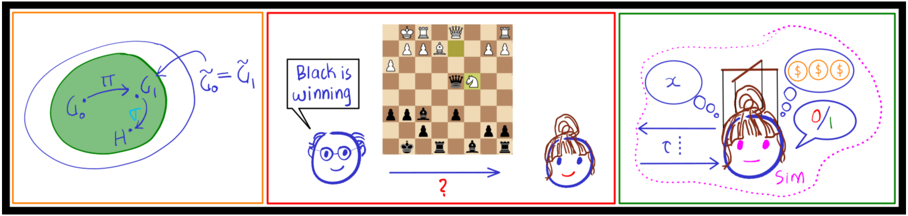
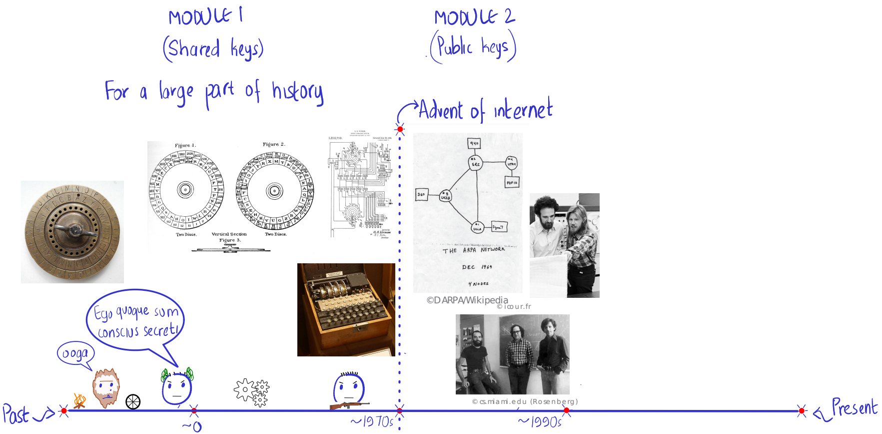


CS409m: Introduction to Cryptography

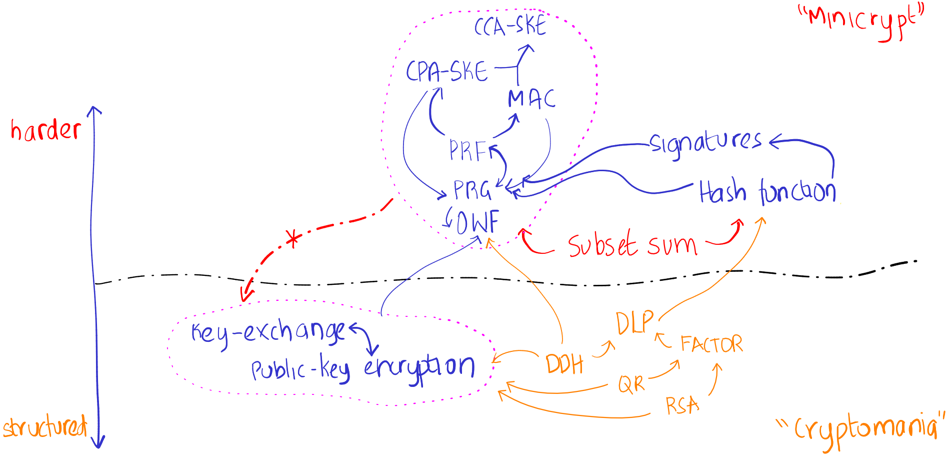
Lecture 18 (17/Oct/25)

Instructor: Chethan Kamath

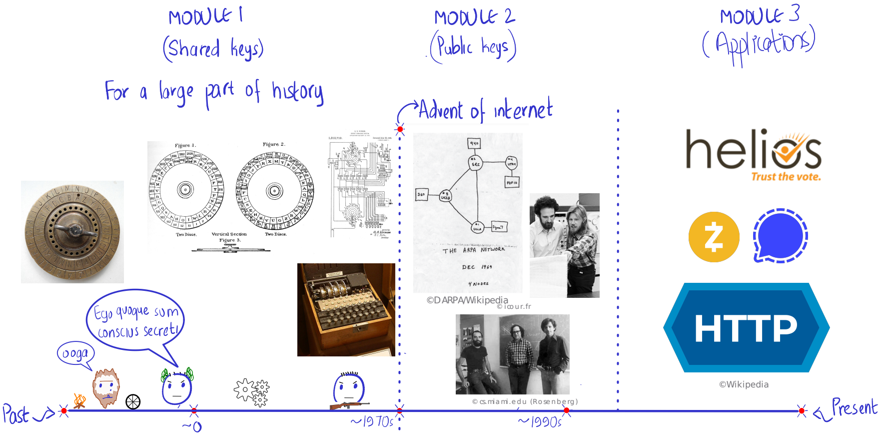
Journey So Far . . .



The Cryptographic Landscape



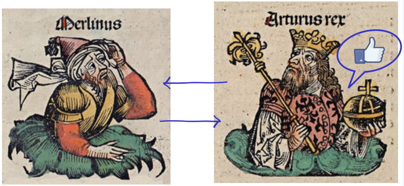
Plan for Module III: Applications!



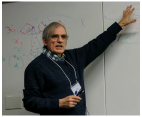
Plan for Today's Lecture...

What really constitutes a proof?

Interactive Proof (IP)



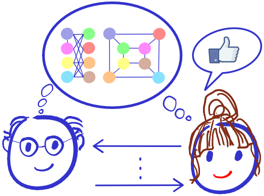
© Wikipedia

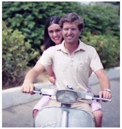


© blog.computationalcomplexity.org

© alchetron.com

Zero-Knowledge IP



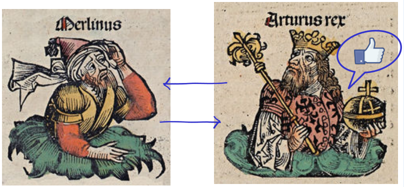


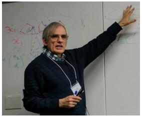
© Oded Goldreich

Plan for Today's Lecture...

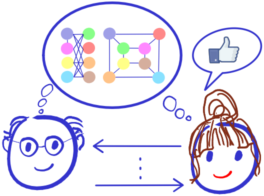
What really constitutes a proof?

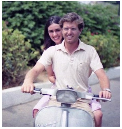
Interactive Proof (IP)





Zero-Knowledge IP

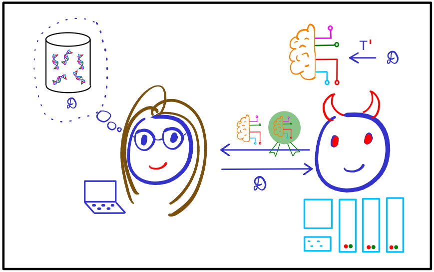




Main tools: simulation paradigm, Chernoff bound...

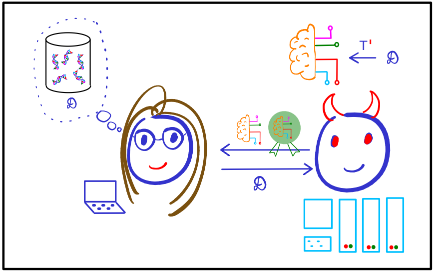
(ZK)IPs are Useful!

- Applications of IP: Verifiable outsourcing



(ZK)IPs are Useful!

- Applications of IP: Verifiable outsourcing

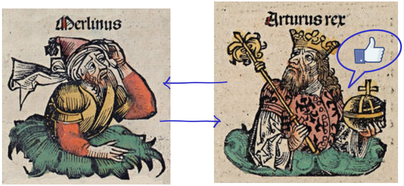


- Applications of ZKP:
  - eVoting: coming up in Lecture 20!
  - Crypto(currencies): prove validity of transaction without revealing details: coming up in Lecture 23!
  - Efficient digital signatures: Schnorr ID protocol

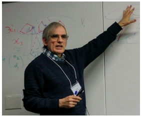
Plan for Today's Lecture...

What really constitutes a proof?

Interactive Proof (IP)



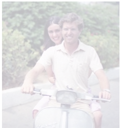
© Wikipedia



© blog.computationalcomplexity.org

© alchetron.com

Zero-Knowledge IP



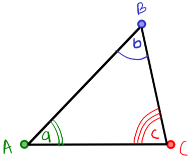
© Oded Goldreich

Main tools: simulation paradigm, Chernoff bound...

Traditional "NP" Proof...

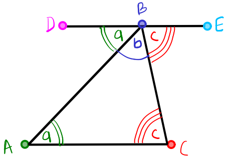
- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

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



- 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

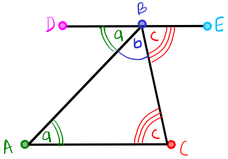

Prover does the heavy lifting: derives the proof

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

statement

witness/proof

$$\forall x \in \mathcal{L}\ \exists \text{ ``short'' } \pi : \mathsf{V}(x, \pi) = 1$$

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

witness/proof

statement $$\forall x \in \mathcal{L} \; \exists \text{"short"} \; \pi : \text{V}(x, \pi) = 1$$

  - NP is the *class* of all such $\mathcal{L}$s (e.g., Sudoku)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

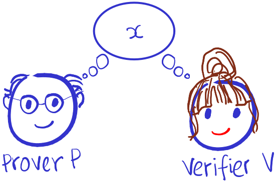
- Corresponds to the *class* NP
  - A language $\mathcal{L} \in$ NP if there exists a polynomial-time *deterministic* machine V such that

witness/proof

statement $$\forall x \in \mathcal{L}\ \exists \text{ “short” } \pi : \mathrm{V}(x, \pi) = 1$$

  - NP is the *class* of all such $\mathcal{L}$s (e.g., Sudoku)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



- "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...

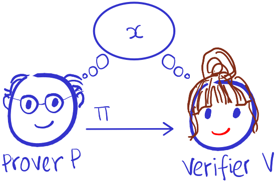
- Corresponds to the *class* NP
  - A language $\mathcal{L} \in$ NP if there exists a polynomial-time *deterministic* machine V such that

statement witness/proof

$$\forall x \in \mathcal{L}\ \exists \text{ "short" } \pi : \mathsf{V}(x, \pi) = 1$$

  - NP is the *class* of all such $\mathcal{L}$s (e.g., Sudoku)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



- "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...

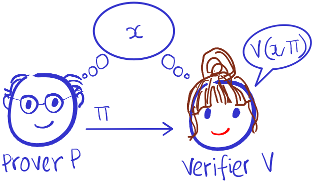
- Corresponds to the *class* NP
  - A language $\mathcal{L} \in$ NP if there exists a polynomial-time *deterministic* machine V such that

statement $\qquad$ witness/proof

$$\forall x \in \mathcal{L} \; \exists \text{ "short" } \pi : \mathrm{V}(x, \pi) = 1$$

  - NP is the *class* of all such $\mathcal{L}$s (e.g., Sudoku)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



- "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

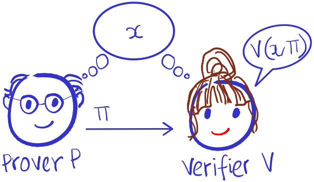
Traditional "NP" Proof...

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

statement witness/proof

$$\forall x \in \mathcal{L}\ \exists \text{ "short" } \pi : \mathrm{V}(x, \pi) = 1$$

  - NP is the *class* of all such $\mathcal{L}$s (e.g., Sudoku)

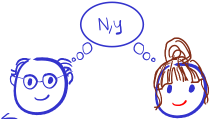


5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- "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 $\pi \Rightarrow \mathrm{V}(x, \pi) = 1$
  - Soundness: $x \notin \mathcal{L} \Rightarrow \nexists$ "short" π s.t. $\mathrm{V}(x, \pi) = 1$

Which Languages have "NP" Proofs?

Quadratic residuosity (QR)

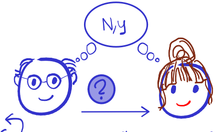


pq

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

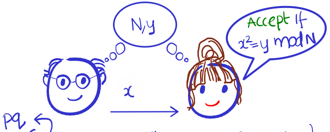
Which Languages have "NP" Proofs?

Quadratic residuosity (QR)



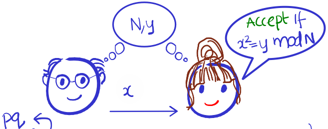
$\mathcal{L}_{QR} = \{(N, y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Quadratic residuosity (QR)



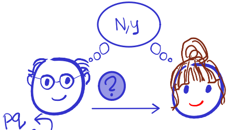
$$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Quadratic residuosity (QR)



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

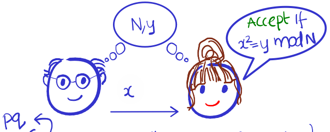
Quad. non-residuosity (QNR)



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

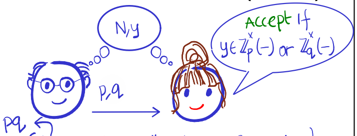
Which Languages have “NP” Proofs?

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?

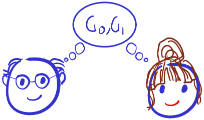
Quadratic residuosity (QR)

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

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?

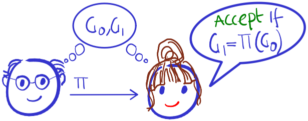
Quadratic residuosity (QR)

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

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?

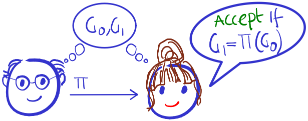
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

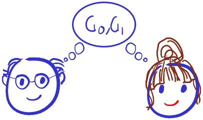
$$\mathcal{L}_{QNR} = \{(\overset{pq}{N}, y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

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)



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

Quadratic residuosity (QR)

p,q

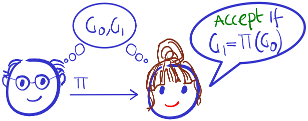
$\mathcal{L}_{QR}=\{(N,y): \exists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Quad. non-residuosity (QNR)

p,q

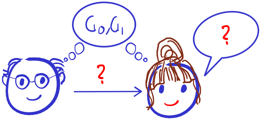
$\mathcal{L}_{QNR}=\{(N,y): \nexists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

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?

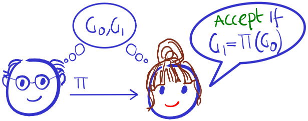
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

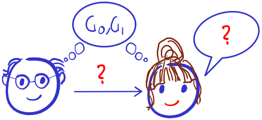
$\mathcal{L}_{QNR}=\{(N,y) : \nexists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Graph isomorphism (GI)



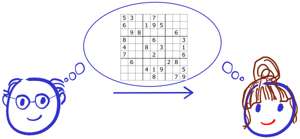
$\mathcal{L}_{GI}=\{(G_0,G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Graph non-isomorphism (GNI)



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

Sudoku



Chess

Which Languages have "NP" Proofs?

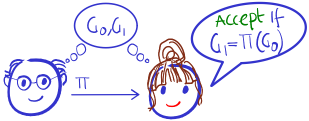
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

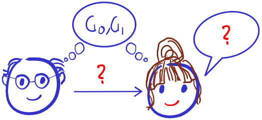
$$\mathcal{L}_{QNR}=\{(N,y):\nexists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$$

Graph isomorphism (GI)



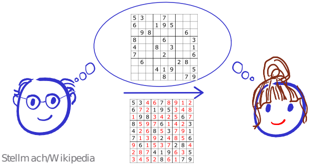
$$\mathcal{L}_{GI}=\{(G_0,G_1):\exists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$$

Graph non-isomorphism (GNI)



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

Sudoku



Chess

Which Languages have “NP” Proofs?

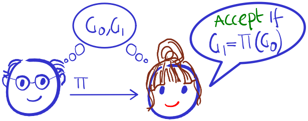
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

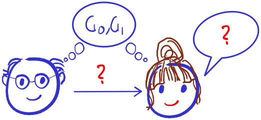
$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Graph isomorphism (GI)



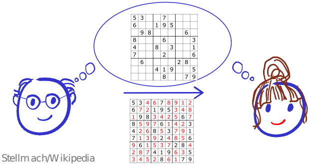
$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph non-isomorphism (GNI)

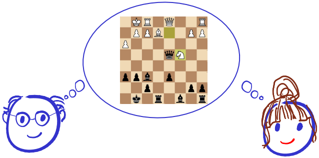


$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Sudoku



Chess



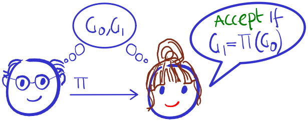
Quadratic residuosity (QR)

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

Quad. non-residuosity (QNR)

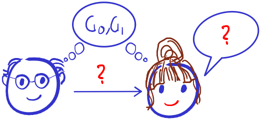
$\mathcal{L}_{QNR}=\{(N,y): \nexists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Graph isomorphism (GI)



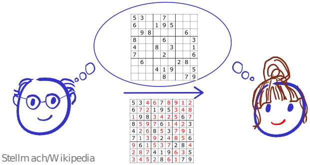
$\mathcal{L}_{GI}=\{(G_0,G_1): \exists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Graph non-isomorphism (GNI)

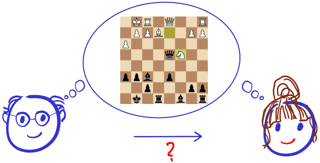


$\mathcal{L}_{GNI}=\{(G_0,G_1): \nexists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Sudoku

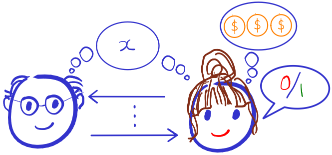


Chess



Interactive Proof (IP)

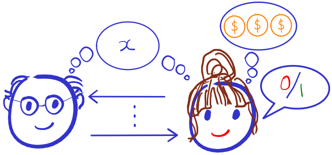
- Difference from NP proofs:
  1. Verifier V is *randomised*
  2. Prover P and V *interact* and V accepts/rejects in the end



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

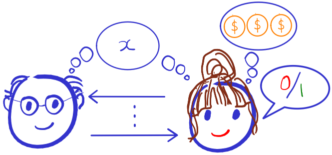


Definition 1

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

Difference from NP proofs:

1. Verifier V is *randomised*
2. Prover P and V *interact* and V accepts/rejects in the end



Definition 1

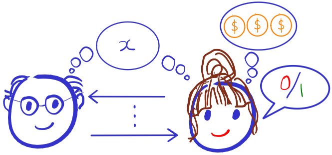
An interactive protocol (P, V) for a language $\mathcal{L}$ is an *interactive proof (IP)* system if the following holds:

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

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



Definition 1

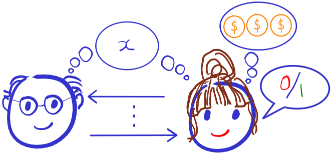
An interactive protocol (P, V) for a language $\mathcal{L}$ is an *interactive proof (IP)* system if the following holds:

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

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



Definition 1

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

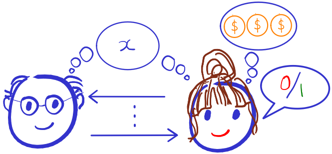
completeness error $\varepsilon_c(n)$

- **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 $\varepsilon_s(n)$

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



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

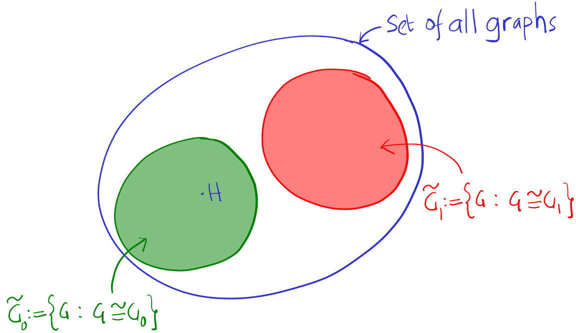
$$\Pr[1 \leftarrow \langle \mathrm{P}^*, \mathrm{V}\rangle(x)] \leq 1/3$$

soundness error $\varepsilon_s(n)$

Exercise 1 (Definition 1 is robust)

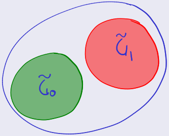
Show that languages captured by Definition 1 doesn't change when $\epsilon_c \leq 1/2^{|x|}$ and $\epsilon_s \leq 1/2^{|x|}$ (Hint: repeat protocol, use Chernoff bound)

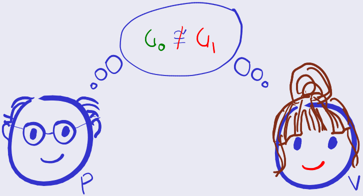
Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, *$G_0 \cong H$ and $G_1 \cong H$ both cannot hold*



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

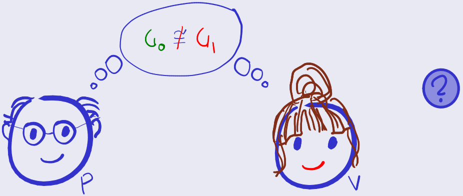
Protocol 1 (Π_{GNI}: IP for GNI)

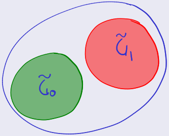




Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

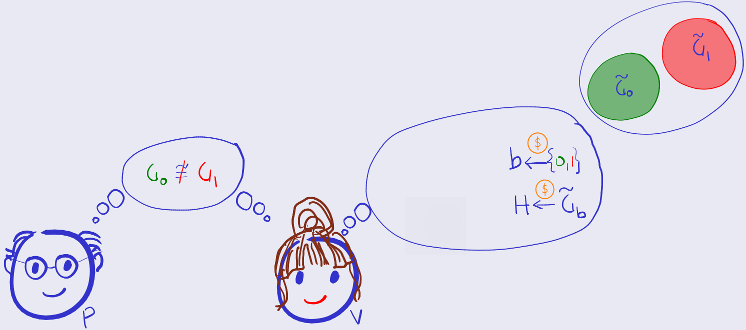
Protocol 1 (Π_{GNI}: IP for GNI)





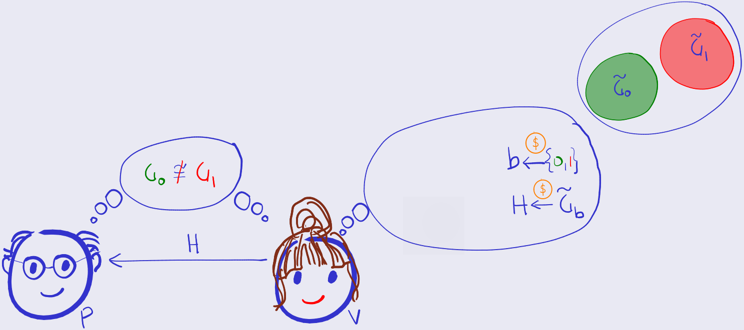
Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

Protocol 1 (Π_{GNI}: IP for GNI)



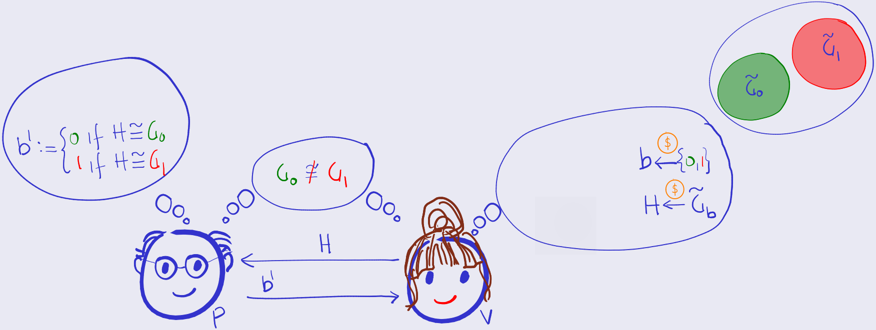
Idea: $G_0 \ncong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

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*

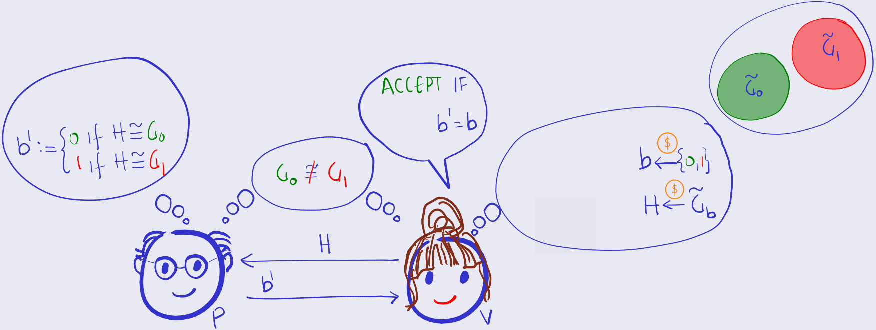
Protocol 1 (Π_{GNI}: IP for GNI)



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

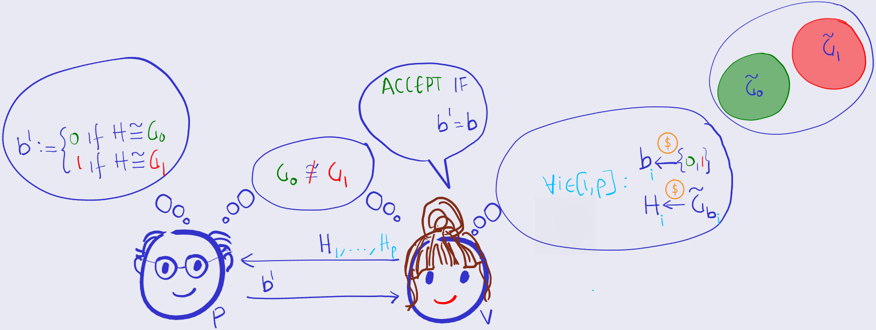
Idea: $G_0 \not\cong G_1 \Rightarrow$ for any graph H, $G_0 \cong H$ *and* $G_1 \cong H$ *both cannot hold*

Protocol 1 (Π_{GNI}: IP for GNI)



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)

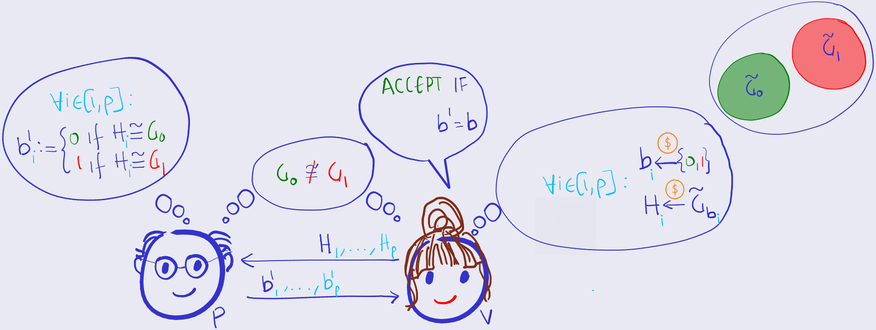


Parallel/sequentially repeat to boost soundness

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*

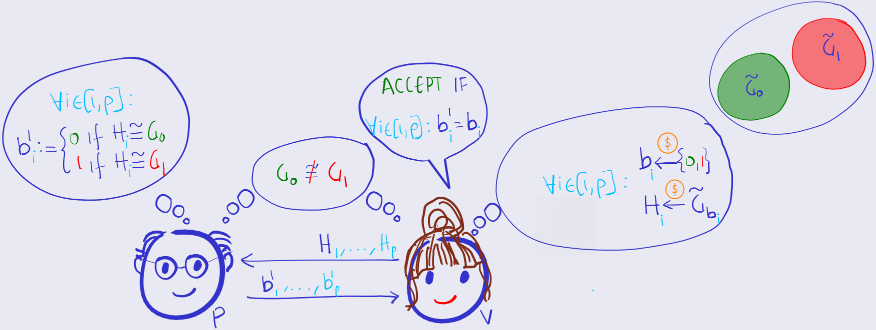
Protocol 1 (Π_{GNI}: IP for GNI)



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

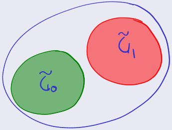
Π_{GNI} *is an IP for* $\mathcal{L}_{GNI}$

Theorem 1

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

Proof.

- Completeness:
  - $G_0 \ncong G_1 \Rightarrow$ P can recover b_i from H_i with certainty



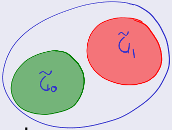
$$\Pr[1 \leftarrow \langle P, V\rangle(G_0, G_1)] = 1 \geq 2/3$$

Theorem 1

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

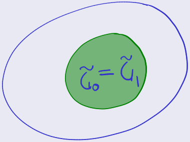
Proof.

- Completeness:
  - $G_0 \ncong G_1 \Rightarrow$ P can recover b_i from H_i with certainty



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

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

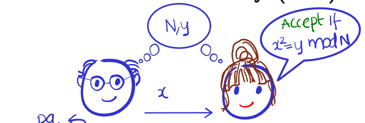


$$\Pr[1 \leftarrow \langle \mathsf{P}^*, \mathsf{V}\rangle(G_0, G_1)] = 1/2^{\rho} < 1/3$$

□

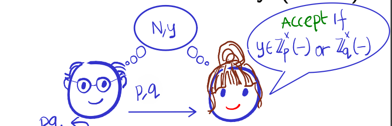
Which Languages have IPs?

Quadratic residuosity (QR)



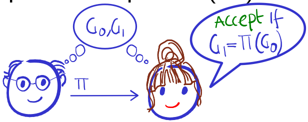
$\mathcal{L}_{QR}=\{(N,y): \exists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Quad. non-residuosity (QNR)



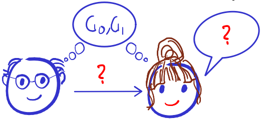
$\mathcal{L}_{QNR}=\{(N,y): \nexists x\in\mathbb{Z}_N^\times \text{ s.t. } y=x^2 \bmod N\}$

Graph isomorphism (GI)



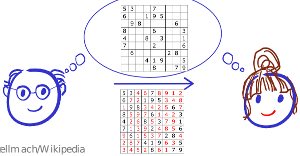
$\mathcal{L}_{GI}=\{(G_0,G_1): \exists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Graph non-isomorphism (GNI)

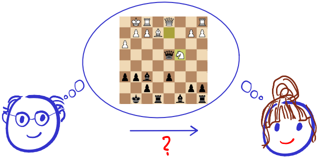


$\mathcal{L}_{GNI}=\{(G_0,G_1): \nexists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Sudoku



Chess



Which Languages have IPs?

Quadratic residuosity (QR)

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

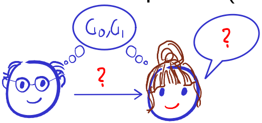
Quad. non-residuosity (QNR)

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

Graph isomorphism (GI)

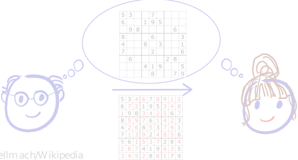
$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph non-isomorphism (GNI)

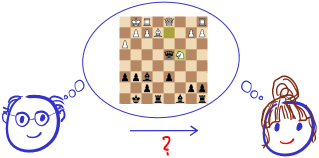


$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Sudoku



Chess



Which Languages have IPs?

Quadratic residuosity (QR)

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

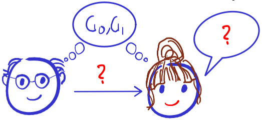
Quad. non-residuosity (QNR)

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

Graph isomorphism (GI)

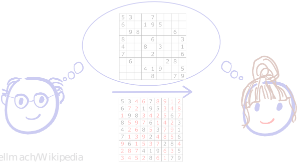
$\mathcal{L}_{GI}=\{(G_0,G_1):\exists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Graph non-isomorphism (GNI)

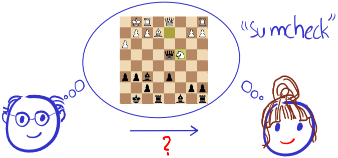


$\mathcal{L}_{GNI}=\{(G_0,G_1):\nexists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Sudoku



Chess



Which Languages have IPs? PSPACE Languages

Quadratic residuosity (QR)

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

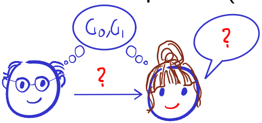
Quad. non-residuosity (QNR)

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

Graph isomorphism (GI)

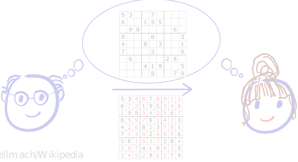
$\mathcal{L}_{GI}=\{(G_0,G_1): \exists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Graph non-isomorphism (GNI)

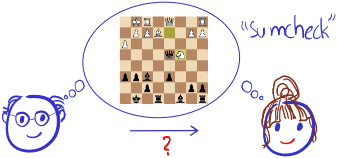


$\mathcal{L}_{GNI}=\{(G_0,G_1): \nexists \text{permutation } \pi \text{ s.t. } G_1=\pi(G_0)\}$

Sudoku



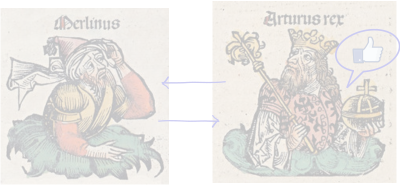
Chess



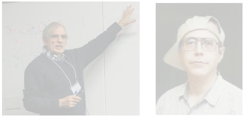
Plan for Today's Lecture...

What really constitutes a proof?

NEW Interactive Proof (IP)



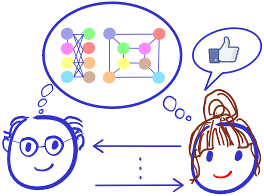
© Wikipedia

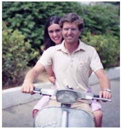


© blog.computationalcomplexity.org

© alchetron.com

Zero-Knowledge IP NEW



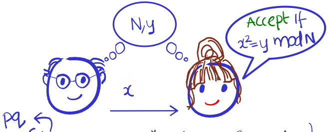


© Oded Goldreich

Main tools: simulation paradigm, Chernoff bound...

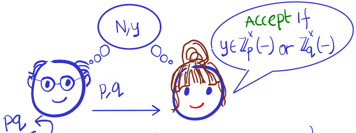
Any Issues with the NP Proofs We Saw?

Quadratic residuosity (QR)



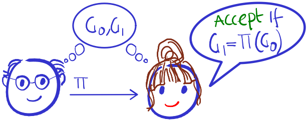
$$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Quad. non-residuosity (QNR)



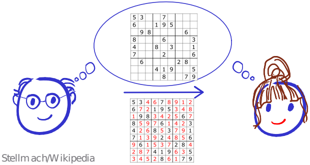
$$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$$

Graph isomorphism (GI)



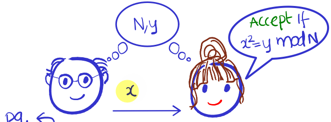
$$\mathcal{L}_{GI} = \{(G_0,G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Sudoku



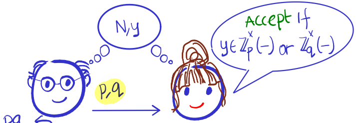
Any Issues with the NP Proofs We Saw?

Quadratic residuosity (QR)



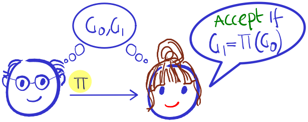
$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Quad. non-residuosity (QNR)



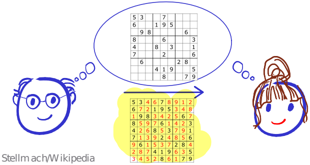
$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Graph isomorphism (GI)



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

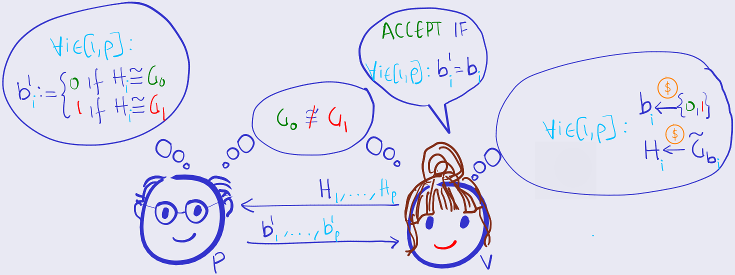
Sudoku



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

What About the IP We Saw?

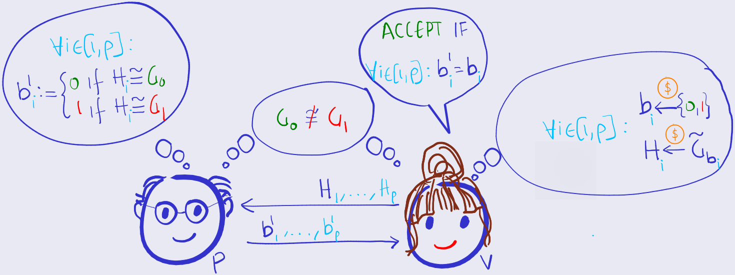
Protocol 1 (Π_{GNI}: IP for GNI)



Parallel/sequentially repeat to boost soundness

What About the IP We Saw?

Protocol 1 (Π_{GNI}: IP for GNI)

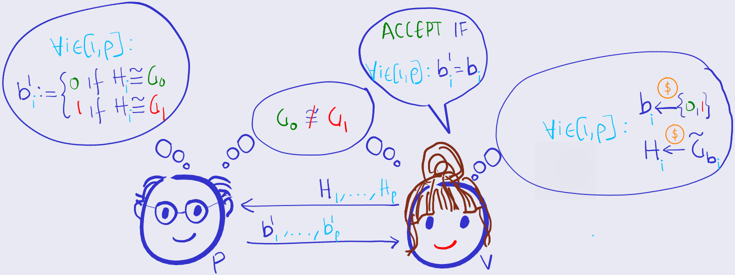


Parallel/sequentially repeat to boost soundness

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

What About the IP We Saw?

Protocol 1 (Π_{GNI}: IP for GNI)



Parallel/sequentially repeat to boost soundness

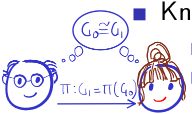
- Seems V *gains no knowledge* beyond validity of the statement
- We will see that Π_{GNI} is (honest-verifier) zero-knowledge!

How to Capture "V Gains No Knowledge"?

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

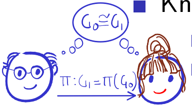
How to Capture "V Gains No Knowledge"?

- 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

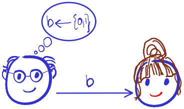


How to Capture "V Gains No Knowledge"?

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



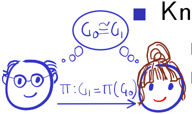
    - Given (G_0, G_1), the isomorphism π contains no *information*
    - But when given π, V "gains knowledge" since she couldn't have computed π herself
  - Knowledge pertains to public objects:



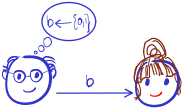
    - Flipping a *private* fair coin b and (later) revealing its outcome leads to V gaining *information*
    - But V *does not gain knowledge*: she could herself have tossed the private coin and revealed it

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:

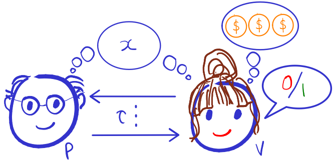


    - Flipping a *private* fair coin b and (later) revealing its outcome leads to V gaining *information*
    - But V *does not gain knowledge*: she could herself have tossed the private coin and revealed it

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*

Defining Zero Knowledge via Simulators...

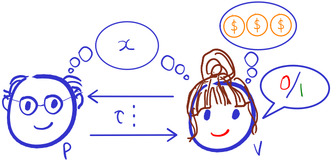
- Formalised via "simulation paradigm": $\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V}\rangle(x))$ can be *efficiently* simulated given only the instance



Defining Zero Knowledge via Simulators...

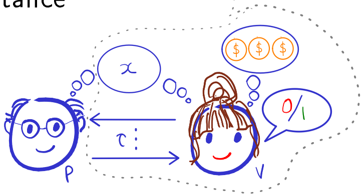
V's "view" = x + transcript τ + coins

- 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

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



Defining Zero Knowledge via Simulators...

V's "view" = x + transcript τ + coins

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

Definition 2 (Honest-Verifier *Perfect* ZK)

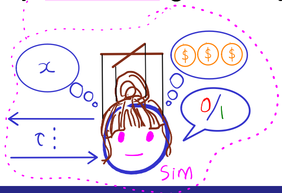
An IP Π is *honest-verifier perfect* ZK if there exists a PPT *simulator* Sim such that for *all* distinguishers D and all $x \in \mathcal{L}$, the following is zero

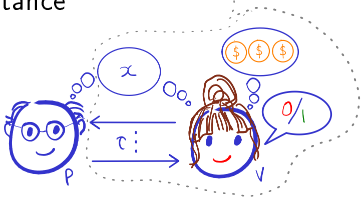
$$\Pr[\mathsf{D}(\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle (x))) = 1] - \Pr[\mathsf{D}(\mathsf{Sim}(x)) = 1]$$

Defining Zero Knowledge via Simulators...

V's "view" = x + transcript τ + coins

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





Definition 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

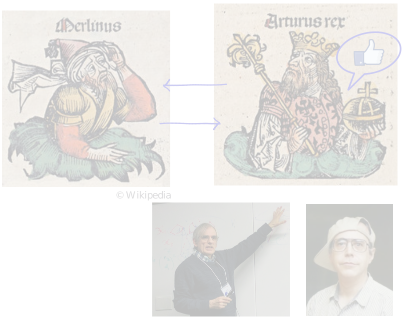
$$\Pr[\mathsf{D}(\mathit{View}_{\mathsf{V}}(\langle \mathsf{P}, \mathsf{V} \rangle (x))) = 1] - \Pr[\mathsf{D}(\mathsf{Sim}(x)) = 1]$$

Exercise 2

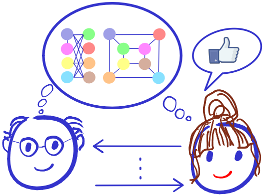
What happens when one invokes the simulator on $x \notin \mathcal{L}$?

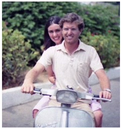
What really constitutes a proof?

Interactive Proof (IP)



Zero-Knowledge IP



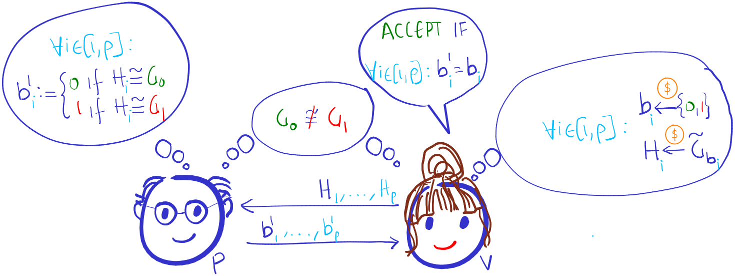


Main tools: simulation paradigm, Chernoff bound...

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

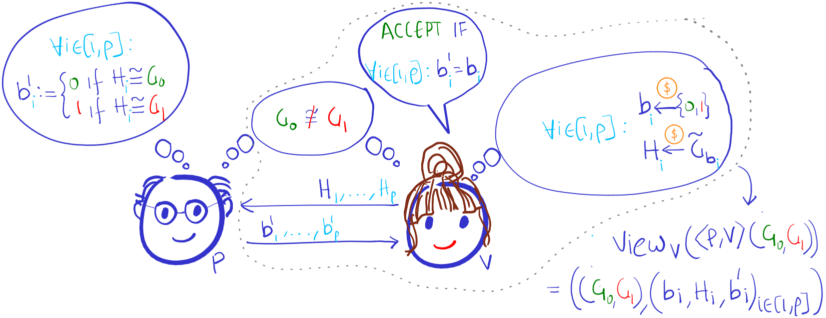


Parallel/sequentially repeat to boost soundness

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.

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

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

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

Proof.

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

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

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

Π_{GNI} is *Honest-Verifier* ZK

Theorem 2

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

Proof.

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

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

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

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.

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

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

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

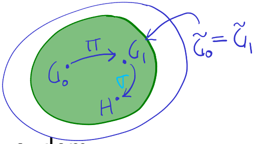
$$\text{o/p } \big((G_0,G_1),(b_i,H_i,b_i)_{i\in[1,\rho]}\big)$$

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

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 $\mathcal{L}_{QNR}$

Honest-Verifier ZKP for GI...

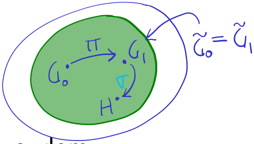


Idea for ZK:

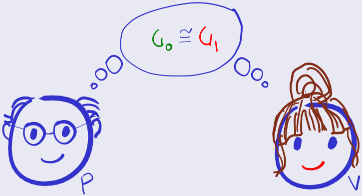
1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

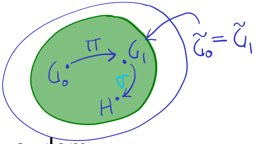
Idea for ZK:

1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random



Protocol 2 (Π_{GI}: IP for GI)

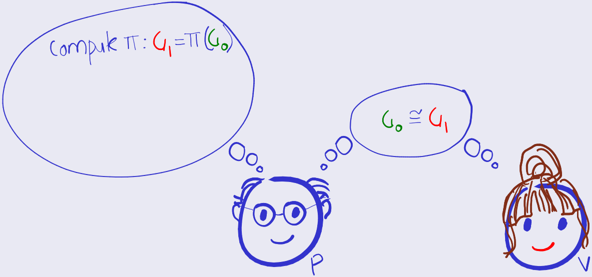


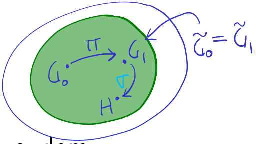


Idea for ZK:

1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)

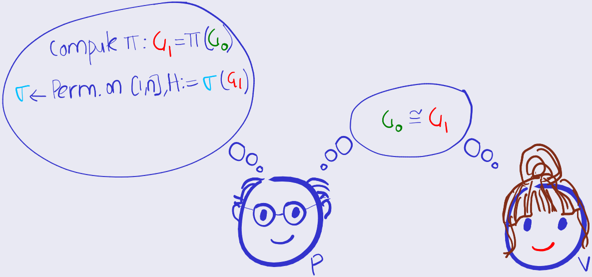




Idea for ZK:

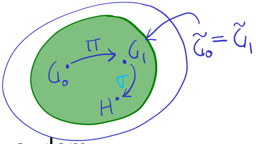
1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)

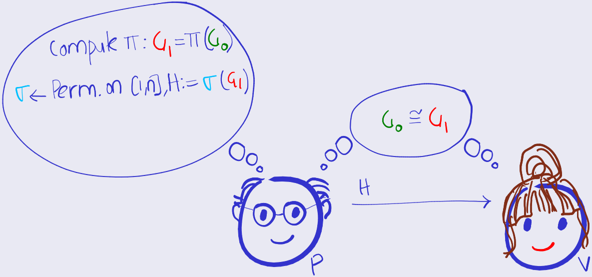


Idea for ZK:

1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

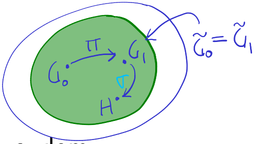


Protocol 2 (Π_{GI}: IP for GI)

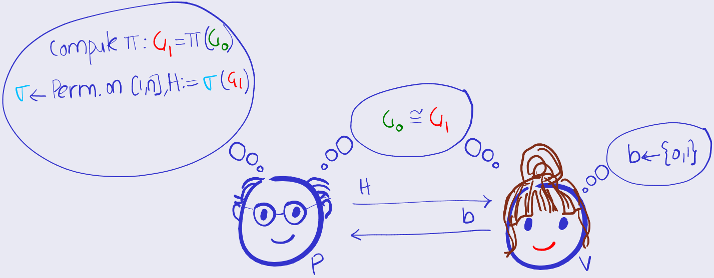


Idea for ZK:

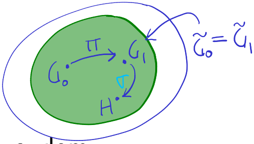
1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random



Protocol 2 (Π_{GI}: IP for GI)



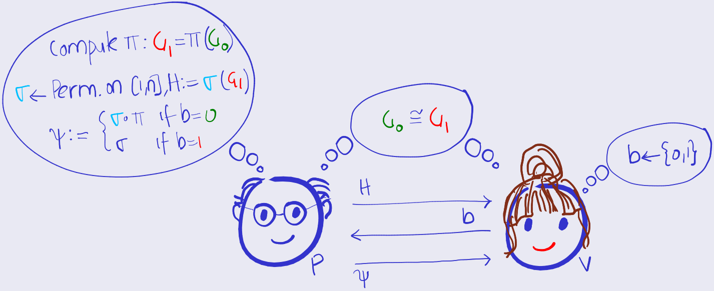
Honest-Verifier ZKP for GI...



Idea for ZK:

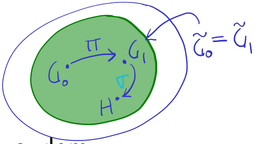
1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)

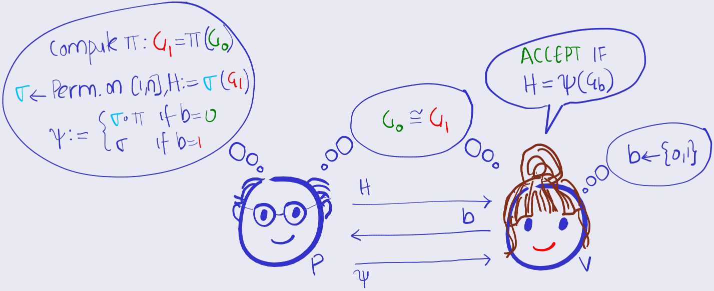


Idea for ZK:

1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random



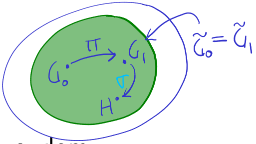
Protocol 2 (Π_{GI}: IP for GI)



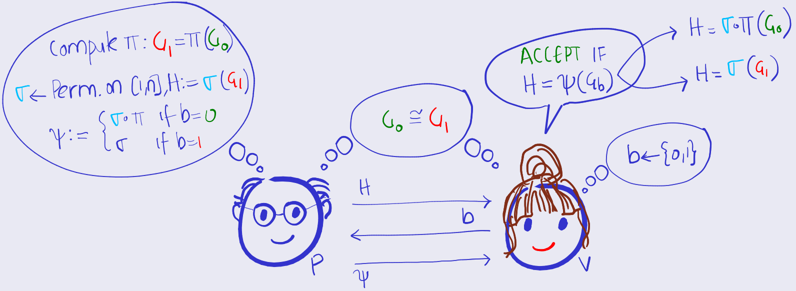
Honest-Verifier ZKP for GI

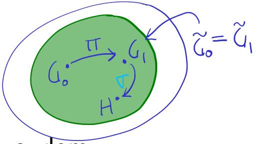
Idea for ZK:

1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random



Protocol 2 (Π_{GI}: IP for GI)

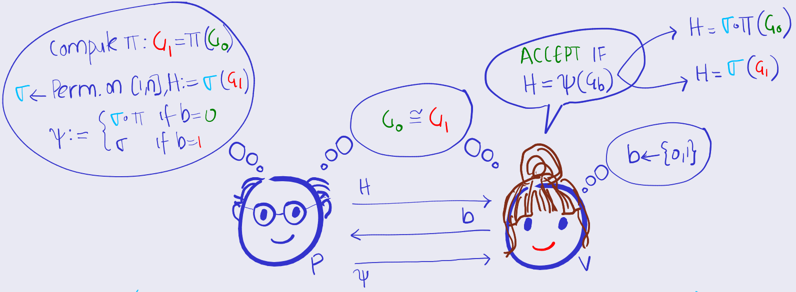




Idea for ZK:

1. $G_0 \overset{\pi}{\cong} G_1 \Rightarrow$ if $G_1 \overset{\sigma}{\cong} H$ then $G_0 \overset{\sigma\circ\pi}{\cong} H$
2. Prover sends a random H s.t. $G_1 \overset{\sigma}{\cong} H$
3. Verifier asks to prove $G_0 \overset{\sigma\circ\pi}{\cong} H$ or $G_1 \overset{\sigma}{\cong} H$ at random

Protocol 2 (Π_{GI}: IP for GI)



(Parallel/sequentially repeat to boost soundness)

Honest-Verifier ZKP for GI...

Theorem 3

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

Honest-Verifier ZKP for GI...

Theorem 3

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 3

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:

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

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

Theorem 3

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:

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

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

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

Theorem 3

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:

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

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

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

Honest-Verifier ZKP for GI...

Theorem 3

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: $G_0 \not\cong G_1 \Rightarrow$ 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:

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

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

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

set $H := \psi(G_b)$

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

$\forall\, G_0 \cong G_1$: $\mathrm{View}_V(\langle P,V\rangle(G_0,G_1))$ identically distributed to $\mathrm{Sim}(G_0,G_1)$. □

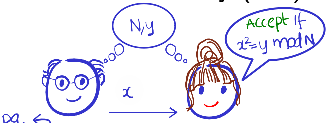
Honest-Verifier ZKP for GI...

Exercise 4

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

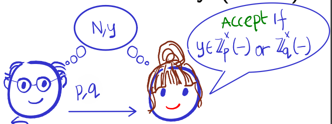
Which Languages have ZKPs?

Quadratic residuosity (QR)



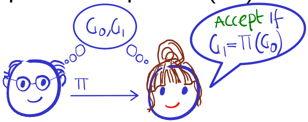
$\mathcal{L}_{QR} = \{(N,y) : \exists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Quad. non-residuosity (QNR)



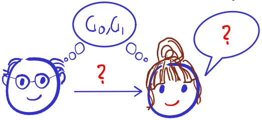
$\mathcal{L}_{QNR} = \{(N,y) : \nexists x \in \mathbb{Z}_N^\times \text{ s.t. } y = x^2 \bmod N\}$

Graph isomorphism (GI)



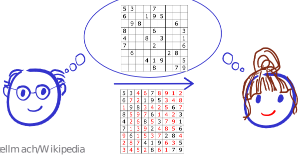
$\mathcal{L}_{GI} = \{(G_0,G_1) : \exists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph non-isomorphism (GNI)

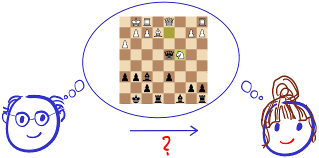


$\mathcal{L}_{GNI} = \{(G_0,G_1) : \nexists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Sudoku



Chess



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)

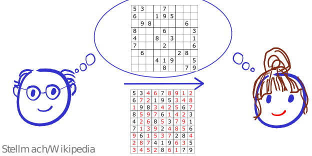
$$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

Graph non-isomorphism (GNI)

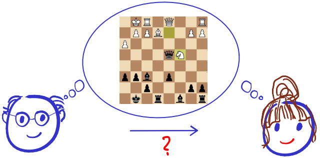
$$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$$

L19

Sudoku



Chess



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)

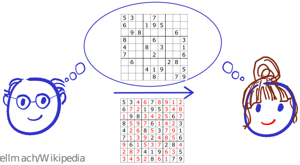
$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph non-isomorphism (GNI)

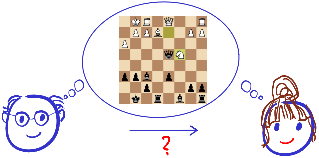
$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

L19

Sudoku



Chess



Are Randomness and Interaction Necessary?

$\leftrightarrows$ Interaction is necessary

Fact 4

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

Are Randomness and Interaction Necessary?

Interaction is necessary

Fact 4

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

Randomness is necessary

Exercise 5

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

Fact 5

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

Recap/Next Lecture

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

Recap/Next 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 (A5: QNR) and GI (A5: QR)

Recap/Next 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 (A5: QNR) and GI (A5: QR)

- Next Lecture:
  - Computational ZKP for all of NP!
    - New cryptographic primitive: commitment schemes
  - Return to ID protocols: zero-knowledge proof of knowledge

References

1. [Gol01, Chapter 4] 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. IP for all of PSPACE is due to [LFKN92, Sha90]. 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.