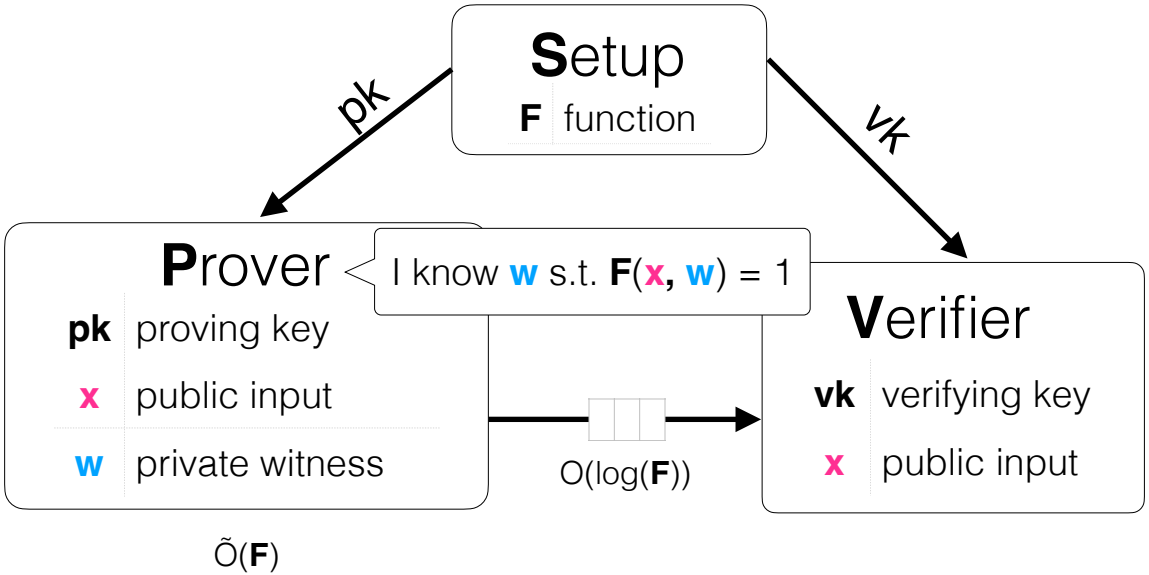
Succinct Arguments

Lecture 02: Modern zkSNARK Constructions

Pratyush Mishra
UPenn
Fall 2025

Succinct Non-Interactive Arguments (SNARGs)

[Mic94, Groth10, GGPR13, Groth16…
…, GWC19, CHM**M**VW20, …]



Succinct Non-Interactive Arguments (SNARGs)

- **Completeness**: If $(F, x, w) \in \mathscr{R}$,

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) = 1 \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w)\end{array}\right] = 1.$$

- **Soundness**: If $(F, x, w) \notin \mathscr{R}$, for all efficient provers $\tilde{\mathbf{P}}$

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) = 1 \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \tilde{\mathbf{P}}(\mathsf{pk}, x)\end{array}\right] \approx 0$$

- **Succinctness**: $|\pi| = O(\mathsf{polylog}\,|F|)$

What if there's always a witness?

Soundness: If $(F, x, w) \notin \mathscr{R}$, then for all efficient provers $\tilde{\mathbf{P}}$

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) = 1 \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \tilde{\mathbf{P}}(\mathsf{pk}, x)\end{array}\right] \approx 0$$

- $F(x, w) := \mathsf{SHA2}(w) \stackrel{?}{=} x$: there is always a preimage!
- $F((m, \mathsf{pk}), \sigma) := \mathsf{VerifySignature}(\mathsf{pk}, m, \sigma) \stackrel{?}{=} 1$: if pk is a valid public key, there is always a valid signature!
- Generally many examples where **witness always exists!**

SNARGs of Knowledge (SNARKs)

- **Completeness**: For all $(F, x, w) \in \mathscr{R}$,

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) = 1 \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w)\end{array}\right] = 1.$$

- **Knowledge Soundness**: If $\mathbf{V}(\mathsf{vk}, x, \pi) = 1$, then $\tilde{\mathbf{P}}$ "knows" w such that $(F, x, w) \in \mathscr{R}$

- **Succinctness**: $|\pi| = O(\log |F|)$

SNARGs of Knowledge (SNARKs)

- **Completeness**: For all $(F, x, w) \in \mathscr{R}$,

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) = 1 \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w)\end{array}\right] = 1.$$

- **Knowledge Soundness**: For each efficient $\tilde{\mathbf{P}}$ there exists an **extractor** $\mathbf{E}$ such that

$$\Pr\left[\begin{array}{c}V(\mathsf{vk}, x, \pi) = 1 \\ \wedge \\ (F, x, w) \notin \mathscr{R}\end{array} \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \tilde{\mathbf{P}}(\mathsf{pk}, x) \\ w \leftarrow \mathbf{E}_{\tilde{\mathbf{P}}}(\mathsf{pk}, x)\end{array}\right] \approx 0$$

- **Succinctness**: $|\pi| = O(\log |F|)$

What about privacy?

- $F(x, w) := \mathsf{SHA2}(w) \stackrel{?}{=} x$:
  Does proof reveal info about preimage?
- $F((m, \mathsf{pk}), \sigma) := \mathsf{VerifySignature}(\mathsf{pk}, m, \sigma) \stackrel{?}{=} 1$:
  Does proof reveal info about which signature was used?
- $F(x = \mathsf{score}, w = \mathsf{credit\_hist}) := \mathsf{CreditModel}(w) \stackrel{?}{=} x$
  Does proof reveal info about credit history?

Verifier is the adversary now!

Zero Knowledge SNARKs (zkSNARKs)

- **Completeness**: For all $(F, x, w) \in \mathscr{R}$, …
- **Knowledge Soundness**: For each efficient $\tilde{\mathbf{P}}$ there exists an **extractor** $\mathbf{E}$ such that …
- **Zero Knowledge**: Proof reveals no information to $\mathbf{V}$ other than validity of w
- **Succinctness**: $|\pi| = O(\log |F|)$

Zero Knowledge SNARKs (zkSNARKs)

- **Completeness**: For all $(F, x, w) \in \mathscr{R}$, …
- **Knowledge Soundness**: For each efficient $\tilde{\mathbf{P}}$ there exists an **extractor** $\mathbf{E}$ such that …
- **Zero Knowledge**: For all $(F, x, w) \in R$, and all efficient $\tilde{\mathbf{V}}$ there exists an **simulator** Sim such that

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk}, x)\end{array}\right] = \Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w)\end{array}\right]$$

- **Succinctness**: $|\pi| = O(\log |F|)$

Doesn't this break soundness?

$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk}, x)\end{array}\right] = \Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \;:\; \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w)\end{array}\right]$$

Sim has same success probability as honest prover!

This is actually okay: we provide Sim with additional powers!

- Interactive case: Sim can rewind verifier
- Non-interactive case: Sim gets "trapdoor"/secret information

What about succinct verification?

Succinctness: $|\pi| = O(\text{polylog}\,|F|)$

- $F(x, w) = \text{SHA2}^{10^6}(w) \stackrel{?}{=} x$:
  Do I need to compute 10^6 hashes to verify proof?
- $F(x = \text{score}, w = \text{credit\_hist}) = \text{CreditModel}(w) \stackrel{?}{=} x$
  Do I need to evaluate complex model to verify proof?

Strongly Succinct zkSNARKs

- **Completeness**: For all $(F, x, w) \in \mathscr{R}$, …
- **Knowledge Soundness**: For each efficient $\tilde{\mathbf{P}}$ there exists an **extractor** $\mathbf{E}$ such that …
- **Zero Knowledge**: For all $(F, x, w) \in R$, and all efficient $\tilde{\mathbf{V}}$ there exists an **simulator** Sim such that

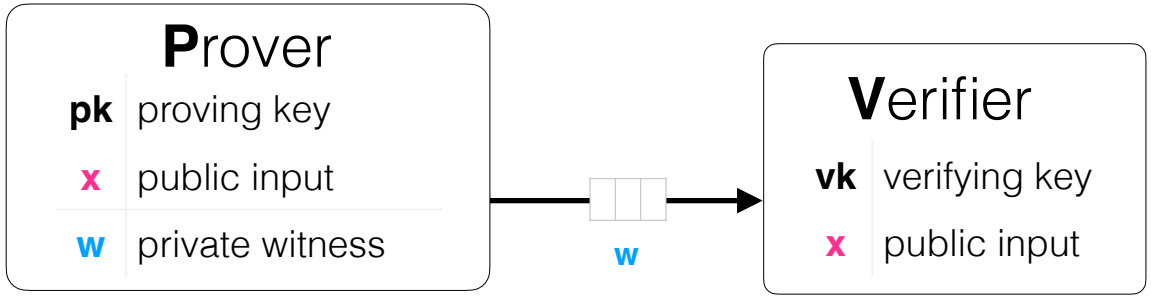
$$\Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathsf{Sim}(\mathsf{pk}, x)\end{array}\right] = \Pr\left[\mathbf{V}(\mathsf{vk}, x, \pi) \ : \ \begin{array}{c}(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(F) \\ \pi \leftarrow \mathbf{P}(\mathsf{pk}, x, w)\end{array}\right]$$

- **Succinctness**: $|\pi| = O(\log |F|)$

and $\mathsf{Time}(\mathbf{V}) = O(\mathsf{polylog}(|F|), |x|)$

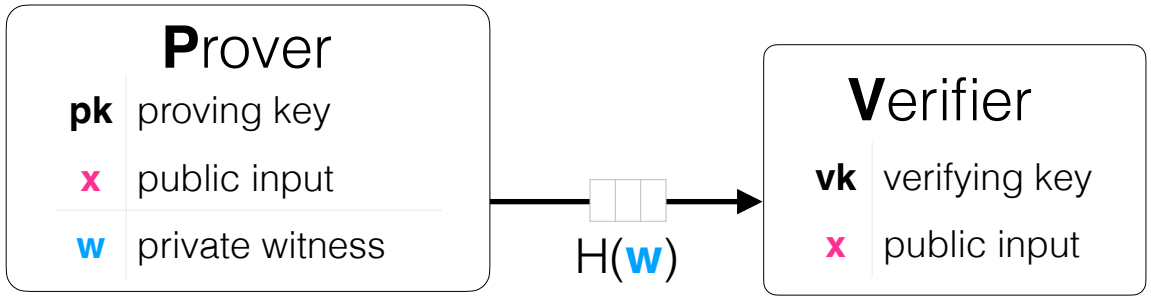
Constructing zkSNARKs

Starting point: Trivial NP Protocol



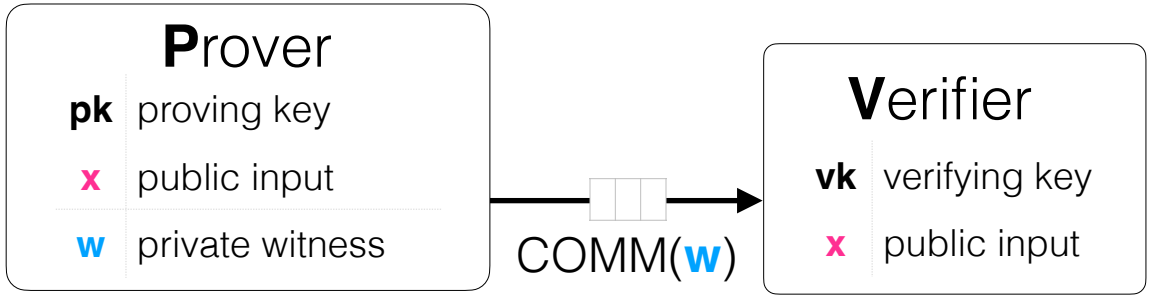
Problem 1: Non-succinct proof!
Problem 2: Non-succinct verification!
Problem 3: Not hiding at all!

Strawman 1: Hash the witness



Problem 1 solved: Succinct proof!
Problem 2: How to verify?
Problem 3: Still might not be hiding!

Strawman 2: Commit to the witness



Problem 1 solved: Succinct proof!
Problem 2: How to verify?
Problem 3: Still might not be hiding!

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)

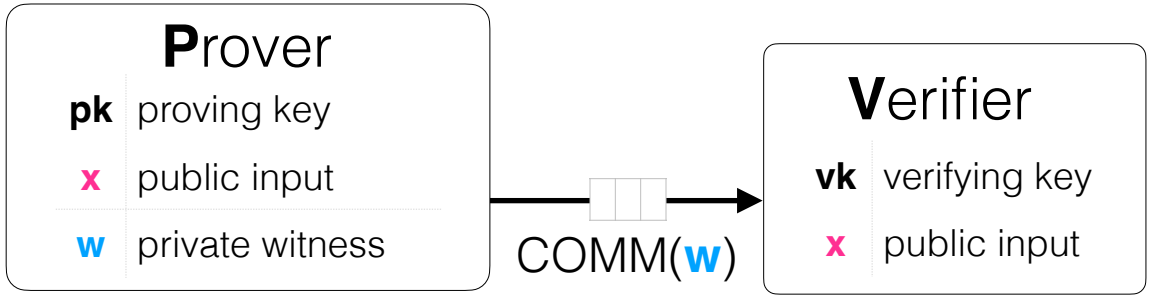
A standard construction

Let H be a cryptographic hash function. Then

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

is a commitment scheme

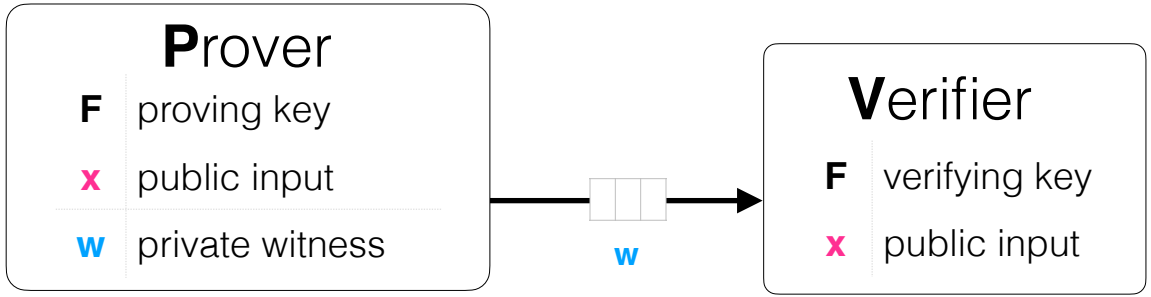
Strawman 2: Commit to the witness



Problem 1 solved: Succinct proof!
Problem 2: How to verify?
Problem 3 solved: COMM hides w!

Performing checks on committed data?

What does V do in the Trivial NP proof?



Evaluate F(x, w)!

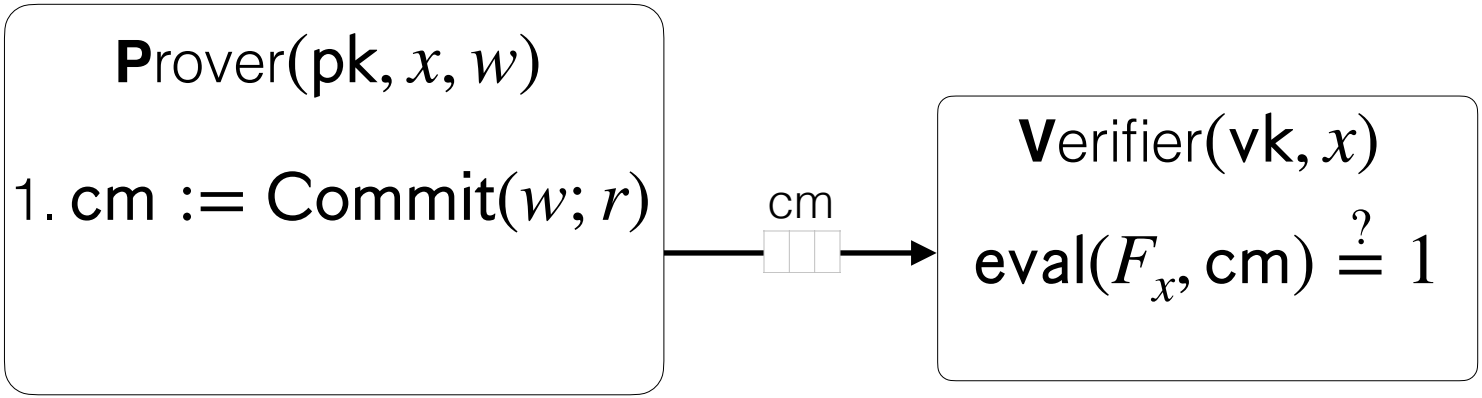
To apply this to our commitment-based protocol, do we need a “fully-homomorphic” commitment?

Homomorphic Commitments?

Pair of algorithms with the following syntax:

- $\mathsf{Commit}(w; r) \rightarrow \mathsf{cm}$
  - Commits to the message
- $\mathsf{Eval}(F_x, \mathsf{cm}) \rightarrow F(x, w)$
  - Evaluates a function over the committed message, and outputs the result in the clear.

Strawman 3: Homomorphic Commitments



Completeness: Follows from that of commitment

Knowledge Soundness: Follows from Trivial NP Proof

Succinct pf size: Follows if eval. proof is succinct

ZK: ???

Problem 1: This would violate ZK: no hiding!

Problem 2: All constructions are inefficient!

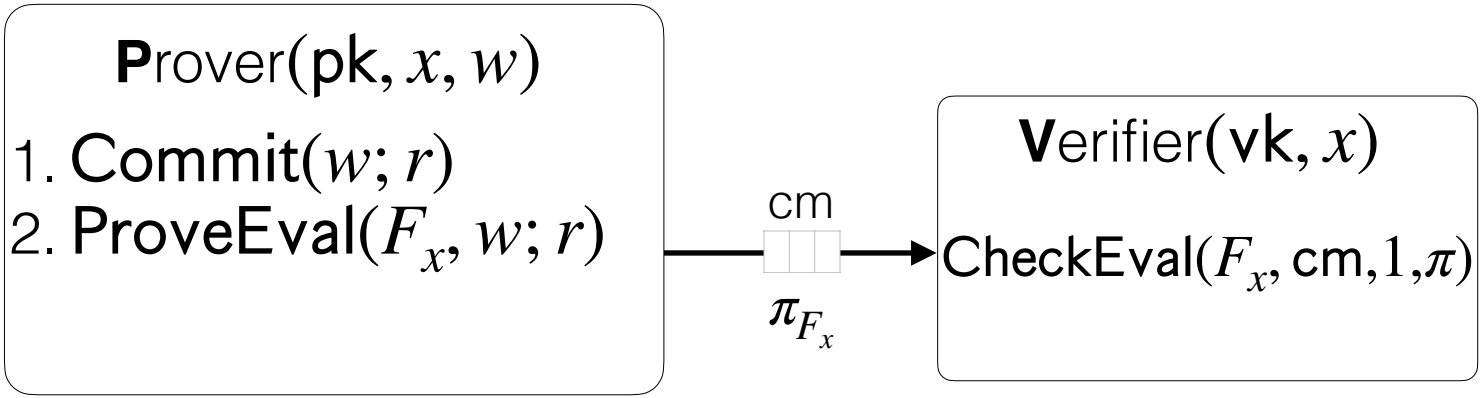
Idea: Ask Prover to help

Triple of algorithms with the following syntax:

- Commit$(m; r) \rightarrow$ cm
  - Commits to the message
- ProveEval$(F, m; r) \rightarrow (F(m), \pi)$
  - Returns proof of correct evaluation of $F(m)$
- CheckEval$(F,$ cm$, v, \pi) \rightarrow b \in \{0,1\}$
  - Checks that π is a valid proof that $F(m) = v$, where m is the msg inside cm

Does this work?

Strawman 4: Functional Commitments



Completeness: Follows from that of (ProveEval, CheckEval)

Knowledge Soundness: Ditto

ZK: Follows from hiding

Succinct pf size: Follows if eval. proof is succinct

Are we done?

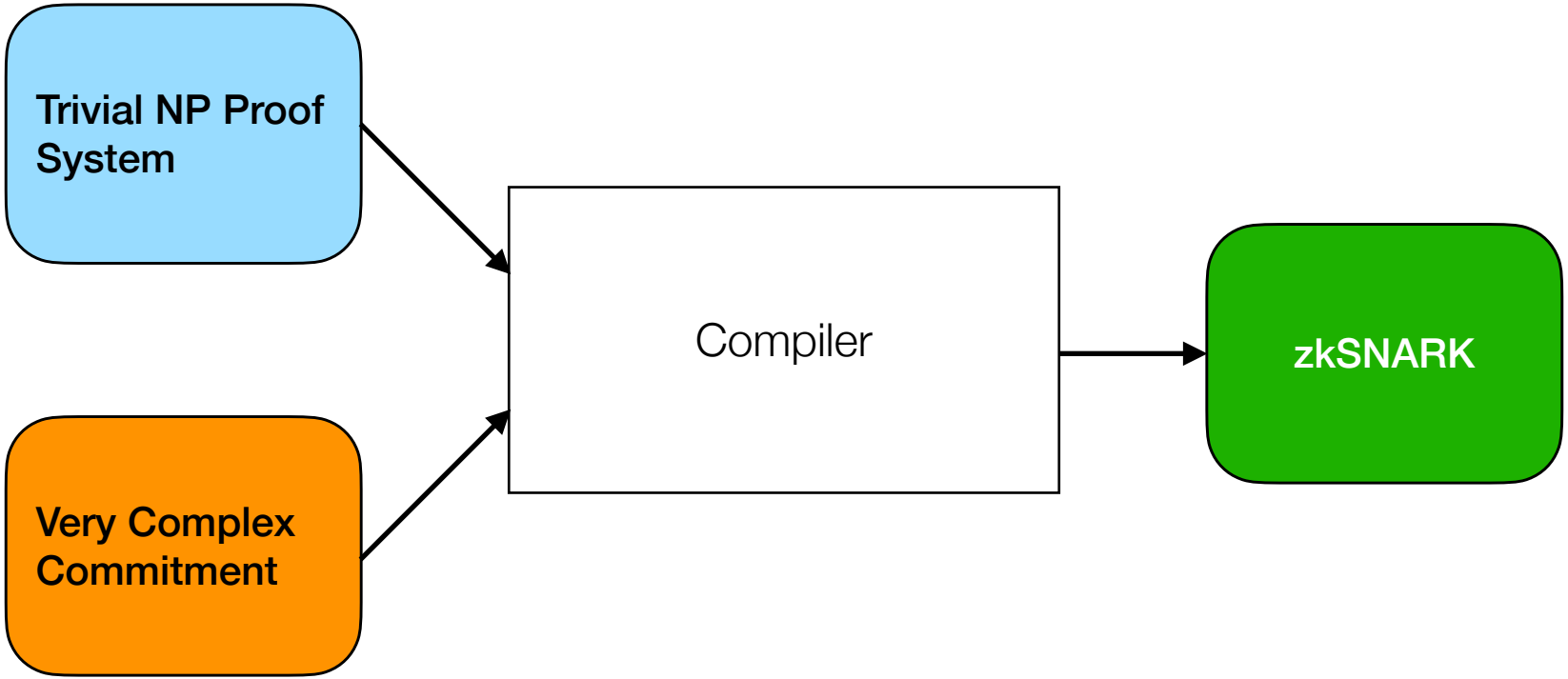
No! We just pushed the problem one layer down!

Problem: This is a zkSNARK for F!

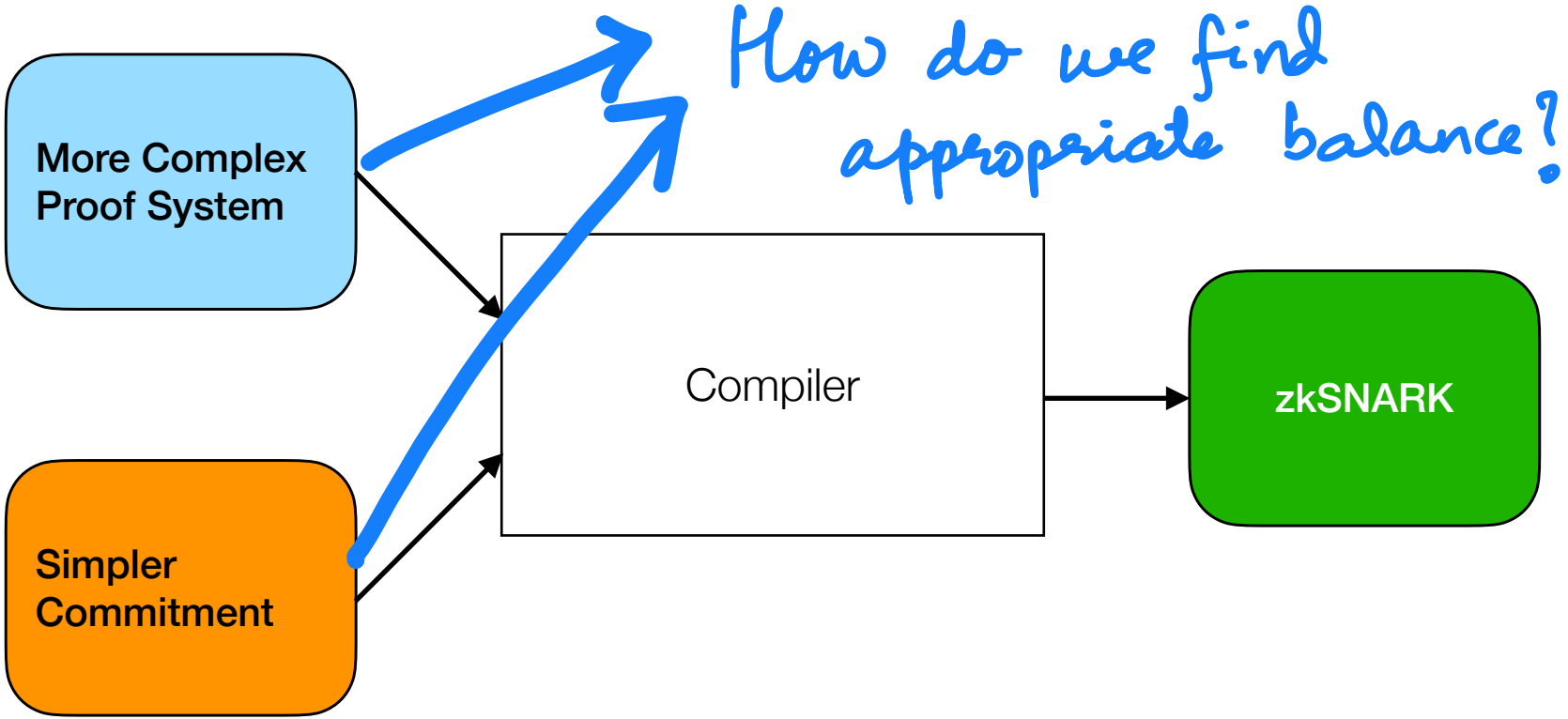
Triple of algorithms with the following syntax:

- Commit$(m; r) \rightarrow$ cm
  - Commits to the message
- ProveEval$(F, m; r) \rightarrow (F(m), \pi)$
  - Returns proof of correct evaluation of $F(m)$
- CheckEval$(F,$ cm$, v, \pi) \rightarrow b \in \{0,1\}$
  - Checks that π is a valid proof that $F(m) = v$, where m is the msg inside cm

Let's Reassess Our Status



How about we rebalance?



What commitment schemes exist?

Polynomial commitments:

- $F_z(m)$: Interpret m as univariate poly $f(X)$ in $\mathbb{F}[X]$ and evaluate at z

Multilinear commitments:

e.g., $f(x_1, \ldots, x_k) = x_1x_3 + x_1x_4x_5 + x_7$

- $F_{\vec{z}}(m)$: Interpret m as multilinear poly $f(X)$ in $\mathbb{F}[\overrightarrow{X}]$ and evaluate at $\vec{z}$

Vector commitments:

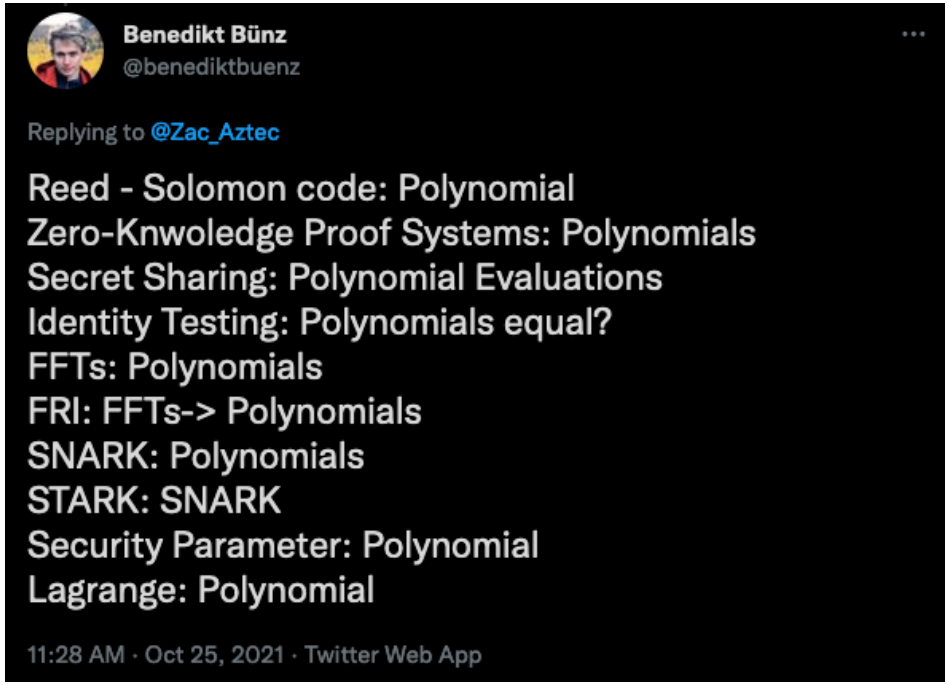
- $F_i(m)$: Interpret m as vector v in $\mathbb{F}^n$ and return v_i

Inner-product commitments:

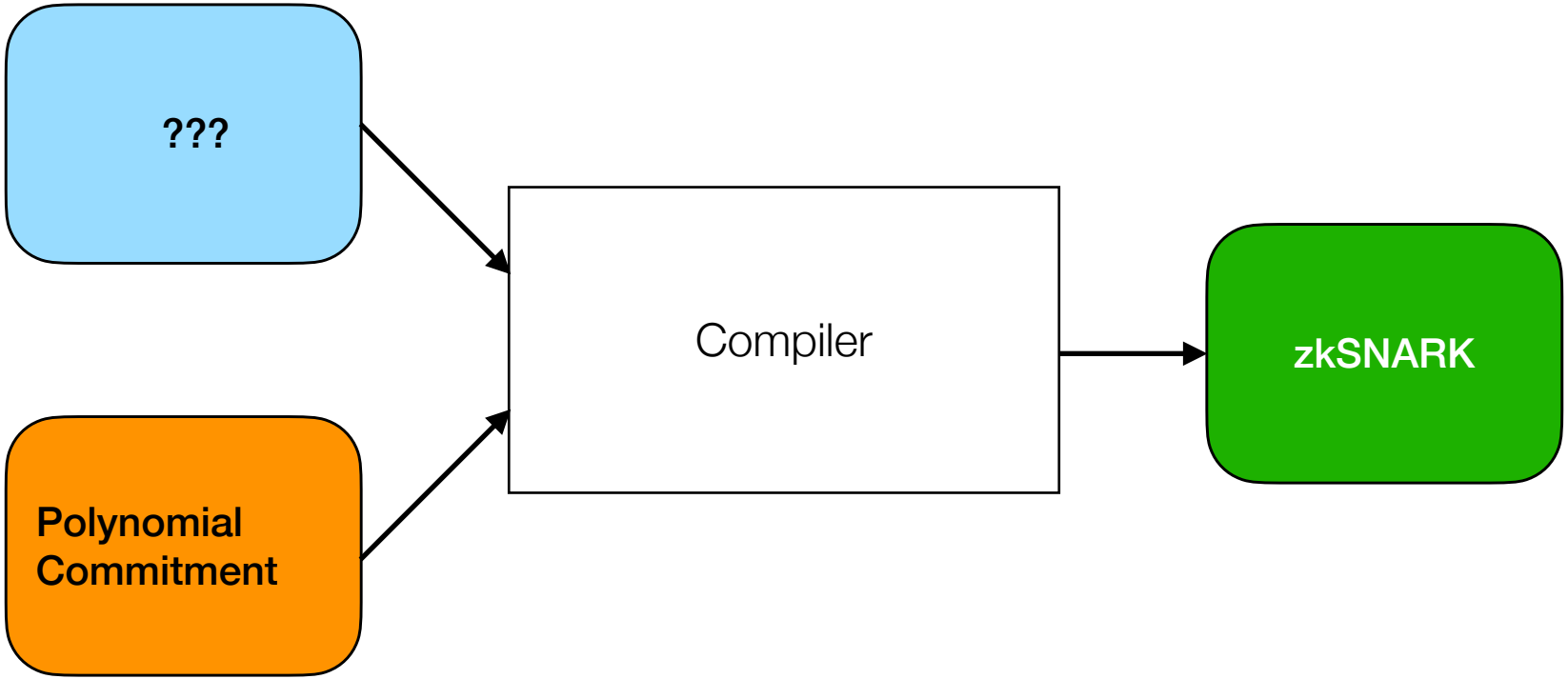
- $F_{\vec{q}}(m)$: Interpret m as vector $\vec{v}$ in $\mathbb{F}^n$ and return $\langle \vec{v}, \vec{q} \rangle$

Which to pick?

A: Polynomials!

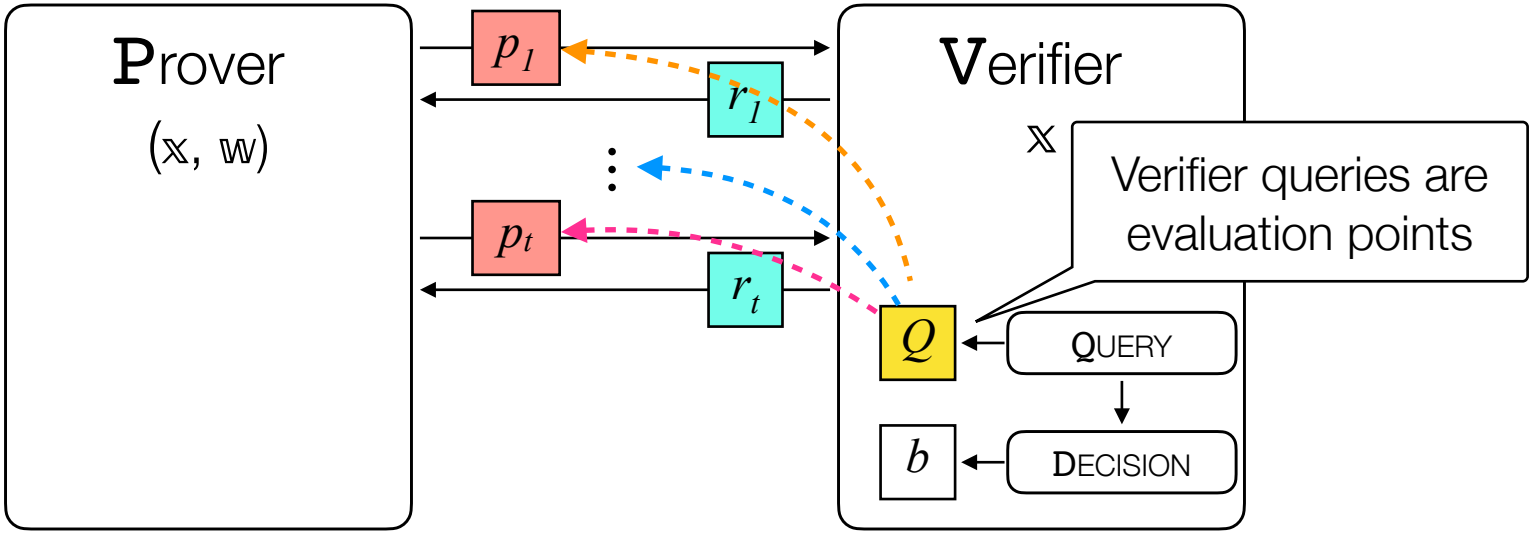


Let’s pick polynomials



Polynomial Interactive Oracle Proofs

Polynomial IOPs [GWC19, CHMMVW20, BFS20]



- **Completeness**: Whenever $(\mathbb{x}, \mathbb{w}) \in R$, there is a strategy for $\mathbf{P}$ that outputs **only polynomials**, and which causes $\mathbf{V}$ to accept.
- **Knowledge Soundness**: Whenever $\mathbf{V}$ accepts against a $\mathbf{P}$ that outputs **only polynomials**, then $\mathbf{P}$ "knows" $\mathbb{w}$ such that $(\mathbb{x}, \mathbb{w}) \in R$.

Majority of innovation is in PIOPs

Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings

Mary Maller
mary.maller.15@ucl.ac.uk
University College London

Sean Bowe
sean@z.cash
Electric Coin Company

Markulf Kohlweiss
mkohlwei@ed.ac.uk
University of Edinburgh
IOHK

Sarah Meiklejohn
s.meiklejohn@ucl.ac.uk
University College London

Plonk: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge

Ariel Gabizon*
Aztec

Zachary J. Williamson
Aztec

Oana Ciobotaru

Flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size

Ariel Gabizon
Function Technologies

Dmitry Khovratovich
Ethereum Foundation

MARLIN:
Preprocessing zkSNARKs
with Universal and Updatable SRS

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Yuncong Hu
yuncong_hu@berkeley.edu
UC Berkeley

Mary Maller
mary.maller.15@ucl.ac.uk
UCL

Pratyush Mishra
pratyush@berkeley.edu
UC Berkeley

Psi Vesely
psi@ucsd.edu
UCL

Nicholas Ward
npward@berkeley.edu
UC Berkeley

Spartan: Efficient and general-purpose zkSNARKs without trusted setup

Srinath Setty
Microsoft Research

HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates

Binyi Chen
Espresso Systems

Benedikt Bünz
Stanford University,
Espresso Systems

Dan Boneh
Stanford University

Zhenfei Zhang
Espresso Systems

Lunar: a Toolbox for More Efficient Universal and Updatable zkSNARKs and Commit-and-Prove Extensions

Matteo Campanelli[1], Antonio Faonio[2], Dario Fiore[3], Anaïs Querol[3,4], and Hadrián Rodríguez[3]

Caulk: Lookup Arguments in Sublinear Time

Arantxa Zapico*[1], Vitalik Buterin[2], Dmitry Khovratovich[2], Mary Maller[2], Anca Nitulescu[3], and Mark Simkin[2]

[1] Universitat Pompeu Fabra†
[2] Ethereum Foundation‡
[3] Protocol Labs§

plookup: A simplified polynomial protocol for lookup tables

Ariel Gabizon
Aztec

Zachary J. Williamson
Aztec

cq:* Cached quotients for fast lookups

Liam Eagen
Blockstream

Dario Fiore
IMDEA software institute

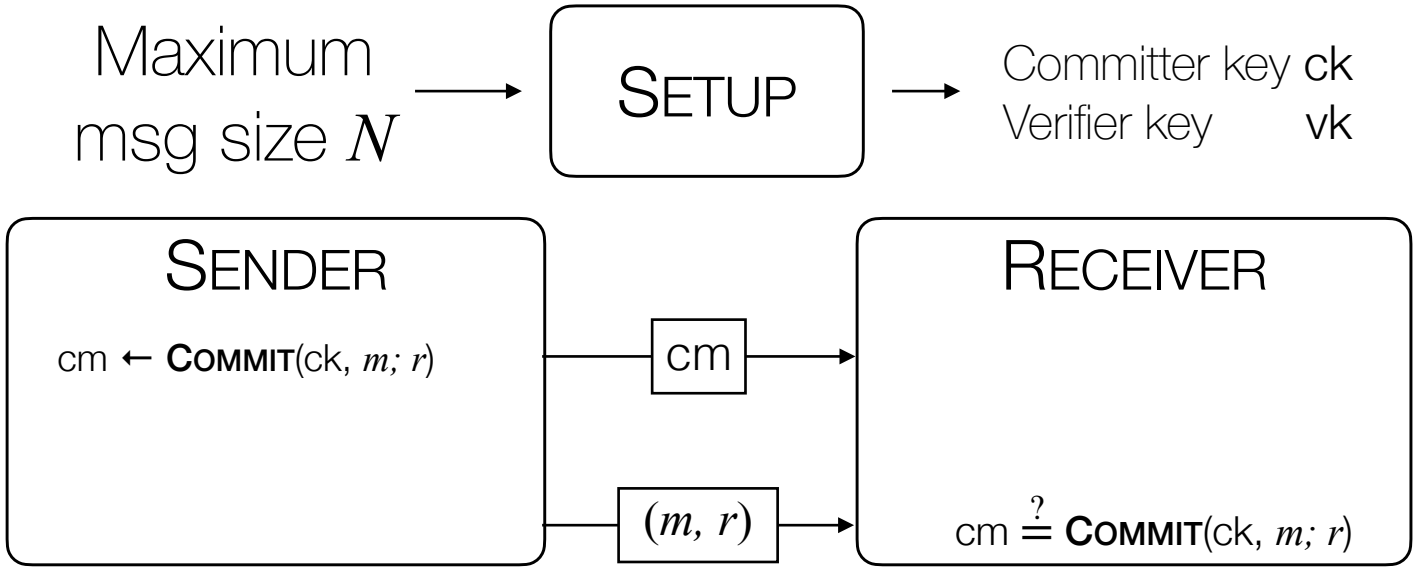
Ariel Gabizon
Zeta Function Technologies

Ba*loo*: Nearly Optimal Lookup Arguments

Arantxa Zapico*, Ariel Gabizon[3], Dmitry Khovratovich[1], Mary Maller[1], and Carla Ràfols[2]

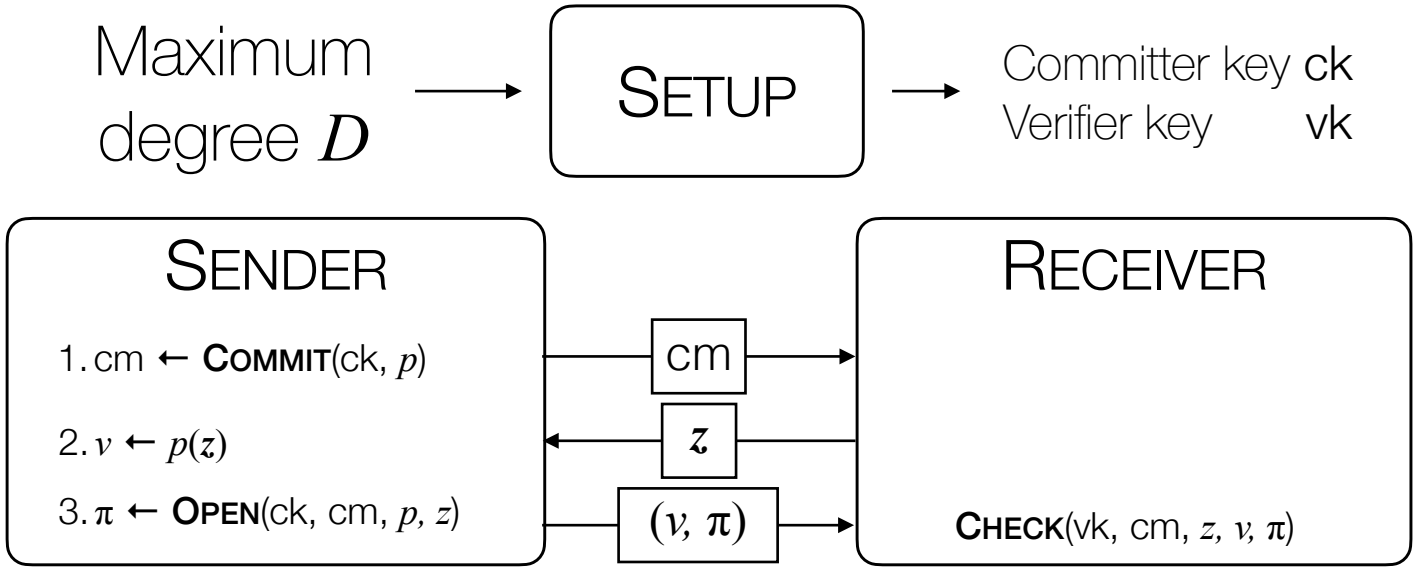
Polynomial Commitments

Recall: Commitment Schemes



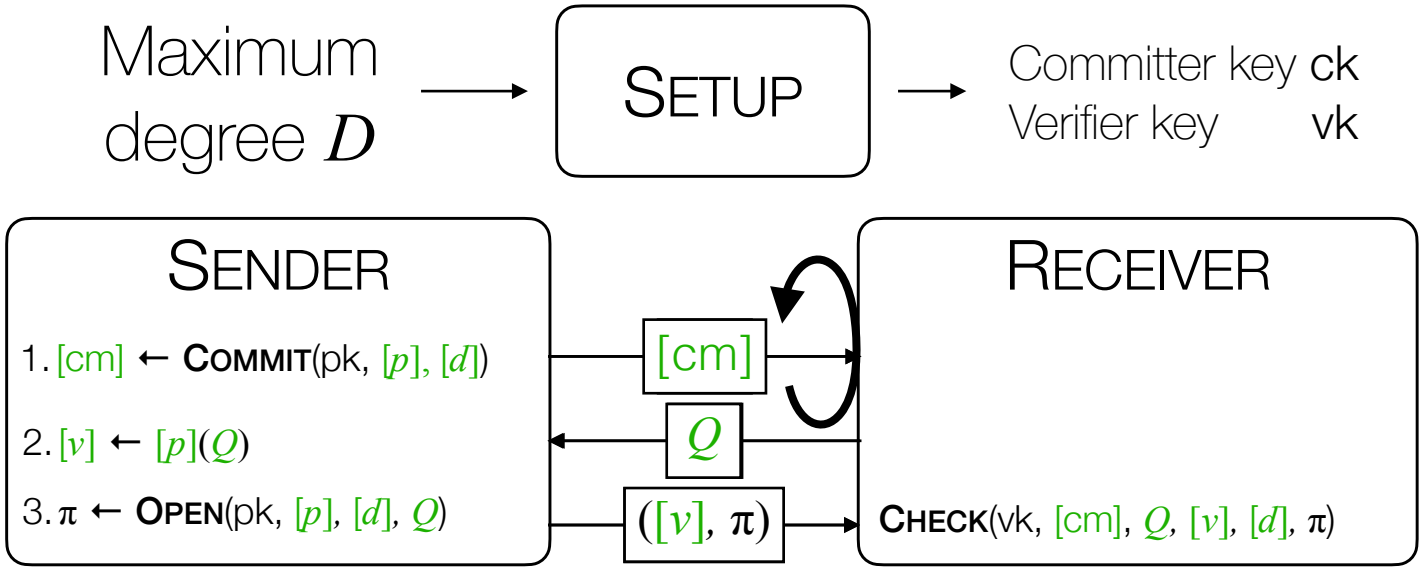
- **Binding**: For $m_1 \neq m_2$, Commit(ck, m; r_1) $\neq$ Commit(ck, m; r_2), for any r_1, r_2
- **Hiding**: **cm** reveals *no* information about m before reveal

Polynomial Commitments



- **Completeness**: Whenever $p(z) = v$, **R** accepts.
- **Extractability**: Whenever **R** accepts, **S**'s commitment **cm** "contains" a polynomial p of degree at most D.
- **Hiding**: **cm** and π reveal *no* information about p other than v

Polynomial Commitments



For efficiency improvements, you need

- Batch commitment
- Batch opening

A selection of constructions

In the last 10 years, several constructions with different

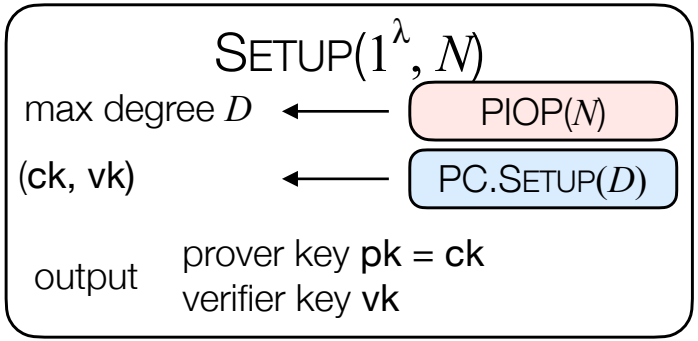
- Cryptographic assumptions
- Prover and verifier efficiency and proof sizes
- Homomorphism and batching properties

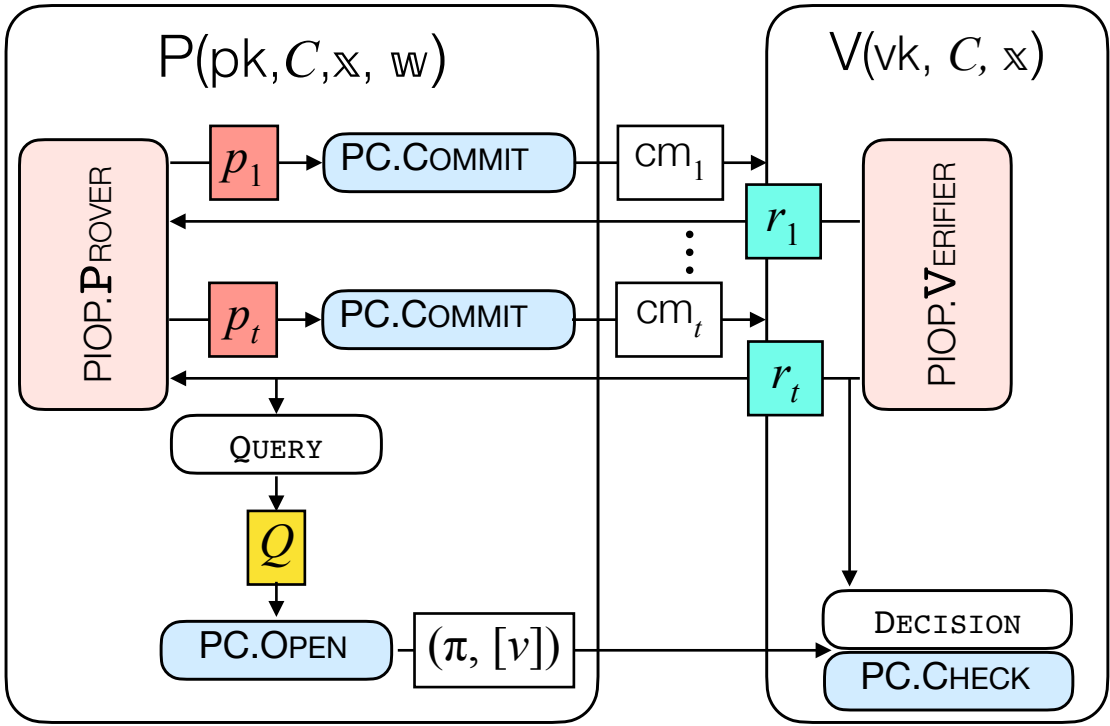
Looking ahead, this enables SNARKs with many different properties

	KZG10	PST13	IPA	Hyrax	Dory	BFS20
crypto	Pairings	Pairings	DLog + RO	DLog + RO	Pairing + RO	GUO + RO
# variables	1	m	1	m	1	1
setup type	Private	Private	Public	Public	Public	Public
commitment size	$O(1)$ G	$O(1)$ G	$O(1)$ G	$O(2^{m/2})$ G	$O(1)$ G	$O(1)$ G
proof size	$O(1)$ G	$O(m)$ G	$O(\log d)$ G	$O(2^{m/2})$ G	$O(\log d)$ G	$O(\log d)$ G
verifier time	$O(1)$ G	$O(m)$ G	$O(d)$ G	$O(2^{m/2})$ G	$O(\log d)$ G	$O(\log d)$ G

PIOP + PC = SNARK

PIOPs + PC Schemes → SNARK





+ Fiat—Shamir to get non-interactivity

Properties

- **Completeness**: Follows from completeness of **PC** and **AHP**.
- **Proof of Knowledge**: Whenever **V** accepts but $C(\mathbb{x}, \mathbb{w}) = 0$, we can construct either an adversarial prover against **PIOP**, or an adversary that breaks extractability of **PC**.
- **Zero Knowledge**: Follows from hiding of **PC** and bounded-query ZK of **AHP**.
- **Verifier efficiency**:
  $T(\textbf{ARG.V}\textsc{erify}) = T(\textbf{PIOP.V}\textsc{erify}) + T(\textbf{PC.C}\textsc{heck})$