

Authenticated Encryption - Active attacks on CPA-secure encryption

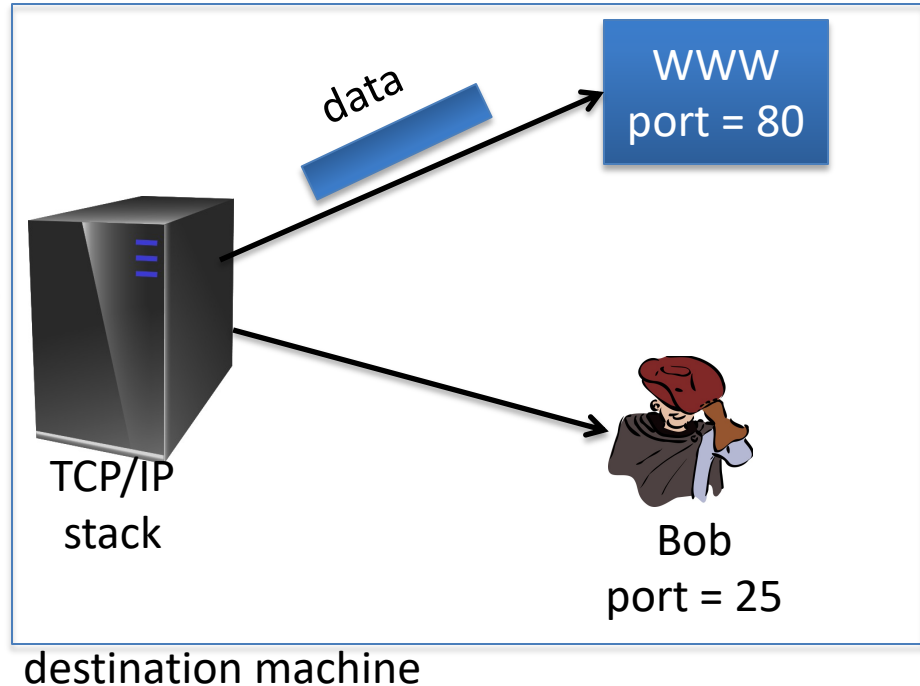
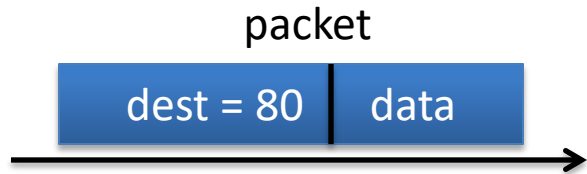
This slide is made based the online course of Cryptography by Dan Boneh

Sample tampering attacks

TCP/IP: (highly abstracted)

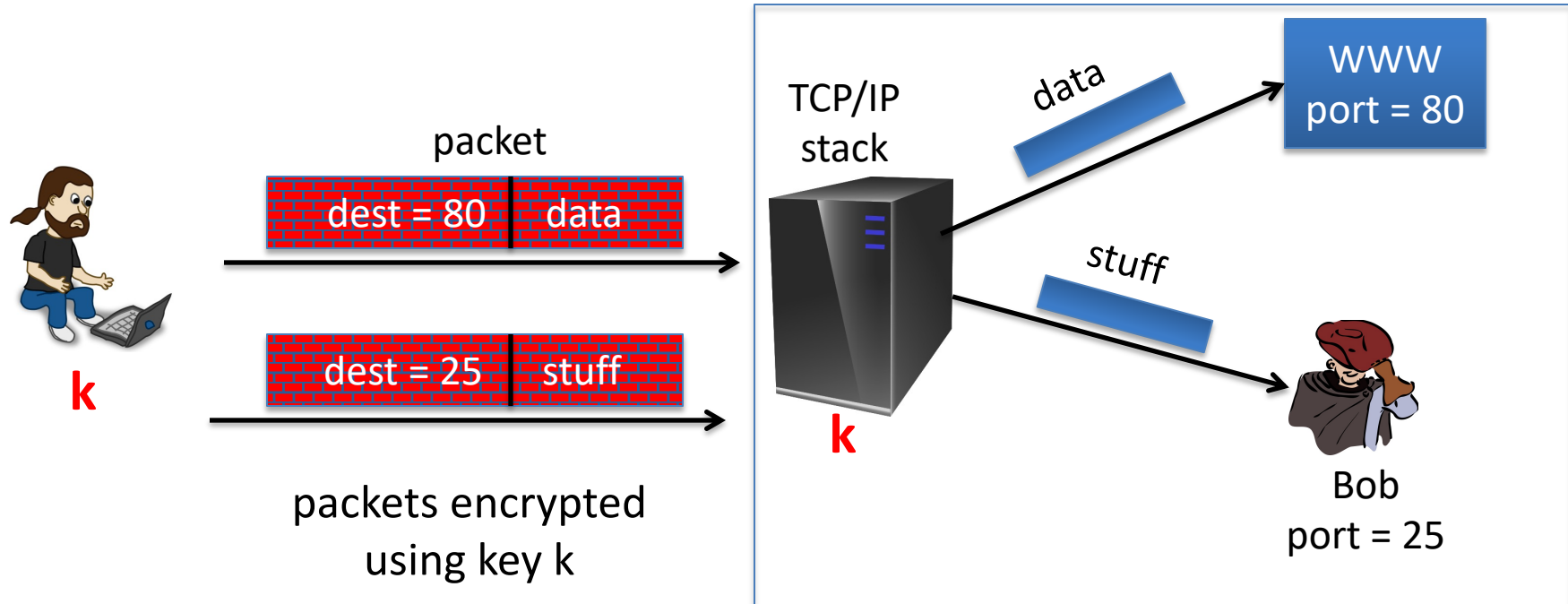


source machine



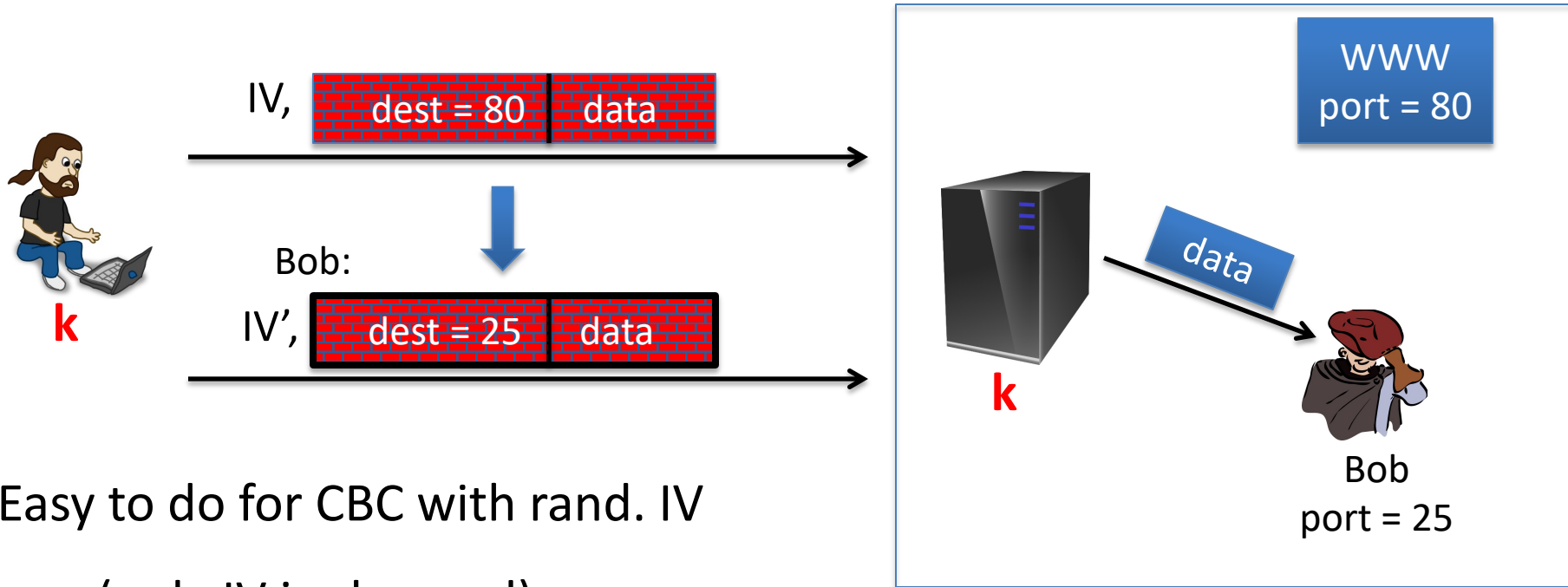
Sample tampering attacks

IPsec: (highly abstracted)



Reading someone else's data

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



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

IV ,

dest = 80	data
-----------	------



IV' ,

dest = 25	data
-----------	------

Encryption is done with CBC with a random IV.

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

- ☐ $IV' = IV \oplus (...25...)$
- ☐ $IV' = IV \oplus (...80...)$
- ☒ $IV' = IV \oplus (...80...) \oplus (...25...)$
- ☐ It can't be done

The lesson

CPA security cannot guarantee secrecy under active attacks.

Only use one of two modes:

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

End of Segment

Authenticated Encryption

Definitions

Goals


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

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

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

Security: the system must provide

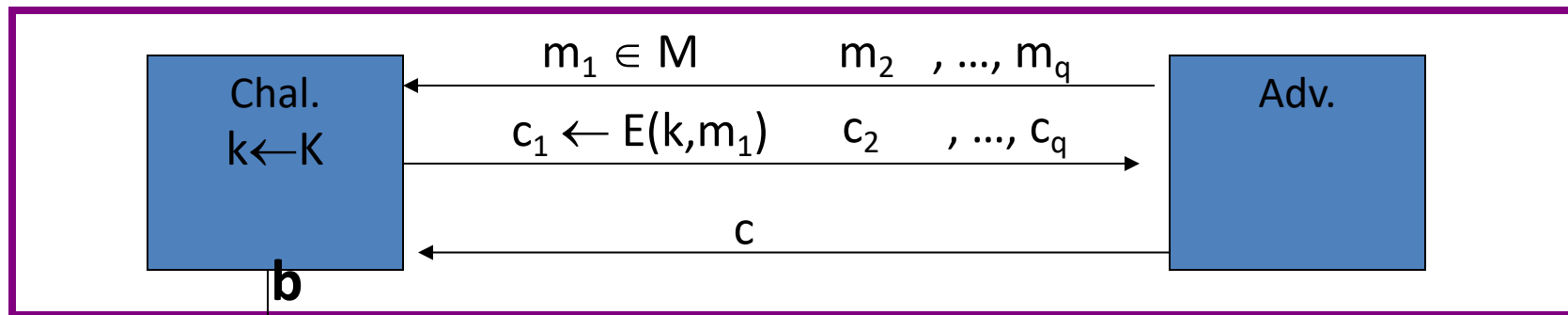
- sem. security under a CPA attack, and
- **ciphertext integrity**:
attacker cannot create new ciphertexts that decrypt properly



ciphertext
is rejected

Ciphertext integrity

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



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

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

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

Authenticated encryption

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

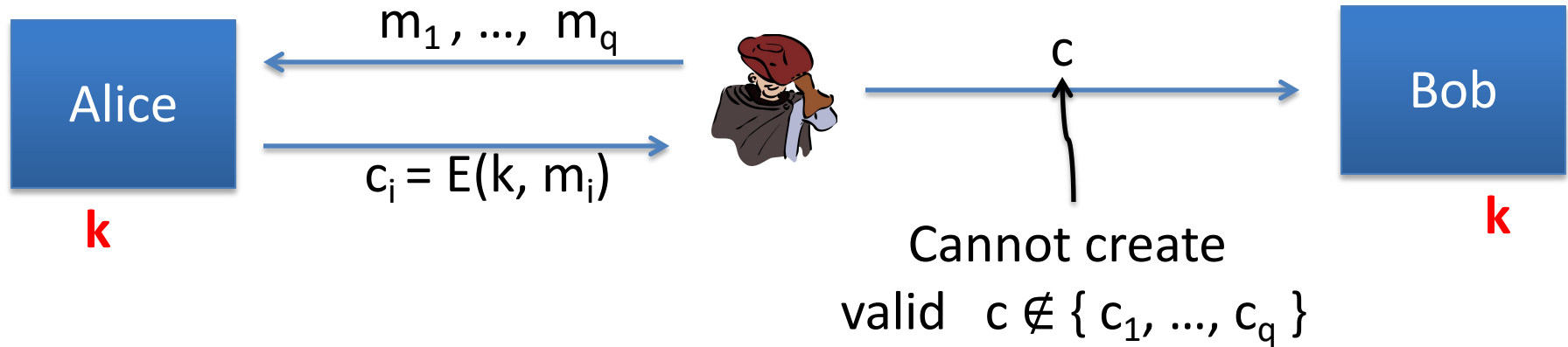
- (1) semantically secure under CPA, and
- (2) has ciphertext integrity

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

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

Implication 1: authenticity

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



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

Implication 2

Authenticated encryption \Rightarrow

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

End of Segment

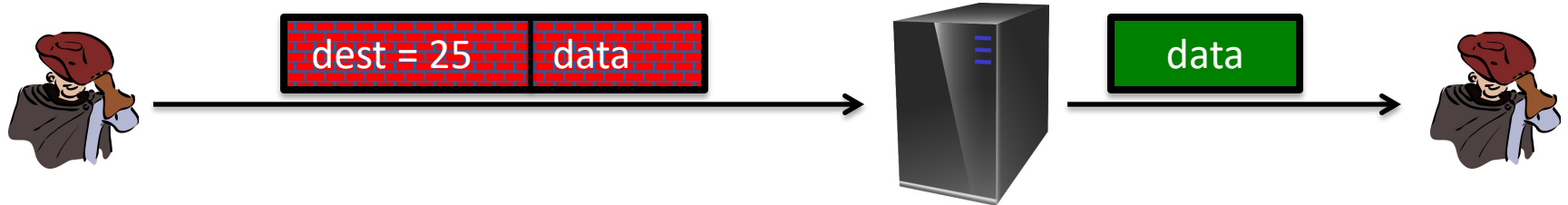
Authenticated Encryption

Chosen ciphertext attacks

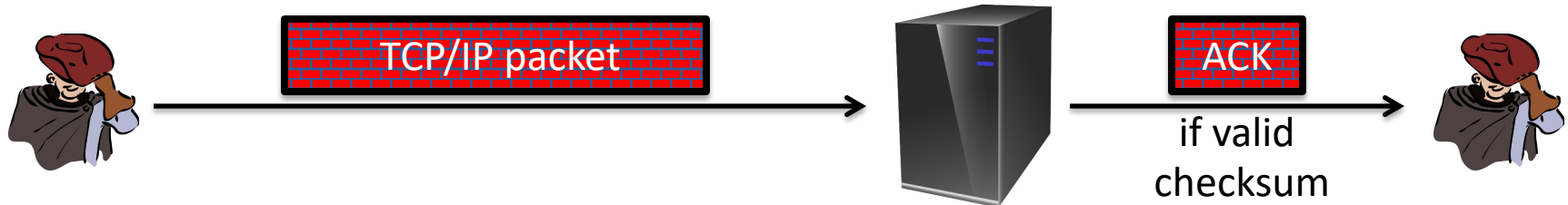
Example chosen ciphertext attacks

Adversary has ciphertext c that it wants to decrypt

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



- Often, adversary can learn partial information about plaintext



Chosen ciphertext security

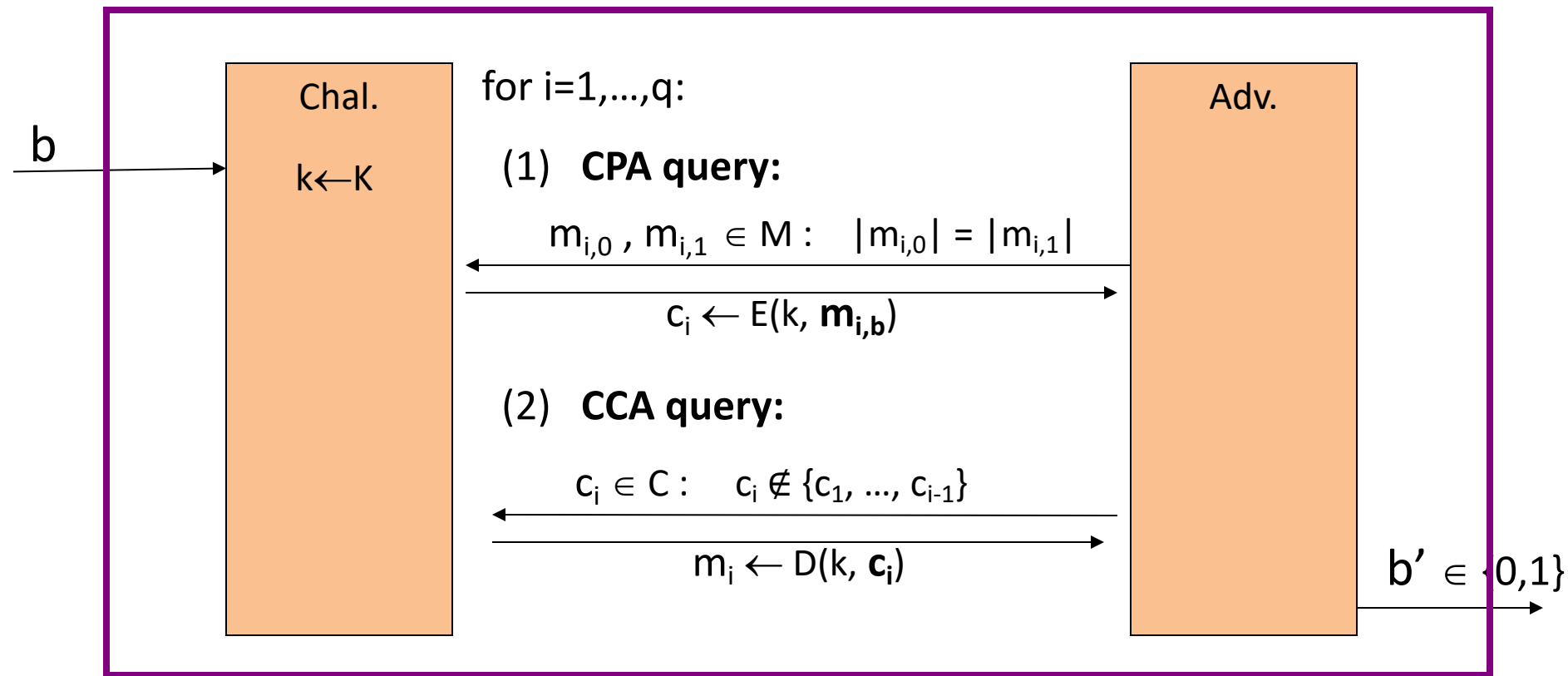
Adversary's power: both CPA and CCA

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

Adversary's goal: Break semantic security

Chosen ciphertext security: definition

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

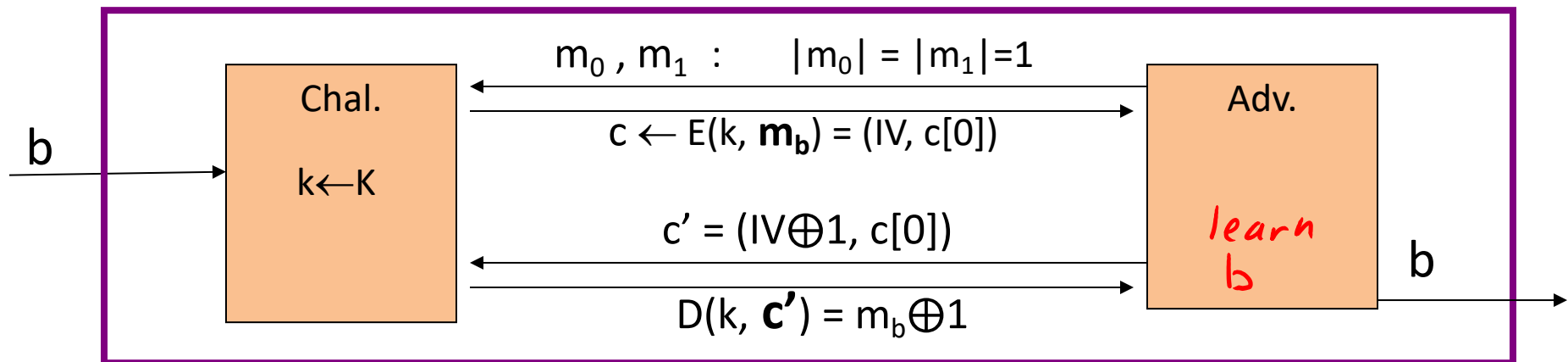


Chosen ciphertext security: definition

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

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

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



Authenticated enc. \Rightarrow CCA security

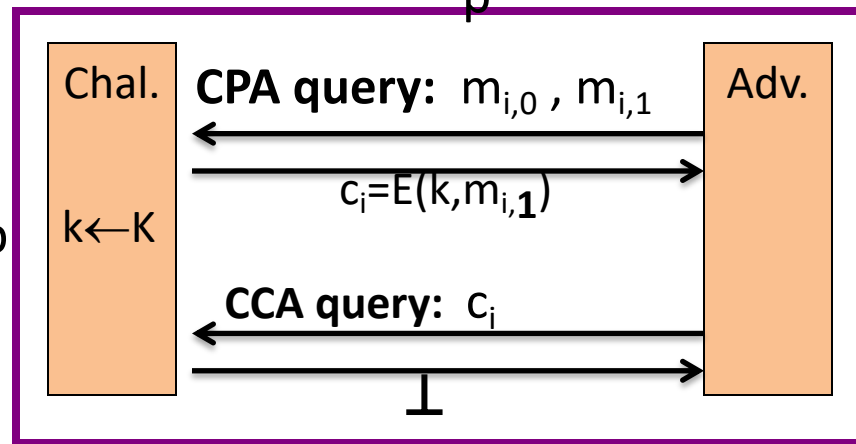
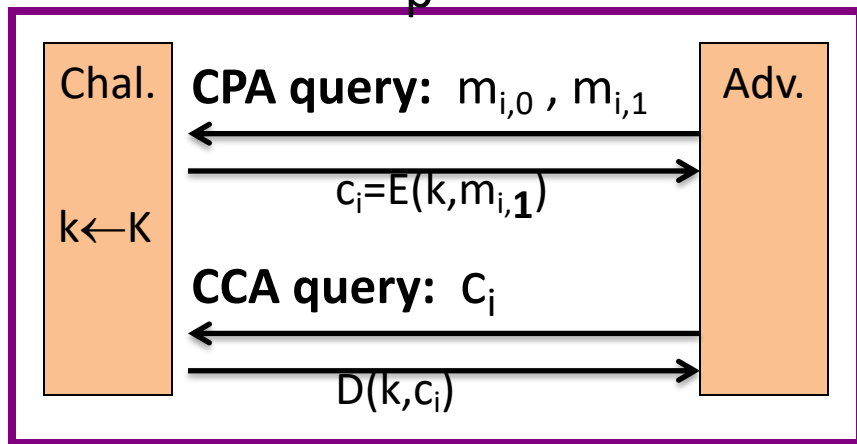
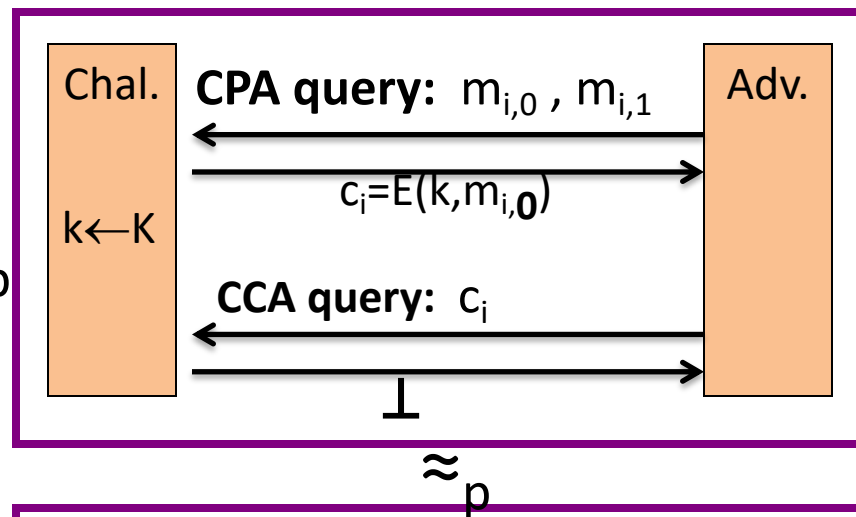
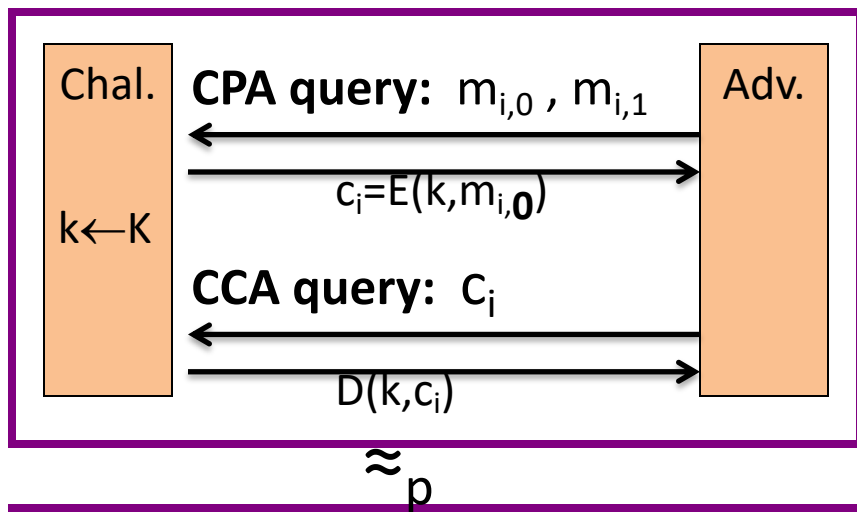
Thm: Let (E,D) be a cipher that provides AE.

Then (E,D) is CCA secure !

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

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

Proof by pictures



So what?

Authenticated encryption:

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

Limitations:

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

End of Segment

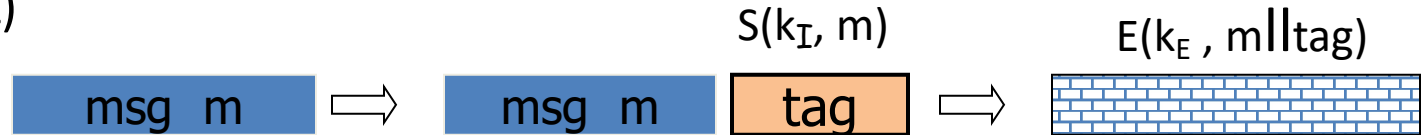
Authenticated Encryption

Constructions from
ciphers and MACs

Combining MAC and ENC (CCA)

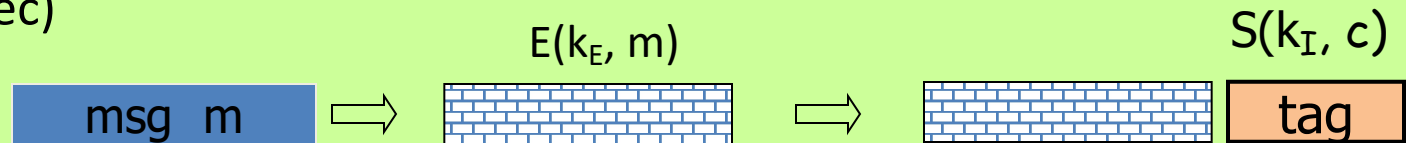
Encryption key k_E . MAC key = k_I

Option 1: (SSL)

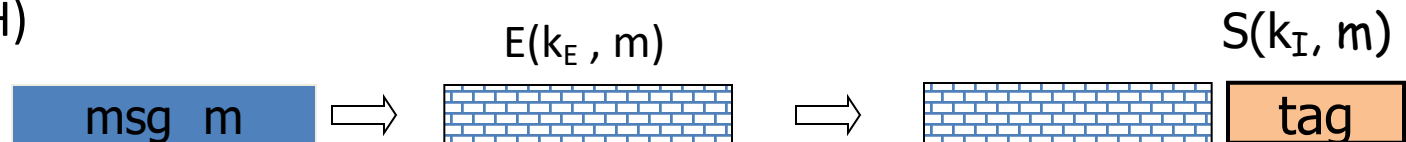


Option 2: (IPsec)

**always
correct**



Option 3: (SSH)



A.E. Theorems

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

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

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

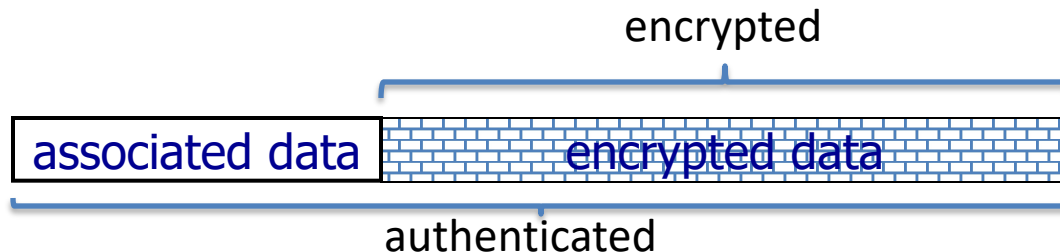
however: when (E,D) is rand-CTR mode or rand-CBC
M-then-E provides A.E.

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

Standards (at a high level)

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

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



An example API (OpenSSL)

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

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

Performance:

Crypto++ 5.6.0 [Wei Dai]

AMD Opteron, 2.2 GHz (Linux)

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

* extrapolated from Ted Kravitz's results

** non-Intel machines

End of Segment

Key Derivation

Deriving many keys from one

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

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

Need many keys to secure session:

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

Goal: generate many keys from this one source key



When source key is uniform

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

Suppose source key SK is uniform in K

- Define Key Derivation Function (KDF) as:

KDF(SK , CTX , L) :=

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

CTX: a string that uniquely identifies the application

KDF(SK, CTX, L) :=

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

What is the purpose of CTX?

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

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

It serves no purpose

What if source key is not uniform?

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

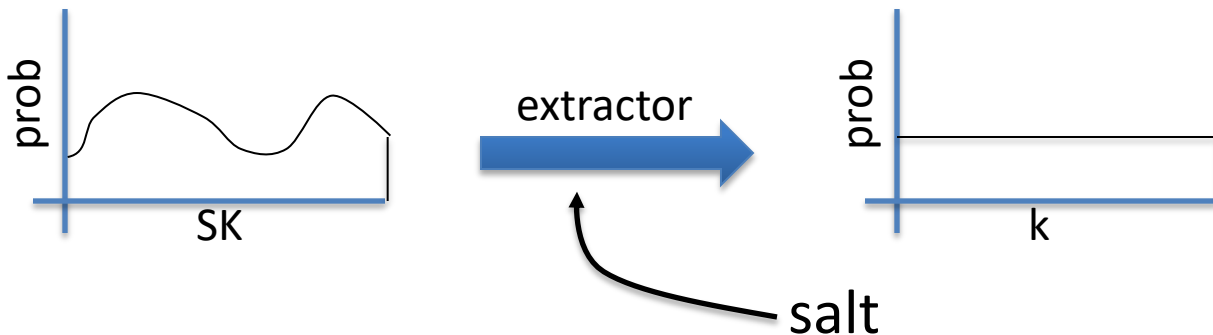
- SK not uniform \Rightarrow PRF output may not look random

Source key often not uniformly random:

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

Extract-then-Expand paradigm

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



salt: a fixed non-secret string chosen at random

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

HKDF: a KDF from HMAC

Implements the extract-then-expand paradigm:

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

Password-Based KDF (PBKDF)

Deriving keys from passwords:

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

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

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

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

End of Segment

Random Oracle Model

Motivation

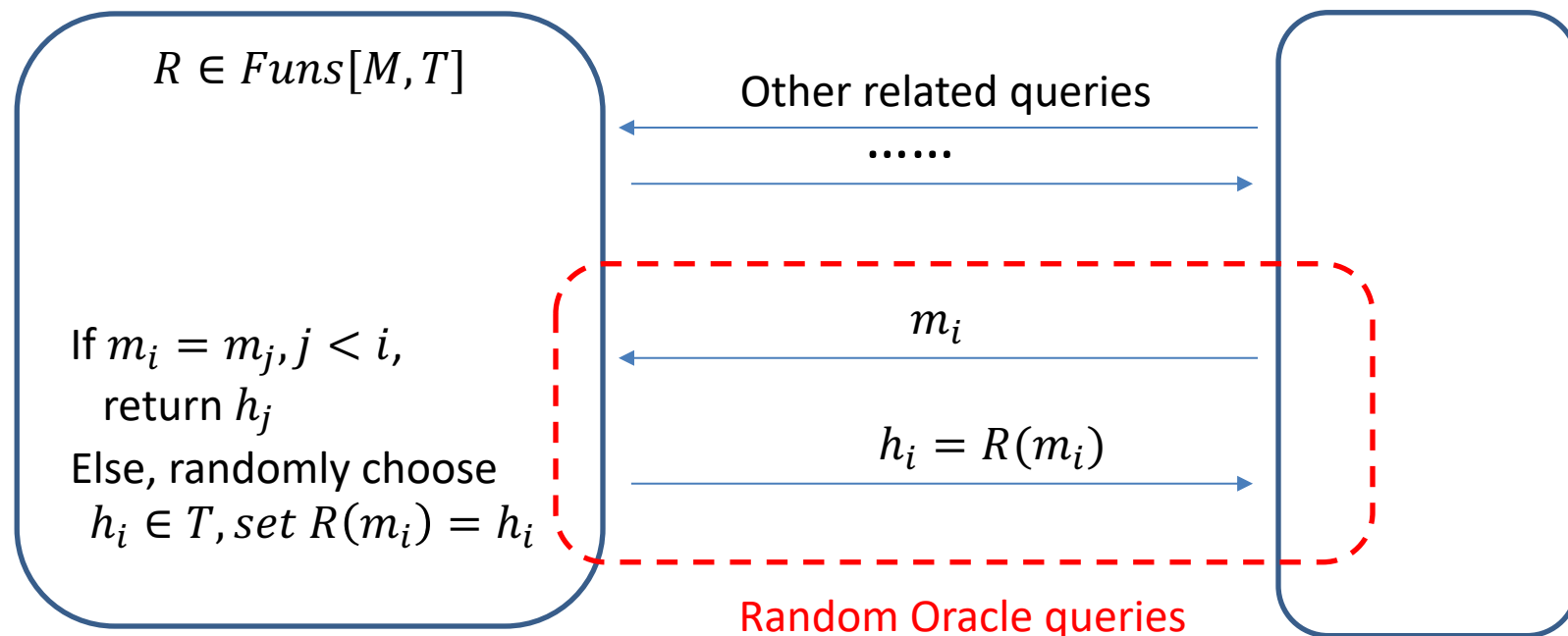
- Hash function is heavily used in many cryptographic applications
- It would be useful to analyze the security if we can simplify the model of hash function H as if it were a truly random function R .
- If H maps from space M to space T , the function R is chosen uniformly at random from the set $\text{Funs}[M,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

Security proof in RO model

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

Challenger

adversary



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

Discussion

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