

Lecture 6

Public-Key Encryption

Stefan Dziembowski
University of Rome
La Sapienza



SAPIENZA
UNIVERSITÀ DI ROMA

BiSS 2009
Bertinoro International
Spring School
2-6 March 2009



Plan



1. Problems with the handbook RSA encryption
2. Security definitions
3. How to encrypt with RSA?
4. Encryption based on discrete-log
 1. first step: Diffie-Hellman key exchange
 2. ElGamal encryption
5. Public-key vs. private key encryption

The “handbook RSA encryption”

$N = pq$ - RSA modulus

e is such that **$\gcd(e, \phi(N)) = 1$,**

d is such that **$ed = 1 \pmod{\phi(N)}$**

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

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

Problems

Enc_{pk} is deterministic, so:
if one encrypts twice the same message then the
ciphertexts are the same

Therefore if the message space M is small, the adversary can
check all possible messages:

given a ciphertext c do:
for every $m \in M$ check if $\text{Enc}_{pk}(m) = c$

for example if $M = \{\text{yes}, \text{no}\}$, then the encryption is not
secure.

RSA has some “algebraic properties”.

Algebraic properties of RSA

1. RSA is homomorphic:

$$\begin{aligned}\text{Enc}_{(e,N)}(m_0 \cdot m_1) &= (m_0 \cdot m_1)^e \\ &= m_0^e \cdot m_1^e \\ &= \text{Enc}_{(e,N)}(m_0) \cdot \text{Enc}_{(e,N)}(m_1)\end{aligned}$$

why is it bad?

By checking if

$$c_0 \cdot c_1 = c$$

the adversary can detect if

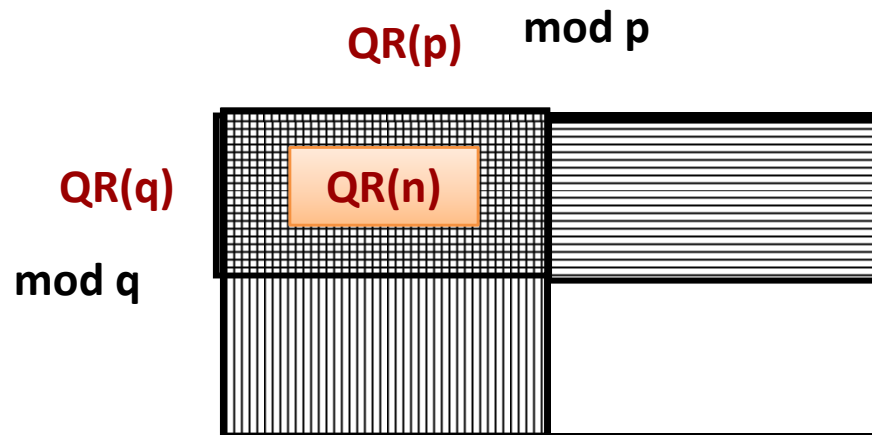
$$\text{Dec}_{(d,N)}(c_0) \cdot \text{Dec}_{(d,N)}(c_1) = \text{Dec}_d(c)$$

2. The **Jacobi symbol** leaks.

Jacobi Symbol

for any prime p define $J_p(x) := \begin{cases} +1 & \text{if } x \in \text{QR}_p \\ -1 & \text{otherwise} \end{cases}$

for $N=pq$ define $J_N(x) := J_p(x) \cdot J_q(x)$



$J_N(x) :=$

+1	-1
-1	+1

It is a subgroup of Z_N^*

$$Z_N^+ := \{x : J_N(x) = +1\}$$

Jacobi symbol can be computed efficiently! (even in p and q are unknown)

Fact

$\text{Enc}_{(N,e)}$ – encryption function of **RSA**

Then:

$$J_n(\text{Enc}_{(N,e)}(m)) = J_n(m)$$

Proof

Left-hand-side:

$$J_n(\text{Enc}_{(N,e)}(m)) = J_p(\text{Enc}_{(N,e)}(m)) \cdot J_q(\text{Enc}_{(N,e)}(m))$$

Right-hand-side:

$$J_n(m) = J_p(m) \cdot J_q(m)$$

Therefore it is enough to prove that:

$$J_p(\text{Enc}_{(N,e)}(m)) = J_p(m)$$

and

$$J_q(\text{Enc}_{(N,e)}(m)) = J_q(m).$$

Since **p** and **q** are symmetric we can just prove it for **p**.

We have to show that

$$J_p(\text{Enc}_e(m)) = J_p(m)$$

In other words:

$$\text{Enc}_{(N,e)}(m) \text{ is a } \text{QR}_p \\ \text{iff} \\ m \text{ is a } \text{QR}_p$$

This is equal to
 $m^e \bmod p$

Now observe that e is always odd

(because $\text{gcd}(e, (p-1)(q-1)) = 1$).

To finish the proof we need to show that for every odd e :

$$m^e \bmod p \text{ is a } \text{QR}_p \\ \text{iff} \\ m \bmod p \text{ is a } \text{QR}_p$$

Fact

For an odd e :

$$\begin{aligned} m^e \bmod p \text{ is a } QR_p \\ \text{iff} \\ m \bmod p \text{ is a } QR_p \end{aligned}$$

Proof

Let g be the generator of Z_p^* . Write $m = g^x$.

Recall that m is a QR_p iff x is an even.

It is easy to see that

$$\begin{aligned} m^e \bmod p \text{ is a } QR_p \\ \text{iff} \\ (g^x)^e \bmod p \text{ is a } QR_p \\ \text{iff} \\ x \cdot e \bmod (p-1) \text{ is even} \\ \text{iff} \\ x \bmod (p-1) \text{ is even} \\ \text{iff} \\ m \bmod p \text{ is a } QR_p \end{aligned}$$

Hence we are done.

Conclusion

The Jacobi symbol “leaks”, i.e.:

from c

one can compute $J_n(\text{Dec}_{(N,d)}(c))$

(without knowing the factorization of N)

Is it a big problem?

Depends on the application...

Question: Is RSA secure?

Looks like it has some weaknesses...

Plan:

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

Plan

1. Problems with the handbook RSA encryption
2. Security definitions
3. How to encrypt with RSA?
4. Encryption based on discrete-log
 1. first step: Diffie-Hellman key exchange
 2. ElGamal encryption
5. Public-key vs. private key encryption



A mathematical view

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

- **Gen** is a **key-generation** randomized algorithm that takes as input a security parameter 1^n and outputs a key pair **(pk,sk)**.
- **Enc** is an **encryption** algorithm that takes as input the **public key pk** and a **message m**, and outputs a **ciphertext c**,
- **Dec** is an **decryption** algorithm that takes as input the **private key pk** and the **ciphertext c**, and outputs a **message m'**.

We will sometimes write **$Enc_{pk}(m)$** and **$Dec_{sk}(c)$** instead of **$Enc(pk,m)$** and **$Dec(sk,c)$** .

Correctness

$P(Dec_{sk}(Enc_{pk}(m)) \neq m)$ is negligible in **n**

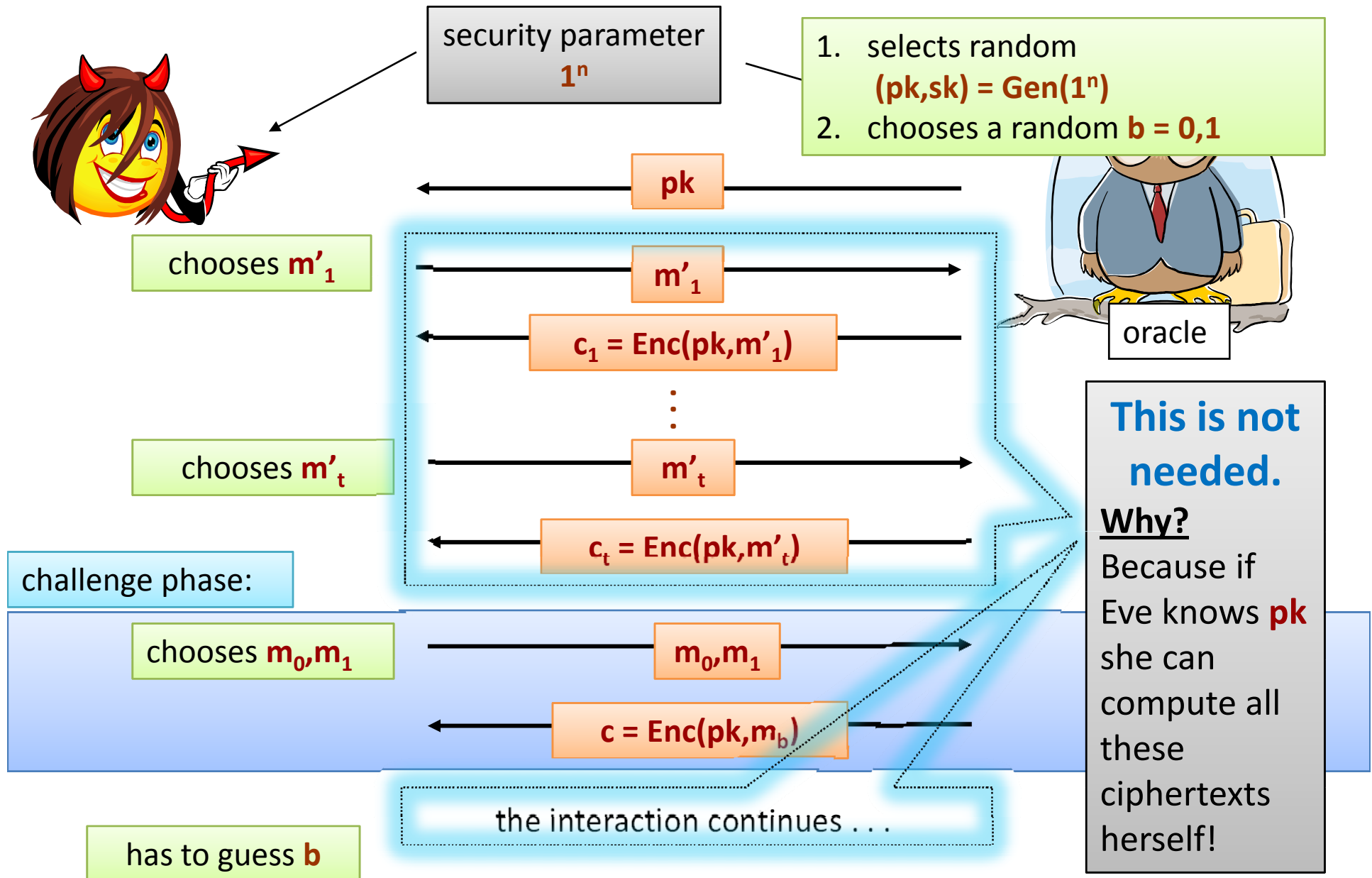
The security definition

Remember the symmetric-key case?

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

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

A chosen-plaintext attack (CPA)



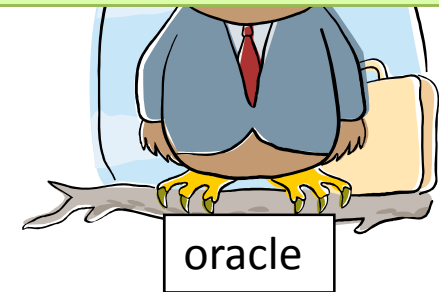
A simplified view



security parameter
 1^n

1. selects random $(pk, sk) = \text{Gen}(1^n)$
2. chooses a random $b = 0, 1$

pk



oracle

challenge phase:

chooses m_0, m_1

m_0, m_1

$c = \text{Enc}(pk, m_b)$

has to guess b

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 **$0.5 + \epsilon(n)$** , where **ϵ** is negligible.

Plan

1. Problems with the handbook RSA encryption
2. Security definitions
3. How to encrypt with RSA?
4. Encryption based on discrete-log
 1. first step: Diffie-Hellman key exchange
 2. ElGamal encryption
5. Public-key vs. private key encryption



Is the “handbook RSA” secure?

the “handbook RSA”

$N = pq$ - RSA modulus

e is such that $\gcd(e,d) = 1$, d is such that $ed = 1 \pmod{\phi(N)}$

$\text{Enc}_{(N,e)}(m) = m^e \pmod N$, and $\text{Dec}_{(d,N)}(c) = c^d \pmod N$.

Not secure!

In fact:

No deterministic encryption scheme is secure.

How can the adversary win the game?

1. he just 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.

Encoding

Therefore, before encrypting a message we usually **encode it** (adding some randomness).

This has the following advantages:

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

How is it done in real-life?

PKCS #1: RSA Encryption Standard Version 1.5:

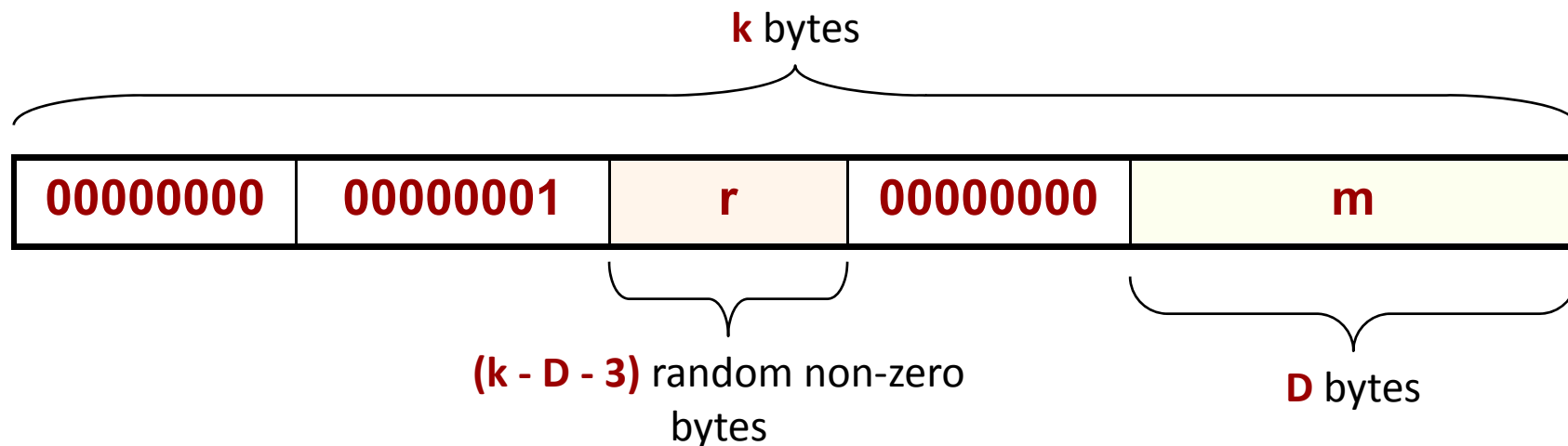
public-key: (N, e)

let k := length on N in bytes.

let D := length of the plaintext

requirement: $D \leq k - 11$.

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



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

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

It has however some weaknesses (it is not “chosen-ciphertext secure”).

Optimal Asymmetric Encryption Padding (OAEP) is a more secure encoding.

(we will discuss it later)

Plan

1. Problems with the handbook RSA encryption
2. Security definitions
3. How to encrypt with RSA?
4. Encryption based on discrete-log
 1. first step: Diffie-Hellman key exchange
 2. ElGamal encryption
5. Public-key vs. private key encryption



How to construct PKE based on the **hardness of discrete log?**

RSA was a trapdoor permutation, so the construction was quite easy...

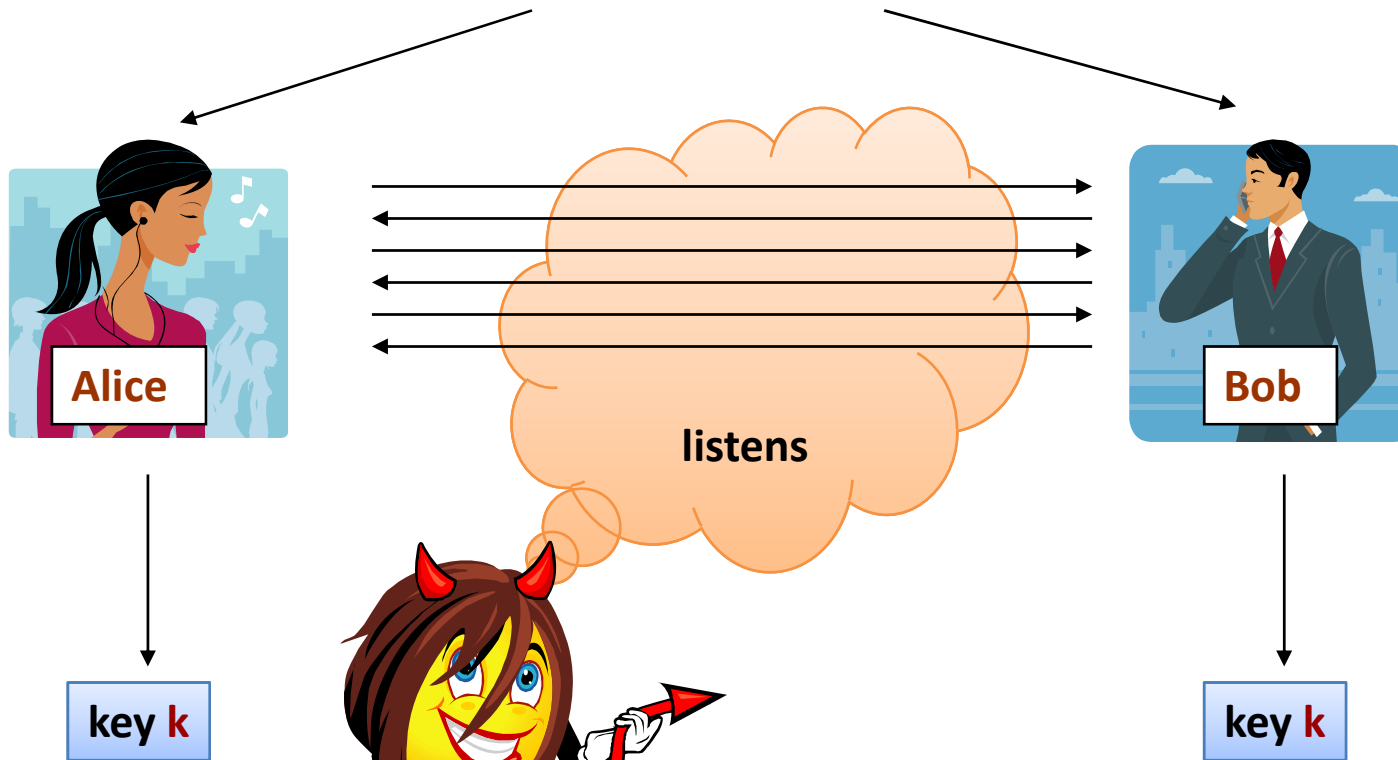
In case of the **discrete log**, we just have a one-way function.

Diffie and Hellman constructed something weaker than PKE: a **key exchange protocol** (also called **key agreement protocol**).

We'll not describe it. Then, we'll show how to "convert it" into a **PKE**.

Key exchange

initially they share no secret



Eve should have no information about **k**

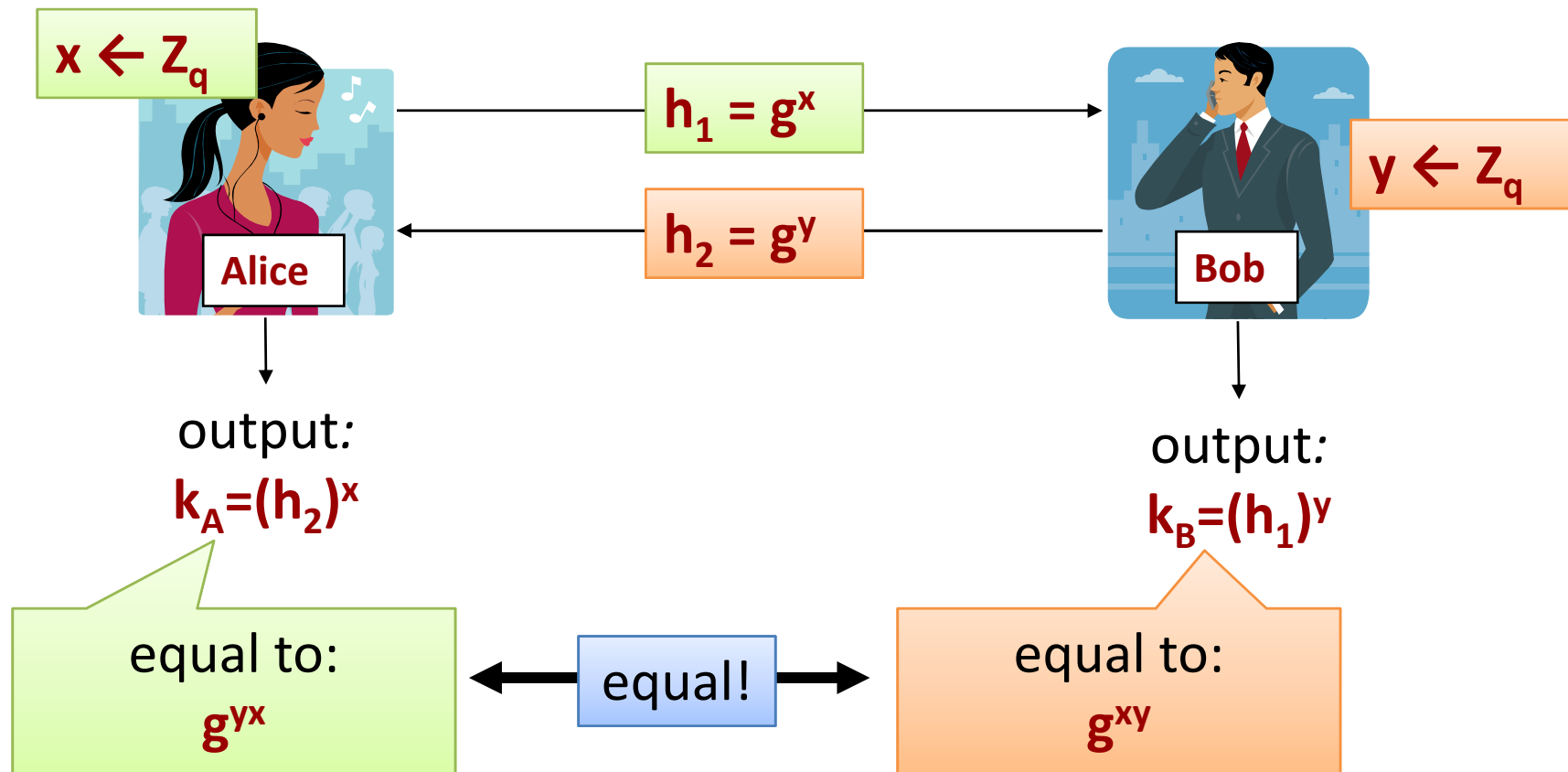
We will formalize it later.
Let's first show the protocol.

The Diffie-Hellman Key exchange

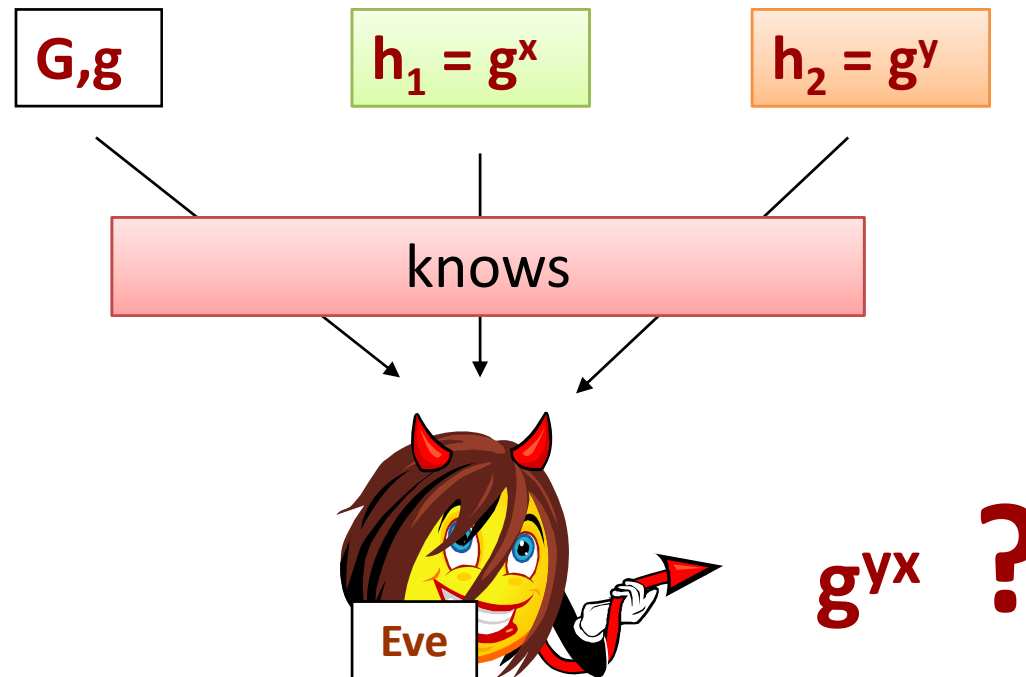
G – a group, where **discrete log is believed to be hard**

q = |G|

g – a generator of **G**



Security of the Diffie-Hellman exchange



Eve should have no information about g^{yx}

Is it secure?

If the **discrete log** in **G** is easy then the **DH key exchange** is **not** secure.

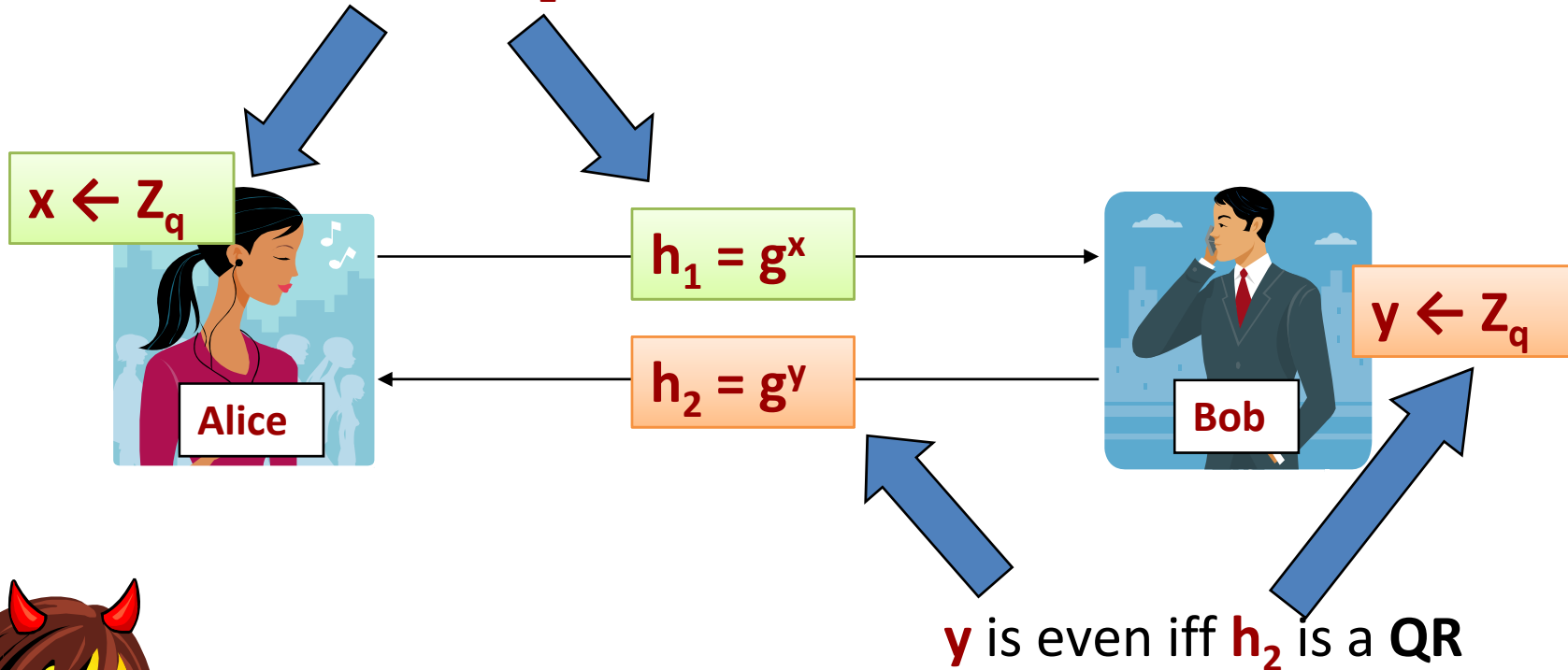
(because the adversary can compute **x** and **y** from **g^x** and **g^y**)

If the discrete log in **G** is hard, then...

it may also not be secure

Example: $G = Z_p^*$

x is even iff h_1 is a QR



Therefore:

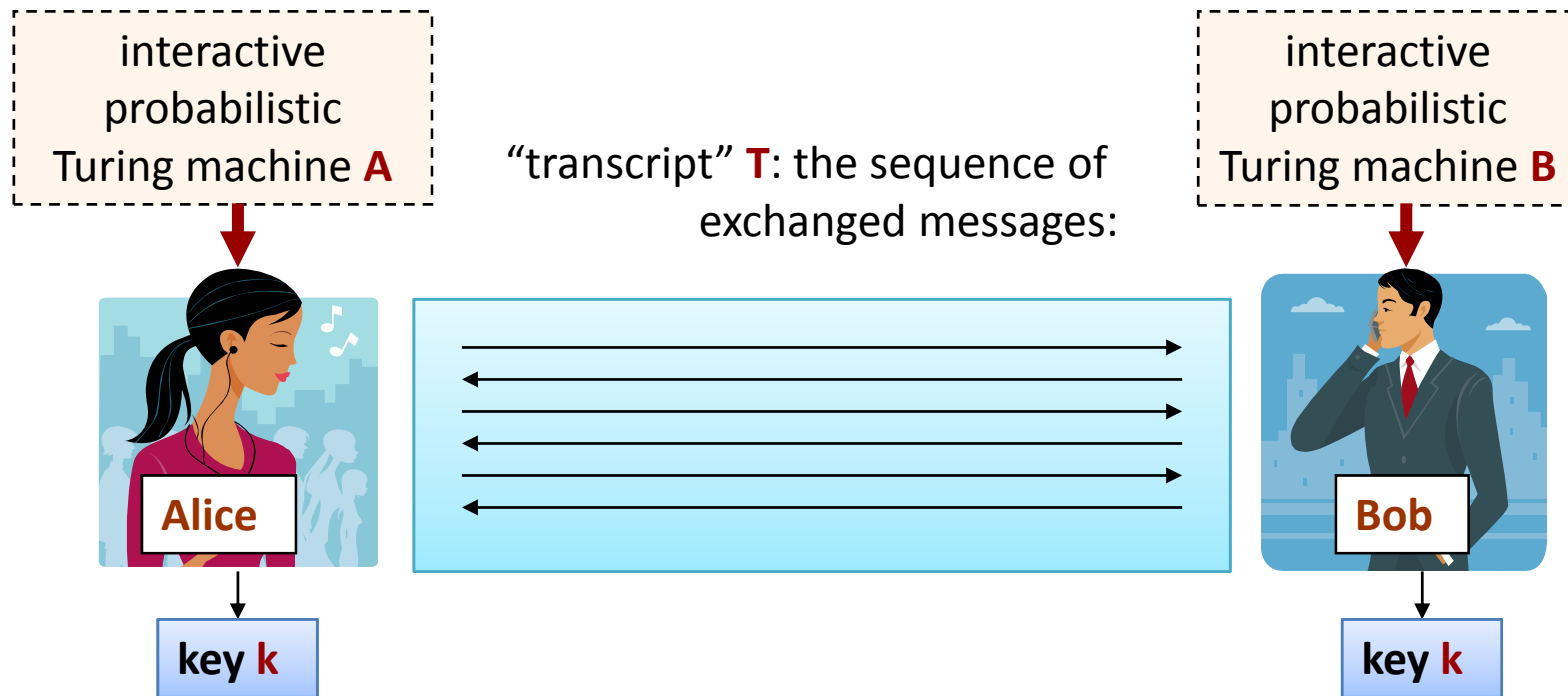
g^{yx} is a QR iff (h_1 is a QR) or (h_2 is a QR)

So, Eve can compute some information about g^{yx} (namely: if it is a QR, or not).

Is it a problem, or not?

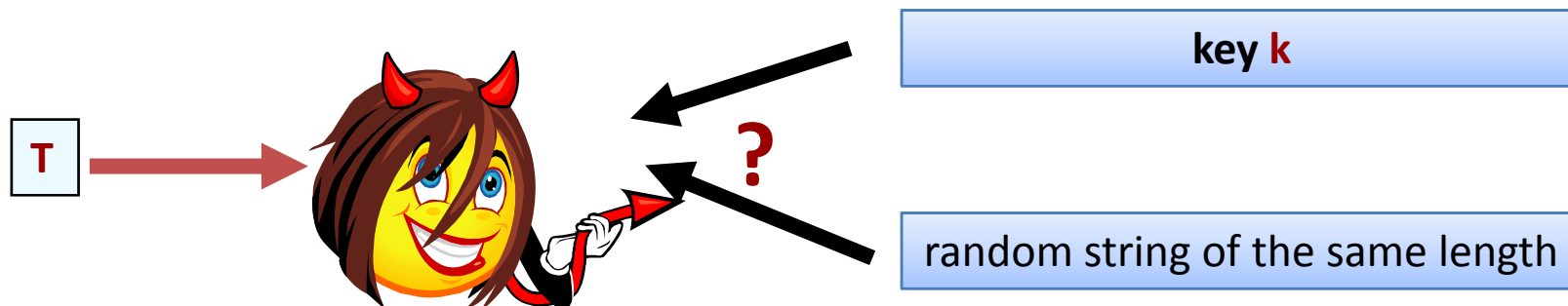
We need to

1. formalize what we mean by secure key exchange,
2. identify the assumptions needed to prove the security.

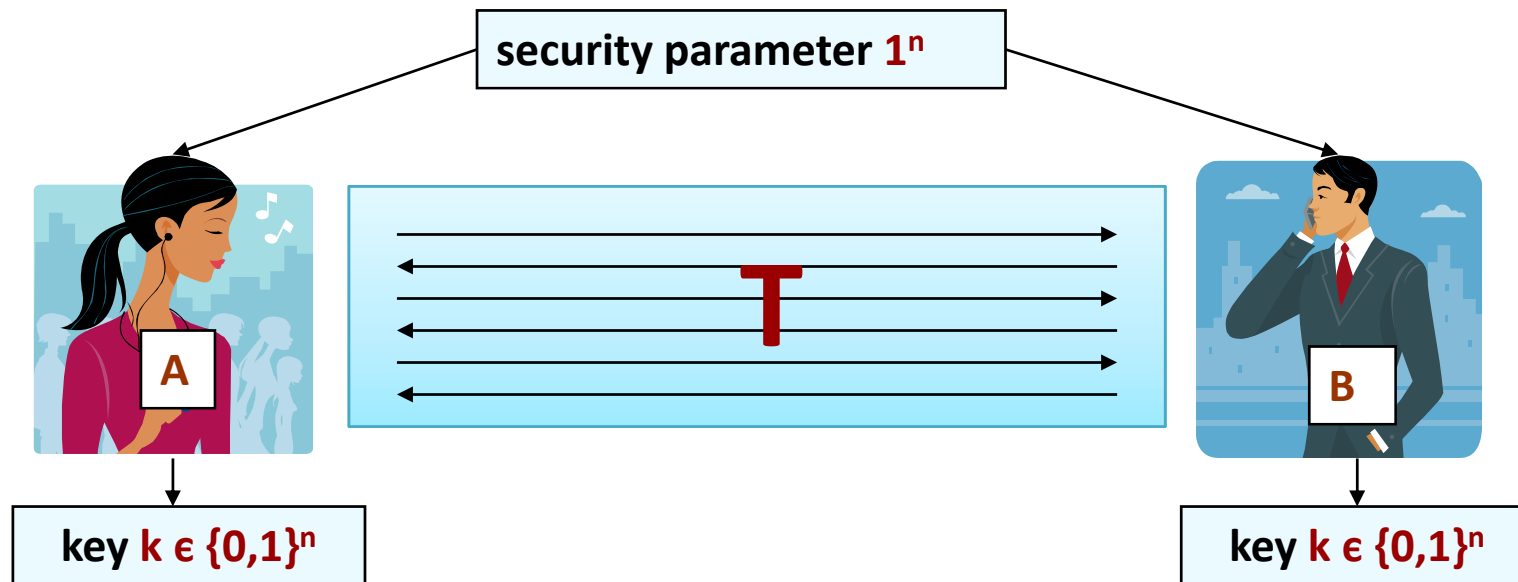


Informal definition:

(A,B) is **secure** if no "efficient adversary" can distinguish k from random, given T , with a "non-negligible advantage".



How to formalize it?



We say (A,B) is secure a secure key-exchange protocol if:
the output of A and B is always the same, and

$$\forall \left| \text{Prob} [M(1^n, T, k) = 1] - \text{Prob} [M(1^n, T, r) = 1] \right| \text{ is negligible in } n$$

polynomial-time M
that outputs 0 or 1

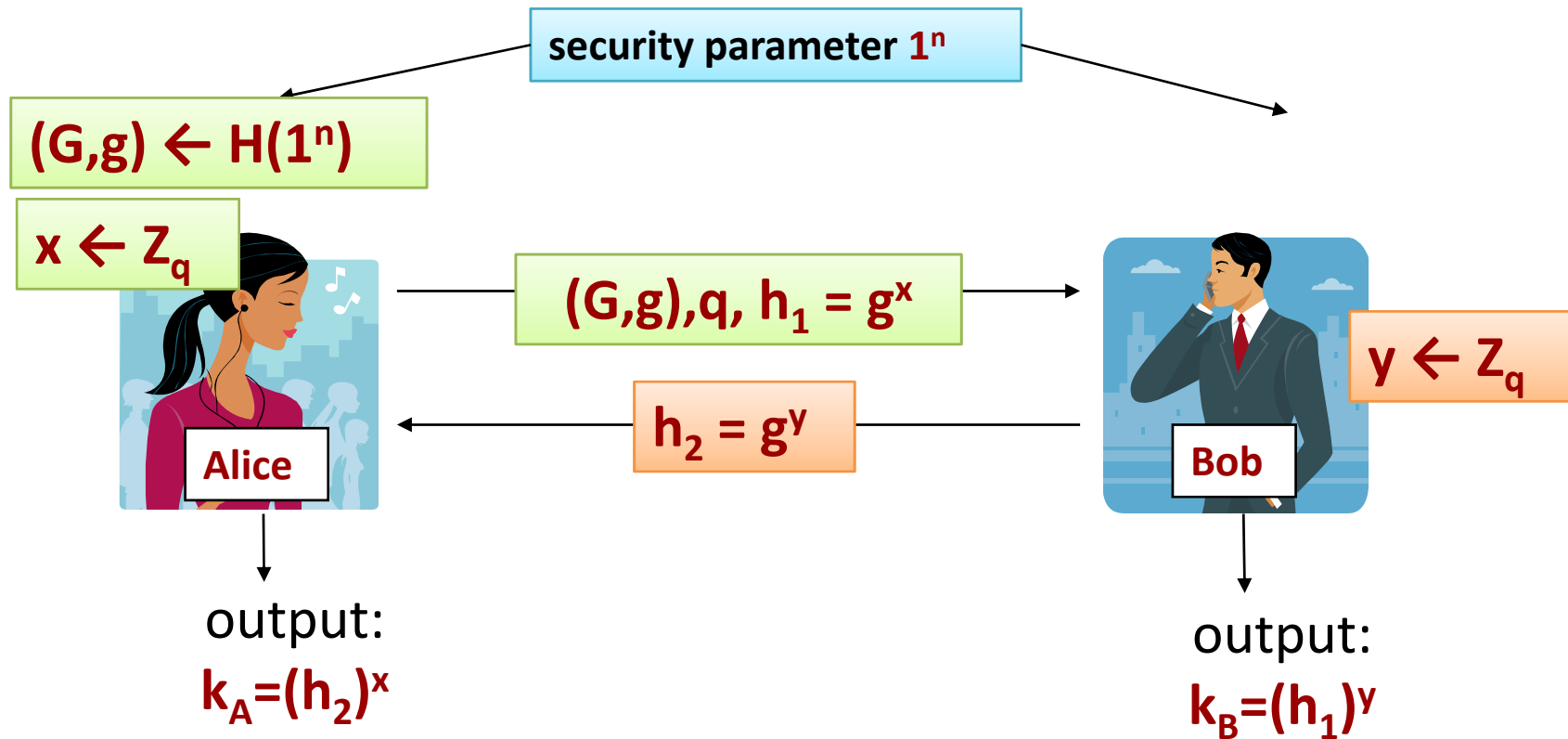
r is random and $|r| = n$

Remember the algorithm **H**?

Algorithm **H**:

- on input $\mathbf{1}^n$
- outputs:
 - a description of **G** of order **q**, such that $|\mathbf{q}| = n$,
 - a generator **g** of **G**.

How does the protocol look now?



(Note that we cheat a bit because k is a “pseudorandom” **group element**, not a **string of bits**.)

If such a key exchange protocol is secure, we say that: the **Decisional Diffie-Hellman (DDH) problem is hard with respect to H**)

An example of **H** where **DDH** is believed to be hard

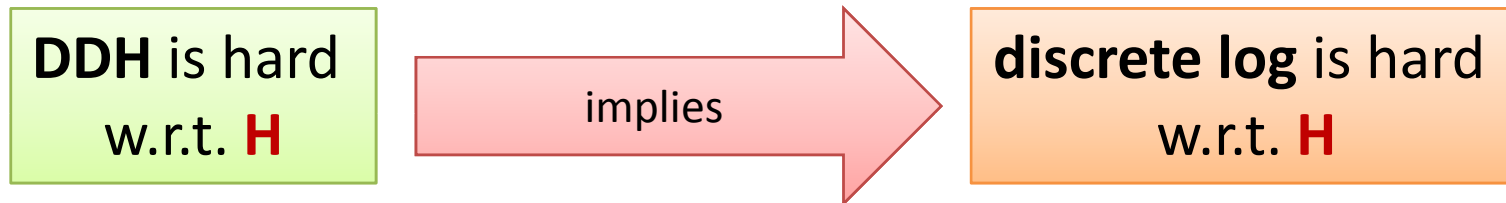
QR(p)

H(1ⁿ):

1. generate a random **strong** prime **p** of length **n+1**.
2. set **q := (p-1)/2**.
3. choose any **x ∈ Z_p^{*}** such that **x ≠ ±1 (mod p)**.
4. set **g := x² mod p**.
5. output **(p,g)**.

Other groups are also used (e.g. groups based on the elliptic curves).

How does DDH compare to the discrete log assumption



The opposite implication is unknown in most of the cases

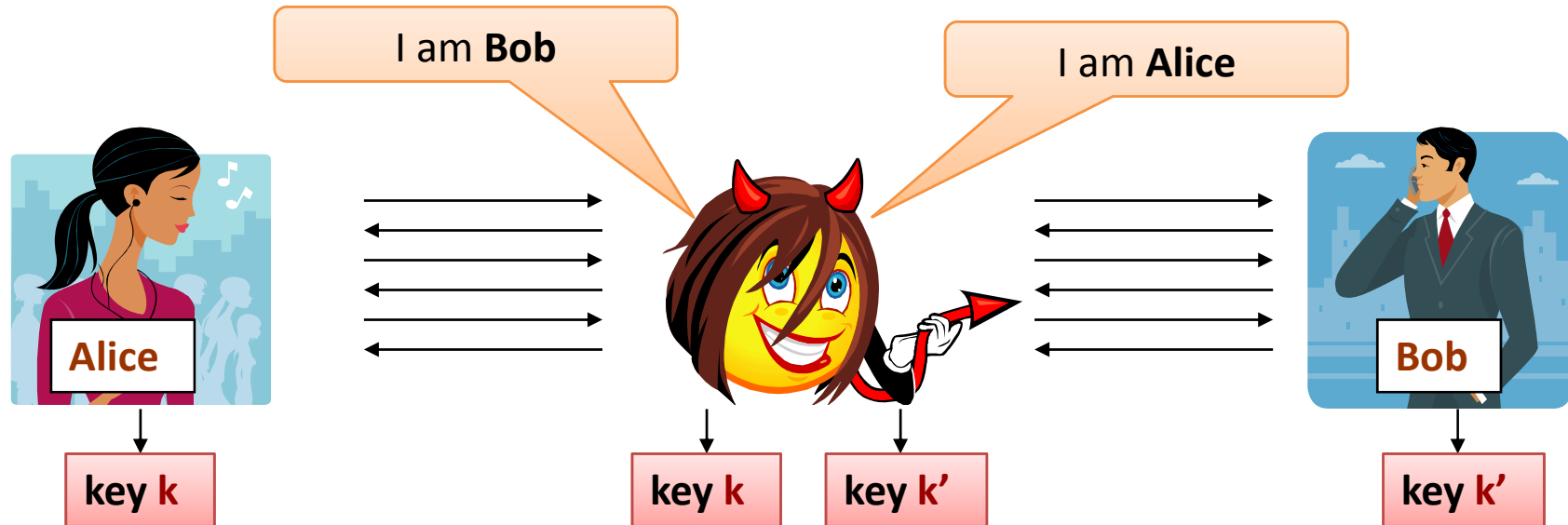
A problem

The protocols that we discussed are secure only against a **passive adversary** (that only eavesdrop).

What if the adversary is **active**?

She can launch a **man-in-the-middle** attack.

Man in the middle attack



A very realistic attack!

So, is this thing totally useless?

No! (it is useful as a building block)

Plan

1. Problems with the handbook RSA encryption
2. Security definitions
3. How to encrypt with RSA?
4. Encryption based on discrete-log
 1. first step: Diffie-Hellman key exchange
 2. ElGamal encryption
5. **Public-key vs. private key encryption**



El Gamal encryption

El Gamal is another popular public-key encryption scheme.

It is based on the **Diffie-Hellman** key-exchange.

First observation

Remember that the one-time pad scheme can be generalized to any group?

E.g.: $\mathcal{K} = \mathcal{M} = \mathcal{C} = \mathbf{G}$.

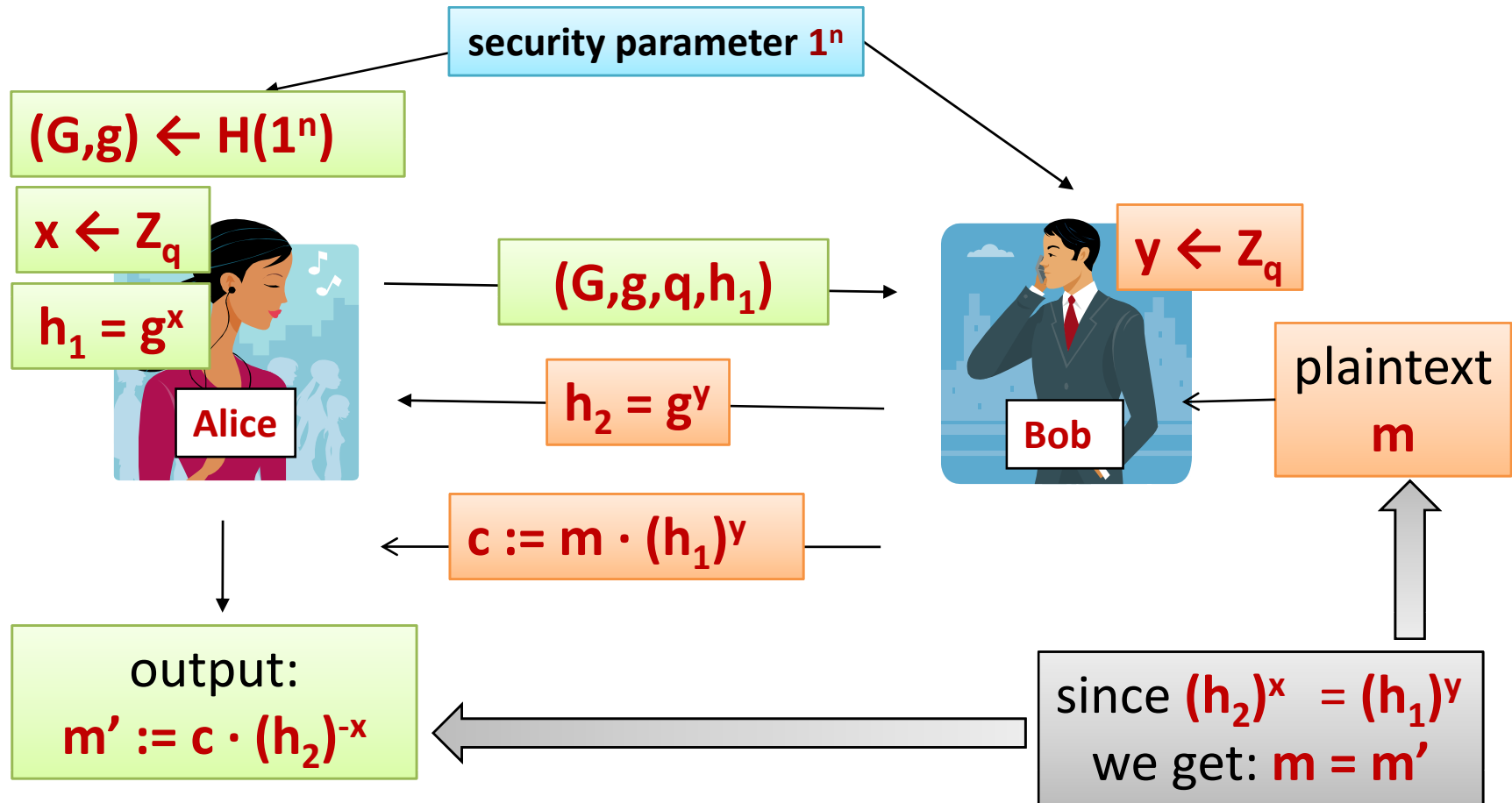
- $\text{Enc}(k,m) = m \cdot k$
- $\text{Dec}(k,m) = m \cdot k^{-1}$

So, if k is the key agreed in the **DH key exchange**, then

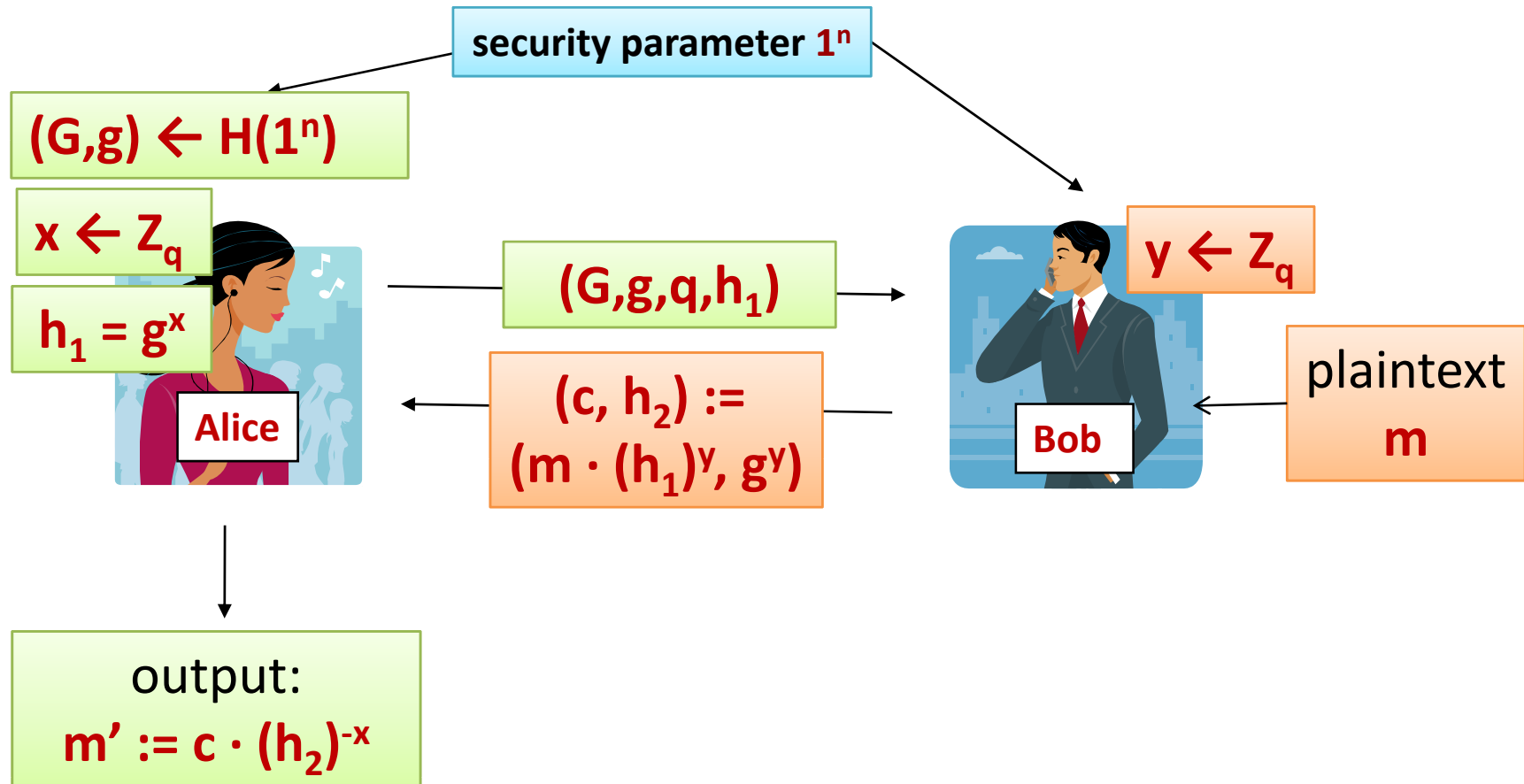
Alice can send a message $m \in \mathbf{G}$ to Bob “encrypting it with k ” by setting:

$$c := m \cdot k$$

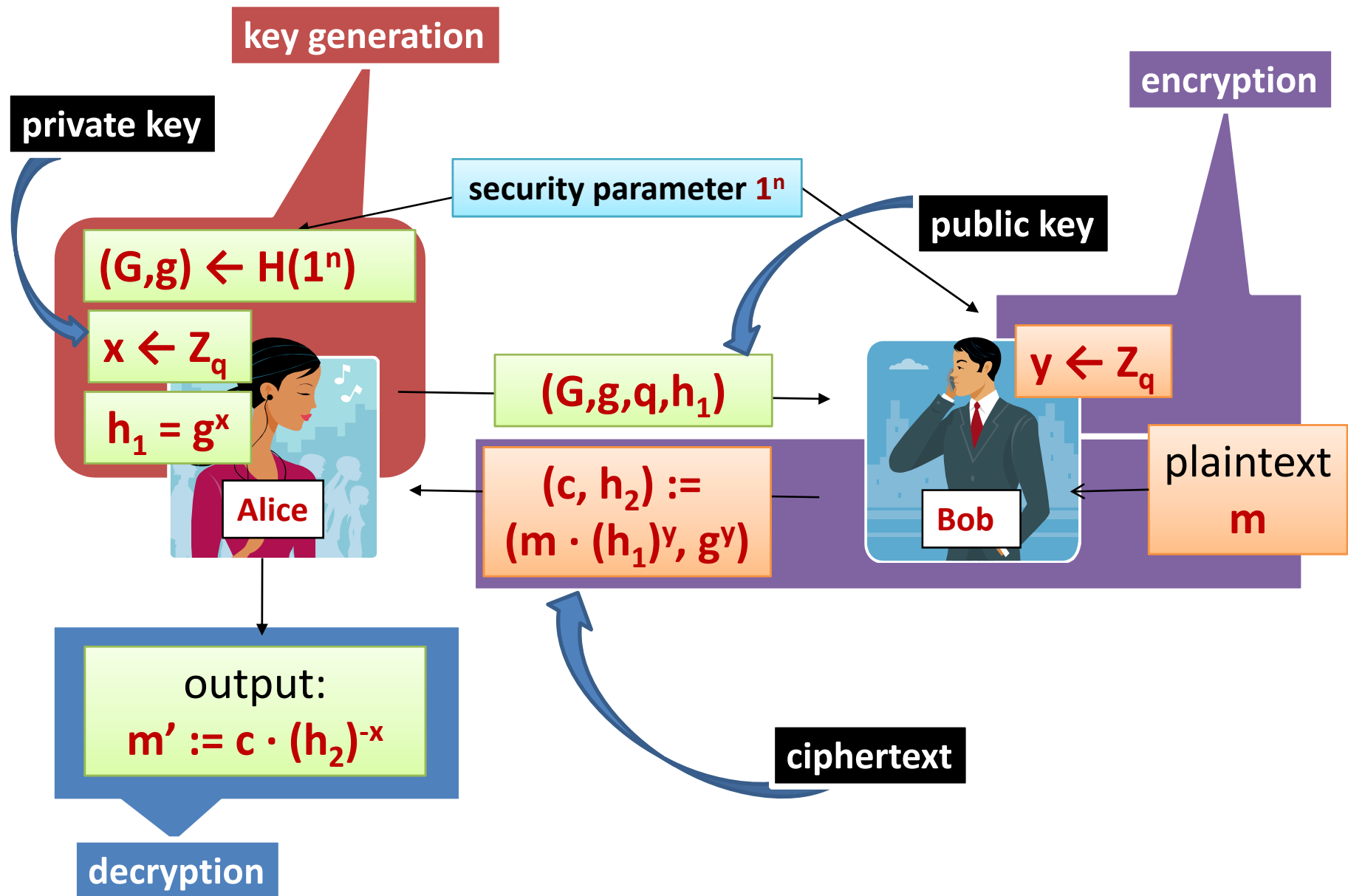
How does it look now?



The last two messages can be sent together



ElGamal encryption



El Gamal encryption

Let H be such that **DDH** is hard with respect to H .

Gen(1^n) first runs H to obtain (G, q) and q . Then, it chooses $x \leftarrow \mathbb{Z}_q$ and computes $h := g^x$. (note: it is **randomized by definition**)

The public key is (G, g, q, h) .

The private key is (G, g, q, x) .

$\text{Enc}((G, g, q, h), m) := (m \cdot h^y, g^y)$,
where $m \in G$ and y is a random element of G

$\text{Dec}((G, g, q, x), (c_1, c_2)) := c_1 \cdot c_2^{-x}$

Correctness

$$h = g^x$$

$$\text{Enc}((G,g,q,h), m) = (m \cdot h^y, g^y)$$

$$\begin{aligned} \text{Dec}((G,g,q,x), (c_1, c_2)) &= c_1 \cdot c_2^{-x} \\ &= m \cdot h^y \cdot (g^y)^{-x} \\ &= m \cdot (g^x)^y \cdot (g^y)^{-x} \\ &= m \cdot g^{xy} \cdot g^{-yx} \\ &= m \end{aligned}$$

El Gamal encryption – implementation issues

Which group to choose?

E.g.: **QR(p)**, where **p** is a strong prime, i.e.: **q = (p-1)/2** is also prime.

Plaintext space is a set of integers **{1,...,q}**.

How to map an integer **$i \in \{1, \dots, q\}$** to **QR(p)**?

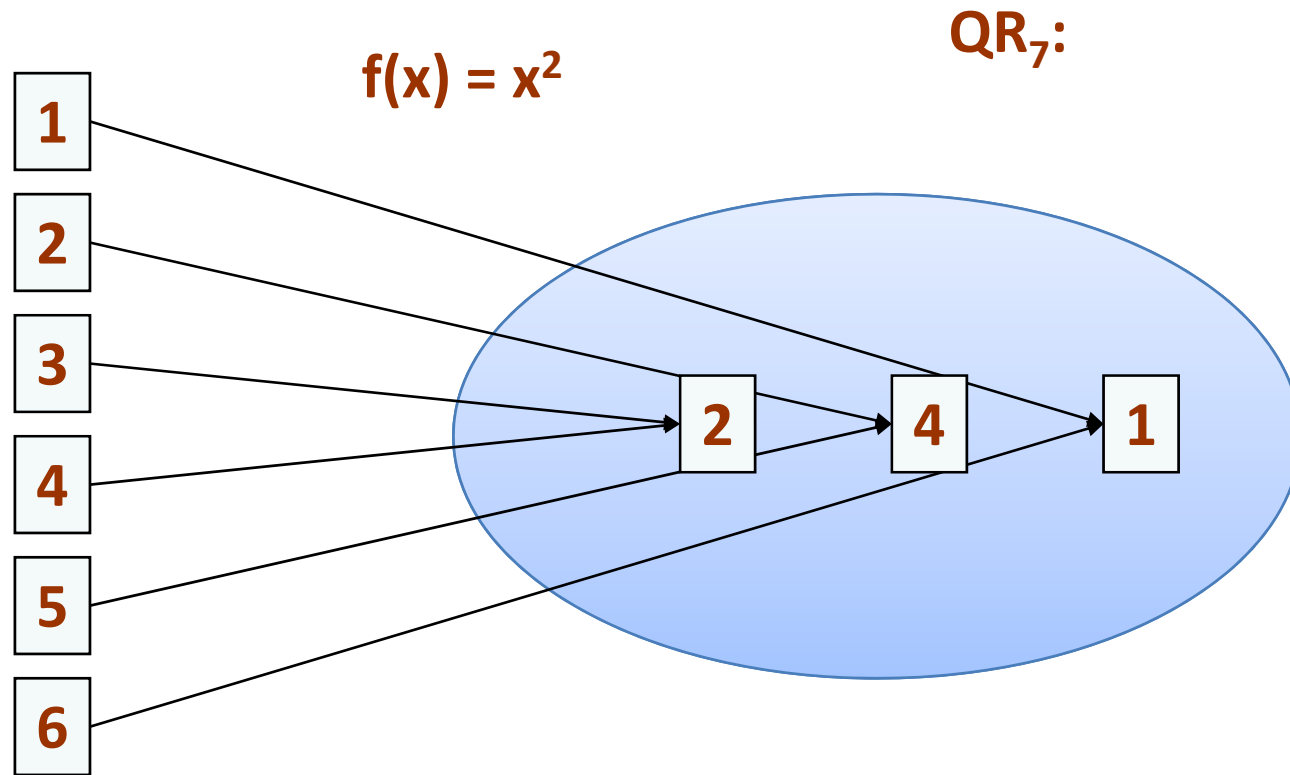
Just square:

$$f(i) = i^2 \bmod p.$$

Why is it **one-to-one**?

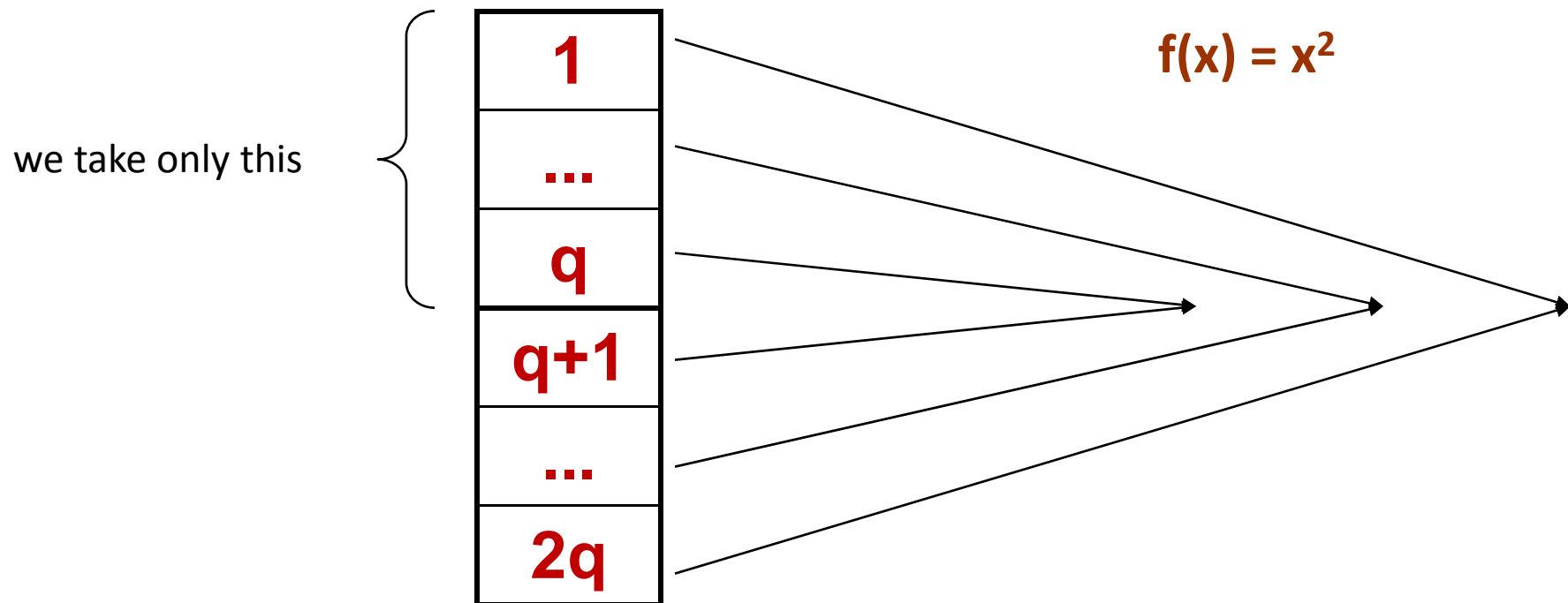
Remember this picture?

Z_7^* :



Observation

In \mathbf{Z}_p^* the function \mathbf{f} “glues” only the elements \mathbf{i} and $\mathbf{p-i}$



The mapping

So

$$f(i) = i^2 \bmod p$$

is **one-to-one** (on $\{1, \dots, q\}$).

Is it also efficiently invertible?

Yes (this was discussed on the previous lecture)

Plan

1. Problems with the handbook RSA encryption
2. Security definitions
3. How to encrypt with RSA?
4. Encryption based on discrete-log
 1. first step: Diffie-Hellman key exchange
 2. ElGamal encryption
5. **Public-key vs. private key encryption**



Public key vs. private key encryption

Private-key encryption has a following advantage:

it is much more efficient.

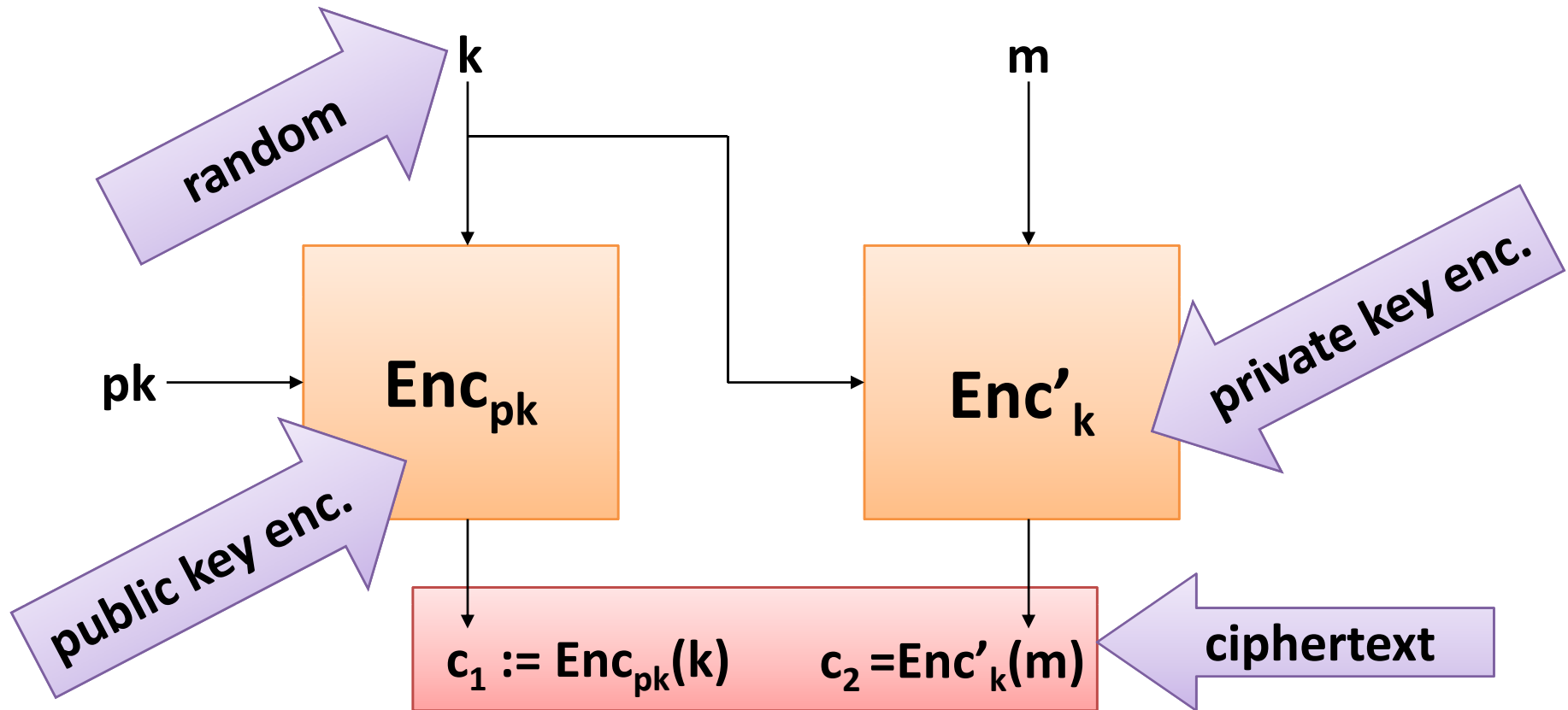
Practical solution:

combine both!

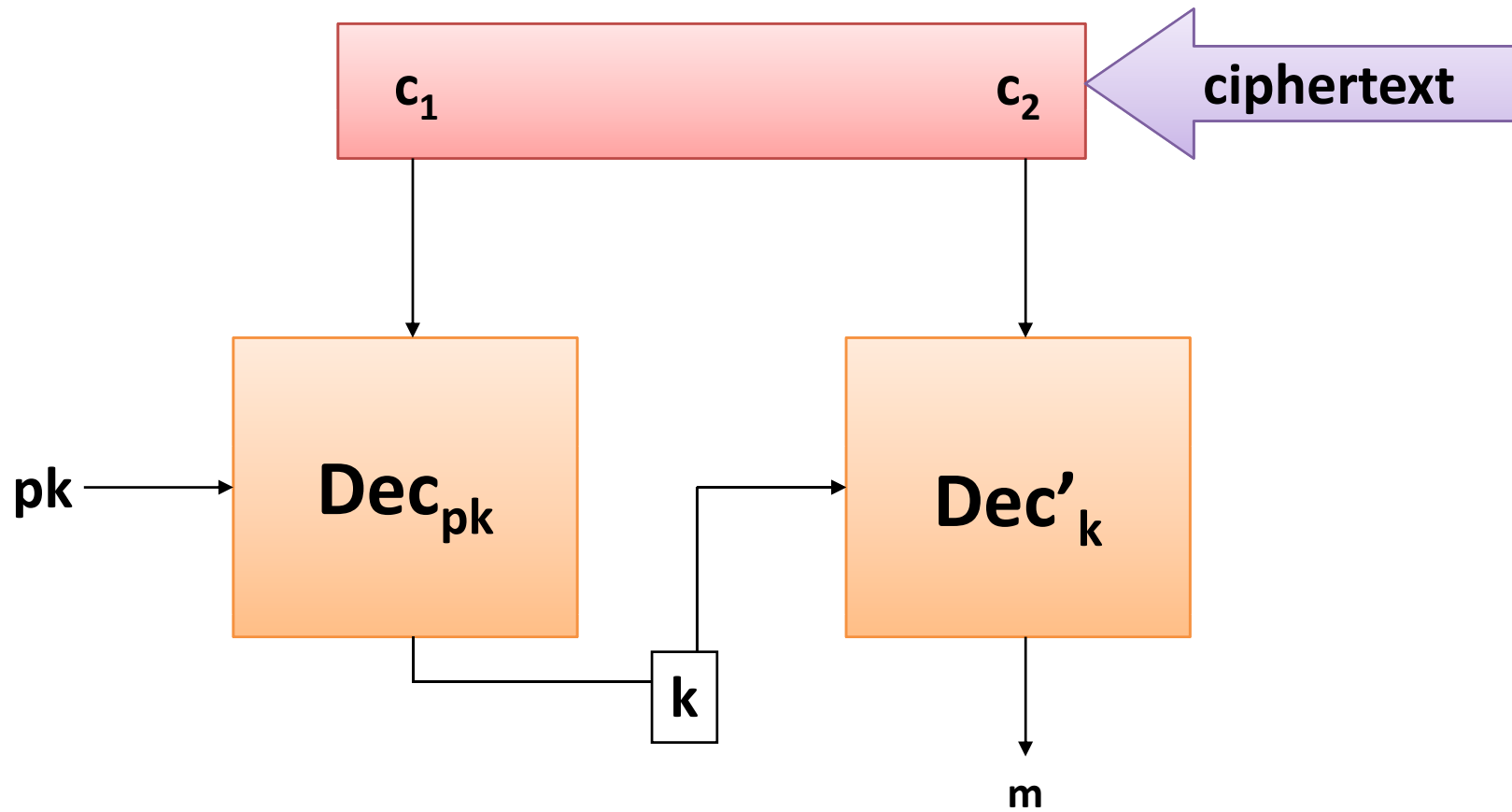
It is called: the **hybrid encryption**.

Hybrid encryption

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



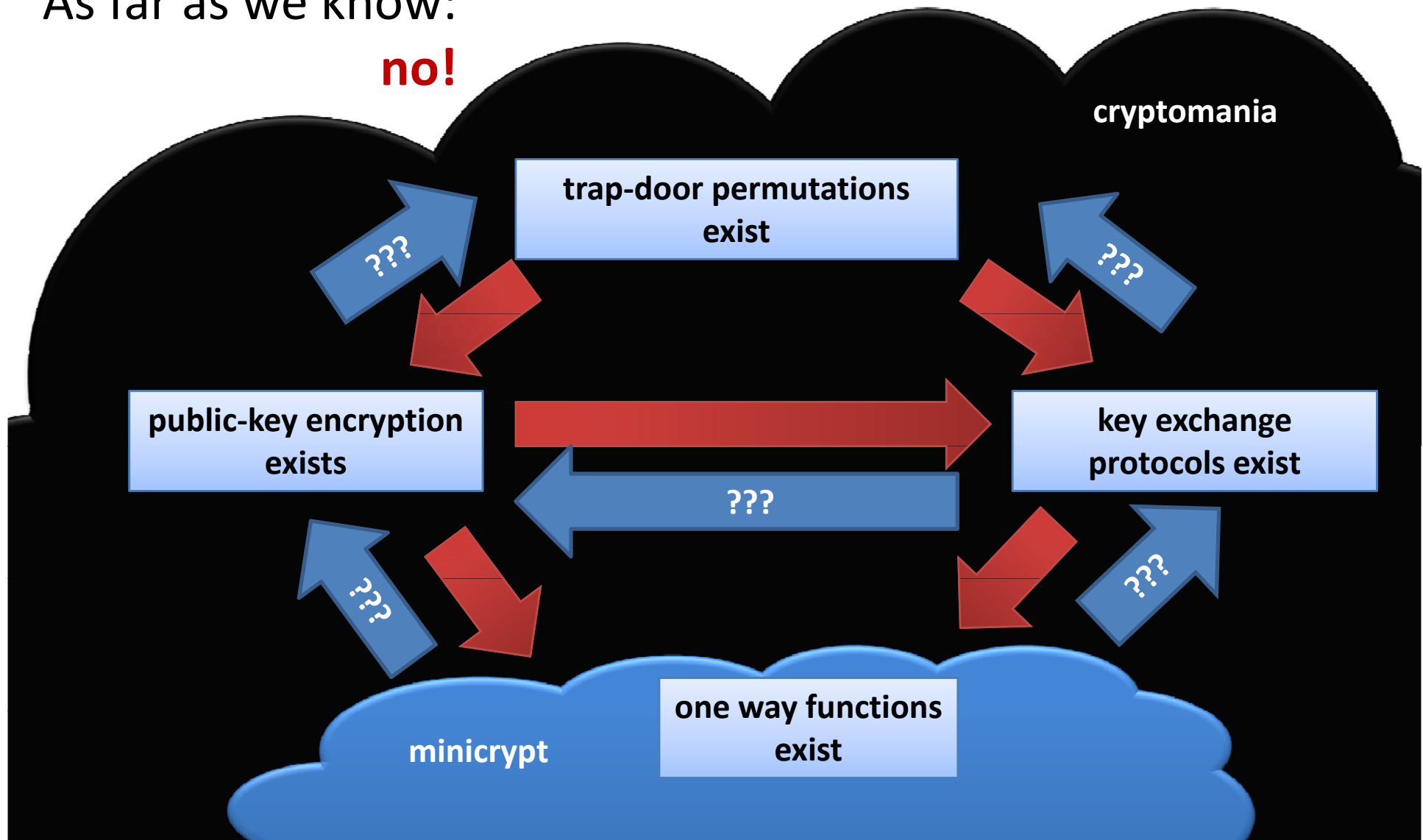
How to decrypt?



Is the public-key encryption in Minicrypt?

As far as we know:

no!



©2009 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.*