Building a SNARK

Dan Boneh
Recap: zk-SNARK applications

Private Tx on a public blockchain:
• Confidential transactions
• Tornado cash, Zcash, IronFish
• Private Dapps: Aleo

Compliance:
• Proving solvency in zero-knowledge
• Zero-knowledge taxes

Scalability: privacy in zk-SNARK Rollup (next week)
(non-interactive) Preprocessing argument systems

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

Preprocessing (setup): \( S(C) \rightarrow \) public parameters \((S_p, S_v)\)
A preprocessing argument system is a triple \( (S, P, V) \):

- \( S(C) \rightarrow \) public parameters \((S_p, S_v)\) for prover and verifier
- \( P(S_p, x, w) \rightarrow \) proof \( \pi \)
- \( V(S_v, x, \pi) \rightarrow \) accept or reject
**Requirements (informal)**

Prover $P(S_p, x, w)$  

Verifier $V(S_v, x, \pi)$  

proof $\pi$  

accept or reject

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

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

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

Optional: **Zero knowledge:** $(S_v, x, \pi)$ “reveals nothing” about $w$
**SNARK: a Succinct ARgument of Knowledge**

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

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

- short “summary” of circuit
- $\lambda = \text{security parameter} = 128$
A simple PCP-based SNARK

[Kilian’92, Micali’94]
A simple construction: PCP-based SNARK

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 $P(S_p, x, w)$**
- Long 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 $\pi$ 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 $h$
- $k$ opening and Merkle proofs

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

Problem: interactive
Making the proof non-interactive

The **Fiat-Shamir transform**:  
• public-coin interactive protocol \( \Rightarrow \) non-interactive protocol  
  public coin: all verifier randomness is public (no secrets)

Prover \( P(S_p, x, w) \)  
Verifier \( V(S_v, x) \)

- Prover sends message \( msg_1 \)
- Verifier sends random bits \( r \)
- Prover sends message \( msg_2 \)
- Verifier accepts or rejects
Making the proof non-interactive

**Fiat-Shamir transform:** $H: M \rightarrow R$ 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 $V(S_v, x)$
- $\pi = (\text{msg1}, \text{msg2})$
- $|\pi| = O(\lambda \log |C|)$
- $r \leftarrow H(x, \text{msg1})$
- accept or reject

Fiat-Shamir: certain secure interactive protocols $\Rightarrow$ non-interactive
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 transform 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: $\text{Time(Prover)} = \tilde{O}(|C|)$
Building an efficient SNARK
Many SNARKs are built in two steps:

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

(zk)SNARK for general circuits
Recall: commitments

Two algorithms:

1. $\text{commit}(m, r) \rightarrow \text{com}$ \hspace{1cm} ($r$ chose at random)

2. $\text{verify}(m, \text{com}, r) \rightarrow$ accept or reject

Properties:

1. binding: cannot produce two valid openings for $\text{com}$.
2. hiding: $\text{com}$ reveals nothing about committed data
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**\((d) \to pp\), public parameters for polynomials of degree \(\leq d\)

- **commit**\((pp, f, r) \to com_f\), 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:

\[
\text{statement } st = (pp, com_f, x, y) \text{ with witness } w = (f, r)
\]

where \(C(st, w) = 0\) iff

\[
[f(x) = y \text{ and } f \in \mathbb{F}_p^{(\leq d)}[X] \text{ and } \text{commit}(pp, f, r) = com_f]
\]
(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 $\mathcal{O}(\log d)$ group elements

• eval verify time is $\mathcal{O}(\log d)$  Prover time: $\mathcal{O}(d)$
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 $P(s_p, x, w)$

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

Verifier $V(s_v, x)$

- $r_1 \leftarrow \mathbb{F}_p$
- $r_2 \leftarrow \mathbb{F}_p$
- $r_{t-1} \leftarrow \mathbb{F}_p$

Fast verify that 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.

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

• Optional: zero knowledge
The resulting SNARK

Poly-IOP params: \( t = \# \text{polynomials}, \quad q = \# \text{eval queries in verify} \)

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 $0 \neq f \in \mathbb{F}_p^{(\leq d)} [X]$

$$\forall r \leftarrow \mathbb{F}_p : \Pr[ f(r) = 0 ] \leq d/p$$

$\Rightarrow$ 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 \( H := \{ 1, \omega, \omega^2, \ldots, \omega^{k-1} \} \subseteq \mathbb{F}_p \)

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 $H$  

$H = \{ 1, \omega, \omega^2, \ldots, \omega^{k-1} \}$

**Prover** $P(f, \perp)$

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

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

**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: $O(\log k)$ and two eval verify (but can be done in one)
Product check on $H$: $\prod_{a \in H} f(a) = 1$

Let $t \in \mathbb{F}_p^{(\leq k)} [X]$ be the degree-$d$ polynomial:

$$t(1) = f(1), \quad t(\omega^s) = \prod_{i=0}^{s} f(\omega^i) \quad \text{for } s = 1, \ldots, k - 1$$

Then

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

and

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

**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 $P((f, c), \bot)$**

construct $t(X) \in \mathbb{F}_p^{(\leq k)}$, $t_1(X) = t(\omega \cdot X) - t(X) \cdot f(\omega \cdot X)$

and $q(X) = t_1(X)/(X^k - 1) \in \mathbb{F}_p^{(\leq k)}$

$q, t \in \mathbb{F}_p^{(\leq k)}[X]$

**Verifier $V([f])$**

$r \leftarrow \mathbb{F}_p$

learn $t(\omega^{k-1}), t(r), t(\omega r), q(r), f(\omega r)$

eval $t(X)$ at $\omega^{k-1}$, $r$, $\omega r$

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

accept if $t(\omega^{k-1}) = 1$ and $t(\omega r) - t(r)f(\omega r) = q(r) \cdot (r^k - 1)$

t_1(H) = 0:
PLONK: a poly-IOP for a general circuit
PLONK: a poly-IOP for a general circuit $C(x, w)$

**Step 1:** compile circuit to a computation trace (gate fan-in = 2)

The computation trace:

- **Gate 0:** 5, 6, 11
- **Gate 1:** 6, 1, 7
- **Gate 2:** 11, 7, 77

**Example input:**

- Left inputs: 5, 6, 1
- Right inputs: 77
- Outputs: 5, 6, 11
Encoding the trace as a polynomial

\[ |C| = \text{total # of gates in } C, \quad |I| = |I_x| + |I_w| = \text{# inputs to } C \]

let \( d = 3 \, |C| + |I| \) (in example, \( d = 12 \)) and \( H = \{ 1, \omega, \omega^2, \ldots, \omega^{d-1} \} \)

The plan: prover interpolates a polynomial \( P \in \mathbb{F}_p(\leq d)[X] \) that encodes the entire trace.

Let’s see how ...

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
Encoding the trace as a polynomial

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

Prover interpolates $P \in \mathbb{F}_p^{(\leq d)} [X]$ such that

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

(2) $\forall \ l = 0, \ldots, |C| - 1$:  

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

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
Encoding the trace as a polynomial

In our example:

inputs: \( P(\omega^{-1}) = 5, \ P(\omega^{-2}) = 6, \ P(\omega^{-3}) = 1, \)

gate 0: \( P(\omega^{0}) = 5, \ P(\omega^{1}) = 6, \ P(\omega^{2}) = 11, \)

gate 1: \( P(\omega^{3}) = 6, \ P(\omega^{4}) = 1, \ P(\omega^{5}) = 7, \)

gate 2: \( P(\omega^{6}) = 11, \ P(\omega^{7}) = 7, \ P(\omega^{8}) = 77 \)

degree(P) = 11
Step 2: proving validity of P

\[ \text{Prover } P(S_p, x, w) \quad \rightarrow \quad \text{Verifier } V(S_v, x) \]

\[ \text{build } P(X) \in \mathbb{F}_p^{(\leq d)}[X] \]

Prover needs to prove that P is a correct computation trace:

1. \( P \) encodes the correct inputs,
2. every gate is evaluated correctly,
3. the wiring is implemented correctly,
4. the final output is 0

Proving (4) is easy: prove \( P(\omega^{3|C|-1}) = 0 \)

<table>
<thead>
<tr>
<th>inputs:</th>
<th>5, 6, 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0:</td>
<td>5, 6, 11</td>
</tr>
<tr>
<td>Gate 1:</td>
<td>6, 1, 7</td>
</tr>
<tr>
<td>Gate 2:</td>
<td>11, 7, 77</td>
</tr>
</tbody>
</table>
Proving (1): P encodes the correct inputs

Both prover and verifier interpolate a polynomial $v(X) \in \mathbb{F}_p^{(\leq |I_x|)}[X]$ that encodes the $x$-inputs to the circuit:

$$v(\omega^{-j}) = \text{input } \#j$$

for $j = 1, \ldots, |I_x|$. In our example: $v(\omega^{-1}) = 5$, $v(\omega^{-2}) = 6$. ($v$ is linear)

constructing $v(X)$ takes time proportional to the size of input $x$
Proving (1): P encodes the correct inputs

Both prover and verifier interpolate a polynomial \( v(X) \in \mathbb{F}_p^{(\leq |I_x|)} [X] \) that encodes the \( x \)-inputs to the circuit:

\[
\text{for } j = 1, \ldots, |I_x|: \quad v(\omega^{-j}) = \text{input } \#j
\]

Let \( H_{\text{inp}} = \{ \omega^{-1}, \omega^{-2}, \ldots, \omega^{-|I_x|} \} \subseteq H \quad \text{(points encoding the input)} \)

Prover proves (1) by using a zero-test to prove that

\[
P(y) - v(y) = 0 \quad \forall \ y \in H_{\text{inp}}
\]
Proving (2): every gate is evaluated correctly

Idea: encode gate types using a selector polynomial $S(X)$

Define $S(X) \in \mathbb{F}_p^{(\le d)}[X]$ 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

In our example $S(\omega^0) = 1$, $S(\omega^3) = 1$, $S(\omega^6) = 0$

(so that $S$ is a quadratic polynomial)
Proving (2): every gate is evaluated correctly

Idea: 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

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

$$S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) = P(\omega^2 y)$$
Proving (2): every gate is evaluated correctly

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

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

 Verify \( V(S_v, x) \)

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

Prover uses zero-test to prove \( \forall y \in H_{\text{gates}} \)

\[ S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) - P(\omega^2 y) = 0 \]
Proving (3): the wiring is correct

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 $W: H \rightarrow H$ that implements a rotation:

$$W(\omega^{-2}, \omega^1, \omega^3) = (\omega^1, \omega^3, \omega^{-2}), \quad W(\omega^{-1}, \omega^0) = (\omega^0, \omega^{-1}), \ldots$$

Example: $x_1=5, x_2=6, w_1=1$

| $\omega^{-1}, \omega^{-2}, \omega^{-3}$ | 5, 6, 1 |
| $\omega^0, \omega^1, \omega^2$ | 5, 6, 11 |
| $\omega^3, \omega^4, \omega^5$ | 6, 1, 7 |
| $\omega^6, \omega^7, \omega^8$ | 11, 7, 77 |

Lemma: $\forall y \in H: P(y) = P(W(y)) \Rightarrow$ wire constraints are satisfied
Proving (3): encoding the circuit wiring

**Problem**: the constraint $P(y) = P(W(y))$ 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:** \( P(y) = P(W(y)) \) for all \( y \in \mathbb{H} \) if and only if \( L(Y, Z) \equiv 1 \), where

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

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 \( x \in \mathbb{H} \)
3. run prod-check to prove \( \prod_{x \in \mathbb{H}} L_1(x) = 1 \)
4. validate \( L_1 \): run zero-test to prove \( L_2(x) = 0 \) for all \( x \in \mathbb{H} \) 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\( (S_p, x, w) \)

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

Prover proves:

- gates: \( (1) \quad S(y) \cdot [P(y) + P(\omega y)] + (1 - S(y)) \cdot P(y) \cdot P(\omega y) - P(\omega^2 y) = 0 \quad \forall y \in H_{\text{gates}} \)
- inputs: \( (2) \quad P(y) - v(y) = 0 \quad \forall y \in H_{\text{inp}} \)
- wires: \( (3) \quad P(y) - P(W(y)) = 0 \quad \forall y \in H \)
- output: \( (4) \quad P(\omega^{3|C|-1}) = 0 \quad \text{(output of last gate = 0)} \)

Verifier V\( (S_v, x) \)

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

- **Plonk proof**: a short proof ($O(1)$ commitments), fast verifier
- 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 zk-SNARK

Main challenge: reduce prover time
Next lecture: scaling the blockchain