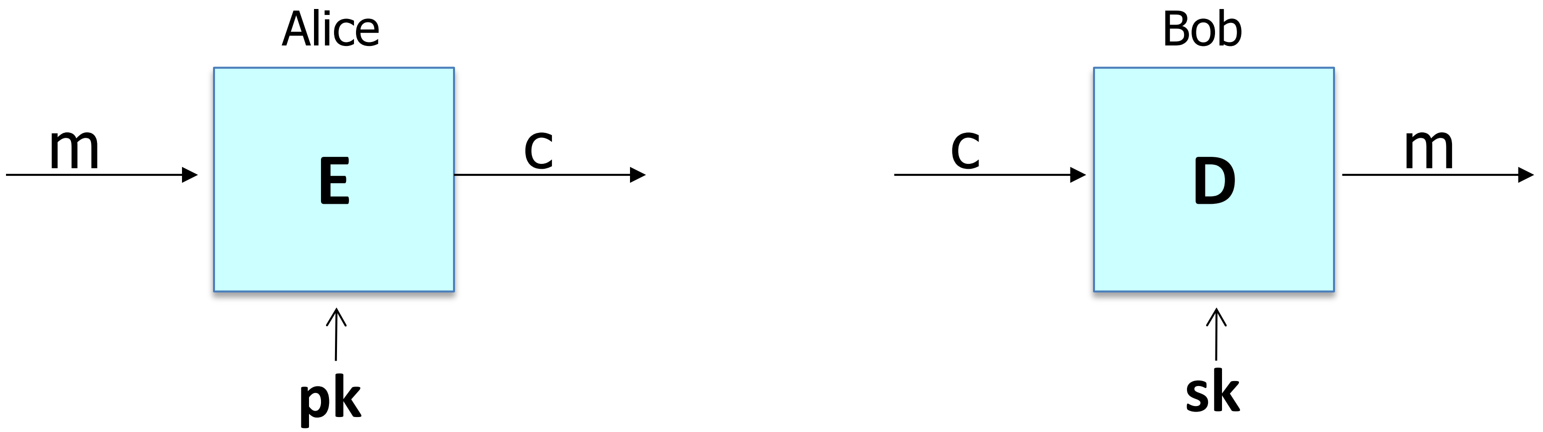
Public Key Encryption from trapdoor permutations

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

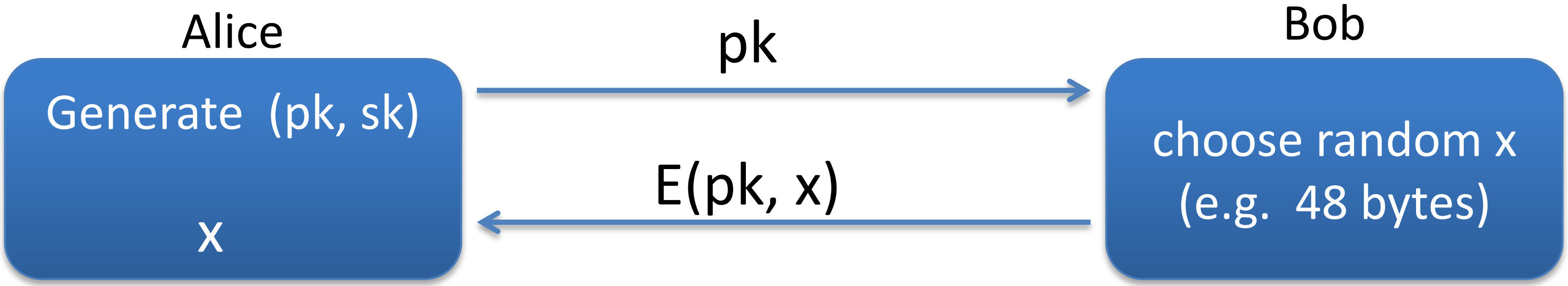
Public key encryption

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)

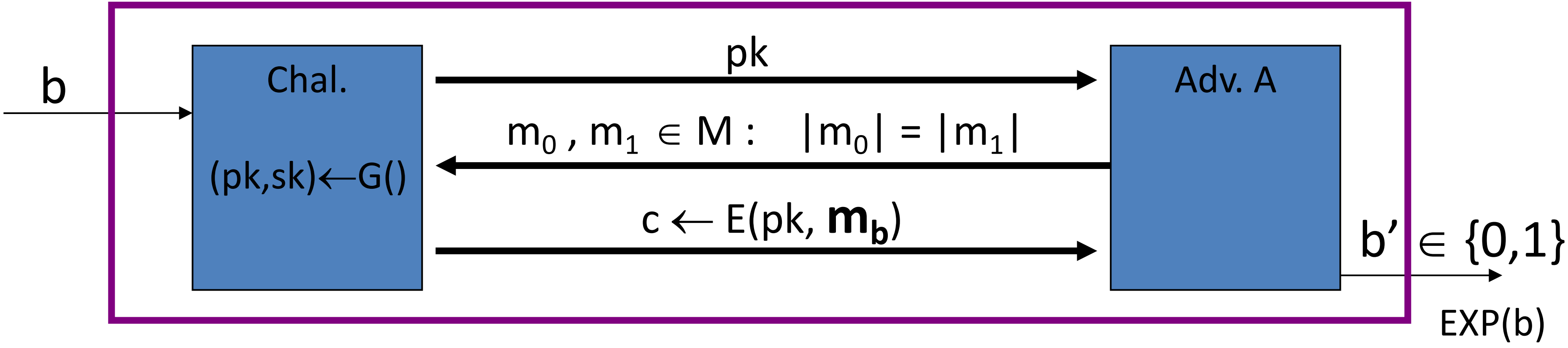
- G(): randomized alg. outputs a key pair (pk, sk)
- E(pk, m): randomized alg. that takes $m \in M$ and outputs $c \in C$
- D(sk,c): det. alg. that takes $c \in C$ and outputs $m \in M$ or $\perp$

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

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

Security: eavesdropping

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



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

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

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:

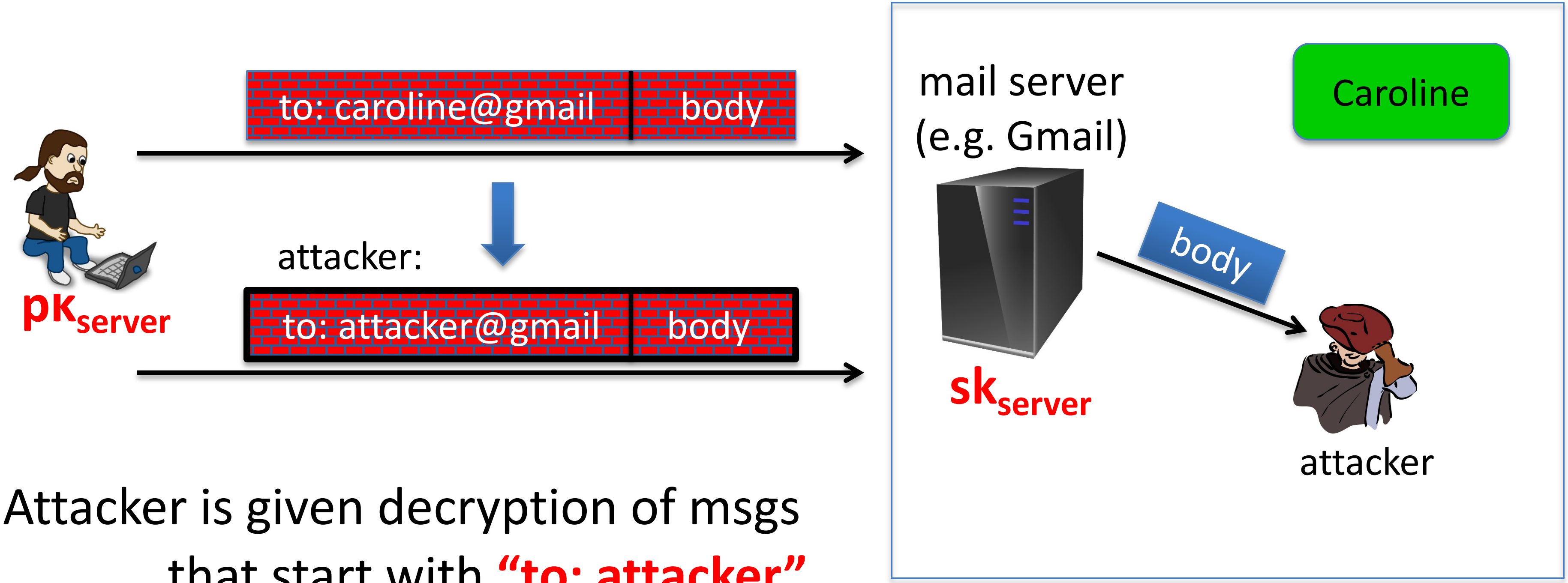
- One-time security $\Rightarrow$ many-time security (CPA)

  (follows from the fact that attacker can encrypt by himself)

- Public key encryption **must** be randomized

Security against active attacks

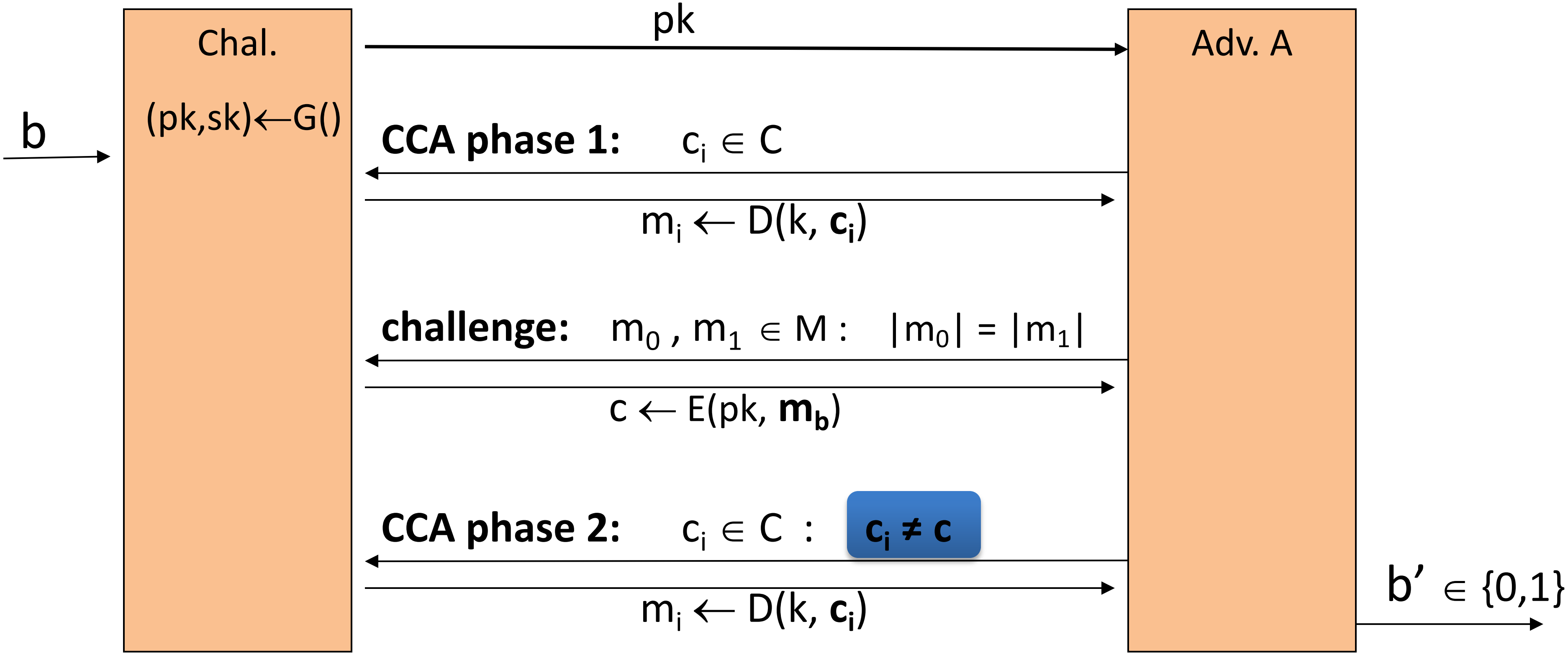
What if attacker can tamper with ciphertext?



Attacker is given decryption of msgs that start with **"to: attacker"**

(pub-key) Chosen Ciphertext Security: definition

$\mathbb{E} = (G,E,D)$ public-key enc. over (M,C). For $b=0,1$ define EXP(b):

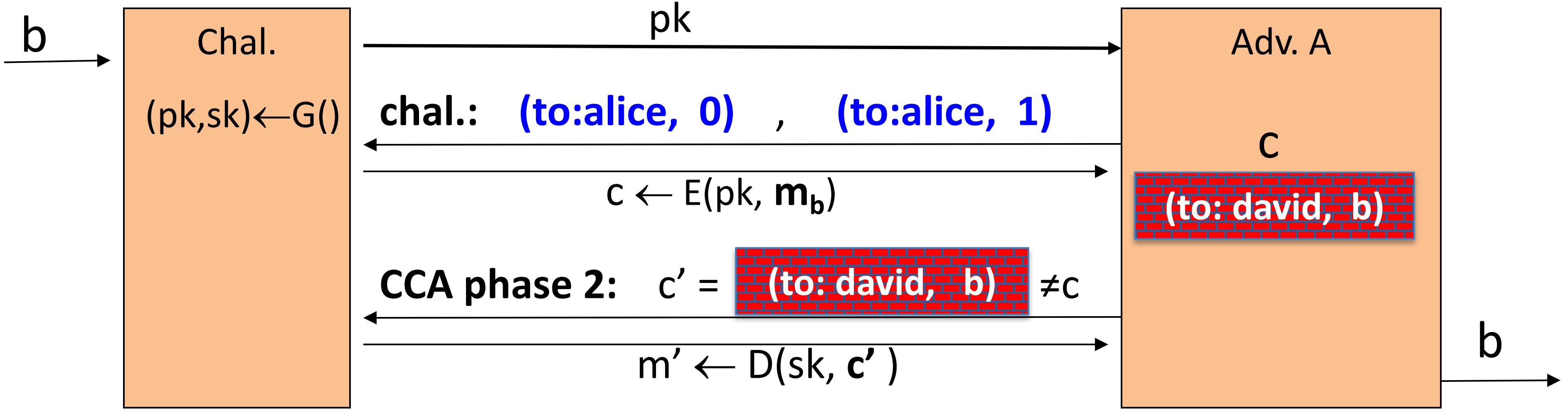


Chosen ciphertext security: definition

Def: $\mathbb{E}$ is CCA secure (a.k.a IND-CCA) if for all efficient A:

$$Adv_{CCA}[A,\mathbb{E}] = \big| \Pr[EXP(0)=1] - \Pr[EXP(1)=1] \big| \quad \text{is negligible.}$$

Example: Suppose **(to: alice, body)** ⟶ **(to: david, body)**



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. $X \longrightarrow Y$ 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 $X \longrightarrow Y$
- $F^{-1}(sk,\cdot)$: defines a function $Y \longrightarrow X$ that inverts $F(pk,\cdot)$

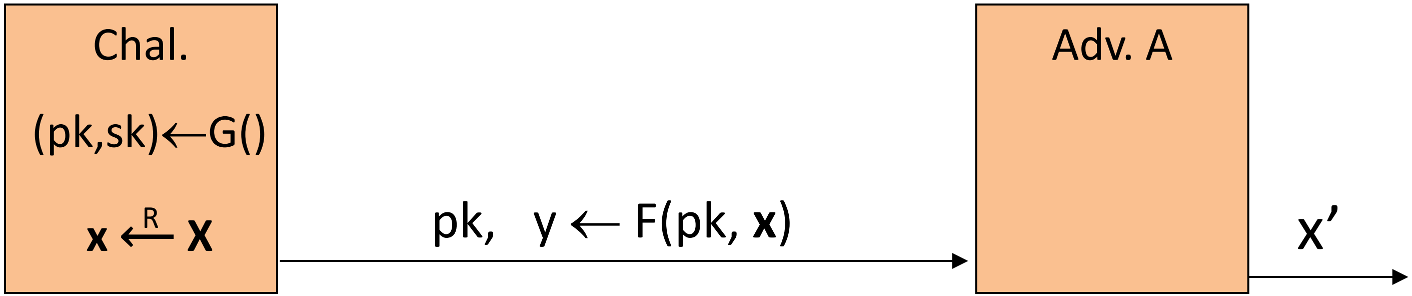
More precisely: $\forall(pk, sk)$ output by G

$$\forall x \in X: \quad F^{-1}(sk, F(pk, x)) = x$$

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:

$$\mathrm{Adv}_{\mathrm{OW}}[A,F] = \Pr[\, x = x' \,] < \text{negligible}$$

Public-key encryption from TDFs

- (G, F, F^{-1}): secure TDF $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ 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 $X \longrightarrow Y$
- (E_s, D_s) : symmetric auth. encryption defined over (K,M,C)
- $H: X \longrightarrow K$ a hash function

E(pk, m) :

$x \xleftarrow{R} X$, $y \longleftarrow F(pk, x)$

$k \longleftarrow H(x)$, $c \longleftarrow E_s(k, m)$

output (y, c)

D(sk, (y,c)) :

$x \longleftarrow F^{-1}(sk, y)$,

$k \longleftarrow H(x)$, $m \longleftarrow D_s(k, c)$

output m

In pictures:

$$\underbrace{F(pk,\ x)}_{\text{header}}\ \underbrace{E_s(\ H(x),\ \ m\ )}_{\text{body}}$$

<u>Security Theorem</u>:

If **(G, F, F^{-1})** is a secure TDF, **(E_s, D_s)** provides auth. enc.

and **H:** $X \longrightarrow K$ 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:

<u>E(pk, m)</u> :

output $c \longleftarrow F(pk, m)$

<u>D(sk, c)</u> :

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 $F(pk, \cdot): X \to X$
- $F(pk, x)$: 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

$Z_N = \{0,1,2,\ldots,N-1\}$; $(Z_N)^* = \{\text{invertible elements in } Z_N\}$

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: $\boxed{\forall\, x \in (Z_N)^* \;:\; x^{\varphi(N)} = 1}$

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 $\mathbf{e}, \mathbf{d}$ s.t. $\mathbf{e \cdot d = 1 \pmod{\varphi(N)}}$

output $pk = (N, e)$, $sk = (N, d)$

F(pk, x): $\mathbb{Z}_N^* \to \mathbb{Z}_N^*$; $\mathbf{RSA(x) = x^e}$ (in Z_N)

$\mathbf{F^{-1}(sk, y)} = y^d$; $y^d = \mathbf{RSA(x)^d} = x^{ed} = x^{k\varphi(N)+1} = \left(x^{\varphi(N)}\right)^k \cdot \mathbf{x} = x$

The RSA assumption

RSA assumption: RSA is one-way permutation

For all efficient algs. A:

$$\Pr\Big[\ A(N,e,y) = y^{1/e}\ \Big] < \text{negligible}$$

where $p,q \xleftarrow{R}$ n-bit primes, $N \leftarrow pq$, $y \xleftarrow{R} Z_N^*$

Review: RSA pub-key encryption (ISO std)

(E_s, D_s): symmetric enc. scheme providing auth. encryption.

H: $Z_N \to K$ 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 $D_s\big(H\big(RSA^{-1}(y)\big), c\big)$

Textbook RSA is insecure

Textbook RSA encryption:

– public key: **(N,e)** Encrypt: $\mathbf{c \longleftarrow m^e}$ (in Z_N)

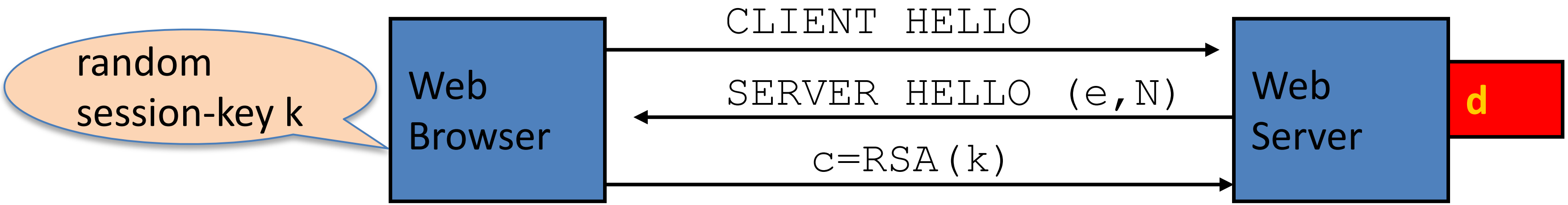
– secret key: **(N,d)** Decrypt: $\mathbf{c^d \longrightarrow m}$

Insecure cryptosystem !!

– Is not semantically secure and many attacks exist

$\Rightarrow$ The RSA trapdoor permutation is not an encryption scheme !

A simple attack on textbook RSA



Suppose k is 64 bits: $k \in \{0,\ldots,2^{64}\}$. Eve sees: $c = k^e$ in Z_N

If $\mathbf{k = k_1 \cdot k_2}$ where $k_1, k_2 < 2^{34}$ (prob. $\approx$20%) then $\mathbf{c/k_1^{\,e} = k_2^{\,e}}$ in Z_N

Step 1: build table: $c/1^e, c/2^e, c/3^e, \ldots, c/2^{34e}$. time: 2^{34}

Step 2: for $k_2 = 0,\ldots, 2^{34}$ 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

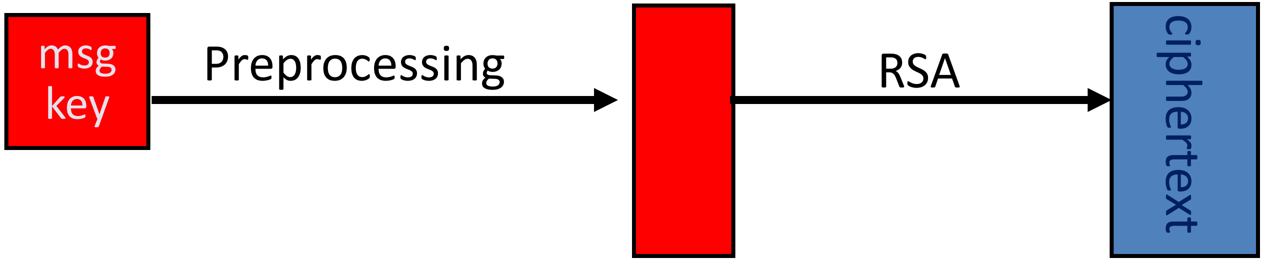
Public Key Encryption from trapdoor permutations

PKCS 1

RSA encryption in practice

Never use textbook RSA.

RSA in practice (since ISO standard is not often used) :

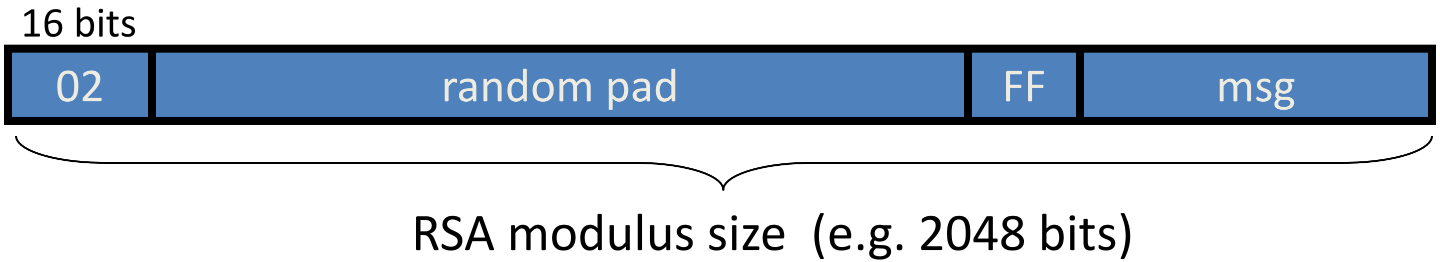


Main questions:

– How should the preprocessing be done?
– Can we argue about security of resulting system?

PKCS1 v1.5

PKCS1 mode 2: (encryption)

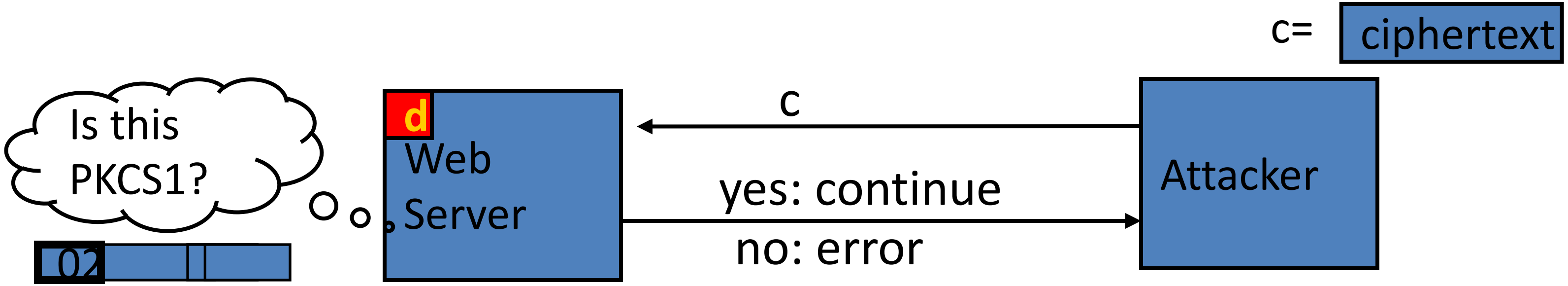


- Resulting value is RSA encrypted

- Widely deployed, e.g. in HTTPS

Attack on PKCS1 v1.5 (Bleichenbacher 1998)

PKCS1 used in HTTPS:



$\Rightarrow$ attacker can test if 16 MSBs of plaintext = ’02’

Chosen-ciphertext attack: to decrypt a given ciphertext c do:

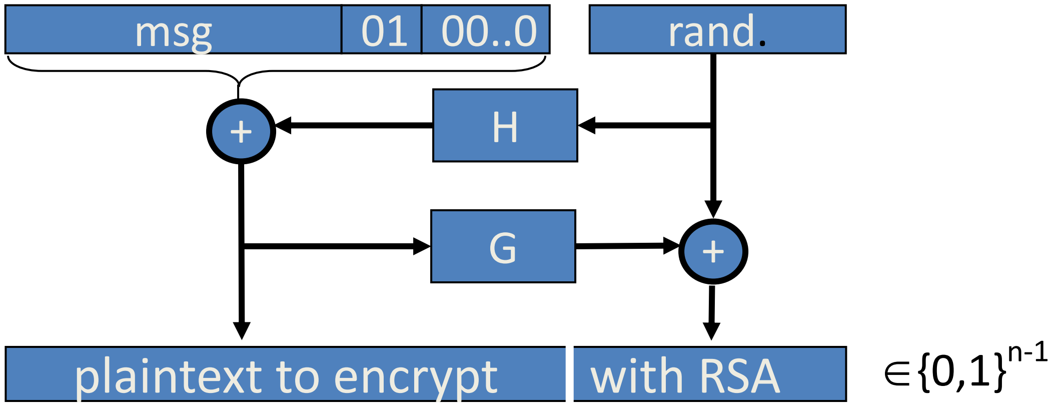
– Choose $r \in Z_N$. Compute $c' \leftarrow r^e \cdot c \;=\; (r \cdot \mathrm{PKCS1}(m))^e$

– Send c’ to web server and use response

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

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:

<u>Cipher key-size</u>	RSA <u>Modulus size</u>
80 bits	1024 bits
128 bits	3072 bits
256 bits (AES)	**<u>15360</u>** 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

$$\left.\begin{array}{ll} \text{decrypt mod p:} & x_p = c^d \text{ in } Z_p \\ \text{decrypt mod q:} & x_q = c^d \text{ in } Z_q \end{array}\right\} \text{combine to get } x = c^d \text{ in } Z_N$$

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 $\Rightarrow$ $\gcd\left((x')^e - c , N \right) = p$

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