An Epistemic Characterization of Zero Knowledge

Joseph Y. Halpern, Rafael Pass and Vasumathi Raman

Cornell University

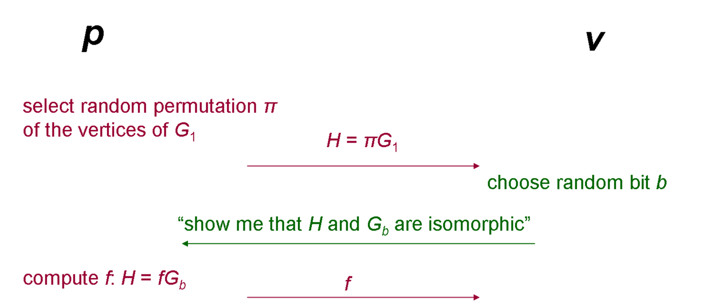
{halpern, rafael, vraman}@cs.cornell.edu

1. Zero Knowledge Proofs

A zero knowledge (ZK) proof system is a way of convincing someone of a fact without giving them any additional knowledge. But what does 'not giving them any additional knowledge' mean?

Let us consider an example of a ZK proof.

- Suppose that a prover (p) wants to prove to a verifier (v) that two graphs G_0 and G_1 are isomorphic.



- v rejects if f is not an isomorphism between G_b and H, otherwise he accepts.

2. Why does this work?

- If p knows an isomorphism between G_0 and G_1, then p can prove upon request that either of (H, G_0) and (H, G_1) are isomorphic (if not, he has a 50 percent chance of failure).
- v repeats this, say 100 times. If p gets it right every time, then v is quite convinced that G_0 and G_1 are isomorphic (or that p is incredibly lucky).
- Moreover, v does not learn anything, because he could have generated the conversation (including p's response) on his own, using a simulator that selects b and then computes a random isomorphic copy of G_b.

3. Intuitive Definition

- A pair of protocols (P, V) for a prover p and verifier v is a *perfect zero knowledge proof system* for L if it is
  - **Sound:** if $x \notin L$, $\Pr(v \texttt{ accepts}) = 1/3$.
  - **Complete:** if $x \in L$, $\Pr(v \texttt{ accepts}) = 2/3$.
  - **Simulable:** no matter what protocol V^* the verifier uses, there is a probabilistic polynomial time "simulator" S_{V^*} that he could use to simulate possible conversations with the prover.
    - ∗ Formally, for every $x \in L$, $(P, V^*)(x)$ (the set of possible runs of the protocol (P, V^*) on input x) and $S_{V^*}(x)$ are identically distributed.
    - ∗ So there is nothing the verifier can do (no protocol he can follow) to learn anything he shouldn't.
- There is an analogous definition of *computational ZK*.
  - This requires only that (P, V^*) on input x) and $S_{V^*}(x)$ be indistinguishable by a polynomial-time verifier.

4. What is "Knowledge"?

CRYPTOGRAPHY	EPISTEMIC LOGIC
• Defined with respect to *computational ability*	• Defined with respect to *what the agent considers possible*
• Bob gains knowledge after interacting with Alice if, after the interaction, Bob can easily compute something that was hard for him earlier	• Bob gains knowledge of fact φ after interacting with Alice if, after the interaction, φ is true in every world Bob considers possible (whereas it was false in some worlds he considered possible before the interaction)

How are these notions related?

5. Previous Work

- Halpern, Moses and Tuttle [HMT 1988] proposed a logical definition of "generating a y satisfying $R(x, y)$" for a relation R.
  - They showed that, if R is testable in polynomial time and the verifier can generate a y satisfying $R(x, y)$ at the end of a ZK proof, he can do so at the start.
  - They called this property *generation security*.
- They left open the question of finding an epistemic statement that is sufficient for ZK.
  - We provide such a statement.

6. The Runs and Systems Framework

- [Fagin, Halpern, Moses and Vardi, 1995]
- Each agents starts in some initial *local state*; its local state then changes over time.
  - A *global state* is a tuple of local states.
- A *run* is an infinite sequence of global states – a possible execution of a protocol. Given a run r and a time m, we refer to (r, m) as a *point*.
- A *system* is a set of runs.
  - often the set of all possible runs of a protocol.
- We start with a collection of primitive facts .
  - e.g. "$x \in L$", where L is some set of strings.
- An interpretation π associates with each primitive fact φ a set $\pi(\varphi)$ of points.
  - $((\mathcal{R}, r, m) \models \varphi$ iff $(r, m) \in \pi(\varphi))$
- $(\mathcal{R}, r, m) \models pr_a^\lambda \varphi$ iff φ holds with probability $\geq \lambda$ over all points where a has the same local state as at (r, m).
- Write $\mathcal{R} \models \varphi$ if $(\mathcal{R}, r, m) \models \varphi$ for all $(r, m) \in \mathcal{R}$.

7. Knowledge as Ability to Generate a Witness

- Intuitively, in a ZK proof, the verifier learns nothing about the initial state of the system.
  - Of course, the verifier may learn facts like "the prover sent 337 in the second round of the interaction."
- Let $\mathcal{I}$ be the set of possible initial states of the system. A fact φ about the initial state of the system can be identified with a binary relation R_φ on $\mathcal{I} \times \{0, 1\}^*$, where φ is true of $i \in \mathcal{I}$ iff there exists a y such that $R_\varphi(i, y)$ holds.
  - y is a witness to φ being true of i.
- We identify "knowing some fact φ about the initial state i" with "being able to generate a witness to φ being true of i".
- In a ZK proof of membership in a language L, the initial global state of the system is a tuple in $S \times T$, where S is the set of prover initial local states and T is the set of verifier initial local states.

8. Formalizing Generating a Witness for R

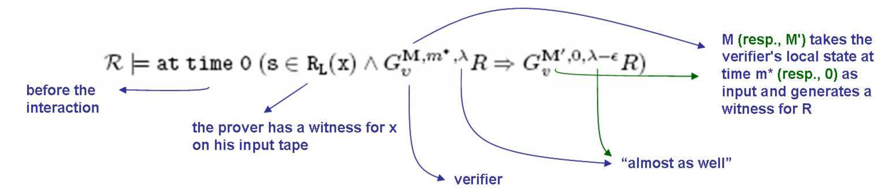
- We want to capture the ability of the verifier to generate witnesses for R using just its local state.
- Formally, the verifier has an algorithm $\mathbf{M}$ that, given as input the local state t of the verifier, generates a witness y such that $R(s, t, y)$ holds.
  - The input x (for which we want to check membership in L) is in the verifier's local state.
  - $\mathbf{M}$ does not get the prover's state s as input.

New primitive propositions

- $\mathbf{M}_{v,R}$ (where $\mathbf{M}$ is an algorithm)
  - Intuitively, $(\mathcal{R}, r, m) \models \mathbf{M}_{v,R}$ if $\mathbf{M}(t)$ returns a y such that $R(s, t, y)$ holds, where s is the prover's state and t is the verifier's state at $(r, 0)$.
- $G_v^{\mathbf{M},m^*,\lambda} R$
  - Read "the verifier can generate a y satisfying relation R using $\mathbf{M}$ with probability λ at time m^*."
  - Formally, $(\mathcal{R}, r, m) \models G_v^{\mathbf{M},m^*,\lambda} R$ if $(\mathcal{R}, r, m) \models pr_v^\lambda(\texttt{at time m}^*\ \mathbf{M}_{v,R})$.

9. Relation Hiding

- We consider interactive proofs of languages L that have a "witness relation" R_L that is computable in time polynomial in $|x|$.
  - $x \in L$ iff there exists a y such that $(x, y) \in R_L$.
  - Let $R_L(x) = \{y : (x, y) \in R_L\}$.
- The system $\mathcal{R}$ is *relation hiding for L* if, for all relations R, algorithms $\mathbf{M}$, and times m^*, there exists an algorithm $\mathbf{M}'$ and a negligible function ϵ such that



In words, for any R, if the verifier can generate a y satisfying R using only the information in his local state at any time m^*, he can do so "almost as well" initially.

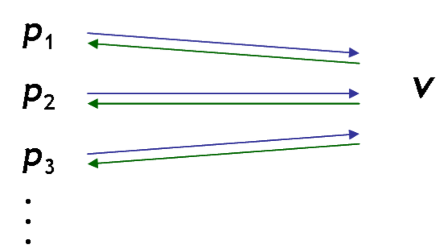
- Perfect relation hiding holds if $\epsilon = 0$.

10. Characterizing ZK

- ***Theorem 1:*** The interactive proof system (P, V) for L is computational *(resp., perfect)* zero knowledge iff the system $P \times \mathcal{V}^{pp}$ is *(perfect)* relation hiding for L.
  - The runs of system $P \times \mathcal{V}^{pp}$ are all possible interactions of a prover running P with a verifier running some probabilistic polynomial time protocol.
- **Unlike HMT's notion of generation security**
  - We consider relations on the entire initial state (i.e., on $S \times T$), not just on L.
  - We require that the probability of generating a y initially be close to the probability at time m^*.
    - ∗ Generation security just requires that if the probability is $\geq 2/3$ at time m^*, then it is $\geq 2/3$ initially.
- We can essentially represent generation security in our language:
  - For all verifier protocols V^*, relations $R(x, y)$, algorithms $\mathbf{M}$, and times m^*, there exists an algorithm $\mathbf{M}'$ and negligible function δ such that $P \times V^* \models \texttt{at time } 0(s \in R_L(x) \implies pr_p^{1-\delta}(G_v^{\mathbf{M},m^*,2/3} R \implies G_v^{\mathbf{M}',0,2/3} R))$.

11. Concurrent ZK

- ZK proofs are often used in the midst of other protocols. When this is done, several ZK proofs may be going on concurrently – an adversary may be able to pass messages between various invocations to gain information.
- Concurrent ZK tries to capture the intuition that no information is leaked even in the presence of several concurrent invocations of a zero-knowledge protocol.



12. Characterizing Concurrent ZK

- We can model a concurrent ZK system with a single verifier and an infinite number of provers.
  - All the provers have the same initial state and use the same protocol P.
  - P is such that provers talk only to the verifier (they do not talk to each other).
- Given a prover protocol P, let $\tilde{P} \times \mathcal{V}^{pp}$ denote the system with runs of this form, where all provers run P and the verifier runs some probabilistic polynomial time protocol.

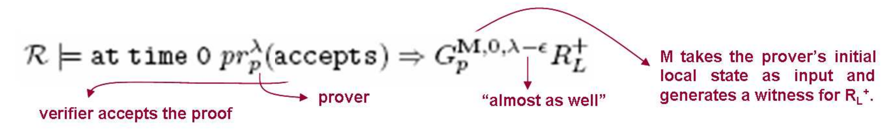
Theorem 2: The interactive proof system (P, V) for L is computational concurrent zero knowledge iff the system $\tilde{P} \times \mathcal{V}^{pp}$ is relation hiding for L.

13. Proofs of Knowledge

In a proof of knowledge, the prover not only convinces the verifier of φ, but also that it possesses, or can "feasibly compute", a witness for φ from its initial secret information.

Witness Convincing

- Define a relation R_L^+ such that $(s, t, y) \in R_L^+$ iff $y \in R_L(x)$.
- The system $\mathcal{R}$ is *witness convincing for L* if, for all algorithms $\mathbf{M}$, there exist an algorithm $\mathbf{M}'$ and negligible function ϵ such that



Intuitively, this says that if the prover convinces the verifier that x is in L, then the prover knows how to generate a witness $y \in R_L(x)$ at the beginning of the protocol.

Theorem 3: The interactive proof system (P, V) for L is a proof of knowledge iff the system $\mathcal{P}^{pp} \times V$ is witness convincing for L.

- The runs of system $\mathcal{P}^{pp} \times V$ are all possible interactions of a verifier running V with a prover running some probabilistic polynomial time protocol.

14. Future Work: The Evolution of Belief

- Relation hiding restricts the verifier's knowledge at the beginning of the interaction (at time 0) about what he can do at some future time m^*.
- Intuitively, we would expect that the verifier does not learn something new at *any* point of zero-knowledge proof.
- This does not hold if we consider only objective probabilities on the verifier's possible worlds.
  - At the end of a run, either the verifier can generate a witness or not.
- Nevertheless, the verifier may have subjective uncertainty about whether he can generate a witness.
- However, subjective beliefs can be arbitrary.
  - What are appropriate constraints/axioms for how the verifier's subjective beliefs change during a ZK proof?