Zero-Knowledge Proofs

Material taken from here, here

Classical Proofs

- The notion of a proof is basic to mathematics
- Proof π is a static string that is written down somewhere and anyone can verify
- Valid proof gives absolute certainty that the statement is true

Redefining proofs

- Proof redefined as a game between a prover and a verifier
- Game can be **interactive**, where the verifier asks questions and the prover answers
- Further generalization to a **probabilistic** proof system
- “Prove that I could prove it if I felt like it”

SIAM J. COMPUT.
Vol. 18, No. 1, pp. 186–208, February 1989

THE KNOWLEDGE COMPLEXITY OF
INTERACTIVE PROOF SYSTEMS*

SHAFI GOLDWASSER†, SILVIO MICALI‡, AND CHARLES RACKOFF‡

Abstract. Usually, a proof of a theorem contains more knowledge than the mere fact that the theorem is true. For instance, to prove that a graph is Hamiltonian it suffices to exhibit a Hamiltonian tour in it; however, this seems to contain more knowledge than the single bit Hamiltonian/non-Hamiltonian.

In this paper a computational complexity theory of the “knowledge” contained in a proof is developed. Zero-knowledge proofs are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Examples of zero-knowledge proof systems are given for the languages of quadratic residuosity and quadratic nonresiduosity. These are the first examples of zero-knowledge proofs for languages not known to be efficiently recognizable.

Key words. cryptography, zero knowledge, interactive proofs, quadratic residues

AMS(MOS) subject classifications. 68Q15, 94A60

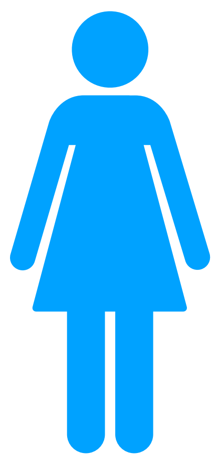
1. Introduction. It is often regarded that saying a language L is in NP (that is, acceptable in nondeterministic polynomial time) is equivalent to saying that there is a polynomial time “proof system” for L. The proof system we have in mind is one where on input x, a “prover” creates a string α, and 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. It is reasonable to ask if there is a more general, and perhaps more natural, notion of a polynomial time proof system. This paper proposes one such notion.

We will still allow the verifier only polynomial time and the prover arbitrary computing power, but will now allow both parties to flip unbiased coins. The result

Graph isomorphism

- Two graphs G_1 and G_2 are isomorphic if there exists a matching between their vertices so that two vertices are connected by an edge in G_1 if and only if corresponding vertices are connected by an edge in G_2
  - Assumption: graph isomorphism is “hard” to solve
- Alice is prover, Bob is verifier
- Alice proves to Bob that G_0 and G_1 are isomorphic
- Classic proof: Alice gives Bob the isomorphism
- Bob knows 1) G_0 and G_1 are isomorphic 2) the isomorphism

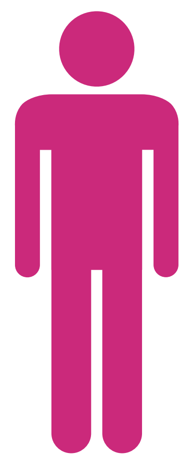
ZK graph isomorphism proof



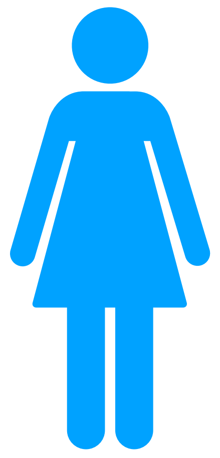
Alice produces a random graph H such that it is isomorphic to both G_0 and G_1

Proof: $H = \gamma_0(G_0), H = \gamma_1(G_1)$, thus $G_1 = \gamma_1^{-1}(\gamma_0(G_0))$ and $\sigma = \gamma_1^{-1}\gamma_0$

If Alice can show both isomorphisms, then there exists an isomorphism from G_0 to G_1



ZK graph isomorphism proof

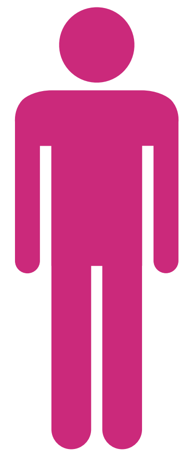


Send H

Send $b \stackrel{R}{\leftarrow} \{0,1\}$

If $b = 0$, send γ_0

If $b = 1$, send $\gamma_1 = \gamma_0 \sigma^{-1}$



Proof properties

- **Completeness:** a proof system is complete if you can prove all true statements using it
  - Previous scheme is complete as verifier will always accept if the prover is proving a true statement
- **Soundness:** a proof system is sound if you can never prove false statements using it
  - If prover is trying to prove a false statement, then the verifier will reject with overwhelming probability
  - Repeat k independent times gives $1 - 2^{-k}$ probability of catching a mistake

Proof properties

- **Zero-knowledge:** a cheating verifier “learns nothing” from the proof
- After an interactive proof, verifier knows
  - Whether the statement is true
  - A view of the interaction (transcript of messages + coins that the verifier tossed)
- The view gives the verifier nothing he couldn’t have obtained on his own

Zero knowledge

- If the verifier's view can be efficiently simulated so that "simulated views" and "real views" are indistinguishable
- Simulator does not take any private input from an honest party
- Simulator S:
  1. Toss coin c
  2. If $c = 0$, choose random γ_0, set $H = \gamma_0(G_0)$; if $c = 1$, choose random γ_1, set $H = \gamma_1(G_1)$
  3. Feed H to the verifier
  4. If verifier outputs $b = c$, then output (H, c, γ_c)
  5. Otherwise, rewind and go to step 1 again

Zero knowledge

- Simulator does not need to know σ
- If $b = c$, then the view of the cheating verifier & view of the simulator are the same: H is a random graph
- Efficient simulation
  - Since H is a random graph, c is independent of b
  - Probability that $b = c$ is $1/2$
  - Expected to halt after two attempts, so expected running time is polynomial
- Sequential composition ensures ZK is preserved over many iterations

Applications

- Maliciously secure MPC - enforce that a malicious party is following the protocol

- Identification scheme: prove identities without revealing

- Verifiable computation: how to verify outsourced (cloud) computation

- Exciting recent developments in zkSNARKs (zero-knowledge Succinct Non-interactive Arguments of Knowledge)

Today’s reading: AUDIT

Next time: guest lecture!

- Bryan Parno will talk about “An Early History of Verifiable Computation”