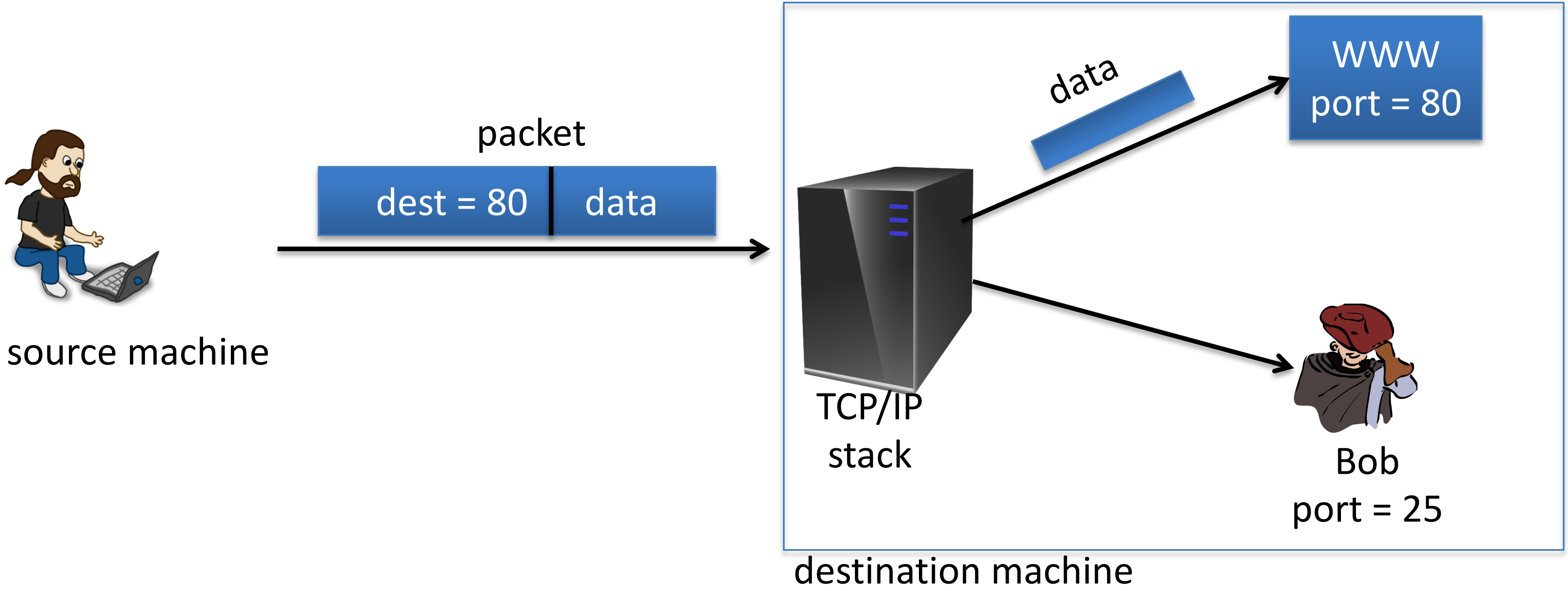
Authenticated Encryption - Active attacks on CPA-secure encryption

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

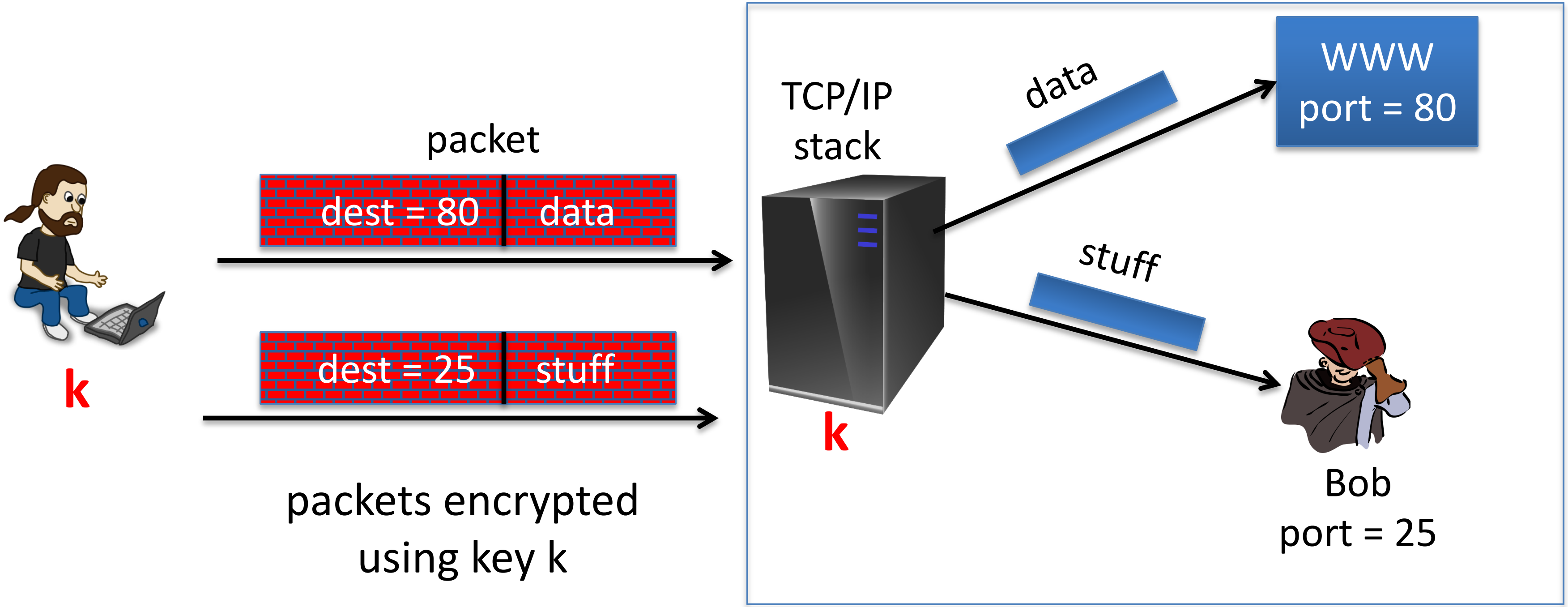
Sample tampering attacks

TCP/IP: (highly abstracted)



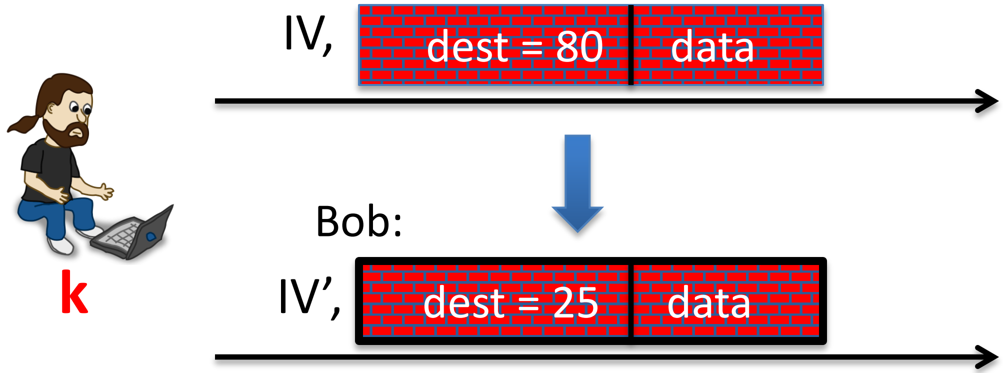
Sample tampering attacks

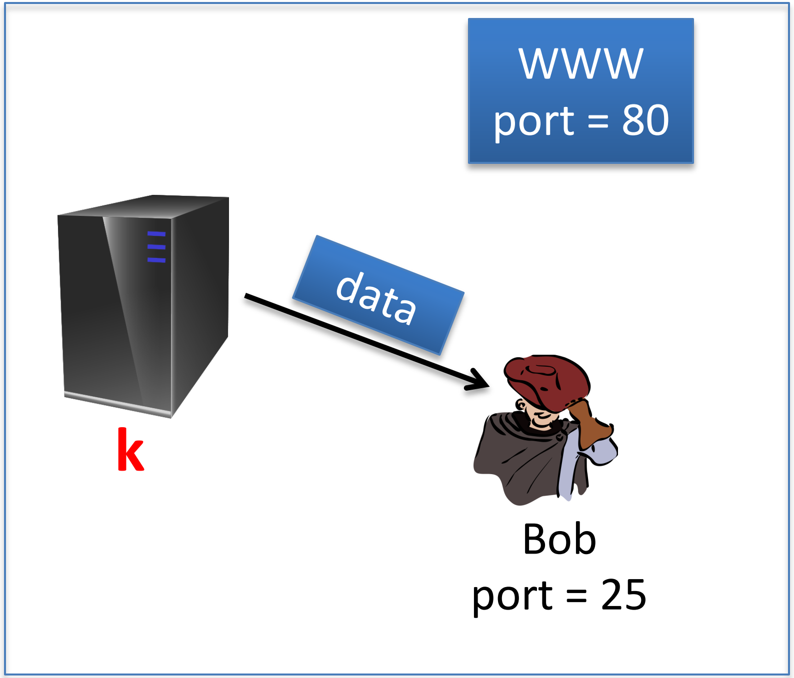
IPsec: (highly abstracted)



Reading someone else's data

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





Easy to do for CBC with rand. IV

(only IV is changed)

Encryption is done with CBC with a random IV.

What should IV' be?

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

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

The lesson

CPA security cannot guarantee secrecy under active attacks.

Only use one of two modes:

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

End of Segment

Authenticated Encryption

Definitions

Goals

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

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

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

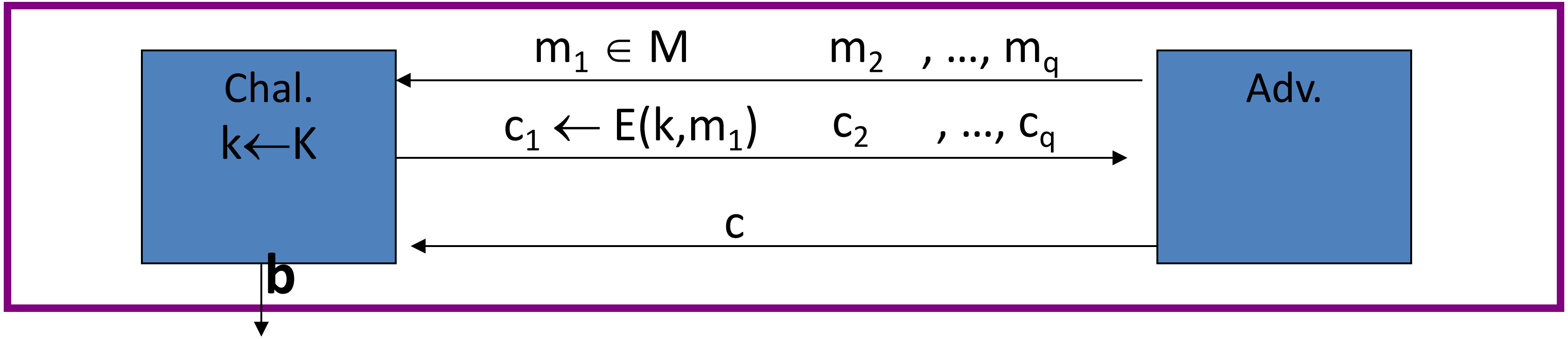
ciphertext is rejected

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.



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

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

$\mathrm{Adv}_{\mathrm{CI}}[A,E]$ = Pr[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

Alice

k

$m_1 , \ldots, \ m_q$

$c_i = E(k, m_i)$

c

Bob

k

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 $\Rightarrow$

Security against **chosen ciphertext attacks**

(next segment)

End of Segment

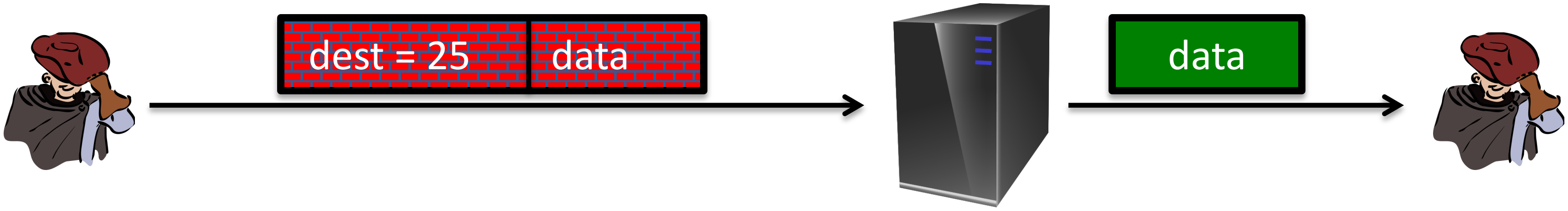
Authenticated Encryption

Chosen ciphertext attacks

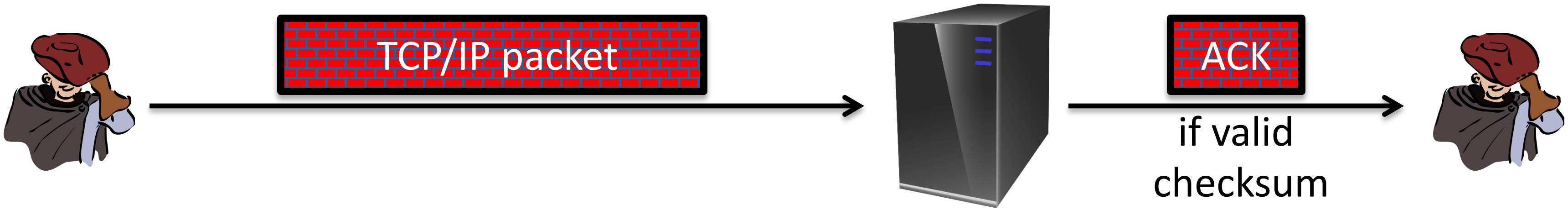
Example chosen ciphertext attacks

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

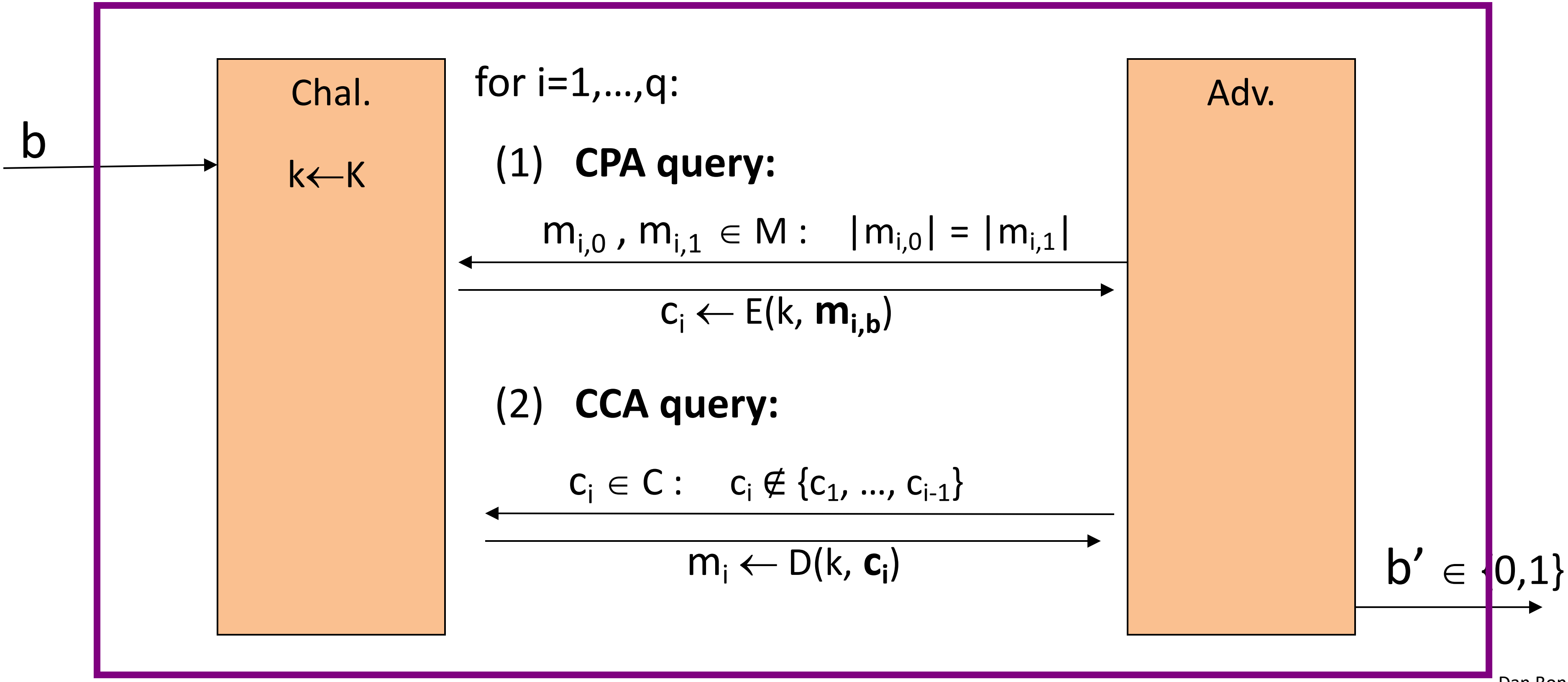
- Can obtain the encryption of arbitrary messages of his choice
- Can decrypt any ciphertext of his choice, other than challenge

(conservative modeling of real life)

Adversary's goal: Break sematic security

Chosen ciphertext security: definition

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

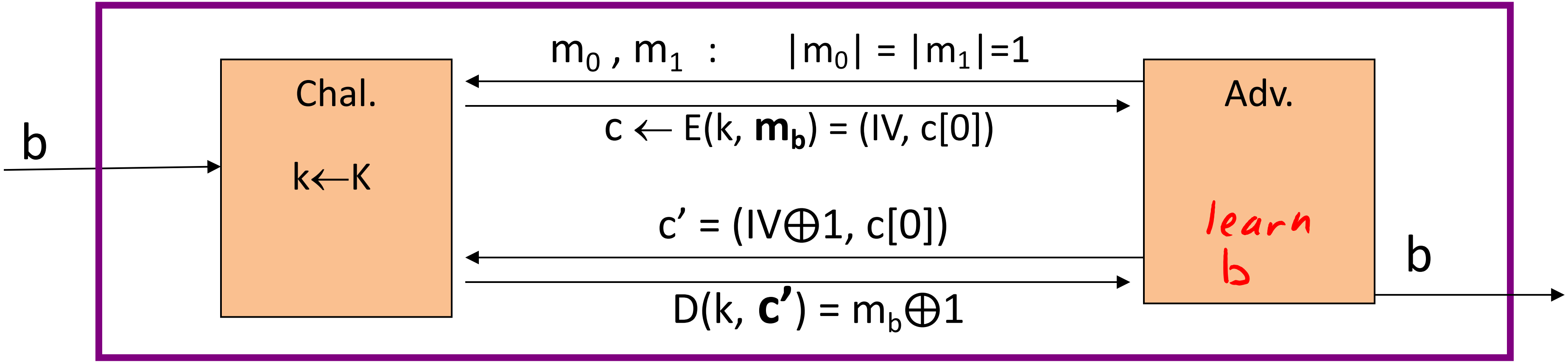


Chosen ciphertext security: definition

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

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

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



Dan Boneh

Authenticated enc. ⇒ CCA security

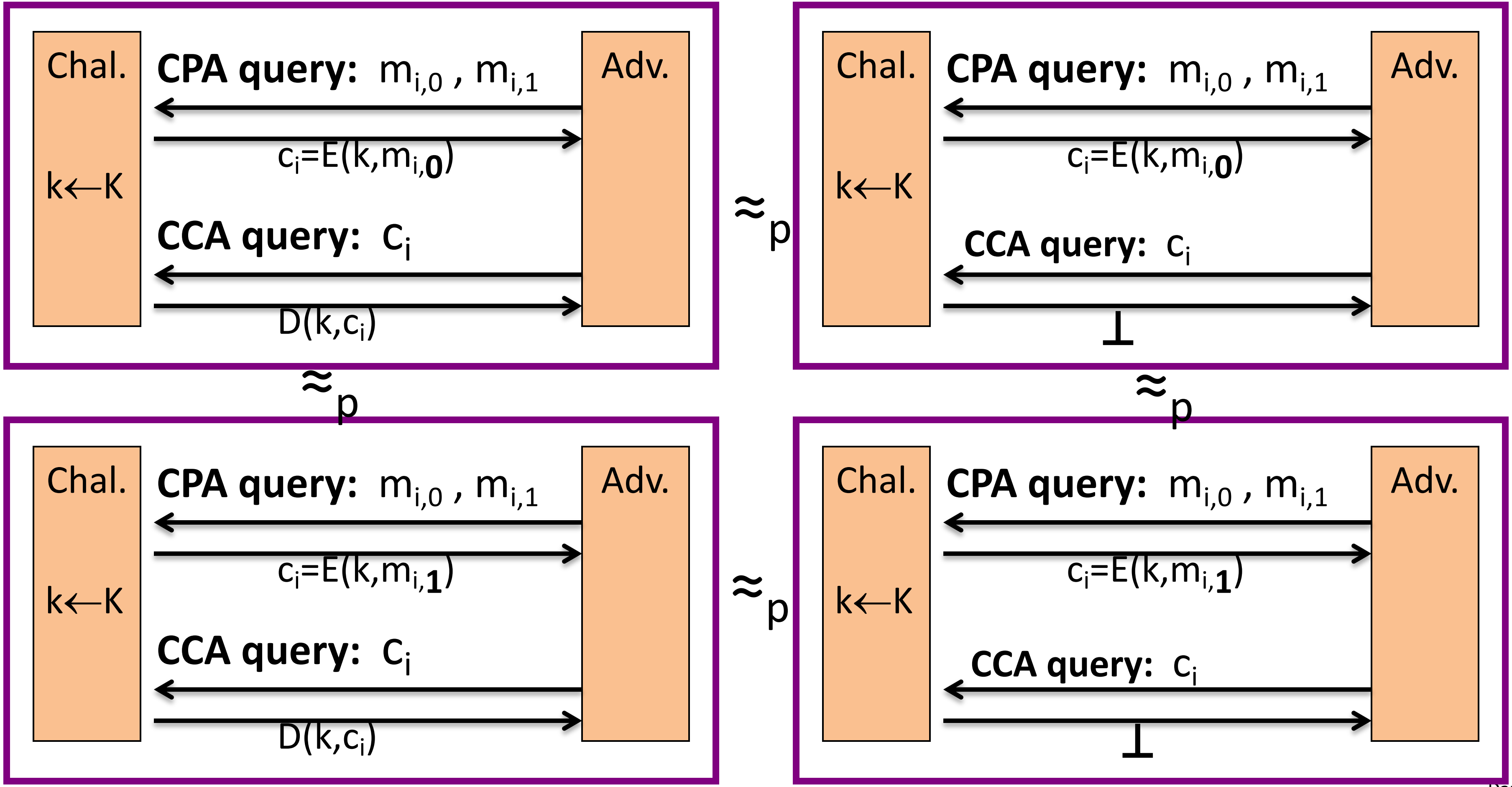
Thm: Let (E,D) be a cipher that provides AE.

Then (E,D) is CCA secure !

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

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

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

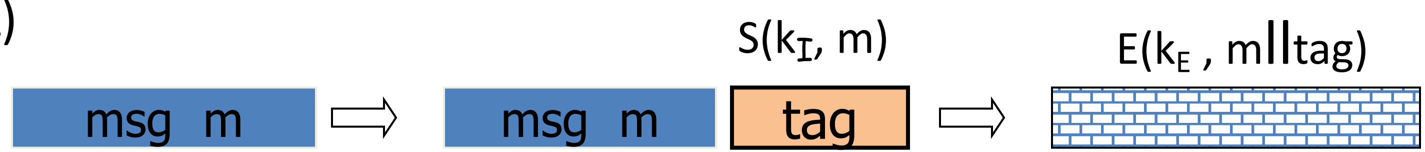
Authenticated Encryption

Constructions from ciphers and MACs

Combining MAC and ENC (CCA)

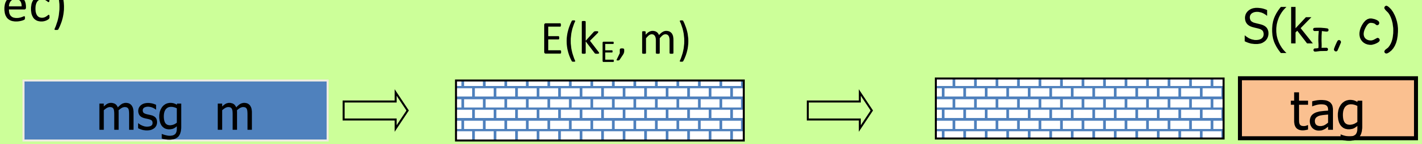
Encryption key k_E. MAC key = k_I

Option 1: (SSL)

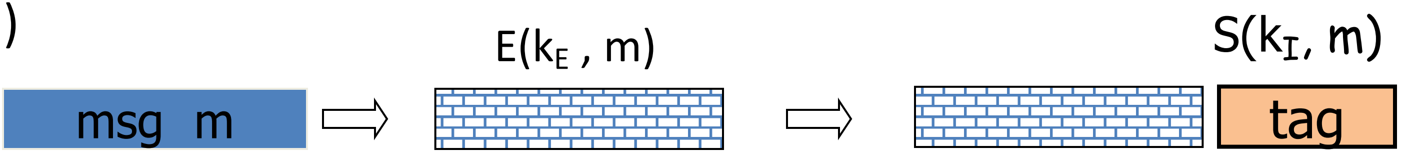


Option 2: (IPsec)

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

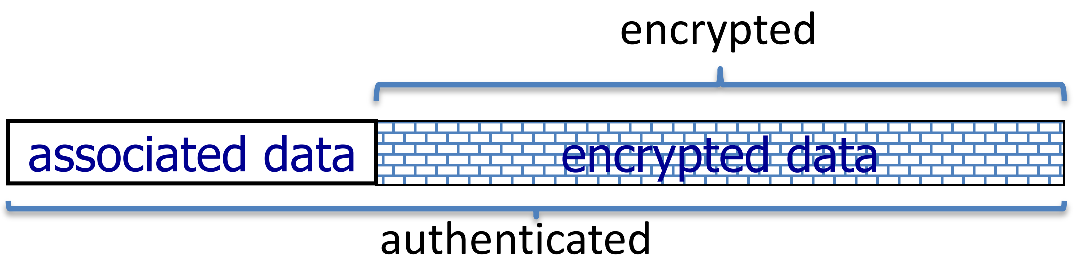
   however: when (E,D) is rand-CTR mode or rand-CBC
   M-then-E provides A.E.

   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)

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

End of Segment

Key Derivation

Deriving many keys from one

Typical scenario. a single <u>source key</u> (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

$$\text{SK} \xrightarrow{\text{KDF}} k_1,\ k_2,\ k_3,\ \dots$$

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) :=

$$F(SK, (\mathbf{CTX \parallel 0})) \parallel F(SK, (\mathbf{CTX \parallel 1})) \parallel \cdots \parallel F(SK, (\mathbf{CTX \parallel L}))$$

CTX: a string that uniquely identifies the application

KDF(SK, CTX, L) :=

$F(SK, (\mathbf{CTX} \parallel \mathbf{0})) \parallel F(SK, (\mathbf{CTX} \parallel \mathbf{1})) \parallel \cdots \parallel F(SK, (\mathbf{CTX} \parallel \mathbf{L}))$

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

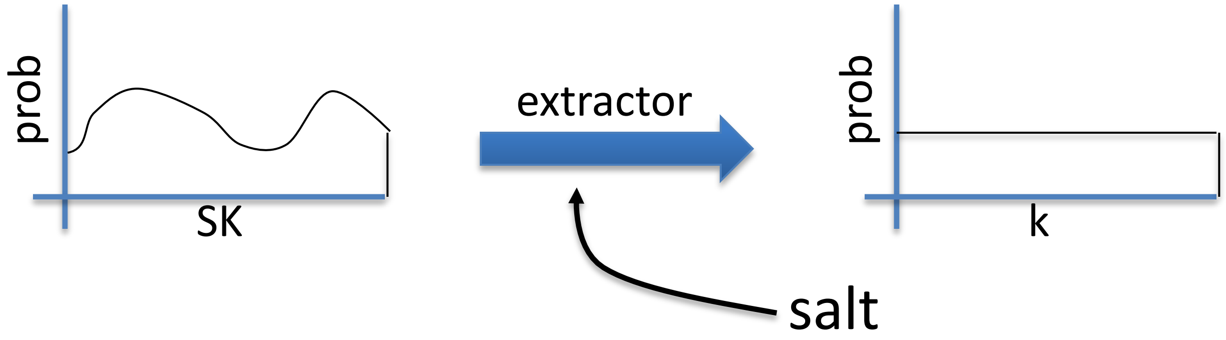
Source key often not uniformly random:

- 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 ⟵ HMAC(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)

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

End of Segment