CIS 5560

Cryptography
Lecture 20

Course website:
pratyushmishra.com/classes/cis-5560-s24/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW8 due tomorrow evening**
- **HW 9 out Wednesday evening**
  - Due **Wednesday Apr 17** at 11:59PM on Gradescope
  - Covers
    - One-time signatures
    - RSA-based signatures

Recap of last lecture

New primitive: Digital Signatures

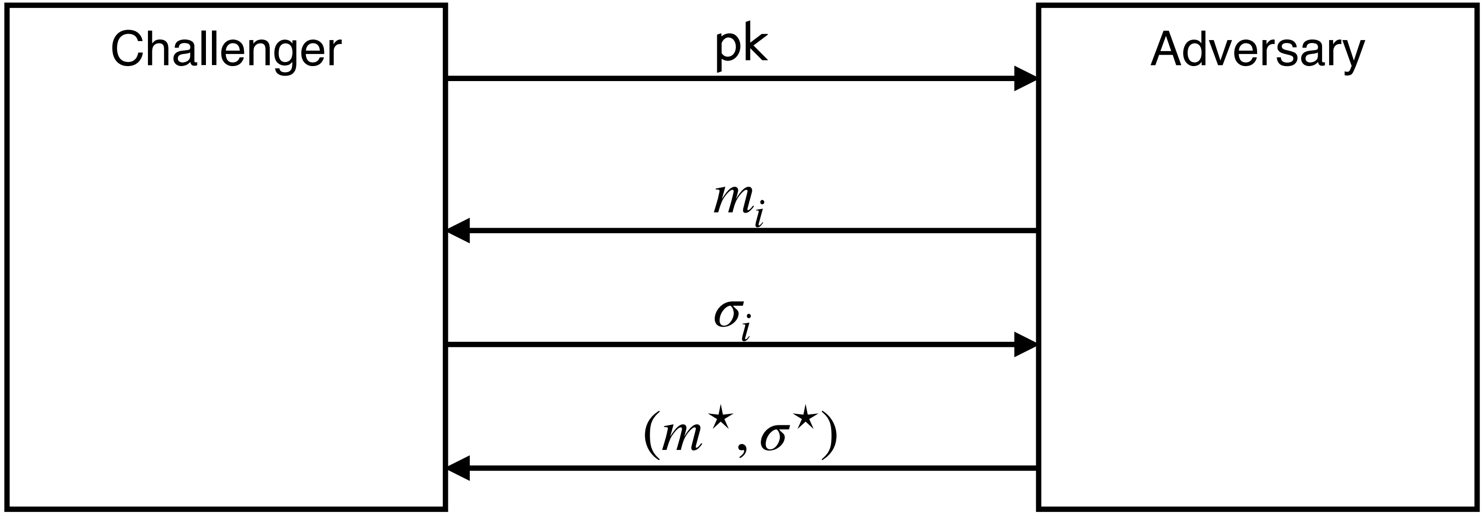
Digital Signatures: Definition

A triple of PPT algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ such that

- Key generation: $\mathsf{Gen}(1^n) \to (\mathsf{sk}, \mathsf{pk})$
- Message signing: $\mathsf{Sign}(\mathsf{sk}, m) \to \sigma$
- Signature verification: $\mathsf{Verify}(\mathsf{pk}, m, \sigma) \to b \in \{0,1\}$

Correctness: For all vk, sk, m: $\mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1$

EUF-CMA for Signatures



$$\Pr\left[\begin{array}{c} m^\star \notin \{m_i\} \\ \text{and} \\ \mathsf{Verify}(\mathsf{pk}, m^\star, \sigma^\star) = 1 \end{array}\right] = \mathsf{negl}(\lambda)$$

Lamport (One-time) Signatures for arbitrary bits

Secret Key sk: $\begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{n,0} \\ x_{1,1} & x_{1,1} & \cdots & x_{n,1} \end{pmatrix}$

Public Key pk: $\begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{n,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{n,1} \end{pmatrix}$ where $y_{i,b} = f(x_{i,b})$.

Signing m:

1. $z := H(m)$
2. $\sigma = (x_{1,z_1}, x_{2,z_2}, \ldots, x_{n,z_n})$

Claim: Assuming H is CRH and f is a OWF, no PPT adv can produce a signature of m given a signature of a single $m' \neq m$.

Claim: Can forge signature on any message given the signatures on (some) two messages.

(Many-time) Signature Scheme

In four+ steps

Step 1. Stateful, Growing Signatures. Idea: Signature ***Chains***

Step 2. How to Shrink the signatures. Idea: Signature ***Trees***

Step 3. How to Shrink Alice's storage.
Idea: ***Pseudorandom Trees***

Step 4. How to make Alice stateless.
Idea: ***Randomization***

Step 5 (*optional*). How to make Alice stateless and deterministic. Idea: ***PRFs.***

How to Fix Vanilla RSA

Start with any trapdoor permutation, e.g. RSA.

Gen(1^λ): Pick primes (P, Q) and let $N = PQ$. Pick e relatively prime to $\varphi(N)$ and let $d = e^{-1} \pmod{\varphi(N)}$.

sk = (N, d) and pk = $(N, e, \boldsymbol{H})$

Sign(sk, m): Output signature $\sigma = \boldsymbol{H}(\boldsymbol{m})^d \pmod{N}$.

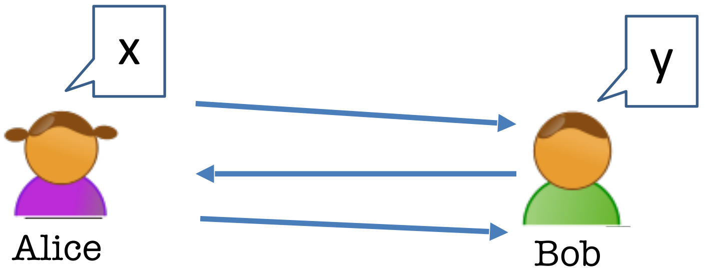
Verify(vk, m, σ): Check if $\sigma^e = \boldsymbol{H}(\boldsymbol{m}) \pmod{N}$.

H is a *random oracle.*

Today’s lecture

- What is a proof?
- Interactive Proofs
- *Zero-knowledge* interactive proofs
-

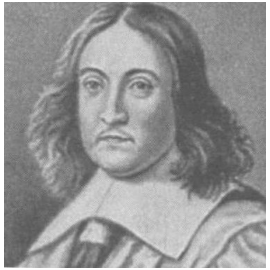
Beyond Secure Communication



Much more than communicating securely.

- Complex Interactions: proofs, computations, games.
- Complex Adversaries: Alice or Bob, adaptively chosen.
- Complex Properties: Correctness, Privacy, Fairness.
- Many Parties: this class, MIT, the internet.

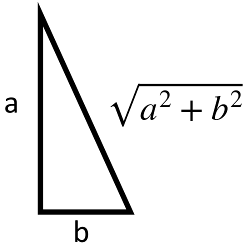
Classical Proofs

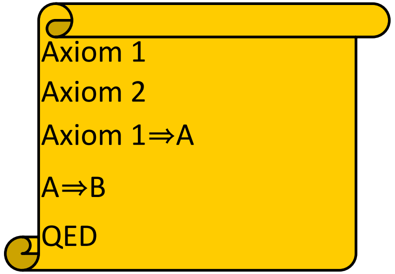


Steve Cook

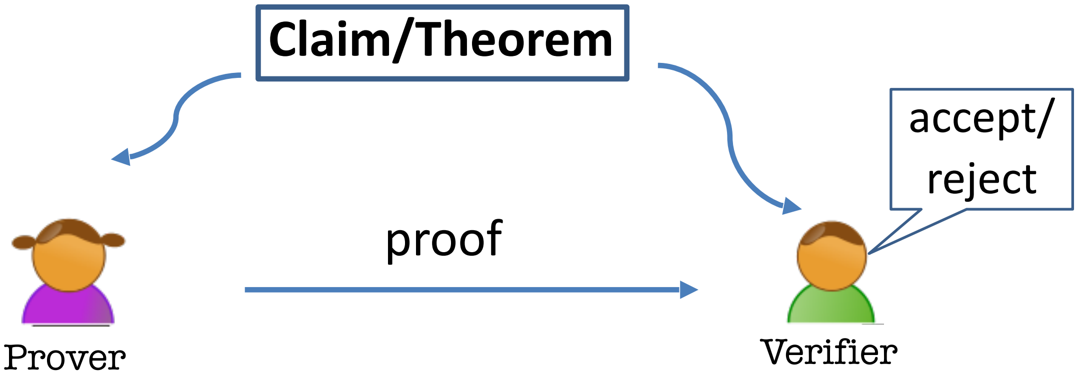
Leonid Levin

Prover writes down a string (proof); Verifier checks.

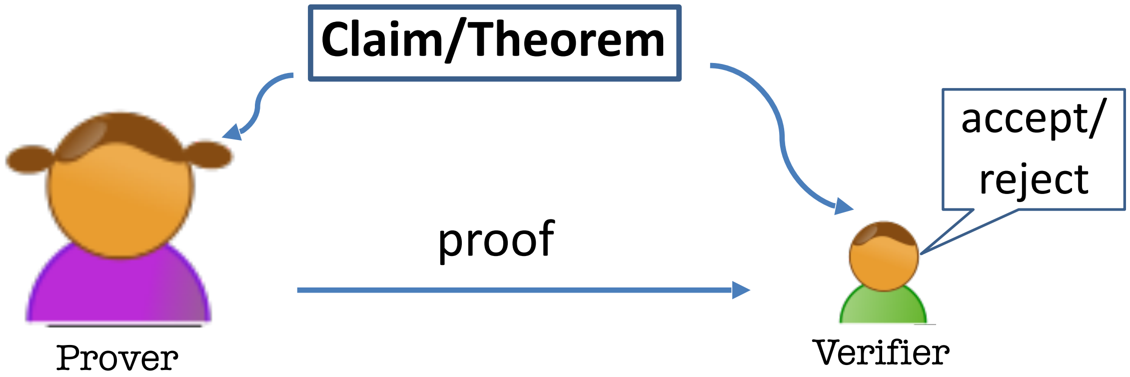




Proofs



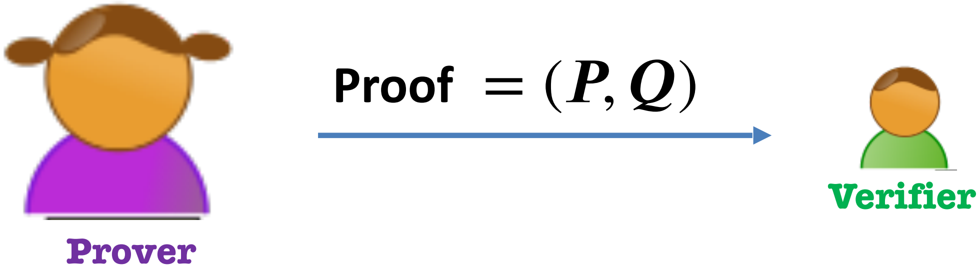
Efficiently Verifiable Proofs: NP



Works hard

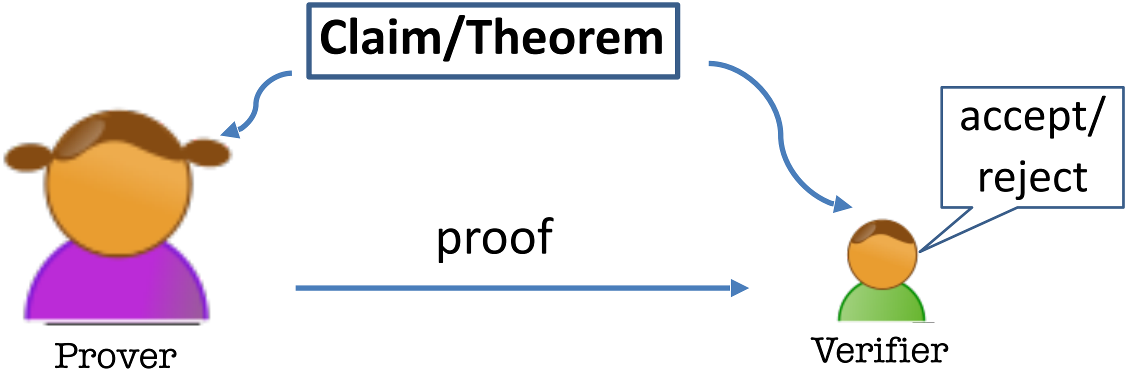
Polynomial-time

Theorem: N is a product of two prime numbers



Accept *iff* $N = PQ$
and P, Q are prime

Efficiently Verifiable Proofs: NP

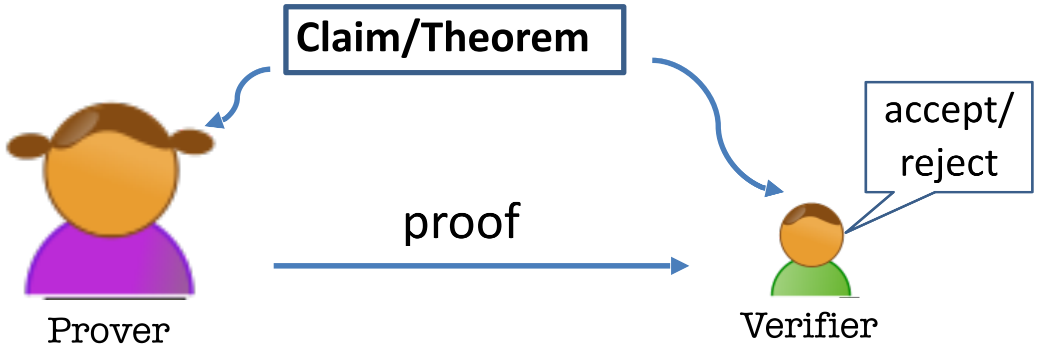


Works hard

Polynomial-time

<u>Def</u>: A language/decision procedure $\underline{\mathscr{L}}$ is simply a set of strings. So, $\underline{\mathscr{L} \subseteq \{0,1\}^*}$.

Efficiently Verifiable Proofs: NP



Def: $\mathscr{L}$ is an NP-language if there is a **poly-time** verifier V where

- **Completeness: True theorems have (short) proofs.**
  for all $x \in \mathscr{L}$, there is a **poly($|x|$)-long** witness (proof) $w \in \{0,1\}^*$ s.t. $V(x, w) = 1$.
- **Soundness: False theorems have no short proofs.**
  for all $x \notin \mathscr{L}$, there is no witness.
  That is, for all polynomially long w, $V(x, w) = 0$.

Theorem: N is a product of two prime numbers

Prover

Proof $= (P, Q)$ $\longrightarrow$

Verifier

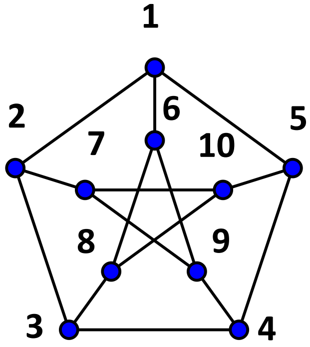
Accept *iff* $N = PQ$
and P, Q are prime

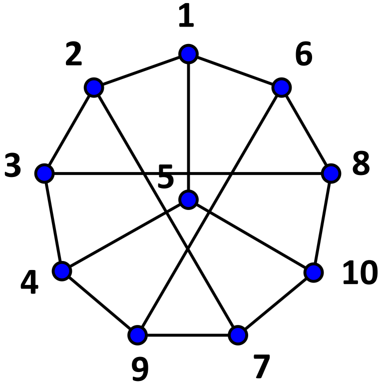
After interaction, the Verifier knows:

1) N is a product of two primes.

2) Also, the two factors of N.

Theorem: Graphs G_0 and G_1 are isomorphic.





Prover

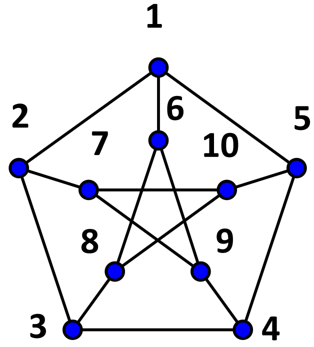
Proof $\pi : [N] \to [N]$,

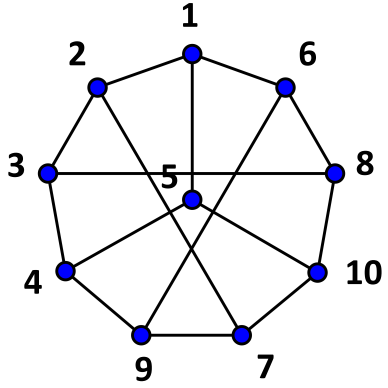
the isomorphism

Verifier

Check $\forall i, j$:
$(\pi(i), \pi(j)) \in E_1$
iff $(i, j) \in E_0$.

Theorem: Graphs G_0 and G_1 are isomorphic.





Prover

Proof $\pi : [N] \rightarrow [N]$**,**

the isomorphism

Verifier

After interaction, Bob the Verifier knows:

1) G_0 and G_1 are isomorphic.

2) **Also**, the isomorphism.

Check $\forall i, j$:
$(\pi(i), \pi(j)) \in E_1$
iff $(i, j) \in E_0$.

Theorem: Boolean Formula φ is satisfiable

$$\phi(X_1, \ldots, X_N) := (X_1 \vee X_3 \vee X_N) \wedge \cdots \wedge (X_5 \vee X_{N-5} \vee X_{10})$$

Prover

Proof = Satisfying assignment

$(x_0, \ldots, x_n)$

Verifier

Check $\varphi(x_1, \ldots, x_n) = 1$

After interaction, Bob the Verifier knows:

1) φ is satisfiable

2) **Also**, the satisfying assignment

Theorem: Boolean Formula φ is satisfiable

$$\phi(X_1, \ldots, X_N) := (X_1 \vee X_3 \vee X_N) \wedge \cdots \wedge (X_5 \vee X_{N-5} \vee X_{10})$$

Prover

Proof = Satisfying assignment

$(x_0, \ldots, x_n)$

Verifier

Check $\varphi(x_1, \ldots, x_n) = 1$

NP-Complete **Problem:**

Every one of the other problems can be reduced to it

Is there any other way?

Zero Knowledge Proofs

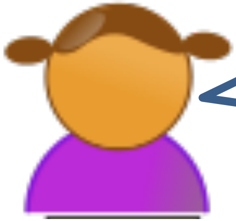
Prover

"I will prove to you that I could've sent you a proof if I felt like it."

Micali Goldwasser Rackoff

Zero Knowledge Proofs

“I will not give you the isomorphism, but will prove to you that I could have one.”



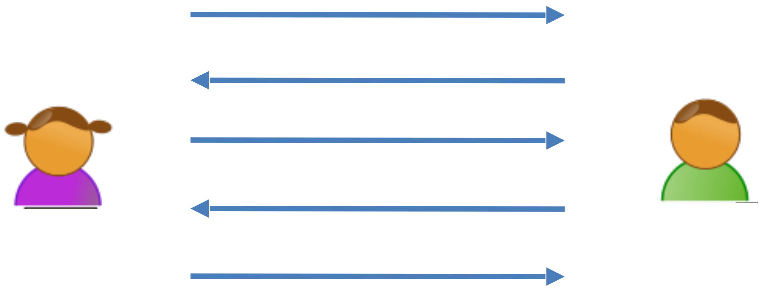
Prover

Micali Goldwasser Rackoff

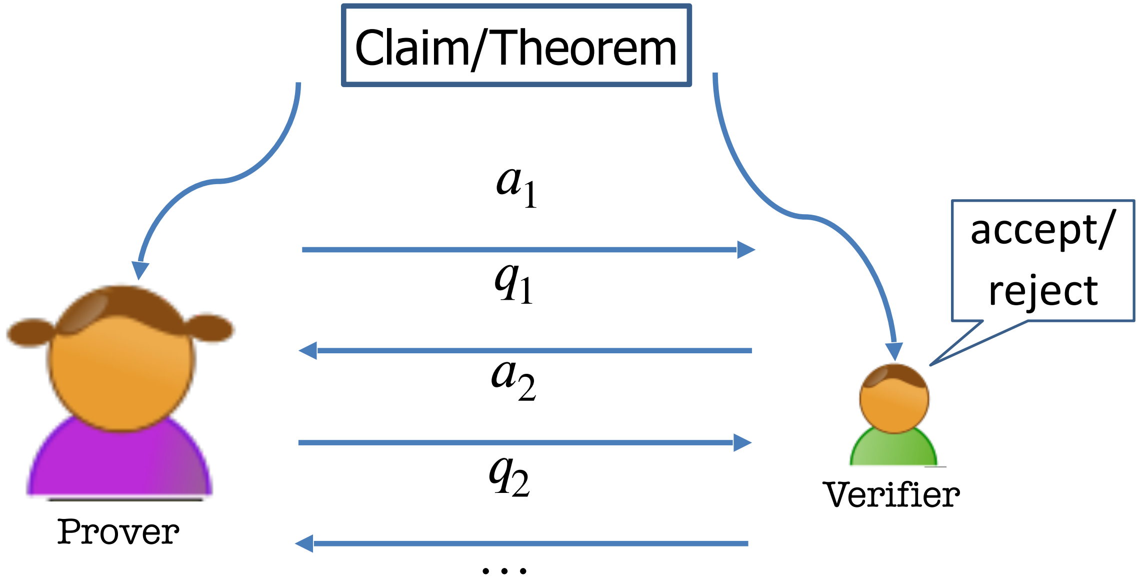
Two (Necessary) New Ingredients

1. **Interaction:** Rather than passively reading the proof, the verifier engages in a conversation with the prover.

2. **Randomness:** The verifier is randomized and can make a mistake with a (exponentially small) probability.



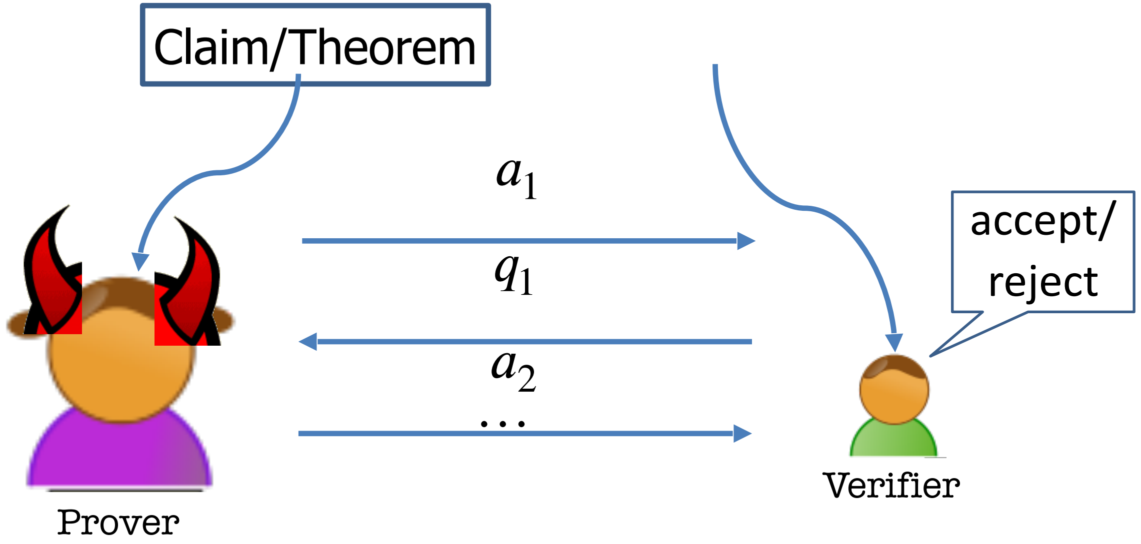
Interactive Proofs for a Language $\mathscr{L}$



Comp. Unbounded

Probabilistic
Polynomial-time

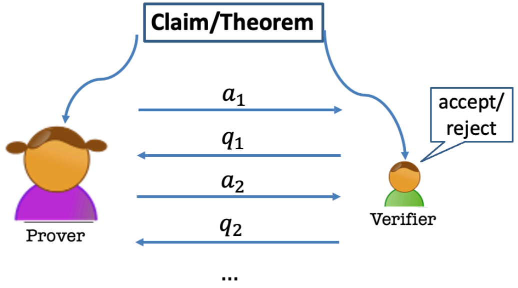
Interactive Proofs for a Language $\mathscr{L}$



Def: $\mathscr{L}$ is an IP-language if there is a unbounded P and **probabilistic poly-time** verifier V where

- **Completeness**: If $x \in \mathscr{L}$, V always accepts.
- **Soundness:** If $x \notin \mathscr{L}$, regardless of the cheating prover strategy, V accepts with negligible probability.

Interactive Proofs for a Language $\mathscr{L}$



Def: $\mathscr{L}$ is an IP-language if there is a **probabilistic poly-time** verifier V where

- **Completeness: If** $x \in \mathscr{L}$**,**

$$\Pr\left[(\mathrm{P}, V)(x) = accept\right] = 1.$$

- **Soundness: If** $x \notin \mathscr{L}$, **there is a negligible function** negl **s.t. for every** P^*,

$$\Pr\left[\left(P^*, V\right)(x) = accept\right] = \mathrm{negl}(\lambda).$$

Interactive Proof for QR

$$\mathscr{L} = \{(N, y) : y \text{ is a quadratic residue mod } N\}.$$

$s = r^2 \pmod N$

(N, y)

$b \leftarrow \{0,1\}$

(N, y)

If b=0: $z = r$

If b=1:

$z = rx$

$$\boxed{\text{Check: } z^2 = sy^b \pmod N}$$

Completeness

Claim: If $(N, y) \in L,$ then the verifier accepts the proof with probability 1.

Proof:

$$z^2 = (rx^b)^2 = r^2(x^2)^b = sy^b \pmod{N}$$

So, the verifier's check passes and he accepts.

Soundness

Claim: If $(N, y) \notin L$, then for every cheating prover P^*, the verifier accepts with probability at most 1/2.

Proof: Suppose the verifier accepts with probability > 1/2.

Then, there is some $s \in Z_N^*$ s.t. the prover produces

$$z_0 : z_0^2 = s \pmod N$$

$$z_1 : z_1^2 = sy \pmod N$$

This means $(z_1/z_0)^2 = y \pmod N$, which tells us that $(N, y) \in L$.

Interactive Proof for QR

$$\mathscr{L} = \{(N, y) : y \text{ is a quadratic residue mod } N\}.$$

(N, y)

$s_i = r_i^2 \pmod{N}$

$b_i \leftarrow \{0,1\}$

(N, y)

If b_i=0:

$z_i = r_i$

If b_i=1:

$z_i = xr_i$

Check for all i:

$z_i^2 = s_i y^b \pmod{N}$

REPEAT sequentially λ times.

Soundness

Claim: If $(N, y) \notin L$, then for every cheating prover P^*, the verifier accepts with probability at most $(\frac{1}{2})^{\lambda}$.

Proof: Exercise.

This is Zero-Knowledge.

But what does that mean?

(N, y)

$$s = r^2 \pmod{N}$$

$$b \leftarrow \{0,1\}$$

If b=0: $z = r$

If b=1:

$$z = rx$$

(N, y)

Check:

$$z^2 = sy^b \pmod{N}$$

How to Define Zero-Knowledge?

After the interaction, *V* knows:

- The theorem is true; and
- A **view** of the interaction
  (= transcript + randomness of V)

***P* gives zero knowledge to *V*:**

When the theorem is true, the view gives V nothing that he couldn’t have obtained on his own without interacting with P.

How to Define Zero-Knowledge?

(P, V) is zero-knowledge if V can generate his **VIEW** of the interaction **all by himself** in **probabilistic polynomial time**.

How to Define Zero-Knowledge?

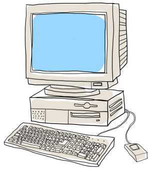
(P, V) is zero-knowledge if V can "simulate" his **VIEW** of the interaction **all by himself** in **probabilistic polynomial time**.

The Simulation Paradigm

n_S:
$b, z)$

$view_V(P, V)$:
Transcript = (s, b, z),
Coins = b

PPT "simulator" S



(N, y)

$s = r^2 \pmod N$ →

$b \leftarrow \{0,1\}$ ←

(N, y)

If b=0: $z = r$ →
If b=1: $z = rx$

Check:
$z^2 = sy^b \pmod N$

Zero Knowledge: Definition

An Interactive Protocol (P,V) is zero-knowledge for a language L if there exists a **PPT** algorithm S (a simulator) such that **for every** $x \in L$, the following two distributions are indistinguishable:

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

(P,V) is a zero-knowledge interactive protocol if it is complete, sound and zero-knowledge.

Perfect Zero Knowledge: Definition

An Interactive Protocol (P,V) is **perfect zero-knowledge** for a language L if there exists a PPT algorithm S (a simulator) such that for every $x \in L$, the following two distributions are **identical**:

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

(P,V) is a zero-knowledge interactive protocol if it is complete, sound and zero-knowledge.

Statistical Zero Knowledge: Definition

An Interactive Protocol (P,V) is **statistical zero-knowledge** for a language L if there exists a PPT algorithm S (a simulator) such that for every $x \in L$, the following two distributions are **statistically indistinguishable**:

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

(P,V) is a zero-knowledge interactive protocol if it is complete, sound and zero-knowledge.

Computational Zero Knowledge: Definition

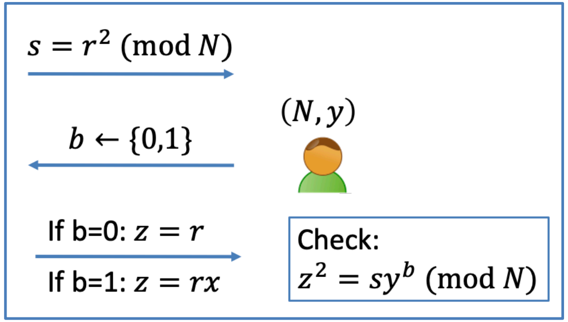
An Interactive Protocol (P,V) is **computational zero-knowledge** for a language L if there exists a PPT algorithm S (a simulator) such that for every $x \in L$, the following two distributions are **computationally indistinguishable**:

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

(P,V) is a zero-knowledge interactive protocol if it is complete, sound and zero-knowledge.

Zero Knowledge

Claim: The QR protocol is zero knowledge.



$view_V(P, V)$:
(s, b, z)

Simulator S works as follows:

1. First pick a random bit b.
2. pick a random $z \in Z_N^*$.
3. compute $s = z^2 / y^b$.
4. output (s, b, z).

Exercise: The simulated transcript is identically distributed as the real transcript in the interaction (P,V).

What if V is NOT HONEST.

OLD DEF

An Interactive Protocol (P,V) is **honest-verifier** perfect zero-knowledge for a language L if there exists a PPT simulator S such that for every $x \in L$, the following two distributions are identical:

$view_V(P, V)$ 2. $S(x, 1^\lambda)$

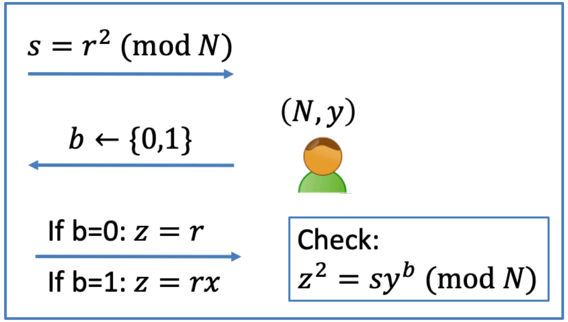
REAL DEF

An Interactive Protocol (P,V) is **perfect zero-knowledge** for a language L if **for every PPT** V^*, there exists a (expected) poly time simulator S s.t. for every $x \in L$, the following two distributions are identical:

1. $view_{V^*}(P, V^*)$ 2. $S(x, 1^\lambda)$

NOW: (Malicious Ver) Zero Knowledge

Theorem: The QR protocol is (malicious verifier) zero knowledge.



$$view_{V^*}(P, V^*):$$
$$(s, b, z)$$

Simulator S works as follows:

1. First pick a random s and "feed it to" V^*.

2. Let $\mathrm{b} = V^*(s)$.

Now what???

(Malicious Ver) Zero Knowledge

Theorem: The QR protocol is (malicious verifier) zero knowledge.

Simulator S works as follows:

1. First set $s = \frac{z^2}{y^b}$ for a random z and b and feed s to V^*.
2. Let $b' = V^*(s)$.
3. If $b' = b$, output (s, b, z) and stop.
4. Otherwise, go back to step 1 and repeat. (also called "rewinding").

Simulator S works as follows:

1. First set $s = \frac{z^2}{y^b}$ for a random z and feed s to V^*.
2. Let $\mathrm{b}' = V^*(s)$.
3. If $b' = b$, output $(\mathrm{s}, \mathrm{b}, \mathrm{z})$ and stop.
4. Otherwise, go back to step 1 and repeat. (also called "rewinding").

Lemma:
(1) S runs in expected polynomial-time.
(2) When S outputs a view, it is identically distributed to the view of V^* in a real execution.

What Made it Possible?

1. **Each statement had multiple proofs** of which the prover chooses one at random.

2. **Each such proof is made of two parts**: seeing either one on its own gives the verifier no knowledge; seeing both imply 100% correctness.

3. **Verifier chooses to see either part, at random**.
The prover's ability to provide either part on demand convinces the verifier.