Zero-Knowledge Proofs of Knowledge

Yehuda Lindell
Bar-Ilan University
Knowledge – Motivation

• Prove that you know the shortest path from A to B
  • A shortest path exists, but who says that you know it?

• Prove identity:
  • For public key \( h = g^x \) in a group where discrete log is hard, prove that I know \( x \)
  • This proves identity since it is my private key and only I know it
  • Attempt: prove in ZK that \( h \in L \) for \( L = \{h \mid \exists x: g^x = h\} \)
  • Problem:
    • This statement is TRUE for all group elements (and so ZK is actually trivial – send YES)
    • Who says that I need to know a witness to prove a true statement
What is Knowledge?

• **Definition**: a student knows the material if she can output it
  • We approximate this by saying that a student knows the material if she can output the answers to the questions on the test

• **Definition**: a machine knows something if it can output it
  • Let R be an NP-relation
    • A machine knows the witness to a statement \( x \) if it can output \( w \) s.t. \( (x, w) \in R \)

• What does it mean for a machine to be able to output it?
Formalizing Knowledge (first attempt)

• Attempt 1: a machine $M$ knows the witness to a statement $x$ if there exists some $M'$ who outputs $w$ s.t. $(x, w) \in R$

• Questions:
  • How does this relate to the machine’s actions (e.g., proving a proof)?
  • How is $M'$ related to $M$; if there is no connection then why does $M$ know it?
Formalizing Knowledge (second attempt)

• Attempt 2:
  • We define a PPT oracle machine $K$, called a knowledge extractor
  • We say that $M$ knows the witness to a statement $x$ if $K^M(\cdot)(x)$ outputs $w$ s.t. $(x, w) \in R$
    • $K$ interacts with $M$ and can use whatever it does to obtain $w$
    • Since $K$ is generic, its ability to output $w$ means that $M$ knows $w$

• Questions:
  • This still doesn’t relate to the machine’s actions (e.g., proving a proof)?
  • $K$ could still just know $w$ independently of $M$
Formalizing Knowledge (third attempt)

• **Definition:**
  - We define a PPT oracle machine $K$, called a knowledge extractor.
  - We say that a prover $P^*$ knows the witness to a statement $x$ if $K^{P^*(\cdot)}(x)$ outputs $w$ s.t. $(x, w) \in R$ whenever $P^*$ convinces $V$ of $x$.

• **Intuition:**
  - $K$ is generic and works for any $x$ and any $P^*$: if $P^*$ can convince $V$ then $K$ can output $w$ and so $M$ knows $w$.

• **Question:** what does it mean: “whenever $P^*$ convinces $V$ of $x$”? 
  - $K$ should run in (expected) polynomial-time and output a witness $w$ with the same probability that $P^*$ convinces $V$ of $x$. 

Formalizing Knowledge (final)

• One can always prove in ZK without knowing, with negligible prob
  • Run the zero-knowledge simulator and hope that the verifier’s queries in the result match the real queries

• The definition is updated to allow a knowledge error $\kappa$, which takes this into account
  • If $P^*$ convinces $V$ of $x$ with probability $> \kappa$, then $K$ should run in (expected) polynomial-time and output a witness $w$ with probability at most $\kappa$ less than $P^*$ convinces $V$ of $x$

• This property is called knowledge soundness
The Definition

• Definition (knowledge soundness):
  • A proof system has **knowledge soundness** with error $\kappa$ if there exists a PPT $K$ s.t. for every prover $P^*$, if $P^*$ convinces $V$ of $x$ with probability $\epsilon > \kappa$, then $K^{P^*} (\cdot) (x)$ outputs $w$ s.t. $(x, w) \in R$ with probability at least $\epsilon(|x|) - \kappa(|x|)$
An Alternative Formulation

**Motivation:** one can trade off running time and success probability
- Definition says: run in PPT and output w.p. $\epsilon$
- Alternative definition: run in expected time $\frac{1}{\epsilon}$ and always output

**Definition (knowledge soundness):**
- A proof system has **knowledge soundness** with error $\kappa$ if there exists a $K$ s.t. for every prover $P^*$, if $P^*$ convinces $V$ of $x$ with probability $\epsilon > \kappa$, then

$$K^{P^*}(\cdot)(x) \text{ outputs } w \text{ s.t. } (x, w) \in R \text{ in expected time } \frac{\text{poly}(|x|)}{\epsilon(|x|) - \kappa(|x|)}$$
Equivalence of the Definitions

• **Original implies alternative:**
  - We are given K that runs in PPT and outputs a witness w.p. $\epsilon$
  - We can run K many times until a witness is output
    - Since it is an **NP relation**, can verify when get correct result
    - Expected number of times needed is $1/\epsilon$

• **Alternative implies original:**
  - We are given K that runs in time $1/\epsilon$ and outputs a witness
  - For $i = 1, ..., n$, run K for $2^{i+1}$ steps; if finish output witness; else proceed w.p. $\frac{1}{2}$
    - Let $i$ be smallest s.t. $2^{i+1} > 1/\epsilon$: probability of getting here is at least $2^{-(i+1)} > \epsilon$
    - Expected running time is $poly(|x|)$
Definition of ZKPOK

• A proof system is a zero-knowledge proof of knowledge if it has
  • Completeness: honest prover convinces honest verifier
  • Zero knowledge: ensures verifier learns nothing
  • Knowledge soundness: ensures prover knows witness

• Zero knowledge is a property of the prover
  • Prover behavior is guaranteed to reveal nothing
  • Protect against a cheating verifier

• Knowledge soundness is a property of the verifier
  • Verifier behavior guarantees that prover knows witness
  • Protect against a cheating prover
Reducing Knowledge Error

• Sequential composition reduces knowledge error exponentially

• Exponentially small error = zero error
  • Assume knowledge error $\kappa < 2^{-|x|}$ and consider alternative definition
  • Run $K^{P^*(\cdot)}(x)$ in parallel to running a brute-force search on witness
    • Assume brute force in time $2^{|x|}$
  • Let $P^*$ be s.t. it convinces $V$ of $x$ with probability $\epsilon$
    • If $\epsilon > 2 \cdot \kappa$ then $\frac{\text{poly}(|x|)}{\epsilon - \kappa} < \frac{2 \cdot \text{poly}(|x|)}{\epsilon}$ and so succeed in time $\frac{\text{poly}'(|x|)}{\epsilon}$
    • If $\epsilon < 2 \cdot \kappa$ then $\frac{\text{poly}(|x|)}{\epsilon} > 2^{|x|} \cdot \text{poly}(|x|)$ and so brute force finishes
Constructing ZKPOKs

A Zero-Knowledge proof for $QR_N$

$x = w^2 \mod N$  \hspace{1cm} P \hspace{1cm} x \in QR_N$  \hspace{1cm} V

$r \in_R \mathbb{Z}^*_N$

\[ y = r^2 \]

\[ b \]

\[ b = 0: \quad z = r \]
\[ b = 1: \quad z = wr \]

\[ b \in_R \{0,1\} \]

\[ z^2 \equiv y \]
\[ z^2 \equiv xy \]
Knowledge Extraction Idea

- $K$ invokes $P^*$ and “receives” some $y$
- $K$ “sends” $P^*$ the query $b = 0$ and receives $z_0$
- $K$ **rewinds** and “sends” $P^*$ the query $b = 1$ and receives $z_1$
- $K$ outputs $w = \frac{z_1}{z_0} \mod N$

**Proof:**

- If $P^*$ convinces w.p. greater than $\kappa = \frac{1}{2}$ then $(z_0)^2 = y$ and $(z_1)^2 = xy$
  - I am assuming for deterministic $P^*$; to discuss!
- Thus $w^2 = \left(\frac{z_1}{z_0}\right)^2 = \frac{xy}{y} = x$ and so $K$ outputs a square root
ZKPOK for NP

An interactive proof for HAM

Ham cycle $w$  
$\pi \in_R S_n$

$P$  
$c = \text{Com}(\pi(G))$

$V$  
$b \in_R \{0,1\}$

$u=\pi(w)$  
$b = 0$: $u \in \text{Dec}(c)$

$b = 1$: $\pi, H = \text{Dec}(c)$

Verify that $u$ is a cycle

Verify that $H = \pi(G)$
ZKPOK for NP

- $K$ invokes $P^*$ and receives a commitment $c$
- $K$ sends $P^*$ the query $b = 0$ and receives a cycle $w$
- $K$ rewinds and sends $P^*$ the query $b = 1$ and receives $\pi, \tilde{G}$

**Proof:**

- If $P^*$ convinces w.p. greater than $\kappa = \frac{1}{2}$ then $w$ is a cycle in $\tilde{G} = \pi(G)$
- Thus, $\pi^{-1}(w)$ is a Hamiltonian cycle in $G$
ZKPOK for NP with Negligible Error

- Run Hamiltonicity $n = |x|$ times sequentially
- Extractor strategy:
  - Consider binary tree of execution
  - Attempt to extract in $i$th execution
    - If $P^*$ answers both queries, get Hamiltonian cycle
    - If $P^*$ answers neither query, $V$ always rejects
    - If $P^*$ answers exactly one query, go down that edge
  - Repeat with next execution
- Extraction fails iff $P^*$ answers exactly one query in each execution
- Thus, extraction works with probability 1 if $\epsilon > 2^{-n}$
Strong Proofs of Knowledge

• **Definition – strong knowledge soundness**
  • A proof system has **strong knowledge soundness** if there exists a negligible function $\mu$ and a PPT $K$ s.t. for every prover $P^*$, if $P^*$ convinces $V$ of $x$ with probability $\epsilon > \mu$, then $K^{P^*(\cdot)}(x)$ outputs $w$ s.t. $(x, w) \in R$ with probability at least $1 - \mu(|x|)$

• **Theorem:** sequential Hamiltonicity is a strong proof of knowledge
Using the Alternative Definition

• **Definition (knowledge soundness):**
  
  A proof system has **knowledge soundness** with error $\kappa$ if there exists a $K$ s.t. for every prover $P^*$, if $P^*$ convinces $V$ of $x$ with probability $\epsilon > \kappa$, then
  
  $$K^{P^*}(\cdot)(x)$$
  
  outputs $w$ s.t. $(x, w) \in R$ in expected time
  
  $$\frac{\text{poly}(|x|)}{\epsilon(|x|) - \kappa(|x|)}$$

• What does it help to run in time
  
  $$\frac{\text{poly}(|x|)}{\epsilon(|x|)}$$
  
  when this may not be polynomial time?
Using the Alternative Definition

• A classic use of zero-knowledge proofs of knowledge:
  • Within a protocol, prover proves the proof
  • To prove security, a simulator (or reduction) needs the witness
    • Unless verifier would reject, in which case it doesn’t matter

• Using ZKPOKs in proofs of security – simulator (or reduction) plays verifier with prover:
  • If the verifier rejects, then the simulator can halt, since a real verifier would
  • If the verifier accepts, then the simulator now has to extract the witness
ZKPOK Inside a Protocol

• Recall simulator (reduction) strategy:
  • Verify, then halt if reject and extract if accept

• What is the expected running time of this simulator (reduction)?
  • Probability that prover convinces verifier is $\epsilon(|x|)$
  • Assuming that the knowledge error $\kappa$ is 0:
    $$E[\text{Time}] = (1 - \epsilon(|x|)) \cdot \text{poly}(|x|) + \epsilon(|x|) \cdot \frac{\text{poly}(|x|)}{\epsilon(|x|)} = \text{poly}(|x|)$$
  • Assuming that the knowledge error $\kappa$ is negligible:
    $$E[\text{Time}] = (1 - \epsilon(|x|)) \cdot \text{poly}(|x|) + \epsilon(|x|) \cdot \frac{\text{poly}(|x|)}{\epsilon(|x|) - \kappa(|x|)} = \text{poly}(|x|) + \frac{\epsilon(|x|)}{\epsilon(|x|) - \kappa(|x|)}$$
  • Actually not polynomial, but can be fixed...
ZKPOK in a Protocol

• The issue that arises is that need to both
  • Simulate the view of the prover in the execution, and
  • Extract a witness

• This is called “witness-extended emulation”

• A witness-extended emulator $E^{P^*}(\cdot)(x)$ outputs a VIEW and some $w$:
  • The view output is indistinguishable from a real execution
  • The probability that the view is accepting and yet $(x, w) \not\in R$ is negligible
  • $E$ runs in expected polynomial-time
Witness-Extended Emulation

• **Lemma**: If \((P, V)\) is a ZKPOK, then there exists a witness extended emulator for \((P, V)\).
  • Very useful when use ZKPOK inside proofs of security (and greatly simplifies)

• Can also formalize an ideal ZK functionality:
  \[
  \mathcal{F}_{zk}(x, w, x) = (\lambda, R(x, w))
  \]

• **Lemma**: If \((P, V)\) is a ZKPOK, then it securely computes the ideal ZK functionality (in the secure computation sense).
Other Applications

- A zero-knowledge proof for $NQR_N$
- Non-oblivious encryption
- Prove that committed value has a property, for statistically hiding
- Identification schemes
A zero-knowledge proof for $QR_N$

Interactive proof for $QR_N$ [GMR’85]

<table>
<thead>
<tr>
<th>P</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \notin QR_N$</td>
<td>$b \in_R {0,1}$</td>
</tr>
<tr>
<td>$z = y^2$</td>
<td>$y \in_R \mathbb{Z}_N^*$</td>
</tr>
<tr>
<td>$z = xy^2$</td>
<td>$b = 0$</td>
</tr>
<tr>
<td>$b'(z) = 0$</td>
<td>$z \in QR_N$</td>
</tr>
<tr>
<td>$b'(z) = 1$</td>
<td>$b \notin QR_N$</td>
</tr>
<tr>
<td>$b' \not= b$</td>
<td></td>
</tr>
</tbody>
</table>
A ZK proof for $\overline{QR_N}$

• Why is the proof not ZK?
  • The verifier may have some $z$ and wants to know if is QR or not

• How can we make this proof ZK?
  • The verifier sends $z$ and proves that it knows $y$ s.t. $z = xy$ or $z = xy^2$

• Why is ZK not enough and why is a ZKPOK needed?
  • Intuitively: for every $z$, there exists a $y$ s.t. $z = xy$ or $z = xy^2$, so statement is always true
  • Formally: simulation strategy
A ZK proof for $\overline{QR}_N$

• Simulation Strategy
  • Simulator $S$ runs $V^*$ and gets $z$
  • Simulator doesn’t know whether it should answer $b = 0$ or $b = 1$
  • Simulator runs the knowledge extractor on the proof from $V^*$ and gets $y$
  • Simulator checks if $z = xy$ or $z = xy^2$, and so knows if $b = 0$ or $b = 1$
Non-Oblivious Encryption

• Provide an encryption and prove that you know what’s encrypted

• Motivation:
  • Prevent copying (e.g., in auction)
  • Guarantee non-malleability (did not take a previous ciphertext and maul)
Prove Property of Statistical Committed Value

• Consider a statistically-hiding commitment scheme
  • A commitment value $c$ can be a commitment to any message
• Committer wishes to prove that it committed to a value in a certain range (or any other property)
• Statement is almost always true for any given $c$
• The question is whether the committer knows a decommitment to a message with this property
• **Rule**: whenever ZK is used with statistical hiding, ZKPOK is needed
Identification Schemes

• Alice has a public key $h = g^x$
• In order to authenticate, she proves that she knows the dlog of $h$
• This must be a ZKPOK, since ZK for the language of DLOG is trivial
Questions?