Zero-Knowledge Interactive Proofs

Michele Ciampi

THE UNIVERSITY of EDINBURGH

Two parties for a proof

- Merlin (prover) has unbounded resources
- Arthur (verifier) has limited resources

Theorem/statement **x**

π

The proof is efficient: **x** is an NP statement and π is its certificate/witness/proof

Graph Isomorphism

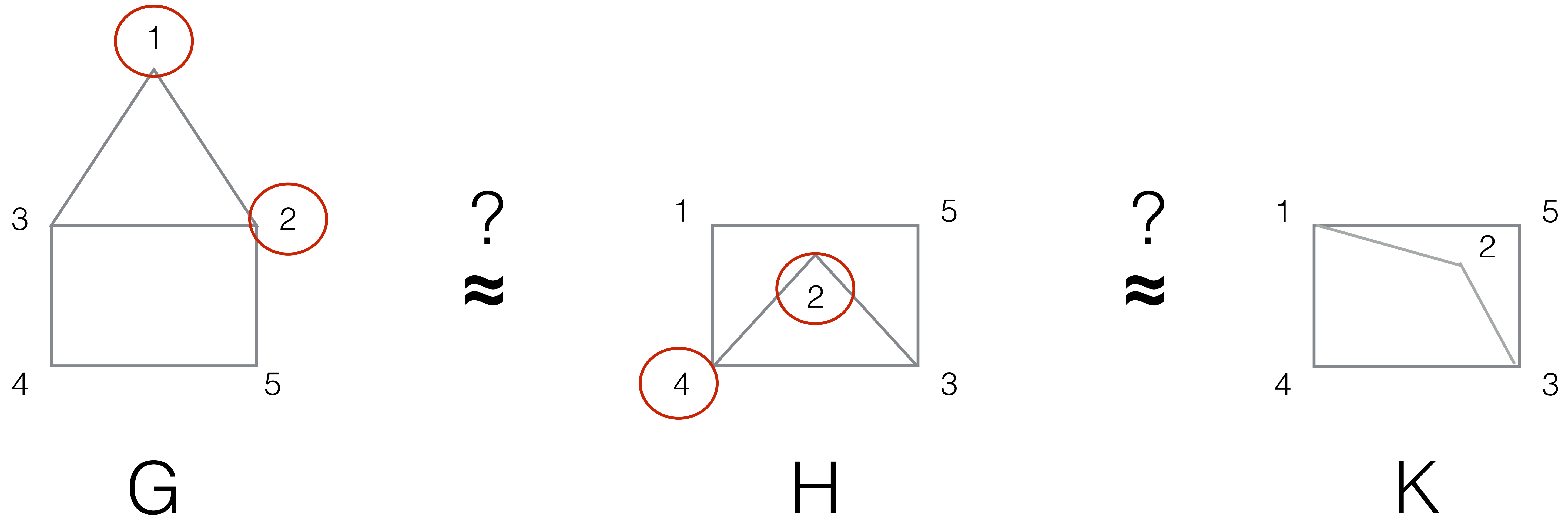
An isomorphism of graphs **G** and **H** is a bijection (permutation) π between the vertex sets of **G** and **H**

π: V(**G**) —> V(**H**)

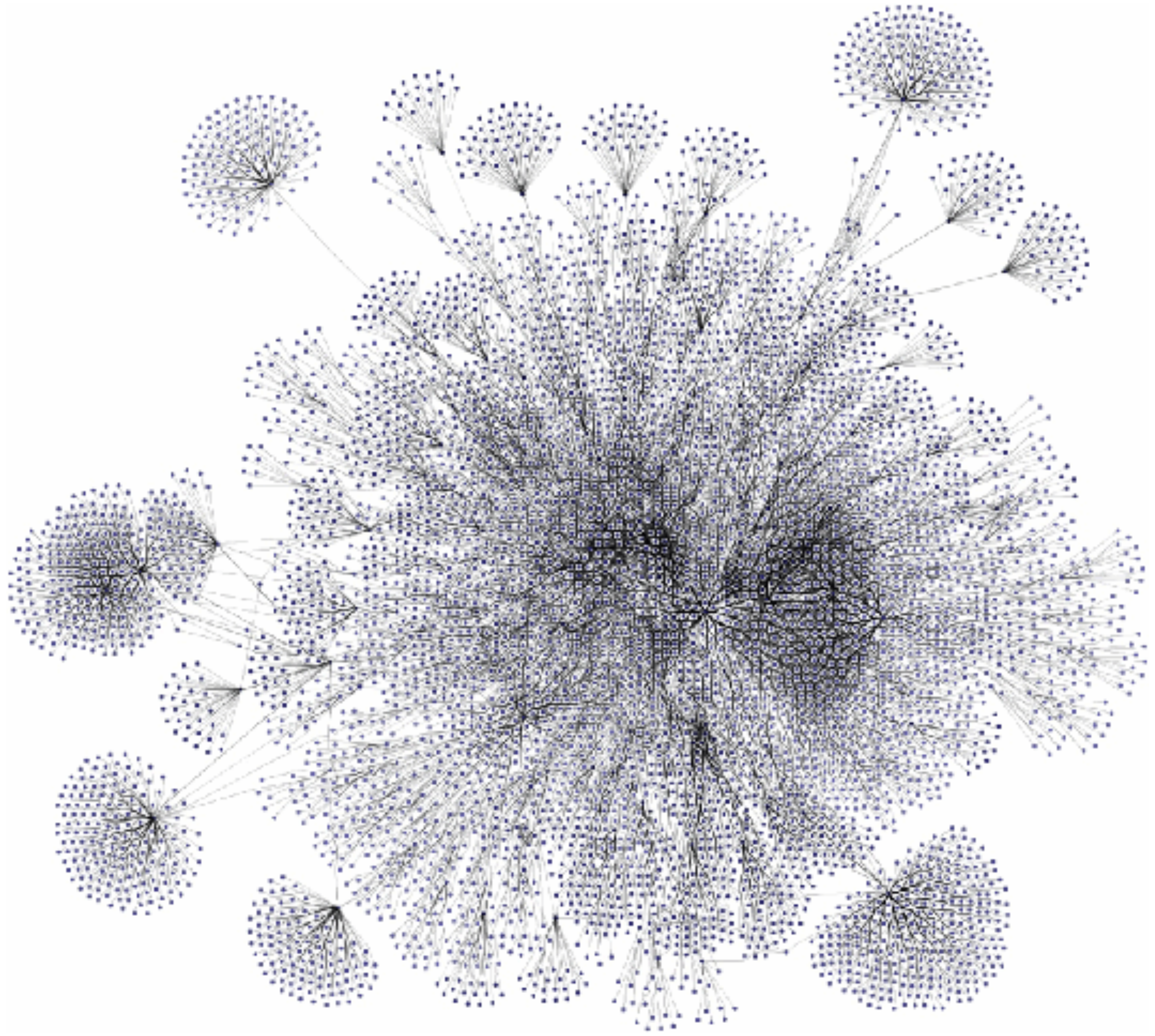
such that any two vertices u and v of **G** are adjacent in **G** if and only if π(u) and π(v) are adjacent in **H**.

π

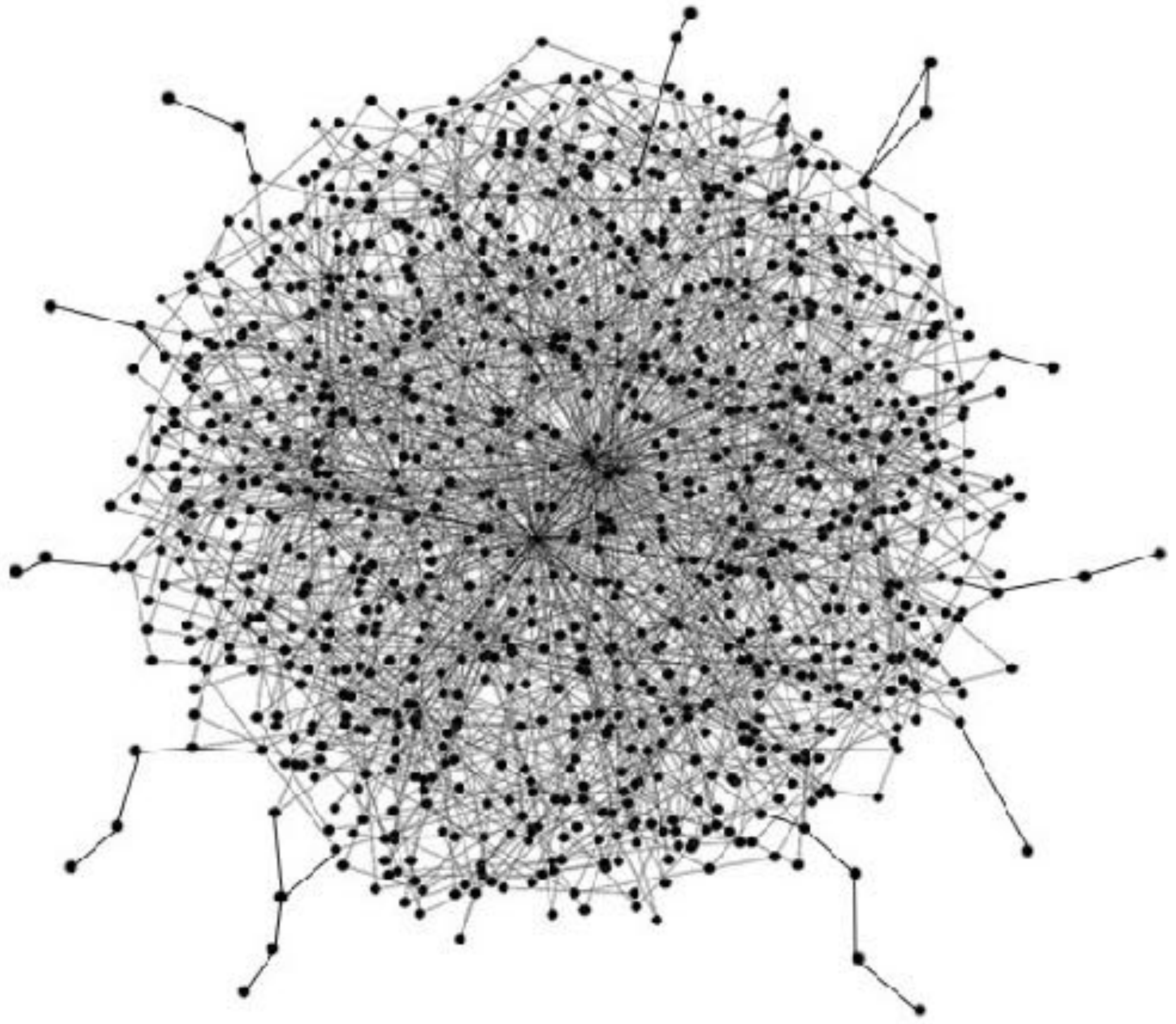
G	**H**
1	2
2	4
3	3
4	5
5	1



Graph Isomorphism



$\overset{?}{\approx}$



The problem belongs to NP

We do not know if it is in P: best known algorithm is quasi-polynomial time

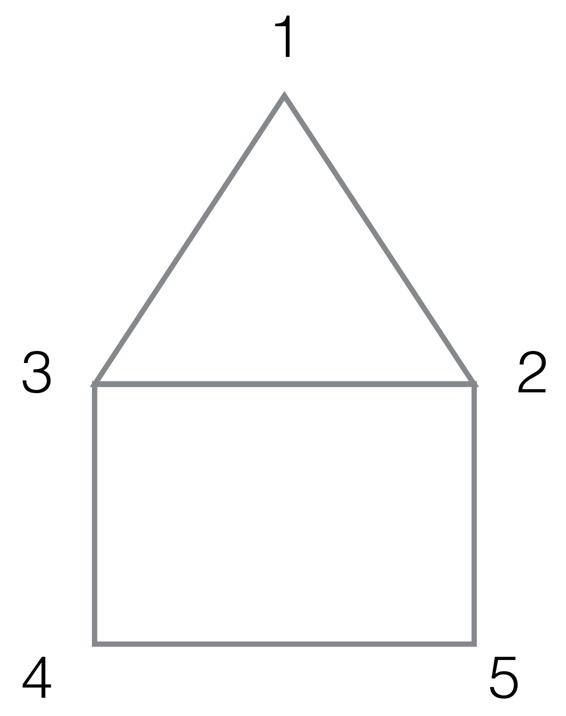
Graph Isomorphism

Witness

π

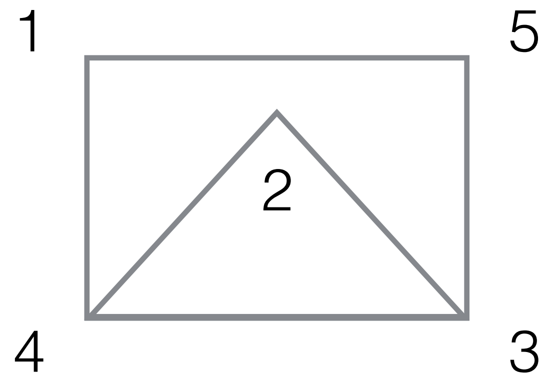
G	H
1	2
2	4
3	3
4	5
5	1

Thm



G

≈



H

π: 1—>2, 4—>1,…

Interactive Proofs

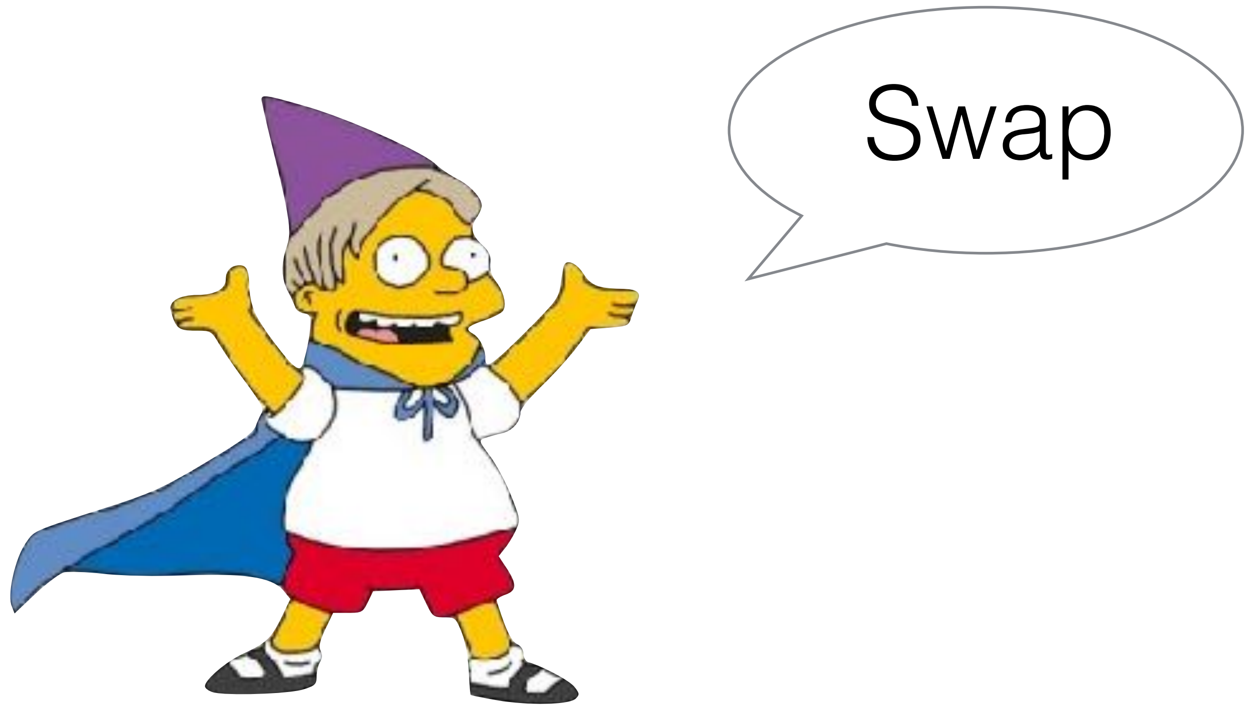
- Suppose now that I want to prove that two graphs are **not isomorphic** or that an equation has no solutions.
- Introduced by Goldwasser, Micali and Rackoff
  - A proof is described as a game between a *prover* and a *verifier*
  - The theorem is true if and only if the prover wins the game always.
  - If the theorem is false then the prover loses the game with 50% probability

Prover (Merlin)

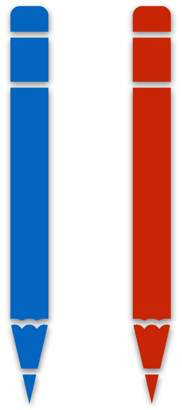
Verifier (Arthur)

Interactive Proofs

A simple example first



Prover



Verifier

Interactive Proofs

A simple example first

Prover

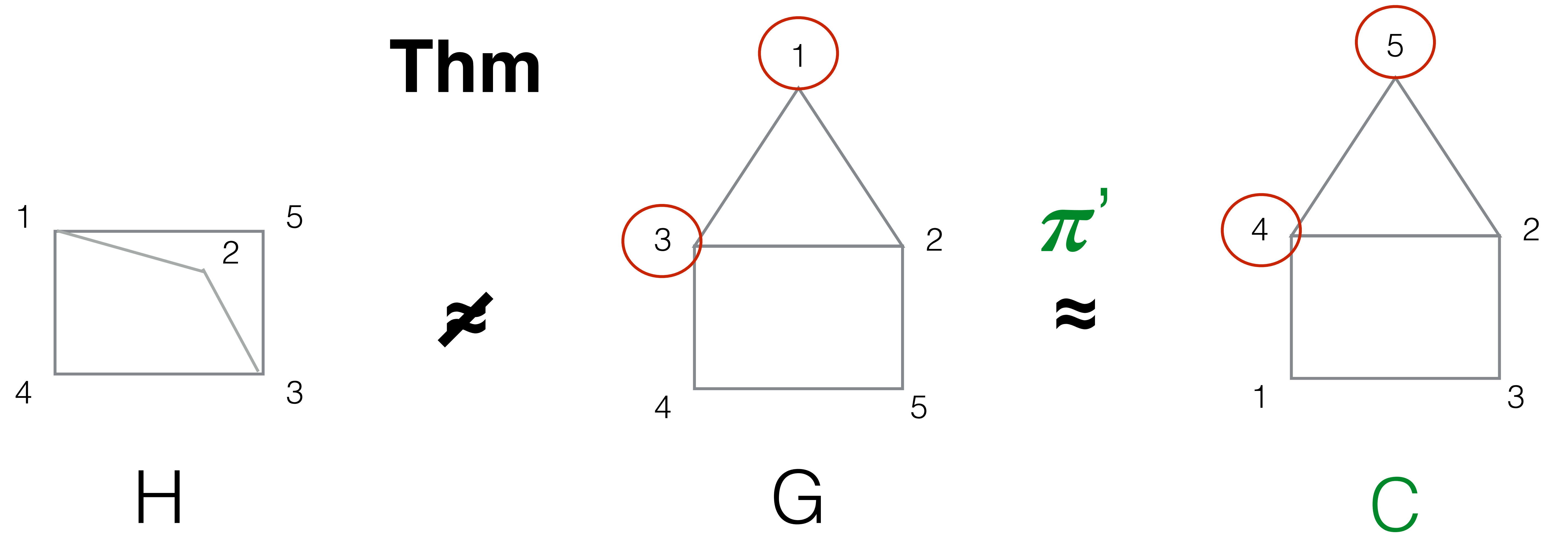
Verifier

If the pencils are both red, then the prover convinces the verifier with a 50% probability

We can repeat the proof many times to make this probability small

Graph Non-Isomorphism

Thm



π'

G	C
1	5
2	2
3	4
4	1
5	3

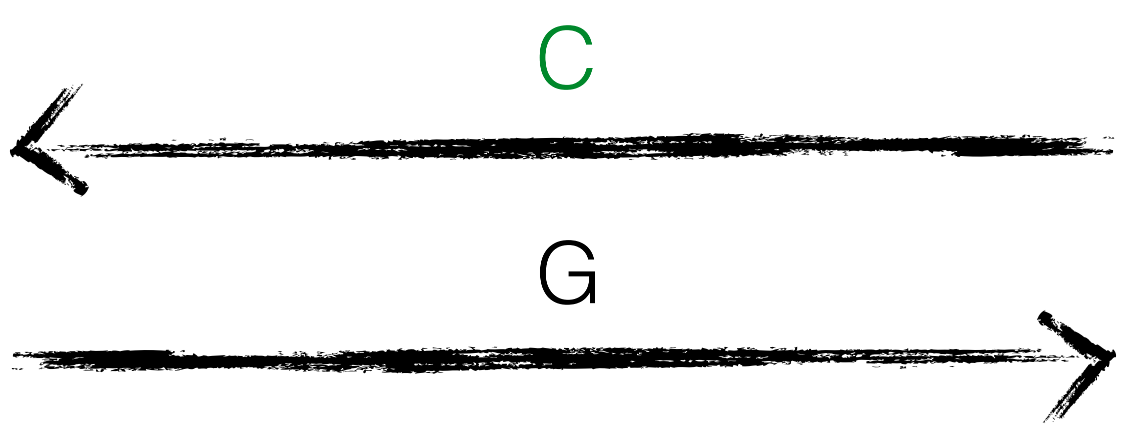
C cannot be isomorphic to H (due to transitivity)

G ⇐

π' ⇐ Random permutations

C ⇐ π'(G)

Unbounded



Poly

Interactive Proofs (formal definition)

Definition 4.2.6 (Generalized Interactive Proof): *Let* $\mathsf{c}, \mathsf{s} : \mathbb{N} \to \mathbb{R}$ *be functions satisfying* $\mathsf{c}(n) > \mathsf{s}(n) + \frac{1}{p(n)}$ *for some polynomial* $p(\cdot)$. *An interactive pair* (P, V) *is called a (generalized) interactive proof system for the language* L, *with* **completeness bound** $\mathsf{c}(\cdot)$ *and* **soundness bound** $\mathsf{s}(\cdot)$, *if*

- (modified) completeness: *for every* $x \in L$,

$$\Pr[\langle P, V\rangle(x) = 1] \geq \mathsf{c}(|x|)$$

- (modified) soundness: *for every* $x \notin L$ *and every interactive machine* B,

$$\Pr[\langle B, V\rangle(x) = 1] \leq \mathsf{s}(|x|)$$

In the previous example c(|x|)=1 and s(|x|)=1/2

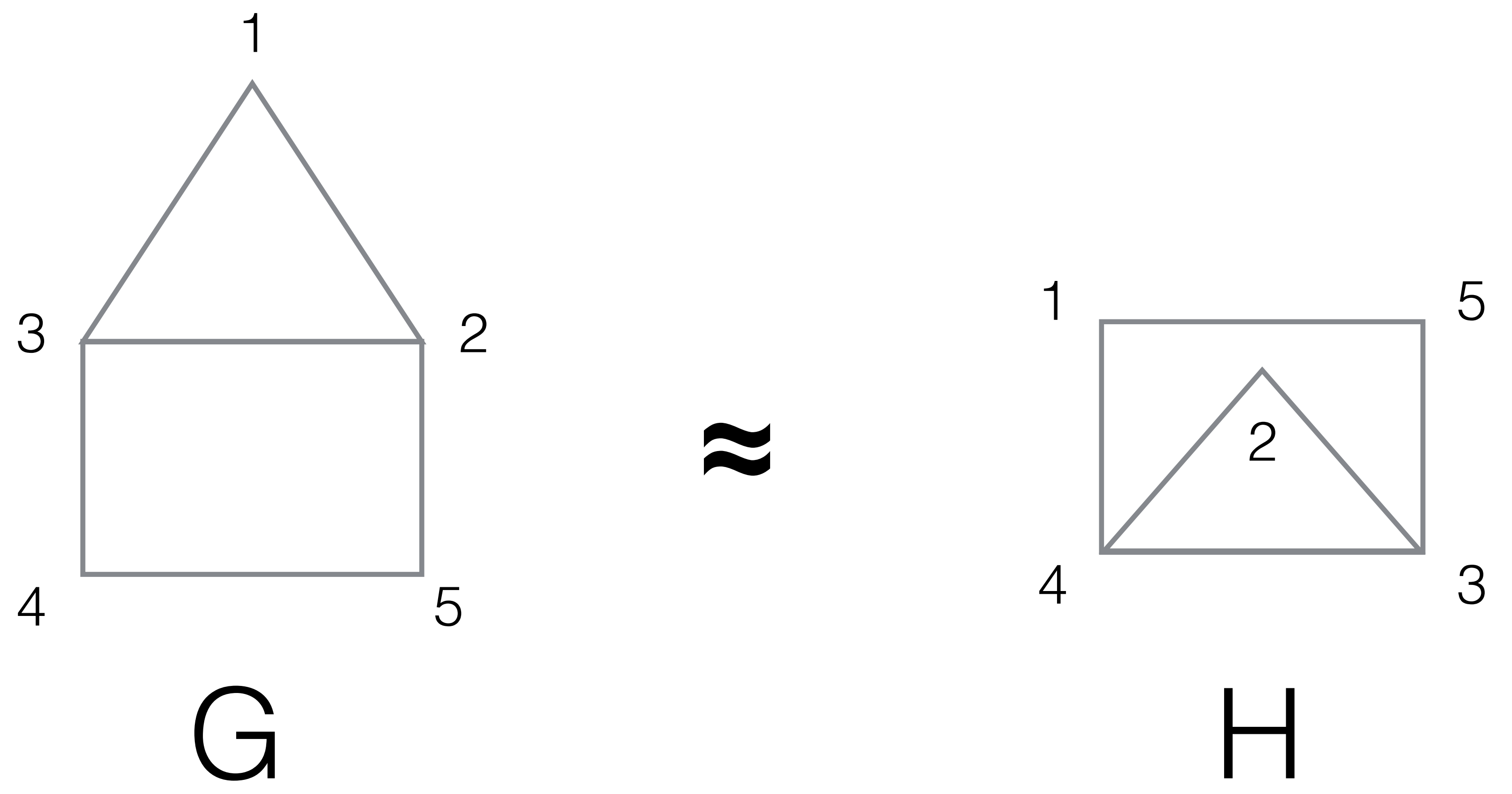
Zero-Knowledge (ZK)

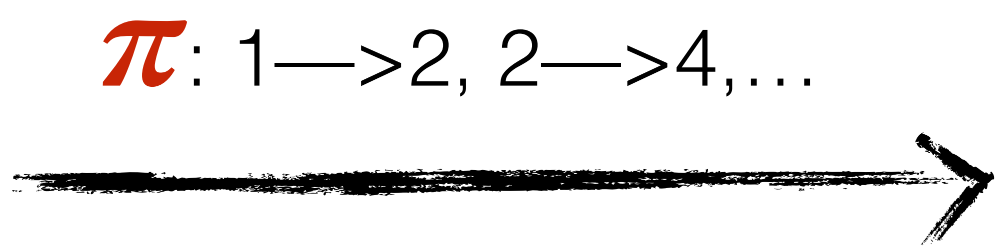
Witness

π

G	H
1	2
2	4
3	3
4	5
5	1

Thm





- How much knowledge is transmitted to the verifier?
- We would like to transmit only one bit: 1 if the theorem is true and 0 otherwise.
- E.g. For the case of graph isomorphism, the prover does not want to disclose the witness

ZK for Graph Isomorphism

Witness

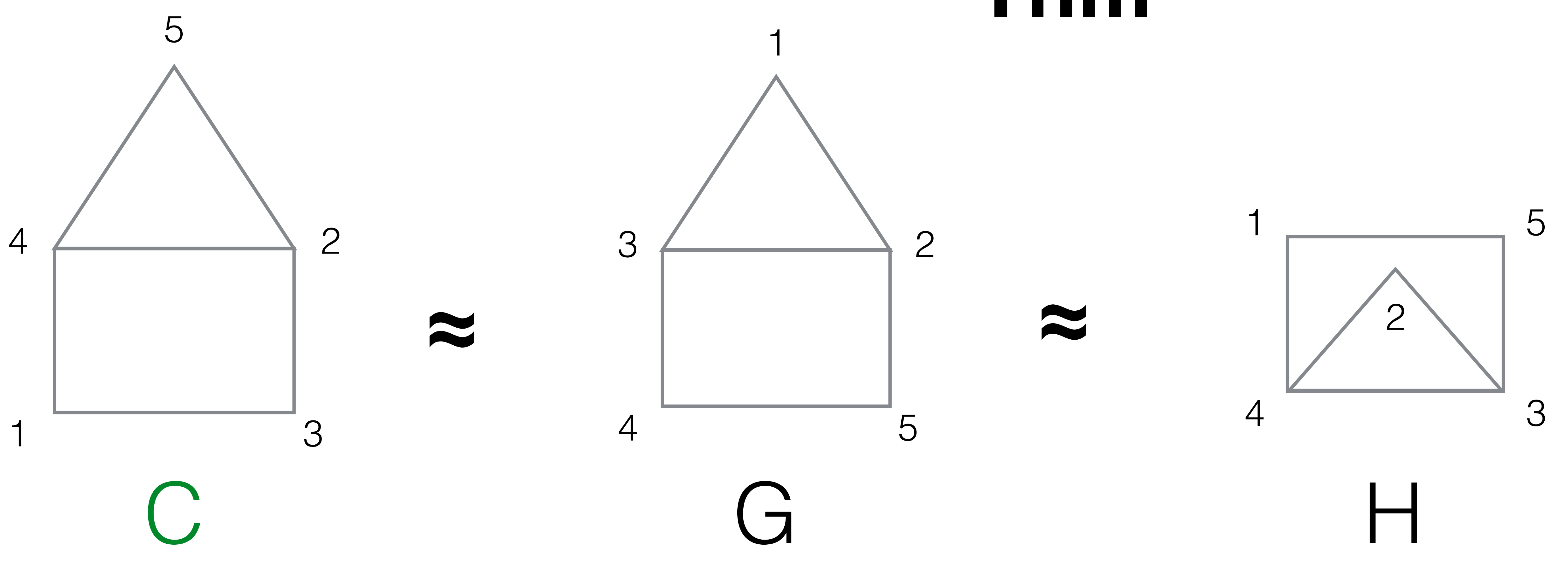
π

G	H
1	2
2	4
3	3
4	5
5	1

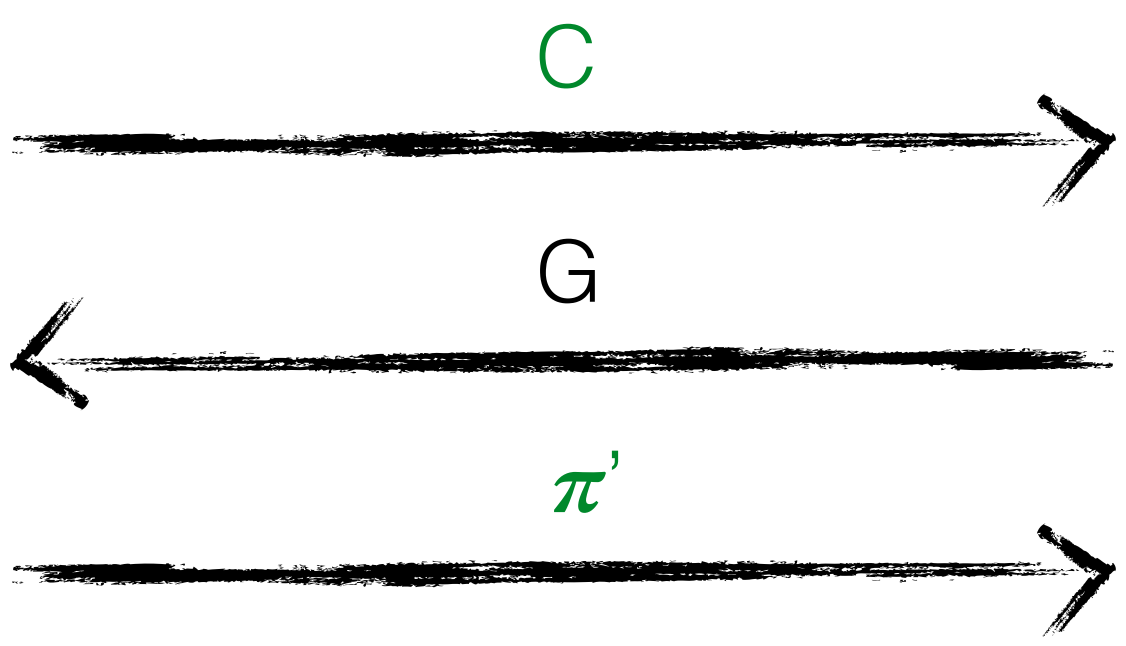
π'

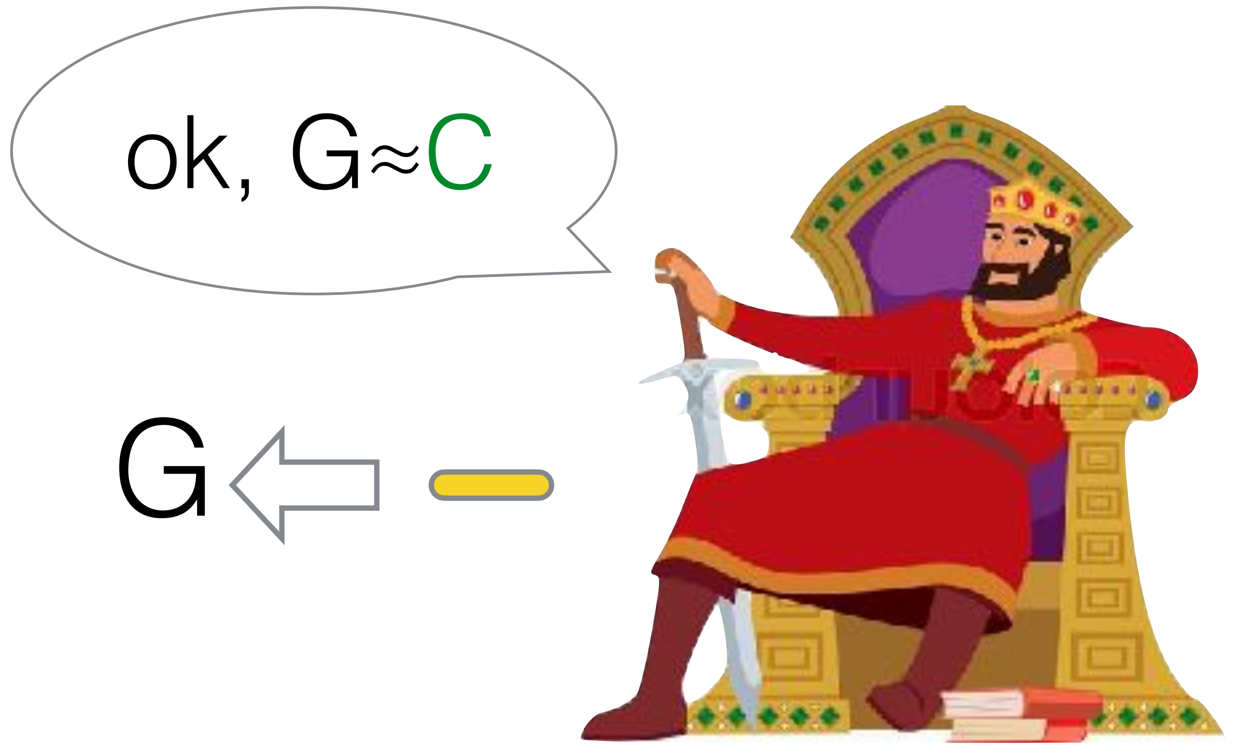
G	C
1	5
2	2
3	4
4	1
5	3

Thm



$C \Leftarrow \pi'(G)$





ZK for Graph Isomorphism

Witness

π

H	G
2	1
4	2
3	3
5	4
1	5

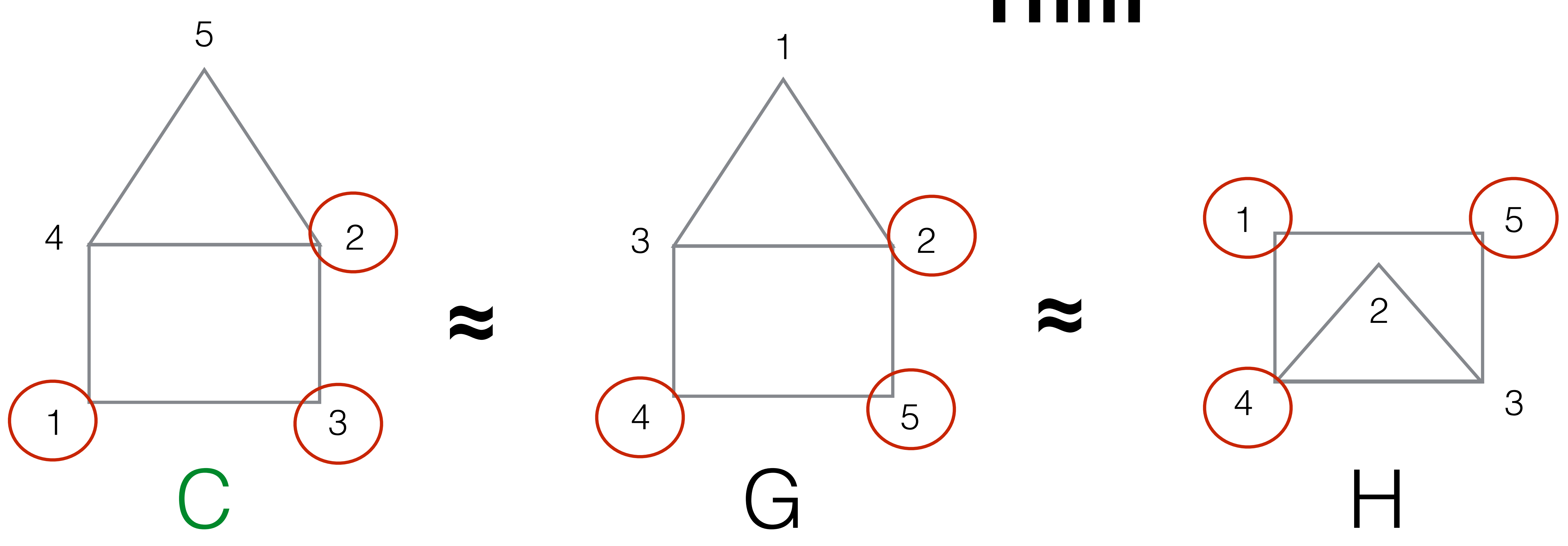
π'

G	C
1	5
2	2
3	4
4	1
5	3

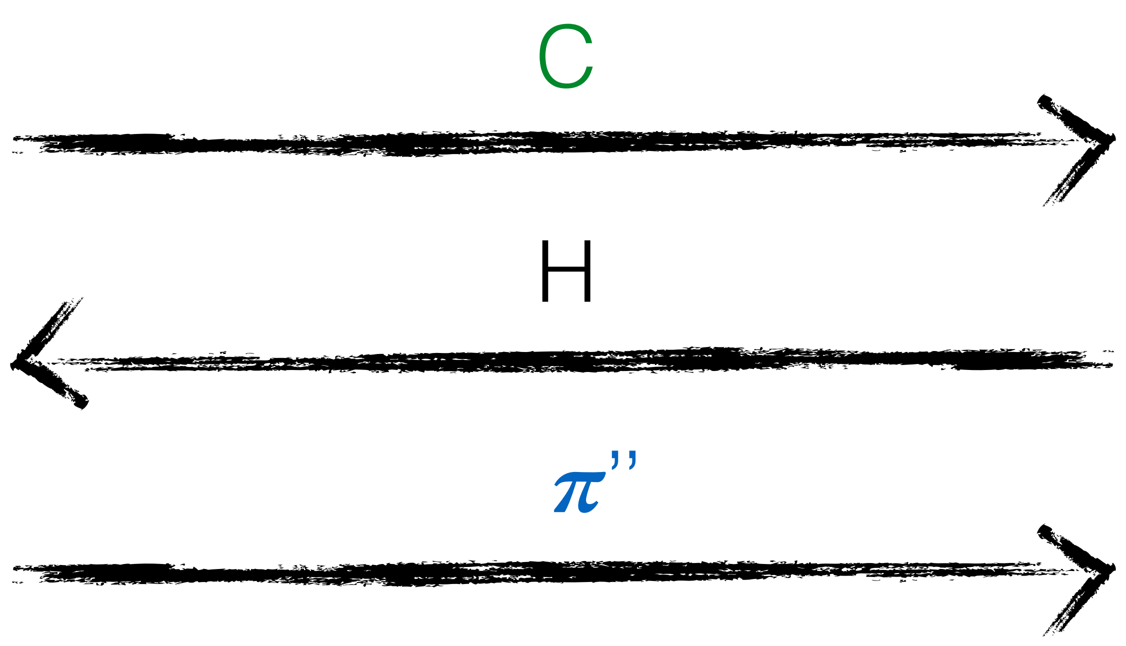
π''

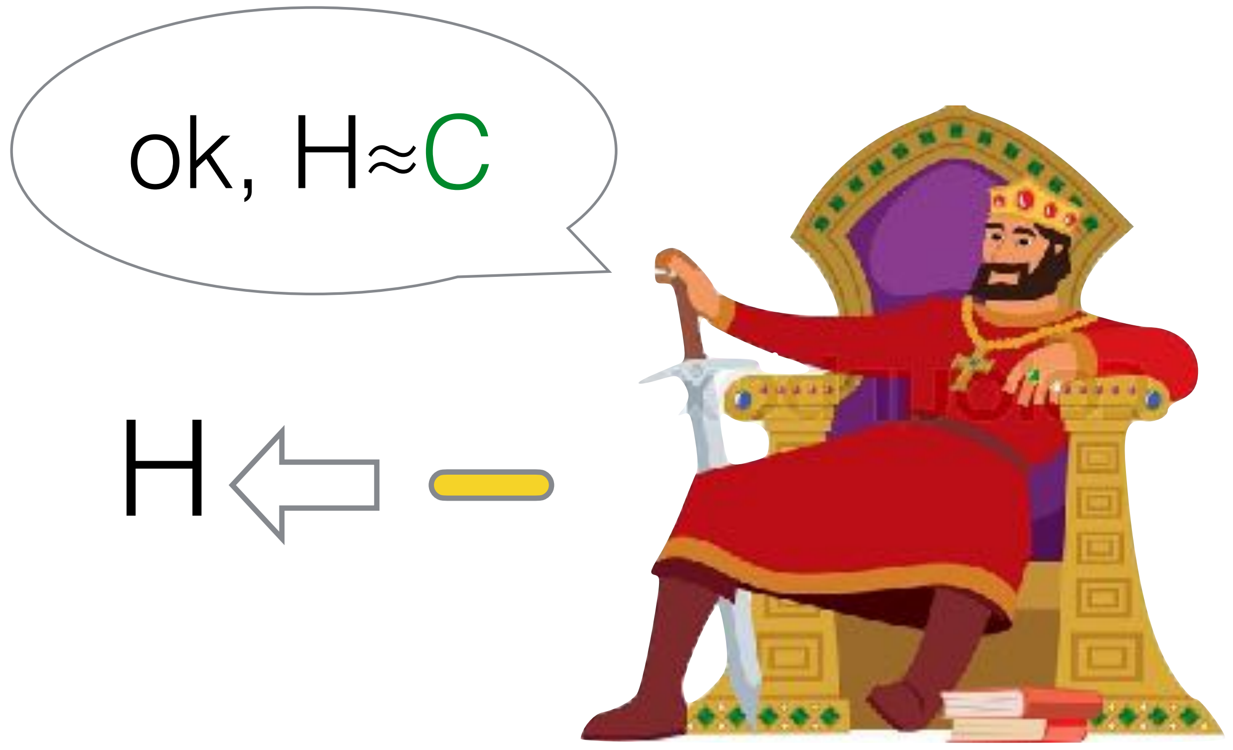
C	H
1	
2	
3	
4	
5	

Thm



C ⇦ π'(G)





If the graphs are non-isomorphic then the prover convinces the verifier with a 50% probability

We can repeat the proof many times to make this probability small

Zero Knowledge

- The notion of zero knowledge requires the existence of a simulator **S** that:
  - knows **only** that the theorem is true
  - is efficient
  - generates a transcript that is distributed similarly* to the real one (when the verifier is honest)
  - has black-box access to the adversary

Honest-Verifier ZK for Graph Isomorphism

π

H	**G**
2	1
4	2
3	3
5	4
1	5

Thm G ≈ H

C →

← G

π' →

ok, G≈C

G ⇐ ▬

Sim **Thm** G ≈ H

G ⇐ ▬

C ←

π^* ⇐ Random permutation of G

G →

C ⇐ π^*(G)

π^* ←

Sigma protocols

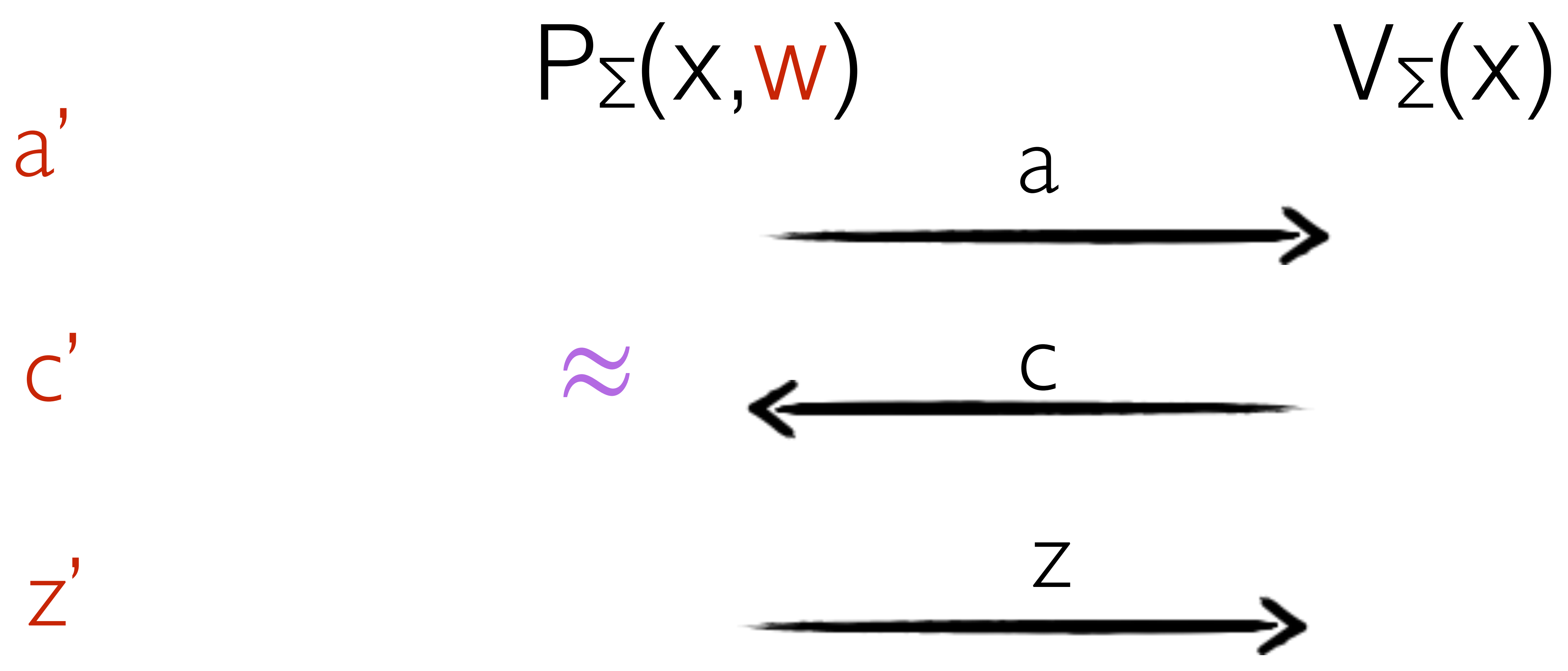
Thm: x

- Completeness

  Computational
- Honest Verifier Zero-Knowledge $\mathcal{HVZK}_{Sim}(x) \Longrightarrow$

  Special Honest Verifier Zero-Knowledge $\mathcal{SHVZK}_{Sim}(x,c) \Rightarrow$ a’,z’

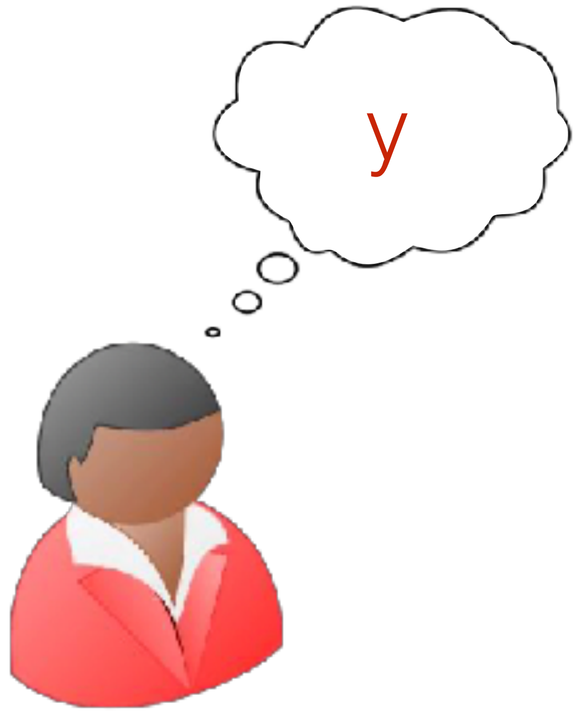
  Computational
- Special Soundness



x, (**a** c z)	c ≠ c’ ⇒	w: (**x**,w)∈ R
x, (**a** c’ z’)		

Schnorr protocol

Let G be a group of order q, with generator g



$x = g^y$

$a = g^r$ →

← c

$z = r + cy$ →

Accept iff $g^z = a x^c$

$g^z = g^{r+cy}$ $\quad$ $a x^c = g^r g^{yc} = g^{r+cy}$

Special-soundness

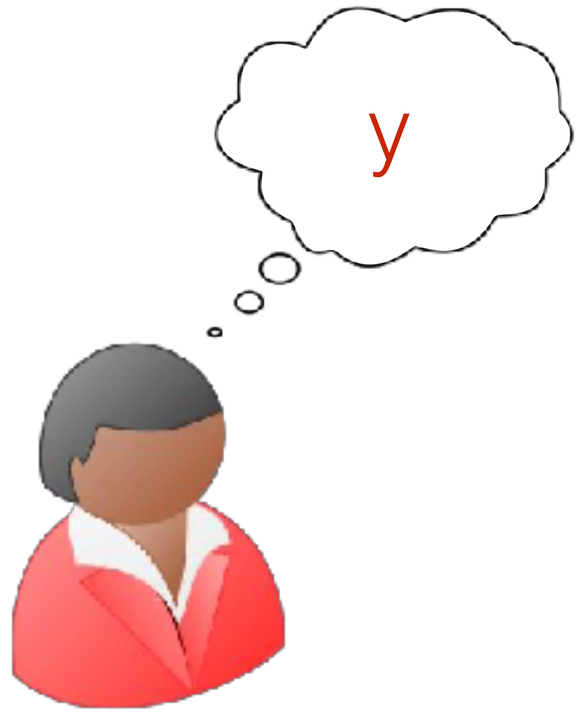
a →

← c $\quad$ ← c'

z → $\quad$ z' →

$$\begin{cases} z = r + cy \\ z' = r + c'y \end{cases} \xrightarrow{c \neq c'} y$$

Schnorr protocol



$x = g^y$

$a = g^r$ →

c ←

$z = r + cy$ →

Accept iff $g^z = a x^c$

HVZK

$\mathcal{HVZK}_{sim}$

$c \Leftarrow Z_q$

$z \Leftarrow Z_q$

$a = g^z / x^c$

a →

c ←

z →

Sigma Protocol for Diffie-Hellman tuples

Let G be a group of order q, with generators g and h

$x=(g, h, u, v)$ Is a DH tuple if $u=g^{y}, v=h^{y}$

$b \leftarrow \{0,1\}$

if $b=0$ then

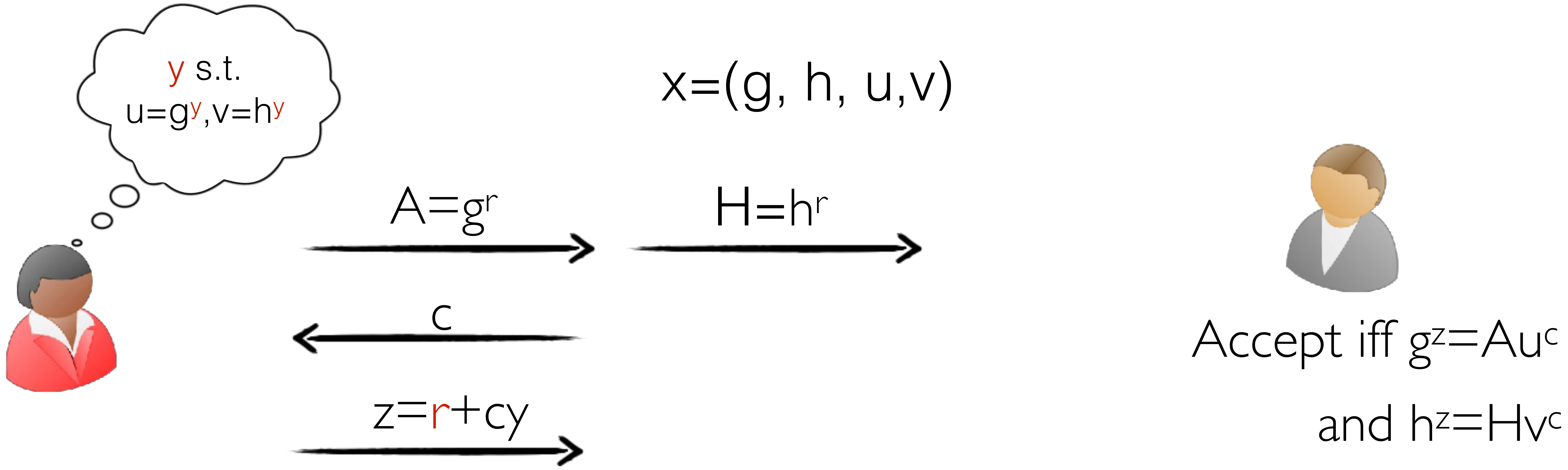
$T=(g, h, u=g^{y}, v=h^{y})$

else

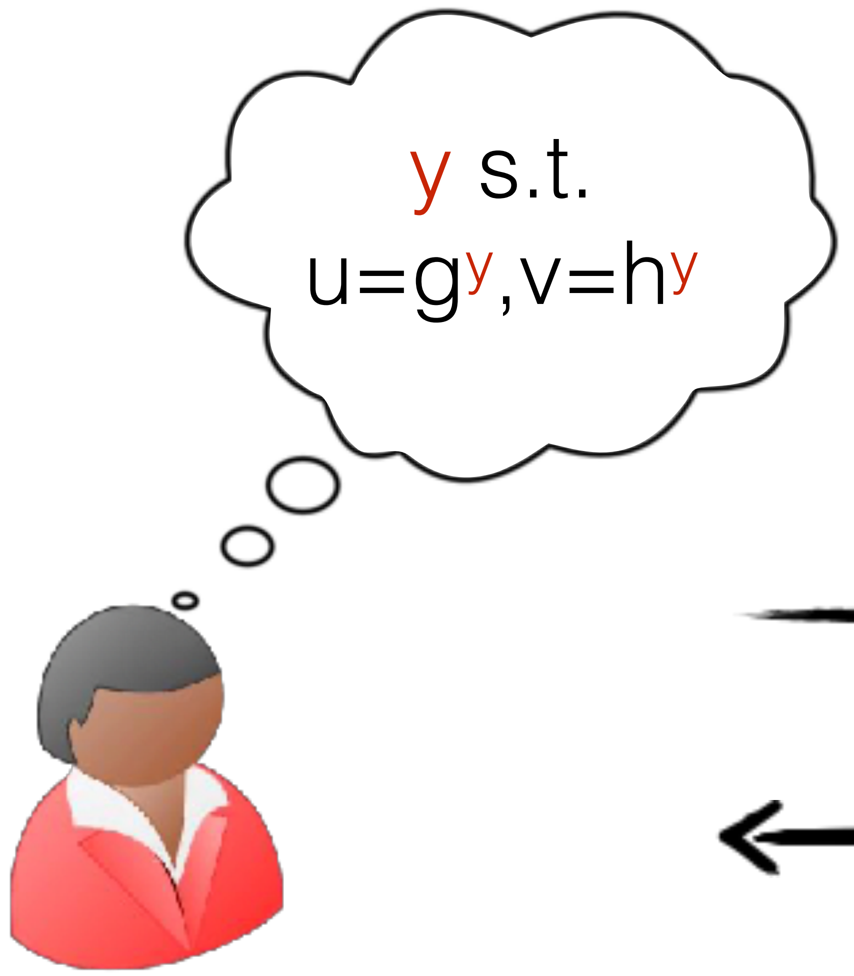
$T=(g, h, u=g^{y}, v=h^{w})$ with $y \neq w$

$\xrightarrow{T}$

Sigma Protocol for Diffie-Hellman tuples



Sigma Protocol for Diffie-Hellman tuples



$x=(g, h, u, v)$

$A=g^r$ $\longrightarrow$ $H=h^r$ $\longrightarrow$

$\longleftarrow$ c

$z=r+cy$ $\longrightarrow$

Accept iff $g^z=Au^c$

and $h^z=Hv^c$

$HVZK_{sim}$

$c \Leftarrow Z_q$

$z \Leftarrow Z_q$

$A=g^z/u^c$

$H=h^z/v^c$

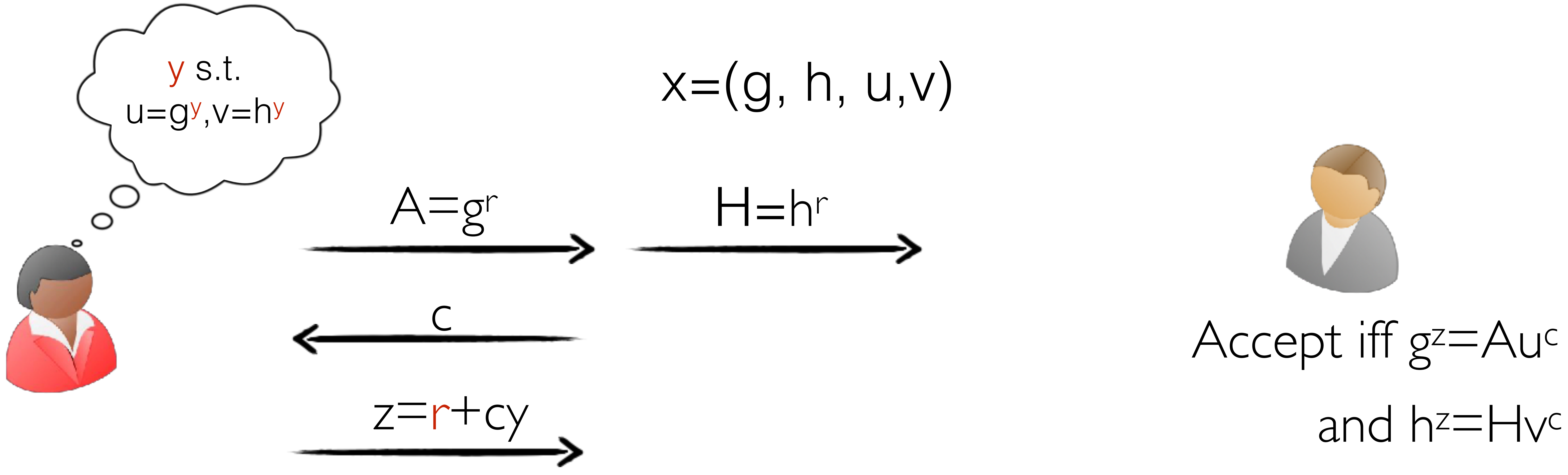
$a=(A,H)$ $\longrightarrow$

$\longleftarrow$ c

z $\longrightarrow$

HVZK

Sigma Protocol for Diffie-Hellman tuples



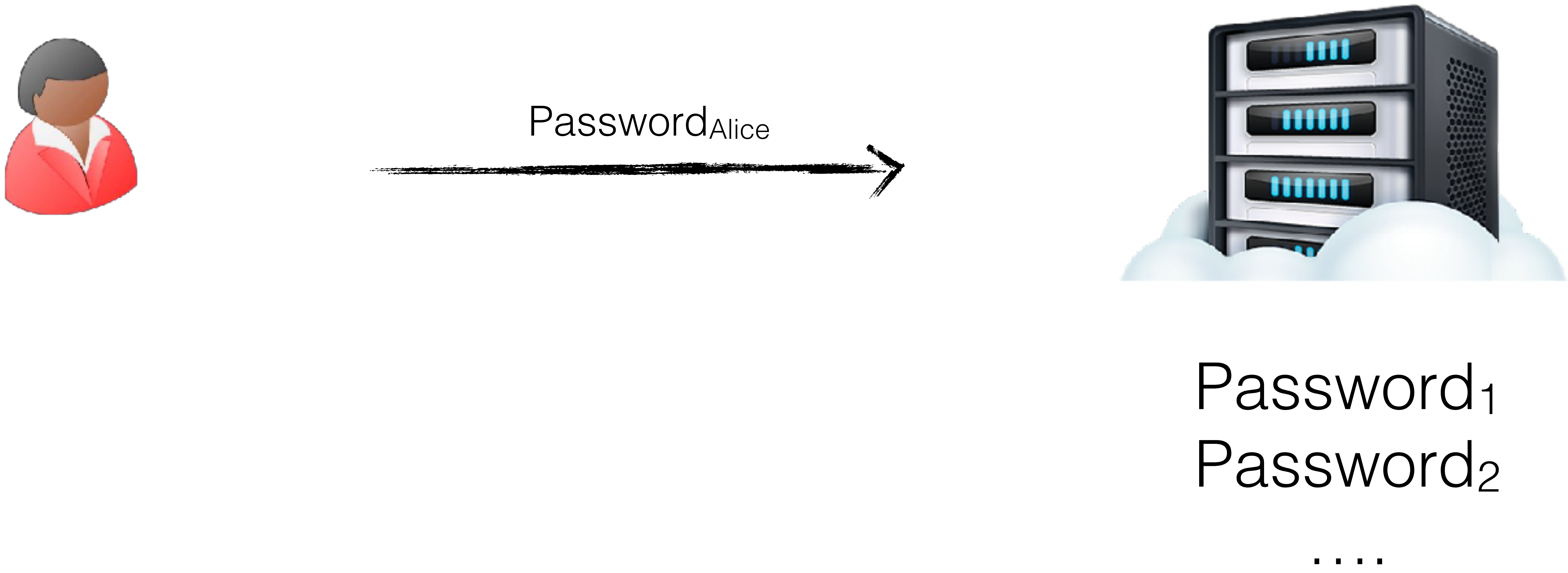
Special-soundness

Exactly the same as the one for the Dlog protocol

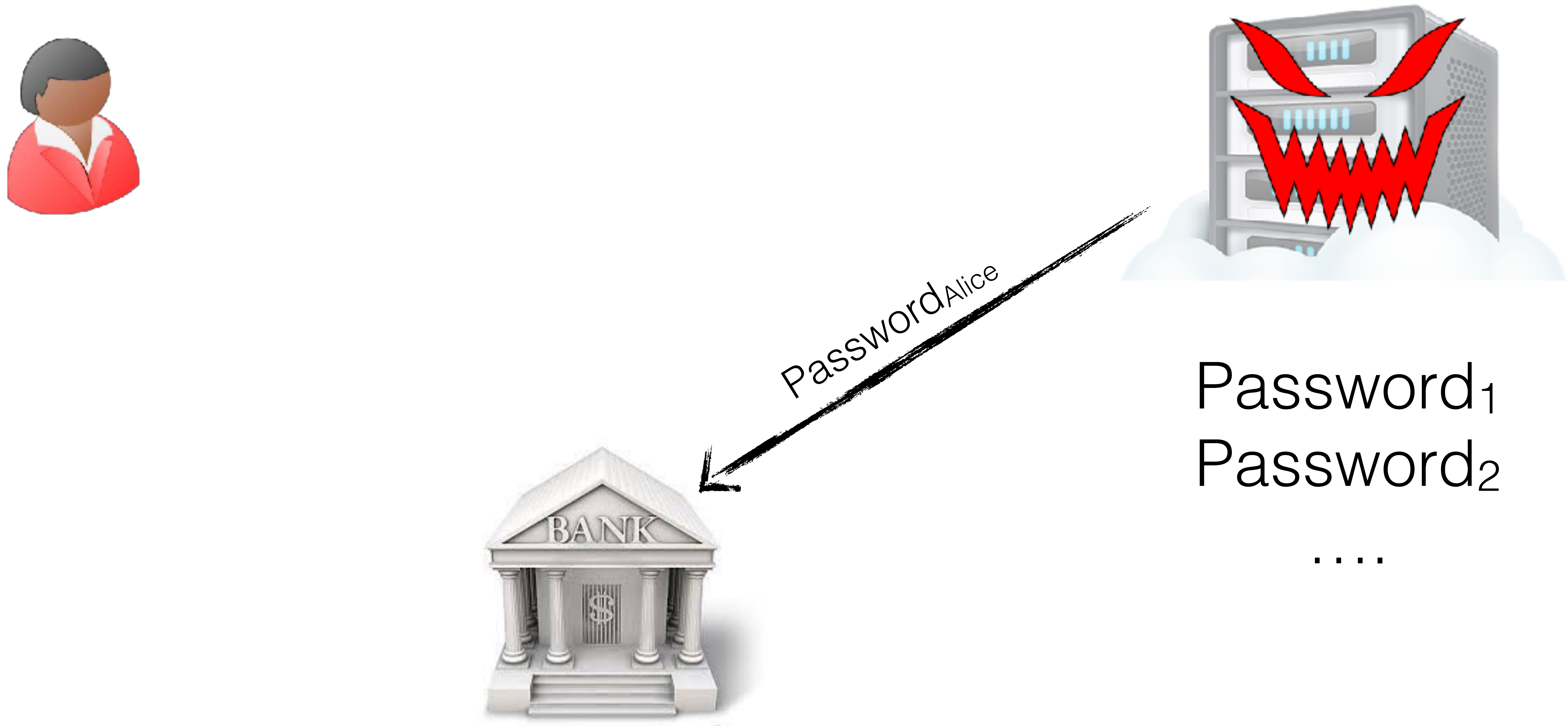
Why do we care?

- We know how to construct ZK proofs for any NP-language (with both efficient prover and verifier)
- CCA-encryption scheme
- Multi-party computation
- Identification schemes
- Privacy-preserving blockchains

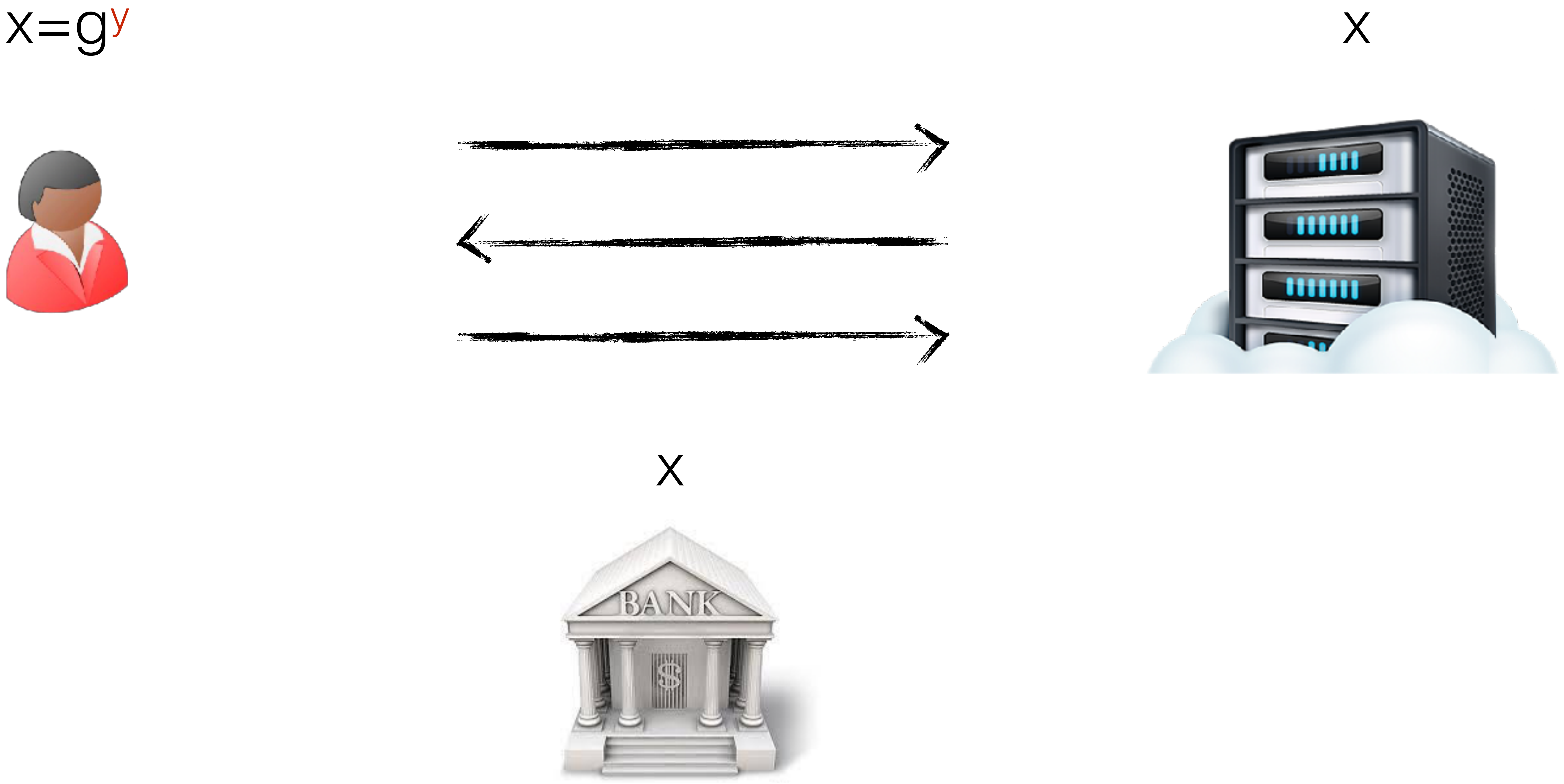
Identification scheme



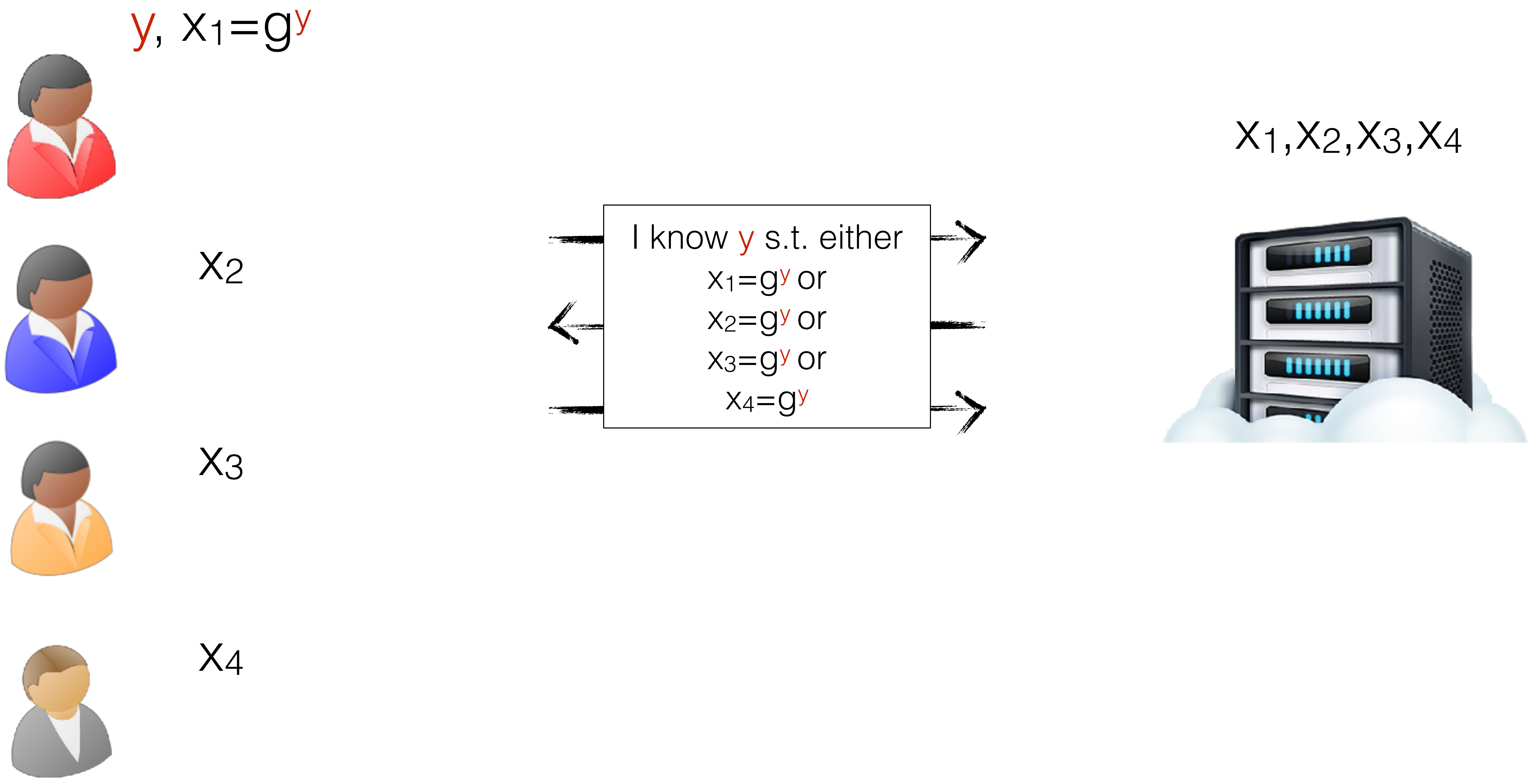
Identification scheme



Identification scheme



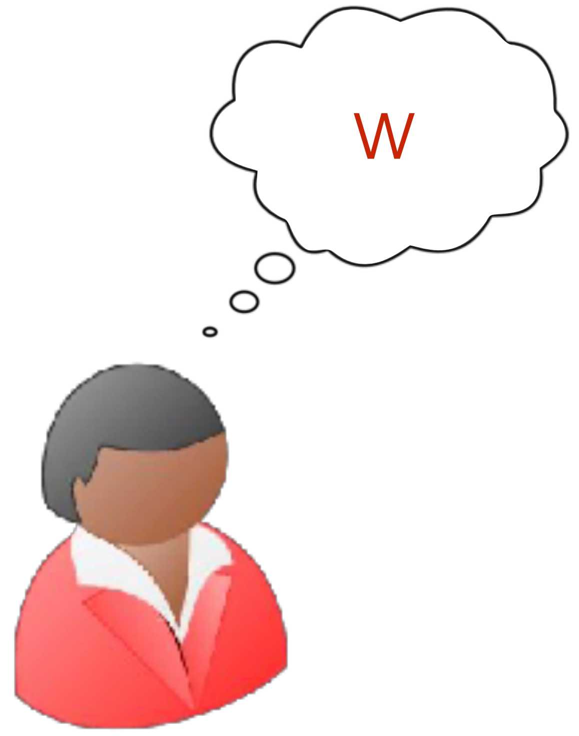
Identification scheme



Summary/Notes

- Sigma-Protocol
- Every language in NP has a sigma-protocol
- Can we circumvent the 3-round impossibility and design an efficient non-interactive argument?

How do we make non-interactive proofs?



$\mathbf{x}$

$\mathcal{O}$

$a \leftarrow P_\Sigma(\mathbf{x})$

$c \leftarrow \mathcal{O}(a, \mathbf{x})$

$z \leftarrow P_\Sigma(\mathbf{x}, w, c)$

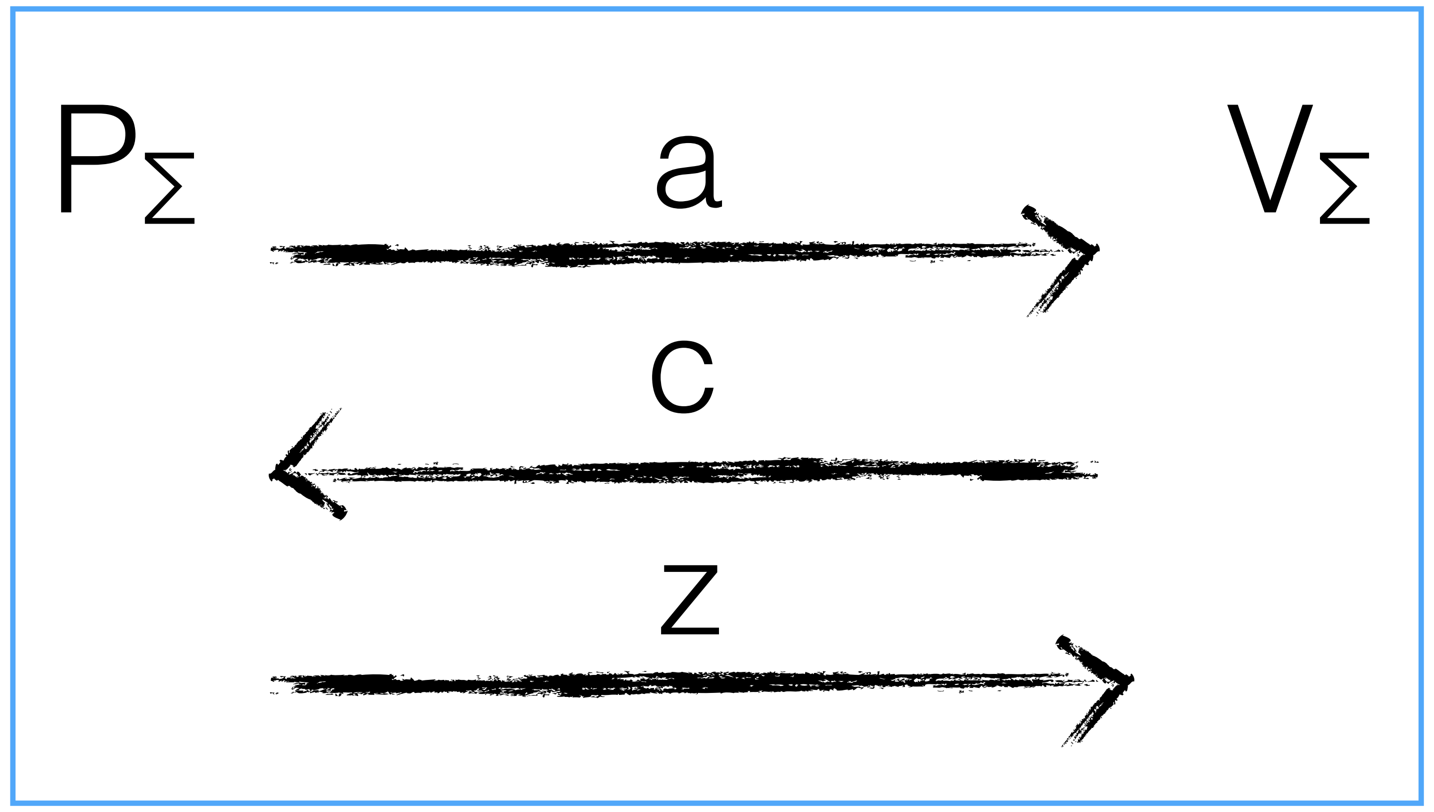
a,z →

$c \leftarrow \mathcal{O}(a, \mathbf{x})$

$V_\Sigma(a,c,z)=1$

- Fiat-Shamir transform
- in practice $\mathcal{O}$ is a hash function (e.g.SHA2)

- Adds very little overhead to the starting sigma-protocol
- Used in practice for identification scheme, signatures, SNARKS, …



Conclusions

- Non-interactive zero-knowledge (NIZK) proofs: length of the proof and verification time dependent on the NP language
  - Known from standard falsifiable assumptions
  - Setup is needed (just RO would suffice)

- SNARKs proofs: length of the proof depends on the security parameter and the verification time is dependent on the instance only
  - Setup is needed (even in the RO model)
  - Based on non-falsifiable assumptions (Knowledge of Exponent Assumptions)

End

References from the book of Goldreich Oded: Foundations of Cryptography: Volume 1, Basic Tools (see the link on learn)

- Sec. 4.2 until (included) Sec. 4.2.2 with no proofs
- Sec. 4.3 until (included) Sec. 4.3.2 with no proofs
- Sec. 4.7 until (included) Definition 4.7.2 with no proofs

More References on Sigma-Protocols: On Sigma-Protocols. Ivan Damgaard. https://www.cs.au.dk/~ivan/Sigma.pdf