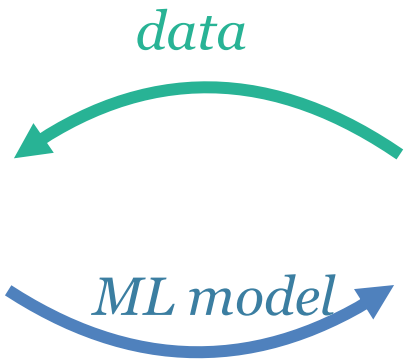
Lecture 9

Verification

Machine Learning as a Service

Service Provider



Client

MLaaS: Amazon SageMaker/AWS, Microsoft Azure, Startups…

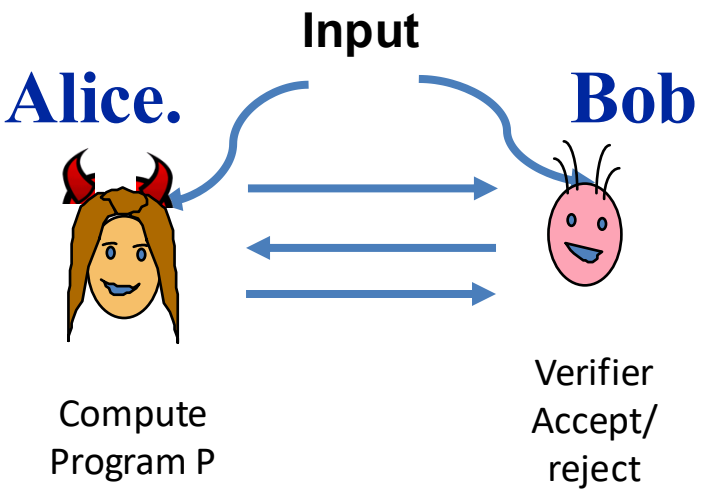
Verifying Model Properties, how?

Can we **verify** properties of the model h:
Accuracy/
Correctness/
Robustness/
Fairness
Safety
Satisfies Regulations

cheaply (**not** retraining) using

- Fewer data samples
- Lower quality data?
- Efficient Time/Memory
- Black box access to h

1980's Theory on Verifiable Computing



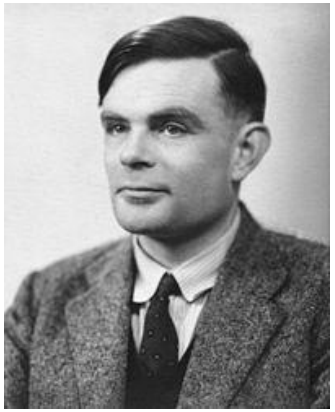
Application to verifiable computing

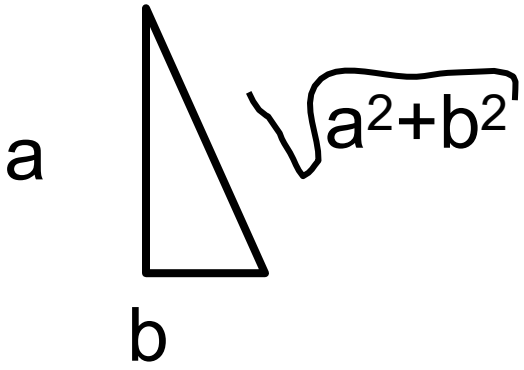
Many Models &Techniques

- Interactive Proofs & arguments for Program Delegation
- Zero Knowledge Interactive Proofs & arguments
- Non-Interactive Zero Knowledge Proofs & arguments, SNARKs, SNARGs,
- Multi Prover Proofs
- Debates
- Probabilistically Checkable Proofs

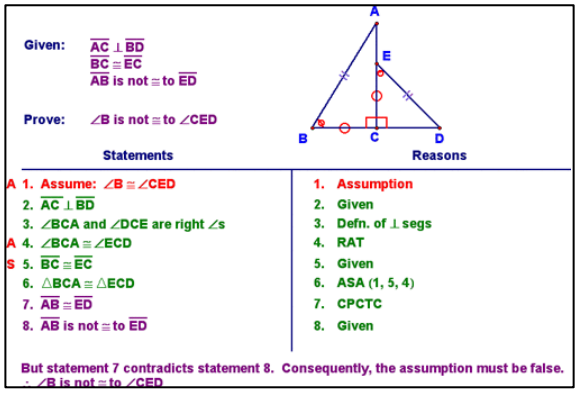
Classical Proofs

. . .





. . .



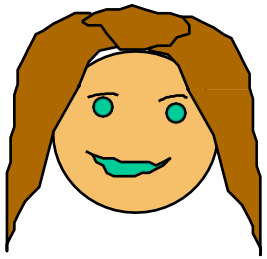
. . .

Prime-
Number Thm

Proofs

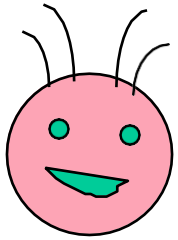
Claim

Prover



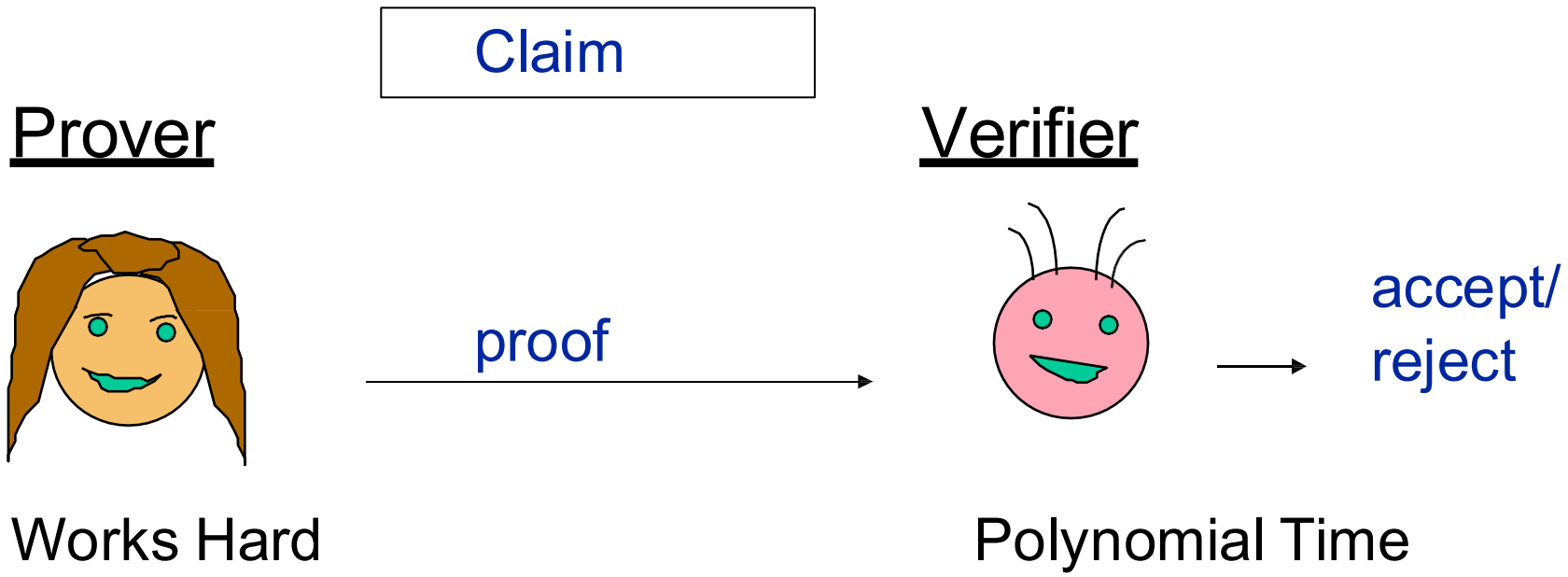
proof

Verifier

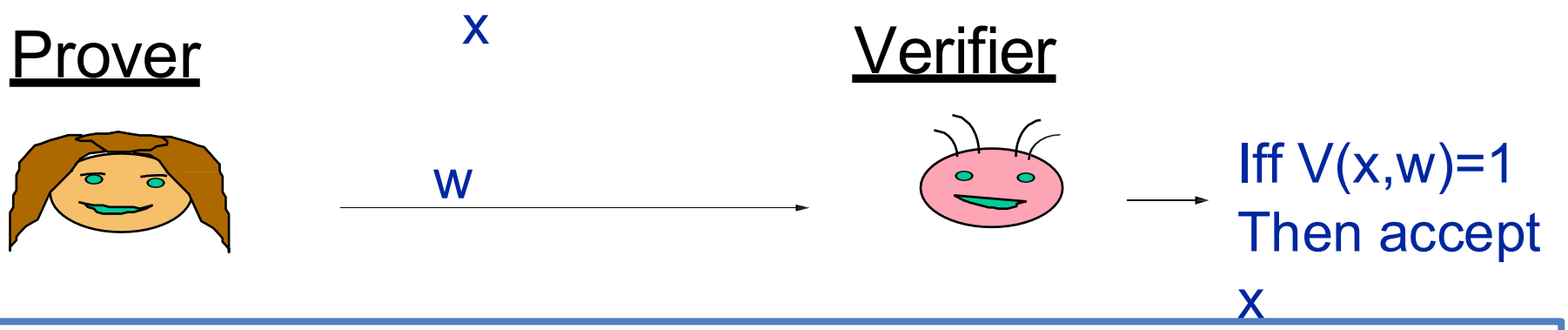


accept/
reject

Efficiently Verifiable Proofs (NP)



Efficiently Verifiable Proofs: $\mathcal{NP}$



Def: D is an $\mathcal{NP}$-language if there is a **poly-time** verifier V function such that

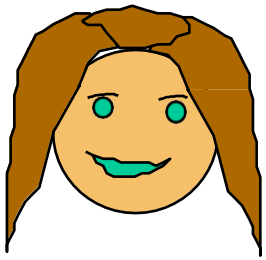
- **Completeness**:

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

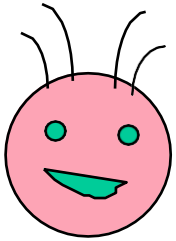
- **Soundness:**

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

Example: N is a product of 2 large primes



p,q



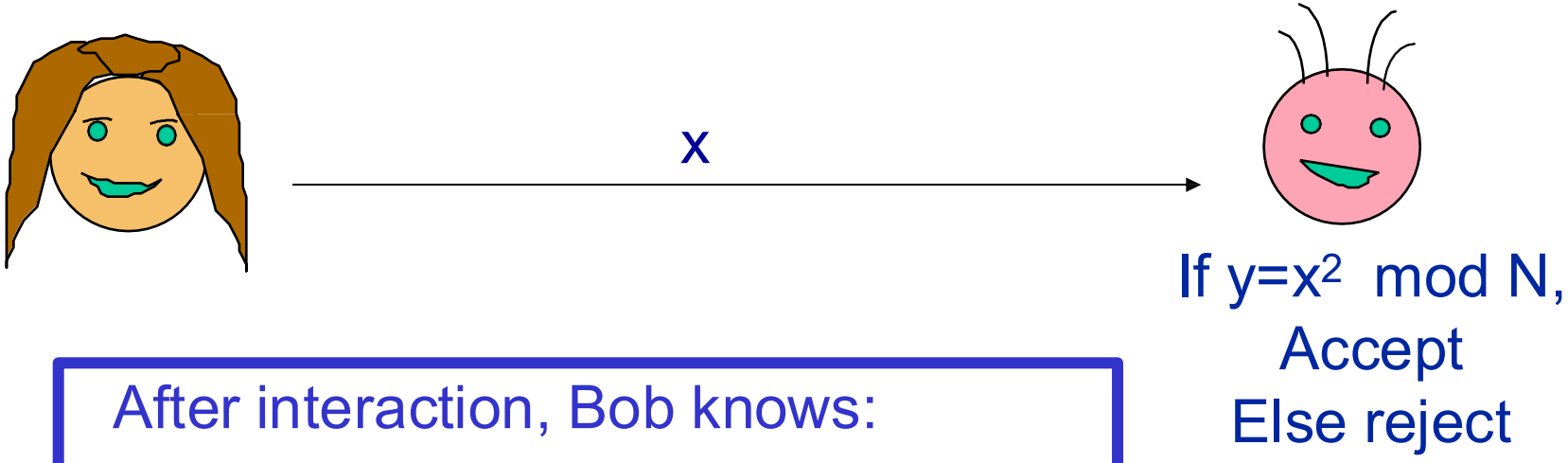
If N=pq and p& q are prime, then accept
Else reject

After interaction, Bob knows:

1) N is product of 2 primes

2) Also the factors of N

Example: y is a quadratic residue mod N (i.e $y=x^2 \bmod N$)

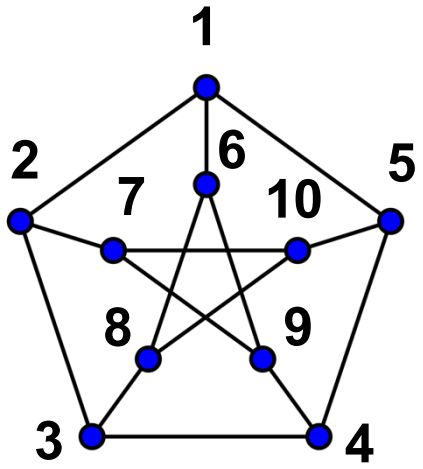


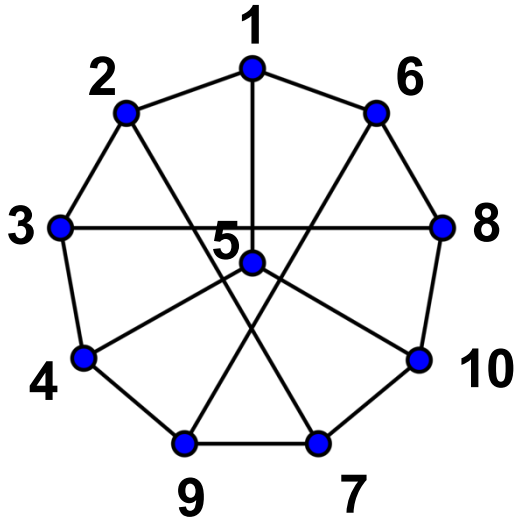
After interaction, Bob knows:

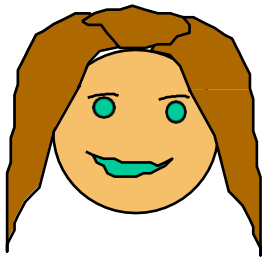
1) y is a quadratic residue mod

2) Also a square root of y mod N

Theorem: Graphs G_0 and G_1 are isomorphic.



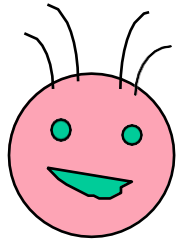




Prover

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

the isomorphism

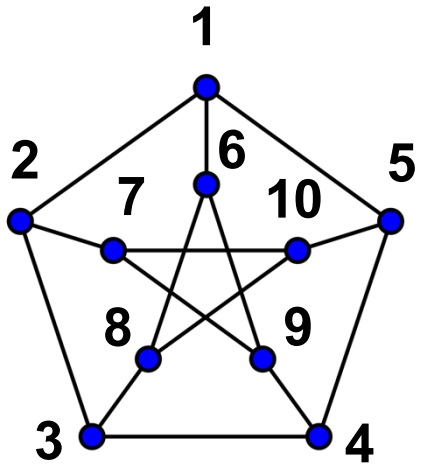


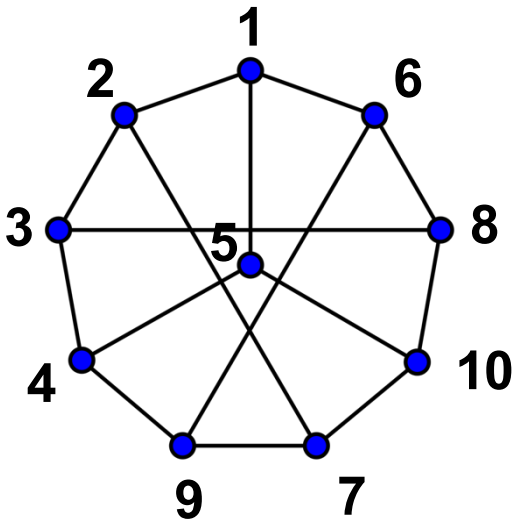
Verifier

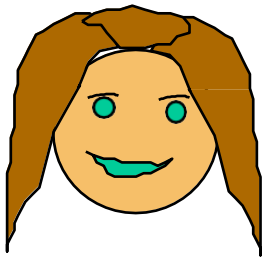
Check $\forall i, j$:

$\big(\pi(i), \pi(j)\big) \in E_1$ iff $(i, j) \in E_0$.

Theorem: Graphs G_0 and G_1 are isomorphic.

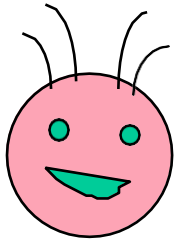






Prover

π



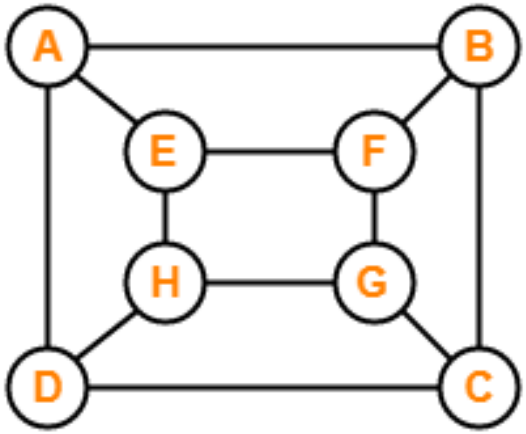
Verifier

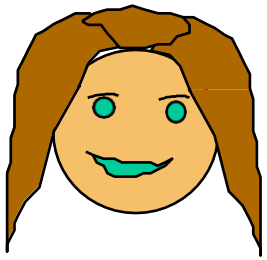
After interaction, Bob knows:

1) G_0 is isomorphic to G_1

2) Also the isomorphism π

Theorem: Graphs G has a Hamiltonian cycle.

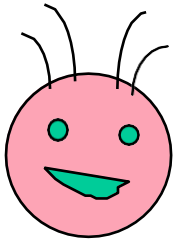




Prover

Proof = Hamiltonian cycle

$(v_0, \ldots, v_{N-1})$

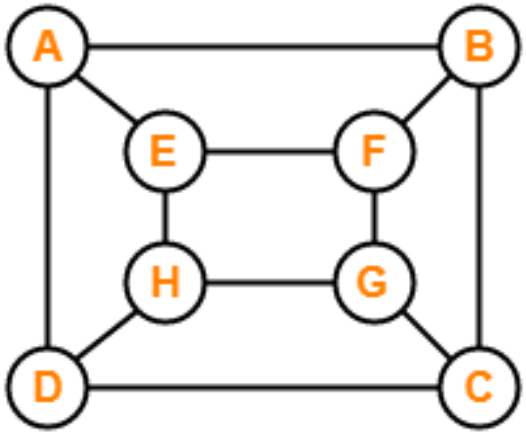


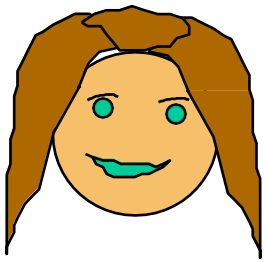
Verifier

Check $\forall i$:

$(v_i, v_{i+1 \bmod N}) \in E$

Theorem: Graphs G has a Hamiltonian cycle.

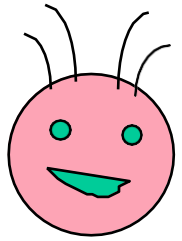




Prover

Hamiltonian cycle

$(v_1, \ldots, v_N)$



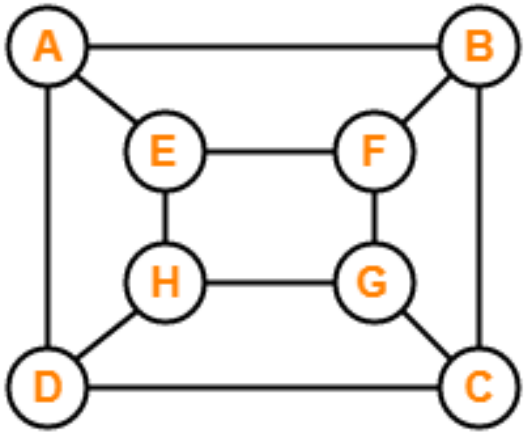
Verifier

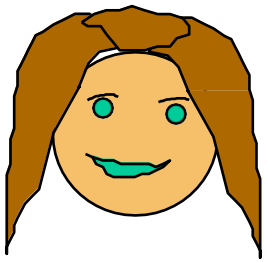
After interaction, Bob the Verifier knows:

1) G ***has*** a Hamiltonian cycle.

2) **Also**, the Hamiltonian cycle itself.

Theorem: Graphs G has a Hamiltonian cycle.

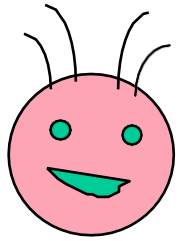




Prover

Hamiltonian cycle

$(v_1, \ldots, v_N)$

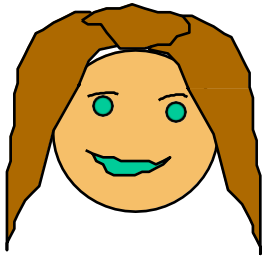


Verifier

NP-Complete Problem:

Every one of the other problems can be reduced to it

Is there any other way**?**



I will not give you
A proof, but I will prove to you
that I **could** provide one.

HOW?

Two New Ingredients

Interactive and Probabilistic Proofs

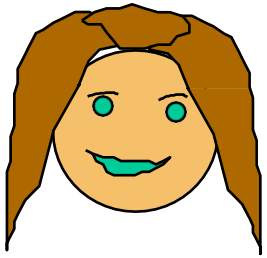
Non-trivial interaction: rather than "reading" proof, verifier engages in an non-trivial **interaction** with the **prover.**

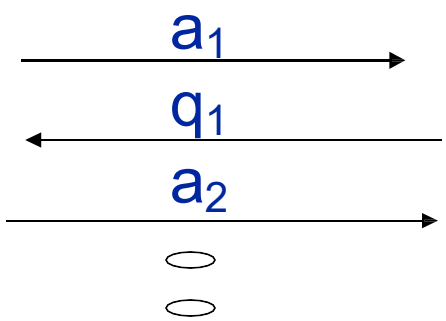
Randomness: verifier is randomized (tosses coins as a primitive operation), and can err with some small probability

Interactive Proofs[GMR85]

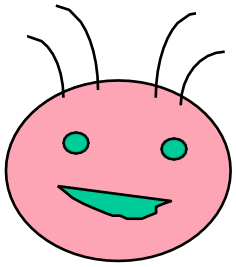
Statement: T

Prover P





Verifier V



Probabilistic Polynomial time algorithm

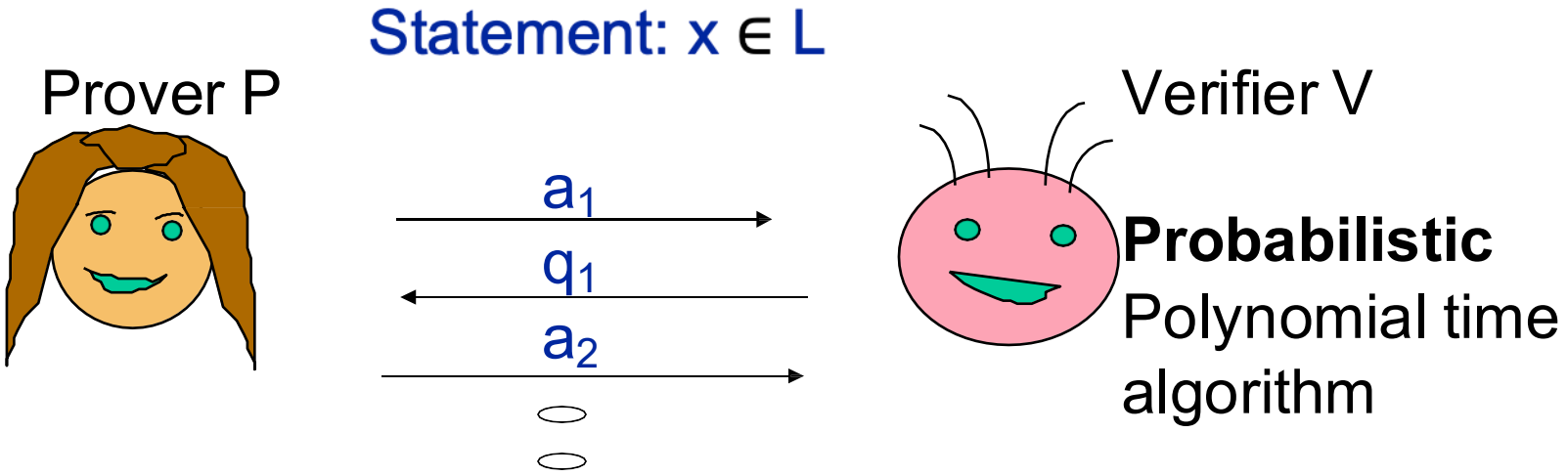
Accepts /Rejects

(P,V) is an interactive proof system for T if

Completeness: if T is true, then V will always accept

Soundness: if T is false, then regardless of prover P*strategy, V will reject with overwhelming probability

Interactive Proofs for Language L[GMR85]



(P,V) is an interactive proof system for L if

Completeness: if $x \in L$, then $Prob[(P,V)[x]= accept]=1$

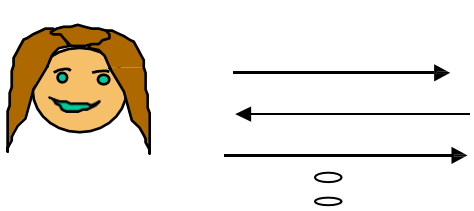
Soundness: if $x \notin L$, then $\forall P^*$

$Prob[(P^*,V)[x]=accept]=neg(|x|)$

Remarks: Interactive Proofs

Prover P

Verifier V



Probabilistic
Polynomial time

Accepts
/Rejects

P and V are a pair of **interactive Algorithms**, each having private inputs and private coins
as well as a common public input.

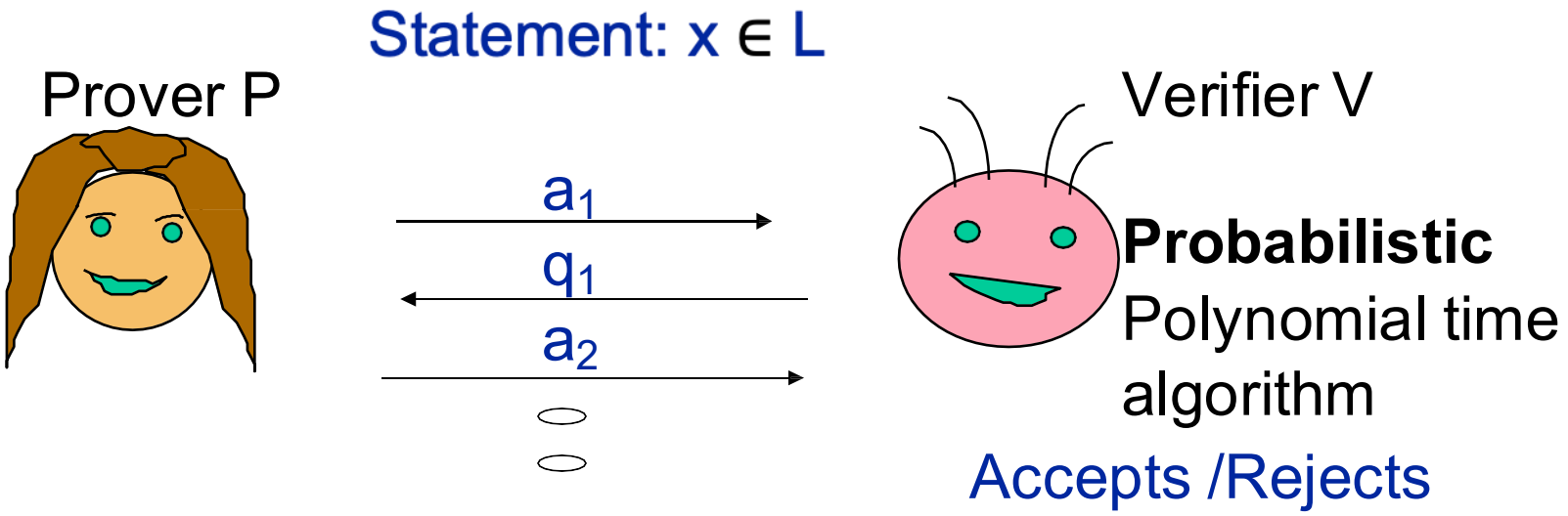
V additionally must run in polynomial time

(P,V) satisfy completeness c(x) & soundness s(x)

- if $x \in L$, Prob((P,V)[x]= accepts)> c(x)
- If $x \notin L$, $\forall P^*$, Prob[(P*,V)[x]=accepts]<s(x)

Suffice to require: c(x)=2/3 and s(x)=1/3

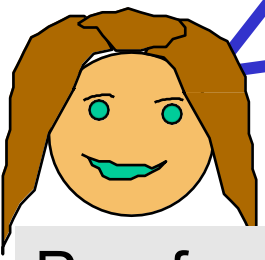
Class IP



IP = {L s.t. there exists (P,V) interactive proof system for L with completeness c(x)=2/3
and soundness s(x)=1/3}

Can you prove more with coins and interactions?
Is IP greater than NP?

Graph ISO



I will produce a random graph H for which

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

OR

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

Hence, there is an isomorphism a from G_0 to G_1 directly

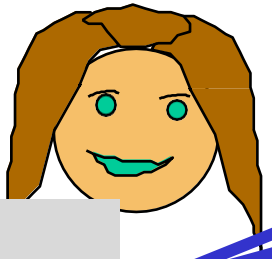
YOU randomly choose if I should demonstrate my ability to do #1 or #2.

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

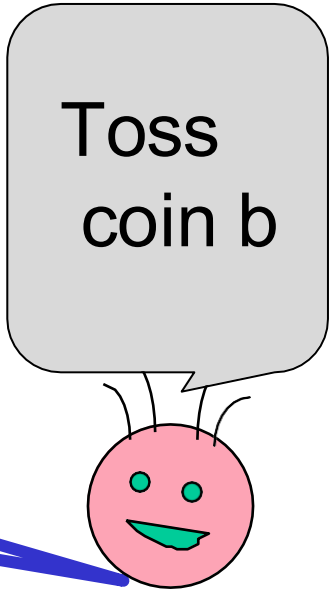
An Interactive Proof

REPEAT K INDEPDENT TIMES.



Graph H

b



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

If b=0: send γ_0

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

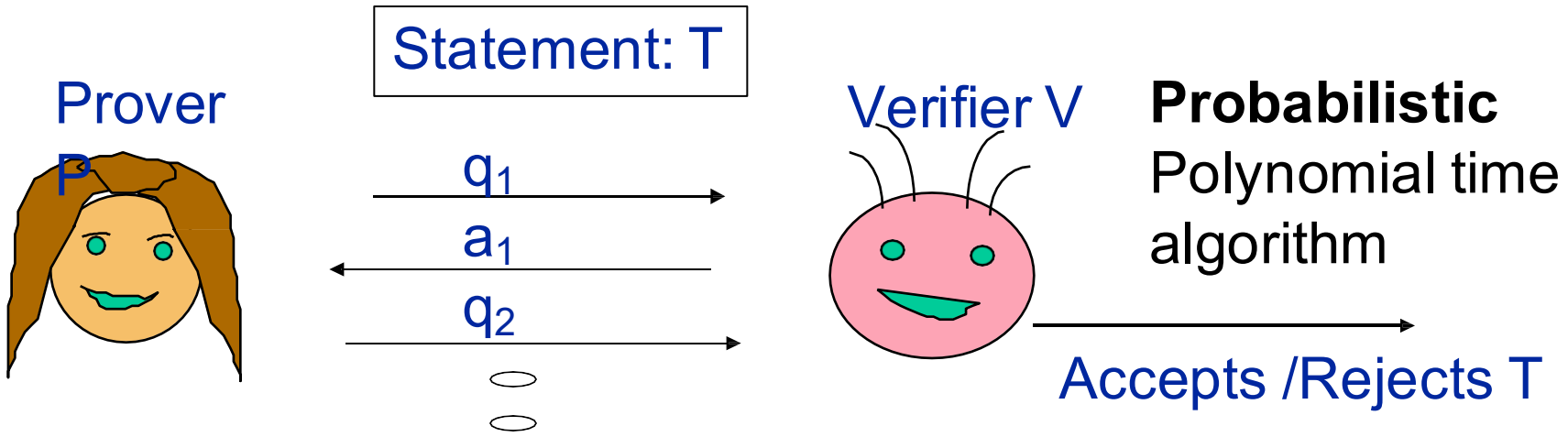
Claims:

(1) Statement true ⇒ can answer correctly for b= 0 and 1

(2) Statement false ⇒ $prob_b$(catch a mistake) = $1-1/2^k$

(3) Zero Knowledge (to be defined)

Zero Knowledge Interactive Proofs



After interactive proof, V "knows":

- T is true (or $x \in L$)
- A **view** of interaction (=transcript + coins V tossed)

P gives Zero- Knowledge to V: when T is true,

the **view** gives V nothing he couldn't have obtained on his own without interacting

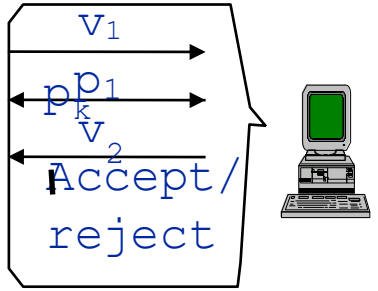
How Do we Capture Getting "Nothing Extra"(when T is true)

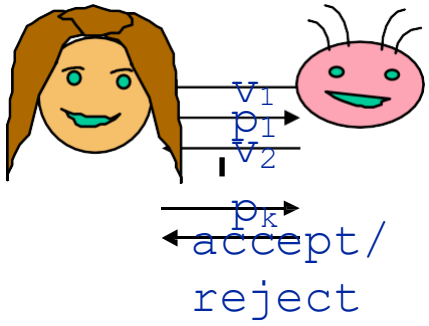
If: the verifier's view can be efficiently simulated so that `simulated views' and `real views' are indistinguishable in polynomial time.

??

SIMULATED VIEWS

REAL VIEWS





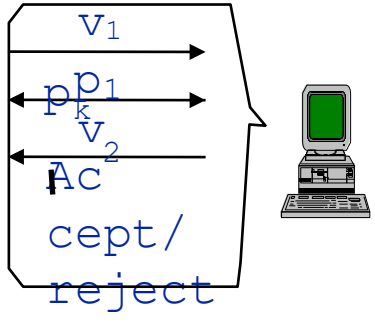
Polynomial time distinguisher

Perfect Zero Knowledge (when T is true)

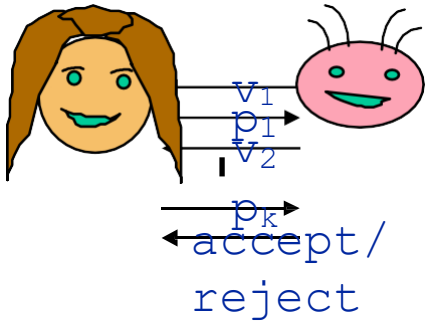
If: the verifier's view can be efficiently simulated so that `Simulated views' = `real views'

??

SIMULATED VIEWS



REAL VIEWS



The distinguisher

Any Algorithm

Formal Definition: Perfect Zero-Knowledge

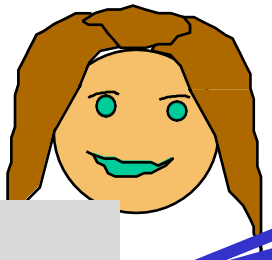
For a given P and V on input x, define probability space $View_{(P,V)}(x)$= $\{(q_1,a_1,q_2,a_2,\ldots,\text{coins of V})\}$ (over coins of V and P)

(P,V) is **honest** verifier perfect zero-knowledge for L if:

∃SIM a polynomial time randomized algorithm s.t. ∀x in L, $View_{(P,V)}(x)$ = SIM(x)

Will allow SIM Expected polynomial time

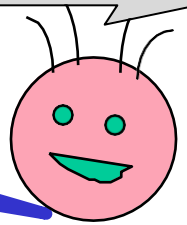
Recall: Isomorphism Example



Graph H

b

Toss coin b



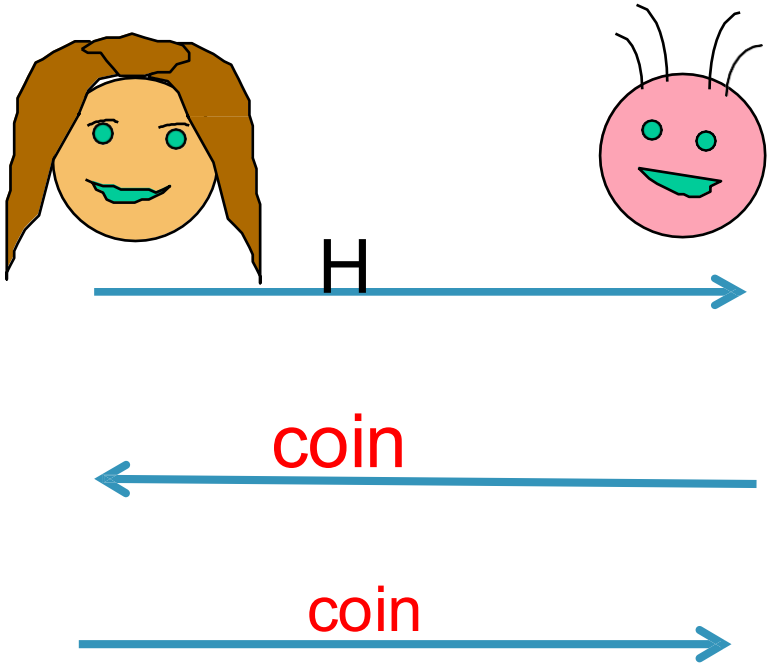
Choose random y_0 permutation of vertices of G_0. Set $H=\gamma_0(G_0)$

If b=0: send γ_0

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

View of Bob=
{(H; b; random isomorphism from G_b to H}

Zero Knowledge



SIMULATOR M:

- toss coin
- If coin=head:
  choose random γ_0
  set H= γ_0 (G_0)
- If coin=tail choose random γ_1
  set H= γ_1 (G_1)

View of Bob=
{(H, coin, random isomorphism of G_b to H}

What if V is not honest: Perfect Zero-Knowledge (Final def)

For a given P and V on input x, define probability space $View_{(P,V)}(x)= \{(q_1,a_1,q_2,a_2,\ldots,coins)\}$ (over coins of V and P)

(P,V) is honest verifier perfect zero-knowledge for L if:

∃SIM expected polynomial time randomized algorithm s.t. ∀x in L, $View_{(P,V)}(x) = SIM(x)$

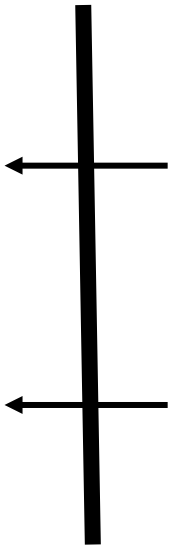
(P,V) is **perfect zero-knowledge** for L if :

∃SIM expected polynomial time randomized algorithm s.t **∀PPT V*** ∀x in L, $View_{(P,V^*)}(x) = SIM(x)$

Prover Gives **Perfect** Zero Knowledge

- If: we can efficiently simulate the view of any verifier s.t. `Simulated views' = `real verifier" for any poly time verifier

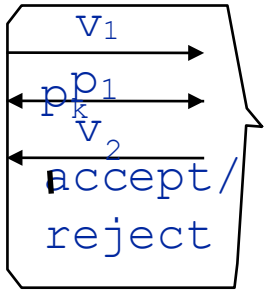
??

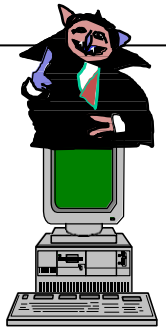


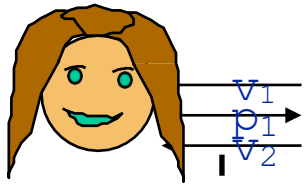
SIM

II

REAL



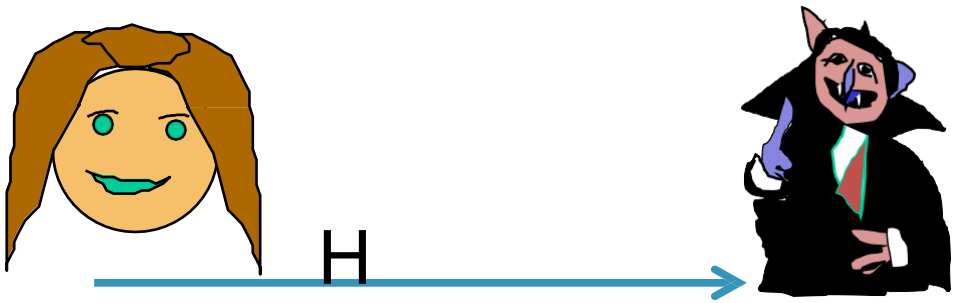




The observer

Any Algorithm

Zero Knowledge Proof that G_1 isomorphic to G_2



if coin=coin. answer
Else abort and try again

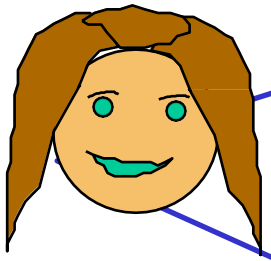
SIMULATOR SIM:

1. toss coin

2. If coin=head:
   choose random y_0
   set H= γ_0 (G_0)
   If coin=tail
   choose random γ_1
   set H= γ_1(G_{21}
3. Feed H to V*=
4. If V* outputs
   coin==coin
   output (H, coin, γ_{coin})
   Else abort and
   goto 1 again.

Claim:
prob[coin=coin] = ½,
Expected [number of repetitions of SIM] = 2.
For k repetitions, SIM expected trials = 2k

Claim: $y = x^2 \mod N$ is solvable



Repeat 100 times ns

$z = [r^2 \mod n]$

$zy = [(rx)^2 \mod n]$

- If I gave you solutions to both, that is r and rx, you would be convinced that the claim is true but also know x
- Instead, I will give you a solution to only one equation, either r or rx but you can choose which!

Choose $1<r<n$ at random

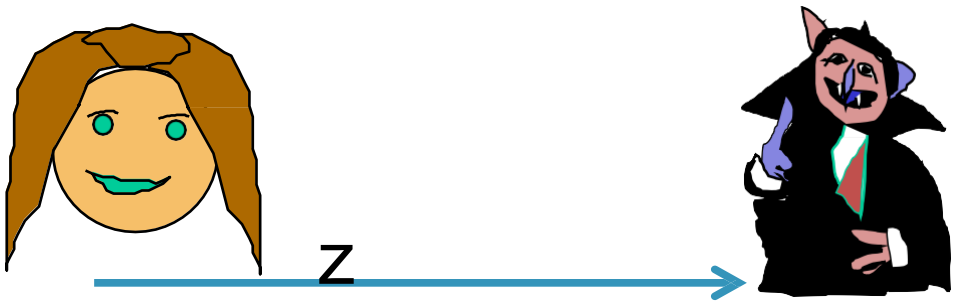
$1-(1/2)^{100}$

Flip a b=  to choose an equation

Gives a solution to the equation requested

Accepts claim only if gets correct solution

Zero Knowledge Proof that $Y=x^2 \bmod N$



coin

if coin ≠ coin abort

If coin=coin, send r

SIMULATOR SIM:

1. toss coin

2. If coin=head:
   choose random r
   set $z=r^2 \bmod n$

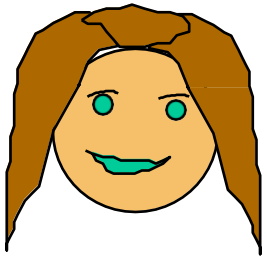
If coin=tail

choose random r

set $z=(ry^{-1})^2 \bmod n$

3. Feed z to V*= 
4. If V*(z) outputs coin≠coin
   abort and goto 1
   else for coin=head
   output(H, coin, r) &
   for coin=tail,
   output(H, coin, r)

Zero Knowledge Proof that $Y=x^2 \bmod N$



z

coin

if coin ≠ coin abort

If coin=coin, send r

Claim:

prob[coin=coin] = ½,

Expected [number of repetitions of M] = 2. For k repetitions, M expected trials = 2k

SIMULATOR SIM:

1. toss coin

2. If coin=head:
   choose random r
   set $z=r^2 \bmod n$

If coin=tail

choose random r

set $z=(ry^{-1})^2 \bmod n$

3. Feed z to V*= 
4. If V*(z) outputs coin≠coin
   abort and goto 1
   else for coin=head
   output(H, coin, r) &
   for coin=tail,
   output(H, coin, r)

SIM: Expected Polynomial Time

- Analysis can be confusing
- Instead can change def to allow
  - SIM(x) to output $\perp$ with probability at most 1/2 and require
  - View (x)= SIM(x) to be conditioned on the event that M(x) does not output $\perp$
  - 1/2 can be relaxed to neg(x)

What Made it possible?

Randomness

- 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

Actually, Alice seems to have proved more: that she actually "knows" the isomorphism (square root)

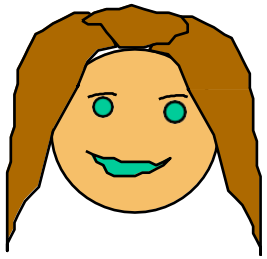
Let V be polynomial time relation. Let (x,w) ∈ V
V defines Language L_V= {x|$\exists w\ s.\ t.\ V(x,w) = 1$}.

We say that (P,V) is a **proof of knowledge** for L_V
[or that P on x knows w] *if:*

∃an extractor algorithm E s.t. for all x
E^P(x) outputs *w* *in expected polynomial time*

E^P can call to P on the same randomness in multiple iterations,
is is called the rewinding technique

ZKPOK that Prover knows an isomorphism from G_1 to G_2



Extractor Algorithm

H

Extractor :

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

Computational Zero-Knowledge

(P,V) is **honest** verifier perfect zero-knowledge for L if:
∃SIM an expected polynomial time randomized algorithm s.t. ∀x in L, $View_{(P,V)}(x) \approx_c SIM(x)$

Relaxed Perfect to "indistinguishable" by any observer who runs In probabilistic polynomial time

(P,V) is computational zero-knowledge for L if :
∃SIM an expected polynomial time randomized algorithm s.t. ∀x in L, PPT V* ,
$View_{(P,V^*)}(x) \approx_c SIM(x)$

Zero Knowledge for all of NP

Theorem: If one-way functions exist, then every problem in NP has a computational zero knowledge interactive proofs

- To prove the theorem, should we construct ZK proof for every **NP** language?

- No. Enough to show zero knowledge interactive proof for **one NP complete problem**

Why is that enough?

Take NP-complete problem

3COLOR = all graphs which can be colored with 3 colors
s.t for for all edges (u,v) color(u) ≠color(v)

NP Completeness Let L in NP.

Every instance x is polynomial time reducible to G_x

$x \in L$ ⇒ G_x is 3 colorable

$x \notin L$ ⇒ G_x is not 3 colorable

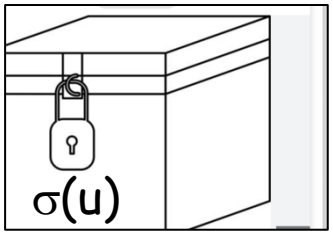
A ZK proof for G_x
Implies ZK for x .

Physical Intuition for Protocol

On common input graph G =(V,E) and

Provers private input coloring $\alpha: V \longrightarrow \{0,1,2\}$

- P picks a random permutation π of the colors {0,1,2}& color the graph with coloring $\sigma=\pi(\alpha)$. It hides the color $\sigma(u)$ of each vertex inside a locked box



- V Select a random edge (u,v)
- P opens boxes corresponding to u and v

- V accepts if and only if $\sigma(u) \neq \sigma(v)$

  [colors are different]

Completeness and Soundness

- Completeness: if prover uses a proper 3-coloring, the verifier will accept.

- Soundness: Let $k = |E|^2$

  If G is not 3-colroable, then for all P*

  Prob[(P*,V)(G) accepts]<1- 1/|E|

  Repeat k times where $k=|E|^2$

  Soundness Prob[(P*,V)(G) accepts]<

  $(1-1/|E|)^{k.} < 1/e^{|E|}$

Commitment Scheme: Digital analogue of locked boxes

- An efficient two-stage protocol between a sender and receiver on security parameter 1^k s.t.:

- commit stage: sender has private input b

  At the end of the commit stage
  - both parties hold output com (called the commitment)
  - sender holds a private output dec (called the de-commitment)

- reveal stage: sender sends the pair (dec' , b') to receiver.

  Receiver accepts or rejects (com, dec, b)

Hiding and Binding Properties

Commitment schemes satisfy two properties:

- **Hiding**: Given comm sent by sender, receiver should not be able to b and b' for $b \neq b'$

- **Binding**: S*ender* can not produce two tuples (comm, dec, b) and (comm, dec, b') that the receiver will accept both with high probability if $b \neq b'$.

Implementation: send computationally secure c=Enc(r,b) to commit to b, and to reveal send r

ZK interactive proof for G3COL

On common input graph $G=(V,E)$ and private prover input coloring $\alpha: V \rightarrow \{0,1,2\}$

- P -> V: Pick a random permutation π of the coloring & color the graph with coloring $\pi(\alpha(v))$. Send commitments $Enc(r_v, \pi(\alpha(v)))$ $\forall$ vertex v.
- V -> P: Select a random edge (u,v) and send it
- P -> V: reveal colors of u and v committed to by releasing r_u and r_v
- If *color u = color v,* V rejects,

  otherwise repeat

  V accepts after k iterations.

Simulation for any Verifier V*

Simulator SIM on input G=(V,E) and verifier

V*: Fix random tape ω for V*

For i = 1 to $|E|^2$:

- Choose random edge (a, b) and permutation π of the colors, generate vector com = Enc(r_v, $\alpha(\pi(v))$) as in honest verifier simulation.
- Run V*(com; ω) to obtain challenge ($a*$, $b*$); if ($a*$, $b*$) = (a,b), then output transcript= (Enc(r_v, $\alpha(\pi(v))$) $\forall v$; (a, b); r_a, r_b)

If all iterations fail, output $\perp$.

Theorem: If Enc is computationally secure with respect to non-uniform adversaries, then

Claim 1: $\forall G$, α (a true coloring) : prob[$\perp$ output]=neg(|E|)

Claim 2 :if $\perp$ is not output, then simulated-view $\approx_c$real-view