CIS 5560

Cryptography
Lecture 22

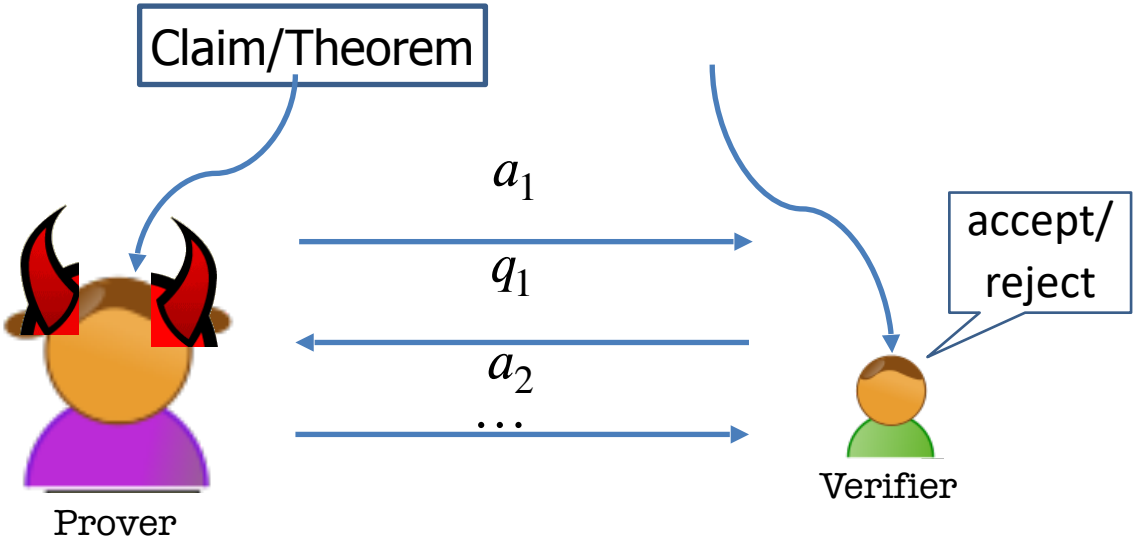
Course website:
pratyushmishra.com/classes/cis-5560-s25

Slides adapted from Dan Boneh and Vinod Vaikuntanathan

Recap of last lecture

- What is a proof?
- Interactive Proofs
- *Zero-knowledge* interactive proofs
  - Definition
- ZKP for Graph Isomorphism
- ZKP for Graph Non-Isomorphism

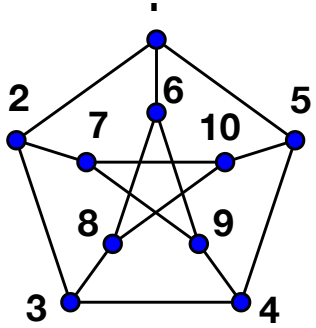
Interactive Proofs for a Language $\mathcal{L}$



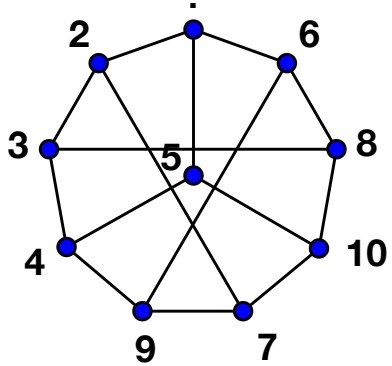
Def: $\mathcal{L}$ is an IP-language if there is a unbounded P and **probabilistic poly-time** verifier V where

- **Completeness**: If $x \in \mathcal{L}$, V always accepts.
- **Soundness:** If $x \notin \mathcal{L}$, regardless of the cheating prover strategy, V accepts with negligible probability.

IP for Graph *Non*-Isomorphism



Graph G



Graph H

Prover

Verifier

Sample random permutation ρ

Sample bit b

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

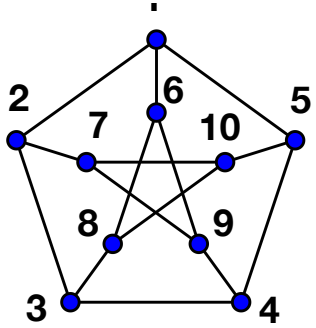
Figure out which graph K_b is isomorphic to.

K_b ←

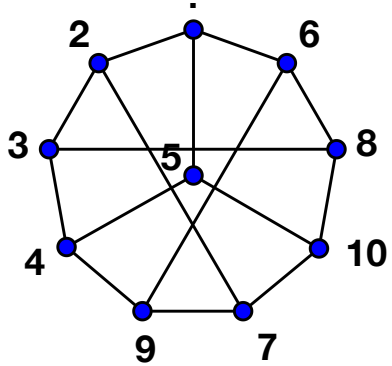
b' →

Accept if $b = b'$

IP for Graph Isomorphism



Graph G



Graph H

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

$K = \rho(G)$

where ρ is a random permutation

random challenge bit b

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

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

Prover

Verifier

How to Define Zero-Knowledge?

After the interaction, V knows:

- The theorem is true; and
- A **view** of the interaction
  (= transcript + randomness of V)

P gives zero knowledge to V:

When the theorem is true, the view gives V nothing that he couldn't have obtained on his own without interacting with P.

Zero Knowledge: Definition

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

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

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

Perfect Zero Knowledge: Definition

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

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

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

Computational Zero Knowledge: Definition

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

1. $view_V(P, V)$
2. $S(x, 1^\lambda)$

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

Today’s Lecture

- Malicious-verifier/“standard” ZK
  - ZKPs for GI achieve standard ZK
- Commitment schemes
  - Pedersen Commitments
- Rank-1 Constraint Systems

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

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 pick a random ϕ
and b and send $\phi(G_b)$ to V.
2. Let $b' = V^*(K)$.

Now what???

$view_{V^*}\left(P, V^*\right)$:
(K, b, ϕ)

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”).

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”).

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.

What Made it Possible?

1. **Each statement had multiple proofs** of which the prover chooses one at random.

2. **Each such proof is made of two parts**: seeing either one on its own gives the verifier no knowledge; seeing both imply 100% correctness.

3. **Verifier chooses to see either part, at random**. The prover's ability to provide either part on demand convinces the verifier.

Do all NP languages have Perfect ZK proofs?

We showed two NP languages with perfect ZK proofs. Can we show this for *all* NP languages?

Theorem [Fortnow'89, Aiello-Hastad'87] No, unless bizarre stuff happens in complexity theory (technically: the polynomial hierarchy collapses.)

Do all NP languages have

Winner of 2024 Turing Award!

Nevertheless, today, we will sho

Theorem [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!

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).

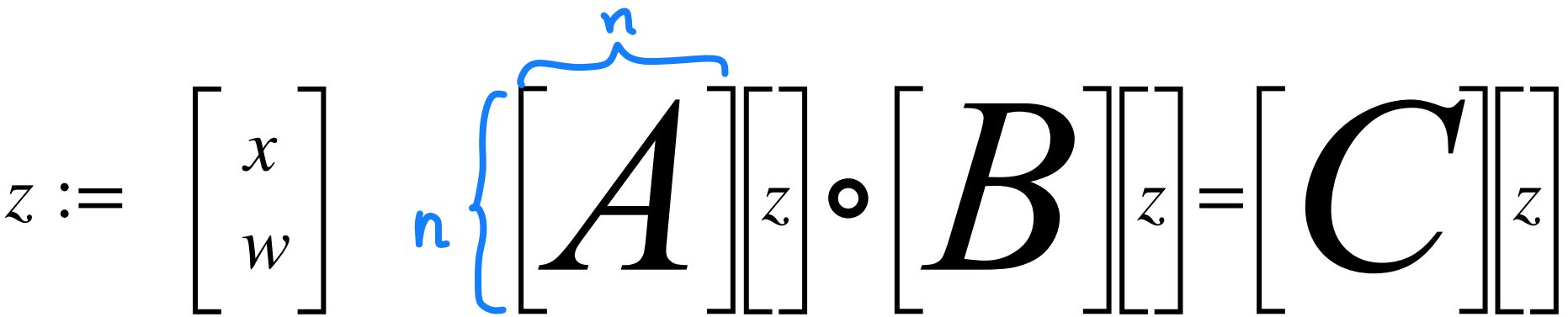
More generally: A tool to enforce honest behavior without revealing information.

Let’s pick a concrete NP relation

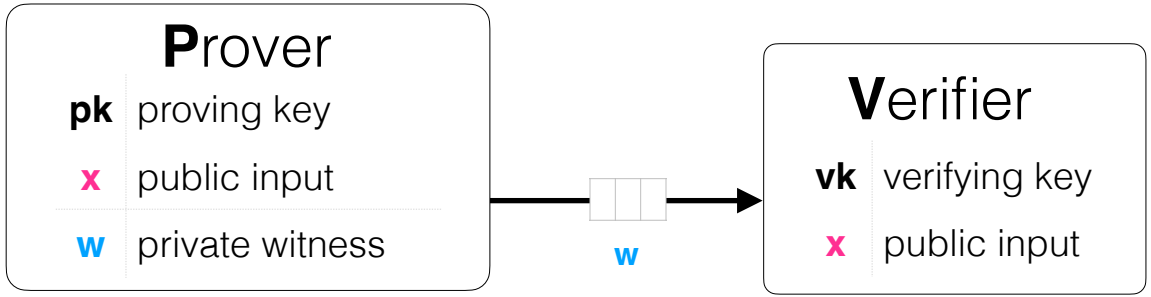
R1CS

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

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

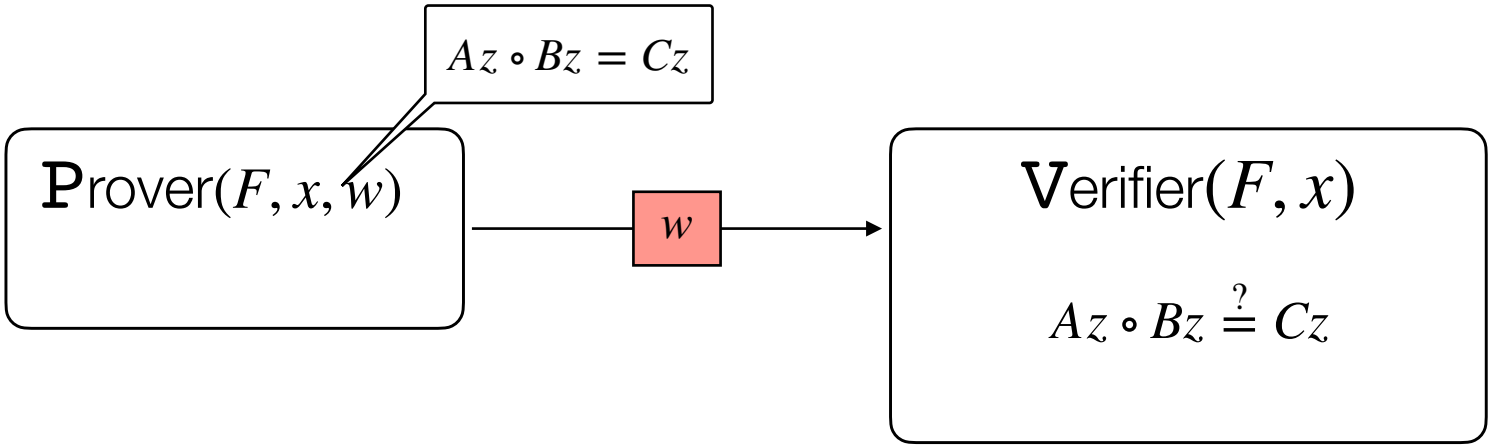


Starting point: Trivial NP Protocol



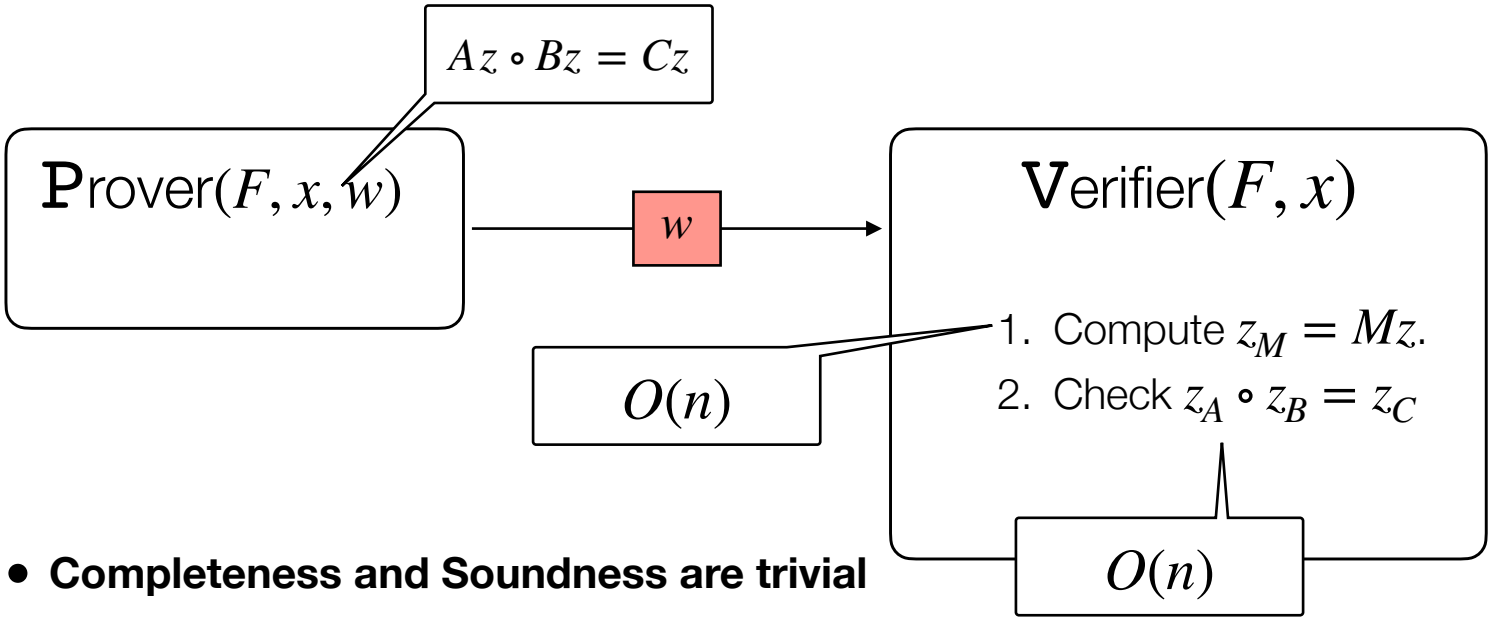
Problem: Not hiding at all!

Strawman 1



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

Strawman 1



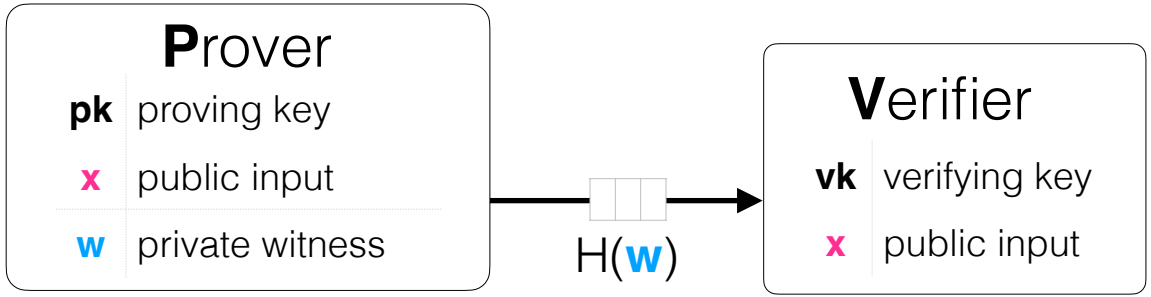
- **Completeness and Soundness are trivial**
- **What about efficiency?**

What checks do we need?

Step 1: Correct element-wise product
check that for each i, $z_A[i] \cdot z_B[i] = z_C[i]$

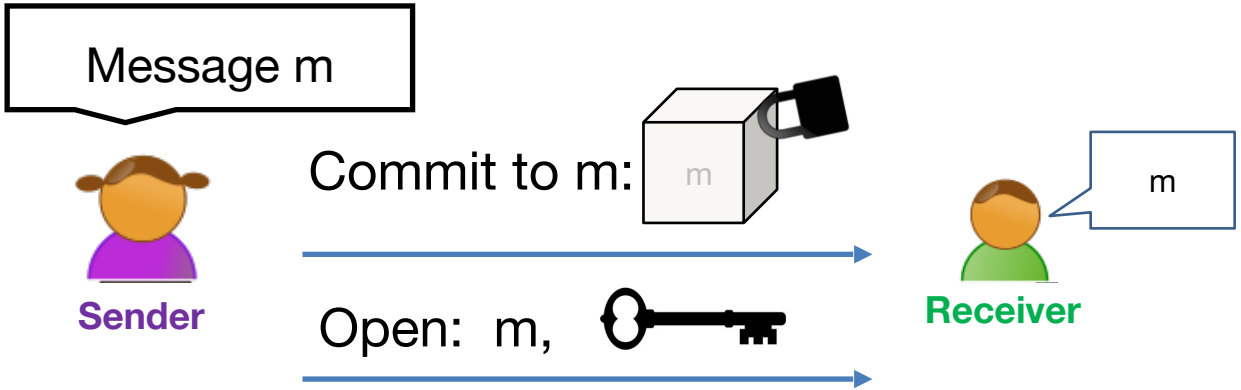
Step 2: Correct matrix multiplication
check that $Mz = z_M \quad \forall M \in \{A, B, C\}$

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

$\mathsf{Commit}(w; r) \rightarrow \mathsf{cm}$
satisfying the following properties

- **Binding**: For all efficient adv. $\mathscr{A}$,
  $\Pr\left[\mathsf{Commit}(w; r) = \mathsf{Commit}(w'; r') \ : \ (w, r, w', r') \leftarrow \mathscr{A}\right] \approx 0$
  (no adv can open commitment to two diff values)

- **Hiding**: For all w, w', and all adv. $\mathscr{A}$,
  $\mathscr{A}(\mathsf{Commit}(w; r)) = \mathscr{A}(\mathsf{Commit}(w'; r'))$
  (no adv can learn committed value, i.e. comms are indistinguishable)

A standard construction

Let H be a cryptographic hash function. Then

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

is a commitment scheme

Pedersen Commitments

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

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

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

1. Output 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. $\mathcal{A}$,
$\mathcal{A}(\mathsf{Commit}(m; r)) = \mathcal{A}(\mathsf{Commit}(m'; r'))$

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: $r' = \frac{m - m'}{x} + r$

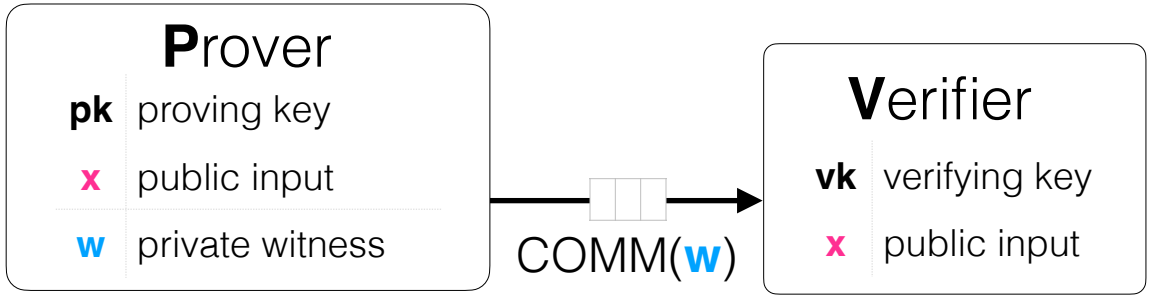
[Note: this doesn't break binding, because $\mathcal{A}$ doesn't know x

Additive Homomorphism

Let cm and cm' be commitments to m and m' wrt r and r'.
Then $\mathsf{cm} + \mathsf{cm}'$ is a commitment to $m + m'$ wrt $r + r'$

$$\begin{aligned}
&\mathsf{cm} := g_1^{m_1} \ldots g_n^{m_n} h^r + \mathsf{cm}' := g_1^{m'_1} \ldots g_n^{m'_n} h^{r'} \\
&= g_1^{m_1 + m'_1} \ldots g_n^{m_n + m'_n} h^{r + r'} \\
&= \mathsf{Commit}(\mathsf{ck}, m + m'; r + r')
\end{aligned}$$

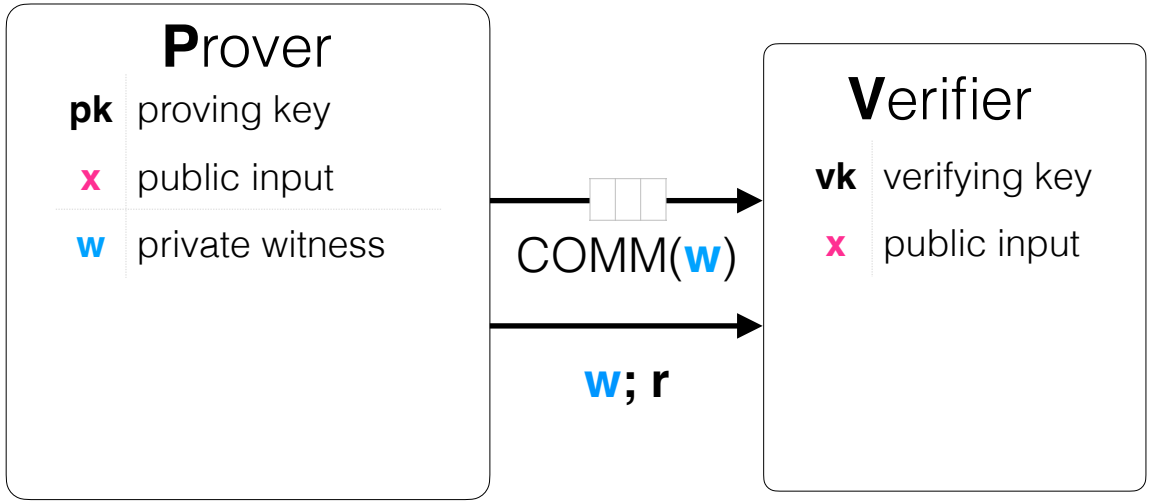
Attempt 2: Commit to the witness



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?