Recap: zk-SNARK applications

Private Tx on a public blockchain: Zcash, IronFish

Compliance:
- Proving that a private Tx are in compliance with banking laws
- Proving solvency in zero-knowledge

Scalability: privacy in a zk-SNARK Rollup (next week)

Bridging between blockchains: zkBridge
(preprocessing) NARK: Non-interactive ARgument of Knowledge

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

Preprocessing (setup): \( S(C) \rightarrow \text{public parameters } (pp, vp) \)

\[ pp, x, w \]

\[ \text{proof } \pi \text{ that } C(x, w) = 0 \]

\[ vp, x \]

accept or reject
NARK: requirements (informal)

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

Proof $\pi$  
accept or reject

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

Adaptively knowledge sound: $V$ accepts $\Rightarrow P$ “knows” $w$ s.t. $C(x, w) = 0$
(an extractor $E$ can extract a valid $w$ from $P$)

Optional: Zero knowledge: $(C, pp, vp, x, \pi)$ “reveal nothing new” about $w$
SNARK: a **Succinct ARgument of Knowledge**

A **succinct preprocessing NARK** is a triple \((S, P, V)\):

- **S(C)** → **public parameters** 
  \((pp, vp)\) for prover and verifier

- **P(pp, x, w)** → **short** proof \(\pi\); 
  \(|\pi| = O_\lambda(\log(|C|))\)

- **V(vp, x, \pi)**  **fast to verify** ;  
  \(\text{time}(V) = O_\lambda(|x|, \log(|C|))\)

short “summary” of circuit
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 $P(pp, x, w)$
- long proof $\pi$
  
Verifier $V(vp, x)$
- 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 $poly(|C|)$. (not succinct)
Converting a PCP proof to a SNARK

Prover $P(pp, x, w)$

Verifier $V(vp, x)$

Merkle $\pi$

$h$

open $k$ positions of $\pi$ \hspace{1cm} (k = O(\lambda))

$k$ opening and Merkle proofs

Merkle root $h$

1 hash

$O(k \log |C|)$ hashes

output accept or reject

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 ⇒ non-interactive protocol
  public coin: all verifier randomness is public (no secrets)

Prover $P(pp, x, w)$

Verifier $V(vp, x)$

msg1

$r$

choose random bits $r$

msg2

accept or reject
Making the proof non-interactive

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

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

<table>
<thead>
<tr>
<th>Prover ( P(pp, x, w) )</th>
<th>Verifier ( V(vp, x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>generate msg1</td>
<td>r ← ( H(x, \text{msg1}) )</td>
</tr>
<tr>
<td>( r \leftarrow H(x, \text{msg1}) )</td>
<td>( \pi = (\text{msg1}, \text{msg2}) )</td>
</tr>
<tr>
<td>generate msg2</td>
<td>(</td>
</tr>
<tr>
<td></td>
<td>accept or reject</td>
</tr>
</tbody>
</table>

Fiat-Shamir: certain secure interactive protocols \( \Rightarrow \) non-interactive
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
General paradigm: two steps

1. A polynomial commitment scheme (cryptographic object)
2. A polynomial interactive oracle proof (PIOP) (info. theoretic object)

Let's explain each concept ...

SNARK for general circuits
Recall: commitments

Two algorithms:

• $commit(m, r) \rightarrow \text{com}$  ($r$ chose at random)

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

Properties:

• **binding**: cannot produce two valid openings for $\text{com}$

• **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

- \textit{setup}(d) \rightarrow pp, \quad \text{public parameters for polynomials of degree } \leq d

- \textit{commit}(pp, f, r) \rightarrow com_f \quad \text{commitment to } f \in \mathbb{F}_p^{(\leq d)}[X]

- \textit{eval}: \quad \text{goal: for a given } com_f \text{ and } x, y \in \mathbb{F}_p, \text{ prove that } f(x) = y.

Formally: \quad \textit{eval} = (s, P, V) \text{ is a SNARK for:}

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

- \text{where } C(st, w) = 0 \iff

\[
\left[ f(x) = y \quad \text{and} \quad f \in \mathbb{F}_p^{(\leq d)}[X] \quad \text{and} \quad \text{commit}(pp, f, r) = com_f \right]
\]
Properties:

- Binding: cannot produce two valid openings $(f_1, r_1), (f_2, r_2)$ for $\text{com}_f$.
- eval is knowledge sounds (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 most widely used in practice (called KZG):

• trusted setup: secret randomness in setup. \( |pp| = O_\lambda(d) \)
• \( \text{com}_f \): constant size (one group element)
• eval proof size: constant size (one group element)
• eval verify time: constant time. Prover time: \( O_\lambda(d) \)
General paradigm: two steps

(1) A polynomial commitment scheme

(2) A polynomial interactive oracle proof (PIOP)

SNARK for general circuits

What is a PIOP?
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 $\exists w: C(x, w) = 0$ as follows:

$$\text{Setup}(C) \rightarrow \text{public parameters } pp \text{ and } vp = (f_0, f_{-1}, \ldots, f_{-s})$$
Polynomial IOP

**Prover** $P(pp, x, w)$

- $f_1 \in \mathbb{F}_p^{(\leq d)}[X]$
- $f_2 \in \mathbb{F}_p^{(\leq d)}[X]$
- $\vdots$
- $f_t \in \mathbb{F}_p^{(\leq d)}[X]$

**Verifier** $V(vp, x)$

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

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

Verify $f_{-s}, \ldots, f_t(x, r_1, \ldots, r_{t-1})$
The Plonk poly-IOP

**Goal:** construct a poly-IOP called *Plonk* \(^{(eprint/2019/953)}\)

\[ \text{[Gabizon – Williamson – Ciobotaru]} \]

\[ \text{Plonk} + \text{PCS} \Rightarrow \text{SNARK} \]

(and also a zk-SNARK)

\[ \text{[PCS = Polynomial Commitment Scheme]} \]
First, a useful observation

A key fact: for non-zero $f \in \mathbb{F}_p^{(\leq d)} [X]$

$$\text{for } r \leftarrow \mathbb{F}_p : \quad \Pr[ f(r) = 0 ] \leq d/p$$  \hspace{1cm} (*)

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

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

\Rightarrow \quad \text{a simple zero test for a committed polynomial}$

**SZDL lemma:** (*) also holds for **multivariate** polynomials (where $d$ is total degree of $f$)
First, a useful observation

Suppose $p \approx 2^{256}$ and $d \leq 2^{40}$ so that $d/p$ is negligible

Let $f, g \in \mathbb{F}_p^{(\leq d)} [X]$.

For $r \leftarrow \mathbb{F}_p$, if $f(r) = g(r)$ then $f = g$ w.h.p

$$f(r) - g(r) = 0 \implies f - g = 0 \text{ w.h.p}$$

$\implies$ a simple equality test for two committed polynomials
Useful proof gadgets

Let $\omega \in \mathbb{F}_p$ be a primitive $k$-th root of unity ($\omega^k = 1$)
Set $\mathcal{H} := \{1, \omega, \omega^2, \ldots, \omega^{k-1}\} \subseteq \mathbb{F}_p$

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

There are efficient poly-IOPs for the following tasks:

Task 1 (zero-test): prove that $f$ is identically zero on $\mathcal{H}$

Task 2 (sum-check): prove that $\sum_{a \in \mathcal{H}} f(a) = b$ (verifier has $f$, $b$)

Task 3 (prod-check): prove that $\prod_{a \in \mathcal{H}} f(a) = c$ (verifier has $f$, $c$)
Zero-test on $H$  

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

Prover $P(f, \bot)$

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

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

Verifier $V(\mathbf{f})$

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

learn $q(r), f(r)$

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

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

Lemma: $f$ is zero on $H$ if and only if $f(X)$ is divisible by $X^k - 1$

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)
Another useful tool: permutation check

$W: H \rightarrow H$ is a permutation of $H$ if $\forall i \in [k]: W(\omega^i) = \omega^j$

ex: $W(\omega^1) = \omega^{17}$, $W(\omega^2) = \omega^5$, $W(\omega^3) = \omega^2$, ...

Let $f, g: H \rightarrow H$ be polynomials in $\mathbb{F}_p^{(\leq d)}[X]$

**Goal:** given commitments to $f, g, W$ prover want to prove that $f(y) = g(W(y))$ for all $y \in H$

$\Rightarrow$ Proves that $g(H)$ is the same as $f(H)$, just permuted by $W$
Another useful tool: permutation check

How? Use our zero-test to prove $f(y) - g(W(y)) = 0$ on H

The problem: the polynomial $f(y) - g(W(y))$ has degree $k^2$

⇒ prover would need to manipulate polynomials of degree $k^2$

⇒ quadratic time prover !! (goal: linear time prover)

Cute trick: reduce this to a prod-check

on a polynomial of degree $2k$ (not $k^2$)
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:

\[(x_1 + x_2)(x_2 + w_1)\]

Example input:

<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>

Diagram:

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

**Diagram Labels:**
- Upper node: \((x_1 + x_2)(x_2 + w_1)\)
- Nodes: \(x_1, x_2, w_1\)
- Edges: ++, --

**Example Input:** 5, 6, 1

**Diagram Arrows:**
- From inputs to gates
- From gates to output
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

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

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

1. **$T$ encodes all inputs:** $T(\omega^{-j}) = \text{input } #j$ for $j = 1, \ldots, |I|$

2. **$T$ encodes all wires:** $\forall \ l = 0, \ldots, |C| - 1$:
   - $T(\omega^{3l})$: left input to gate #$l$
   - $T(\omega^{3l+1})$: right input to gate #$l$
   - $T(\omega^{3l+2})$: 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, Prover interpolates $T(X)$ such that:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>$T(\omega^{-1}) = 5$, $T(\omega^{-2}) = 6$, $T(\omega^{-3}) = 1$,</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate 0</td>
<td>$T(\omega^0) = 5$, $T(\omega^1) = 6$, $T(\omega^2) = 11$,</td>
</tr>
<tr>
<td>Gate 1</td>
<td>$T(\omega^3) = 6$, $T(\omega^4) = 1$, $T(\omega^5) = 7$,</td>
</tr>
<tr>
<td>Gate 2</td>
<td>$T(\omega^6) = 11$, $T(\omega^7) = 7$, $T(\omega^8) = 77$</td>
</tr>
</tbody>
</table>

Degree($T$) = 11

Prover uses FFT to compute the coefficients of $T$ in time $d \log_2 d$
Step 2: proving validity of P

Prover $P(S_p, x, w)$

Verifier $V(S_v, x)$

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

T is a commitment (commitment)

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

1. $T$ encodes the correct inputs,
2. every gate is evaluated correctly,
3. the wiring is implemented correctly,
4. the output of last gate is 0

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

Inputs:
- $5, 6, 1$

Gate 0:
- $5, 6, 11$

Gate 1:
- $6, 1, 7$

Gate 2:
- $11, 7, 77$
Proving (1): T 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:

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

In our example: $v(\omega^{-1}) = 5$, $v(\omega^{-2}) = 6$, $v(\omega^{-3}) = 1$. ($v$ is quadratic)

constructing $v(X)$ takes time proportional to the size of input $x$

$\Rightarrow$ verifier has time do this
Proving (1): T 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 $H_{\text{inp}} := \{ \omega^{-1}, \omega^{-2}, \ldots, \omega^{-|I_x|} \} \subseteq H$ (points encoding the input)

Prover proves (1) by using a zero-test on $H_{\text{inp}}$ to prove that

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

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

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

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

$$S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) = T(\omega^2 y)$$

left input right input left input right input output
Proving (2): every gate is evaluated correctly

Setup($C$) $\rightarrow$ $pp := S$ and $vp := (\boxed{S})$

Prover $P(pp, x, w)$

Verifier $V(vp, x)$

Prover uses zero-test on the set $H_{gates}$ to prove that $\forall y \in H_{gates}$

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

**Step 4:** encode the wires of $C$:

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

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

\[
W(\omega^{-2}, \omega^1, \omega^3) = (\omega^1, \omega^3, \omega^{-2}) , \quad W(\omega^{-1}, \omega^0) = (\omega^0, \omega^{-1}) , \quad ...
\]

**Lemma:** $\forall y \in H : T(y) = T(W(y))$ $\Rightarrow$ wire constraints are satisfied
Proving (3): the wiring is correct

Step 4: encode the wires of $C$:

\[
\begin{align*}
T(\omega^{-2}) &= T(\omega^1) = T(\omega^3) \\
T(\omega^{-1}) &= T(\omega^2) = T(\omega^5) \\
T(\omega^2) &= T(\omega^6) = T(\omega^7) \\
T(\omega^{-3}) &= T(\omega^4) = T(\omega^5)
\end{align*}
\]

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

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

**Lemma:** $\forall y \in H: T(y) = T(W(y)) \Rightarrow$ wire constraints are satisfied

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

$\omega^{-1}, \omega^{-2}, \omega^{-3} : 5, 6, 1$

$\omega^0, \omega^1, \omega^2 : 6, 11$

$\omega^3, \omega^4, \omega^5 : 1, 7, 77$

Proved using a permutation check.
The final Plonk Poly-IOP  (and SNARK)

Setup($C$) $\rightarrow$  $pp := (S,W)$ and  $vp := (S$ and $W)$  (untrusted)

Prover $P(pp, x, w)$

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

Prover proves:

gates:  (1) $S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) - T(\omega^2 y) = 0 \quad \forall \ y \in H_{\text{gates}}$

inputs:  (2) $T(y) - v(y) = 0 \quad \forall \ y \in H_{\text{inp}}$

wires:  (3) $T(y) - T(W(y)) = 0 \quad \forall \ y \in H$

output:  (4) $T(\omega^{3|C|^{-1}}) = 0 \quad \text{(output of last gate = 0)}$

Verifier $V(vp, x)$

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

Prover $P(pp, x, w)$ builds $T(X) \in \mathbb{F}_p^{\leq d}[X]$ (commitment)

Prover proves:

gates:  (1) $S(y) \cdot [T(y) + T(\omega y)] + (1 - S(y)) \cdot T(y) \cdot T(\omega y) - T(\omega^2 y) = 0 \quad \forall \ y \in H_{\text{gates}}$

inputs:  (2) $T(y) - v(y) = 0 \quad \forall \ y \in H_{\text{inp}}$

wires:  (3) $T(y) - T(W(y)) = 0 \quad \forall \ y \in H$

output:  (4) $T(\omega^{3|C|^{-1}}) = 0 \quad \text{(output of last gate = 0)}$
The final Plonk Poly-IOP (and SNARK)

Setup($C$) $\rightarrow$  $pp := (S,W)$ and $vp := (S \text{ and } W)$ (untrusted)

Prover $P(pp, x, w)$

build $T(X) \in F_p^{(\leq d)}[X]$

Verifier $V(vp, x)$

build $v(X) \in F_p^{(\leq |I_x|)}[X]$

Thm: The Plonk Poly-IOP is complete and knowledge sound (eprint/2019/953)
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