LatticeFold & its Applications

Binyi Chen, Dan Boneh
Stanford University

Succinct Non-Interactive Argument of Knowledge

(zk)SNARK ≈ Proof of correct computation

Given circuit $\boldsymbol{C}$, instance x, I know witness w s.t. $\boldsymbol{C}(x, w) = 0$

E.g. knowledge of secret key/hash preimage

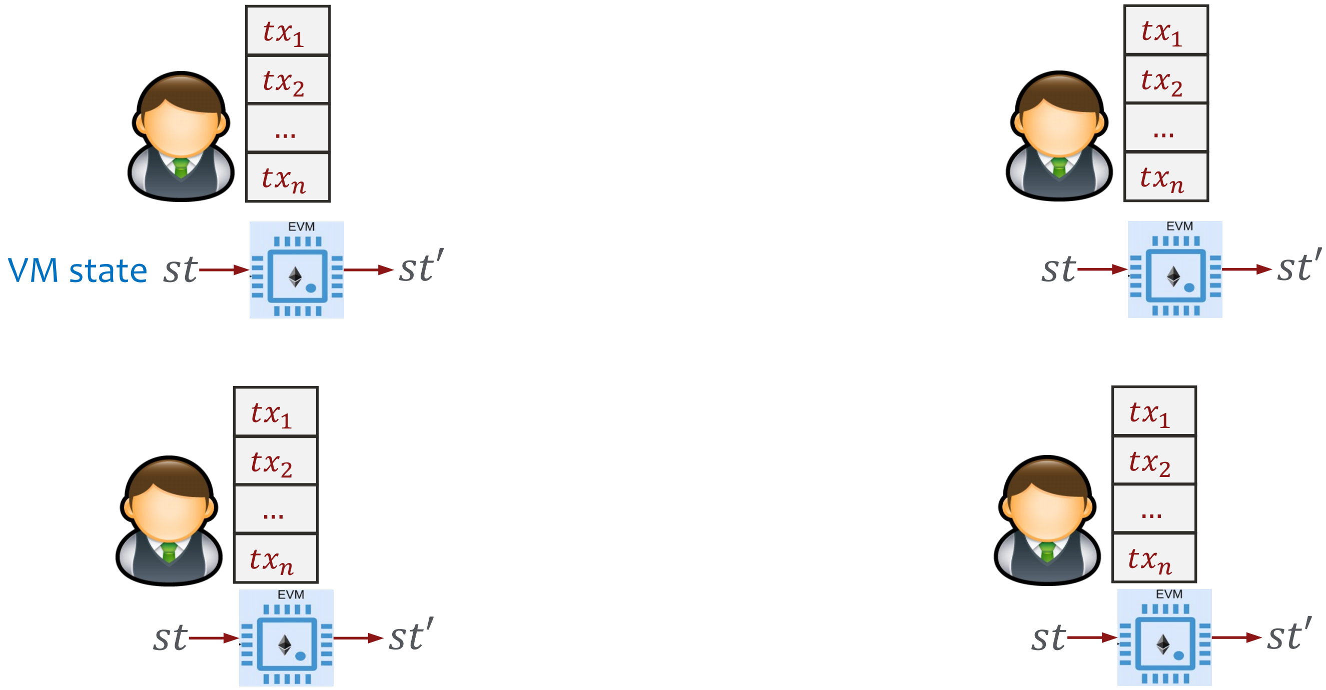
$$(pk_C, vk_C) \leftarrow \text{Setup}(\boldsymbol{C})$$

$$\text{Prove}(pk_C, x, w) \rightarrow \pi \qquad \text{Verify}(vk_C, x, \pi) \rightarrow 0/1$$

<u>Succinctness:</u> π is **small** and **cheap** to verify

Scaling Blockchains

Smart-contract Blockchain: (oversimplified)

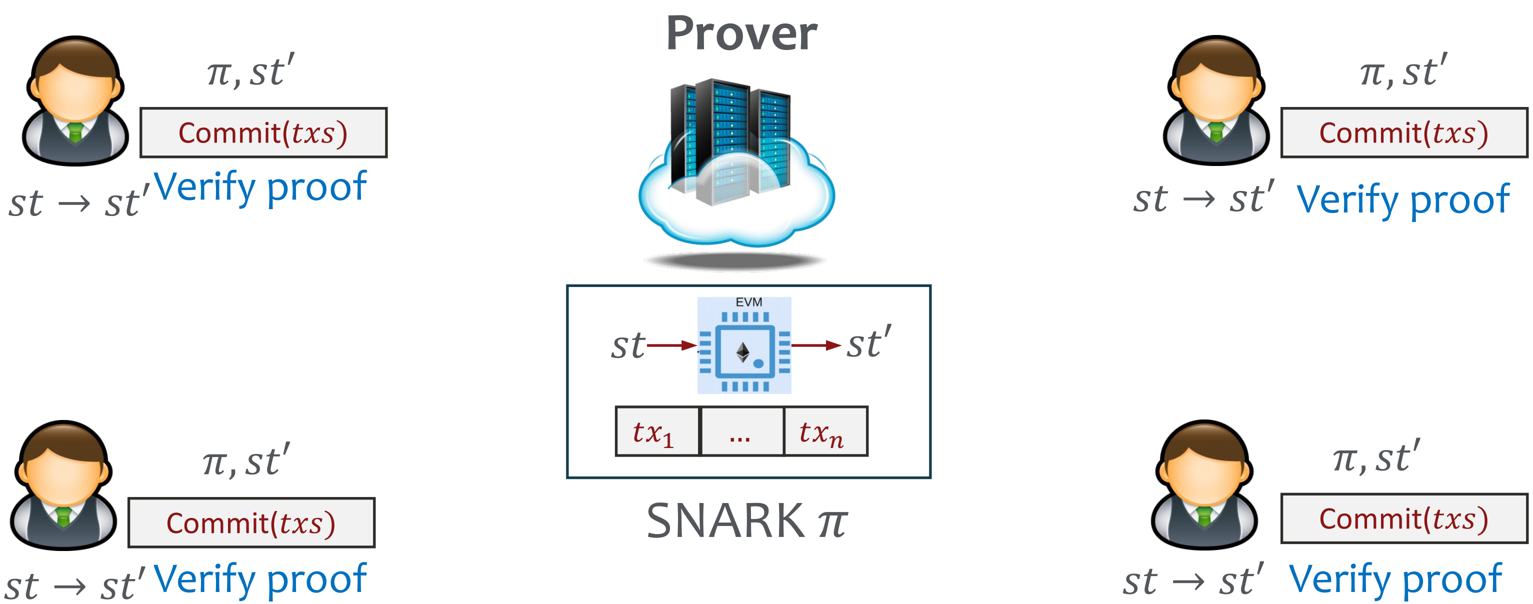


Redundant execution ⇒ poor throughput/latency

Scaling Blockchains

Based Rollup: (oversimplified)

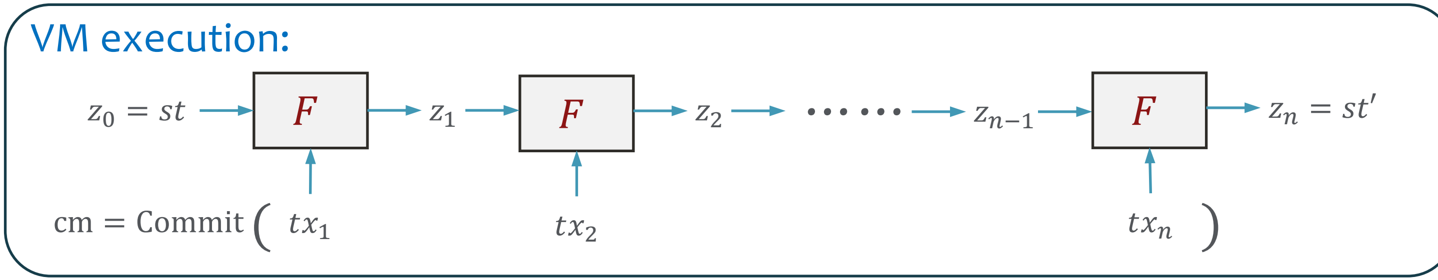
How to compute π efficiently?



Much cheaper!

Monolithic SNARKs

Huge circuit



Can't support dynamic n

Fix n

transform to a circuit

Large, can't start proving without it

$$C(x = [z_0, z_n, \mathrm{cm}], w \leftarrow f(\mathrm{exec\_trace})) = 0$$

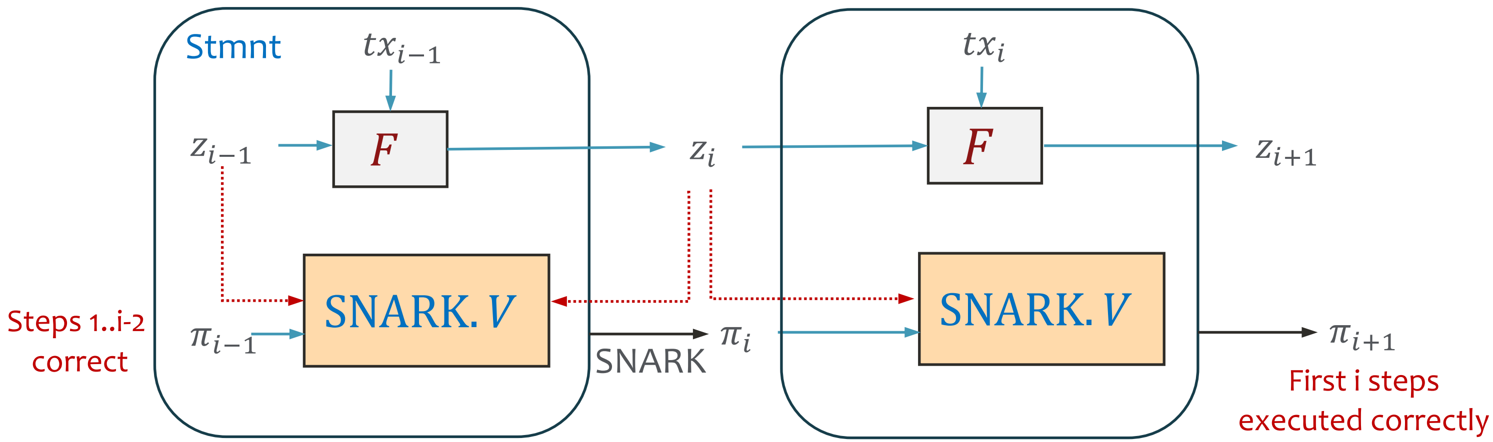
Memory/computation intensive

E.g., FFTs, MSMs

Run a SNARK (e.g., Plonk/STARK)

Proof π

Piecemeal SNARKs (IVC/PCD) [Valiant08, BCCT12]



Pros:
- Pipeline proving/witness-gen
- Small memory overhead
- Parallelizable using PCD

E.g., Mangrove [NDCTB24]

Cons:
- Expensive SNARK.V circuit
- SNARK proving still not *that* cheap

Any better way to construct IVC?

IVC/PCD from Folding [BCLMS20,KST21]

Homomorphic commitment:

Commit: long vector w $\longrightarrow$ short c_w

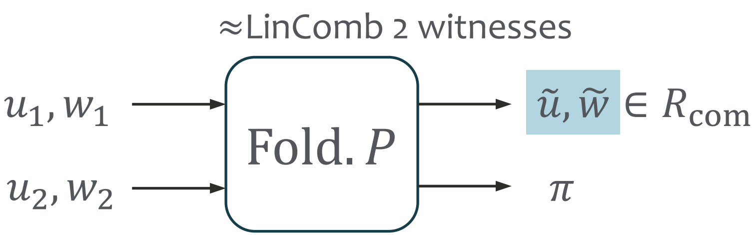
Homomorphism: $w_1 + w_2$ $\longrightarrow$ $c_{w_1+w_2} = c_{w_1} + c_{w_2}$

Why useful? Expensive chk $f(w_1, w_2) =_? 0$ $\longrightarrow$ Easy chk $f(c_{w_1}, c_{w_2}) =_? 0$

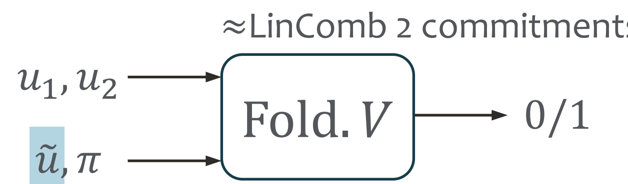
IVC/PCD from Folding [BCLMS20,KST21]

Folding scheme: ≈ Compress multiple NP statements into one

$$R_{\text{com}} \coloneqq \{(u = (x, c_w), w) : (x, w) \in R_{NP} \wedge c_w = \text{Comm}(w)\}$$



Faster than SNARK.P!

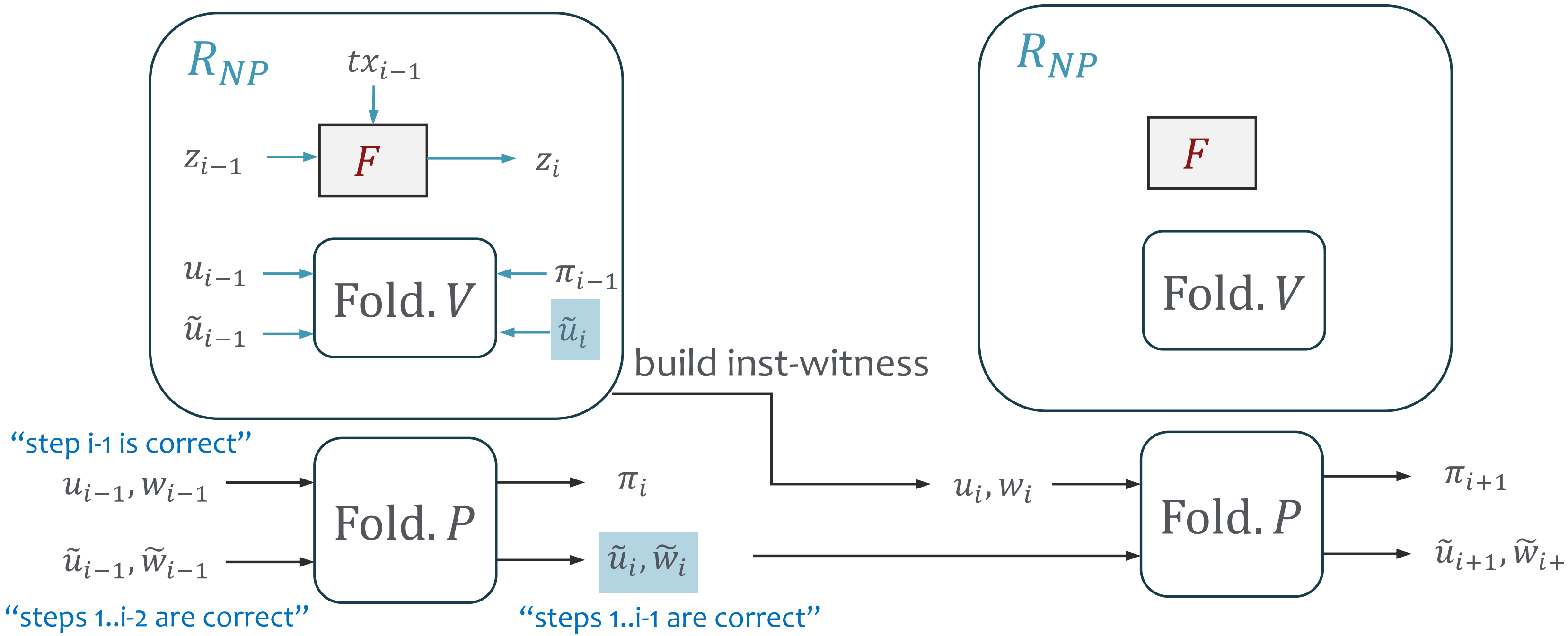


Cheaper than SNARK.V!

Completeness + Knowledge soundness

[BCLMS20,KST21]: We can construct IVC/PCD from folding schemes!

IVC/PCD from Folding [BCLMS20,KST21]



IVC/PCD from Folding [BCLMS20,KST21]

IVC from folding vs IVC from SNARK:

Proving algorithm:

SNARK.P

$\Downarrow$

$\boxed{\text{Fold}.\,P}$ Much faster

Extra embedded circuit:

$\boxed{\text{SNARK}.\,V}$

$\Downarrow$

$\boxed{\text{Fold}.\,V}$ Much smaller

Which homomorphic commitment to use?

Homomorphic Commitment

Option 1: Pedersen p, q: ≈256-bit primes

$$w := (w_1, w_2 \ldots, w_n) \in \mathbb{F}_p^n \longrightarrow c_w := g_1^{w_1} g_2^{w_2} \cdots g_n^{w_n} \in \mathbb{G} \approx \mathbb{F}_q \times \mathbb{F}_q$$

Cons:

- Expensive group exponentiations over **large** fields $\mathbb{F}_p, \mathbb{F}_q$ (256-bit)
- Fold.V ≈ 1 $\mathbb{G}$-exp + hash/field ops over $\mathbb{F}_p$
    - need to support both $\mathbb{F}_p, \mathbb{F}_q$ ⇒ field emulation (e.g. $\mathbb{F}_p$-ops over $\mathbb{F}_q$)
- Vulnerable to quantum attacks

LatticeFold: Contributions

The first folding scheme from lattice-based commitments

- *Fast & small* fields arithmetics (e.g., 64-bit or 32-bit prime fields)
- Eliminate *non-native* field emulation in Fold.V
  - Messages and commitments live in the same space
- Quantum attacks resistant (based on Lattice assumptions)
- Support *high-degree* constraint systems (e.g., CCS [STW23])

Ajtai Binding Commitments [Ajtai96]

E.g., $q \approx$ 64-bit prime, $\beta = 2^{16}, n \gg \lambda$

long vector $w \in [-\beta, \beta]^n \xrightarrow{\boxed{A \leftarrow \mathbb{Z}_q^{\lambda \times n}}}$ short $c_w = Aw \bmod q \in \mathbb{Z}_q^{\lambda}$

Essential for binding

<u>Homomorphic Property:</u>

Assumption: $w_1 + w_2 \in [-\beta, \beta]^n$

$$c_{w_1} + c_{w_2} = (Aw_1 + Aw_2) \bmod q = A(w_1 + w_2) \bmod q = c_{w_1+w_2}$$

Cons: committing complexity = $O(\lambda n)$ $\mathbb{F}$-ops

Ring/Module-based Ajtai [LM07,PR07]

E.g., $R_q = \mathbb{Z}_q[X]/(X^d + 1)$ (Polynomials with deg $< d$ and $\mathbb{Z}_q$-coefficients)

long vector $w \in \{-\beta, \ldots, \beta\}^n \xrightarrow{A \leftarrow \mathbb{Z}_q^{\lambda \times n}}$ short $c_w = Aw \bmod q \in \mathbb{Z}_q^\lambda$

$\Big\Downarrow\ \mathbb{Z}_q^d \rightarrow R_q$

$\widetilde{w} \in R_q^{n/d} \xrightarrow[\tilde{A} \leftarrow R_q^{\lambda/d \times n/d}]{} \tilde{c}_w = \tilde{A}\widetilde{w} \in R_q^{\lambda/d}$

Coefficients in $\{-\beta, \ldots, \beta\}$

Pros:

- E.g., $\lambda = d$, committing complexity: $O(n/d)$ R_q-ops $\approx O(n\log\lambda)$ $\mathbb{F}_q$-ops
- Many *hardware optimizations* in the FHE/Lattice-signature literature

Challenges of Folding with Ajtai

Naïve folding:

$$c_{w_1}, w_1 \qquad \xrightarrow{\text{random } \gamma} \qquad c_{w_1} + \gamma c_{w_2}, \quad \boxed{w_1 + \gamma w_2} \notin [-\beta, \beta]^n \text{ anymore}$$
$$c_{w_2}, w_2$$

Challenge: Keep folded witness stay in the ***bounded*** msg space

Essential for binding/soundness

Re-represent witnesses w/ lower norms

Extend to the case where $a \in R_q$

Decomposition: $a \in (-\beta, \beta) \xrightarrow[(\beta = b^k)]{\text{split algorithm}} a_1, \ldots, a_k \in (-b, b)$

$a = a_1 + b \cdot a_2 + \cdots + b^{k-1} \cdot a_k$

Folding: (e.g., $k = 2$ and $\beta = b^2$)

$c_{w_1}, w_1 \xrightarrow{\text{split}} c_{w_1}^1, w_1^1 \in (-\boldsymbol{b}, \boldsymbol{b})^n$; $c_{w_1}^2, w_1^2$

$c_{w_2}, w_2 \xrightarrow{\text{split}} c_{w_2}^1, w_2^1$; $c_{w_2}^2, w_2^2$

random $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in R_q$

with small coefficients!

$c^* = \text{combine}([\gamma_i], [c_{w_1}^1 \ldots c_{w_2}^2])$

$\boxed{w^* = \text{combine}([\gamma_i], [w_1^1 \ldots w_2^2])}$

$\in [-\beta, \beta]^n$

Complication: Fold.P must prove that witnesses are low-norm (i.e. in $(-b, b)^n$)

Novel range-proofs from Sumchecks

Performance

$n \approx$ # of constraints

	LatticeFold	Pedersen Folding [KST21, BC23, KS23]	Hash-based Folding [BMNW24]
Prover time	$O(n\log\lambda)$ $\mathbb{Z}_q$-mul w/ **small** q 🙂	$O(n)$-sized MSM over **large** field	$O(n)$ hash 🙂
Verifier circuit	$\approx O(b\log n)$ hash 🙂	$O(1)$ $\mathbb{G}$-exps + **non-native** $\mathbb{F}$-ops	$O(\lambda\log n) \gg O(b\log n)$ hash ☹️
"Unbounded" folding steps	✅	✅	❌
Efficient commit for sparse vector	✅	✅	❌

Summary & Future Work

- LatticeFold: the **first** lattice-based folding scheme
  - Fast & small field; efficient verifier circuit; quantum attacks resistant
  - Hardware optimization-friendly + Support high-deg constraint systems

- Updated version
  - Optimized folding for high-degree constraint systems (CCS)
    - 2 sequential Sumchecks previously, now only 1!

- Future work
  - Integrate with Lasso to support table lookups
  - Remove the need for witness decomposition/range-check

Thank You

https://eprint.iacr.org/2024/257.pdf