Composition and Iteration for SNARKs

Lecture 23
Composition

- Outsourcing the verification back to the prover
  - Proof for statement $\sigma$: Proof $\pi_1$ that there exists a proof $\pi_0$ for $\sigma$ which will be accepted by a verifier $V_0$
  - Can make the proof shorter and the verification faster
  - Prover’s work increases, but not by much if $\pi_0$ short & $V_0$ fast
  - Can make the overall proof ZK if the outer proof $\pi_1$ is ZK
Composition

- **Overheads/limiting factors**
  - Once the witness is short enough, succinctness not possible
  - No more a secure scheme in ROM even if $\pi_0, \pi_1$ were: Hash function used in $\pi_0$ is implemented as a hash function
  - Knowledge soundness degrades: For a time $t$ adversary, Ext$_1$ runs in time $\text{poly}_1(t)$ to extract $\pi_0$. Ext$_0$ extracts witness from Ext$_1$ in time $\text{poly}_0(\text{poly}_1(t))$.
  - Suppose $\sigma$ is for circuits over a field $F(0)$ and the computation of $V_0$ is over a field $F(1)$, then $\pi_1$ should support statements over $F(1)$. May limit choice of the underlying proof systems.
Composition

Proof for statement $\sigma$: Proof $\pi_1$ that there exists a proof $\pi_0$ for $\sigma$ which will be accepted by a verifier $V_0$

Suppose $\sigma$ over a field $F_{(0)}$ and verifying $\pi_0$ uses computations over a field $F_{(1)}$, then $\pi_1$ should support statements over $F_{(1)}$

Recall: when using discrete-log based polynomial commitment schemes, the field for the polynomial is $F_p$, where $p$ is the order of the cyclic group where discrete log is assumed to be hard

The most efficient candidates for such groups are elliptic curve groups. Such a group consists of a set of $p$ points of the form $(x,y) \in F^2$ for some other “base field” $F$, and the group operations use field operations over $F$. Here $F_{(0)}$ is $F_p$, $F_{(1)}$ is $F$.

Verification of $\pi_1$ will be over another field $F_{(2)}$. For deeper composition a proof system for statements in $F_{(2)}$ needed.

E.g., From 2 groups of order $p$, $q$, with base fields $F_q$, $F_p$, resp.
Many computations encountered in applications are iterative in nature.

- e.g., evaluating a Merkle-Damgard iterated hash function

**Goals:**

- To reduce prover's total effort
  - Linear in the number of steps
- To allow different provers to carry out different steps
Iterative Statements

When propagating proof

- Omit $x_i$ (except $x_1$) as verifier cannot take all $x_i$ directly as input. Can instead include a Merkle hash of all $x_i$ in $x_1$, and $w_i$ can include the opening to $x_i$.

- Omit $u_i$ by instead including a hiding commitment of $u_i$ in $y_i$, and opening it in $w_{i+1}$.

- Omit $v_i$ and include it in $y_i$ as all of public output needs to go into the next stage’s verifier.
Iterative Statements

- Naïve scheme: At each step, give a proof that the computation in the last step is consistent with the previous step’s output.
- Allows incremental proving/verification: can forget witnesses/proofs from previous steps.
- Prover’s time is linear in $n$ (if each step of computation is taken to be of constant time).
- But overall proof is long, and verification takes time proportional to $n$. 
Iterative Statements

With Composition

Idea: Move verification into the iterated computation

Statement proven by $P_{i+1}$ includes that $V_i$ accepted its proof

Note: Cannot prove soundness of the overall scheme in the Random Oracle Model, since the hash function modelled as the RO is now part of the computation

Note: Provable knowledge soundness degrades exponentially with number of steps. May be directly assumed as a heuristic.

Prover’s time and space requirements are still linear in the number of steps. But each step now has a larger computation.
Folding

- A technique for realising iteration more efficiently for the prover
- Will use R1CS representation of $F$
  $$F(y_{j-1}) = y_j \iff \exists z = (y_{j-1}, y_j, w) \text{ s.t. } Az \circ Bz = Cz$$
- Idea: Combine two R1CS instances (with same $A, B, C$) into a single R1CS instance, so that the latter can be satisfied only if the original instances can both be (with high probability)
- Uses a more general representation: “Extended” R1CS
  $$Az \circ Bz = u Cz + E,$$ where $u \in \mathbb{F}$ and $E \in \mathbb{F}^m$ is a vector
- Given instances $Az_1 \circ Bz_1 = u_1 Cz_1 + E_1$ and $Az_2 \circ Bz_2 = u_2 Cz_2 + E_2$, define instance $Az \circ Bz = u Cz + E$, where $u = u_1 + ru_2$ and $E$ is such that $z = z_1 + rz_2$ satisfies
  $$E = E_1 + r^2 E_2 + rT,$$ where $T$ has the “cross terms”:
  $$T = Az_1 \circ Bz_2 + Az_2 \circ Bz_1 - u_1 Cz_2 - u_2 Cz_1$$
- For any row, $<a_i, z_1 + rz_2> <b_i, z_1 + rz_2> = (u_1 + ru_2) <c_i, z_1 + rz_2> + E_{1i} + r^2 E_{2i} + rT_i$ holds for at most two values of $r$, unless all 3 coefficients of $r$ are 0: requiring the three conditions above to hold for that row
Folding for Iteration

Expanded R1CS representation of $F$:

$$F(y_{j-1}) = y_j \text{ iff } \exists z=(y_{j-1}, y_j, w_j) \text{ s.t. } Az \circ Bz = uCz + E, \text{ where } u=1, \ E=0$$

At each round, the prover commits (via homomorphic commitment) to $w_j$ and $y_j$ to complete the commitment of $z_j = (y_{j-1}, y_j, w_j)$. Also to $T_j$.

“Folder” picks $r_j$ and computes commitment of $z^{(j)} = z^{(j-1)} + r_j z_j$. Also of $E^{(j)} = E^{(j-1)} + r_j^2 E + r_j T_j = E^{(j-1)} + r_j T_j$ (since $E=0$). Let $u^{(j)} = u^{(j-1)} + r_j$.

Finally prover gives a SNARK for ER1CS with $E^{(n)}$, $z^{(n)}$ (committed), $u^{(n)}$, $y_{n+1}$

Adapting R1CS SNARKs that use homomorphic commitments to ER1CS
Folding for Iteration

Nova: Convert to a SNARK, by moving folders into the computation represented by the ER1CS

Only one verification at the end

Verifier may be interested in \((y_1, ..., y_{n+1})\), not just \(y_{n+1}\)

Fix: The output of the computation will be \(y'_j = (y_j, y^{(j)})\) where \(y^{(j)} = \text{Hash}(y_j, y^{(j-1)})\). In addition to the above proof, prover can send \(y_1, ..., y_{n+1}\), so that \(y^{(n+1)}\) can be checked.

Note: As before, provable knowledge soundness degrades exponentially with number of steps. May be directly assumed as a heuristic.