Building a SNARK

Dan Boneh
Recap: high-level goals

- Private transactions on a public blockchain
- Blockchain scaling, such as proof-based Rollup
-Privately prove compliance, such as a private proof of solvency
Recap: non-interactive proof systems (for NP)

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

Let \( \mathbf{x} \in \mathbb{F}_p^n \). Two standard goals for prover P:

1. **Soundness**: convince Verifier that \( \exists \mathbf{w} \text{ s.t. } C(\mathbf{x}, \mathbf{w}) = 0 \)
   (e.g., \( \exists \mathbf{w} \text{ such that } [H(\mathbf{w}) = \mathbf{x} \text{ and } 0 < \mathbf{w} < 2^{60}] \))

2. **Knowledge**: convince Verifier that P “knows” \( \mathbf{w} \) s.t. \( C(\mathbf{x}, \mathbf{w}) = 0 \)
   (e.g., P knows a \( \mathbf{w} \) such that \( H(\mathbf{w}) = \mathbf{x} \))
A non-interactive proof system is a triple \((S, P, V)\):

- \(S(C) \rightarrow\) public parameters \((S_p, S_v)\) for prover and verifier
  
  \((S_p, S_v)\) is called a reference string

- \(P(S_p, x, w) \rightarrow\) proof \(\pi\)

- \(V(S_v, x, \pi) \rightarrow\) accept or reject
proof systems: properties (informal)

Prover $P(pp, x, w)$  
Verifier $V(pp, x, \pi)$

$\Pi$  
proof accept or reject

Complete: $\forall x, w: C(x, w) = 0 \Rightarrow V(S_v, x, P(S_p, x, w)) = \text{accept}$

Proof of knowledge: $V$ accepts $\Rightarrow$ P “knows” $w$ s.t. $C(x, w) = 0$

in some cases, soundness is sufficient: $\exists w$ s.t. $C(x, w) = 0$

Zero knowledge (optional): $(x, \pi)$ “reveals nothing” about $w$
SNARK: succinct argument of knowledge

Goal: P wants to show that it knows $w$ s.t. $C(x, w) = 0$

**Succinct:**

- Proof $\pi$ should be short \[ i.e., |\pi| = O(\log(|C|), \lambda) \]
- Verifying $\pi$ should be fast \[ i.e., \text{time}(V) = O(|x|, \log(|C|), \lambda) \]

Note: if SNARK is zero-knowledge, then called a zkSNARK
A simple PCP-based SNARK

[Kilian’92, Micali’94]
The PCP theorem: Let $C(x, w)$ be an arithmetic circuit.
there is a proof system that for every $x$ proves $\exists w: C(x, w) = 0$ as follows:

Prover $P(S_p, x, w)$

- proof $\pi$

Verifier $V(S_v, x)$

- read only $O(\lambda)$ bits of $\pi$
- output accept or reject

V always accepts valid proof. If no $w$, then V rejects with high prob.

size of proof is $\text{poly}(|C|)$. (not succinct)
Converting a PCP proof to a SNARK

Prover $P(S_p, x, w)$

Verifier $V(S_v, x)$

Merkle root $h$

Open $k$ positions of $\pi$ ($k = O(\lambda)$)

$k$ opening and Merkle proofs

Verifier sees $O(\lambda \log |C|)$ data $\Rightarrow$ succinct proof.

Problem: interactive
Making the proof non-interactive

The Fiat-Shamir heuristic:

- public-coin interactive protocol $\Rightarrow$ non-interactive protocol
  
  public coin: all verifier randomness is public (no secrets)

Prover $P(S_p, x, w)$

Verifier $V(S_v, x)$

- $r$ chosen randomly
- $\text{msg1}$
- $\text{msg2}$
- accept or reject
Making the proof non-interactive

**Fiat-Shamir heuristic:** \( H: M \rightarrow R \) is a cryptographic hash function

- idea: prover generates random bits on its own (!)

---

**Prover** \( P(S_p, x, w) \)

- generate msg1
- \( r \leftarrow H(x, \text{msg1}) \)
- generate msg2

**Verifier** \( V(S_v, x) \)

\[ \pi = (\text{msg1, msg2}) \]

\[ |\pi| = O(\lambda \log |C|) \]

- \( r \leftarrow H(x, \text{msg1}) \)
- accept or reject

**Thm:** this is a secure SNARK assuming \( H \) is a random oracle
Let’s build an extractor $E$ for the interactive protocol:

- After prover commits to Merkle root of proof
  $E$ asks prover to open many batches of $k = O(\lambda)$ positions of $\pi$
  (by rewinding prover)

- $E$ fails to extract cell #j of $\pi$ if
  1. prover produces a false Merkle proofs (efficient prover cannot), or
  2. prover fails (i.e., verifier rejects) whenever j is in batch to open:

  $\Pr[\text{prover fails}] \geq \Pr[\text{j in batch}] = 1 - (1 - 1/|\pi|)^k$.
  
  so: this cannot happen if $k$ is sufficiently large

$\Rightarrow E$ extracts entire proof $\pi$. Once $\pi$ is known, $E$ can obtain $w$ from $\pi$. 
Are we done?

Simple transparent SNARK from the PCP theorem

• Use Fiat-Shamir heuristic to make non-interactive
• We will apply Fiat-Shamir in many other settings

The bad news: an impractical SNARK --- Prover time too high

Better SNARKs: Goal: $\text{Time}(\text{Prover}) = \tilde{O}(|C|)$
Building an efficient SNARK
Many SNARKs are built in two steps:

- Polynomial interactive oracle proofs (poly-IOP)
- Polynomial commitment scheme

(zk)SNARK for general circuits
Recall: commitments

Two algorithms:

- \texttt{commit}(m, r) \rightarrow \texttt{com}
- \texttt{verify}(m, \texttt{com}, r) \rightarrow \text{accept or reject}

Properties:

- binding: cannot produce two valid openings for \texttt{com}.
- hiding: \texttt{com} reveals nothing about committed data
Polynomial commitment schemes

Notation:

Fix a finite field: \( \mathbb{F}_p = \{0, 1, \ldots, p - 1\} \)

\( \mathbb{F}_p^{(\leq d)}[X] \): all polynomials in \( \mathbb{F}_p[X] \) of degree \( \leq d \).
(1) Polynomial commitment schemes

- **setup**($d$) $\rightarrow$ $pp$, public parameters for polynomials of degree $\leq d$
- **commit**($pp$, $f$, $r$) $\rightarrow$ $com_f$ commitment to $f \in \mathbb{F}_p^{(\leq d)}[X]$
- **eval**: goal: for a given $com_f$ and $x, y \in \mathbb{F}_p$, prove that $f(x) = y$.

Formally: $eval = (P, V)$ is a SNARK for:

- statement $st = (pp, com_f, x, y)$ with witness = $w = (f, r)$
- where $C(st, w) = 0$ iff

$$\left[ f(x) = y \ \text{and} \ f \in \mathbb{F}_p^{(\leq d)}[X] \ \text{and} \ \text{commit}(pp, f, r) = com_f \right]$$
(1) Polynomial commitment schemes

Properties:

• Binding: cannot produce two valid openings \((f_1, r_1), (f_2, r_2)\) for \(\text{com}_f\).
• eval is an argument of knowledge (can extract \((f, r)\) from a successful prover)
• optional:
  • commitment is hiding
  • eval is zero knowledge
Constructing polynomial commitments

Not today ... (see readings or CS355)

Properties of the best ones:

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

simple construction without this requirement
Goal: polynomial commitment scheme \( \Rightarrow \) SNARK for a general circuit \( C(x, w) \).

... done using a polynomial-IOP

Fix an arithmetic circuit \( C(x, w) \). Let \( x \in \mathbb{F}_p^n \).

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

Prover $P(x, w)$

- $f_1 \in \mathbb{F}_p^{(\leq d)} [X]$
- $r_1 \leftarrow \mathbb{F}_p$
- $f_2 \in \mathbb{F}_p^{(\leq d)} [X]$
- $r_2 \leftarrow \mathbb{F}_p$
- $\vdots$
- $r_{t-1} \leftarrow \mathbb{F}_p$
- $f_t \in \mathbb{F}_p^{(\leq d)} [X]$
- $r_{t-1} \leftarrow \mathbb{F}_p$

Verifier $V(x)$

- can evaluate $f_i$ at any $x$ in $\mathbb{F}_p$
- verify $f_1, \ldots, f_t(r_1, \ldots, r_{t-1})$
Properties

• complete: if $\exists w: C(x, w) = 0$ then verifier always accepts

• Soundness or proof of knowledge: (informal) Let $x \in \mathbb{F}_p^n$.

  $P^*$: a prover that convinces the verifier with prob. $\geq 1/10^6$

  then there is an efficient extractor $E$ s.t.

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

• Optional: zero knowledge
The resulting SNARK

Poly-IOP params:  \#polynomials = t, \# eval queries in verify = q

The SNARK:
• During interactive phase of poly-IOP: send t poly commitments
• During poly-IOP verify: run poly-commit eval protocol q times
• Use Fiat-Shamir to make the proof system non-interactive

Length of SNARK proof:  t poly-commits + q eval proofs
SNARK verify time:  q poly eval proof verifications + time(IOP-verify)
SNARK prover time:  t poly commits + time(IOP-prover)
First some useful tricks ...

The fundamental theorem of algebra: for $0 \neq f \in \mathbb{F}_p^{(\leq d)} [X]$

$$\text{for } r \leftarrow \mathbb{F}_p : \quad \Pr [ f(r) = 0 ] \leq \frac{d}{p}$$

$\Rightarrow$ suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ then $d/p$ is negligible

$\Rightarrow$ for $r \leftarrow \mathbb{F}_p$ , if $f(r) = 0$ then $f$ is identically zero w.h.p

$\Rightarrow$ simple zero test for a committed polynomial
Some useful gadgets

Let \( \omega \in \mathbb{F}_p \) be a primitive \( k \)-th root of unity \( (\omega^k = 1) \)

Set \( H := \{ 1, \omega, \omega^2, \ldots, \omega^{k-1} \} \).

Let \( f \in \mathbb{F}_p^{(\leq d)} [X] \) and \( b, c \in \mathbb{F}_p \). \( (d \geq k) \)

Want poly-IOPs for the following tasks:

**Task 1 (zero-test):** prove that \( f \) is identically zero on \( H \)

**Task 2 (sum-check):** prove that \( \sum_{a \in H} f(a) = b \)

**Task 3 (prod-check):** prove that \( \prod_{a \in H} f(a) = c \)
Zero test on H \( (H = \{1, \omega, \omega^2, ..., \omega^{k-1}\}) \)

**Prover** \(P(f, \bot)\)

\[
q(X) \leftarrow f(X)/(X^k - 1)
\]

\[
q \in \mathbb{F}^{(\leq d)}_p [X]
\]

**Verifier** \(V(f)\)

\[
r \leftarrow \mathbb{F}_p
\]

learn \(q(r), f(r)\)

accept if \(f(r) \equiv q(r) \cdot (r^k - 1)\)

(implies that \(f(X) = q(X)(X^k - 1)\))

**Thm:** this protocol is complete and sound, assuming \(d/p\) is negligible.

Verifier time: \(O(\log d)\) and one eval verify
Let \( t \in \mathbb{F}_p^{(\leq k)} [X] \) be the degree-\( d \) polynomial:

\[
t(1) = f(1), \quad t(\omega^s) = \prod_{i=1}^s f(\omega^i) \quad \text{for} \quad s = 1, \ldots, k - 1
\]

Then

\[
t(\omega^{k-1}) = \prod_{a \in H} f(a) = 1
\]

and

\[
t(\omega \cdot x) = t(x) \cdot f(\omega \cdot x) \quad \text{for all} \quad x \in H \quad \text{(including} \quad x = \omega^{k-1} \text{)}
\]

**Lemma**: if

1. \( t(\omega^{k-1}) = 1 \) and
2. \( t(\omega \cdot x) - t(x) \cdot f(\omega \cdot x) = 0 \) for all \( x \in H \)

then

\[
\prod_{a \in H} f(a) = 1
\]
Product check on H  (unoptimized)

Prover P((f, c), ⊥)

```
construct t(X) ∈ \mathbb{F}_p^{(≤k)} , \quad t_1(X) = t(\omega \cdot X) - t(X) \cdot f(\omega \cdot X)
and q(X) = t_1(X)/(X^k - 1) ∈ \mathbb{F}_p^{(≤k)}
```

Verifier V([f])

```
r ← \mathbb{F}_p

\begin{align*}
q, \ t & \in \mathbb{F}_p^{(≤k)}[X] \\
eval \ t(X) \ at \ \omega^{k-1}, \ r, \ \omega r
\end{align*}
```

```
eval \ q(X) \ at \ r, \ and \ f(X) \ at \ \omega r
```

```
\begin{align*}
t_1(H) &= 0 : \\
t_1(H) &= 0
\end{align*}
```

```
accept if \quad t(\omega^{k-1}) \equiv 1 \quad and \quad t(\omega r) - t(r)f(\omega r) \equiv q(r) \cdot (r^k - 1)
```

```
r ⇾ \mathbb{F}_p
learn t(\omega^{k-1}), t(r), t(\omega r), q(r), f(\omega r)
```
PLONK: a poly-IOP for a general circuit $C(x, w)$

**Step 1:** compile circuit to a sequence of ops (gate fan-in = 2)

$$\mathbf{P} = (x_1 + x_2)(x_2 + w_1)$$

**Program P**

- **0:** inp$_1$, inp$_2$ : +
- **1:** inp$_2$, inp$_3$ : +
- **2:** out$_0$, out$_1$ : $\times$

(topological sort of gates)
**Encoding the inputs to the circuit**

**Step 2:**

Let \( d = 3 |C| + |I| \) and \( H = \{ 1, \omega, \omega^2, \ldots, \omega^{d-1} \} \)

\(|C| = \text{total } \# \text{ of gates in } C\), \( |I| = |I_x| + |I_w| = \# \text{ inputs to } C\)

- encode the \( x \)-inputs to the circuit in a polynomial \( v \in \mathbb{F}_p^{(\leq |I_x|)}[X] \)
  
  for \( j = 1, \ldots, |I_x| \): \( v(\omega^{-j}) = \text{input } \#j \)

- constructing \( v(X) \) takes time proportional to the size of the input

- Let \( H_{\text{inp}} = \{ \omega^{-1}, \omega^{-2}, \ldots, \omega^{-|I_x|} \} \) (points encoding the input)
Encoding the circuit internal values

The plan: (prover uses FFT to compute coefficients of P in time \(d \log_2 d\))

Define a polynomial \(P \in \mathbb{F}_p^{(\leq d)}[X]\) such that \(\forall l = 0, \ldots, |C| - 1:\)

- \(P(\omega^{3l})\): left input to gate \#l
- \(P(\omega^{3l+1})\): right input to gate \#l
- \(P(\omega^{3l+2})\): output of gate \#l

and \(P(\omega^{-j}) = \text{input } \#j\) for \(j = 1, \ldots, |I|\) (all inputs)

**example:** \(x_1=5, x_2=6, w_1=1\)

\[\begin{align*}
\omega^{-1}, \omega^{-2}, \omega^{-3} & : 5, 6, 1 \\
0 & : \omega^0, \omega^1, \omega^2 : 5, 6, 11 \\
1 & : \omega^3, \omega^4, \omega^5 : 6, 1, 7 \\
2 & : \omega^6, \omega^7, \omega^8 : 11, 7, \text{77}
\end{align*}\]
Encoding the gates of the circuit

**Step 3:** encode gate types using a selector polynomial $S(X)$

Define $S(X) \in \mathbb{F}_p^{(\leq d)}[X]$ such that $\forall \ l = 0, \ldots, |C| - 1$:

- $S(\omega^{3l}) = 1$ if gate $\#l$ is an addition gate
- $S(\omega^{3l}) = 0$ if gate $\#l$ is a multiplication gate

Then, $\forall \ x \in H_{\text{gates}} = \{ 1, \omega^3, \omega^6, \omega^9, \ldots, \omega^{3(|C|-1)} \}$:

$$S(x) \cdot [P(x) + P(\omega x)] + (1 - S(x)) \cdot P(x) \cdot P(\omega x) = P(\omega^2 x)$$
Encoding the circuit wiring

**Step 4:** encode the wires of \( C \):

\[
\begin{align*}
P(\omega^{-2}) &= P(\omega^1) = P(\omega^3) \\
P(\omega^{-1}) &= P(\omega^0) \\
P(\omega^2) &= P(\omega^6) \\
P(\omega^{-3}) &= P(\omega^4)
\end{align*}
\]

Define a polynomial \( W : H \to H \) that implements a rotation:

\[
\begin{align*}
W(\omega^{-2}, \omega^1, \omega^3) &= (\omega^1, \omega^3, \omega^{-2}) \\
W(\omega^{-1}, \omega^0) &= (\omega^0, \omega^{-1})
\end{align*}
\]

**Lemma:** \( \forall \ x \in H: \ P(x) = P(W(x)) \ \Rightarrow \ \text{wire constraints are satisfied} \)
Encoding the circuit wiring

**Problem**: the constraint $P(x) = P(W(x))$ has degree $d^2$

$\Rightarrow$ prover would need to manipulate polynomials of degree $d^2$

$\Rightarrow$ quadratic time prover !! (goal: linear time prover)

Cute trick: use prod-check proof to reduce this to a constraint of linear degree
Reducing wiring check to a linear degree

**Lemma:**  \( P(x) = P(W(x)) \) for all \( x \in H \) if and only if  \( L(Y, Z) \equiv 1 \),

where  \( L(Y, Z) = \prod_{x \in H} \frac{P(x) + y \cdot W(x) + z}{P(x) + y \cdot x + z} \)

To prove that  \( L(Y, Z) \equiv 1 \) do:

1. verifier chooses random  \( y, z \in \mathbb{F}_p \)
2. prover builds  \( L_1(X) \) s.t.  \( L_1(x) = \frac{P(x) + y \cdot W(x) + z}{P(x) + y \cdot x + z} \) for all  \( x \in H \)
3. run prod-check to prove  \( \prod_{x \in H} L_1(x) = 1 \)
4. validate  \( L_1 \): run zero-test to prove  \( L_2(x) = 0 \) for all  \( x \in H \) where

\[
L_2(x) = (P(x) + y \cdot x + z) L_1(x) - (P(x) + y \cdot W(x) + z)
\]
The final (S, P, V) SNARK

Setup(\(C\)): \(S_v = (\text{poly commitment to } S(X) \text{ and } W(X))\)

Prover \(P(x, w)\)

- build \(P(X) \in \mathbb{F}_p^{(\leq d)}[X]\)
- Prove:
  - gates: (1) \(S(x) \cdot [P(x) + P(\omega x)] + (1 - S(x)) \cdot P(x) \cdot P(\omega x) - P(\omega^2 x) = 0\) \(\forall x \in H_{\text{gates}}\)
  - inputs: (2) \(P(x) - \nu(x) = 0\) \(\forall x \in H_{\text{inp}}\)
  - wires: (3) \(P(x) - P(W(x)) = 0\) \(\forall x \in H\)
  - output: (4) \(P(\omega^{3|C|-1}) = 0\) (output of last gate = 0)

Verifier \(V(S_v, x)\)

- build \(\nu(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]\)
Many extensions ...

- Can handle circuits with more general gates than + and \times
  - PLOOKUP: efficient SNARK for circuits with lookup tables

- The SNARK can easily be made into a zkSNARK

- Main challenge: reduce prover time
END OF LECTURE

Next lecture: recursive SNARKs