CIS 5560

Cryptography
Lecture 23

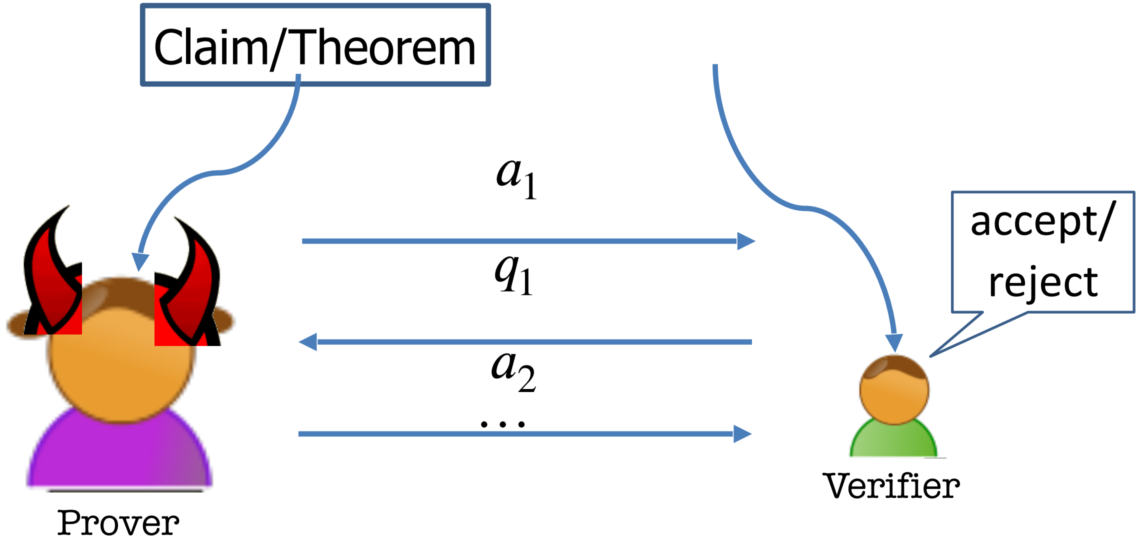
Course website:
pratyushmishra.com/classes/cis-5560-s25/

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Recap of Last Lecture

- Malicious-verifier/“standard” ZK
  - ZKPs for GI and for QR achieve standard ZK
- ZKP for 3-coloring

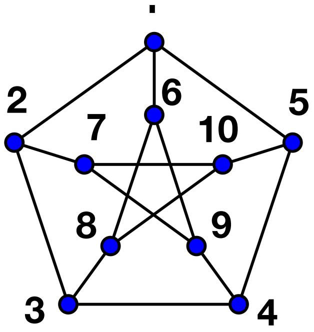
Interactive Proofs for a Language $\mathscr{L}$



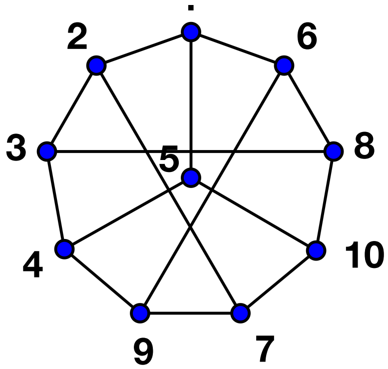
Def: $\mathscr{L}$ is an IP-language if there is a unbounded P and **probabilistic poly-time** verifier V where

- **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.

IP for Graph Isomorphism



Graph G



Graph H

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

Prover

$K = \rho(G)$

where ρ is a random permutation

Verifier

random challenge bit b

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

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

Old: Honest-Verifier ZK

Claim: The GI protocol is honest-verifier zero knowledge.

Simulator S works as follows:

1. First pick a random bit b.
2. Sample random permutation ϕ.
3. Compute $K = \phi(G_b)$.
4. output (K, b, ϕ).

$view_V(P, V)$:
(K, b, ϕ)

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

Now: Malicious Verifier ZK

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

Simulator S works as follows:

1. First set $K = \phi(G_b)$ for a random ϕ and b and feed K to V^*.

2. Let $b' = V^*(s)$.

3. If $b' = b$, output (K, b, ϕ) and stop.

4. Otherwise, go back to step 1 and repeat. (also called “rewinding”).

Do all NP languages have

Winner of 2024 Turing Award!

Nevertheless, today, we will sho

<u>Theorem</u> [Goldreich-Micali-Wigderson'87] Assuming one-way functions exist, all of NP has computational zero-knowledge proofs.

This theorem is amazing: it tells us that everything that can be proved (in the sense of Euclid) can be proved in zero knowledge!

R1CS

An rank-1 constraint system (R1CS) is a generalization of arithmetic circuits

$$(F := (\mathbb{F}, n \in \mathbb{N}, A, B, C), x, w)$$

$$z := \begin{bmatrix} x \\ w \end{bmatrix} \qquad n\left\{\overbrace{\begin{bmatrix} A \end{bmatrix}}^{n}\right.\begin{bmatrix} z \end{bmatrix} \circ \begin{bmatrix} B \end{bmatrix}\begin{bmatrix} z \end{bmatrix} = \begin{bmatrix} C \end{bmatrix}\begin{bmatrix} z \end{bmatrix}$$

Today's Lecture

- Commitment Schemes
- Pedersen Commitment
- Complete proof of ZK for R1CS
- "Proof of Knowledge"
- Non-Interactive Zero-Knowledge

R1CS

An rank-1 constraint system (R1CS) is a generalization of arithmetic circuits

$$(F := (\mathbb{F}, n \in \mathbb{N}, A, B, C), x, w)$$

$$z := \begin{bmatrix} x \\ w \end{bmatrix} \quad \underbrace{\overbrace{[A]}^{n}}_{n}[z] \circ [B][z] = [C][z]$$

What checks do we need?

$$z := \begin{bmatrix} x \\ w \end{bmatrix} \qquad \begin{bmatrix} A \end{bmatrix}\begin{bmatrix} z \end{bmatrix} \circ \begin{bmatrix} B \end{bmatrix}\begin{bmatrix} z \end{bmatrix} = \begin{bmatrix} C \end{bmatrix}\begin{bmatrix} z \end{bmatrix}$$

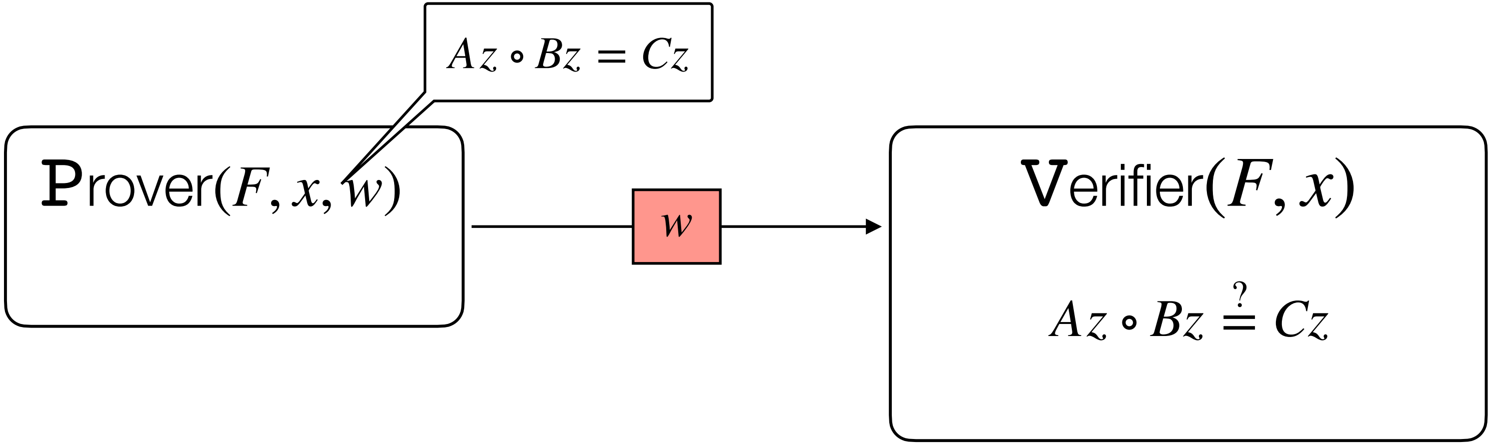
Step 1: Correct matrix multiplication

check that $Mz = z_M \quad \forall M \in \{A, B, C\}$

Step 2: Correct element-wise product

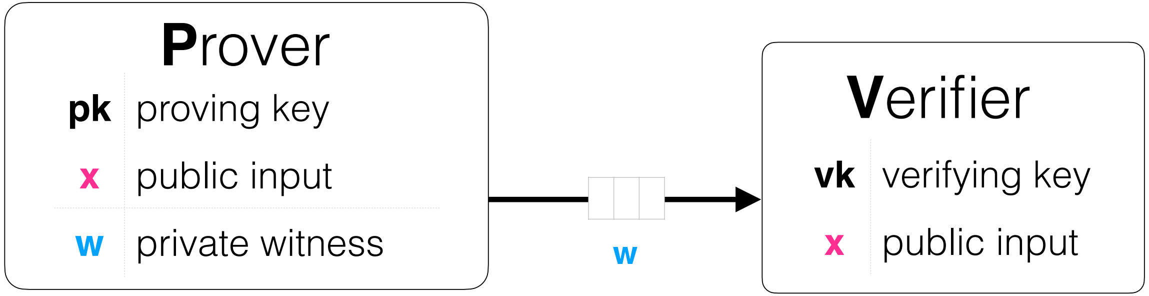
check that for each i, $z_A[i] \cdot z_B[i] = z_C[i]$

Attempt 1: Trivial NP Protocol



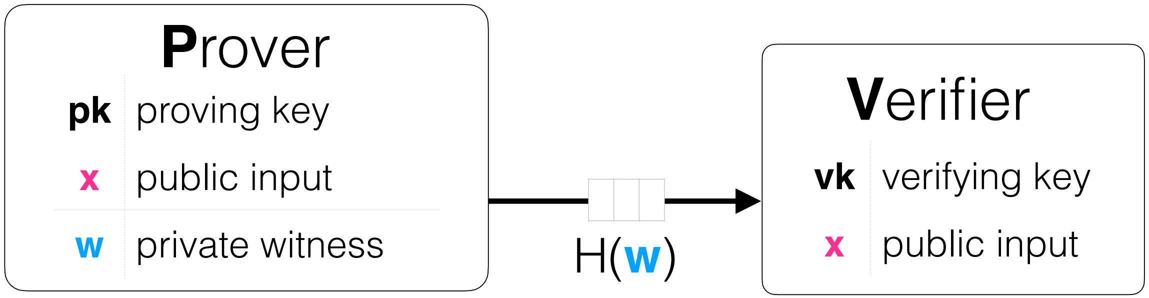
- **Completeness and Soundness are trivial**
- **What about ZK?**

Attempt 1: Trivial NP Protocol



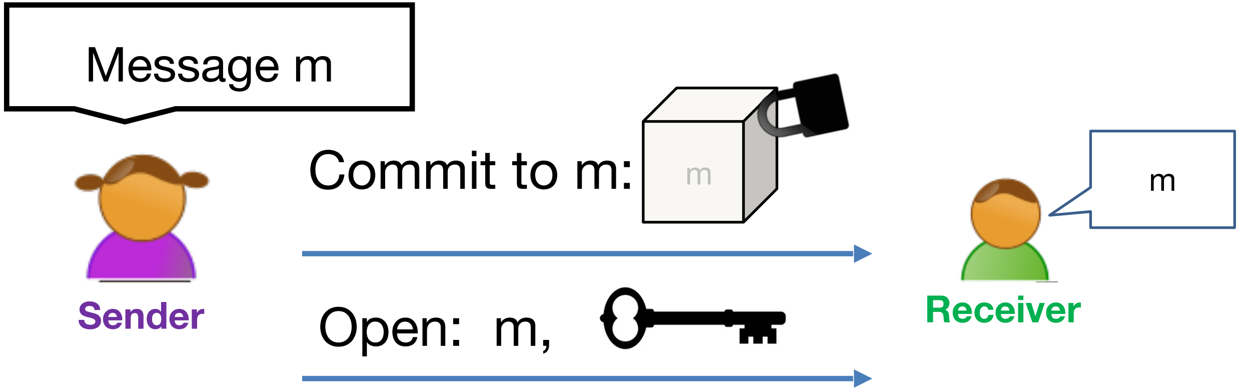
Problem: Not hiding at all!

Attempt 1: Hash the witness



Problem 1: How to verify?
Problem 2: Still might not be hiding!

We need a *commitment scheme*



1. **Hiding:** The locked box should completely hide m.

2. **Binding:** Sender shouldn't be able to open to different msg m'.

Commitment Schemes

$\text{Commit}(w; r) \rightarrow \text{cm}$

satisfying the following properties

- **Binding**: For all efficient adv. $\mathscr{A}$,

  $\Pr\left[\text{Commit}(w; r) = \text{Commit}(w'; r') \;:\; (w, r, w', r') \leftarrow \mathscr{A}\right] \approx 0$

  (no adv can open commitment to two diff values)

- **Hiding**: For all w, w', and all adv. $\mathscr{A}$,

  $\mathscr{A}(\text{Commit}(w; r)) = \mathscr{A}(\text{Commit}(w'; r'))$

  (no adv can learn committed value, i.e. comms are indistinguishable)

A standard construction

Let H be a cryptographic hash function. Then

$$\text{Commit}(w; r) := H(w, r)$$

is a commitment scheme

Pedersen Commitments

$\mathsf{Setup}(n \in \mathbb{N}) \to \mathsf{ck}$

1. Sample random elements $g_1, \ldots, g_n, h \leftarrow \mathbb{G}$

$\mathsf{Commit}(\mathsf{ck}, m \in \mathbb{F}_p^n; r \in \mathbb{F}_p) \to \mathsf{cm}$

1. Output $\mathsf{cm} := g_1^{m_1} g_2^{m_2} \ldots g_n^{m_n} h^r$

Binding

Goal: For all efficient adv. $\mathscr{A}$,

$$\Pr\left[\mathsf{Commit}(m;r) = \mathsf{Commit}(m';r') \;:\; \begin{array}{c}\mathsf{ck} \leftarrow \mathsf{Setup}(n) \\ (m,r,m',r') \leftarrow \mathscr{A}(\mathsf{ck})\end{array}\right] \approx 0$$

Proof: We will reduce to hardness of DL. Assume that $\mathscr{A}$ did indeed find breaking (m, r, m', r'). Let's construct $\mathscr{B}$ that breaks DL. Assume that $n = 1$.

Key idea: Let $h = g^x$. Then

$$g^m h^r = g^{m'} h^{r'} \implies g^{m+xr} = g^{m'+xr'}$$

Can recover $x = \dfrac{m - m'}{r' - r}$

$\mathscr{B}(g, h)$

1. $(m, r, m', r') \leftarrow \mathscr{A}(\mathsf{ck} = (g, h))$
2. Output $x = \dfrac{m - m'}{r' - r}$

Hiding

Goal: For all m, m', and all adv. $\mathscr{A}$,
$\mathscr{A}(\mathsf{Commit}(m; r)) = \mathscr{A}(\mathsf{Commit}(m'; r'))$

Proof idea: Basically one-time pad!
Let $\mathsf{cm} := \mathsf{Commit}(\mathsf{ck}, m; r)$. Let $h = g^x$.
Then, for any m', there exists r' such that $\mathsf{cm} := \mathsf{Commit}(\mathsf{ck}, m'; r')$.

We could compute it, if we knew x: $r' = \dfrac{m - m'}{x} + r$

[Note: this doesn't break binding, because $\mathscr{A}$ doesn't know x

Additive Homomorphism

Let cm and cm' be commitments to m and m' wrt r and r'.

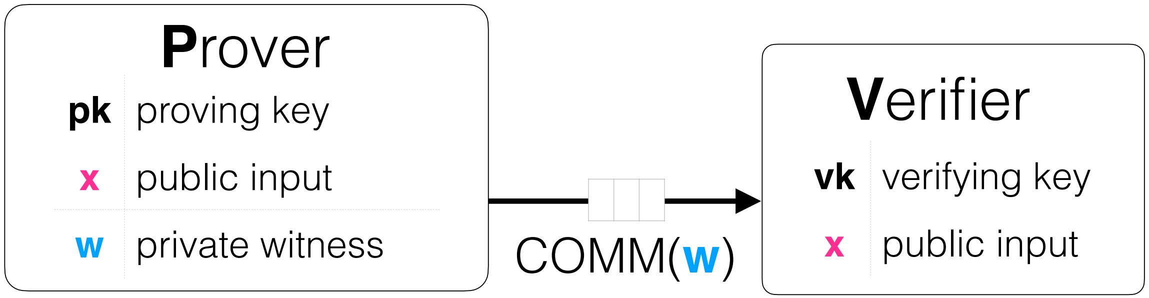
Then $\mathsf{cm} \cdot \mathsf{cm}'$ is a commitment to $m + m'$ wrt $r + r'$

$$\mathsf{cm} := g_1^{m_1} \ldots g_n^{m_n} h^r \cdot \mathsf{cm}' := g_1^{m_1'} \ldots g_n^{m_n'} h^{r'}$$

$$= g_1^{m_1 + m_1'} \ldots g_n^{m_n + m_n'} h^{r + r'}$$

$$= \mathsf{Commit}(\mathsf{ck}, m + m'; r + r')$$

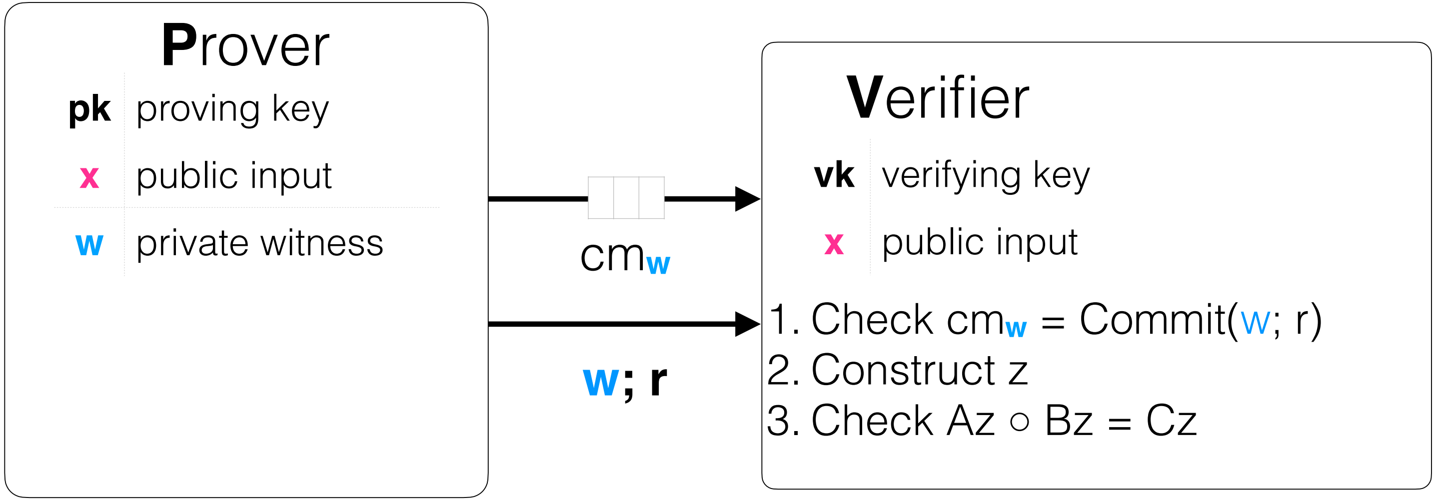
Attempt 2: Commit to the witness



Problem 1: How to verify?

Solution 2: Hiding from COMM!

Attempt 3: Commit to the witness

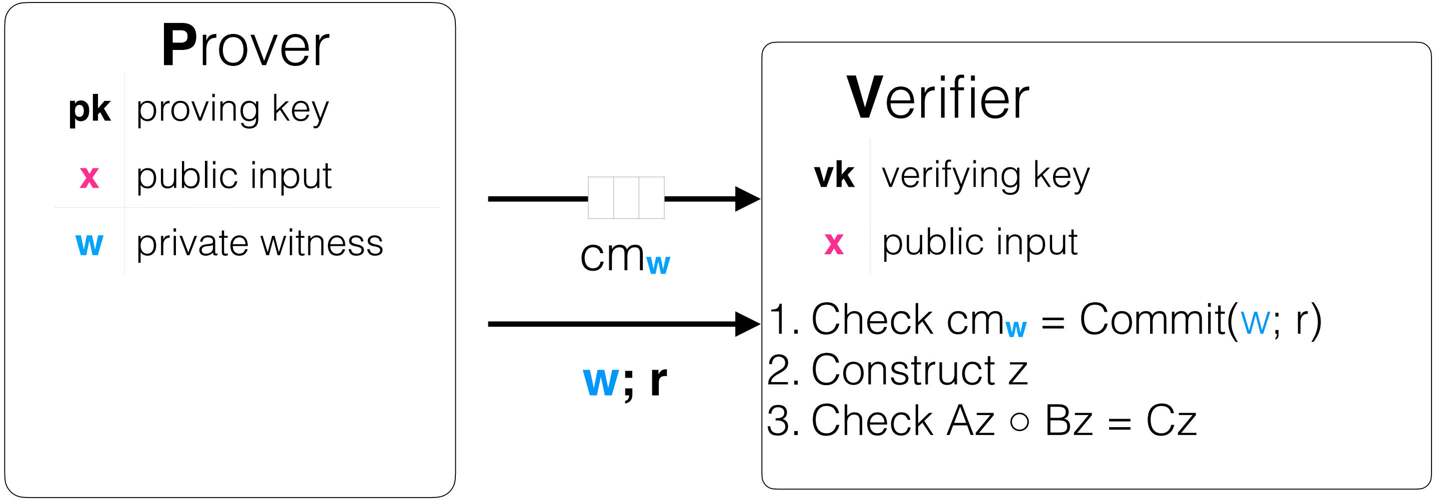


Solution 1: Just check!

Problem 2: No hiding again!

Performing checks on committed data?

Attempt 3: Commit to the witness



Solution 1: Just check!

Idea 2: Blind it!!

Examples of NP Assertions

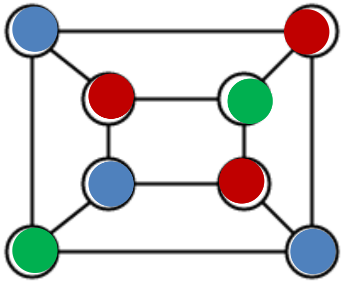
- **My public key is well-formed** (e.g. in RSA, the public key is N, a product of two primes together with an e that is relatively prime to $\varphi(N)$.)
- **Encrypted bitcoin (or Zcash): "I have enough money to pay you."** (e.g. I will publish an encryption of my bank account and prove to you that my balance is $\geq \$X$.)
- **Running programs on encrypted inputs:** Given Enc(x) and y, prove that y = PROG(x).

Examples of NP Assertions

- **Running programs on encrypted inputs:** Given Enc(x) and y, prove that y = PROG(x).

More generally: A tool to enforce honest behavior without revealing information.

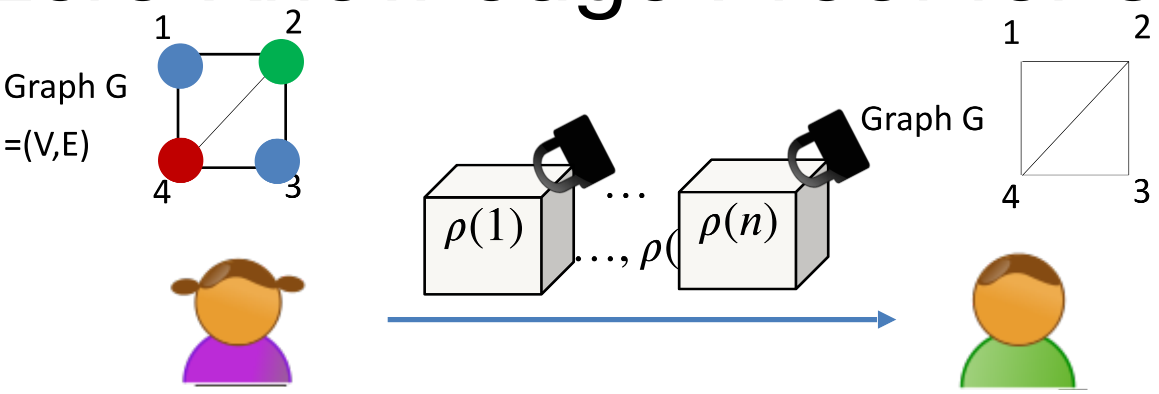
Zero Knowledge Proof for 3-Coloring



***NP-Complete* Problem:**

Every other problem in NP can be reduced to it.

Zero Knowledge Proof for 3COL



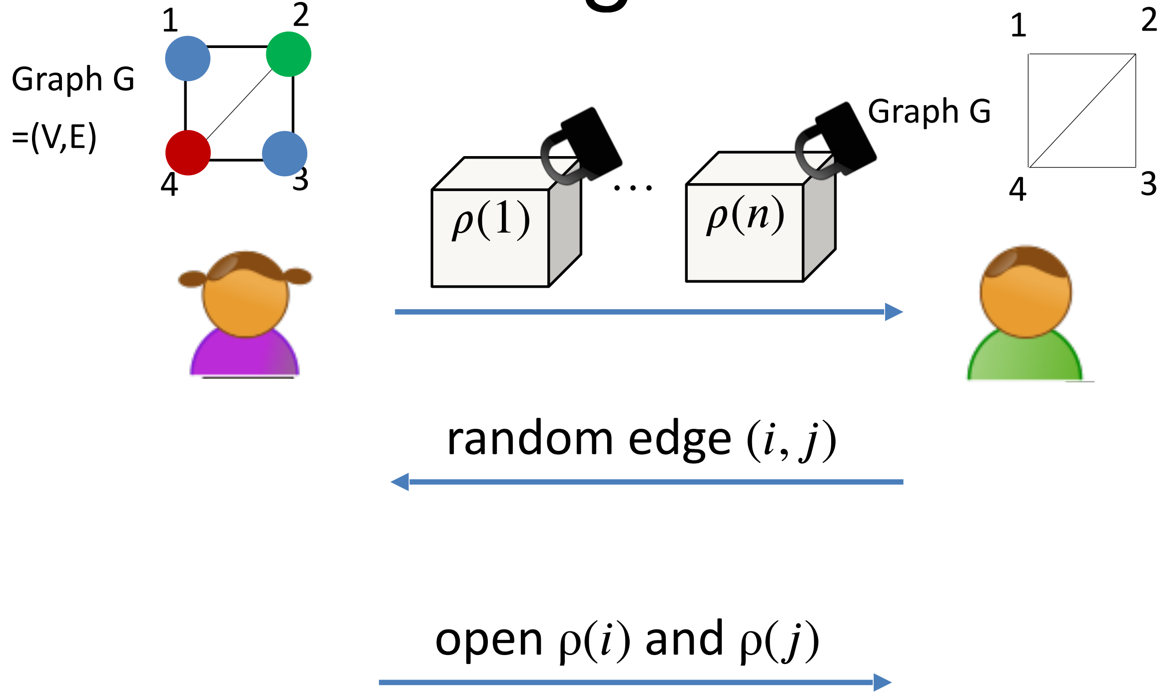
Come up with a random perm of the colors $\rho: V \rightarrow \{R, B, G\}$

random edge (i, j)

open $\rho(i)$ and $\rho(j)$

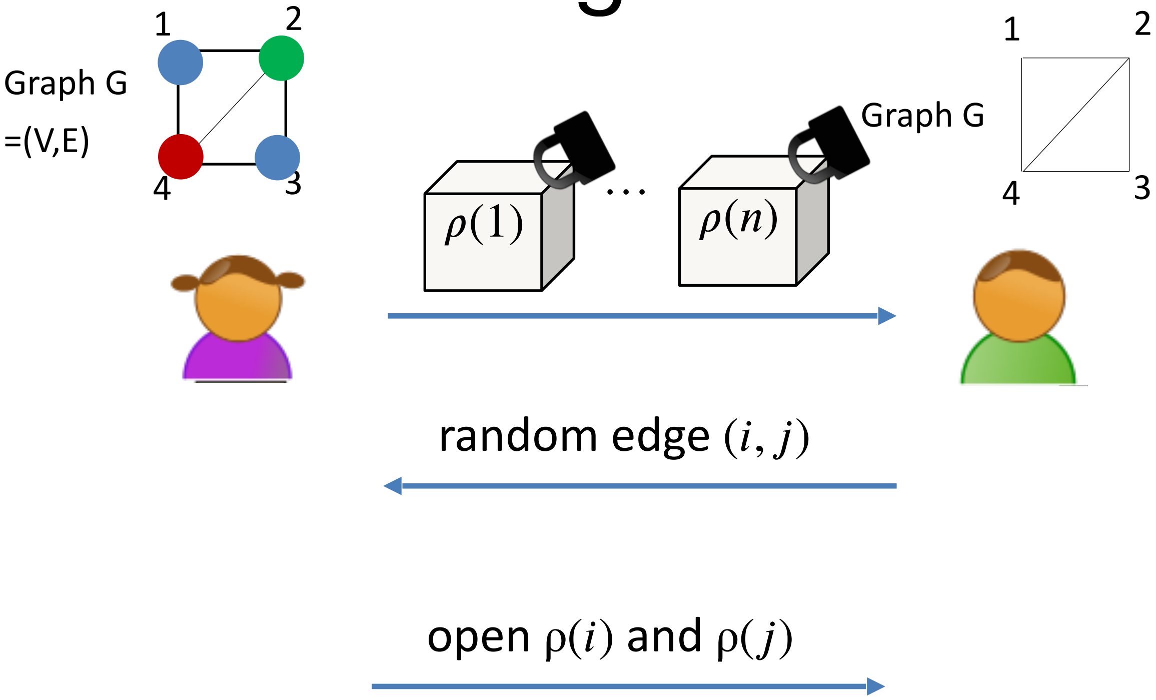
1. Check the openings
2. Check: $\rho(i),\ \rho(j) \in \{R, B, G\}$
3. Check: $\rho(i) \neq \rho(j)$.

Zero Knowledge Proof for 3COL



Completeness: Exercise.

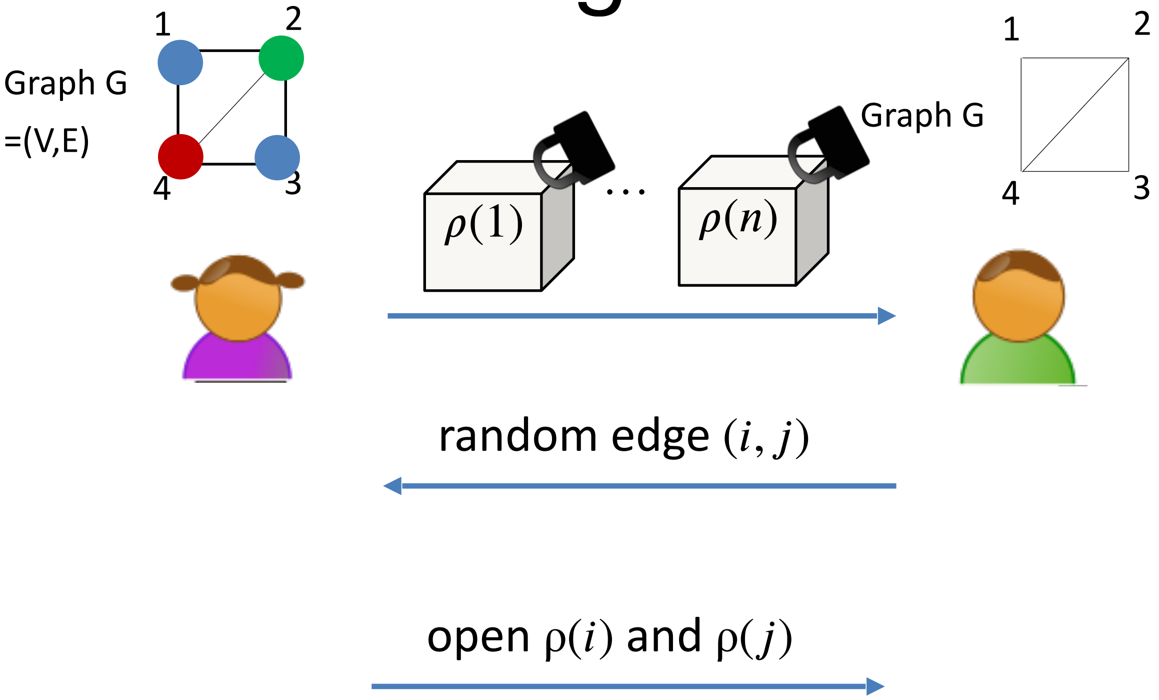
Zero Knowledge Proof for 3COL



Soundness: If the graph is not 3COL, in every 3-coloring (that P commits to), there is some edge whose end-points have the same color.

V will catch this edge and reject with probability $\geq 1/|E|$.

Zero Knowledge Proof for 3COL



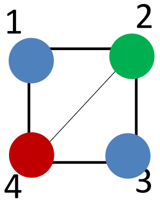
Repeat $|E| \bullet \lambda$ times to get the verifier to accept with probability

$$\leq (1 - 1/|E|)^{|E| \bullet \lambda} \leq 2^{-\lambda}$$

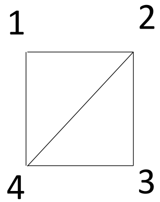
Constructing
Commitment Schemes

Back to ZK Proof for 3COL

Graph G =(V,E)



Graph G



$\{Com(\rho(k); r_k)\}_{k=1}^{n}$ →

random edge (i, j) ←

send openings $\rho(i), r_i$ and $\rho(j), r_j$ →

Why is this zero-knowledge?

Simulator S works as follows:

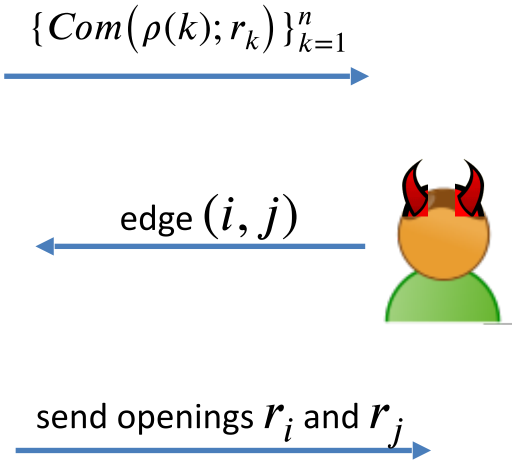
1. First pick a random edge (i^*, j^*)

   Color vertices i^* and j^* with random, different colors
   Color all other vertices red.

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.

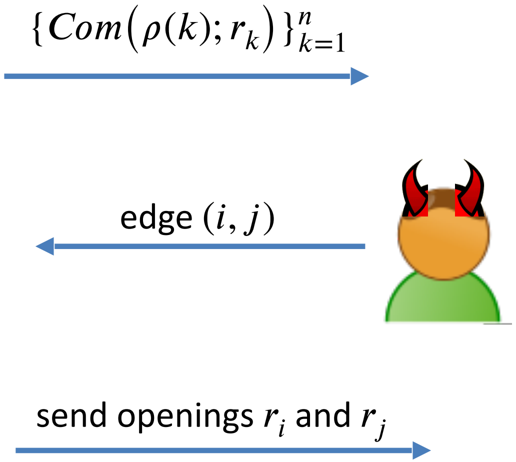


Why is this zero-knowledge?

Lemma:

(1) Assuming the commitment is hiding, S runs in expected polynomial-time.

(2) When S outputs a view, it is comp. indist. from the view of V^* in a real execution.



Why is this zero-knowledge?

Simulator S works as follows (call this Hybrid 0)

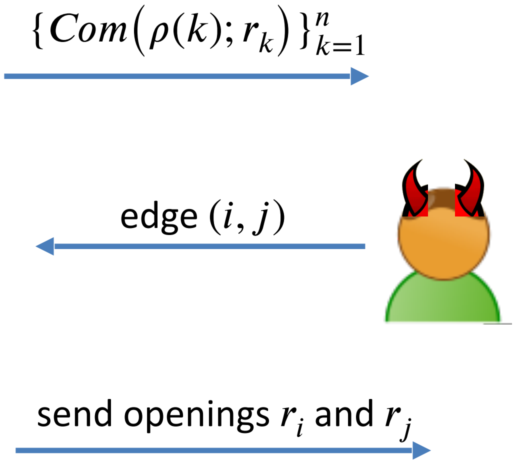
1. First pick a random edge (i^*, j^*)

   Color vertices i^* and j^* with random, different colors
   Color all other vertices red.

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.



Why is this zero-knowledge?

Not-a-Simulator S works as follows (call this Hybrid 1)

1. First pick a random edge (i^*, j^*)

   Permute a legal coloring and color all vertices correctly.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$

edge (i, j)

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

send openings r_i and r_j

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.

Why is this zero-knowledge?

Claim: Hybrids 0 and 1 are computationally indistinguishable, assuming the commitment scheme is computationally hiding.

Proof: By contradiction. Show a reduction that breaks the hiding property of the commitment scheme, assuming there is a distinguisher between hybrids 0 and 1.

Why is this zero-knowledge?

Not-a-Simulator S works as follows (call this Hybrid 1)

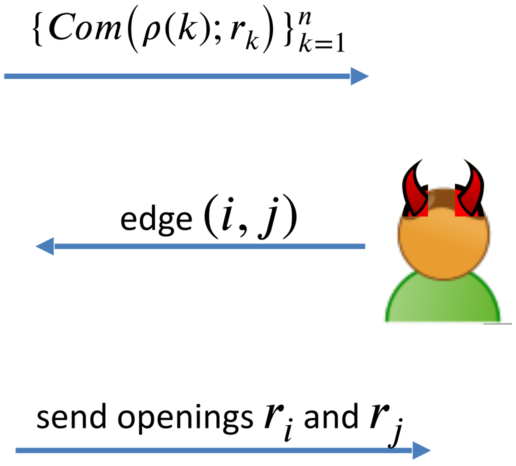
1. First pick a random edge (i^*, j^*)

   Permute a legal coloring and color all vertices correctly.

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. If $(i, j) \neq (i^*, j^*)$, go back and repeat.

4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.



Why is this zero-knowledge?

Here is the real view of V* (Hybrid 2)

1. ~~First pick a random edge (i^*, j^*)~~

   Permute a legal coloring and color all edges correctly.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$ →

edge (i, j) ←

2. Feed the commitments of the colors to V^* and get edge (i, j)

3. ~~If $(i, j) \neq (i^*, j^*)$, go back and repeat.~~

send openings r_i and r_j →

4. ~~If $(i, j) = (i^*, j^*)$,~~ output the commitments and openings r_i and r_j as the transcript.

Why is this zero-knowledge?

Claim: Hybrids 1 and 2 are identical.

Hybrid 1 merely samples from the same distribution as Hybrid 2 and, with probability $1 - 1/|E|$, decides to throw it away and resample.

Put together:

Theorem: The 3COL protocol is zero knowledge.