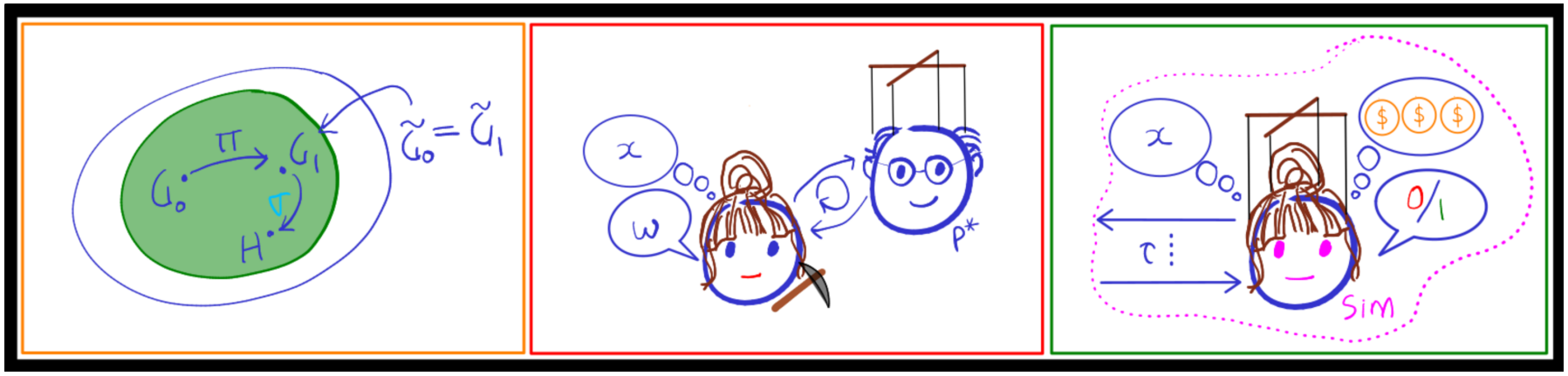
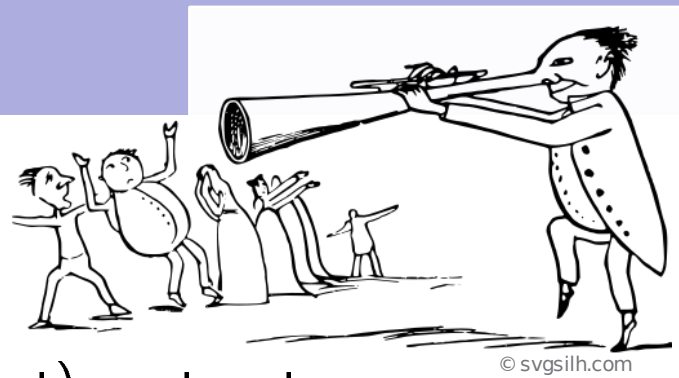


CS409m: Introduction to Cryptography

Lecture 19 (22/Oct/25)

Instructor: Chethan Kamath

Announcements



- Feedback form for course (post mid-sem part) sent out
- Assignment 5 out yesterday (21/Oct)

- Quiz 2 viewing on 24/Oct (Friday), 12:30-14:30
  - Submit your cribs online by 29/Oct (next Wednesday)
- Quiz 3 on 29/Oct (next Wednesday)
  - 08:25-09:25, in CC103/CC105

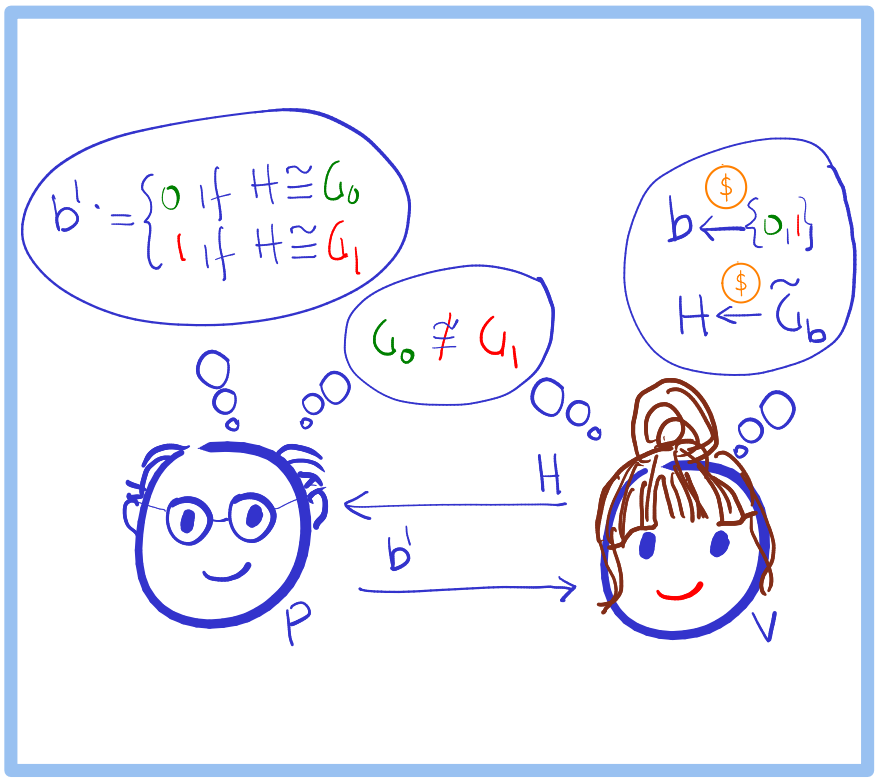
- Lab Exercise 4 will be released today (22/Oct)
  - Submit flag by 29/Oct EoD (Wednesday)
  - Submit write-up by 31/Oct EoD (Friday)

Recall from Last Lecture...

- Interactive proofs vs NP proofs:
  - Prover convinces verifier using *interaction*
  - Verifier is *random*

Interactive Proof (IP)

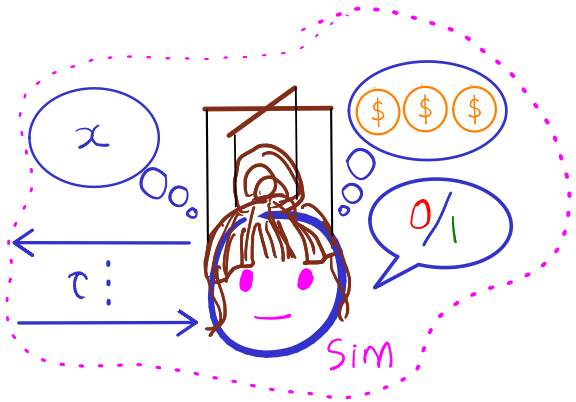
IP for GNI



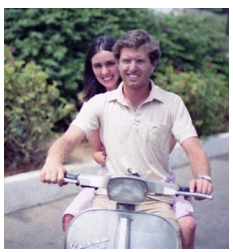
Plan for Today's Lecture...

IP where prover reveals *no non-trivial knowledge* to the verifier

NEW Zero-knowledge (ZK) IP

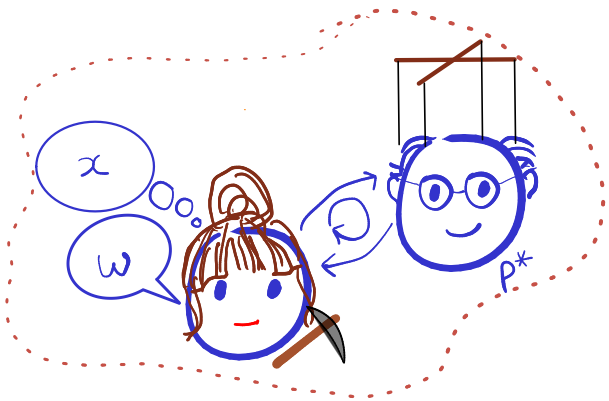


© alchetron.com



©Oded Goldreich

ZK Proof of Knowledge (PoK) NEW



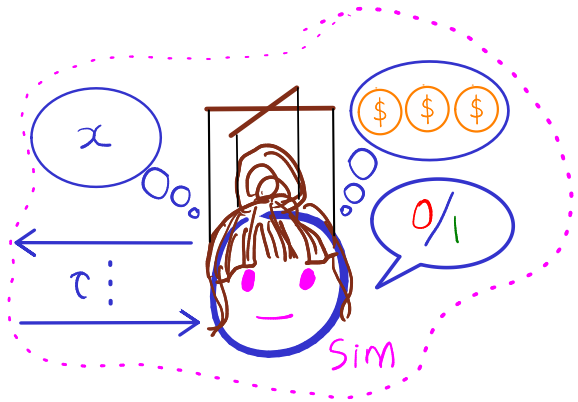
©Konrad Jacobs/Wkipedia

Plan for Today's Lecture...

IP where prover reveals *no non-trivial knowledge* to the verifier

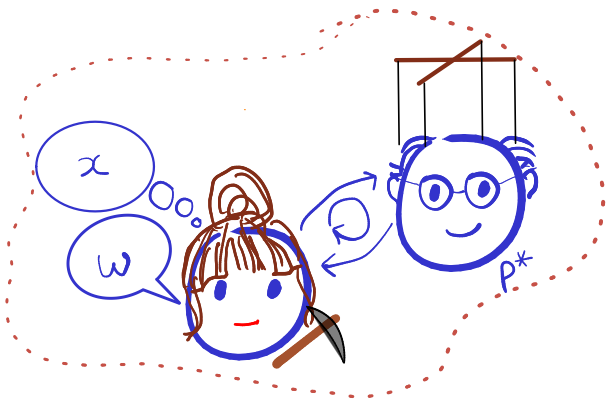
NEW Zero-knowledge (ZK) IP

ZK Proof of Knowledge (PoK) NEW



© alchetron.com

©Oded Goldreich



©Konrad Jacobs/Wkipedia

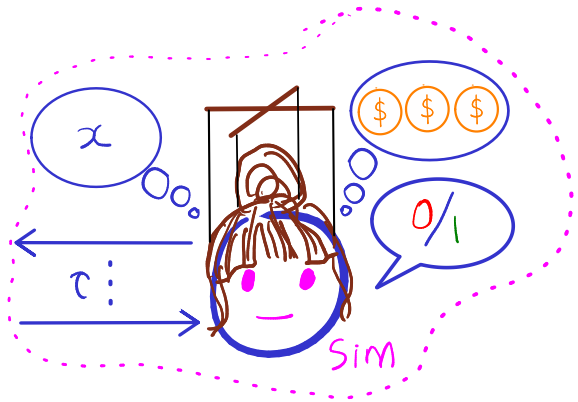
Main tools: simulator and extractor

Plan for Today's Lecture...

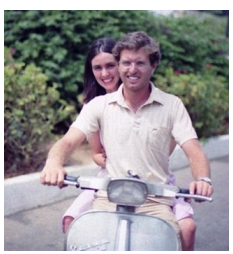
IP where prover reveals *no non-trivial knowledge* to the verifier

NEW Zero-knowledge (ZK) IP

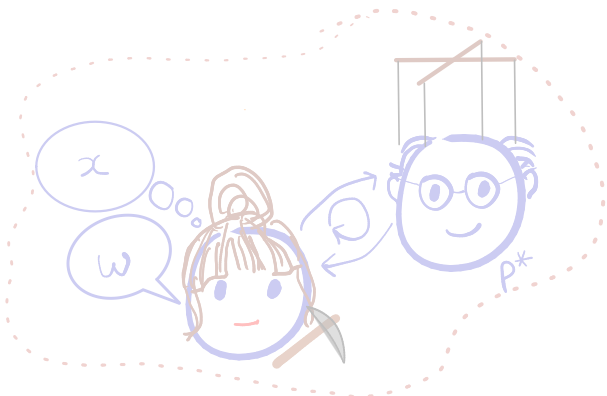
ZK Proof of Knowledge (PoK) NEW



© alchetron.com



©Oded Goldreich

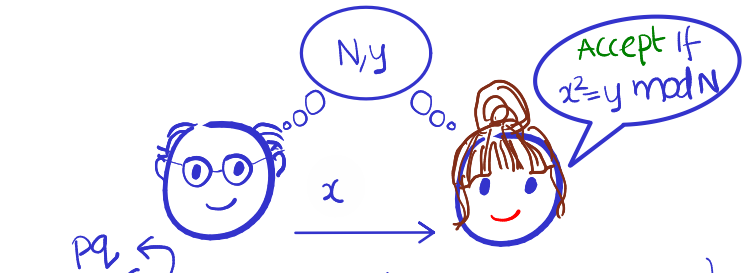


©Konrad Jacobs/Wikipedia

Main tools: simulator and extractor

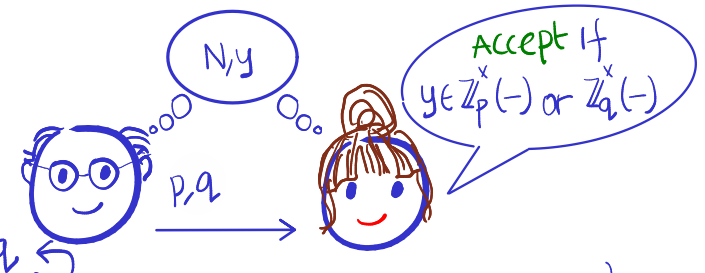
The NP Proofs We Saw Leaked Information

Quadratic residuosity (QR)



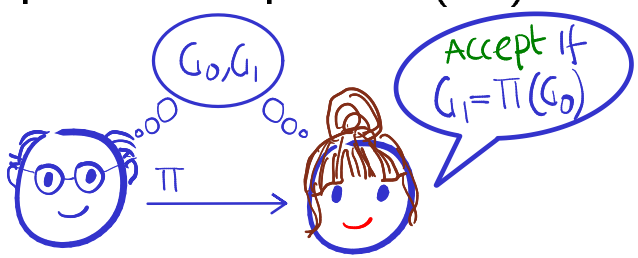
$$\alpha_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Quad. non-residuosity (QNR)



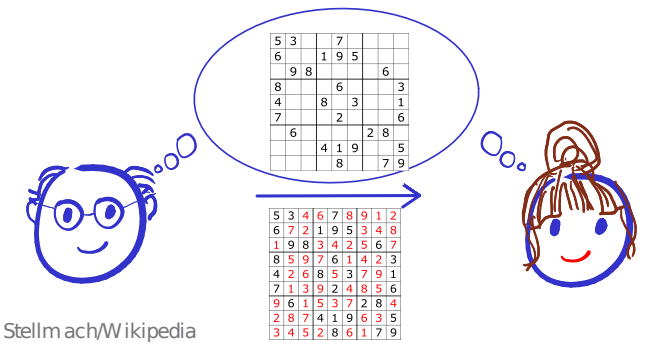
$$\alpha_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Graph isomorphism (GI)



$$\alpha_{GI} = \{(G_0, G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

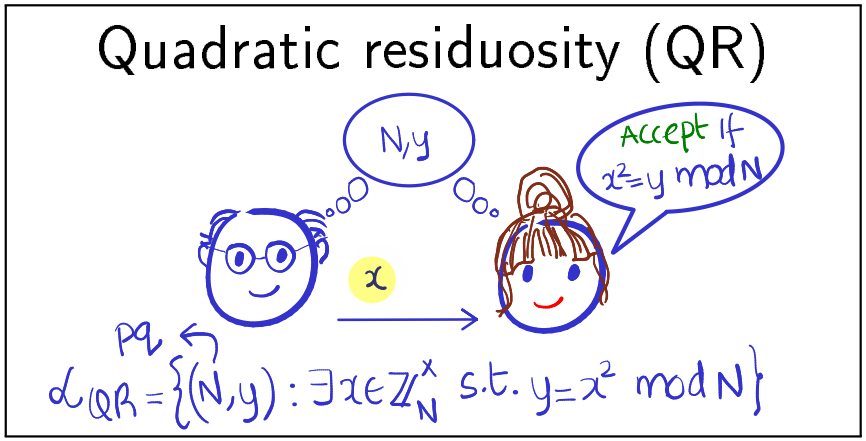
Sudoku

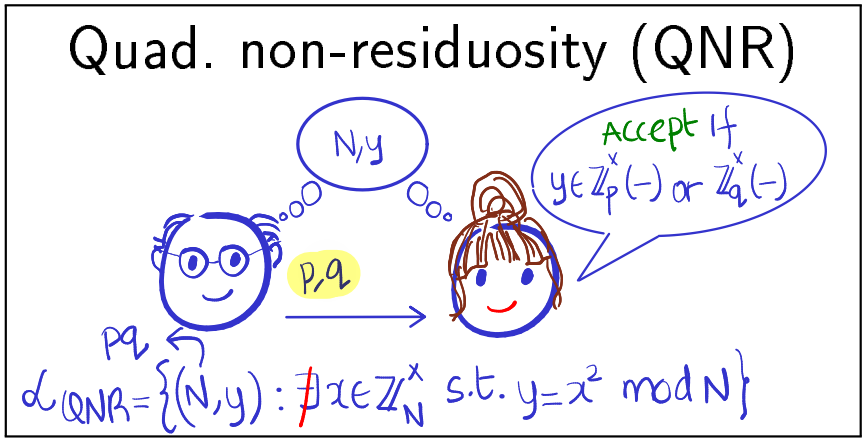


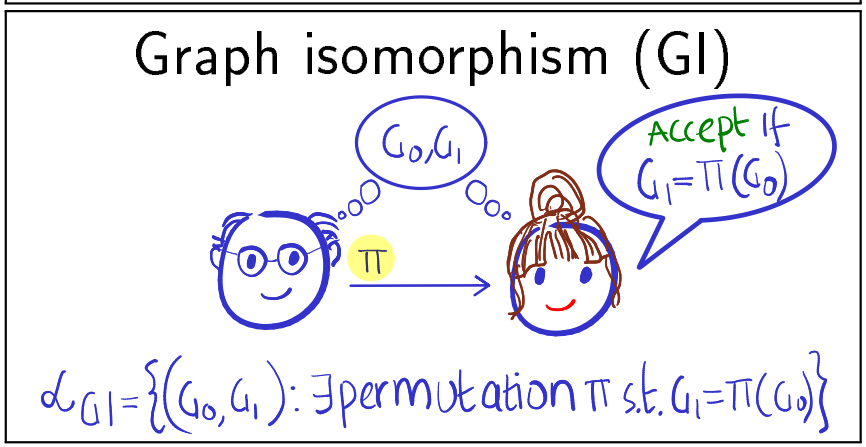
© Tim Stellmach/Wikipedia

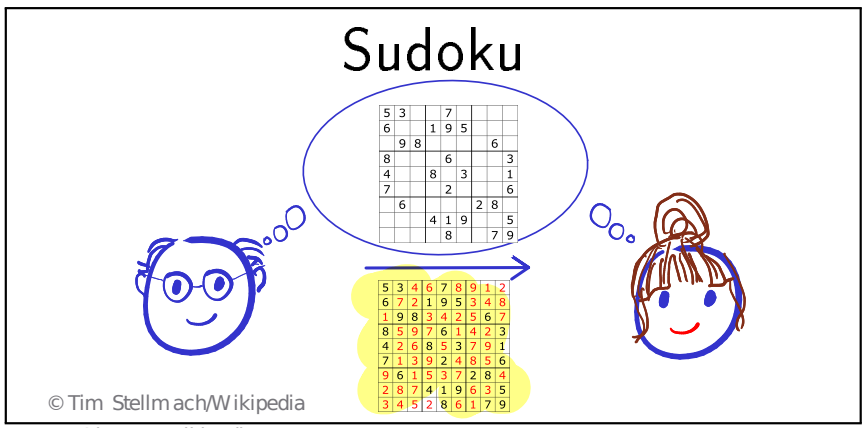
© Cburnett/Wikipedia

The NP Proofs We Saw Leaked Information







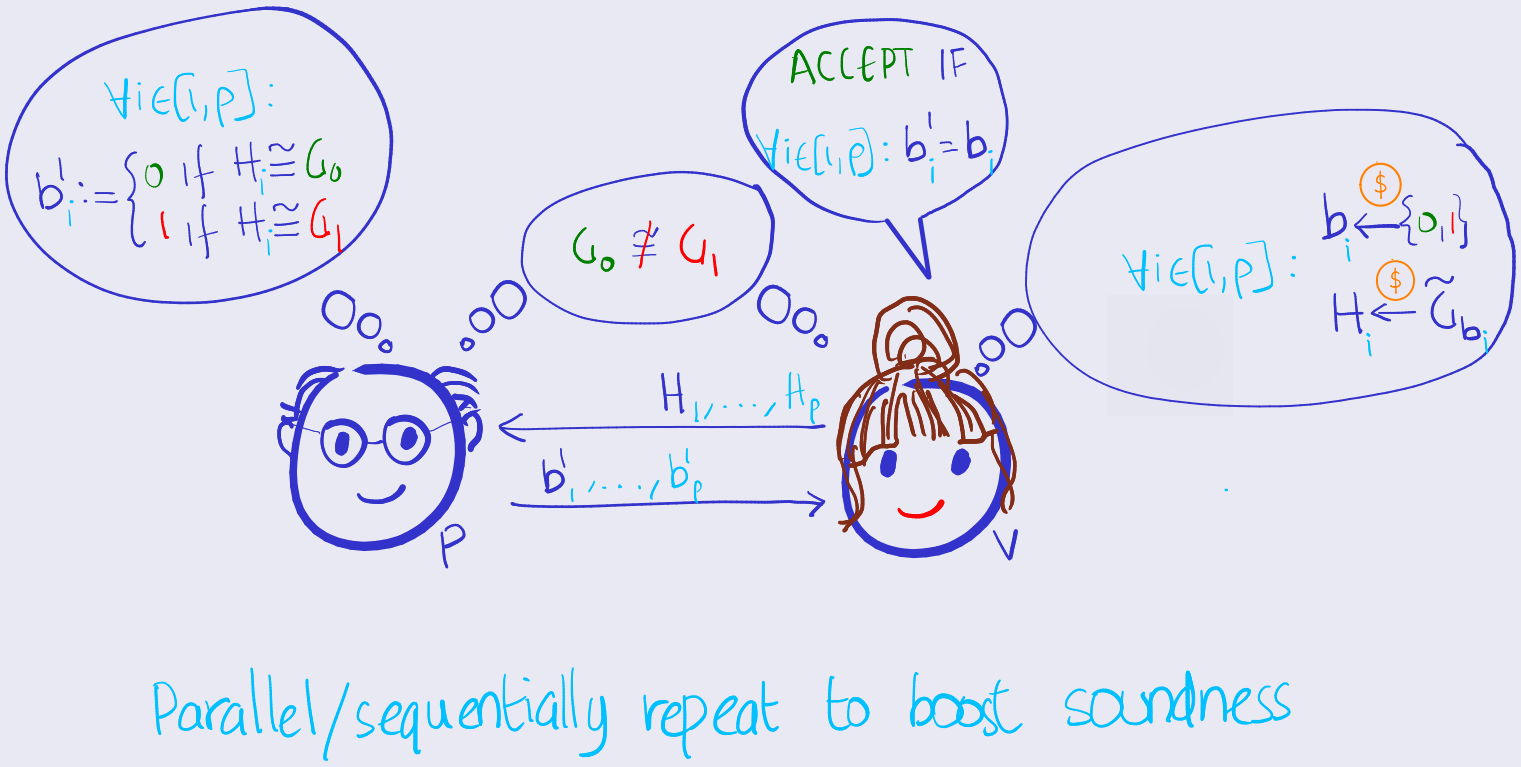


© Cburnett/Wikipedia

- Verifier gains "non-trivial knowledge" about witness w
  - Not desirable, e.g., when $x = pk$ and $w = sk$ (identification)

But the IP for GNI We Saw Doesn't Seem to

Protocol 1 (Π_{GNI}: IP for GNI)



Parallel/sequentially repeat to boost soundness

- Seems V *gains no knowledge* beyond validity of the statement
- We will show that Π_{GNI} is (honest-verifier) zero-knowledge!

How to Capture "V Gains No Knowledge"?

Knowledge vs. information ← in the information-theoretic sense

- Knowledge is computational

How to Capture "V Gains No Knowledge"?

Knowledge vs. information

- Knowledge is computational: e.g., consider NP proof for GI
  - Given (G_0, G_1), the isomorphism π contains no *information*
  - But when given π, V "gains knowledge" since she couldn't have computed π herself

How to Capture "V Gains No Knowledge"?

Knowledge vs. information

- Knowledge is computational: e.g., consider NP proof for GI
  - Given (G_0, G_1), the isomorphism π contains no *information*
  - But when given π, V "gains knowledge" since she couldn't have computed π herself
- Knowledge pertains to public objects:
  - Flipping a *private* fair coin b and (later) revealing its outcome leads to V gaining *information*
  - But V *does not gain knowledge*: she could herself have tossed the private coin and revealed it

How to Capture "V Gains No Knowledge"?

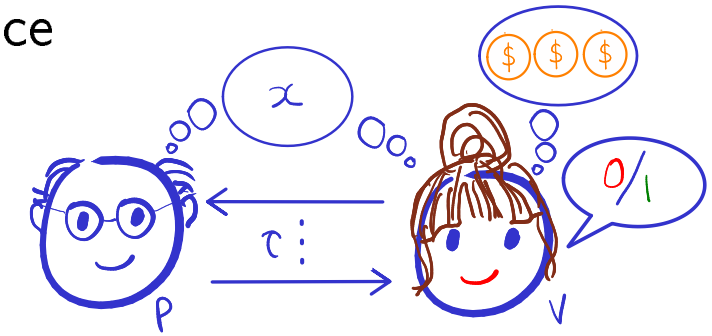
Knowledge vs. information

- Knowledge is computational: e.g., consider NP proof for GI
  - Given (G_0, G_1), the isomorphism π contains no *information*
  - But when given π, V "gains knowledge" since she couldn't have computed π herself
- Knowledge pertains to public objects:
  - Flipping a *private* fair coin b and (later) revealing its outcome leads to V gaining *information*
  - But V *does not gain knowledge*: she could herself have tossed the private coin and revealed it

Intuitively, "V gains no knowledge" if anything V can *compute* after the interaction, V could have computed *without it*

Defining Zero Knowledge via Simulators...

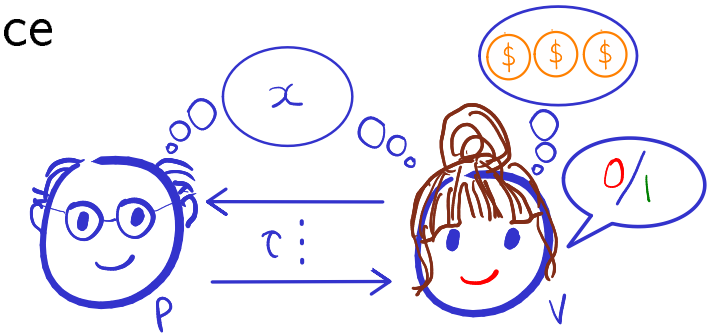
- Formalised via "simulation paradigm": $\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))$ can be *efficiently* simulated given only the instance



Defining Zero Knowledge via Simulators...

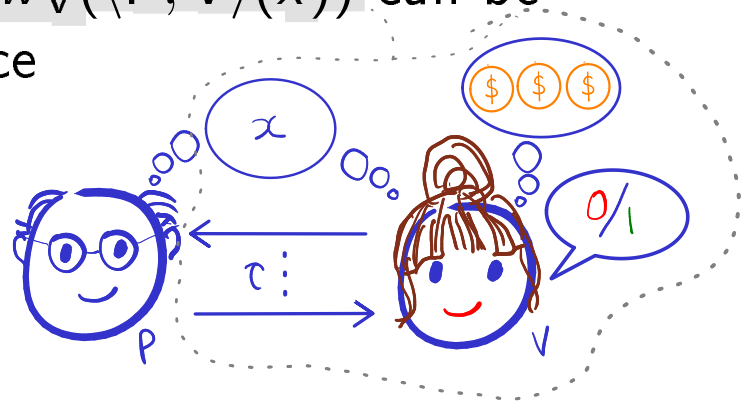
- Formalised via "simulation paradigm": $\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))$ can be *efficiently* simulated given only the instance

V's "view" = x + transcript τ + coins



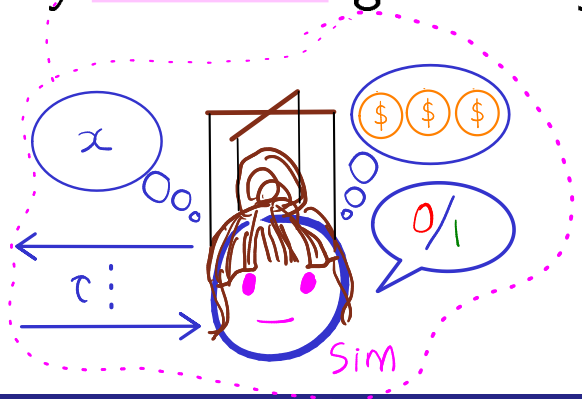
Defining Zero Knowledge via Simulators...

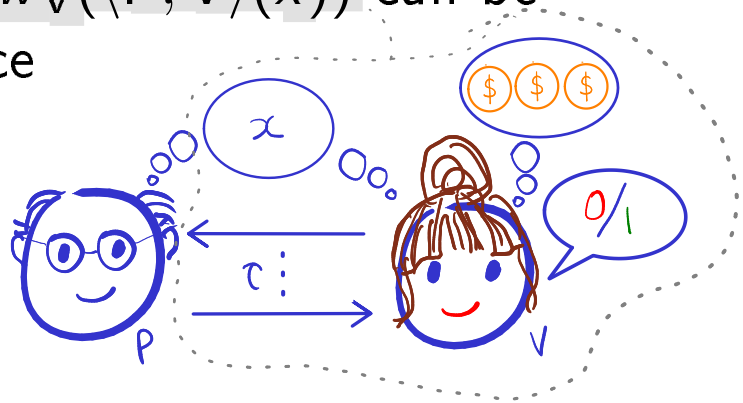
- Formalised via "simulation paradigm": $\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))$ can be *efficiently* simulated given only the instance



Defining Zero Knowledge via Simulators

- Formalised via "simulation paradigm": $\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))$ can be *efficiently* simulated given only the instance





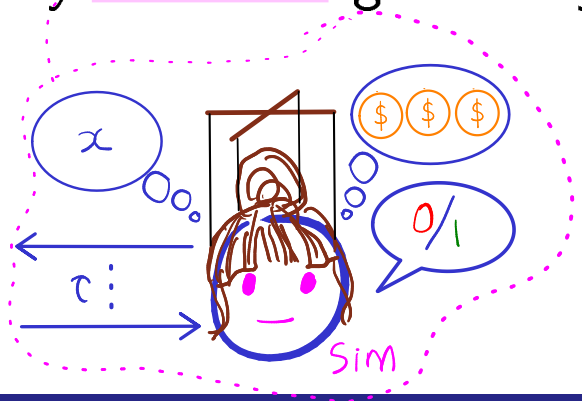
Definition 1 (Honest-Verifier *Perfect* ZK)

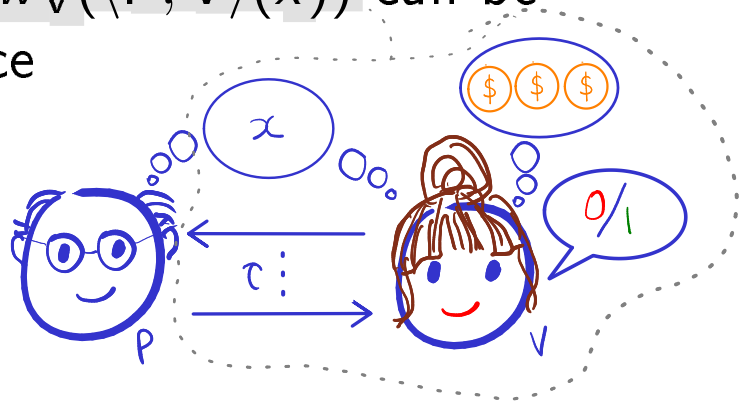
An IP Π is *honest-verifier perfect* ZK if there exists a PPT *simulator* Sim such that for *all* distinguishers D and all $x \in \mathcal{L}$

$$\Pr[\mathsf{D}(\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))) = 1] = \Pr[\mathsf{D}(\mathsf{Sim}(x)) = 1]$$

Defining Zero Knowledge via Simulators

- Formalised via "simulation paradigm": $\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))$ can be *efficiently* simulated given only the instance





Definition 1 (Honest-Verifier *Perfect* ZK)

An IP Π is *honest-verifier perfect* ZK if there exists a PPT *simulator* Sim such that for *all* distinguishers D and all $x \in \mathcal{L}$

$$\Pr[\mathsf{D}(\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle(x))) = 1] = \Pr[\mathsf{D}(\mathsf{Sim}(x)) = 1]$$

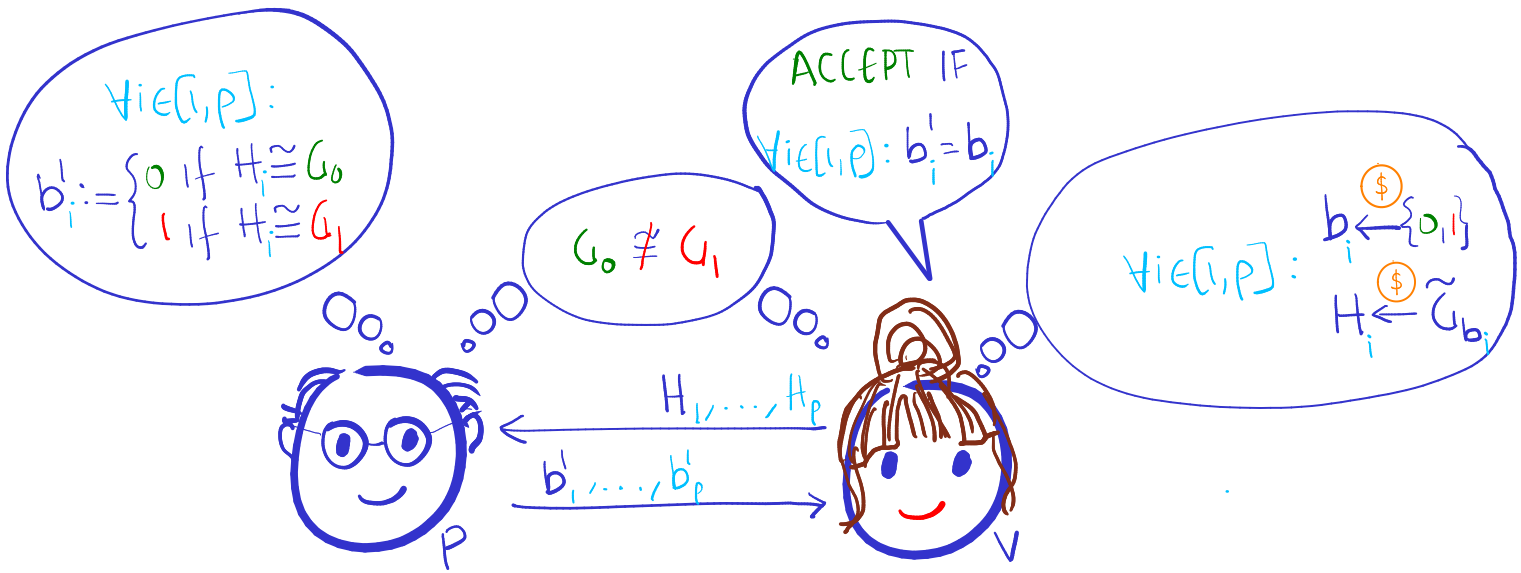
Exercise 1

What happens when one invokes the simulator on $x \notin \mathcal{L}$?

Π_{GNI} is *Honest-Verifier* ZK

Theorem 1

Π_{GNI} is honest-verifier perfect zero-knowledge IP for $\mathcal{L}_{GNI}$

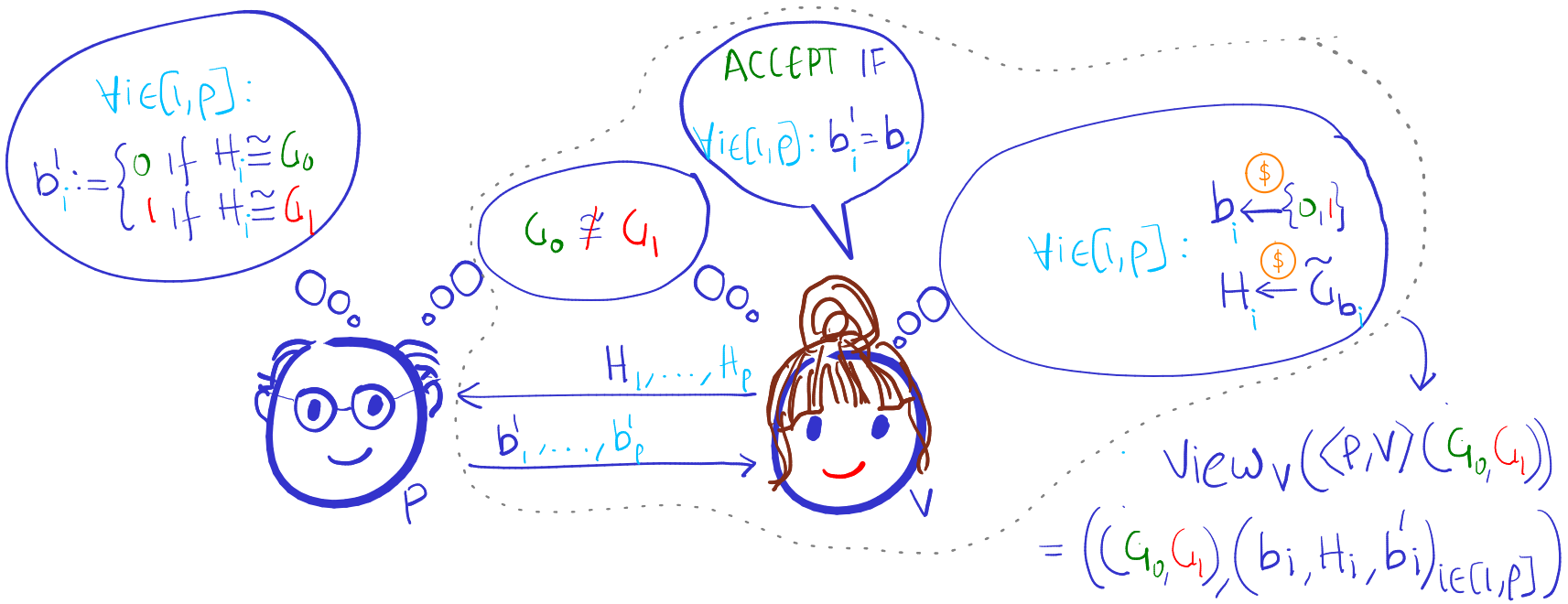


Parallel/sequentially repeat to boost soundness

Π_{GNI} is *Honest-Verifier* ZK

Theorem 1

Π_{GNI} is honest-verifier perfect zero-knowledge IP for $\mathcal{L}_{GNI}$



Parallel/sequentially repeat to boost soundness

Π_{GNI} is *Honest-Verifier* ZK

Theorem 1

Π_{GNI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GNI}$

Proof.

$$\text{view}_V(\langle P, V\rangle)(G_0, G_1) := \left((G_0, G_1), \left(b_i, H_i, b_i'\right)_{i\in[1,p]}\right)$$

$\forall\, G_0 \not\cong G_1$:

Π_{GNI} is *Honest-Verifier* ZK

Theorem 1

Π_{GNI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GNI}$

Proof.

$$\text{View}_V(\langle P, V\rangle)(G_0, G_1) := \left((G_0, G_1), \left(b_i, H_i, b_i'\right)_{i\in[1,\rho]}\right)$$

$\forall\, G_0 \not\cong G_1$:

$\text{Sim}(G_0, G_1) :=$?

Π_{GNI} is *Honest-Verifier* ZK

Theorem 1

Π_{GNI} is honest-verifier perfect zero-knowledge IP for $\mathcal{L}_{GNI}$

Proof.

$$\text{View}_V(\langle P,V\rangle(G_0,G_1)) := \left((G_0,G_1),\left(b_i,H_i,b_i'\right)_{i\in[1,p]}\right)$$

$\forall\ G_0 \not\cong G_1$:

$\text{Sim}(G_0,G_1)$:= $\forall i\in[1,p]$: sample $b_i \leftarrow \{0,1\}$ and $H_i \leftarrow \tilde{G}_{b_i}$

o/p $\left((G_0,G_1),\left(b_i,H_i,b_i\right)_{i\in[1,p]}\right)$

$\forall\ G_0 \not\cong G_1$: $\text{View}_V(\langle P,V\rangle(G_0,G_1))$ identically distributed to $\text{Sim}(G_0,G_1)$.

Π_{GNI} is *Honest-Verifier* ZK

Theorem 1

Π_{GNI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GNI}$

Proof.

$\text{View}_V(\langle P,V\rangle(G_0,G_1)) := \left((G_0,G_1),\left(b_i,H_i,b_i'\right)_{i\in[1,p]}\right)$

$\forall\, G_0 \not\cong G_1$:

$\text{Sim}(G_0,G_1) :=$ $\forall i\in[1,p]$: sample $b_i \leftarrow \{0,1\}$ and $H_i \leftarrow \tilde{G}_{b_i}$

o/p $\left((G_0,G_1),\left(b_i,H_i,b_i\right)_{i\in[1,p]}\right)$

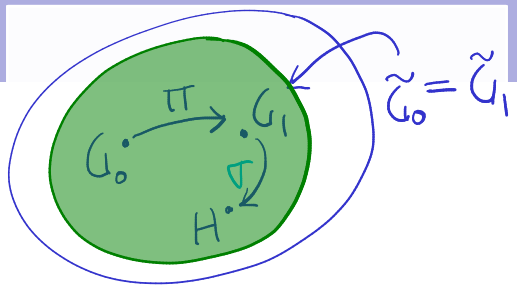
$\forall\, G_0 \not\cong G_1$: $\text{View}_V(\langle P,V\rangle(G_0,G_1))$ identically distributed to $\text{Sim}(G_0,G_1)$. □

Exercise 2

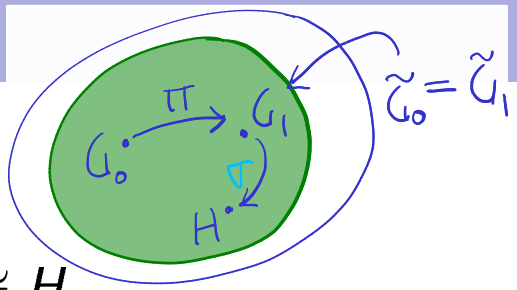
1. What happens if V is "malicious" and can deviate from protocol?
2. Using ideas from Π_{GNI}, build honest-verifier ZKP for $\mathcal{L}_{QNR}$

Honest-Verifier ZKP for GI...

Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$



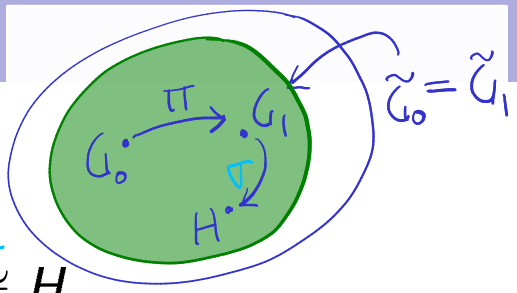
Honest-Verifier ZKP for GI...



Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \cong H$
2. Verifier challenges to "open" $G_0 \cong H$ or $G_1 \cong H$ at random

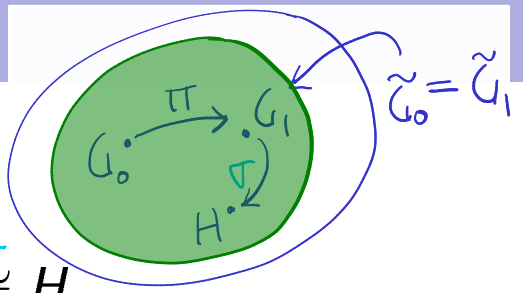
Honest-Verifier ZKP for GI...



Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

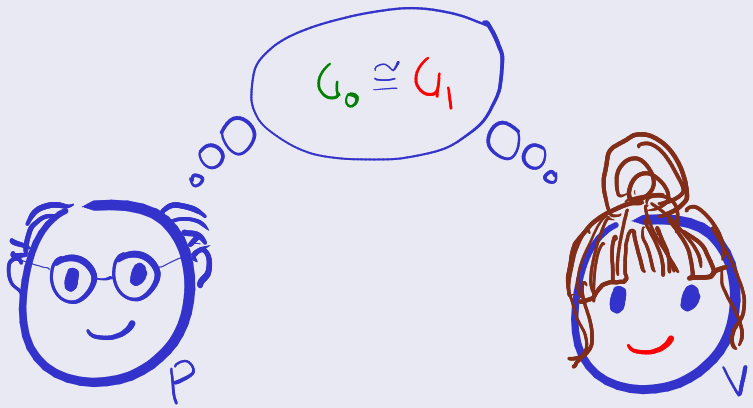
Honest-Verifier ZKP for GI...



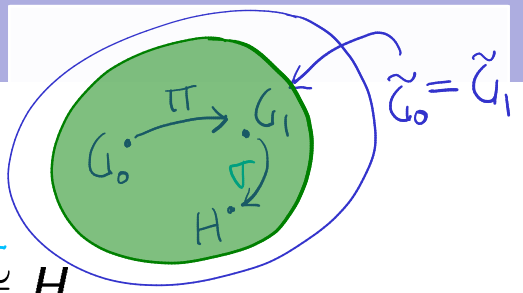
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



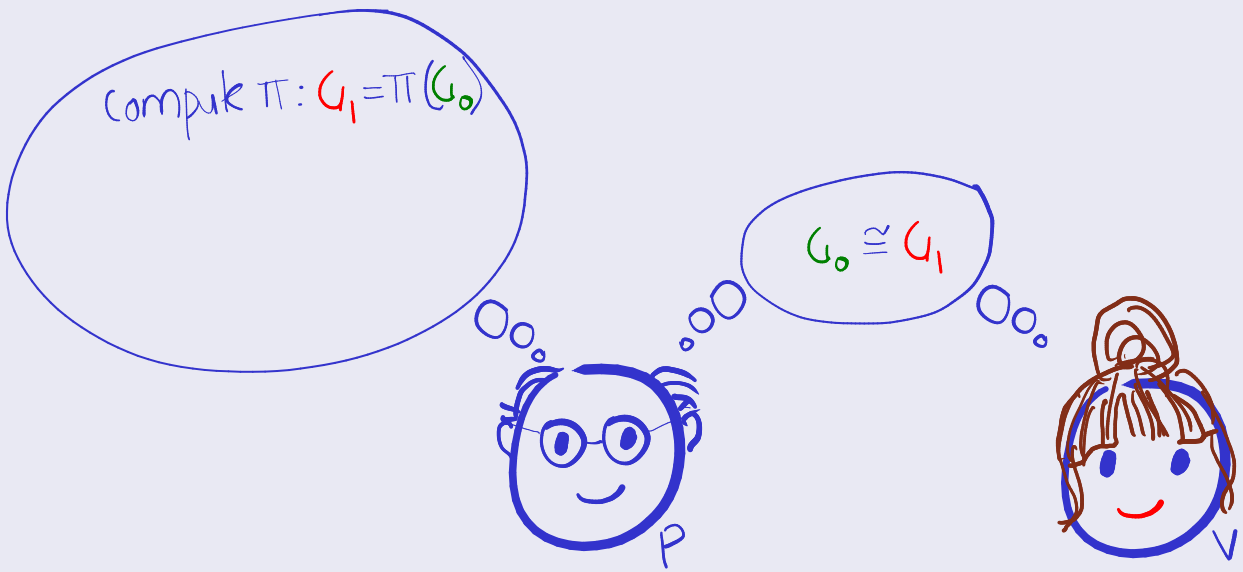
Honest-Verifier ZKP for GI...



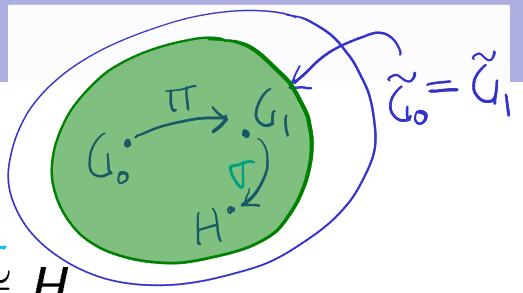
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



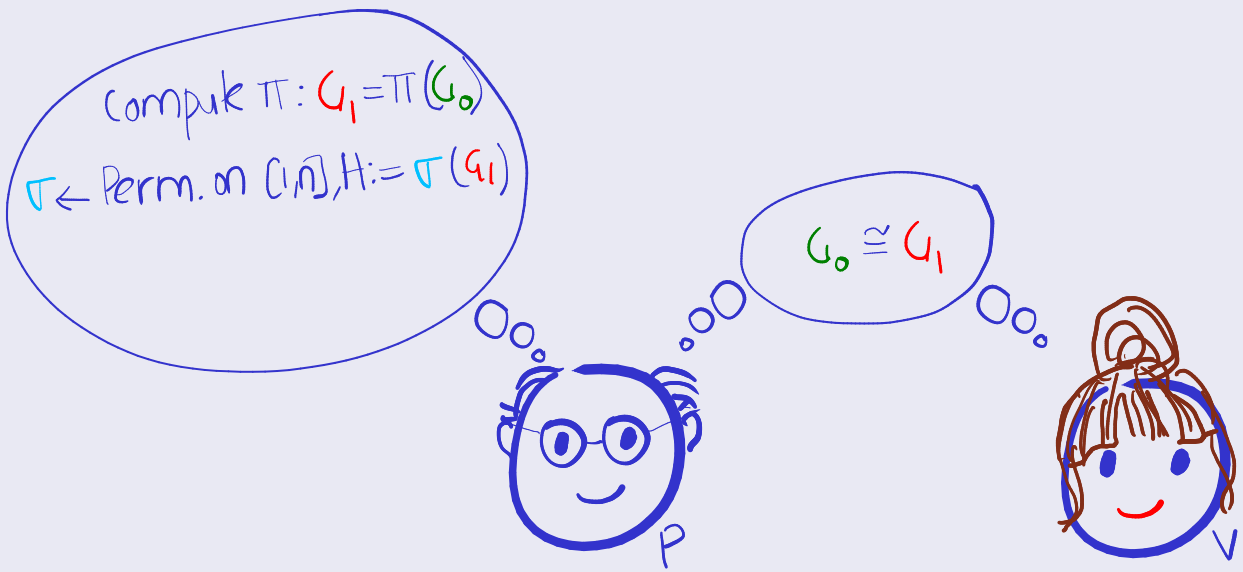
Honest-Verifier ZKP for GI...



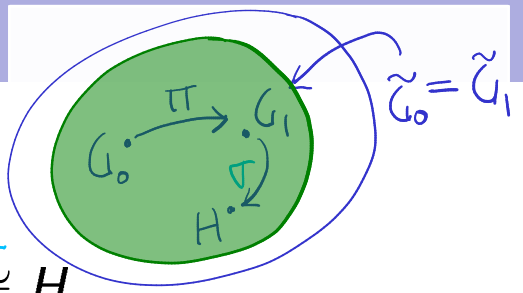
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



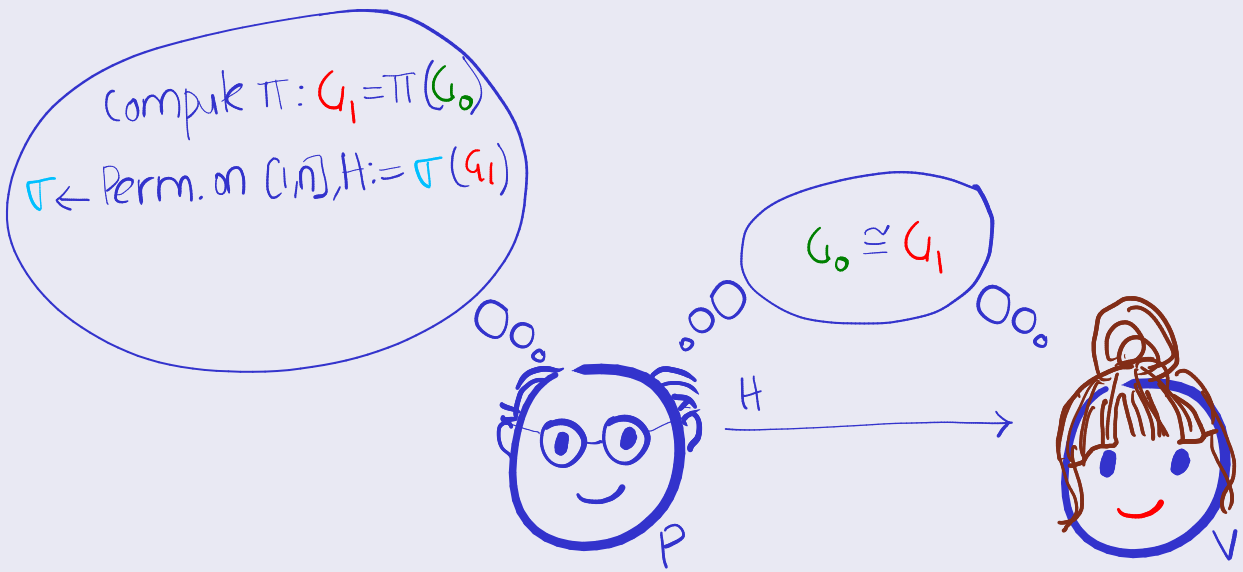
Honest-Verifier ZKP for GI...



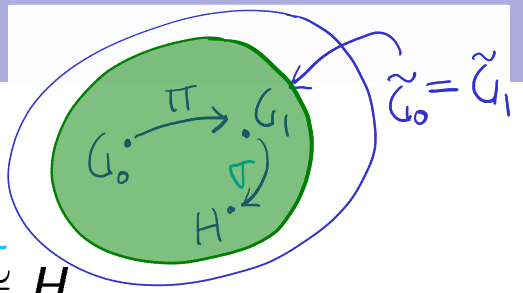
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



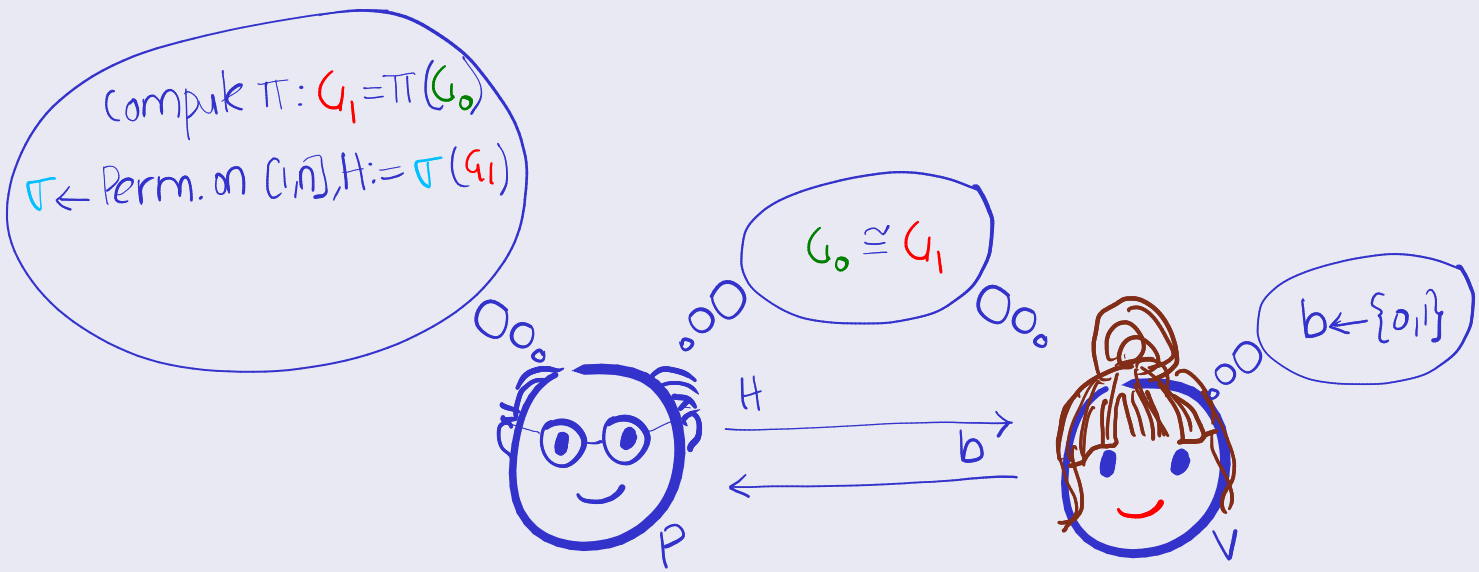
Honest-Verifier ZKP for GI...



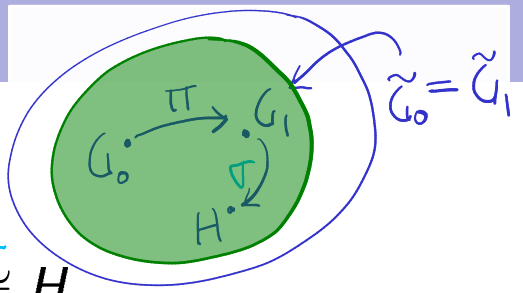
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



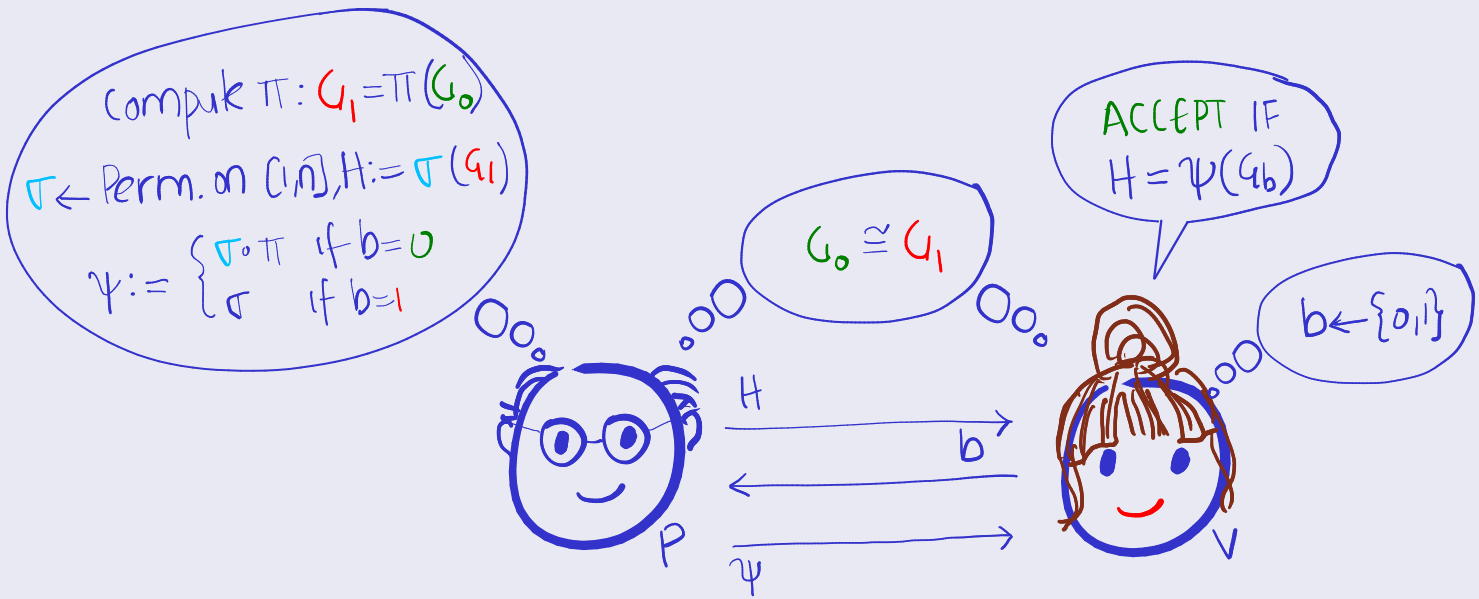
Honest-Verifier ZKP for GI...



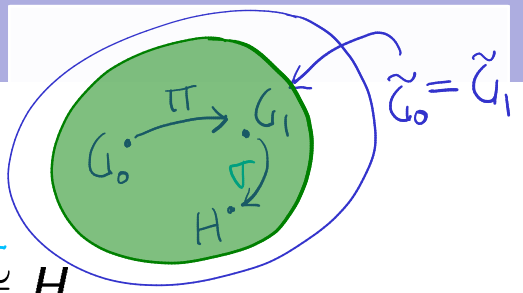
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



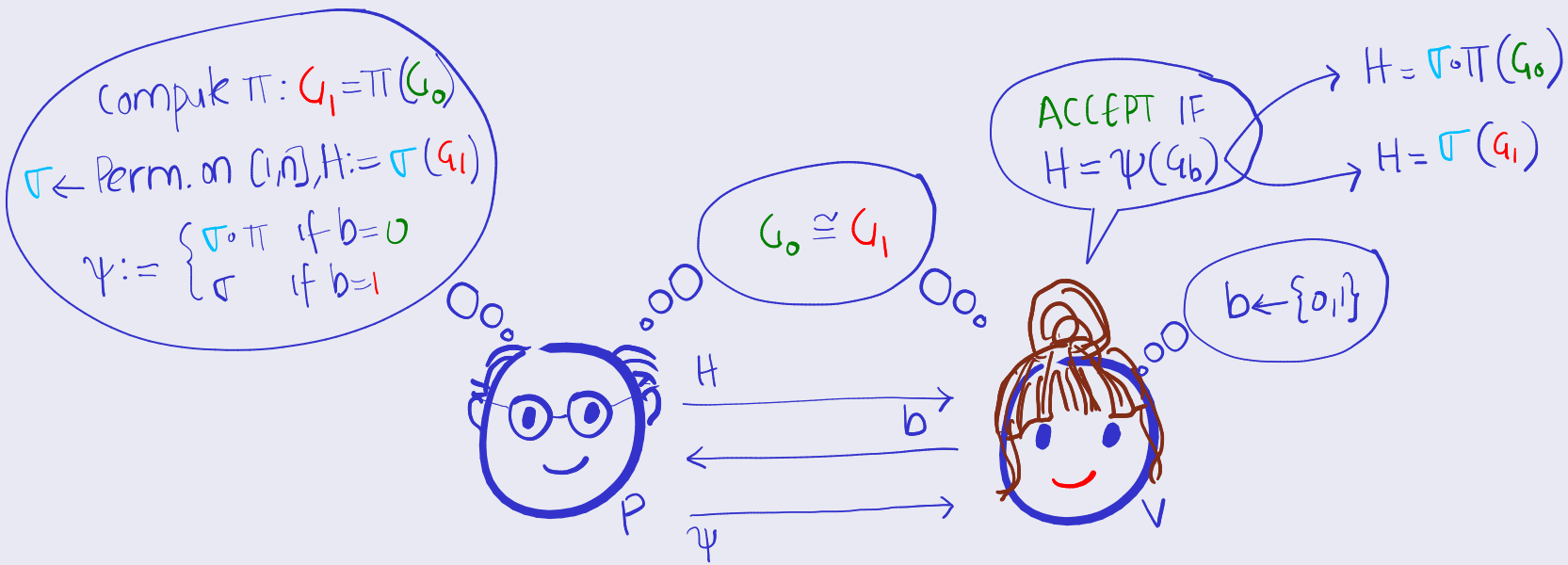
Honest-Verifier ZKP for GI...



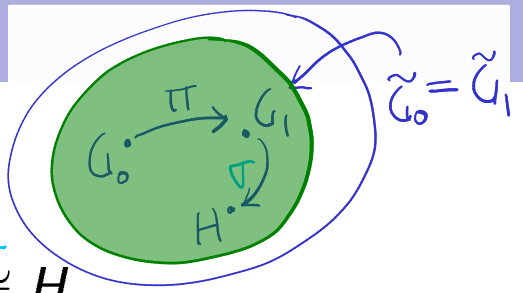
Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



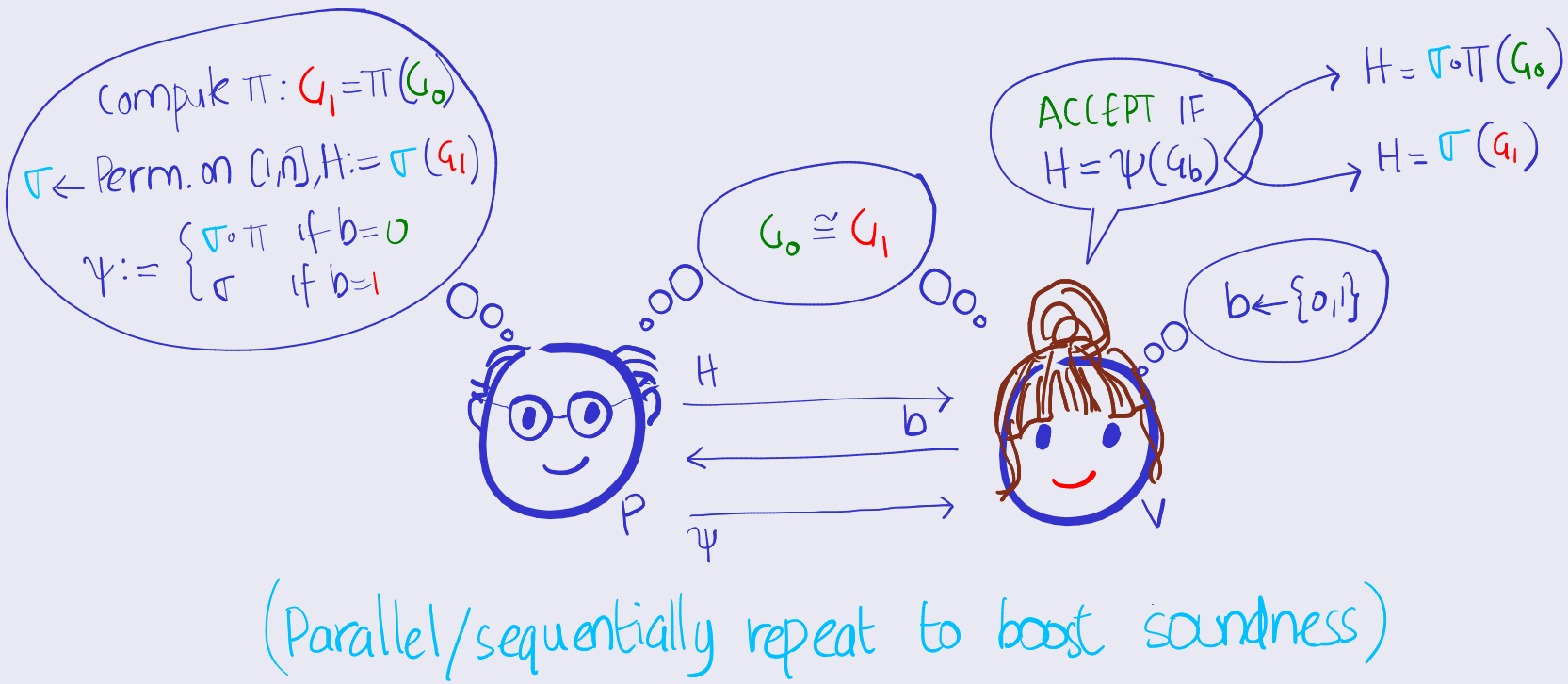
Honest-Verifier ZKP for GI...



Idea for ZK: $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$

1. Prover "commits" by sending random H s.t. $G_1 \overset{\sigma}{\cong} H$
2. Verifier challenges to "open" $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



Honest-Verifier ZKP for GI...

Theorem 2

Π_{GI} is honest-verifier perfect zero-knowledge IP for $\mathcal{L}_{GI}$

Honest-Verifier ZKP for GI...

Theorem 2

Π_{GI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GI}$

Proof ( idea for ZK: out of order sampling).

- Completeness: $G_0 \cong G_1 \Rightarrow$ P can answer both challenges $\Rightarrow$ V always accepts
- Soundness: $G_0 \ncong G_1 \Rightarrow$ for *any* H P* commits to, $G_0 \cong H$ *and* $G_1 \cong H$ *cannot both hold* $\Rightarrow$ best P* can do is guess b

Honest-Verifier ZKP for GI...

Theorem 2

Π_{GI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GI}$

Proof (idea for ZK: out of order sampling).

- Completeness: $G_0 \cong G_1 \Rightarrow$ P can answer both challenges $\Rightarrow$ V always accepts
- Soundness: $G_0 \ncong G_1 \Rightarrow$ for *any* H P^* commits to, $G_0 \cong H$ *and* $G_1 \cong H$ *cannot both hold* $\Rightarrow$ best P^* can do is guess b
- Zero knowledge:

$$\text{view}_V(\langle P, V\rangle(G_0, G_1)) := \left((G_0, G_1), (H, b, \psi)\right)$$

$$\forall\, G_0 \overset{\pi}{\cong} G_1 :$$

Theorem 2

Π_{GI} is honest-verifier perfect zero-knowledge IP for $\mathcal{L}_{GI}$

Proof (idea for ZK: out of order sampling).

- Completeness: $G_0 \cong G_1 \Rightarrow$ P can answer both challenges $\Rightarrow$ V always accepts
- Soundness: $G_0 \ncong G_1 \Rightarrow$ for *any* H P* commits to, $G_0 \cong H$ *and* $G_1 \cong H$ *cannot both hold* $\Rightarrow$ best P* can do is guess b
- Zero knowledge:

$$\mathrm{View}_V(\langle P,V\rangle(G_0,G_1)) := ((G_0,G_1),(H,b,\psi))$$

$H = \sigma(G_1)$, b = bit, $\psi = \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$

$\forall\, G_0 \overset{\pi}{\cong} G_1 :$

□

Honest-Verifier ZKP for GI...

Theorem 2

Π_{GI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GI}$

Proof (idea for ZK: out of order sampling).

- Completeness: $G_0 \cong G_1 \Rightarrow$ P can answer both challenges $\Rightarrow$ V always accepts
- Soundness: $G_0 \ncong G_1 \Rightarrow$ for *any* H P* commits to, $G_0 \cong H$ *and* $G_1 \cong H$ *cannot both hold* $\Rightarrow$ best P* can do is guess b
- Zero knowledge:

$$\text{view}_V(\langle P,V\rangle(G_0,G_1)) := \left((G_0,G_1),\left(H,b,\psi\right)\right)$$

$\psi(G_1)$, bit, $\begin{cases}\sigma\circ\pi & \text{if } b=0\\ \sigma & \text{if } b=1\end{cases}$

$\forall\, G_0 \overset{\pi}{\cong} G_1$: $\text{Sim}(G_0,G_1)$:

Honest-Verifier ZKP for GI...

Theorem 2

Π_{GI} *is honest-verifier perfect zero-knowledge IP for* $\mathcal{L}_{GI}$

Proof (idea for ZK: out of order sampling).

- Completeness: $G_0 \cong G_1 \Rightarrow$ P can answer both challenges $\Rightarrow$ V always accepts
- Soundness: $G_0 \ncong G_1 \Rightarrow$ for *any* H P* commits to, $G_0 \cong H$ *and* $G_1 \cong H$ *cannot both hold* $\Rightarrow$ best P* can do is guess b
- Zero knowledge:

$\pi(G_1)$, bit

$$\text{view}_V(\langle P,V\rangle(G_0,G_1)) := \left((G_0,G_1),\left(H,b,\psi\right)\right) \quad \begin{cases} \pi\circ\psi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$$

$\forall\ G_0 \overset{\pi}{\cong} G_1$: $\text{sim}(G_0,G_1)$: sample $b \leftarrow \{0,1\}$, $\psi \leftarrow$ permutation on $[1,n]$

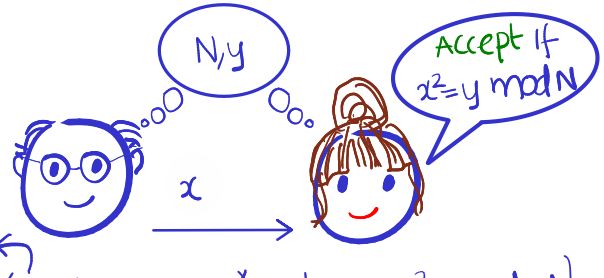
set $H := \psi(G_b)$

o/p $\left((G_0,G_1),\left(H,b,\psi\right)\right)$

$\forall\ G_0 \cong G_1$: $\text{view}_V(\langle P,V\rangle(G_0,G_1))$ identically distributed to $\text{sim}(G_0,G_1)$. □

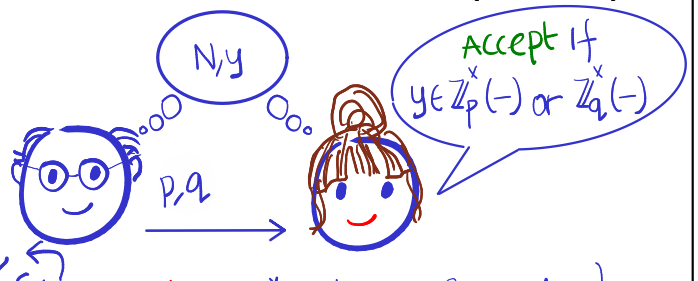
Which Languages have ZKPs?

Quadratic residuosity (QR)



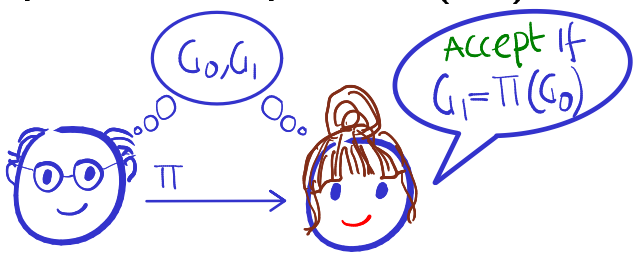
$$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^{\times} \text{ s.t. } y = x^2 \bmod N\}$$

Quad. non-residuosity (QNR)



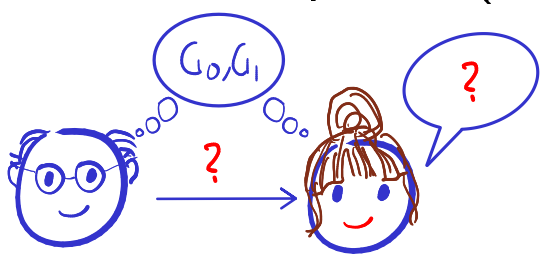
$$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^{\times} \text{ s.t. } y = x^2 \bmod N\}$$

Graph isomorphism (GI)



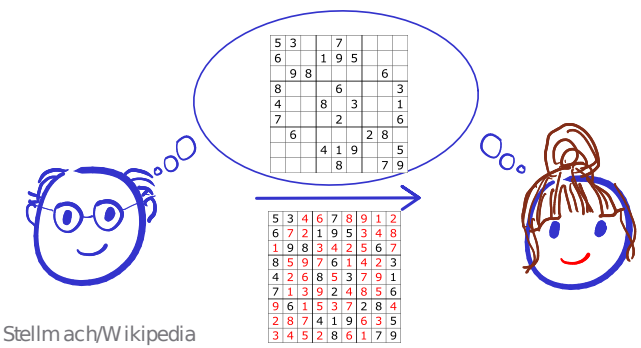
$$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Graph non-isomorphism (GNI)



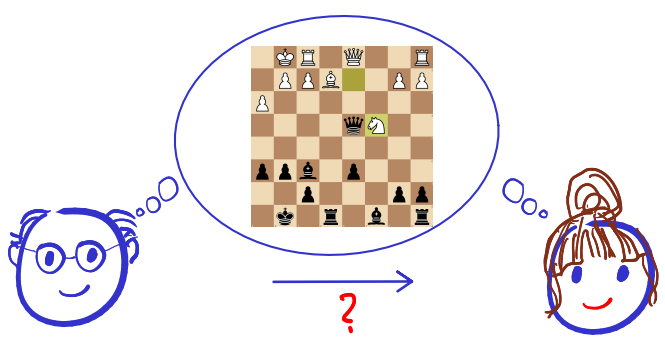
$$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Sudoku



© Tim Stellmach/Wikipedia

Chess



© Cburnett/Wikipedia

Which Languages have ZKPs?

Quadratic residuosity (QR)

$$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Quad. non-residuosity (QNR)

$$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

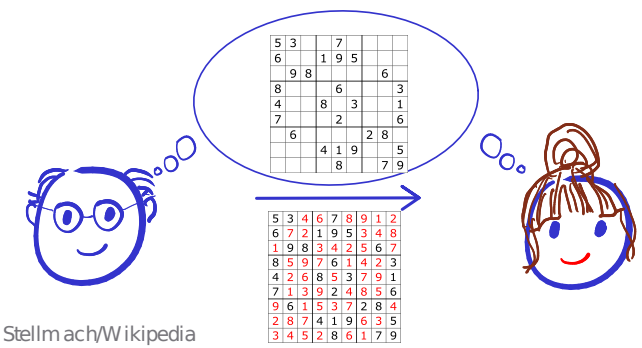
Graph isomorphism (GI)

$$\mathcal{L}_{GI} = \{(G_0,G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Graph non-isomorphism (GNI)

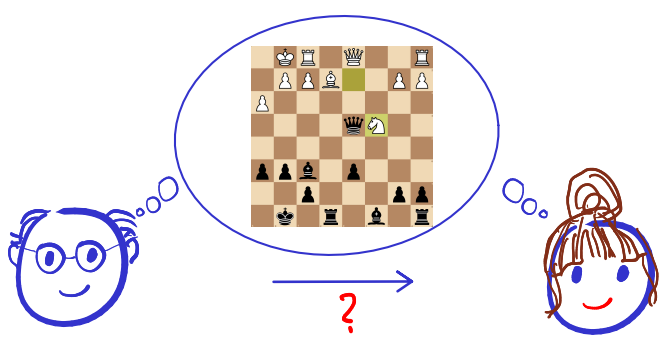
$$\mathcal{L}_{GNI} = \{(G_0,G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Sudoku



© Tim Stellmach/Wikipedia

Chess



L19

Which Languages have ZKPs? PSPACE Languages

Quadratic residuosity (QR)

$$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Quad. non-residuosity (QNR)

$$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

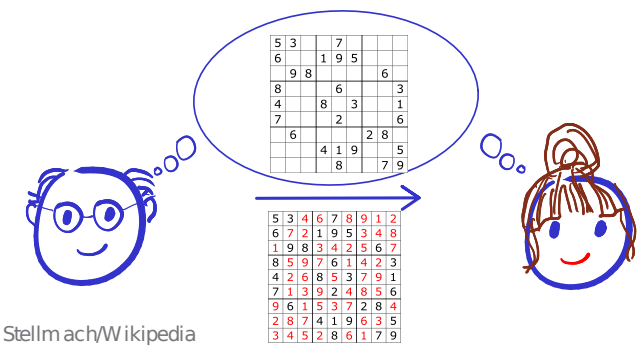
Graph isomorphism (GI)

$$\mathcal{L}_{GI} = \{(G_0,G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Graph non-isomorphism (GNI)

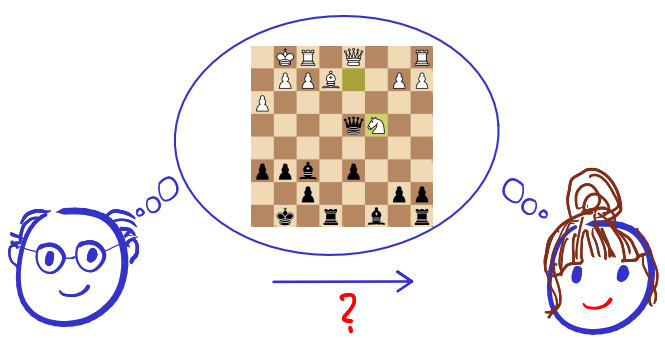
$$\mathcal{L}_{GNI} = \{(G_0,G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Sudoku



© Tim Stellmach/Wikipedia

Chess



L19

Are Randomness and Interaction Necessary?

$\leftrightarrows$ Interaction is necessary

Fact 3

If $\mathcal{L}$ has a non-interactive (i.e, one-message) ZKP then $\mathcal{L}$ is "trivial"

Are Randomness and Interaction Necessary?

$\leftrightarrows$ Interaction is necessary

Fact 3

If $\mathcal{L}$ has a non-interactive (i.e, one-message) ZKP then $\mathcal{L}$ is "trivial"

Randomness is necessary

Exercise 3

If $\mathcal{L}$ has an IP with deterministic verifier then $\mathcal{L} \in$ NP

Fact 4

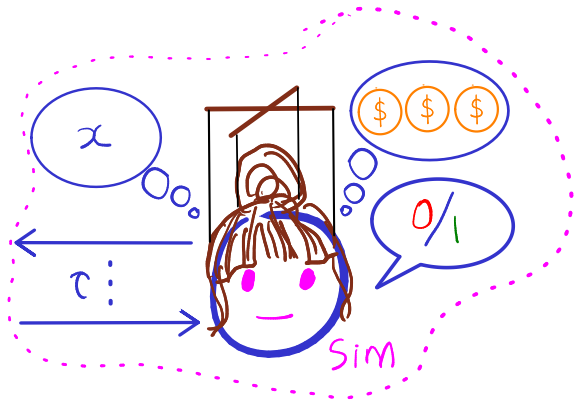
If $\mathcal{L}$ has an ZKP with deterministic verifier then $\mathcal{L}$ is "trivial"

Plan for Today's Lecture...

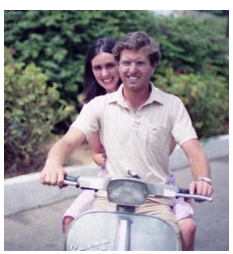
IP where prover reveals *no non-trivial knowledge* to the verifier

NEW Zero-knowledge (ZK) IP

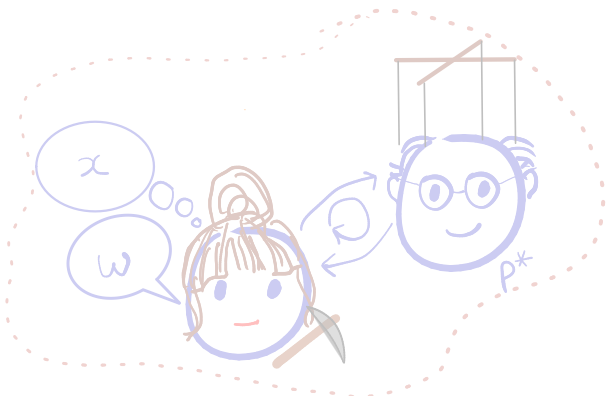
ZK Proof of Knowledge (PoK) NEW



© alchetron.com



©Oded Goldreich



©Konrad Jacobs/Wikipedia

Main tools: simulator and extractor

Sometimes Stronger Guarantees than ZK Needed

- Recall ZK IP requirements:
  1. Completeness
  2. Soundness
  3. Zero-knowledge (ZK)

Sometimes Stronger Guarantees than ZK Needed

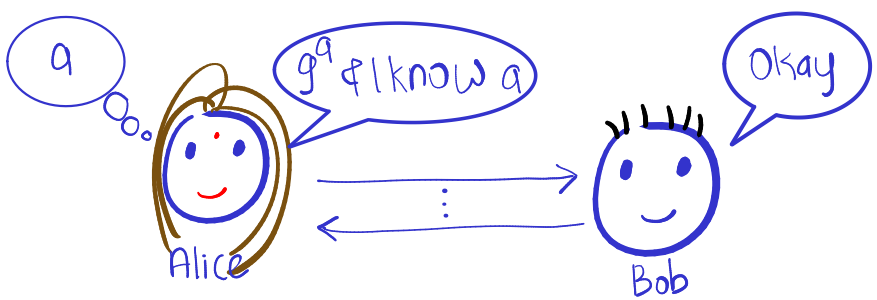
- Recall ZK IP requirements:
  1. Completeness
  2. Soundness
  3. Zero-knowledge (ZK)

- Sometimes V needs to be convinced that P *knows* a witness

Sometimes Stronger Guarantees than ZK Needed

- Recall ZK IP requirements:
  1. Completeness
  2. Soundness
  3. Zero-knowledge (ZK)

- Sometimes V needs to be convinced that P *knows* a witness
- E.g. Identification for ElGamal PKE in cyclic group $\mathbb{G}$
  - Public key is $h := g^a$ and secret key is the discrete log a
  - Owner has to prove they *possess* a (such an a always exists)



How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via simulator: V's view can be *efficiently simulated* given only the instance x

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via simulator: V's view can be *efficiently simulated* given only the instance x

How would you quantify "knowledge" itself?

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via simulator: V's view can be *efficiently simulated* given only the instance x

How would you quantify "knowledge" itself?

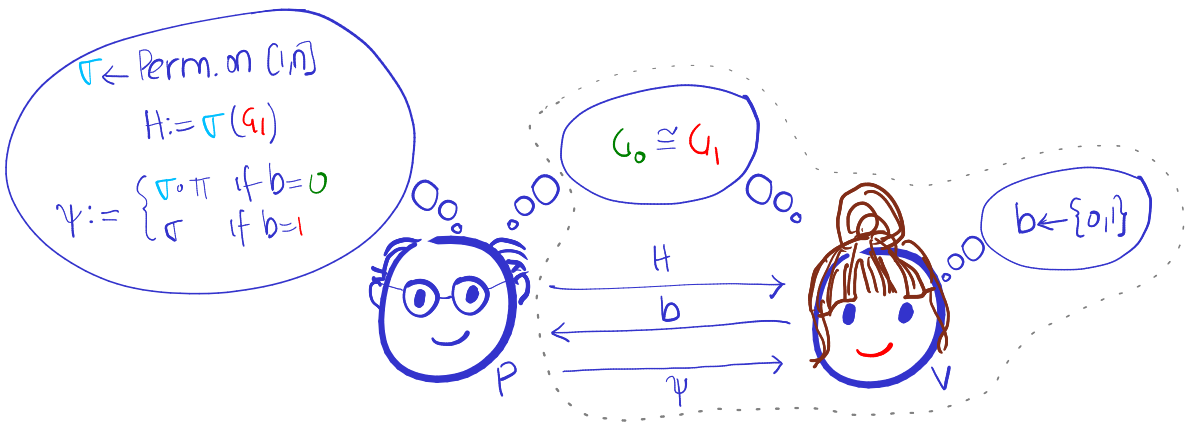
  - For a student: get hold of student, hold viva, extract answers

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via simulator: V's view can be *efficiently simulated* given only the instance x

❓ How would you quantify "knowledge" itself?

  - For a student: get hold of student, hold viva, extract answers



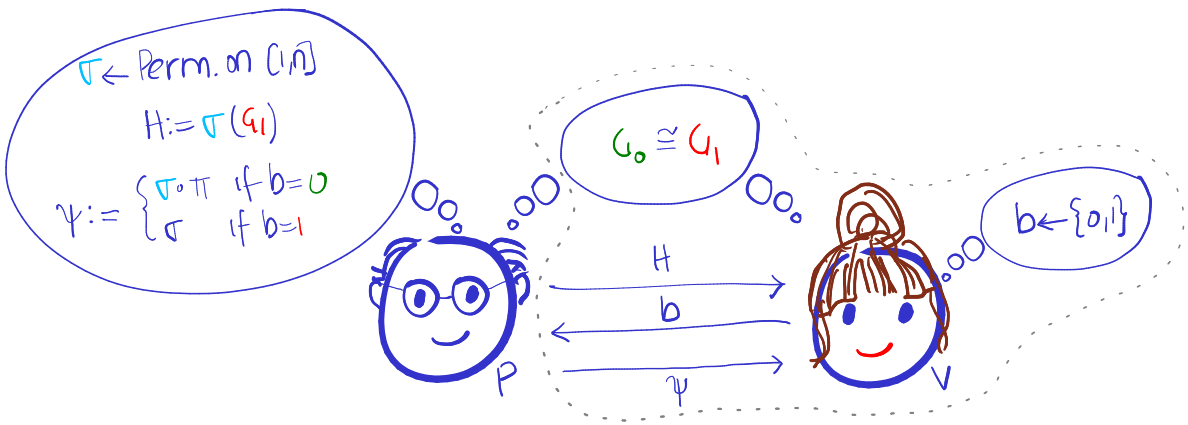
  - For P in Π_{GI}?

How to Quantify Knowledge?

- For defining ZK, we only quantified "gain of knowledge"
  - "V gains no knowledge" if anything V can *compute* after the interaction with P, it could have computed *without it*
  - Formalised via simulator: V's view can be *efficiently simulated* given only the instance x

❓ How would you quantify "knowledge" itself?

  - For a student: get hold of student, hold viva, extract answers



  - For P in Π_{GI}? Should be possible to *efficiently extract* isomorphism π given access to P
  - In general, for NP: should be possible to extract a witness w

Let's Define ZK *Proof of Knowledge*

Definition 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for an NP language $\mathcal{L}$ is a zero-knowledge *proof of knowledge* if it is

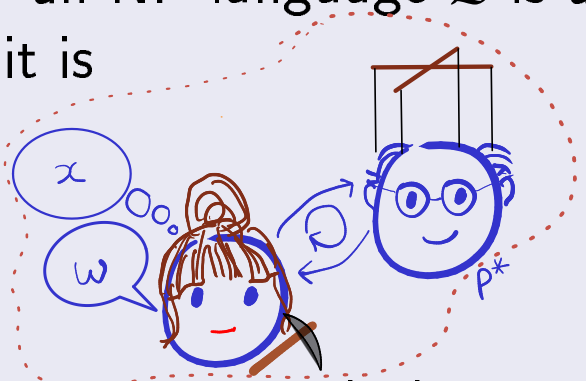
1. Complete
2. Zero knowledge

Let's Define ZK *Proof of Knowledge*

Definition 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for an NP language $\mathcal{L}$ is a zero-knowledge *proof of knowledge* if it is

1. Complete
2. Zero knowledge
3. Knowledge sound:
   - $\exists$ expected polynomial-time *extractor* Ext such that
   - $\forall$ prover P^* and instance x:



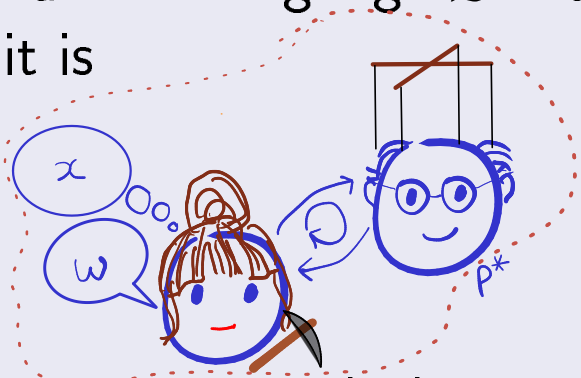
$$\Pr_{w \leftarrow \mathsf{Ext}^{\mathbf{P}^*}(x)}[w \text{ is a witness for } x] \geq \Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(x)] - \epsilon_k$$

Let's Define ZK *Proof of Knowledge*

Definition 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for an NP language $\mathcal{L}$ is a zero-knowledge *proof of knowledge* if it is

1. Complete
2. Zero knowledge
3. Knowledge sound:
   - $\exists$ expected polynomial-time *extractor* Ext such that
   - $\forall$ prover P^* and instance x:



"knowledge error"

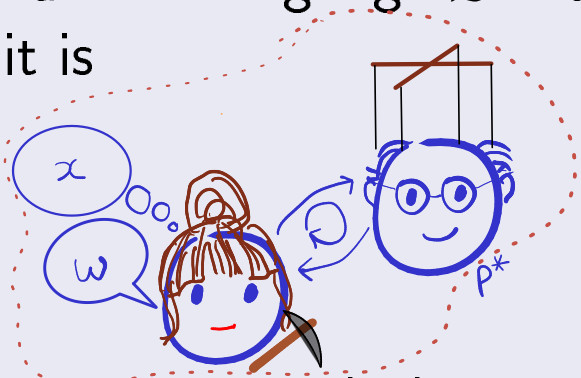
$$\Pr_{w \leftarrow \mathsf{Ext}^{\mathbf{P}^*}(x)}[w \text{ is a witness for } x] \geq \Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(x)] - \epsilon_k$$

Let's Define ZK *Proof of Knowledge*

Definition 2 (ZKPoK)

An interactive protocol $\Pi = (\mathsf{P}, \mathsf{V})$ for an NP language $\mathcal{L}$ is a zero-knowledge *proof of knowledge* if it is

1. Complete
2. Zero knowledge
3. Knowledge sound:
   - $\exists$ expected polynomial-time *extractor* Ext such that
   - $\forall$ prover P^* and instance x:



"knowledge error"

$$\Pr_{w \leftarrow \mathsf{Ext}^{\mathbf{P}^*}(x)}[w \text{ is a witness for } x] \geq \Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(x)] - \epsilon_k$$

- Trivial if we omit either of requirement 2 or 3

Ext must do something *more* than V, e.g. "rewind" P^*

Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

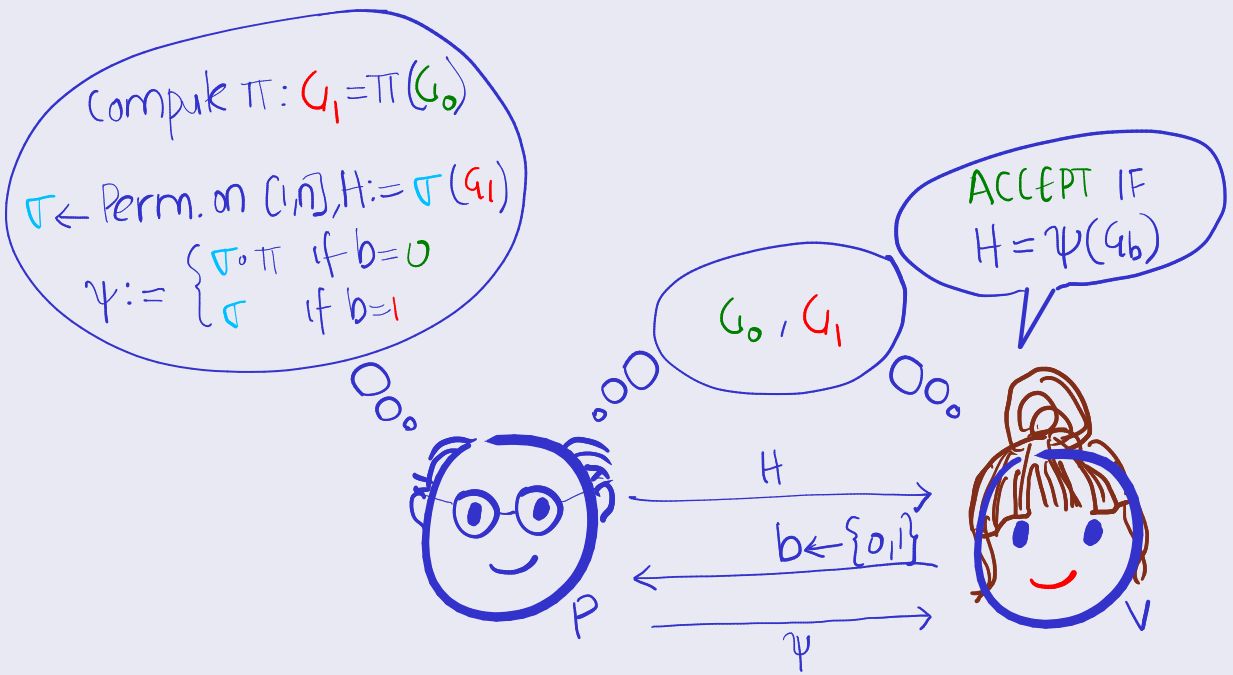
Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

response to challenge 1 ↖ ↗ response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.



□

Π_{GI} is ZKPoK: How to Extract π?

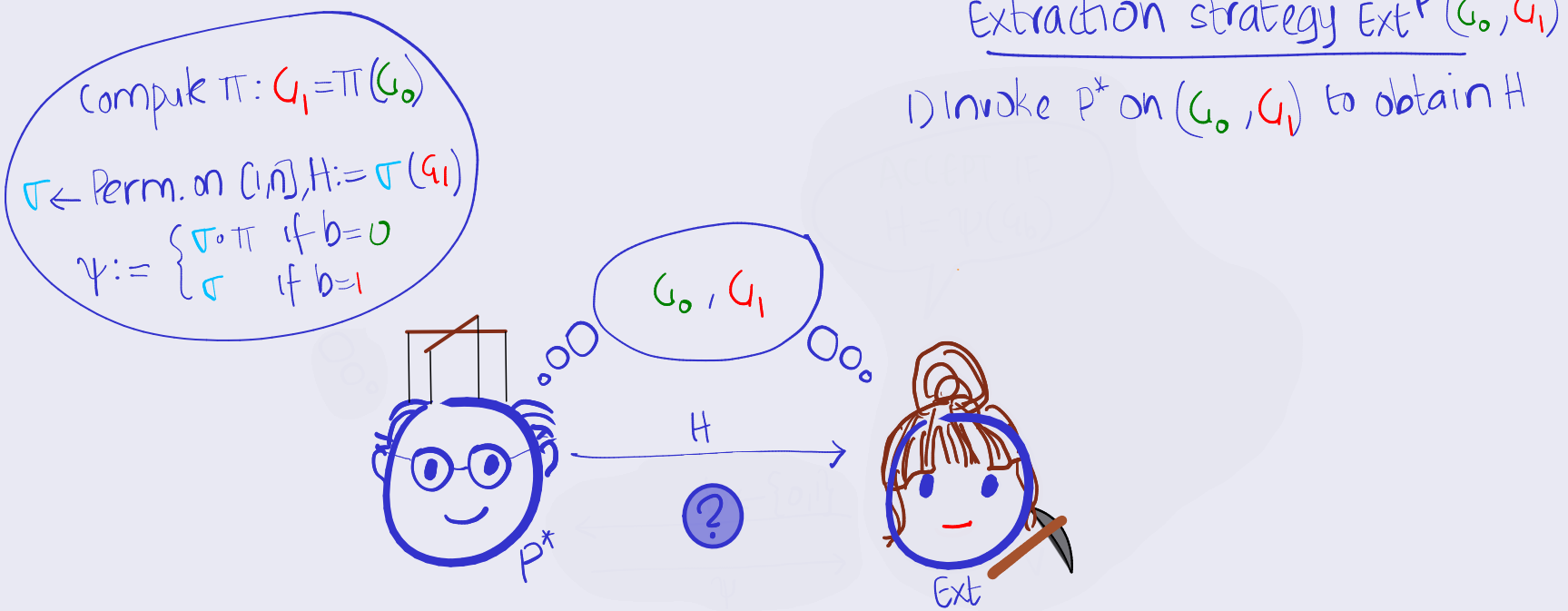
Theorem 5

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

Proof (of PoK)

response to challenge 1 ↖ ↗ response to challenge 0

Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.



Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

response to challenge 1 ↖ ↗ response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

Compute $\pi : G_1 = \pi(G_0)$

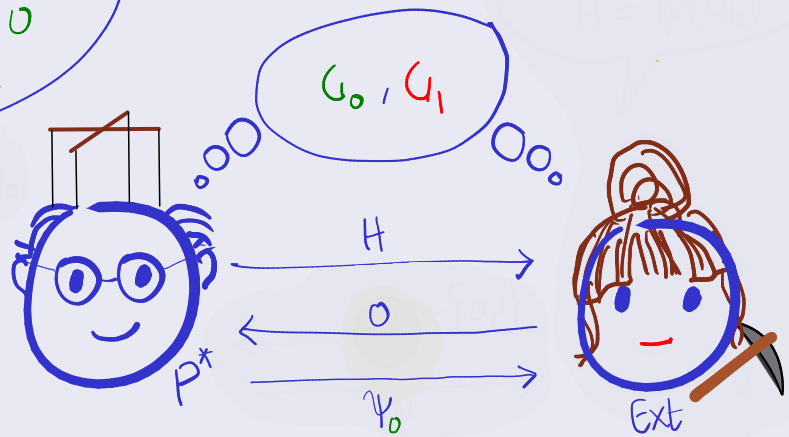
$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$

Extraction strategy $\text{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H

2) Challenge on 0 to get ψ_0



□

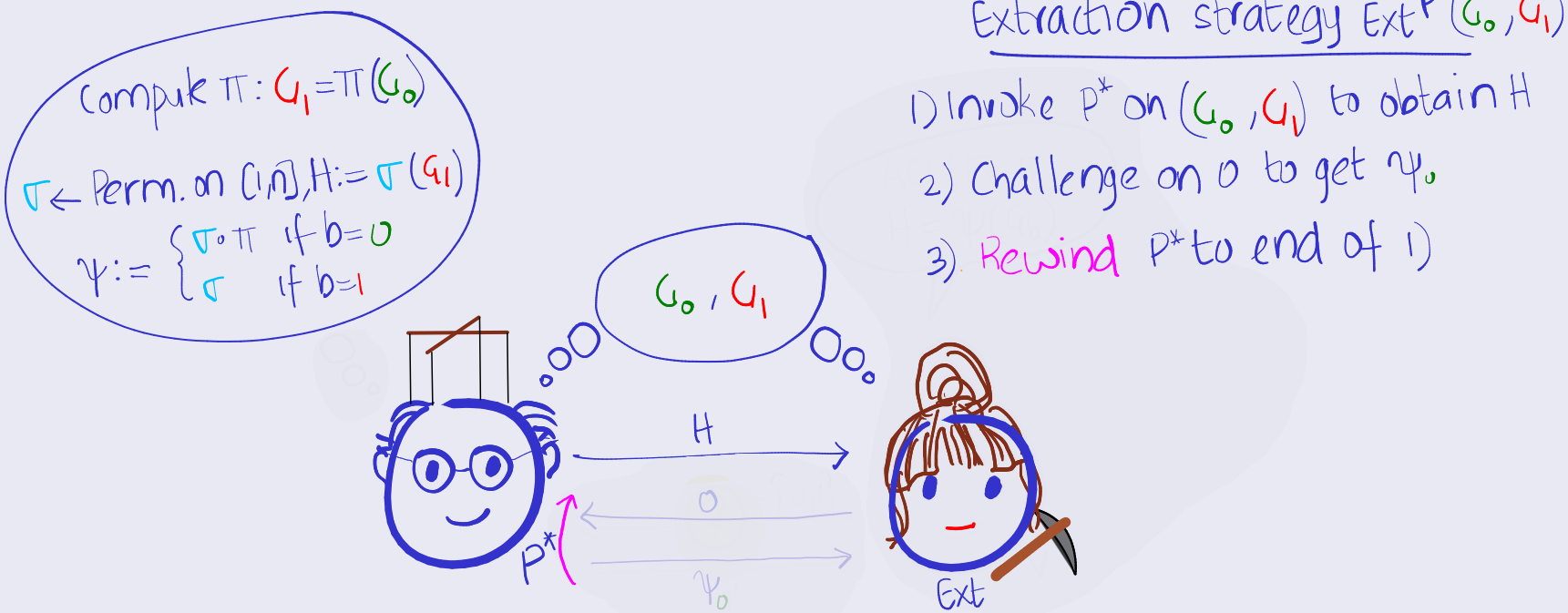
Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} *is a ZKPoK for* $\mathcal{L}_{GI}$ *with* $\epsilon_k \leq 1/2$

response to challenge 1 ↖ ↗ response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.



Extraction strategy $\text{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)

□

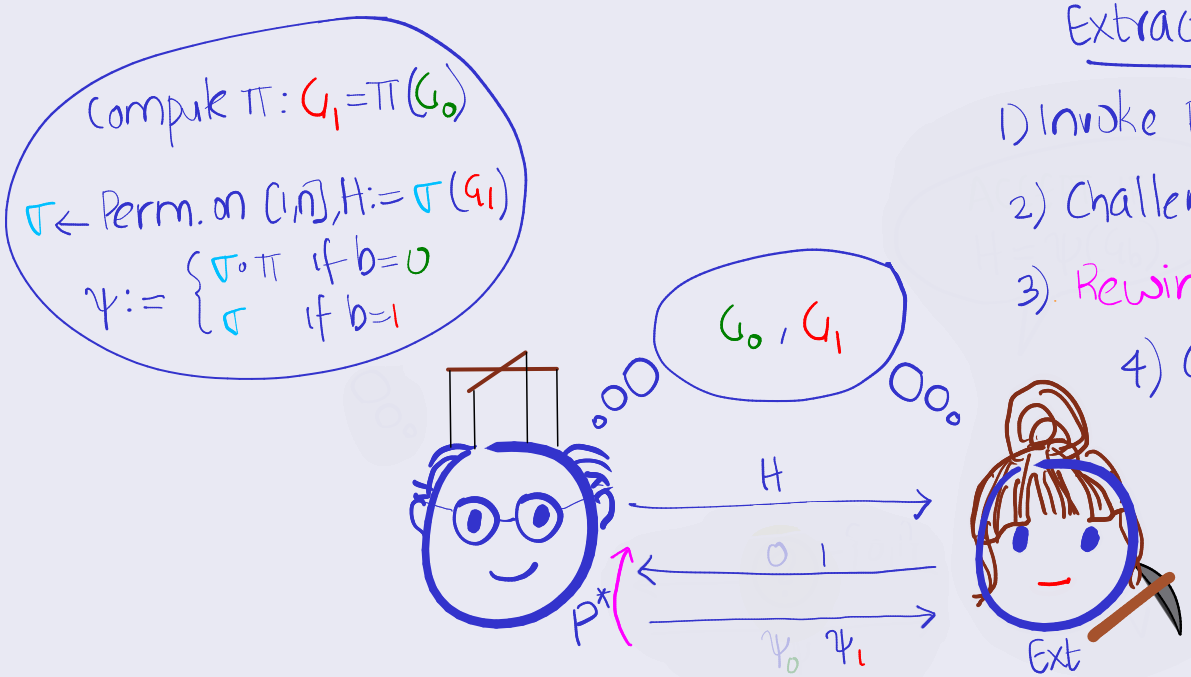
Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

response to challenge 1 ↖ ↗ response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.



Extraction strategy $\text{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)
4) Challenge on 1 to get ψ_1

□

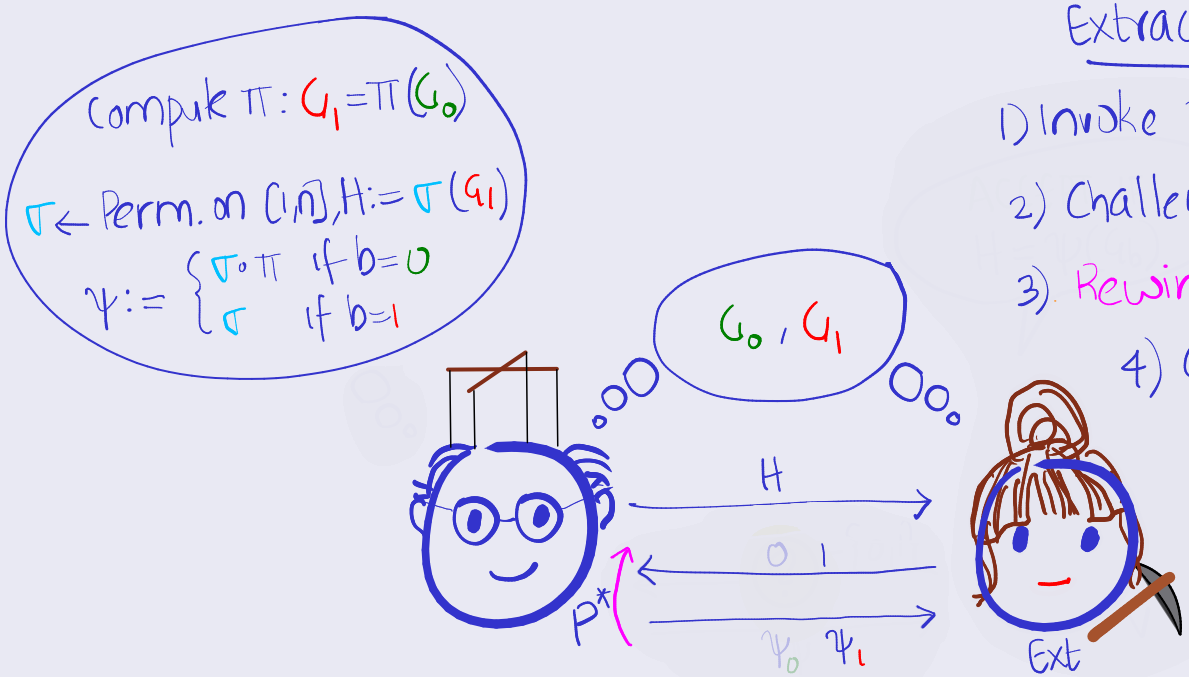
Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} is a ZKPoK for $\mathcal{L}_{GI}$ with $\epsilon_k \leq 1/2$

response to challenge 1 ↖ ↗ response to challenge 0

Proof (of PoK) Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.



Extraction strategy $\text{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)
4) Challenge on 1 to get ψ_1
5) Output $\psi_1^{-1} \circ \psi_0$

□

Π_{GI} is ZKPoK: How to Extract π?

Theorem 5

Π_{GI} *is a ZKPoK for* $\mathcal{L}_{GI}$ *with* $\epsilon_k \leq 1/2$

Proof (of PoK)

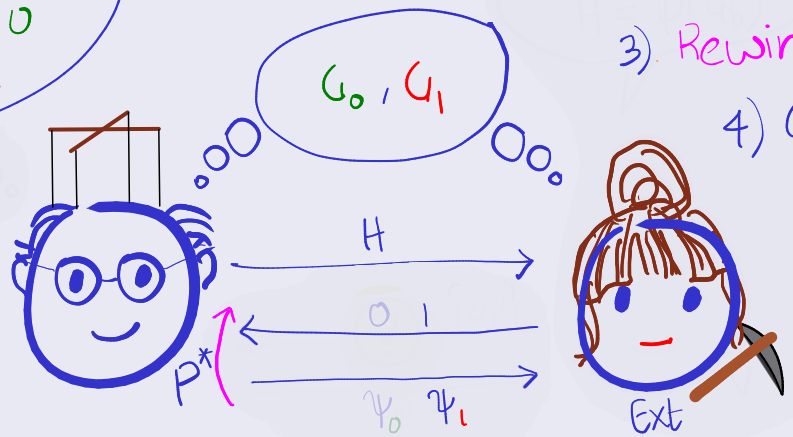
response to challenge 1 ↖ ↗ response to challenge 0

Hint $\psi_1^{-1} \circ \psi_0 = \sigma^{-1} \circ \sigma \circ \pi = \pi$.

Compute π: $G_1 = \pi(G_0)$

$\sigma \leftarrow$ Perm. on $[1,n]$, $H := \sigma(G_1)$

$\psi := \begin{cases} \sigma \circ \pi & \text{if } b=0 \\ \sigma & \text{if } b=1 \end{cases}$



Extraction strategy $\text{Ext}^{P^*}(G_0, G_1)$

1) Invoke P^* on (G_0, G_1) to obtain H
2) Challenge on 0 to get ψ_0
3) Rewind P^* to end of 1)
4) Challenge on 1 to get ψ_1
5) Output $\psi_1^{-1} \circ \psi_0$

V accepts 2) & 4) $\Rightarrow H = \psi_0(G_0) = \psi_1(G_1) \Rightarrow G_1 = \psi_1^{-1} \circ \psi_0(G_0)$ □

ZKPoK for DLP: Schnorr's Protocol...

Definition 3 (Lecture 12, DLP in prime-order $\mathbb{G}$ w.r.to S)

- Input:
  1. $(\mathbb{G}, \ell, g)$ sampled by a group sampler $\mathsf{S}(1^n)$
  2. $h := g^a$ for $a \leftarrow \mathbb{Z}_\ell$
- Solution: a

ZKPoK for DLP: Schnorr's Protocol...

Definition 3 (Lecture 12, DLP in prime-order $\mathbb{G}$ w.r.to S)

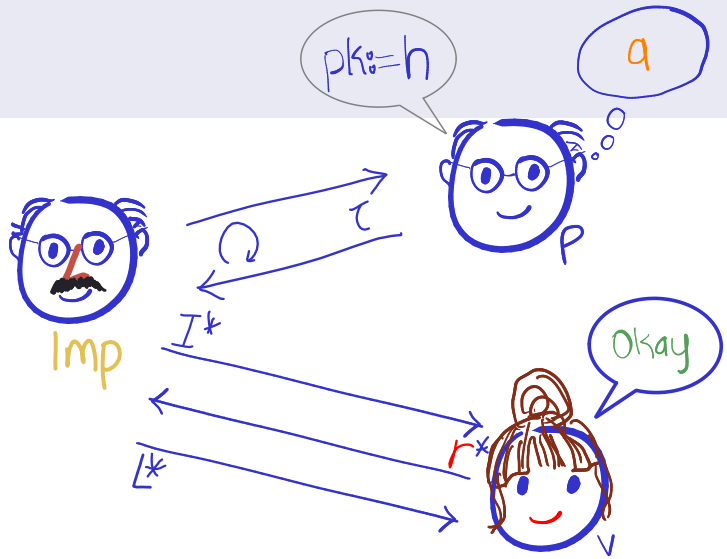
- Input:
  1. $(\mathbb{G}, \ell, g)$ sampled by a group sampler $\mathsf{S}(1^n)$
  2. $h := g^a$ for $a \leftarrow \mathbb{Z}_\ell$
- Solution: a

- ElGamal PKE:
  - Public key: $h := g^a$
  - Secret key: a

ZKPoK for DLP: Schnorr's Protocol...

Definition 3 (Lecture 12, DLP in prime-order $\mathbb{G}$ w.r.to S)

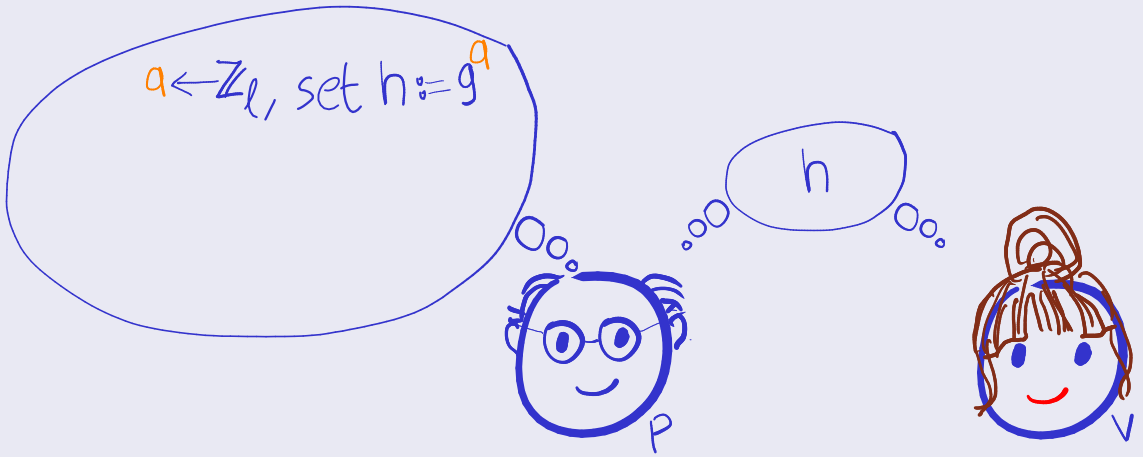
- Input:
  1. $(\mathbb{G}, \ell, g)$ sampled by a group sampler $S(1^n)$
  2. $h := g^a$ for $a \leftarrow \mathbb{Z}_\ell$
- Solution: a



- ElGamal PKE:
  - Public key: $h := g^a$
  - Secret key: a
- In ID protocol for ElGamal PK, the impostor (who doesn't know a):
  - May see several transcripts via Auth_{sk} oracle
  - Should not be able to fool the verifier into accepting *in the protocol*

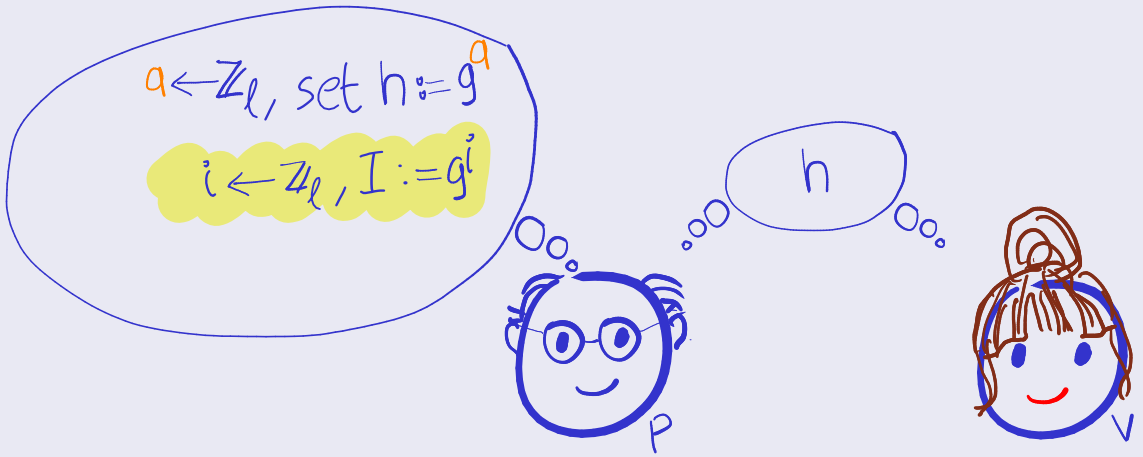
ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



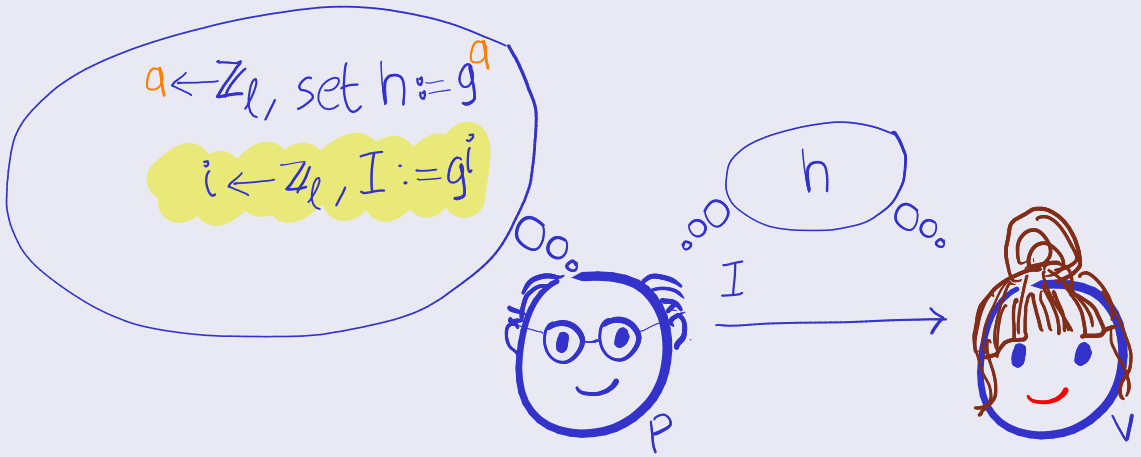
ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



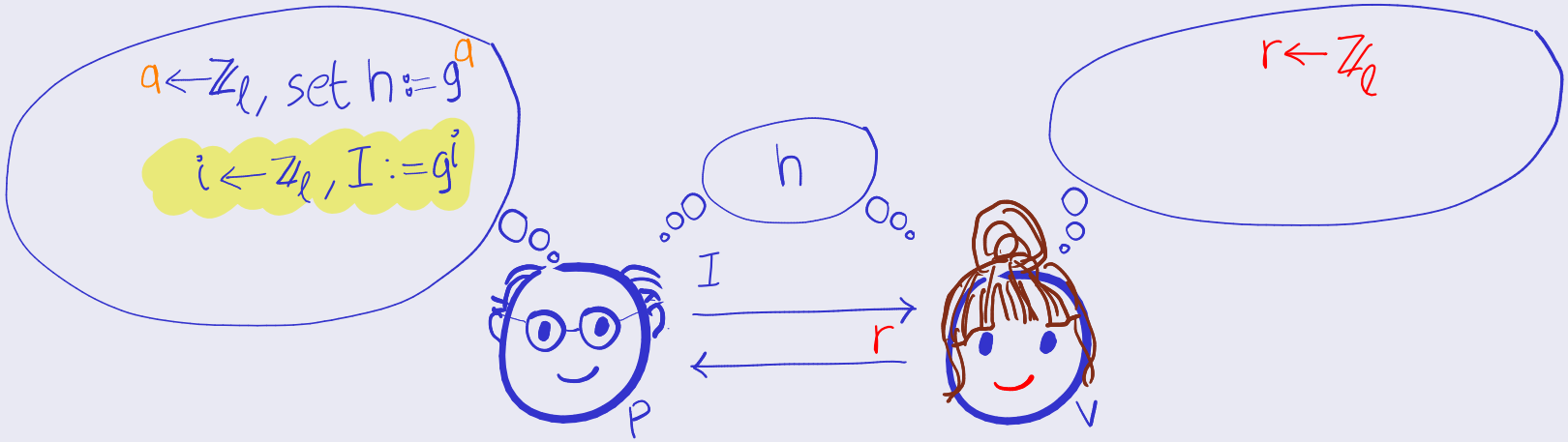
ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



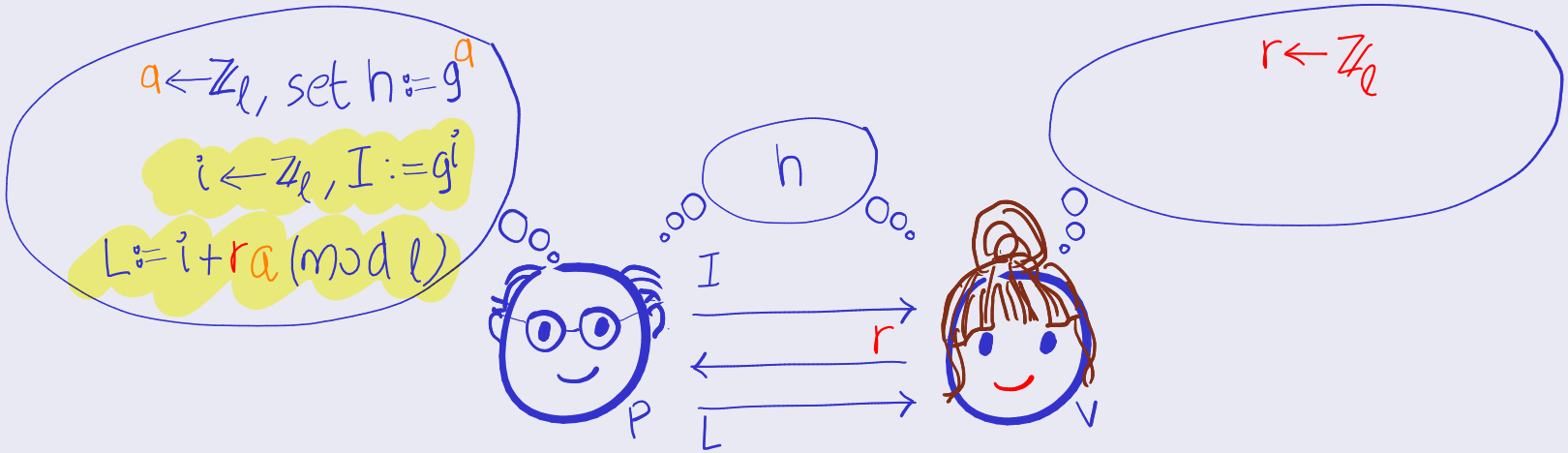
ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



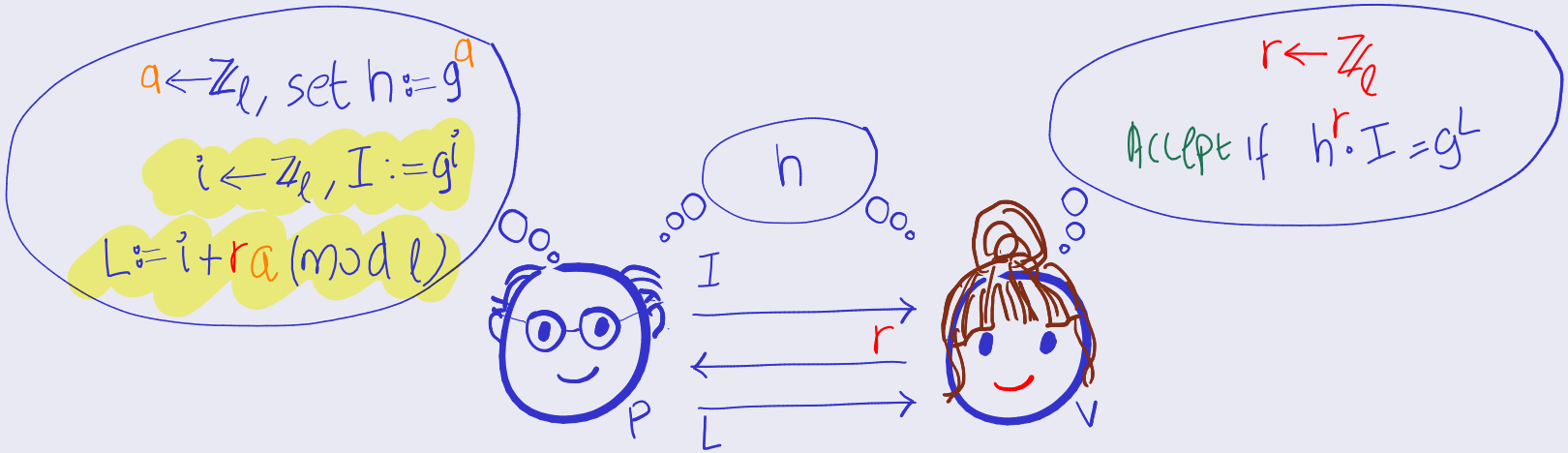
ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



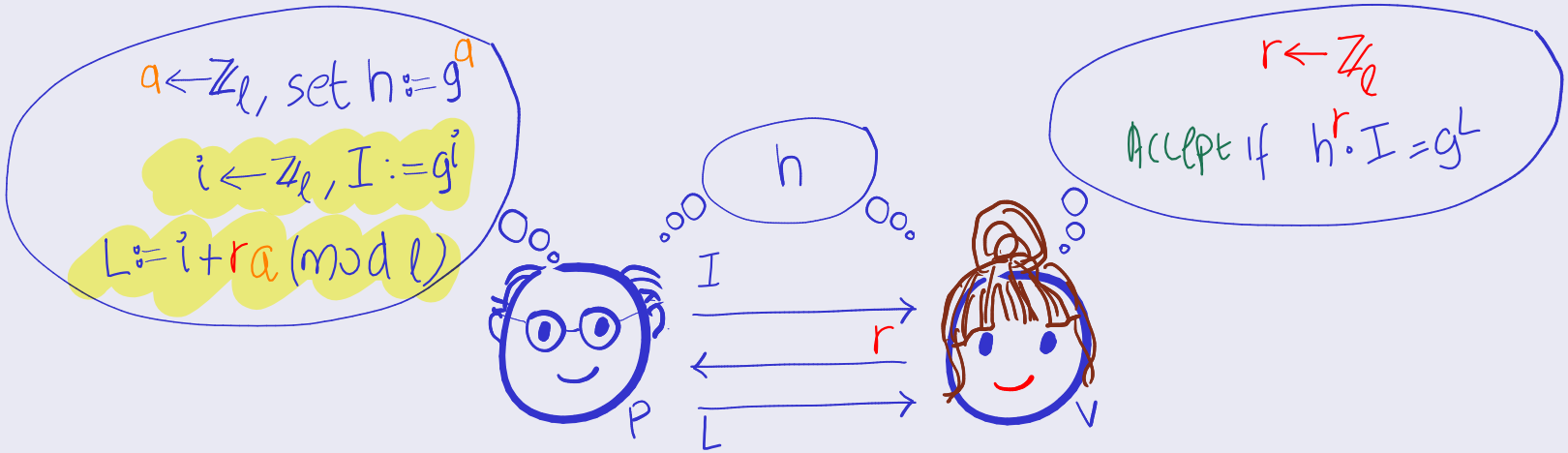
ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



ZKPoK for DLog: Schnorr's Protocol...

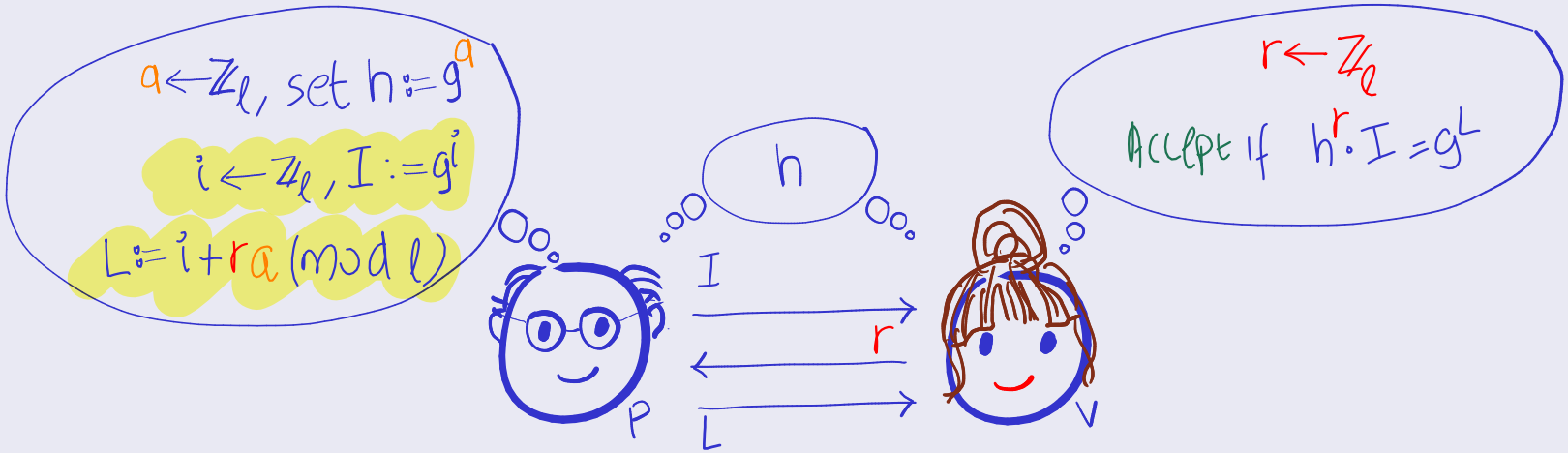
Protocol 3 (Π_{DLP}: Schnorr's protocol)



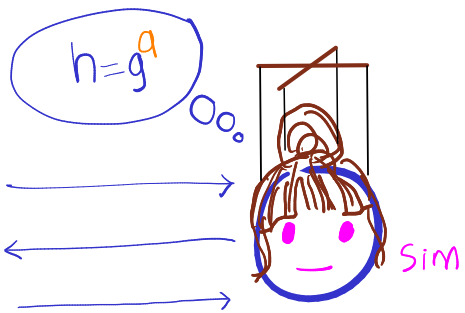
- Completeness: $h^r \cdot I = (g^a)^r \cdot g^i = g^{ar+i} = g^L$ (by group axioms)

ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



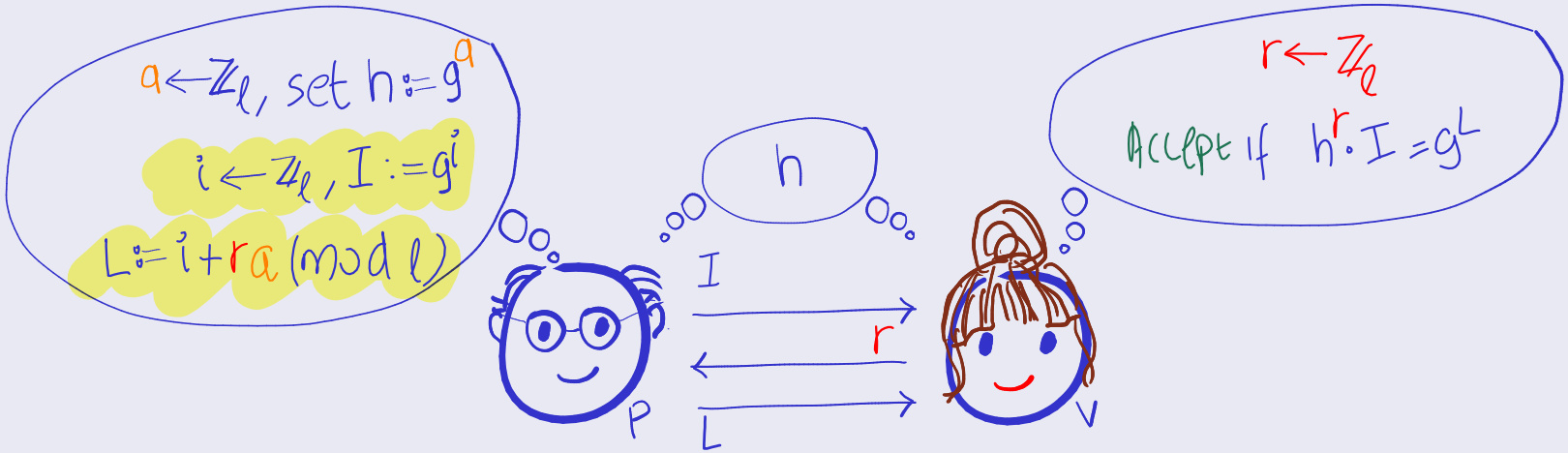
- Completeness: $h^r \cdot I = (g^a)^r \cdot g^i = g^{ar+i} = g^L$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again



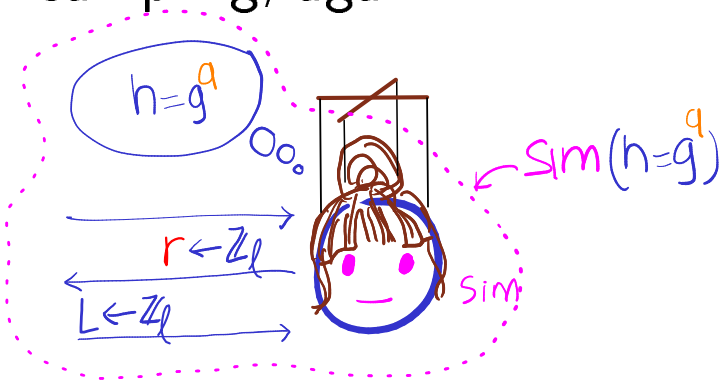
- Allows simulation of transcripts in ID protocol

ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



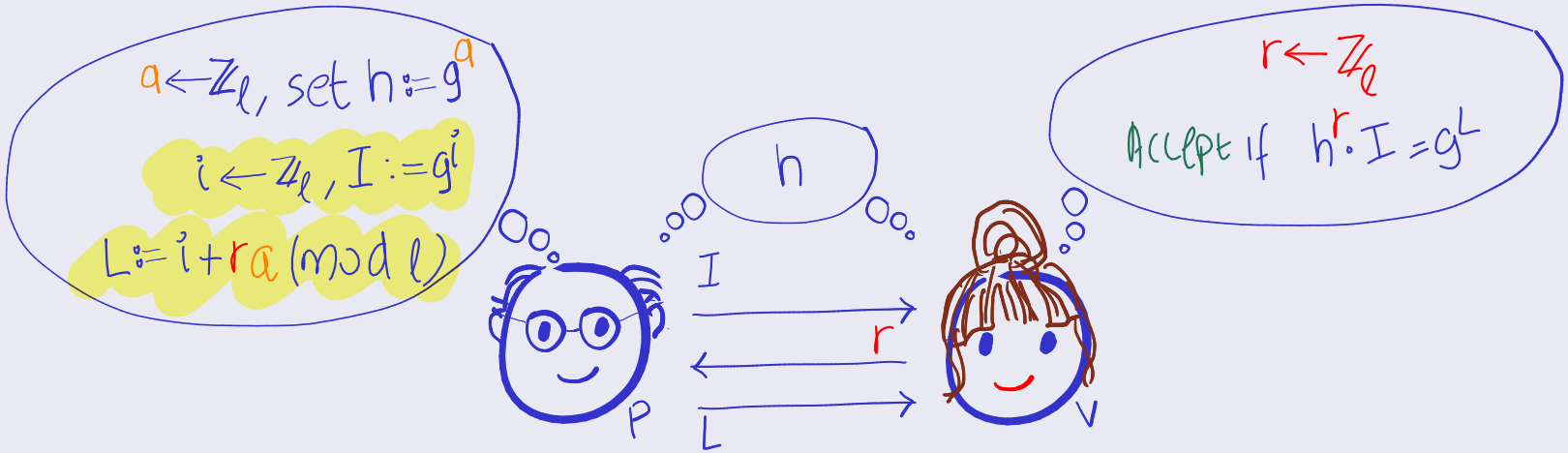
- Completeness: $h^r \cdot I = (g^a)^r \cdot g^i = g^{ar+i} = g^L$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again



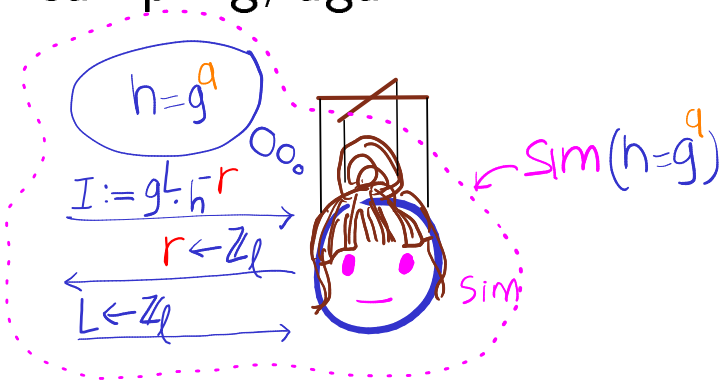
- Allows simulation of transcripts in ID protocol

ZKPoK for DLog: Schnorr's Protocol...

Protocol 3 (Π_{DLP}: Schnorr's protocol)



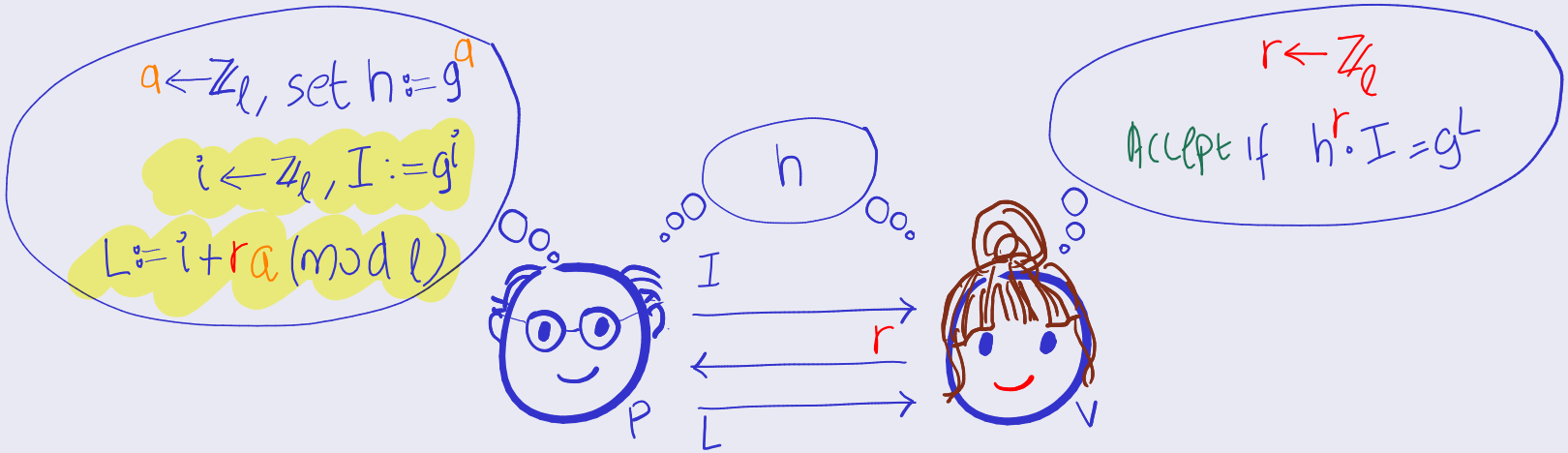
- Completeness: $h^r \cdot I = (g^a)^r \cdot g^i = g^{ar+i} = g^L$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again



- Allows simulation of transcripts in ID protocol

ZKPoK for DLog: Schnorr's Protocol...

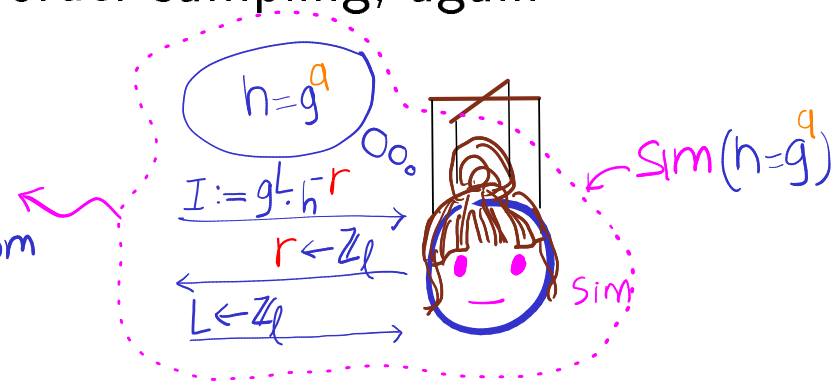
Protocol 3 (Π_{DLP}: Schnorr's protocol)



- Completeness: $h^r \cdot I = (g^a)^r \cdot g^i = g^{ar+i} = g^L$ (by group axioms)
- Honest-verifier ZK: out of order sampling, again

Distributed identically to view_V since

$g^L \cdot h^{-r} = g^{L-ar}$ is random



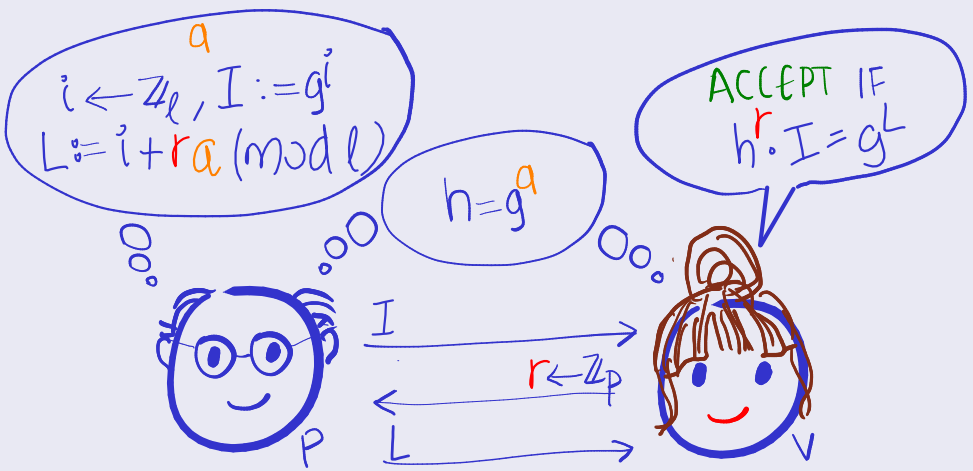
- Allows simulation of transcripts in ID protocol

How to Extract a from P^*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.

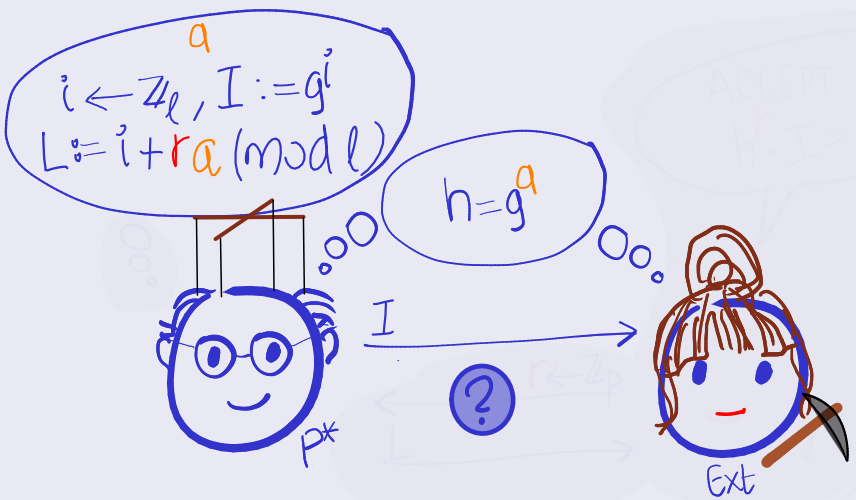


How to Extract a from P*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

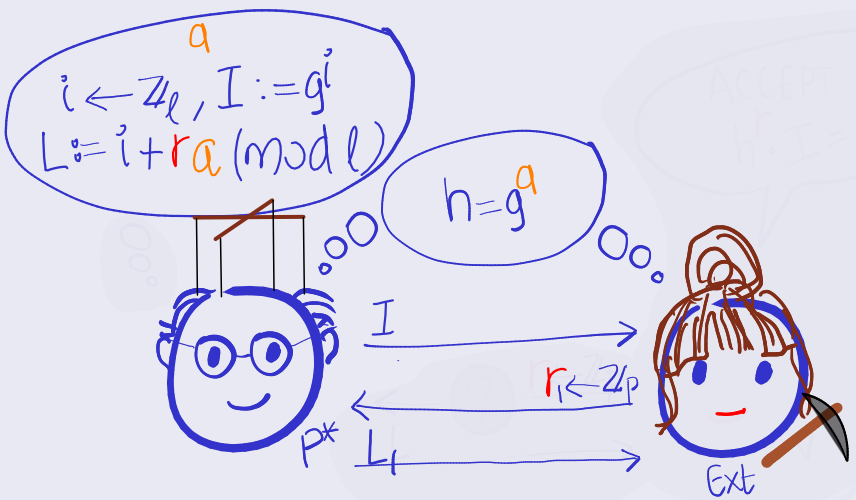
1) Invoke P^* on h to obtain I

How to Extract a from P^*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

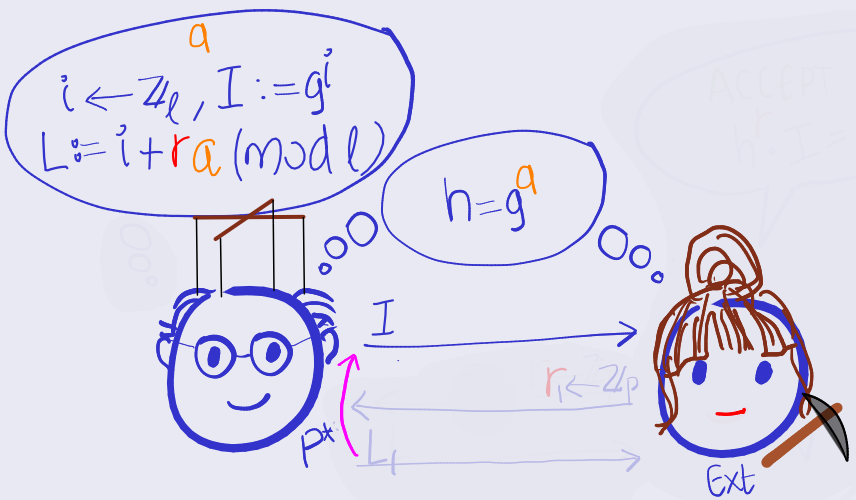
1) Invoke P^* on h to obtain I
2) Challenge on $r_1 \leftarrow Z_\ell$ to get L_1

How to Extract a from P^*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

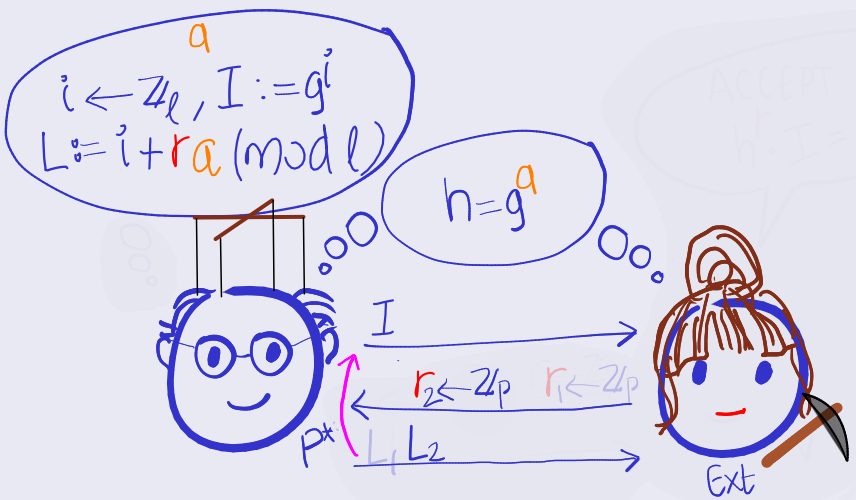
1) Invoke P^* on h to obtain I
2) Challenge on $r \leftarrow Z_\ell$ to get L
3) Rewind P to 1)

How to Extract a from P^*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

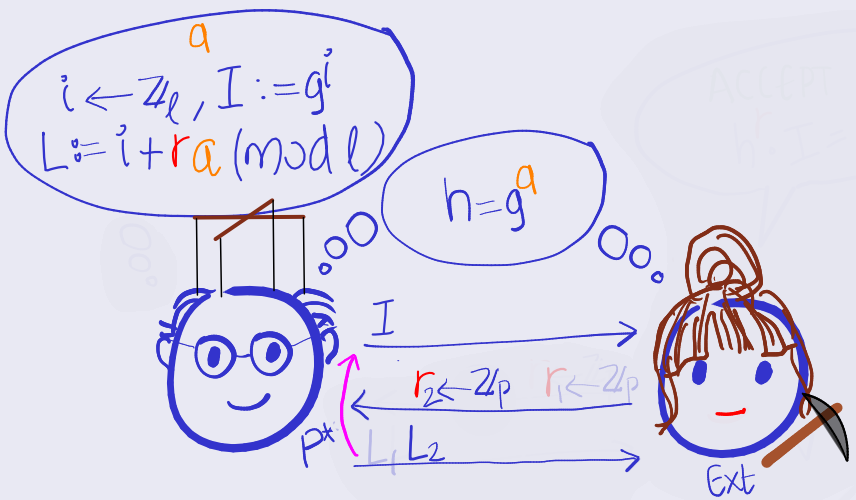
1) Invoke P^* on h to obtain I
2) Challenge on $r_1 \leftarrow Z_\ell$ to get L_1
3) Rewind P to 1)
4) Challenge on $r_2 \leftarrow Z_p$ to get L_2

How to Extract a from P*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

1) Invoke P^* on h to obtain I
2) Challenge on $r_1 \leftarrow Z_\ell$ to get L_1
3) Rewind P to 1)
4) Challenge on $r_2 \leftarrow Z_p$ to get L_2
5) Output $L_1 - L_2 / r_1 - r_2$

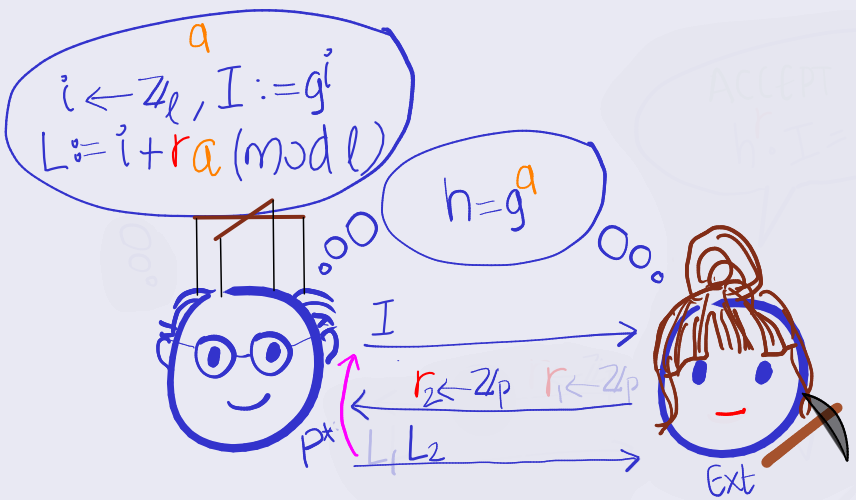
V accepts both executions $\Rightarrow$ $g^{L_1} = I \cdot h^{r_1}$ & $g^{L_2} = I \cdot h^{r_2}$ $\Rightarrow$ $g^{L_1 - L_2} = h^{r_1 - r_2}$ $\Rightarrow$ $a = \frac{L_1 - L_2}{r_1 - r_2}$

How to Extract a from P*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

1) Invoke P^* on h to obtain I
2) Challenge on $r_1 \leftarrow Z_\ell$ to get L_1
3) Rewind P to 1)
4) Challenge on $r_2 \leftarrow Z_p$ to get L_2
5) Output $L_1 - L_2 / r_1 - r_2$

V accepts both executions $\Rightarrow$ $g^{L_1} = I \cdot h^{r_1}$ & $g^{L_2} = I \cdot h^{r_2}$ $\Rightarrow$ $g^{L_1 - L_2} = h^{r_1 - r_2}$ $\Rightarrow$ $a = \frac{L_1 - L_2}{r_1 - r_2}$

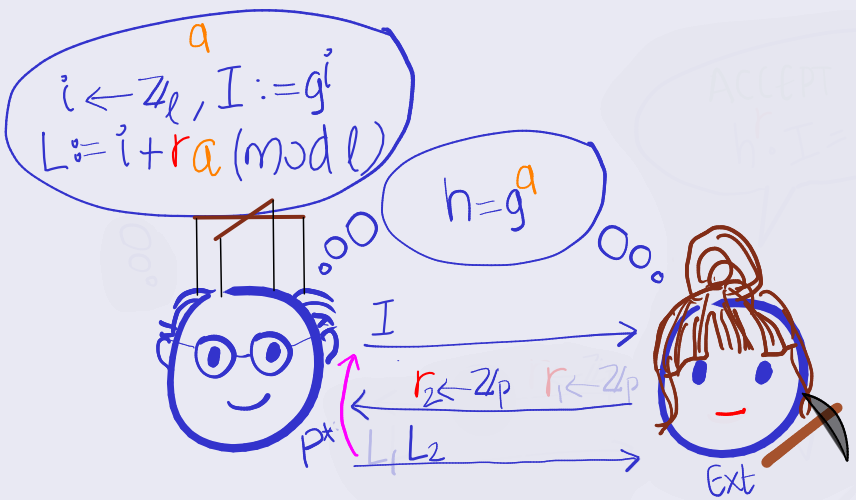
Extraction error $1/p$ $\Leftarrow$ Fails if $r_1 = r_2$

How to Extract a from P*?

Theorem 6

Π_{DLP} *is a PoK for* $\mathcal{L}_{DLP}$ *with* $\epsilon_k \leq 1/p$

Proof (of PoK) Hint Obtain two eqns of form $L = I + ra \bmod \ell$.



Extraction strategy $Ext^{P^*}(h)$

1) Invoke P^* on h to obtain I
2) Challenge on $r_1 \leftarrow Z_\ell$ to get L_1
3) Rewind P to 1)
4) Challenge on $r_2 \leftarrow Z_p$ to get L_2
5) Output $L_1 - L_2 / r_1 - r_2$

V accepts both executions $\Rightarrow$ $g^{L_1} = I \cdot h^{r_1}$ & $g^{L_2} = I \cdot h^{r_2}$ $\Rightarrow$ $g^{L_1 - L_2} = h^{r_1 - r_2}$ $\Rightarrow$ $a = \frac{L_1 - L_2}{r_1 - r_2}$

Extraction error $1/p$ $\Leftarrow$ Fails if $r_1 = r_2$ □

Recap/Next Lecture

- ZK IP
  - Knowledge vs. information
  - Modelled "zero knowledge" via simulator
  - (Honest-verifier) ZKP for GNI (A5: QNR) and GI (A5: QR)

Recap/Next Lecture

- ZK IP
  - Knowledge vs. information
  - Modelled "zero knowledge" via simulator
  - (Honest-verifier) ZKP for GNI (A5: QNR) and GI (A5: QR)
- ZK PoK
  - Modelled "knowledge" via extractor
  - ZKPoK for GI, DLP

Recap/Next Lecture

- ZK IP
  - Knowledge vs. information
  - Modelled "zero knowledge" via simulator
  - (Honest-verifier) ZKP for GNI (A5: QNR) and GI (A5: QR)
- ZK PoK
  - Modelled "knowledge" via extractor
  - ZKPoK for GI, DLP
- Next Lecture:
  - Application: eVoting
  - Tools used: Elgamal PKE, ZK (PoK)

References

1. [Gol01, Chapters 4.3 and 4.7] for details of today's lecture
2. [GMR89] for definitional and philosophical discussion on ZK
3. The ZKPs for GI and GNI are taken from [GMR89, GMW91]
4. Computational ZKP for all of PSPACE is due to [GMW91].


Shafi Goldwasser, Silvio Micali, and Charles Rackoff.

The knowledge complexity of interactive proof systems.

SIAM J. Comput., 18(1):186–208, 1989.


Oded Goldreich, Silvio Micali, and Avi Wigderson.

Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems.

J. ACM, 38(3):691–729, 1991.


Oded Goldreich.

The Foundations of Cryptography - Volume 1: Basic Techniques.

Cambridge University Press, 2001.


Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan.

Algebraic methods for interactive proof systems.

J. ACM, 39(4):859–868, October 1992.


Adi Shamir.

IP=PSPACE.

In *31st FOCS*, pages 11–15. IEEE Computer Society Press, October 1990.