

Lecture 21: Zero Knowledge Proofs

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

November 27, 2024

Overview

- Why Zero Knowledge?
- Zero-Knowledge Proofs
- Conclusion
- Acknowledgements

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there
- Situation
 - Alice has all her files encrypted (in public database)
 - Bob requests from her the contents of one of her files

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there
- Situation
 - Alice has all her files encrypted (in public database)
 - Bob requests from her the contents of one of her files
 - She could simply send the decrypted file to Bob

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there
- Situation
 - Alice has all her files encrypted (in public database)
 - Bob requests from her the contents of one of her files
 - She could simply send the decrypted file to Bob
 - Bob has no way of *knowing* that this message comes from Alice (or that this is indeed the right file)

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there
- Situation
 - Alice has all her files encrypted (in public database)
 - Bob requests from her the contents of one of her files
 - She could simply send the decrypted file to Bob
 - Bob has no way of *knowing* that this message comes from Alice (or that this is indeed the right file)
 - Alice could *prove* to Bob this is the correct file by sending her encryption key

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there
- Situation
 - Alice has all her files encrypted (in public database)
 - Bob requests from her the contents of one of her files
 - She could simply send the decrypted file to Bob
 - Bob has no way of *knowing* that this message comes from Alice (or that this is indeed the right file)
 - Alice could *prove* to Bob this is the correct file by sending her encryption key
 - But then Bob has access to her entire database!

Cryptography

- In cryptography, want to communicate with other people/entities whom we may not trust.
- Or we may not trust the channel of communication
 - someone may *eavesdrop* our messages
 - messages could be *corrupted*
 - someone may try to *impersonate* us
 - it's a wild world out there
- Situation
 - Alice has all her files encrypted (in public database)
 - Bob requests from her the contents of one of her files
 - She could simply send the decrypted file to Bob
 - Bob has no way of *knowing* that this message comes from Alice (or that this is indeed the right file)
 - Alice could *prove* to Bob this is the correct file by sending her encryption key
 - But then Bob has access to her entire database!
 - Can Alice convince Bob that she gave right file *without giving any more knowledge* beyond that she gave right file?

Zero-Knowledge Proofs

Proofs in which the verifier
gains *no knowledge*
beyond the *validity of the assertion*.

Knowledge vs Information

- ① What do you mean by knowledge?
- ② What does it mean to “learn something/gain knowledge”?
- ③ What is difference between knowledge and information?

Knowledge vs Information

- ① What do you mean by knowledge?
- ② What does it mean to “learn something/gain knowledge”?
- ③ What is difference between knowledge and information?

First question is quite complex.

Let's only talk about the second and third.

Knowledge vs Information

- 1 What do you mean by knowledge?
 - 2 What does it mean to “learn something/gain knowledge”?
 - 3 What is difference between knowledge and information?
- *Knowledge*:
 - related to *computational difficulty*
If you could have found the answer (i.e. computed it) without help,
then you *gained no knowledge*
 - about *publicly known* objects
One gains knowledge when one obtains something one *could not compute* before!

Knowledge vs Information

- 1 What do you mean by knowledge?
- 2 What does it mean to “learn something/gain knowledge”?
- 3 What is difference between knowledge and information?
 - **Knowledge:**
 - related to *computational difficulty*
If you could have found the answer (i.e. computed it) without help, then you *gained no knowledge*
 - about *publicly known* objects
One gains knowledge when one obtains something one *could not compute* before!
 - **Information:**
 - unrelated to computational difficulty
 - about *partially known* objects
One gains information when one obtains something one *could not access* before!

Example: Knowledge vs Information

- Bob asks Alice whether a graph G is Eulerian

Example: Knowledge vs Information

- Bob asks Alice whether a graph G is Eulerian
- Bob gains no knowledge in this interaction, since he could have computed it by himself

Euler's theorem: check that all vertices have even degree

Example: Knowledge vs Information

- Bob asks Alice whether a graph G is Eulerian
- Bob gains no knowledge in this interaction, since he could have computed it by himself

Euler's theorem: check that all vertices have even degree

- Bob asks Alice if graph G has Hamiltonian cycle

Example: Knowledge vs Information

- Bob asks Alice whether a graph G is Eulerian
- Bob gains no knowledge in this interaction, since he could have computed it by himself

Euler's theorem: check that all vertices have even degree

- Bob asks Alice if graph G has Hamiltonian cycle
- Bob now gains knowledge ($P \neq NP \Rightarrow$ Bob could not compute it)

Example: Knowledge vs Information

- Bob asks Alice whether a graph G is Eulerian
- Bob gains no knowledge in this interaction, since he could have computed it by himself

Euler's theorem: check that all vertices have even degree

- Bob asks Alice if graph G has Hamiltonian cycle
- Bob now gains knowledge ($P \neq NP \Rightarrow$ Bob could not compute it)
- In both cases Alice *didn't* convey any *information*!

The graph is given, so all information about it is “available” to everyone.

Example: Graph Isomorphism

- Setup:
 - A claim $\mathcal{C} :=$ graphs G_0, G_1 are isomorphic

Example: Graph Isomorphism

- Setup:
 - A claim $\mathcal{C} :=$ graphs G_0, G_1 are isomorphic
 - A prover P writes down an isomorphism ρ such that $\rho(G_0) = G_1$

Example: Graph Isomorphism

- Setup:
 - A claim $\mathcal{C} :=$ graphs G_0, G_1 are isomorphic
 - A prover P writes down an isomorphism ρ such that $\rho(G_0) = G_1$
 - Prover P sends ρ to a verifier V

Example: Graph Isomorphism

- Setup:
 - A claim $\mathcal{C} :=$ graphs G_0, G_1 are isomorphic
 - A prover P writes down an isomorphism ρ such that $\rho(G_0) = G_1$
 - Prover P sends ρ to a verifier V
 - Verifier checks that ρ is a permutation of vertices, and that $\rho(G_0) = G_1$ (*deterministic, polynomial time algorithm*)

Example: Graph Isomorphism

- Setup:
 - A claim $\mathcal{C} :=$ graphs G_0, G_1 are isomorphic
 - A prover P writes down an isomorphism ρ such that $\rho(G_0) = G_1$
 - Prover P sends ρ to a verifier V
 - Verifier checks that ρ is a permutation of vertices, and that $\rho(G_0) = G_1$ (*deterministic, polynomial time algorithm*)
 - Verifier accepts iff the above is correct.

Example: Graph Isomorphism

- Setup:
 - A claim $\mathcal{C} :=$ graphs G_0, G_1 are isomorphic
 - A prover P writes down an isomorphism ρ such that $\rho(G_0) = G_1$
 - Prover P sends ρ to a verifier V
 - Verifier checks that ρ is a permutation of vertices, and that $\rho(G_0) = G_1$ (*deterministic, polynomial time algorithm*)
 - Verifier accepts iff the above is correct.
- In this setting, verifier *learns the isomorphism* (i.e., the proof)!

Can we convince people differently?

- Yes! But we need to modify the way proofs are checked.

Can we convince people differently?

- Yes! But we need to modify the way proofs are checked.
 - Make proofs *interactive*, instead of only one-way

Can we convince people differently?

- Yes! But we need to modify the way proofs are checked.
 - Make proofs *interactive*, instead of only one-way
 - Verifier is allowed *private randomness*

Can we convince people differently?

- Yes! But we need to modify the way proofs are checked.
 - Make proofs *interactive*, instead of only one-way
 - Verifier is allowed *private randomness*
- In the end, we will see a (zero-knowledge) proof for graph isomorphism as follows:

Alice: I will not give you an isomorphism, but I will prove that I could give you one, if I wanted to.

- Why Zero Knowledge?
- Zero-Knowledge Proofs
- Conclusion
- Acknowledgements

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !
- Note that:
 - Claim is *true* \Rightarrow prover can *always* give isomorphism!
 - Claim is *false* \Rightarrow can catch *bad proof* with probability = $1/2$

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !
- Note that:
 - Claim is *true* \Rightarrow prover can *always* give isomorphism!
 - Claim is *false* \Rightarrow can catch *bad proof* with probability = $1/2$
 - Can amplify probability of catching bad proof by repeating protocol above!

Example: Graph Isomorphism

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !
- Note that:
 - Claim is *true* \Rightarrow prover can *always* give isomorphism!
 - Claim is *false* \Rightarrow can catch *bad proof* with probability = $1/2$
 - Can amplify probability of catching bad proof by repeating protocol above!
- How can we model the fact that verifier does not gain knowledge?!

Simulation!

(Attempt at) Perfect Zero Knowledge

Note that we usually talked about not trusting provers so far, but for Zero-Knowledge, we will *not trust verifiers* (as they may try to obtain information about the proof!)¹

¹In particular, the zero knowledge is a property of the *prover*. 


(Attempt at) Perfect Zero Knowledge

Note that we usually talked about not trusting provers so far, but for Zero-Knowledge, we will *not trust verifiers* (as they may try to obtain information about the proof!)¹

Definition ((Ideal/Naive) Perfect Zero Knowledge)

A proof system (P, V) is *perfect zero-knowledge* for language L if for every polynomial time, randomized verifier V^* , there is a randomized algorithm M^* such that for every $x \in L$ the following random variables are *identically* distributed:

- $\langle P, V^* \rangle(x)$, that is, output of interaction between prover P and verifier V^* on input x
- $M^*(x)$, that is, output of algorithm M^* (simulation) on input x

¹In particular, the zero knowledge is a property of the *prover*. 


(Attempt at) Perfect Zero Knowledge

Note that we usually talked about not trusting provers so far, but for Zero-Knowledge, we will *not trust verifiers* (as they may try to obtain information about the proof!)¹

Definition ((Ideal/Naive) Perfect Zero Knowledge)

A proof system (P, V) is *perfect zero-knowledge* for language L if for every polynomial time, randomized verifier V^* , there is a randomized algorithm M^* such that for every $x \in L$ the following random variables are *identically* distributed:

- $\langle P, V^* \rangle(x)$, that is, output of interaction between prover P and verifier V^* on input x
- $M^*(x)$, that is, output of algorithm M^* (simulation) on input x
- The above captures the idea that V^* is not gaining any extra computational power by interacting with P , since same output could have been generated by M^*

¹In particular, the zero knowledge is a property of the *prover*. 

Perfect Zero Knowledge

- Previous definition is a bit too strict to be useful, so we relax it.²
- We will allow simulator to fail with small probability (denoted by outputting \perp)

²Very common phenomenon in crypto, that statistical indistinguishability too strict

Perfect Zero Knowledge

- Previous definition is a bit too strict to be useful, so we relax it.²
- We will allow simulator to fail with small probability (denoted by outputting \perp)

Definition (Perfect Zero Knowledge)

A proof system (P, V) is *perfect zero-knowledge* for language L if for every polynomial time, randomized verifier V^* , there is a probabilistic, poly-time TM M^* such that the following holds:

- 1 For each $x \in \{0, 1\}^*$,

$$\Pr[M^*(x) = \perp] \leq 1/2$$

- 2 The following variables are *identically distributed* for $x \in L$:
 - $\langle P, V^* \rangle(x)$, that is, output of interaction between prover P and verifier V^* on input x
 - $B_x := (M^*(x) \mid M^*(x) \neq \perp)$, that is, output of simulator M^* on input x conditioned on not outputting \perp

²Very common phenomenon in crypto, that statistical indistinguishability is too strict.

Statistical Zero Knowledge

- Given two random variables A, B over the same discrete set \mathcal{D} , their statistical distance (or Total Variation distance – TV) is given by

$$\Delta_{TV}(A, B) := \frac{1}{2} \cdot \sum_{\alpha \in \mathcal{D}} |\Pr[A = \alpha] - \Pr[B = \alpha]|$$

Statistical Zero Knowledge

- Given two random variables A, B over the same discrete set \mathcal{D} , their statistical distance (or Total Variation distance – TV) is given by

$$\Delta_{TV}(A, B) := \frac{1}{2} \cdot \sum_{\alpha \in \mathcal{D}} |\Pr[A = \alpha] - \Pr[B = \alpha]|$$

- Given a language $L \subseteq \{0, 1\}^*$, consider the ensembles of random variables $\{A_x\}_{x \in L}, \{B_x\}_{x \in L}$ – that is, for each $x \in L$, we have two random variables A_x, B_x

Statistical Zero Knowledge

- Given two random variables A, B over the same discrete set \mathcal{D} , their statistical distance (or Total Variation distance – TV) is given by

$$\Delta_{TV}(A, B) := \frac{1}{2} \cdot \sum_{\alpha \in \mathcal{D}} |\Pr[A = \alpha] - \Pr[B = \alpha]|$$

- Given a language $L \subseteq \{0, 1\}^*$, consider the ensembles of random variables $\{A_x\}_{x \in L}, \{B_x\}_{x \in L}$ – that is, for each $x \in L$, we have two random variables A_x, B_x
- Two ensembles of random variables $\{A_x\}_{x \in L}, \{B_x\}_{x \in L}$ are *statistically indistinguishable* if for every polynomial function $p : \mathbb{N} \rightarrow \mathbb{N}$ and all sufficiently large $x \in L$

$$\Delta_{TV}(A_x, B_x) < \frac{1}{p(|x|)}.$$

Statistical Zero Knowledge

Definition (Statistical Zero Knowledge)

A proof system (P, V) is *statistical zero-knowledge* for language L if for every PPT verifier V^* , there is a PPT TM M^* such that:

- 1 For each $x \in \{0, 1\}^*$,

$$\Pr[M^*(x) = \perp] \leq 1/2$$

- 2 The following ensembles are *statistically indistinguishable* for $x \in L$:
 - $A_x := \langle P, V^* \rangle(x)$, (output of interaction)
 - $B_x := (M^*(x) \mid M^*(x) \neq \perp)$, that is, output of simulator M^* on input x conditioned on not outputting \perp

Computational Zero Knowledge

- Given a language $L \subseteq \{0, 1\}^*$, consider the ensembles of random variables $\{A_x\}_{x \in L}, \{B_x\}_{x \in L}$ – that is, for each $x \in L$, we have two random variables A_x, B_x

Computational Zero Knowledge

- Given a language $L \subseteq \{0, 1\}^*$, consider the ensembles of random variables $\{A_x\}_{x \in L}, \{B_x\}_{x \in L}$ – that is, for each $x \in L$, we have two random variables A_x, B_x
- The ensembles are *computationally indistinguishable* if for every PPT TM D , for every polynomial function $p : \mathbb{N} \rightarrow \mathbb{N}$, and for all sufficiently large $x \in L$

$$|\Pr[D(x, A_x) = 1] - \Pr[D(x, B_x) = 1]| < \frac{1}{p(|x|)}$$

Computational Zero Knowledge

- Given a language $L \subseteq \{0, 1\}^*$, consider the ensembles of random variables $\{A_x\}_{x \in L}, \{B_x\}_{x \in L}$ – that is, for each $x \in L$, we have two random variables A_x, B_x
- The ensembles are *computationally indistinguishable* if for every PPT TM D , for every polynomial function $p : \mathbb{N} \rightarrow \mathbb{N}$, and for all sufficiently large $x \in L$

$$|\Pr[D(x, A_x) = 1] - \Pr[D(x, B_x) = 1]| < \frac{1}{p(|x|)}$$

- That is, no poly-time randomized algorithm can distinguish between above ensembles

Computational Zero Knowledge

Definition (Computational Zero Knowledge)

A proof system (P, V) is *computationally zero-knowledge* for language L if for every PPT verifier V^* , there is a PPT simulator M^* such that:

- 1 For each $x \in \{0, 1\}^*$,

$$\Pr[M^*(x) = \perp] \leq 1/2$$

- 2 The following ensembles are *computationally indistinguishable* for $x \in L$:

- $\langle P, V^* \rangle(x)$, (output of interaction)
- $(M^*(x) \mid M^*(x) \neq \perp)$, that is, output of simulator M^* on input x conditioned on not outputting \perp

Complexity Classes & Relations

- Let PZK be the class of languages that have perfect zero-knowledge proof.
- Similarly, define SZK and CZK for the statistical and computational zero-knowledge

Complexity Classes & Relations

- Let PZK be the class of languages that have perfect zero-knowledge proof.
- Similarly, define SZK and CZK for the statistical and computational zero-knowledge
- From the definitions, we have:

$$\text{BPP} \subseteq \text{PZK} \subseteq \text{SZK} \subseteq \text{CZK} \subseteq \text{IP}$$

Complexity Classes & Relations

- Let PZK be the class of languages that have perfect zero-knowledge proof.
- Similarly, define SZK and CZK for the statistical and computational zero-knowledge
- From the definitions, we have:

$$\text{BPP} \subseteq \text{PZK} \subseteq \text{SZK} \subseteq \text{CZK} \subseteq \text{IP}$$

- Fun fact:

$$\text{IP} = \text{PSPACE}$$

Complexity Classes & Relations

- Let PZK be the class of languages that have perfect zero-knowledge proof.
- Similarly, define SZK and CZK for the statistical and computational zero-knowledge
- From the definitions, we have:

$$\text{BPP} \subseteq \text{PZK} \subseteq \text{SZK} \subseteq \text{CZK} \subseteq \text{IP}$$

- Fun fact:

$$\text{IP} = \text{PSPACE}$$

- Fun (conditional under some strong conditions) semi-fact:

existence of non-uniformly hard OWFs \Rightarrow CZK = IP

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !
- Note that:
 - Claim is *true* \Rightarrow prover can *always* give isomorphism!
 - Claim is *false* \Rightarrow can catch *bad proof* with probability = $1/2$

PZK for Graph Isomorphism - Proof

Setup:

- A claim $\mathcal{C} :=$ graphs $G_0([n], E_0), G_1([n], E_1)$ are isomorphic

Protocol:

- A prover P picks $\pi \sim S_n$ and sends $H := \pi(G_1)$ to verifier
- Verifier picks $b \sim \{0, 1\}$ and sends b to prover
- Upon receiving b , prover does
 - If $b = 0$, then prover sends $\rho := \pi \circ \sigma$ (if there is σ s.t. $\sigma(G_0) = G_1$)
 - If $b = 1$, then prover sends $\rho := \pi$
- Verifier checks that $\rho(G_b) = H$
- Note that verifier will not learn isomorphism between G_0 and G_1 !
- Note that:
 - Claim is *true* \Rightarrow prover can *always* give isomorphism!
 - Claim is *false* \Rightarrow can catch *bad proof* with probability = $1/2$
 - Can amplify probability of catching bad proof by repeating protocol above!

PZK for Graph Isomorphism - Simulator

Protocol:

- Let $x := (G_0, G_1)$ be the input

PZK for Graph Isomorphism - Simulator

Protocol:

- Let $x := (G_0, G_1)$ be the input
- The simulator M^* selects a random string $R \sim \{0, 1\}^{q(|x|)}$

PZK for Graph Isomorphism - Simulator

Protocol:

- Let $x := (G_0, G_1)$ be the input
- The simulator M^* selects a random string $R \sim \{0, 1\}^{q(|x|)}$
- M^* then picks $b \sim \{0, 1\}$ and $\pi \sim S_n$ and sends $\pi(G_b)$ to "verifier"

PZK for Graph Isomorphism - Simulator

Protocol:

- Let $x := (G_0, G_1)$ be the input
- The simulator M^* selects a random string $R \sim \{0, 1\}^{q(|x|)}$
- M^* then picks $b \sim \{0, 1\}$ and $\pi \sim S_n$ and sends $\pi(G_b)$ to "verifier"
- Now, M^* simulates $V^*(x, R, \pi(G_b))$, and "sends" bit \tilde{b} (the outcome of V^*) to "prover"

PZK for Graph Isomorphism - Simulator

Protocol:

- Let $x := (G_0, G_1)$ be the input
- The simulator M^* selects a random string $R \sim \{0, 1\}^{q(|x|)}$
- M^* then picks $b \sim \{0, 1\}$ and $\pi \sim S_n$ and sends $\pi(G_b)$ to "verifier"
- Now, M^* simulates $V^*(x, R, \pi(G_b))$, and "sends" bit \tilde{b} (the outcome of V^*) to "prover"
- If $b = \tilde{b}$, then M^* halts with output $(x, R, \pi(G_b), \pi)$. Else, output \perp .

PZK for Graph Isomorphism - Simulator

Protocol:

- Let $x := (G_0, G_1)$ be the input
- The simulator M^* selects a random string $R \sim \{0, 1\}^{q(|x|)}$
- M^* then picks $b \sim \{0, 1\}$ and $\pi \sim S_n$ and sends $\pi(G_b)$ to "verifier"
- Now, M^* simulates $V^*(x, R, \pi(G_b))$, and "sends" bit \tilde{b} (the outcome of V^*) to "prover"
- If $b = \tilde{b}$, then M^* halts with output $(x, R, \pi(G_b), \pi)$. Else, output \perp .
- Need to prove: whenever we don't fail, we output *same distribution* as the original protocol!

Proof of same distribution

- Let $x = (G_0, G_1) \in L$, i.e. G_0 and G_1 are isomorphic³
- Let $A_x := \langle P, V^* \rangle(x)$ and $B_x := (M^*(x) \mid M^*(x) \neq \perp)$.
- A_x, B_x take values on tuples (x, R, H, π)

Note that the bit b that V^* (same for M^*) sends is determined by
 $x, R, H,$
so no need to include it in the description of the random variable.

³Simplifying assumption: G_0, G_1 are *asymmetric*, i.e., $\text{Aut}(G_b) = \{1\}$.

Proof of same distribution

- Let $A_x := \langle P, V^* \rangle(x)$ and $B_x := (M^*(x) \mid M^*(x) \neq \perp)$.
- A_x, B_x take values on tuples (x, R, H, π)
- Proof that A_x and B_x are identically distributed:
 - Enough to show that for each choice of $x \in L$ and $R \in \{0, 1\}^{q(|x|)}$, the random variables

$$\mu := A_x(x, R, -, -) \quad \text{and} \quad \nu := B_x(x, R, -, -)$$

are identically distributed



Conclusion

- We saw today how the power of interaction can be used to verify validity of “proofs” without conveying information about it

Conclusion

- We saw today how the power of interaction can be used to verify validity of “proofs” without conveying information about it
- Has applications in
 - Modern cryptography
 - Credit Cards
 - Passwords
 - Complexity Theory (can use zero-knowledge to construct complexity classes)
 - Used in cryptocurrencies (validate transactions without giving details about transactions)

Acknowledgement

- Lecture based largely on:

- Oded Goldreich's Foundations of Cryptography book, Chapter 4
- Oded Goldreich's Computational Complexity book, Chapter 9.2
- Berkeley & MIT's 6.875 Lecture 14

<https://inst.eecs.berkeley.edu/~cs276/fa20/slides/lec14.pdf>