Building a SNARK, Part I

Dan Boneh
Stanford University

Review: Preprocessing argument systems

Public arithmetic circuit: $C(\,x,\ w\,) \rightarrow \mathbb{F}$

public statement in $\mathbb{F}^n$ ⟶ x

secret witness in $\mathbb{F}^m$ ⟶ w

Preprocessing (setup): $\mathbf{S}(C) \rightarrow$ public parameters $(\,S_p, S_v\,)$

$S_p,\ x,\ w$

Prover

proof π

(I know a w s.t. $C(x, w) = 0$)

$S_v,\ x$

Verifier

accept or reject

Preprocessing argument System

A **preprocessing argument system** is a triple (S, P, V):

- **S**(C) → public parameters (S_p, S_v) for prover and verifier
- **P**$(S_p, \boldsymbol{x}, \boldsymbol{w})$ → proof π
- **V**$(S_v, \boldsymbol{x}, \boldsymbol{\pi})$ → accept or reject

Requirements (informal)

Prover P(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$) $\xrightarrow{\text{proof } \pi}$ Verifier V (S_v, $\boldsymbol{x}$, $\boldsymbol{\pi}$)

accept or reject

Complete: $\forall x, w$: $C(\boldsymbol{x}, \boldsymbol{w}) = 0 \Rightarrow$ Pr[V(S_v, $\boldsymbol{x}$, P(S_p, $\boldsymbol{x}$, $\boldsymbol{w}$)) = accept] = 1

Knowledge sound: V accepts $\Rightarrow$ P "knows" $\boldsymbol{w}$ s.t. $C(\boldsymbol{x}, \boldsymbol{w}) = 0$

example: P "knows" $\boldsymbol{w}$ s.t. [$H(\boldsymbol{w}) = \boldsymbol{x}$ and $0 \leq \boldsymbol{w} \leq 2^{128}$]

Optional: **Zero knowledge**: $(S_v, \boldsymbol{x}, \pi)$ "reveals nothing" about $\boldsymbol{w}$

SNARK: a Succinct ARgument of Knowledge

A **succinct preprocessing argument system** is a triple (S, P, V):

- **S**(C) → public parameters (S_p, S_v) for prover and verifier
- **P**$(S_p, \boldsymbol{x}, \boldsymbol{w})$ → **short** proof π ; $|\pi| = O_\lambda(\, \mathbf{log}(|C|)\,)$
- **V**$(S_v, \boldsymbol{x}, \boldsymbol{\pi})$ **fast to verify** ; $\text{time(V)} = O_\lambda(\, |x|,\ \mathbf{log}(|C|)\,)$

short "summary" of circuit

$\lambda :=$ security parameter = 128

Building an efficient SNARK

General paradigm: two steps

(1) A functional commitment scheme

(2) A compatible interactive oracle proof (IOP)

(zk)SNARK for general circuits

Let’s explain each concept …

(1) Commitments

Cryptographic commitment: emulates an envelope

Recall: commitments

Two algorithms:

- *commit*(m, r) → ***com***   (r chose at random)
- *verify*$(m,$ ***com***$, r)$ → accept or reject

Properties:

- **binding**: cannot produce ***com*** and two valid openings for ***com***
- **hiding**: ***com*** reveals nothing about committed data

A standard construction

Fix a hash function $H : \mathcal{M} \times \mathcal{R} \to C$

commit(m, r): **com** $:= H(m, r)$

verify$(m,$ ***com***$, r)$: accept if ***com*** $= H(m, r)$

Hiding and Binding for a suitable function H

Committing to a function

choose a family of functions $\mathcal{F} = \{f: X \to Y\}$

Prover		Verifier
$f \in \mathcal{F}$	$\mathbf{com}_f := commit(f, r)$ →	$\mathbf{com}_f$
	← $x \in X$	
	$y \in Y$, proof π →	accept or reject

proof that: $f(x) = y$ and $f \in \mathcal{F}$

Committing to a function: syntax

A **functional commitment** scheme for $\mathcal{F}$:

- <u>*setup*</u>$(\lambda) \rightarrow$ *pp*, outputs public parameters *pp*
- <u>*commit*</u>$(pp, f, r) \rightarrow$ $\boldsymbol{com_f}$ commitment to $f \in \mathcal{F}$ with $r \in \mathcal{R}$

  a **binding** (and optionally **hiding**) commitment scheme for $\mathcal{F}$

- <u>*eval*</u>(Prover P, verifier V): for a given $\boldsymbol{com_f}$ and $x \in X$, $y \in Y$:

  $\mathrm{P}(pp, f, x, y, r) \rightarrow$ short proof π

  $\mathrm{V}(pp, \boldsymbol{com_f}, x, y, \pi) \rightarrow$ accept/reject

a SNARK for the relation:

$f(x) = y$ and $f \in \mathcal{F}$ and *commit*$(pp, f, r) = \boldsymbol{com_f}$

Three examples

Polynomial commitments:

- Committing to a univariate polynomial $f(X)$ in $\mathbb{F}_p^{(\leq d)}[X]$

(univariate polynomials of degree at most d)

Multilinear commitments:

- Committing to a multilinear polynomial in $\mathbb{F}_p^{(\leq 1)}[X_1, \ldots, X_k]$

e.g., $f(x_1, \ldots, x_k) = x_1 x_3 + x_1 x_4 x_5 + x_7$

Linear commitments:

- Committing to a linear function $f_{\vec{v}}(\vec{u}) = \langle \vec{u}, \vec{v} \rangle = \sum_{i=1}^{n} u_i v_i$

Polynomial Commitment Scheme (PCS)

A PCS is a functional commitment for the family $\mathcal{F} = \mathbb{F}_p^{(\le d)}[X]$

$\implies$ prover commits to a univariate polynomial f in $\mathbb{F}_p^{(\le d)}[X]$, later, can prove that $v = f(u)$ for public $u, v \in \mathbb{F}_p$

Proof size and verifier time should be $O_\lambda(\log d)$

Examples:

- Using basic elliptic curves: Bulletproofs (verifier's work is linear in d)
- Using bilinear groups: KZG'10 (trusted setup), Dory'20
- Using groups of unknown order: Dark'20
- Using hash functions only: based on FRI

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

Group $\mathbb{G} := \{\, 0,\ \ G,\ \ 2 \cdot G,\ \ 3 \cdot G\,,\ \dots\,,\ (p-1) \cdot G \,\}$ of order p.

<u>*setup*</u>$(\lambda) \rightarrow$ *pp:*

- Sample random $\alpha \in \mathbb{F}_p$
- *pp* $= \left(H_0 = G,\ \ H_1 = \alpha \cdot G,\ \ H_2 = \alpha^2 \cdot G,\ \ \dots,\ \ H_d = \alpha^d \cdot G\right) \in \mathbb{G}^{d+1}$
- delete α !! (trusted setup)

a binding commitment, but not hiding

<u>*commit*</u>$(pp, f) \rightarrow$ ***com$_f$*** where ***com$_f$*** $:= f(\alpha) \cdot G \in \mathbb{G}$

- $f(X) = f_0 + f_1 X + \cdots + f_d X^d \quad \Rightarrow \quad$ ***com$_f$*** $= f_0 \cdot H_0 + \cdots + f_d \cdot H_d$

$$= f_0 \cdot G + f_1 \alpha \cdot G + f_2 \alpha^2 \cdot G + \cdots = f(\alpha) \cdot G$$

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

commit$(pp, f) \rightarrow$ $\mathbf{com}_f$ where $\mathbf{com}_f = \mathrm{f}(\alpha) \cdot \mathrm{G} \in \mathbb{G}$

eval: *Prover(pp, **f**, u, v)* *Verifier(pp, **com**$_f$, u, v)*

Goal: prove $f(u) = v$

$f(u) = v \iff u$ is a root of $\hat{f} := f - v \iff (X\text{-}u)$ divides $\hat{f}$

$\iff$ exists $q \in \mathbb{F}_p[X]$ s.t. $q(X)\cdot(X\text{-}u) = f(X) - v$

compute $q(X)$ and $\mathbf{com}_q$

$\xrightarrow{\pi := \mathbf{com}_q \in \mathbb{G}}$

(short: proof size indep. of d)

accept if

$(\alpha - u)\cdot\mathbf{com}_q = \mathbf{com}_f - v\cdot G$

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

comm

How to prove that this is a secure PCS? Not today …

eval: Prover(pp, **f**, u, v) Verifier(pp, $\mathbf{com_f}$, u, v)

Goal: prove

An expensive computation for large d

oot of

Verifier does not know α

$\Rightarrow$ uses a "pairing"

(and only needs G, H_1 from *pp*)

$\Leftrightarrow$ exists $q \in \mathbb{F}_p$ [X]

compute $q(X)$ and $\mathbf{com_q}$

$\pi := \mathbf{com_q} \in \mathbb{G}$

(short: proof size indep. of d)

accept if

$(\alpha - u) \cdot \mathbf{com_q} = \mathbf{com_f} - v \cdot G$

The KZG poly-commit scheme (Kate-Zaverucha-Goldberg'2010)

Generalizations:

- KZG for committing to ***k*-variate polynomials** [PST'13] (eprint/2011/587)

  ... but eval proof size is *k* group elements

- **Batch proofs**:
  - suppose verifier has commitments ***com$_{f1}$* , ... *com$_{fn}$***
  - prover wants to prove $f_i(u_{i,j}) = v_{i,j}$ for $i \in [n]$, $j \in [m]$

  $\Rightarrow$ batch proof π is one or two group elements !

The Dory polynomial commitment (eprint/2020/1274)

Difficulties with KZG: trusted setup for *pp*, and *pp* size is linear in *d*.

<u>Dory</u>:

- **transparent setup**: no secret randomness in setup
- ***com***$_f$ is a single group element (independent of degree d)
- eval proof size for $f \in \mathbb{F}_p^{(\leq d)}[X]$ is $\boxed{\text{O}(\log d)}$ group elements
- eval verify time is $\boxed{\text{O}(\log d)}$ Prover time: $\boxed{O(d)}$

PCS have many applications

Example: **vector commitment** (a drop-in replacement for Merkle trees)

Bob: vector $(u_1, \dots, u_k) \in \mathbb{F}_p^{(\leq d)}$

interpolate poly f s.t.:

$f(i) = u_i$ for $i = 1, \dots, k$

Alice

$\mathbf{com}_u :=$ commit(pp, f)

prove $u_2 = a$, $u_4 = b$

$\pi :=$ eval proof that $f(2) = a$, $f(4) = b$

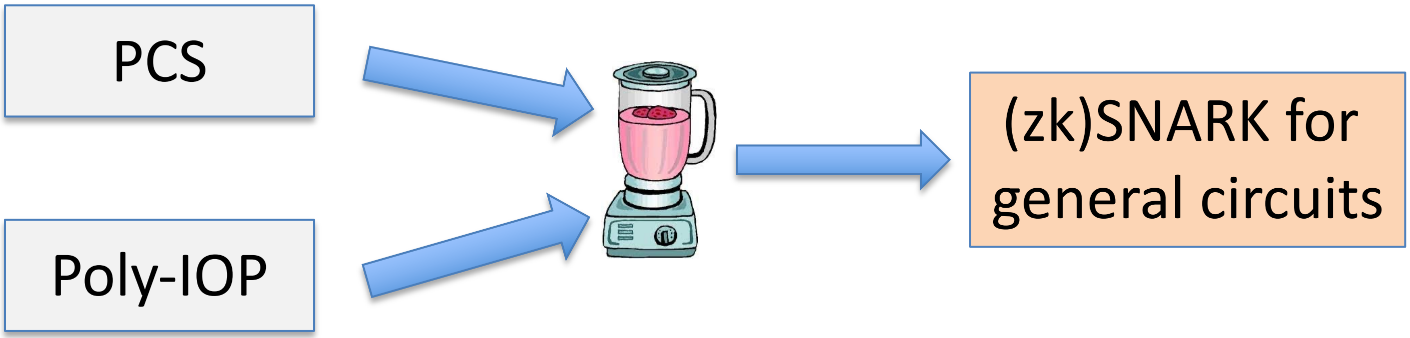
$\pi \in \mathbb{G}$

(KZG: π is a single group element)

shorter than a Merkle proof

accept or reject

Component 2:   Polynomial IOP



Component 2: Polynomial IOP

Let $C(x, w)$ be some arithmetic circuit. Let $x \in \mathbb{F}_p^n$.

Poly-IOP: a proof system that proves $\exists w\colon C(x, w) = 0$ as follows:

Setup(C) $\rightarrow$ public parameters $\boldsymbol{S_p}$ and $\boldsymbol{S_v}$ = ($\boxed{f_0}$, $\boxed{f_{-1}}$, … , $\boxed{f_{-s}}$)

Polynomial IOP

Prover P(s_p,x,w)

Verifier V(s_v, x)

commit $\boxed{f_1 \in \mathbb{F}_p^{(\leq d)}[X]}$

$r_1 \leftarrow \mathbb{F}_p$

r_1

commit $\boxed{f_2 \in \mathbb{F}_p^{(\leq d)}[X]}$

$r_2 \leftarrow \mathbb{F}_p$

r_2

$\vdots$

r_{t-1}

$r_{t-1} \leftarrow \mathbb{F}_p$

commit $\boxed{f_t \in \mathbb{F}_p^{(\leq d)}[X]}$

$\text{verify}^{f_{-s},\ldots,f_t}(\mathbf{x}, r_1, , \ldots, r_{t-1})$

fast verify that can evaluate f_i at any point in $\mathbb{F}_p$ (outputs yes/no)

Properties

- **Complete**: if $\exists w: C(x,w) = 0$ then verifier always accepts

- **Knowledge sound**: (informal) Let $x \in \mathbb{F}_p^n$.

  for every P* that convinces the verifier with prob. ≥ $1/10^6$

  there is an efficient extractor E s.t.

  $$\Pr[\ E(x, f_1, r_1, \dots, r_{t-1}, f_t) \rightarrow w \ \text{ s.t. } C(x,w) = 0\ ] \geq 1/10^6 - \varepsilon$$

- Optional: **zero knowledge** (for a zk-SNARK)

The resulting SNARK

(t, q) Poly-IOP: $t :=$ #polys. committed, $q :=$ # eval queries in verify

(usually $t, q \leq 3$)

<u>The SNARK</u>:

- Prover sends t poly commitments
- During poly-IOP verify: run PCS eval protocol q times
- Use **Fiat-Shamir** to make the proof system non-interactive

Length of SNARK proof: t poly-commits + q eval proofs

Verifier time: $q \times$ time(eval verify) + time(IOP-verify)

Prover time: $t \times$ time(commit) + $q \times$ time(prove) + time(IOP-prover)

END OF LECTURE

Next lecture: Constructing a Poly-IOP