Zero Knowledge Proofs

Introduction to Zero Knowledge Interactive Proofs

Dan Boneh, **Shafi Goldwasser,** Dawn Song, Justin Thaler, Yupeng Zhang





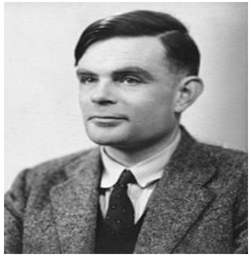
Stanford University

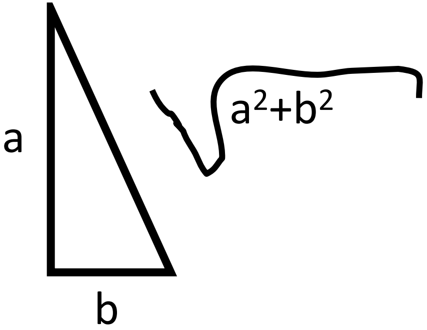
Berkeley UNIVERSITY OF CALIFORNIA

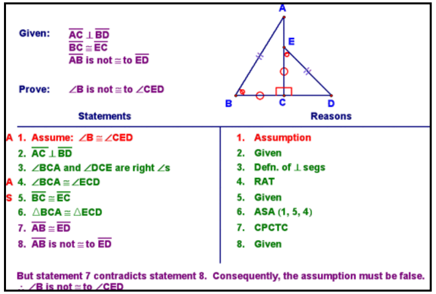
GEORGETOWN UNIVERSITY

TEXAS A&M UNIVERSITY

Classical Proofs







Given: $\overline{AC} \perp \overline{BD}$; $\overline{BC} \cong \overline{EC}$; $\overline{AB}$ is not $\cong$ to $\overline{ED}$

Prove: $\angle B$ is not $\cong$ to $\angle CED$

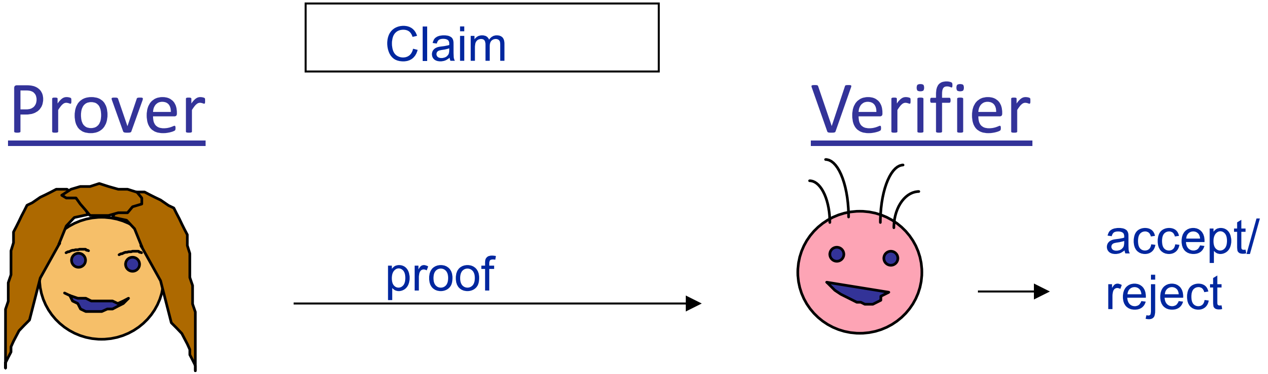
	Statements	Reasons
A	1. Assume: $\angle B \cong \angle CED$	1. Assumption
	2. $\overline{AC} \perp \overline{BD}$	2. Given
	3. $\angle BCA$ and $\angle DCE$ are right $\angle$s	3. Defn. of $\perp$ segs
A	4. $\angle BCA \cong \angle ECD$	4. RAT
S	5. $\overline{BC} \cong \overline{EC}$	5. Given
	6. $\triangle BCA \cong \triangle ECD$	6. ASA (1, 5, 4)
	7. $\overline{AB} \cong \overline{ED}$	7. CPCTC
	8. $\overline{AB}$ is not $\cong$ to $\overline{ED}$	8. Given

But statement 7 contradicts statement 8. Consequently, the assumption must be false.
$\therefore$ $\angle B$ is not $\cong$ to $\angle CED$

. . .

Prime-Number Thm

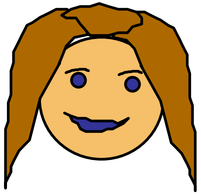
Proofs



Efficiently Verifiable Proofs (NP-proofs)

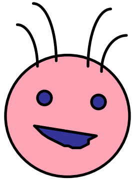
Claim

Prover



short proof →

Verifier



→ accept/ reject

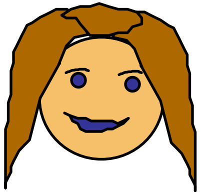
Works Hard

Polynomial Time

Efficiently Verifiable Proofs (NP-proofs)

Claim x

Prover P

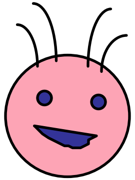


w

$|w|$ = polynomial in $|x|$

Unbounded

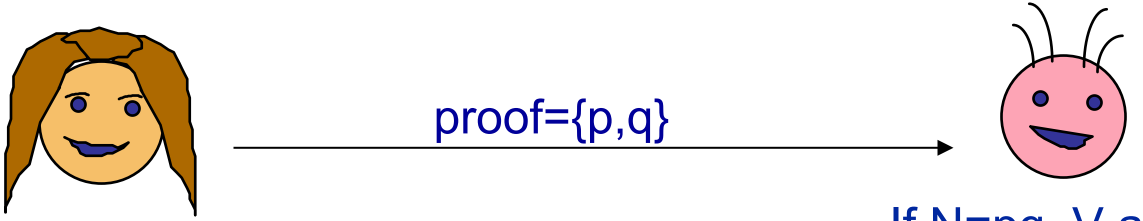
Verifier V



Accepts x If
V(x,w)=1
Else reject

V takes time Polynomial **in |x|**

Claim: N is a product of 2 large primes

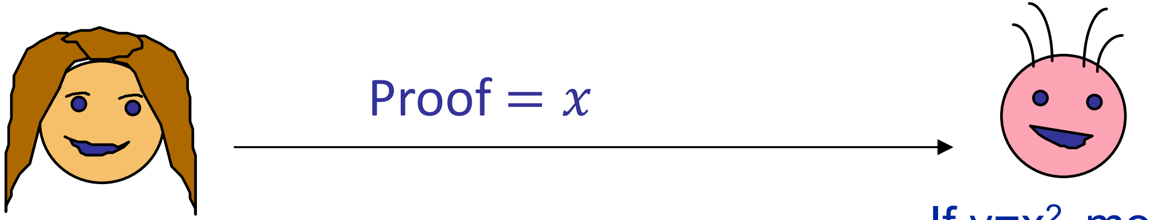


If N=pq, V accepts
Else V rejects

After interaction, V knows:

1) N is product of 2 primes
2) The two primes p and q

Claim: y is a quadratic residue mod N
(i.e $\exists x$ in Z_N^* s. t. y=x^2 mod N)

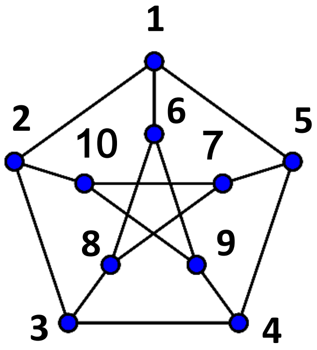


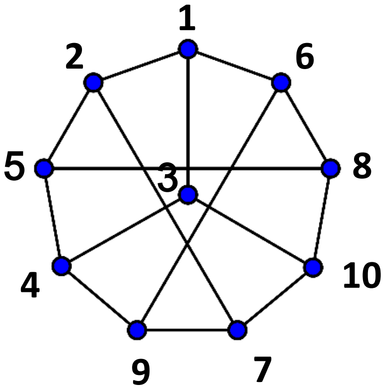
If y=x^2 mod N, V accepts
Else V rejects

After interaction, V knows:

1. y is a quadratic residue mod
2. Square root of y (hard problem equivalent to factoring N)

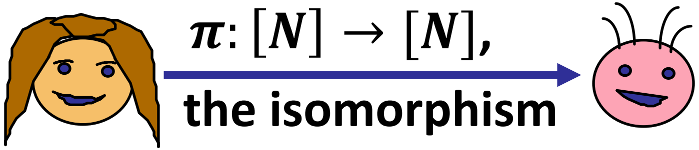
Claim: the two graphs are isomorphic





After interaction, V knows:

1) G_0 is isomorphic to G_1
2) The isomorphism $\boldsymbol{\pi}$



$\boldsymbol{\pi}: [N] \to [N]$,

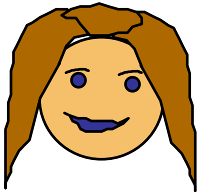
the isomorphism

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

Efficiently Verifiable Proofs (NP-Languages)

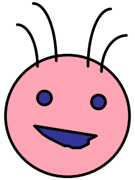
Claim x

Prover P



w →

Verifier V



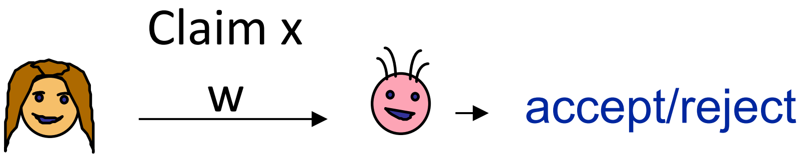
→ Accepts x If V(x,w)=1 Else reject

Works Hard

V Polynomial time

Def: A language L is a set of binary strings x.

Efficiently Verifiable Proofs (NP-Languages)



<u>Def</u>: $\mathcal{L}$ is an **NP**-language (or NP-decision problem), if there is a **poly (|x|) time** verifier V where

- **Completeness [True claims have (short) proofs].**

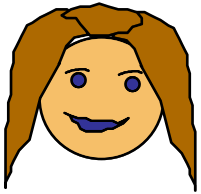
  if x $\in \mathcal{L}$, there is a **poly(|x|)-long** witness w $\in \{0,1\}^*$ s.t. $V(x,w) = 1$.

- **Soundness [False theorems have no proofs}.**

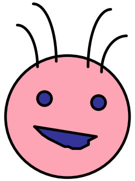
  if x $\notin \mathcal{L}$, there is no witness. That is, for all w $\in \{0,1\}^*$, $V(x,w) = 0$.

1982-1985: Is there any other way?

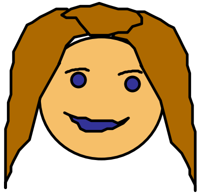
Theorem: y is a quadratic residue mod N



Proof=$\sqrt{y} \ mod \ N \in Z_N^*$ →

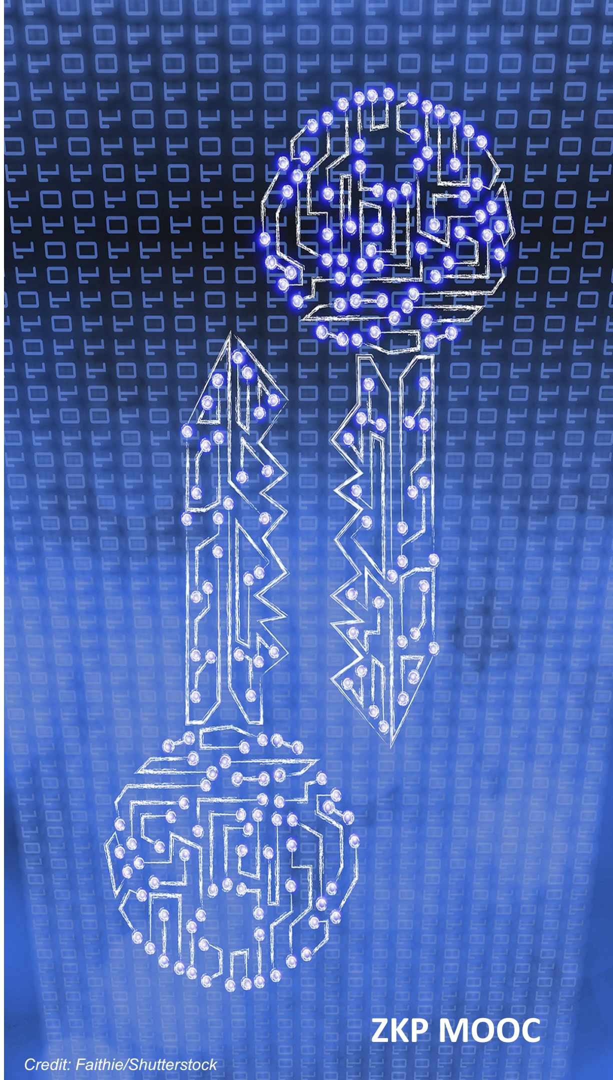


Zero Knowledge Proofs: Yes



Main Idea:
Prove that
I **could** prove it
If I felt like it

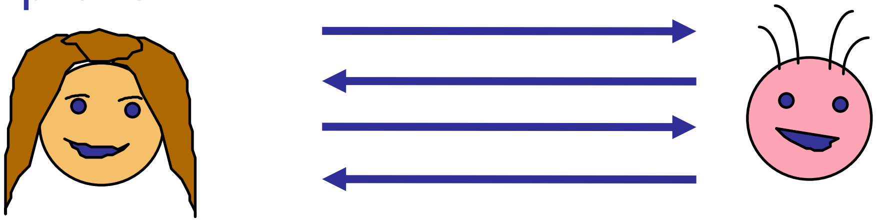
Zero Knowledge Interactive Proofs



Two New Ingredients

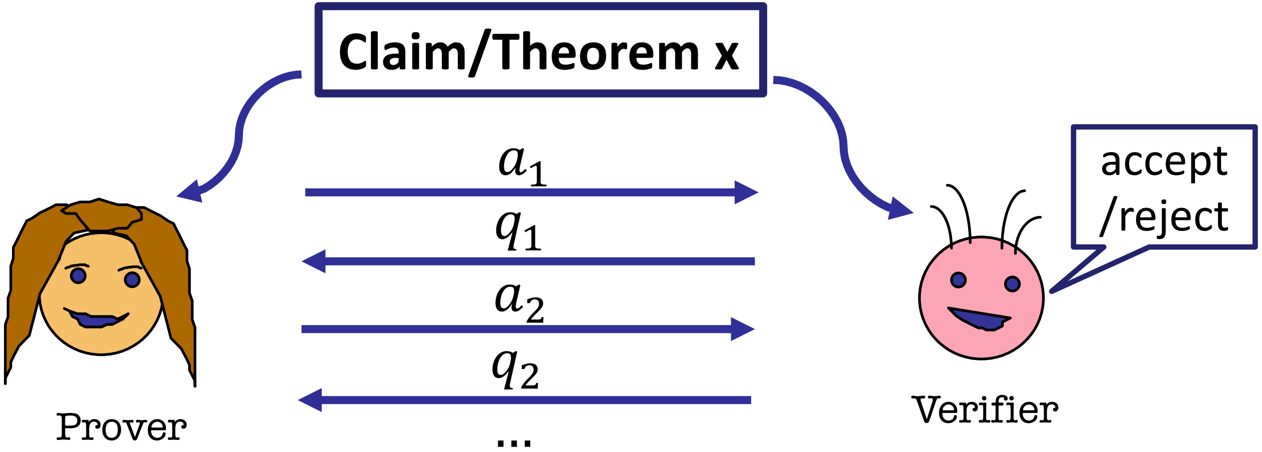
Interactive and Probabilistic Proofs

Interaction: rather than passively "reading" proof, verifier engages in a non-trivial interaction with the prover.



Randomness: verifier is randomized (tosses coins as a primitive operation), and can err in accept/reject with small probability

Interactive Proof Model



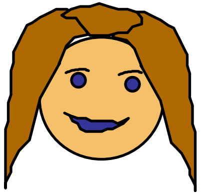
Comp. Unbounded

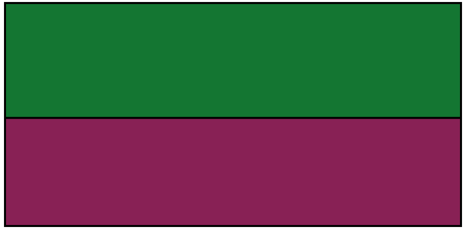
Probabilistic Polynomial-time (PPT)

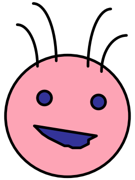
Here is the idea:

How to prove colors are different to a **blind verifier**

Claim: This page contains 2 colors







Toss coin to decide if to flip page over or not
Heads flip, Tails don't

Sends resulting page

If page is flipped
Set coin'=heads
Else coin'=tails

I guess you tossed coin'

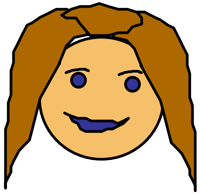
If coin $\neq$ coin',
reject, else accept

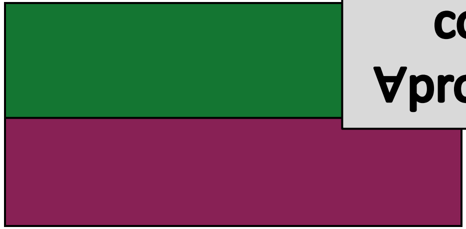
Here is the idea:
How to prove colors are differ

Claim: This page contains 2 c

- If there are 2 colors, then Verifier will accept
- If there is a single color, ∀provers $\text{Prob}_{\text{coins}}$(Verifier accept)$\leq$ 1/2
- If repeat i=1..k times and V accept if $\text{coin}_i' = \text{coin}_i$ every repetition,

∀provers $\text{Prob}_{\text{coins}}$(Verifier accept)$\leq 1/2^k$





Toss coin to decide if to flip page over or not
Heads flip, Tails don't

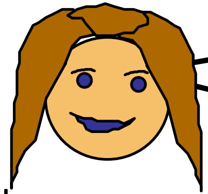
Sends resulting page

If page is flipped
Set coin'=heads
Else coin'=tails

I guess you tossed coin'

If coin $\neq$ coin',
reject, else accept

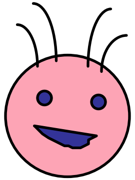
Interactive Proof for QR$= \{(N, y) : \exists x \ s.t.\ y = x^2 \ mod \ N\}$



Choose random $1 \leq r \leq N$ s.t. gcd(r,N)=1

Send $s=r^2$ mod n and say

- If I gave you square roots of both s and sy mod N you would be convinced that the claim is true (but also know $\sqrt{y}$ mod N)
- Instead, I will give you a square root of either s or of sy mod N but you get to choose which!



Flip a b=  to choose

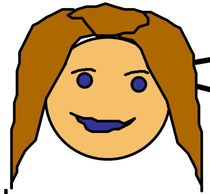
If b=1: send z=r

If b=0: send z=r$\sqrt{y}$ mod N

Accepts only if $z^2=sy^{1-b}$ mod N

Interactive Proof for QR= {

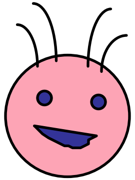
- **Completeness: If Claim is true, then Verifier will accept**
- **Soundness: If Claim is false, ∀provers $\text{Prob}_{\text{coins}}$(Verifier accept)≤ 1/2**
- **Prover only needs to know $\mathbf{x = \sqrt{y}}$**



Sends $s=r^2$ mod n and says

- If I gave you square roots [text covered by the grey box] be convinced that the claim is true (but also know $\sqrt{y}$ mod N)
- Instead, I will give you a a square rot of s or of sy mod N but you get to choose which!

Choose random $1 \leq r \leq N$ s.t. gcd(r,N)=1



Flip a b=  to choose

Accepts only if $z^2 = sy^{1-b}$ mod N

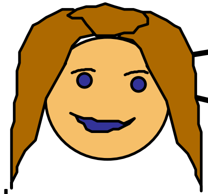
If b=1: send z=r

If b=0: send z=r $\sqrt{y}$ mod N

Interactive Proof for Q

- **Completeness: If Claim is true, then Verifier will accept**
- **Soundness: If Claim is false, ∀provers Prob_{coins}(Verifier accept)≤ $(1/2)^{100}$**
- **Prover only needs to know $x = \sqrt{y}$**

Repeat 100 times



Choose random $1 \leq r \leq N$ s.t. gcd(r,N)=1

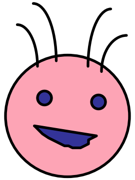
Send s=r^2 mod n and s

- If I gave you square
  convinced that the claim is true (but also know $\sqrt{y}$ mod N)
- Instead, I will give you a a square rot of s or of sy mod N but you get to choose which!
- The fact that I COULD (in principle) do both, should convince you

Flip a b=  to choose

If b=1: send z=r

If b=0: send z=r$\sqrt{y}$ mod N

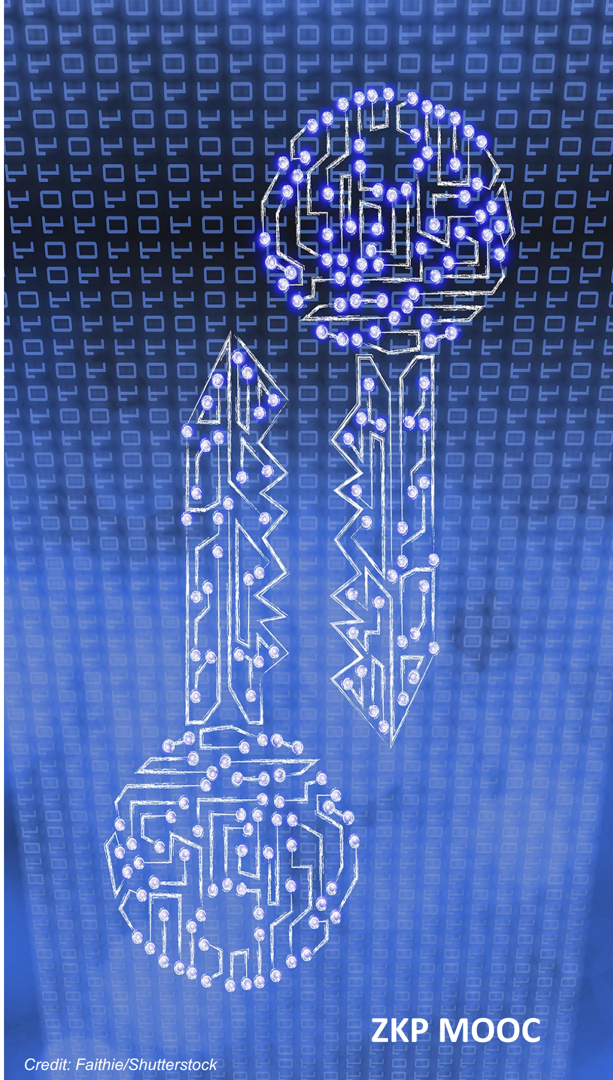


Accepts only if $z^2 = sy^{1-b}$ mod N

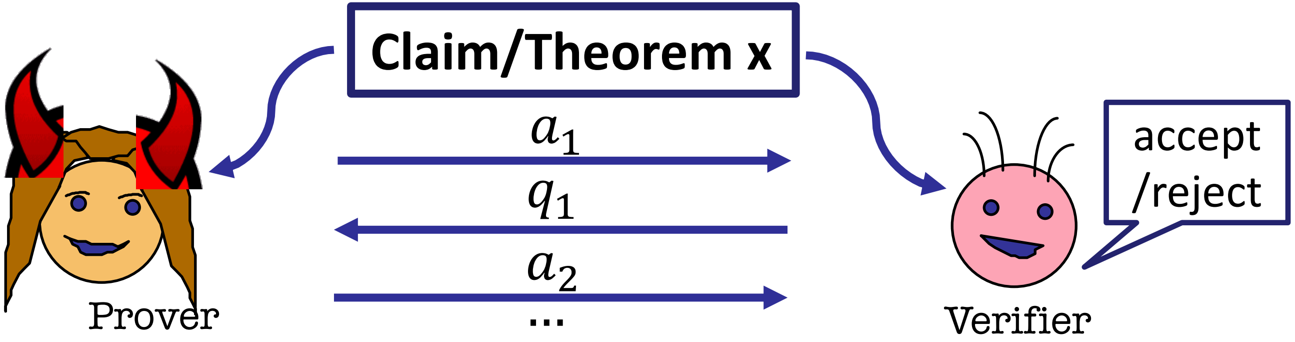
What Made it possible?

- The statement to be proven has **many possible proofs** of which the prover chooses one ***at random***.
- Each such proof is made up of exactly 2 parts: seeing either part on its own gives the verifier no knowledge; seeing both parts imply 100% correctness.
- Verifier chooses **at random** which of the two parts of the proof he wants the prover to give him. The ability of the prover to provide either part, convinces the verifier

Definitions : of Zero Knowledge Interactive Proofs



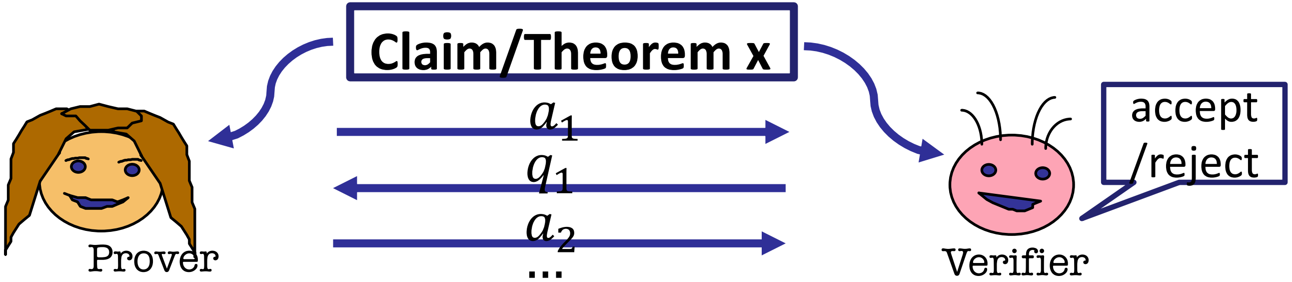
Interactive Proofs for a Language $\mathcal{L}$



Def: (P, V) is an interactive proof for L, if V is probabilistic poly (|x|) time &

- **Completeness**: If $x \in \mathcal{L}$, V always accepts.
- **Soundness:** If $x \notin \mathcal{L}$, for all cheating prover strategy, V will not accept except with negligible probability.

Interactive Proofs: Notation

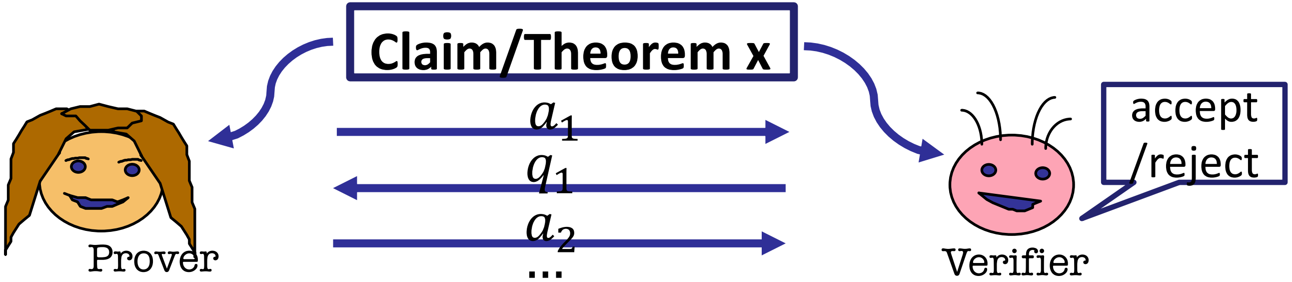


Def: (P, V) is an interactive proof for L, if V *is* probabilistic poly (|x|) and

- **Completeness**: **If** $x \in \mathcal{L}$, $Pr[(P,V)(x) = accept] = 1$.
- **Soundness: If** $x \notin \mathcal{L}$, **for every** $\boldsymbol{P^*}$, $\Pr[(P^*, V)(x) = accept] = negl(|x|)]$

where $\text{negl}(\lambda) < \frac{1}{\text{polynomial}(\lambda)}$ for all polynomial functions

Interactive Proofs: Notation

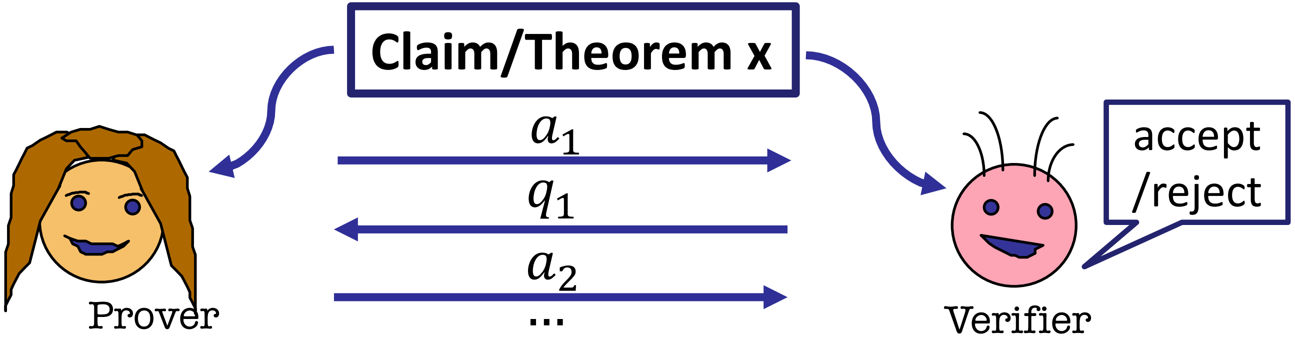


Def

This is what a proof ultimately is!

whe

Interactive Proofs for a Language $\mathcal{L}$: Notation

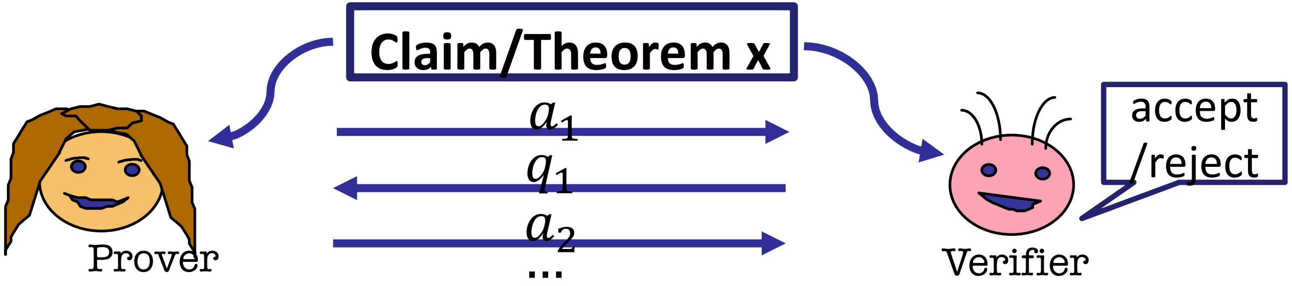


<u>Def</u>: (P, V) is an interactive proof for L, if V *is* probabilistic poly (|x|) and

- **Completeness: If** $\mathrm{x} \in \mathcal{L}$**,** $\Pr[(\mathrm{P}, V)(x) = accept] \geq c$
- **Soundness: If** $\mathrm{x} \notin \mathcal{L}$, **for every** $\boldsymbol{P}^*$, $\Pr[(P^*, V)(x) = accept] \leq s$

Equivalent as long as $\boldsymbol{c} - \mathbf{s} \geq 1/\mathrm{poly}(|x|)$

The class of Interactive Proofs (IP)



<u>Def</u>: class of languages **IP =**

{L for which there is an interactive proof}

What is zero-knowledge?

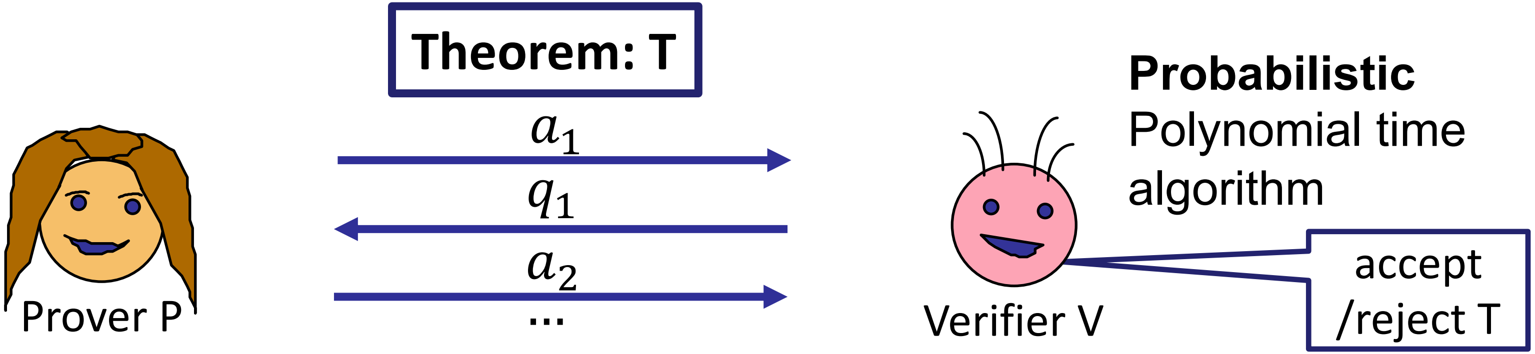
for every verifier

For true Statements,

What the verifier can compute
after the interaction =
What the verifier could have computed
before interaction

How do we capture this mathematically?

The Verifier's View



- After interactive proof, V "learned":
  - T is true (or $x \in \mathcal{L}$)
  - A **view** of interaction (= transcript + coins V tossed)

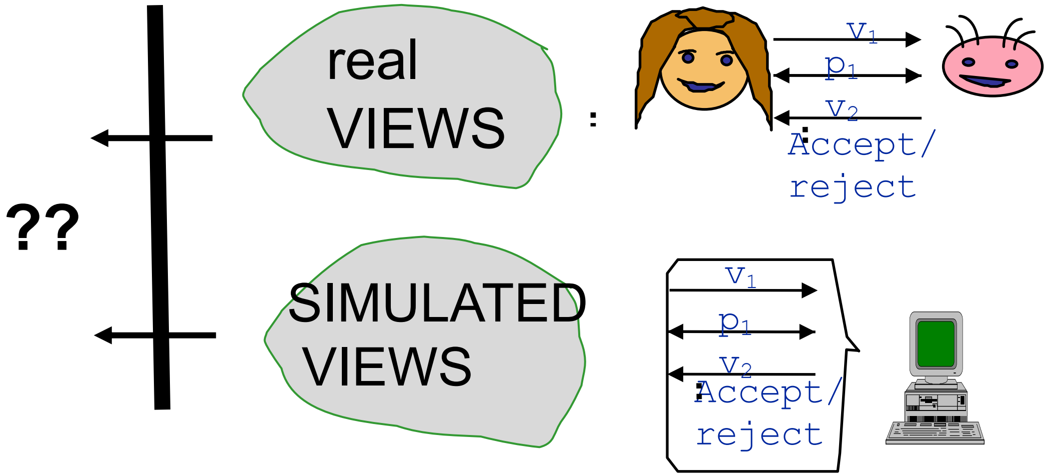
Def: $\textbf{view}_V(\boldsymbol{P}, \boldsymbol{V})[\boldsymbol{x}] =$ $\{(q_1,a_1,q_2,a_2,\ldots,\text{coins of V})\}$.

(probability distribution over coins of V and P)

The Simulation Paradigm

V’s view gives him nothing new, if he could have simulated it its own s.t `simulated view’ and `real-view’ are computationally-Indistinguishable

The poly-time Distinguisher

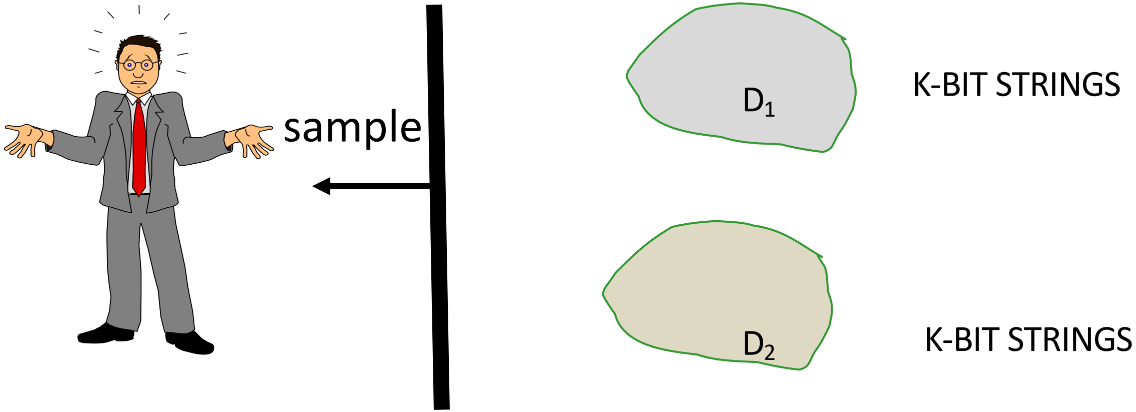


When Theorem is true

Computational Indistinguishability

If no "distinguisher" can tell apart two different probability distributions they are "effectively the same".

Probabilistic
Poly Time (k)
Distinguisher D



For all distinguisher algorithms D, even after receiving a polynomial number of samples from D_b, Prob[D guesses b] <1/2+negl(k)

Zero Knowledge: Definition

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

1. $view_V(P,V)[x] =$ {(q_1,a_1,q_2,a_2,…,coins of V)} (over coins of V and P)
2. $Sim(x)$

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

Zero Knowledge: Definition

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

Allow simulator S
Expected
Poly-time

1. $view_V(P, V)[x, 1^\lambda] =$ {(q$_1$,a$_1$,q$_2$,a$_2$,…,coins of V)} (over coins of V and P)
2. $Sim(x, 1^\lambda)$

Technicality:
Allows sufficient
Runtime onn small x
λ- security parameter

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

What if V is NOT HONEST

OLD DEF

An Interactive Protocol (P,V) is **honest-verifier** zero-knowledge for a language L if there exists a PPT simulator Sim such that for every $x \in L$,

$$view_V(P,V)[x] \approx \quad Sim(x, 1^\lambda)$$

REAL DEF

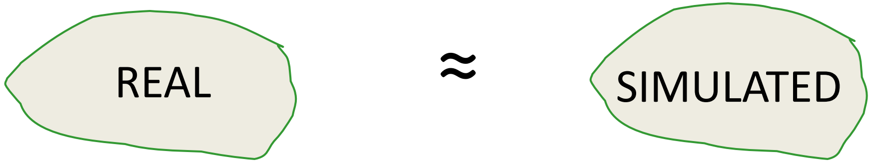
An Interactive Protocol (P,V) is **zero-knowledge** for a language L if **for every PPT V^***, there exists a poly time simulator Sim s.t. for every $x \in L$,

$$view_V(P,V)[x] \approx \quad Sim(x, 1^\lambda)$$

34

Flavors of Zero Knowledge

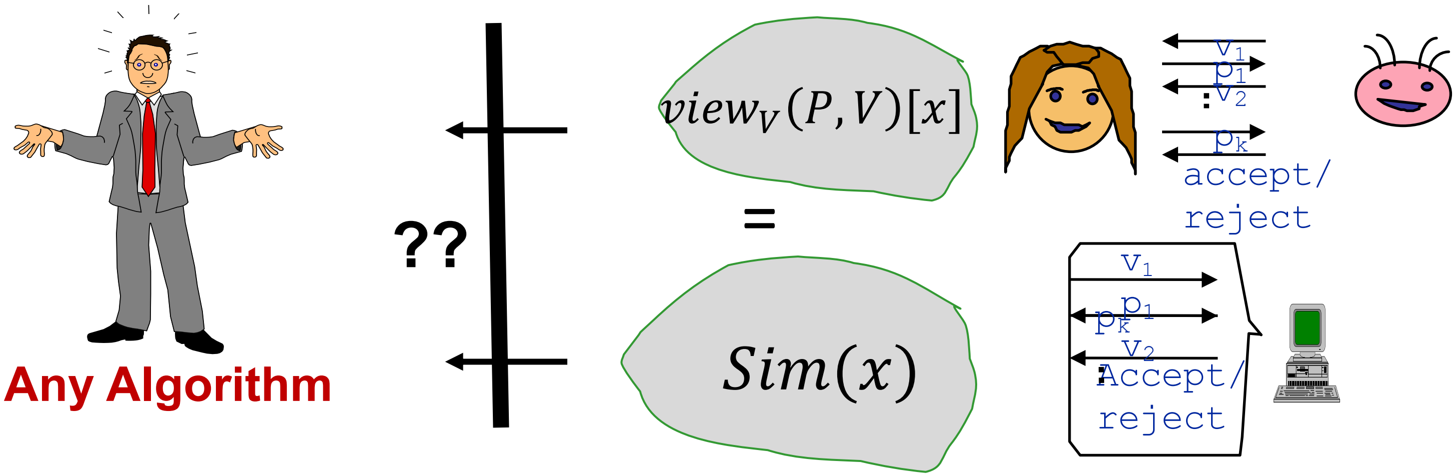
$view_V(P, V)[x]$ $\qquad$ $Sim(x, 1^\lambda)$



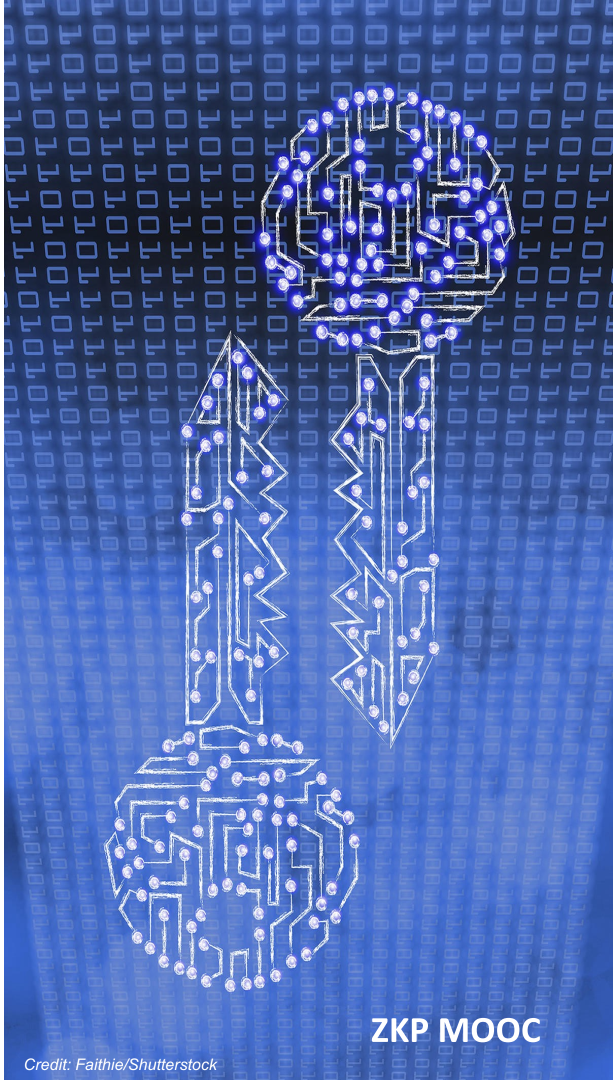
- Computationally indistinguishable distributions = CZK
- Perfectly identical distributions = PZK
- Statistically close distributions = SZK

Special Case: **Perfect** Zero Knowledge

verifier's view can be exactly efficiently simulated
`Simulated views' = `real views'



Working through a Simulation for QR Protocol



Recall the Simulation Paradigm

$view_V(P, V)$:

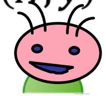
Transcript $= (s, b, z)$,

Coins = b

$s = r^2 \pmod N$ →

$b \leftarrow \{0,1\}$ ←

(N, y)



If b=0: $z = r$ →

If b=1: $z = rx$

Check:

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

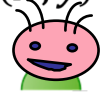
Recall the Simulation Paradigm

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

sim :
(s,b,z)

$s = r^2 \pmod N$ →

(N,y)

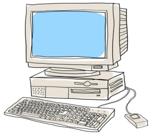


← $b \leftarrow \{0,1\}$

If b=0: $z = r$ →
If b=1: $z = rx$

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

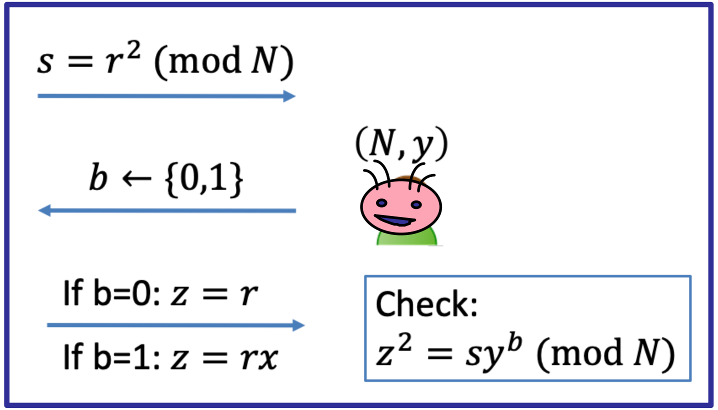
PPT "simulator" *Sim*



(N,y)

(Honest Verifier) Perfect Zero Knowledge

Claim: The QR protocol is perfect zero knowledge.



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).

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

claim: The simulated transcript is identically distributed as the real transcript

Perfect Zero Knowledge: for all V*

Claim: The QR protocol is perfect zero knowledge.

$s = r^2 \pmod N$ →

(N, y)

$b \leftarrow \{0,1\}$ ←

If b=0: $z = r$ →

If b=1: $z = rx$

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

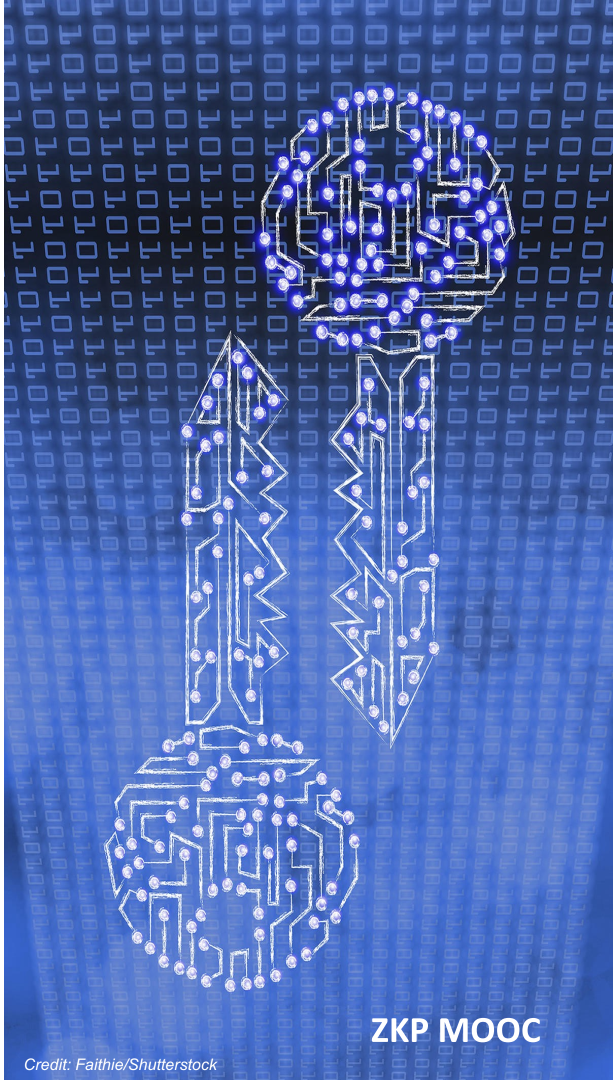
$view_V(P, V)$:
(s, b, z)

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. If V*((N,y),s) = b output (s, b, z)
   if not goto 1 and repeat

Claim: Expected number of repetitions is two

ZK proof of Knowledge

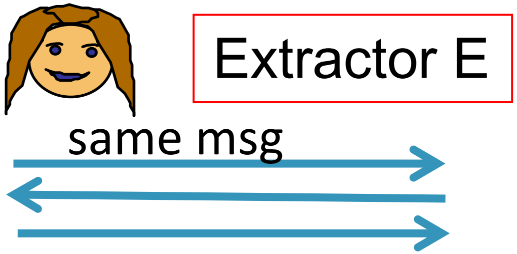


Prover seems to have proved more: theorem is correct and that she "knows" a square root mod N

Consider $\mathbf{L_R} = \{x : \exists w \; s.t.\; R(x,w) = accept\}$ for poly-time relation R.

Def: (P,V) is a proof of knowledge (POK) for L_R if :
$\exists$ PPT (knowledge) extractor algorithm E s. t. $\forall$x in L,
in expected poly-time $E^P(x)$ outputs w s.t. R(x,w)=accept.

$E^P(x)$ (E may run P repeatedly on the same randomness)
possibly asking different questions in every executions
This is called the rewinding technique



Prover seems to have proved more not only that theorem is correct, but that she "knows" a square root mod N

Consider $L_R = \{x : \exists \boldsymbol{w}\ \boldsymbol{s.t.}\ \boldsymbol{R(x,w) = accept}\}$ for poly-time relation R.

Def: (P,V) is a proof of knowledge (POK) for L_R if :

$\exists$ PPT (knowledge) extractor algorithm E s.t. $\forall$x in L,

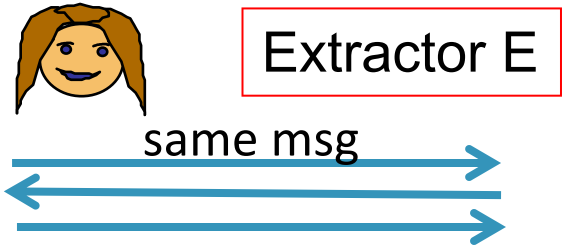
in expected poly-time $E^P(x)$ outputs w s.t. R(x,w)=accept.

[if Prob[(P,V)(x)=accept] > α, then $E^P(x)$ runs in expected poly(|x|,1/α) time]

$E^P(x)$ (may run P repeatedly on the same randomness)

Possibly asking different questions in every executions

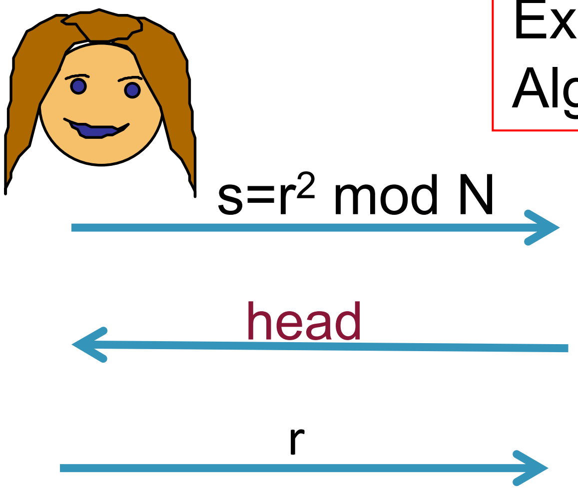
This is called the rewinding technique

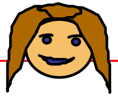


ZKPOK that Prover knows a square root x of y mod N

Input: (y,N)

Extractor Algorithm





Extractor:

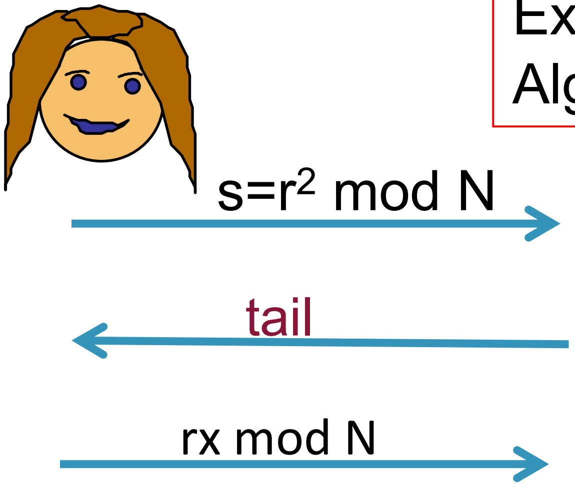
On input (y,N),

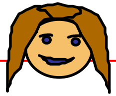
1. Run prover & receive s
2. Set verifier message to head; Store r

The Rewinding Method

Input: (y,N)

Extractor Algorithm



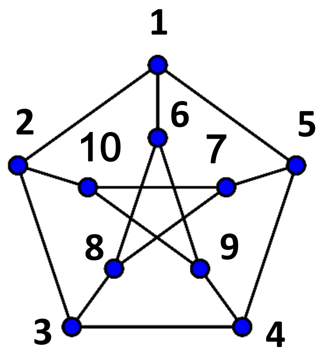


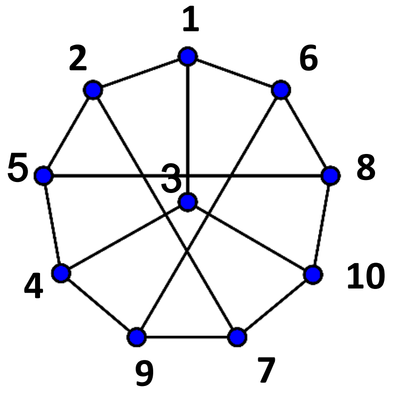
Extractor:

On input (y,N)

1. Run prover & receive s
2. Set verifier message to head; receive and store r
3. Rewind and 2^{nd} time set verifier message to tail receive rx
4. Output rx/r=x mod N

ZK Proof for Graph Isomorphism



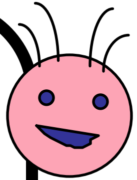


Recall:

G_0 is isomorphic to G_1

If ∃ isomorphism $\boldsymbol{\pi}: [N] \to [N]$, $\forall i, j$: $\left(\pi(i), \pi(j)\right) \in E_1$ iff $(i, j) \in E_0$.

ZK Interactive Proof for Graph Isomorphism



I will produce a random graph H for which

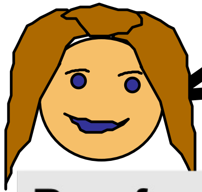
1: I can give an isomorphism γ_0 from G_0 to H

OR

2: I can give an isomorphism γ_1 from G_1 to H

Thus, $\exists$isomorphism σ from G_0 to G_1

Verifier, please randomly choose if I should demonstrate my ability to do #1 or #2.



Proof:

$H = \gamma_0(G_0)$,

$H = \gamma_1(G_1)$,

Thus

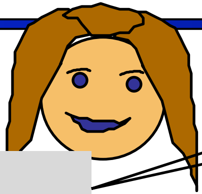
$G_1 = \gamma_1^{-1}(\gamma_0(G_0))$

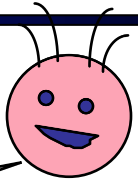
Set $\sigma = \gamma_1^{-1} \cdot \gamma_0$

POINT IS: If I can do both,
there exists an isomorphism from G_0 to G_1

REPEAT K INDEPENDENT TIMES.

Input: (G_0,G_1)





Choose random γ_0 permutation of vertices of G_0. Set $H=\gamma_0(G_0)$

Graph H

Toss coin b

b

If b=0: send γ_0

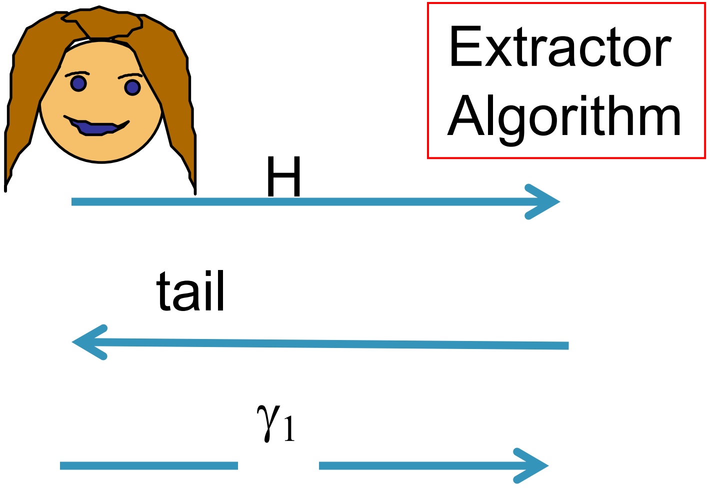
If b=1: send $\gamma_0\,\sigma^{-1}$ (where $\sigma(G_0)=G_1$)

Claims: (1) Statement true ➡ can answer correctly for b= 0 and 1

(2) Statement false ➡ prob_b(catch a mistake)$\geq 1\text{-}1/2^k$

(3) Perfect ZK [Exercise]

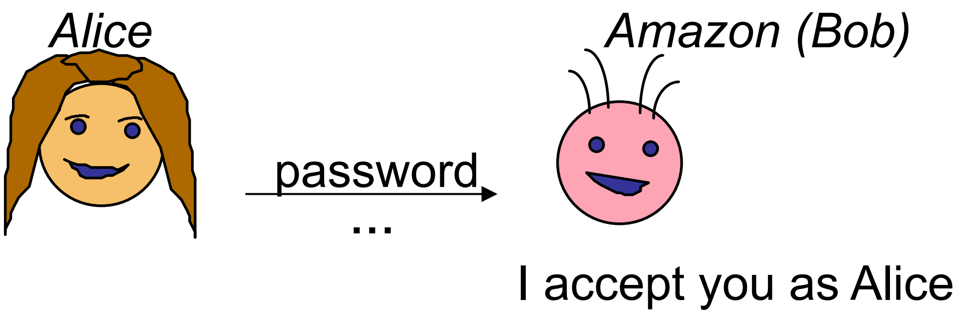
ZKPOK that **Prover knows an isomorphism** from G_1 to G_2



Extractor :

1) On input H
   set coin=head
   Store γ_0
2) Rewind and 2nd time
   set coin=tail
   Store γ_1
3) Output $\gamma_1^{-1}(\gamma_0)$

The first application: Identity Theft [FS86]



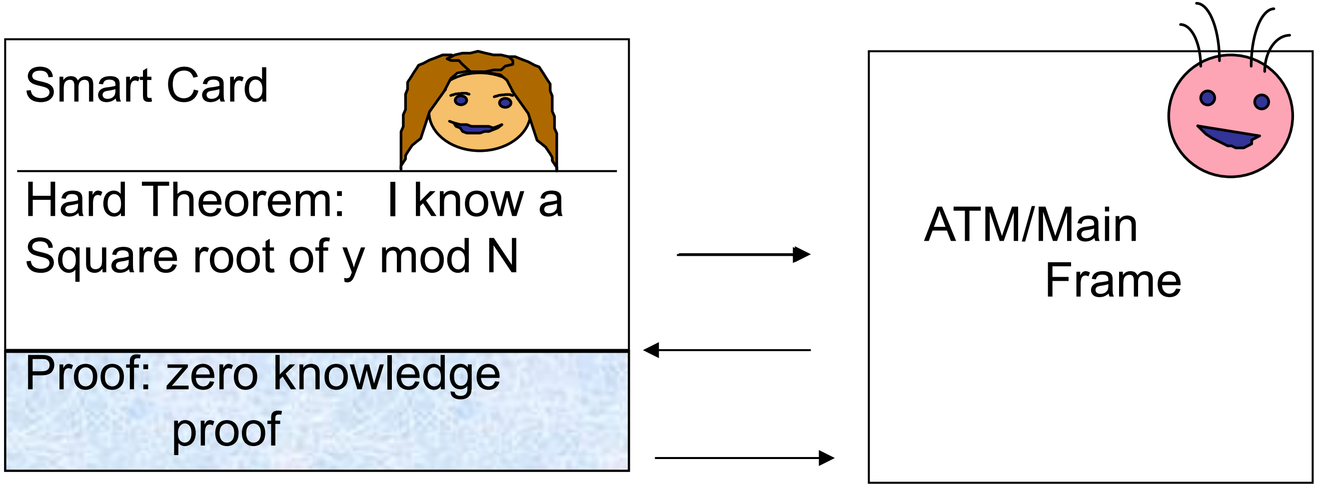
For Settings:

- Alice = Smart Card.
- Over the Net
- Breaking ins at Bob/Amazon are possible

Passwords are no good

Zero Knowledge: Preventing Identity Theft

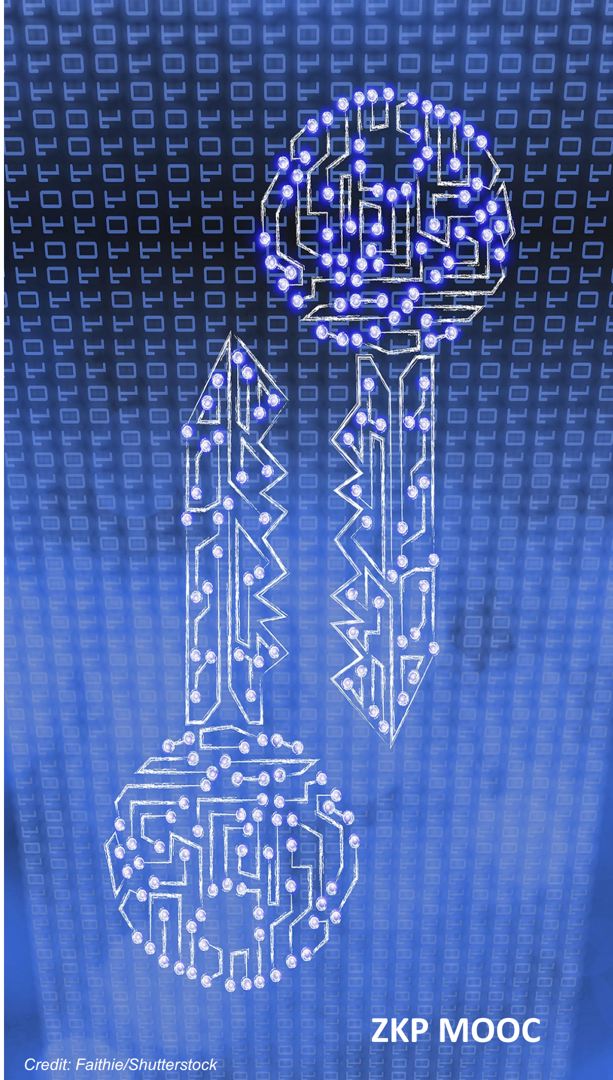
PROVER VERIFIER



To identify itself prover proves a hard theorem.

Interesting examples, one application

But, do all NP Languages have Zero Knowledge Interactive Proofs?



Yes: All of NP is in Zero Knowledge

Theorem[GMW86,Naor]: If one-way functions exist, then every L in NP has computational zero knowledge interactive proofs

Ideas of the proof:

1. Show that an NP-Complete Problem has a ZK interactive Proof

[GMW87] Showed ZK interactive proof for G3-COLOR using bit- commitments

⇒For any other L in NP, L $<_p$ G3-COLOR (due to NPC reducibility)

⇒ Every instance x can be reduced to graph G_x such that

- if x in L then G_x is 3 colorable
- if x not in L then G_x is not 3 colorable

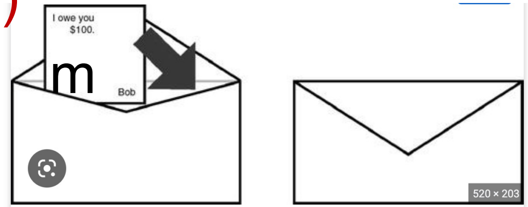
Can you show Zero Knowledge for all of NP [GMW87]

Theorem[GMW86, Naor]: If one-way functions exist, then every L in NP has computational ZK interactive proofs

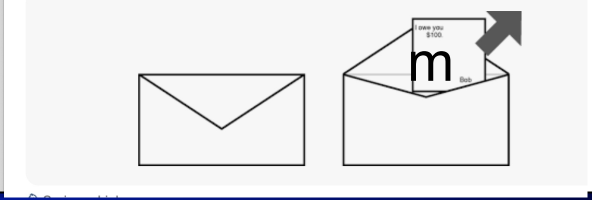
Ideas of the proof:

1. [GMW87] Show that an NP-Complete Problem has a ZK interactive Proof if bit commitments exist
2. [Naor]One Way functions ➡ bit commitment protocol exist

Commit(m)



Decommit



hiding

binding

Properties of a Bit Commitment Protocol (Commit, Decommit) between Sender S and Receiver R

Hiding: ∀ receiver R∗ , after commit stage ∀ b, b′ ∈ {0,1}, view of sender R*

$\{View_{R*}\{Sender(b),R^*)(1^k)\} \approx_c \{View_{R*}(Sender(b'),R^*)(1^k)\}$ [k=sec. param]

Binding: ∀ sender S*, after commit and decommit stage

Prob[R will accept two different values b and b’] < negl(k)

K-security parameter

Ex: Use (semantically) secure probabilistic encryption scheme Enc

Commit(b)= “sender chooses r and sends c=Enc(b;r)”

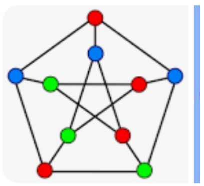
Decommit(c) =“sender sends r and b. Receiver rejects unless c=Enc(b;r)”

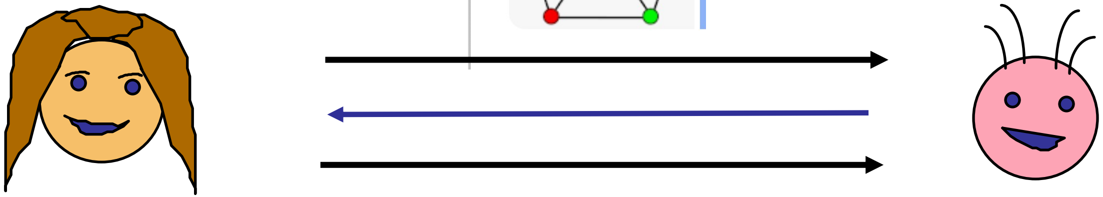
All of NP is in Zero Knowledge

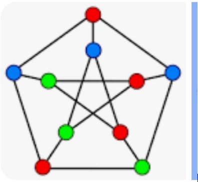
Theorem[GMW86,Naor]: If one-way functions exist, then every L in NP has computational zero knowledge interactive proofs

Ideas of the proof:

1. Show that an G3-COLOR has a ZK interactive Proof

Theorem :  is 3-color



Theorem :  is G3-COLORABLE

On common input graph G =(V,E) & prover input coloring π: V → {0,1,2}

1. **Prover:** pick a random permutation σ of colors {0,1,2} & color the graph with coloring $\phi(v):=\sigma(\pi(v))$, and **commit** to each color of each vertex v *by running Commit(ϕ(v)) protocol*
2. **Verifier**: select a random edge e=(a, b) to send to Prover
3. **Prover**: **Decommit** colors *ϕ(a) & ϕ(b)* of vertices a and b

Decision: Verifier rejects If $\phi(a)) \neq \phi(b)$, otherwise Verifier repeats steps 1-3 and accepts after k iterations

Completeness and Soundness

- Completeness: if G is 3-colorable, then the honest prover uses a proper 3-coloring& the verifier always accept.

- Soundness: If G is not 3-colroable, then for all P*,
  Prob[Verifier accepts]< $(1-1/|E|)^k$ < $1/e^{|E|}$
  for $k = |E|^2$.

- Zero Knowledge: Easy to see informally, Messy to prove formally

Honest Verifier Computational ZK

Simulator S in input G=(V,E) : choose at random in advance a challenge (a,b) of the honest verifier V.

- Choose random edge (a,b) in G
- Choose colors $\phi_{a,}\phi_b$ in {0,1,2} s.t $\phi_a \neq \phi_b$ at random and for all other v ≠ a,b set ϕ_a= 2. Output simulated-view =

  (commit-transcript to $\phi(v)$ for all v , edge =(a, b),

  decommit-transcript to colors $\phi_{a,}\phi_b$)

Computational ZK: Simulation for any Verifier V*

Simulator S on input G and verifier V*: For i = 1 to $|E|^2$:

- Choose random edge (a, b) and generate commitments *com* to colors as in honest verifier simulation.
- Run V* on *com* to obtain challenge (a^*, b^*);

  if $(a^*, b^*) = (a, b)$, then output simulation as honest verifier case,

If all iterations fail, then output ⊥.

Claim: If Commitment scheme is Hiding & Binding, then

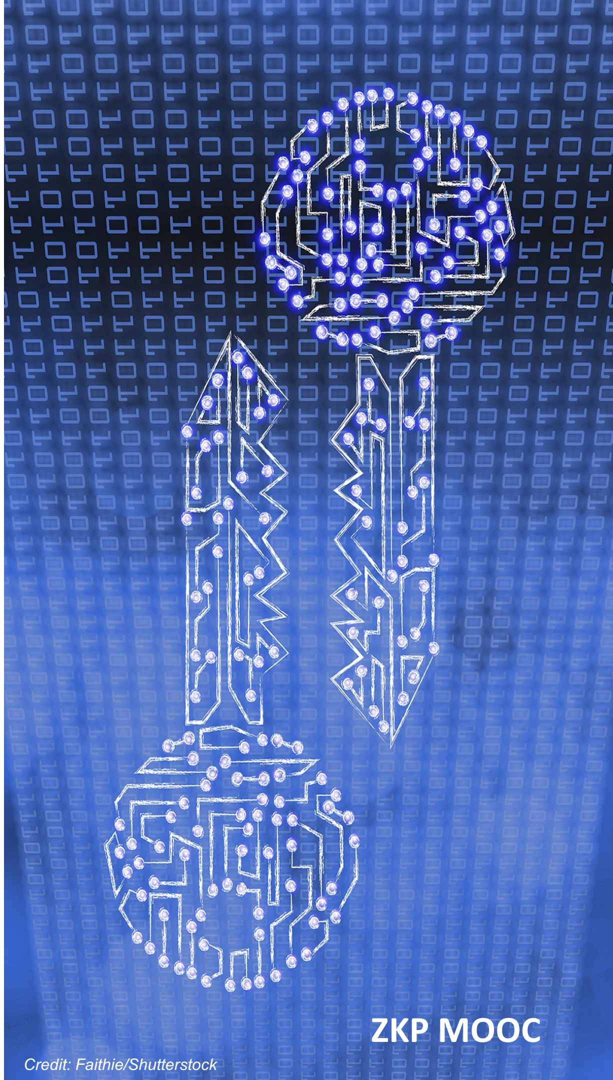
$\forall G,\pi$ (a true coloring) : prob[⊥ output]=neg(|E|) and if ⊥ is not output, then simulated-view $\approx_c$real-view

Now, we have as many CZK examples as NP-languages

- n is the product of 2 primes
- x is a square mod n
- (G_0,G_1) are isomorphic
- Any SAT Boolean Formula has satisfying assignment
- Given encrypted inputs E(x) & program PROG, y=PROG(x)
- Given encrypted inputs E(x) & encrypted program E(PROG), y=PROG(x)

} Stronger Guarantee: PZK

Applications in practice and in theory



ZKP MOOC

Protocol design applications

- Can prove relationships between m_1 and m_2 never revealing either one, only commit(m1) and commit(m2).

Examples: $m_1=m_2$, $m_1 \neq m_2$ *or more generally* $v=f(m_1, m_2)$ *for any poly-time f*

Generally: A tool to enforce honest behavior in protocols without revealing any information. Idea: protocol players sends along with each *next-msg*, a ZK proof that *next-msg*= Protocol(history h, randomness r) on history h & c=commit(r) Possible since L={$\exists r\ s.t.\ next-msg = Protocol(h,r)\ and$ c=commit(r)} in NP.

Uses for Zero Knowledge Proofs 90-onwards

Computation Delegation [Kalai, Rothblum x 2, Tromer,…]

Zero Knowledge and Nuclear Disarmament [Barak et al]

Zero Knowledge and Forensics [Naor etal]

Zcash: Bit Coin with privacy and anonymity [BenSasson, Chiesa et al]

Zero Knowledge and Verification Dilemmas in the Law [Bamberger etal]

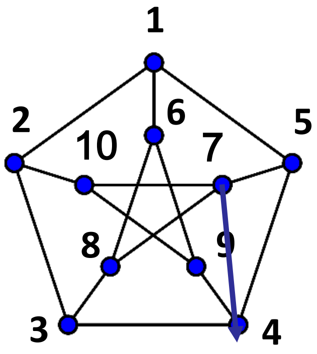
Complexity Theory: Randomized Analogue to NP

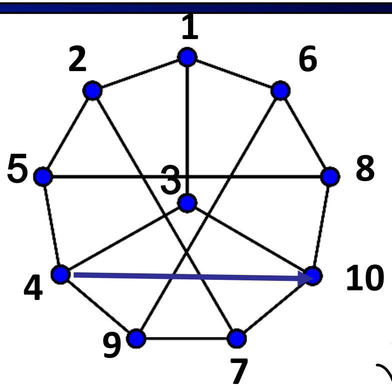
Efficiently Solvable	P	BPP (randomized poly time)
Efficiently Verifiable	NP	**IP**

Q: Is **IP** greater than NP?

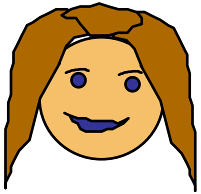
Claim: G_0 is **Not Isomorphic** to G_1 (in co-NP, not known to be in NP)

G_0

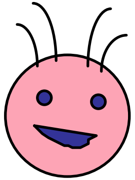




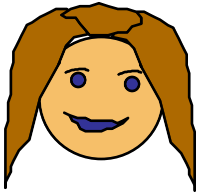
G_1



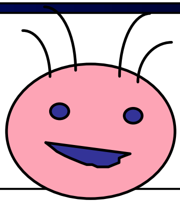
Shortest classical proof:
≈exponential n! for n vertices
But can convince with an efficient interactive proof



Graph **Non**-Isomorphism in **IP**



input: (G_0, G_1)



$H = \gamma(G_c)$ ←

flip coin $c \in \{0,1\}$
pick random γ

if H isomorphic to G_0 then b = 0, else b = 1

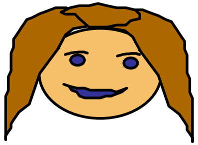
b →

Reject if $b \neq c$

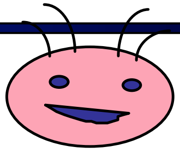
Accept after n repetitions

Claim: Completeness&Soundness hold

Graph Non-Isomorphism in **IP**



input: (G_0, G_1)



flip coin $c \in \{0,1\}$ pick random γ

$H = \gamma(G_c)$ ←

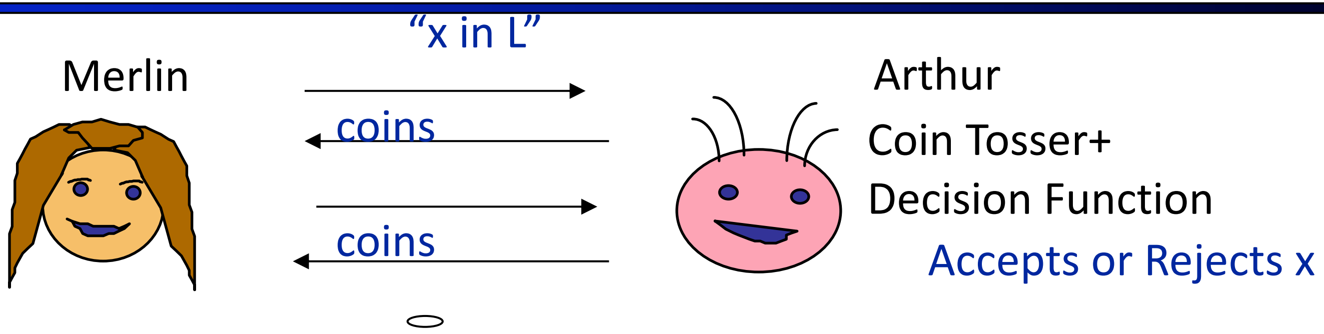
if H isomorphic to G_0 then b = 0, else b = 1

b →

Reject if $b \neq c$
Accept otherwise

Not ZK! V* can learn if graph H of its choice is isomorphic to G_0 or G_1.
Idea for fix: V proves to P in ZK that he knows an isomorphism γ

Arthur-Merlin Games [BaM85]



GNI requires verifier to keep its coins secret as in IP protocols

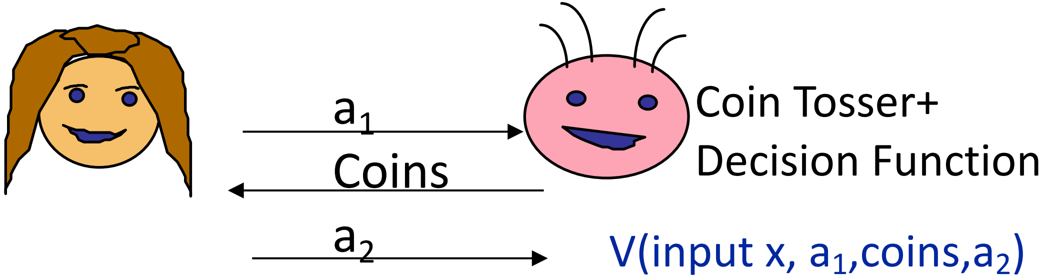
Is coin privacy necessary?

Theorem[GoldwasserSipser86]: AM (protocols with Public Coins) = IP

Idea: Merlin proves to Arthur "the set of private coin executions that would make Verifer accept" is large. Technique= prove lower bound on size of sets

AM Protocols enable "in practice" removal of interaction: the Fiat-Shamir Paradigm[FS87]

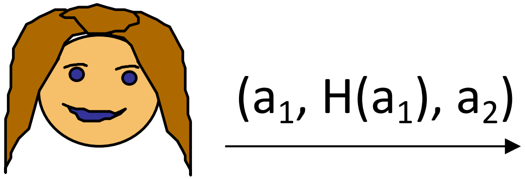
- Let $H:\{0,1\}^* \longrightarrow \{0,1\}^k$ be a cryptographic Hash function
- Can take an AM protocol



V(input x, a_1,coins,a_2)
=Accept or Rejects

- Replace by

Fiat-Shamir Heuristic:
If H is random-oracle,then
completeness&soundness hold,
Use H –hash function



V(x, a_1,H(x,a_1), a_2)
=Accept or Rejects

AM Protocols suggest “in practice” removal of interaction: the Fiat-Shamir Paradigm[FS87]

Warning: this does **NOT** mean every interactive ZK proof can transform to AM protocols and then use Fiat-Shamir heuristic,

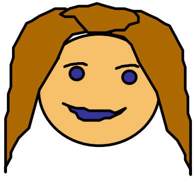
Since IP =AM transformation requires extra super-polynomial powers from Merlin

And for Fiat-Shamir heuristic to work, Prover must be computationally bounded so not to be able to invert H

Yet, many specific protocols, can benefit from this heuristic

Fiat-Shamir Heuristic:

If H is random-oracle,then completeness&soundness hold

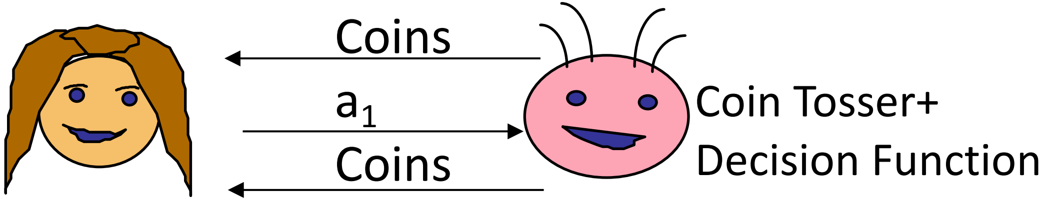


$(a_1, H(a_1), a_2)$ →

$V(x, a_1, H(x, a_1), a_2)$ =Accept or Rejects

AM Protocols suggest "in practice" removal of interaction: the Fiat-Shamir Paradigm[FS87]

- Let H:{0,1}* ⟶ {0,1}^{k} be a cryptographic Hash function
- Can take an AM protocol



Q: What if first message are coins from Arthur?

Idea(used later in course extensively):
Post first message coins as a "publicly" chosen randomness for all to see and then apply Fiat-Shamir heuristics to get non-interactive proofs

, a_2)=Accept

a_2)=Accept
or Rejects

IP: Complexity Theory Catalyst

Decoupled "Correctness" from "Knowledge of the proof"

Ask new questions about nature of proof

Questions have been asked and answered in last 30+ years leading up to current research on Provably outsourcing computation

Classically: Can Efficiently Verify

NP	✓	∃solution
Co-NP	?	0 solutions
#P	?	$2^{100}-13$ solutions
PSPACE	?	∀∃∀…∃

Can you prove more via interactive proofs?

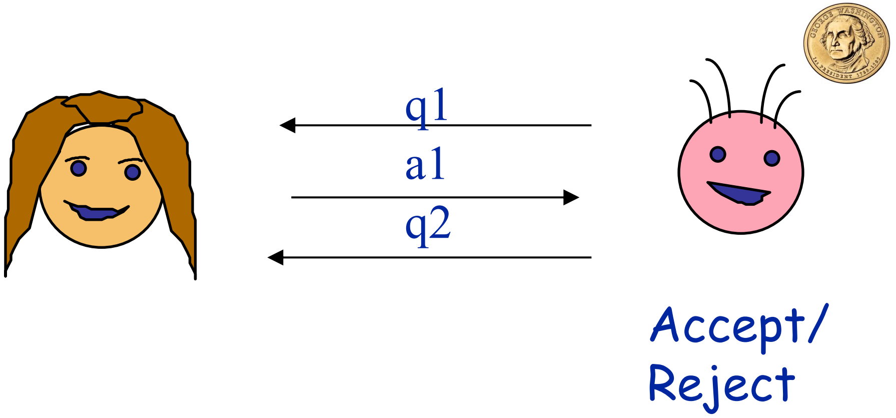
Interactively Provable= PSPACE [FortnowKarloffLundNissan89, Shamir89]

NP ✓

Co-NP ✓

#P ✓

PSPACE =IP ✓



Other Ways to define probabilistic proof systems?

The Arrival of the Second Prover (MIP)

[BenorGoldwasserKilianWigderson88]

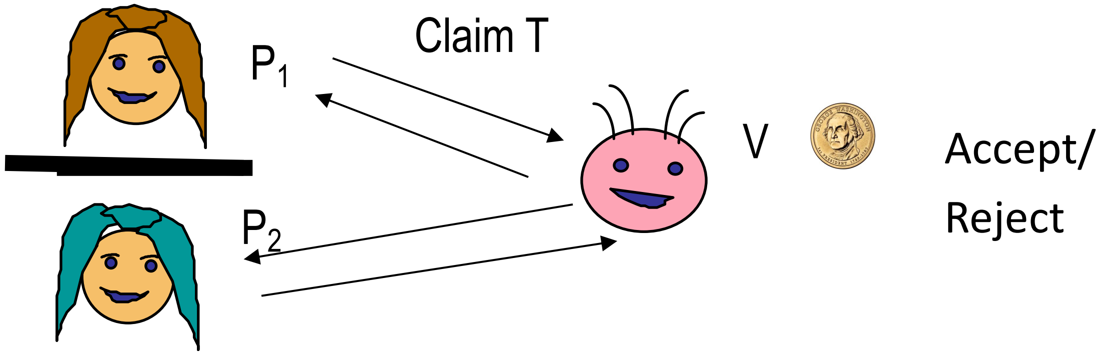
NP ✓

Co-NP ✓

#P ✓

PSPACE ✓

Unconditionally: NP =PZK



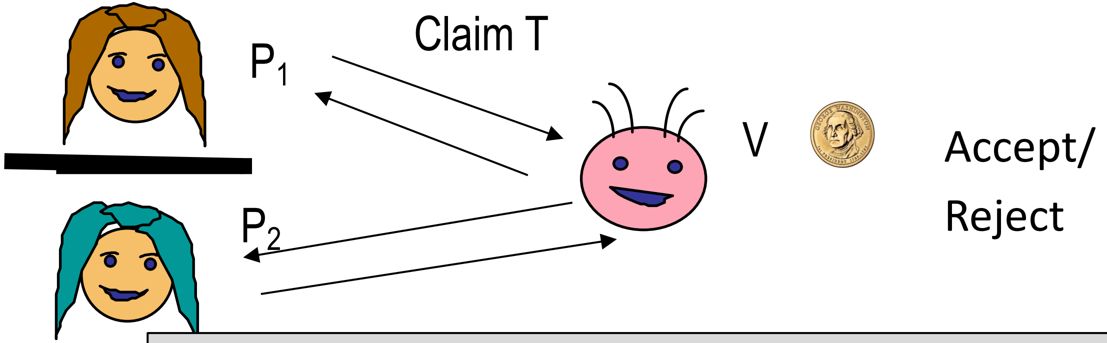
Could two prove more than one?

Intuition: Can check **consistency,**
Verifier catches provers if deviate

The Second Prover is a Game Changer (MIP)

- NP ✓
- Co-NP ✓
- #P ✓
- PSPACE ✓
- NEXPTIME ✓

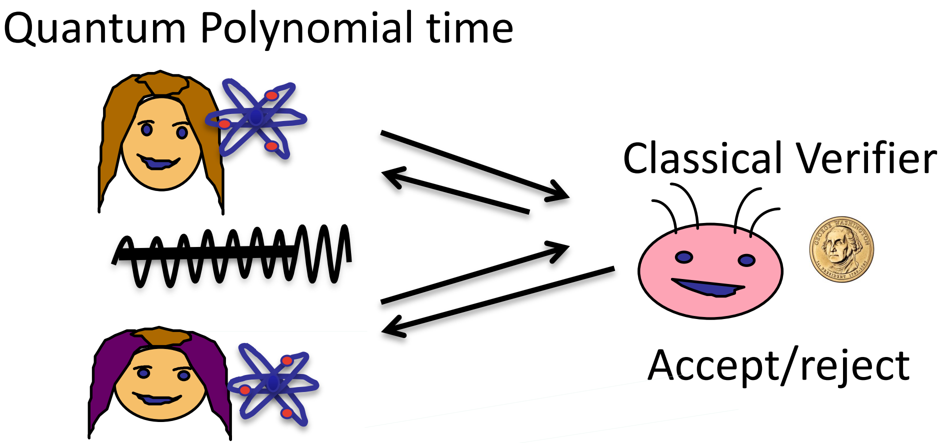
[BabaiFortnowLund90]



PCP theorem [FGLSS, AS, **ALMSS91**]:
NP statements can be verified with high prob. by only reading a **constant** number of bits of the proof
→NP-Hardness of Approximation Problems

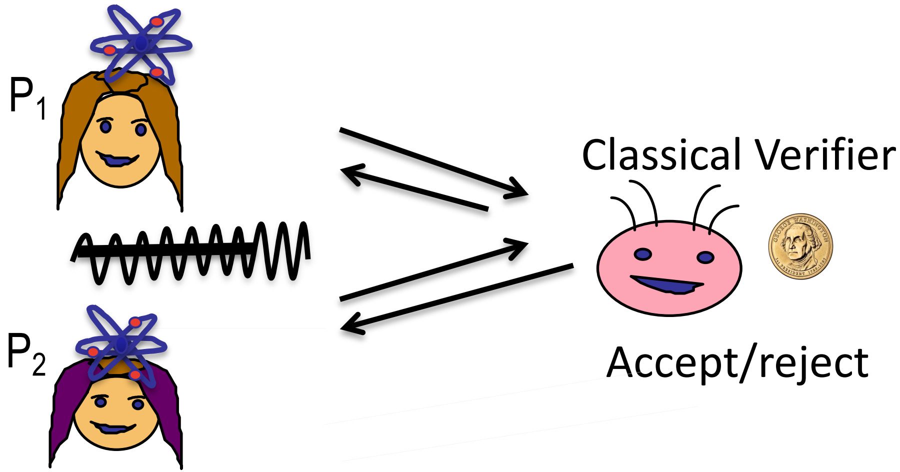
Impact on Quantum Computing

Q: Can the correctness of a Quantum polynomial time computation be checked by a classical verifier?



Theorem[ReichardtUngerVazirani13]:
A classical Verifier can verify the computation of two entangled but non-communicating poly-time quantum algorithms

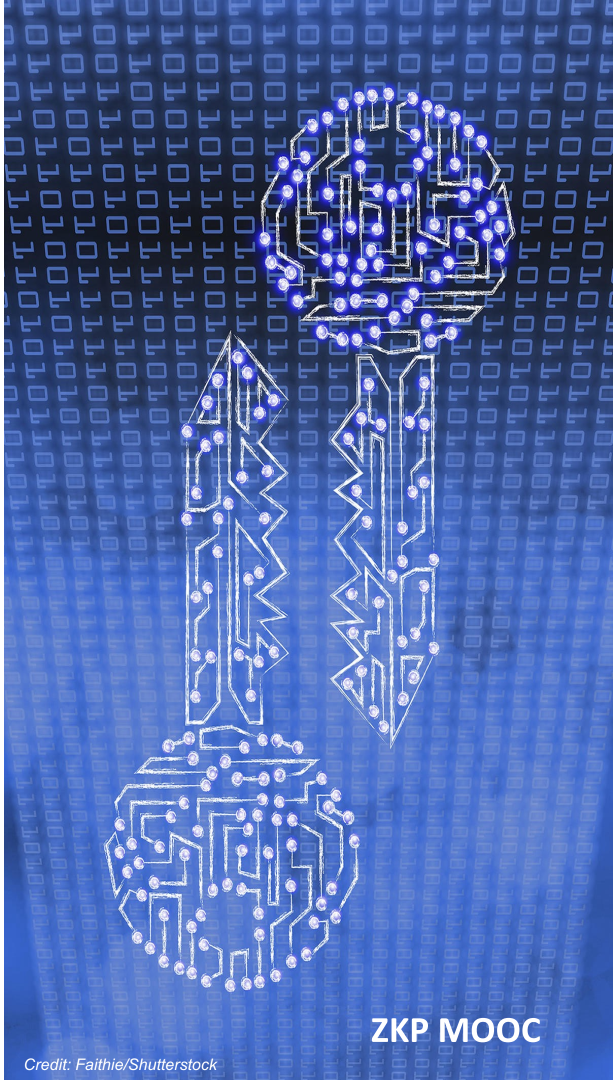
Quantum MIP is All Powerful



2020: **MIP* = Recursively Enumerable Languages**

[Ji, Natarajan,Vidick, Wright, Yuen]

Aside:
The Resistance



1983–1985 (The Resistance)

We … me) to be
appl … ural exten-
sion … ontext, as
neec

π w … $\subseteq D_\pi$ the
subs … computa-
tion

Inte

An Interactive Non-deterministic Turing Machine (INDTM) is formed by two communicating modules: a *guesser* G and a *checker* C. The checker is a probabilistic Turing Machine. G and C share a read-only tape in which the input is

1983–1985 (The Resistance)

Revised Version, Dec 8, 198

The Information Content of Proof Systems

1. Introduct

1.1 The goal

The goal
contained in a
Example
associated wit
graphs. A stan
sist in exhibiti
than *G* is Han
sist of all weig
than B. Simil

1983–1985 (The Resistance)

Let me show you how to do it!
Dec 7, 1984

ocols

1. Intr

Co ... In traditional
computa ... municate as much
knowled ... dered good friends,
no one ... on with respect to
cryptogra ... em at all communi-
cating the knowledge efficiently, but the whole problem is making sure not *too much* knowledge
has been communicated.

1985 (The Acceptance)

We are very happy to inform you that your paper
"*The Knowledge Complexity of Interactive Proof Systems*"
has been selected for presentation at the 17th Symposium on Theory of Computing

Broader Lessons

- Pay attention to good ideas
- It may take a long time >30 years to go from the basic idea to impact