

Lecture 11

Commitment Schemes and

Zero Knowledge

Stefan Dziembowski

www.crypto.edu.pl/Dziembowski

University of Warsaw



Plan

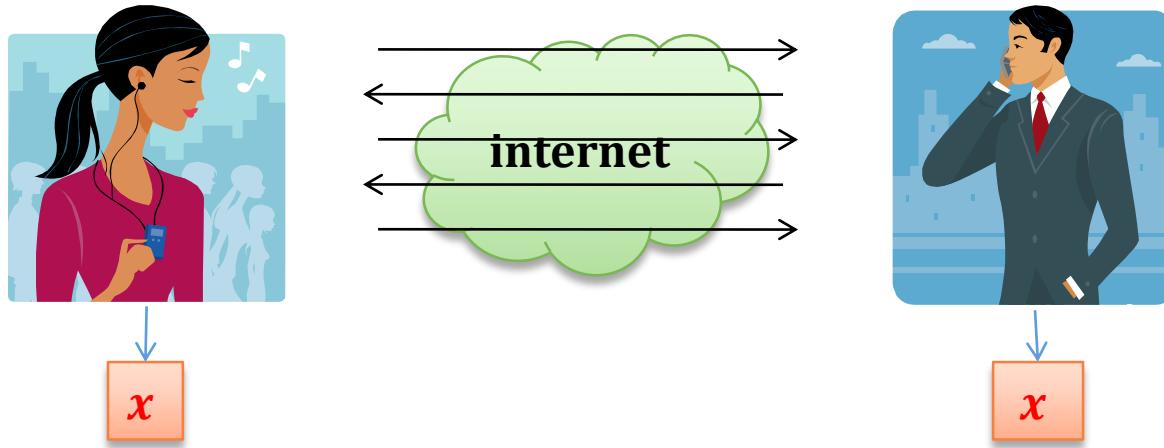


1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications

Coin-flipping by telephone [Blum'81]

privacy and authenticity is not a problem

Suppose Alice and Bob are connected by a secure internet link:



The goal of Alice and Bob is to toss a coin.

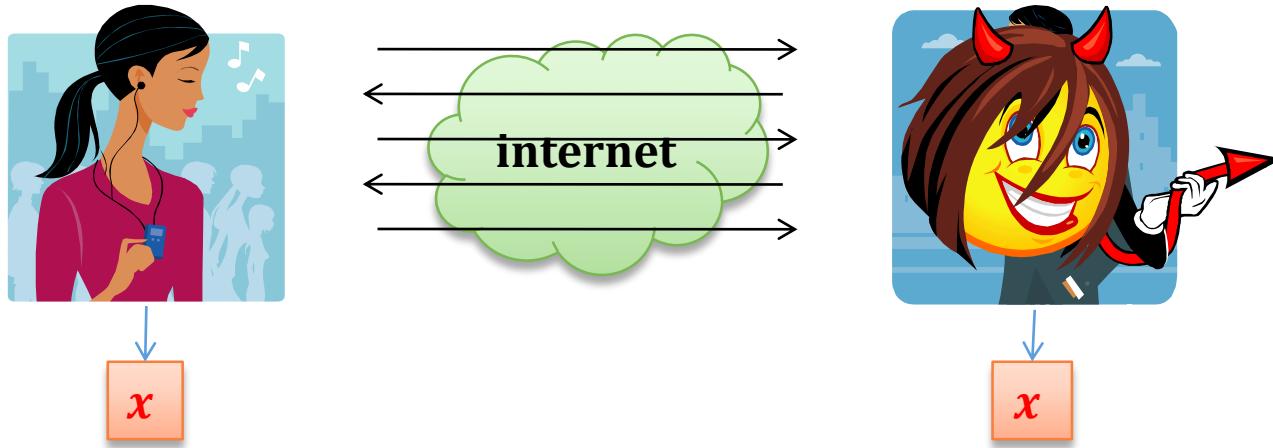
In other words:

They want to execute some protocol π in such a way that at the end of the execution they both output the same bit x distributed uniformly over $\{0, 1\}$.

How to define security? [1/2]

Let us just stay at an informal level...

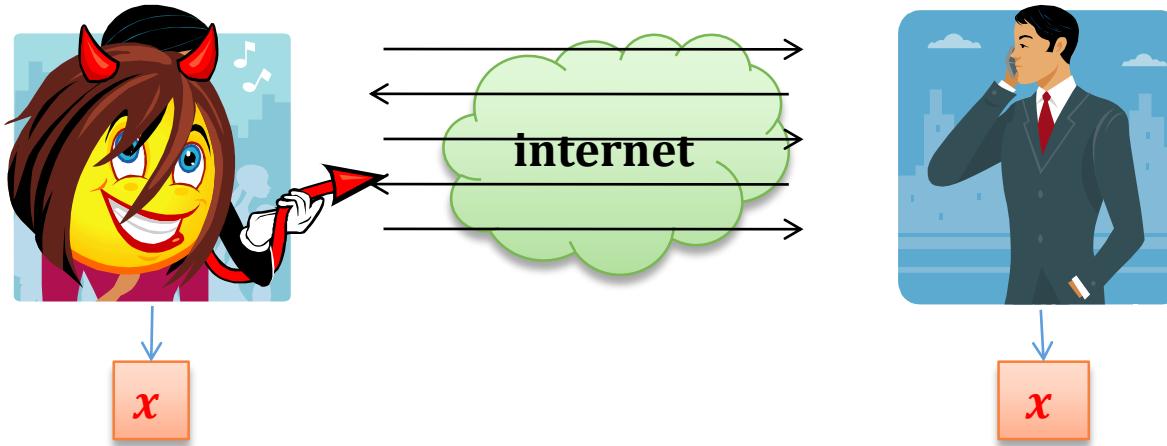
From the point of view of Alice:



even if **Bob** is **cheating** (i.e.: he doesn't follow the protocol):
if the protocol terminates successfully, then **x** has a
uniform distribution

How to define security? [2/2]

The same holds from the point of view of Bob



even if Alice is **cheating** (i.e.: he doesn't follow the protocol):
if the protocol terminates successfully, then x has a
uniform distribution

Note the difference

Unlike what we saw on the previous lectures:

the enemy can be one of the parties

(**not** an external entity)

A cheating party is sometimes called a **corrupted** party, or a **malicious** party.

We will see many other examples of this later!

How to solve this problem?

Idea

Remember the old
game:

rock-paper-scissors?





Alice

Bob

| | | | |
|--|---|---|---|
| |  |  |  |
| | draw | Alice wins | Bob wins |
| |  | Bob wins | Alice wins |
| |  | Alice wins | Bob wins |

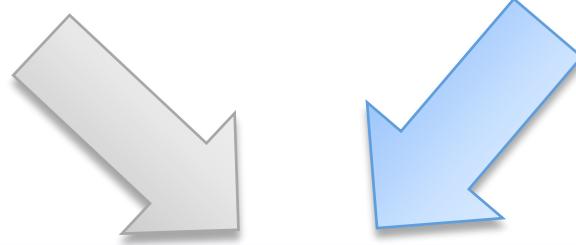
Let's simplify this game

| | | Alice | | |
|--|--|---------|------------|------------|
| | | $A = 0$ | $A = 1$ | |
| | | $B = 0$ | Alice wins | Bob wins |
| | | $B = 1$ | Bob wins | Alice wins |

In other words: Alice wins iff $A \oplus B = 0$.

Another way to look at it

Alice
has an input A Bob
has an input B

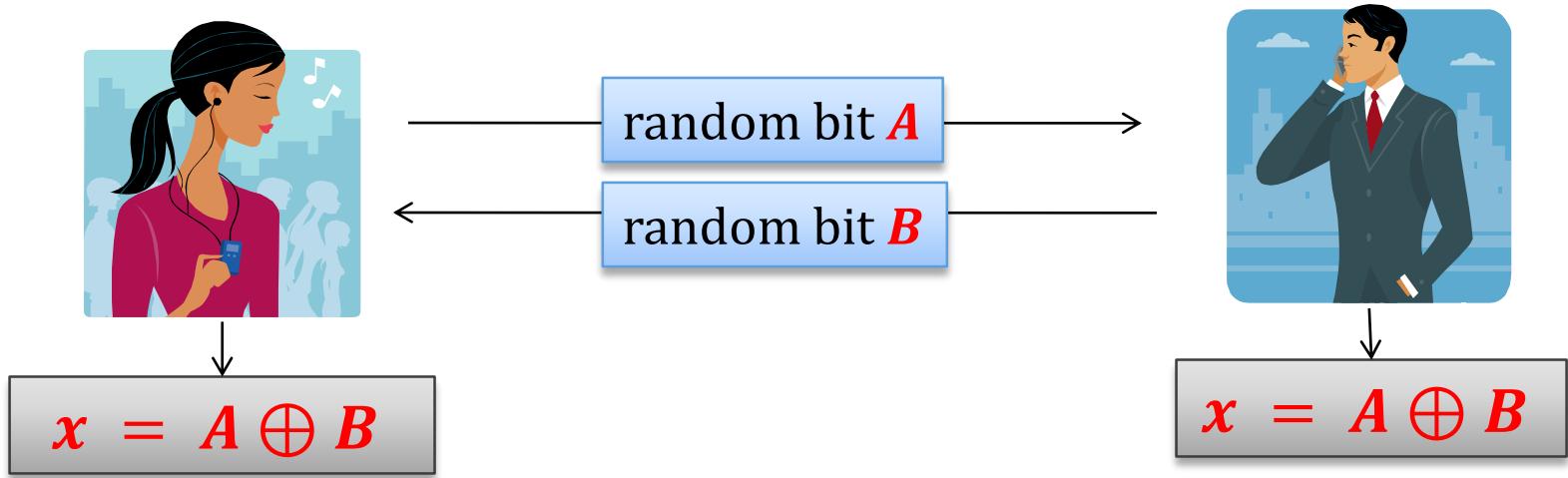


they should jointly compute

$$x = A \oplus B$$

(in a secure way)

What to do?



Problem:

A and B should be sent at the same time

(e.g. if A is sent before B then a malicious **Bob** can set $B := x \oplus A$, where x is chosen by him).

How to prevent this?

Seems hard:

the internet is not synchronous...

A solution:

bit commitments

Plan



1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. Applications

Commitment schemes – an intuition

a bit **b**



Alice “commits
herself to **b**”

Alice sends a **locked** box to Bob



[binding] from now Alice cannot change **b**,
[hiding] but Bob doesn't know **b**



Alice can later send the key to Bob

Alice “opens the
commitment”



Commitment schemes – a functional definition

A **commitment scheme** is a protocol executed between Alice and Bob consisting of two phases: **commit** and **open**.

In the **commit** phase:

- **Alice** takes some input bit b .
- **Bob** takes no input.

In the **open** phase:

- **Alice** outputs nothing
- **Bob** outputs b , or **error**

Security requirements - informally

[binding]

After the **commit** phase there exists at most one value b that can be open in the **open** phase.

[hiding]

As long as the **open** phase did not start **Bob** has no information about b .

How to define security formally?

Not so trivial – remember that the parties can misbehave arbitrarily.

We do not present a complete definition here.

(The hiding property can be defined using the “**indistinguishability**” principle.)

The definition depends on some options.

1. What is the computational power of a **cheating Alice**?
2. What is the computational power of a **cheating Bob**?

The computational power of the adversary

If a cheating Alice can be infinitely powerful, we say that the protocol is **unconditionally binding**.

Otherwise it is **computationally binding**.

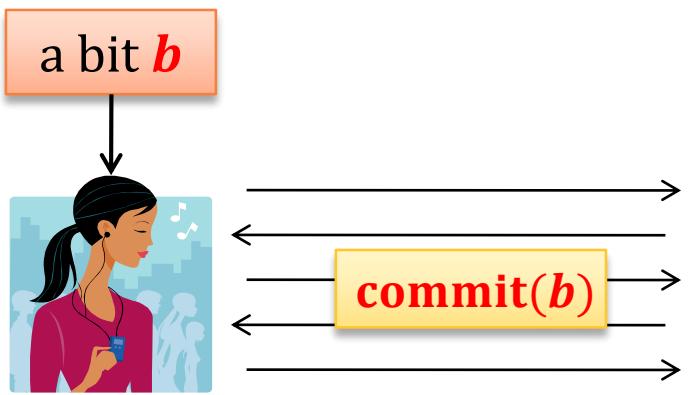
If a cheating Bob can be infinitely powerful, we say that the protocol is **unconditionally hiding**.

Otherwise it is **computationally hiding**.

Of course, to be formal we would need to introduce a security parameter...

Unconditionally hiding and binding commitment schemes do not exist

Proof (intuition)



There are two options:

1. there exists a way to open $1 - b$, or

in this case “infinitely powerful”
Alice can cheat

2. there doesn't exist such a way

in this case “infinitely powerful”
Bob can learn b

So, how does it solve the coin-flipping problem?

chooses a random bit $A \leftarrow \{0, 1\}$



commits to A



sends B

output
 $A \oplus B$

chooses a random bit $B \leftarrow \{0, 1\}$

opens A

output
 $A \oplus B$

Problem

Alice can always refuse to send the last message.

This is unavoidable (there has to be the **last message** in the protocol).

But they can use a convention:

if **Alice** didn't send the last message – she lost!

Plan

1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. Applications



Remember quadratic residues modulo an RSA modulus?

$$\mathbf{QR}_N = \{x^2 \bmod N : x \in \mathbf{Z}_N^*\}$$

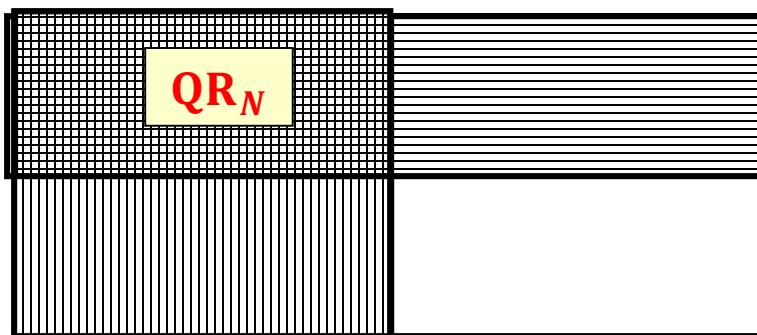
Fact: For $N = pq$ we have $|\mathbf{QR}_N| = |\mathbf{Z}_N^*| / 4$.

\mathbf{Z}_N^* :

$\mathbf{QR}_p \bmod p$

\mathbf{QR}_q

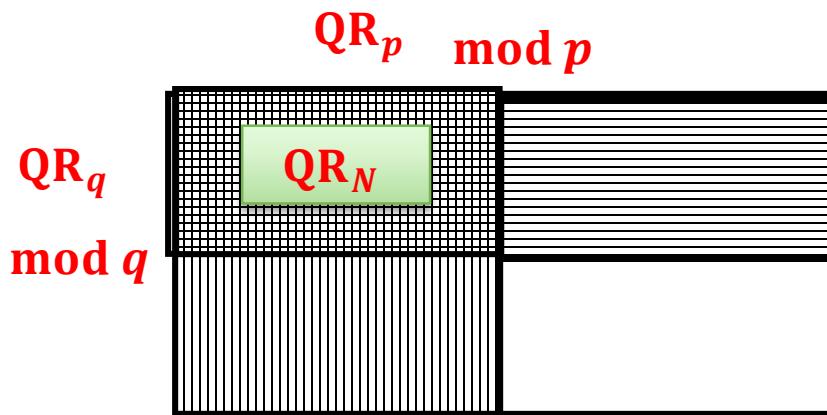
$\bmod q$



Jacobi Symbol

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

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



$J_N(x) :=$

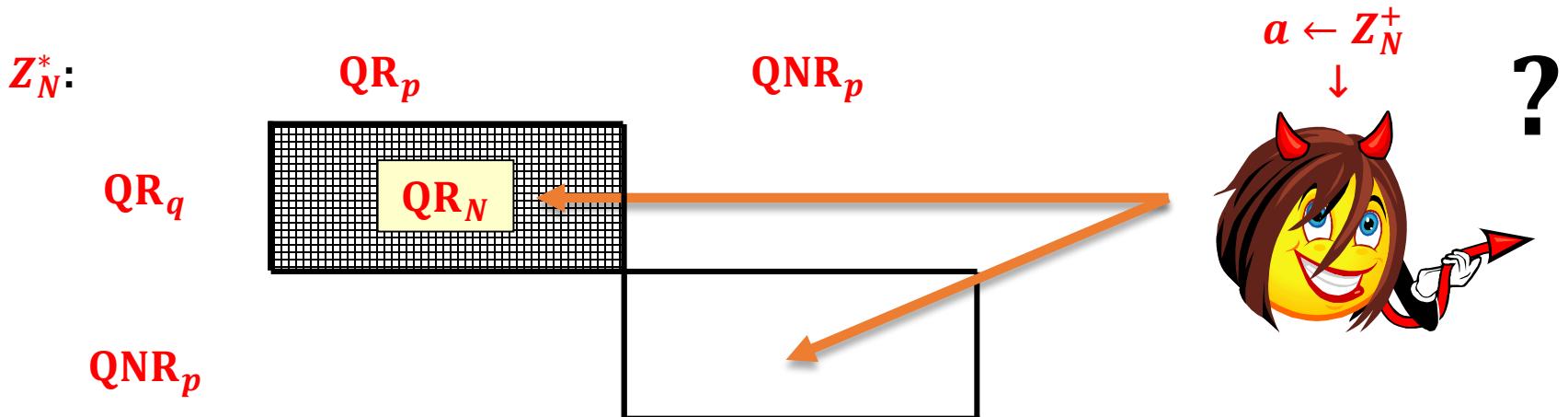
| | |
|----|----|
| +1 | -1 |
| -1 | +1 |

It is a subgroup of \mathbf{Z}_N^*

$\mathbf{Z}_N^+ := \{x \in \mathbf{Z}_N^*: J_N(x) = +1\}$

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

Quadratic Residuosity Assumption



Quadratic Residuosity Assumption (QRA):

For a random $a \leftarrow \mathbf{Z}_N^+$ it is computationally hard to determine if $a \in \mathbf{QR}_N$.

Formally: for every **polynomial-time** probabilistic algorithm D the value:

$$\left| P(D(N, a) = Q_N(a)) - \frac{1}{2} \right|$$

(where $a \leftarrow \mathbf{Z}_N^+$) is **negligible**.

Where a predicate $Q_N: \mathbf{Z}_N^+ \rightarrow \{0, 1\}$ is defined as follows:
 $Q_N(a) = 1$ if $a \in \mathbf{QR}_N$
 $Q_N(a) = 0$ otherwise

A construction based on QRA



select a random RSA modulus $N = pq$.

Let

$$x := \begin{cases} \text{a random } \mathbf{QR}_N & \text{if } b = 1 \\ \text{a random } \mathbf{Z}_N^+ \setminus \mathbf{QR}_N & \text{if } b = 0 \end{cases}$$



to **commit** to b sends (N, x)

to **open** a commitment send (p, q) .

if $N \neq pq$.
then output **error**

Otherwise output
 $Q_N(x)$

This commitment scheme is unconditionally binding

Why?

Suppose **Alice** has sent (N, x) to **Bob**.

What can **Bob** output at the end of the opening phase?

There exists the following options:

- N is not an RSA modulus – in this case **Bob** will always output **error**,
- $x \in \text{QR}_N$ – in this case **Bob** can only output **0** or **error**,
- $x \notin \text{QR}_N$ – in this case **Bob** can only output **1** or **error**.

This commitment scheme is computationally hiding, assuming QRA holds

Proof (intuition)

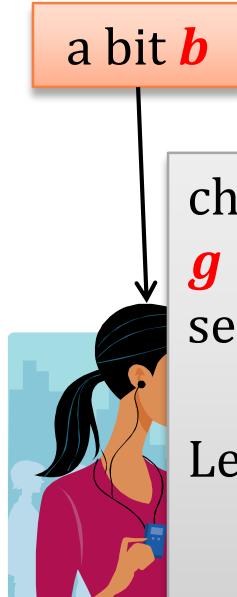
To distinguish between $\mathbf{b} = \mathbf{0}$ and $\mathbf{b} = \mathbf{1}$ a malicious **Bob** would need to distinguish \mathbf{QR}_N from the other elements of \mathbf{Z}_N^* ...

Plan



1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications

A construction based on discrete log



check if p is prime and
 g and s are in QR_p
select a random
 $y \leftarrow \mathbb{Z}_{(p-1)/2}$
Let
 $x := \begin{cases} g^y & \text{if } b = 0 \\ s \cdot g^y & \text{if } b = 1 \end{cases}$

Select
 p – a random prime,
 g – a generator of QR_p
 s – a random element of QR_p



p, g, s

to commit to b send x

to open a commitment send y .

if $x = g^y$ output $b = 0$
if $x = s \cdot g^y$ output $b = 1$

This commitment scheme is **computationally binding**, assuming that the discrete log is hard in QR_p

Proof (intuition)

To be able to open the commitment in two ways, a **cheating Alice** needs to know y and y' such that there exists x such that

$$g^y = x = s \cdot g^{y'}$$

But this means that $g^{y-y'} = s$. So, she would know the discrete log of s .

This commitment scheme is unconditionally hiding

Why?

Because $\textcolor{red}{x}$ is just a random element of $\textcolor{red}{QR}_N$.

Plan

- 
1. Coin-flipping by telephone
 2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
 3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications

A construction based on PRGs [Naor'91]

$G: \{0, 1\}^L \rightarrow \{0, 1\}^{3L}$ a PRG

a bit b



select a random string
 $X \leftarrow \{0, 1\}^{3L}$



select a random string

$$Z \leftarrow \{0, 1\}^L$$

and let

$$Y := \begin{cases} G(Z) \oplus X & \text{if } b = 0 \\ G(Z) & \text{if } b = 1 \end{cases}$$

X

to commit to b send Y

to open a commitment send Z .

if $Y = G(Z) \oplus X$ output $b = 0$
if $Y = G(Z)$ output $b = 1$

This commitment scheme is unconditionally binding

Proof (intuition)

To be able to open the commitment in two ways, a **cheating Alice** needs to find \mathbf{Z} and \mathbf{Z}' such that there exists \mathbf{Y} such that:

$$\mathbf{G}(\mathbf{Z}) \oplus \mathbf{X} = \mathbf{Y} = \mathbf{G}(\mathbf{Z}')$$

This means that $\mathbf{G}(\mathbf{Z}) \oplus \mathbf{G}(\mathbf{Z}') = \mathbf{X}$.

How many \mathbf{X} 's have the property that

there exist \mathbf{Z} and \mathbf{Z}' such that $\mathbf{G}(\mathbf{Z}) \oplus \mathbf{G}(\mathbf{Z}') = \mathbf{X}$?

By the counting argument: at most $(2^L)^2 = 2^{2L}$.

Therefore, the probability that a **random $\mathbf{X} \in \{0, 1\}^{3L}$** has this property is at most

$$\frac{2^{2L}}{2^{3L}} = 2^{-L}.$$

QED

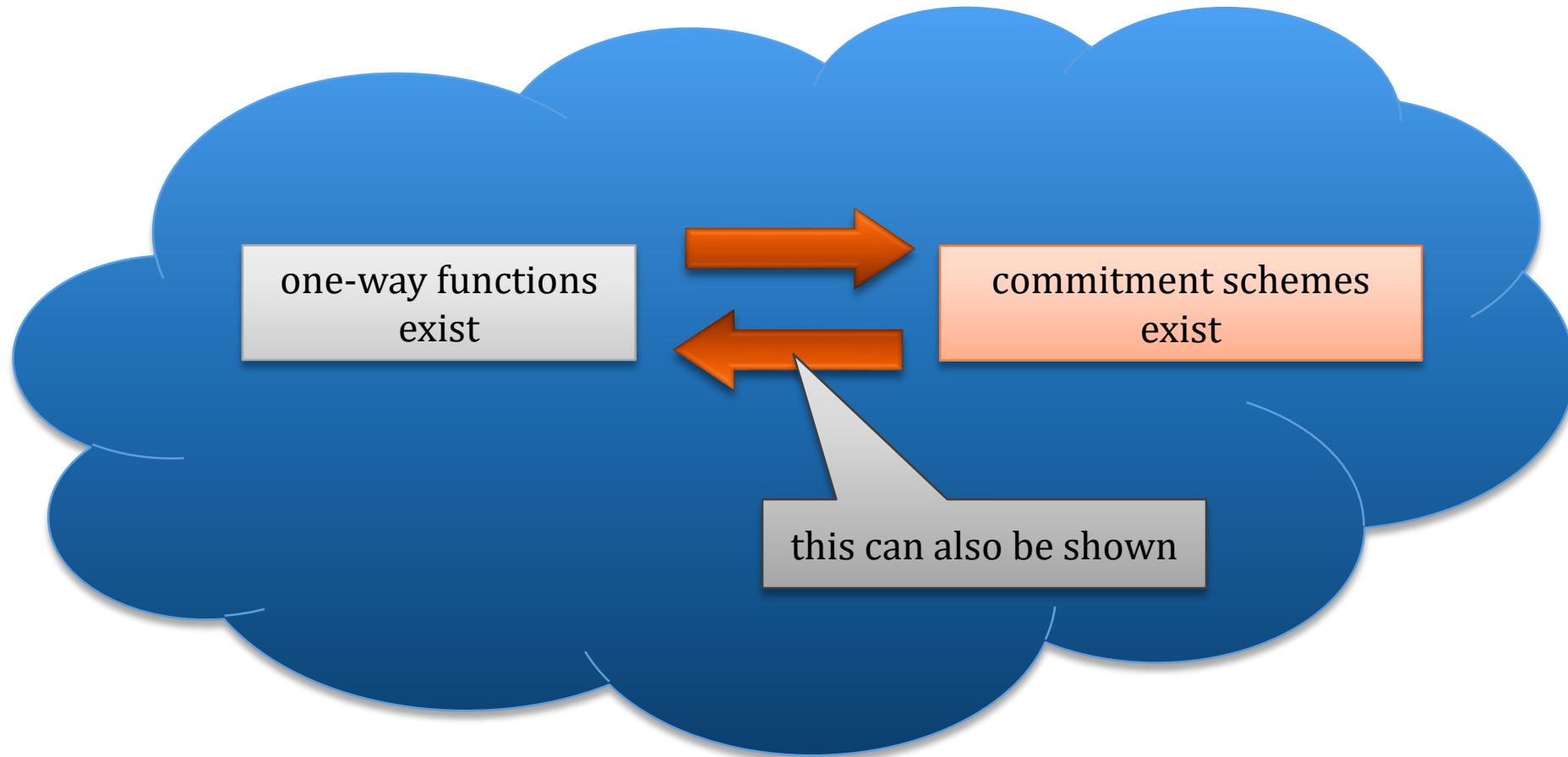
This commitment scheme is computationally hiding, assuming \mathbf{G} is a secure **PRG**

Why?

Obviously, if, instead of $\mathbf{G}(\mathbf{Z})$ Alice uses a completely random string \mathbf{R} , then the scheme is secure against a **cheating Bob**.

If a scheme behaved differently with \mathbf{R} and with $\mathbf{G}(\mathbf{Z})$, then a **cheating Bob** could be used as a distinguisher for \mathbf{G} .

Moral



Commitment schemes are a part of **Minicrypt!**

String commitment

How to commit to a longer string $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_n)$?

Just commit to every \mathbf{x}_i separately.

To open the commitment, open each commitment to \mathbf{x}_i .

(**Bob** accepts only if all the openings were **ok**)

Plan

1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications

Zero-knowledge (ZK)

We will now talk about the **zero-knowledge proofs**.

Informally:

A proof of some statement φ is **zero-knowledge**, if it doesn't reveal any information (besides the fact that φ holds).

Introduced in:

[**Shafi Goldwasser, Silvio Micali, Charles Rackoff: The Knowledge Complexity of Interactive Proof-Systems, STOC 1985, SIAM J. Comput. 1989**]



S. Goldwasser

Turing Award
in 2012



S. Micali



C. Rackoff

A motivating example: public-key identification (see: the last lecture)

(Enc, Dec) – a **public key** encryption scheme

(pk, sk) – a **(public key, private key)**
pair of **Alice**



pk

Everybody the knows pk can
verify the identity of **Alice**

Take a random
message m

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



$m' := \text{Dec}(sk, c)$

Check if
 $m = m'$

Is it secure against actively cheating verifier?



can ask arbitrary queries



So is it secure?

(we didn't define security, so this is just an informal question)

To impersonate **Alice** one needs to be able to decrypt c without the knowledge of m .

What does the verifier learn about sk ?

If the verifier follows the protocol – he doesn't learn anything that he didn't know before (he already knows m).

But what if the verifier is malicious?

Alice acts as a decryption oracle!

(so the verifier learns something that he didn't know)

is it a problem? – depends on the application

A question

Is it possible to design a protocol where

- a verifier **learns nothing**,
- **besides of the fact that he is talking to Alice?**

A new variant of the protocol

(pk, sk) – a (public key, private key) pair of Alice



pk

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

Take a random message m



commit to $m' := \text{Dec}(sk, c)$

abort if
 $m \neq m'$

m

open the commitment to m'

check if
 $m = m'$

Can a malicious verifier learn something from this protocol?

Intuition:

No, because he
doesn't learn m'

(he already knows m').

Can this be proven formally?

Yes!

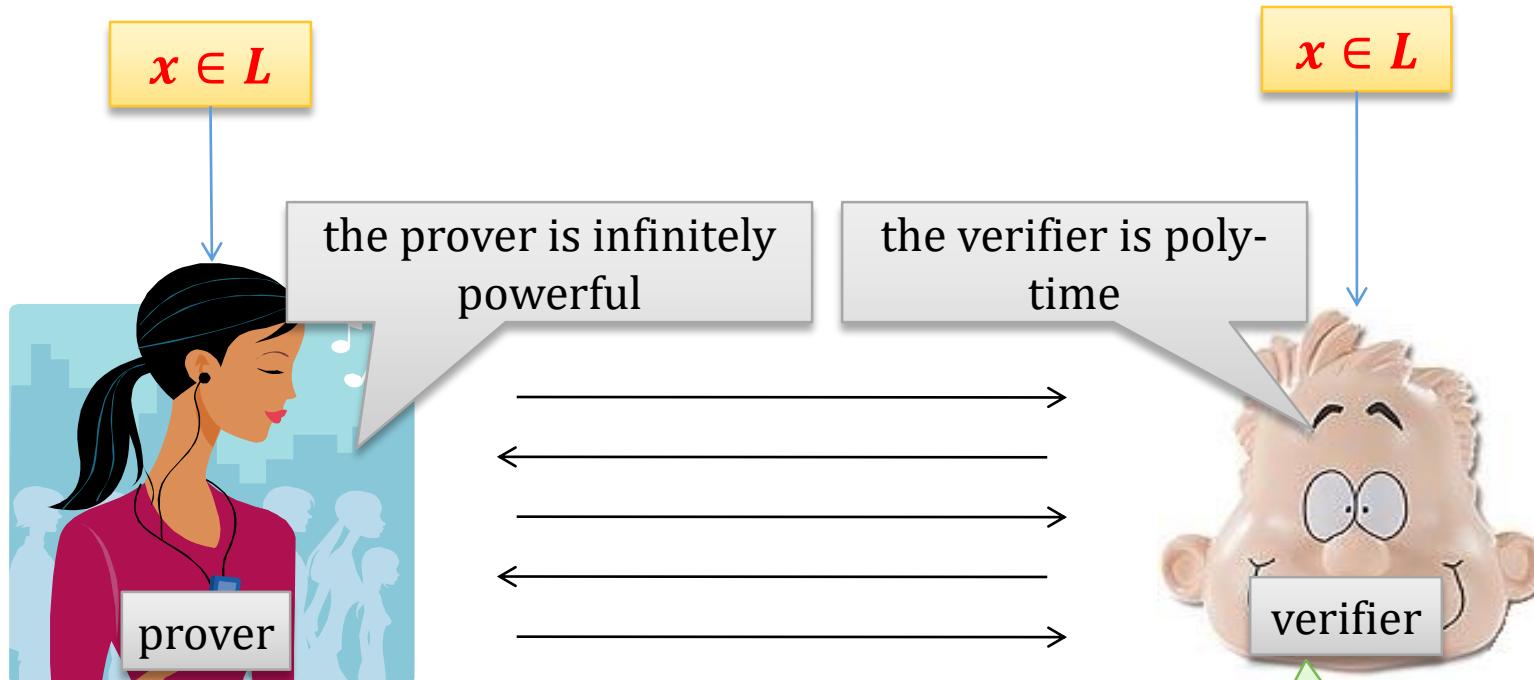
But we first need to

**define what it means that
“the verifier learns nothing”.**

This will lead us to the concept of
zero knowledge

The general picture

L – some language (usually not in P)



Two main properties:

1. soundness
2. zero-knowledge

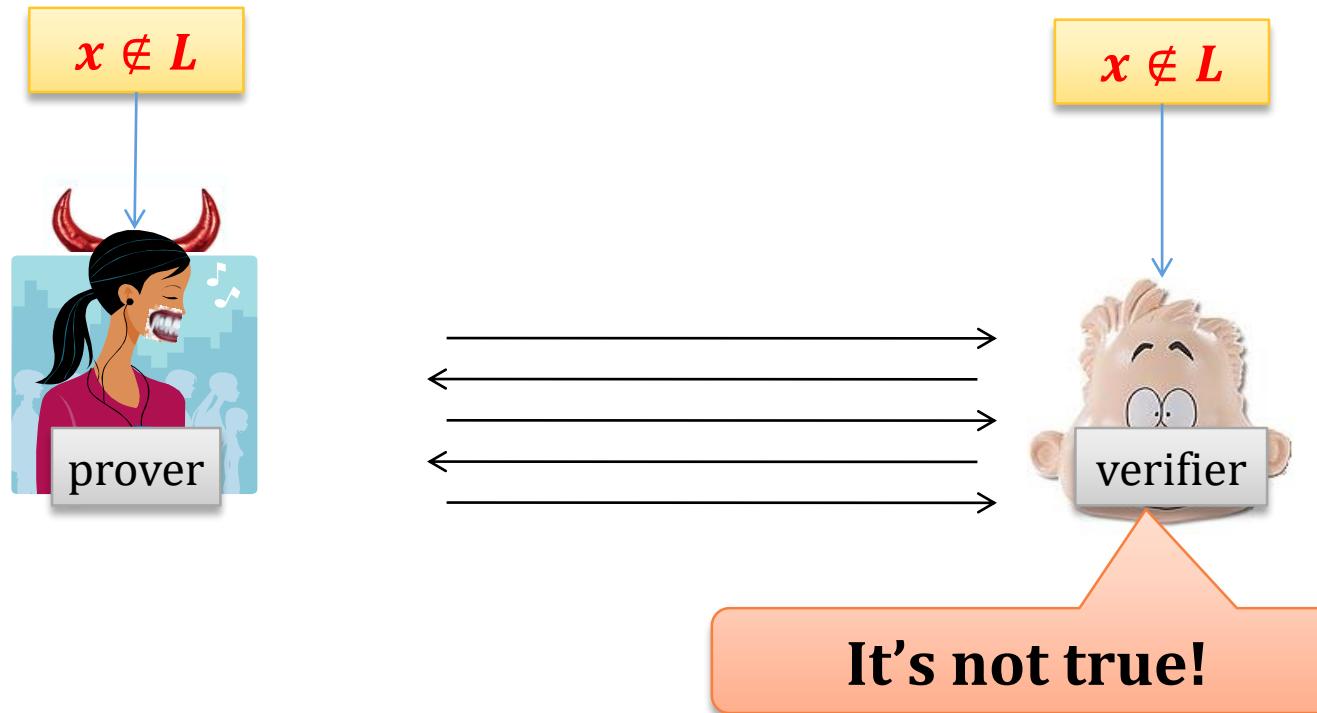
Soundness - informally

A cheating prover cannot convince the verifier that

$$x \in L$$

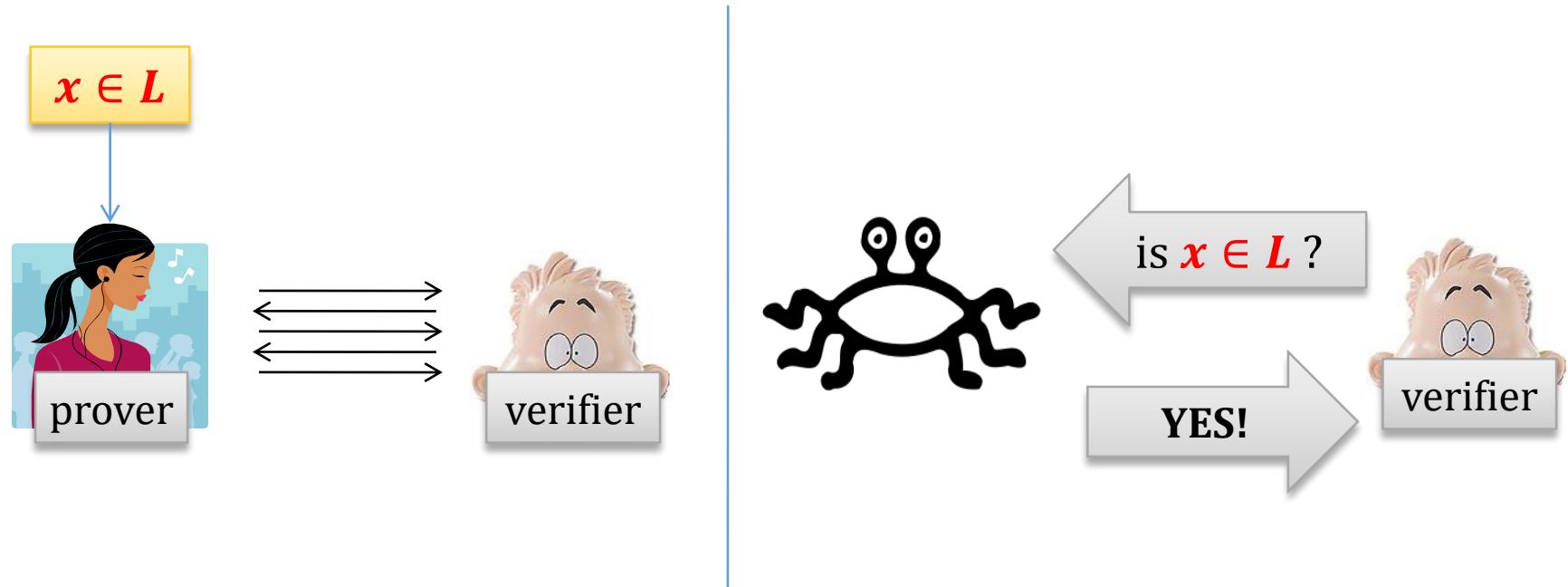
if it is **not** true

(negligible error probability is allowed)



Zero Knowledge

The only thing that the verifier should learn is that $x \in L$

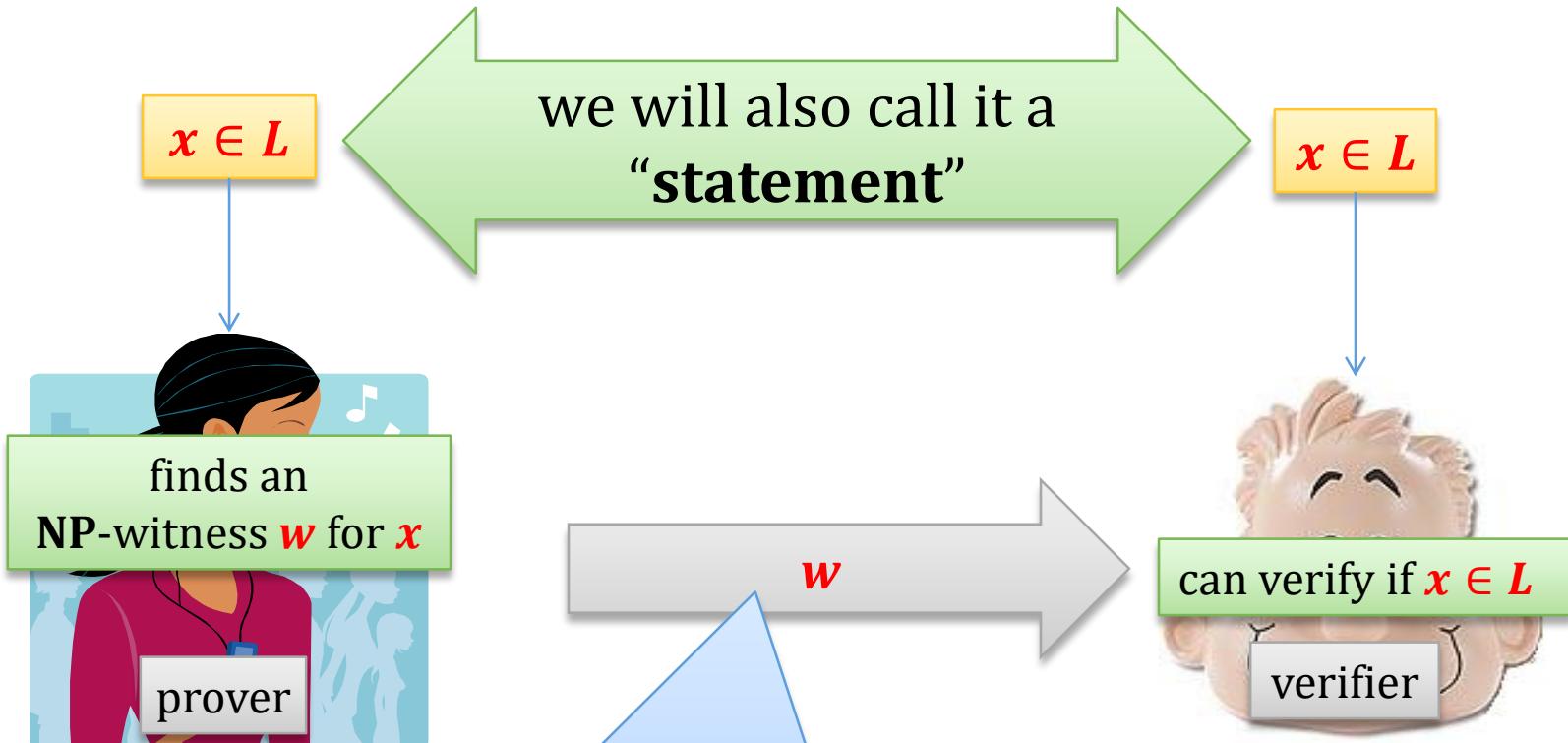


This should hold even if the verifier doesn't follow the protocol.

(again: we allow some negligible error)

An example of a protocol that is **not** Zero Knowledge

L – some NP-complete language



Why it is **not ZK**?
Because the verifier learned **w**

Notation

Suppose we are given a protocol consisting of two randomized machines P and V .

Suppose P and V take some common input x , and then V outputs **yes** or **no**.

We say that (P, V) **accepts** x if V outputs **yes**. Otherwise we say that it **rejects** x .

View(P, V, x) – a random variable denoting the “view of V ”, i.e.:

1. the random input of V and the input x ,
2. the **transcript of the communication**.

Zero-knowledge proofs

A pair (P, V) is a **zero-knowledge proof system** for L if it satisfies the following conditions:

- P has an infinite computing power and V is poly-time.
- Completeness: If $x \in L$, then the probability that (P, V) rejects x is negligible in the length of x .
- Soundness: If $x \notin L$ then for any prover P^* , the probability that (P^*, V) accepts x is negligible in the length of x .
- Zero-Knowledge: “a cheating V should not learn anything besides of the fact that $x \in L$ ”

How to define it formally?

“a cheating V should not learn anything besides of the fact that $x \in L$ ”

“What a cheating V can learn can be simulated without interacting with P ”

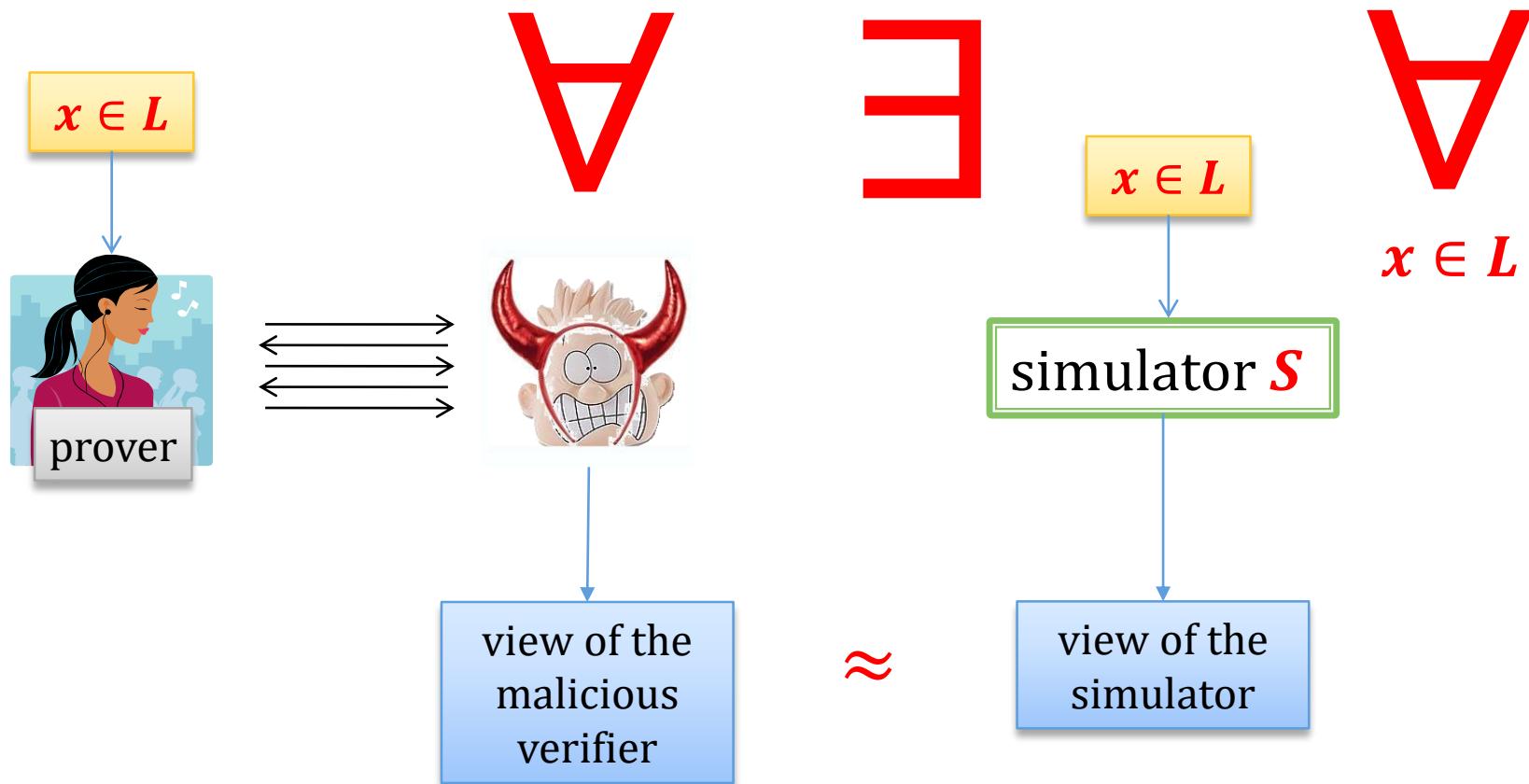
Definition (main idea)

For every (even malicious) poly-time V^* there exists an (expected) poly-time machine S such that

$\{\text{View}(P, V^*, x)\}_{x \in L}$ is
“*indistinguishable from*” $\{S(x)\}_{x \in L}$

we will formalize it in a moment

The idea of simulation



Indistinguishability

Let $\alpha = \{A(x)\}_{x \in L}$ and $\beta = \{B(x)\}_{x \in L}$ be two **sets of distributions**.

α and β are **computationally indistinguishable** if for every poly-time D there exists a negligible function ε such that for every $x \in L$

$$|P(D(x, A(x)) = 1) - P(D(x, B(x)) = 1)| \leq \varepsilon(|x|) \quad (*)$$

α and β are **statistically indistinguishable** if $(*)$ holds also for infinitely powerful D .

α and β are **perfectly indistinguishable** if $(*)$ holds also for infinitely powerful D , and $\varepsilon = 0$.

“a cheating V should not learn anything besides of the fact that $x \in L$ ”

Definition (a bit more formally)

For every (even malicious) poly-time V^* there exists an (expected) poly-time machine S such that

$$\{\text{View}(P, V^*, x)\}_{x \in L}$$

is computationally indistinguishable from $\{S(x)\}_{x \in L}$

This is a definition of a **computational zero-knowledge**.

By changing the “**computational** indistinguishability” into

- “**statistical** indistinguishability” we get a **statistical zero-knowledge**
- “**perfect** indistinguishability” we get a **perfect zero-knowledge**

Plan

1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications



Graph isomorphism

A **graph** is a pair (V, E) , where E is a binary symmetric relation on V .

A **graph isomorphism between (V, E) and (V', E')** is a function:

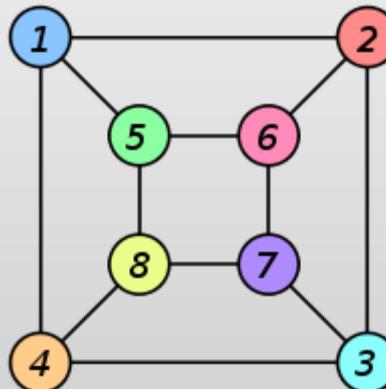
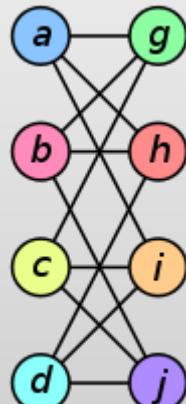
$$\varphi : V \rightarrow V'$$

such that

$$(e_1, e_2) \in E \text{ iff } (\varphi(e_1), \varphi(e_2)) \in E'$$

Graphs G and H are **isomorphic** if there exists an isomorphism between them.

Example



isomorphism:

$$\begin{aligned}f(a) &= 1 \\f(b) &= 6 \\f(c) &= 8 \\f(d) &= 3 \\f(g) &= 5 \\f(h) &= 2 \\f(i) &= 4 \\f(j) &= 7\end{aligned}$$

Hardness of graph isomorphism

No poly-time algorithm for the graph isomorphism problem is known.

Without loss of generality we will consider only isomorphism between (V, E) and (V', E') , where

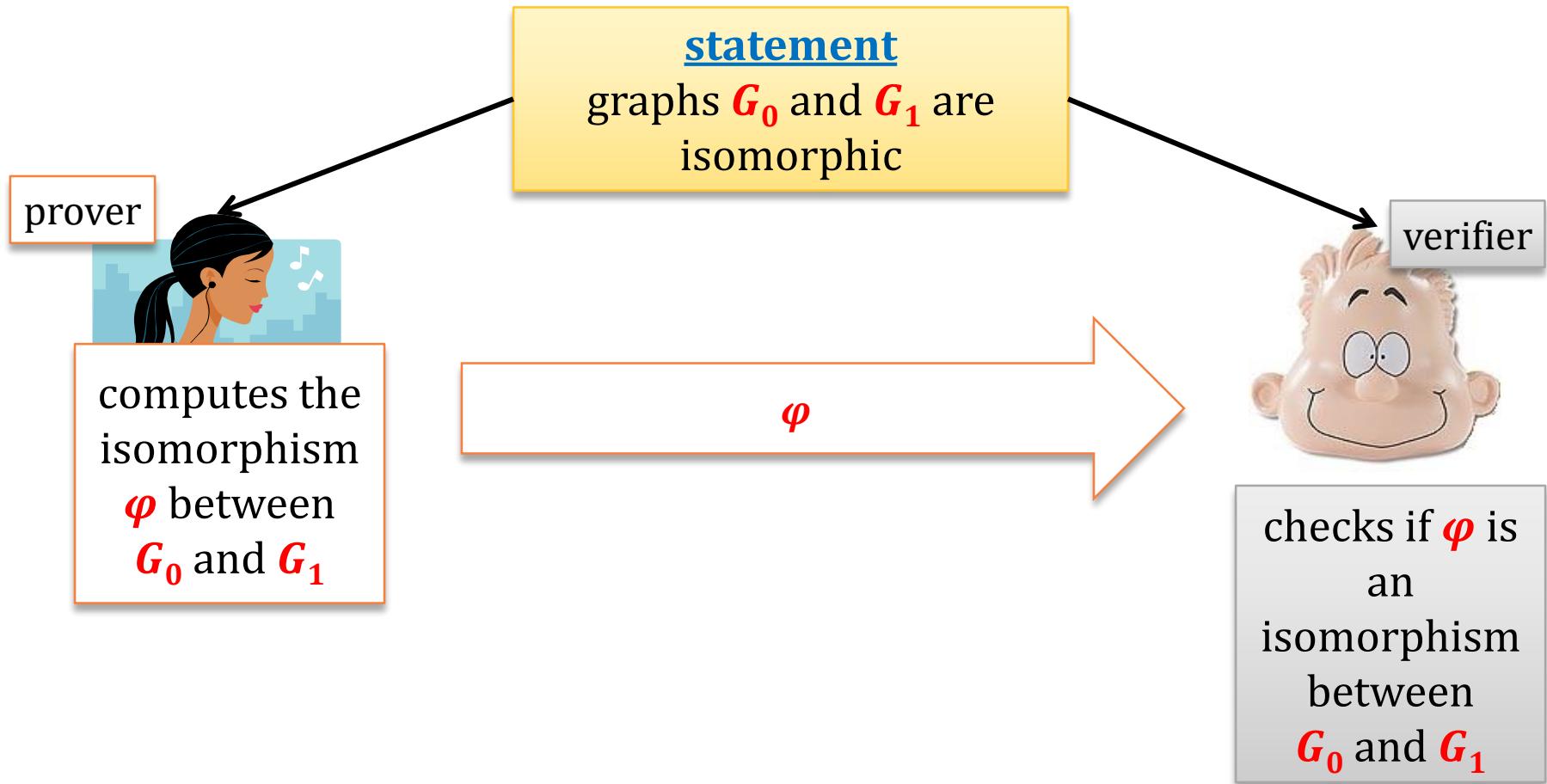
$$V = V' = \{1, \dots, n\} \text{ (for some } n\text{)}.$$

That is, a bijection:

$$\varphi : V \rightarrow V'$$

is a permutation of the set $\{1, \dots, n\}$.

A zero knowledge proof of graph isomorphism – a wrong solution



Notation

If $\mathbf{G} = (V, E)$ is a **graph**, and

$\pi: V \rightarrow V$ is a **permutation**

then by $\pi(\mathbf{G})$ we mean a graph

$$\mathbf{G}' = (V', E')$$

where

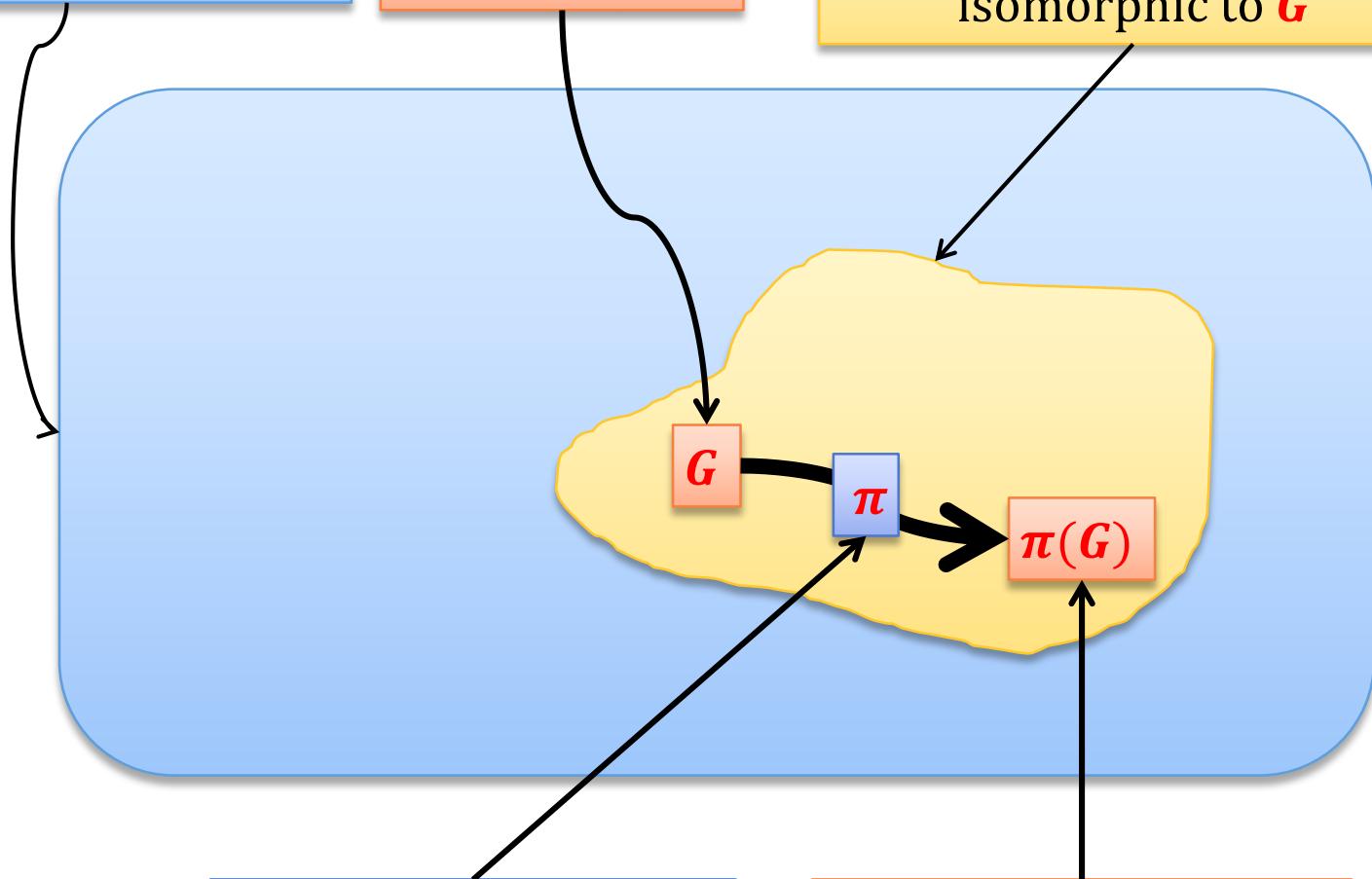
$$(a, b) \in E \text{ iff } (\pi(a), \pi(b)) \in E'$$

A fact

a set of all graphs with vertices in some set V

some graph G

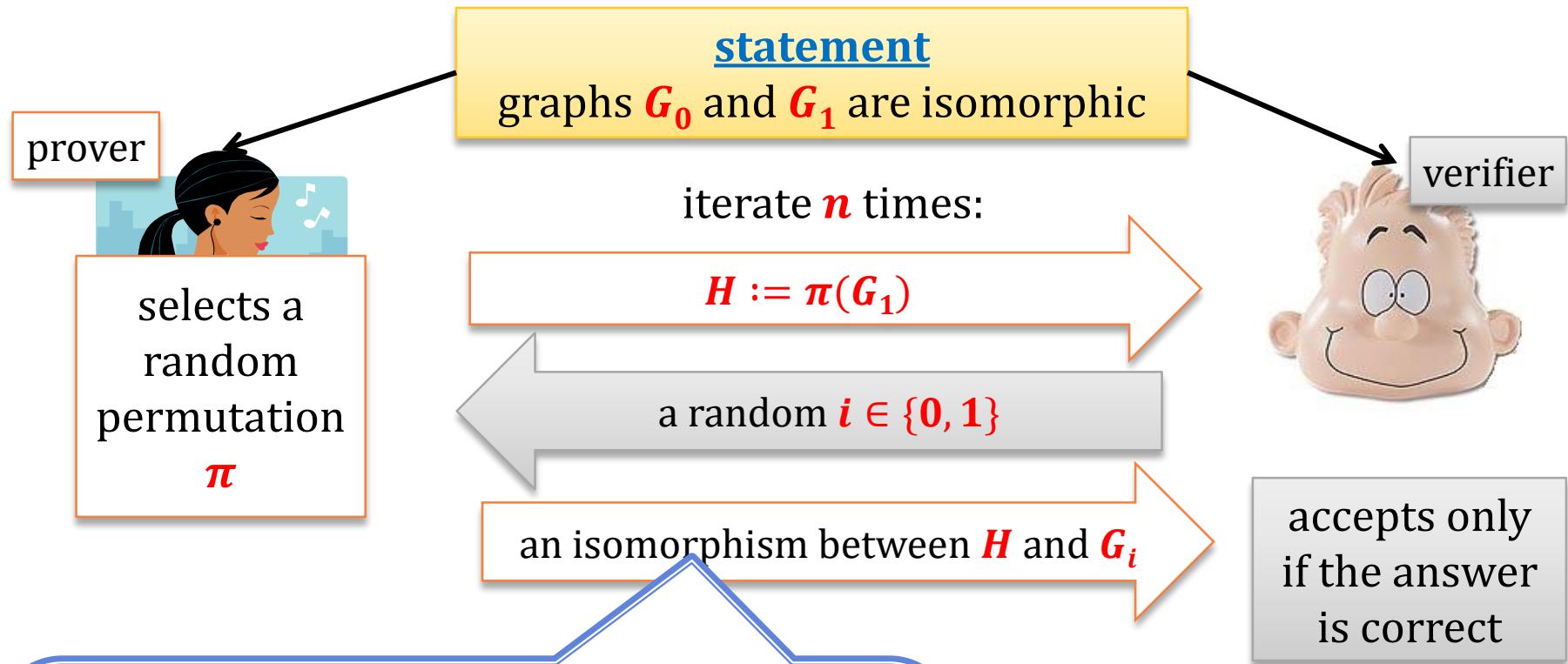
Γ - a class of all graphs isomorphic to G



If π is a random permutation

then $\pi(G)$ is a random element of Γ

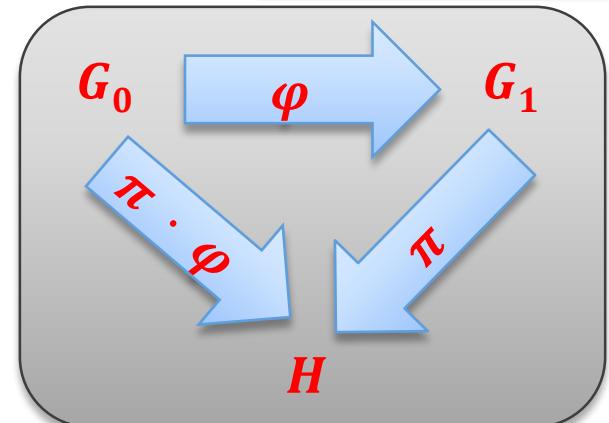
A zero knowledge proof of graph isomorphism



Note:

Prover does not need to be infinitely powerful, if he knows the isomorphism isomorphism φ between G_0 and G_1 .

- if $i = 1$ then he just sends π
- if $i = 0$ then he sends $\pi \cdot \varphi$

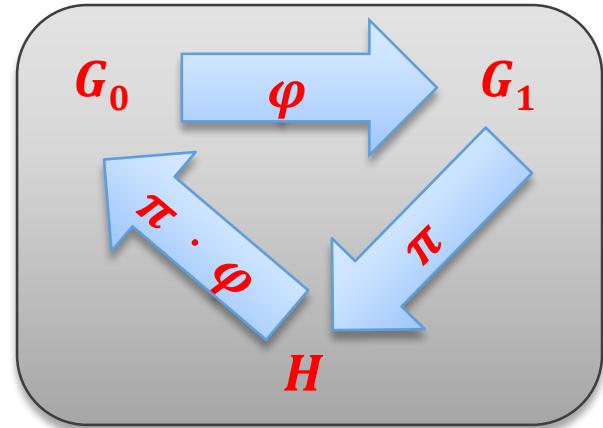


Why is this a zero-knowledge proof system?

- **Completeness:** trivial

- **Soundness:**

Suppose G_0 and G_1 are **not** isomorphic



Then, **at least one of the following** has to hold:

- G_0 and H are **not** isomorphic
- H and G_1 are **not** isomorphic



probability that a verifier rejects is at least **1/2**.

Since the protocol is repeated n times, the probability that the verifier rejects is at least $1 - \left(\frac{1}{2}\right)^n$.

Setting $n := |G_0| + |G_1|$ we are done!

Zero-knowledge?

Intuitively, the zero-knowledge property comes from the fact that:

The only thing that verifier learns is:

- a **permutation** between H and G_0 or G_1
where
- graph H is random graph isomorphic to G_0
(and isomorphic to G_1).

(In fact: we can show that this is a **perfect zero knowledge proof system**.)

More formally

For every poly-time



malicious
verifier V^*

there exists an (expected) poly-time

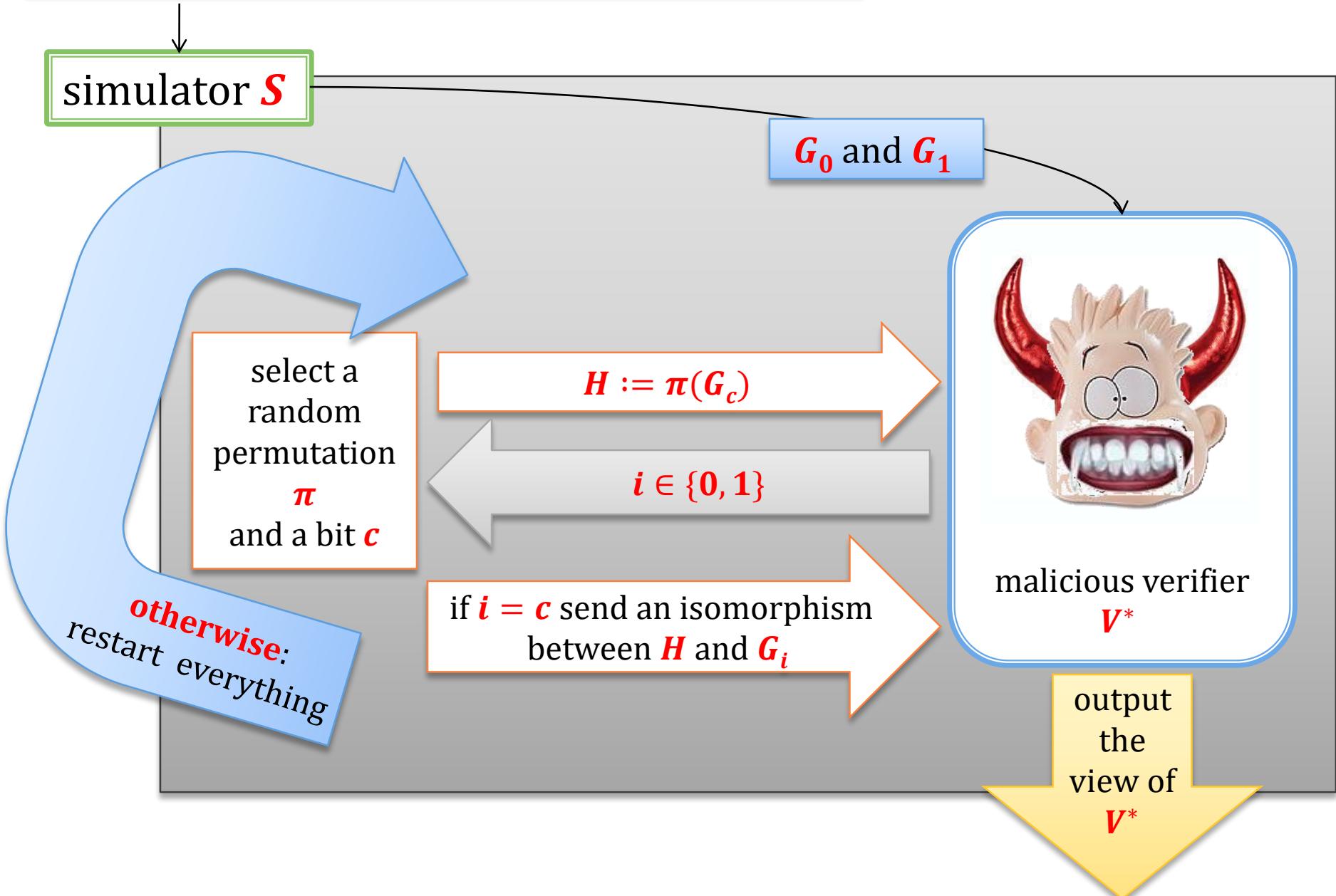
simulator S

such that

$$\{\mathbf{View}(P, V^*, x)\}_{x \in L}$$

is perfectly indistinguishable from $\{S(x)\}_{x \in L}$

statement: graphs G_0 and G_1 are isomorphic



The running time

First, observe, that the distribution of H doesn't depend on c (since it is uniform in the class of graphs isomorphic with G_0 and G_1)

Therefore the probability that S needs to restart V^* is equal to $\mathbf{1/2}$.

So the expected number of restarts is $\mathbf{2}$.

Therefore, the running time is (expected) polynomial time.

Indistinguishability of the distributions

Suppose $i = c$, and hence we didn't restart.

In this case, the simulator simply simulated “perfectly” execution of V^* against P .

uniform in the class of graphs isomorphic with G_0 and G_1

$$H := \pi(G_i)$$

a random $i \in \{0, 1\}$

an isomorphism between H and G_i

QED

Plan

1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications



What is provable in NP?

Theorem [Goldreich, Micali, Wigderson, 1986]

Assume that the **one-way functions exist**.

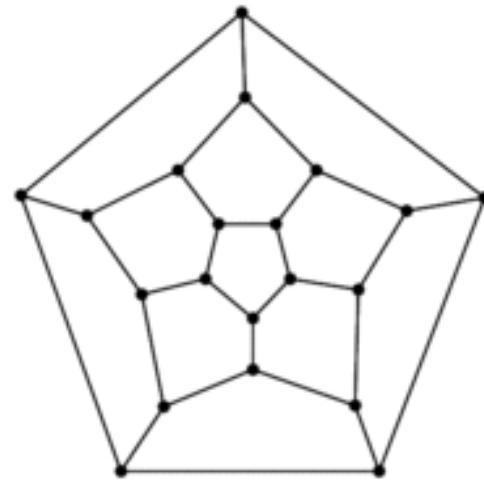
Then, every language $L \in \text{NP}$ has a computational zero-knowledge proof system.

How to prove it?

It is enough to show it for one **NP-complete** problem!

Take the following NP-complete problem: **Hamiltonian graphs**

Example of a **Hamiltonian cycle**:



Hamiltonian graph – a graph that has a **Hamiltonian cycle**

$$L := \{G : G \text{ is Hamiltonian}\}$$

How to construct a ZK proof that a graph G is Hamiltonian?

Of course:

sending the Hamiltonian cycle in a graph G to the verifier doesn't work.

H is Hamiltonian
iff
 G is Hamiltonian

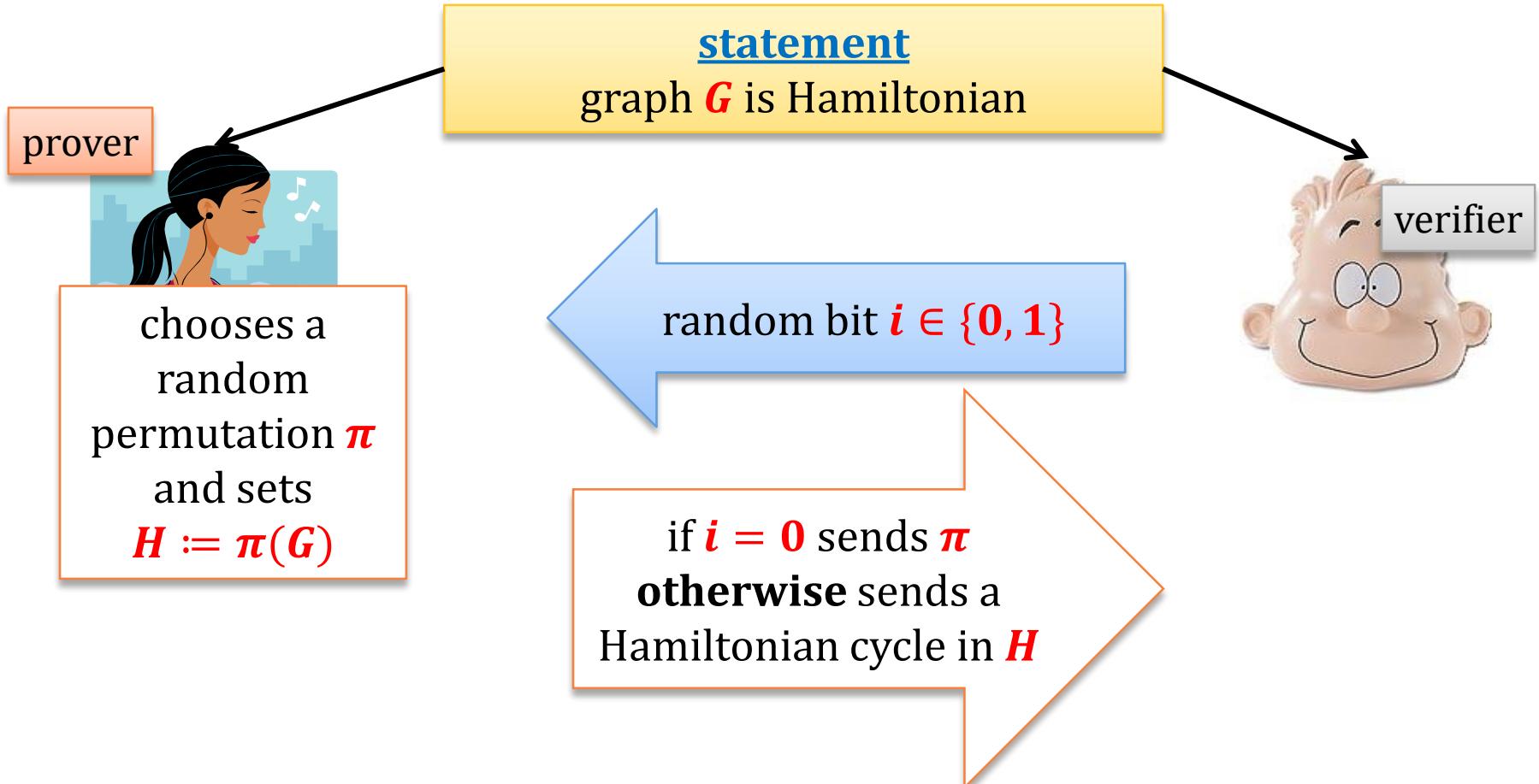
Idea:

We permute the graph G randomly – let H be the permuted graph.

Then we prove that

1. H is Hamiltonian,
2. H is a permutation of G .

The first idea:



Problem: Prover can choose his response depending on i .

Solution: use commitments

Remember, that we assumed that the one-way functions exist, so we are “allowed” to use commitments!

Assume the vertices of the graph are natural numbers $\{1, \dots, n\}$.

How to commit to a permutation of a graph?

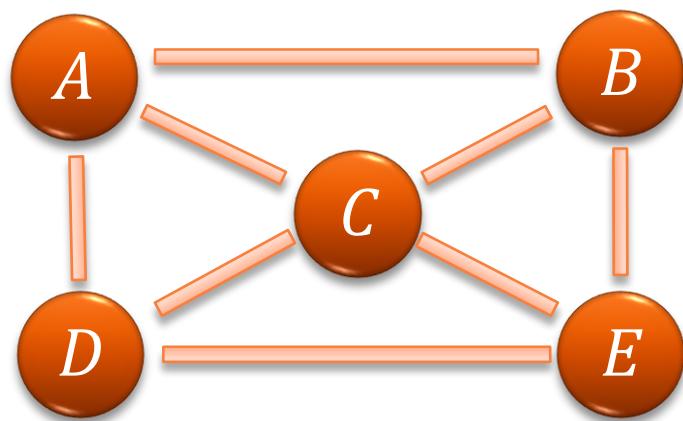
Represent it as a string

How to commit to a graph?

Represent it as an **adjacency matrix**,
and commit to each bit in the matrix separately.

Example

graph H :



$$M = \{M_{ij}\}_{i,j \in \{A, \dots, B\}}$$

| | A | B | C | D | E |
|----------|----------|----------|----------|----------|----------|
| A | 0 | 1 | 1 | 1 | 0 |
| B | 1 | 0 | 1 | 0 | 1 |
| C | 1 | 1 | 0 | 1 | 1 |
| D | 1 | 0 | 1 | 0 | 1 |
| E | 0 | 1 | 1 | 1 | 0 |

to commit to H :

for $i = A, \dots, E$

for $j = A, \dots, E$

Commit(M_{ij})

statement: graph G is Hamiltonian

prover



chooses a random permutation π and sets $H := \pi(G)$

iterate n times:

verifier



commit to π

commit to every bit M_{ij}

random bit $i \in \{0, 1\}$

if $i = 0$:

open all the commitments

check if everything was done correctly

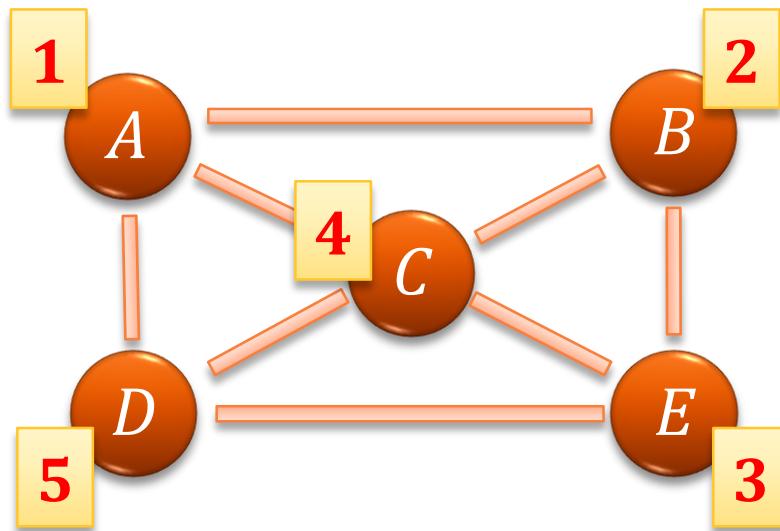
if $i = 1$:

open only the commitments to the edges that represent a Hamiltonian cycle in H

check if it is indeed a Hamiltonian cycle

verifier accepts only if all commitments were open correctly and all checks are ok

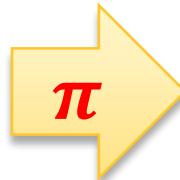
Example of a Hamiltonian graph



| | | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 4 | 5 | 3 |
| A | 0 | 1 | 1 | 1 | 0 |
| B | 1 | 0 | 1 | 0 | 1 |
| C | 1 | 1 | 0 | 1 | 1 |
| D | 1 | 0 | 1 | 0 | 1 |
| E | 0 | 1 | 1 | 1 | 0 |

Example of a “permuted graph”

| | <i>A</i> | <i>B</i> | <i>C</i> | <i>D</i> | <i>E</i> |
|----------|----------|----------|----------|----------|----------|
| <i>A</i> | 0 | 1 | 1 | 1 | 0 |
| <i>B</i> | 1 | 0 | 1 | 0 | 1 |
| <i>C</i> | 1 | 1 | 0 | 1 | 1 |
| <i>D</i> | 1 | 0 | 1 | 0 | 1 |
| <i>E</i> | 0 | 1 | 1 | 1 | 0 |



| | $\pi(B)$ | $\pi(C)$ | $\pi(D)$ | $\pi(E)$ | $\pi(A)$ |
|----------|----------|----------|----------|----------|----------|
| $\pi(B)$ | 0 | 1 | 0 | 1 | 1 |
| $\pi(C)$ | 1 | 0 | 1 | 1 | 1 |
| $\pi(D)$ | 0 | 1 | 0 | 1 | 1 |
| $\pi(E)$ | 1 | 1 | 1 | 0 | 0 |
| $\pi(A)$ | 1 | 1 | 1 | 0 | 0 |

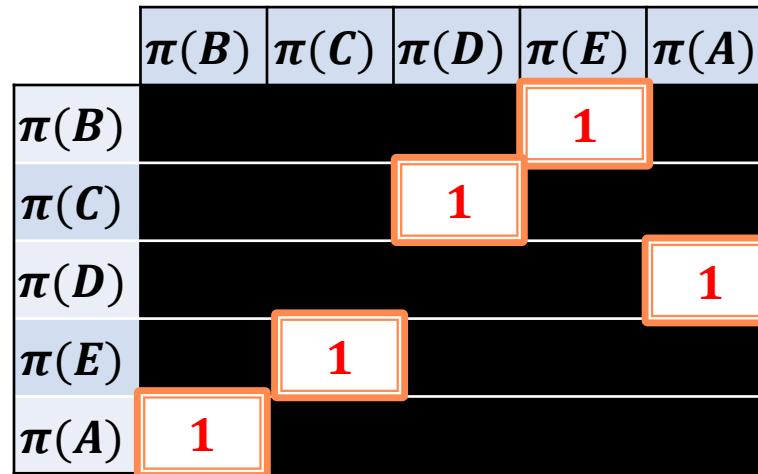
$$\pi(A) = E, \pi(B) = A, \pi(C) = B, \pi(D) = C, \pi(E) = D$$

Case 0:

open everything but **don't show the Hamiltonian cycle**

Case 1

Open **only** the Hamiltonian cycle



Why is it a ZK proof?

Completeness: trivial

Soundness: If G is not Hamiltonian, then either
 H is not Hamiltonian or π is not a permutation.

Therefore, to cheat with probability higher than $1/2$ the prover needs to **break the binding property of the commitment scheme**.

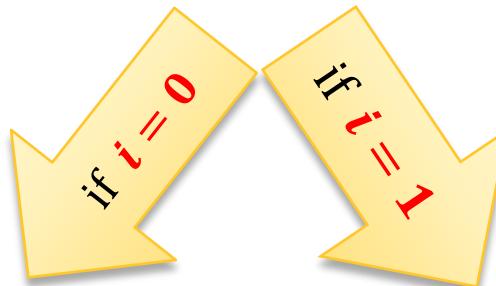
If we use the commitment scheme of **Naor**, this probability is **negligible**, even against an infinitely-powerful adversary

Since the protocol is repeated n times, the probability that the verifier rejects is at least

$$1 - \left(\frac{1}{2}\right)^n.$$

Zero-Knowledge - intuition

“a cheating V should not learn anything besides of the fact that $x \in L$ ”



P “opens everything”, so V just learns a randomly permuted graph G .

P “opens only the Hamiltonian cycle”, so V just learns a randomly permuted cycle of vertices

Note, that this gives us only **computational** indistinguishability. This is because the commitment scheme is only computationally binding.

Observation

The honest prover doesn't need to be infinitely powerful, if he receives the **NP**-witness as an additional input!

Corollary

“Everything that is provable is provable in Zero Knowledge!”

Plan

1. Coin-flipping by telephone
2. Commitment schemes
 1. definition
 2. construction based on QRA
 3. construction based on discrete log
 4. construction based on PRG
3. Zero-knowledge (ZK)
 1. motivation and definition
 2. ZK protocol for graph isomorphism
 3. ZK protocol for Hamiltonian cycles
 4. applications



Example

Suppose, **Alice** knows a signature σ of Bob on some document $m = (m_1 || m_2)$.

$$\sigma = \text{Sign}_{sk}(m)$$

She want to reveal the first part m_1 of m to **Carol**, and convince her that it was signed by **Bob**, while keeping m_2 and σ secret.

$$L = \{m_1 : \text{there exists } m_2 \text{ and } \sigma \text{ such that } \text{Vrfy}_{pk}(m_1 || m_2, \sigma) = \text{yes}\}$$

L is in **NP**. So (in principle) **Alice** can do it!

Another example

Alice has a document (signed by some public authority) saying:

“Alice was born on **DD-MM-YYYY**”.

She can now prove in zero-knowledge that she is at least **18** years old (without revealing her exact age)

There are many other examples!

For instance:

Alice can show that some message m was signed by Bob or by Carol,

without revealing which was the case.

etc...

Other applications of ZK

- a building block in some other protocols
- A recent application: Zcash – a fully anonymous cryptocurrency (deployed in 2016)
- zero-knowledge identification (e.g. a **Feige-Fiat-Shamir** protocol, based on quadratic residues)



Example

We show a zero-knowledge proof that some x is a quadratic residue modulo N .

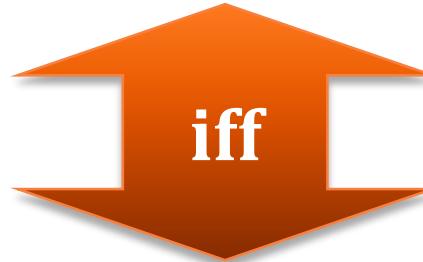
How does it work?

Similarly to the proof that two graphs are isomorphic!

Fact

For $a, b \in Z_N^+$ we have

$$a \cdot b \in QR_N$$



- $a \in QR_N$ and $b \in QR_N$
- or
- $a \notin QR_N$ or $b \notin QR_N$

Main idea

G_0 is isomorphic with H

H is isomorphic with G_1



G_0 is isomorphic with G_1

v is a QR

$v \cdot x$ is a QR



x is a QR

RSA modulus N ,
statement: $x \in \mathbb{Z}_N^+$ in QR_N

y such that
 $y^2 = x \pmod N$



chose a
random
 $u \leftarrow \mathbb{Z}_N^*$

iterate n times:



$$v := u^2 \pmod N$$

random bit $i \leftarrow \{0, 1\}$

$$w := u \cdot y^i \pmod N$$

$$\begin{aligned} &= u && \text{if } i = 0 \\ &= u \cdot y && \text{if } i = 1 \end{aligned}$$

accept if
 $w^2 = v \cdot x^i \pmod N$

y such that
 $y^2 = x \pmod{N}$

choose a random
 $u \leftarrow \mathbb{Z}_N^*$

$$v := u^2 \pmod{N}$$

random bit $i \leftarrow \{0, 1\}$

$$w := u \cdot y^i \pmod{N}$$

accept if
 $w^2 = v \cdot x^i \pmod{N}$

Why is this a zero-knowledge proof system?

1. Completeness:

$$\begin{aligned} w^2 &= (u \cdot y^i)^2 \\ &= u^2 \cdot (y^2)^i \\ &= v \cdot x^i \end{aligned}$$

y such that
 $y^2 = x \pmod{N}$

chose a random
 $u \leftarrow \mathbb{Z}_N^*$

$$v := u^2 \pmod{N}$$

random bit $i \leftarrow \{0, 1\}$

$$w := u \cdot y^i \pmod{N}$$

accept if
 $w^2 = v \cdot x^i \pmod{N}$

2. Soundness:

suppose that $x \notin \text{QR}_N$

Then

- if v is a QR_N then the cheating prover will be caught when $i = 1$ since we cannot have

$$\text{QR} \cdot \text{QNR} = \text{QR}$$

- if v is a QNR_N the cheating prover gets caught when $i = 0$.

So, the prover can cheat with probability at most $1/2$ (in each iteration of the protocol).

y such that
 $y^2 = x \pmod{N}$

chose a random
 $u \leftarrow \mathbb{Z}_N^*$

$$v := u^2 \pmod{N}$$

random bit $i \leftarrow \{0, 1\}$

$$w := u \cdot y^i \pmod{N}$$

accept if
 $w^2 = v \cdot x^i \pmod{N}$

3. Zero-knowledge (intuition)

The only information that the verifier gets is:

$$v := u^2$$

and

- $w := u$ if $i = 0$, or
- $w := u \cdot y$ if $i = 1$.

This obviously gives him
no information on x

This also gives him no information
on y , since y is “encrypted” with u

Observation

In fact, the prover demonstrated not only that x in QR_N ,
but also that **she knows the square root of x .**

This is called a **zero-knowledge proof of knowledge**.

It can be defined formally!

Zero-knowledge public-key identification

The protocol on the previous slides can be used as a simple **zero-knowledge public-key identification scheme**:

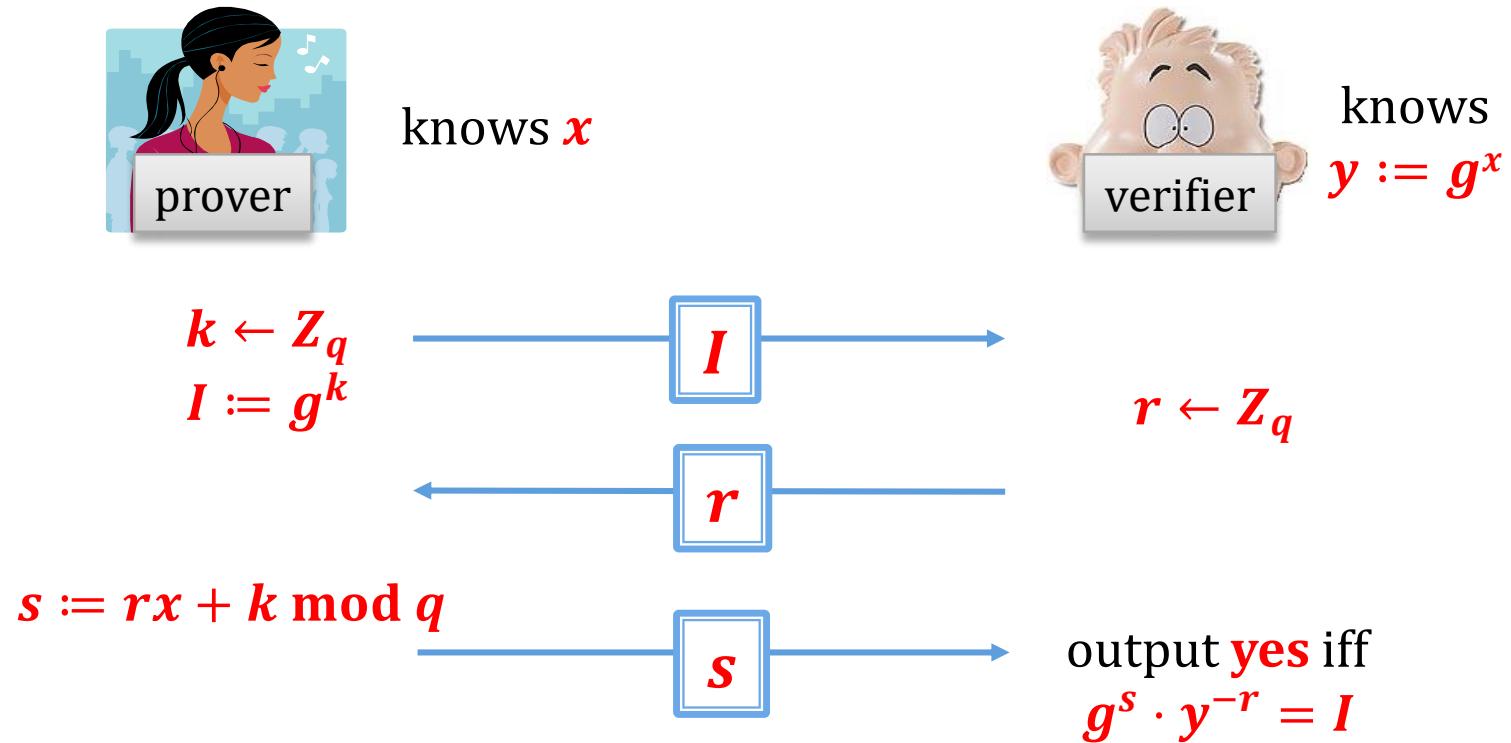
- public key: N, x
- private key: y such that $y^2 = x \bmod N$

Its extension is called a **Feige-Fiat-Shamir** protocol.

Is Shorr's protocol zero-knowledge proof of knowledge?

From last lecture:

G - group, $q = |G|$
 g - generator



The situation

We have proven that the transcripts
 (I, r, s)

do not reveal any information about prover's secret x .

This is a **weaker** property than **zero-knowledge** (because the **verifier does not choose r**).

It is called an **honest verifier zero-knowledge**.

Schnorr's protocol is **believed** to be also **zero knowledge**, but nobody can prove it (from standard assumptions).

©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.*