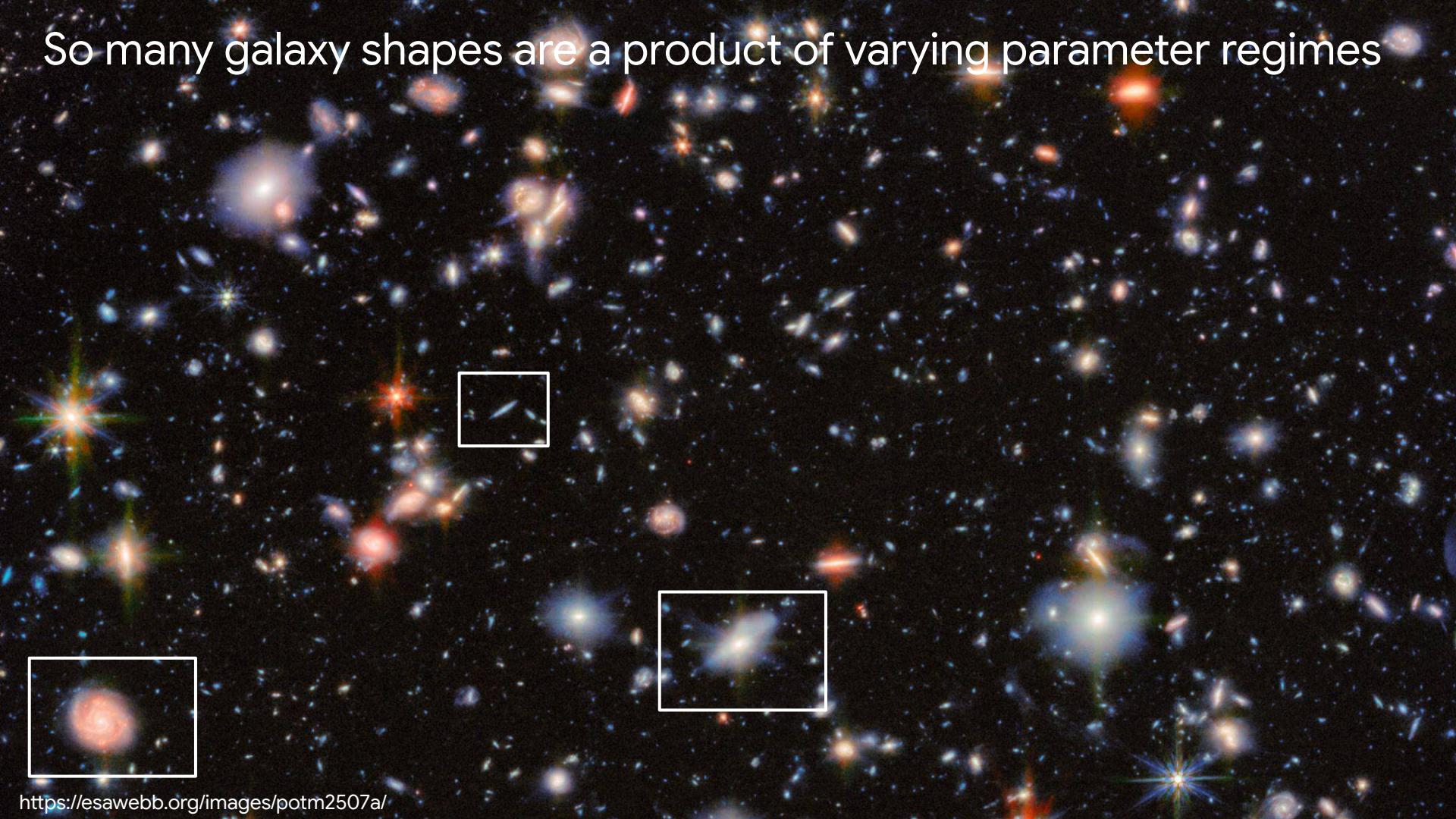


ZK for legacy schemes

Presented on January 29 at MPTS 2026
NIST Workshop on Multi-Party Threshold Schemes

Matteo Frigo, abhi shelat

So many galaxy shapes are a product of varying parameter regimes



Zero-knowledge proofs are natural computational artifacts.

Different parameter regimes result in different shapes.

Lesson today: adding sumcheck to any scheme seems to be a good idea.

Zero-Knowledge (ZK) recipes

Approaches:

Requires Pairing-Friendly curves

BBS+

Elegant proof system for anonymous credentials, but requires changes to secure element.

Groth16

Small proofs, Fastest V, slowest P, and requires a large trusted parameter setup (GBs)

STARK

Too slow, due to requirement of Reed-Solomon encoding **every wire** in the circuit.

Requires only a hash function

Ligero

Hyrax

(sumcheck + com)

Our approach, substantially improved.

{MPC,VOLE}-ith

GF v F_p performance?
Fast VOLE needs LPW?
Sumcheck better if NTT used

2nd oldest recipe for ZK [BGGHKMR88]

Run an **IP** \rightarrow Transcript.

Commit to Transcript \rightarrow Com.

ZK for “Com contains T and $\text{IP-verifier}(T)=1$ ”

It just remains to pick, IP, Commit, ZK.

Our Choices

IP: Layered Sumcheck [GKR] (Prover can run in $O(C)$ time!)

Commit/ZK: Ligerio [AHIV] (Only uses SHA256. Simple, fast.)

P convinces V that “there exists w such that $f(\vec{x}, \vec{w}) = 0$ ”

Prover (f, \vec{x}, \vec{w})

Verifier (f, \vec{x})

Pick a random pad.

Ligero-Commit(w, pad)

com



Run Sumcheck-P(C, x, w)



Run Sumcheck-V(C, x)

When Sumcheck sends message
 m_i , send $m_i + \text{pad}_i$.
Record transcript T .

...

Send random challenges.
Record transcript T

Run Ligero-Prover($\text{com}, \text{Enc-Sumcheck-Verify}(T, x, \text{pad})$)



Ligero-Verify($\text{com},$
 $\text{Enc-Sumcheck-Verify}, T, x, \text{proof}$)



The scheme is inspired by Hyrax [WTSTW17].

Our Choices

IP: Layered Sumcheck [GKR] (Prover can run in $O(C)$ time!)

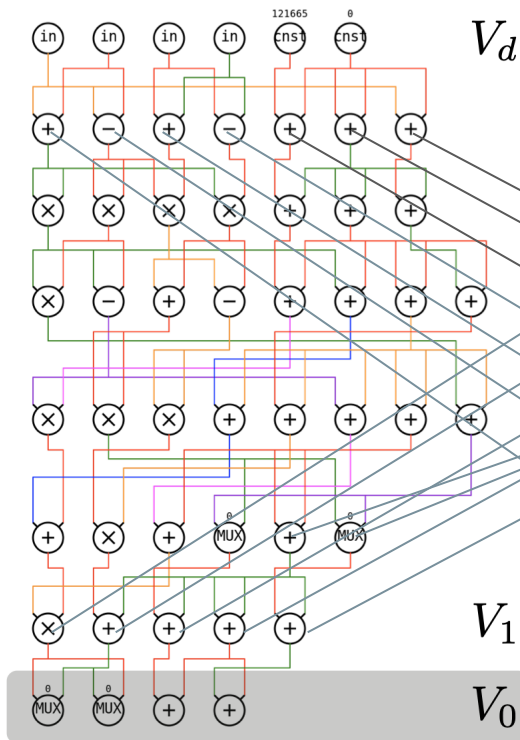
Commit/ZK: Ligerio [AHIV] (Only uses SHA256. Simple, fast.)

**Why is this
combination
useful?**

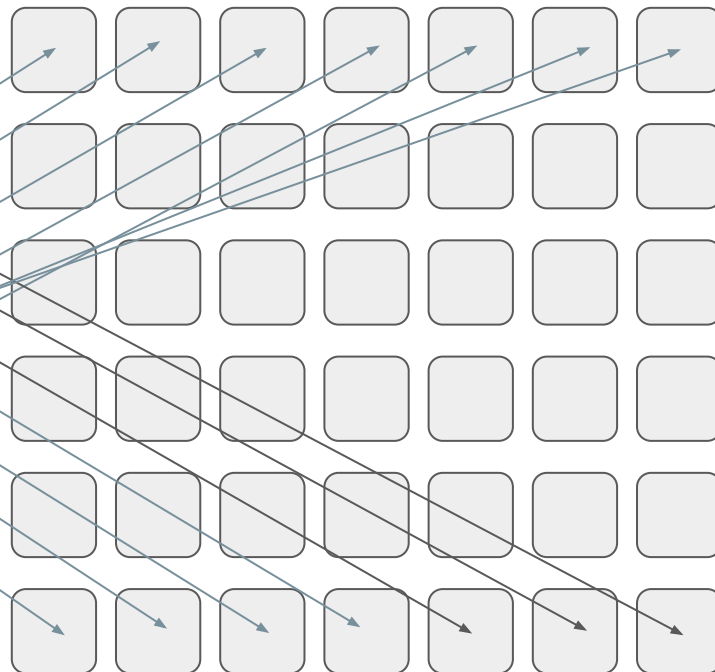
Most TARKs commit to every intermediate wire

E.g., Ligerio, Starkware, SmallWood, v-ith...

Circuit(x,w)

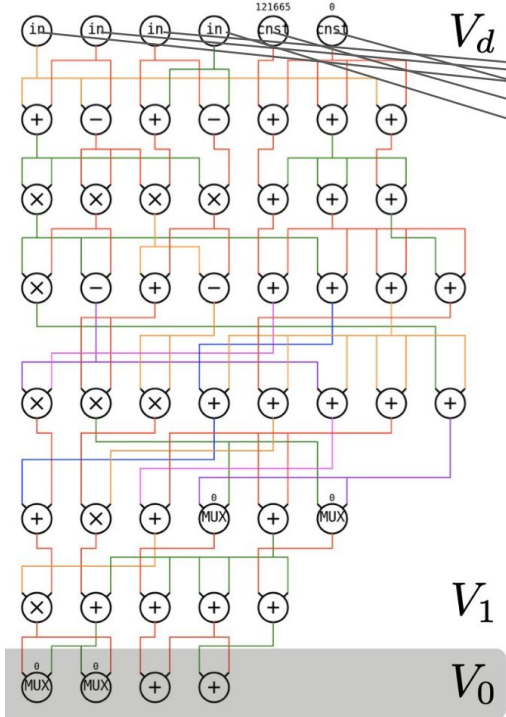


Commitment

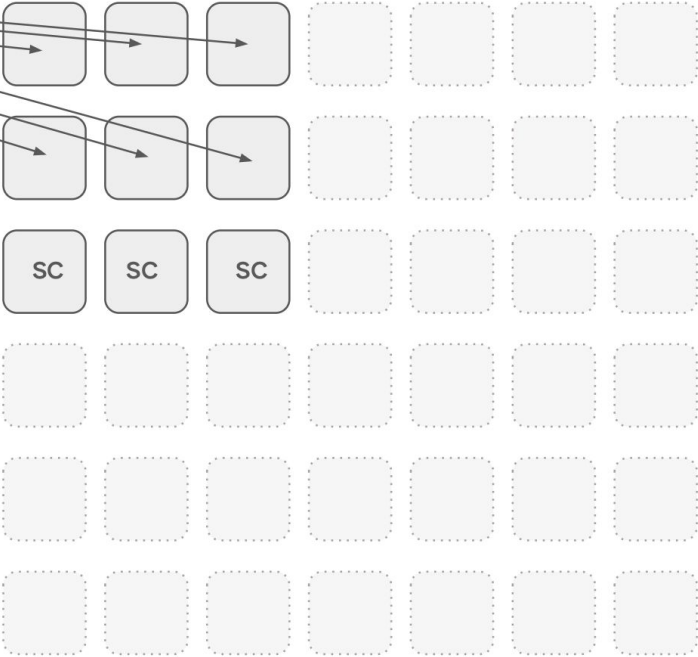


Ligero(Layered-sumcheck) commits to input+sc proof

and only verifies the sumcheck proof



Ligero(Layered-SC) Commitment



Our Choices

IP: Layered Sumcheck [GKR] (Prover can run in $O(C)$ time!)

Commit/ZK: Ligerio [AHIV] (Only uses SHA256. Simple, fast.)

Reduction in Commitment size leads
to 20-50x improvements;
Fast enough for human applications.

Example: Verifying an ECDSA signature

Algorithm 2 Verification Circuit $V(Q, H(m), r, s, r_y)$

- 1: Derive integer e from $H(m)$.
 - 2: Verify that $r, s \in [1, q - 1]$
 - 3: Verify that $Q, R = (r, r_y) \in E$ and $R \neq id$.
 - 4: Verify $id = G \cdot e + Q \cdot r - R \cdot s$
-

Prior work (eg circom-ECDSA, Stark, Plonk, ...) needs to emulate the field math of F_p, F_q .

ECDSA verification is a small circuit in F_{P256}

Table 1: Circuit size and depth for ECDSA verification.

	DEPTH	QUADS	TERMS	INPUTS
Multi-exponentiation	7	19,534	37,550	1,038
Range check + rest	12	5,475	10,569	1,038
Total	12	23,453	47,598	1,038

Finite field used by our circuit is over the base field of the P256 curve. Prior work was expensive b/c it simulated math in F_p in an NTT-friendly F_q .

Input + SC is roughly ~50x fewer wires to commit to.

Possible because we can perform efficient NTTs for F_{p256} .

ECDSA: Mac M4 performance

BM_ECDSAZKProver/1	16.7 ms	42
BM_ECDSAZKProver/2	26.5 ms	27
BM_ECDSAZKProver/3	38.3 ms	18
BM_ECDSAZKVerifier/1	10.3 ms	67
BM_ECDSAZKVerifier/2	16.0 ms	44
BM_ECDSAZKVerifier/3	23.4 ms	31

Example: ZK for SHA-256 pre-image (n block message)

BM_ShaZK_fp2_128/1	5300363	ns	5300367	ns	109
BM_ShaZK_fp2_128/2	9602156	ns	9600622	ns	74
BM_ShaZK_fp2_128/4	18730299	ns	18730225	ns	40
BM_ShaZK_fp2_128/8	35389356	ns	35289150	ns	20
BM_ShaZK_fp2_128/16	65615658	ns	65553800	ns	10
BM_ShaZK_fp2_128/32	125226342	ns	125226400	ns	5
BM_ShaZK_fp2_128/33	132710525	ns	132710400	ns	5

Proof size ~200kb.

Using field F_2^{128}

Single thread

Cf, Vole-in-head: 17.7ms

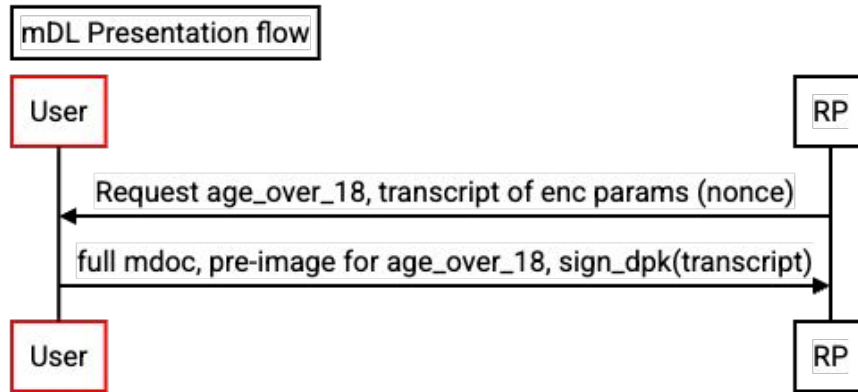
Ligero ZK SHA-256 preimage systems

		Time (ms)		
		$n = 1$	2	4
x64	Commit	222	384	736
	Total ZK Prover	273	493	939
	Overhead wrt this work	26x	25x	27x
	[WHV24] <small>Smaller field, multi-threaded</small>	250	450	750
Pixel	Commit	294	536	1022
	Total ZK Prover	380	717	1370
	Overhead wrt this work	20x	19x	20x

This slide shows the performance of our implementation of the same circuit using only Ligero.

Example: Legacy ISO MDOC identity protocol

“The state of California has produced a signature on an mDL document stored on my cell phone that includes the attribute ‘age_over_18 = true’.”



1. Verify Signature of mdoc by Massachusetts.
2. Parse mdoc to find DPK.
3. Verify Signature of transcript under DPK.
4. Verify pre-image of the "age_over_18" attribute, verify it is set to True.
5. Verify credential expiry condition.

ZK statement to prove

Given the public values (PK_II, “age_over_18”, transcript, time_now), there exists

a 2231 byte string MDL, a hash e_1, a hash h_2, an index X, a signature sig_1, ..., a 32-b nonce, a pk DPK, a pair of strings time_start, time_end, and a signature sig_2

such that:

```
e_1 = SHA-256(MDL) and  
p256.verify( sig_1, e_1, PK_II ) = true and
```

← Costly part

```
h_2 = MDL[valueDigests][org.iso.18013.5.1][‘age_over_18’] and  
h_2 = SHA-256(nonce, ‘age_over_18’, ‘True’) and
```

```
DPK = MDL[deviceKeyInfo][deviceKey][-2, -3] and  
p256.verify( sig_2, transcript, DPK ) = true and
```

```
time_start = MDL[validityInfo][validFrom] and  
time_end = MDL[validityInfo][validUntil] and  
time_start < time_now < time_end and
```

~ 7.2m “wires”,
3m “quads”,
600k in Fp,
6.7m in 2¹²⁸

Device
binding

Proof: ~350kb

Status

2 separate open-source implementations
Google, ISRG

Deployed in a few products

4 independent security reviews

Alternative ZK systems

Hypothesis

For problems in the 1000 – 20m witness regime,

With no setup parameters,

With security that only relies on HASH,

{Sumcheck + Ligerio} is a hard to beat.

Alternative #1: Proofs use Setup strings

Many options, small proofs.

Large common setup parameters make these systems impractical to deploy.

Proofs that use bilinear pairings:

Similar problems with deployment.

WHIR vs Ligerio ?

- Proofs are ~ same size for these parameters
- Prover takes longer

Prover time: 163.2ms
Proof size: 138.0 KiB

Prover time: 45.9ms
Proof size: 158.6 KiB

“All the complexity for such little gain”

When does it overtake ? after 2^{26} witness size?

Other MPC-in-the-head

- “With preprocessing”
 - The “repetition code” that is implicit in these constructions make them unsuitable for all but very small problems.

M (# of pre-processed “triples”) *
 n (# of participants)
easily in the 1000s

	$\rho = 128$						$\rho = 256$					
n	4	8	16	32	64	128	4	8	16	32	64	128
M	218	252	352	462	631	916	456	533	781	1024	1662	2540
τ	65	44	33	27	23	20	129	87	65	53	44	38

Table 1: Sample values of M , n , and τ to achieve statistical security $\rho \in \{128, 256\}$. M is the number of executions simulated by the prover; n is number of parties in the MPC protocol; τ is the number of executions of the online phase of the MPC protocol.

Lookup arguments? [ArunSettyThaler24]

No clear win for lookups in SHA, ECDSA, parsing, etc.

“All the complexity for such little gain”

Performing NTT on P256

An n-th root of unity is a solution to the equation $x^n = 1 \pmod p$. For FFTs, one requires $n=2^k$ roots of unity for $k \in [10,30]$.

The P256 finite field doesn't have enough roots of unity for an NTT.

$$\underline{P256 - 1} = 2 * 3 * 5^2 * 17 * 257 * 641 * 1531 * 65537 * 490463 * 6700417 * 835945042244614951780389953367877943453916927241$$

Its quadratic extension does. So we can lift the NTT to F_{P256^2} .

$$\underline{P256^2 - 1} = 2^{97} * 3 * 7 * 5^2 * 17 * 257 * 641 * 1531 * 65537 * 274177 * 490463 * 6700417 * 67280421310721 * 11318308927973941931404914103 * 835945042244614951780389953367877943453916927241$$

zk-ECDSA PoK(r,s) for (e,pk)		Time (ms)		
		$n = 1$	2	3
x64 single thread	Ligero com	38.7	51.0	60.8
	Verify transcript	13.5	26.5	38.6
	Total ZK	58.8	87.0	110
	Verifier	6.09	11.0	14.5
Pixel Single thread	Ligero com	51.0	67.2	80.0
	Verify transcript	20.3	40.1	58.4
	Total ZK	80.5	120.0	152.0
	Verifier	8.50	16.2	21.2

circom-ECDSA:
140000ms (single core)
 973MB trusted parameter

Woo et al (IEEE S&P'25):
900ms (uses Spartan zk
 system + sidecar sigma
 prot) [Last talk this session]

Update: 300ms