① IVC idea:

(1) $(t_1, s_1, s_2) + (t_2, s_2, s_3) = (t_1 + t_2, s_1, s_3)$.

(2) Efficient prover, verifier, proof size (proof merging).

(3). Completeness

(4). Soundness

Canonical language $\mathcal{L}_C = \{(M, x) : |x| = k,\ \exists\ w, |w| = k^c \text{ s.t. } M(x, w) \text{ accepts w/in } k^c\}$.

② Framework:

(1) Canonical language that we want to build an incrementally verifiable IP system for.

(2). The IP system consists of the following interactions

Statement $\mathcal{X}$ ←

Commitment →

Uniformly randomly sampled challenge ←

Opened secret as an answer to the challenge →

(3). Incrementally build S_{i+1} from S_i, as well as other circuits when needed.

(4). Prover strategy:

- Some level of uniqueness → make the hard solution by that prover's strategy the sink of rSVL.
- No uniqueness but any solution is hard to find

rSVL: (S, V, s_0, T)
ECLS

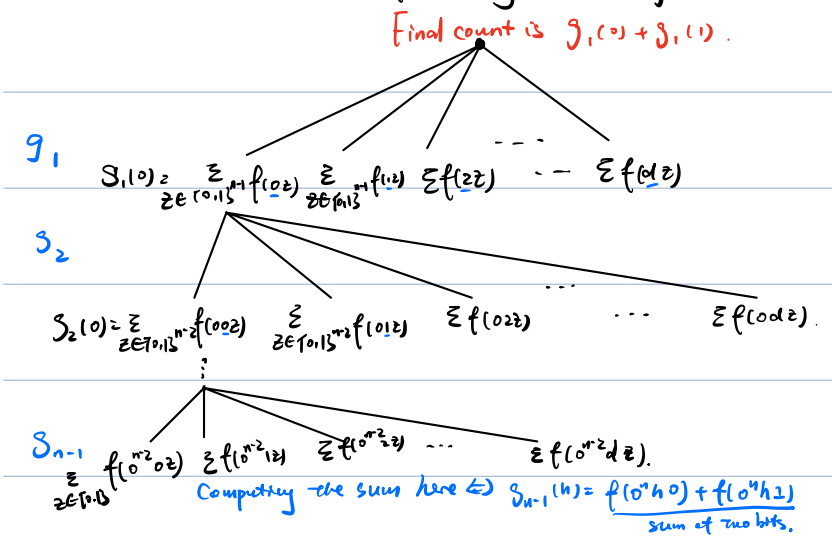
Local-Search: (S, F, v_0).
PLS-complete
Find v s.t. $F(S(v)) \le F(v)$

EOL: (S, P, v_0)
PPAD-complete
Find $v \neq v_0$ s.t. $S(P(v)) \neq v$ or $P(S(v)) \neq v$

EOUL: (C, v_0) s.t. ONLY (C, v_0).
PPA-complete
$C: V \times \{0,1\} \to V$. Given $C(v_0, 0) = v_0$ and $C(v_0, 1) \neq v_0$, find $v \neq v_0$ s.t. v also has degree of 1

Sumcheck Protocol + #SAT is hard + Fiat-Shamir $\Rightarrow$ rSUL$^{\mathcal{O}}$ HoA.

Given SAT $\to$ 3SAT-4 $\to$ low-degree polynomial of degree $\le d$.
Then, we can build the following $(d+1)$-ary recursion tree, w/ $(d+1)^{\Theta(n)}$ nodes.



At each level, the prover convinces the verifier by:

$\leftarrow$ β_{j-1} partial sum
$g_j(h)$ for $h \in H$ where $|H| = d+1$
$\leftarrow$ challenge r_j
compute $g_j(r_j)$

Observation: Why is the proof for $g_j(h)$ efficiently computable?

Recall: $g_j(h) = \sum_{z\in\{0,1\}^{n-j}} f(\beta \| h \| z)$ ($\beta \in \{0,1\}^{j-1}$) and this sum is over 2^{n-j} many z's $\Rightarrow$ the sum can be huge

* Solution: This is where the recursion comes in:

① Base level: $g_{n-2}(h) = \sum_{z\in\{0,1\}} f(\beta \| h \| z)$ ($\beta \in \{0,1\}^{n-2}$) is a sum over two bit values $\Rightarrow$ efficiently computable $\Rightarrow$ verification by direct computation.

② Inductive steps: $g_j(h) = \sum_{z\in\{0,1\}^{n-j}} f(\beta \| h \| z)$. This can be computed by fixing h and let $\beta' = \beta \| h$ and interpolate $g_{j+1}(x) = \sum_{z\in\{0,1\}^{n-j-1}} f(\beta' \| x \| z)$.

$\Rightarrow$ Once $g_{j+1}(x)$ is interpolated and computed, $g_j(h) = g_{j+1}(0) + g_{j+1}(1)$.

Remark: Locally what each subtree does is the same process as the main tree, but with a smaller size of variables.

$$\mathcal{L}_{SC} = \{(\mathbb{F}, y, f, \beta_1, \ldots, \beta_j, \tilde{\alpha}_{j+1}, \beta_{j+1}, \ldots, \alpha_{j+\bar{j}})\}$$

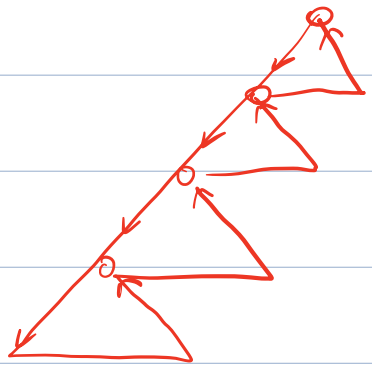
① $j = \bar{j} = 0$ $\Rightarrow$ input to sumcheck

② $j > 0$, $\bar{j} = 0$ $\Rightarrow$ partial sumcheck

③ $\bar{j} \ge 1$ $\Rightarrow$ in $\mathcal{L}_{SC}$ $\Leftrightarrow$ $\tilde{\alpha}_{j+1}$ is consistent w/ prescribed prover.

S_{n-j+1} from S_{n-j}.

Incrementally compute the proofs of each node of the recursion tree above in the DFS order



Input: $S_{n-j+1}(\beta, t, s, \text{tran})$.

① $\beta_{j-1} \in \mathbb{H}^{j-1}$ for some $g_j(x) = \sum_{x \in \{0,1\}^{n-j}} f(\beta_{j-1} \| x \| z)$.

② Index: $(t_1, \ldots, t_\ell) \in [d+1]^{\leq (n-j+1)}$

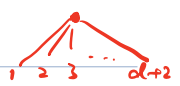
Index of a node that indicates where the computation of the current proof is at.

③ State: Sum and proofs $\{(y_0, \pi_0), (y_1, \pi_1), \ldots\}$

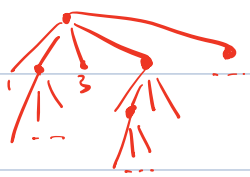
Only enough that are needed for the current computation.

④ Transcript: Statement being proved and the partial proof.

Do what?

Case A: All substates are final ⇒ aggregate to one proof 

Case B: Not enough final substates (we need $(d+2) = \begin{cases} d+1 \text{ for Schwartz Zippel} \\ +1 \text{ for challenge computation as a proof} \end{cases}$ many final states) but ongoing substates have been final ⇒ start the next substate  $i < d+2$

Case C: Substates in progress ⇒ increment one step. 

V_{n-j+1} from V_{n-j}

By nature of the recursion, can always recursively check the proofs using the S and transcript from node indexed by t for the partial / full computation of the partial sum at B_{j-1}.

rSVL

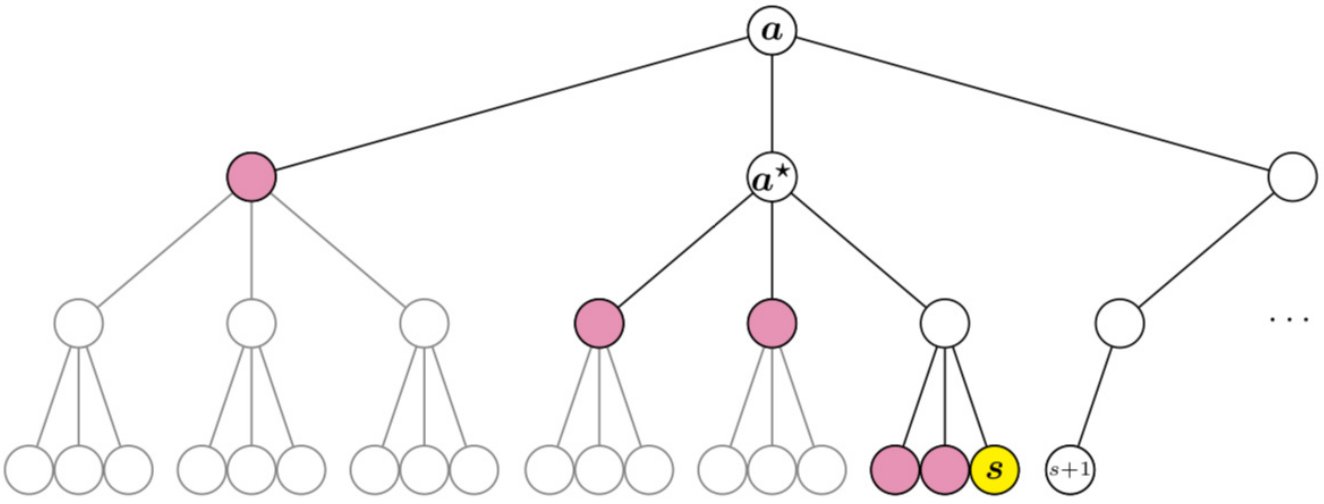
① #SAT is hard ⇒ finding sink on the main line is hard.

② The non-interactive sumcheck protocol is unambiguously sound

⇒ finding a false positive solution is hard.

Iterated squaring + RWS assumption $\Rightarrow$ rSVL$^{\mathcal{O}}$ HOA

$\Rightarrow$ Factoring is hard



① To prove $y = x^{2^t} \pmod N$, it suffices to prove

$$\left(x^{2^{t/k}}, x^{2^{2t/k}}, \cdots, x^{2^{it/k}}, \cdots, x^{2^{kt/k}} = x^{2^t} \right).$$

② Proof at each recursion level is

$$\left(\prod_{i=1}^{k} (x_{i-1})^{r_i} \right)^{2^{t/k}} = \prod_{i=1}^{k} \left(x_{i-1}^{2^{t/k}}\right)^{r_i} = \prod_{i=1}^{k} x_i^{r_i}, \quad \underline{x_0 = x}.$$

- $\{r_i\}$ are the uniformly randomly sampled challenges in ROM for Fiat-Shamir.
- Such nodes are stored in the last nodes of each sub-level, and these nodes are called the **Sketch node**.

$$\xleftarrow{\quad y = x^{2^t} \pmod N \quad}$$

$$\xrightarrow{\quad \left(x^{2^{ie/k}}\right)_{i \in [k]} \quad}$$

$$\xleftarrow{\quad \text{Challenge } \{r_i\}_{i \in [k]} \quad}$$

$$\xrightarrow{\quad \left(\prod_{i=1}^{k} (x_{i-1})^{r_i} \right)^{2^{t/k}} = \prod_{i=1}^{k} x_i^{r_i} \quad}$$

if x_i's are correct, these are efficiently computable.

$$\mathcal{I}_{N,B} = \{ (x_0, y, t) \mid \begin{array}{l} y^2 = (x_0^2)^{2^t} \pmod N \text{ if } x_0 \text{ is valid and } t \le B \\ y = \perp, \text{ o/w} \end{array} \}$$

$S(v) = cVDF.Eval(1^\lambda, pp, v)$

① $v \triangleq (y, c, F)$ where $y = x_0$

F is the frontier (left siblings are all that's needed for computing a leaf node).

S: can be computed w/

(I) Input $x =$ parent input, if leftmost child
left sibling output, if middle child
All left siblings' outputs, if rightmost child
↑ in F.

(II) Compute

$(y, \pi) \leftarrow cVDF.Eval(1^\lambda, PP_{cVDF}, (x, k^{d'}))$

similar to the traversal of the sumcheck protocol.

② Compute frontier of $S+1 \to F'$

③ Output $(y, S+1, F')$.

$V(v, i) = cVDF.Verify(1^\lambda, pp, (v_0, i), v)$

Similarly recursively utilizes the proof in sketch nodes.

$v_0 \leftarrow (y, 0^h, \emptyset)$ where $y \leftarrow \mathbb{Z}_N^*$ s.t. $\gcd(y \pm 1, N) = 1$ and $y = |S|_N$.

no frontiers.

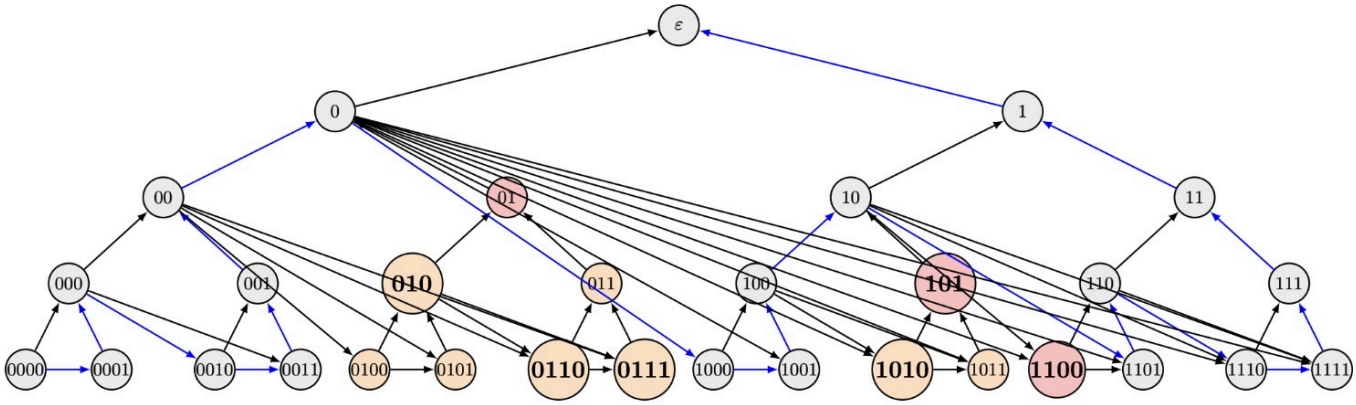
leftmost leaf node

Remark: It's said that it ① uses a weaker Fiat-Shamir assumption in terms of round soundness

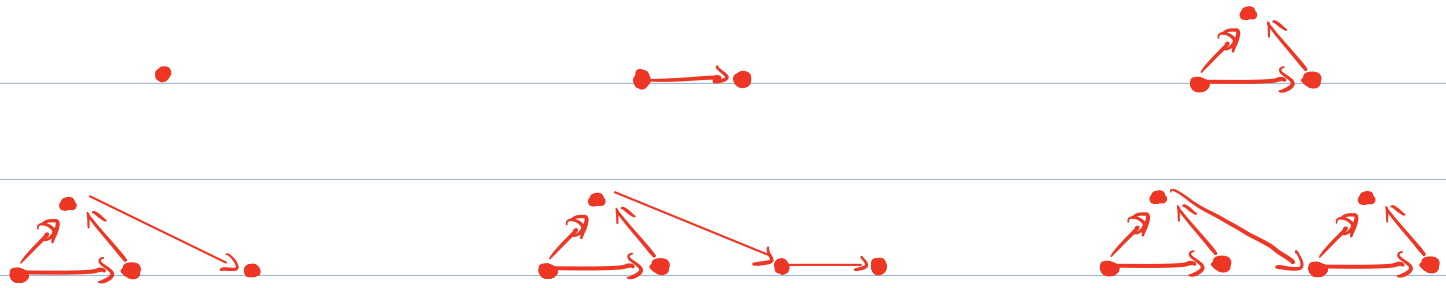
② Achieves "optimally" HOA. instances of rSUL

$\Leftarrow$ $\sqrt{T}$ steps must be taken to find any solutions $\Rightarrow$ no short-cut / false positive solutions that are trivial.

PoSW + unconditional in RoM $\Rightarrow$ $PLS^{\mathcal{O}}$ HOA.



* Unlike before, a very strong hash is used directly in the construction:



Prover

Verifier

random puzzle $\in \{0,1\}^w$

$x \xleftarrow{\$} D$

Stores ϕ_p = $\{\ell_i\}_{i \in \{0,1\}^{\leq n}}$

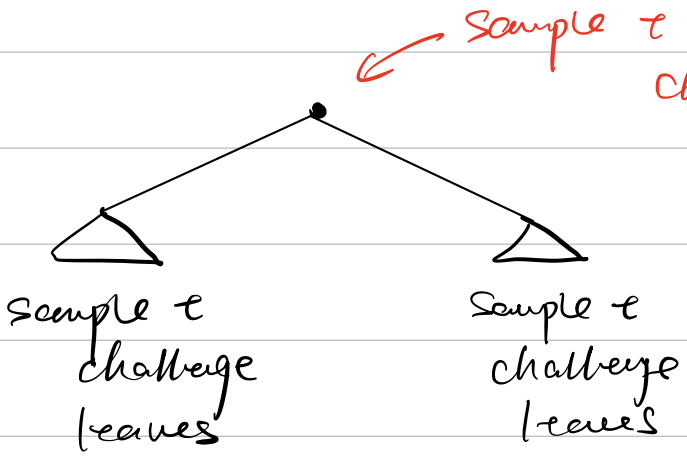
height

Compute $\mathcal{O}_n$, PoSW, ℓ_ε is the root node hash

Challenge $\gamma = (\gamma_1, \ldots, \gamma_t)$ ← these are out of the 2^n many leaf nodes, $\gamma_i \in \{0,1\}^w$ w.t

$\{(\ell_{\gamma_i}$, labels of all siblings of nodes on the $\gamma_i - \varepsilon$ path$)\}_{i \in [t]}$

How to make this incremental?



This can be thought of as random omissions of paths:

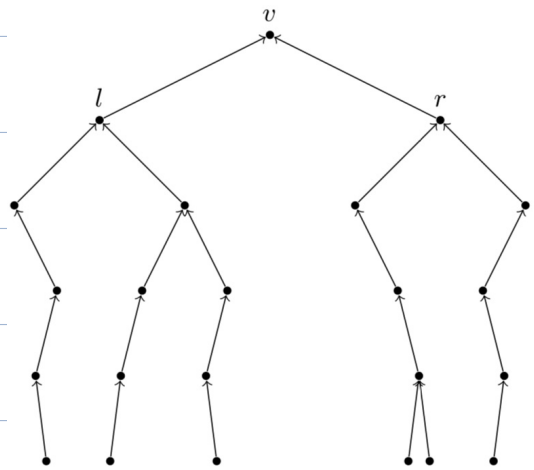


Fig. 1: Before choosing challenge subset.

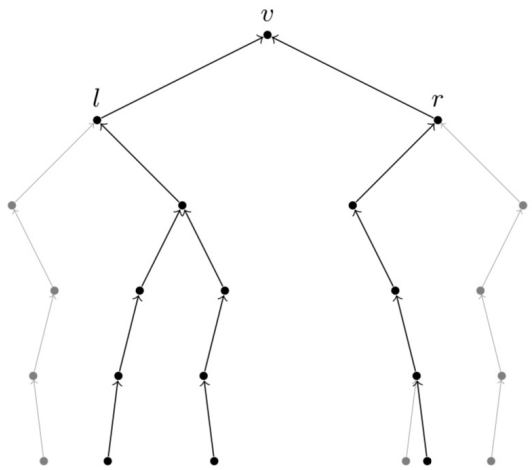


Fig. 2: After choosing challenge subset.

$\chi \xleftarrow{\$} D$

$\ell \varepsilon \longrightarrow$

$\mathcal{P}_P$

Thin down to only relevant authentication path set

$\xleftarrow{\text{Challenge Set}}$

$\xrightarrow{\text{Same proof}}$

$R_d^\theta = \{(x, \pi) \mid \text{DLM. } V^\theta(x, T_d, \pi) = \text{ACCEP}\}$

$x \in R_d^\theta$

① S^θ : $(t+1, \pi_i')$ by generate π_i' from (t, π_i) by joining two subtrees and selecting new
- challenge set
- authentication path

② V^θ : Value of the state is t

$x \notin R_d^\theta$

① S^θ : $(1, \pi_1)$

② V^θ : -1