Zero-Knowledge and An Epistemic Characterization of it

Yiting Wang

Department of Philosophy, Peking University

2020.2.25

Motivation of Zero-Knowledge

Intuitively, a zero knowledge proof system is a way of convincing someone of a fact without giving them any additional knowledge.

- Prime Factorisation(Counterexample)
  - A know that 26781 is not a prime number since he knows that 26781 is equal to 113 times 237. To prove "26781 is not a prime number" to B, it's natural for A to demonstrate that $26781 = 113 \times 237$. However, in this example, B not only is convinced that 26781 is not a prime number, but also learns its factorization.
- Color Non-blindness
  - A have two balls: one is red, the other is green. They are identical except for the color. To B, who is color-blind, they are completely identical. A wants to prove to B that these two balls have different colors without revealing the exact color each ball has. What should A do?

What does “Proof” mean?

It is often regarded that saying a language (a subset of string over a given alphabet) $L \in NP$ is equivalent to saying that there is a polynomial time “proof system” for L. The proof system we have in mind is one where on input x, a “prover” creates a string α, and the “verifier” then computes on x and α in time polynomial in the length of the binary representation of x to check that x is indeed in L.

Desired Properties of Zero-Knowledge Interactive Proof System

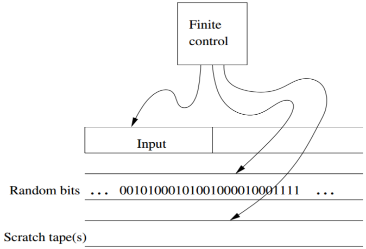
- Completeness: If $x \in L$, then, with very high probability, the verifier is "convinced" of this statement, after interacting with the prover.
- Soundness:If $x \notin L$, then no matter what the prover does, with very high probability, prover fails to fool the verifier.
- Zero-Knowledge: The verifier will not learn anything from the interaction apart from the fact that the statement is true.

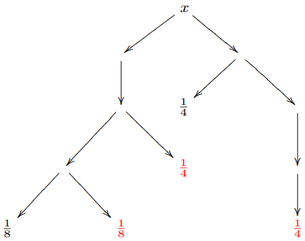
Probabilistic Polynomial Time Turing Machine

A probabilistic Turing machine (PTM) M is a type of nondeterministic Turing machine where each nondeterministic step is called a (fair) coin-flip step and has two at most legal next moves. We assign a probability to each branch b of M 's computation on input x as follows. Define the probability of branch b to be

$$\Pr[b] = 2^{-k}$$

where k is the number of coin-flip steps that occur on branch b.





Define the probability that M accepts x to be

$$\Pr[M \text{ accepts } x] = \sum_{b \text{ is an accept branch}} \Pr[b]$$

In other words, the probability that M accepts x is the probability that A would reach an accepting configuration if A simulated M on x by flipping a coin to determine which move to follow at each coin-flip step. We let

$$\Pr[M \text{ rejects } x] = 1 - \Pr[M \text{ accepts } x]$$

Define that M recognizes language L if

(1) $x \in L$ implies $\Pr[M \text{ accepts } w] > \frac{1}{2}$, and

(2) $x \notin L$ implies $\Pr[M \text{ rejects } w] \geq \frac{1}{2}$

Expected Polynomial-Time

Assume $PTM\,\mathcal{M}$ always halts. Let $A_x = \{\alpha \mid M(x,\alpha)$ accpts after$|\alpha|$ moves $\}$.

$$expecttime_M(x) = \Sigma_{\alpha \in A_x} |\alpha| \cdot 2^{-|\alpha|}$$

Then expected time complexity of M is

$$t_M(n) = max\{expecttime_M(x) \mid |x| = n\}$$

$TM\ \mathcal{M}$ is called “expected polynomial-time” if there exists a polynomial function $Q(x)$ such that, for all $x \in L(\mathcal{M})$, $t_M(|x|)$ is bounded above by $Q(|x|)$.
Let

$$ETIME(t(n)) = \{L(M) \mid M \text{ is a } PTM \text{ and } t_M(n) \leq (t(n))\}$$

Let PP be the set of all languages recognized by Polynomial PTM. Then

$$PP = \bigcup_{k>0} ETIME(n^k)$$

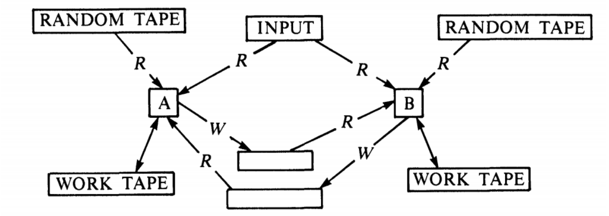
Interactive Turing machine

An interactive Turing machine (ITM) is essentially a PTM equipped with a read-only input tape, a work tape, a random tape, one read-only communication tape, and one write-only communication tape.

Interactive Protocol

An interactive protocol is an ordered pair of ITMs $\langle P, V\rangle$ such that P and V share the same input tape, V's write-only communication tape is P's read-only communication tape and vice versa.

- The two machines take turns in being active, with P being active first.
- P first performs some internal computation using its input tape, work tapes, communication tape and random tape, then writes a string (for V) on its write-only communication tape.
- The ith message of P is the entire string that P writes on its communication tape during its ith active stage.
- As soon as machine P writes its message, it is deactivated and machine V becomes active, unless the protocol has been terminated.



Interactive protocol

- Let P be an ITM. Then $P(x, r, \alpha_1, \alpha_2, \ldots, \alpha_n)$ denotes the message sent by P on input x, random tape contents r, after receiving the messages α_1 through α_n.
- Let P and V be an interacting pair of ITMs. Then, $[P(x), V(y)]$ denotes the output distribution of V on input y, when P has input x. The probability space is induced by the unbiased coin tosses of both machines.

Interactive Proof System

Definition [GMW91]

An interactive proof system for a language L is a pair of ITMs, $\langle P, V\rangle$, such that V is expected polynomial-time and the following two conditions hold:

(1) Completeness Condition. For every constant $c > 0$, if $x \in L$

$$\Pr([P(x), V(x)] = 1) \geq 1 - |x|^{-c}$$

(2) Soundness Condition. For every constant $c > 0$, every interactive Turing machine P^*, if $x \notin L$

$$\Pr\left([P^*(x), V(x)] = 0\right) \geq 1 - |x|^{-c}$$

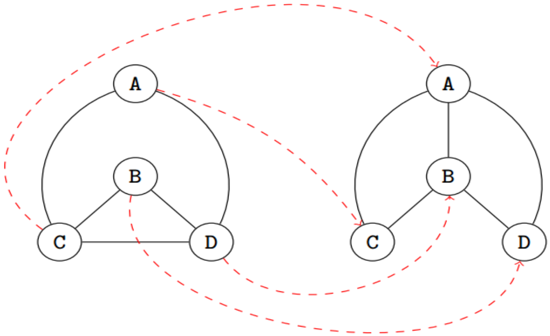
W denote by IP the class of languages having interactive proof systems.

Note that NP is a special case of IP ($|x|^{-c} = 0$)

An Example: Graph Isomorphism

Definition

Let $V(G), E(G)$ denote the vertex set and edge set of G respectively. Two graph G_0, G_1 are isomorphic ($G_1 \simeq G_2$) if there exists a permutation $\pi : V(G_0) \mapsto V(G_1)$ such that $\forall x, y \in V(G_0), (x, y) \in E(G_0)$ if and only if $(\pi(x), \pi(y)) \in E(G_1)$ ". The permutation π is called an isomorphism (i.e. $G_1 = \pi G_0$).



Zero-Knowledge Proof for Graph Isomorphism

Statement: Two (undirected) graphs $G_0 = (V, E_0)$, $G_1 = (V, E_1)$ are isomorphic.

Prover's witness: An isomorphism f between G_0 and G_1.

Protocol:

- (P_1) The prover picks a random permutation h from $Sym(V)$, which the set of permutations on V and computes $H = h(G_1)$. It stores h and sends H to the verifier.
- (V_1) The verifier picks a uniformly random challenge $b \in \{0, 1\}$ and send b to P (Intuitively, the verifier asks the prover to show him that H and G_b are indeed isomorphic).
- (P_2) If $b = 0$, the prover sends $g = h \circ f$ to the verifier. If $b = 1$, the prover sends $g = h$ to the verifier.
- (V_2) If the permutation h received from the prover is not an isomorphism between G_b and H (i.e., $H \neq g(G_b)$), then the verifier stops and rejects; otherwise, back to (P_1).

If the verifier has completed n iterations of the above steps without rejecting, then he accepts.

Completeness

Suppose $G_0 \simeq G_1$ and the verifier acts according to the protocol (denoted "honest verifier")

- If $b = 0$, then verifier will ask the prover to show that $H \simeq G_0$). The prover does this by sending the verifier $g := h \circ f$ as a witness. Since $H := h(G_1)$ and $g(G_1) = h \circ f(G_1) = h(G_0)$, the verifier will find that $H = g(G_1)$, and accepts.
- If $b = 1$, then the verifier will the prover to show that $H \simeq h(G_1)$. Recall that $H := h(G_1)$, and that when the verifier sends $b = 1$ the prover returns $g = h$. Certainly, $h(G_1) \simeq h(G_1)$, so the verifier will accepts.

Soundness

Suppose $G_0 \not\simeq G_1$. For any graph G^*, $G^* \simeq G_0$ and $G' \simeq G_1$ can not both hold. If the verifier sends $b = 1$, then the prover sends h in which case the verifier will accept. However, when $b = 0$, the prover needs to come up with a permutation g that shows $h(G_0) = g(G_1)$, and if $G_0 \not\simeq G_1$, then such a task is not possible, causing the verifier to reject. Since the verifier uniformly randomly samples b from $\{0, 1\}$ it follows that, for any P^*

$$\Pr[P^* \text{ convinces } V \text{ that } G_0 \simeq G_1] \leq \frac{1}{2}$$

Thus, after performing n repetitions, for any P^*

$$\Pr[P^* \text{ convinces } V \text{ that } G_0 \simeq G_1] \leq 2^{-n}$$

Intuition behind Zero-knowledge Property

(1) All information V gets from the interactive proof is that $x \in L$, which means whatever is computed after interacting with P could have been computed without interactions.

(2) V's view (V's random coins and messages it receives in the interactive proof system) can be efficiently (polynomially) "simulated" by a simulator S_V.

(3) For every polynomial time V^*, the distribution that V^* "sees" on all its tapes, when interacting with P on input $x \in L$, is "indistinguishable" from a distribution that can be computed from x in polynomial time.

Polynomially Indistinguishable

Definition [GMW91]

Let $S \subseteq \{0,1\}^*$ be an infinite set of strings, $\Pi_1 = \{\Pi_1(x)\}_{x \in S}$ and $\Pi_2 = \{\Pi_2(x)\}_{x \in S}$ be two probability ensembles (i.e., for every $i \in \{1,2\}$ and $x \in S$, $\Pi_i(x)$ is a random variable assuming values in $\{0,1\}^*$).
For every algorithm$A$, let $p_i^A(x)$ denote the probability that A outputs 1 on input x and an element chosen according to the probability distribution $\Pi_i(x)$. Formally,

$$p_i^A(x) = \sum_{\alpha} \Pr(A(x,\alpha) = 1) \cdot \Pr\left(\Pi_i(x) = \alpha\right)$$

The ensembles $\Pi_1 = \{\Pi_1(x)\}_{x \in S}$ and $\Pi_2 = \{\Pi_2(x)\}_{x \in S}$ are polynomially indistinguishable if for every expected polynomial-time algorithm A, for every constant $c > 0$ and for all sufficiently long $x \in S$

$$\left|p_1^A(x) - p_2^A(x)\right| \leq |x|^{-c}$$

Computational Zero-Knowledge

Definition [GMW91]

Let (P, V) be an interactive proof system for a language L. (P, V) is computational Zero-Knowledge (for L) if for every expected polynomial-time interactive Turing machine V^*, there exists an expected polynomial-time machine S_{V^*} such that the probability ensembles $\{S_{V^*}(x)\}_{x \in L}$ and $\{[P(x), V^*(x)]\}_{x \in L}$ are polynomially indistinguishable.

Zero-Knowledge Proof for Graph Isomorphism

Protocol:

- (P_1) The prover picks a random permutation h from $Sym(V)$, which the set of permutations on V and computes $H = h(G_1)$. It stores h and sends H to the verifier.
- (V_1) The verifier picks a uniformly random challenge $b \in \{0, 1\}$ and send b to P. (Intuitively, the verifier asks the prover to show him that H and G_b are indeed isomorphic)
- (P_2) If $b = 0$, the prover sends $g = h \circ f$ to the verifier. If $b = 1$, the prover sends $g = h$ to the verifier.
- (V_2) If the permutation h received from the prover is not an isomorphism between G_b and H (i.e., $H \neq g(G_b)$.), then the verifier stops and rejects; otherwise, back to (V_1).

If the verifier has completed n iterations of the above steps, then he accepts.

The simulator S_{V^*} will do the following (Note that the simulator can depend on V^* and hence in particular can use the strategy V^* in its computation) :

(1) Sample uniformly σ from $Sym(V)$ and b' from $\{0,1\}$, and set $H := \sigma\left(G_{b'}\right)$.

(2) Send H into V^* to get b.

(3) If $b' = b$ output σ, otherwise repeat from (1)

(4) Halts and accepts.

To argue that $\{S_{V^*}(x)\}_{x\in L}$ and $\{[P(x), V^*(x)]\}_{x\in L}$ are polynomially indistinguishable (actually identical), it is sufficient to show that the distribution of the output in step (1) is indistinguishable from step (P_1) in the original protocol, since when $G_0 \simeq G_1$ the rest of S_{V^*} is identical to the original protocol.

System and Runs

- Each agents starts in some initial local state; its local state then changes over time.
  - The agent's local state at time $m \geq 0$ consists of the time on the global clock, the agent's initial information (if any), the history of messages the agent has received from other agents and read, and the history of coin flips used.
- A global state is a tuple of local states, one for each processor. (Sometimes there is one for nature, which keeps track of information about the system not known to any of the agents.)
- A run is an infinite sequence of global states. Given a run r and a time piont m, we refer to (r, m) as a point.
  - we denote the local state of processor q in $r(m)$ by $r_q(m)$
- A system is a set of all possible runs of a particular protocol.

For each processor q, K_q is a binary relations on System. $K_q(r, m) = \{(r', m') : r'_q(m') = r_q(m)\}$ can be thought of as the set of points that a considers possible at (r, m), because he has the same local state at all of them. Since the agent's local state at time m consists of the time on the global clock, any point that a considers possible at (r, m) is also at time m, so $K_q(r, m) = \{(r', m) : r_q(m) = r'_q(m)\}$

We denote by $P \times V$ the system consisting of all possible executions of (P, V) and by $P \times \mathcal{V}^{pp}$ the system consisting of the union of the systems $P \times V^*$ for all probabilistic, polynomial-time protocols V^*

Probability in System

When reasoning about probabilistic systems, we want to talk about that λ is the conditional probability of φ, given q 's local state.

Given a system $\mathcal{R}$, we associate with every processor q and every point (r, m) a probability space $\mathcal{P}(q, (r, m)) = \big(S_{q,(r,m)}, X_{q,(r,m)}, \mu_{q,(r,m)}\big)$ where $S_{q,(r,m)}$ is a set of points, $X_{q,(r,m)}$ is a set of measurable subsets of $S_{q,(r,m)}$, and $\mu_{q,(r,m)}$ is a probability measure.

Intuitively, the set $S_{q,(r,m)}$ is a subset of the points q thinks possible at (r, m), and $\mu_{q,(r,m)}$ determines the probability with which q considers a particular point in $S_{q,(r,m)}$ to be the actual point (r, m).

The set $S_{q,(r,m)}(\varphi)$ is then defined to consist of those points in $S_{q,(r,m)}$ at which φ holds.

Semantics

- $(\mathcal{R}, r, m) \models s \in \mathrm{R_L}(x)$ if $r_c(0) \in L$ and $r_p(0) \in \mathrm{R_L}\,(r_c(0))$,
    - $r_c(0)$ is the common inputs.
    - $\mathrm{R_L}$ is the "witness" relation such that $x \in L$ iff there exists a y such that $(x, y) \in \mathrm{R_L}$.
- $(\mathcal{R}, r, m) \models \phi$ iff $(r, m) \in \pi(\varphi)$ where π is an interpretation associating with each primitive fact φ a set $\pi(\varphi)$ of points.
- $(\mathcal{R}, r, m) \models K_q\varphi$ iff $(\mathcal{R}, r', m') \models \varphi$ for all $(r', m') \in \mathcal{K}_q(r, m)$.
    - Intuitively, agent q knows φ if φ is true at all the worlds that agent q considers possible.
- $(\mathcal{R}, r, m) \models$ at time m^* φ iff $(\mathcal{R}, r, m^*) \models \varphi$
- $(\mathcal{R}, r, m) \models pr_q^\lambda(\varphi)$ iff where φ holds with probability at least λ over all points where agent q has the same local state as at (r, m)
    - It intuitively says that processor q knows that φ must hold with probability at least λ.

We write $\mathcal{R} \models \varphi$ if $(\mathcal{R}, r, m) \models \varphi$ holds for all points (r, m) in $\mathcal{R}$.

A fact φ about the initial state of the system can be identified with a binary relation R_φ on $S \times T \times \{0,1\}^*$, where φ is true of $(s,t) \in S \times T$ iff there exists a y such that $R_\varphi((s,t),y)$ holds. y is a witness to φ being true of (s,t).

$[HPR09]$ identify "knowing some fact φ about the initial state i " with "being able to generate a witness to φ being true of (s,t) ".

We will capture verifier's ability to generate such witnesses for R by using an algorithm M that takes as input the verifier's local state t and is supposed to return a y such that $R(s,t,y)$ holds.

Generate a witness for some relation R

Define a function $\mathbf{M} : \mathcal{TM} \rightarrow \mathcal{TM}$; intuitively $\mathbf{M}\left(V^*\right)$ is the decoding procedure for the verifier protocol V^*.
To reason about this in the language, we add a primitive proposition $M_{v,R}$ to the language, where

$$(\mathcal{R}, r, m) \vDash \mathbf{M}_{v,R} \text{ if } R\left(r_p(0), r_v(0), \mathbf{M}\left(V^*\right)\left(r_v(m)\right)\left(\rho^r\right)\right) \text{ holds}$$

and V^* is the verifier protocol in run r and ρ^r is the extra random tape that is part of nature's local state in run r

For any constant λ, let $G_v^{\mathrm{M},m^*,\lambda}\ R$ be an abbreviation of $pr_v^{\lambda}\ (at\ time\ m^*\ \mathbf{M}_{v,R})$, which intuitively says that "the verifier can generate a y satisfying R using M with probability λ at time m^*". Similarly, $G_p^{M,m^*,\lambda}R$ intuitively says prover can generate a y satisfying R using M with probability λ at time m^*.

Now we can capture the intuitively that if the verifier knows that he can, at some future time during the interaction with the prover, generate a witness for some relation R on the initial state with some probability, then he knows that he can generate a witness for R at time 0, with almost the same probability.

Relation Hiding

Let $\mathcal{EPPT}$ be the set of all expected probabilistic polynomial time algorithms.

A function $\epsilon : \mathbb{N} \to [0,1]$ is negligible if for every positive integer k there exists an $n_0 \in \mathbb{N}$ such that for all $n > n_0, \epsilon(n) < \frac{1}{n^k}$, that is, ϵ is eventually smaller than any inverse polynomial.

The system $\mathcal{R}$ is relation hiding for L if for every polynomial-time relation R on $S \times T \times \{0,1\}^*$ and function $\mathbf{M} : \mathcal{TM} \to \mathcal{EPPT}$, there exist functions $\mathbf{M}' : \mathcal{TM} \to \mathcal{EPPT}, \epsilon : \mathcal{TM} \times \mathbb{N} \to [0,1]$ such that for every Turing machine $V^*, \epsilon(V^*, n)$ (where $n = |x|$) is a negligible function, and for every $0 \leq \lambda \leq 1$ and time m^*

$$\mathcal{R} \models \textit{at time } 0 \left(\mathrm{s} \in \mathrm{R_L}(\mathrm{x}) \wedge G_v^{\mathrm{M}, m^*, \lambda} R \Rightarrow G_v^{\mathrm{M}', 0, \lambda - \epsilon} R \right)$$

Characterizing ZK

Theorm [HPR09]

The interactive proof system (P, V) for L is computational zero knowledge iff the system $P \times \mathcal{V}^{pp}$ is relation hiding for L.

Proof. (Intuition)

For "if" direction, suppose that (P, V) is a computational zero knowledge system. If V^* is the verifier protocol in run $r \in P \times \mathcal{V}^{pp}$, then there is a simulator machine S_{V^*} that produces verifier views that no polynomial distinguisher D can distinguish from views during possible interactions with the prover.
If the verifier has an algorithm $M(V^*)$ that takes as input his view at a final point of the interaction and generates a y satisfying the relation R, then he can generate such a y before the interaction by running the simulating machine S_{V^*} at the initial point to get a final view, and then running $M(V^*)$ on this view to generate y. We can therefore construct the function M' using M and S_{V^*}.

Proof. (Intuition)

For the "only if" direction, given an arbitrary protocol V^*, we construct a relation R such that the verifier has an algorithm for generating witnesses for R after the interaction. Since $P \times \mathcal{V}^{pp}$ is relation hiding for L, the verifier has an algorithm for generating witnesses for R at initial points of the interaction. We then use this generating machine to implement a simulator S_{V^*} that fools any distinguisher.

Thanks for your attention!

References I

Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, and Jens Groth.
Efficient zero-knowledge proof systems.
In *Foundations of security analysis and design VIII*, pages 1–31. Springer, 2016.

Ronald Fagin, Yoram Moses, Joseph Y Halpern, and Moshe Y Vardi.
Reasoning about knowledge.
MIT press, 2003.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff.
The knowledge complexity of interactive proof systems.
SIAM Journal on computing, 18(1):186–208, 1989.

Oded Goldreich, Silvio Micali, and Avi Wigderson.
Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems.
Journal of the ACM (JACM), 38(3):690–728, 1991.

References II

Joseph Halpern, Yjoram Moses, and Mark Tuttle.
A knowledge-based analysis of zero knowledge.
In *Proceedings of the twentieth annual ACM symposium on Theory of computing - STOC '88*, pages 132–147. ACM Press.

Joseph Y Halpern, Rafael Pass, and Vasumathi Raman.
An epistemic characterization of zero knowledge.
In *Proceedings of the 12th Conference on Theoretical Aspects of Rationality and Knowledge*, pages 156–165, 2009.