Zero-Knowledge Proofs

CS 601.642/442 Modern Cryptography

Fall 2019
Zero-Knowledge Proofs: Brief History

- Invented by Goldwasser, Micali and Rackoff
- Paper rejected three times! Eventually published in 1985
- 30 years later: ZK proofs are now practical. Used in many systems such as ZCash. Very active area of research
- Goldwasser and Micali won the 2012 Turing Award for their foundational work on encryption schemes and proof systems
Example Scenario

- Suppose Alice and Bob are trying to find Waldo in a large picture.
- Alice quickly manages to find Waldo’s location and wants to convince Bob of her knowledge.
- One way to accomplish this is to simply reveal the location.
- Is there a better way to convince Bob without revealing the location?
- ZK proofs provide such a way!
In general, Alice may want to convince Bob that she knows a secret without revealing the secret to Bob.

This is very useful in applications. A simple example is identification (e.g., login to server without revealing the password).

Classical proofs inherently convey “knowledge”.

ZK proofs, as the name suggests, avoid any transmission of knowledge beyond the truthfulness of the statement.

Remarkably, Goldwasser, Micali and Wigderson proved that anything efficiently provable can also be provable in “zero knowledge”.

Today’s Agenda

- To understand ZK proofs, we first need to understand what are proofs
- The power of interaction in proving statements
- What is (zero) knowledge?
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.
1. **Question 1:** How to model efficient verifiability?
   - Verifier must be polynomial time in the length of the statement

2. **Question 2:** Must a proof be *non-interactive*?
   - Or can a proof be a conversation? (i.e., *interactive*)
Interactive Protocols

- Interactive Turing Machine (ITM): A Turing machine with two additional tapes: a read-only communication tape for receiving messages, a write-only communication tape for sending messages.

- An interactive protocol \((M_1, M_2)\) is a pair of ITMs that share communication tapes s.t. the send-tape of the first ITM is the receive-tape of the second, and vice-versa.

- Protocol proceeds in rounds. In each round, only one ITM is active, the other is idle. Protocol ends when both ITMs halt.

- \(M_1(x_1, z_1) \leftrightarrow M_2(x_2, z_2)\): A (randomized) protocol execution where \(x_i\) is input and \(z_i\) is auxiliary input of \(M_i\).

- \(\text{Out}_{M_i}(e)\): Output of \(M_i\) in an execution \(e\).

- \(\text{View}_{M_i}(e)\): View of \(M_i\) in an execution \(e\) consists of its input, random tape, auxiliary input and all the protocol messages it sees.
Interactive Proofs

**Definition (Interactive Proofs)**

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

- **Completeness**: For every \(x \in L\),
  \[
  \Pr \left[ \text{Out}_V[P(x) \leftrightarrow V(x)] = 1 \right] = 1
  \]

- **Soundness**: There exists a negligible function \(\nu(\cdot)\) s.t. \(\forall x \notin L\) and for all adversarial provers \(P^*\),
  \[
  \Pr \left[ \text{Out}_V[P^*(x) \leftrightarrow V(x)] = 1 \right] \leq \nu(|x|)
  \]

**Remark**: In the above definition, prover is not required to be efficient. Later, we will also consider efficient provers.
Why Interactive proofs?

- Let $L$ be a language in $\mathbf{NP}$ and let $R$ be the associated relation.
- For any $x \in L$, there exists a “small” (polynomial-size) witness $w$.
- By checking that $R(x, w) = 1$, we can verify that $x \in L$.
- Therefore, $w$ is a non-interactive proof for $x$.
- E.g. Graph Isomorphism: Two graphs $G_0$ and $G_1$ are isomorphic if there exists a permutation $\pi$ that maps the vertices of $G_0$ onto the vertices of $G_1$.

So why use interactive proofs after all?
Two main reasons for interaction:

1. Proving statements in languages not known to be in $\text{NP}$
   - Single prover [Shamir]: $\text{IP} = \text{PSPACE}$
   - Multiple provers [Babai-Fortnow-Lund]: $\text{MIP} = \text{NEXP}$

2. Achieving privacy guarantee for prover
   - Zero knowledge [Goldwasser-Micali-Rackoff]: Verifier learns nothing from the proof beyond the validity of the statement!
Notation for Graphs

- Graph $G = (V, E)$ where $V$ is set of vertices and $E$ is set of edges
- $|V| = n$, $|E| = m$
- $\Pi_n$ is the set of all permutations $\pi$ over $n$ vertices
- Graph Isomorphism: $G_0 = (V_0, E_0)$ and $G_1 = (V_1, E_1)$ are isomorphic if there exists a permutation $\pi$ s.t.:
  - $V_1 = \{\pi(v) \mid v \in V_0\}$
  - $E_1 = \{\pi(v_1), \pi(v_2) \mid (v_1, v_2) \in E_0\}$
- Alternatively, $G_1 = \pi(G_0)$
- Graph Isomorphism is in $\textbf{NP}$
Graph Non-Isomorphism: $G_0$ and $G_1$ are non-isomorphic if there exists no permutation $\pi \in \Pi_n$ s.t. $G_1 = \pi(G_0)$

Graph Non-Isomorphism is in co-NP, and not known to be in NP
How to Prove Graph Non-Isomorphism?

- Suppose $P$ wants to prove to $V$ that $G_0$ and $G_1$ are not isomorphic.
- One way to prove this is to write down all possible permutations $\pi$ over $n$ vertices and show that for every $\pi$, $G_1 \neq \pi(G_0)$. However, this is not efficiently verifiable.
- How to design an efficiently verifiable interactive proof?
Interactive Proof for Graph Non-Isomorphism

Common Input: \( x = (G_0, G_1) \)

Protocol \((P, V)\): Repeat the following procedure \(n\) times using fresh randomness

\[ V \to P: \] V chooses a random bit \( b \in \{0, 1\} \) and a random permutation \( \pi \in \Pi_n \). It computes \( H = \pi(G_b) \) and sends \( H \) to \( P \)

\[ P \to V: \] P computes \( b' \) s.t. \( H \) and \( G_{b'} \) are isomorphic and sends \( b' \) to \( V \)

\[ V(x, b, b'): \] V outputs 1 if \( b' = b \) and 0 otherwise
$(P, V)$ is an Interactive Proof

- **Completeness**: If $G_0$ and $G_1$ are not isomorphic, then an unbounded prover can always find $b'$ s.t. $b' = b$

- **Soundness**: If $G_0$ and $G_1$ are isomorphic, then $H$ is isomorphic to both $G_0$ and $G_1$! Therefore, in one iteration, any (unbounded) prover can correctly guess $b$ with probability at most $\frac{1}{2}$. Since each iteration is independent, prover can succeed in all iterations with probability at most $2^{-n}$. 
Interactive Proofs with Efficient Provers

- Prover in graph non-isomorphism protocol is inefficient.
- For languages in \textbf{NP}, we can design interactive proofs with \textit{efficient} provers
- Prover strategy must be efficient when it is given a witness \(w\) for a statement \(x\) that it attempts to prove

\textbf{Definition}

An interactive proof system \((P,V)\) for a language \(L\) with witness relation \(R\) is said to have an \textit{efficient prover} if \(P\) is PPT and the completeness condition holds for every \(w \in R(x)\)

\textbf{Main Goal:} \textit{Zero Knowledge}, i.e., ensuring that verifier does not gain any knowledge from its interaction with prover beyond learning the validity of the statement \(x\) (e.g., \(P\)'s witness \(w\) remains private from \(V\))
Towards Zero Knowledge

Q. 1: How to formalize “does not gain any knowledge?”

Q. 2: What is knowledge?
Rules for formalizing “(zero) knowledge”:

**Rule 1:** Randomness is for free

**Rule 2:** Polynomial-time computation is for free

That is, by learning the result of a random process or result of a polynomial time computation, we gain no knowledge.
When is knowledge conveyed?

**Scenario 1:** Someone tells you he will sell you a 100-bit random string for $1000.

**Scenario 2:** Someone tells you he will sell you the product of two prime numbers of your choice for $1000.

**Scenario 3:** Someone tells you he will sell you the output of an exponential time computation (e.g., isomorphism between two graphs) for $1000.

**Think:** Should you accept any of these offers?

We can generate 100-bit random string for free by flipping a coin, and we can also multiply on our own for free. But an exponential-time computation is hard to perform on our own, since we are PPT. So we should reject first and second offers, but seriously consider the third one!
Zero Knowledge: Intuition

- 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*, as in definition of semantic security for encryption.
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 \(\nu(n)\):

- \(\{\text{View}_V[P(x, w) \leftrightarrow V(x, z)]\}\)
- \(\{S(1^n, x, z)\}\)
Remarks on the Definition

- Captures that whatever $V$ “saw” in the interactive proof, it could have generated it on its own by running the simulator $S$
- The auxiliary input to $V$ captures any a priori information $V$ may have about $x$. Definition promises that $V$ does not learn anything “new”
- **Problem:** However, the above is promised only if verifier $V$ follows the protocol
- What if $V$ is malicious and deviates from the honest strategy?
- **Want:** Existence of a simulator $S$ for every, possibly malicious (efficient) verifier strategy $V^*$
- For now, will relax the simulator and allow it to be *expected* PPT, i.e., a machine whose expected running time is polynomial
Zero Knowledge: Definition II

**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 $\nu(n)$:

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

- If the distributions are statistically close, then we call it statistical zero knowledge
- If the distributions are identical, then we call it perfect zero knowledge
Reflections on Zero Knowledge

Paradox?

- Protocol execution convinces $V$ of the validity of $x$
- Yet, $V$ could have generated the protocol transcript on its own

To understand why there is no paradox, consider the following story:

- Alice and Bob run $(P, V)$ on input $x$ where Alice acts as $P$ and Bob as $V$
- Now, Bob goes to Eve: “$x$ is true”
- Eve: “Oh really?”
- Bob: “Yes, you can see this accepting transcript”
- Eve: “That doesn’t mean anything. Anyone can come up with such a transcript without knowing a witness for $x$!”
- Bob: “But I computed this transcript by talking to Alice who answered my challenge correctly every time!”
Moral of the story:

- Bob participated in a “live” conversation with Alice, and was convinced by how the transcript was generated.
- But to Eve, who did not see the live conversation, there is no way to tell whether the transcript is from real execution or produced by simulator.