Zero-Knowledge Proofs

CS 601.641/441

Spring 2018

Abhishek Jain, JHU

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement
- Mathematical proof: Deductive argument for a statement, by reducing the validity of the statement to a set of axioms or assumptions

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement
- Mathematical proof: Deductive argument for a statement, by reducing the validity of the statement to a set of axioms or assumptions
- Desirable features in a proof:

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement
- Mathematical proof: Deductive argument for a statement, by reducing the validity of the statement to a set of axioms or assumptions
- Desirable features in a proof:
  - The verifier should accept the proof if the statement is true

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement
- Mathematical proof: Deductive argument for a statement, by reducing the validity of the statement to a set of axioms or assumptions
- Desirable features in a proof:
  - The verifier should accept the proof if the statement is true
  - The verifier should reject *any* proof if the statement is false

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement
- Mathematical proof: Deductive argument for a statement, by reducing the validity of the statement to a set of axioms or assumptions
- Desirable features in a proof:
    - The verifier should accept the proof if the statement is true
    - The verifier should reject *any* proof if the statement is false
    - Proof must be finite (or succinct) and efficiently verifiable

What is a Proof?

- An argument (or sufficient evidence) that can convince a reader of the truth of some statement
- Mathematical proof: Deductive argument for a statement, by reducing the validity of the statement to a set of axioms or assumptions
- Desirable features in a proof:
  - The verifier should accept the proof if the statement is true
  - The verifier should reject *any* proof if the statement is false
  - Proof must be finite (or succinct) and efficiently verifiable
- E.g., Proof that there are infinitely many primes should not simply be a list of all the primes. Not only would it take forever to generate that proof, it would also take forever to verify it

Interactive Proofs

1. Typically, proofs are *non-interactive*: a prover can write down the proof and send it to a verifier who can then verify it for correctness

Interactive Proofs

1. Typically, proofs are *non-interactive*: a prover can write down the proof and send it to a verifier who can then verify it for correctness
2. Interestingly, interactive proofs (i.e., a back-and-forth conversation between a prover and verifier) can be very powerful

Interactive Proofs

Definition (Interactive Proofs)

A pair of PPT algorithms (P, V) is an interactive proof system for a language L if the following properties hold:

Interactive Proofs

Definition (Interactive Proofs)

A pair of PPT algorithms (P, V) is an interactive proof system for a language L if the following properties hold:

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathsf{Out}_V[P(x) \leftrightarrow V(x)] = 1\right] = 1$$

Interactive Proofs

Definition (Interactive Proofs)

A pair of PPT algorithms (P, V) is an interactive proof system for a language L if the following properties hold:

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathsf{Out}_V[P(x) \leftrightarrow V(x)] = 1\right] = 1$$

- **Soundness:** There exists a negligible function $\nu(\cdot)$ s.t. $\forall x \notin L$ and for all adversarial provers P^*,

$$\Pr\left[\mathsf{Out}_V[P^*(x) \leftrightarrow V(x)] = 1\right] \leqslant \nu(|x|)$$

Interactive Proofs

Definition (Interactive Proofs)

A pair of PPT algorithms (P, V) is an interactive proof system for a language L if the following properties hold:

- **Completeness:** For every $x \in L$,

$$\Pr\left[\mathsf{Out}_V[P(x) \leftrightarrow V(x)] = 1\right] = 1$$

- **Soundness:** There exists a negligible function $\nu(\cdot)$ s.t. $\forall x \notin L$ and for all adversarial provers P^*,

$$\Pr\left[\mathsf{Out}_V[P^*(x) \leftrightarrow V(x)] = 1\right] \leqslant \nu(|x|)$$

Remark: If the soundness property only holds against PPT adversarial provers, then we refer to it as an **argument**

Zero Knowledge: Intuition

- An interactive proof is zero knowledge if the verifier does not gain any knowledge from its interaction with the prover beyond the fact that the statement is true

Zero Knowledge: Intuition

- An interactive proof is zero knowledge if the verifier does not gain any knowledge from its interaction with the prover beyond the fact that the statement is true
- We do not gain any knowledge from an interaction if we could have carried it out on our own

Zero Knowledge: Intuition

- An interactive proof is zero knowledge if the verifier does not gain any knowledge from its interaction with the prover beyond the fact that the statement is true

- We do not gain any knowledge from an interaction if we could have carried it out on our own

- Intuition for ZK: V can generate a protocol transcript on its own, without talking to P. If this transcript is indistinguishable from a real execution, then clearly V does not learn anything by talking to P

Zero Knowledge: Intuition

- An interactive proof is zero knowledge if the verifier does not gain any knowledge from its interaction with the prover beyond the fact that the statement is true
- We do not gain any knowledge from an interaction if we could have carried it out on our own
- Intuition for ZK: V can generate a protocol transcript on its own, without talking to P. If this transcript is indistinguishable from a real execution, then clearly V does not learn anything by talking to P
- Formalized via notion of *Simulator*

Zero Knowledge: Definition I

We first define Honest Verifier Zero Knowledge, i.e., ZK property against verifiers who behave honestly during the protocol but may try to learn extra information from the transcript.

Definition (Honest Verifier Zero Knowledge)

An interactive proof (P, V) for a language L with witness relation R is said to be *honest verifier zero knowledge* if there exists a PPT simulator S s.t. for every non-uniform PPT distinguisher D, there exists a negligible function $\nu(\cdot)$ s.t. for every $x \in L$, $w \in R(x)$, $z \in \{0,1\}^*$, D distinguishes between the following distributions with probability at most $\nu(n)$:

- $\left\{\mathsf{View}_V[P(x,w) \leftrightarrow V(x,z)]\right\}$
- $\left\{S(1^n, x, z)\right\}$

Zero Knowledge: Definition II

We now define ZK against malicious verifiers, who may use arbitrary PPT malicious strategy during the protocol

Definition (Zero Knowledge)

An interactive proof (P, V) for a language L with witness relation R is said to be *zero knowledge* if for every non-uniform PPT adversary V^*, there exists an expected PPT simulator S s.t. for every non-uniform PPT distinguisher D, there exists a negligible function $\nu(\cdot)$ s.t. for every $x \in L$, $w \in R(x)$, $z \in \{0,1\}^*$, D distinguishes between the following distributions with probability at most $\nu(n)$:

- $\left\{\mathsf{View}^*_V[P(x, w) \leftrightarrow V^*(x, z)]\right\}$
- $\left\{S(1^n, x, z)\right\}$

Notation for Graphs

- Graph $G = (V, E)$ where V is set of vertices and E is set of edges
- $|V| = n$, $|E| = m$
- **Graph 3-Coloring**: Language of all graphs whose vertices can be colored using only three colors s.t. no two connected vertices have the same color
- Graph 3-Coloring is NP-Complete (so any NP instance can be efficiently transformed into a graph 3-coloring instance)

ZK Proof for Graph 3-Coloring

Common Input: $G = (V, E)$**, where** $|V| = n$

P**'s witness: Colors** $\mathsf{color}_1, \ldots, \mathsf{color}_n \in \{1, 2, 3\}$

Protocol (P, V)**:** Repeat the following procedure $n|E|$ times *using fresh randomness*

$P \to V$: P chooses a random permutation π over $\{1, 2, 3\}$. For every $i \in [n]$, it computes $C_i = \mathsf{Com}(\widetilde{\mathsf{color}}_i)$ where $\widetilde{\mathsf{color}}_i = \pi(\mathsf{color}_i)$. It sends $(C_1, \ldots, C_n)$ to V

$V \to P$: V chooses a random edge $(i, j) \in E$ and sends it to P

$P \to V$: Prover opens C_i and C_j to reveal $(\widetilde{\mathsf{color}}_i, \widetilde{\mathsf{color}}_j)$

V: If the openings of C_i, C_j are valid and $\widetilde{\mathsf{color}}_i \neq \widetilde{\mathsf{color}}_j$, then V accepts the proof. Otherwise, it rejects.

Proof of Soundness

- If G is not 3-colorable, then for any coloring $\mathsf{color}_1, \ldots, \mathsf{color}_n$, there exists at least one edge which has the same colors on both endpoints

Proof of Soundness

- If G is not 3-colorable, then for any coloring $\mathsf{color}_1, \ldots, \mathsf{color}_n$, there exists at least one edge which has the same colors on both endpoints
- From the binding property of Com, it follows that $C_1, \ldots, C_n$ have unique openings $\widetilde{\mathsf{color}}_1, \ldots, \widetilde{\mathsf{color}}_n$

Proof of Soundness

- If G is not 3-colorable, then for any coloring $\mathsf{color}_1, \ldots, \mathsf{color}_n$, there exists at least one edge which has the same colors on both endpoints
- From the binding property of Com, it follows that $C_1, \ldots, C_n$ have unique openings $\widetilde{\mathsf{color}}_1, \ldots, \widetilde{\mathsf{color}}_n$
- Combining the above, let $(i^*, j^*) \in E$ be s.t. $\widetilde{\mathsf{color}}_{i^*} = \widetilde{\mathsf{color}}_{j^*}$

Proof of Soundness

- If G is not 3-colorable, then for any coloring $\mathsf{color}_1, \ldots, \mathsf{color}_n$, there exists at least one edge which has the same colors on both endpoints
- From the binding property of Com, it follows that $C_1, \ldots, C_n$ have unique openings $\widetilde{\mathsf{color}}_1, \ldots, \widetilde{\mathsf{color}}_n$
- Combining the above, let $(i^*, j^*) \in E$ be s.t. $\widetilde{\mathsf{color}}_{i^*} = \widetilde{\mathsf{color}}_{j^*}$
- Then, with probability $\frac{1}{|E|}$, V chooses $i = i^*, j = j^*$ and catches P

Proof of Soundness

- If G is not 3-colorable, then for any coloring $\mathsf{color}_1, \ldots, \mathsf{color}_n$, there exists at least one edge which has the same colors on both endpoints
- From the binding property of Com, it follows that $C_1, \ldots, C_n$ have unique openings $\widetilde{\mathsf{color}}_1, \ldots, \widetilde{\mathsf{color}}_n$
- Combining the above, let $(i^*, j^*) \in E$ be s.t. $\widetilde{\mathsf{color}}_{i^*} = \widetilde{\mathsf{color}}_{j^*}$
- Then, with probability $\frac{1}{|E|}$, V chooses $i = i^*, j = j^*$ and catches P
- In $n|E|$ independent repetitions, P successfully cheats in all repetitions with probability at most

$$\left(1 - \frac{1}{|E|}\right)^{n|E|} \approx e^{-n}$$

Proving Zero Knowledge

Intuition:

- In each iteration, V only sees two random colors

Proving Zero Knowledge

Intuition:

- In each iteration, V only sees two random colors
- Hiding property of Com guarantees that everything else remains hidden from V

Proving Zero Knowledge

Intuition:

- In each iteration, V only sees two random colors
- Hiding property of Com guarantees that everything else remains hidden from V
- We will prove zero knowledge for one iteration.

Proving Zero Knowledge

Intuition:

- In each iteration, V only sees two random colors
- Hiding property of Com guarantees that everything else remains hidden from V
- We will prove zero knowledge for one iteration.
- ZK for one iteration implies Honest-Verifier ZK for one iteration. Honest-Verifier ZK is preserved under parallel repetition

Proving Zero Knowledge

Intuition:

- In each iteration, V only sees two random colors
- Hiding property of Com guarantees that everything else remains hidden from V
- We will prove zero knowledge for one iteration.
- ZK for one iteration implies Honest-Verifier ZK for one iteration. Honest-Verifier ZK is preserved under parallel repetition
- (Malicious-verifier) ZK does not compose under parallel repetition. But it composes under sequential repetition.

Proving Zero Knowledge: Simulator

Simulator $S(x = G, z)$**:**

- Choose a random edge $(i', j') \xleftarrow{\$} E$ and pick random colors $\mathsf{color}'_{i'}, \mathsf{color}'_{j'} \xleftarrow{\$} \{1, 2, 3\}$ s.t. $\mathsf{color}'_{i'} \neq \mathsf{color}'_{j'}$. For every other $k \in [n] \setminus \{i', j'\}$, set $\mathsf{color}'_k = 1$
- For every $\ell \in [n]$, compute $C_\ell = \mathsf{Com}(\mathsf{color}'_\ell)$
- Emulate execution of $V^*(x, z)$ by feeding it $(C_1, \ldots, C_n)$. Let (i, j) denote its response
- If $(i, j) = (i', j')$, then feed the openings of C_i, C_j to V^* and output its view. Otherwise, restart the above procedure, at most $n|E|$ times
- If simulation has not succeeded after $n|E|$ attempts, then output `fail`

Correctness of Simulation

- Can be argued using the hiding property of commitments
- Must also argue that Sim fails with negligible probability
- Full proof using "Hybrid Arguments" (attend Modern Cryptography for more details)

Non-Interactive Zero Knowledge

- In the Random Oracle model, 3-round Honest Verifier ZK where the verifier's messages are "public coin," can be transformed into a non-interactive ZK proof
- **Main Idea [Fiat-Shamir]:** Let (α, β, γ) denote the messages of an HVZK protocol. Then, for any α computed by the prover, β can be computed as $H(\alpha)$, where H is a Hash function (like SHA-256). So now, prover can compute (α, β, γ) on its own and send it as a *non-interactive* proof
- Soundness and ZK property are argued in the programmable Random Oracle model, where the Hash function is viewed as Random Oracle.

Succinct Arguments

- Typically, the total "size" of an argument depends upon the size of the statement and the witness used to compute the proof

Succinct Arguments

- Typically, the total "size" of an argument depends upon the size of the statement and the witness used to compute the proof
- In some applications, the statement and witnesses may be very large (Later: see such an example in ZeroCash)

Succinct Arguments

- Typically, the total "size" of an argument depends upon the size of the statement and the witness used to compute the proof
- In some applications, the statement and witnesses may be very large (Later: see such an example in ZeroCash)
- **Succinct argument:** The size of an argument is *independent* (or poly-logarithmic) in the size of the statement and the witness

Succinct Arguments

- Typically, the total "size" of an argument depends upon the size of the statement and the witness used to compute the proof
- In some applications, the statement and witnesses may be very large (Later: see such an example in ZeroCash)
- **Succinct argument:** The size of an argument is *independent* (or poly-logarithmic) in the size of the statement and the witness
- Interactive succinct arguments were first constructed by Kilian using Probabilistically-Checkable Proofs (PCPs)

Succinct Arguments

- Typically, the total "size" of an argument depends upon the size of the statement and the witness used to compute the proof
- In some applications, the statement and witnesses may be very large (Later: see such an example in ZeroCash)
- **Succinct argument:** The size of an argument is *independent* (or poly-logarithmic) in the size of the statement and the witness
- Interactive succinct arguments were first constructed by Kilian using Probabilistically-Checkable Proofs (PCPs)
- Micali constructed non-interactive succinct arguments in Random Oracle model by applying Fiat-Shamir heuristic on Kilian's protocol

Zero Knowledge SNARKS

- **Succinct Non-Interactive Argument of Knowledge (SNARK):** A succinct argument is called an *argument of knowledge* if there exists a PPT extractor algorithm $\mathcal{E}$ who can extract a valid witness w for any statement x proven by a prover algorithm P^*, given access to the code of P^*

Zero Knowledge SNARKS

- **Succinct Non-Interactive Argument of Knowledge (SNARK):** A succinct argument is called an *argument of knowledge* if there exists a PPT extractor algorithm $\mathcal{E}$ who can extract a valid witness w for any statement x proven by a prover algorithm P^*, given access to the code of P^*
- SNARKs are presently only known from *knowledge assumptions*, which are considered "non-standard" in cryptography (in fact, in certain restricted settings, some knowledge assumptions and program obfuscation have been shown to be in contention)

Zero Knowledge SNARKS

- **Succinct Non-Interactive Argument of Knowledge (SNARK):** A succinct argument is called an *argument of knowledge* if there exists a PPT extractor algorithm $\mathcal{E}$ who can extract a valid witness w for any statement x proven by a prover algorithm P^*, given access to the code of P^*
- SNARKs are presently only known from *knowledge assumptions*, which are considered "non-standard" in cryptography (in fact, in certain restricted settings, some knowledge assumptions and program obfuscation have been shown to be in contention)
- **Zero-Knowledge SNARK:** A SNARK that also achieves zero knowledge property. Most constructions of ZK-SNARKs require a *common random string* (CRS) setup, namely, where some trusted party is supposed to have computed the CRS and "destroyed" the secret randomness used in its computation. This CRS is used by the prover and the verifier to compute and verify the proof.