CIS 5560

Cryptography
Lecture 20

Course website:
pratyushmishra.com/classes/cis-5560-s25/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Recap of last lecture

New primitive: Digital Signatures

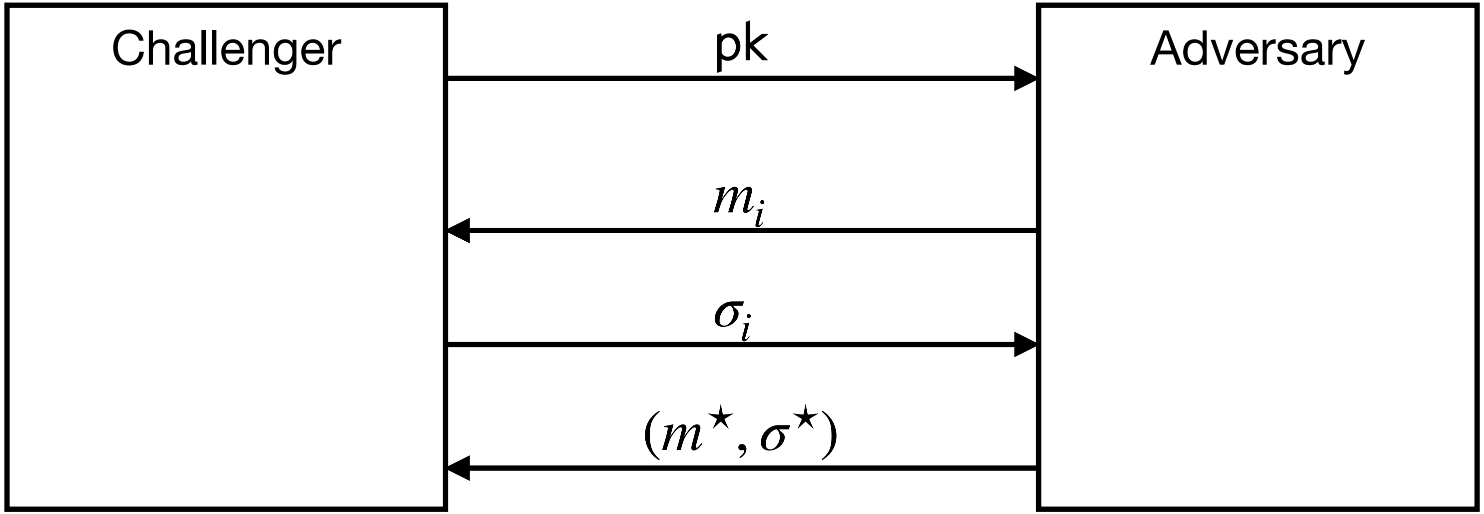
Digital Signatures: Definition

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 $\underline{f}$ is a OWF, no PPT adv can produce a signature of $\underline{m}$ given a signature of a single $\underline{m' \neq m}$.

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.***

Today's lecture

- RSA Signatures
- Proof systems
  - What is a proof?
  - Interactive Proofs
  - *Zero-knowledge* interactive proofs

“Vanilla” RSA Signatures

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)

Sign(sk, m): Output signature $\sigma = m^d \left(\mathrm{mod}\ N\right)$.

Verify(pk, m, σ): Check if $\sigma^e = m \left(\mathrm{mod}\ N\right)$.

Problem: Existentially forgeable!

“Vanilla” RSA Signatures

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

Verify(pk, m, σ): Check if $\sigma^e = m \pmod N$.

Problem: Existentially forgeable!

Attack: Pick a random σ and output ($m = \sigma^e$, σ) as the forgery.

Problem: Malleable!

Attack: Given a signature of m, you can produce a signature of $2^e * m, 3^e * m, \ldots, m^2, m^3, \ldots$

“Vanilla” RSA Signatures

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

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

Fundamental Issues:

1. Can ”reverse-engineer” the message starting from the signature (Attack 1)

2. Algebraic structure allows malleability (Attack 2)

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 $\sigma = \boldsymbol{H(m)}^d \pmod{N}$.

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

So, what is H? Some very complicated “hash” function.

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 $\sigma = \boldsymbol{H(m)}^d \pmod{N}$.

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

H should be at least one-way to prevent Attack #1.

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

Hard to “algebraically manipulate” H(m) into H(related m’).

(to prevent Attack #2.)

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 $d = e^{-1} \pmod{\varphi(N)}$.

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

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

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

Collision-resistance does not seem to be enough. (Given a CRHF h(m), you may be able to produce h(m’) for related m’.)

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 $d = e^{-1} \pmod{\varphi(N)}$.

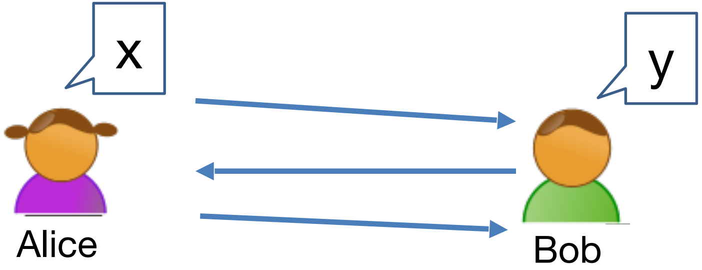
sk = (N, d) and pk = $(N, e, \boldsymbol{H})$

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

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

Collision-resistance does not seem to be enough. (Given a CRHF h(m), you may be able to produce H(m') for related m'.)

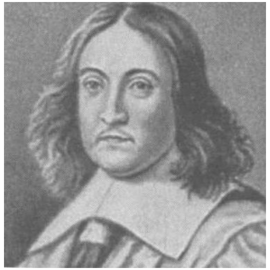
Beyond Secure Communication



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, Penn students, the internet.

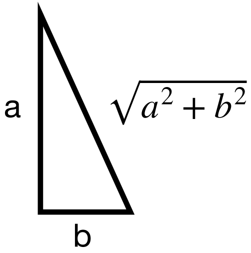
Classical Proofs

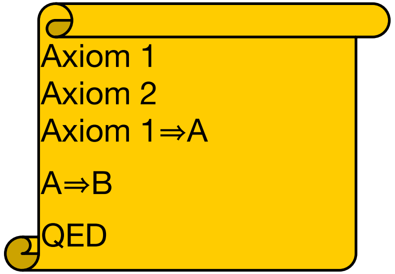



Steve Cook


Leonid Levin

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

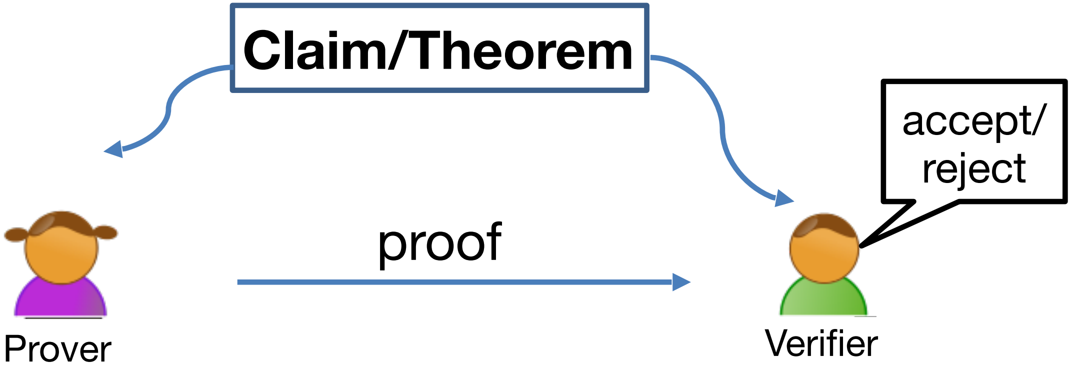



Axiom 1
Axiom 2
Axiom 1⇒A

A⇒B

QED

Proofs



Claim/Theorem

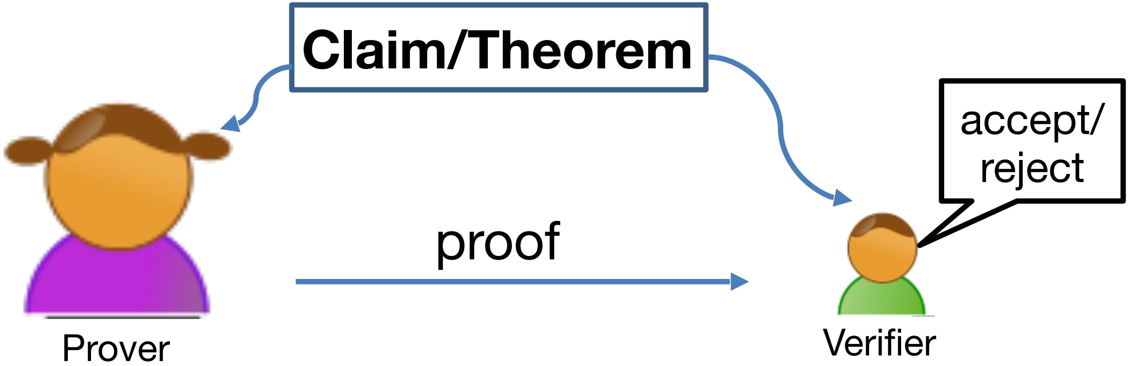
proof

Prover

Verifier

accept/
reject

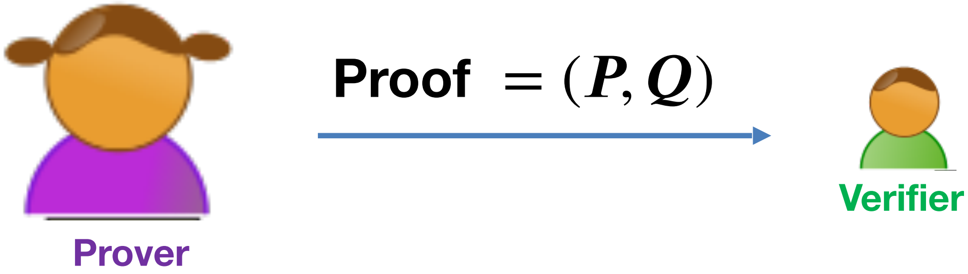
Efficiently Verifiable Proofs: NP



Works hard

Polynomial-time

Theorem: N is a product of two prime numbers



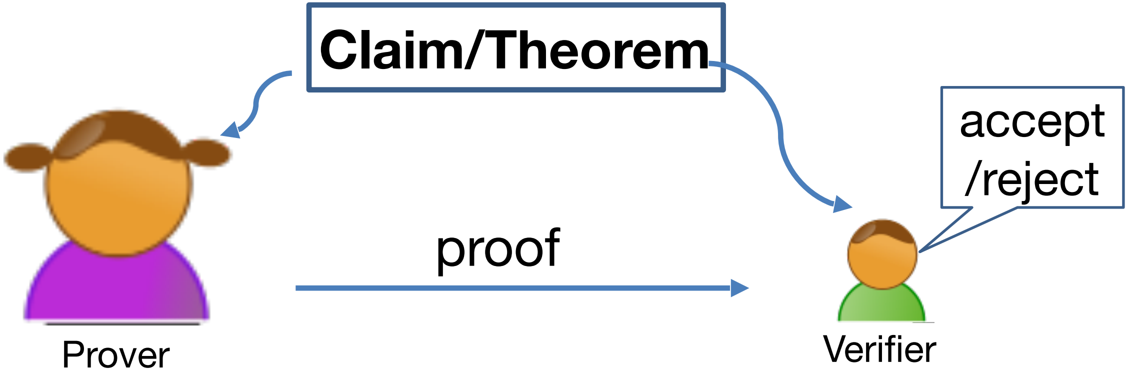
Proof $= (P, Q)$

Verifier

Prover

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

Efficiently Verifiable Proofs: NP



Claim/Theorem

accept
/reject

proof

Prover

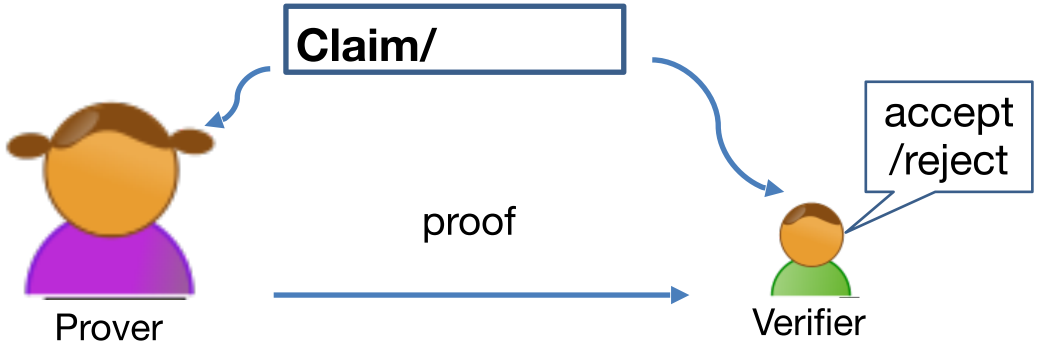
Verifier

Works hard

Polynomial-time

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

Efficiently Verifiable Proofs: NP



Claim/

accept /reject

proof

Prover

Verifier

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

- **Completeness: True theorems have (short) proofs.**

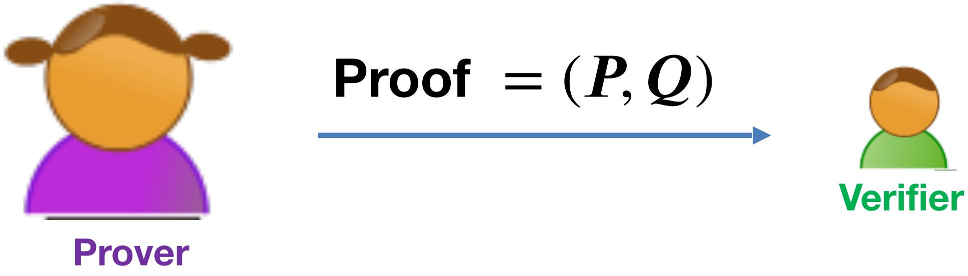
  for all $x \in \mathscr{L}$, there is a **poly($|x|$)-long** witness (proof) $w \in \{0,1\}^*$ s.t. $V(x, w) = 1$.

- **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



Proof $= \boldsymbol{(P, Q)}$

Prover

Verifier

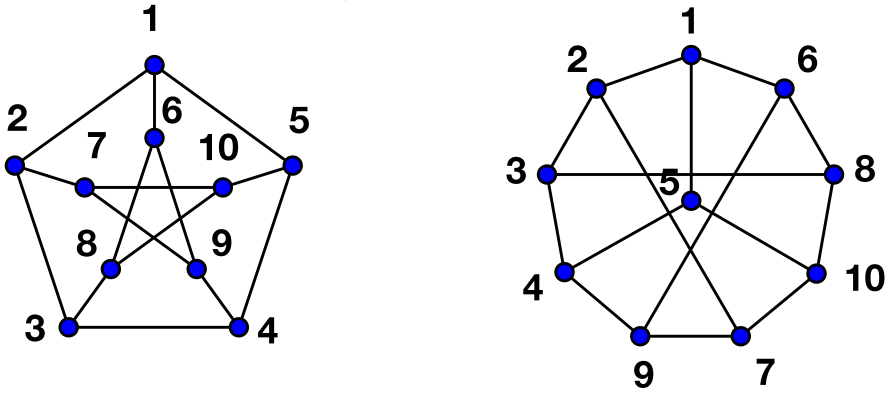
Accept *iff* $N = PQ$
and P, Q are prime

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

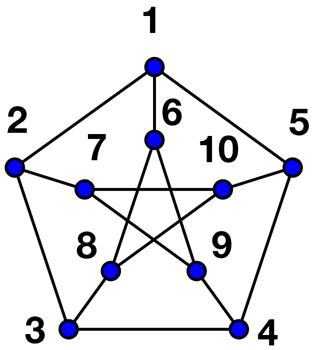
Proof $\pi : [N] \rightarrow [N]$,

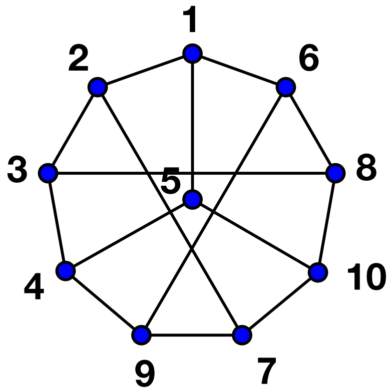
the isomorphism

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

the isomorphism

Verifier

After interaction, Bob the Verifier knows:

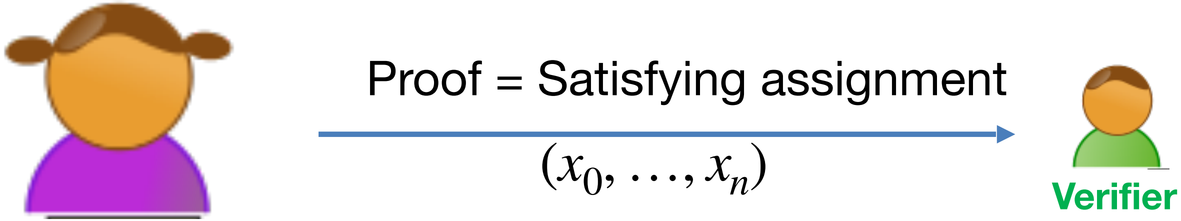
1) G_0 and G_1 are isomorphic.

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

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



Proof = Satisfying assignment

$(x_0, \ldots, x_n)$

Prover

Verifier

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

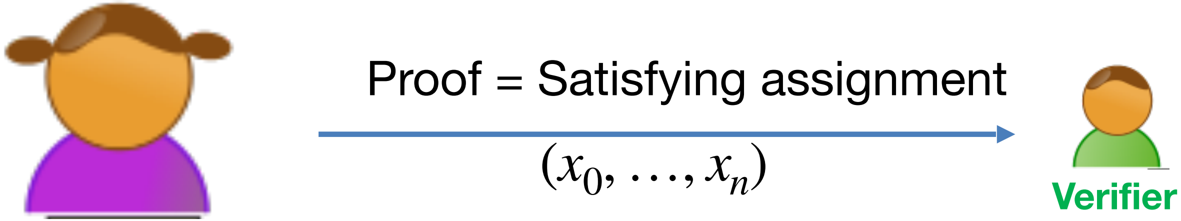
After interaction, Bob the Verifier knows:

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?

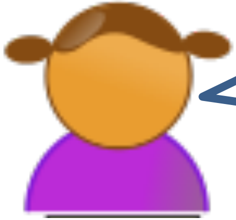
Zero Knowledge Proofs

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



Prover

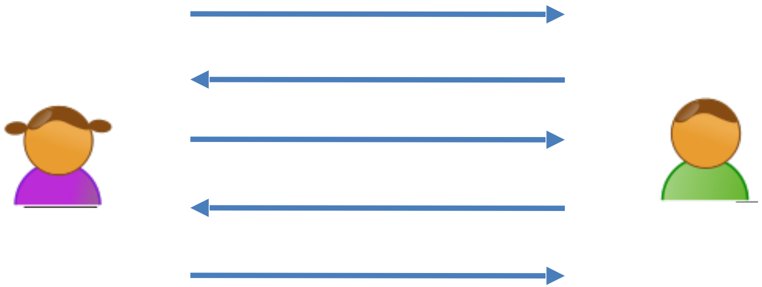
"I will not give you the isomorphism, but will prove to you that I could have one."

Micali **Goldwasser** **Rackoff**

Two (Necessary) New Ingredients

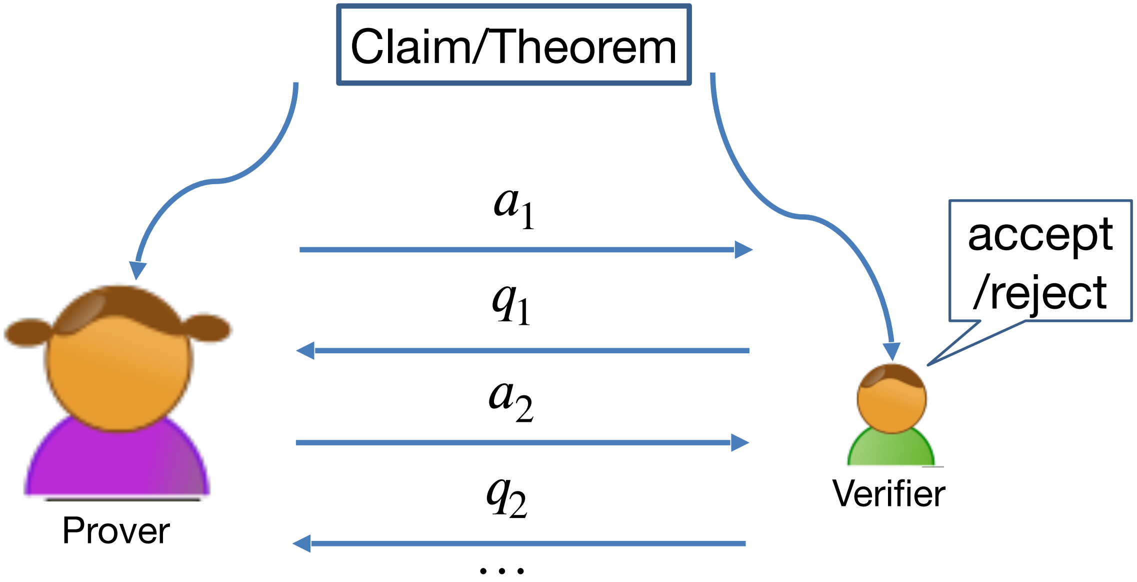
1. **Interaction:** Rather than passively reading the proof, the verifier engages in a conversation with the prover.

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



Marker example

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



Claim/Theorem

a_1

q_1

a_2

q_2

$\cdots$

accept
/reject

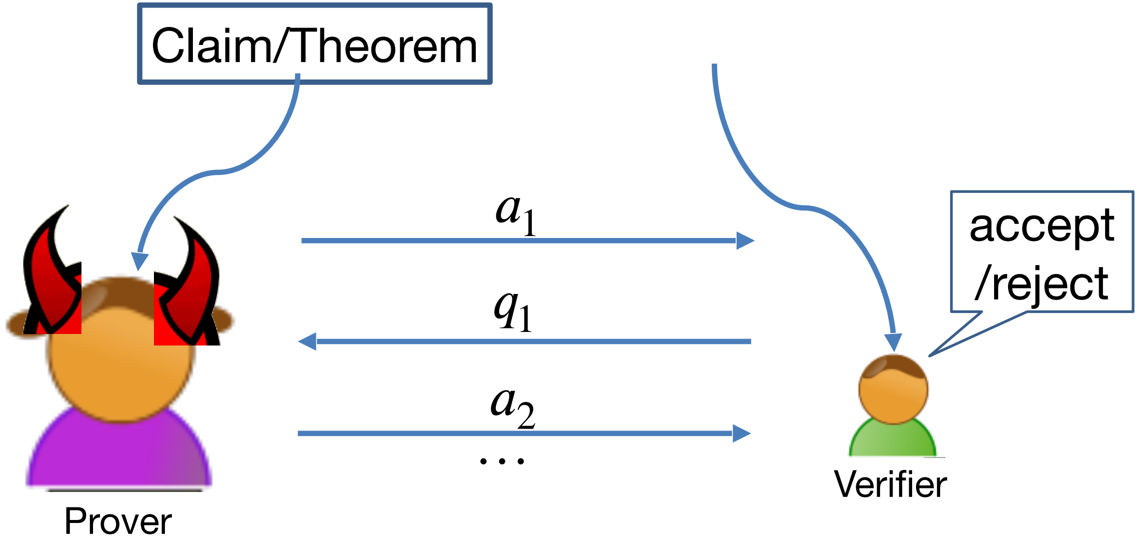
Prover

Verifier

Comp. Unbounded

Probabilistic
Polynomial-time

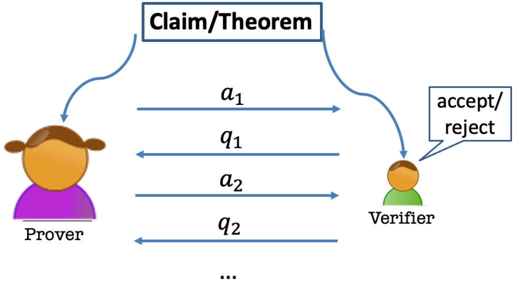
Interactive Proofs for a Language $\mathscr{L}$



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: $\underline{\mathscr{L}}$ is an <u>IP</u>-language if there is a **probabilistic poly-time** verifier $\underline{V}$ where

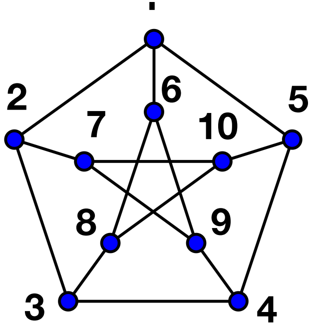
- **Completeness: If** $x \in \mathscr{L}$**,**

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

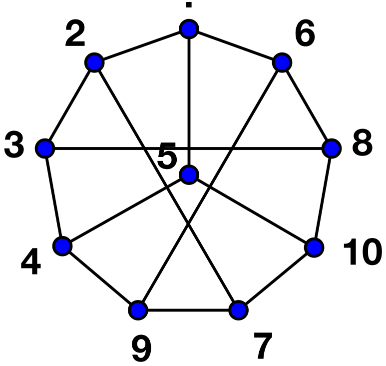
- **Soundness: If** $x \notin \mathscr{L}$, **there is a negligible function** negl **s.t. for every** $\boldsymbol{P^*}$,

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

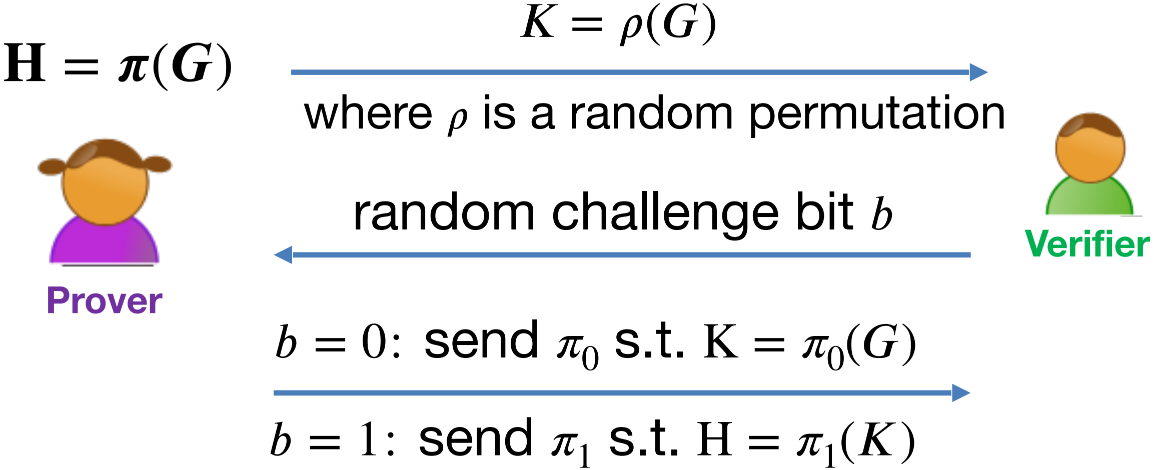
IP for Graph Isomorphism



Graph G



Graph H



$\mathbf{H} = \pi(G)$

$K = \rho(G)$

where ρ is a random permutation

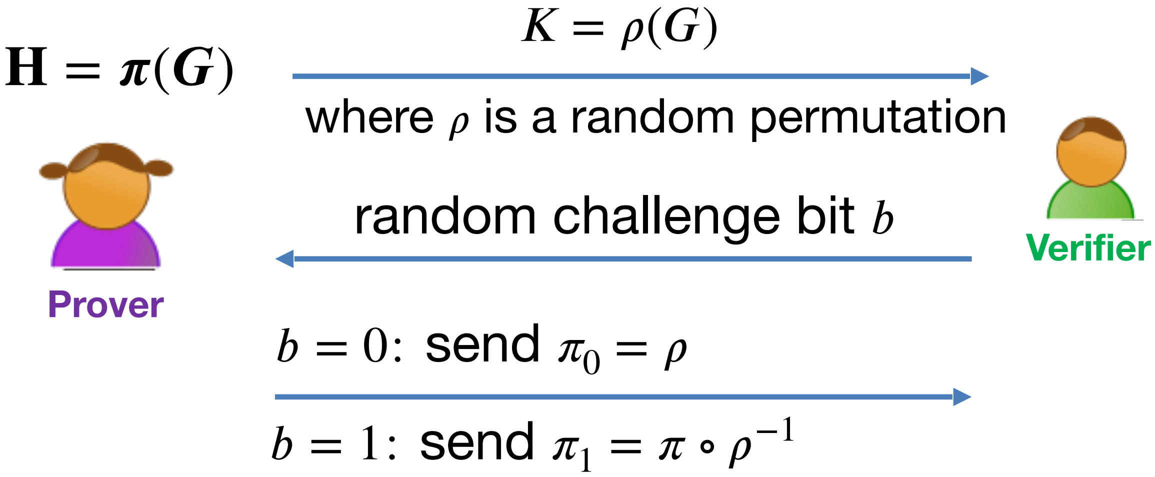
random challenge bit b

$b = 0$: send π_0 s.t. $\mathrm{K} = \pi_0(G)$

$b = 1$: send π_1 s.t. $\mathrm{H} = \pi_1(K)$

IP for Graph Isomorphism

Completeness?



$\mathbf{H} = \pi(G)$

Prover

$K = \rho(G)$

where ρ is a random permutation

random challenge bit b

$b = 0$: send $\pi_0 = \rho$

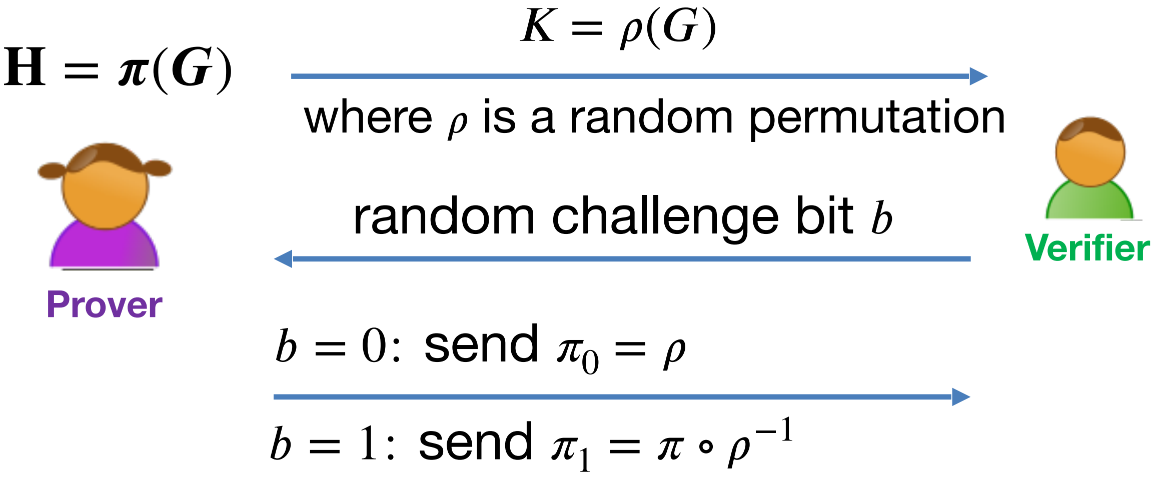
$b = 1$: send $\pi_1 = \pi \circ \rho^{-1}$

Verifier

IP for Graph Isomorphism

Soundness: Suppose G and H are non-isomorphic, and a prover could answer both the verifier challenges. Then, $K = \pi_0(G)$ and $H = \pi_1(K)$

In other words, $\mathrm{H} = \pi_1 \circ \pi_0(G)$, a contradiction!



$\mathbf{H} = \boldsymbol{\pi}(\boldsymbol{G})$

$K = \rho(G)$

where ρ is a random permutation

random challenge bit b

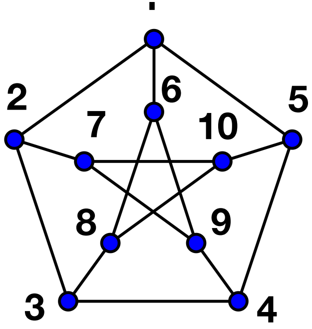
$b = 0:$ send $\pi_0 = \rho$

$b = 1:$ send $\pi_1 = \pi \circ \rho^{-1}$

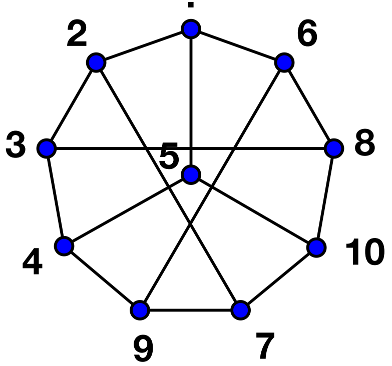
Prover

Verifier

IP for Graph *Non*-Isomorphism



Graph G



Graph H

Prover

Verifier

Sample random permutation ρ

Sample bit b

Set $K_0 = \rho(G)$ and $K_1 = \rho(H)$

K_b

Figure out which graph K_b is isomorphic to.

b'

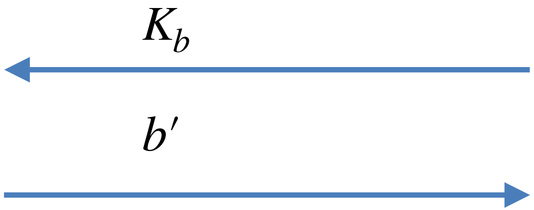
Accept if $b = b'$

IP for Graph Non-Isomorphism

Completeness?

Prover

Figure out which graph K_b is isomorphic to.



Verifier

Sample random permutation ρ
Sample bit b
Set $K_0 = \rho(G)$ and $K_1 = \rho(H)$

Accept if $b = b'$

IP for Graph Non-Isomorphism

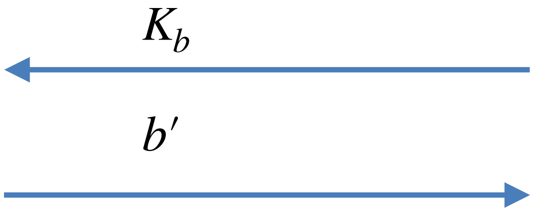
Soundness: Suppose G and H are isomorphic.
Then K_b is isomorphic to *both graphs.* Prover can't figure out which one it is isomorphic to
So best it can do is guess!

Prover

Verifier

Sample random permutation ρ
Sample bit b
Set $K_0 = \rho(G)$ and $K_1 = \rho(H)$

Figure out which graph K_b is isomorphic to.



Accept if $b = b'$

IP for Graph Non-Isomorphism

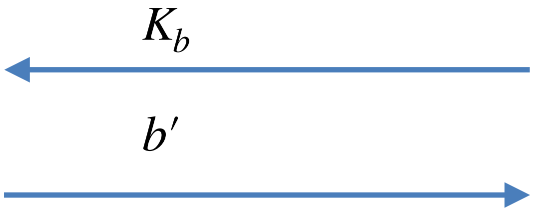
What else does the verifier learn?

Prover

Verifier

Sample random permutation ρ
Sample bit b
Set $K_0 = \rho(G)$ and $K_1 = \rho(H)$

Figure out which graph K_b is isomorphic to.



K_b

b'

Accept if $b = b'$

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?

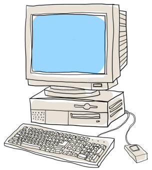
(P, V) is zero-knowledge if V can "simulate" his **VIEW** of the interaction **all by himself** in **probabilistic polynomial time**.

The Simulation Paradigm

sim S

(s, b, z)

PPT "simulator" S



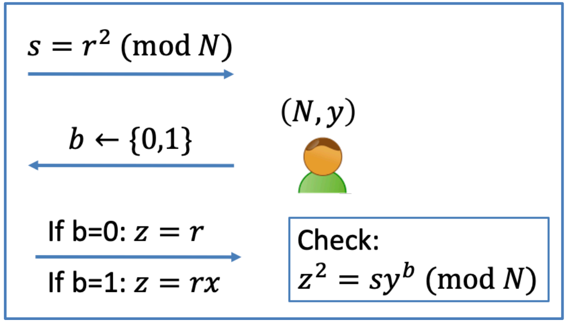
(N, y)

$view_V(P, V)$:

(s, b, z)

Transcript $= (s, b, z)$,

Coins $= b$



$s = r^2 \pmod N$

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

(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-

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-

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. $S(x, 1^\lambda)$