Introduction to (Zero-Knowledge) Proofs

Jonathan Katz
Google and University of Maryland

Proofs

[redacted]

Proofs

What is a proof of a (theorem) statement x?

Proofs

What is a proof of a (theorem) statement x?

- Static object

Proofs

What is a proof of a (theorem) statement x?

- Static object
- Verified by some deterministic procedure

Proofs

What is a proof of a (theorem) statement x?

- Static object
- Verified by some deterministic procedure
- False statements do not have proofs that verify

Proofs

What is a proof of a (theorem) statement x?

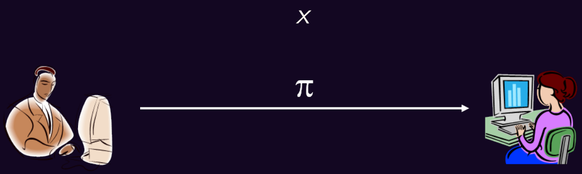
- Static object
- Verified by some deterministic procedure
- False statements do not have proofs that verify

x

Proofs

What is a proof of a (theorem) statement x?

- Static object
- Verified by some deterministic procedure
- False statements do not have proofs that verify



The class NP

A language $L \subseteq \{0,1\}^*$ is in NP if there is a deterministic verifier V_L running in polynomial time (in its first input) such that

$$x \in L \Leftrightarrow \exists \pi \text{ s.t. } V_L(x, \pi) = 1$$

The class NP

A language $L \subseteq \{0,1\}^*$ is in NP if there is a deterministic verifier V_L running in polynomial time (in its first input) such that

$$x \in L \Leftrightarrow \exists \pi \text{ s.t. } V_L(x, \pi) = 1$$

I.e.,

- Completeness: If $x \in L$ then there is a proof (aka a *witness*) π such that $V_L(x, \pi) = 1$
- Soundness: If $x \notin L$ then for all π^* we have $V_L(x, \pi^*) = 0$

Proofs

Why limit ourselves?

Proofs

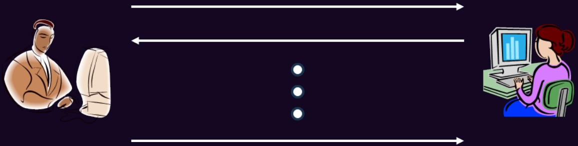
Why limit ourselves?

Traditional view	New view
Static object	Interactive process!
Deterministic verification	Allow randomization!
False statements do not have proofs that verify	Might accept proofs for false statements*

*with small probability

Proofs

x



Accept/reject

Proof systems and the class IP

A proof system for a language L is a pair of algorithms (P, V), where V runs in probabilistic, polynomial time (PPT), such that

1. Completeness: if $x \in L$ then for all λ we have

$$\Pr[\langle P, V\rangle(1^\lambda, x) = 1] = 1$$

2. Soundness: if $x \notin L$ then for all P^*, λ we have

$$\Pr[\langle P^*, V\rangle(1^\lambda, x) = 1] \leq 2^{-\lambda}$$

Proof systems and the class IP

A proof system for a language L is a pair of algorithms (P, V), where V runs in probabilistic, polynomial time (PPT), such that

1. Completeness: if $x \in L$ then for all λ we have
$$\Pr[\langle P, V\rangle(1^\lambda, x) = 1] = 1$$
2. Soundness: if $x \notin L$ then for all P^*, λ we have
$$\Pr[\langle P^*, V\rangle(1^\lambda, x) = 1] \leq 2^{-\lambda}$$

IP is the class of languages L that have a proof system

Proof systems and the class IP

A proof system for a language L is a pair of algorithms (P, V), where V runs in probabilistic, polynomial time (PPT), such that

1. Completeness: if $x \in L$ then for all λ we have

$$\Pr[\langle P, V\rangle(1^\lambda, x) = 1] = 1$$

2. Soundness: if $x \notin L$ then for all P^*, λ we have

$$\Pr[\langle P^*, V\rangle(1^\lambda, x) = 1] \leq 2^{-\lambda}$$

IP is the class of languages L that have a proof system

- Clearly $NP \subseteq IP$

Proof systems and the class IP

A proof system for a language L is a pair of algorithms (P, V), where V runs in probabilistic, polynomial time (PPT), such that

1. Completeness: if $x \in L$ then for all λ we have
$$\Pr[\langle P, V\rangle(1^\lambda, x) = 1] = 1$$
2. Soundness: if $x \notin L$ then for all P^*, λ we have
$$\Pr[\langle P^*, V\rangle(1^\lambda, x) = 1] \leq 2^{-\lambda}$$

IP is the class of languages L that have a proof system

- Clearly $NP \subseteq IP$

Notes:

- (P, V) is an argument system if soundness only holds for PPT P^*

Proof systems and the class IP

A proof system for a language L is a pair of algorithms (P, V), where V runs in probabilistic, polynomial time (PPT), such that

1. Completeness: if $x \in L$ then for all λ we have

$$\Pr[\langle P, V\rangle(1^\lambda, x) = 1] = 1$$

2. Soundness: if $x \notin L$ then for all P^*, λ we have

$$\Pr[\langle P^*, V\rangle(1^\lambda, x) = 1] \leq 2^{-\lambda}$$

IP is the class of languages L that have a proof system

- Clearly $NP \subseteq IP$

Notes:

- (P, V) is an argument system if soundness only holds for PPT P^*
- If $L \in NP$ and $x \in L$, would like P to be efficient (given a witness)

Advantages?

(Potential) advantages?

Advantages?

(Potential) advantages?

- Applicable to languages beyond *NP*

Advantages?

(Potential) advantages?

- Applicable to languages beyond *NP*
- Stronger properties (e.g., zero knowledge)
- More-efficient verification

Advantages?

(Potential) advantages?

- Applicable to languages beyond *NP*
- Stronger properties (e.g., zero knowledge)
- More-efficient verification

Zero-knowledge (ZK) proofs

Goal:

- Convince a verifier that some statement is true (i.e., $x \in L$)...

Zero-knowledge (ZK) proofs

Goal:

- Convince a verifier that some statement is true (i.e., $x \in L$)...
- ...without revealing any information beyond that!

ZK proofs

How to define...?

ZK proofs

How to define...?

Main idea:

The verifier can simulate (by itself) its interaction with the prover!

$\Rightarrow$ Anything the verifier learns from its interaction with the prover, it could have learned on its own

Computational indistinguishability

[redacted]

Computational indistinguishability

Let $\mathcal{X}, \mathcal{X}'$ be such that $\mathcal{X}(1^\lambda, x)$, $\mathcal{X}'(1^\lambda, x)$ are probability distributions for any $\lambda \in \mathbb{N}$ and $x \in S$

Computational indistinguishability

Let $\mathcal{X}, \mathcal{X}'$ be such that $\mathcal{X}(1^\lambda, x)$, $\mathcal{X}'(1^\lambda, x)$ are probability distributions for any $\lambda \in \mathbb{N}$ and $x \in S$

$\mathcal{X}, \mathcal{X}'$ are computationally indistinguishable if for all D running in poly(λ) time and all λ, $x \in S$, and $z \in \{0,1\}^*$

$$\left| \Pr\left[D(1^\lambda, x, \mathcal{X}(1^\lambda, x), z) = 1 \right] - \Pr\left[D(1^\lambda, x, \mathcal{X}'(1^\lambda, x), z) = 1 \right] \right| \leq \mathsf{negl}(\lambda)$$

Computational indistinguishability

Let $\mathcal{X}, \mathcal{X}'$ be such that $\mathcal{X}(1^\lambda, x)$, $\mathcal{X}'(1^\lambda, x)$ are probability distributions for any $\lambda \in \mathbb{N}$ and $x \in S$

$\mathcal{X}, \mathcal{X}'$ are computationally indistinguishable if for all D running in poly(λ) time and all λ, $x \in S$, and $z \in \{0,1\}^*$

$$\left| \Pr\left[D(1^\lambda, x, \mathcal{X}(1^\lambda, x), z) = 1 \right] - \Pr\left[D(1^\lambda, x, \mathcal{X}'(1^\lambda, x), z) = 1 \right] \right| \leq \mathsf{negl}(\lambda)$$

Write $\{\mathcal{X}(1^\lambda, x)\}_{x \in S} \approx \{\mathcal{X}'(1^\lambda, x)\}_{x \in S}$

ZK proofs

Let (P, V) be a proof/argument system for a language L with relation R

ZK proofs

Let (P, V) be a proof/argument system for a language L with relation R

Honest-verifier zero knowledge

(P, V) is (computational) honest-verifier zero knowledge if there is a PPT simulator $\mathcal{S}$ such that

$$\{\langle P(w), V\rangle(1^\lambda, x)\}_{(x,w)\in R} \approx \{\mathcal{S}(1^\lambda, x)\}_{(x,w)\in R}$$

i.e., $\mathcal{S}$ can simulate the (transcript of the) interaction of the prover with the honest verifier, without the witness

ZK proofs

Let (P, V) be a proof/argument system for a language L with relation R

Zero knowledge

(P, V) is (computational) zero knowledge if for every PPT V^* there is an expected polynomial-time simulator $\mathcal{S}_{V^*}$ such that

$$\{\langle P(w), V^*\rangle(1^\lambda, x)\}_{(x,w)\in R} \approx \{\mathcal{S}_{V^*}(1^\lambda, x)\}_{(x,w)\in R}$$

i.e., the (transcript of the) interaction of the prover with any verifier can be simulated

Knowledge soundness/proofs of knowledge (PoKs)

It is often useful to also have a stronger notion of soundness

Knowledge soundness/proofs of knowledge (PoKs)

It is often useful to also have a stronger notion of soundness

Intuition:

If a (malicious) prover can successfully convince the honest verifier, then the prover must know a witness

Knowledge soundness/proofs of knowledge (PoKs)

It is often useful to also have a stronger notion of soundness

Intuition:
If a (malicious) prover can successfully convince the honest verifier, then the prover must know a witness

Why is this useful?

- "Trivial" languages, e.g., $L = \{y \mid \exists x : y = g^x\}$
- When proofs are used as a building block for larger protocols

Proofs of knowledge

Let (P, V) be a proof/argument system for a language $L \in NP$ with associated relation R

Proofs of knowledge

Let (P, V) be a proof/argument system for a language $L \in NP$ with associated relation R

Proof of knowledge

(P, V) is a proof of knowledge (PoK) with respect to R if for every PPT P^* there is an expected polynomial-time knowledge extractor $\mathcal{E}$ such that

- $\Pr[(v, w) \leftarrow \mathcal{E}(1^\lambda, x) : v \text{ is accepting } \wedge \ (x, w) \notin R] \leq \mathsf{negl}(\lambda)$
- $\{\langle P^*, V\rangle(1^\lambda, x)\}_{x \in \{0,1\}^*} \approx \{\mathcal{E}_1(1^\lambda, x)\}_{x \in \{0,1\}^*}$

Proofs of knowledge

Let (P, V) be a proof/argument system for a language $L \in NP$ with associated relation R

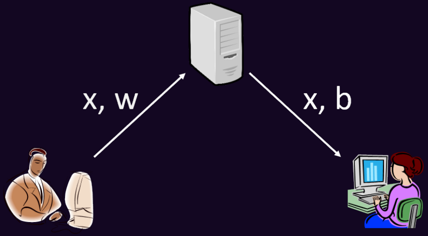
Proof of knowledge

(P, V) is a proof of knowledge (PoK) with respect to R if for every PPT P^* there is an expected polynomial-time knowledge extractor $\mathcal{E}$ such that

- $\Pr[(v, w) \leftarrow \mathcal{E}(1^\lambda, x) : v \text{ is accepting } \wedge \ (x, w) \notin R] \leq \mathsf{negl}(\lambda)$
- $\{\langle P^*, V\rangle(1^\lambda, x)\}_{x \in \{0,1\}^*} \approx \{\mathcal{E}_1(1^\lambda, x)\}_{x \in \{0,1\}^*}$

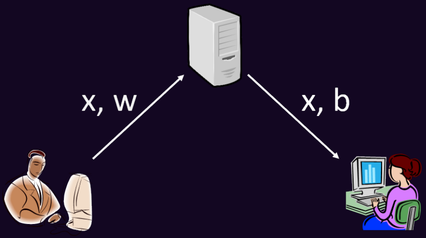
Proof of knowledge $\Rightarrow$ soundness

An aside



If $(x, w) \in R$, set $b := 1$
else set $b := 0$

An aside



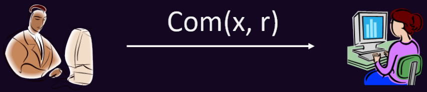
If $(x, w) \in R$, set $b := 1$
else set $b := 0$

Secure computation of this function $\Longleftrightarrow$ a ZKPoK for relation R

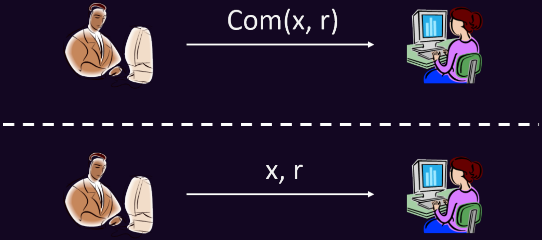
ZKPoKs for NP

Commitment schemes

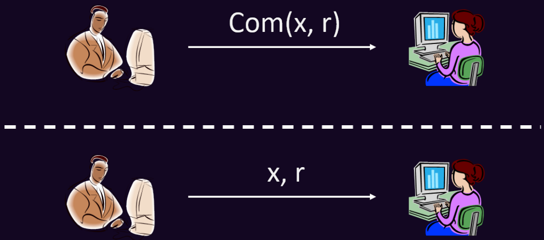
Commitment schemes



Commitment schemes



Commitment schemes



Properties:

- Binding: Sender cannot send a commitment that it can later open to two different values x, x'
- Hiding: Receiver cannot learn anything about x from the commitment

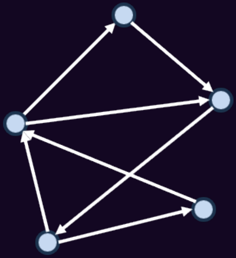
Either property can be computational or perfect/statistical

ZKPoK from commitments

We show a ZKPoK for the NP-complete Hamiltonian cycle problem; this implies a ZKPoK for any language in NP

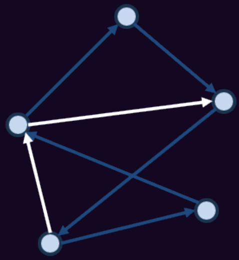
ZKPoK from commitments

We show a ZKPoK for the NP-complete Hamiltonian cycle problem; this implies a ZKPoK for any language in NP



ZKPoK from commitments

We show a ZKPoK for the NP-complete Hamiltonian cycle problem; this implies a ZKPoK for any language in NP



ZKPoK from commitments

Inputs: the prover and verifier share a directed graph G; the prover also knows a Hamiltonian cycle c in G

Three-round subroutine

1. Prover chooses uniform permutation π, and commits entrywise to the adjacency matrix of $\pi(G)$

ZKPoK from commitments

Inputs: the prover and verifier share a directed graph G; the prover also knows a Hamiltonian cycle c in G

Three-round subroutine

1. Prover chooses uniform permutation π, and commits entrywise to the adjacency matrix of $\pi(G)$
2. Verifier sends a uniform challenge $b \in \{0, 1\}$

ZKPoK from commitments

Inputs: the prover and verifier share a directed graph G; the prover also knows a Hamiltonian cycle c in G

Three-round subroutine

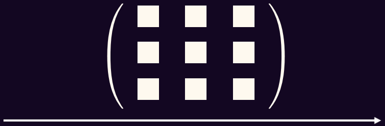
1. Prover chooses uniform permutation π, and commits entrywise to the adjacency matrix of $\pi(G)$
2. Verifier sends a uniform challenge $b \in \{0, 1\}$
3. Prover does:
   - If $b = 0$, open all commitments and send π
   - If $b = 1$, open $\pi(c)$ only
4. Verifier checks:
   - If $b = 0$, check that committed graph corresponds to $\pi(G)$
   - If $b = 1$, check that opened entries are a cycle

ZKPoK from commitments

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

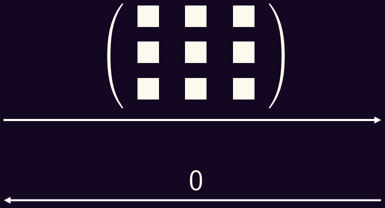
ZKPoK from commitments

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



ZKPoK from commitments

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



ZKPoK from commitments

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

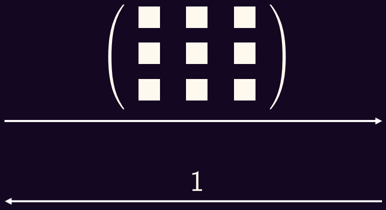
$$\xrightarrow{\begin{pmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{pmatrix}}$$

$$\xleftarrow{\quad 0 \quad}$$

$$\xrightarrow{\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix},\ \pi = (2, 3)}$$

ZKPoK from commitments

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



ZKPoK from commitments

$$G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare \end{pmatrix}$$
$$\longrightarrow$$

$$1$$
$$\longleftarrow$$

$$\begin{pmatrix} \blacksquare & 1 & \blacksquare \\ \blacksquare & \blacksquare & 1 \\ 1 & \blacksquare & \blacksquare \end{pmatrix}$$
$$\longrightarrow$$

ZKPoK from commitments

To obtain soundness error $2^{-\lambda}$, sequentially repeat this 3-round subroutine λ times

ZKPoK from commitments

To obtain soundness error $2^{-\lambda}$, sequentially repeat this 3-round subroutine λ times

Theorem

If the commitment scheme is statistically binding and computationally hiding, this is a ZKPoK for graph Hamiltonicity

PoK analysis

PoK analysis

Key property of 3-round subroutine:

- Given an initial message and correct responses to both challenges, possible to efficiently compute a cycle in G

PoK analysis

Note: assume P^* is deterministic (if not, fix its randomness)

Extractor $\mathcal{E}$

1. Run $P^*(G)$ using uniform $(b_1, \ldots, b_\lambda)$ to obtain transcript v
   - If v is not accepting, output $(v, \perp)$; otherwise, continue
2. For $i = 1, \ldots, \lambda$:
   - Run $P^*(G)$ using $(b_1, \ldots, b_{i-1}, \bar{b}_i)$
   - If P^* responds correctly to $\bar{b}_i$, compute cycle c in G; output (v, c)
3. Output (v, fail)

PoK Analysis

v is identically distributed to an interaction of P^* with V

PoK Analysis

v is identically distributed to an interaction of P^* with V

Need to show $\Pr[v \text{ is accepting} \wedge (x, w) \notin R] \leq 2^{-\lambda}$:

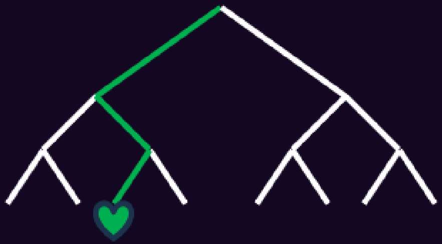
PoK Analysis

v is identically distributed to an interaction of P^* with V

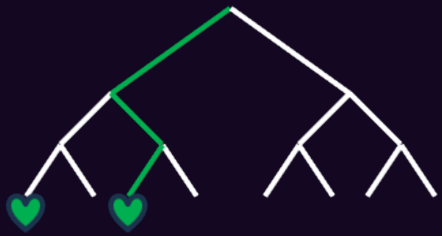
Need to show $\Pr[v \text{ is accepting} \wedge (x, w) \notin R] \leq 2^{-\lambda}$:

- If P^* responds correctly to no sequence of challenges, trivial
- If P^* responds correctly to exactly one sequence of challenges, then $\Pr[v \text{ is accepting}] \leq 2^{-\lambda}$
- If P^* responds correctly to two or more sequences of challenges, then $\mathcal{E}$ will compute a correct witness when v is accepting

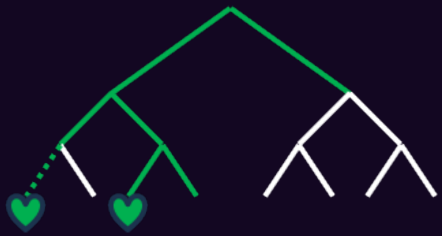
PoK Analysis



PoK Analysis



PoK Analysis



ZK analysis

(Assume perfectly hiding commitments for simplicity)

Key property of 3-round subroutine:

- Easy to simulate if we know the verifier's challenge in advance
  - If the challenge will be 0, commit to a random permutation of G
  - If the challenge will be 1, commit to a random cycle

ZK analysis

Simulator $\mathcal{S}$

For $i = 1, \ldots, \lambda$ do:

- Repeat up to λ times:
  - Choose uniform b_i
    - If $b_i = 0$, choose uniform π and send commitments to $\pi(G)$ to V^*
    - If $b_i = 1$, send commitments to a random cycle to V^*
  - If V^* responds with b_i, answer correctly and continue to next i

ZK analysis

If inner loop never fails, simulation is perfect

ZK analysis

If inner loop never fails, simulation is perfect

(Assuming perfectly hiding commitments)

$$\begin{aligned} \Pr[\text{inner loop fails in any given iteration}] &= 2^{-\lambda} \\ \Rightarrow \Pr[\text{inner loop fails in some iteration}] &\leq \lambda \cdot 2^{-\lambda} \end{aligned}$$

ZKPoKs

The ZKPoK we presented has $\Theta(\lambda)$ rounds

Constant-round ZKPoKs for NP are possible

- Running the 3-round subroutine in parallel does not (seem to) work... why?

ZKPoKs

The ZKPoK we presented has $\Theta(\lambda)$ rounds

Constant-round ZKPoKs for NP are possible

- Running the 3-round subroutine in parallel does not (seem to) work. . . why?

Possible to show (assuming commitment schemes) that every language in *IP* has a zero-knowledge proof. . .

Thank you!