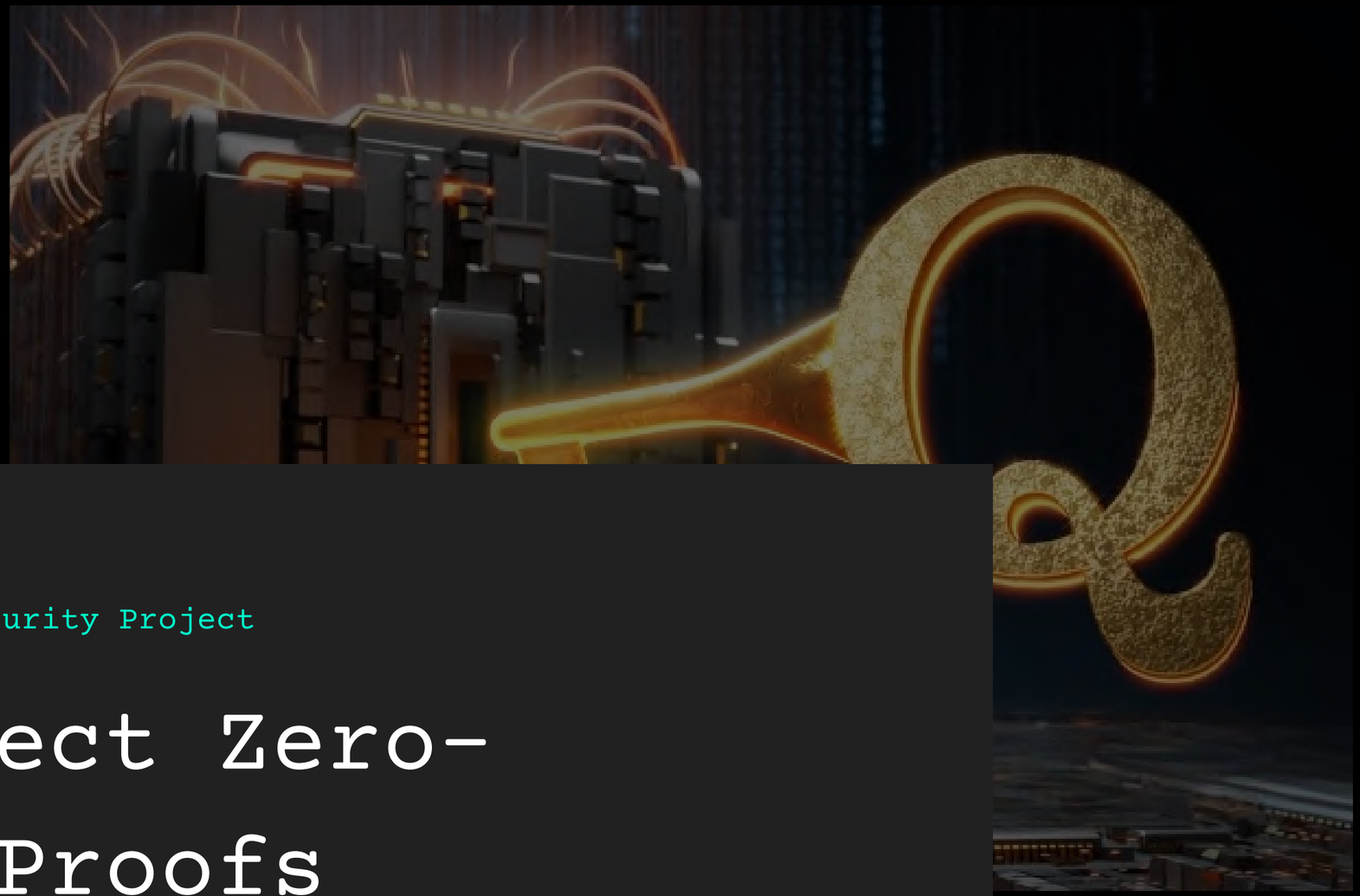

Principles of Information Security Project

Super-Perfect Zero-Knowledge Proofs

Akshit Gureja
Gaurav Singh
George Paul

Table of Contents

Super-Perfect Zero-Knowledge Proofs

Preliminaries

Things you need to know

What is a proof ?

A formal argument that demonstrates the correctness of a mathematical or computational system or program or algorithm. It essentially provides evidence to consider that a particular statement is true

The process by which the validity of an assertion is established

Prover and Verifier

Prover

- It is the party trying to prove a claim
- Goal to construct a proof that is convincing enough to persuade the verifier that the statement is true

Verifier

- Takes as input the statement and also the proof constructed by the prover
- Responsible for validating the assertion and truth of the statement.

The verifier is required to be a PPTM but no resource bounds are places on the computing power of the prover

Soundness and Completeness

Soundness

- Asserts that the verification procedure cannot be 'tricked' into accepting false statements.
- Captures the verifier's ability to protect itself from being convinced of false statements (irrespective of what the prover does in order to fool the verifier).
- Hence, if the statement is false, no prover, even if it doesn't follow the protocol should be able to convince to the honest verifier that it is true.

Completeness

- Captures the ability of some prover to convince the verifier of true statements (belonging to some predetermined set of true statements)
- Hence, if the statement is true, the honest prover can convince the honest verifier that it is indeed true

Interactive Proof Systems

Proof Systems are generally defined in terms of the verification procedure (which can be viewed as the verifier) and the proof, which is always considered as coming from outside (viewed as prover). The verification procedure itself doesn't generate proofs but merely verifies their validity.

Interactive Proof Systems

- Interactive proof systems are intended to capture whatever can be efficiently verified via interaction with the outside.
- The verifier's verdict of whether to accept or reject a claim is probabilistic and hence a bounding error probability is allowed.
- Loosely speaking, we require that the prover be able to convince the verifier of the validity of true statements, while nobody can fool the verifier into believing false statements.
- Both conditions are given a probabilistic interpretation: it is required that the verifier accepts valid statements with 'high' probability whereas the probability that it will accept a false statement is 'low.'

Interactive Proof Systems

The prover possesses unlimited computational resources but cannot be trusted, while the verifier has bounded computation power but is assumed to be always honest.

PPT Strategy

We say that a strategy is probabilistic polynomial time (PPT) if the total time spent when it interacts with any other strategy on common input x is in poly(|x|), where the total time accounts for all computations performed at all stages of the interaction (including the final generation of the output)

Definition 4.2.6 (Generalized Interactive Proof): *Let* $\mathsf{c}, \mathsf{s} : \mathbb{N} \to \mathbb{R}$ *be functions satisfying* $\mathsf{c}(n) > \mathsf{s}(n) + \frac{1}{p(n)}$ *for some polynomial* $p(\cdot)$. *An interactive pair* (P, V) *is called a (generalized) interactive proof system for the language* L, *with* **completeness bound** $\mathsf{c}(\cdot)$ *and* **soundness bound** $\mathsf{s}(\cdot)$, *if*

- (modified) completeness: *for every* $x \in L$,

$$\Pr[\langle P, V\rangle(x) = 1] \geq \mathsf{c}(|x|)$$

- (modified) soundness: *for every* $x \notin L$ *and every interactive machine* B,

$$\Pr[\langle B, V\rangle(x) = 1] \leq \mathsf{s}(|x|)$$

The function $\mathsf{g}(\cdot)$ *defined as* $\mathsf{g}(n) \stackrel{\text{def}}{=} \mathsf{c}(n) - \mathsf{s}(n)$ *is called the* acceptance gap *of* (P, V), *and the function* $\mathsf{e}(\cdot)$, *defined as* $\mathsf{e}(n) \stackrel{\text{def}}{=} \max\{1 - \mathsf{c}(n), \mathsf{s}(n)\}$, *is called the* error probability *of* (P, V). *In particular,* s *is the* soundness error *of* (P, V), *and* $1 - \mathsf{c}$ *is its* completeness error.

Interactive Proof Systems

Notation clatification

For randomized interactive Turing Machines A and B and supposing that all possible interactions on each common input terminate in a finite number of steps.

We denote <A,B>(x) as the (local) output of B after interacting with A on common input x, when the random input to each machine is uniformly and independently chosen. Since A and B are randomized, <A,B>(x) is a random variable.

Alternate way of defining interactive proof systems

Definition 2.1 (interactive proof systems, following Goldwasser, Micali and Rackoff [14]): *Let* $\epsilon_c, \epsilon_s : \mathbb{N} \to [0,1)$ *such that* $\epsilon_c(\ell)$ *and* $\epsilon_s(\ell)$ *are computable in* $\text{poly}(\ell)$*-time and* $\epsilon_c(\ell) + \epsilon_s(\ell) < 1 - 1/\text{poly}(\ell)$*. Let* P *and* V *be interactive strategy such that* V *is PPT. We say that* (P, V) *is an* interactive proof system for a set S with completeness error ϵ_c and soundness error ϵ_s *if the following two conditions hold.*

Completeness: *For every* $x \in S$*, it holds that* $\mathbf{Pr}[\langle P, V\rangle(x) = 1] \geq 1 - \epsilon_c(|x|)$.

Soundness: *For every* $x \notin S$ *and every interactive* P^**, it holds that* $\mathbf{Pr}[\langle P^*, V\rangle(x) = 1] \leq \epsilon_s(|x|)$.

If $\epsilon_c \equiv 0$*, then the system has* perfect completeness.

Zero-knowledge proofs

The original task

What is meant by zero-knowledge

Zero-knowledge refers to the fact that the verifier of the proof gains no knowledge beyond the fact that the statement being proven is true. Hence the **knowledge gain** is 0.

What is knowledge gain?

- Consider a case where Bob asks Alice whether or not a graph is Eulerian. Clearly, Bob gains no knowledge from Alice's answer, as Bob has the computational capability to determine the answer on his own.
- Now if Bob asks Alice if the graph is Hamiltonian and lets assume Alice is able to answer this question, then we say that Bob has gained knowledge from Alice as we don't know of an efficient procedure by which Bob could have determined this himself.

Thus, whenever Bob can efficiently compute the answer to a question *after interacting* with Alice, and he can also efficiently compute the answer by himself, we say he has *gained no knowledge.* Bob gains knowledge only if he receives the result of a computation that computationally infeasible for him

Zero-knowledge proof systems

- Zero-knowledge proofs are cryptographic protocols that enable one party, the prover, to prove to another party, the verifier, that a certain statement is true without revealing any additional information beyond the fact that the statement is indeed true. In other words, the verifier learns nothing other than the truth of the statement being proven.
- The zero-knowledge property requires that there exists a simulator, Sim, that can generate a "fake" proof that is indistinguishable from a real proof, meaning that the verifier cannot tell the difference between a real proof and a fake proof generated by the simulator.

Zero-knowledge proof systems

A zero-knowledge proof of some statement must satisfy three properties:

1. **Completeness**: if the statement is true, an honest verifier (that is, one following the protocol properly) will be convinced of this fact by an honest prover.
2. **Soundness**: if the statement is false, no cheating prover can convince an honest verifier that it is true, except with some small probability.
3. **Zero-knowledge**: if the statement is true, no verifier learns anything other than the fact that the statement is true. In other words, just knowing the statement (not the secret) is sufficient to imagine a scenario showing that the prover knows the secret. This is formalized by showing that every verifier has some *simulator* that, given only the statement to be proved (and no access to the prover), can produce a transcript that "looks like" an interaction between an honest prover and the verifier in question.

Zero-knowledge proof systems

They have the remarkable property of being convincing and yielding nothing beyond the validity of the assertion.

- The paper defines zero-knowledge proof systems by "the following oversimplified definition":
  An interactive proof system (P, V) for a set S is called zero-knowledge (ZK) if for every probabilistic polynomial-time strategy V ∗ there exists a (strict) probabilistic polynomial-time algorithm (called a simulator) A∗ such that A∗ (x) is distributed identically to the output of V ∗ after interacting with P on common input x.

Interactive and Non-interactive ZKPs

Interactive ZKP

An interactive proof system (P,V) for a set S is called zero-knowledge (ZK) if for every PPT strategy V* there exists a (strict) probabilistic PPT algorithm (called a simulator) A* such that A*(x) is distributed identically to the output of V* after interacting with P on common input x.

Non-interactive ZKP

Let $\epsilon_c, \epsilon_s : \mathbb{N} \to [0,1)$ such that $\epsilon_c(l)$ and $\epsilon_s(l)$ are computable in poly(l)-time and $\epsilon_c(l) + \epsilon_s(l) < 1 - 1/poly(l)$. Let P and V be algorithms such that V is a PPT. Let 'p' be a positive polynomial. We say that (P,V) is a non-interactive proof system for a set S with completeness error and soundness error being ϵ_c, ϵ_s respectively if the following 2 conditions hold -

- Completeness: For every $x \in S$, it holds that

$$Pr_{\omega \leftarrow U_{p(|x|)}}[V(x, \omega, P(x, \omega)) = 1] \geq 1 - \epsilon_c(|x|)$$

- Soundness: For every $x \notin S$, and every function P*, it holds that

$$Pr_{\omega \leftarrow U_{p(|x|)}}[V(x, \omega, P^*(x, \omega)) = 1] \leq \epsilon_s(|x|)$$

If $\epsilon_c \equiv 0$ then the system has perfect completeness

Perfect Zero-Knowledge Proofs

The precursor

Perfect Zero-Knowledge

Definition of Perfect-Zero Knowledge

Let (P, V) be an interactive proof system for some language. L. We say that. (P, V) is **perfect zero-knowledge** if for every PPT interactive machine. V^* there exists a PPTM M^* such that for every $x \in L$, the following two conditions hold:

1. With probability at most $\frac{1}{2}$, on input x, machine M^* outputs a special symbol $\perp$, that is:

$$Pr[M^*(x) = \perp] \leq \frac{1}{2}$$

2. Let $m^*(x)$ be a random variable describing the distribution of $M^*(x)$ conditioned on $M^*(x) \neq \perp$, that is:

$$Pr[m^*(x) = \alpha] = Pr[M*(x) = \alpha | M^*(x) \neq \perp, \forall \alpha \in \{0,1\}^*$$

Then the following random variables are identically distributed:

- $\langle P, V^* \rangle(x)$ (i.e., the output of the interactive machine V^* after interacting with the interactive machine P on a common input x)
- $m^*(x)$ (i.e., the output of machine M^* on input x, conditioned on not being $\perp$)

Machine M^* is called a **perfect simulator** of the interaction of V^* with P.

Unfortunately, the preceding formulation of perfect zero-knowledge is slightly too strict. Thus a relaxed formulation allows for the simulator to fail with bounded probability, to produce an interaction.

Interactive and Non-interactive PZKPs

Interactive PZKPs

Let (P,V) be an interactive proof system for S. The system (P,V) is perfect zero-knowledge if for every probabilistic polynomial-time strategy V* there exists a (strict) PPT algorithm A* such that for every x ∈ S it holds

$$Pr[A^*(x) = \perp] \leq 1/2$$

and

$$Pr[A^*(x) = \gamma | A^* \neq \perp] = Pr[< P, V^* > (x) = \gamma]$$

for every $\gamma \in \{0,1\}^*$

Non interactive PZKPs

The system (P,V) is perfect zero-knowledge if there exists a (strict) PPT algorithm A such that for every $x \in S$, it holds that

$$Pr[A(x) = \perp] \leq 1/2$$

and

$$Pr[A(x) = \gamma | A \neq \perp] = Pr_{\omega \leftarrow U_{p(|x|)}}[(\omega, P(x, \omega)) = \gamma]$$

for every $\gamma \in \{0,1\}^*$

Super-Perfect Zero-Knowledge Proofs

The original task

Interactive and Non-interactive SPZKPs

Interactive PZKPs

The system (P, V) is super perfect zero-knowledge if for every probabilistic polynomial time strategy V* there exists a (strict) probabilistic polynomial-time algorithm A* such that for every $x \in S$, it holds that A*(x) is distributed identically to <P, V*>(x)

Non interactive PZKPs

The system (P,V) is super perfect zero-knowledge if there exists a (strict) PPT algorithm A such that for every $x \in S$, it holds that A(x) is distributed identically to $(\omega, P(x, \omega))$ where $\omega \leftarrow Up(|x|)$.

From Perfect ZK to Super-Perfect ZK

Yielding super-perfect zero-knowledge proofs of exponentially vanishing completeness error

Theorem 1

Suppose (P,V) is an interactive proof system for S and there exists a function p:S→ [0,0.5] such that for every probabilistic polynomial time strategy V* there exists a probabilistic polynomial time algorithm A* such that for every x ∈ S, it holds that -

$$Pr[A^*(x) = \perp] \leq p(x)$$

and

$$Pr[A^*(x) = \gamma | A^*(x) \neq \perp] = Pr[< P, V^*(x) = \gamma]$$

for every $\gamma \in \{0,1\}^*$

Then S has a super perfect zero-knowledge proof system. Furthermore -

- The soundness error is preserved, and the increase in the completeness error is exponentially vanishing
- black-box simulation is preserved
- The communication complexities (number of rounds and length of messages) are preserved
- The new prover strategy can be implemented by a PPT oracle machine that is given access to the original prover strategy.

It is proved by observing the transformation proposed by Malka, which applies whenever all simulators fil with the same probability

Proof

Proposed Construction

Construction 3.1 (the transformation): *Let (P, V), S and p be as in the hypothesis of Theorem 1, and suppose that A' is a simulator for any fixed PPT strategy V'* (e.g., V' may equal V or V_{hon}). *On common input x, the two parties proceed as follows.*

- *The prover invokes $A'(x)$ and sends the empty message λ if and only if $A'(x) = \perp$. In such a case, the verifier will reject.*
- *Otherwise, the parties execute (P, V) on the common input x.*

For every PPT strategy V^*, consider the simulator A^* guaranteed by the hypothesis of Theorem 1. On input x, the corresponding new simulator (for V^*) *computes $\gamma \leftarrow A^*(x)$, outputs γ if $\gamma \neq \perp$ and $V^*(x, \lambda)$ otherwise.*

Here, w.l.o.g. we assume that p(x) is a negligible function for any x ∈ S. The transformation lets the prover perform the original protocol with probability 1-p(x) and abort otherwise. The verifier will reject in case the prover aborts and so perfect completeness is lost. This transformation relies in the assumption that all simulators output $\perp$ with the same probability. For sake of simplicity, we also assume w.l.o.g. that the original prover never sends the empty string denoted by **λ**

Proof

As in this transformation, we haven't modified anything on the verifier, the protocol would preserver the soundness error of V.

The completeness error on input $x \in S$ increases by at-most $p(x)$. The verifier rejects when the prover aborts which occurs with a probability $p(x)$, hence the interaction error with the original completeness bound takes place with probability $1-p(x)$. Hence, the completeness error changes from $\varepsilon(|x|)$ to $p(x) + (1-p(x)).\varepsilon(|x|)$

As the verifier rejects whenever the prover gets unlucky, a cheating prover gains nothing by claiming that it got $\perp$. The new simulators establish SPZK feature and hence, theorem 1 follows.

Corollary:

As the conditions posed in Theorem 1 are for nothing but a honest-verifier ZK, the proof vacuously holds for honest-verifier perfect ZK proof systems. Hence, every set S that has an honest-verifier perfect ZK proof system has an honest-verifier SPZK proof system.

From Perfect ZK to Super-Perfect ZK

On super-perfect ZK arguments with perfect completeness

Theorem 1

Super-perfect zero-knowledge arguments with perfect completeness while assuming

Assumption: the existence of perfectly binding commitment schemes.

Definition: an admissible class of perfect ZK protocols

Let (P,V) be an argument system for S, let P0 denote the strategy derived from P by having it abort just before sending the last message. We say that P is admissible if for every PPT strategy V* there exists a (strict) PPT algorithm A* such that for every $x \in S$ the following 3 conditions hold -

1. $Pr[A^*(x) = (1, .)] = Pr[A^*(x) = (0, .)] = 1/2$
2. $Pr[A^*(x) = (1, .) | A^*(x) = (1, .)] = Pr[< P, V^* > (x) = \gamma]$ for every $\gamma \in \{0,1\}^*$
3. $Pr[A^*(x) = (0, .) | A^*(x) = (0, .)] = Pr[< P_0, V^* > (x) = \gamma]$ for every $\gamma \in \{0,1\}^*$

In addition to requiring A* to output <P0,V(>(x) whenever it fails (Condition 3) we also require the failure probability to be exactly equal to one half. The latter condition can be assumed WLOG whenever p is efficiently computable provided that we adopt the non-standard model of PPT machines in which a machine can sample [n] uniformly at the cost n.

Theorem 1

Super-perfect zero-knowledge arguments with perfect completeness while assuming

Assumption: the existence of perfectly binding commitment schemes.

Construction (the transformation):

1. *The parties execute* (P_0, V) *on common input* x*; that is, they invoke the original protocol, except that the prover does not send its last message, denoted* β.
2. *The parties performs a standard coin tossing protocol* (see [11, Sec. 7.4.3.1]). *Specifically, the verifier sends a commitment* $c \leftarrow C(v)$ *to a random bit* v*, the prover responds* (in the clear) *with a random bit* u*, and the verifier de-commits to the commitment* (i.e., provides (v, s) such that $c = C_s(v)$).
3. *If the verifier has de-committed improperly* (i.e., $c \neq C_s(v)$)*, then the prover sends the empty message, denoted* λ*. Otherwise, if* $u = v$ *then the prover sends* 0*, and otherwise it sends* β (where we assume, w.l.o.g, that $\beta \notin \{0, \lambda\}$).
4. *If* $u = v$ *then the verifier accepts, otherwise* (i.e., $u \neq v$) *it acts as* $V(\alpha, \beta)$*, where* α *denotes the view of* V *in the interaction with* P_0 (as conducted in Step 1).

Theorem 1

Super-perfect zero-knowledge arguments with perfect completeness while assuming

Assumption: the existence of perfectly binding commitment schemes.

- This transformation preserves the completeness error of (P, V), but the computational-soundness error grows from $\epsilon(\ell)$ to $(1+\epsilon(\ell)+\mu(\ell))/2$, where μ is a negligible function.
- Indeed, computational soundness is established by observing that the prover can cause the verifier to ignore the transcript (α, β) only if it guessed correctly the value committed by the verifier (i.e., if $u = v$).
- Assuming that $V*$ always de-commits properly, the super-perfect ZK feature is based on the simulator's ability to set $u = v$ whenever it fails to produce a full transcript.

From perfect NIZK to super-perfect NIZK

Making it non-interactive

> … this can be done by “transferring” the simulation attempt from the prover (who cannot be trusted to perform it at random) to the common reference string (which is uniformly distributed by definition).

Goldreich and Teichner

From the paper under review

The Construction

Common Random String

An $(\rho(\ell) + \rho'(\ell))$-bit string, denoted (ω,r), where r is interpreted as an integer in $\{0,\ldots,2\ \rho'(\ell) - 1\}$

Prover

(on input $x \in \{0,1\}^{\ell}$): If $r < p(x) \cdot 2\ \rho'(\ell)$, then the prover outputs the empty message λ. Otherwise (i.e., $r \geq p(x) \cdot 2\ \rho'(\ell)$), the prover outputs $P(x,\omega)$

Verifier

(on input $x \in \{0,1\}^{\ell}$ and alleged proof y): If $r < p(x) \cdot 2\rho'(\ell)$, then the verifier accepts. Otherwise (i.e., $r \geq p(x) \cdot 2\rho'(\ell)$), the verifier decides according to $V(x,\omega,y)$.

The Simulator

1. **Invoke A(x)**
   The new simulator invokes A(x), an algorithm that simulates (P,V), obtaining the value v.

2. **If v = ⊥**
   Then the simulator selects uniformly $\omega \in \{0,1\}$ $\rho(\ell)$ and $r \in \{0,\ldots,p(x) \cdot 2\rho'(\ell) - 1\}$, and outputs $((\omega,r),\lambda)$.

3. **Else if v != ⊥**
   Otherwise (i.e., $v = (\omega,y)$), the simulator selects uniformly $r \in \{p(x) \cdot 2\rho'(\ell) ,\ldots,2\,\rho\,'\,(\ell) - 1\}$, and outputs $((\omega,r),y)$.

Aha!

Recall that the construction of the new simulator relies on the ability to generate uniform distributions on the sets $[p(x) \cdot 2\rho'(\ell)]$ and $[2\rho'(\ell) - p(x) \cdot 2\rho'(\ell)]$

However, if we start with $p \equiv 1/2$

However, if we start with $p \equiv 1/2$

Corollary

Then, S has a super-perfect non-interactive zero-knowledge proof system, where simulation is in the standard PPT model

A super-perfect NIZK for Blum Integers

Putting it into practice

Constructing the Proof System

Input	Common reference string	Prover
A natural number n. Let $\ell = \lceil \log_2(n) \rceil$.	An ℓ-bit string, denoted ω, interpreted as an integer in the group $Z_{2\ell}$	If $\omega \in S_n$ and there exists $r \in S_n$ such that fn (r) = ω, then the prover outputs r (otherwise it outputs 0).

The Verifier

A part of the Proof System

1. If n is a prime power or is not $n \equiv 1 \pmod 4$, then the verifier halts outputting 0 (indicating reject)

2. The verifier, the basically checks in poly time whether n is a Blum Integer within the group $Z_{2\ell}$.

3. If not $\omega \in S_n$, then the verifier halts outputting 1 (indicating accept).

4. If $r \in S_n$ and $fn(r) = \omega$, then the verifier outputs 1. Otherwise, it outputs 0

The Proof in short

for $n \in B$ and $\omega \in S_n$ ⟹ there exists a unique $r \in S_n$ such that $fn(r) = \omega$.

The super-perfect simulation proceeds as follows:

1. Select uniformly $r \in Z_{2\ell}$

2. If $r \in S\ n$ then output $(fn(r), r)$, else output $(r,0)$

Note that the simulator's output is distributed identically to the distribution produced by the prover

Soundness error = 16/17

Who needs knowledge?
Thanks!

Brought to you by
Proofs and Consequences

Akshit Gureja
Gaurav Singh
George Paul