$, $c \leftarrow E_s(k, m)$

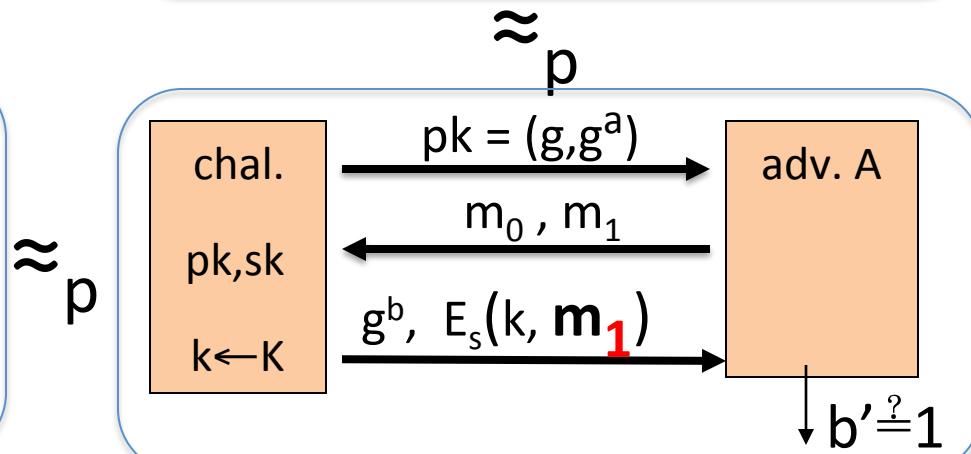
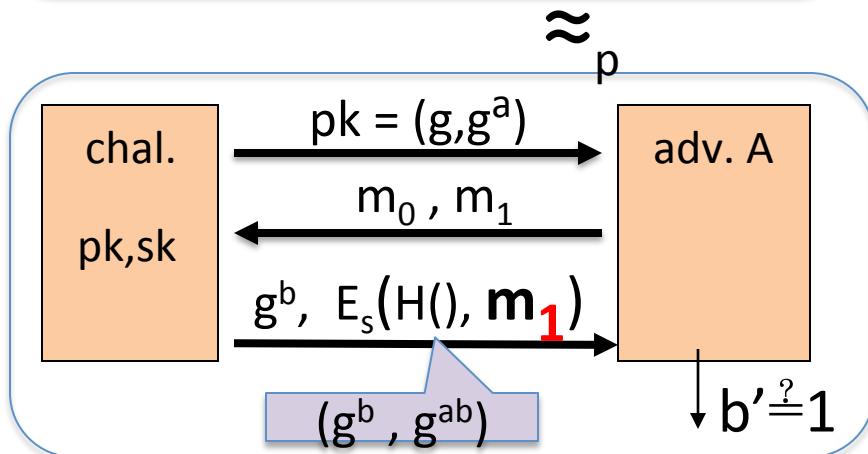
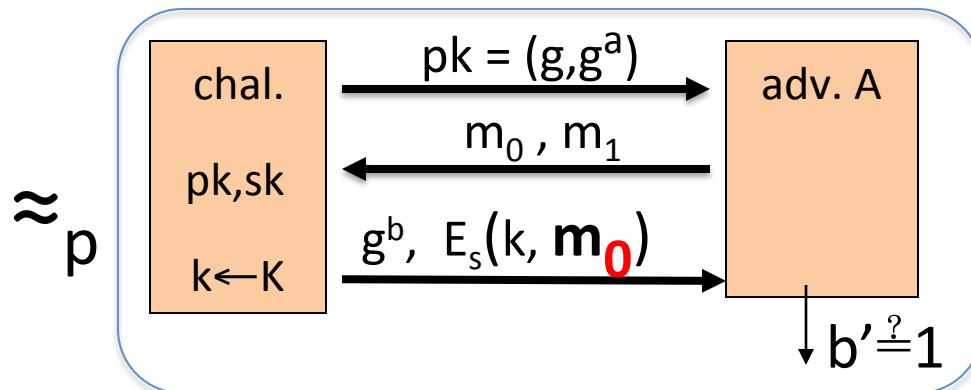
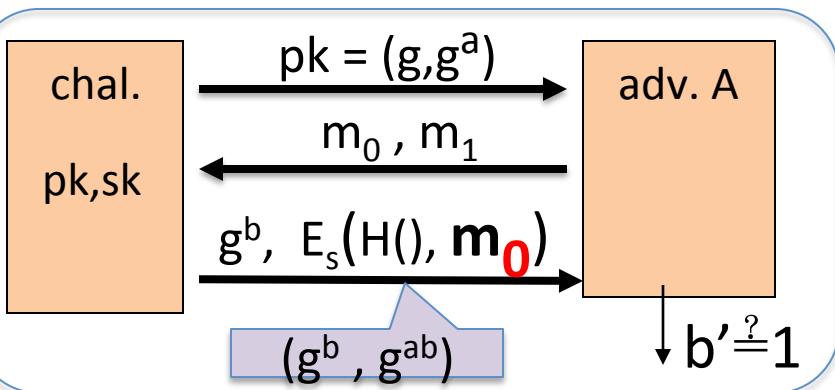
output (g^b, c)

D(sk=a, (u,c)) :

$k \leftarrow H(u, u^a)$, $m \leftarrow D_s(k, c)$

output m

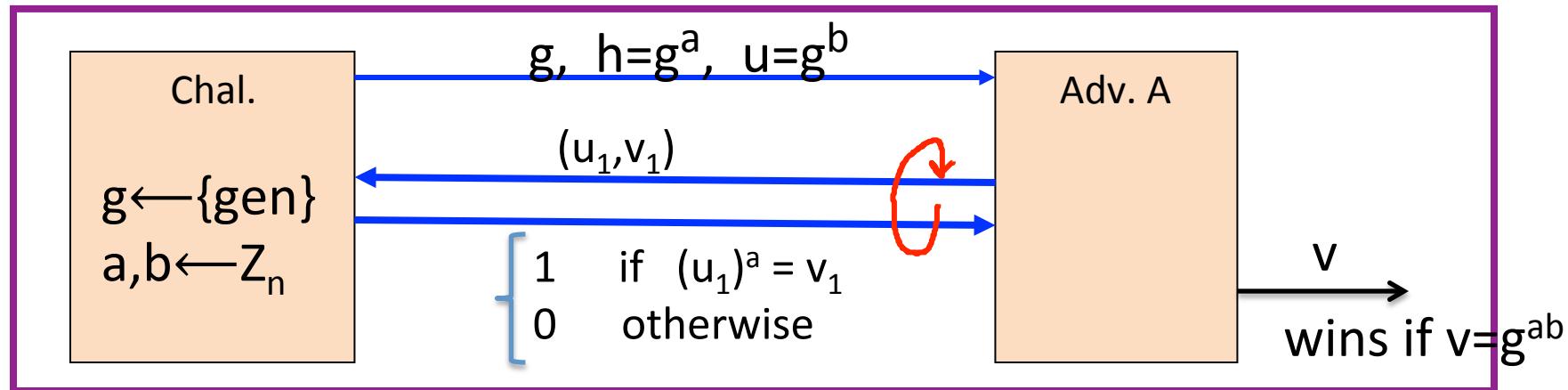
ElGamal is sem. secure under Hash-DH



ElGamal chosen ciphertext security?

To prove chosen ciphertext security need stronger assumption

Interactive Diffie-Hellman (IDH) in group G:



IDH holds in G if: $\forall \text{efficient A: } \Pr[\text{A outputs } g^{ab}] < \text{negligible}$

ElGamal chosen ciphertext security?

Security Theorem:

If IDH holds in the group G , (E_s, D_s) provides auth. enc.
and $H: G^2 \rightarrow K$ is a “random oracle”
then ElGamal is CCA^{ro} secure.

- Questions:
- (1) can we prove CCA security based on CDH?
 - (2) can we prove CCA security without random oracles?

End of Segment



Public key encryption from Diffie-Hellman

ElGamal Variants With Better Security

Review: ElGamal encryption

KeyGen: $g \leftarrow \{\text{generators of } G\}$, $a \leftarrow \mathbb{Z}_n$

output $\text{pk} = (g, h=g^a)$, $\text{sk} = a$

E(pk=(g,h), m) : $b \leftarrow \mathbb{Z}_n$

$k \leftarrow H(g^b, h^b)$, $c \leftarrow E_s(k, m)$

output (g^b, c)

D(sk=a, (u,c)) :

$k \leftarrow H(u, u^a)$, $m \leftarrow D_s(k, c)$

output m

ElGamal chosen ciphertext security

Security Theorem:

If **IDH** holds in the group G , (E_s, D_s) provides auth. enc.
and $H: G^2 \rightarrow K$ is a “random oracle”
then **ElGamal** is CCA^{ro} secure.

Can we prove CCA security based on CDH $(g, g^a, g^b \rightarrow g^{ab})$?

- Option 1: use group G where $CDH = IDH$ (a.k.a bilinear group)
- Option 2: change the ElGamal system

Variants: twin ElGamal [CKS'08]

KeyGen: $g \leftarrow \{\text{generators of } G\}$, $a1, a2 \leftarrow \mathbb{Z}_n$

output $\text{pk} = (g, h_1=g^{a1}, h_2=g^{a2})$, $\text{sk} = (a1, a2)$

E(pk=(g,h₁,h₂), m) : $b \leftarrow \mathbb{Z}_n$

$$k \leftarrow H(g^b, h_1^b, h_2^b)$$
$$c \leftarrow E_s(k, m)$$

output (g^b, c)

D(sk=(a1,a2), (u,c)) :

$$k \leftarrow H(u, u^{a1}, u^{a2})$$
$$m \leftarrow D_s(k, c)$$

output m

Chosen ciphertext security

Security Theorem:

If **CDH** holds in the group G , (E_s, D_s) provides auth. enc.
and $H: G^3 \rightarrow K$ is a “random oracle”
then **twin ElGamal** is CCA^{ro} secure.

Cost: one more exponentiation during enc/dec

- Is it worth it? No one knows ...

ElGamal security w/o random oracles?

Can we prove CCA security without random oracles?

- Option 1: use Hash-DH assumption in “bilinear groups”
 - Special elliptic curve with more structure [CHK’04 + BB’04]
- Option 2: use Decision-DH assumption in any group [CS’98]

Further Reading

- The Decision Diffie-Hellman problem. D. Boneh, ANTS 3, 1998
- Universal hash proofs and a paradigm for chosen ciphertext secure public key encryption. R. Cramer and V. Shoup, Eurocrypt 2002
- Chosen-ciphertext security from Identity-Based Encryption.
D. Boneh, R. Canetti, S. Halevi, and J. Katz, SICOMP 2007
- The Twin Diffie-Hellman problem and applications.
D. Cash, E. Kiltz, V. Shoup, Eurocrypt 2008
- Efficient chosen-ciphertext security via extractable hash proofs.
H. Wee, Crypto 2010



Public key encryption from Diffie-Hellman

A Unifying Theme

One-way functions (informal)

A function $f: X \rightarrow Y$ is one-way if

- There is an efficient algorithm to evaluate $f(\cdot)$, but
- Inverting f is hard:

for all efficient A and $x \leftarrow X$:

$$\Pr[\textcolor{red}{f(A(f(x)))} = \textcolor{red}{f(x)}] < \text{negligible}$$

Functions that are not one-way: $f(x) = x$, $f(x) = 0$

Ex. 1: generic one-way functions

Let $f: X \rightarrow Y$ be a secure PRG (where $|Y| \gg |X|$)

(e.g. f built using det. counter mode)

Lemma: f a secure PRG \Rightarrow f is one-way

Proof sketch:

A inverts $f \Rightarrow B(y) = \begin{cases} 0 & \text{if } f(A(y)) = y \\ 1 & \text{otherwise} \end{cases}$ is a distinguisher

Generic: no special properties. Difficult to use for key exchange.

Ex 2: The DLOG one-way function

Fix a finite cyclic group G (e.g. $G = (\mathbb{Z}_p)^*$) of order n

g : a random generator in G (i.e. $G = \{1, g, g^2, g^3, \dots, g^{n-1}\}$)

Define: $f: \mathbb{Z}_n \rightarrow G$ as $f(x) = g^x \in G$

Lemma: Dlog hard in $G \Rightarrow f$ is one-way

Properties: $f(x), f(y) \Rightarrow f(x+y) = f(x) \cdot f(y)$

\Rightarrow key-exchange and public-key encryption

Ex. 3: The RSA one-way function

- choose random primes $p, q \approx 1024$ bits. Set $N = pq$.
- choose integers e, d s.t. $e \cdot d = 1 \pmod{\varphi(N)}$

Define: $f: \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^*$ as $f(x) = x^e \text{ in } \mathbb{Z}_N$

Lemma: f is one-way under the RSA assumption

Properties: $f(x \cdot y) = f(x) \cdot f(y)$ and **f has a trapdoor**

Summary

Public key encryption:

made possible by one-way functions
with special properties

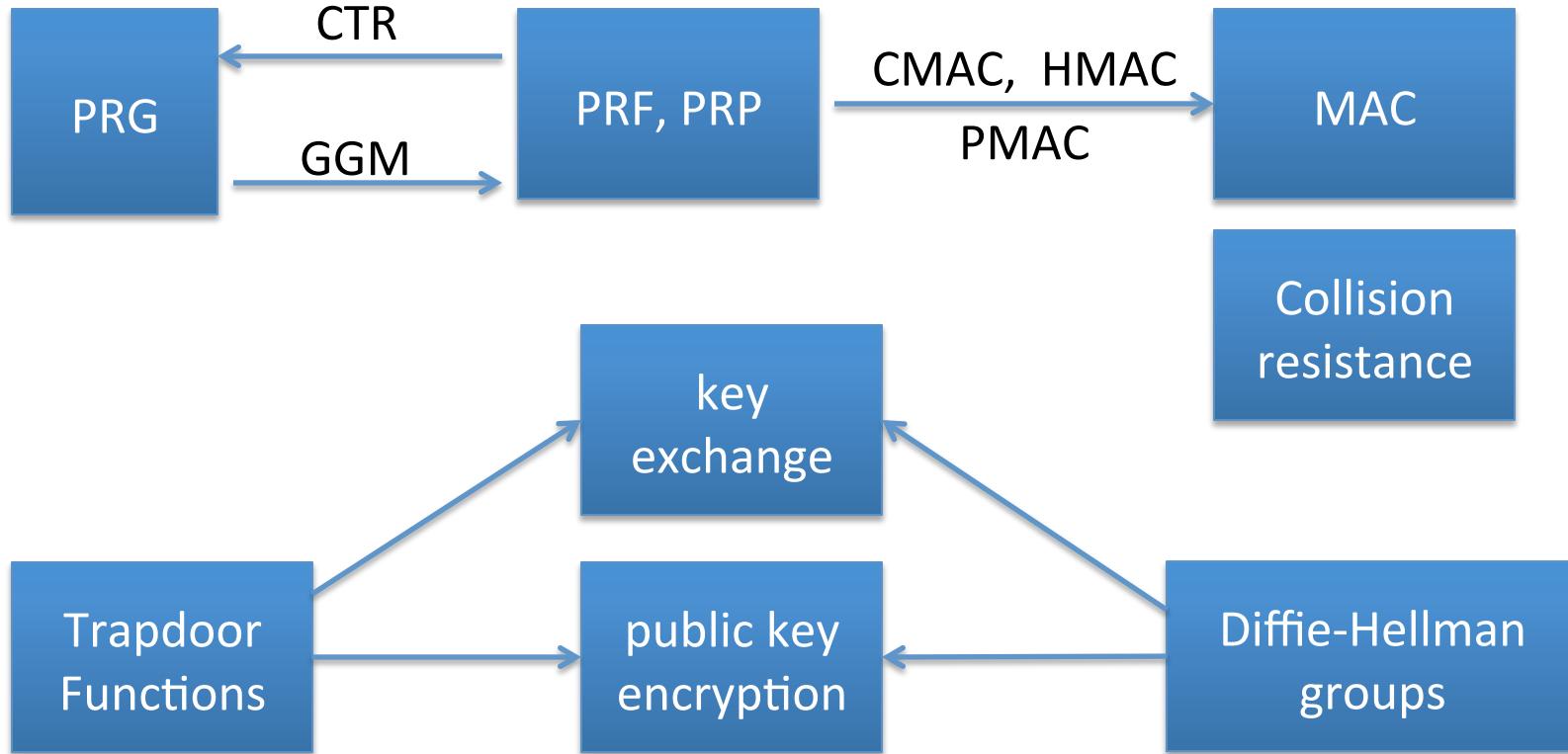
homomorphic properties and trapdoors

End of Segment



Farewell (for now)

Quick Review: primitives



Quick Review: primitives

To protect non-secret data: (data integrity)

- using small read-only storage: use collision resistant hash
- no read-only space: use MAC ... requires secret key

To protect sensitive data: only use authenticated encryption
(eavesdropping security by itself is insufficient)

Session setup:

- Interactive settings: use authenticated key-exchange protocol
- When no-interaction allowed: use public-key encryption

Remaining Core Topics (part II)

- Digital signatures and certificates
- Authenticated key exchange
- User authentication:
 - passwords, one-time passwords, challenge-response
- Privacy mechanisms
- Zero-knowledge protocols

Many more topics to cover ...

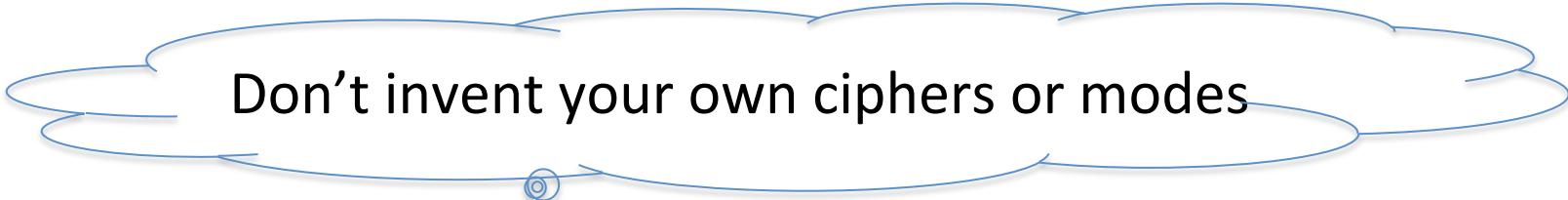
- Elliptic Curve Crypto
- Quantum computing
- New key management paradigms:
 - identity based encryption and functional encryption
- Anonymous digital cash
- Private voting and auction systems
- Computing on ciphertexts: fully homomorphic encryption
- Lattice-based crypto
- Two party and multi-party computation

Final Words

Be careful when using crypto:

- A tremendous tool, but if incorrectly implemented:
system will work, but may be easily attacked

Make sure to have others review your designs and code



Don't invent your own ciphers or modes

End of part I