Lecture 9

Public-Key Encryption II

Stefan Dziembowski

www.crypto.edu.pl/Dziembowski

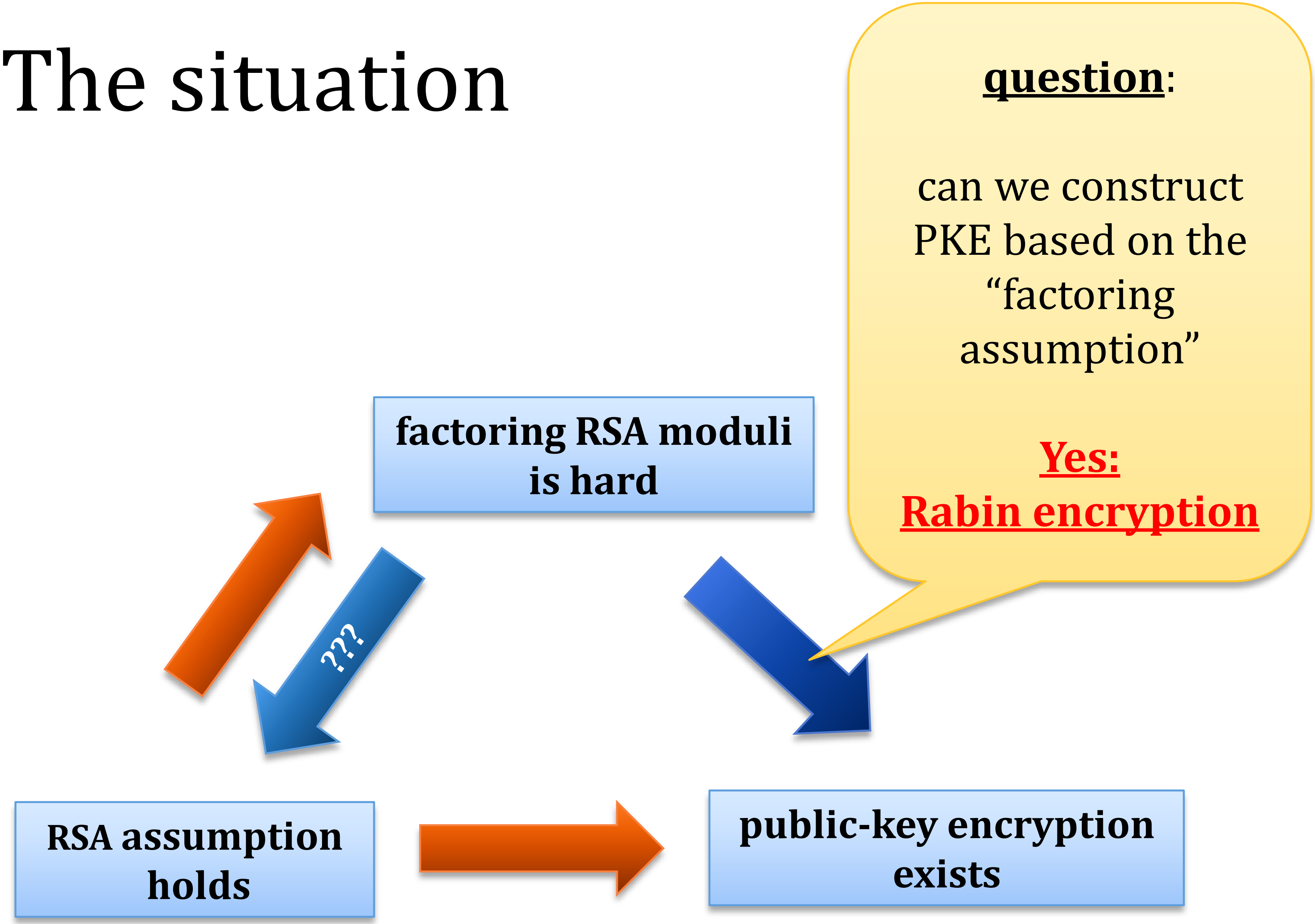
University of Warsaw

28.11.18 **version 1.0**

Plan

1. Rabin encryption
2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

The situation



Rabin encrypion

Michael O. Rabin (Wrocław 1931 –)

One of the founding fathers of computer science.

- introduced **non-determinism**
- decidability of the **monadic second order logic**
- efficient **primality testing**
- **oblivious transfer**,
- **....**

received **Turing Award** in **1976**

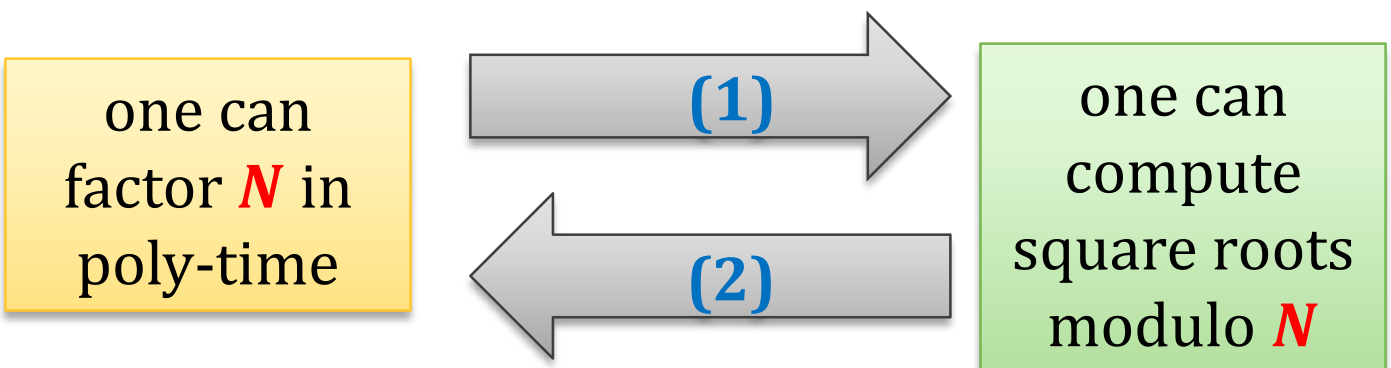
- introduced by **Michael O. Rabin** in 1979
- based on squaring in Z_N^*
- security **equivalent** to factoring

On previous lectures we proven the following

Fact

Let N be a random **RSA** modulus.

The problem of computing square roots (modulo N) of random elements in $\mathbf{QR}_N$ is poly-time equivalent to the problem of factoring N.



In other words

"squaring in $\boldsymbol{Z}_{\boldsymbol{N}}^{*}$" is a one-way function (assuming the **factoring RSA moduli** is hard).

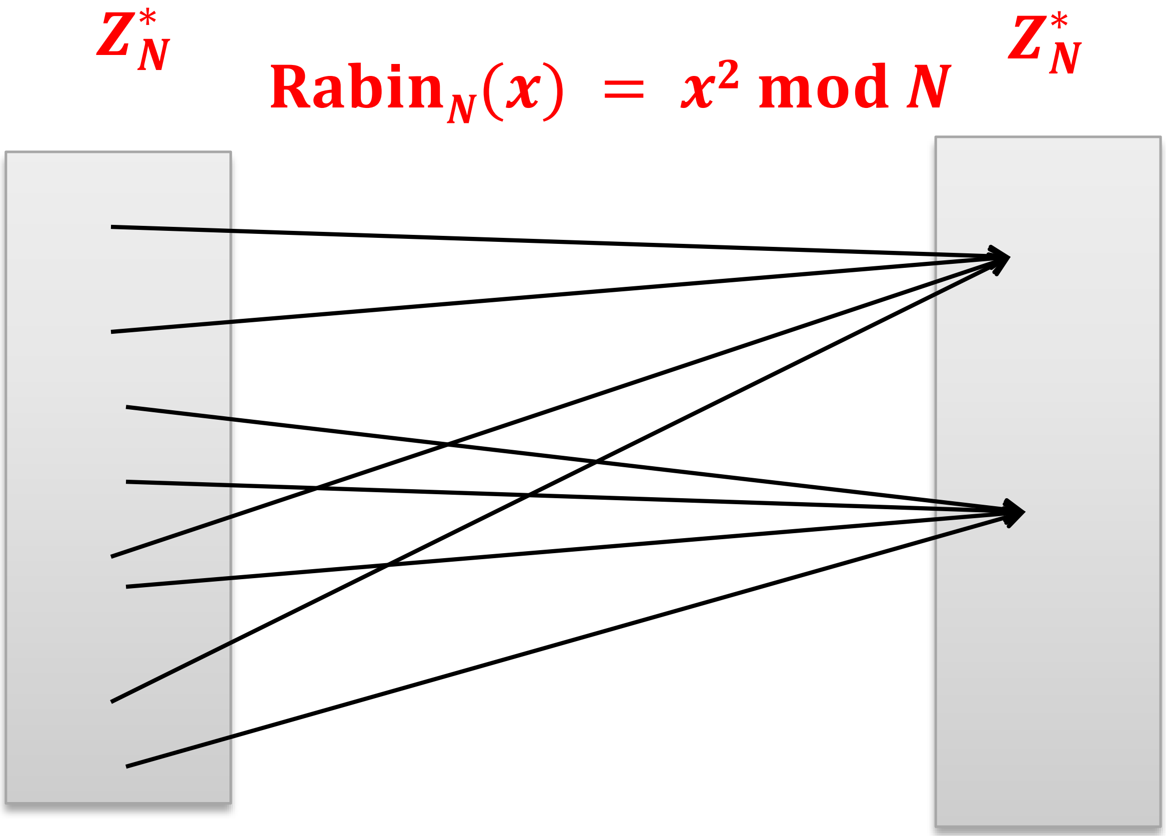
Define:

$$\textbf{Rabin}\colon \boldsymbol{Z}_{\boldsymbol{N}}^{*} \to \boldsymbol{Z}_{\boldsymbol{N}}^{*}$$

as

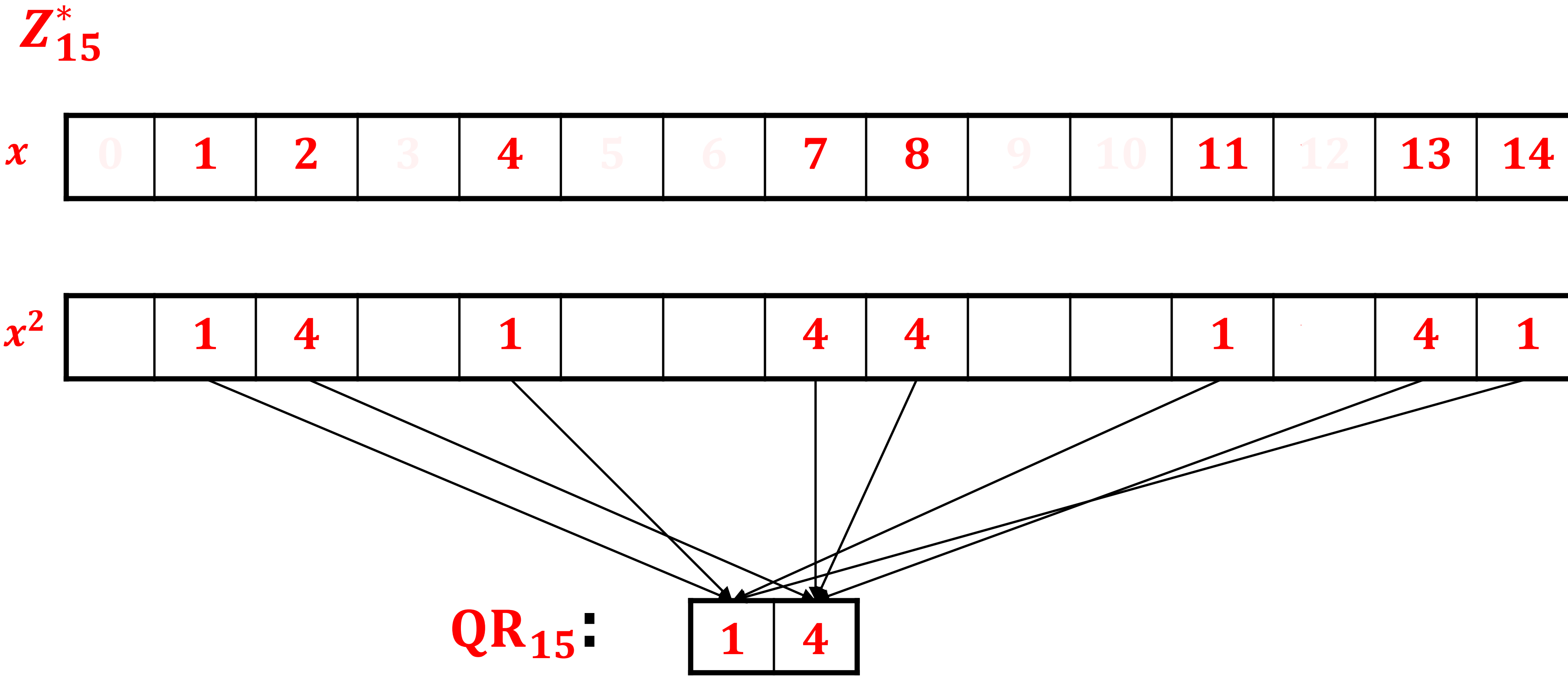
$$\textbf{Rabin}(\boldsymbol{x}) \coloneqq \boldsymbol{x}^{\mathbf{2}} \textbf{ mod } \boldsymbol{N}$$

A fact about squaring modulo $N = pq$?



This function "glues" **4** elements together.

Example for $N = 15$



How to base encryption on this?

Idea:

- **public key**: $N = pq$
- **private key**: (p, q)
- **encryption**: $\mathbf{Enc}_N(x) = x^2 \bmod N$
- **decryption**: $\mathbf{Dec}_{(p,q)}(y) = \sqrt{y} \bmod N$

can be computed efficiently if one knows p and q (see **Lecture 7**)

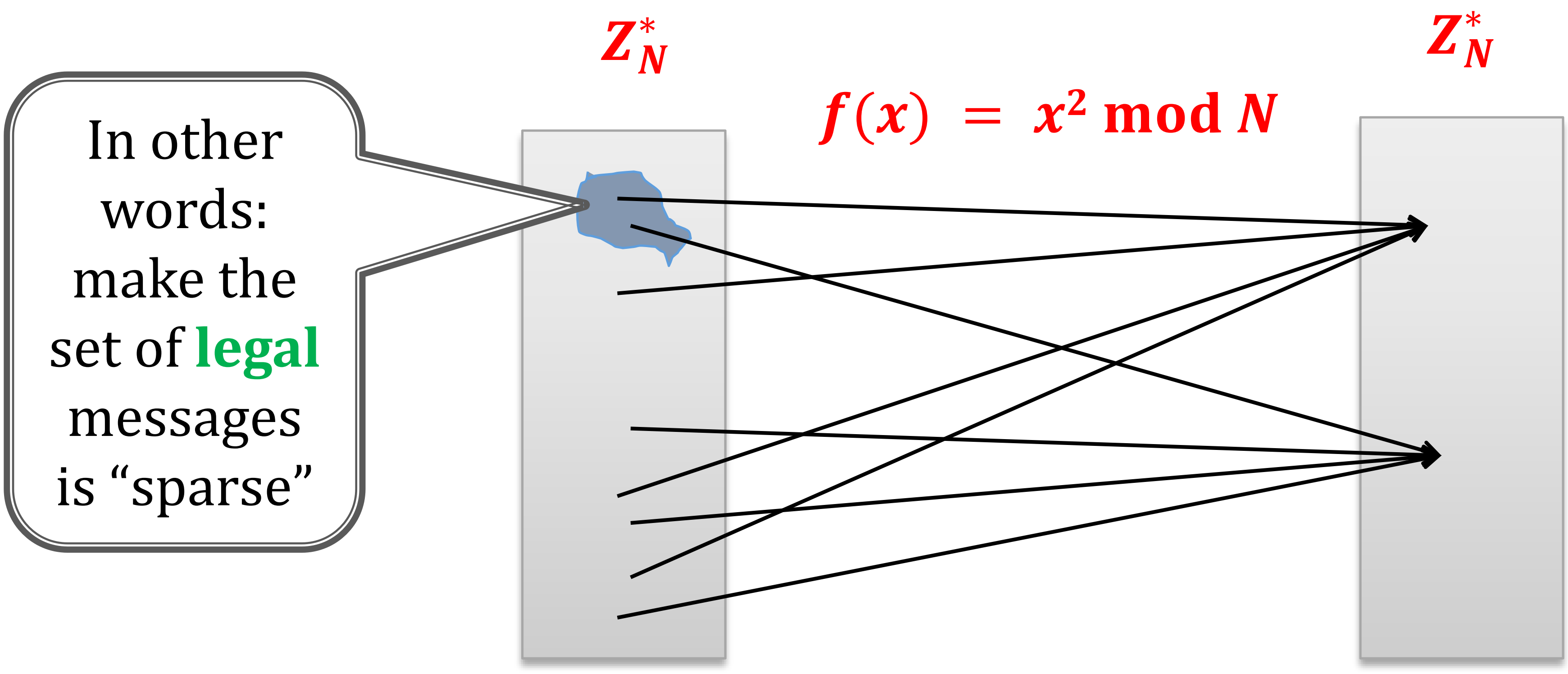
Problem: there are **4** square roots.

Solution: "make the inversion unique".

How to do it?

An ad-hoc method: add an encoding (like in the "real **RSA** encryption").

In such a way that only **1** out of the **4** square roots "make sense".



Another approach

Such an N is called a "**Blum integer**"

Fact

Suppose $N = pq$ where

$$p = q = 3 \ (\mathbf{mod}\ 4)$$

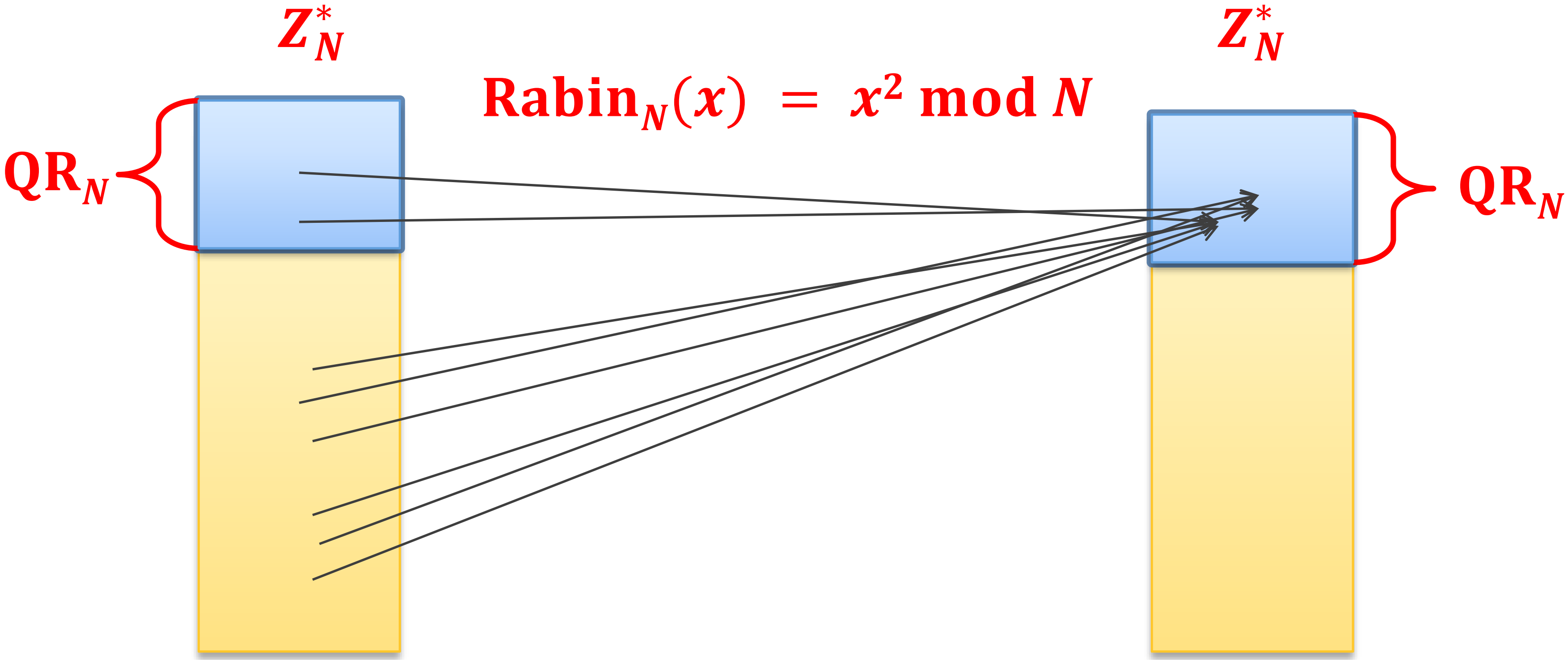
Then the function

$$\mathbf{Rabin}_N(x) \ = \ x^2 \ \mathbf{mod}\ N$$

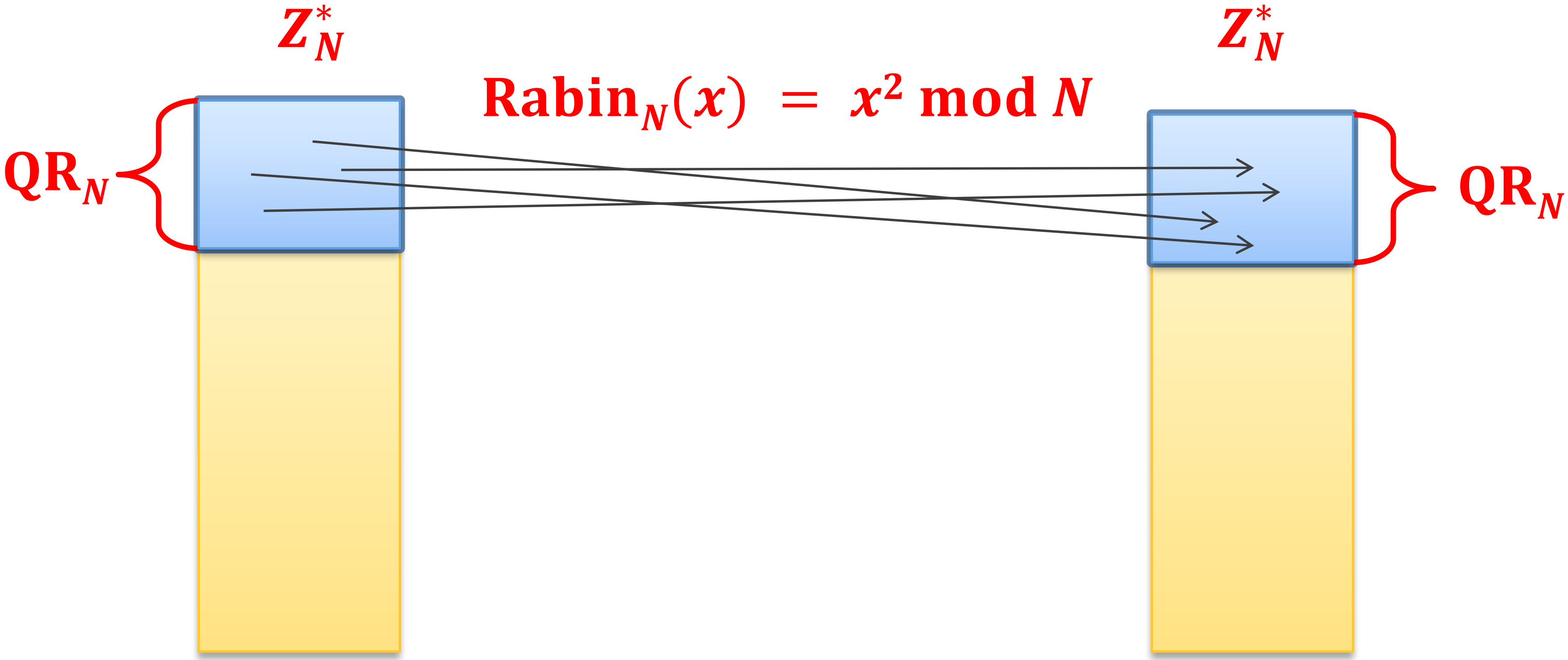
is a permutation when restricted to $\mathbf{QR}_N$

$$\mathbf{Rabin}_N : \ \mathbf{QR}_N \rightarrow \mathbf{QR}_N$$

How does it look?



Rabin restricted to $\mathbf{QR}_N$ is a permutation



Proof that $\mathbf{Rabin}_N(x) = x^2 \bmod N$ restricted to $\mathbf{QR}_N$ is a permutation

($N = pq$, where $p = q = 3 \bmod 4$)

We prove that **Rabin** is injective, i.e. for every $x, y \in \mathbf{QR}_\mathrm{N}$ we have that

$$x^2 = y^2 \implies x = y$$

Observation: by **CRT** it is enough to show that

- $x^2 = y^2 \implies x = y \bmod p$ and
- $x^2 = y^2 \implies x = y \bmod q$.

By symmetry it's also enough to show it just for p.

Proof

Suppose we have $x, y \in \mathbf{QR_N}$ such that
$x^2 = y^2 \bmod N$

$x^2 = y^2 \bmod p$

$g^{4i} = g^{4j} \bmod p$

$g^{4(i-j)} = 1 \bmod p$

$p - 1 \mid 4(i-j)$

$4k + 2 \mid 4(i-j)$

$2k + 1 \mid 2(i-j)$

$2k + 1 \mid i - j$

$i = j$

$x = y \bmod p$

QED

Let $p = 4k + 3$, where $k \in \mathbf{N}$

Let $i, j \in \mathbf{N}$ be such that

- $x = g^{2i} \bmod p$ and
- $y = g^{2j} \bmod p$

where g is a generator of Z_p^* and

$$0 \le j \le i < \frac{p-1}{2} = \frac{4k+2}{2} = 2k+1$$

How to encrypt a one-bit message b?

Fact: the least significant bit is a **hard-core bit for the Rabin permutation**.

a Blum integer

N – public key

(p, q) – private key

$$\mathbf{Rabin}_N(x) = x^2 \bmod N$$
$$\mathbf{Rabin}_N : \mathbf{QR}_N \to \mathbf{QR}_N$$

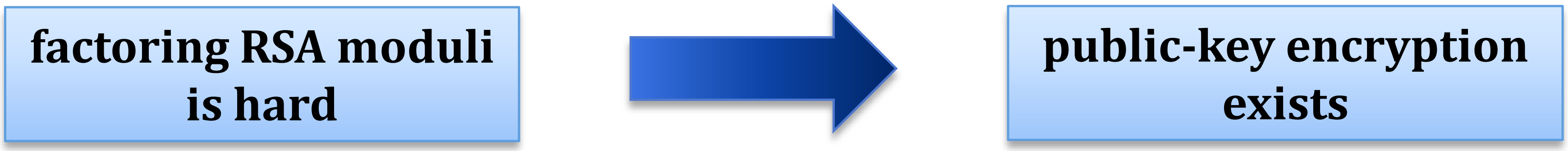
$$\mathbf{Enc}_N(b) = (\mathbf{LSB}(x) \oplus b, \mathbf{Rabin}_N(x)),$$

where $x \in \mathbf{QR}_N$ is random.

this can be computed if one knows p and q

$$\mathbf{Dec}_{p,q}(b', y) = \mathbf{LSB}\left(\mathbf{Rabin}_N^{-1}(y)\right) \oplus b'$$

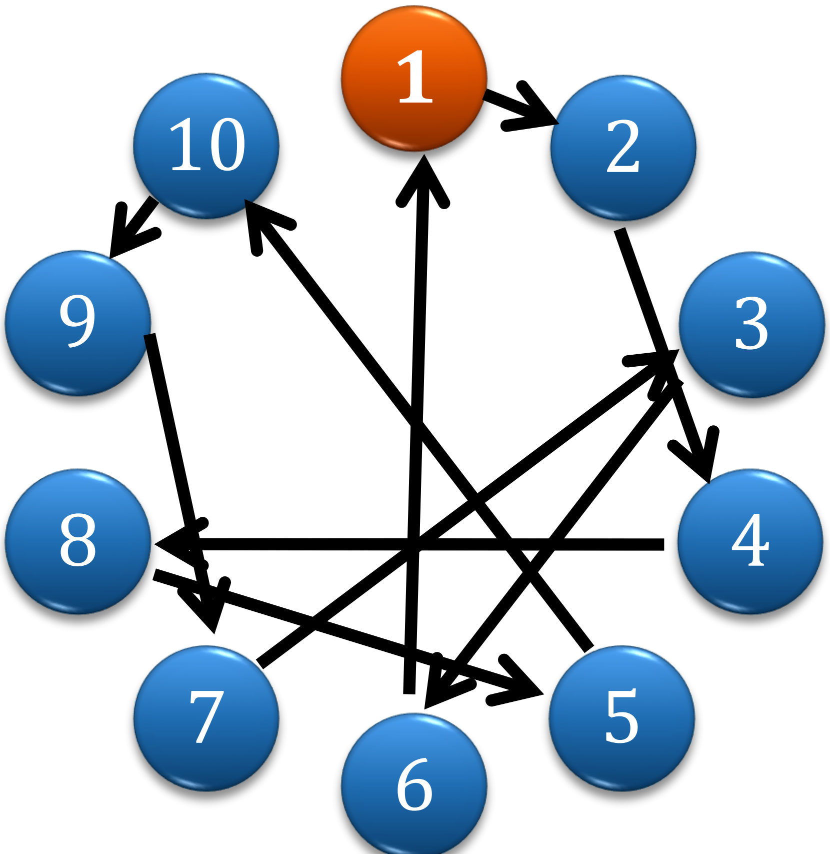
Moral



Plan

1. Rabin encryption
2. ElGamal encryption
   1. a tool: Diffie-Hellman key exchange
   2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

Remember the exponentiation modulo a prime?



x	$2^x \bmod 11$
0	1
1	2
2	4
3	8
4	5
5	10
6	9
7	7
8	3
9	6

2 is a generator of Z_{11}^*

Discrete log

x	g^x
0	1
1	2
2	4
3	8
4	5
5	10
6	9
7	7
8	3
9	6

Function

$$f(x) = g^x \bmod p$$

easy to compute

believed to be **hard to compute** for large p

f^{-1} is also denoted $\log_g$ and called the **discrete logarithm**

Discrete log is hard in many other groups!

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.

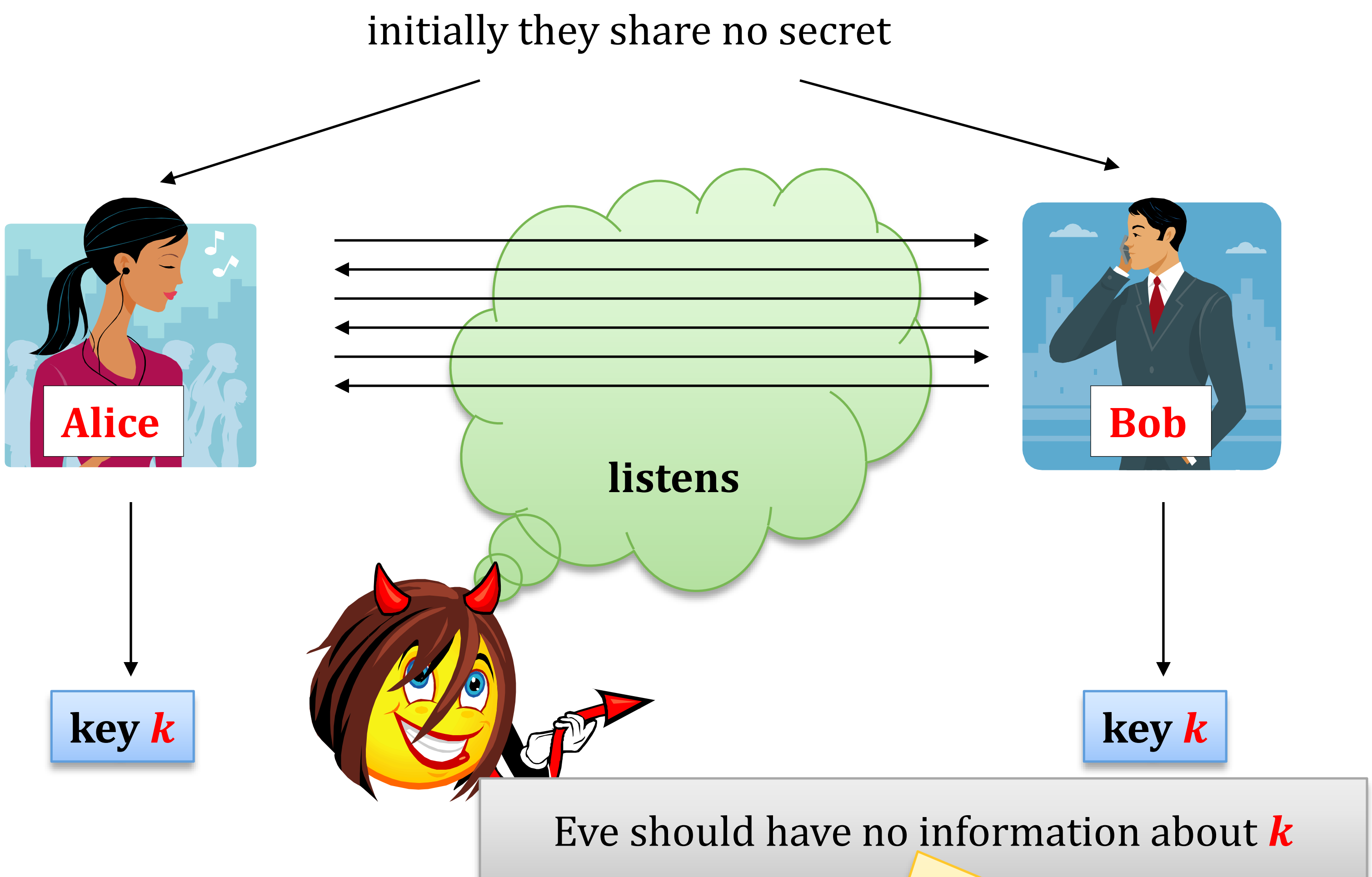
Diffie and Hellman constructed something weaker than PKE: a **key exchange protocol** (also called key **agreement** protocol).

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

Plan

1. Rabin encryption
2. ElGamal encryption
    1. a tool: Diffie-Hellman key exchange
    2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

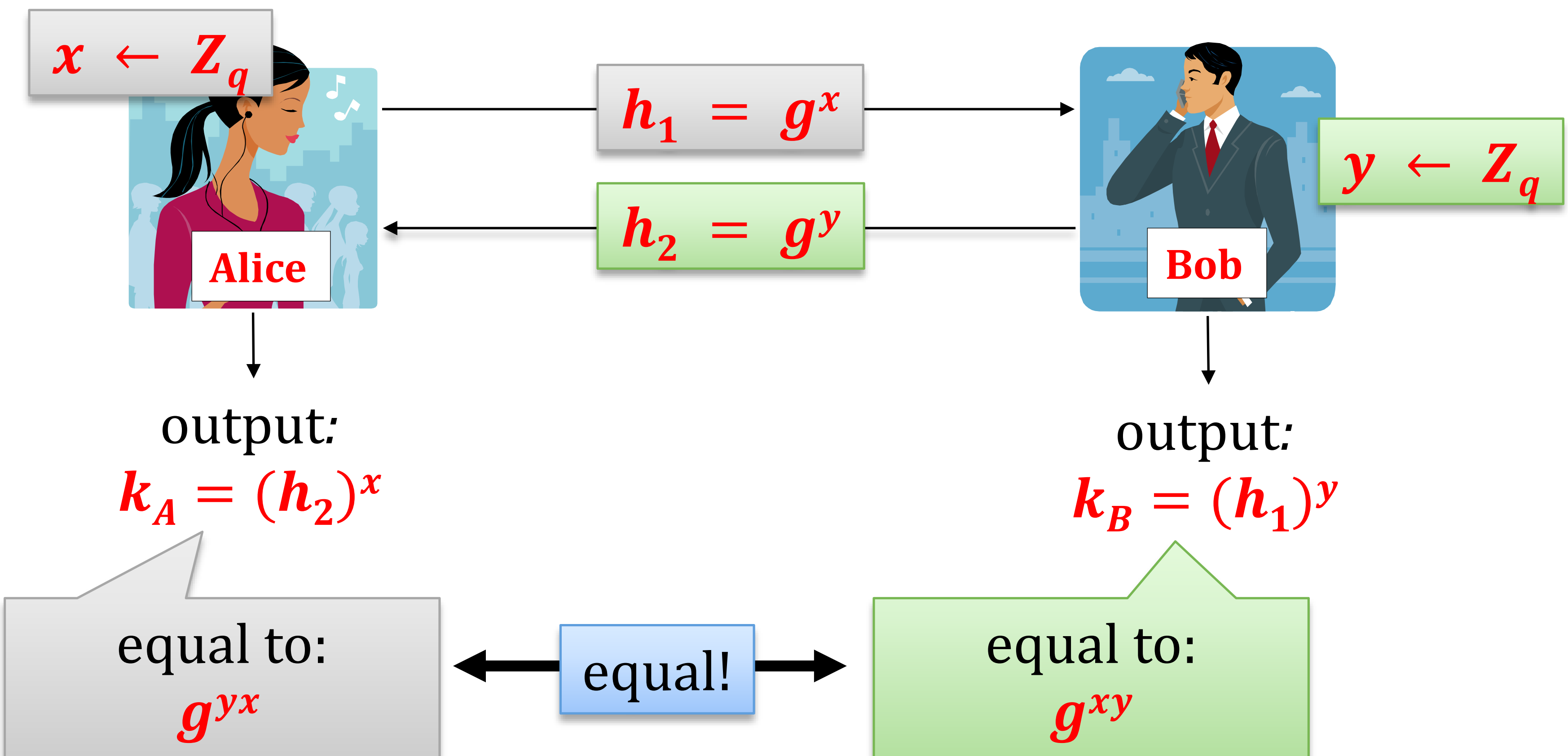
Key exchange



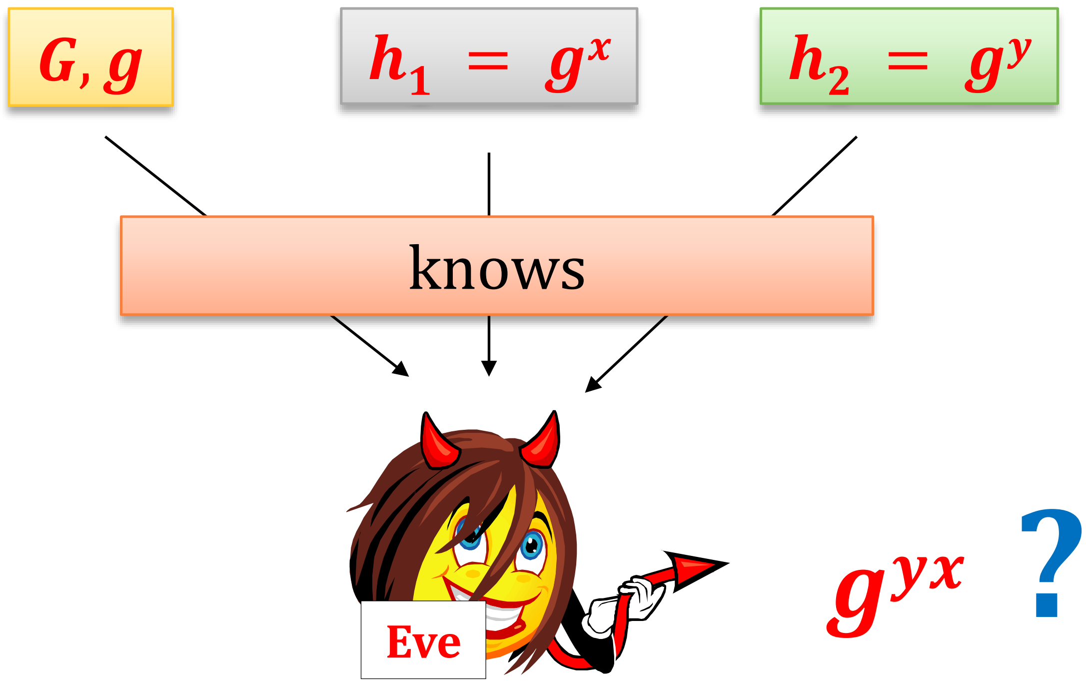
We will formalize it later.
Let's first show the protocol.

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 key 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...

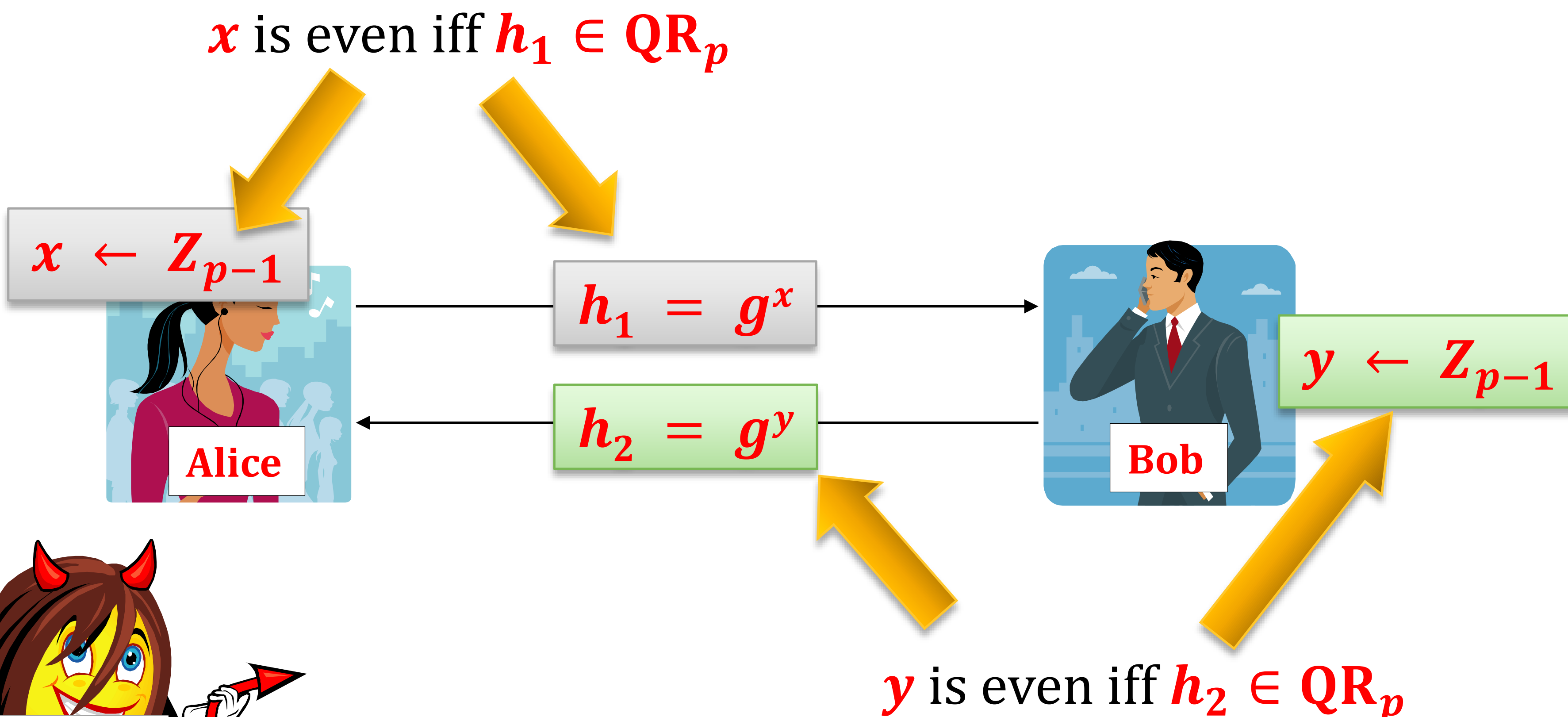
it may also not be secure

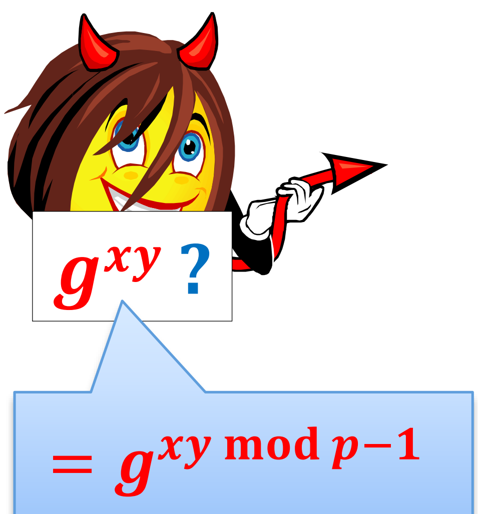
Example for $G = Z_p^*$

We use the facts that:

- **quadratic residues** in Z_p^* are **even powers of the generator**, and
- testing **membership in $\mathbf{QR}_p$** is computationally **easy** (even for large p).

Suppose $G = Z_p^*$





Therefore:

$g^{xy} \in \mathrm{QR}_p$ iff ($h_1 \in \mathrm{QR}_p$ or $h_2 \in \mathrm{QR}_p$)

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

Solution (see previous lectures)

Instead of working in $\boldsymbol{Z_p^*}$ work in its **subgroup**: $\mathbf{QR}_{\boldsymbol{p}}$

How to find a generator of $\mathbf{QR}_{\boldsymbol{p}}$?

A practical method: Choose $\boldsymbol{p}$ that is a **strong prime**, which means that:

$\boldsymbol{p = 2 \cdot q + 1}$, with $\boldsymbol{q}$ prime.

Hence: $\mathbf{QR}_{\boldsymbol{p}}$ has a **prime order** ($\boldsymbol{q}$).

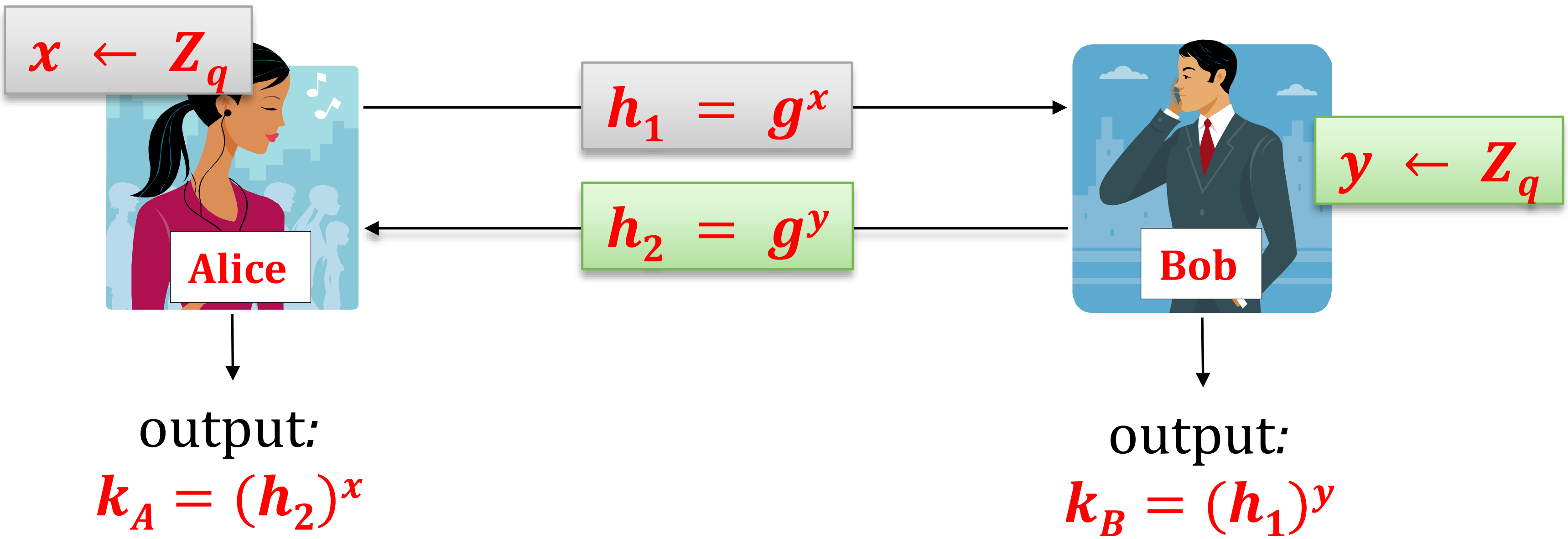
Every element (except of **1**) of a group of a prime order is its **generator**!

Therefore: every element of $\mathbf{QR}_{\boldsymbol{p}}$ is a generator.

The DH Key exchange over QR group

Take a prime $p = 2 \cdot q + 1$, with q prime.

Take any $h \in Z_p$ such that $h \neq \pm 1$ and let $g = h^2 \bmod p$.

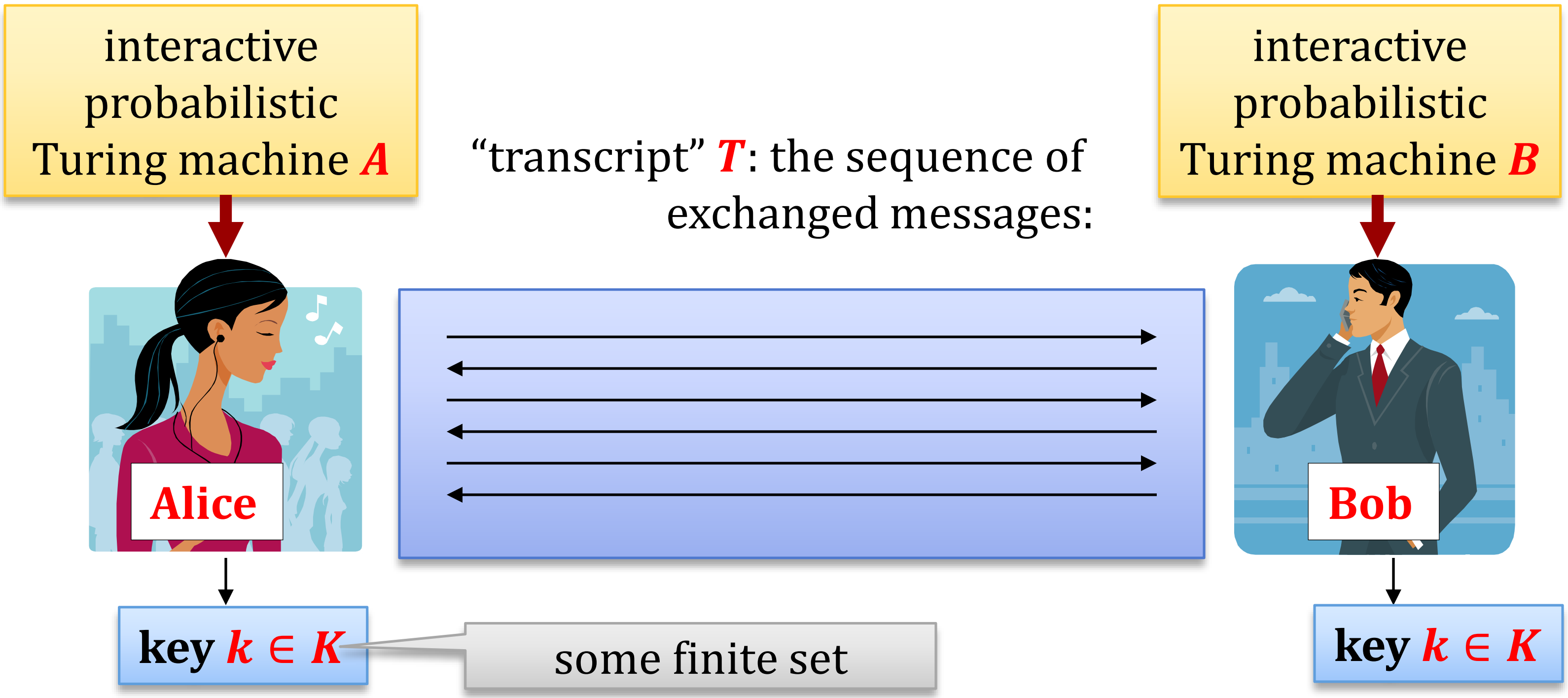


But is the partial information leakage really a problem?

We need to

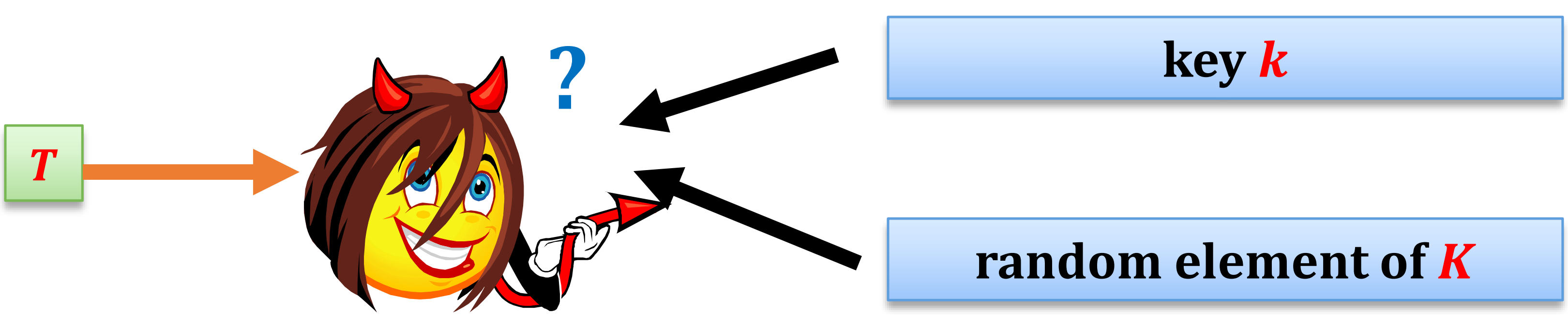
1. **formalize** what we mean by secure key exchange,

2. identify the **assumptions needed** to prove the security.

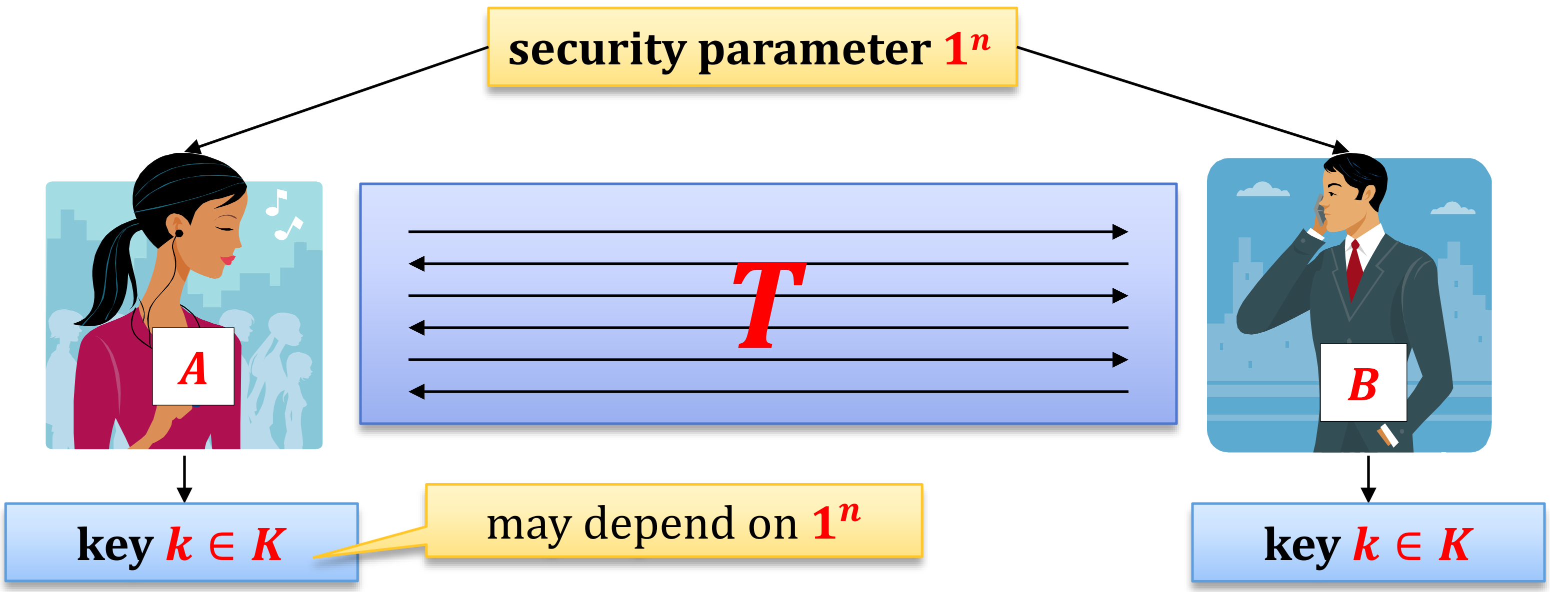


Informal definition:

(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

$$\forall \; |P(M(1^n, T, k) = 1) - P(M(1^n, T, r) = 1)| \leq \mathbf{negl}(n)$$

poly-time M

$r \leftarrow K$

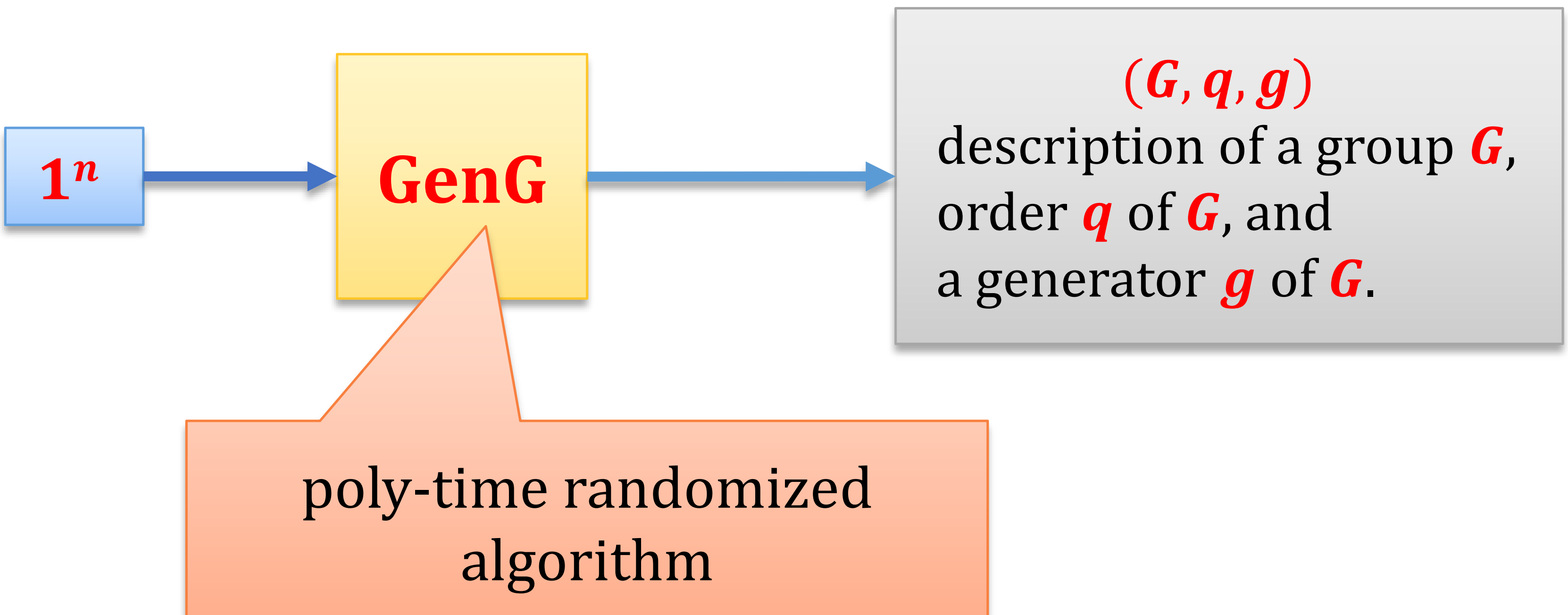
How to make G dependent on 1^n?

In **practice** often a fixed group is used.

In **theory** we need to have a **new group** G for every value of 1^n.

So, we need to define an algorithm that generates G and its generator g.

Group generating algorithm **GenG**



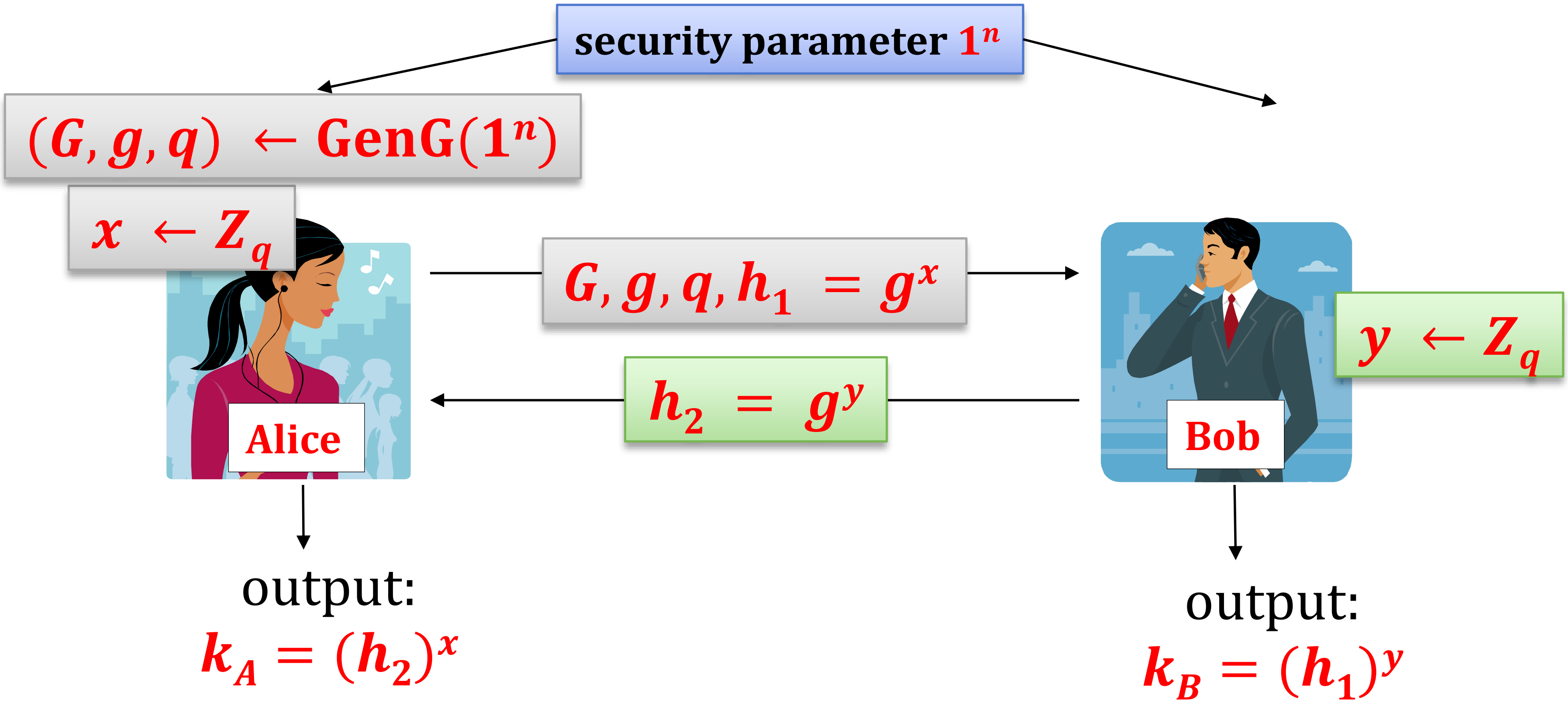
Example of GenG

$$(Z_p, q, g)$$

where:

- p, q – random primes such that $|p| = n$ and $p = 2 \cdot q + 1$
- and $g = h^2 \bmod p$, where $h \leftarrow Z_p \setminus \{-1, 1\}$

How does the protocol look now?



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

Formally

Decisional Diffie-Hellman (DDH) problem is hard relative to **GenG** if for every poly-time algorithm $\boldsymbol{A}$ we have that

$$|P(A(G,q,g,g^x,g^y,g^z)=1) - P(A(G,q,g,g^x,g^y,g^{xy})=1)| \leq \mathbf{negl}(n)$$

where

$$(G,q,g) \leftarrow \mathbf{GenG}(1^n)$$

and

$$x,y,z \leftarrow Z_q$$

Examples

DDH is believed to be hard relative to $\mathbf{GenG_{QR}}$

Other examples: elliptic curves

How does DDH compare to the discrete log assumption

DDH is hard relative to **GenG**

implies

discrete log is hard relative to ***H***

The opposite implication is unknown in most of the cases

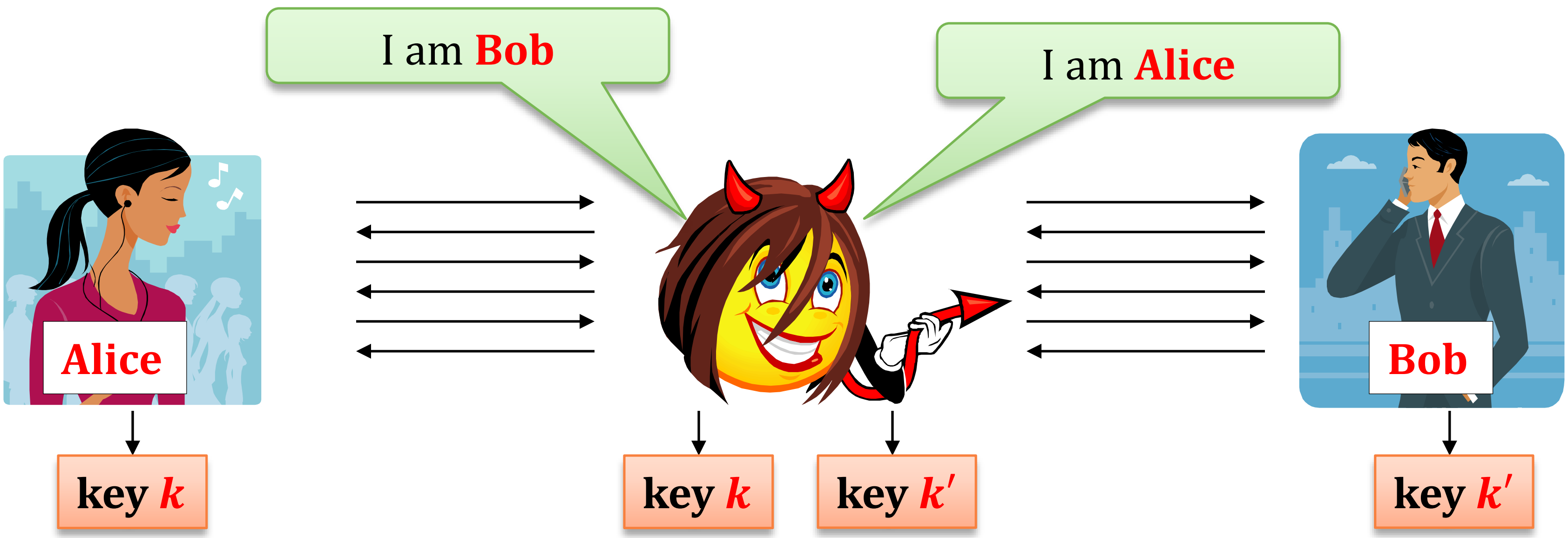
A problem

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. Rabin encryption
2. ElGamal encryption
    1. tool: Diffie-Hellman key exchange
    2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

ElGamal encryption

ElGamal is another popular public-key encryption scheme.

Introduced in:

[Taher ElGamal "A Public key Cryptosystem and A Signature Scheme based on discrete Logarithms". ***IEEE Transactions on Information Theory*****. 1985]**

Taher ElGamal
(1955–)

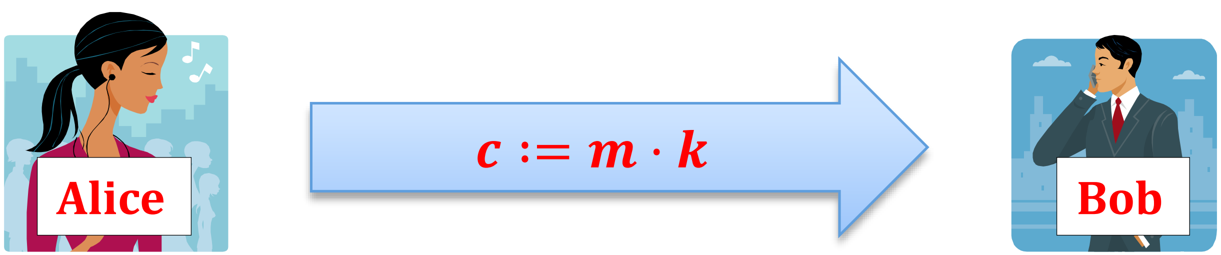
It is based on the **Diffie-Hellman** key-exchange.

First observation

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

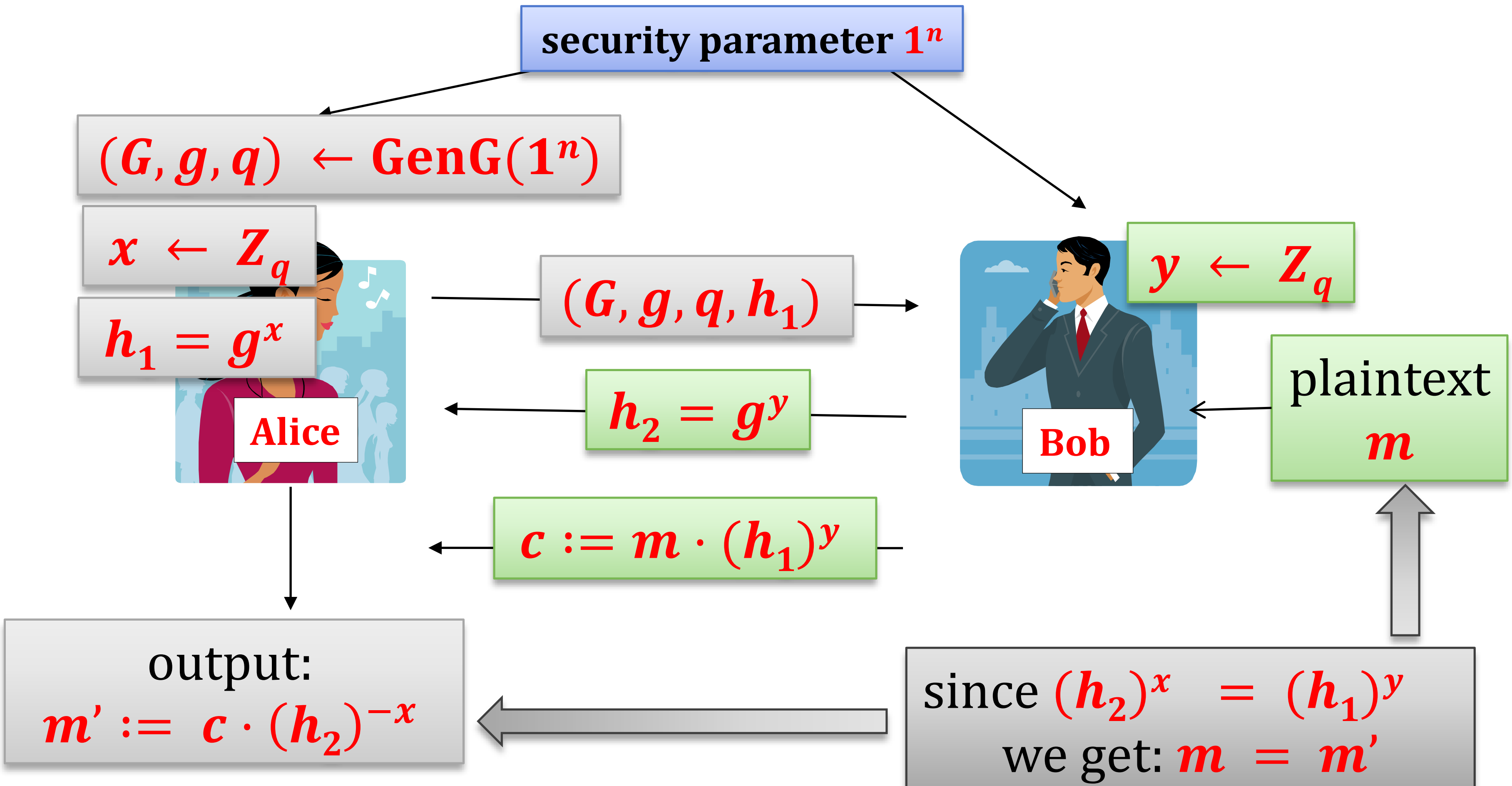
$$\mathcal{K} = \mathcal{M} = \mathcal{C} = G$$
$$\mathbf{Enc}(k, m) = m \cdot k$$
$$\mathbf{Dec}(k, m) = m \cdot k^{-1}$$

So, if k is the key agreed in the **DH key exchange**, then **Alice** can send a message $M \in G$ to **Bob** "encrypting it with k" by setting: $c := m \cdot k$

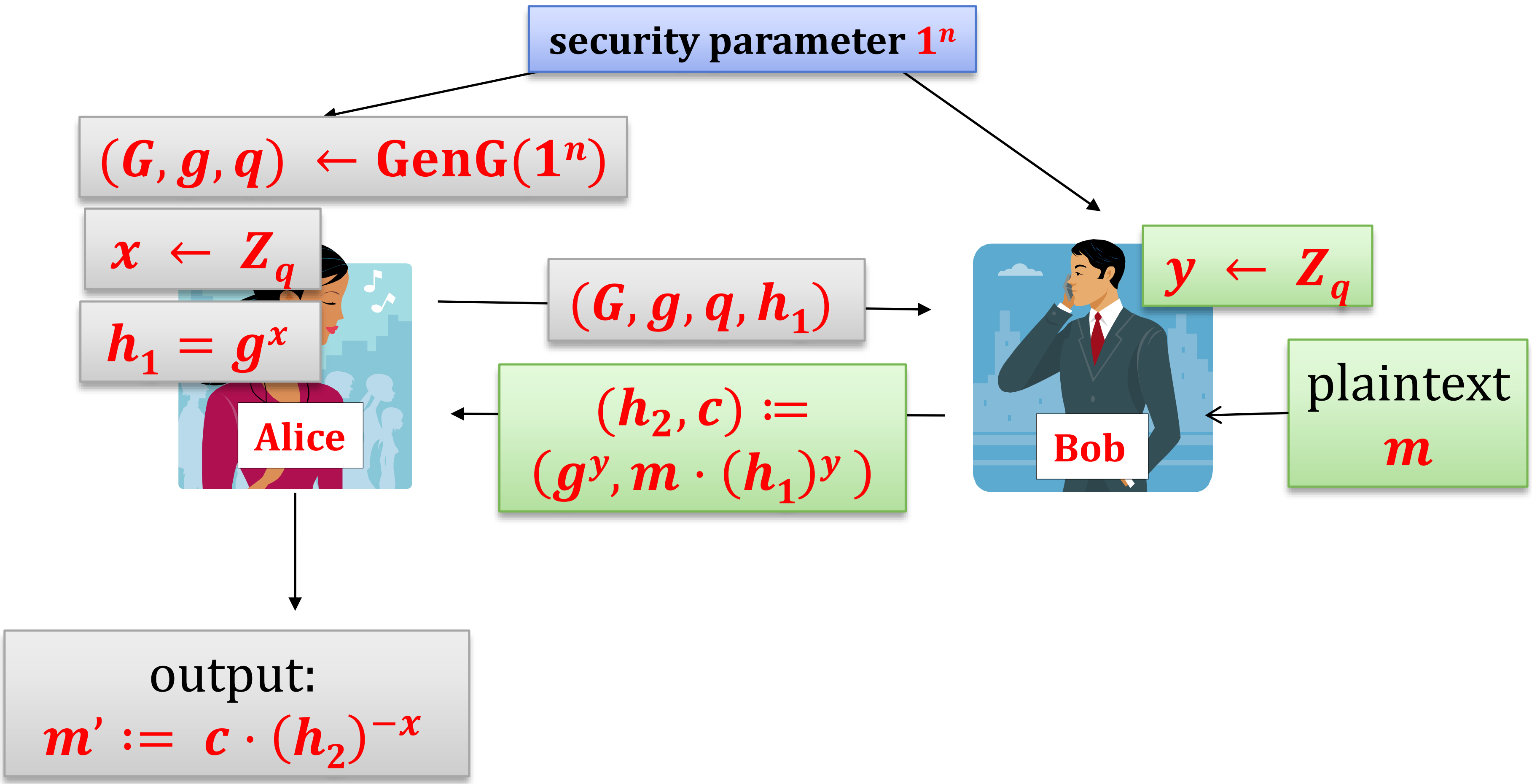


Note: this is essentially the **KEM/DEM** method from **Lecture 8**.

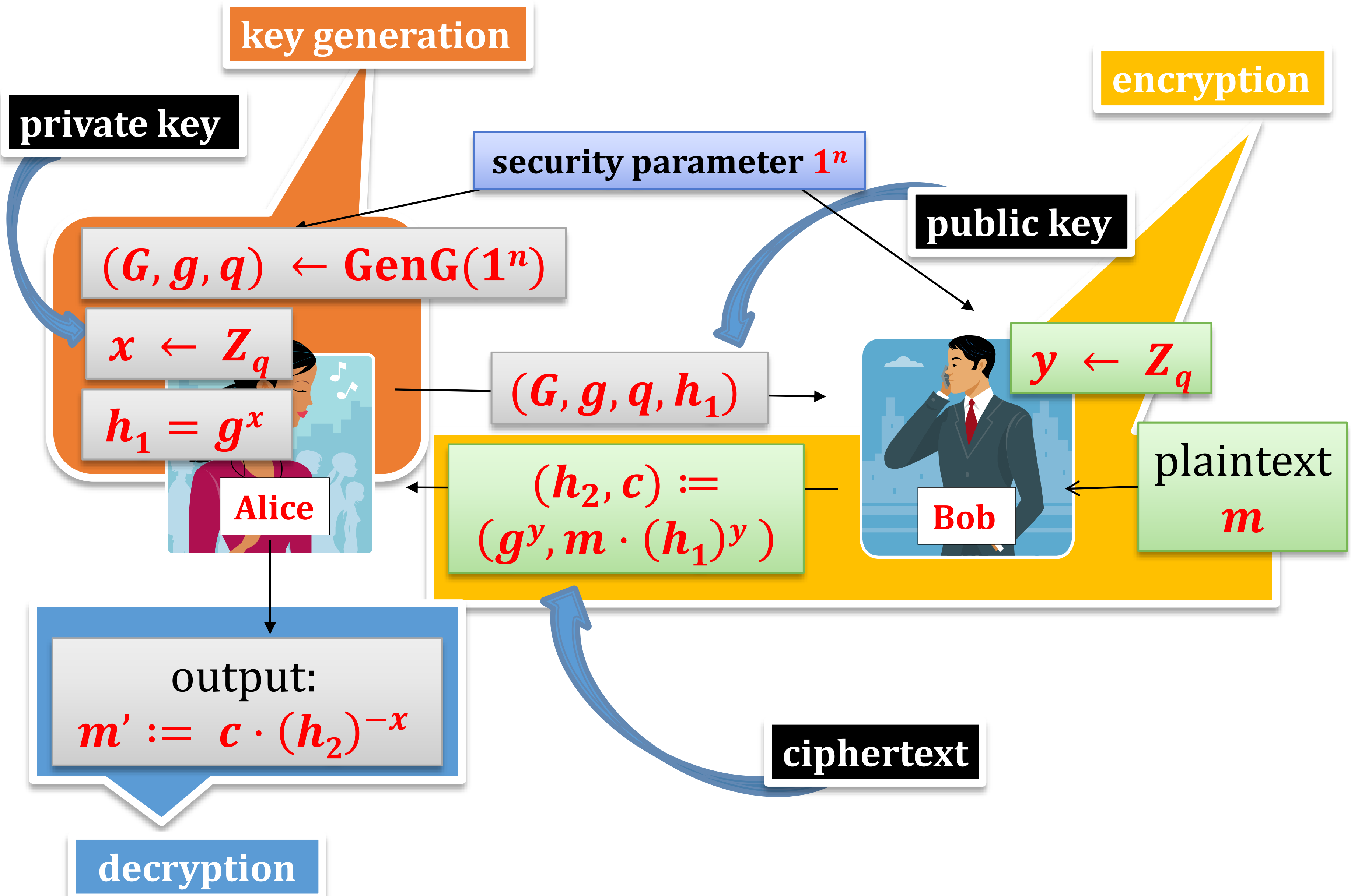
How does it look now?



The last two messages can be sent together



ElGamal encryption



ElGamal encryption

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

Gen(1^n) first runs **GenG** to obtain G, g and q. Then, it chooses $x \leftarrow Z_q$ and computes $h_1 := g^x$.

The public key is (G, g, q, h_1).
The private key is (G, g, q, x).

Enc$((G, g, q, h_1), m) := (m \cdot h_1^y, g^y)$,

where $m \in G$ and y is a random element of G

(note: it is **randomized by definition**)

Dec$\big((G, g, q, x), (c_1, h_2)\big) := c_1 \cdot h_2^{-x}$

Correctness

$$h = g^x$$

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

$$\begin{aligned}
\mathbf{Dec}\big((G, g, q, x), (c_1, h_2)\big) &= c_1 \cdot h_2^{-x} \\
&= m \cdot h^y \cdot (g^y)^{-x} \\
&= m \cdot (g^x)^y \cdot (g^y)^{-x} \\
&= m \cdot g^{xy} \cdot g^{-yx} \\
&= m
\end{aligned}$$

ElGamal – implementation issues

Which group to choose?

E.g.: $\mathbf{QR}_p$, where p is a strong prime, i.e.: $q = \frac{p-1}{2}$ is also prime.

Plaintext space is a set of integers $\{1, \dots, q\}$.

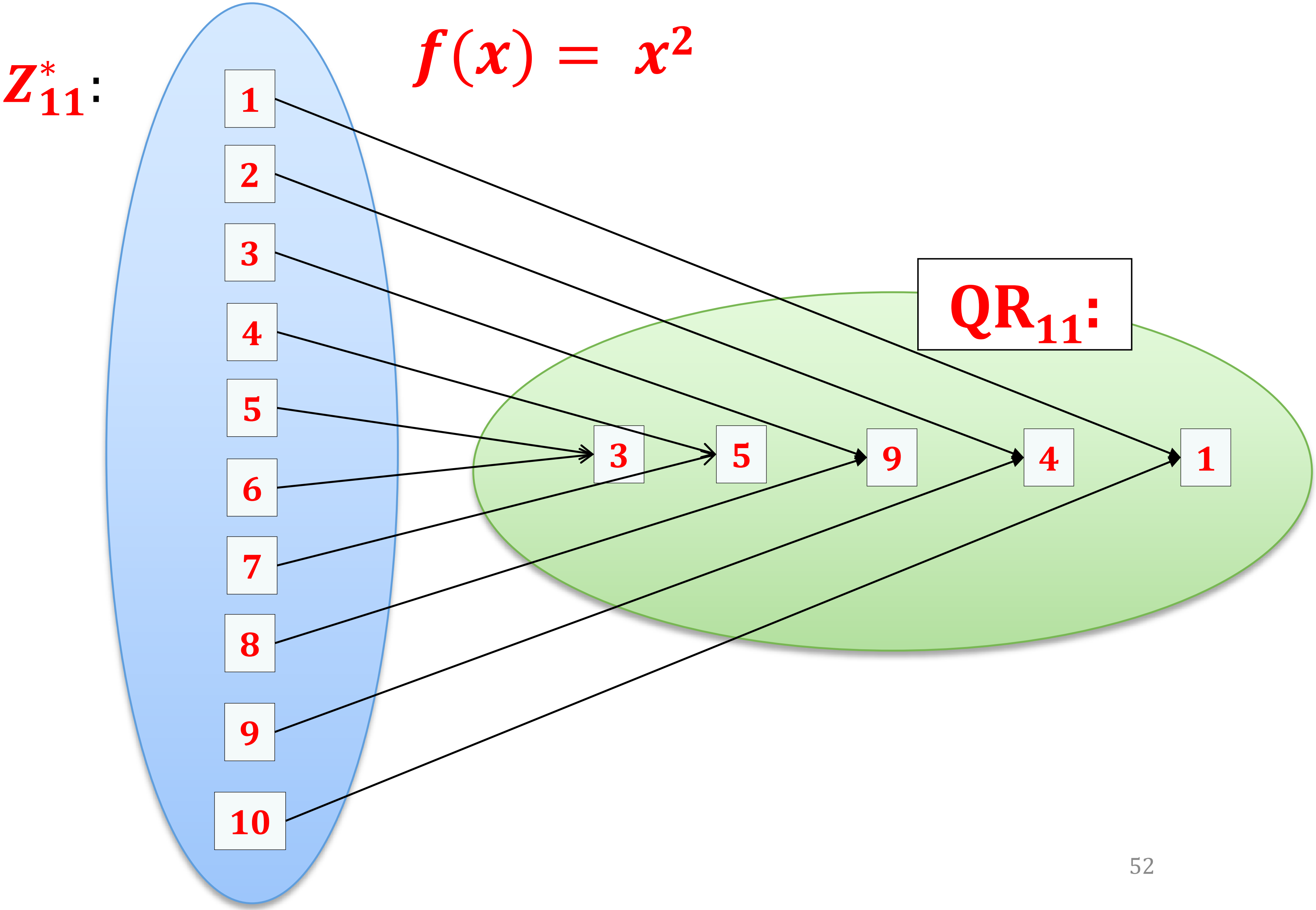
How to map an integer $i \in \{1, \dots, q\}$ to $\mathbf{QR}_p$?

Just square:

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

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

Remember this picture (from previous lectures)?



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 **Lecture 7**)

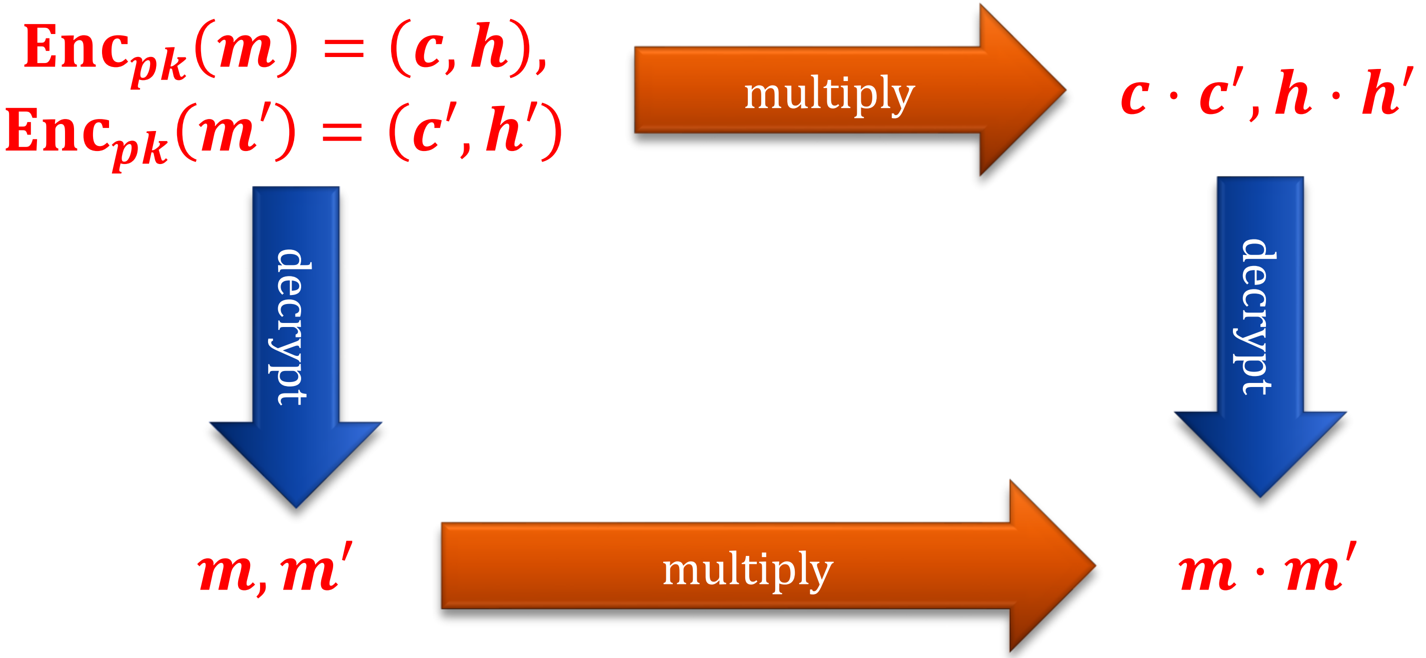
Plan

1. Rabin encryption
2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

ElGamal has an interesting property

homomorphism with respect to multiplication:

A "product of two ciphertexts" decrypts to a product of their corresponding messages.



Why?

- **public key**: (G, g, q, h)
- **private key**: (G, g, q, x)

$c := \mathbf{Enc}((G, g, q, h), m) := (m \cdot h^y, g^y)$, where $y \leftarrow G$

$c' := \mathbf{Enc}((G, g, q, h), m') := (m' \cdot h^{y'}, g^{y'})$, where $y' \leftarrow G$

product of c and c':

$$(m \cdot m' \cdot h^y \cdot h^{y'}, g^y \cdot g^{y'})$$
$$= \left(m \cdot m' \cdot h^{y+y'}, g^{y+y'}\right)$$

this is an encryption of $m \cdot m'$ with randomness $y + y'$

Homomorphism – good or bad?

Sometimes homomorphism is a security weakness (think of the **CCA security**).

On the other hand: it can also be a plus.

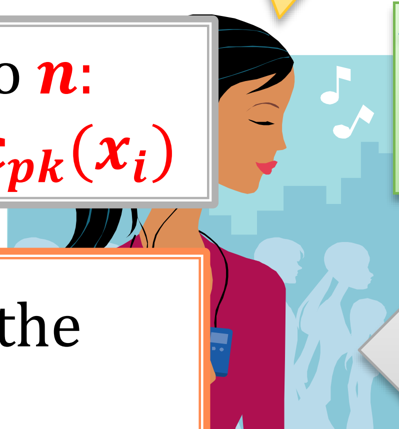
One example: cloud computing

Example: outsourcing computation

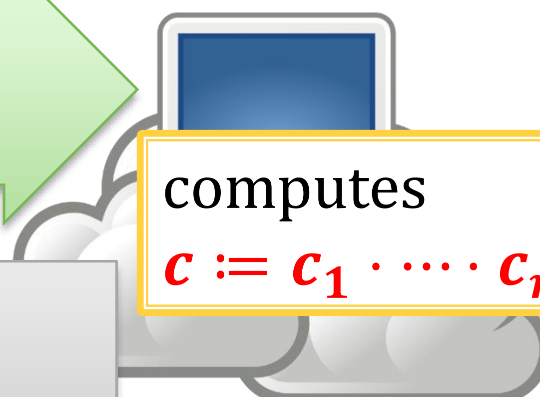
has a large set
$\{x_1, \dots, x_n\} \subseteq Z_p^*$
and wants to learn
$x = x_1 \cdot \dots \cdot x_n \bmod p$

generated a key pair
$pk = (Z_p, g, p-1, h)$
$sk = (Z_p, g, p-1, x)$

for $i = 1$ to n:
$c_i := \mathrm{Enc}_{pk}(x_i)$



$p,$
$c_1, \dots, c_n$



computes
$c := c_1 \cdot \dots \cdot c_n \bmod p$

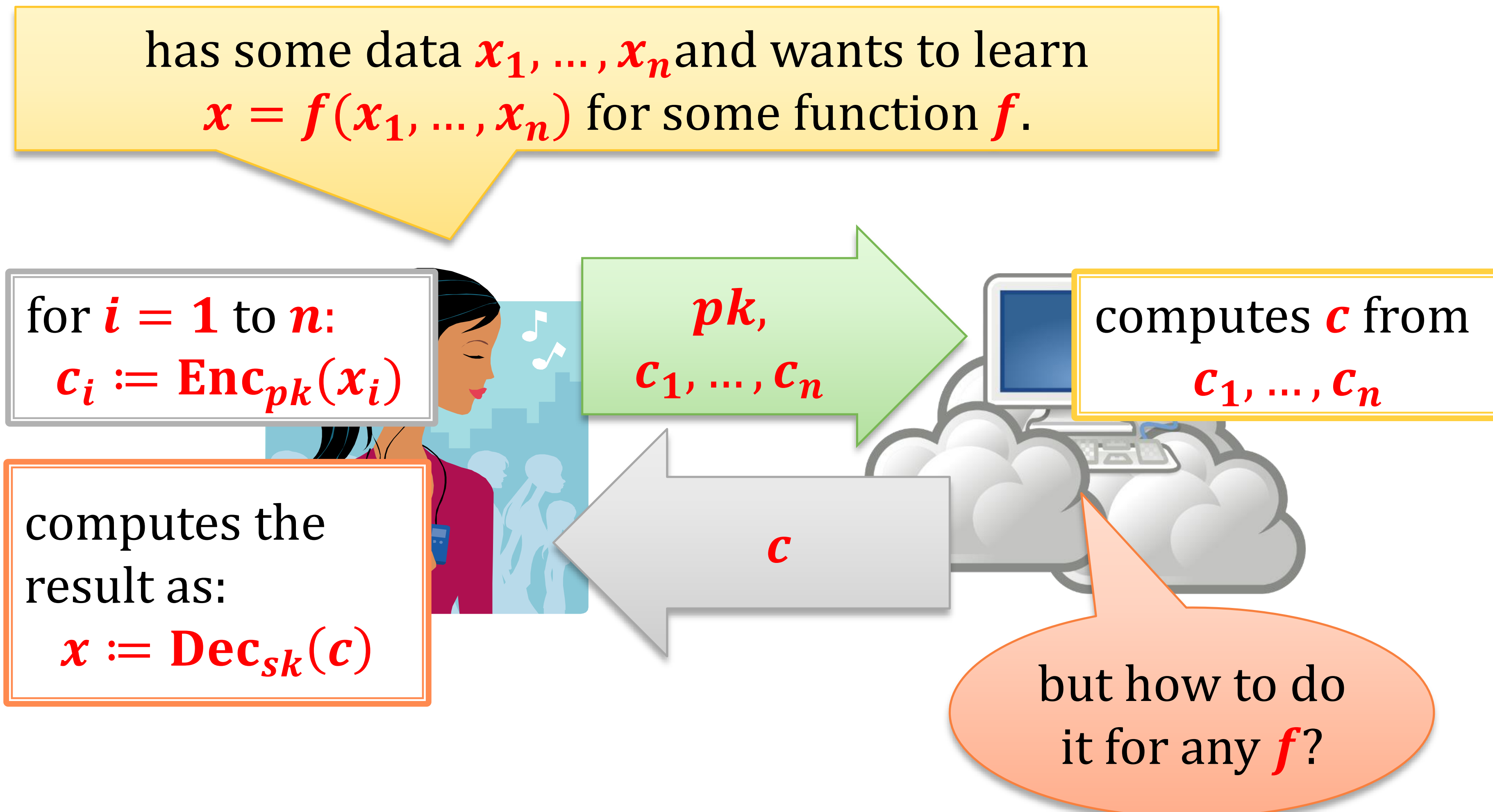
c

computes the result as:
$x := \mathrm{Dec}_{sk}(c)$

Observe: the server doesn't learn the x_i's!

This can be generalized!

The example on the previous slide was a bit artificial.
But think about the following.



Fully homomorphic encryption (FHE)

Constructing encryption scheme that would allow "**homomorphic computation**" of any function ***f*** was an **open problem** until **2009**.

The first such construction was given in:

> **Craig Gentry. Fully Homomorphic Encryption Using Ideal Lattices. ACM Symposium on Theory of Computing (STOC), 2009**.

Working towards construction of **practical** **FHE** is an active research area.

A natural (but much simpler) question

Can we construct an encryption scheme that is homomorphic **with respect to addition**?

Answer: **Yes, Paillier cryptosystem**

[Pascal Paillier "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes". EUROCRYPT 1999]

Paillier cryptosystem works over $\mathbb{Z}_{N^2}^*$, where N is an **RSA modulus**

Let $N := pq$.

public key: N

private key: (p, q)

How does $\mathbb{Z}_{N^2}^*$ look like?

Observe:

$$\varphi(N^2) = p(p-1) \cdot q(q-1)$$
$$= pq \cdot (p-1)(q-1)$$
$$= N \cdot \varphi(N)$$

Fact

$Z_{N^2}^*$ is isomorphic to $Z_N \times Z_N^*$ with the following isomorphism

$$f: Z_N \times Z_N^* \to Z_{N^2}^*$$

$$f(a, b) = (1 + N)^a \cdot b^N \bmod N^2$$

If $x = f(a, b)$ then we will also write: $x \leftrightarrow (a, b)$

[proof: exercise]

Another fact

Fact: for any integer $\boldsymbol{a}$ we have that

$$(1+N)^a = 1 + a \cdot N \pmod{N^2}$$

Proof:

$$(1+N)^a = 1 + \binom{a}{1} N^1 + \binom{a}{2} N^2 + \cdots + \binom{a}{1} N^a$$

$$= 1 + \binom{a}{1} N \pmod{N^2}$$

$$= 1 + a \cdot N \pmod{N^2}$$

QED

A consequence of this fact

Fact: for any integer $\boldsymbol{a}$ we have that

$$(\mathbf{1}+\boldsymbol{N})^{\boldsymbol{a}} = \mathbf{1} + \boldsymbol{a} \cdot \boldsymbol{N} \pmod{\boldsymbol{N}^2}$$

Consequence: order of $\mathbf{1}+\boldsymbol{N}$ in $\boldsymbol{Z}^*_{\boldsymbol{N}^2}$ is $\boldsymbol{N}$.

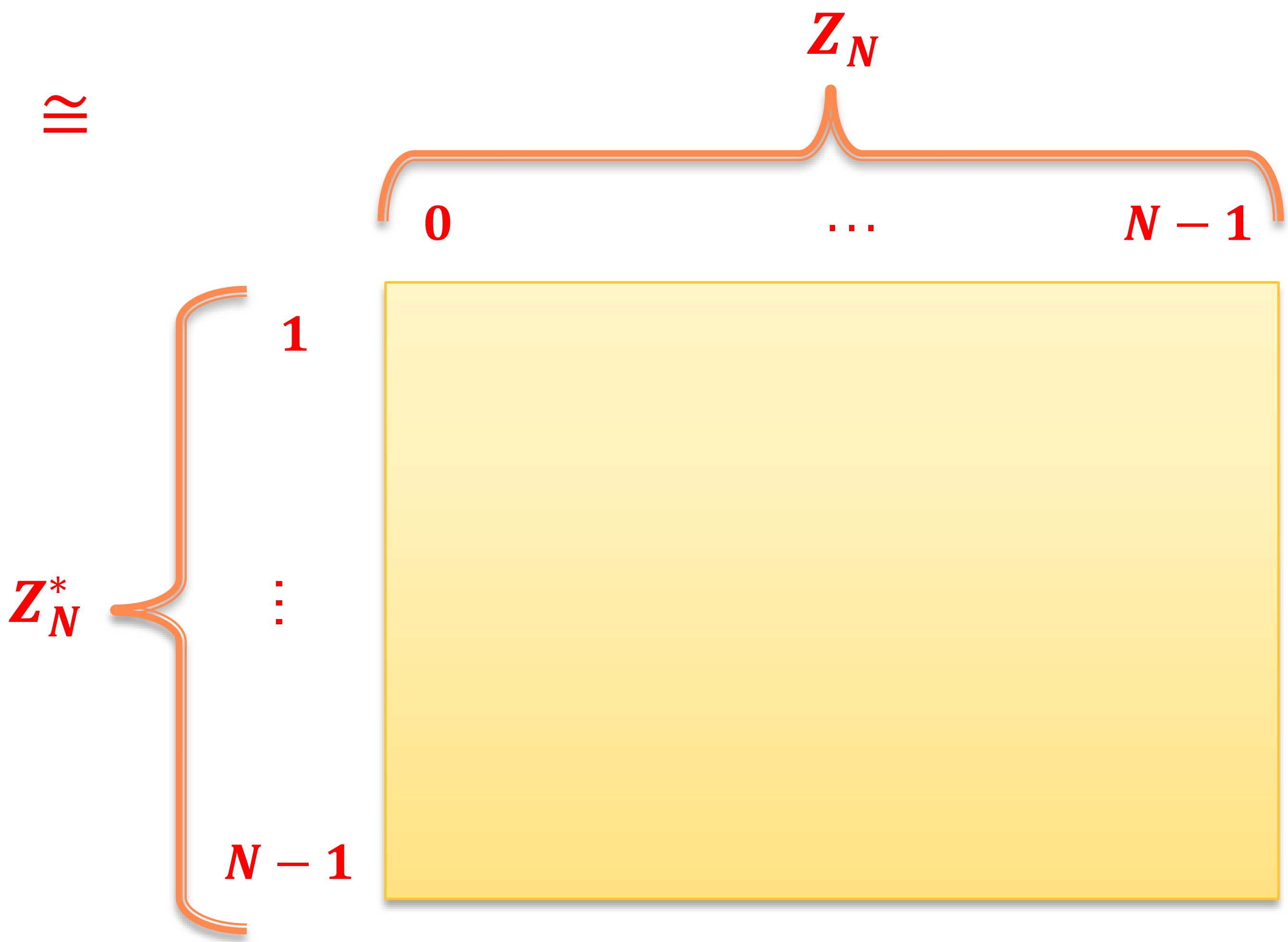
why?

because:

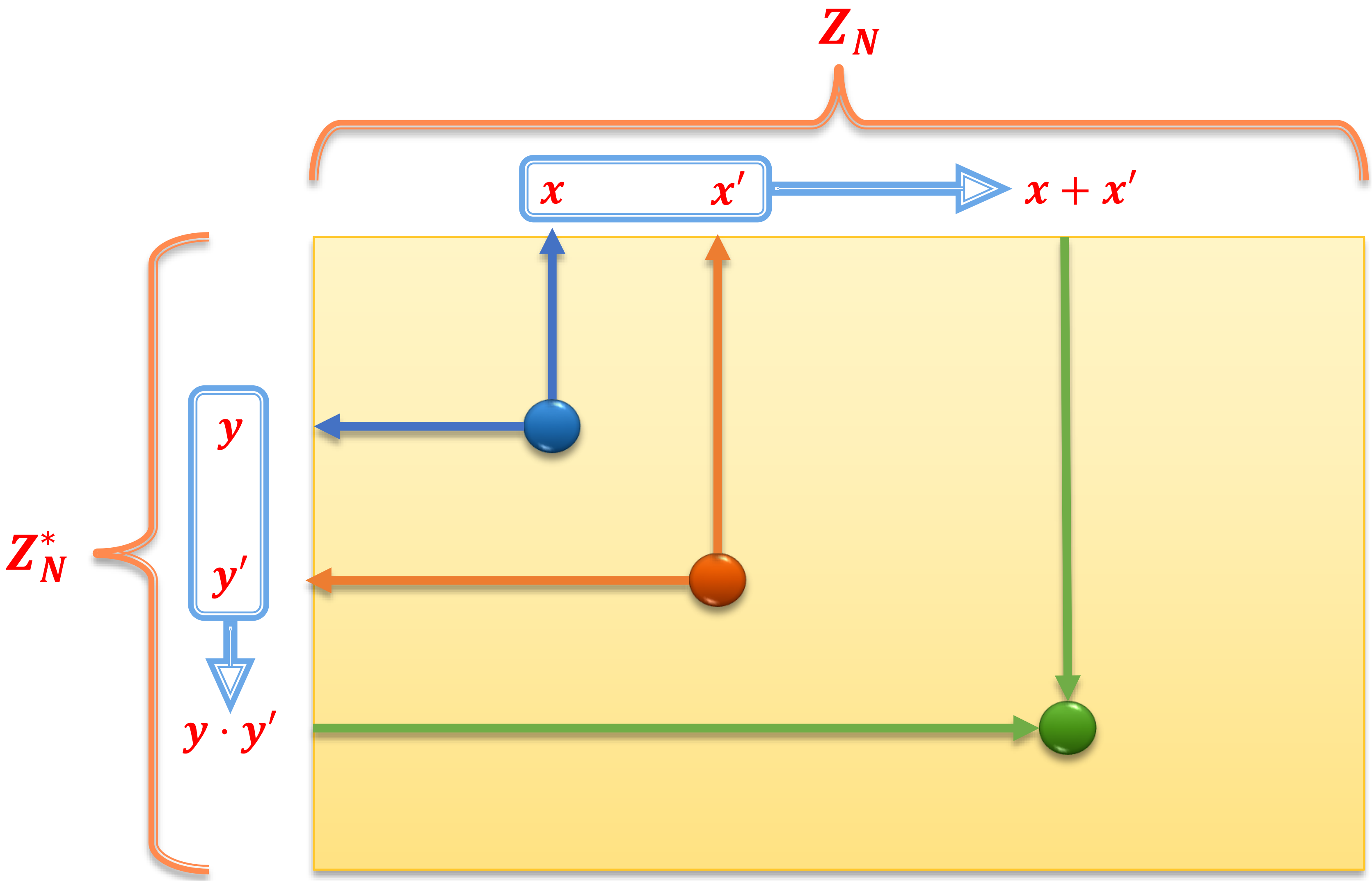
- for $\mathbf{0} < \boldsymbol{a} < \boldsymbol{N}$ we have $\mathbf{1} < \mathbf{1} + \boldsymbol{a} \cdot \boldsymbol{N} < \boldsymbol{N}^2$
- and $\mathbf{1} + \boldsymbol{N} \cdot \boldsymbol{N} = \mathbf{1} \pmod{\boldsymbol{N}^2}$

Structure of $Z^*_{N^2}$

$Z^*_{N^2} \cong$



Multiplication in $Z_{N^2}^*$



Nth residues in $Z_{N^2}^*$

A number $y \in Z_{N^2}^*$ is called an N**th residue modulo** N^2 if there exists $x \in Z_{N^2}^*$ such that

$$y = x^N \bmod N^2$$

How do the Nth residues look like?

A form of every $\boldsymbol{N}$th residue

Suppose $\boldsymbol{x} \leftrightarrow (\boldsymbol{a}, \boldsymbol{b})$.

Then

$$\begin{aligned} \boldsymbol{x}^{\boldsymbol{N}} &\leftrightarrow (\boldsymbol{N} \cdot \boldsymbol{a} \textbf{ mod } \boldsymbol{N}, \boldsymbol{b}^{\boldsymbol{N}} \textbf{ mod } \boldsymbol{N}) \\ &= (\mathbf{0}, \boldsymbol{b}^{\boldsymbol{N}} \textbf{mod } \boldsymbol{N}) \end{aligned}$$

So every $\boldsymbol{N}$th residue is of a form

$$\boldsymbol{y} \leftrightarrow (\mathbf{0}, \boldsymbol{c})$$

Is every element of this form an $\boldsymbol{N}$th residue?

Yes!

A proof that every element $(\mathbf{0}, \boldsymbol{c})$ is an $\boldsymbol{N}$th residue

Take $\boldsymbol{y} \leftrightarrow (\mathbf{0}, \boldsymbol{c})$. Let $\boldsymbol{d} = \boldsymbol{N}^{-1} \textbf{ mod } \boldsymbol{\varphi}(\boldsymbol{N})$.

For an arbitrary $\boldsymbol{a} \in \boldsymbol{Z}_{\boldsymbol{N}}$ let $\boldsymbol{x}$ be such that

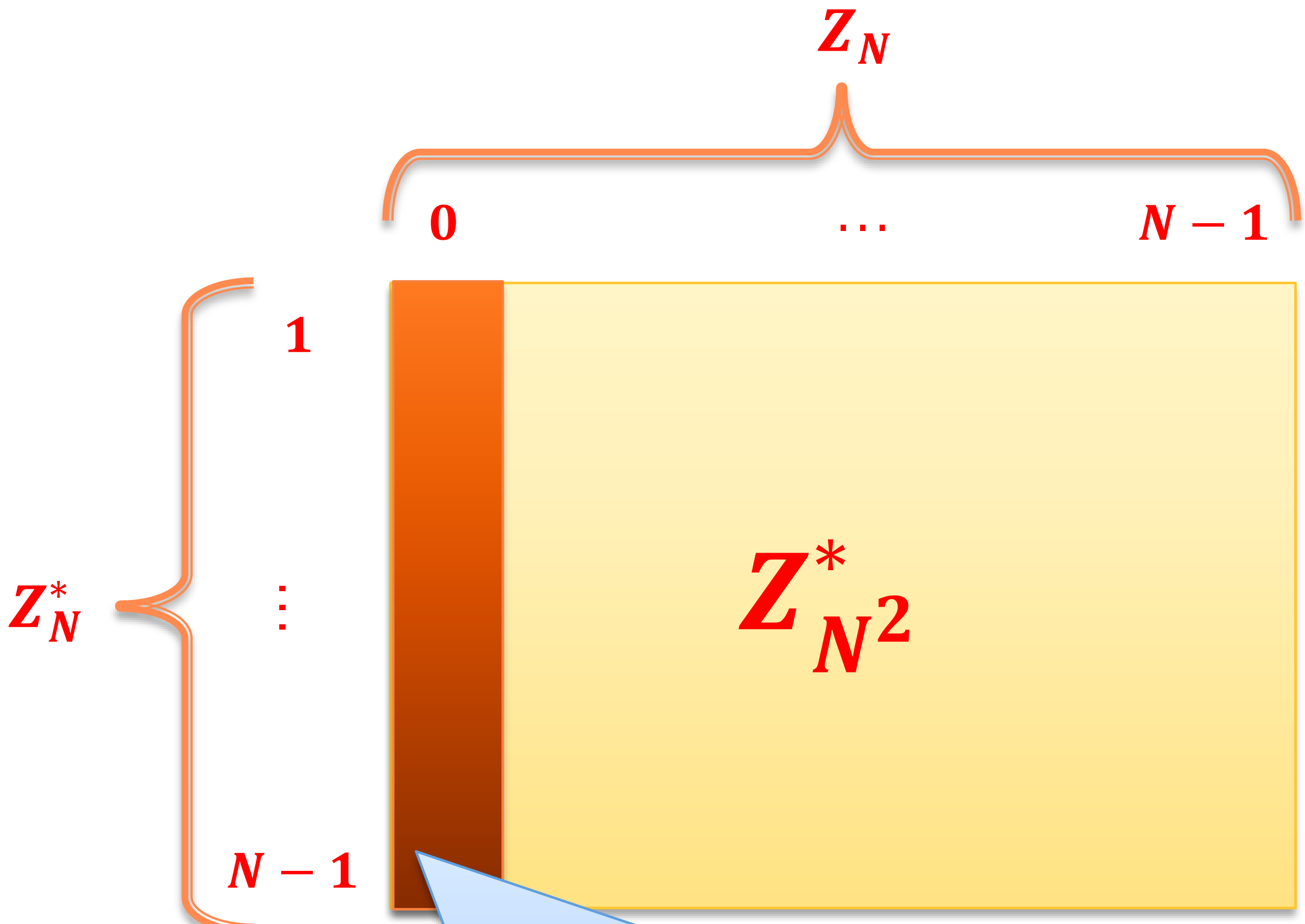
$$\boldsymbol{x} \leftrightarrow \left(\boldsymbol{a}, \boldsymbol{c}^{\boldsymbol{d}}\right)$$

We have:

$$\begin{aligned} \boldsymbol{x}^{\boldsymbol{N}} &\leftrightarrow \left(\boldsymbol{N}\boldsymbol{a} \textbf{ mod } \boldsymbol{N}, \boldsymbol{c}^{\boldsymbol{dN}} \textbf{ mod } \boldsymbol{N}\right) \\ &= \left(\mathbf{0}, \boldsymbol{c}^{\boldsymbol{dN} \textbf{ mod } \boldsymbol{\varphi}(\boldsymbol{N})}\right) \\ &= (\mathbf{0}, \boldsymbol{c}^{\mathbf{1}}) \\ &= (\mathbf{0}, \boldsymbol{c}) \end{aligned}$$

Observe: this also shows that every $\boldsymbol{N}$th residue $\boldsymbol{y}$ has exactly $\boldsymbol{N}$ roots $\sqrt[\boldsymbol{N}]{\boldsymbol{y}}$.

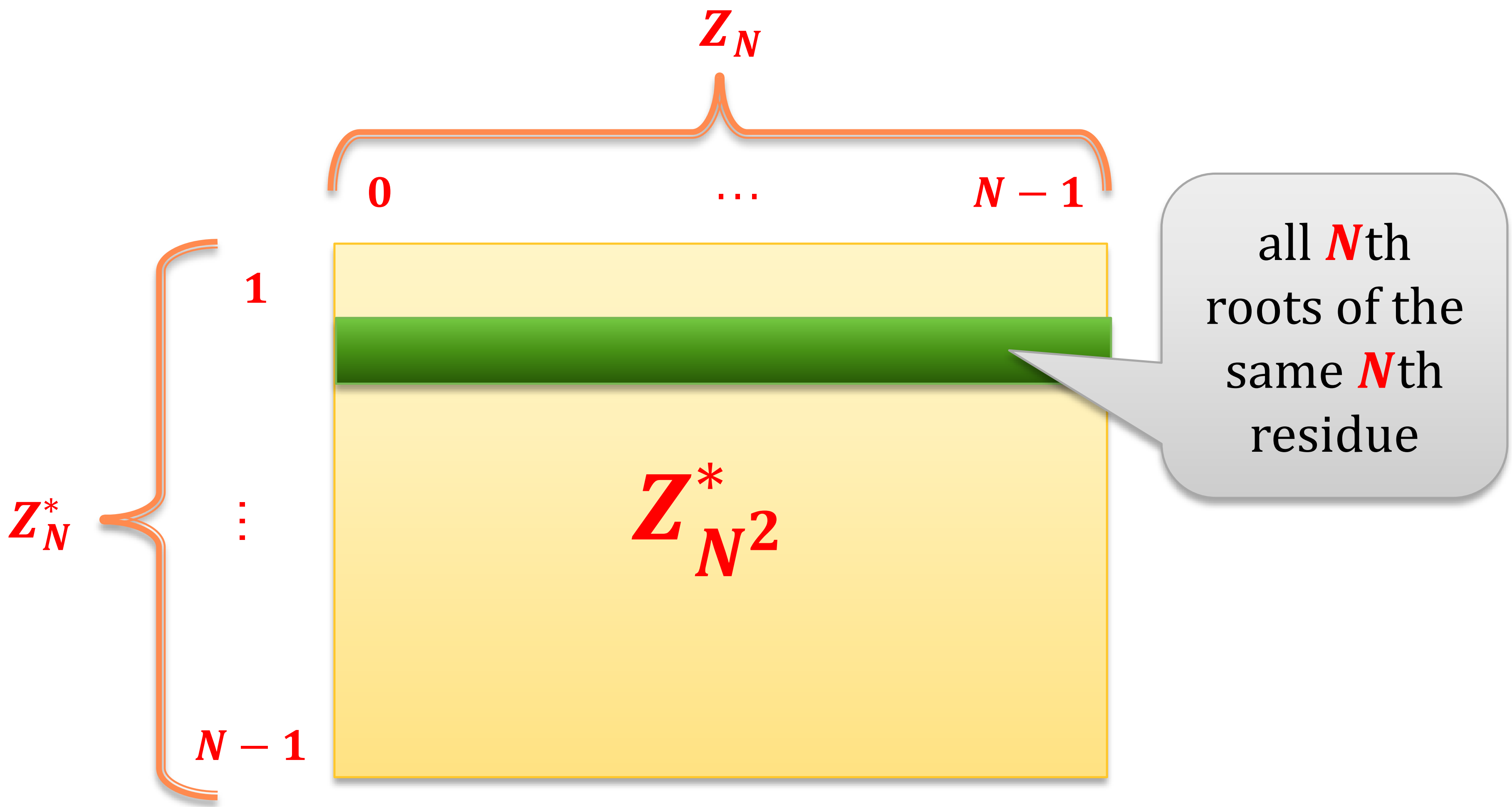
The Nth residues pictorially



Nth residues. Denote this set $\mathbf{Res}(N^2)$

Also

The Nth roots of every $(0, c)$ have a form (a, c^d):



Corollary

It's easy to choose a random $\boldsymbol{N}$th residue:

Just take a random element $\boldsymbol{x} \leftarrow \boldsymbol{Z}^*_{\boldsymbol{N}^2}$ and compute $\boldsymbol{y} = \boldsymbol{x}^{\boldsymbol{N}} \textbf{ mod } \boldsymbol{N}^2$.

Which problem is **hard** $\boldsymbol{Z}^*_{\boldsymbol{N}^2}$ (if one doesn't know $\boldsymbol{p}$ and $\boldsymbol{q}$)?

Decisional composite residuosity (**DCR**) assumption

Informally:

It is hard to distinguish random element of $\mathbf{Res}(N^2)$ from a random element of $Z^*_{N^2}$.

How to encrypt?

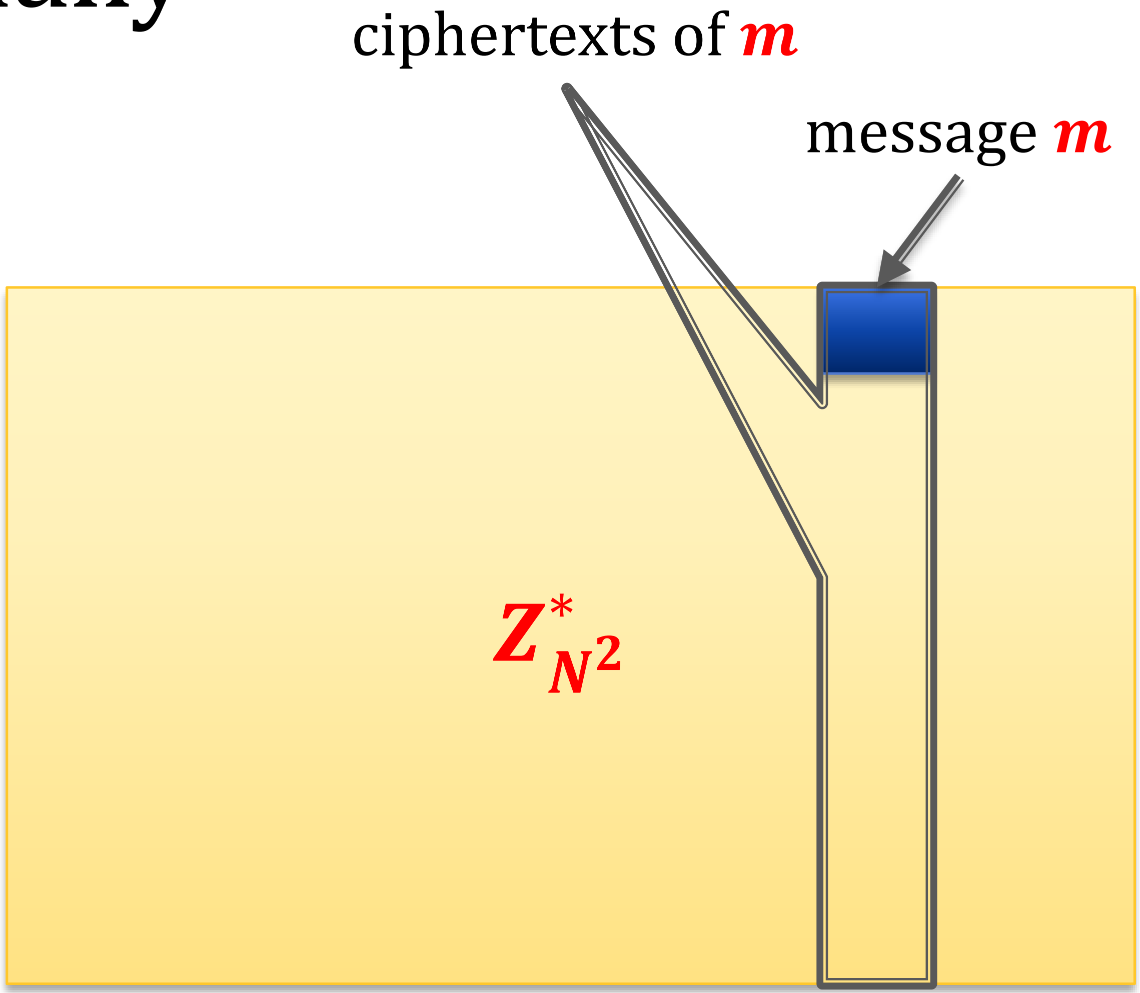
Main idea: messages are elements $x \leftrightarrow (a, 1)$ (for $a \in Z_N$)

$$Z^*_{N^2}$$

To encrypt a message m multiply it by a random $r \leftarrow \mathbf{Res}(N^2)$:

$$\mathbf{Enc}_N(m) = m \cdot r$$

Pictorially



Two questions

1. Is this **secure**?
2. How to **decrypt**?

Security follows from the **DCR** assumption

Proof (sketch):
Take the original scheme

$$\mathbf{Enc}_N(m) = m \cdot r \text{ where } r \leftarrow \mathbf{Res}(N^2)$$

and modify it as follows:

$$\mathbf{Enc}_N(m) = m \cdot r \text{ where } r \leftarrow Z^*_{N^2}$$

Easy to see:
1. the **modified scheme hides the message completely** (it's a "generalized one-time pad")
2. if these **two schemes can be distinguished then the DCR assumption is broken**.

How to decrypt?

$\mathbf{Enc}_N(m) = m \cdot r$ where
$r \leftarrow \mathbf{Res}(N^2)$

Let's view encryption as a function in $Z_N \times Z_N^*$:

$$\mathbf{Enc}_N(a, 1) \leftrightarrow (a + 0, 1 \cdot b) \text{ where } b \leftarrow Z_N^*$$
$$= (a, b)$$

Problem:
the receiver can only see $f(a, b)$.
How can he "extract" a from it?

Observation

$$\left(f(a,b)\right)^{\varphi(N)} \bmod N^2 \leftrightarrow \left(\varphi(N)\cdot a \bmod N, b^{\varphi(N)} \bmod N\right)$$
$$= \left(\varphi(N)\cdot a \bmod N, 1\right)$$
$$\leftrightarrow f\left(\varphi(N)\cdot a \bmod N, 1\right)$$
$$= (1+N)^{\varphi(N)\cdot a \bmod N}\cdot 1^n \bmod N^2$$
$$= (1+N)^{\varphi(N)\cdot a \bmod N} \bmod N^2$$
$$= \underbrace{1 + \left(\varphi(N)\cdot a \bmod N\right)\cdot N}_{< N^2} \bmod N^2$$
$$= 1 + \left(\varphi(N)\cdot a \bmod N\right)\cdot N$$

here we use the fact that
$$(1+N)^a = 1 + a\cdot N \pmod{N^2}$$

So:

$$\varphi(N)\cdot a \bmod N = \frac{\left(f(a,b)\right)^{\varphi(N)} \bmod N^2 - 1}{N}$$

Continued:

denote it $\boldsymbol{z}$

We got that

$$\boldsymbol{\varphi(N) \cdot a \bmod N = \frac{\left(f(a,b)\right)^{\varphi(N)} \bmod N^2 - 1}{N}}$$

Therefore

$$\boldsymbol{a = z \cdot \left(\varphi(N)\right)^{-1} \bmod N}$$

Paillier encryption

Key generation: let $N := pq$ like in RSA

public key: N

private key: (p, q)

Encryption:

$\mathbf{Enc}_N(m) = (1 + N)^m \cdot r^N \bmod N^2$ where $r \leftarrow Z_N^*$

Decryption:

$$\mathbf{Dec}_{p,q}(c) = \frac{(c^{\varphi(N)} \bmod N^2) - 1}{N} \cdot \varphi(N)^{-1} \bmod N$$

Why is this additively homomorphic?

$\boldsymbol{c} = \mathbf{Enc}_{\boldsymbol{N}}(\boldsymbol{m}) \leftrightarrow (\boldsymbol{m}, \boldsymbol{r})$ where $\boldsymbol{r} \leftarrow \boldsymbol{Z}_{\boldsymbol{N}}^{*}$

$\boldsymbol{c}' = \mathbf{Enc}_{\boldsymbol{N}}(\boldsymbol{m}') \leftrightarrow (\boldsymbol{m}', \boldsymbol{r}')$ where $\boldsymbol{r}' \leftarrow \boldsymbol{Z}_{\boldsymbol{N}}^{*}$

We have:

$$\boldsymbol{c} \cdot \boldsymbol{c}' \leftrightarrow (\boldsymbol{m}, \boldsymbol{r}) \cdot (\boldsymbol{m}, \boldsymbol{r})$$
$$= (\boldsymbol{m} + \boldsymbol{m}', \boldsymbol{r} \cdot \boldsymbol{r}')$$
$$\leftrightarrow \mathbf{Enc}_{\boldsymbol{N}}(\boldsymbol{m} + \boldsymbol{m}') \text{ with randomness } \boldsymbol{r} \cdot \boldsymbol{r}'$$

Plan

1. Rabin encryption
2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

ElGamal vs. RSA

In practice **RSA** and **ElGamal** (in $\boldsymbol{Z_p^*}$) have similar security for equivalent key lengths.

- **RSA** is slightly more efficient
- **ElGamal** has a ciphertext twice as long as the plaintext
- But **ElGamal** can be generalized to other groups (e.g. the **elliptic curves**) where it is much more efficient!

NIST recommendations

bits of security	RSA modulus length	discrete log in order q subgroups of Z_p^*	discrete log in elliptic curves of order:
≤ 80	1024	$\lvert p\rvert = 1024$ $\lvert q\rvert = 160$	160
112	2048	$\lvert p\rvert = 2048$ $\lvert q\rvert = 224$	224
128	3072	$\lvert p\rvert = 3072$ $\lvert q\rvert = 256$	256
192	7680	$\lvert p\rvert = 7680$ $\lvert q\rvert = 384$	384
256	15360	$\lvert p\rvert = 15360$ $\lvert q\rvert = 512$	512

[NIST Special Publication 800-57 Part 1 Revision 4 Recommendation for Key Management]

Quantum attacks

All the schemes presented so far can be broken by quantum computers using Shor's algorithm.

[Peter W. Shor "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer" 1995]

Peter Shor
1959—

There exists public-key encryption schemes that are believed to be secure against quantum computers (see **post-quantum cryptography**)

Plan

1. Rabin encryption
2. ElGamal encryption
3. Homomorphic encryption and Paillier cryptosystem
4. Practical considerations
5. Theoretical overview

A natural question

Is public-key encryption a member of **Minicrypt**?

Answer: **NO** (as far as we know).

More precisely: nobody knows how to construct **PKE** from **one-way functions**.

However, the following implication is known:

This is proven using the **hardcore predicates**.

Hard-core predicates

Hard-core **predicates** are a generalization of hard-core **bits**.

Definition (informal)

$\boldsymbol{\pi}: \{\mathbf{0}, \mathbf{1}\}^{n} \rightarrow \{\mathbf{0}, \mathbf{1}\}$ is a hard core predicate for a trap-door permutation $\boldsymbol{f}: \{\mathbf{0}, \mathbf{1}\}^{n} \rightarrow \{\mathbf{0}, \mathbf{1}\}^{n}$ if it is hard to guess $\boldsymbol{\pi}(\boldsymbol{f}^{-1}(\boldsymbol{y}))$ from $\boldsymbol{y}$ (with probability significantly better than $\mathbf{1/2}$).

A fact

Does every trap-door permutation have a hard-core predicate?

Almost:

Suppose that $\boldsymbol{f}$ is a trap-door permutation.

It can be used to build a trap-door permutation $\boldsymbol{g}$ that has a hard-core predicate.

How to encrypt with such an $\boldsymbol{g}$?

Encryption for messages of length **1**:

public key: description of $\boldsymbol{g}$

private key: trapdoor $\boldsymbol{t}$ for $\boldsymbol{g}$

$$\mathbf{Enc}_{g}(b) = (\pi(x) \oplus b, g(x))$$

where $x \in Z_N^*$ is random.

$$\mathbf{Dec}_{t}(b', y) = \pi\left(g^{-1}(y)\right) \oplus b$$

The general picture