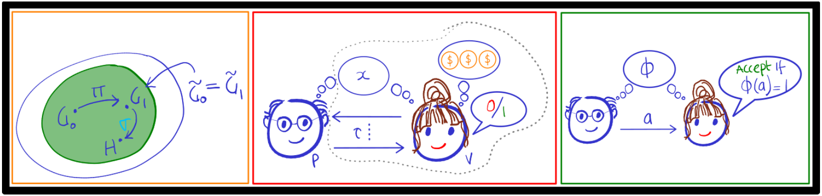


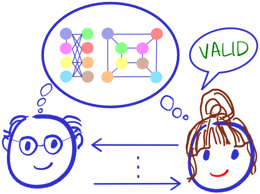
CS760: Topics in Computational Complexity

Lecture 21(?) (18/Oct/24)

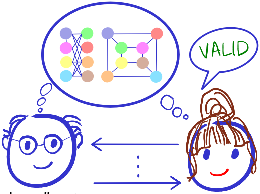
Instructor: Chethan Kamath (Stand-in for Rohit Gurjar)

Plan for Today's Lecture

- What constitutes a proof?
  - Traditional "NP" proofs vs *interactive* proofs

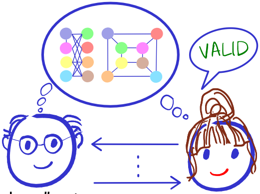


Plan for Today's Lecture



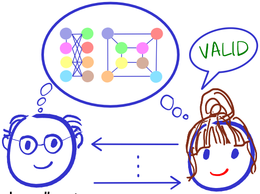
- What constitutes a proof?
  - Traditional "NP" proofs vs *interactive* proofs
- Zero-knowledge proofs: capture "zero knowledge" via simulation paradigm

Plan for Today's Lecture

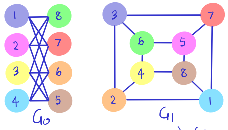


- What constitutes a proof?
  - Traditional "**NP**" proofs vs *interactive* proofs
- Zero-knowledge proofs: capture "zero knowledge" via simulation paradigm
- Examples. ZKP for:
  - Graph isomorphism (GI)
  - Quadratic residuosity (QR)
  - Graph non-isomorphism (GNI)
  - Quadratic non-residuosity (QNR)

Plan for Today's Lecture



- What constitutes a proof?
  - Traditional "**NP**" proofs vs *interactive* proofs
- Zero-knowledge proofs: capture "zero knowledge" via simulation paradigm

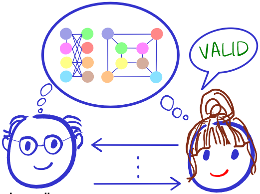


$G_0=([1,n],E_0) \cong G_1=([1,n],E_1)$ iff $\exists$ permutation π on $[1,n]$ s.t. $(u,v)\in E_0 \Rightarrow (\pi(u),\pi(v))\in E_1$

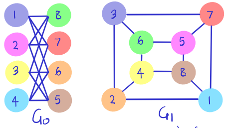
$G_0 \overset{\pi}{\cong} G_1$ or $G_1=\pi(G_0)$

- Examples. ZKP for:
  - Graph isomorphism (GI)
  - Quadratic residuosity (QR)
  - Graph non-isomorphism (GNI)
  - Quadratic non-residuosity (QNR)

Plan for Today's Lecture

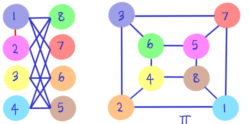


- What constitutes a proof?
  - Traditional "**NP**" proofs vs *interactive* proofs
- Zero-knowledge proofs: capture "zero knowledge" via simulation paradigm



$G_0=([1,n],E_0)\cong G_1=([1,n],E_1)$ if $\exists$ permutation π on $[1,n]$ s.t. $(u,v)\in E_0 \Rightarrow (\pi(u),\pi(v))\in E_1$ $\to$ $G_0 \overset{\pi}{\cong} G_1$ or $G_1=\pi(G_0)$

- Examples. ZKP for:
  - Graph isomorphism (GI)
  - Quadratic residuosity (QR)
  - Graph non-isomorphism (GNI)
  - Quadratic non-residuosity (QNR)



$G_0 \not\cong G_1$ if $\nexists \pi$ s.t. $G_0 \overset{\pi}{\cong} G_1$

Plan for Today's Lecture

1. Interactive Proof (IP)

2. Zero Knowledge (Interactive) Proof (ZKP)

3. Class **SZK**: Statistical Zero-Knowledge Proof

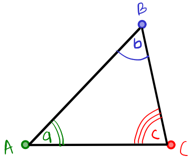
Plan for Today's Lecture

1. Interactive Proof (IP)
2. Zero Knowledge (Interactive) Proof (ZKP)
3. Class **SZK**: Statistical Zero-Knowledge Proof

Traditional "NP" Proof...

- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

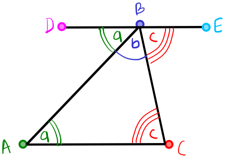
Theorem: "Sum of angles of a triangle equals 180°"



Traditional “NP” Proof...

- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

    Theorem: “Sum of angles of a triangle equals 180°”

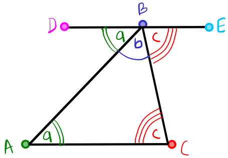


- Prover vs. verifier
  - Prover does the heavy lifting: derives the proof
    1. Construct a line through B parallel to $\overline{AC}$
    2. $\angle DBA = \angle a$ and $\angle EBC = \angle c$ (alternate interior angles)
    3. $2 \Rightarrow \angle a + \angle b + \angle c = \angle DBA + \angle b + \angle EBC = 180°$

Traditional "NP" Proof...

- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

    Theorem: "Sum of angles of a triangle equals 180°"



- Prover vs. verifier
  - Prover does the heavy lifting: derives the proof
    1. Construct a line through B parallel to $\overline{AC}$
    2. $\angle DBA = \angle a$ and $\angle EBC = \angle c$ (alternate interior angles)
    3. $2 \Rightarrow \angle a + \angle b + \angle c = \angle DBA + \angle b + \angle EBC = 180°$
  - Verifier checks the proof, step by step

Traditional "NP" Proof...

- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

$$\forall x \in \mathcal{L}\ \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x, \pi) = 1$$

Traditional "NP" Proof...

- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement — witness/proof

$$\forall x \in \mathcal{L}\ \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x, \pi) = 1$$

Traditional "NP" Proof...

- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement　　　　　　　　　　　　　　　　　　　　witness/proof

$$\underbrace{\forall x \in \mathcal{L}}_{\text{statement}} \; \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x, \pi) = 1$$

  - NP is the class of all such $\mathcal{L}$s

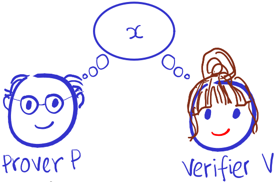
Traditional "NP" Proof...

- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement — witness/proof

$$\forall x \in \mathcal{L}\ \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x, \pi) = 1$$

  - NP is the class of all such $\mathcal{L}$s



- "Proof system" view of NP
  - Prover P is *unbounded*: finds short proof π for x (if one exists)
  - Verifier V is *efficient*: checks proof π against the statement x

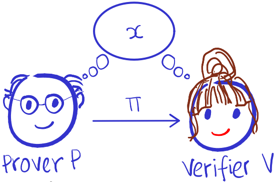
Traditional "NP" Proof...

- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement   witness/proof

$$\forall x \in \mathcal{L}\ \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x, \pi) = 1$$

  - NP is the class of all such $\mathcal{L}$s



- "Proof system" view of NP
  - Prover P is *unbounded*: finds short proof π for x (if one exists)
  - Verifier V is *efficient*: checks proof π against the statement x

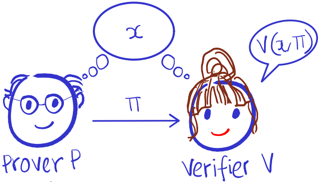
Traditional "NP" Proof...

- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement witness/proof

$$\forall x \in \mathcal{L}\ \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x,\pi) = 1$$

  - NP is the class of all such $\mathcal{L}$s



- "Proof system" view of NP
  - Prover P is *unbounded*: finds short proof π for x (if one exists)
  - Verifier V is *efficient*: checks proof π against the statement x

Traditional "NP" Proof...

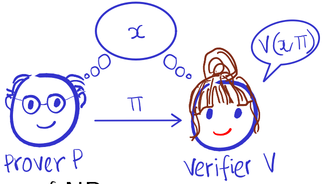
- Corresponds to class NP
  - A language $\mathcal{L} \in \mathsf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

witness/proof

statement

$$\forall x \in \mathcal{L}\ \exists \pi \in \{0,1\}^{\mathsf{poly}(|x|)} : \mathsf{V}(x, \pi) = 1$$

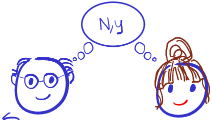
  - NP is the class of all such $\mathcal{L}$s



- "Proof system" view of NP
  - Prover P is *unbounded*: finds short proof π for x (if one exists)
  - Verifier V is *efficient*: checks proof π against the statement x
  - Completeness: $x \in \mathcal{L} \Rightarrow$ P finds $\pi \Rightarrow \mathsf{V}(x, \pi) = 1$
  - Soundness: $x \notin \mathcal{L} \Rightarrow\ \nexists \pi \in \{0,1\}^{\mathsf{poly}(|x|)}$ s.t. $\mathsf{V}(x, \pi) = 1$

Which Languages have "NP" Proofs?

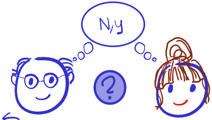
Quadratic residuosity (QR)



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

Which Languages have “NP” Proofs?

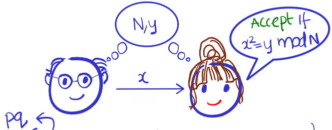
Quadratic residuosity (QR)



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

Which Languages have "NP" Proofs?

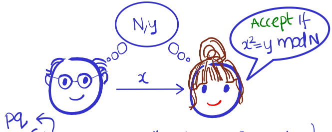
Quadratic residuosity (QR)



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

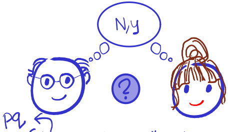
Which Languages have "NP" Proofs?

Quadratic residuosity (QR)



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

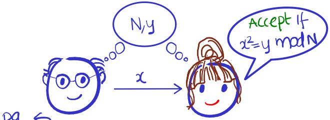
Quad. non-residuosity (QNR)



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

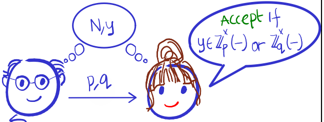
Which Languages have "NP" Proofs?

Quadratic residuosity (QR)



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

Quad. non-residuosity (QNR)



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

Which Languages have “NP” Proofs?

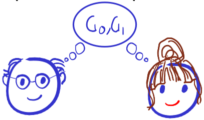
Quadratic residuosity (QR)

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

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)



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

Which Languages have "NP" Proofs?

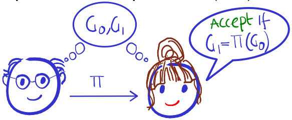
Quadratic residuosity (QR)

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

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)



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

Which Languages have "NP" Proofs?

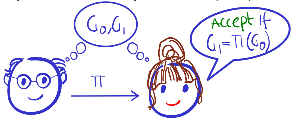
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

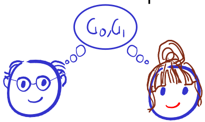
$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

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)



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

Which Languages have “NP” Proofs?

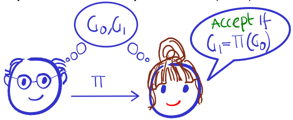
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

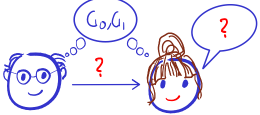
$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

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)



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

Which Languages have “NP” Proofs?

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)

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

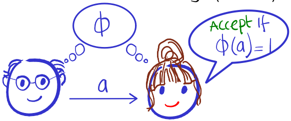
Graph isomorphism (GI)

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

Graph non-isomorphism (GNI)

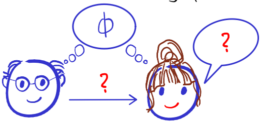
$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Boolean satisfiability (SAT)



$\mathcal{L}_{SAT} = \{\phi : \exists \text{assignment } a \text{ s.t. } \phi(a) = 1\}$

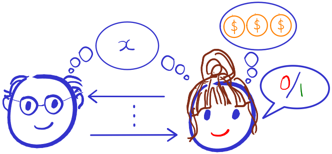
Bool. unsatisfiability (UNSAT)



$\mathcal{L}_{UNSAT} = \{\phi : \nexists \text{assignment } a \text{ s.t. } \phi(a) = 1\}$

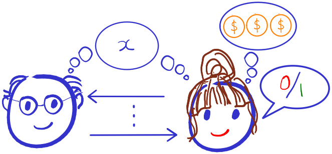
Interactive Proof (IP)

- Difference from **NP** proofs:
  1. Verifier **V** is *randomised*
  2. Prover **P** and **V** *interact* and **V** accepts/rejects in the end



Interactive Proof (IP)

- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end

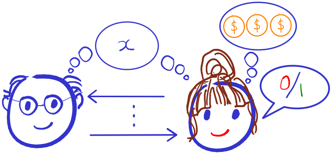


Defintion 1

An interactive protocol (P, V) *for a language* $\mathcal{L}$ *is an interactive proof (IP) system if the following holds:*

Interactive Proof (IP)

- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end



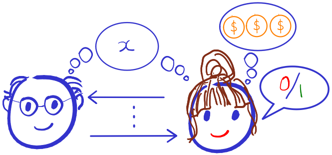
Defintion 1

An interactive protocol (P, V) *for a language* $\mathcal{L}$ *is an interactive proof (IP) system if the following holds:*

- *Completeness: for every* $x \in \mathcal{L}$, $\Pr[1 \leftarrow \langle \mathsf{P}, \mathsf{V}\rangle(x)] \geq 1 - 1/3$

Interactive Proof (IP)

- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end



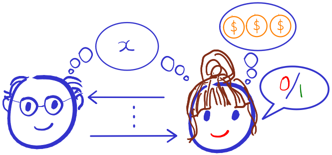
Defintion 1

An interactive protocol (P, V) *for a language* $\mathcal{L}$ *is an interactive proof (IP) system if the following holds:*

- *Completeness: for every* $x \in \mathcal{L}$, $\Pr[1 \leftarrow \langle \mathsf{P}, \mathsf{V}\rangle(x)] \geq 1 - 1/3$
- *Soundness: for every* $x \notin \mathcal{L}$ *and malicious prover* P^*,
  $$\Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V}\rangle(x)] \leq 1/3$$

Interactive Proof (IP)

- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end



Defintion 1

An interactive protocol (P, V) *for a language* $\mathcal{L}$ *is an interactive proof (IP) system if the following holds:* completeness error $\varepsilon_c(n)$

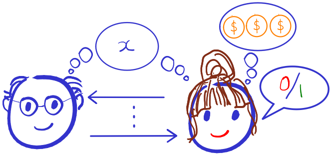
- *Completeness: for every* $x \in \mathcal{L}$, $\Pr[1 \leftarrow \langle \mathsf{P}, \mathsf{V}\rangle(x)] \geq 1 - 1/3$
- *Soundness: for every* $x \notin \mathcal{L}$ *and malicious prover* P^*,

$$\Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V}\rangle(x)] \leq 1/3$$

soundness error $\varepsilon_s(n)$

Interactive Proof (IP)

- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end



Defintion 1

An interactive protocol (P, V) *for a language* $\mathcal{L}$ *is an interactive proof (IP) system if the following holds:* completeness error $\varepsilon_c(n)$

- *Completeness: for every* $x \in \mathcal{L}$, $\Pr[1 \leftarrow \langle \mathsf{P}, \mathsf{V} \rangle(x)] \geq 1 - 1/3$
- *Soundness: for every* $x \notin \mathcal{L}$ *and malicious prover* P^*,

$$\Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle(x)] \leq 1/3$$

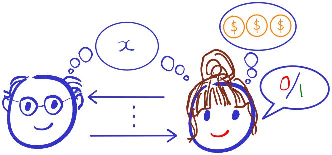
soundness error $\varepsilon_s(n)$

Exercise 1 (Robustness of Defintion 1)

Show that languages captured by Defintion 1 doesn't change when 1) $\epsilon_c \leq 1/2^{|x|}$, $\epsilon_s \leq 1/2^{|x|}$*; 2)* $\epsilon_c \leq 1/2 - 1/|x|$, $\epsilon_s \leq 1/2 - 1/|x|$

Interactive Proof (IP)

- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end



Defintion 1

An interactive protocol (P, V) *for a language* $\mathcal{L}$ *is an interactive proof (IP) system if the following holds:* completeness error $\varepsilon_c(n)$

- *Completeness: for every* $x \in \mathcal{L}$, $\Pr[1 \leftarrow \langle \mathsf{P}, \mathsf{V} \rangle (x)] \geq 1 - 1/3$
- *Soundness: for every* $x \notin \mathcal{L}$ *and malicious prover* P^*,

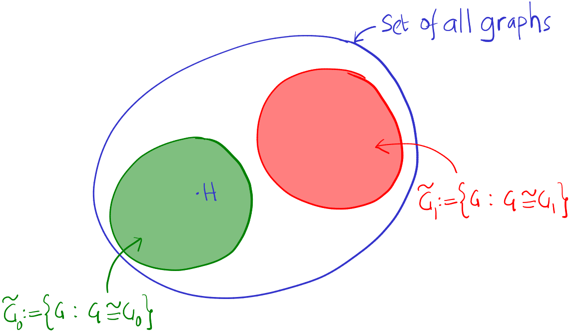
$$\Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V} \rangle (x)] \leq 1/3$$

soundness error $\varepsilon_s(n)$

Exercise 1 (Robustness of Defintion 1)

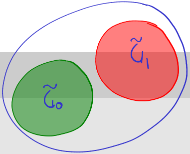
Show that languages captured by Defintion 1 doesn't change when 1) $\epsilon_c \leq 1/2^{|x|}$, $\epsilon_s \leq 1/2^{|x|}$; *2)* $\epsilon_c \leq 1/2 - 1/|x|$, $\epsilon_s \leq 1/2 - 1/|x|$

Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

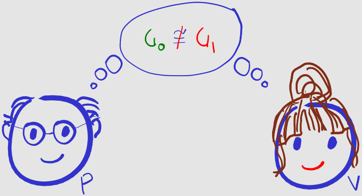


Power of Randomness+Interaction: IP for GNI...

Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

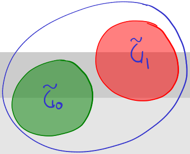


Protocol 1 (Π_{GNI}: IP for GNI)

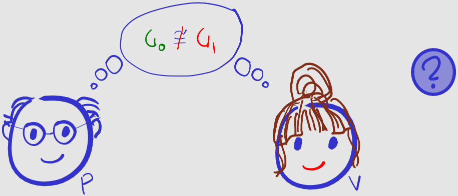


Power of Randomness+Interaction: IP for GNI...

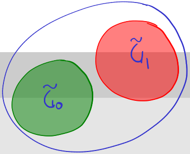
Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*



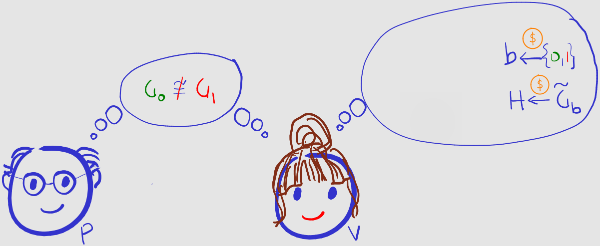
Protocol 1 (Π_{GNI}: IP for GNI)



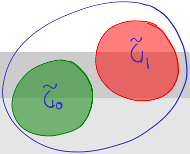
Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*



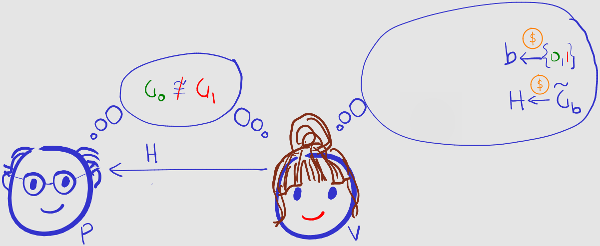
Protocol 1 (Π_{GNI}: IP for GNI)



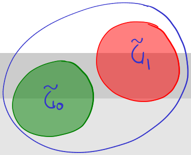
Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*



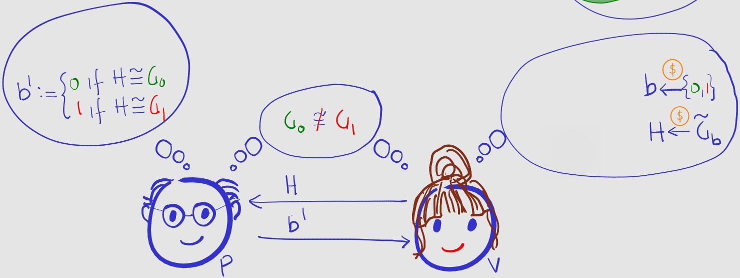
Protocol 1 (Π_{GNI}: IP for GNI)



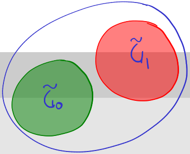
Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*



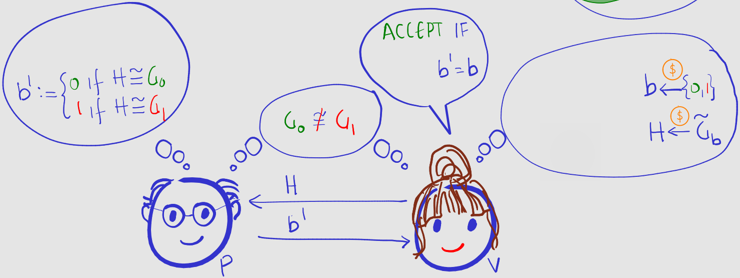
Protocol 1 (Π_{GNI}: IP for GNI)



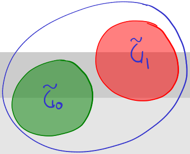
Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, *$G_0 \cong H$ and $G_1 \cong H$ both cannot hold*



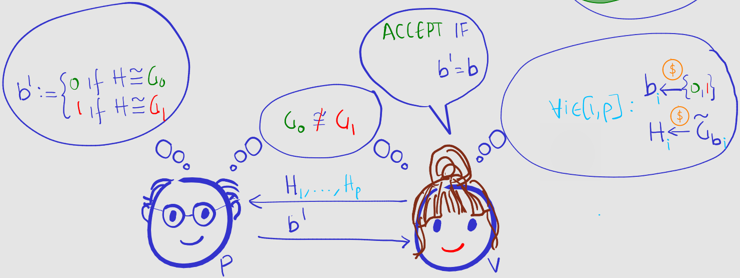
Protocol 1 (Π_{GNI}: IP for GNI)



Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

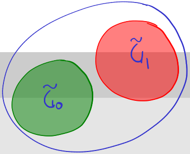


Protocol 1 (Π_{GNI}: IP for GNI)

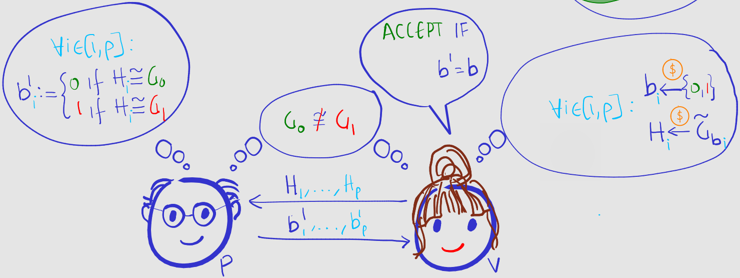


Parallel/sequentially repeat to boost soundness

Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*



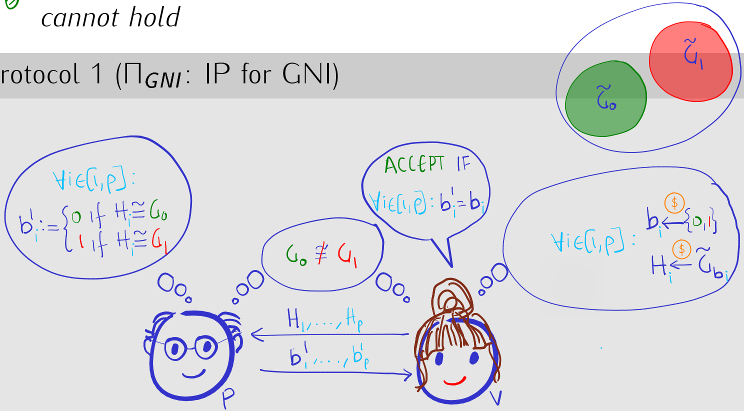
Protocol 1 (Π_{GNI}: IP for GNI)



Parallel/sequentially repeat to boost soundness

Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

Protocol 1 (Π_{GNI}: IP for GNI)



Parallel/sequentially repeat to boost soundness

Power of Randomness+Interaction: IP for GNI...

Theorem 1

Π_{GNI} *is an IP for* $\mathcal{L}_{GNI}$

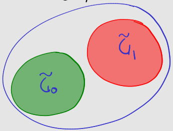
Power of Randomness+Interaction: IP for GNI...

Theorem 1

Π_{GNI} is an IP for $\mathcal{L}_{GNI}$

Proof.

- Completeness:
  - $G_0 \not\cong G_1 \Rightarrow$ P can recover b_i from H_i with certainty



$$\Pr[1 \leftarrow \langle \mathsf{P}, \mathsf{V} \rangle (G_0, G_1)] = 1 \geq 2/3$$

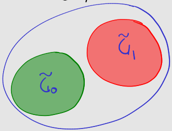
Power of Randomness+Interaction: IP for GNI...

Theorem 1

Π_{GNI} is an IP for $\mathcal{L}_{GNI}$

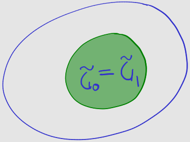
Proof.

- Completeness:
  - $G_0 \not\cong G_1 \Rightarrow$ P can recover b_i from H_i with certainty



$$\Pr[1 \leftarrow \langle \mathrm{P}, \mathrm{V} \rangle (G_0, G_1)] = 1 \geq 2/3$$

- Soundness:
  - $G_0 \cong G_1 \Rightarrow H_i$ loses information about bits b_i
  - Hence best P* can do is guess b_is

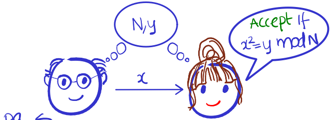


$$\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V} \rangle (G_0, G_1)] = 1/2^{\rho} < 1/3$$

□

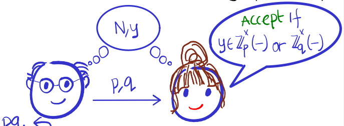
Which Languages have IPs?

Quadratic residuosity (QR)



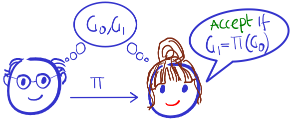
$\mathcal{L}_{QR}=\{(N,y): \exists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Quad. non-residuosity (QNR)



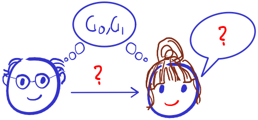
$\mathcal{L}_{QNR}=\{(N,y): \nexists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Graph isomorphism (GI)



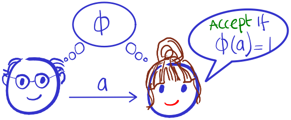
$\mathcal{L}_{GI}=\{(G_0,G_1): \exists \text{ permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Graph non-isomorphism (GNI)



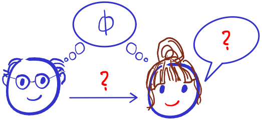
$\mathcal{L}_{GNI}=\{(G_0,G_1): \nexists \text{ permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Boolean satisfiability (SAT)



$\mathcal{L}_{SAT}=\{\phi : \exists \text{ assignment } a \text{ s.t. } \phi(a)=1\}$

Bool. unsatisfiability (UNSAT)



$\mathcal{L}_{UNSAT}=\{\phi : \nexists \text{ assignment } a \text{ s.t. } \phi(a)=1\}$

Which Languages have IPs?

Quadratic residuosity (QR)

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

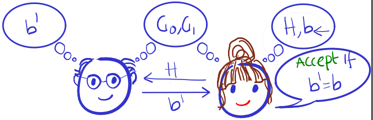
Quad. non-residuosity (QNR)

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

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)

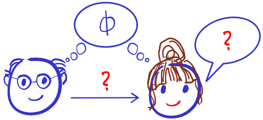


$\mathcal{L}_{GNI}=\{(G_0,G_1): \nexists \text{ permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Boolean satisfiability (SAT)

$\mathcal{L}_{SAT}=\{\phi: \exists \text{ assignment } a \text{ s.t. } \phi(a)=1\}$

Bool. unsatisfiability (UNSAT)



$\mathcal{L}_{UNSAT}=\{\phi: \nexists \text{ assignment } a \text{ s.t. } \phi(a)=1\}$

Which Languages have IPs?

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)

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

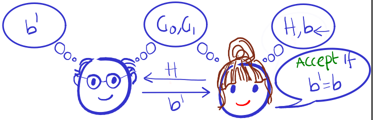
✓

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)



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

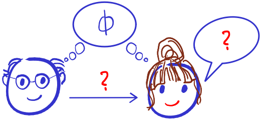
✓

Boolean satisfiability (SAT)

$\mathcal{L}_{SAT}=\{\phi: \exists \text{assignment } a \text{ s.t. } \phi(a)=1\}$

✓

Bool. unsatisfiability (UNSAT)



$\mathcal{L}_{UNSAT}=\{\phi: \nexists \text{assignment } a \text{ s.t. } \phi(a)=1\}$

"Sumcheck"

✓

Which Languages have IPs? PSPACE Languages

Quadratic residuosity (QR)

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

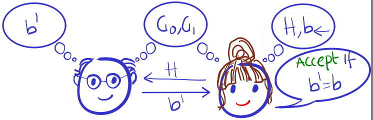
Quad. non-residuosity (QNR)

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

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)



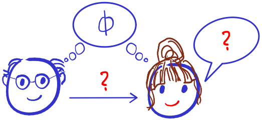
$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Boolean satisfiability (SAT)

$\mathcal{L}_{SAT} = \{\phi : \exists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Bool. unsatisfiability (UNSAT)

"Sumcheck"



$\mathcal{L}_{UNSAT} = \{\phi : \nexists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Plan for Today's Lecture

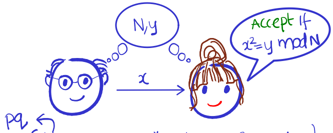
1. Interactive Proof (IP)

2. Zero Knowledge (Interactive) Proof (ZKP)

3. Class **SZK**: Statistical Zero-Knowledge Proof

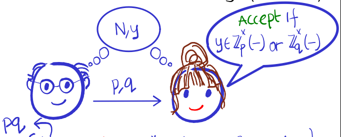
Any Issues with the NP Proofs We Saw?

Quadratic residuosity (QR)



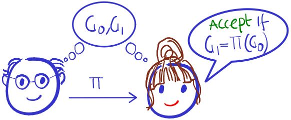
$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Quad. non-residuosity (QNR)



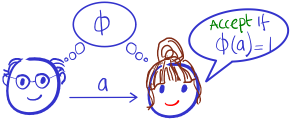
$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Graph isomorphism (GI)



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

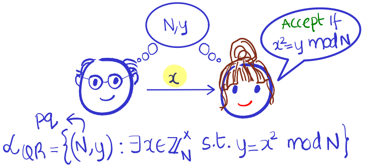
Boolean satisfiability (SAT)



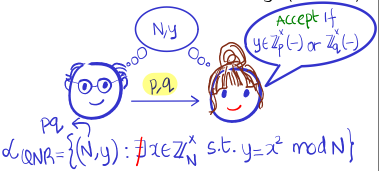
$\mathcal{L}_{SAT} = \{\phi : \exists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Any Issues with the NP Proofs We Saw?

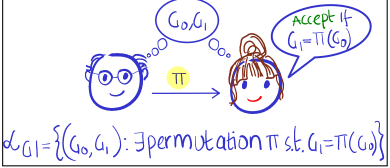
Quadratic residuosity (QR)



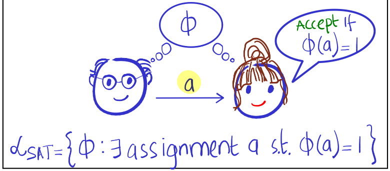
Quad. non-residuosity (QNR)



Graph isomorphism (GI)

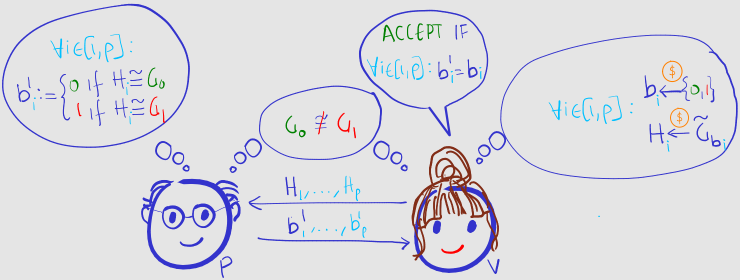


Boolean satisfiability (SAT)



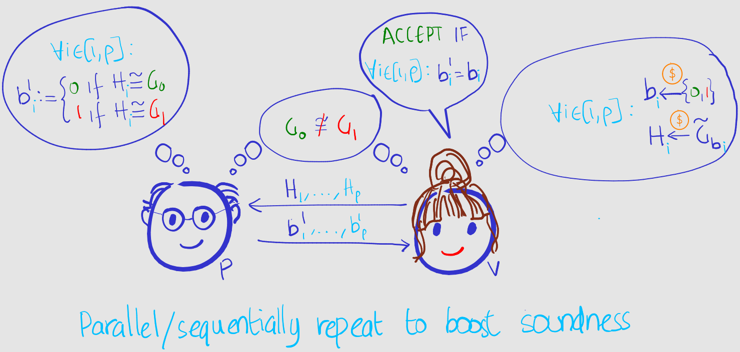
- Verifier gains "non-trivial knowledge" about witness w
  - Not desirable, e.g., when $x = pk$ and $w = sk$ (identification)

Protocol 1 (Π_{GNI}: IP for GNI)



Parallel/sequentially repeat to boost soundness

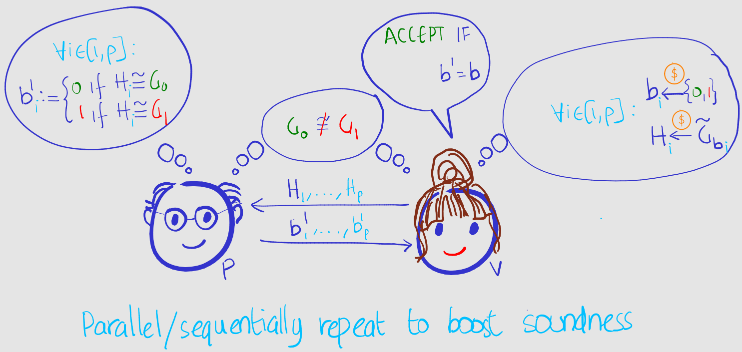
Protocol 1 (Π_{GNI}: IP for GNI)



- Seems V *gains no knowledge* beyond validity of the statement

What About the IP We Saw?

Protocol 1 (Π_{GNI}: IP for GNI)



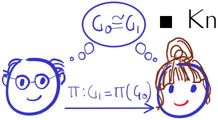
- Seems V *gains no knowledge* beyond validity of the statement
- We will see that Π_{GNI} is (honest-verifier) zero-knowledge!

How to Capture "V Gains No Knowledge"?

- Knowledge vs. information in the information theoretic sense
  - Knowledge is computational

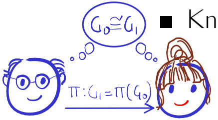
How to Capture “V Gains No Knowledge”?

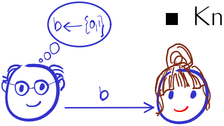
- Knowledge vs. information in the information theoretic sense
  - Knowledge is computational: e.g., consider **NP** proof for GI



    - 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”?

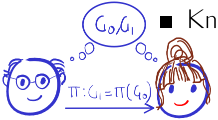
- Knowledge vs. information ← in the information theoretic sense
  

  - 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
  

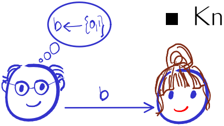
  - Knowledge pertains to public objects:
    - Flipping a *private* fair coin b and (later) revealing its outcome leads to V gaining *information*
    - But V *does not gain knowledge*: she could herself have tossed the private coin and revealed it

How to Capture "V Gains No Knowledge"?

- Knowledge vs. information ← in the information theoretic sense
  - Knowledge is computational: e.g., consider **NP** proof for GI



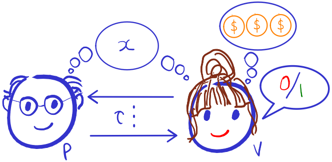
    - Given (G_0, G_1), the isomorphism π contains no *information*
    - But when given π, V "gains knowledge" since she couldn't have computed π herself
  - Knowledge pertains to public objects:



    - 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

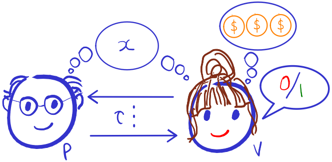
Intuitively, "V gains no knowledge" if anything (other than the validity of x) V can *compute* after the interaction, V could have computed *without it*

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



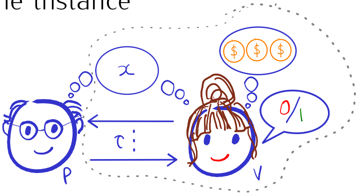
V's "view" = x + transcript τ + coins

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



V's "view" = x + transcript τ + coins

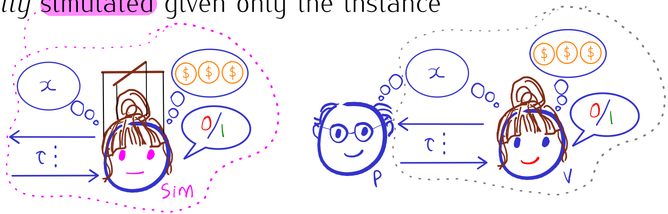
- Formalised via "simulation paradigm": $\mathsf{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V}\rangle(x))$ can be *efficiently* simulated given only the instance



Defining Zero Knowledge via Simulators...

V's "view" = x + transcript τ + coins

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



Defintion 2 (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 $x \in \mathcal{L}$, the following is zero*

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

Defining Zero Knowledge via Simulators...

Malicious-verifier

Defintion 2 (~~Honest-Verifier~~ *Perfect* ZK)

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

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

- Malicious-Verifier ZK: honest verifier V → all verifiers V^*
  - For every V^* there exists a PPT simulator Sim

Defining Zero Knowledge via Simulators...

Defintion 2 (Honest-Verifier ~~*Perfect*~~ statistical ZK)

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

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

- Malicious-Verifier ZK: honest verifier $\mathsf{V} \rightarrow$ all verifiers V^*
  - For every V^* there exists a PPT simulator Sim
- Statistical ZK: relax “zero” to “negligible”
  - Equivalently: relax “distributions identical” to “distributions statistically close”

Defining Zero Knowledge via Simulators...

Defintion 2 (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* $x \in \mathcal{L}$*, the following is zero*

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

- Malicious-Verifier ZK: honest verifier $\mathsf{V} \rightarrow$ all verifiers V^*
  - For every V^* there exists a PPT simulator Sim
- Statistical ZK: relax "zero" to "negligible"
  - Equivalently: relax "distributions identical" to "distributions statistically close"

Exercise 2

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

Π_{GNI} is *Honest-Verifier* ZK

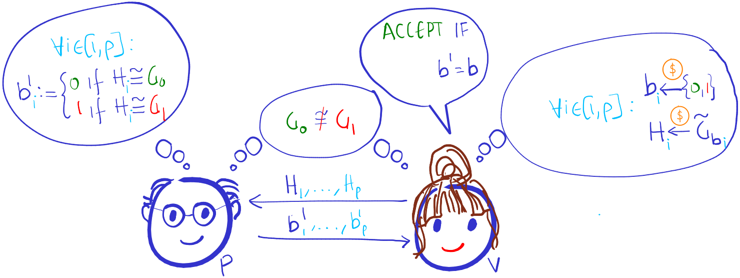
Theorem 2

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

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

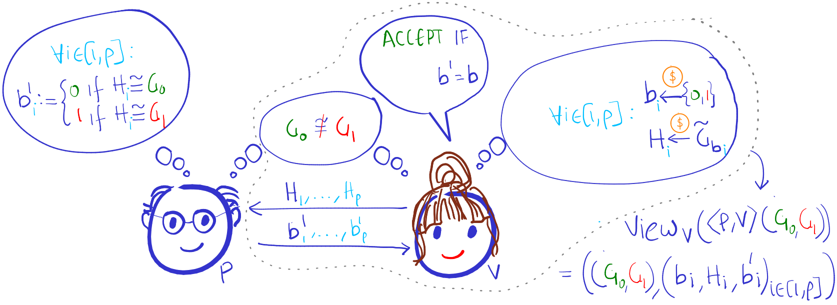


Parallel/sequentially repeat to boost soundness

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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 2

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

Proof.

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

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

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

Proof.

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

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

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

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

Proof.

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

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

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

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

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

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

Proof.

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

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

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

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

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

Exercise 3

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

Are Randomness and Interaction Necessary?

$\leftrightarrows$ Interaction is necessary

Exercise 4

If $\mathcal{L}$ has a non-interactive (i.e, one-message) ZKP then $\mathcal{L} \in \mathsf{BPP}$

Are Randomness and Interaction Necessary?

\$ $\leftrightarrows$

$L \in \mathrm{BPP}$ if $\exists$ PPT V : $\forall x \in L : \Pr_V[V(x)=1] \geq 2/3$

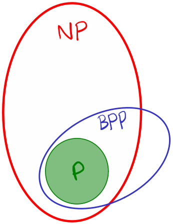
$\forall x \notin L : \Pr_V[V(x)=1] \leq 1/3$

Bounded-error probabilistic polynomial (BPP)

$\leftrightarrows$ Interaction is necessary

Exercise 4

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



Are Randomness and Interaction Necessary?

$\mathcal{L} \in \mathrm{BPP}$ if $\exists$ PPT V : $\forall x \in \mathcal{L} : \Pr_V[V(x)=1] \geq 2/3$
$\forall x \notin \mathcal{L} : \Pr_V[V(x)=1] \leq 1/3$

Bounded-error probabilistic polynomial (BPP)

$\leftrightarrows$ Interaction is necessary

Exercise 4

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

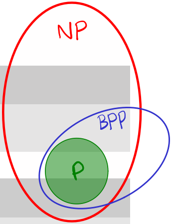
\$ Randomness is necessary

Exercise 5

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

Exercise 6

If $\mathcal{L}$ has an ZKP with deterministic verifier then $\mathcal{L} \in$ BPP



Plan for Today's Lecture

1 Interactive Proof (IP)

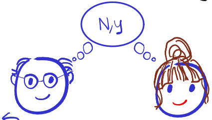
2 Zero Knowledge (Interactive) Proof (ZKP)

3 Class **SZK**: Statistical Zero-Knowledge Proof

Which Languages have ZKPs?

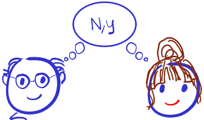
✓

Quadratic residuosity (QR)



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

Quad. non-residuosity (QNR)

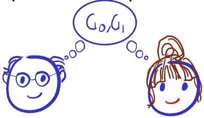


$\mathcal{L}_{QNR} = \{(\overset{pq}{N},y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

✓

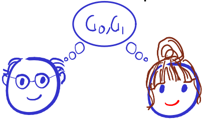
✓

Graph isomorphism (GI)



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

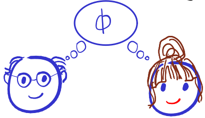
Graph non-isomorphism (GNI)



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

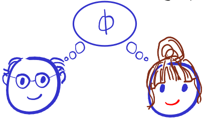
✓

Boolean satisfiability (SAT)



$\mathcal{L}_{SAT} = \{\phi : \exists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Bool. unsatisfiability (UNSAT)



$\mathcal{L}_{UNSAT} = \{\phi : \nexists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Which Languages have ZKPs?

✓

Quadratic residuosity (QR)

pq

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

Quad. non-residuosity (QNR)

✓

pq

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

✓

Graph isomorphism (GI)

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

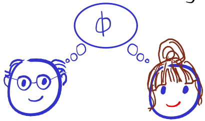
Graph non-isomorphism (GNI)

✓

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

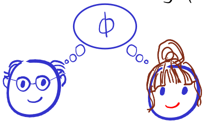
L15 ✓

Boolean satisfiability (SAT)



$\mathcal{L}_{SAT} = \{\phi : \exists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Bool. unsatisfiability (UNSAT)



$\mathcal{L}_{UNSAT} = \{\phi : \nexists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Which Languages have Computational 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)

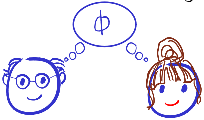
✓

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

L15

✓

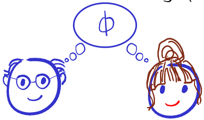
Boolean satisfiability (SAT)



$\mathcal{L}_{SAT}=\{\phi: \exists \text{ assignment } a \text{ s.t. } \phi(a)=1\}$

Bool. unsatisfiability (UNSAT)

✓



$\mathcal{L}_{UNSAT}=\{\phi: \nexists \text{ assignment } a \text{ s.t. } \phi(a)=1\}$

Class Statistical Zero Knowledge Proofs (SZK)

- The class of languages which admit *malicious verifier statistical* zero-knowledge proofs

Class Statistical Zero Knowledge Proofs (SZK)

- The class of languages which admit *malicious verifier statistical* zero-knowledge proofs
- Why is it interesting?
  - Contains a host of languages with connections to cryptography.
    1. Number-theoretic problems: QR, QNR
    2. Lattice problems: approximate shortest-vector problem, closest vector problems
    3. ...

Class Statistical Zero Knowledge Proofs (SZK)

- The class of languages which admit *malicious verifier statistical* zero-knowledge proofs
- Why is it interesting?
  - Contains a host of languages with connections to cryptography.
    1. Number-theoretic problems: QR, QNR
    2. Lattice problems: approximate shortest-vector problem, closest vector problems
    3. ...
  - Has complete problems: e.g., statistical difference (SD)
    - Given two circuits $C_0, C_1 : \{0,1\}^n \to \{0,1\}^m$, decide whether the distributions induced inputting C_0 and C_1 are statistically "close"or "far".

Class Statistical Zero Knowledge Proofs (SZK)

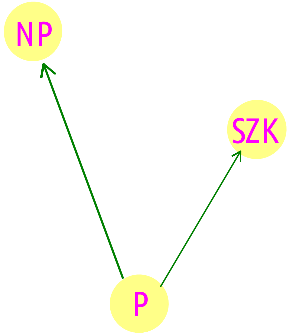
- The class of languages which admit *malicious verifier statistical* zero-knowledge proofs
- Why is it interesting?
  - Contains a host of languages with connections to cryptography.
    1. Number-theoretic problems: QR, QNR
    2. Lattice problems: approximate shortest-vector problem, closest vector problems
    3. ...
  - Has complete problems: e.g., statistical difference (SD)
    - Given two circuits $C_0, C_1 : \{0,1\}^n \rightarrow \{0,1\}^m$, decide whether the distributions induced inputting C_0 and C_1 are statistically "close"or "far".

Exercise 7

Can you think of a honest-verifier SZK proof for SD*?*

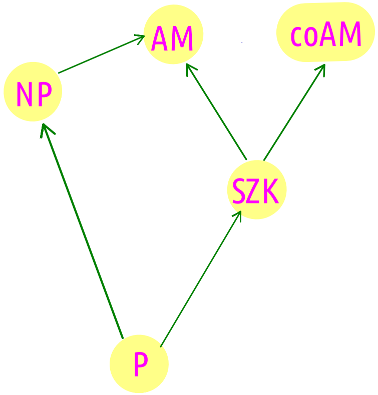
What do we know about SZK?

+ Closed under complement, i.e., $\mathsf{SZK} = \mathsf{coSZK}$



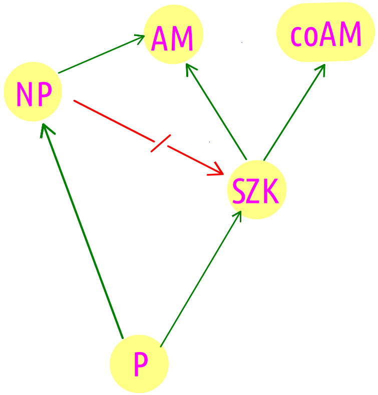
What do we know about SZK?

+ Closed under complement, i.e., SZK = coSZK
+ Contained in AM ∩ coAM
  - AM: *public-coin*, one-round IP



What do we know about SZK?

- + Closed under complement, i.e., SZK = coSZK
- + Contained in AM ∩ coAM
  - AM: *public-coin*, one-round IP
- − NP cannot be contained in SZK (unless polynomial hierarchy collapses)



To Recap Today's Lecture

- Traditional "NP" proofs vs *interactive* proofs
  - IP is more powerful: IP for GNI

To Recap Today's Lecture

- Traditional "NP" proofs vs *interactive* proofs
  - IP is more powerful: IP for GNI
- Zero-knowledge proofs
  - Knowledge vs. information
  - Modelled "zero knowledge" via simulation paradigm

To Recap Today's Lecture

- Traditional "NP" proofs vs *interactive* proofs
  - IP is more powerful: IP for GNI
- Zero-knowledge proofs
  - Knowledge vs. information
  - Modelled "zero knowledge" via simulation paradigm
- *Honest-verifier* ZKP for GNI (HW: QNR)

To Recap Today's Lecture

- Traditional "**NP**" proofs vs *interactive* proofs
  - IP is more powerful: IP for GNI
- Zero-knowledge proofs
  - Knowledge vs. information
  - Modelled "zero knowledge" via simulation paradigm
- *Honest-verifier* ZKP for GNI (HW: QNR)
- Class **SZK**

References

1. [Gol01, Chapter 4] for details of today's lecture
2. [GMR89] for definitional and philosophical discussion on ZK. Salil Vadhan's thesis [Vad99] is also an excellent resource.
3. The ZKPs for GI and GNI are taken from [GMR89, GMW91]
4. IP for all of **PSPACE** is due to [LFKN92, Sha90]. Computational ZKP for all of **PSPACE** is due to [GMW91].
5. [Oka96] showed that **SZK** is closed under complement. That **NP** cannot be contained in **SZK** (unless PH collapses) is due to [For87]

Lance Fortnow.
The complexity of perfect zero-knowledge (extended abstract).
In Alfred Aho, editor, *19th ACM STOC*, pages 204–209. ACM Press, May 1987.

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.

Tatsuaki Okamoto.

On relationships between statistical zero-knowledge proofs.
In *28th ACM STOC*, pages 649–658. ACM Press, May 1996.

Adi Shamir.
IP=PSPACE.
In *31st FOCS*, pages 11–15. IEEE Computer Society Press, October 1990.

Salil Vadhan.
A study of statistical zero-knowledge proofs.
PhD thesis, Massachusetts Institute of Technology, 1999.