Single-Server Private Outsourcing of zk-SNARKs (IEEE S&P 2026)

Kasra Abbaszadeh: UMD
Hossein Hafezi: NYU / University of Cambridge
Jonathan Katz: Google
Sarah Meiklejohn: UCL

dd May 2026

ZKPROOF
Paving the path to adoption

8th Workshop

Rome, Italy
9-10 May 2026

Zero-Knowledge Succinct ARGuments of Knowledge

R public relation

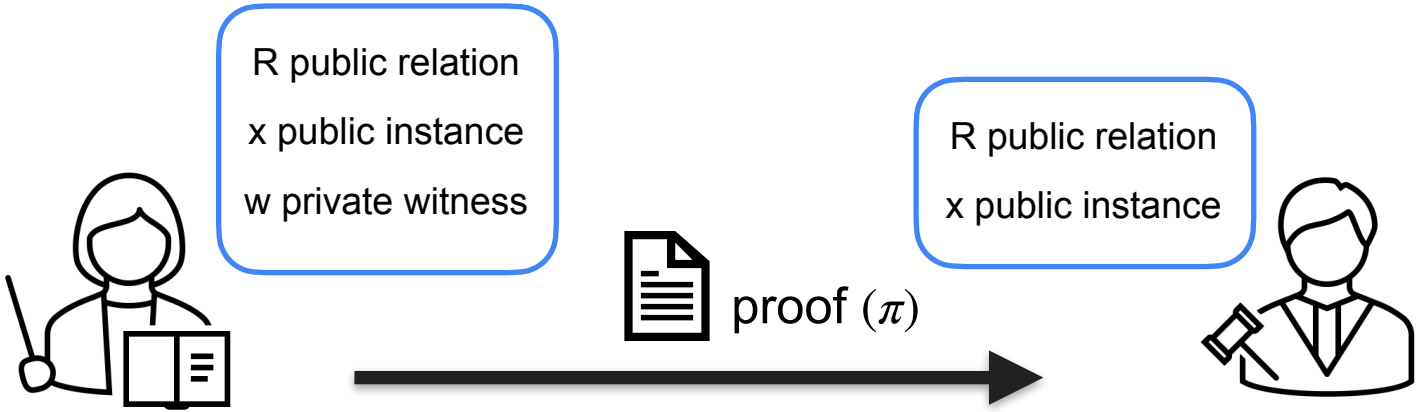
x public instance

w private witness

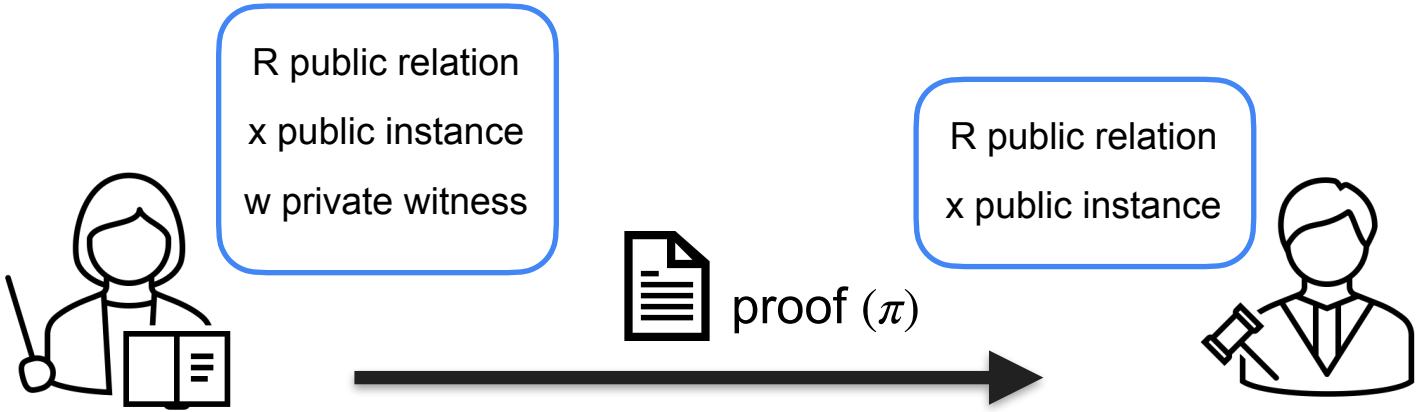
R public relation

x public instance

Zero-Knowledge Succinct ARGuments of Knowledge



Zero-Knowledge Succinct ARGuments of Knowledge



Completeness. For every $(x, w) \in R$, it holds that Pr[Verify(x, π)] $= 1$

Knowledge Soundness.If Verify$(x, \pi) = 1$, then the prover must know w.

Zero-Knowledge. π reveals nothing about the underlying witness w.

Sunccinctness. size of π is sublinear in w.

Many Real-World Applications

Espresso

Aleo

STARKWARE

Aztec

zk-creds: Flexible Anonymous Credentials from zkSNARKs and Existing Identity Infrastructure

Michael Rosenberg*[1], Jacob White†[2], Christina Garman[2], and Ian Miers[1]

ZEXE: Enabling Decentralized Private Computation

Sean Bowe
sean@z.cash
Zcash

Alessandro Chiesa
alexch@berkeley.edu
UC Berkeley

Matthew Green
mgreen@cs.jhu.edu
Johns Hopkins University

Ian Miers
imiers@cs.cornell.edu
Cornell Tech

Pratyush Mishra
pratyush@berkeley.edu
UC Berkeley

Howard Wu
howardwu@berkeley.edu
UC Berkeley

Efficient Proofs of Software Exploitability for Real-world Processors

Matthew Green
mgreen@cs.jhu.edu
Johns Hopkins University
USA

Mathias Hall-Andersen
mathias@hall-andersen.dk
Aarhus University
Denmark

Eric Hennenfent
eric.hennenfent@trailofbits.com
Trail of Bits
USA

Gabriel Kaptchuk
kaptchuk@bu.edu
Boston University
USA

Benjamin Perez
benperez1227@gmail.com
Trail of Bits
USA

Gijs Van Laer
gijs.vanlaer@jhu.edu
Johns Hopkins University
USA

What's Not There Yet?

High prover cost

Prover is 1000 × slower than native computation.

computing SHA256 hash of 100KB takes 1ms; however, proving this takes up to 20 min

Groth16, on a laptop

What's Not There Yet?

High prover cost

Prover is 1000 × slower than native computation.

computing SHA256 hash of 100KB takes 1ms; however, proving this takes up to 20 min

Many applications require generating proof for millions of arithmetic constraints.

zk-login, proof of training, auditable key transparency, etc.

What's Not There Yet?

High prover cost

Prover is 1000 × slower than native computation.

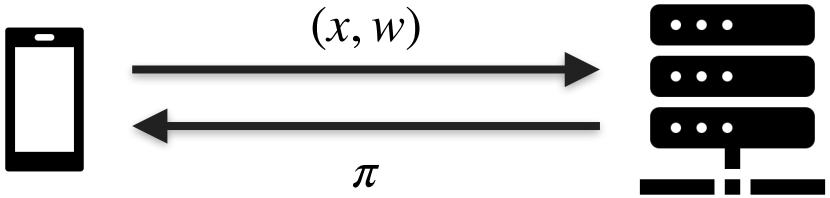
computing SHA256 hash of 100KB takes 1ms; however, proving this takes up to 20 min

Many applications require generating proof for millions of arithmetic constraints.

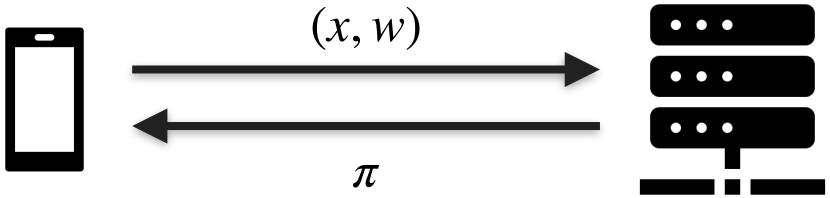
zk-login, proof of training, auditable key transparency, etc.

Can we outsource proof generation to more powerful servers to lower the prover's workload and latency?

Outsourcing zk-SNARK prover

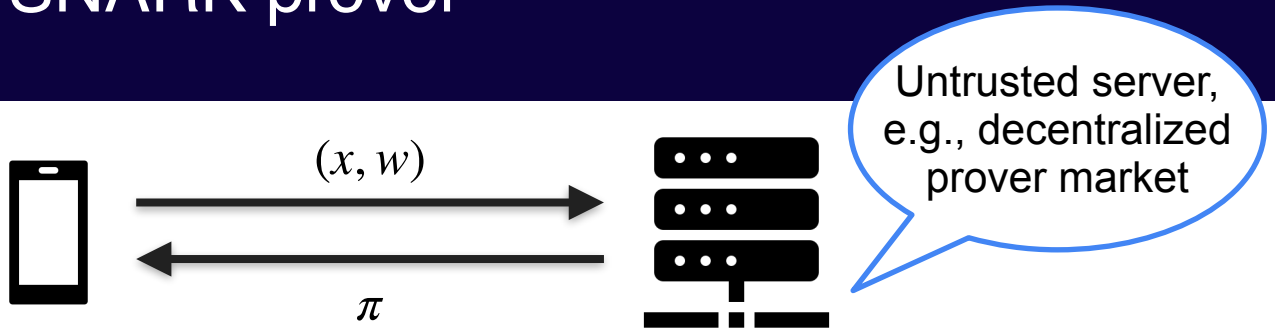


Outsourcing zk-SNARK prover



A cluster of servers can generate the proof 100x faster [Wu+18].

Outsourcing zk-SNARK prover

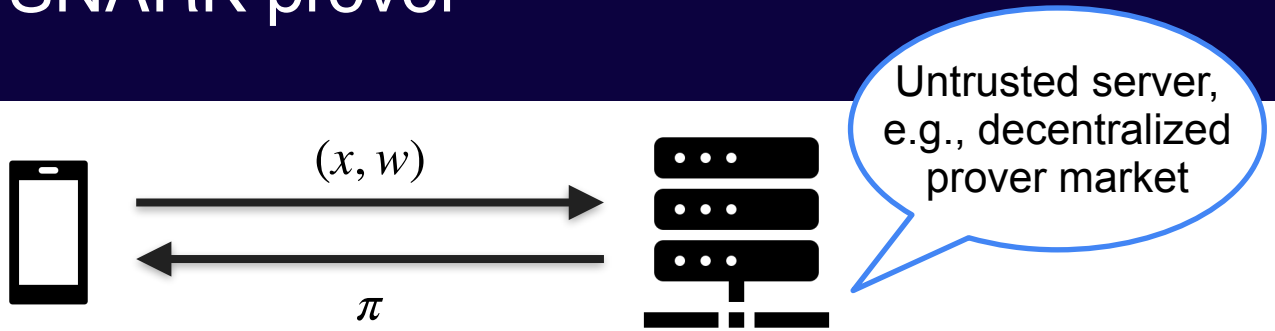


A cluster of servers can generate the proof 100x faster [Wu+18].

The problem: Exposing secrets to the servers; not ideal for privacy-preserving applications

e.g., private transactions, anynonymous credentials, etc.

Outsourcing zk-SNARK prover



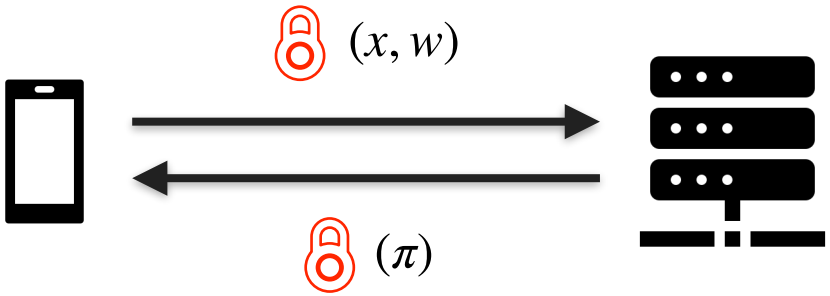
A cluster of servers can generate the proof 100x faster [Wu+18].

The problem: Exposing secrets to the servers; not ideal for privacy-preserving applications

e.g., private transactions, anynonymous credentials, etc.

Ideally: (1) do not reveal the witness (2) unlinkablity

Private Outsourcing of zk-SNARKs Prover



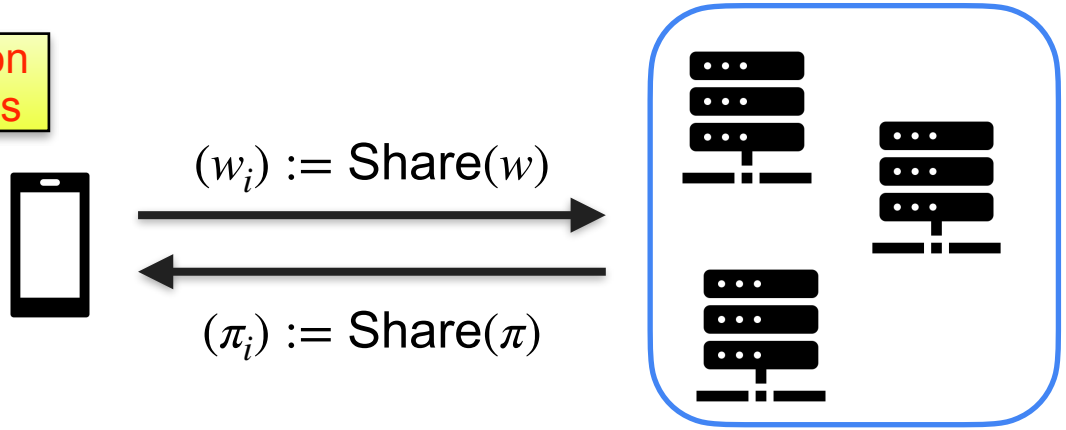
Generic approaches to computation over private data:

i) Fully Homomorphic Encryption (FHE)

ii) Secure Multi-Party Computation via secret sharing (MPC)

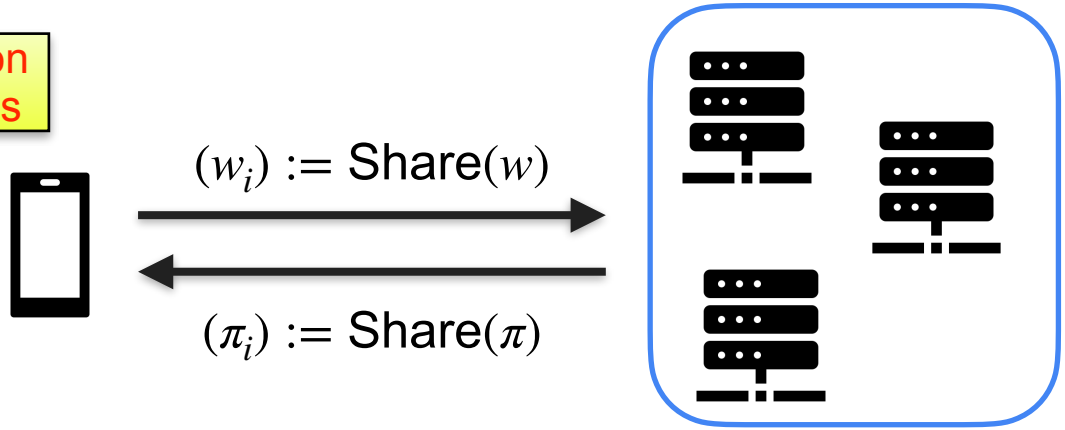
Private Outsourcing of zk-SNARKs via MPC

Client only does construction and reconstruction of shares



Private Outsourcing of zk-SNARKs via MPC

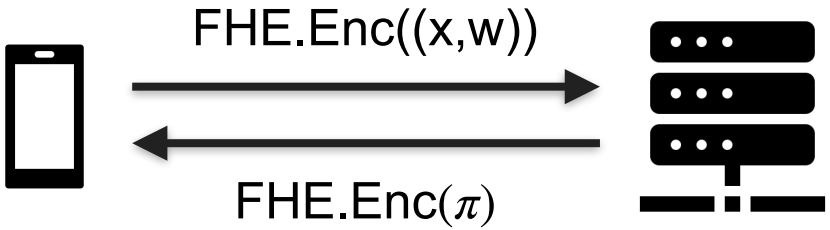
Client only does construction and reconstruction of shares



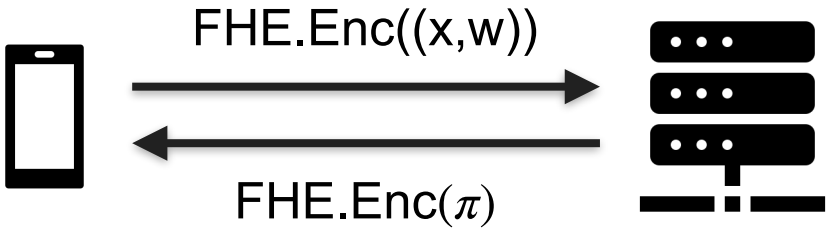
[GGJ+23, CLMZ24]: Similar to non-private schemes, low latency and efficient client.

The key limitation: non-collusion assumption

Private Outsourcing of zk-SNARKs via FHE



Private Outsourcing of zk-SNARKs via FHE

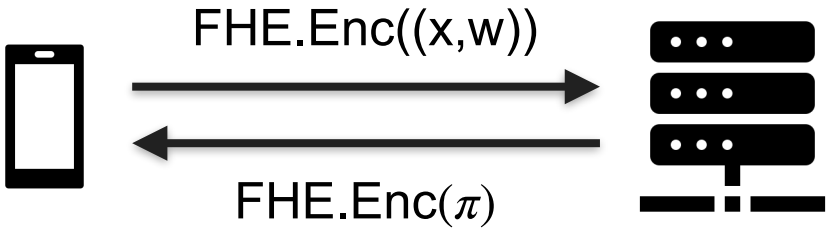


Generic FHE incurs 10,000x slowdown

[GGW24]: Single-server, minimal interctive and efficient client

Main Barrier: Latency overhead, e.g., 10x in [GGW24] for (adapted) FRI-based zk-SNARKs

Private Outsourcing of zk-SNARKs via FHE



[GGW24]: Single-server, minimal interctive and efficient client

Main Barrier: Latency overhead, e.g., 10x in [GGW24] for (adapted) FRI-based zk-SNARKs

No malicious security and unlinkablity

Comparison with Prior Work

Scheme	# of servers	Improved latency	Unlinkability	Malicious security	Instantiations
co-SNARKs [52]	≥ 2	✗	✗	✓	Groth16; Marlin; Plonk; Fractal
EOS [21]	≥ 2	✓	✗	✗	Marlin
ZKSaaS [33]	≥ 8	✓	✗	✗	Groth16; Plonk; Marlin
DFS [38]	≥ 2	✓	✗	✓	Marlin*
Siniel [64]	≥ 2	✓	✗	✓	Marlin
Liu et al. [46]	≥ 8	✓	✗	✓	HyperPlonk
Fang et al. [28]	≥ 2	✓	✗	✓	Plonk*
Blind FRI [35, 32]	1	✗	✗	✗	Fractal*
This paper	1	✓	✓	✓	Nova; Groth16; Plonk

* Delegation-specific adaptations.

Table 1: Private delegation schemes. Number of servers assumes the scheme tolerates at least one corrupted server. Although EOS claims malicious security, an attack is known [38].

Our Contribution

- Server-aided zk-SNARKs: single-server framework for private outsourcing of zk-SNARKs, which supports latency reduction, malicious security, unlinkability and group-based zk-SNARKs.
- EMSM (encrypted multi-scalar multiplication): we construction EMSM from LPN assumption with lightweight cryptographic overhead.

What is Multi-Scalar Multiplication (MSM)?

$$\text{MSM}(\vec{g}, \vec{z}) = \prod_{i \in [n]} g_i^{z_i} \text{ such that } g_i \in \mathbb{G}, z_i \in \mathbb{F}$$

What is Multi-Scalar Multiplication (MSM)?

$$\text{MSM}(\vec{g}, \vec{z}) = \prod_{i \in [n]} g_i^{z_i} \text{ such that } g_i \in \mathbb{G}, z_i \in \mathbb{F}$$

The computation bottleneck for group-based zk-SNARKs.For a circuit with 1 million constraints: MSM is **97%** percent of computation with Nova and **91%** with groth16

What is Multi-Scalar Multiplication (MSM)?

$$\text{MSM}(\vec{g}, \vec{z}) = \prod_{i \in [n]} g_i^{z_i} \text{ such that } g_i \in \mathbb{G}, z_i \in \mathbb{F}$$

The computation bottleneck for group-based zk-SNARKs.For a circuit with 1 million constraints: MSM is **97%** percent of computation with Nova and **91%** with groth16

Each group exponentiation costs ≈ 3000 arithmetic gates

Outsourcing with FHE is impractical

What is Multi-Scalar Multiplication (MSM)?

$$\text{MSM}(\vec{g}, \vec{z}) = \prod_{i \in [n]} g_i^{z_i} \text{ such that } g_i \in \mathbb{G}, z_i \in \mathbb{F}$$

The computation bottleneck for group-based zk-SNARKs.For a circuit with 1 million constraints: MSM is **97%** percent of computation with Nova and **91%** with groth16

Each group exponentiation costs ≈ 3000 arithmetic gates

Outsourcing with FHE is impractical

Use linear structure of MSM to fully outsource it to the server, and instead, the client does field operations

Dual Learning Parity with Noice (LPN)

Dual LPN

$(\mathbf{H}, \mathbf{H}\vec{e}) \approx_c (\mathbf{H}, \vec{b})$ such that $\vec{b} \leftarrow \mathbb{F}^n$

the noise is sparse, $\vec{e} \leftarrow \text{Bern}_p$

Dual Learning Parity with Noice (LPN)

Dual LPN

$(\mathbf{H}, \mathbf{H}\vec{e}) \approx_c (\mathbf{H}, \vec{b})$ such that $\vec{b} \leftarrow \mathbb{F}^n$

the noise is sparse, $\vec{e} \leftarrow \text{Bern}_p$

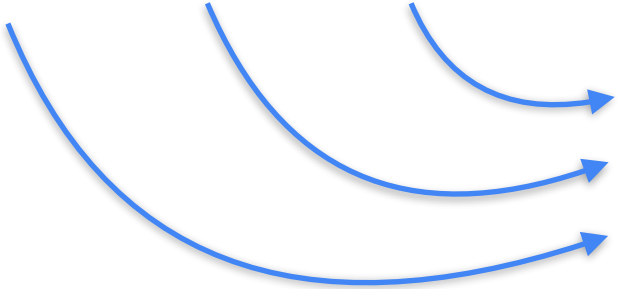
The noise vector has only a constant number of non-zeros.

Building EMSM from Dual-LPN

Fix the bases $\vec{g}$ then $\text{MSM}(\vec{g}, \vec{z} + \vec{r}) = \text{MSM}(\vec{g}, \vec{z}) + \text{MSM}(\vec{g}, \vec{r})$

Building EMSM from Dual-LPN

Fix the bases $\vec{g}$ then $\text{MSM}(\vec{g}, \vec{z} + \vec{r}) = \text{MSM}(\vec{g}, \vec{z}) + \text{MSM}(\vec{g}, \vec{r})$



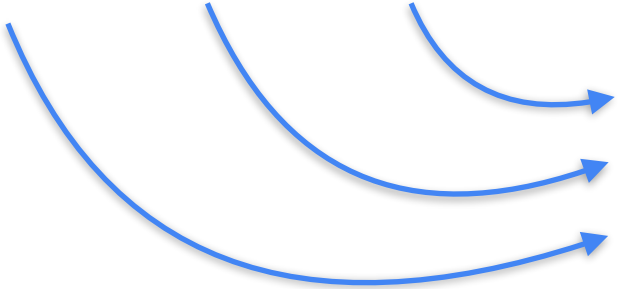
Client computes this on their own

Client want to outsource this to the server

The server computes this for the client

Building EMSM from Dual-LPN

Fix the bases $\vec{g}$ then $\text{MSM}(\vec{g}, \vec{z} + \vec{r}) = \text{MSM}(\vec{g}, \vec{z}) + \text{MSM}(\vec{g}, \vec{r})$



Client computes this on their own

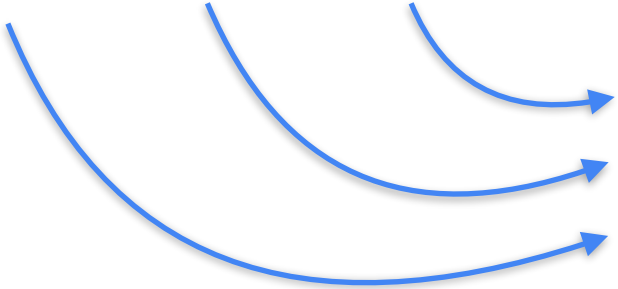
Client want to outsource this to the server

The server computes this for the client

$$\vec{r} = \mathbf{H} \cdot \vec{e} \implies \text{MSM}(\vec{g}, \vec{r}) = \text{MSM}(\vec{g}, \mathbf{H} \cdot \vec{e}) = \text{MSM}(\mathbf{H}^T \cdot \vec{g}, \vec{e})$$

Building EMSM from Dual-LPN

Fix the bases $\vec{g}$ then $\text{MSM}(\vec{g}, \vec{z} + \vec{r}) = \text{MSM}(\vec{g}, \vec{z}) + \text{MSM}(\vec{g}, \vec{r})$



Client computes this on their own

Client want to outsource this to the server

The server computes this for the client

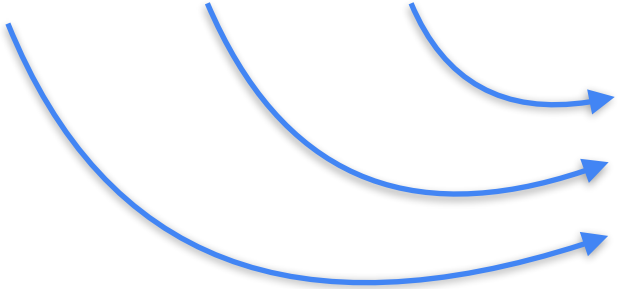
$\vec{r} = \mathbf{H} \cdot \vec{e} \implies \text{MSM}(\vec{g}, \vec{r}) = \text{MSM}(\vec{g}, \mathbf{H} \cdot \vec{e}) = \text{MSM}(\mathbf{H}^T \cdot \vec{g}, \vec{e})$

since $\vec{e}$ is sparse, this MSM is efficient

$\vec{h} = \mathbf{H}^T \cdot \vec{g}$ can be precomputed

Building EMSM from Dual-LPN

Fix the bases $\vec{g}$ then $\text{MSM}(\vec{g}, \vec{z} + \vec{r}) = \text{MSM}(\vec{g}, \vec{z}) + \text{MSM}(\vec{g}, \vec{r})$



Client computes this on their own

Client want to outsource this to the server

The server computes this for the client

$\vec{r} = \mathbf{H} \cdot \vec{e} \implies \text{MSM}(\vec{g}, \vec{r}) = \text{MSM}(\vec{g}, \mathbf{H} \cdot \vec{e}) = \text{MSM}(\mathbf{H}^T \cdot \vec{g}, \vec{e})$

since $\vec{e}$ is sparse, this MSM is efficient

$\vec{h} = \mathbf{H}^T \cdot \vec{g}$ can be precomputed

However if H has no structure, it requires quadric group operations, it can take days for an MSM of size on million

Concrete Instantiation

We instantiate dual-LPN with RAA codes [DJM98, BCF+25]

The generator matrix: $\mathbf{H} = \mathbf{F}_r \cdot \mathbf{M}_{\sigma_1} \cdot \mathbf{A} \cdot \mathbf{M}_{\sigma_2} \cdot \mathbf{A} \in \mathbb{F}^{n \times N}$ where N=nr, such that:

1) $\mathbf{F}_r$: r-repetition matrix, repeats each entry of input r times.

2) $\mathbf{M}_\sigma$: permutation matrix, corresponding to a matrix $\sigma : [N] \to [N]$

3) $\mathbf{A}$: an accumulation matrix, i.e., upper triangle with all 1 non-zero entries.

Concrete Instantiation

We instantiate dual-LPN with RAA codes [DJM98, BCF+25]

The generator matrix: $\mathbf{H} = \mathbf{F}_r \cdot \mathbf{M}_{\sigma_1} \cdot \mathbf{A} \cdot \mathbf{M}_{\sigma_2} \cdot \mathbf{A} \in \mathbb{F}^{n \times N}$ where N=nr, such that:

1) $\mathbf{F}_r$: r-repetition matrix, repeats each entry of input r times.

2) $\mathbf{M}_\sigma$: permutation matrix, corresponding to a matrix $\sigma : [N] \to [N]$

3) $\mathbf{A}$: an accumulation matrix, i.e., upper triangle with all 1 non-zero entries.

For concrete terms, the client does $(12 \cdot n)\ \mathbb{F}_{\text{ADD}} + t\ \mathbb{G}_{\text{EXP}}$ instead of $n\ \mathbb{G}_{\text{EXP}}$

Concrete Instantiation

We instantiate dual-LPN with RAA codes [DJM98, BCF+25]

The generator matrix: $\mathbf{H} = \mathbf{F}_r \cdot \mathbf{M}_{\sigma_1} \cdot \mathbf{A} \cdot \mathbf{M}_{\sigma_2} \cdot \mathbf{A} \in \mathbb{F}^{n \times N}$ where N=nr, such that:

1) $\mathbf{F}_r$: r-repetition matrix, repeats each entry of input r times.

2) $\mathbf{M}_\sigma$: permutation matrix, corresponding to a matrix $\sigma : [N] \to [N]$

3) $\mathbf{A}$: an accumulation matrix, i.e., upper triangle with all 1 non-zero entries.

For concrete terms, the client does $(12 \cdot n)\ \mathbb{F}_{\text{ADD}} + t\ \mathbb{G}_{\text{EXP}}$ instead of $n\ \mathbb{G}_{\text{EXP}}$

The preprocessing requires only $12 \cdot n\ \mathbb{G}_{\text{EXP}}$ for the setup. This setup is untrusted and can be used many times.

Security of the RAA Code

- At the time of writing the paper, RAA codes were only analyzed for binary field [BCF+25].
- We propose an empirical analysis over arbitrary finite fields.

Security of the RAA Code

- At the time of writing the paper, RAA codes were only analyzed for binary field [BCF+25].
- We propose an empirical analysis over arbitrary finite fields.

Distance of RAA Codes over Large Finite Fields (with Applications in zkSNARKs and PCGs)

Pariya Akhiani and Yupeng Zhang

University of Illinois Urbana Champaign
{akhiani2,zhangyp}@illinois.edu

Server-Aided zk-SNARKs via EMSM

- At a high-level, the prover does the field operations locally and outsources the MSMs.
- This simple idea yields efficient server-aided zk-SNARK for many group-based schemes: Groth16, Plonk, Nova, Sonic, HyperNova, KZH-Fold, etc.
- The prover for these schemes only does field operation and only a constant number of group operations.

Server-Aided zk-SNARKs via EMSM

- At a high-level, the prover does the field operations locally and outsources the MSMs.
- This simple idea yields efficient server-aided zk-SNARK for many group-based schemes: Groth16, Plonk, Nova, Sonic, HyperNova, KZH-Fold, etc.
- The prover for these schemes only does field operation and only a constant number of group operations.

Field addition is 10000x cheaper than group exponentiation

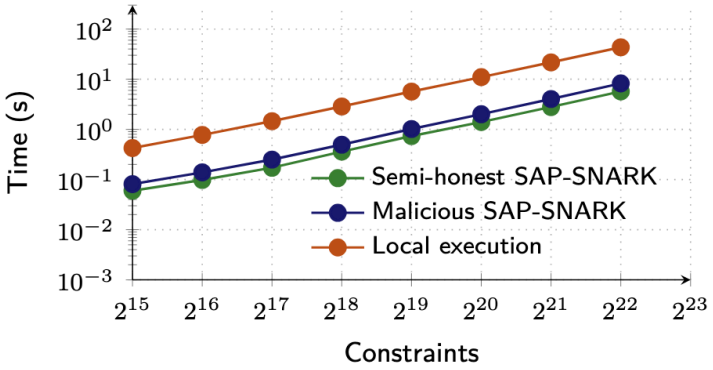
Security against Malicious Server

The client chooses a random $c \in \mathbb{F}$ and outsources MSM$(\vec{g}, \vec{z})$ and MSM$(\vec{g}, c \cdot \vec{z})$

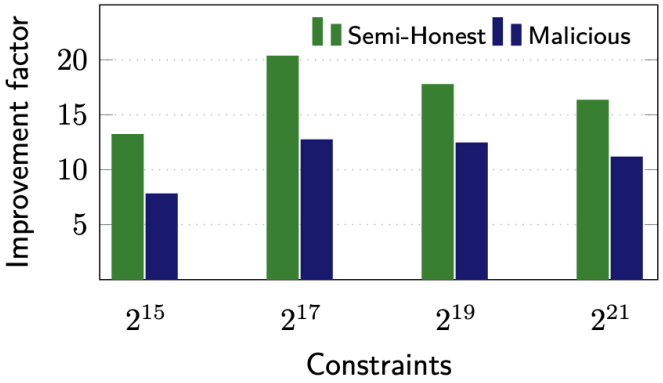
Given MSM$(\vec{g}, \vec{z})$ and MSM$(\vec{g}, c \cdot \vec{z})$, the client checks that $\implies$ $c \cdot \text{MSM}(\vec{g}, \vec{z}) = \text{MSM}(\vec{g}, c \cdot \vec{z})$

Performance of Server-Aided Nova

Lowers the client's active computation by up 20x and latency by 10x



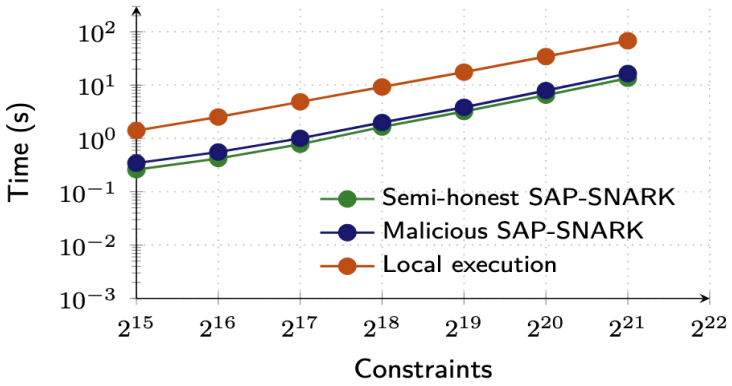
(a) Latency of proof generation (log-log scale).



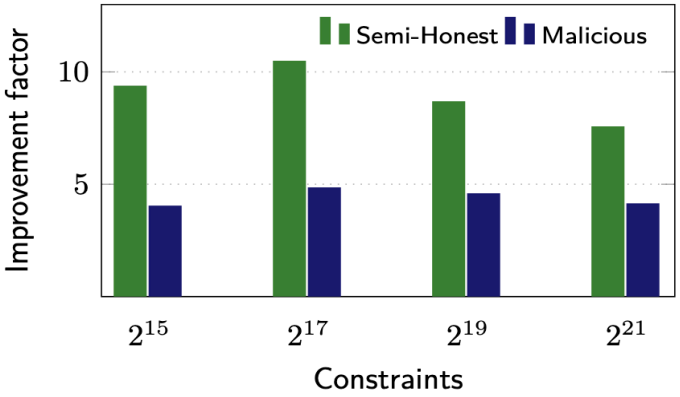
(b) Improvement in the client's active runtime.

Performance of Server-Aided Groth16

Lowers the client's active computation by up 10x and latency by 6x



(a) Latency of proof generation (log-log scale).



(b) Improvement in the client's active runtime.

Conclusion and Future Work

- We introduce server-aided zk-SNARKs for popular group-based zk-SNARKs such as Plonk, Nova and Groth16.
- The techniques of EMSM and analysis of RAA code over large fields might be of independent interest.

Conclusion and Future Work

- We introduce server-aided zk-SNARKs for popular group-based zk-SNARKs such as Plonk, Nova and Groth16.
- The techniques of EMSM and analysis of RAA code over large fields might be of independent interest.

Future Work: Memory improvement for the client side

ZKPROOF

Paving the path to adoption

Thank you!

ZKPROOF

Paving the path to adoption

Questions?