CS 4770: Cryptography

CS 6750: Cryptography and Communication Security

Alina Oprea
Associate Professor, CCIS
Northeastern University

March 22 2017

Outline

- Public-key encryption
  - Definition
  - Security notions: CPA, CCA
- Trapdoor functions
  - Construct public-key encryption from trapdoor functions
- Constructions of trapdoor functions
  - RSA trapdoor and encryption scheme
- RSA encryption in practice
  - PKCS, OAEP standards
  - Attacks on RSA

The Diffie-Hellman protocol

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}

$$p, g, A \leftarrow g^a \bmod p \longrightarrow$$

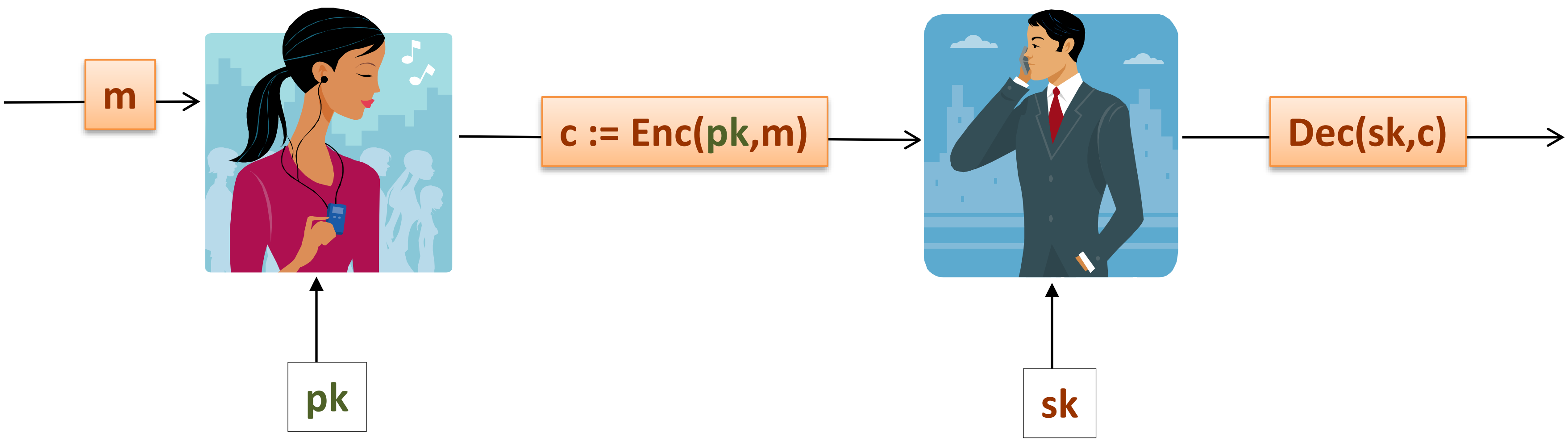
$$\longleftarrow B \leftarrow g^b \bmod p$$

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

Public-key encryption

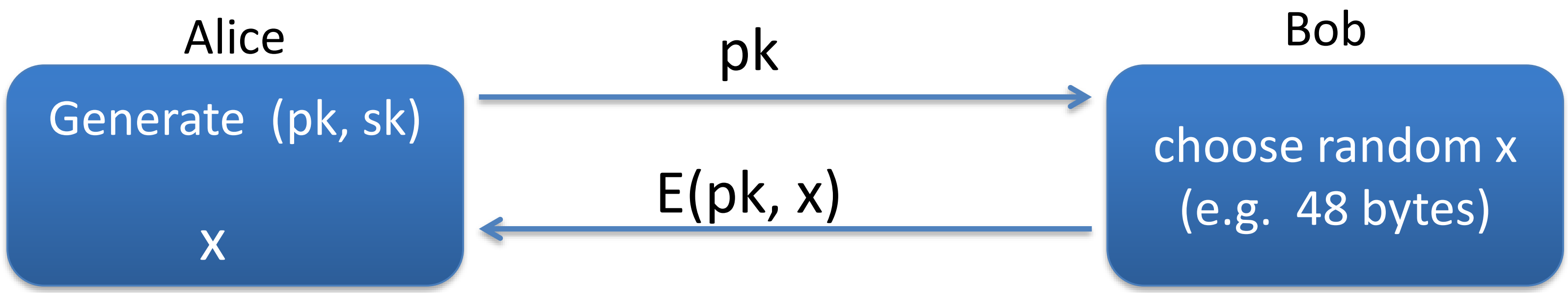
Instead of using one key **k**,
use **2** keys **(pk,sk)**, where

pk - public key used for **encryption**,
sk – secret key used for **decryption**.



Applications

Key exchange (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

Definition: a public-key encryption system is a triple of algs. (Gen, Enc, Dec)

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

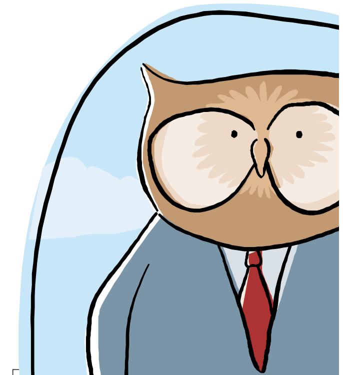
Correctness: $\forall$(pk, sk) output by G :

$$\forall m \in M: \quad \text{Dec}(sk, \text{Enc}(pk, m)) = m$$

CPA Security Game – Public key

Π= (Enc,Dec): an encryption scheme

security parameter
n



PPT Adversary A

Challenger

pk

chooses m_0, m_1 such that $|m_0|=|m_1|$

m_0, m_1

1. **(pk,sk) ← Gen(n)**
2. chooses random **b** ← **{0,1}**
3. calculate **c** ← **Enc(pk,m_b)**

c

Makes a guess **b'**

Security definition:
We say that **(Enc,Dec)** is **CPA-secure** if any **polynomial time** adversary,
| Pr[b'=b] - ½ | is negligible in n.

CPA security definition

- Experiment $\mathrm{Exp}_{\Pi,A}^{\mathrm{CPA}}(n)$:
  1. Choose $(pk, sk) \leftarrow^R Gen(1^n)$
  2. $m_0, m_1 \leftarrow A_1\ (pk)$
  3. $b \leftarrow^R \{0,1\}; c \leftarrow Enc_{pk}(m_b)$
  4. $b' \leftarrow A_2\ (pk, m_0, m_1, c)$
  5. Output 1 if $b = b'$ and 0 otherwise

We say that **(Enc,Dec)** is **chosen-plaintext attack (CPA) secure** if

For every **PPT** adversary $A = (A_1, A_2)$:

|Pr[$\mathrm{Exp}_{\Pi,A}^{\mathrm{CPA}}(n)$ = 1]- ½ | negligible in n

Relation to symmetric cipher security

Recall: for symmetric ciphers we had two security notions:

- EAV security and CPA security
- We showed that CPA security is strictly stronger than EAV security

For public key encryption:

- EAV security $\Rightarrow$ CPA security

follows from the fact that attacker can encrypt by himself

- Public key encryption **must** be randomized

CCA security definition

- Experiment $\text{Exp}_{\Pi,A}^{\text{CCA}}(n)$:
  1. Choose $(pk, sk) \leftarrow^{R} Gen(1^n)$
  2. $m_0, m_1 \leftarrow A_1^{Dec_{sk}(\cdot)}(pk)$
  3. $b \leftarrow^{R} \{0,1\}; c \leftarrow Enc_{pk}(m_b)$
  4. $b' \leftarrow A_2^{Dec_{sk}(\cdot)}(m_0, m_1, c)$
  5. Output 1 if $b = b'$ and 0 otherwise

Adversary can not submit c to decryption oracle

We say that **(Enc,Dec)** is **chosen-ciphertext attack (CCA) secure** if

For every **PPT** adversary $A = (A_1, A_2)$:

|Pr[$\text{Exp}_{\Pi,A}^{\text{CCA}}(n)$ = 1]- ½ | negligible in n

Construct public-key encryption from trapdoor functions

Trapdoor functions (TDF)

Def: a trapdoor func. $X \longrightarrow Y$ is a triple of efficient algorithms (Gen, F, F^{-1})

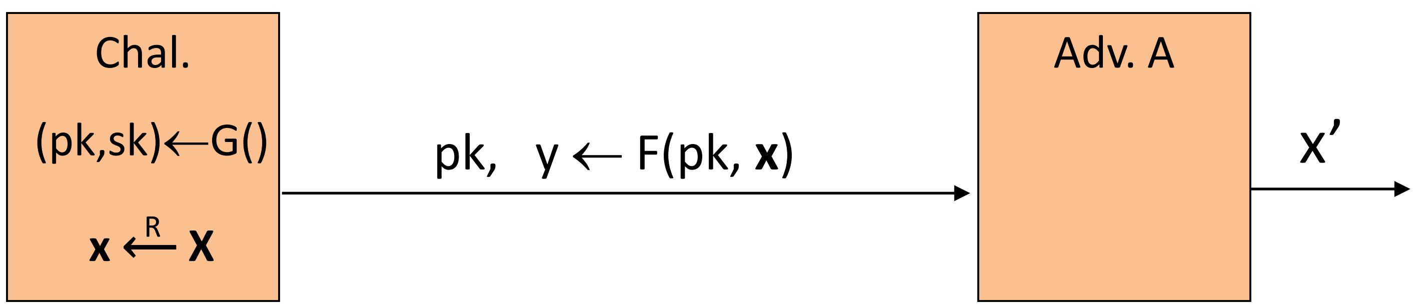
- Gen(): randomized alg. outputs a key pair (pk, sk)
- $F(pk,\cdot)$: deterministic alg. that defines a function $X \longrightarrow Y$
- $F^{-1}(sk,\cdot)$: defines a function $Y \longrightarrow X$ that inverts $F(pk,\cdot)$

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

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

Secure Trapdoor Functions (TDFs)

(Gen, F, F^{-1}) is secure if $F(pk, \cdot)$ is a "one-way" function: can be evaluated, but cannot be inverted without sk



Def: (Gen, F, F^{-1}) is a secure TDF if for all PPT A:

$$\Pr[\, x \leftarrow X,\ y = F(pk,x),\ x' \leftarrow A(pk,y),\ x = x' \,]$$

is negligible

Construct trapdoor functions

RSA trapdoor

Review: arithmetic mod composites

Let $N = p \cdot q$ where p,q are prime

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

<u>Facts</u>: $x \in Z_N$ is invertible $\Leftrightarrow$ $\gcd(x,N) = 1$

– Number of elements in $(Z_N)^*$ is $\varphi(N) = (p-1)(q-1) = N-p-q+1$

<u>Euler's theorem</u>: $\boxed{\forall\, x \in (Z_N)^* \; : \; x^{\varphi(N)} = 1 \bmod N}$

The RSA trapdoor permutation

First published: Scientific American, Aug. 1977

R. Rivest, A. Shamir, and L. Adelman

Very widely used:

- SSL/TLS: certificates and key-exchange
- Secure e-mail and file systems

  … many others

The RSA trapdoor permutation

Gen(): Choose random primes $p,q \approx 1024$ bits.

Set **N=pq**. **RSA modulus**

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

Output pk = (N, e) , sk = (d)

F(pk, x): $\mathbb{Z}_N^* \to \mathbb{Z}_N^*$; **F(pk, x) = x^e mod N**

F^{-1}(sk, y) = y^d mod N

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

The RSA assumption

RSA assumption: RSA is trapdoor permutation

For all PPT algorithms A:

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

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

Textbook RSA is insecure

Textbook RSA encryption:

- public key: **(N,e)**
- secret key: **(N,d)**

Encrypt: $\mathbf{c \longleftarrow m^e \bmod N}$

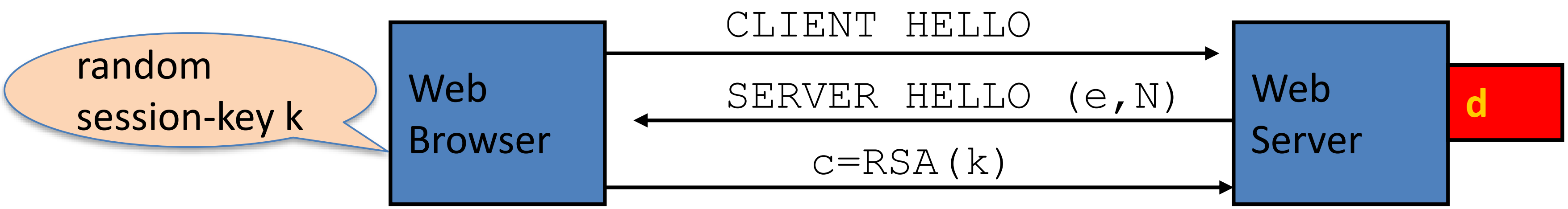
Decrypt: $\mathbf{c^d \longrightarrow m \bmod N}$

Insecure cryptosystem !!

- Is not semantically secure and many attacks exist
- Deterministic encryption

⇒ The RSA trapdoor permutation is not an encryption scheme !

Attack 1: Meet in the middle



Suppose k is 64 bits: $k \in \{0,\dots,2^{64}-1\}$. Eve sees: $c = k^e$ in Z_N

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

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

Step 2: for $k_2 = 0,\dots, 2^{34}-1$ test if k_2^e is in table. time: 2^{34}

Output matching (k_1, k_2). Total attack time: $\approx 2^{40} \ll 2^{64}$

Attack 2: Small messages

- Encrypt: $\mathbf{c \longleftarrow m^e \bmod N}$
- Use small exponents (e.g., $e = 3$)
- Assume that m is small
  - m is 300 bits, N is 1024 bits
  - $m^e < N \Rightarrow c = m^e$ over integers
  - Then can compute $m = c^{1/e}$ over integers
- Finding e-th roots is easy over integers, but hard mod N, for N=pq

Attack 3: Small decryption exponent

To speed up RSA decryption use small private key d ($d \approx 2^{128}$)

$$\boxed{c^d = m \pmod{N}}$$

Wiener'87: If $d < N^{0.25}$ then RSA is insecure.

BD'98: If $d < N^{0.292}$ then RSA is insecure (open: $d < N^{0.5}$)

<u>Insecure:</u> private key d can be found from (N,e)

Attack 4: RSA with related public keys

- Assume 2 users share the same module N
  - Public keys (N,e_1) and (N,e_2) with $\gcd(e_1,e_2) = 1$
- Same message m encrypted under both keys
  - Adversary sees $c_1 = m^{e_1} \bmod N$; $c_2 = m^{e_2} \bmod N$
- Attacker can recover m
  - e_1 and e_2 are public => there exists X and Y such that $e_1X + e_2Y = 1$ (X and Y can be found with extended Euclidian algorithm)
  - $c_1^X c_2^Y = m^{e_1X} m^{e_2Y} = m \bmod N$
- Morale: do not reuse the same RSA modulus for multiple keys

RSA public-key encryption

(E, D): authenticated encryption scheme

H: $Z_N \to K$ where K is key space of (E_s, D_s)

- **Gen**(): generate RSA parameters:
  pk = (N,e), sk = (d)

- **Enc**(pk, m): (1) choose random x in Z_N
  (2) $y \leftarrow \text{RSA}(x) = x^e$, $k \leftarrow H(x)$
  (3) output (y , E(k,m))

- **Dec**(sk, (y, c)): output D($H(\text{RSA}^{-1}(y))$, c)

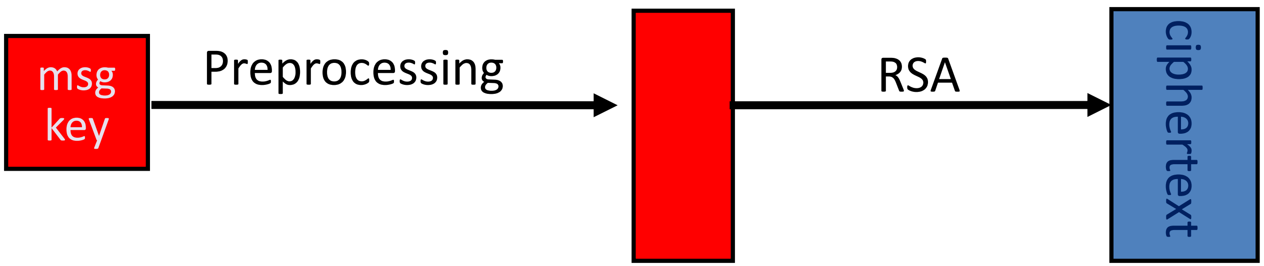
CCA secure

ISO Standard

RSA encryption in practice

Never use textbook RSA.

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

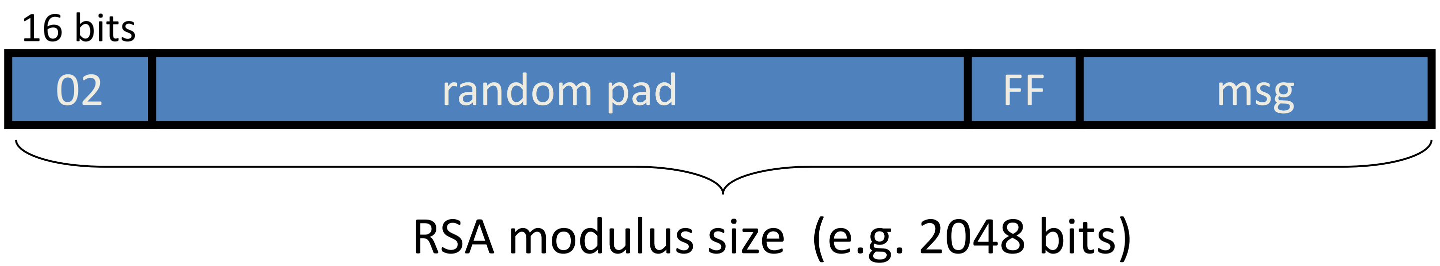


Main questions:

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

PKCS1 v1.5

PKCS1 mode 2: (encryption)

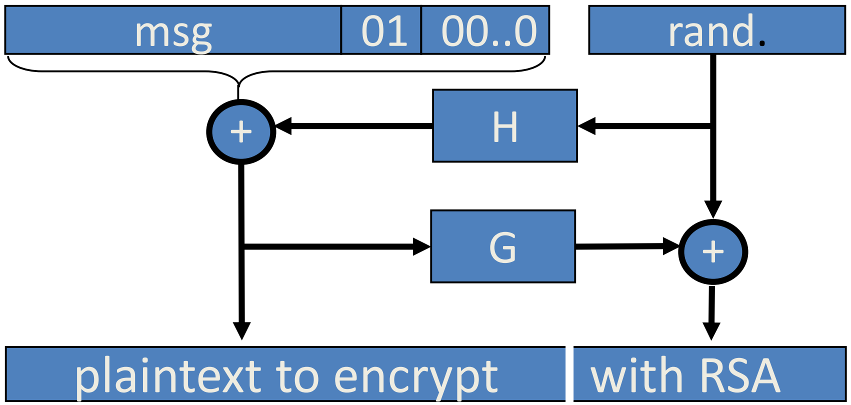


- Resulting value is RSA encrypted
- Widely deployed, e.g. in HTTPS

PKCS1 v2.0: OAEP

New preprocessing function: OAEP [BR94]

check pad
on decryption.
reject CT if invalid.



$\in \{0,1\}^{n-1}$

Theorem [FOPS'01] : RSA is a trapdoor permutation $\Rightarrow$
RSA-OAEP is CCA secure when H,G are *random functions*

in practice: use SHA-256 for H and G

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

Key insights

- CCA security is the desired notion of security for public-key encryption to handle active attackers
  - CPA security is equivalent to EAV security
- RSA trapdoor
  - Relies on hardness of computing e-th roots mod N
- CCA secure public-key encryption can be constructed from trapdoor permutations
  - Trapdoor permutations (e.g., RSA) are not by themselves secure encryption schemes
  - Need to use a method to transform them to CCA-secure encryption (ISO standard, OAEP)

Acknowledgement

Some of the slides and slide contents are taken from
http://www.crypto.edu.pl/Dziembowski/teaching
and fall under the following:

We have also used slides from Prof. Dan Boneh online cryptography course at Stanford University:

http://crypto.stanford.edu/~dabo/courses/OnlineCrypto/