

Lecture 8

Public-Key Encryption I

Stefan Dziembowski
www.crypto.edu.pl/Dziembowski

University of Warsaw



Is the “handbook **RSA**” secure?

On the last lecture we saw that it has **some weaknesses** (homomorphism, determinism...)

Plan for today:

1. Provide a formal security definition.
2. Modify **RSA** so that it is secure according to this definition.

Plan



1. Definition of the **CPA security**
2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
3. The **hybrid encryption** and the **KEM/DEM** paradigm
4. Definition of the **CCA security**
5. Constructions of **CCA-secure** symmetric encryption
6. Constructions of **CCA-secure RSA** encryption schemes

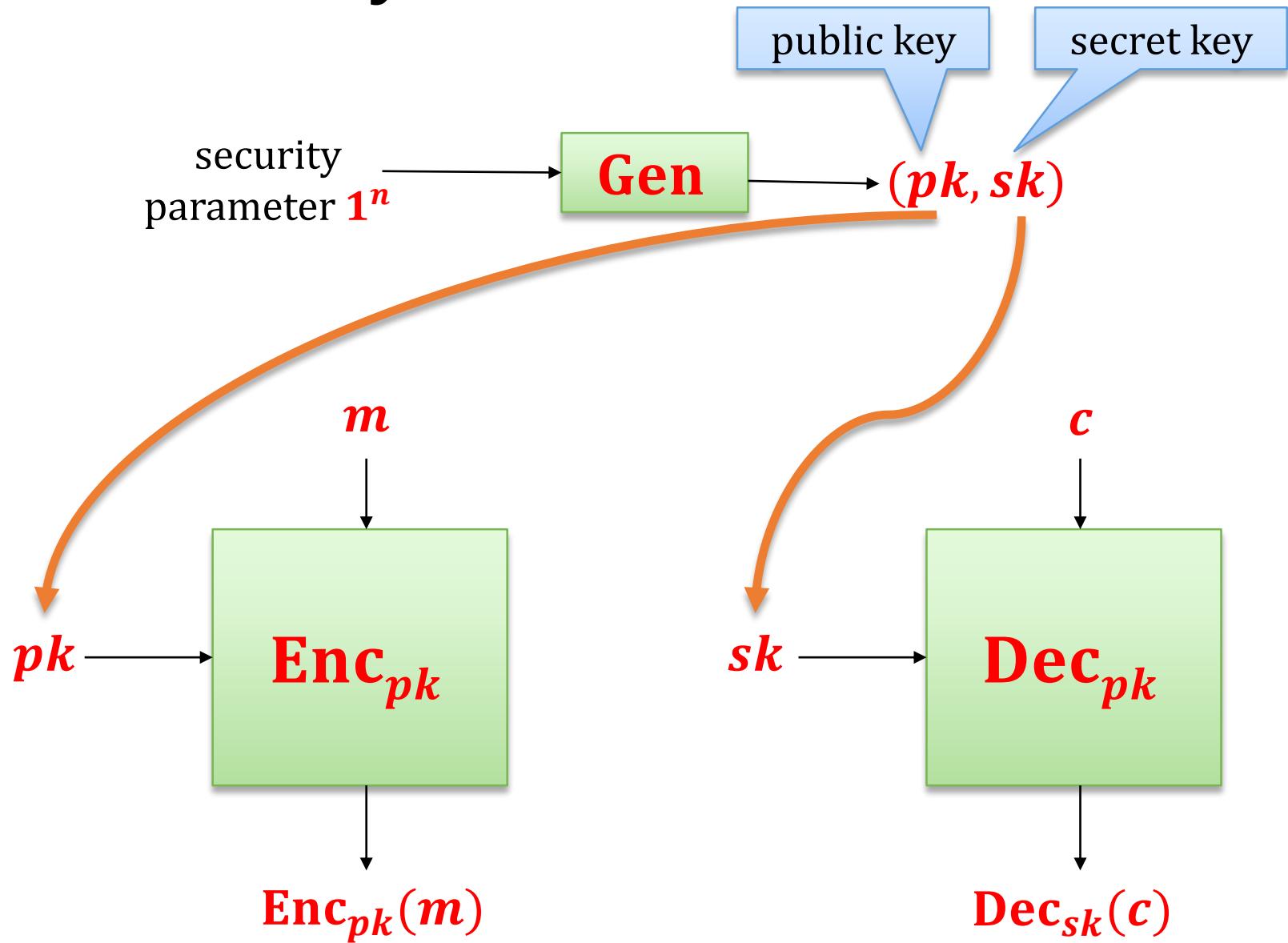
A mathematical view

A **public-key encryption (PKE)** scheme is a triple $(\mathbf{Gen}, \mathbf{Enc}, \mathbf{Dec})$ of poly-time algorithms, where

- **Gen** is a **key-generation** randomized algorithm that takes as input a security parameter $\mathbf{1}^n$ and outputs a key pair $(\mathbf{pk}, \mathbf{sk}) \in (\{0, 1\}^*)^2$.
- **Enc** is an **encryption** algorithm that takes as input the **public key** \mathbf{pk} and a **message** \mathbf{m} (from some set that **may depend** on \mathbf{pk}), and outputs a **ciphertext** \mathbf{c} ,
- **Dec** is a **decryption** algorithm that takes as input the **private key** \mathbf{sk} and the **ciphertext** \mathbf{c} , and outputs a **message** $\mathbf{m}' \in \{0, 1\}^* \cup \{\perp\}$.

We will sometimes write $\mathbf{Enc}_{\mathbf{pk}}(\mathbf{m})$ and $\mathbf{Dec}_{\mathbf{sk}}(\mathbf{c})$ instead of $\mathbf{Enc}(\mathbf{pk}, \mathbf{m})$ and $\mathbf{Dec}(\mathbf{sk}, \mathbf{c})$.

Pictorially



Correctness

We will require that it always holds that

$$P(\text{Dec}_{sk}(\text{Enc}_{pk}(m)) \neq m) \text{ is negligible in } n$$

assuming that:

- $(pk, sk) \leftarrow \text{Gen}(1^n)$
- and m is a “legal” plaintext for pk .

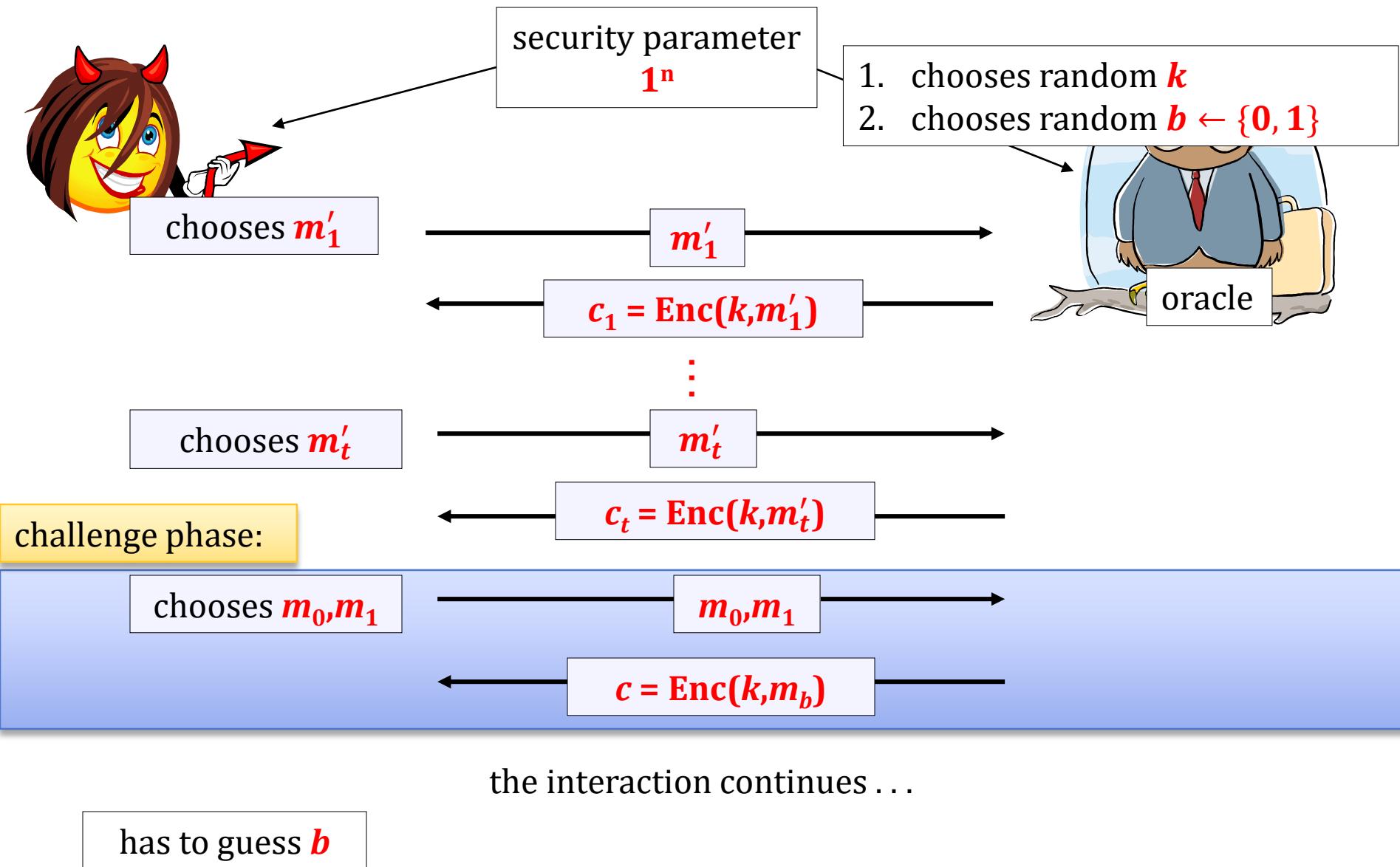
The security definition

Remember the symmetric-key case?

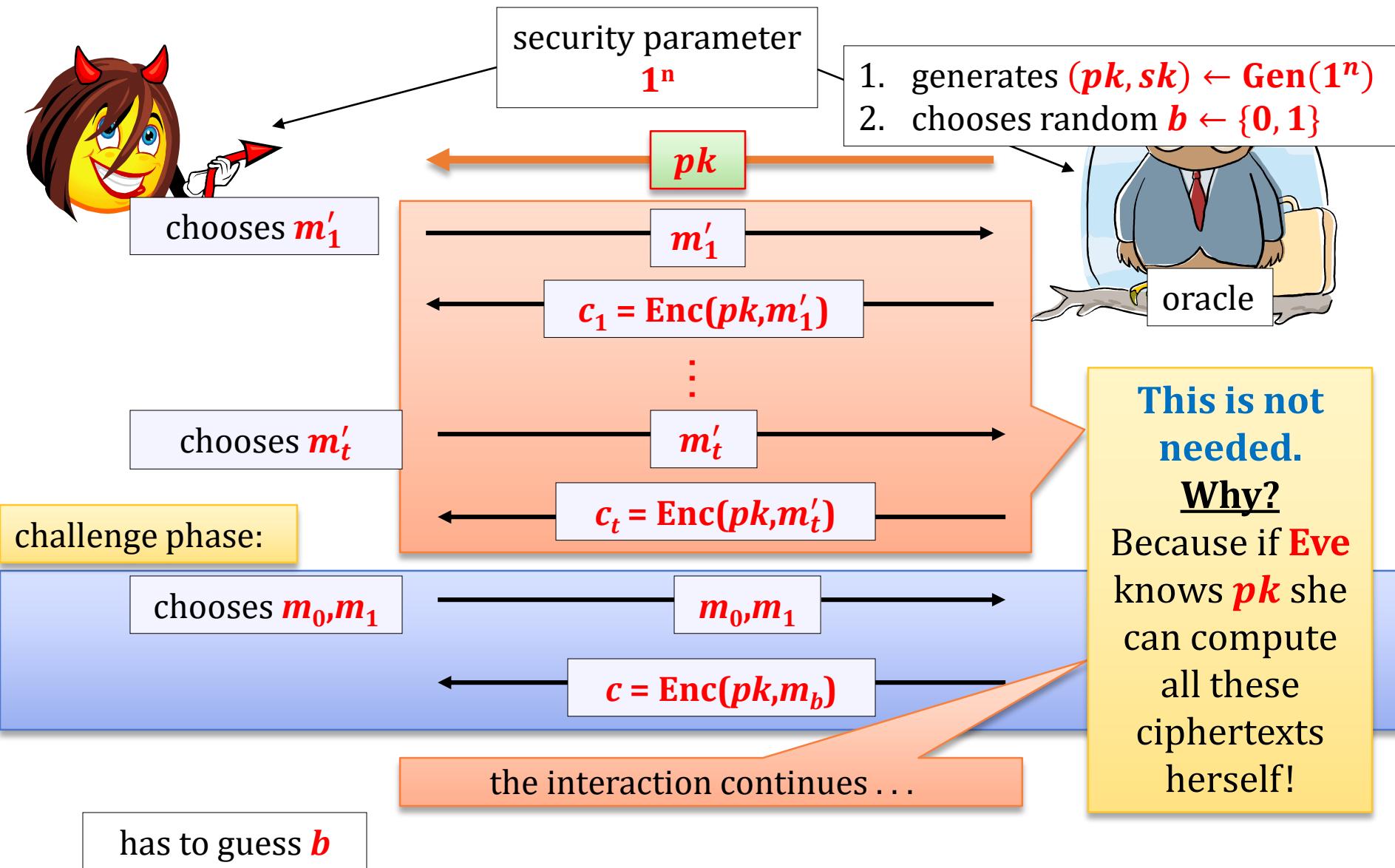
We considered a **chosen-plaintext attack**.

How would it look in the case of the **public-key encryption**?

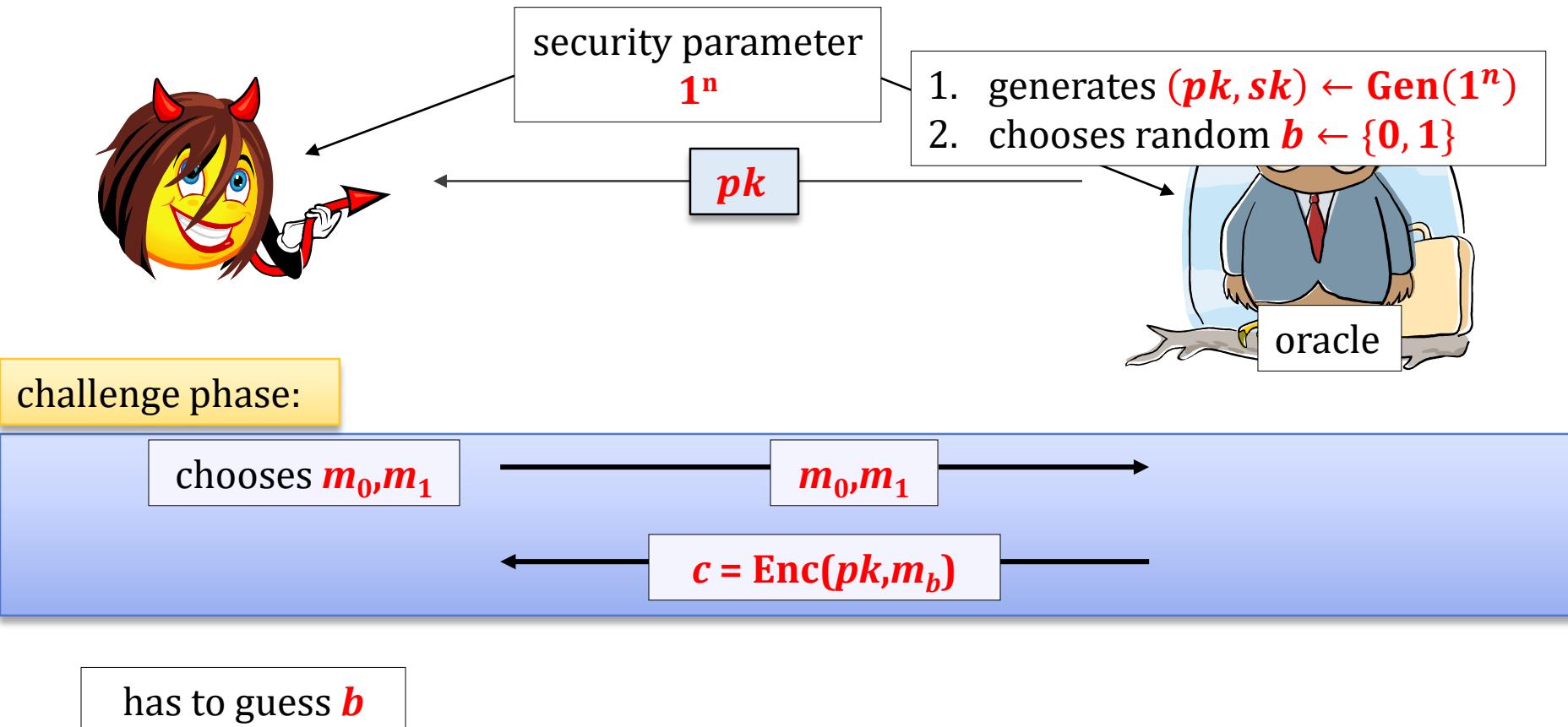
CPA in the symmetric settings



CPA in the asymmetric settings



The game after simplifications



CPA-security

Alternative name: CPA-secure

Security definition:

We say that **(Gen, Enc, Dec)** has indistinguishable encryptions under a chosen-plaintext attack (CPA) if any

randomized polynomial time adversary

guesses **b** correctly

with probability at most **$1/2 + \varepsilon(n)$** , where **ε** is negligible.

Plan



1. Definition of the **CPA security**
2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
3. The **hybrid encryption** and the **KEM/DEM** paradigm
4. Definition of the **CCA security**
5. Constructions of **CCA-secure** symmetric encryption
6. Constructions of **CCA-secure RSA** encryption schemes

Is the “handbook RSA” CPA-secure?

$N = pq$, such that p and q are random primes,
and $|p| = |q|$

e – random such that $e \perp (p - 1)(q - 1)$

d – random such that $ed \equiv 1 \pmod{(p - 1)(q - 1)}$
 $pk := (N, e)$ $sk := (N, d)$

$\text{Enc}_{pk}(m) = m^e \pmod{N}$.

$\text{Dec}_{sk}(c) = c^d \pmod{N}$.

Not CPA-secure!

In fact: no deterministic encryption scheme is secure.

How can the adversary win the game?

1. he chooses any m_0, m_1 ,
2. computes $c_0 = \text{Enc}_{pk}(m_0)$ himself
3. compares the result.

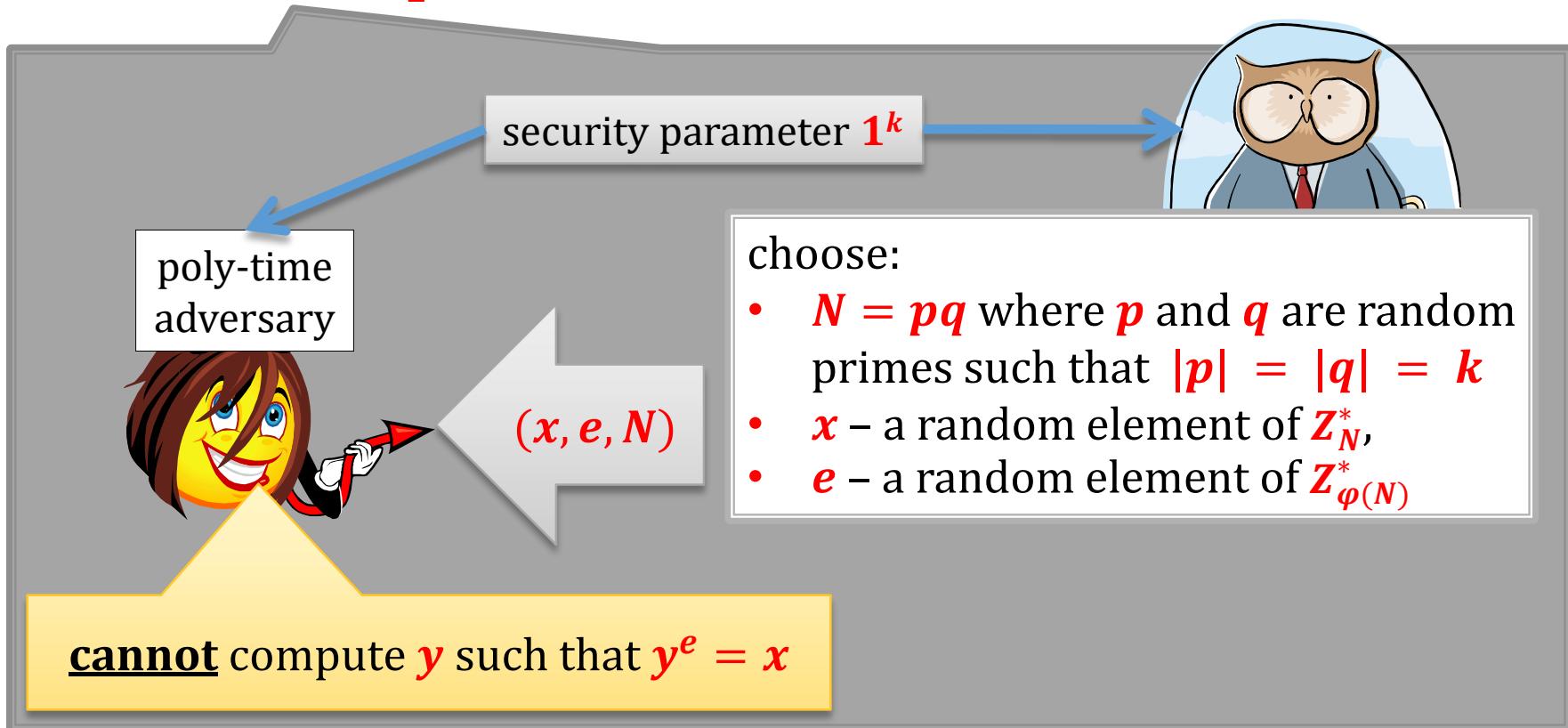
Moral: encryption has to be randomized.

Plan

- 
1. Definition of the **CPA security**
 2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
 3. The **hybrid encryption** and the **KEM/DEM** paradigm
 4. Definition of the **CCA security**
 5. Constructions of **CCA-secure** symmetric encryption
 6. Constructions of **CCA-secure RSA** encryption schemes

CPA-secure encryption from the RSA assumption

We now show how to construct a provably secure encryption scheme whose security is based on the **RSA assumption**.



Outline of the construction

1. We prove that the **least significant bit** is a **hard-core bit** for **RSA**.
2. We show how to “**encrypt using this bit**”

RSA hardcore bit

We know that “**the Jacobi symbol leaks**”.

Question: does **RSA** have a bit that is for sure well-hidden?

Answer: if **RSA assumption** doesn’t hold, then: **no**.

But what if it holds?

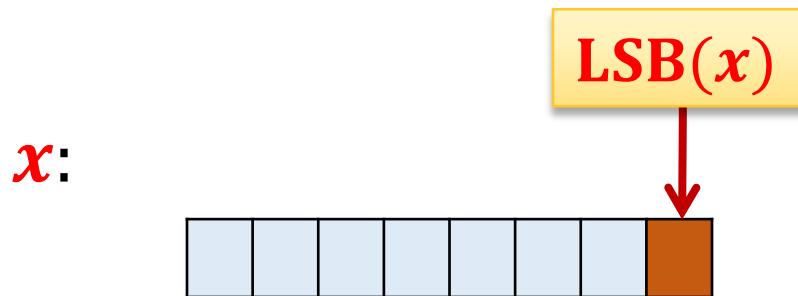
Answer: **yes** – the **least significant bit is hardcore** for RSA.

Notation

For an integer x we will write

LSB(x)

to denote the least significant bit of x .



In other words: **LSB(x) = $x \bmod 2$**

Fact (informally)

LSB is the “hardest bit to compute” in **RSA**.

(it is called a “hard-core bit”).

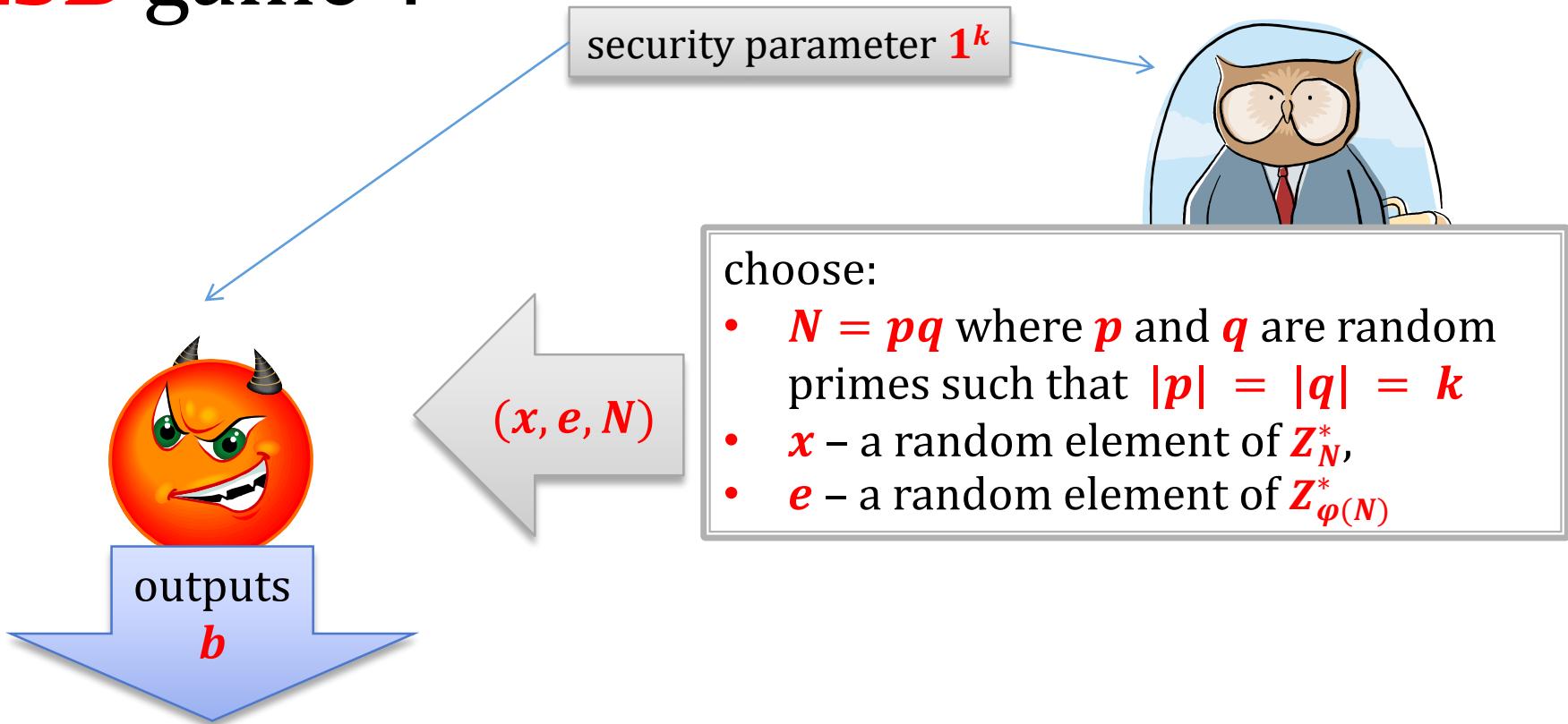
More precisely:

If you can compute **LSB** then you can invert **RSA**.

Note:

In some sense it is a “dual” predicate to Jacobi symbol...

“LSB game”:



The adversary **wins** if

b is the **least significant bit** of $y = \text{Enc}_{e,N}^{-1}(x)$

Theorem

Suppose the **RSA assumption** holds.

Then every poly-time adversary wins **Game 2** with a probability at most

$$0.5 + \varepsilon(k)$$

where ε is negligible.

W. Alexi, B. Chor, O. Goldreich, and C.P. Schnorr

[On the hardness of the least-significant bits of the RSA and Rabin functions,](#)
1984

In other words:

The least significant bit is a **hard-core bit for RSA**.

Proof strategy

Suppose we are given a poly-time adversary

For simplicity suppose
that this happens with
probability **1**

(not: **$0.5 + \varepsilon(k)$**)



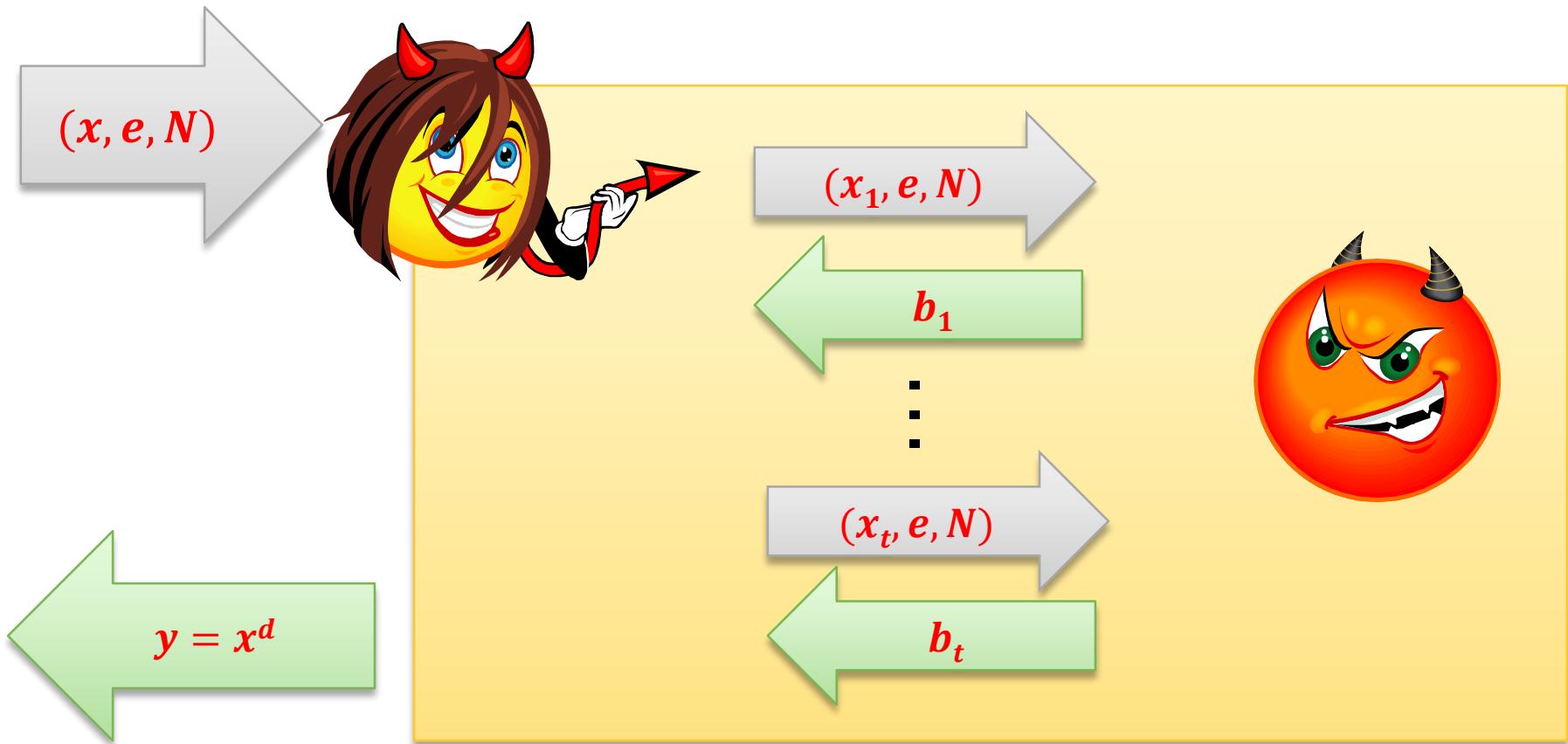
that wins the **LSB game**.

We construct a poly-time adversary



that breaks the **RSA assumption**.

Outline of the construction



Observation

Adversary  can be used to compute

LSB of $x^d \bmod N$.

It can also be used to compute (for any c)
LSB of $c \cdot x^d \bmod N$.

How?

$(c^e \cdot x, e, N)$



outputs

$$\begin{aligned} b' &= \text{LSB}((c^e \cdot x)^d) \\ &= \text{LSB}(c^{ed} \cdot x^d) \\ &= \text{LSB}(c \cdot x^d) \end{aligned}$$

The method

The adversary  will use 

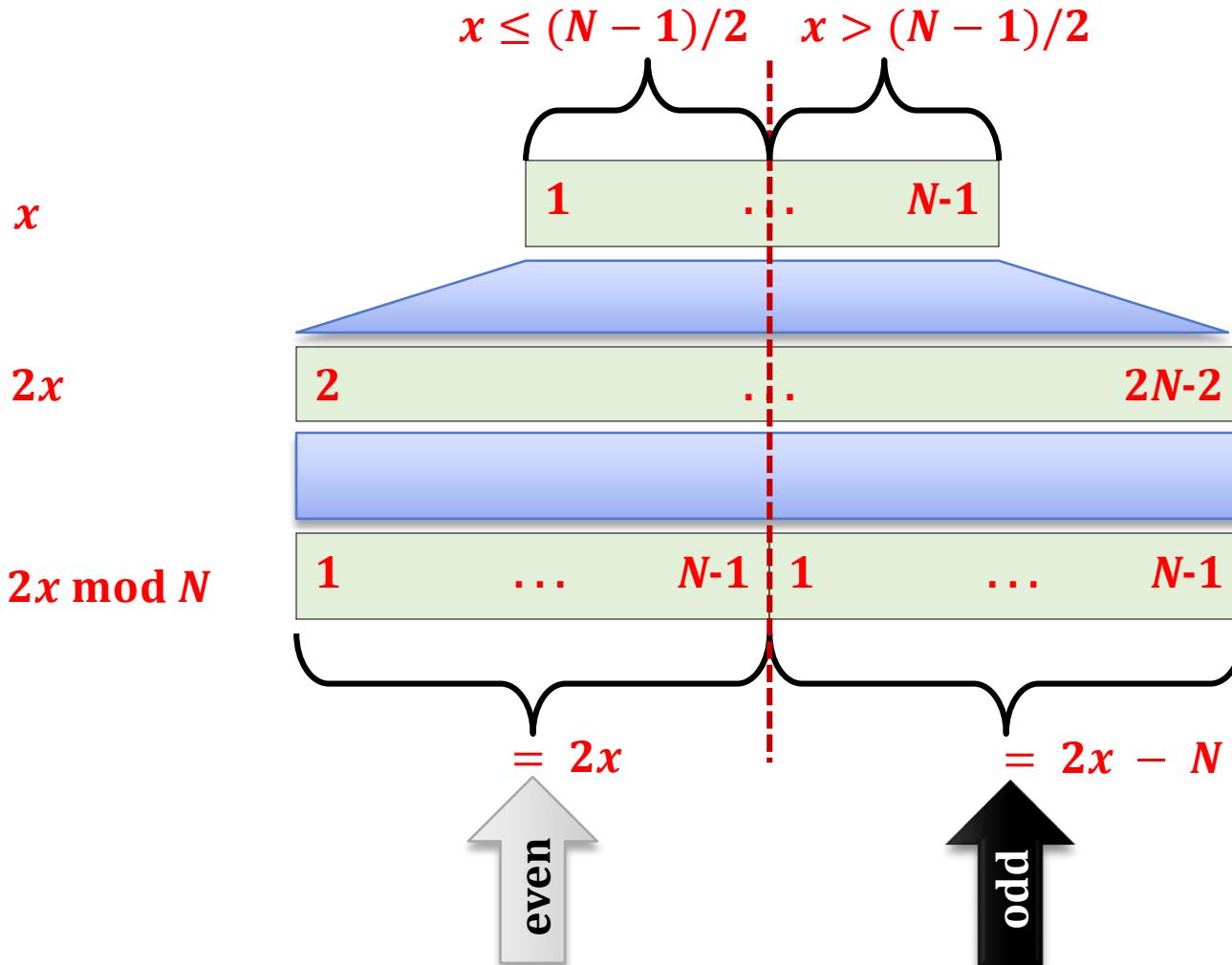
compute:

- $\text{LSB}(2x)$
- $\text{LSB}(4x)$
- $\text{LSB}(8x)$

⋮

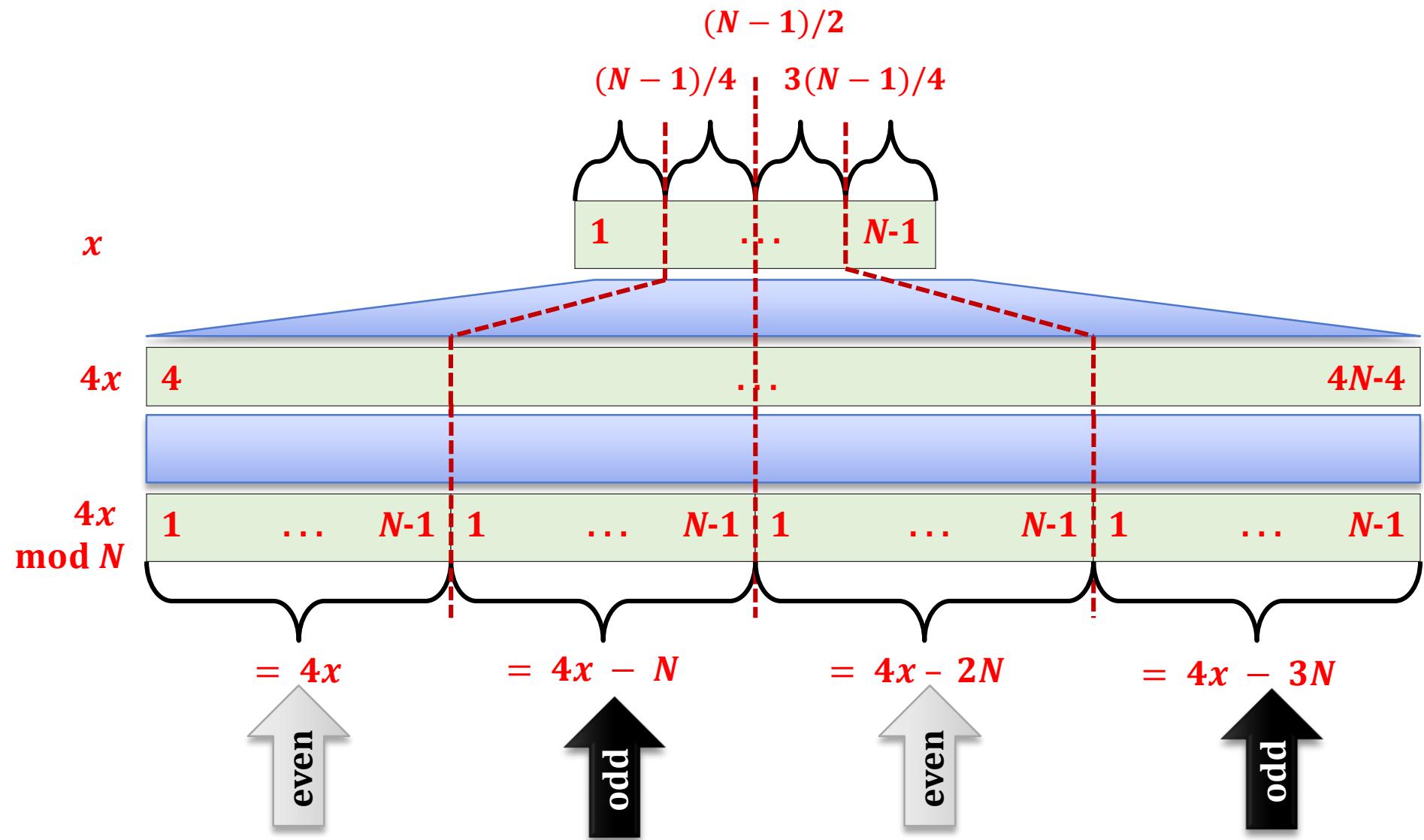
Why is it useful?

Observation

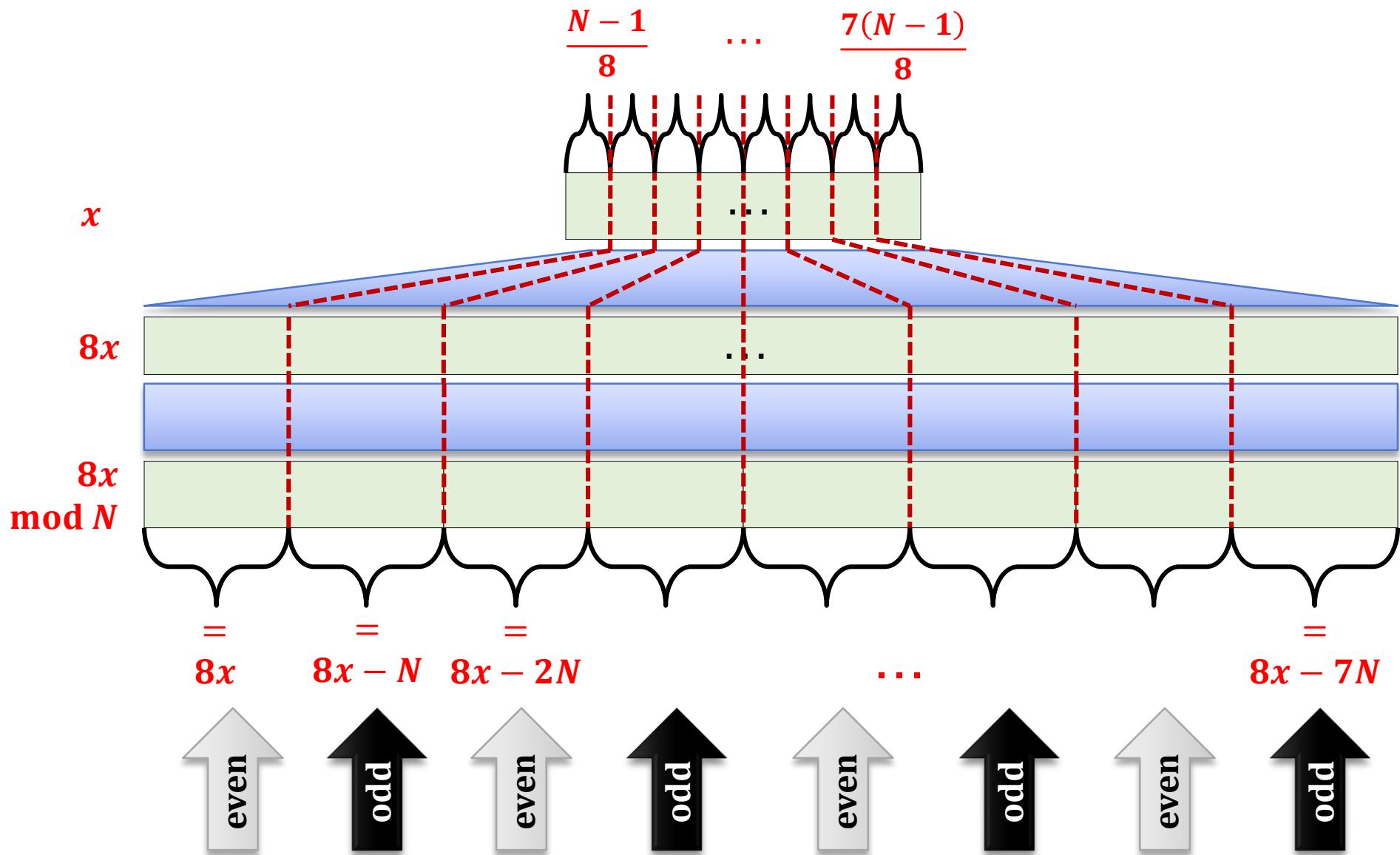


Remember:
 $N = pq$ is odd

Moral: $x \in [1, \dots, (N - 1)/2]$ iff $2x \bmod N$ is even



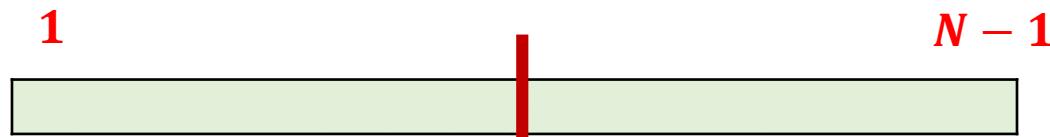
Moral: $x \in \left[1, \dots, \frac{N-1}{4}\right] \cup \left[\frac{N}{2} + 1, \dots, \frac{3(N-1)}{4}\right]$ iff $4x \bmod N$ is even



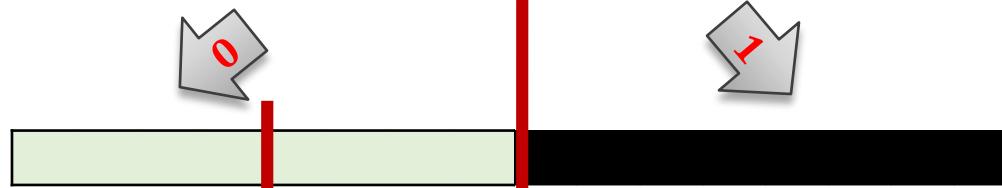
Moral: $x \in \left[1, \dots, \frac{N-1}{8}\right] \cup \left[\frac{2N}{8} + 1, \dots, \frac{3(N-1)}{8}\right]$
 $\cup \left[\frac{4N}{8} + 1, \dots, \frac{5(N-1)}{8}\right] \cup \left[\frac{6N}{8} + 1, \dots, \frac{7(N-1)}{8}\right]$ iff **$8x \bmod N$** is even

So we can use bisection

calculate
 $\text{LSB}((2 \cdot x)^d)$



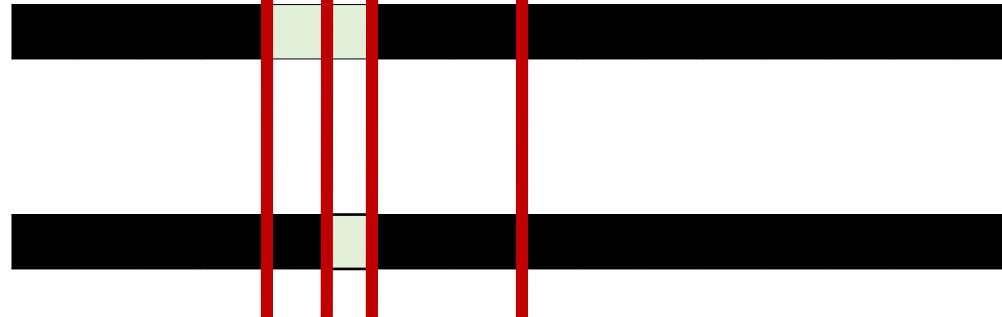
calculate
 $\text{LSB}((4 \cdot x)^d)$



calculate
 $\text{LSB}((8 \cdot x)^d)$



calculate
 $\text{LSB}((16 \cdot x)^d)$



⋮

QED

Why is it interesting?

We can encrypt **one bit messages** as follows:

(N, e) – public key

(N, d) – private key

$$\text{Enc}_{e,N}(b) = (\text{LSB}(x) \oplus b, x^e)$$

(where $x \leftarrow \mathbf{Z}_N^*$)

$$\text{Dec}_{d,N}(c, y) = \text{LSB}(y^d) \oplus c$$

This is secure **under the RSA assumption**

How to extend it to longer messages?

Encrypt **bit-by-bit**:

$$\begin{aligned}\mathbf{Enc}_{e,N}(\mathbf{m}_1, \dots, \mathbf{m}_k) = \\ ((\mathbf{LSB}(x_1) \oplus \mathbf{m}_1, \dots, \mathbf{LSB}(x_k) \oplus \mathbf{m}_k), (x_1^e, \dots, x_k^e))\end{aligned}$$

where $\mathbf{x}_1, \dots, \mathbf{x}_k \leftarrow \mathbf{Z}_N^*$

$$\begin{aligned}\mathbf{Dec}_{d,N}((\mathbf{c}_1, \dots, \mathbf{c}_k), (\mathbf{y}_1, \dots, \mathbf{y}_k)) = \\ (\mathbf{LSB}(y_1^d) \oplus \mathbf{c}_1, \dots, \mathbf{LSB}(y_k^d) \oplus \mathbf{c}_k)\end{aligned}$$

Lemma

Assume that the **RSA assumption holds**. Then the encryption scheme from the previous slide is **CPA-secure**.

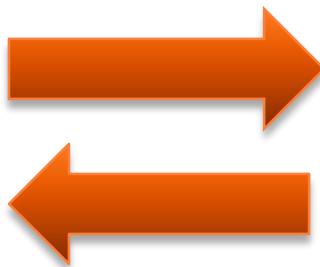
Proof: exercise

Conclusion

Advantage:

Security of this scheme is implied by the RSA assumption.

RSA assumption holds



the public-key
encryption scheme
that we just
constructed is secure

Disadvantage:

The ciphertext is much longer than the plaintext.

It is a rather theoretical construction!

Plan

- 
1. Definition of the **CPA security**
 2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
 3. The **hybrid encryption** and the **KEM/DEM** paradigm
 4. Definition of the **CCA security**
 5. Constructions of **CCA-secure** symmetric encryption
 6. Constructions of **CCA-secure RSA** encryption schemes

Encoding (also called: “padding”)

Before encrypting a message we usually **encode it** (adding some randomness).

This has the following advantages:

- it makes the encryption **non-deterministic**
- it **breaks the “algebraic properties”** of encryption.

How is it done in real-life?

PKCS #1: RSA Encryption Standard Version 1.5:

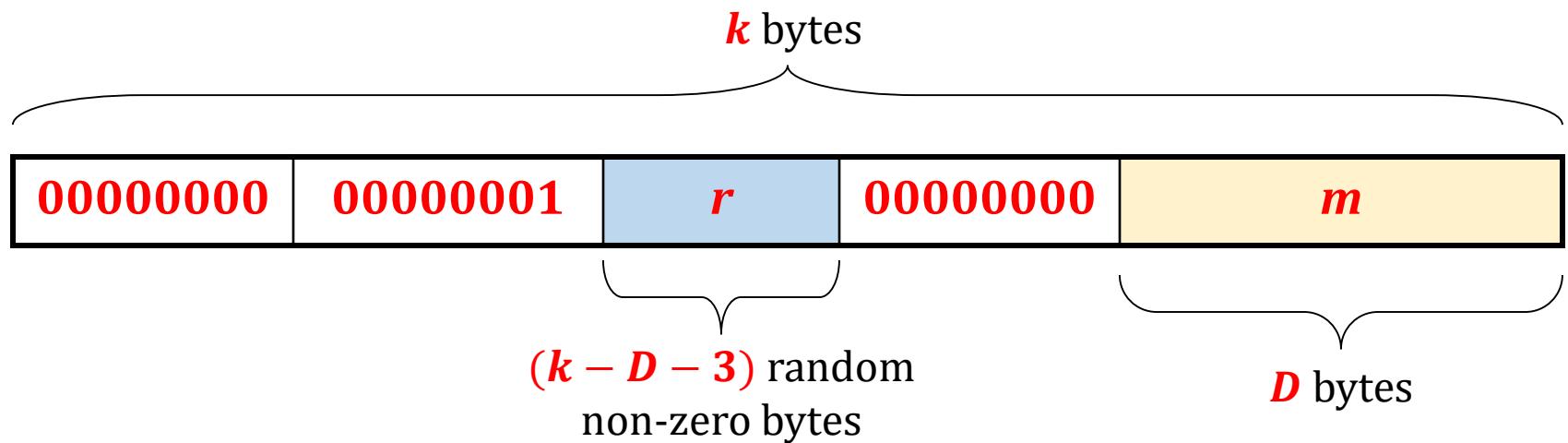
public-key: (N, e)

k := length on N in bytes.

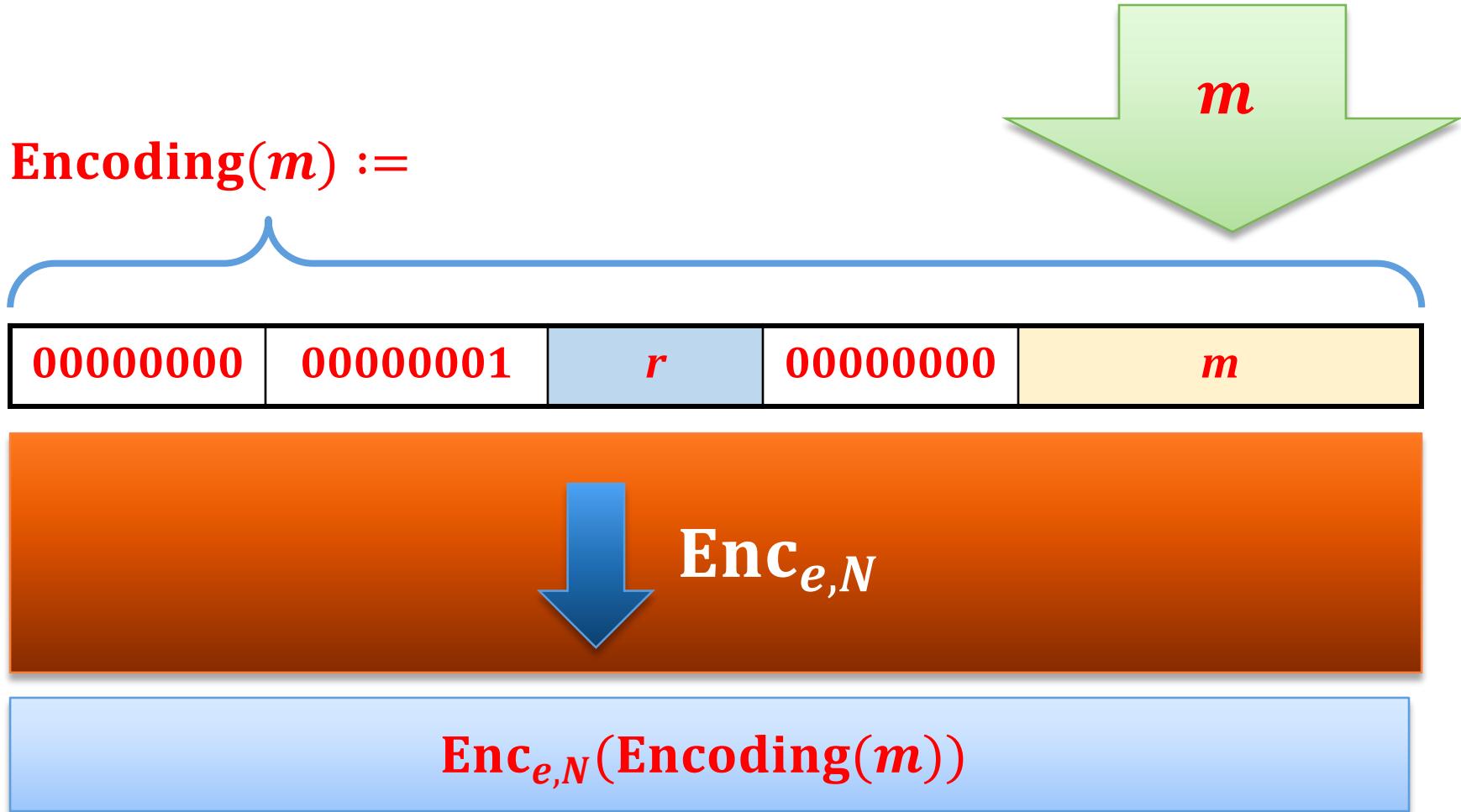
D := length of the plaintext

requirement: $D \leq k - 11$.

$\text{Enc}((N, e), m) := x^e \bmod N$, where x is equal to:



How to encrypt?



How to decrypt?

ciphertext y

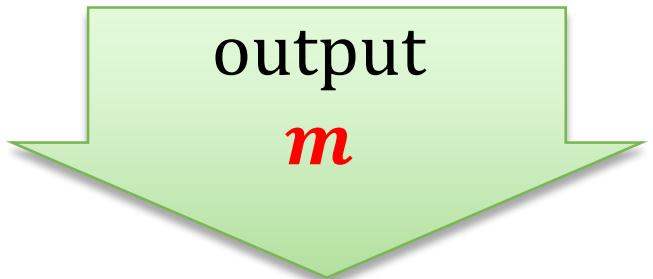


$\text{Dec}_{d,N}$

check if the format agrees....

00000000	00000001	r	00000000	m
----------	----------	-----	----------	-----

if not then output \perp , otherwise



Example

If the adversary can calculate the Jacobi symbol of

0000000	0000001	r	0000000	m
---------	---------	-----	---------	-----

most probably it doesn't help him in learning any information about m ...

Security of the PKCS #1: RSA Encryption Standard Version 1.5 – security

It is **believed** to be **CPA-secure**.

(as we will later learn: it's not "**CCA-secure**")

Plan



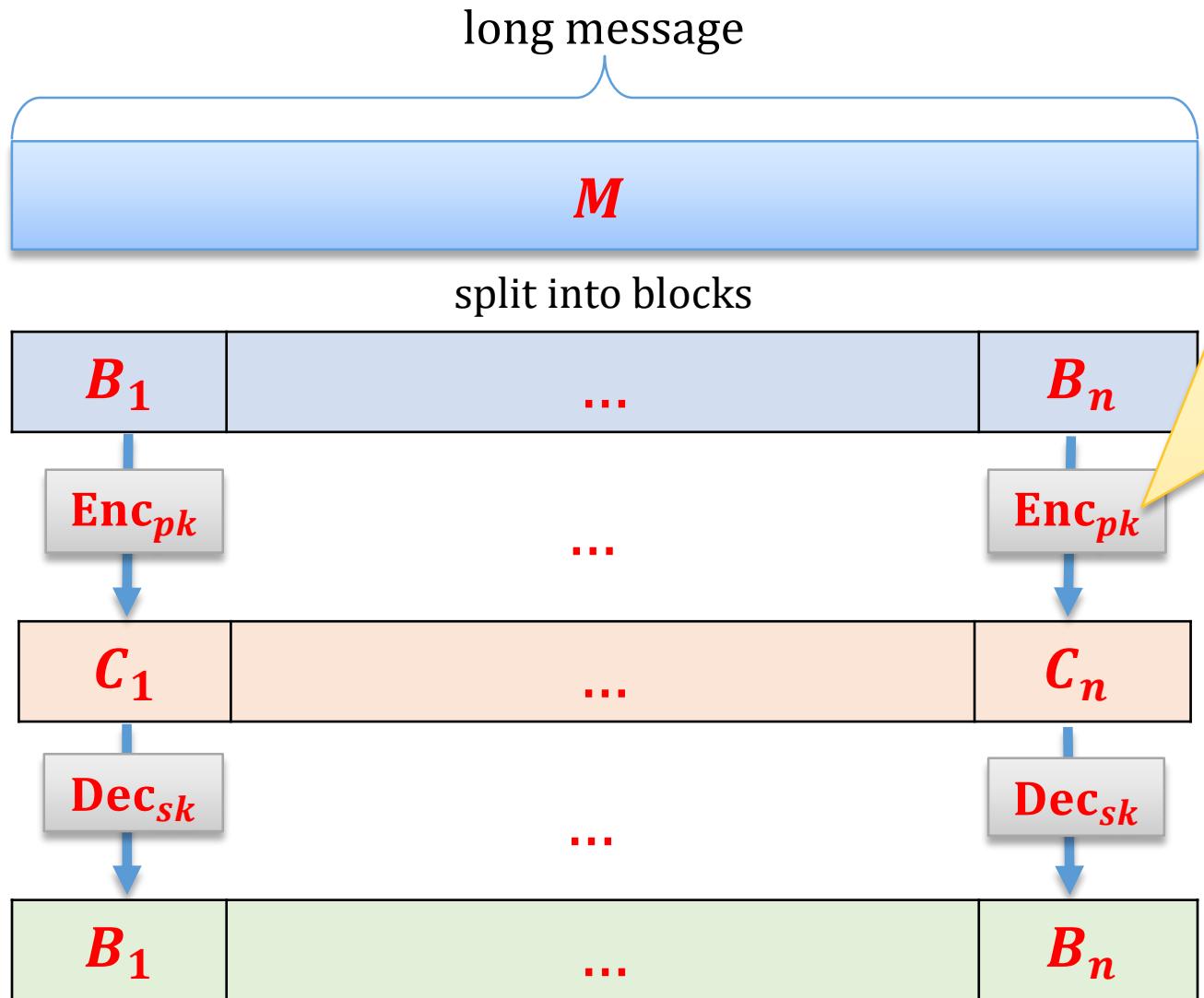
1. Definition of the **CPA security**
2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
3. The **hybrid encryption** and the **KEM/DEM** paradigm
4. Definition of the **CCA security**
5. Constructions of **CCA-secure** symmetric encryption
6. Constructions of **CCA-secure RSA** encryption schemes

How to encrypt longer messages?

Two options:

1. divide the message in blocks and **encrypt each block separately**.
2. combine the **public-key encryption with the private-key encryption**.

Encrypting block-by-block



note: this is **randomized**, so we don't have the same problem as with the ECB mode

A problem with this solution

It's rather inefficient (the number of public-key operations is proportional to $|M|$)

A more efficient solution:

hybrid encryption

Ingredients for the hybrid encryption

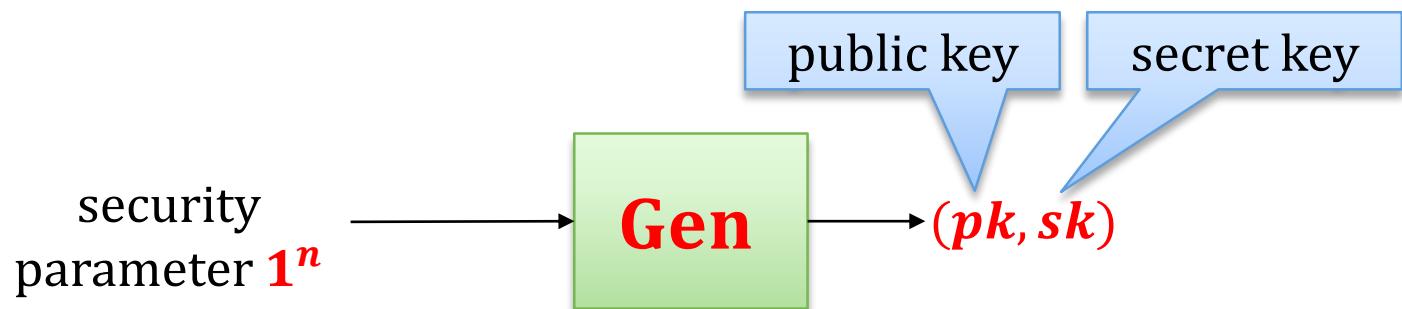
- **(Gen, Enc, Dec)**- a **public key encryption scheme**
- **(Enc', Dec')**- a **private key encryption scheme**

Main idea:

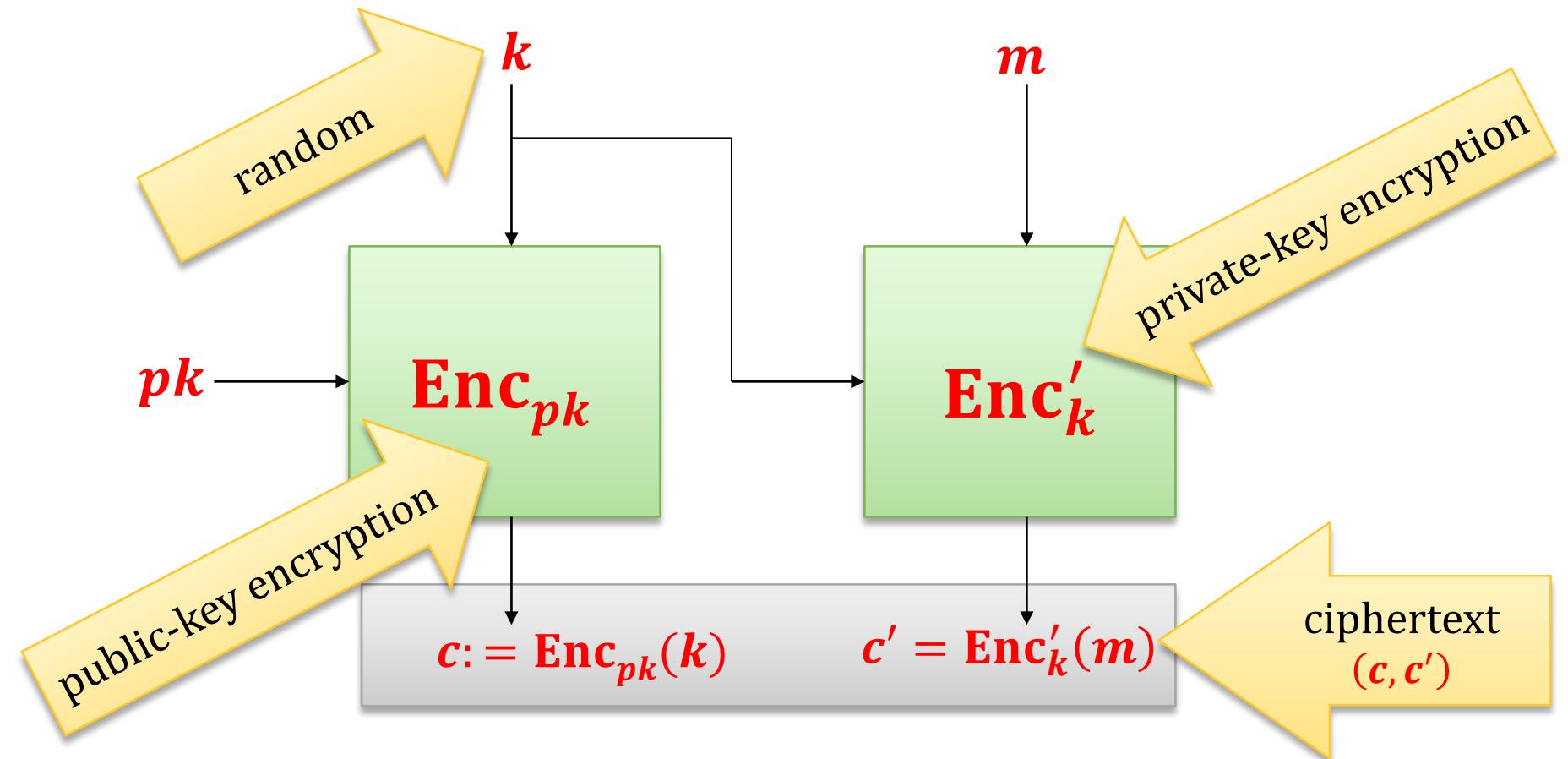
Encrypt the symmetric key with a public-key encryption scheme.

Key generation

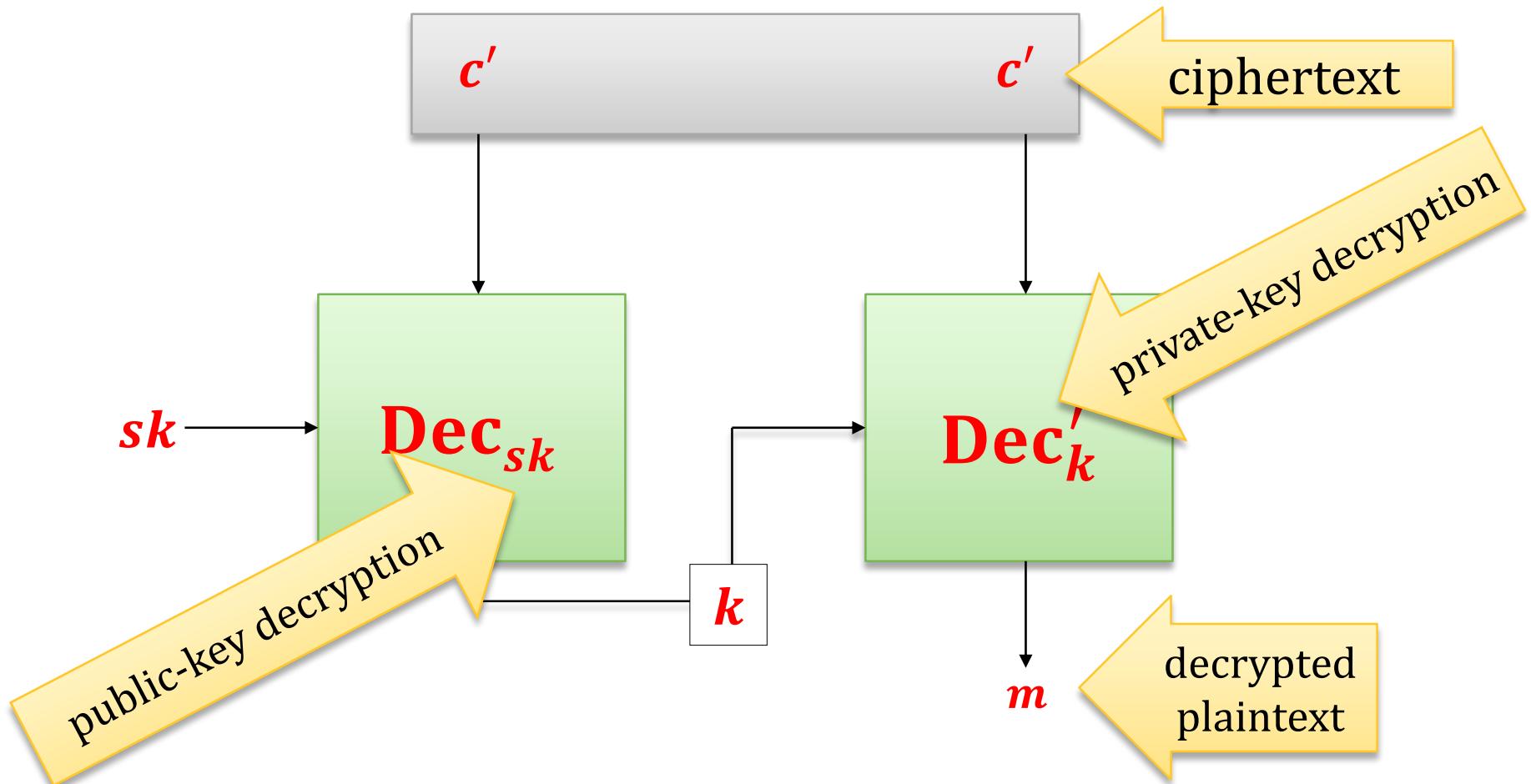
The same as in the public-key scheme:



How to encrypt?



How to decrypt?



A more direct method: the **KEM/DEM** paradigm

DEM – Data Encapsulation Mechanism

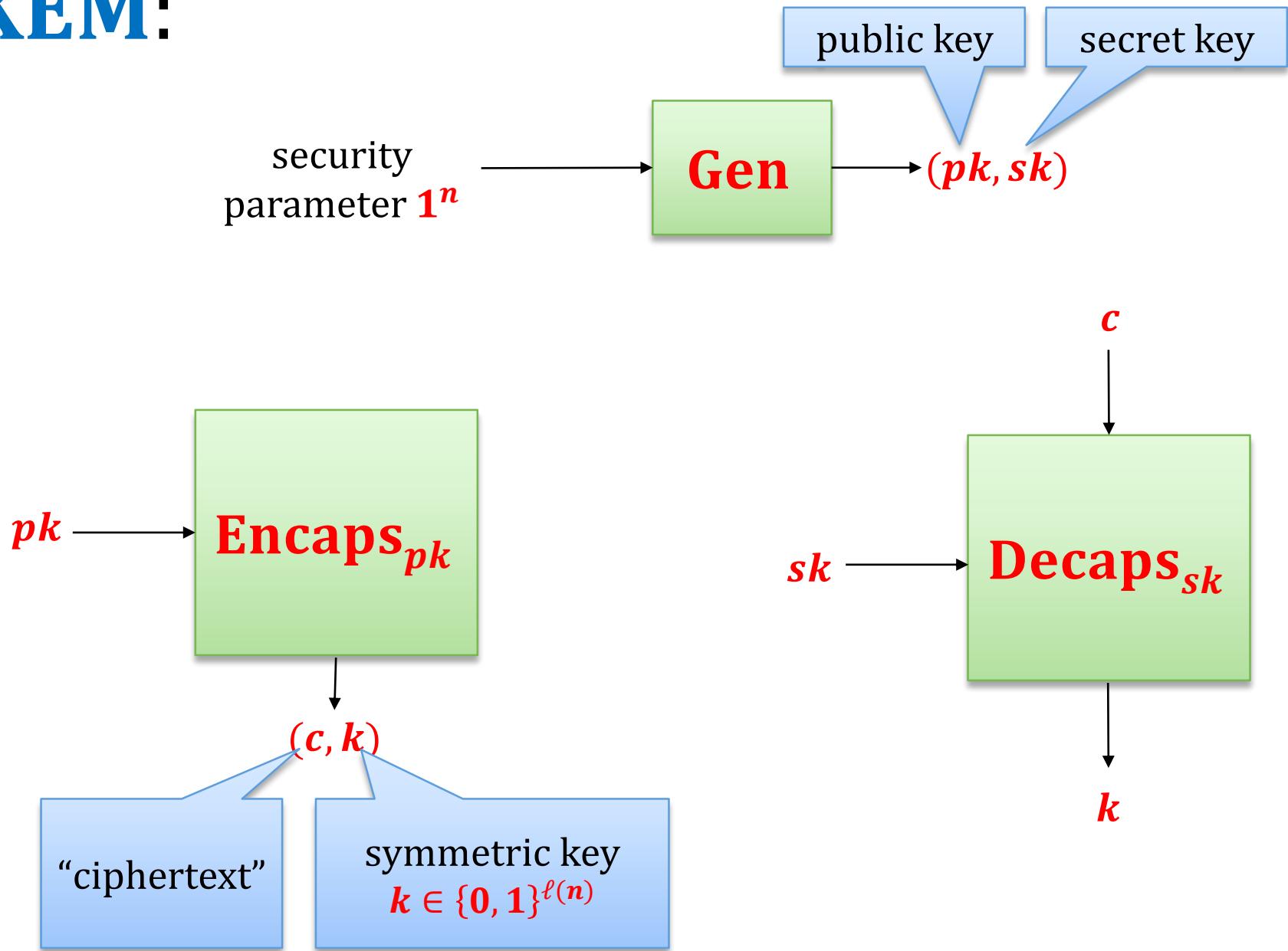
= private key encryption

KEM – Key Encapsulation Mechanism

consists of the following algorithms:

- **key generation algorithm Gen** – as in **PKE**,
- **encapsulation algorithm Encaps**,
- **decapsulation algorithm Decaps**.

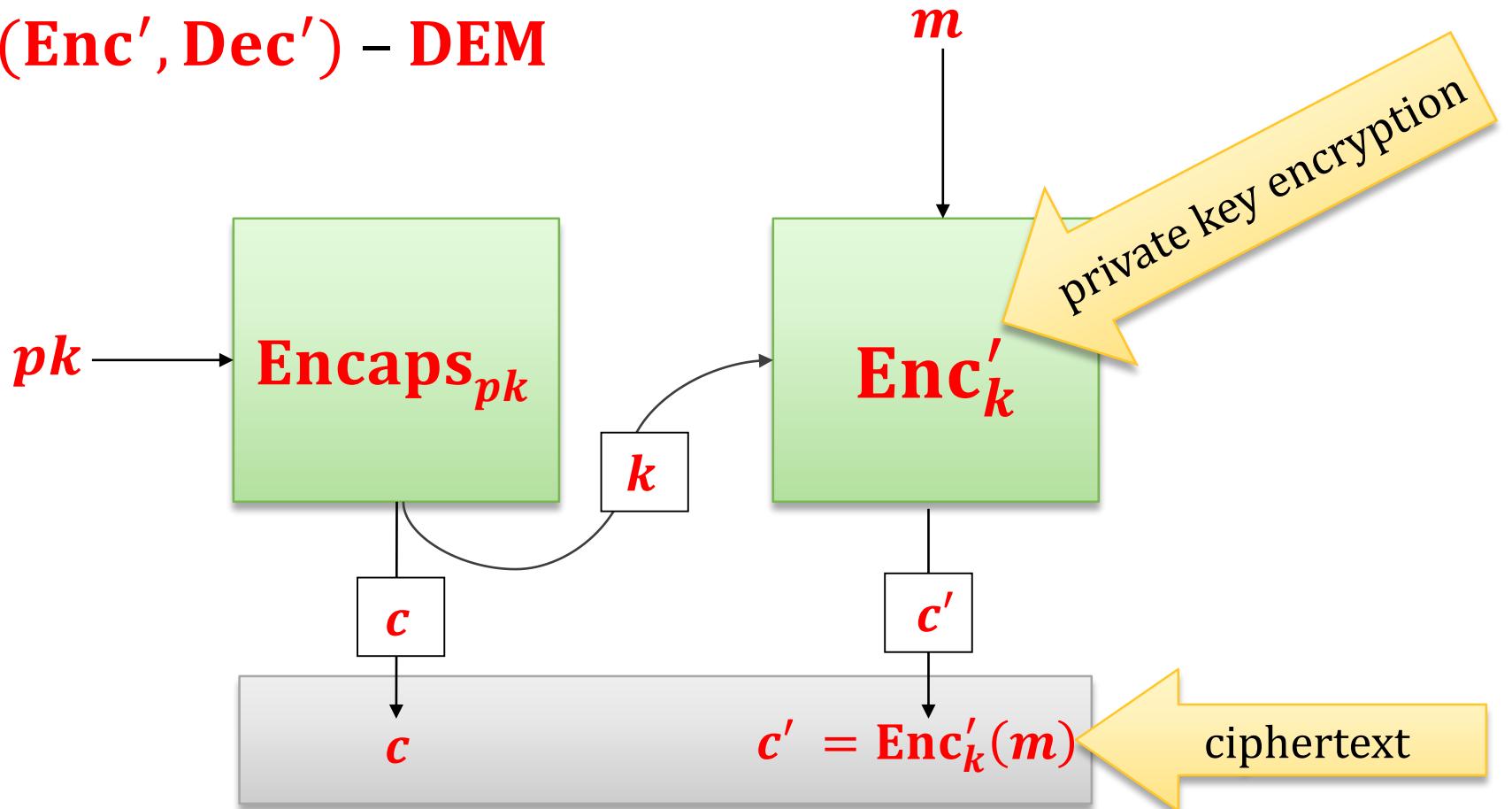
KEM:



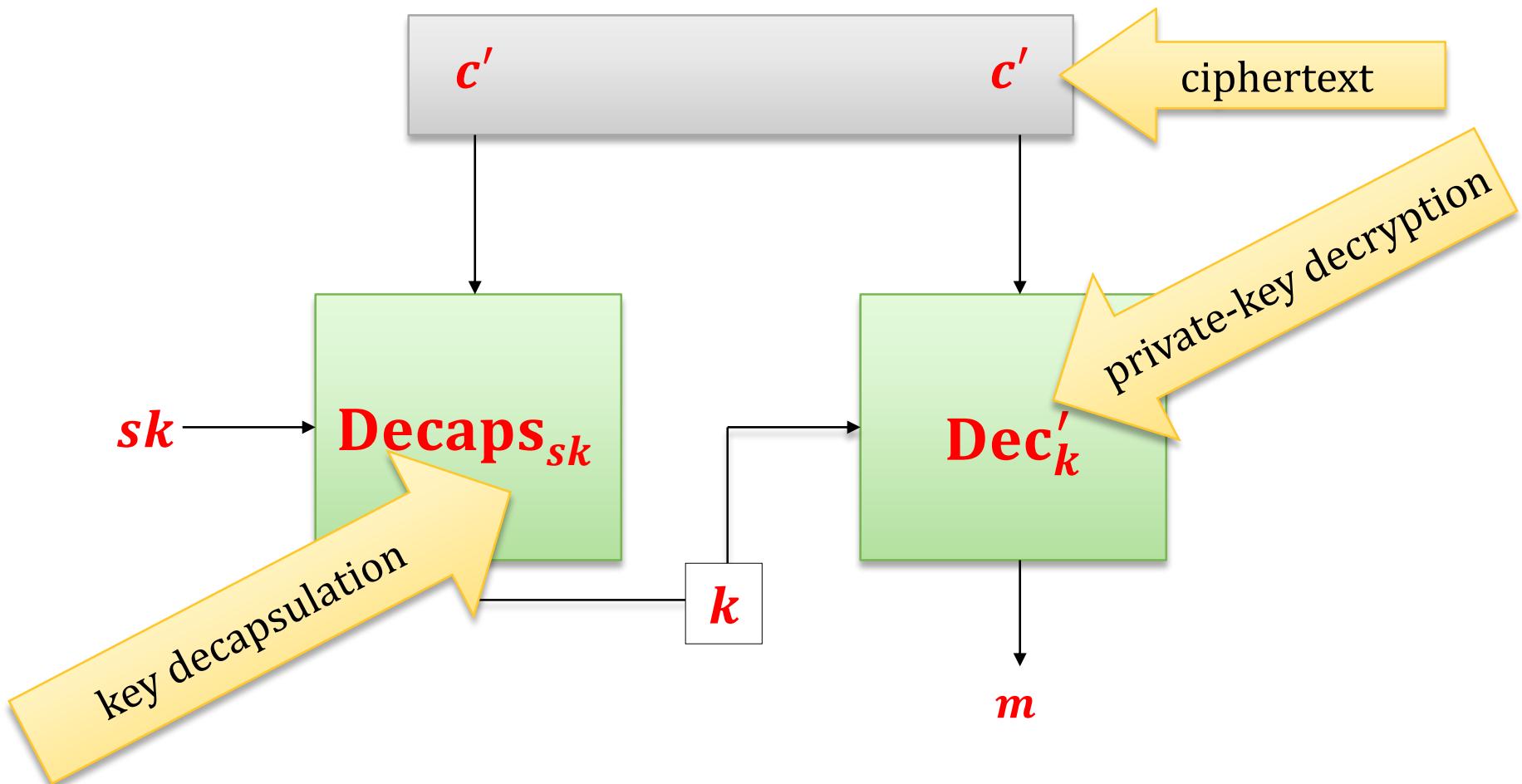
How to encrypt?

Gen – as in **KEM**

(Enc', Dec') – **DEM**



How to decrypt?



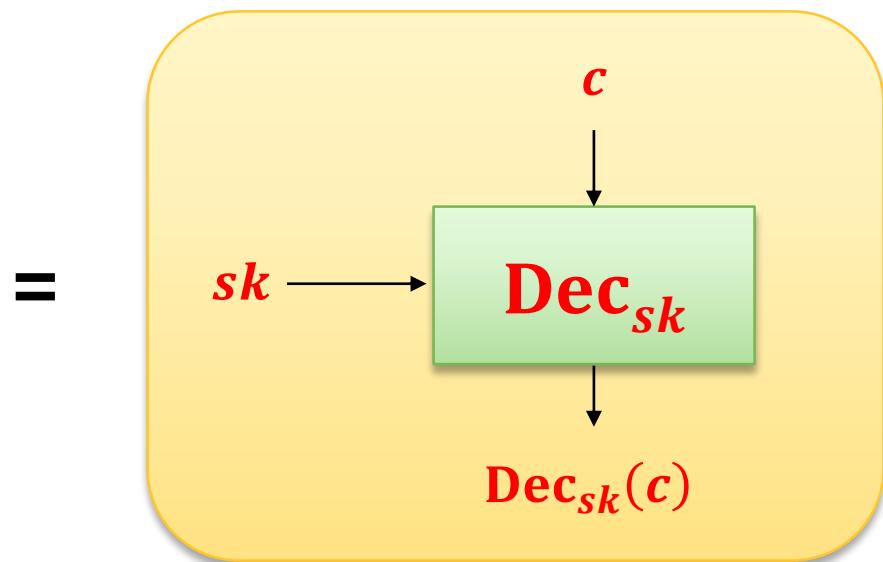
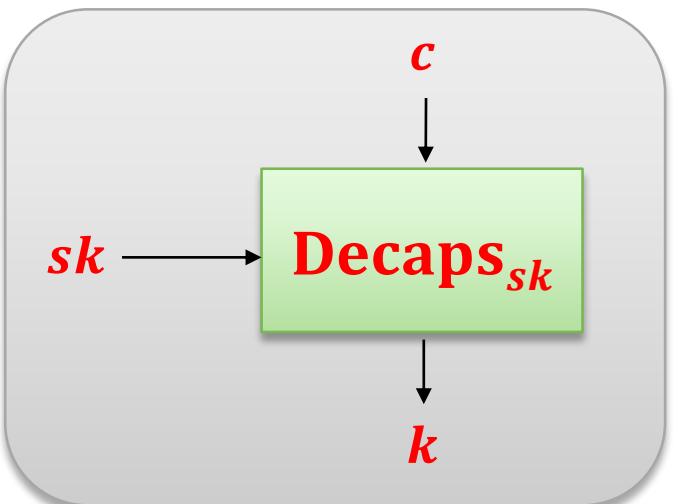
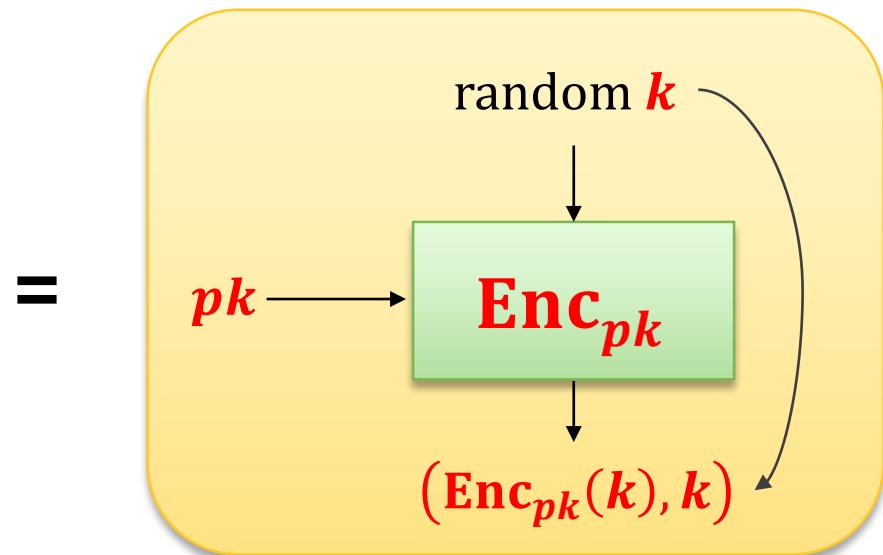
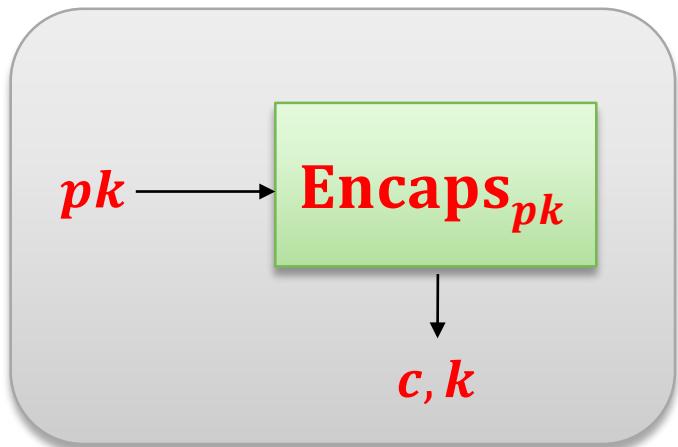
One method to implement **KEM**

Take a **public-key encryption** scheme **(Gen, Enc, Dec)**.

Define **KEM** as follows:

- **Gen** is the same
- **Encaps_{pk}** = generate a **random symmetric key k** and **output**
$$(\text{Enc}_{pk}(k), k)$$
- **Decaps_{sk}** = on input **c** **output** **Dec_{sk}(c)**

Pictorially:



Note

In this case **KEM/DEM** method is simply equal to the **hybrid encryption**.

However: there exist other, direct methods for key encapsulation.

(they are more efficient)

Consequences of these approaches

For longer messages the cost of encryption is
dominated by the cost of symmetric operations.

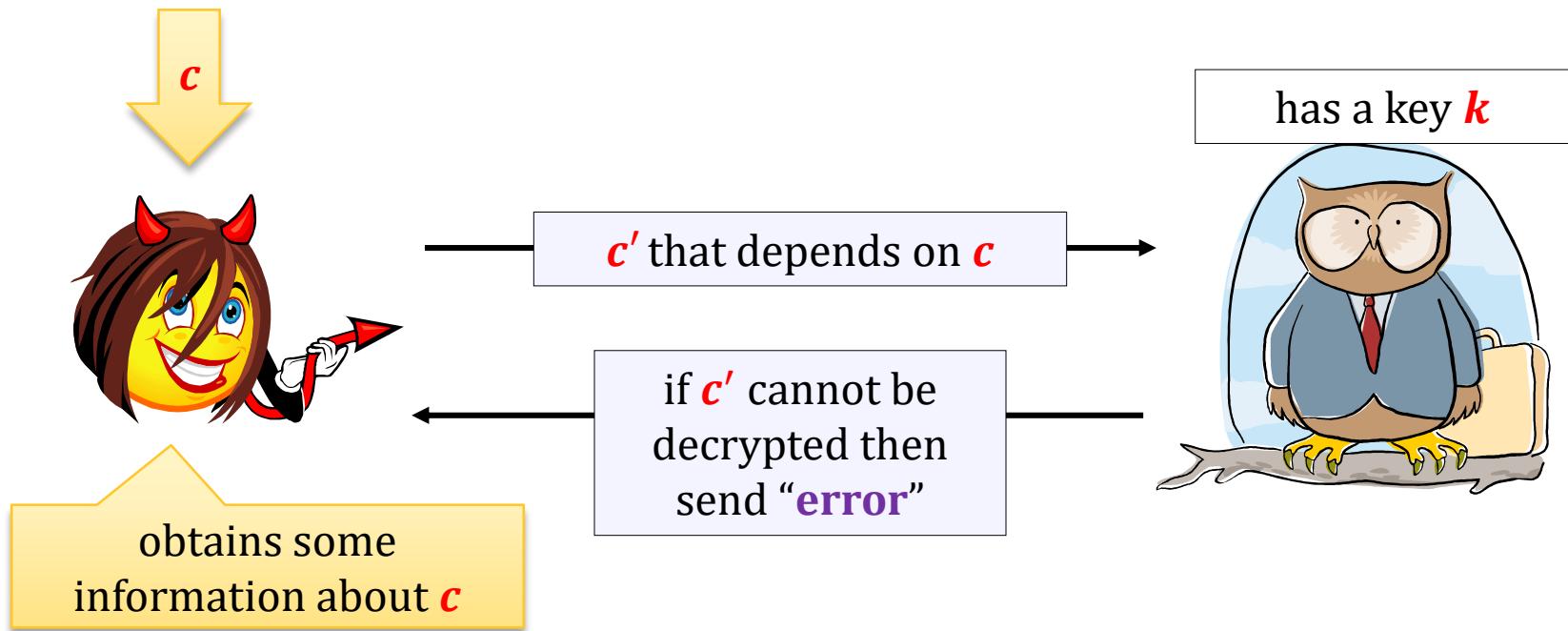
Hence: the public-key operations (amortized over the length of the messages) are almost “for free”.

Plan

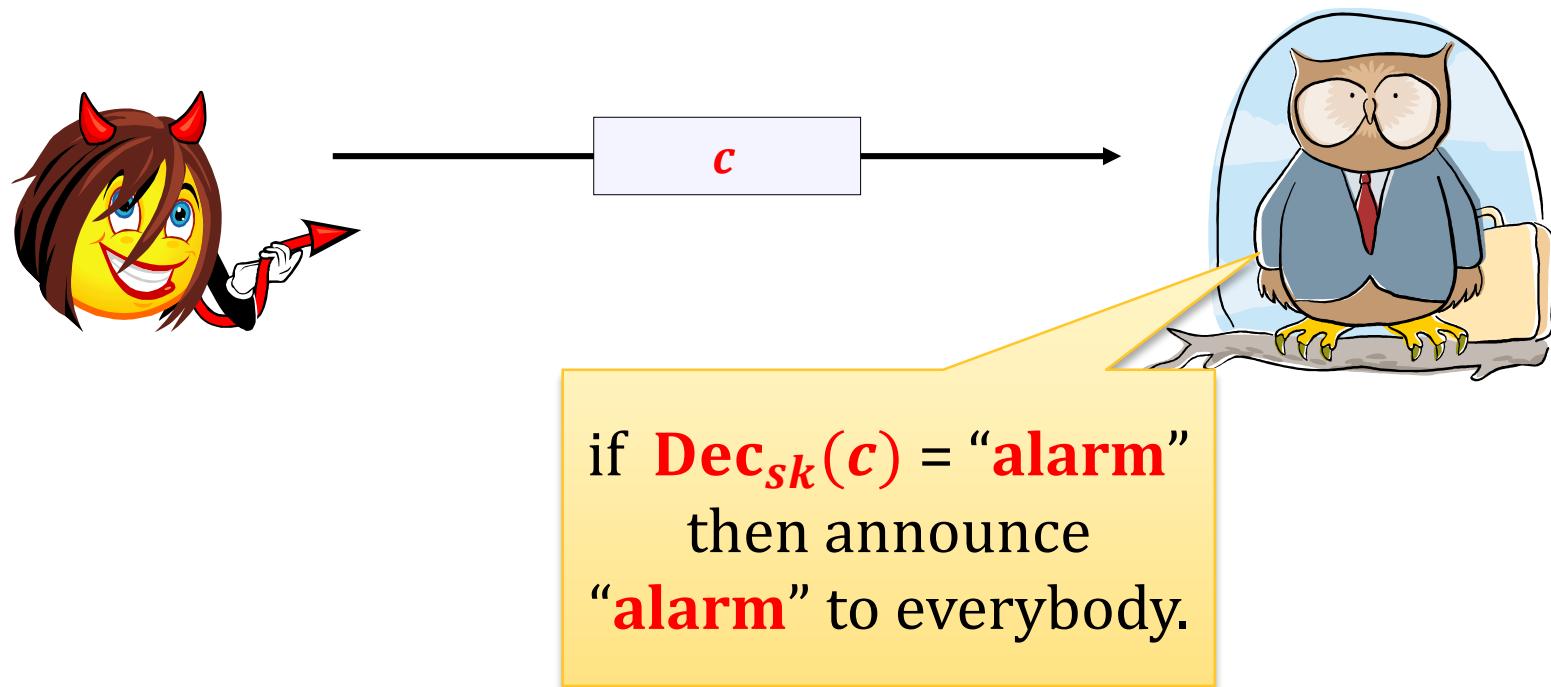
- 
1. Definition of the **CPA security**
 2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
 3. The **hybrid encryption** and the **KEM/DEM** paradigm
 4. Definition of the **CCA security**
 5. Constructions of **CCA-secure** symmetric encryption
 6. Constructions of **CCA-secure RSA** encryption schemes

Chosen-plaintext attacks – motivation

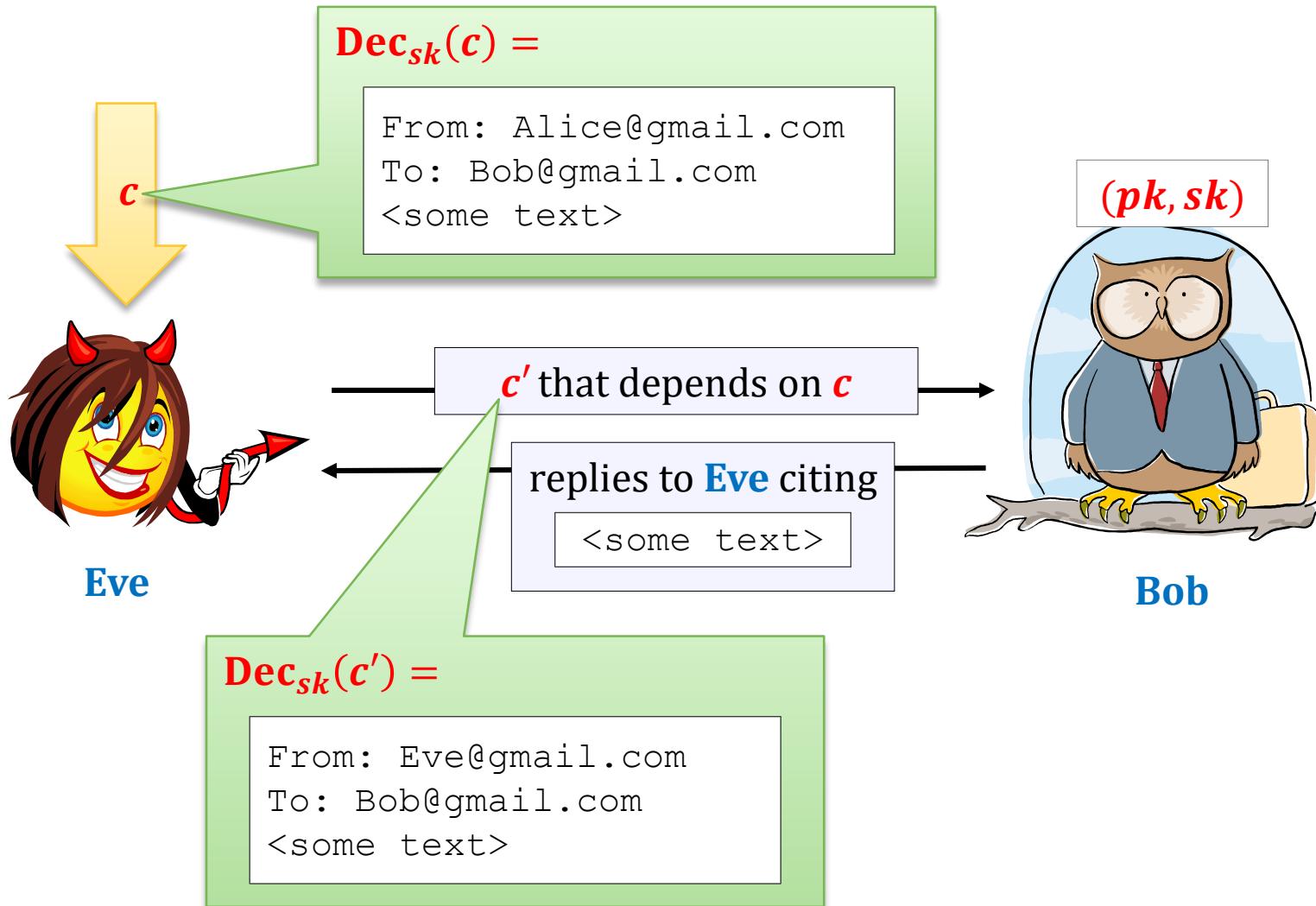
Remember the attack on the **symmetric** encryption based on the **error messages from the decryption oracle** ([exercise 3, Nov 9, 2016](#))?



Another scenario



A more advanced example



Note

CPA security does not imply that such attacks are impossible.

We need a **stronger** security definition.

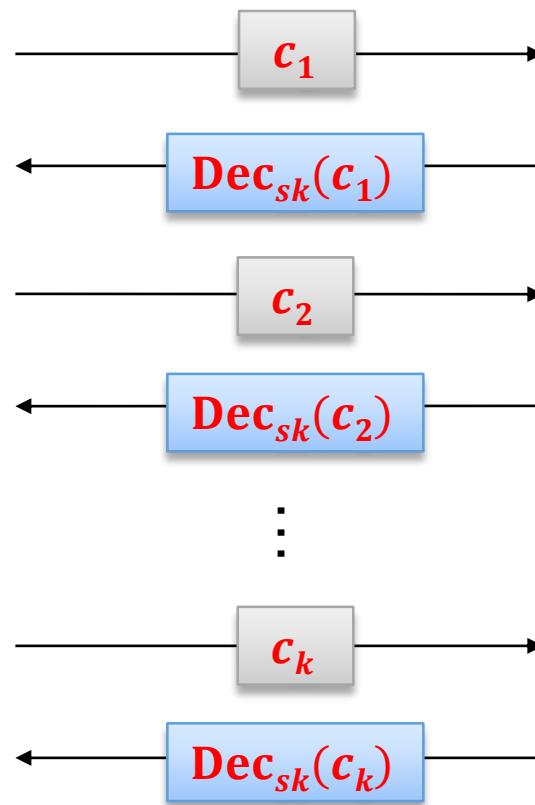
This will be called:

chosen-ciphertext security (CCA)

It can be defined both for the **symmetric** and **asymmetric** case.

Decryption oracle

To define the CCA-security we consider a **decryption oracle**.



convention:

$\text{Dec}_{sk}(c_i) \coloneqq \perp$
if c_i cannot be decrypted

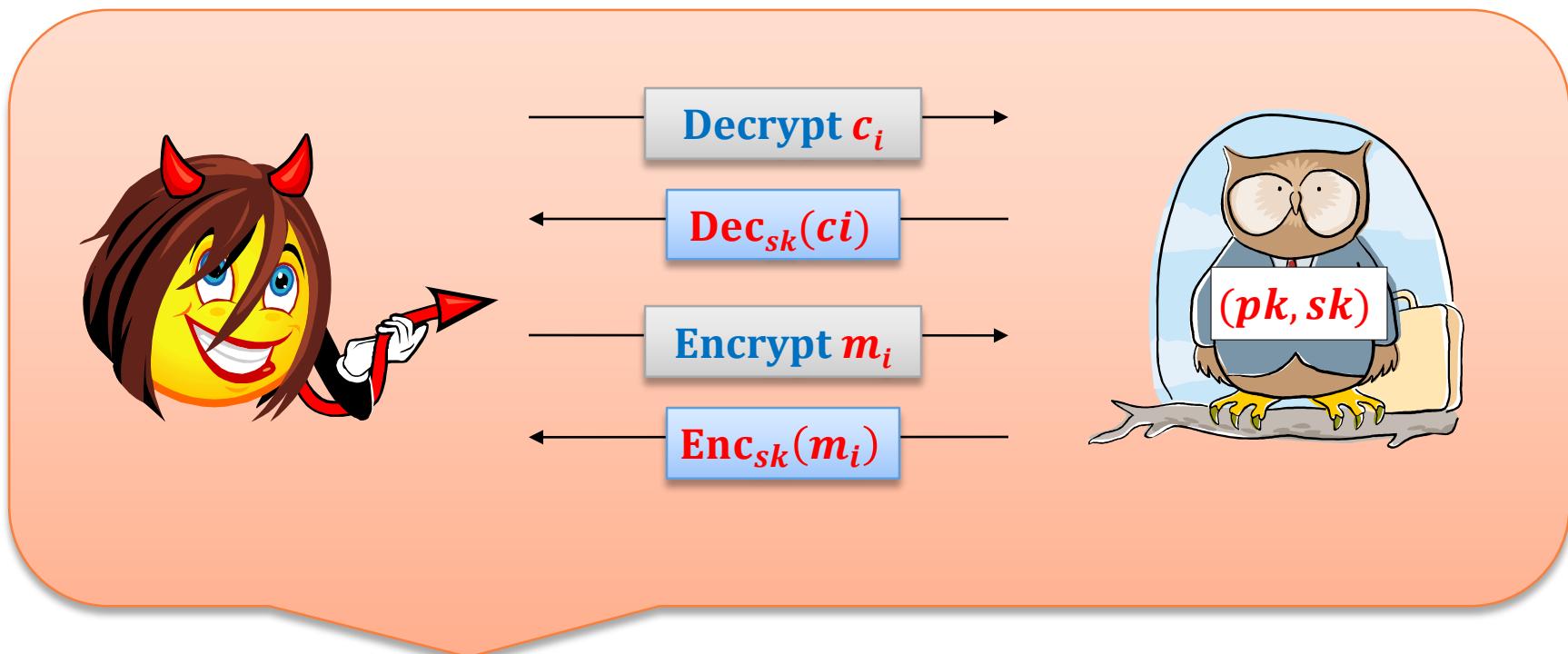
we call such a ciphertext
"invalid"

Decryption/encryption oracle

We assume that **also CPA** is allowed.

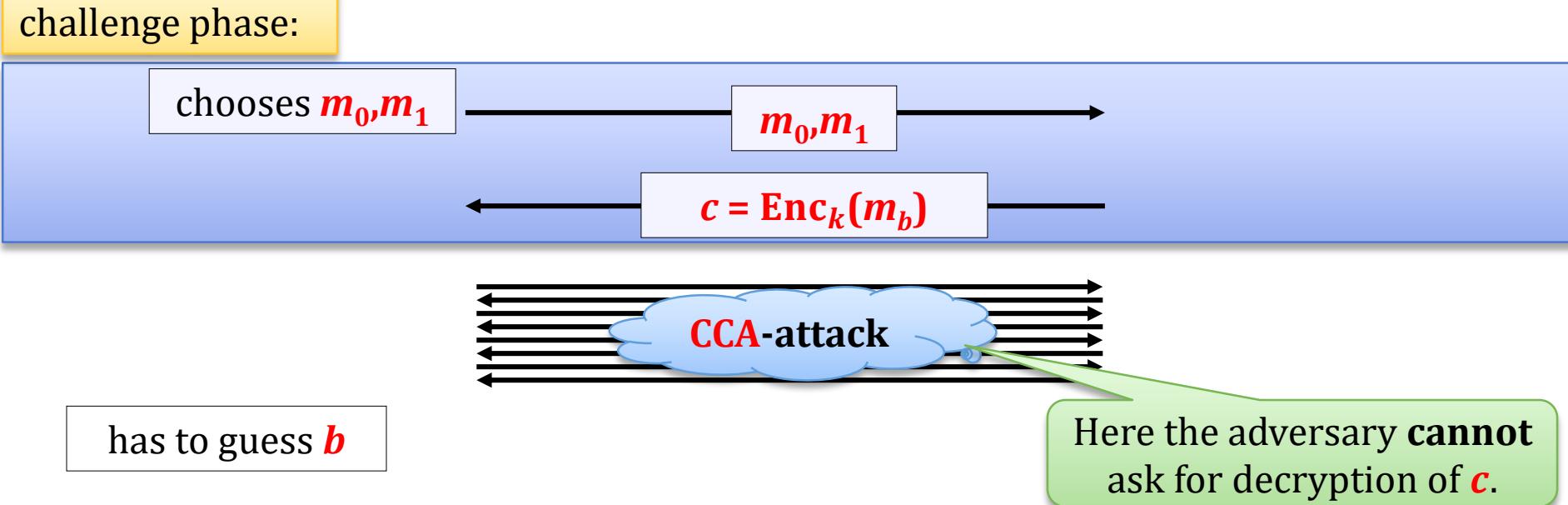
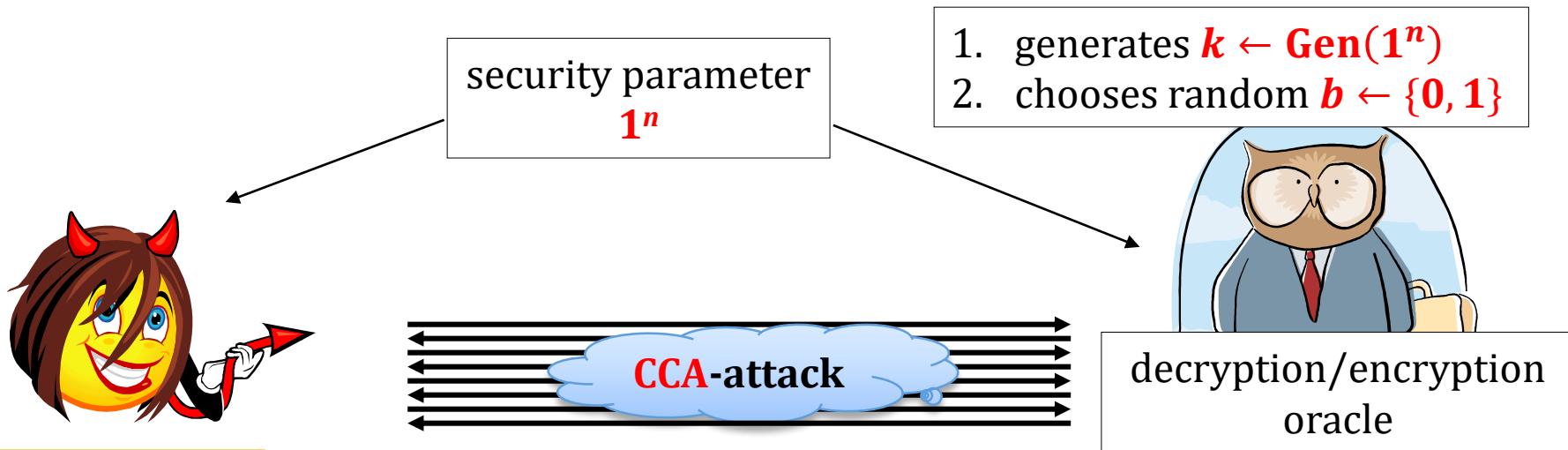
Two types of queries:

this will be used in the symmetric case

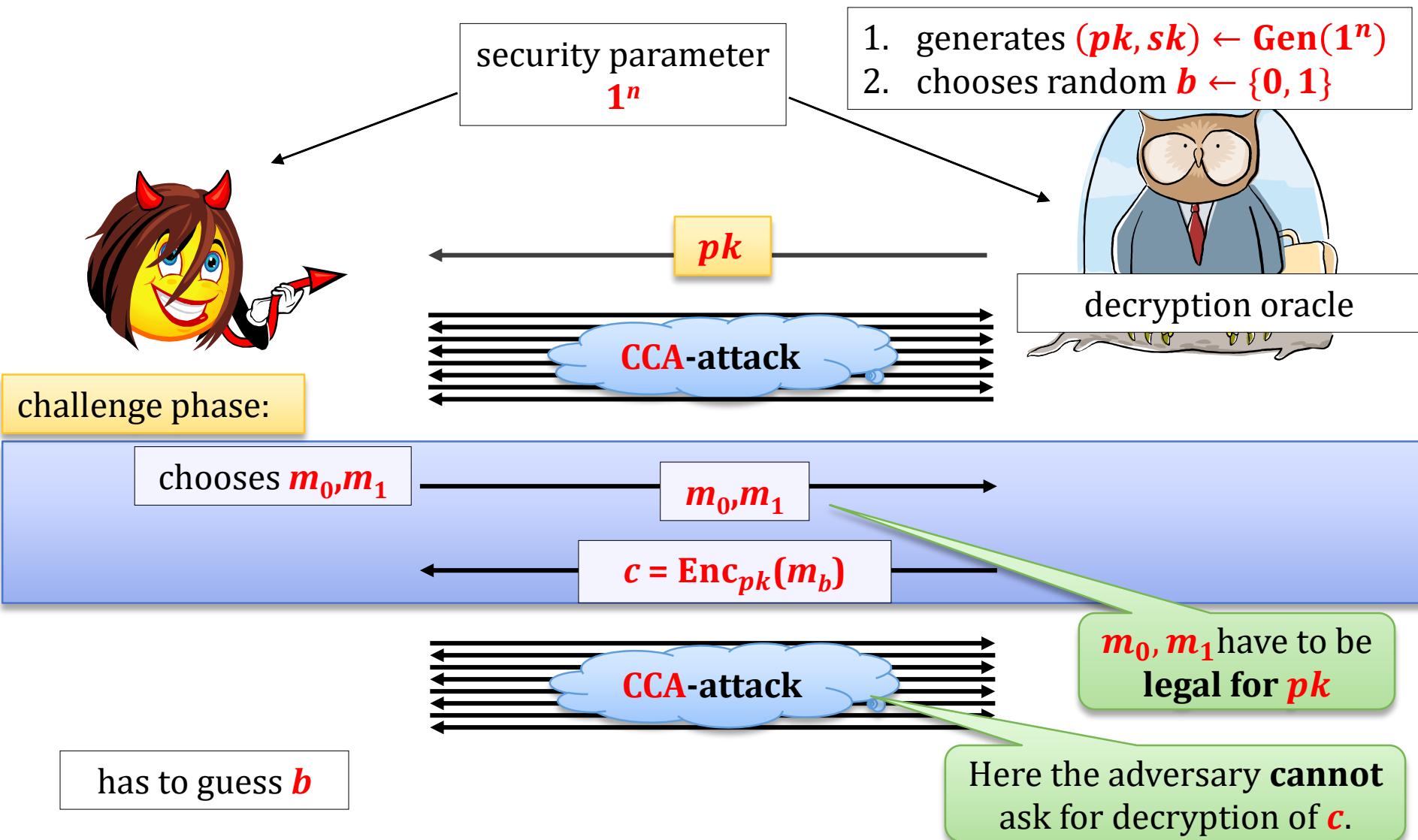


this is called a **CCA-attack**

CCA-security – the game in the symmetric case



CCA-security – the game in the asymmetric case



CCA security

Alternative name:
CCA-secure

Security definition (in the **asymmetric** case):

In the **symmetric** case: (**Enc, Dec**)

We say that (**Gen, Enc, Dec**) has indistinguishable encryptions under a chosen-ciphertext attack (**CCA**) if any

randomized polynomial time adversary

guesses ***b*** correctly

with probability at most **$1/2 + \epsilon(n)$** , where **ϵ** is negligible.

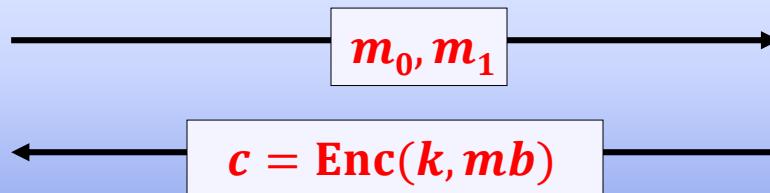
Easy to see

CCA-security implies **CPA security**

(because the adversary in the “**CCA game**” is at least as powerful as the one in the “**CPA game**”)

What about the implication in the other direction?

CPA-security does **not** imply the CCA-security



here ask about
the “related” c'

CCA-attack

Here Eve cannot ask for
decryption of c .

Informally:

To win the game it is enough that Eve is computes some c' such that $\text{Dec}_k(c')$ is “related to” $\text{Dec}_k(c)$.

(Why? Because then she is allowed to ask for it.)

For example: it is possible for any stream cipher!

if $c' = c' \oplus (1, \dots, 1)$ then $\text{Dec}_k(c') = \text{Dec}_k(c) \oplus (1, \dots, 1)$

How to construct CCA-secure schemes?

- in the **symmetric case**: **easy**
- in the **asymmetric case**: usually **harder** (also: in this case the “**CCA attacks**” are more realistic).

Plan

- 
1. Definition of the **CPA security**
 2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
 3. The **hybrid encryption** and the **KEM/DEM** paradigm
 4. Definition of the **CCA security**
 5. Constructions of **CCA-secure** symmetric encryption
 6. Constructions of **CCA-secure RSA** encryption schemes

Symmetric case

Simplest method: **authenticate** every ciphertext with a MAC.

Ingredients:

- **(Enc, Dec)** – a **CPA-secure symmetric encryption** scheme
- **(Tag, Vrfy)** – a **message authentication code** that is **strongly secure**.

A MAC is **strongly secure** if the adversary cannot produce a valid tag t' on a message m even if he saw a valid pair (m, t) (where $t \neq t'$)

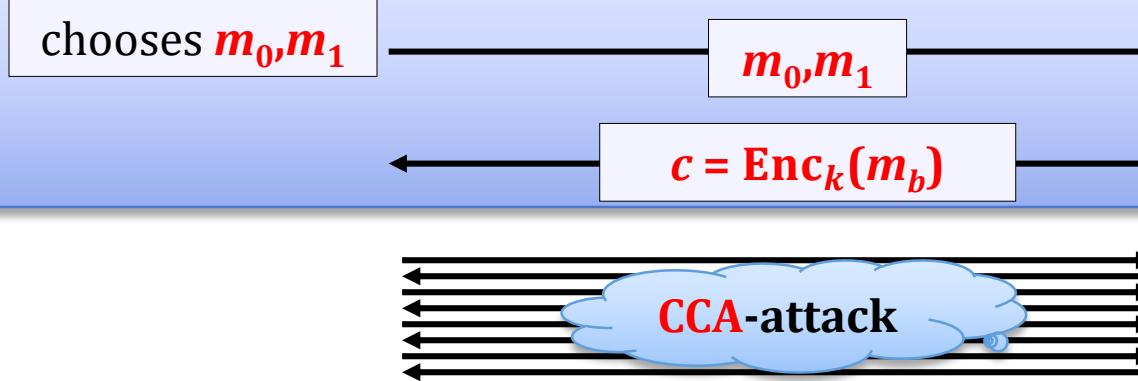
The method from Lecture 5

encrypt-then-authenticate:

key: a pair (k_1, k_2)

- to **encrypt** m compute $c := \text{Enc}_{k_1}(m)$ and $t := \text{Tag}_{k_2}(c)$, and output (c, t)
- to **decrypt** (c, t) :
if $\text{Vrfy}_{k_2}(c, t) = \text{no}$ then output \perp
otherwise output $\text{Dec}_{k_1}(c)$

Why is this secure?



The adversary **cannot “produce himself” a valid ciphertext.**

The only decryption queries **Decrypt c'** on which he doesn't get \perp are such that he received c' from the oracle before.

But he already knows the decryptions of such c' 's.

So: the **CCS** attack does not help him!

Plan

- 
1. Definition of the **CPA security**
 2. Constructions of **CPA-secure RSA** encryption schemes
 1. theoretical
 2. practical
 3. The **hybrid encryption** and the **KEM/DEM** paradigm
 4. Definition of the **CCA security**
 5. Constructions of **CCA-secure** symmetric encryption
 6. Constructions of **CCA-secure RSA** encryption schemes

PKCS #1 v.2 is not CCA-secure

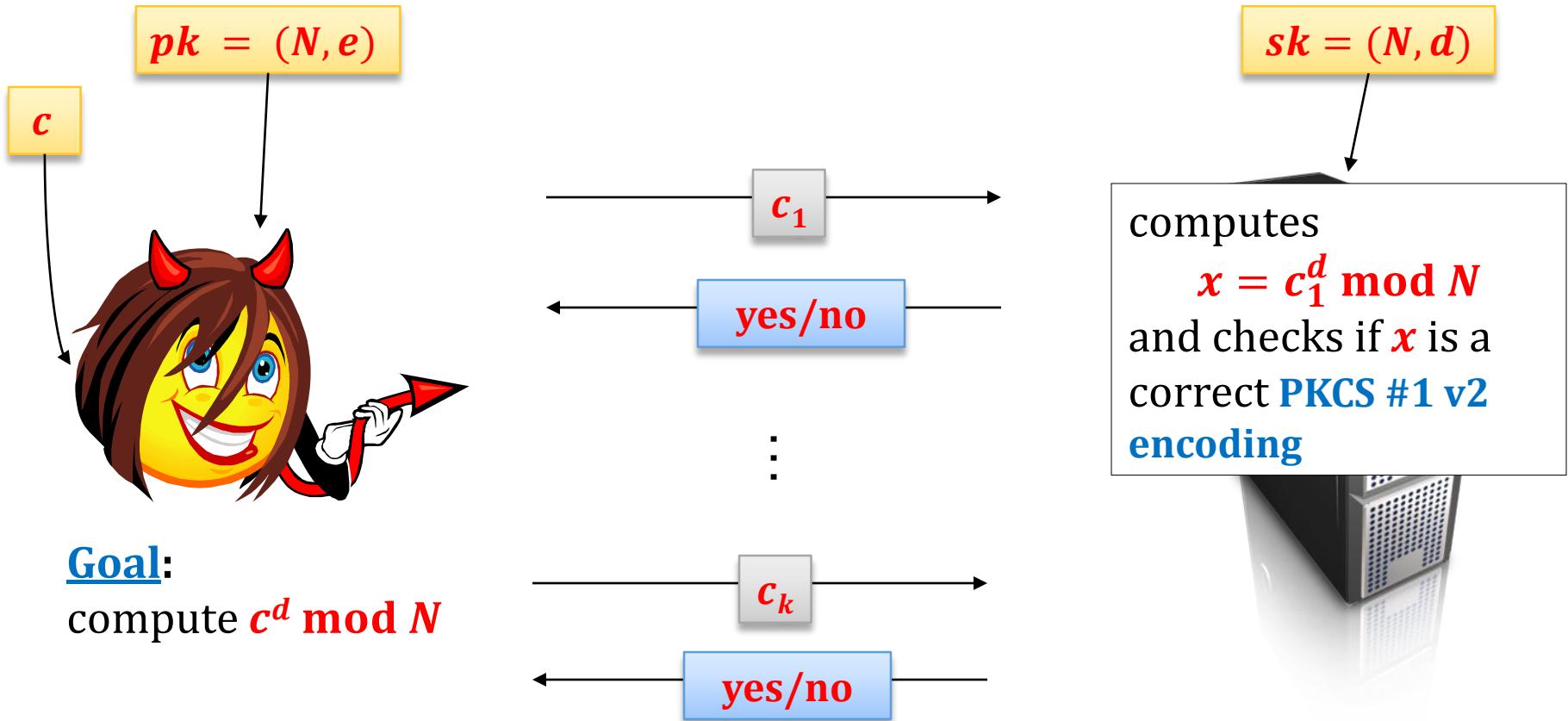
Bleichenbacher [1998] showed a “practical” chosen ciphertext attack on encoding proposed for the PKCS #1 v.2 standard.

[see also: Bleichenbacher, D., Kaliski B., Staddon J., "Recent results on PKCS #1: RSA encryption standard", *RSA Laboratories' bulletin* #7,
<ftp://ftp.rsasecurity.com/pub/pdfs/bulletn7.pdf>]

Why is Blaichenbacher's attack practical?

Because it assumes that the adversary can get only one bit of information about the plaintext...

Bleichenbacher's attack – the scenario



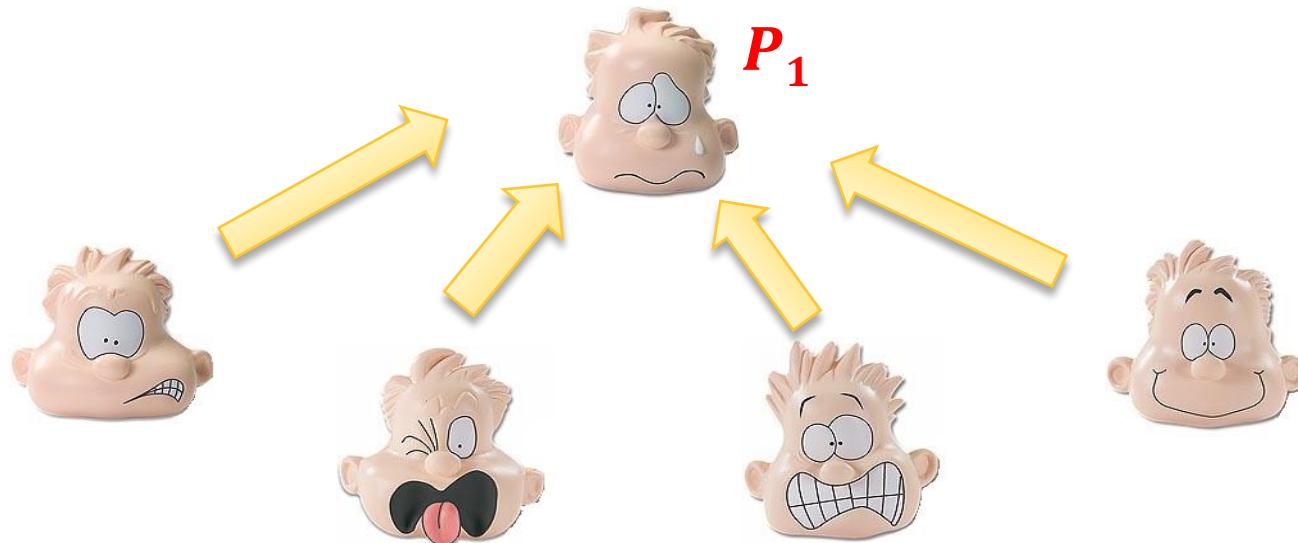
Bleichenbacher [1998]:

There exists a successful attack that requires $k = 2^{20}$ questions for $|N| = 1024$.

How to construct CCA-secure encryption scheme from RSA?

Observation: MACs don't help (at least directly).

Because in the asymmetric case the parties don't share a key for a MAC.



First attempt

Idea: take a **symmetric-key CCA-secure** scheme **(Enc', Dec')** and use it in the **KEM/DEM** method.

r is random from \mathbb{Z}_N^*

public key: (N, e)

private key: (N, d)

$$\text{Enc}((N, e), m) := (r^e \bmod N, \text{Enc}'(r, m))$$

$$\text{Dec}((N, d), (c_0, c_1)) := \text{Dec}'(c_0^d \bmod N, c_1)$$

Problem

$$\text{Enc}((N, e), m) := (r^e \bmod N, \text{Enc}'(r, m))$$

$|N|$ is normally much larger than the length of a key for symmetric encryption.

Typically $|N| = 1024$ and length of the symmetric key is **128**.

First idea: **truncate**.

But is it secure?

It may be the case that

- **RSA** is hard to invert, but
- **128** first bits are easy to compute...

Idea: instead of truncating – hash!

t – length of the symmetric key

$H: \{0, 1\}^* \rightarrow \{0, 1\}^t$ – a hash function

$$\text{Enc}((N, e), m) := (r^e \bmod N, \text{Enc}'(H(r), m))$$

$$\text{Dec}((N, d), (c_0, c_1)) := \text{Dec}'(H(c_0^d \bmod N), c_1)$$

But can we prove anything about it?

depends...

Which properties should H have?

If we just assume that H is collision-resistant we cannot prove anything...

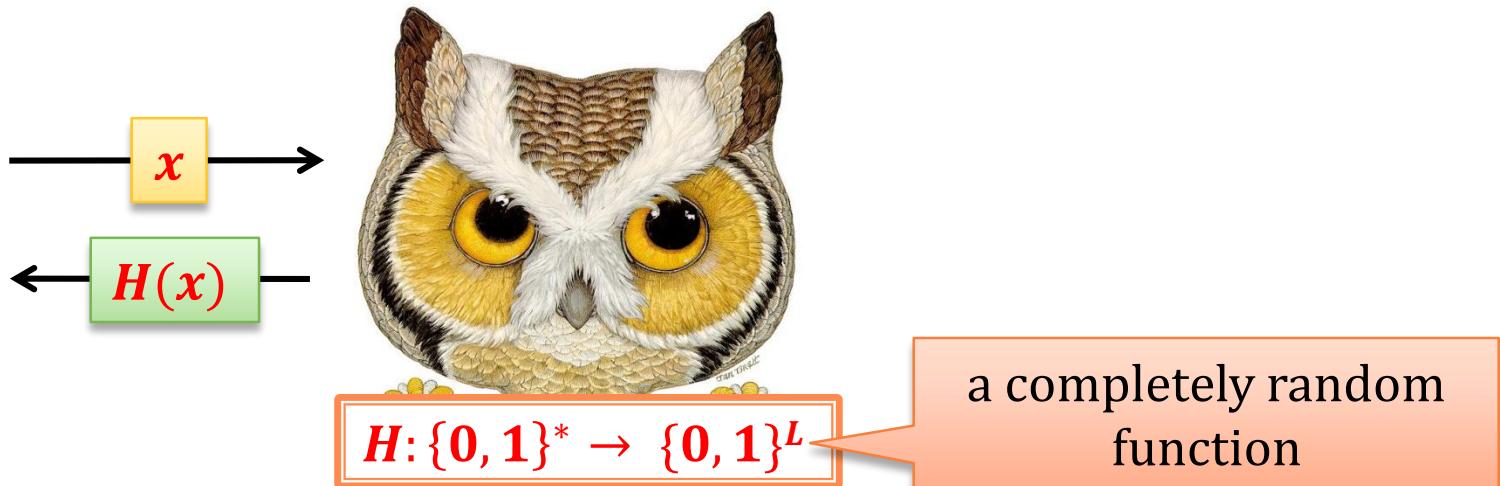
We have to assume that H “outputs **random values** on different inputs”.

This can be formalized by modeling H as **random oracle**.

Remember the **Random Oracle Model**?

Random oracle model

hash functions \approx **random oracles**



Security proof – the intuition

H – a hash function $\text{Enc}((N, e), m) := (r^e \bmod N, \text{Enc}'(H(r), m))$

Why is this scheme secure in the **random oracle model**?

Because, as long as the adversary did not query the oracle on r , the value of $H(r)$ is completely random.

To learn r the adversary would need to compute it from $r^e \bmod N$, so he would need to invert **RSA**.

So (with a very high probability) from the point of view of the adversary $H(r)$ is random.

Therefore the **CCA-security** of (Enc, Dec) follows from the **CCA-security** of $(\text{Enc}', \text{Dec}')$.

A drawback of this method

$$\text{Enc}((N, e), m) := (r^e \bmod N, \text{Enc}'(H(r), m))$$

The ciphertext is longer than N even if the message is short.

Therefore in practice another method is used:

Optimal Asymmetric Encryption Padding (OAEP).

Optimal Asymmetric Encryption Padding (OAEP) – the history

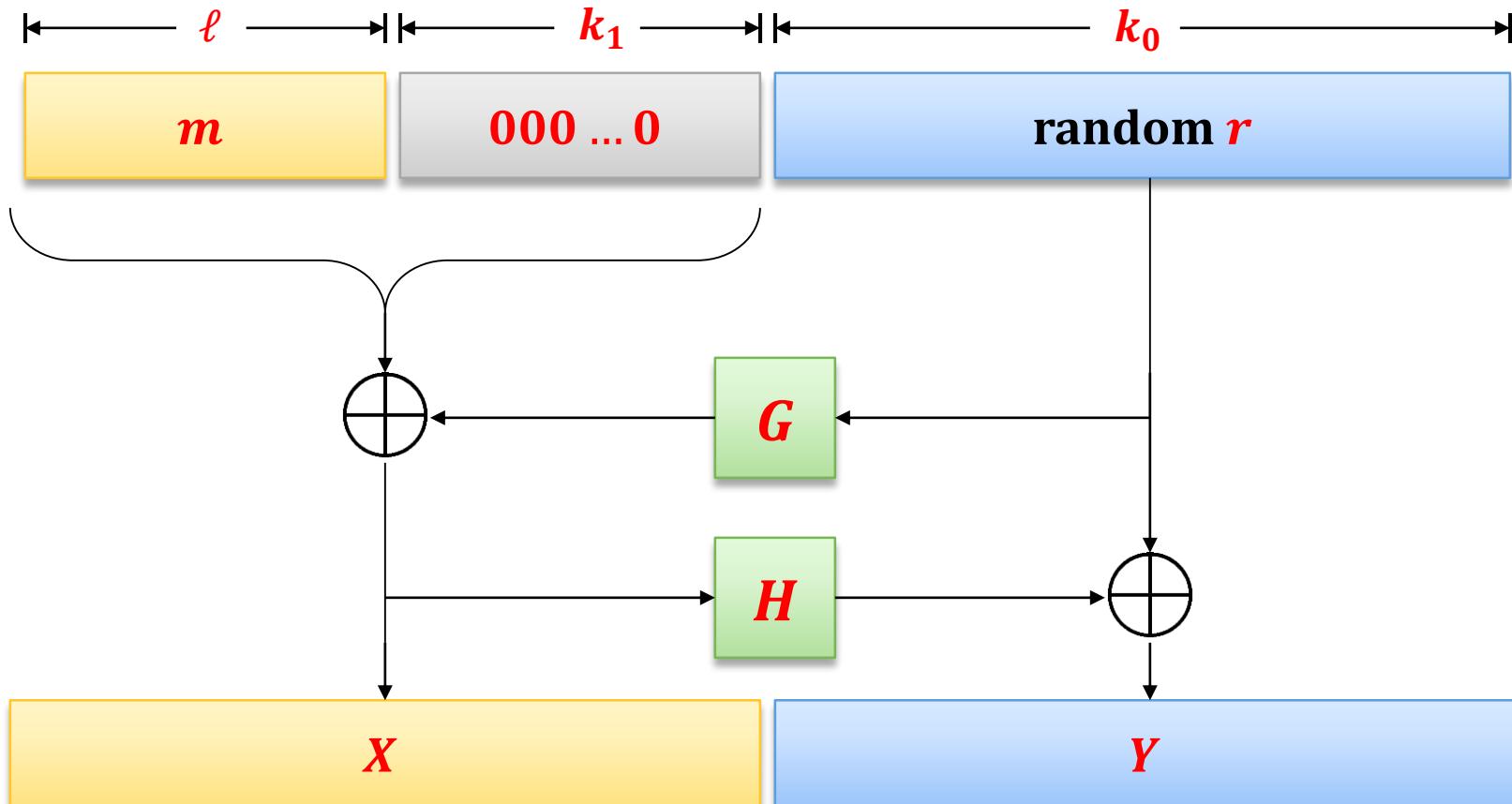
- Introduced in:
[M. Bellare, P. Rogaway. *Optimal Asymmetric Encryption -- How to encrypt with RSA.* Eurocrypt '94]
- An error in the security proof was spotted in
[V. Shoup. *OAEP Reconsidered.* Crypto '01]
- This error was repaired in
[E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. *RSA-OAEP is secure under the RSA assumption.* Crypto '01]

It is now a part of a **PKCS#1 v. 2.0** standard.

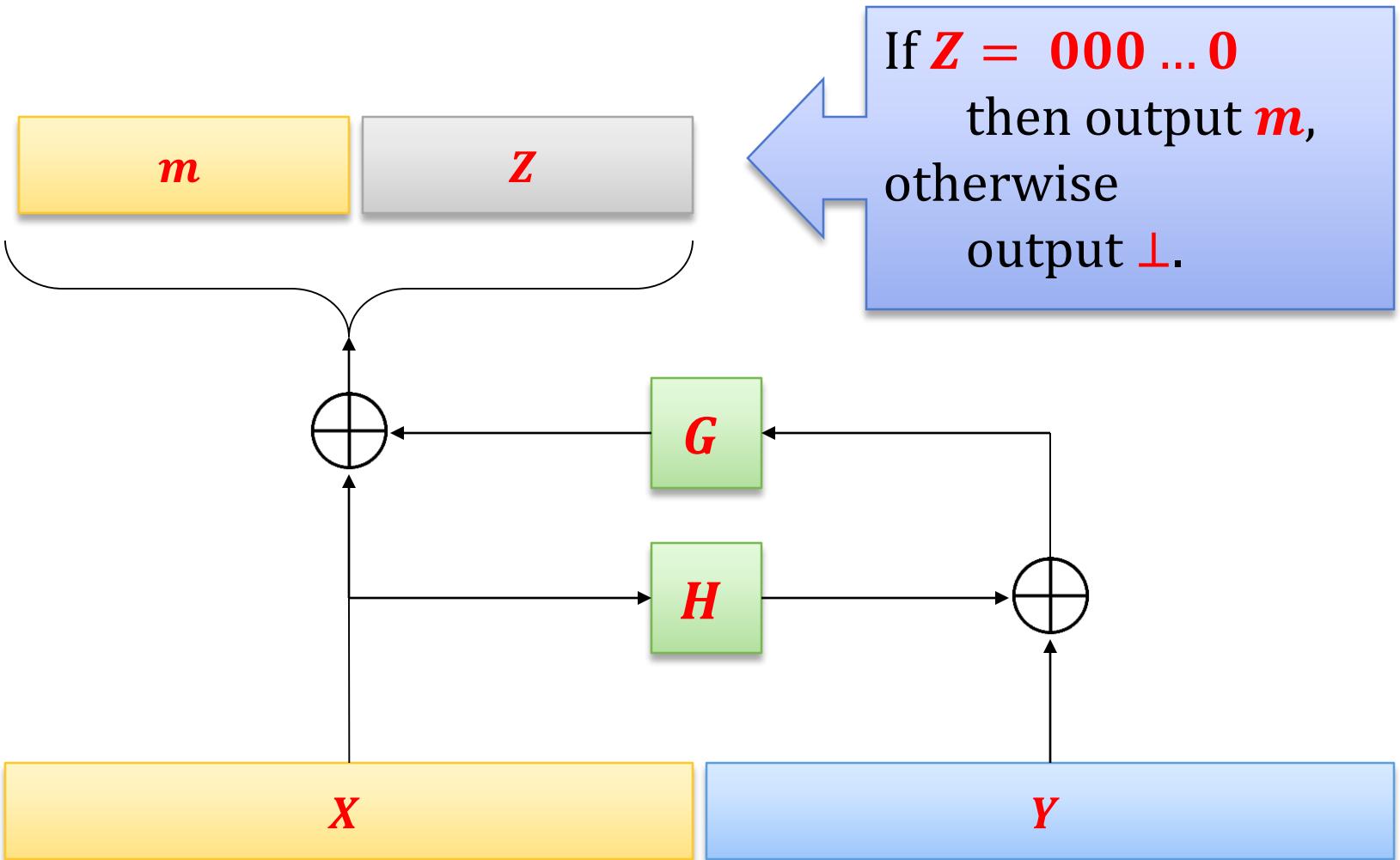
OAEP

OAEP(m) :=

N – RSA modulus
 ℓ, k_1, k_2 – parameters such that
 $\ell + k_1 + k_2 \leq \lceil \log_2 N \rceil$
hash functions:
• $G: \{0, 1\}^{k_0} \rightarrow \{0, 1\}^{\ell+k_1}$,
• $H: \{0, 1\}^{\ell+k_1} \rightarrow \{0, 1\}^{k_0}$



How to invert?



RSA-OAEP

key pair like in the **handbook RSA**:

private key: (N, d)

public key: (N, e)

$\text{Enc}((N, e), m) := (\text{OAEP}(m))^e \bmod N$

$\text{Dec}((N, e), c) :=$ let $x := c^d \bmod N$
if $x > 2^{\ell+k_1+k_2}$ then output \perp
otherwise output $\text{OAEP}^{-1}(x)$

Security of RSA-OAEP

Security of **RSA-OAEP** can be proven

- if one models **H** and **G** as random oracles
- assuming the **RSA assumption** holds.

We do not present the proof here.

We just mention some **nice properties** of this encoding.

Nice properties of OAEP (for the right choice of parameters)

good for
the **CPA-**
security

good for
the **CCA-**
security

- it is **invertible**
- but **to invert you need to know (X, Y) completely**
- for every message m the encoding **$OAEP(m)$** is uniformly **random**
- It is hard to produce a valid (X, Y) “without knowing m first”

OAEP is hard to invert if you don't know
X and **Y** completely.

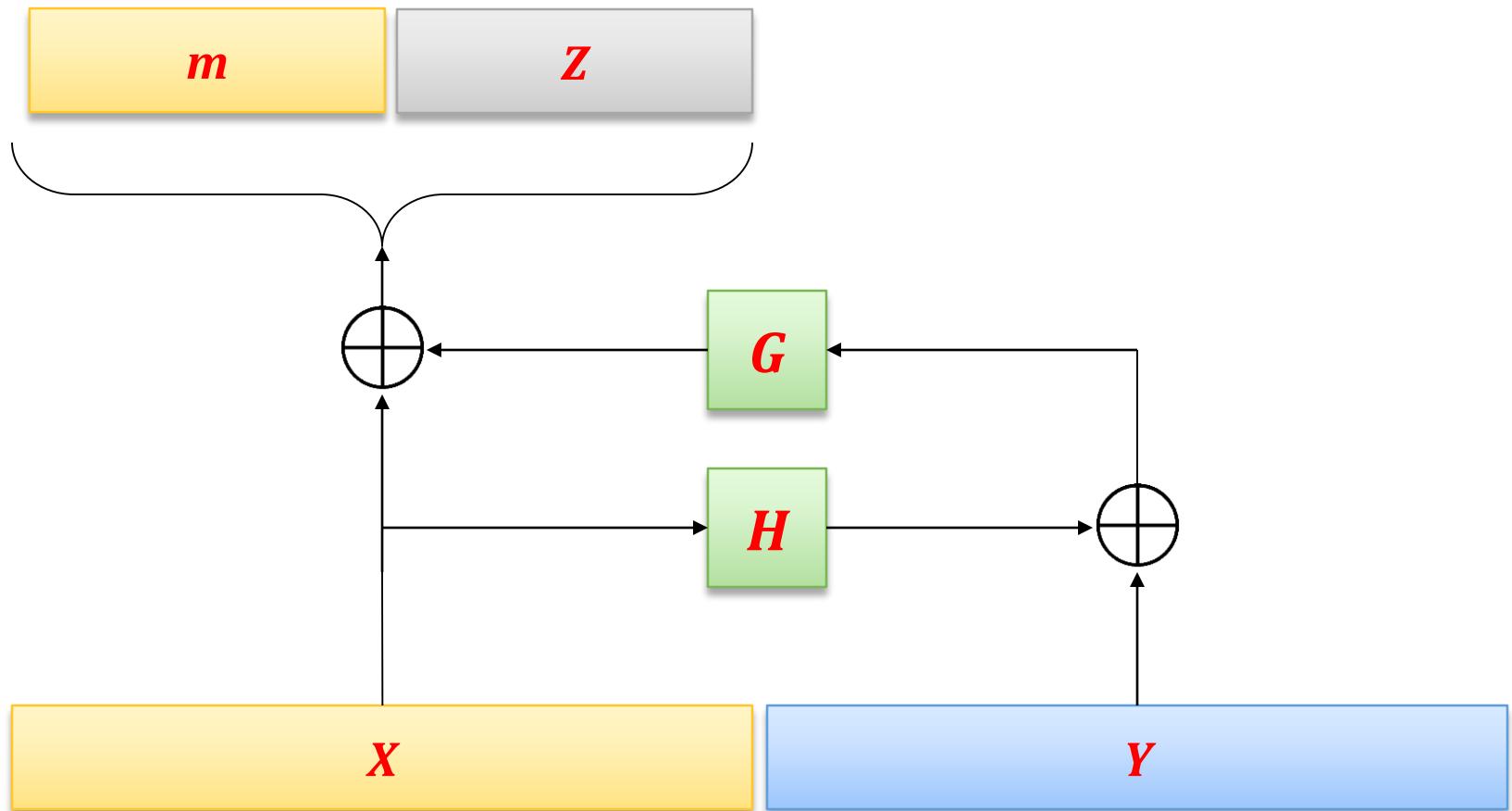
Actually:

m is completely hidden in such a case.

(assuming **G** and **H** are random oracles)

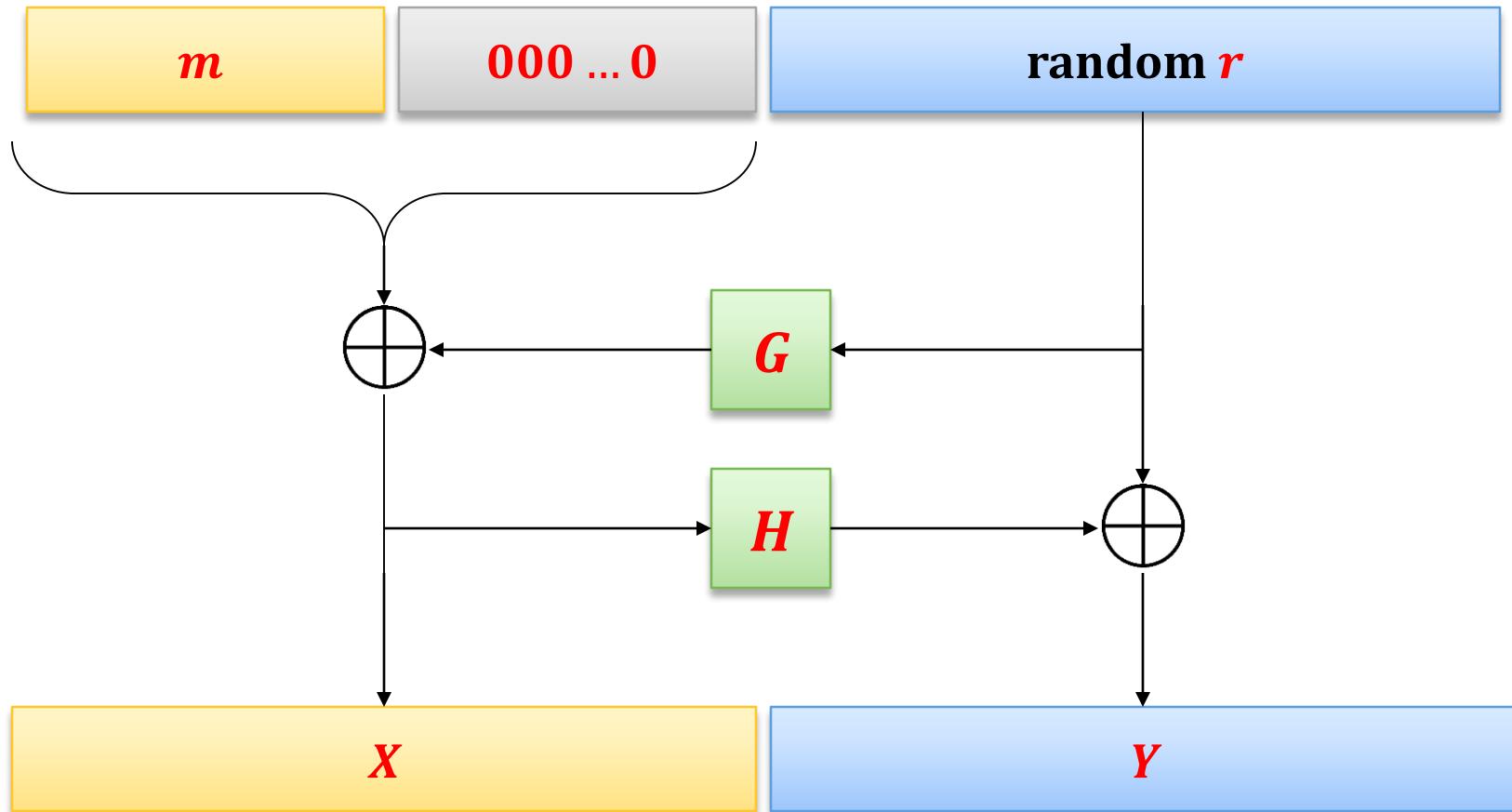
Why?

Look at the picture:



The encoding $\text{OAEP}(m)$ is uniformly **random**

Again look at the picture:



Why are these two properties useful for **CPA-security**?

The adversary obtains

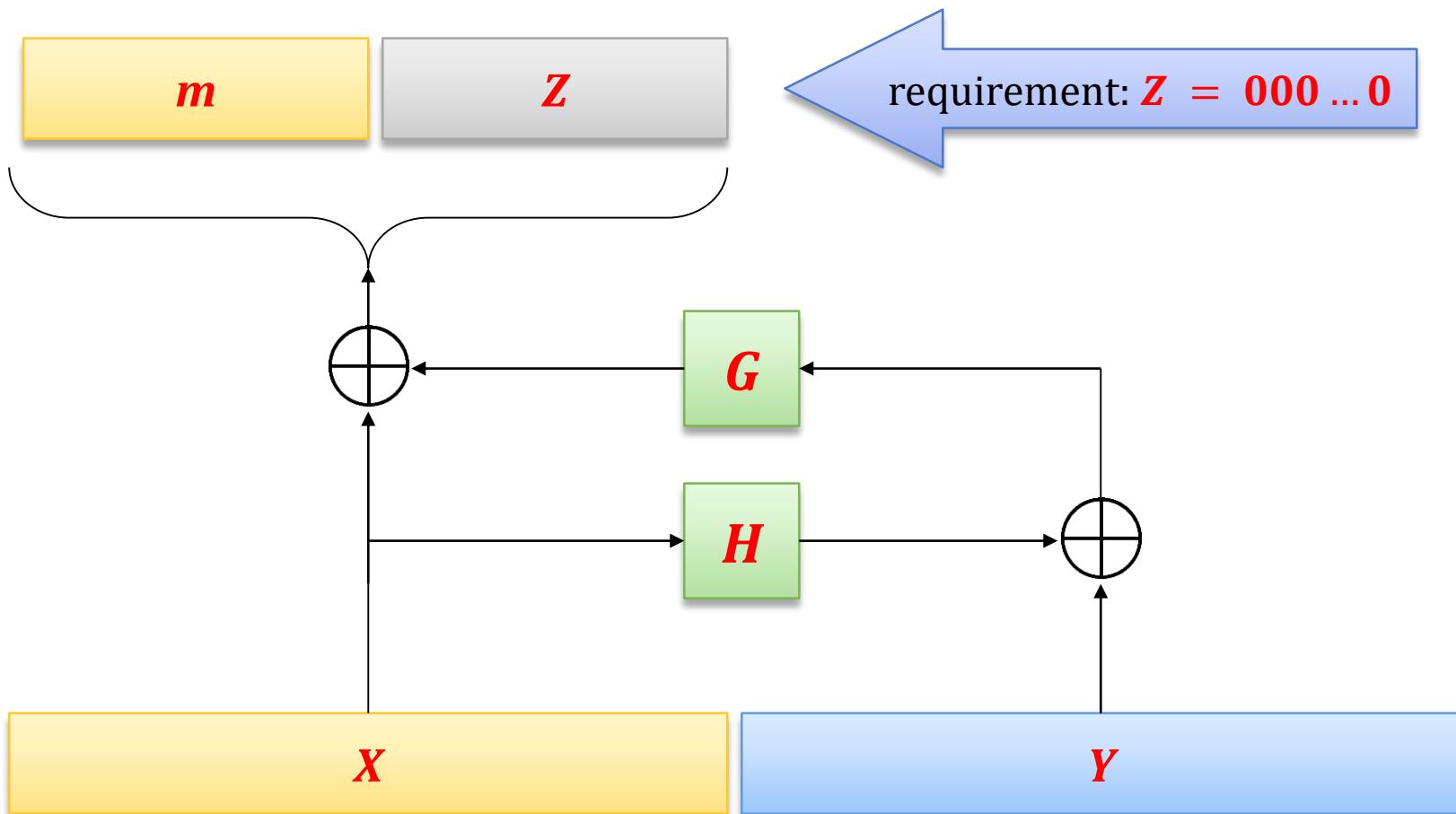
$$\text{Enc}_{pk}(m_b) = x^e \bmod N$$

where $x = \text{OAEP}(m_b)$.

In order to get **any information about m_b** needs to compute the **entire value of x** , where x is uniformly random.

Hardness of this problem is equivalent to the **RSA assumption**.

It is hard to produce a valid (X, Y) “without knowing m first”



This last property is useful for CCA-security

Why?

Informally:

Eve can produce valid ciphertexts only of those messages that she knows...

The only way to produce a valid ciphertext is to do the following:

- choose \mathbf{m}
- compute $\mathbf{c} := (\text{OAEP}(\mathbf{m}))^e \text{ mod } N.$

Note

In “handbook RSA” this is not the case since every $c \in Z_N^*$ is a valid ciphertext.

Also in the **PKCS #1: RSA Encryption Standard Version 1.5** standard the probability of producing a valid ciphertext is noticeable.

An interesting attack on OAEP

J. Manger: **A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0.** CRYPTO 2001

Based on the following fact:

the decryption algorithm outputs \perp in two cases:

1. “ $x > 2^{\ell+k_1+k_2}$ ”,
2. or $Z \neq 000 \dots 0$.

The attack exploits the fact that in the **PKCS #1 v2.0** standard the **error messages in these two cases were different**.

Moral: implementation details matter!

©2016 by Stefan Dziembowski. Permission to make digital or hard copies of part or all of this material is currently granted without fee *provided that copies are made only for personal or classroom use, are not distributed for profit or commercial advantage, and that new copies bear this notice and the full citation.*