SNARKs Lecture 5: Transparency, Recursive Proving

Instructor: Ben Fisch
Transparent SNARKs
Problems with Trusted Setup

• If subverted, prover can create fake proof
• Can be alleviated through distributed setup
  • Expensive and difficult, but done for Zcash
• Low flexibility: New functionality → New setup
  • HAWK: Every smart contract has a new setup
IOPs with Succinct Queries

• Classical IOP (point queries)

• Polynomial IOP
  - Each oracle is a degree d polynomial $f: \mathbb{F}^\mu \rightarrow \mathbb{F}$
  - Verifier queries $q \in \mathbb{F}^\mu$ and receives $f(q)$
Polynomial IOP Compilation

- Polynomial IOPs
- Interactive protocol (argument of knowledge)
- SNARK

Cryptographic compilers
- Polynomial commitment
- Fiat Shamir
IOP with Preprocessing

- A preprocessing phase establishes several oracles which are then used in future IOP instances.
- The IOP verifier is given query access not only to oracles sent by the prover, but also oracles established in the preprocessing.
- “Transparent setup”
  - Some of the IOP oracles may be derived from the R1CS program independent of the input or witness.
  - These oracles may be preprocessed so that the verifiers (or public auditors) perform a one-time expensive operation to check the correctness of their full description.
  - Preprocessed oracles may then be queried cheaply in an IOP for the program on any input/witness pair \((x, w)\).
Theorem [MBKM19]:

- There exists a **2-round** polynomial* IOP for any NP relation R (with multiplicative complexity \( n \)) that makes **1 bivariate** and **6 univariate** queries to degree \( 2n \) polynomials. Also preprocess degree \( 2n \) polynomials.

- Transforms to **5-round** polynomial IOP with **39 univariate** oracle queries overall, degree \( 2n \)
Theorem [CHMMVW19]:

- There is a 4-round polynomial IOP for any NP relation R (with R1CS complexity $n$) that makes 20 queries at 3 distinct query points to 19 univariate polynomials (max degree 6n). The preprocessing checks 8 univariate degree n polynomials.
Polynomial Commitment

\[ f(X) \in \mathbb{F}_p[X] \text{ degree at most } d \]

Small Proof that \( f(z) = y \) and 
\[ \text{deg}(f) \leq d \]
Polynomial Commitment

Input: $f(X) \in \mathbb{F}_p[X]$ degree at most $d$

- **Setup**($1^\lambda$) $\rightarrow pp$
- **Commit**($pp, f$) $\rightarrow C$
- **Open** ($pp, C, f$) $\rightarrow b \in \{0,1\}$

(Interactive protocol)

- **Eval** ($pp, C, z, y, d; f$) \Prover claim: exists $f(X)$ s.t. $f(z) = y$ AND **Commit**($pp, f$) = $C$
Efficiency: Succinctness

Input: $f(X) \in \mathbb{F}_p[X]$ degree at most $d$

- **Setup**($1^\lambda$) $\rightarrow$ $pp$
- **Commit**($pp, f$) $\rightarrow$ $C$
- **Open**($pp, C, f$) $\rightarrow$ $b \in \{0, 1\}$

(Interactive protocol)

- **Eval**($pp, C, z, y, d; f$) \ \Prover\ claim: exists $f(X)$ s.t. $f(z) = y$ AND **Commit**($pp, f$) = $C$

\[ |C| \ll f(X) \text{ ideally } O(\lambda) \]

Communication \textit{sublinear} in $d$
Security: Binding / Knowledge

Input: \( f(X) \in \mathbb{F}_p[X] \) degree at most \( d \)

- **Setup**(1^\text{\text{\text{\tau}}}) \rightarrow pp
- **Commit**(pp, f) \rightarrow C
- **Open** (pp, C, f) \rightarrow b \in \{0,1\}

(Interactive protocol)

- **Eval** (pp, C, z, y, d; f) \ \text{Prover claim: exists} f(X) \text{ s.t.} \ f(z) = y \text{ AND Commit}(pp, f) = C
Three Commitment Schemes

- **FRI (BenSasson-Bentov-Horesh-Riabzev)**
  - No trusted-setup, large proofs (>100KB), quantum secure

- **DARK (Bünz-Fisch-Szepieniec)**
  - No trusted-setup, smaller (8KB), not quantum secure

- **Bilinear group commitment (Kate-Zaverucha-Golberg)**
  - Trusted-setup, very small (32 Bytes), not quantum secure

**Size estimates for polynomial degree 1M**
Three SNARK Schemes

• FRI-STA**RK (using FRI)
  - **No trusted-setup**, large proofs (>100KB), quantum secure

• Supersonic (using DARK)
  - **No trusted-setup**, smaller (~10KB), not quantum secure

• Sonic/PLONK/Marlin (using Bilinear group)
  - **One-time** trusted-setup, small (<1KB), not quantum secure

**Size estimates for circuit size 1M gates**
Recursive SNARKs
Incremental Verifiable Computation

• Can we take a SNARK proof $\pi$ for a statement $C(x,w) = 0$ and produce an “updated” $\pi'$ that says $C(x, w) = 0 \land C(x', w') = 0$?
  • Naïve solution: redo entire SNARK for new combined statement
  • Provide both $\pi'$ and $\pi$ for new $C(x', w')$
  • Incremental SNARK: new statement that says $C(x', w')$ is true and knowledge of $\pi$ that proves $C(x,w) = 0$
“Proof of a proof”

• A proof $\pi_1$ that I know a proof $\pi_0$ that $C(x, w) = 0$

”proof of a proof of a proof…”

• A proof $\pi_2$ that I know a proof $\pi_1$ which proves knowledge of a proof $\pi_0$ that $C(x, w) = 0$. 
SNARK of SNARK

\[ S(C) \rightarrow S_P, S_V \]
\[ \pi \leftarrow P(x, w) \]

Now write a circuit \( C' \) that verifies \( \pi \):

- Input \( x' \) is \( x \)
- Witness \( w' \) is \( \pi \)
- \( C'(x', w') = 0 \) iff \( V(S_V, \pi, x) = \text{Accept} \)

Finally:

\[ S(C') \rightarrow S'_P, S'_V \]
\[ \pi' \leftarrow P(x', w') \]
SNARK of SNARK...

• Note that $C'$ depends only on $V$ and $S_V$! Can represent code for verifier $V$ as an R1CS program.

• We can also make $C'$ more complex...
  
  • Input $x'$ is $x_0, x_1$
  
  • Witness $w'$ is $\pi, w_1$
  
  • $C'(x', w') = 0$ iff $V(S_V, \pi, x_0) = \text{Accept}$ AND $C(x_1, w_1) = 0$

• What about proving verification of $\pi'$? Needs new $C''$ that has $S'_V$ hardcoded? Do we need to re-run setup for every additional level of recursion?
• Do verifiers need to re-run setup for every additional level of recursion? *Proposed solution: make* $S_V$ *part of the witness*

• Modify definition $C'_i$:
  - Input $x'$ is $x_0, x_1, \ldots, x_i$
  - Witness $w'$ is $\pi, w_i, S_V$
  - $C'(x', w') = 0$ iff $V(S_V, \pi, x_0, \ldots, x_{i-1}) = \text{Accept}$ AND $S(C'_{i-1}) \rightarrow (S_P, S_V)$ AND $C(x_i, w_i) = 0$

• Now $C'_i$ may accept as witness a proof $\pi$ that is a proof for $C'_{i-1}$ generated using parameters $(S_{P}^{i-1}, S_{V}^{i-1}) \leftarrow S(C'_{i-1})$. *Prover still needs to re-run setup as part of proving.*
Universal SNARK Verifier

- What goes wrong when the setup is trusted? Making $S_V$ a witness does not work.
  - $S_V$ “exists” even for proofs of false statements
  - Proof system only sound when prover does not know the secrets involved in the generation of $S_V$.
  - The existence of $(S_V, \pi)$ that the algorithm $V$ would accept is meaningless.

- New solution: universal SNARK verifier
Universal SNARK verifier

- UC is a circuit that takes inputs \((C, x, w)\) and outputs \(C(x, w)\)
- \(S(UC) \rightarrow (US_p, US_v)\) are parameters of SNARK system for UC
- Define \(C'_i\):
  - Input \(x'\) is \(x_0, x_1, \ldots, x_i\)
  - Witness \(w'\) is \(\pi, w_i, S_v\)
  - \(C'(x', w') = 0 \text{ iff } V(US_v, \pi, C'_{i-1}, x_0, \ldots, x_{i-1}) = \text{Accept AND AND } C(x_i, w_i) = 0\)
Application: constant size blockchain

• **Recall roll-up application**: server processes 1000 txs and sends overall state change to blockchain along with SNARK proof of knowledge of the 1000 valid txs for the transition.
  - Blockchain content still grows linearly
  - Syncing with blockchain still requires verifying many historical SNARK proofs (every 1000 txs)

• **Constant-size blockchain**: Blockchain always has just one SNARK proof. Based on recursive proofs.
  - Upon each state transition, the blockchain proof is replaced with a new SNARK that proves both correctness of transition, and existence/correctness of previous SNARK.