EE309 Advanced Programming Techniques for EE

Lecture 22:
Authenticated encryption + Introduction to public key encryption

INSU YUN (윤인수)

School of Electrical Engineering, KAIST

[Slides from Cryptography at Coursera by Dan boneh]

Active attacks on
CPA-secure encryption

Recap: the story so far

Confidentiality: semantic security against a CPA attack

- Encryption secure against **eavesdropping only**

Integrity:

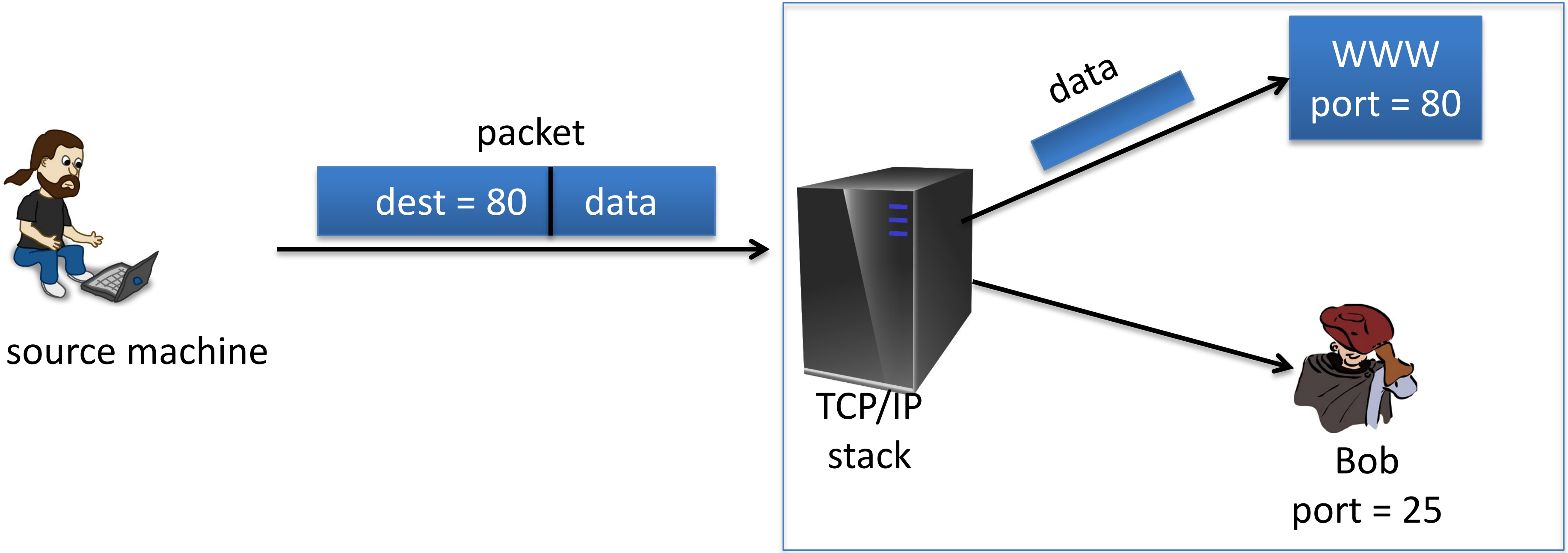
- Existential unforgeability under a chosen message attack
- CBC-MAC, HMAC, PMAC, CW-MAC

This module: encryption secure against **tampering**

- Ensuring both confidentiality and integrity

Sample tampering attacks

TCP/IP: (highly abstracted)



packet

dest = 80

data

source machine

data

WWW

port = 80

TCP/IP stack

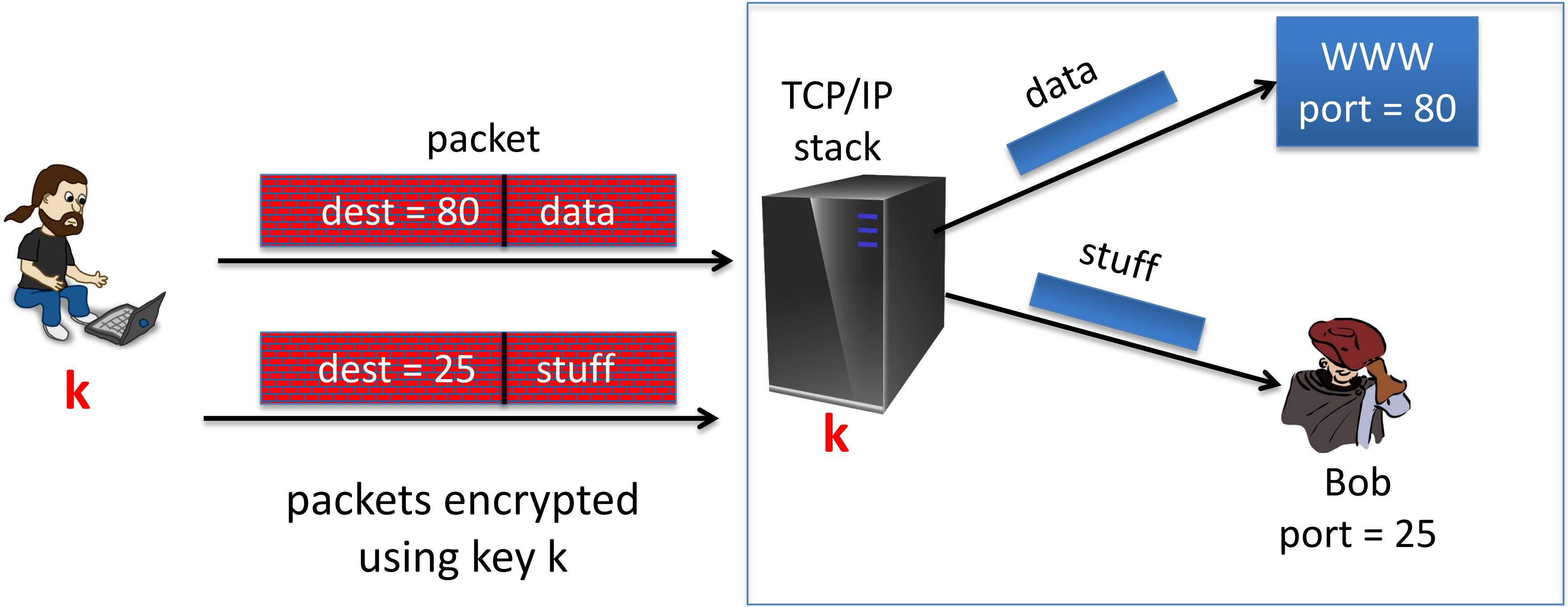
Bob

port = 25

destination machine

Sample tampering attacks

IPsec: (highly abstracted)



packet

dest = 80 | data

dest = 25 | stuff

k

packets encrypted using key k

TCP/IP stack

k

data

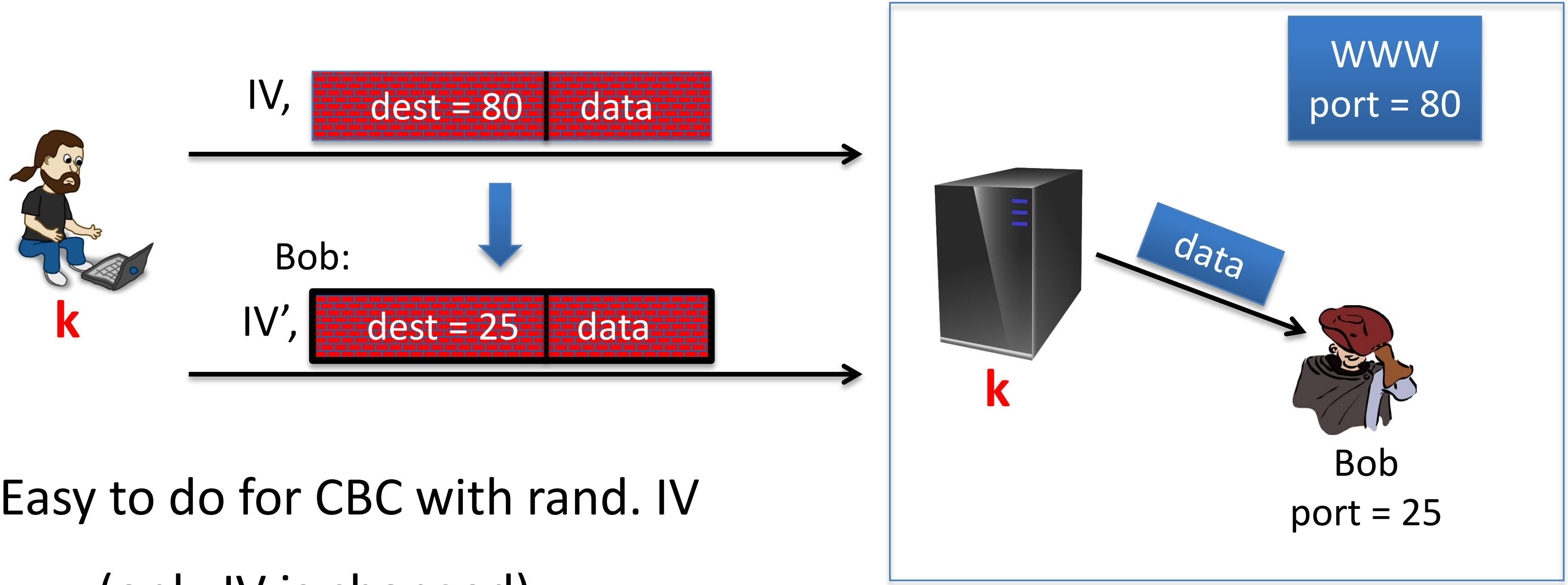
stuff

WWW
port = 80

Bob
port = 25

Reading someone else's data

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



Easy to do for CBC with rand. IV

(only IV is changed)

Encryption is done with CBC with a random IV.

What should IV' be?

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

○ $IV' = IV \oplus (\ldots 25 \ldots)$

○ $IV' = IV \oplus (\ldots 80 \ldots)$

○ $IV' = IV \oplus (\underset{00}{\ldots}80\underset{00}{\ldots}) \oplus (\underset{00}{\ldots}25\underset{00}{\ldots})$ ⟸

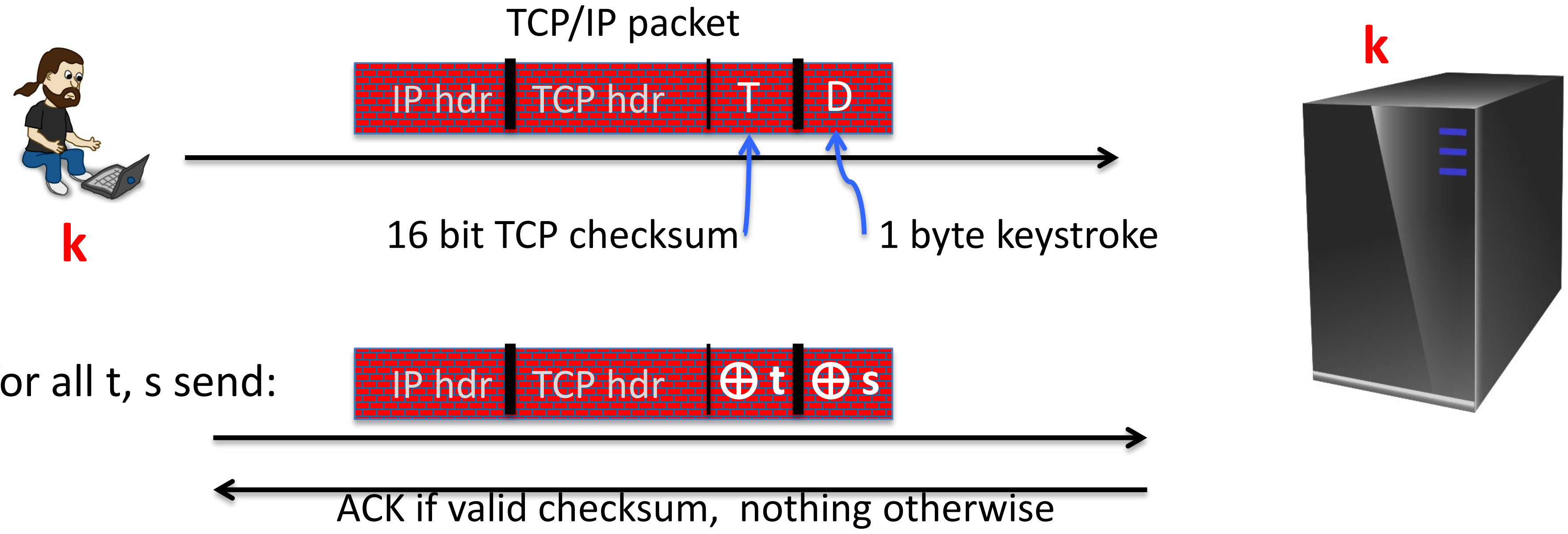
○ It can't be done

$$\underline{D(k, c[0]) \oplus IV'} = \overbrace{D(k, c[0]) \oplus IV}^{\cancel{\ldots 80 \ldots}} \oplus \cancel{80} \oplus 25$$

$$= \ldots 25 \ldots$$

An attack using only network access

Remote terminal app.: each keystroke encrypted with CTR mode



TCP/IP packet

16 bit TCP checksum

1 byte keystroke

for all t, s send:

ACK if valid checksum, nothing otherwise

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

Definitions

Goals

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

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

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

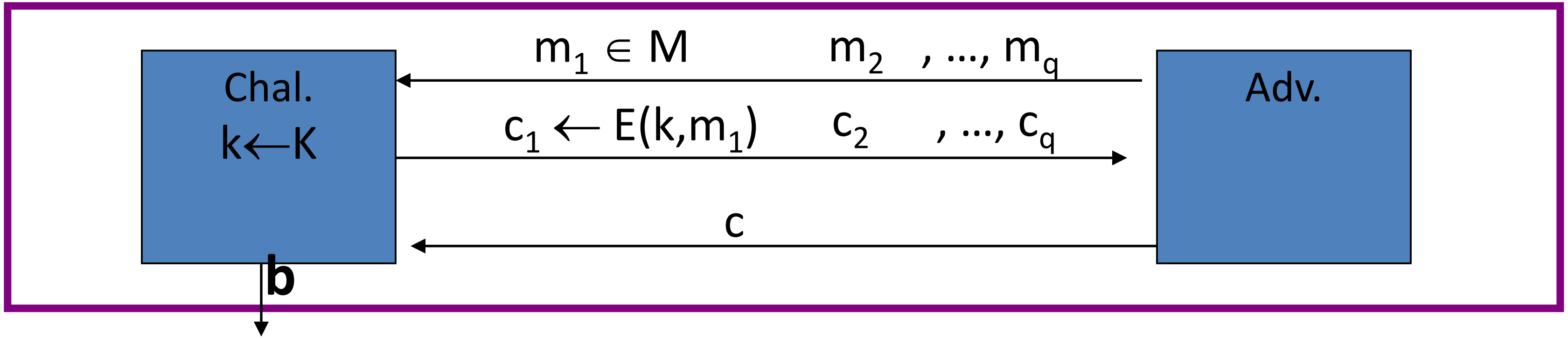
ciphertext is rejected

Security: the system must provide

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

Ciphertext integrity

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



Chal.

$k \leftarrow K$

Adv.

$m_1 \in M \quad m_2 \; , \ldots, m_q$

$c_1 \leftarrow E(k,m_1) \quad c_2 \quad , \ldots, c_q$

c

b

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

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

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

Authenticated encryption

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

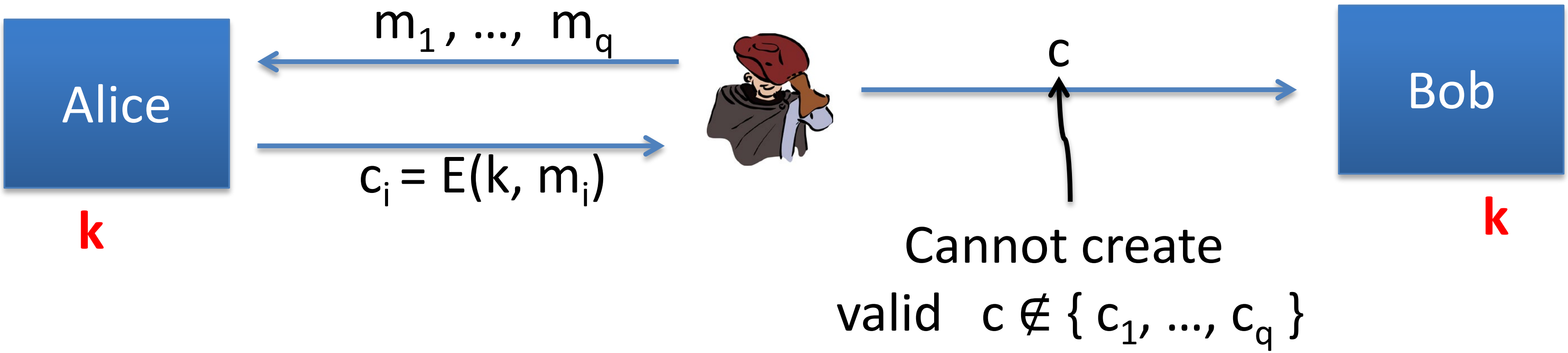
1. semantically secure under CPA, and
2. has ciphertext integrity

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



Cannot create valid $c \notin \{ c_1, \ldots, c_q \}$

$\Rightarrow$ if $D(k,c) \neq \perp$ Bob knows message is from someone who knows k

(but message could be a replay)

Implication 2

Authenticated encryption ⇒

Security against **chosen ciphertext attacks**

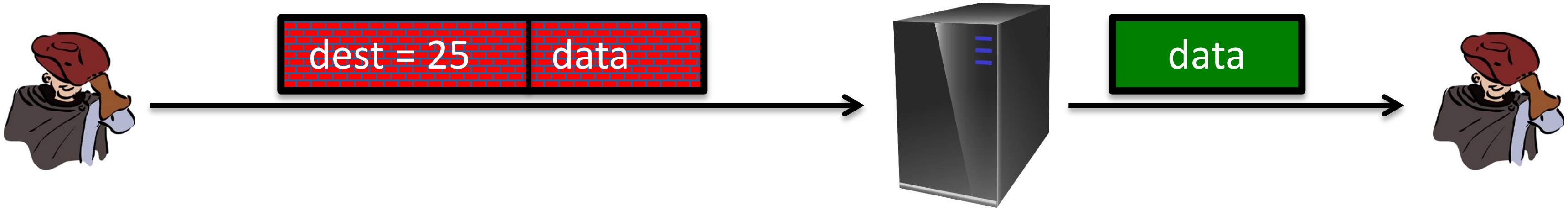
(next segment)

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)

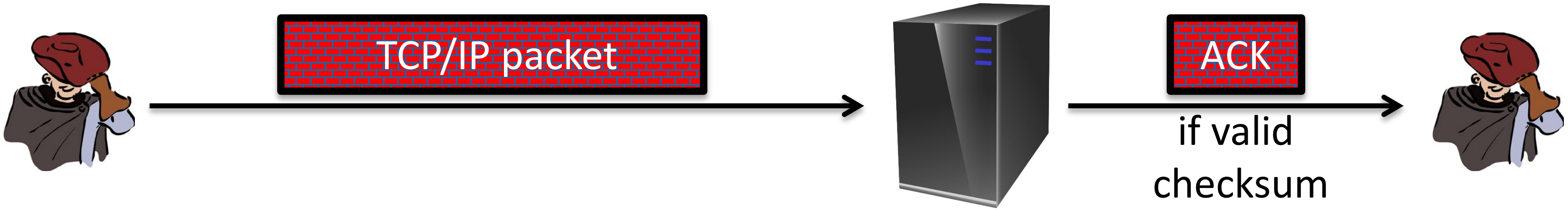


dest = 25

data

data

- Often, adversary can learn partial information about plaintext



TCP/IP packet

ACK

if valid checksum

Chosen ciphertext security

Adversary's power: both CPA and CCA

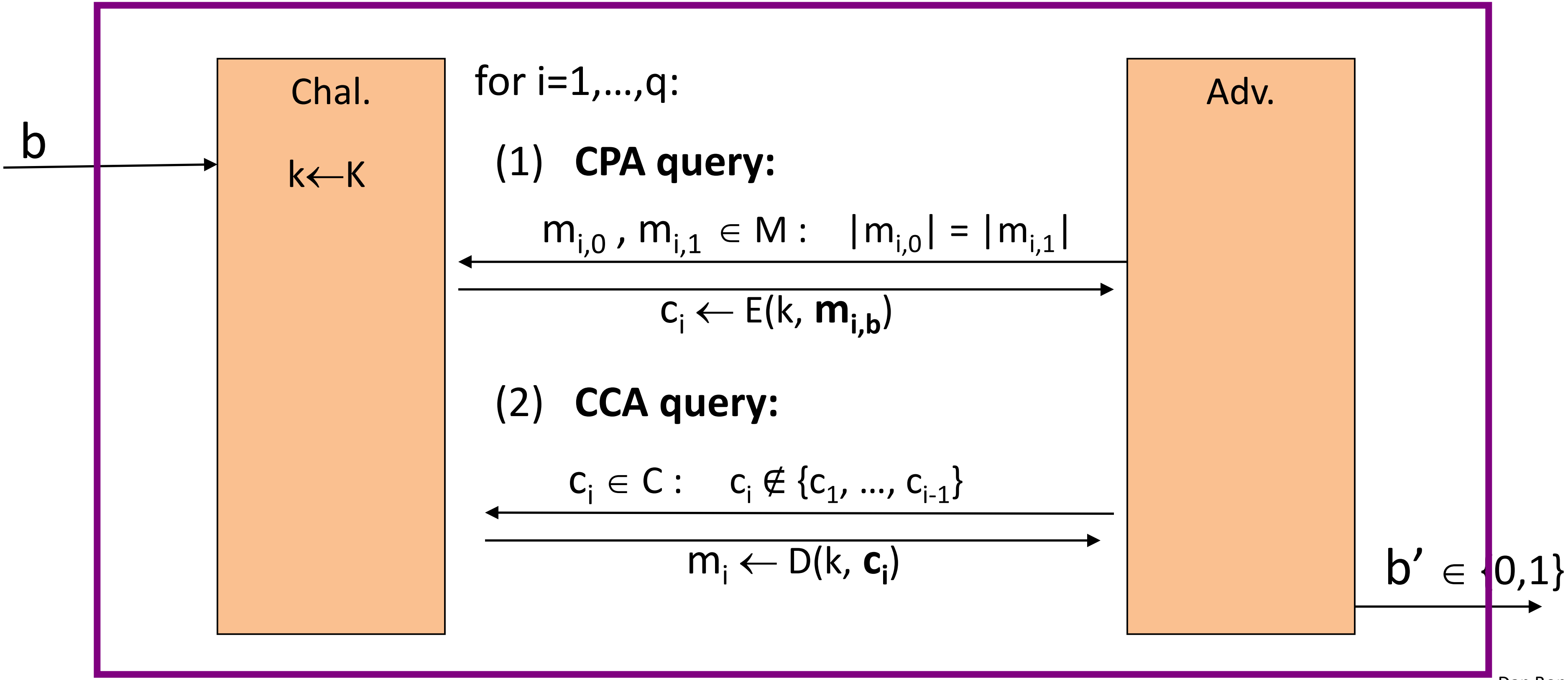
- Can obtain the encryption of arbitrary messages of his choice
- Can decrypt any ciphertext of his choice, other than challenge

(conservative modeling of real life)

Adversary's goal: Break sematic security

Chosen ciphertext security: definition

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



b

Chal.

$k \leftarrow K$

for $i=1,\ldots,q$:

(1) **CPA query:**

$m_{i,0}\,,\, m_{i,1} \in M: \quad |m_{i,0}| = |m_{i,1}|$

$c_i \leftarrow E(k, \mathbf{m_{i,b}})$

(2) **CCA query:**

$c_i \in C: \quad c_i \notin \{c_1, \ldots, c_{i-1}\}$

$m_i \leftarrow D(k, \mathbf{c_i})$

Adv.

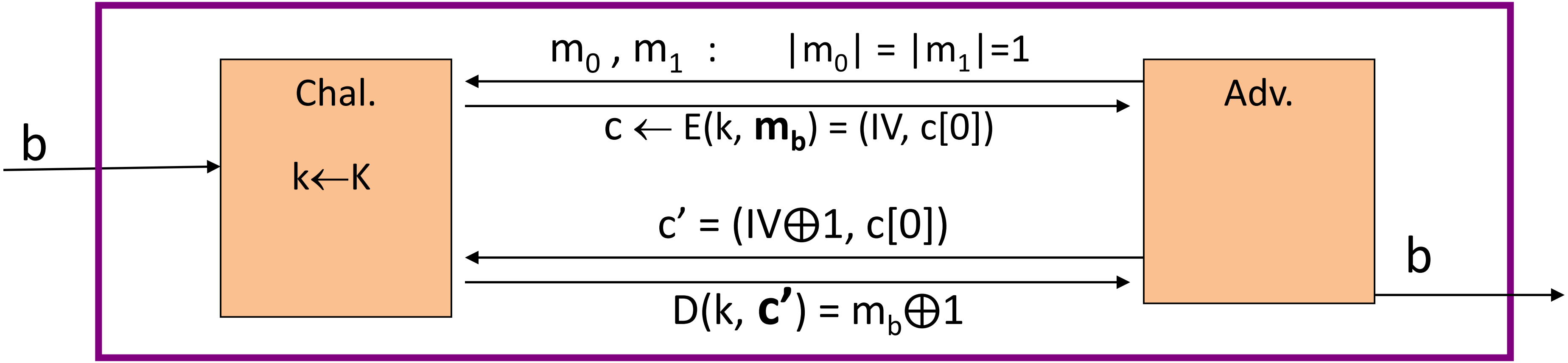
$b' \in \{0,1\}$

Chosen ciphertext security: definition

$\mathbb{E}$ is CCA secure if for all "efficient" A:

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

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



Authenticated enc. ⇒ CCA security

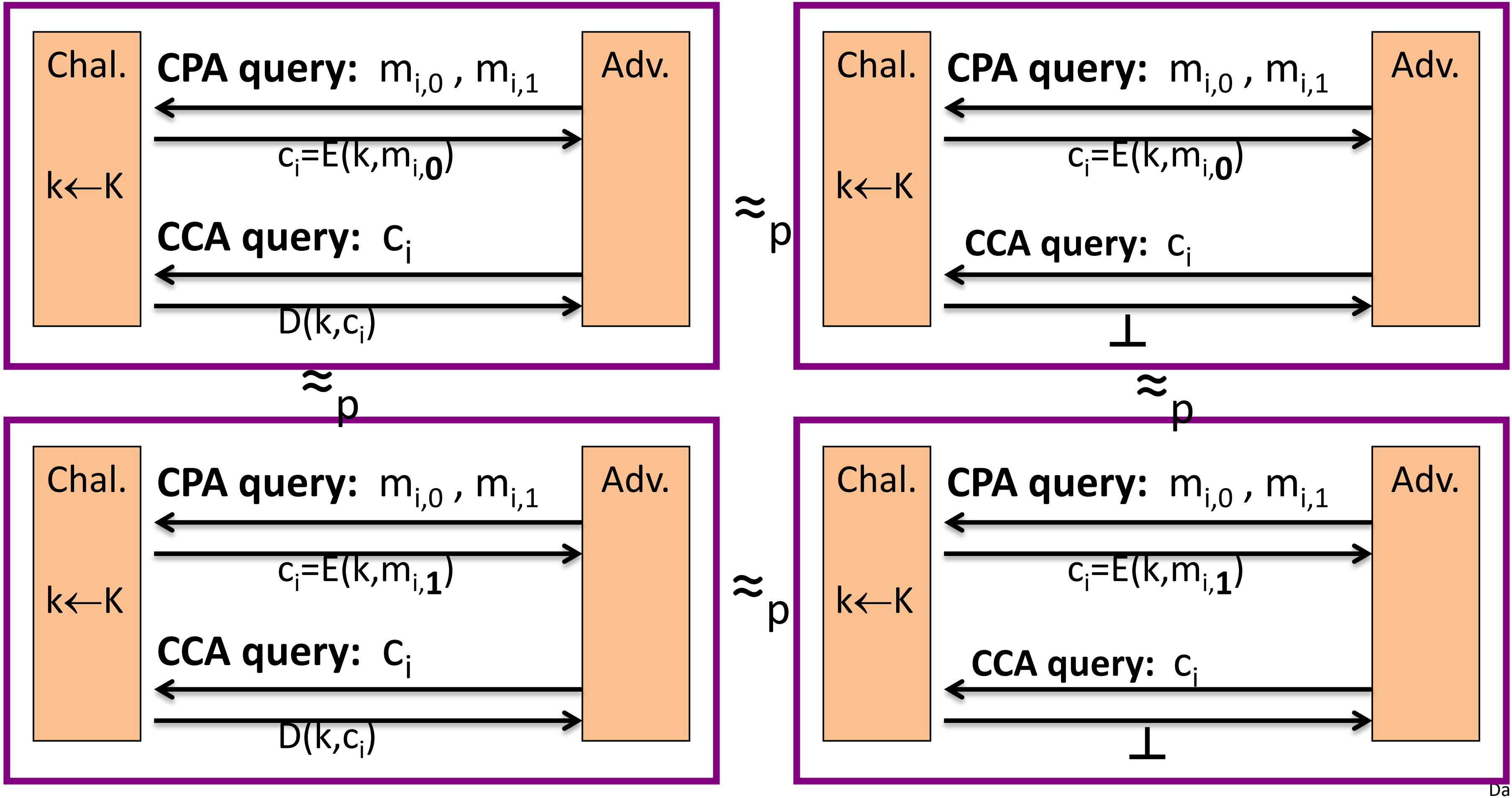
Thm: Let (E,D) be a cipher that provides AE.

Then (E,D) is CCA secure !

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

$$\mathrm{Adv_{CCA}}[A,E] \leq 2q\cdot\mathrm{Adv_{CI}}[B_1,E] + \mathrm{Adv_{CPA}}[B_2,E]$$

Proof by pictures



Chal.

$k \leftarrow K$

Adv.

CPA query: $m_{i,0}$, $m_{i,1}$

$c_i=E(k,m_{i,\mathbf{0}})$

CCA query: c_i

$D(k,c_i)$

$\approx_p$

Chal.

$k \leftarrow K$

Adv.

CPA query: $m_{i,0}$, $m_{i,1}$

$c_i=E(k,m_{i,\mathbf{0}})$

CCA query: c_i

$\perp$

$\approx_p$

$\approx_p$

Chal.

$k \leftarrow K$

Adv.

CPA query: $m_{i,0}$, $m_{i,1}$

$c_i=E(k,m_{i,\mathbf{1}})$

CCA query: c_i

$D(k,c_i)$

$\approx_p$

Chal.

$k \leftarrow K$

Adv.

CPA query: $m_{i,0}$, $m_{i,1}$

$c_i=E(k,m_{i,\mathbf{1}})$

CCA query: c_i

$\perp$

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)

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)

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

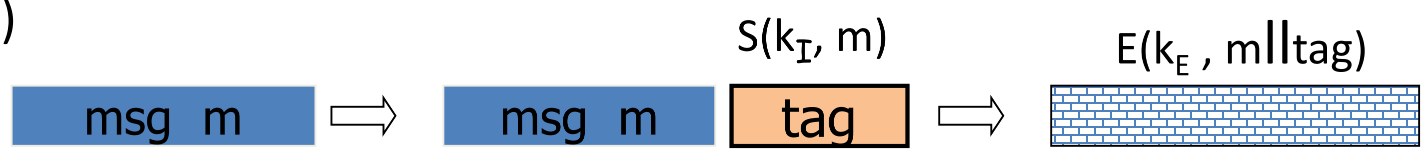
Every project had to combine the two itself without
a well defined goal

- Not all combinations provide AE …

Combining MAC and ENC (CCA)

Encryption key k_E. MAC key = k_I

Option 1: (SSL)

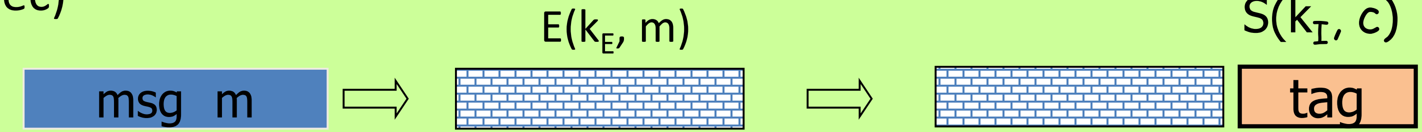


msg m ⇨ msg m | tag ⇨ $E(k_E, m||tag)$

$S(k_I, m)$

Option 2: (IPsec)

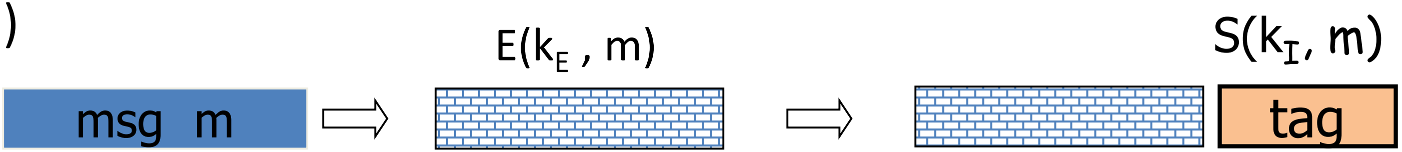
always correct



msg m ⇨ $E(k_E, m)$ ⇨ ciphertext | tag

$S(k_I, c)$

Option 3: (SSH)



msg m ⇨ $E(k_E, m)$ ⇨ ciphertext | tag

$S(k_I, m)$

A.E. Theorems

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

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

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

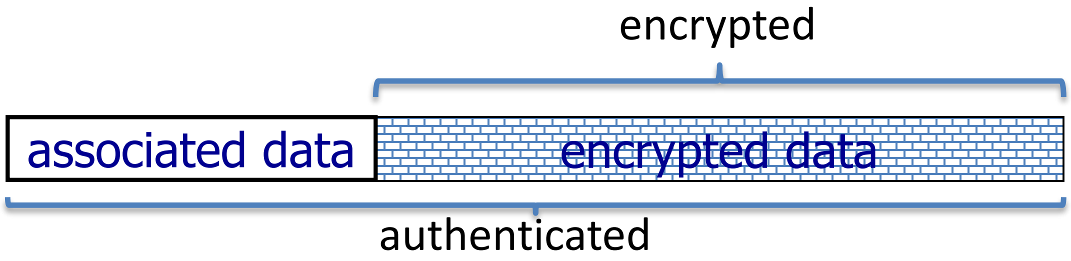
however: when (E,D) is rand-CTR mode or rand-CBC
M-then-E provides A.E.

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

Standards (at a high level)

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

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



An example API (OpenSSL)

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


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

Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

Cipher	code size	Speed (MB/sec)		
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

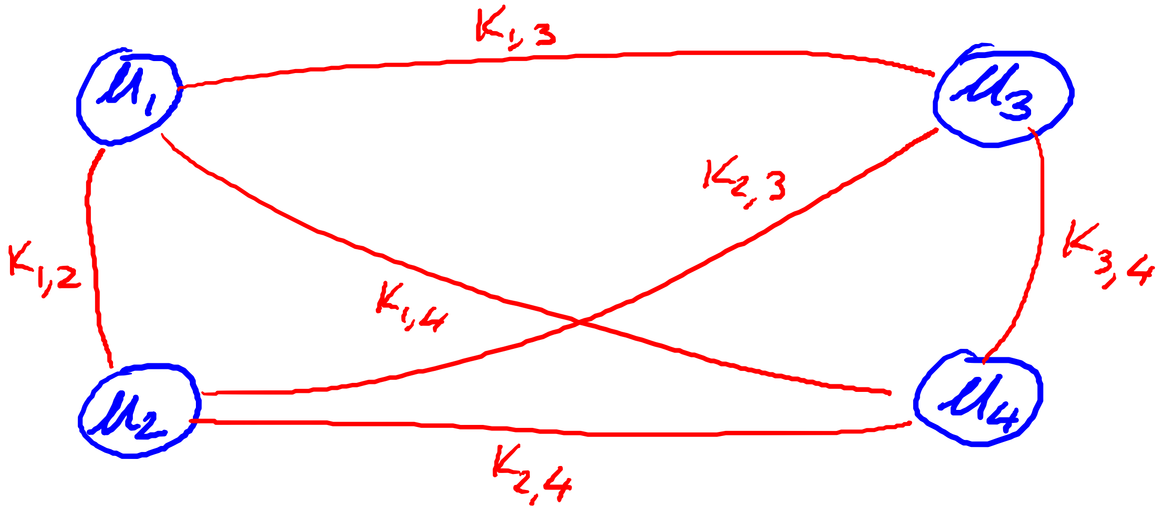
* extrapolated from Ted Kravitz's results

** non-Intel machines

Key management

Key management

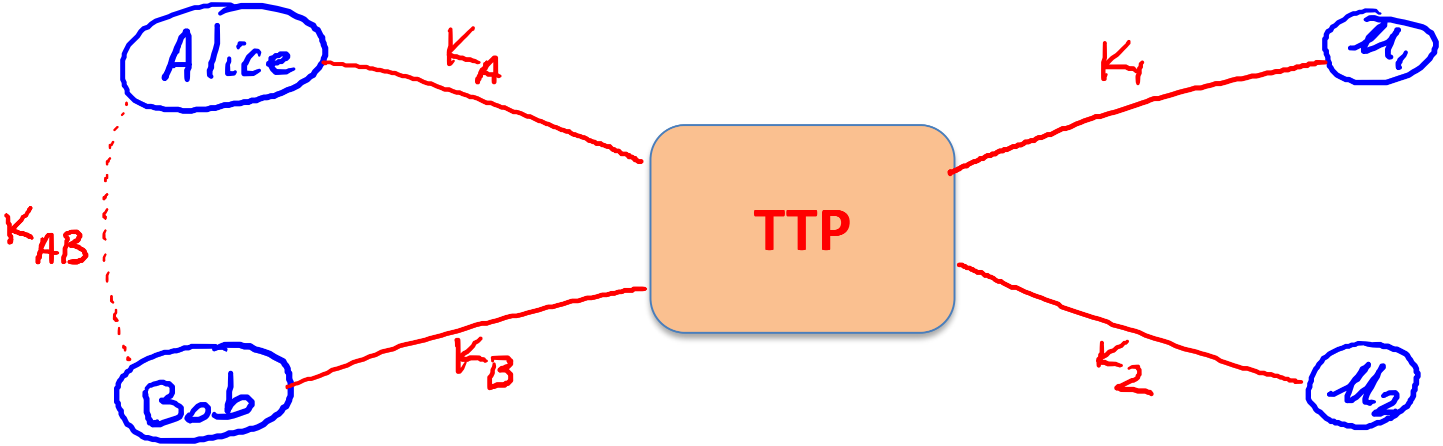
Problem: n users. Storing mutual secret keys is difficult



Total: O(n) keys per user

A better solution

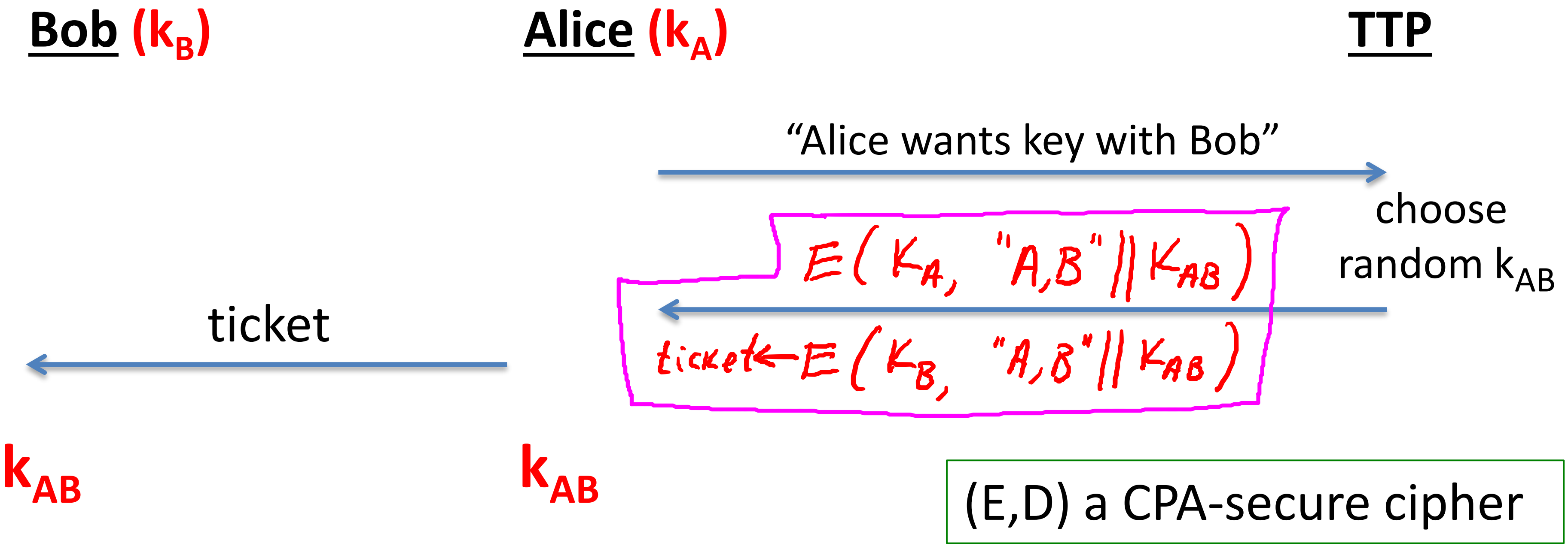
Online Trusted 3rd Party (TTP)



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**

"Alice wants key with Bob"

choose random k_{AB}

$E(K_A, \text{"A,B"} \| K_{AB})$

$\text{ticket} \leftarrow E(K_B, \text{"A,B"} \| K_{AB})$

ticket

k_{AB} **k_{AB}**

(E,D) a CPA-secure cipher

Generating keys: a toy protocol

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

Eavesdropper sees: $E(k_A,$ "A, B" ll $k_{AB})$; $E(k_B,$ "A, B" ll $k_{AB})$

(E,D) is CPA-secure ⇒

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:

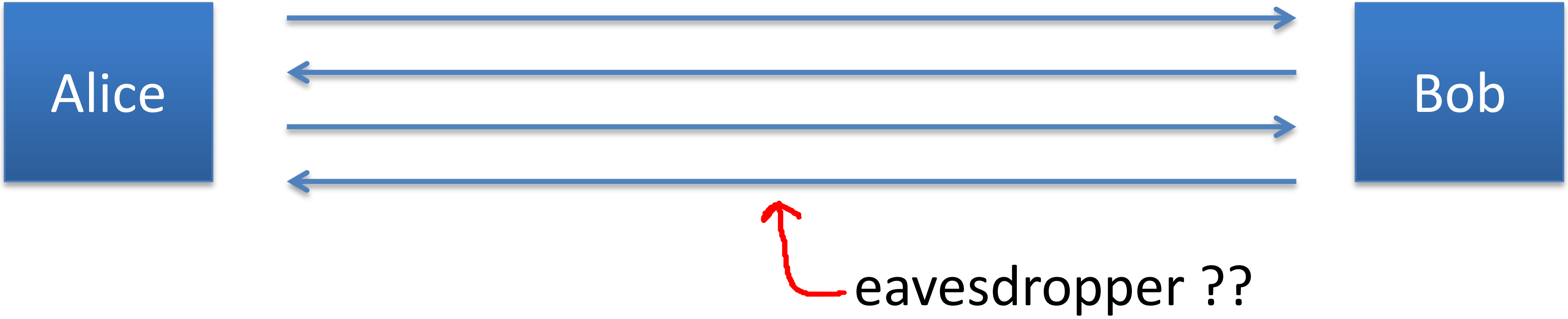
- Merkle (1974), Diffie-Hellman (1976), RSA (1977)
- More recently: ID-based enc. (BF 2001), Functional enc. (BSW 2011)

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?

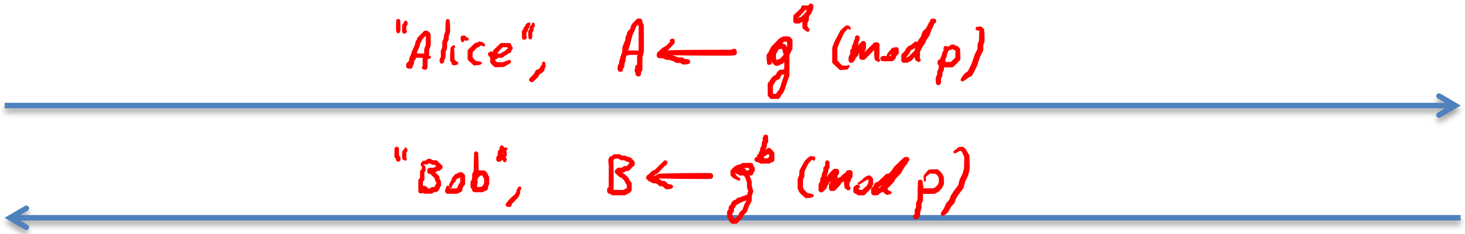
The Diffie-Hellman protocol (informally)

Fix a large prime p (e.g. 600 digits)

Fix an integer g in {1, …, p}

Alice **Bob**

choose random **a** in {1,…,p-1} choose random **b** in {1,…,p-1}



"Alice", $A \leftarrow g^a \pmod{p}$

"Bob", $B \leftarrow g^b \pmod{p}$

$$\mathbf{B^a} \pmod{p} = (g^b)^a = \mathbf{k_{AB} = g^{ab}} \pmod{p} = (g^a)^b = \mathbf{A^b} \pmod{p}$$

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($\tilde{O}(\sqrt[3]{n})$)

cipher key size	modulus size	Elliptic Curve size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	**15360** bits	512 bits

As a result: slow transition away from (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.

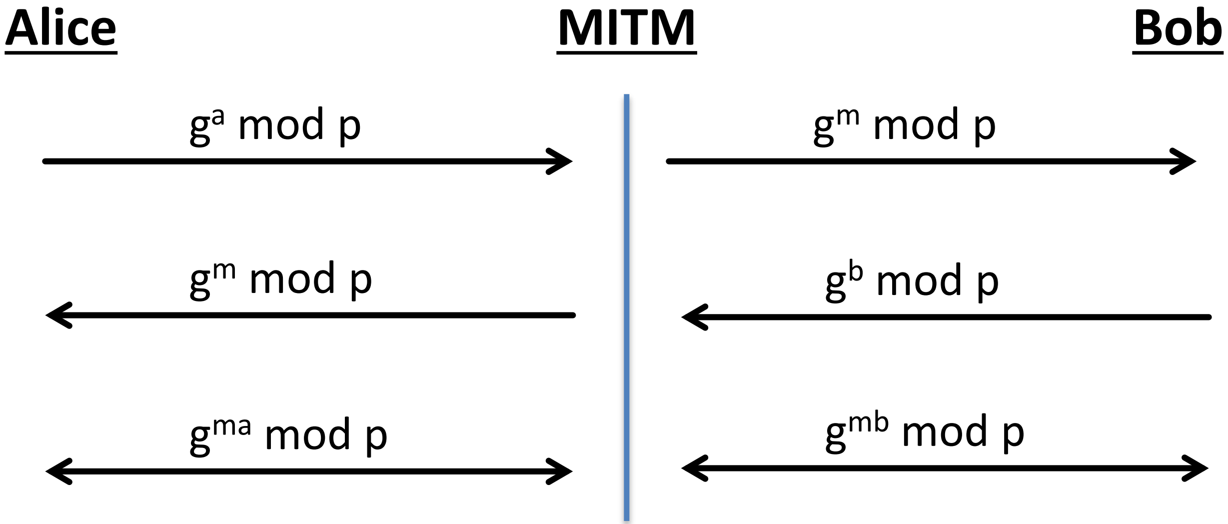
The connection uses TLS 1.0.

The connection is encrypted using RC4_128, with SHA1 for message authentication and ECDHE_RSA as the key exchange mechanism.

Elliptic curve
Diffie-Hellman

MITM Adversary

As described, Diffie-Hellman is *insecure* against *active* Man In The Middle (MITM) attacks



Alice **MITM** **Bob**

g^a mod p

g^m mod p

g^m mod p

g^b mod p

g^{ma} mod p

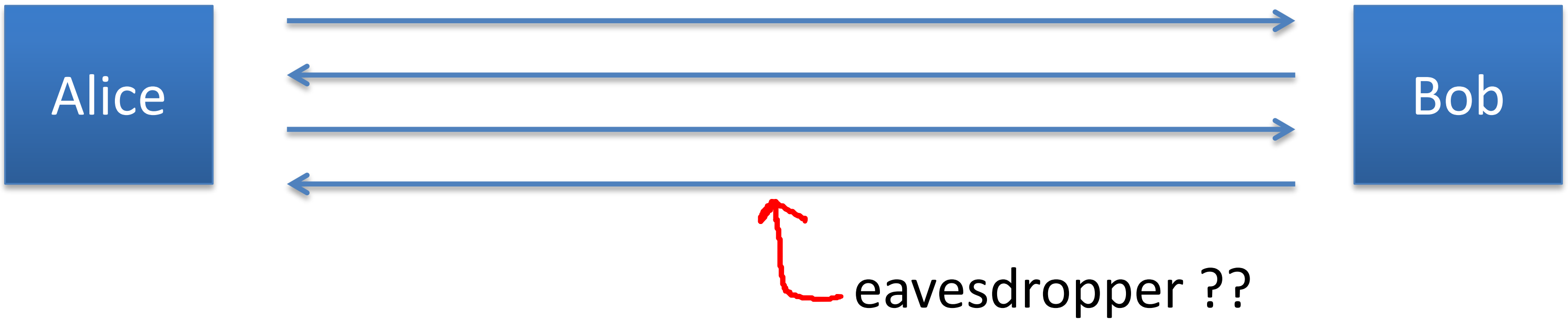
g^{mb} mod p

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)



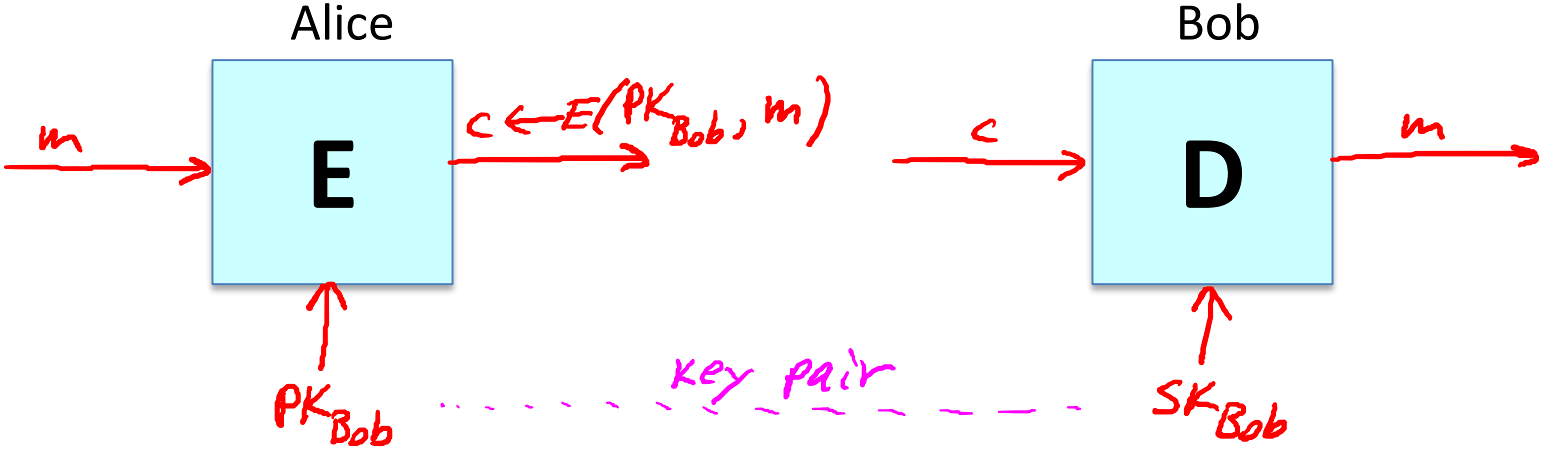
Alice

Bob

eavesdropper ??

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)

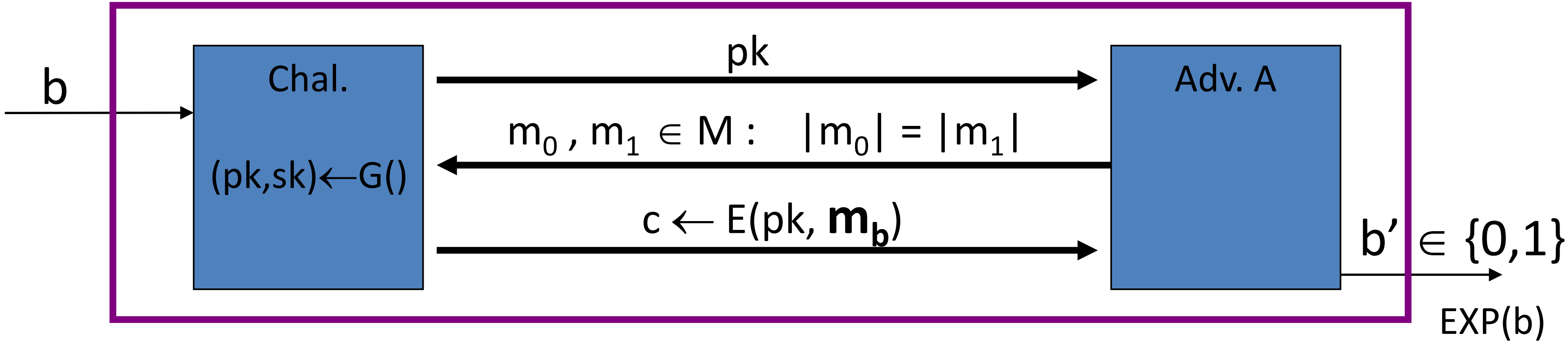
- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

Consistency: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad D(sk,\ E(pk,\ m)\ ) = m$$

Semantic Security

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



b

Chal.

$(pk,sk) \leftarrow G()$

pk

$m_0, m_1 \in M: \quad |m_0| = |m_1|$

$c \leftarrow E(pk, \mathbf{m_b})$

Adv. A

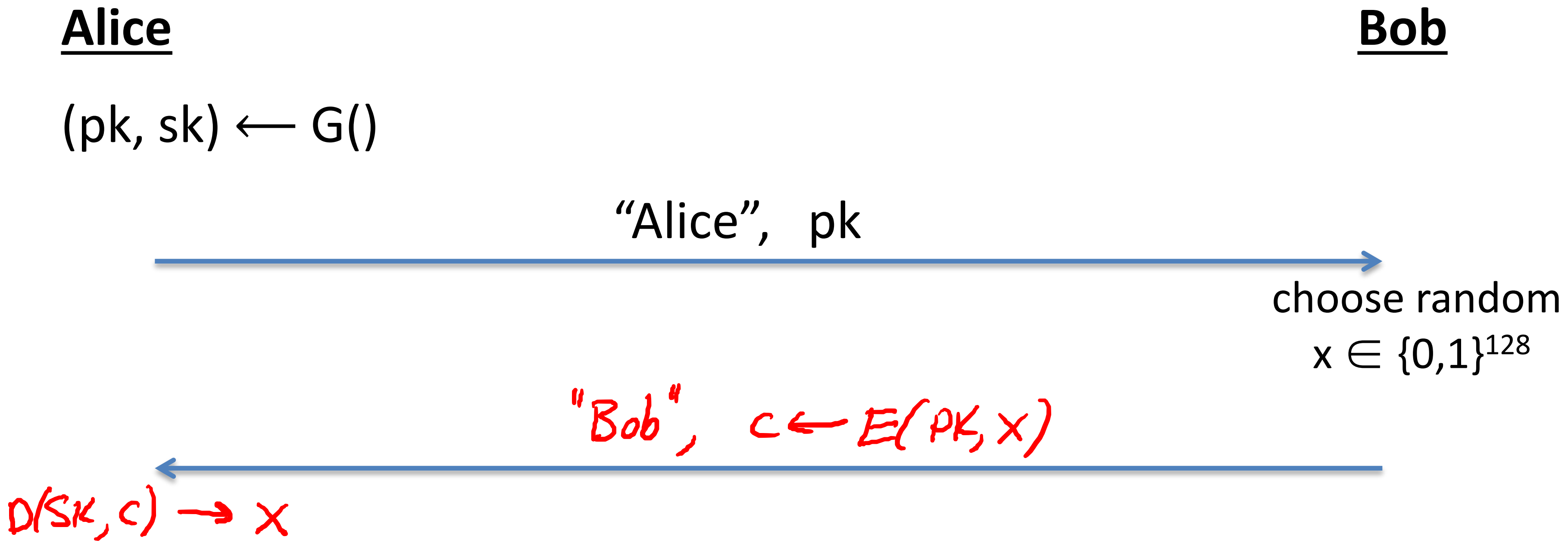
$b' \in \{0,1\}$

EXP(b)

Def: $\mathbb{E}$ =(G,E,D) is sem. secure (a.k.a IND-CPA) if for all efficient A:

$$\text{Adv}_{SS}[A,\mathbb{E}] = \Big| \Pr[\text{EXP}(0)=1] - \Pr[\text{EXP}(1)=1] \Big| < \text{negligible}$$

Establishing a shared secret



Alice

Bob

(pk, sk) ⟵ G()

"Alice", pk

choose random

$x \in \{0,1\}^{128}$

"Bob", $c \leftarrow E(PK, x)$

$D(SK, c) \rightarrow x$

x: shared secret

Security (eavesdropping)

Adversary sees **pk, E(pk, x)** and wants $x \in M$

Semantic security ⇒

adversary cannot distinguish

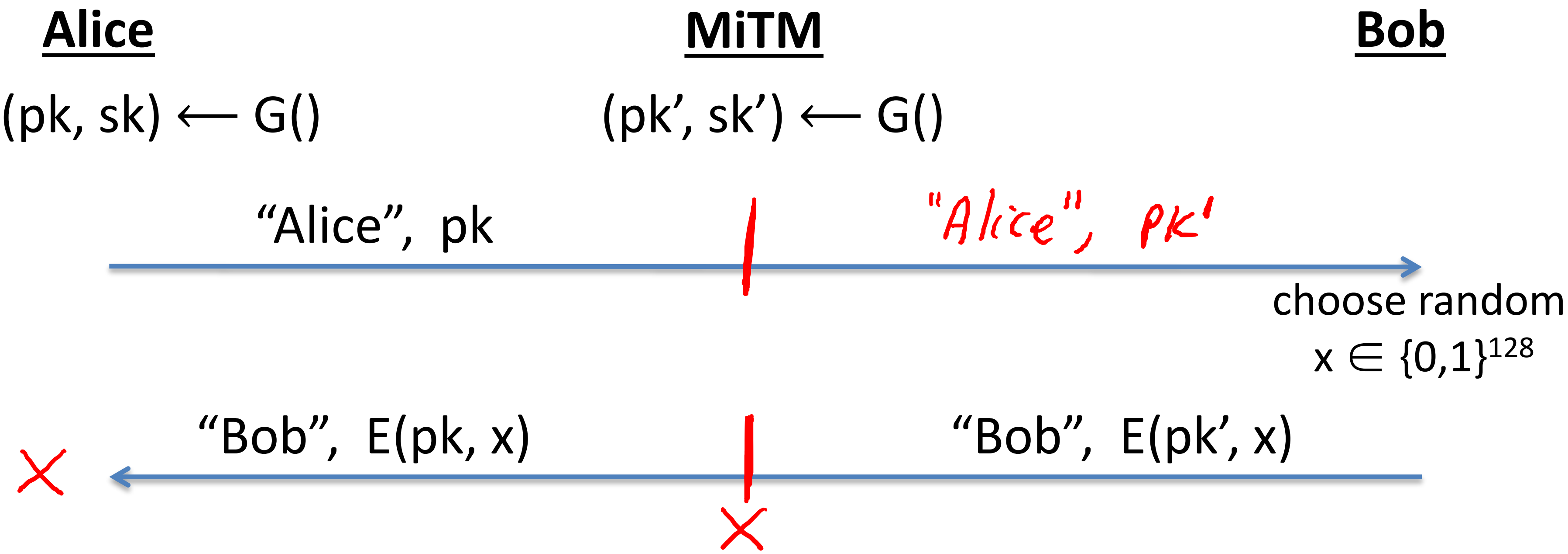
$\{ pk, E(pk, x), x \}$ from $\{ pk, E(pk, x), rand \in M \}$

⇒ 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



Alice **MiTM** **Bob**

$(pk, sk) \longleftarrow G()$ $(pk', sk') \longleftarrow G()$

“Alice”, pk "Alice", pk'

choose random $x \in \{0,1\}^{128}$

“Bob”, E(pk, x) “Bob”, E(pk’, x)

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