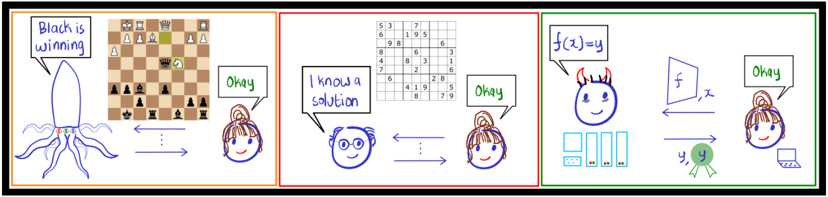


CS789: Introduction to Probabilistic Proof Systems

Lecture 1 (06/Jan/25)

Instructor: Chethan Kamath

Administrivia...

- Timing and Venue: Slot 8 (14:00-15:25, Mondays and Thursdays) in CC101
- Contact hours: drop by my office (CC305) any time!
- Teaching assistants: TBA

Administrivia...

- Timing and Venue: Slot 8 (14:00-15:25, Mondays and Thursdays) in CC101
- Contact hours: drop by my office (CC305) any time!
- Teaching assistants: TBA

- Resources
  - Slides and other resources will be posted on course website
    - cse.iitb.ac.in/~ckamath/courses/2025s/CS789.html
  - Announcements/online discussion on Moodle:
    - moodle.iitb.ac.in/course/view.php?id=5986

Administrivia...

- Grading Scheme

Weightage	Towards
30%	End-sem
25%	Mid-sem
20%	Paper presentation (two students per paper)
15%	Three graded assignments
5%	Class participation
5%	Scribing (~2 lectures per student)

  - Attendance is not mandatory (but encouraged)

Administrivia...

- Grading Scheme

Weightage	Towards
40% 30%	End-sem
35% 25%	Mid-sem
~~20%~~	~~Paper presentation (two students per paper)~~
15%	Three graded assignments
5%	Class participation
5%	Scribing (~2 lectures per student)

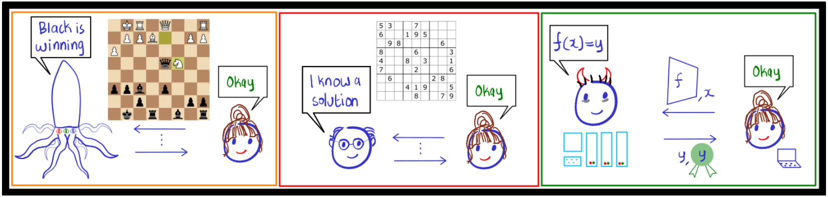
  - Attendance is not mandatory (but encouraged)
  - Will scrap paper presentation and readjust grades if number of creditors > 20

Administrivia...

- Grading Scheme

Weightage	Towards
40% 30%	End-sem
35% 25%	Mid-sem
~~20%~~	~~Paper presentation (two students per paper)~~
15%	Three graded assignments
5%	Class participation
5%	Scribing (~2 lectures per student)

  - Attendance is not mandatory (but encouraged)
  - Will scrap paper presentation and readjust grades if number of creditors > 20
- Any volunteers for class rep?



CS789: Introduction to Probabilistic Proof Systems

Lecture 1 (06/Jan/25)

Instructor: Chethan Kamath

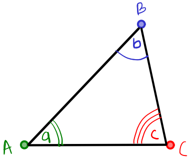
Traditional (Mathematical) Proofs...

- Taught in *CS208: Automata Theory and Logic*

- Taught in *CS208: Automata Theory and Logic*
- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

    $\downarrow$

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

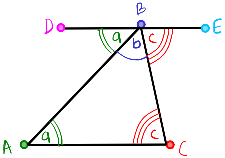


Traditional (Mathematical) Proofs...

- Taught in *CS208: Automata Theory and Logic*
- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

    $\downarrow$

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



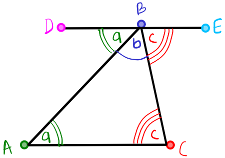
- Prover vs. verifier
  - Prover does the heavy lifting: *derives* the proof
    1. Construct a line $\overline{DE}$ through B, parallel to $\overline{AC}$
    2. $\angle DBA = \angle a$ and $\angle EBC = \angle c$ (alternate interior angles)
    3. $2 \Rightarrow \angle a + \angle b + \angle c = \angle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

Traditional (Mathematical) Proofs...

- Taught in *CS208: Automata Theory and Logic*
- Axioms $\xrightarrow{\text{derivation rules}}$ theorems=true statements
  - E.g.: Axioms of Euclidean geometry

    $\downarrow$

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



- Prover vs. verifier
  - Prover does the heavy lifting: *derives* the proof
    1. Construct a line $\overline{DE}$ 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 can *check/verify* the proof, step by step

Traditional (Mathematical) Proofs...

- Can be considered to correspond to class **NP**
  - A language $\mathcal{L} \in \mathbf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement

witness/proof

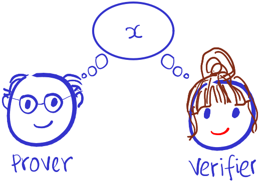
$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathbf{poly}(|x|)} : \mathsf{V}(x, w) = 1$$

Traditional (Mathematical) Proofs...

- Can be considered to correspond to class **NP**
  - A language $\mathcal{L} \in \mathbf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement witness/proof

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



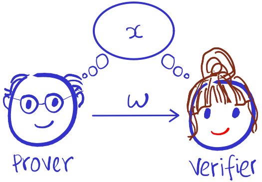
- "Proof system" view of **NP**
  - Prover is *unbounded*: finds witness w for x (if one exists)
  - Verifier is *efficient*: checks whether $\mathsf{V}(x, w) = 1$

Traditional (Mathematical) Proofs...

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

statement witness/proof

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



- "Proof system" view of **NP**
  - Prover is *unbounded*: finds witness w for x (if one exists)
  - Verifier is *efficient*: checks whether $\mathsf{V}(x, w) = 1$

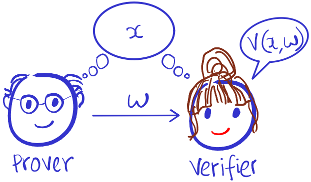
Traditional (Mathematical) Proofs...

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

witness/proof

statement

$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathbf{poly}(|x|)} : \mathrm{V}(x, w) = 1$$

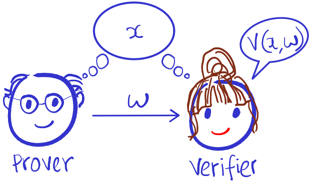


- "Proof system" view of **NP**
  - Prover is *unbounded*: finds witness w for x (if one exists)
  - Verifier is *efficient*: checks whether $\mathrm{V}(x, w) = 1$

- Can be considered to correspond to class **NP**
  - A language $\mathcal{L} \in \mathbf{NP}$ if there exists a polynomial-time *deterministic* machine V such that

statement   witness/proof

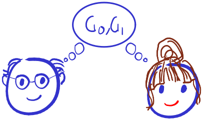
$$x \in \mathcal{L} \Leftrightarrow \exists w \in \{0,1\}^{\mathbf{poly}(|x|)} : \mathsf{V}(x, w) = 1$$



- "Proof system" view of **NP**
  - Prover is *unbounded*: finds witness w for x (if one exists)
  - Verifier is *efficient*: checks whether $\mathsf{V}(x, w) = 1$
  - Completeness: $x \in \mathcal{L} \Rightarrow$ prover finds $w \Rightarrow \mathsf{V}(x, w) = 1$
  - Soundness: $x \notin \mathcal{L} \Rightarrow \nexists w \in \{0,1\}^{\mathbf{poly}(|x|)}$ s.t. $\mathsf{V}(x, w) = 1$

Which Languages Have "**NP**" Proofs?

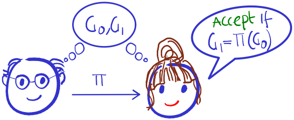
Graph isomorphism (GI)



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

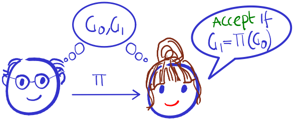
Which Languages Have "**NP**" Proofs?

Graph isomorphism (GI)



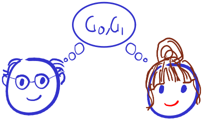
$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph isomorphism (GI)



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

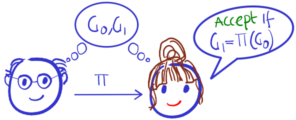
Graph non-isomorphism (GNI)



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

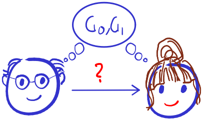
Which Languages Have "**NP**" Proofs?

Graph isomorphism (GI)



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

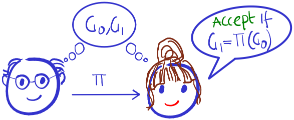
Graph non-isomorphism (GNI)



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

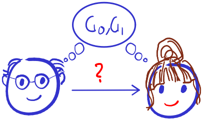
Which Languages Have "**NP**" Proofs?

Graph isomorphism (GI)



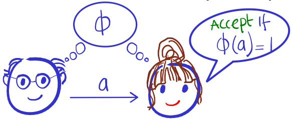
$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph non-isomorphism (GNI)



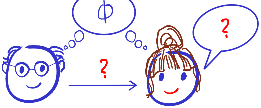
$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Boolean satisfiability (SAT)



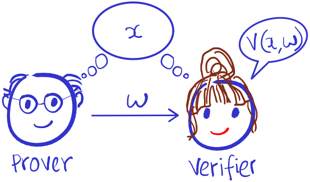
$\mathcal{L}_{SAT} = \{\phi : \exists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Bool. unsatisfiability (UNSAT)



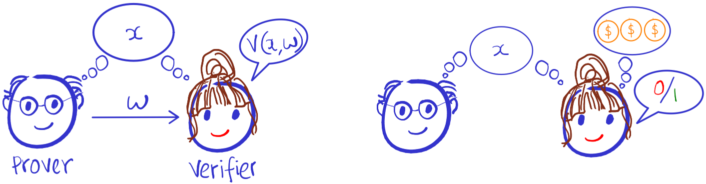
$\mathcal{L}_{UNSAT} = \{\phi : \nexists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

What Are *Probabilistic* Proofs?



- *Relaxation* of traditional notion of proof

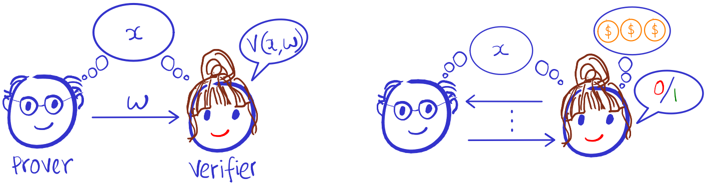
What Are *Probabilistic* Proofs?



- *Relaxation* of traditional notion of proof
- Key differences from traditional proofs:
  1. Verifier is *randomised*

     $\Longrightarrow$ Verifier may accept false statements ("soundness error")

What Are *Probabilistic* Proofs?



- *Relaxation* of traditional notion of proof
- Key differences from traditional proofs:
  1. Verifier is *randomised*

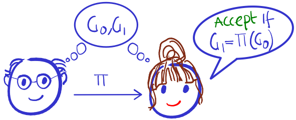
     $\Longrightarrow$ Verifier may accept false statements ("soundness error")
  2. Verifier may *interact* with the prover

     $\Longrightarrow$ proof not necessarily a string

Why Study Probabilistic Proofs?

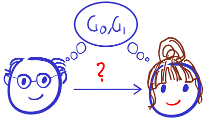
1. Far more expressive than traditional proofs

Graph isomorphism (GI)



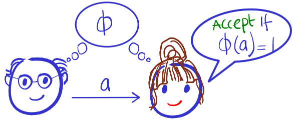
$\mathcal{L}_{GI} = \{(G_0, G_1) : \exists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Graph non-isomorphism (GNI)



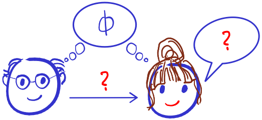
$\mathcal{L}_{GNI} = \{(G_0, G_1) : \nexists \text{ permutation } \pi \text{ s.t. } G_1 = \pi(G_0)\}$

Boolean satisfiability (SAT)



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

Bool. unsatisfiability (UNSAT)

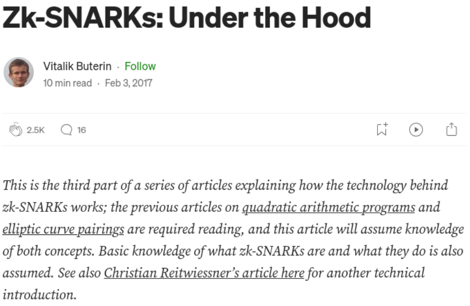


$\mathcal{L}_{UNSAT} = \{\phi : \nexists \text{ assignment } a \text{ s.t. } \phi(a) = 1\}$

Prob. proofs known

2 Extremely useful for real-world applications

- Blockchain applications, verifiable computation etc



Zk-SNARKs: Under the Hood

Vitalik Buterin · Follow
10 min read · Feb 3, 2017

2.5K 16

This is the third part of a series of articles explaining how the technology behind zk-SNARKs works; the previous articles on quadratic arithmetic programs and elliptic curve pairings are required reading, and this article will assume knowledge of both concepts. Basic knowledge of what zk-SNARKs are and what they do is also assumed. See also Christian Reitwiessner's article here for another technical introduction.

Why Study Probabilistic Proofs?

2 Extremely useful for real-world applications

- Blockchain applications, verifiable computation etc

Zk-SNARKs: Under the Hood

Vitalik Buterin · Follow
10 min read · Feb 3, 2017

2.5K 16

This is the third part of a series of articles explaining how the technology behind zk-SNARKs works; the previous articles on quadratic arithmetic programs and elliptic curve pairings are required reading, and this article will assume knowledge of both concepts. Basic knowledge of what zk-SNARKs are and what they do is also assumed. See also Christian Reitwiessner's article here for another technical introduction.

STARKWARE Microsoft | Research polygon

Succinct arithmic

3 Still an area of active research

Quanta magazine

PROOFS

Computer Scientists Combine Two 'Beautiful' Proof Methods

Ben Brubaker

Staff Writer

October 4, 2024

How do you prove something is true? For mathematicians, the answer is simple: Start with some basic assumptions and proceed, step by step, to the conclusion. QED, proof complete. If there's a mistake anywhere, an expert who reads the proof carefully should be able to spot it. Otherwise, the proof must be valid. Mathematicians have been following this basic approach for well over 2,000 years.

About this Course: What to Expect?

- Goal: *formally* study several types of probabilistic proofs

About this Course: What to Expect?

- Goal: *formally* study several types of probabilistic proofs
  1. Introduce the formal model (e.g., interactive proof)
  2. Construct proof for various problems of interest (e.g., UNSAT)
  3. Formally prove its properties (e.g., soundness)

About this Course: What to Expect?

- Goal: *formally* study several types of probabilistic proofs
  1. Introduce the formal model (e.g., interactive proof)
  2. Construct proof for various problems of interest (e.g., UNSAT)
  3. Formally prove its properties (e.g., soundness)

- *Advanced*, theoretical course
  - Prerequisites: discrete mathematics, probability theory, familiarity with formal proofs
  - Soft prerequisites: basic complexity theory and cryptography

About this Course: What to Expect?

- Goal: *formally* study several types of probabilistic proofs
  1. Introduce the formal model (e.g., interactive proof)
  2. Construct proof for various problems of interest (e.g., UNSAT)
  3. Formally prove its properties (e.g., soundness)

- *Advanced*, theoretical course
  - Prerequisites: discrete mathematics, probability theory, familiarity with formal proofs
  - Soft prerequisites: basic complexity theory and cryptography
  - Focus on depth

About this Course: What to Expect?

- Goal: *formally* study several types of probabilistic proofs
  1. Introduce the formal model (e.g., interactive proof)
  2. Construct proof for various problems of interest (e.g., UNSAT)
  3. Formally prove its properties (e.g., soundness)

- *Advanced*, theoretical course
  - Prerequisites: discrete mathematics, probability theory, familiarity with formal proofs
  - Soft prerequisites: basic complexity theory and cryptography
  - Focus on depth

- You will enjoy the course if you enjoyed other theory courses (CS105, CS208, CS760, CS783)
  - We'll encounter lots of new tools

This Lecture: An Overview of the Course

1 Module I: Interactive Proof

This Lecture: An Overview of the Course

1 Module I: Interactive Proof

2 Module II: Zero Knowledge, Probabilistically-Checkable Proof

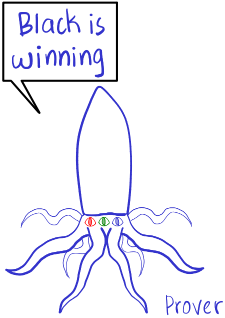
This Lecture: An Overview of the Course

1. Module I: Interactive Proof
2. Module II: Zero Knowledge, Probabilistically-Checkable Proof
3. Module III: Cryptographic Proof Systems

This Lecture: An Overview of the Course

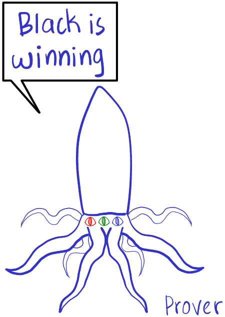
1. Module I: Interactive Proof
2. Module II: Zero Knowledge, Probabilistically-Checkable Proof
3. Module III: Cryptographic Proof Systems

Interactive Proof (IP)...

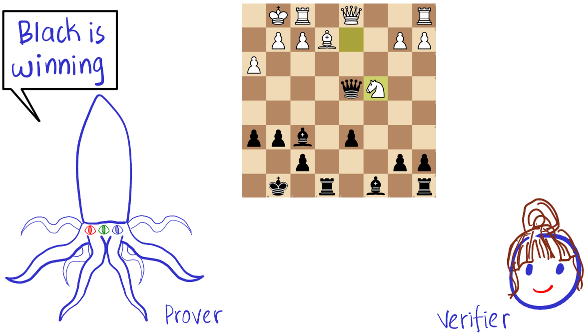


- When can we say "this position is winning for black"?

Interactive Proof (IP)...

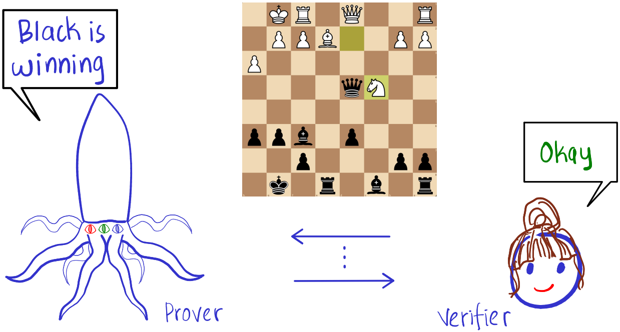


- When can we say "this position is winning for black"?
- What could a traditional proof for this statement look like?



- When can we say "this position is winning for black"?
- What could a traditional proof for this statement look like?
  - Seems too big to write down

Interactive Proof (IP)...



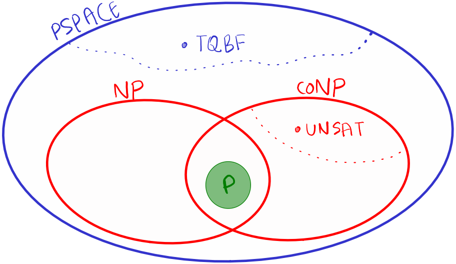
- When can we say "this position is winning for black"?
- What could a traditional proof for this statement look like?
  - Seems too big to write down
- We will learn how a verifier can *interactively check* that black is winning!

Interactive Proof (IP)...

- IP is powerful. In this course, we'll construct IP for:
  - UNSAT, the celebrated *Sumcheck Protocol*
  - TQBF, which captures polynomial-space computations!

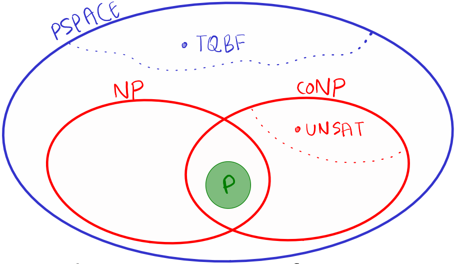
Interactive Proof (IP)...

- IP is powerful. In this course, we'll construct IP for:
  - UNSAT, the celebrated *Sumcheck Protocol*
  - TQBF, which captures polynomial-space computations!



Interactive Proof (IP)...

- IP is powerful. In this course, we'll construct IP for:
  - UNSAT, the celebrated *Sumcheck Protocol*
  - TQBF, which captures polynomial-space computations!



- Some issues with interactive proof:
  - Requires interaction
    - Undesirable in practical applications (latency)
  - May leak unnecessary information
    - Undesirable, e.g., when the witness is a secret

This Lecture: An Overview of the Course

1 Module I: Interactive Proof

2 Module II: Zero Knowledge, Probabilistically-Checkable Proof

3 Module III: Cryptographic Proof Systems

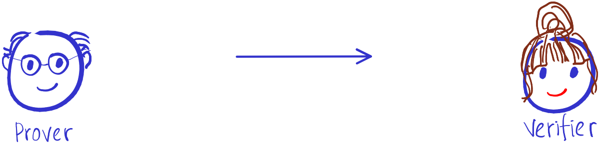
Zero-Knowledge (ZK) IP

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

- **NP** proof *reveals* the witness

Zero-Knowledge (ZK) IP

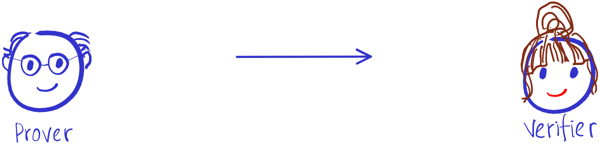
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



- **NP** proof *reveals* the witness

Zero-Knowledge (ZK) IP

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



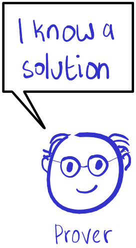
- **NP** proof *reveals* the witness
- Can a prover convince the verifier that it knows a witness *without leaking any information about it*?

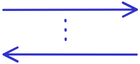
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

- **NP** proof *reveals* the witness
- Can a prover convince the verifier that it knows a witness *without leaking any information about it*?
- We will learn how this is possible using *interaction*

Zero-Knowledge (ZK) IP

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

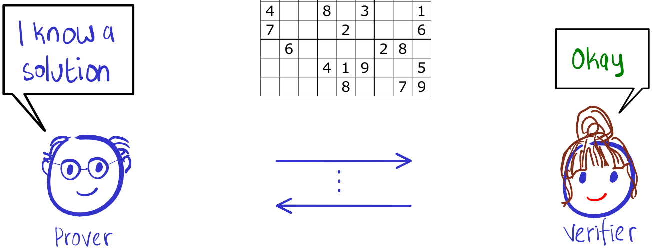




- **NP** proof *reveals* the witness
- Can a prover convince the verifier that it knows a witness *without leaking any information about it*?
- We will learn how this is possible using *interaction*

Zero-Knowledge (ZK) IP

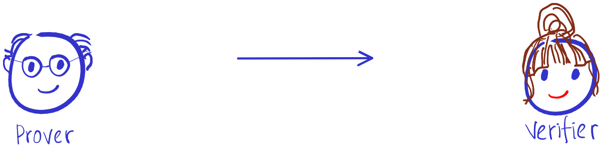
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9



- **NP** proof *reveals* the witness
- Can a prover convince the verifier that it knows a witness *without leaking any information about it*?
- We will learn how this is possible using *interaction*
  - Construct ZK IP for GNI and some number-theoretic problems
  - Study class **SZK**, which is important for cryptography

Probabilistically-Checkable Proof (PCP)...

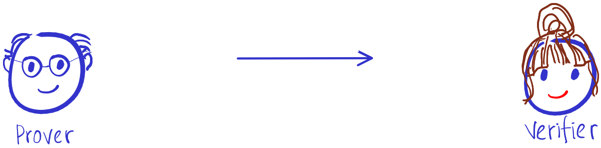
5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9



- To verify **NP** proof, verifier must check *the whole witness*

Probabilistically-Checkable Proof (PCP)...

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



- To verify **NP** proof, verifier must check *the whole witness*
- In PCPs, the proof is written such that the verifier needs to check only *a few* random parts of the proof!

Probabilistically-Checkable Proof (PCP)...

- In this course, we will:
  - Construct PCP for **NP** (the celebrated PCP Theorem)

- In this course, we will:
  - Construct PCP for **NP** (the celebrated PCP Theorem)

nature

News | Published: 10 September 2012

Proof claimed for deep connection between primes

Philip Ball

Nature (2012) | Cite this article

7407 Accesses | **1372** Altmetric | Metrics

If it is true, a solution to the *abc* conjecture about whole numbers would be an 'astounding' achievement.

Mathematician Shinichi Mochizuki of Kyoto University in Japan has released a 500-page proof of the *abc* conjecture, which proposes a relationship between whole numbers – a 'Diophantine' problem.

This Lecture: An Overview of the Course

What are *Cryptographic* Proof Systems?

- Probabilistic proofs where certain guarantees (e.g., soundness) hold only under *cryptographic hardness assumptions*
  - E.g.: hardness of factoring integers for probabilistic polynomial-time algorithms

What are *Cryptographic* Proof Systems?

- Probabilistic proofs where certain guarantees (e.g., soundness) hold only under *cryptographic hardness assumptions*
  - E.g.: hardness of factoring integers for probabilistic polynomial-time algorithms

- What is the motivation/necessity?

What are *Cryptographic* Proof Systems?

- Probabilistic proofs where certain guarantees (e.g., soundness) hold only under *cryptographic hardness assumptions*
  - E.g.: hardness of factoring integers for probabilistic polynomial-time algorithms

- What is the motivation/necessity?
  1. Bypass impossibility results or limitations
     - E.g.: (Statistical) ZK IP for **NP** unlikely to exist

What are *Cryptographic* Proof Systems?

- Probabilistic proofs where certain guarantees (e.g., soundness) hold only under *cryptographic hardness assumptions*
  - E.g.: hardness of factoring integers for probabilistic polynomial-time algorithms

- What is the motivation/necessity?
  1. Bypass impossibility results or limitations
     - E.g.: (Statistical) ZK IP for **NP** unlikely to exist
  2. Bridge from theory to practice, where emphasis is on efficiency
     - Efficient prover and verifier
     - Non-interactive
     - Short proofs (succinctness)

What are *Cryptographic* Proof Systems?

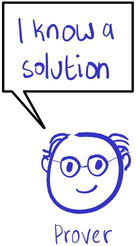
- Probabilistic proofs where certain guarantees (e.g., soundness) hold only under *cryptographic hardness assumptions*
  - E.g.: hardness of factoring integers for probabilistic polynomial-time algorithms

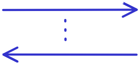
- What is the motivation/necessity?
  1. Bypass impossibility results or limitations
     - E.g.: (Statistical) ZK IP for **NP** unlikely to exist
  2. Bridge from theory to practice, where emphasis is on efficiency
     - Efficient prover and verifier
     - Non-interactive
     - Short proofs (succinctness)

     - Reasonable to assume all parties (including adversaries) are bounded in practice

(Non-Interactive) Computational Zero-Knowledge

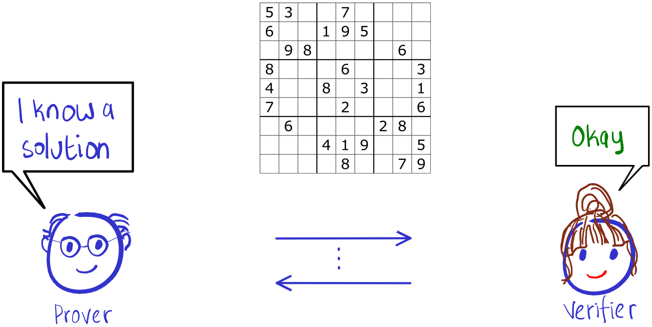
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9





- Recall: (Statistical) ZK IP for **NP** unlikely to exist

(Non-Interactive) Computational Zero-Knowledge



- Recall: (Statistical) ZK IP for **NP** unlikely to exist
- Under appropriate hardness assumptions, we construct
  - *Computational* ZK IP for **NP**
  - *Non-interactive* computational ZK for **NP**

Succinct Non-Interactive Arguments (SNArg)

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

- Recall: in **NP** proof, prover must send *whole witness*

Succinct Non-Interactive Arguments (SNArg)

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

- Recall: in **NP** proof, prover must send *whole witness*
- Can prover send something *shorter* and still convince verifier?

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

- Recall: in **NP** proof, prover must send *whole witness*
- Can prover send something *shorter* and still convince verifier?
- We will learn how by relaxing to *computational soundness*
    - Construction relies on PCP

Succinct Non-Interactive Arguments (SNArg)

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

- Recall: in **NP** proof, prover must send *whole witness*
- Can prover send something *shorter* and still convince verifier?
- We will learn how by relaxing to *computational soundness*
  - Construction relies on PCP
- Useful for verifiable computation

An Example

Probabilistically Checking Matrix Multiplication
(On whiteboard)

References

1. For a discussion on **NP** and mathematical proofs, see [AB09, §2.7.2]
2. Most of Module I will be based on Alessandro Chiesa's CS294 (Fall 2020)
3. Module II will largely be based on the above course and [Gol01, AB09]
4. For Module III we will mostly follow Justin Thaler's monograph [Tha22]
5. The description of Freivalds' algorithm here is from [Mat10, Miniature 11]
6. More resources can be found on the course website

Sanjeev Arora and Boaz Barak.
Computational Complexity - A Modern Approach.
Cambridge University Press, 2009.

Oded Goldreich.
The Foundations of Cryptography - Volume 1: Basic Techniques.
Cambridge University Press, 2001.

Jiří Matoušek.
Thirty-three Miniatures: Mathematical and Algorithmic Applications of Linear Algebra.
American Mathematical Society, 2010.

Justin Thaler.
Proofs, Arguments, and Zero-Knowledge.
https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html, 2022.