CIS 5560

Cryptography
Lecture 23

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

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Announcements

- **HW10 d**ue **Wednesday Apr 24** at 11:59PM on Gradescope

Recap of Last Lecture

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

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:

1. $\text{view}_V(P, V)$ 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. $\text{view}_{V^*}(P, V^*)$ 2. $S(x, 1^\lambda)$

Simulator S works as follows:

1. First set $s = \frac{z^2}{y^b}$ for a random z and feed s to V^*.

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

3. If $b' = b$, output $(\mathrm{s}, \mathrm{b}, \mathrm{z})$ and stop.

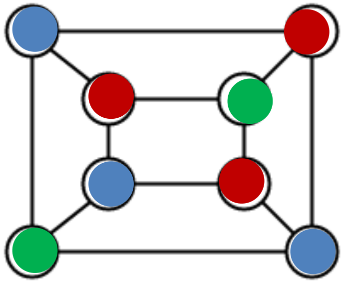
4. Otherwise, go back to step 1 and repeat. (also called "rewinding").

Lemma:

(1) S runs in expected polynomial-time.

(2) When S outputs a view, it is identically distributed to the view of V^* in a real execution.

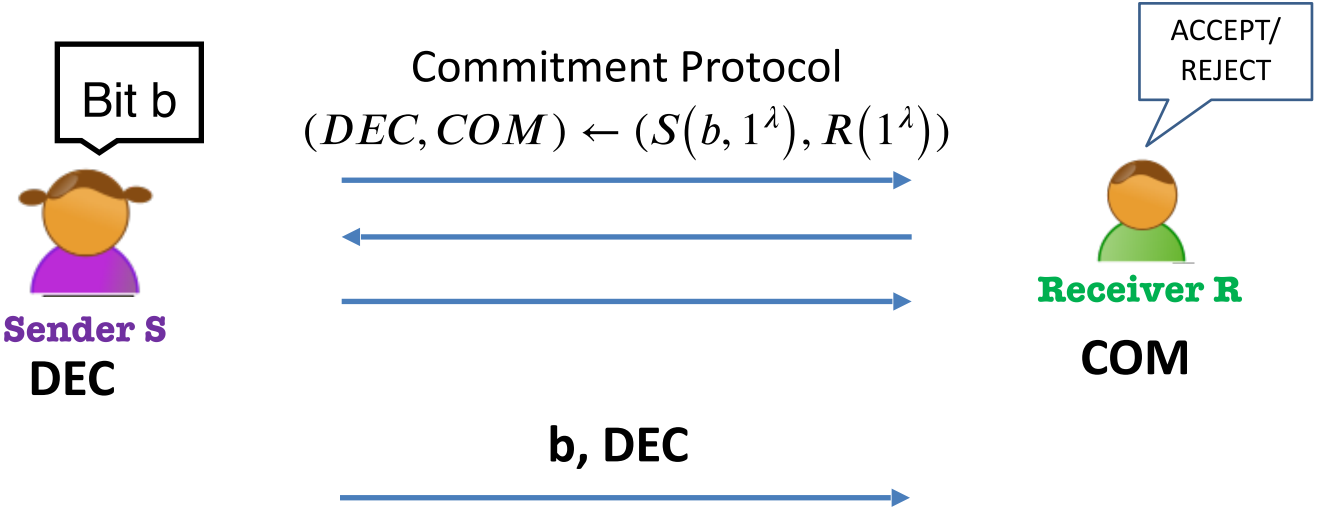
Zero Knowledge Proof for 3-Coloring



***NP-Complete* Problem:**

Every other problem in NP can be reduced to it.

Commitment Schemes



Completeness: R always accepts in an honest execution.

Hiding: COM reveals no information about b.

Binding: Sender cannot find (b', DEC') such that $b \neq b'$ and yet R accepts (b', DEC').

A Commitment Scheme from any OWP

Sender S

$DEC = r$

$COM = (f(r), HCB(r) \oplus b)$

Receiver R

Let $COM = \left(x, y\right)$.

Check that

1. $f(r) = x$ and
2. $HCB(r) \oplus b = \mathrm{y}$

$OPEN: (b, r)$

1. **Completeness:** Exercise.

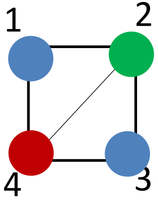
2. **Comp. Hiding:** by the hardcore bit property.

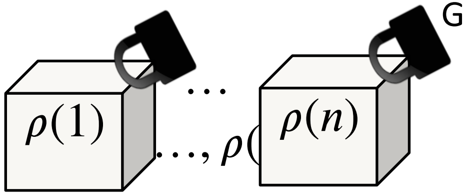
3. **Perfect Binding:** because f is a permutation.

Zero Knowledge Proof for 3COL

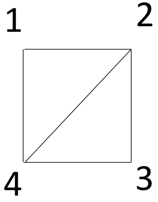
Graph G

=(V,E)





Graph G



random edge (i, j)

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

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

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

Today's Lecture

- Complete proof of ZK for 3COL
- "Proof of Knowledge"
- Non-Interactive Zero-Knowledge

Why is 3COL Protocol ZK?

Simulator S works as follows:

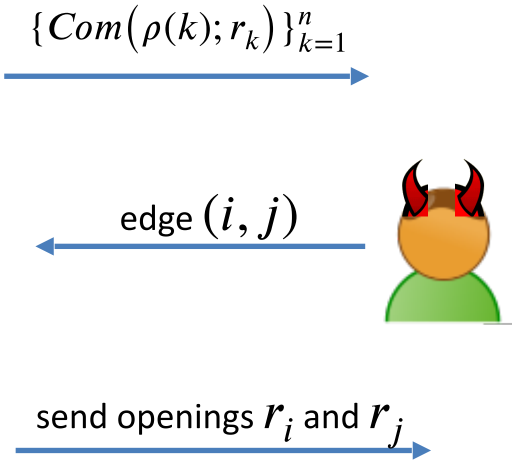
1. First pick a random edge (i^*, j^*)

   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.

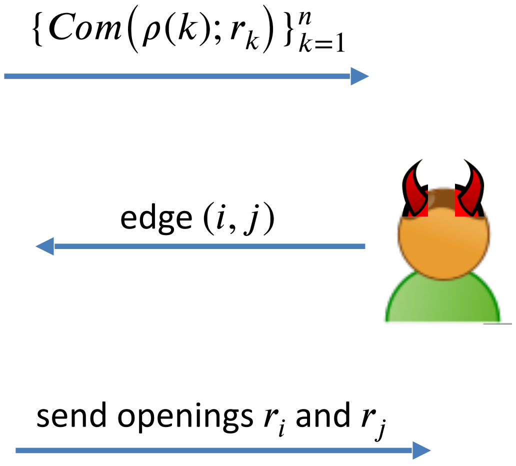


Why is this zero-knowledge?

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.

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

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

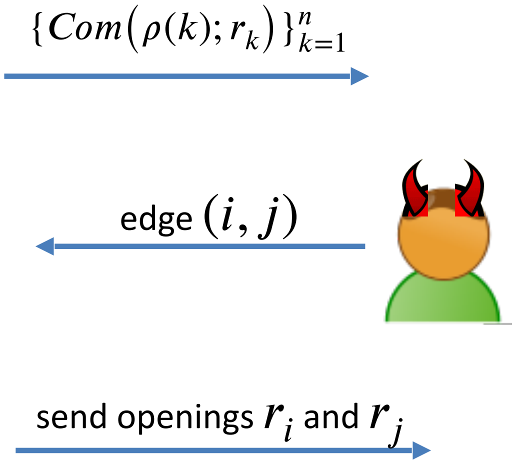
4. If $(i, j) = (i^*, j^*)$, output the commitments and openings r_i and r_j as the simulated transcript.

Why is this zero-knowledge?

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?

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.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$ →

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

← 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?

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.

$\{Com(\rho(k); r_k)\}_{k=1}^{n}$ →

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

← 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.

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.