# Post-Quantum Designated-Verifier zkSNARKs from Lattices

David Wu

October 2021

# Argument Systems

[GMR85]

$x \in \{0,1\}^{n(\lambda)}$

$C = \{C_\lambda\}_{\lambda \in \mathbb{N}}$: family of Boolean circuits of size $s(\lambda)$

accept if $x \in \mathcal{L}$

$$\mathcal{L}_C = \{x : C_\lambda(x, w) = 1 \text{ for some } w\}$$

prover

verifier

**Completeness:** $\forall x \in \mathcal{L}_C : \Pr[\langle P(x,w), V(x)\rangle = \text{accept}] = 1$

*“Honest prover convinces honest verifier of true statements”*

**Soundness:** $\forall x \notin \mathcal{L}_C,\ \forall \text{ efficient } P^* : \Pr\left[\langle P^*(1^\lambda, x), V(x)\rangle = \text{accept}\right] = \text{negl}(\lambda)$

*“Efficient prover cannot convince honest verifier of false*

# Argument Systems

[GMR85]

$x \in \{0,1\}^{n(\lambda)}$

accept if
$x \in \mathcal{L}$

$$\mathcal{L}_C = \{x : C_\lambda(x, w) = 1 \text{ for some } w\}$$

prover

verifier

Argument system is **succinct** if:

- Prover communication is $\mathrm{poly}(\lambda + \log|C_\lambda|)$
- Running time of $V$ is $\mathrm{poly}(\lambda + |x| + \log|C_\lambda|)$

Both must be smaller than classic NP verification

# Succinct Non-Interactive Arguments (SNARGs)

[Kil92, Mic00, GW11]

$$\mathcal{L}_C = \{x : C_\lambda(x, w) = 1 \text{ for some } w\}$$

prover

$(x, w)$

$\pi = P(x, w)$

Argument consists of a single message

verifier

$x$

accept if $V(x, \pi) = 1$

Additional properties of interest:

- **Proof of knowledge**: succinct non-interactive argument of knowledge (SNARK):

*"There exists an efficient extractor that can recover a witness from any prover that convinces an honest verifier"*

# Succinct Non-Interactive Arguments (SNARGs)

[Kil92, Mic00, GW11]

$$\mathcal{L}_C = \{x : C_\lambda(x, w) = 1 \text{ for some } w\}$$

prover

$(x, w)$

$\pi = P(x, w)$

Argument consists of a single message

verifier

$x$

accept if $V(x, \pi) = 1$

Additional properties of interest:

- **Zero-knowledge**: *"Proof does not leak information about the prover's witness"*
- **zkSNARK:** zero-knowledge succinct non-interactive argument of knowledge

# Succinct Non-Interactive Arguments (SNARGs)

[Kil92, Mic00, GW11]

$$\mathcal{L}_C = \{x : C_\lambda(x, w) = 1 \text{ for some } w\}$$

prover

$(x, w)$

$$\pi = P(x, w) \longrightarrow$$

verifier

$x$

accept if $V(x, \pi) = 1$

For general NP languages, SNARGs are
<u>unlikely</u> to exist in standard model [BP04, Wee05]

# Succinct Non-Interactive Arguments (SNARGs)

[Kil92, Mic00, GW11]

**Instantiation:** “CS proofs” in the random oracle model [Mic94]


random oracle ($\mathcal{RO}$)

prover

$(x, w)$

$$\pi = P^{\mathcal{RO}}(x, w)$$

Argument consists of a single message

verifier

$x$

accept if $V^{\mathcal{RO}}(x, \pi) = 1$

# Succinct Non-Interactive Arguments (SNARGs)

[Kil92, Mic00, GW11]

Preprocessing SNARGs: allow “expensive” setup

$\text{Setup}(1^\lambda)$

Can consider publicly-verifiable and secretly-verifiable SNARGs

common reference string (CRS)

verification state

$\sigma$ $\tau$

prover

$(x, w)$

$\pi = P(\sigma, x, w)$

verifier

$x$

accept if $V(\tau, x, \pi) = 1$

# Succinct Non-Interactive Arguments (SNARGs)

[Kil92, Mic00, GW11]

Very active area of research (encompassing both theory and practice):

PHGR13, BCI⁺13, BCC⁺16, Gro16, ZGK⁺17, AHIV17, WTS⁺18, GMNO18, BBB⁺18, BBHR19, BCR⁺19, XZZ⁺19, LM19, CHM⁺20, BFS20, SL20, Set20, COS20, CY21, GNS21, GMN21, GLS⁺21, and *many, many more…*

**This talk:** post-quantum constructions (specifically, from lattice-based assumptions)

prover

$(x, w)$

$\pi = P(\sigma, x, w)$

Argument consists of a single message

verifier

$x$

accept if $V(\tau, x, \pi) = 1$

# zkSNARK Constructions (with Implementation)

| Construction | Prover Complexity | Proof Size: Asymptotic | Proof Size: Concrete | Assumption |
|---|---|---|---|---|
| [Gro16] | $N \log N$ | $1$ | 128 bytes | Pairings |
| Marlin [CHM+20] | $N \log N$ | $1$ | 704 bytes | Pairings |
| Xiphos [SL20] | $N$ | $\log N$ | 61 KB | Pairings |
| Fractal [COS20] | $N \log N$ | $\log^2 N$ | 215 KB | Random Oracle |
| STARK [BBHR19] | $N \operatorname{polylog} N$ | $\log^2 N$ | 127 KB* | Random Oracle |
| [GMNO18]† | $N \log N$ | $1$ | 640 KB | Lattices |

***Pre-Quantum*** (first three rows) · ***Post-Quantum*** (last three rows)

†designated-verifier

*for a structured computation

Focus is on constructions with a *succinct* verifier

$N$: size of NP relation being verified ($N \approx 2^{20}$ for concrete values)

Asymptotic metrics are given up to $\operatorname{poly}(\lambda)$ factors (for a security parameter $\lambda$)

# zkSNARK Constructions (with Implementation)

| Construction | Prover Complexity | Proof Size: Asymptotic | Proof Size: Concrete | Assumption |
|---|---|---|---|---|
| [Gro16] | $N \log N$ | $1$ | 128 bytes | Pairings |
| Marlin [CHM+20] | $N \log N$ | $1$ | 704 bytes | Pairings |
| Xiphos [SL20] | $N$ | $\log N$ | 61 KB | Pairings |
| *Pre-Quantum* | | | | |
| Fractal [COS20] | $N \log N$ | $\log^2 N$ | 215 KB | Random Oracle |
| STARK [BBHR19] | $N \operatorname{polylog} N$ | $\log^2 N$ | 127 KB* | Random Oracle |
| [GMNO18]† | $N \log N$ | $1$ | 640 KB | Lattices |
| *Post-Quantum* | | | | |

1000× gap between size of pre-quantum zkSNARKs and post-quantum ones

**This talk:** constructing shorter post-quantum zkSNARKs (via lattice-based assumptions)

# zkSNARK Constructions (with Implementation)

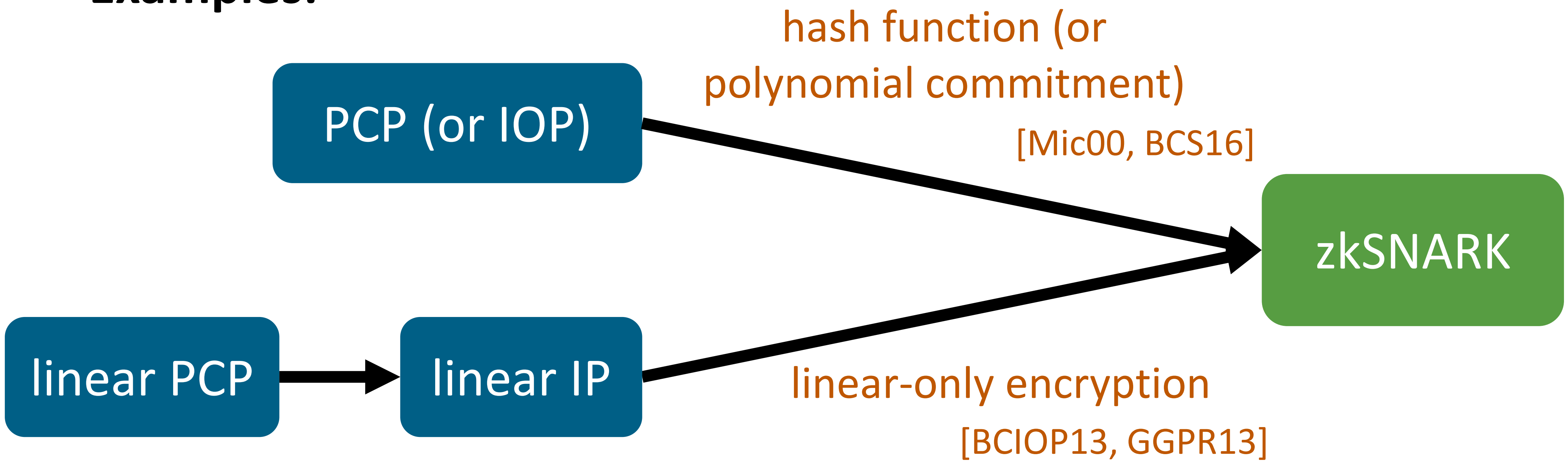
| Construction | Prover Complexity | Proof Size: Asymptotic | Proof Size: Concrete | Assumption |
|---|---|---|---|---|
| [Gro16] | $N \log N$ | $1$ | 128 bytes | Pairings |
| Marlin [CHM+20] | $N \log N$ | $1$ | 704 bytes | Pairings |
| Xiphos [SL20] | $N$ | $\log N$ | 61 KB | Pairings |
| *Pre-Quantum* | | | | |
| Fractal [COS20] | $N \log N$ | $\log^2 N$ | 215 KB | Random Oracle |
| STARK [BBHR19] | $N \operatorname{polylog} N$ | $\log^2 N$ | 127 KB* | Random Oracle |
| [GMNO18]† | $N \log N$ | $1$ | 640 KB | Lattices |
| **This work** | $\boldsymbol{N \log N}$ | $\mathbf{1}$ | **16 KB** | **Lattices** |
| *Post-Quantum* | | | | |

- ≈10× shorter proofs compared to previous post-quantum zkSNARKs for general NP relations
- Prover and verifier are concretely faster compared to most succinct pre-quantum construction [Gro16]
- Construction is designated-verifier (need secret key to check proofs) and has long CRS

# Construction Overview

Follows the classic approach of combining an information-theoretic proof system (for NP) with a cryptographic compiler

**Examples:**

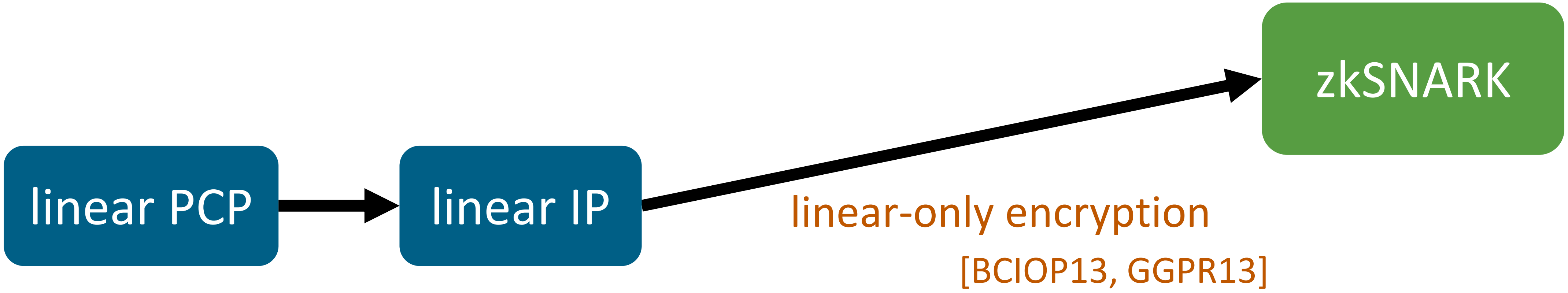


# Construction Overview

Follows the classic approach of combining an information-theoretic proof system (for NP) with a cryptographic compiler

**Starting point:** the [BCIOP13] compiler from linear PCPs to zkSNARKs

- Yields the most succinct pre-quantum zkSNARKs [GGPR13, Gro16]
- Basis of several lattice-based zkSNARKs [BISW17, GMNO18]



# Linear Probabilistically-Checkable Proofs (LPCPs)

[IKO07]

$(x, w)$ → $\boldsymbol{\pi} \in \mathbb{F}^m$

linear PCP

"encoding" of statement/witness

$\boldsymbol{\pi} \in \mathbb{F}^m$

$\boldsymbol{q} \in \mathbb{F}^m$

$\boldsymbol{q}^{\mathrm{T}}\boldsymbol{\pi} \in \mathbb{F}$

verifier

→ accept/reject

- Verifier given oracle access to a *linear* function $\pi \in \mathbb{F}^m$
- Several instantiations:
  - 3-query LPCP based on the Walsh-Hadamard code: $m = O(|C|^2)$ [ALMSS92]
  - 4-query LPCP based on quadratic arithmetic programs: $m = O(|C|)$ [GGPR13]

# Linear Probabilistically-Checkable Proofs (LPCPs)

[IKO07]

$(x, w)$ → $\boldsymbol{\pi} \in \mathbb{F}^m$

linear PCP
"encoding" of statement/witness

$\boldsymbol{\pi} \in \mathbb{F}^m$

$\boldsymbol{q} \in \mathbb{F}^m$

$\boldsymbol{q}^{\mathrm{T}}\boldsymbol{\pi} \in \mathbb{F}$

verifier

→ accept/reject

Oftentimes, verifier is *oblivious*:
the queries $q$ do not depend on
the statement $x$

# Linear Probabilistically-Checkable Proofs (LPCPs)

[IKO07]

Equivalent view (if verifier is oblivious):

$\boldsymbol{\pi} \in \mathbb{F}^m$

$\boldsymbol{Q} \in \mathbb{F}^{m \times k}$

$\boldsymbol{Q}^{\mathrm{T}} \boldsymbol{\pi} \in \mathbb{F}^k$

verifier

$\longrightarrow$ accept/reject

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix} \in \mathbb{F}^{m \times k}$$

pack all queries into single matrix

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}}_{\text{part of the CRS}}$$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

Two problems:

- Malicious prover can choose $\boldsymbol{\pi}$ based on queries
- Malicious prover can apply different $\boldsymbol{\pi}$ to the different columns of $\boldsymbol{Q}$

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}}_{\text{part of the CRS}}$$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^\mathrm{T}\boldsymbol{\pi}$

**Step 1:** Encrypt elements of $\boldsymbol{Q}$ using additively homomorphic encryption scheme

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}}_{\text{part of the CRS}}$$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

homomorphic evaluation

$\boldsymbol{q}_1^{\mathrm{T}}\boldsymbol{\pi}$ $\cdots$ $\boldsymbol{q}_k^{\mathrm{T}}\boldsymbol{\pi}$

SNARK proof

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

**Designated-verifier SNARK:**
decryption key needed to verify

If LPCP verification can be performed directly on ciphertexts (e.g., with pairing-based instantiations), then SNARK is **publicly-verifiable**

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

homomorphic evaluation

$\boldsymbol{q}_1^{\mathrm{T}}\boldsymbol{\pi}$ $\cdots$ $\boldsymbol{q}_k^{\mathrm{T}}\boldsymbol{\pi}$

SNARK proof

Verifier decrypts to learn $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$ and runs linear PCP decision procedure

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can “commit” to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\left[\, \boldsymbol{q}_1 \,\middle|\, \boldsymbol{q}_2 \,\middle|\, \boldsymbol{q}_3 \,\middle|\, \cdots \,\middle|\, \boldsymbol{q}_k \,\right]}_{\text{part of the CRS}}$$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

Two problems:

- Malicious prover can choose $\boldsymbol{\pi}$ based on queries
- Malicious prover can apply different $\boldsymbol{\pi}$ to the different columns of $\boldsymbol{Q}$

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}}_{\text{part of the CRS}}$$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

**[BCIOP13] approach:**

- Add a linear consistency check and view construction as a linear IP (LIP)
- Encrypt the LIP queries using a "linear-only" encryption scheme

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}}_{\text{part of the CRS}}$$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

**Intuitively:** an encryption scheme that <u>only</u> supports additive homomorphism

- Encrypt the LIP queries using a "linear-only" encryption scheme

# Linear-Only Encryption

[BCIOP13]

$x_1 \in \mathbb{F}$

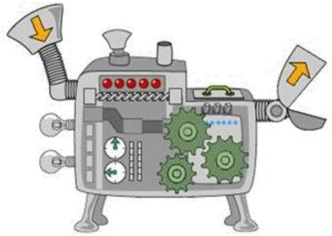
$x_2 \in \mathbb{F}$

$\vdots$

$x_n \in \mathbb{F}$

adversary

ct



extractor

$\alpha_1, \dots, \alpha_m \in \mathbb{F}$

**Requirement:** If $\mathrm{Decypt}(\mathrm{sk}, \mathrm{ct}) \neq \perp$, then $\mathrm{Decrypt}(\mathrm{sk}, \mathrm{ct}) = \sum_{i \in [n]} \alpha_i x_i$

**Intuition:** adversary's strategy can be "explained" by a linear function

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time

$$\boldsymbol{Q} = \underbrace{\begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}}_{\text{part of the CRS}}$$

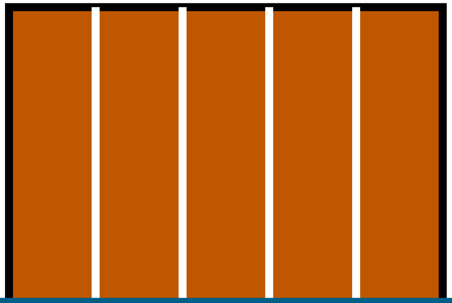
Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

All adversarial strategies can be explained by a linear function of the encrypted query components ⇒ soundness can now be based on the soundness of the linear PCP

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13]

*Oblivious* verifier can "commit" to its queries ahead of time



part of the CRS

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

For zero-knowledge, require that LPCP is (honest-verifier) ZK and encryption scheme is circuit private (hides linear combination)

**Rest of this talk:** will <u>not</u> focus on ZK

All adversarial strategies can be explained by a linear function of the encrypted query components ⇒ soundness can now be based on the soundness of the linear PCP

# Candidate Linear-Only Encryption from Lattices

[BISW17, GMNO18]

**Conjecture:** Regev encryption is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$

$\mathrm{Encrypt}(\boldsymbol{s}, \mu \in \mathbb{Z}_p)$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{s}^{\mathrm{T}} \boldsymbol{a} + pe + \mu)$$

$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, b)$ and output

$$\left(b - \boldsymbol{s}^{\mathrm{T}} \boldsymbol{a} \bmod q\right) \bmod p$$

Correct as long as $|e| \leq \frac{q}{2p}$

# Candidate Linear-Only Encryption from Lattices

[BISW17, GMNO18]

**Conjecture:** Regev encryption is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$

$\mathrm{Encrypt}(\boldsymbol{s}, \mu \in \mathbb{Z}_p)$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} + pe + \mu)$$

$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$:

Additive homomorphism:

- $\mathrm{ct}_1 = (\boldsymbol{a}_1, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a}_1 + pe_1 + \mu_1)$
- $\mathrm{ct}_2 = (\boldsymbol{a}_2, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a}_2 + pe_2 + \mu_2)$

Then:

$$\mathrm{ct}_1 + \mathrm{ct}_2 = (\boldsymbol{a}_1 + \boldsymbol{a}_2, \boldsymbol{s}^{\mathrm{T}}(\boldsymbol{a}_1 + \boldsymbol{a}_2) + p(e_1 + e_2) + (\mu_1 + \mu_2))$$

Homomorphic operations increase noise growth

# Candidate Linear-Only Encryption from Lattices

[BISW17, GMNO18]

**Conjecture:** Regev encryption is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$

$\mathrm{Encrypt}(\boldsymbol{s}, \mu \in \mathbb{Z}_p)$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} + pe + \mu)$$

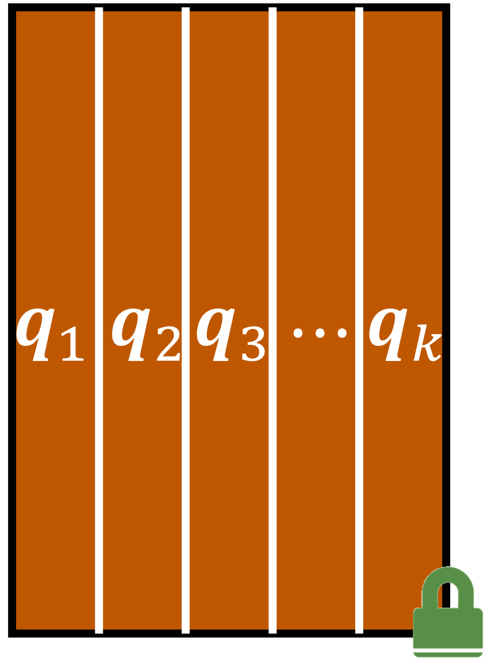
$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, b)$ and output

$$\left(b - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} \bmod q\right) \bmod p$$

While Regev encryption can be extended to obtain FHE, existing constructions require additional components or different message embedding

*Can we get more homomorphism from vanilla Regev?*

# Concrete Efficiency of Basic Instantiation

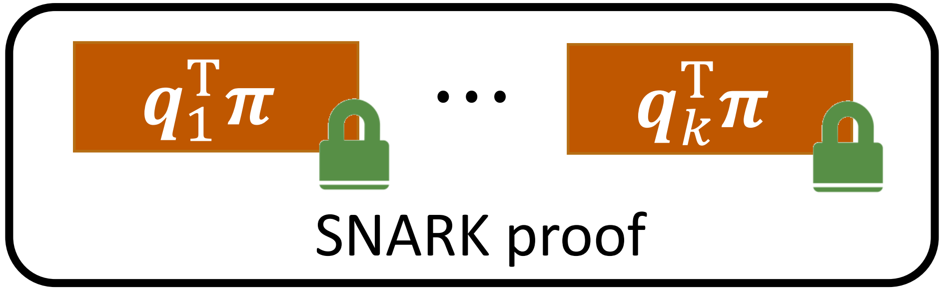
$\boldsymbol{Q} =$ 


common reference string

homomorphic evaluation

linear combinations of length $m$ over $\mathbb{F}_p$

Amount of homomorphism determines scheme parameters




Using quadratic arithmetic programs (for verifying circuit $C$):

- $k = 4$
- $m = O(|C|)$
- $\text{soundness} \approx \frac{2|C|}{|\mathbb{F}_p|} = \frac{2|C|}{p}$

# Concrete Efficiency of Basic Instantiation

$\boldsymbol{Q} = [\, q_1 \mid q_2 \mid q_3 \mid \cdots \mid q_k \,]$

homomorphic evaluation

$\longrightarrow$

linear combinations of length $m$ over $\mathbb{F}_p$

Amount of homomorphism determines scheme parameters

$\boldsymbol{q}_1^{\mathrm{T}}\boldsymbol{\pi} \quad \cdots \quad \boldsymbol{q}_k^{\mathrm{T}}\boldsymbol{\pi}$

SNARK proof

Need to choose encryption modulus $q$ to support this amount of homomorphism:

$$q/2p > p \cdot m \cdot B$$

where $B$ is the initial noise term

Using quadratic arithmetic programs (for verifying circuit $C$):

- $k = 4$
- $m = O(|C|)$
- $\text{soundness} \approx \frac{2|C|}{|\mathbb{F}_p|} = \frac{2|C|}{p}$

# Concrete Efficiency of Basic Instantiation

For a circuit with $m = 2^{20}$ gates and requiring 128 bits of soundness, we require:

- $p > 2^{148}$, so $q > 2^{300}$
- At 128 bits of security, lattice dimension $n > 10^4$, so a single Regev ciphertext is *over 350 KB* (longer than other post-quantum constructions based on IOPs)
- Proof contains $k$ ciphertexts, so proof is even longer

**Alternatively:** Use a small plaintext field $\mathbb{F}_p$ and amplify soundness via parallel repetition

- $p \approx 2^{20}$ and $q \approx 2^{100}$: single ciphertext is 45 KB
- Need many copies in this case ($\approx 128$ copies), so proof is again very long

[GMNO18]: use an instantiation where $p = 2^{32}$ *without* soundness amplification

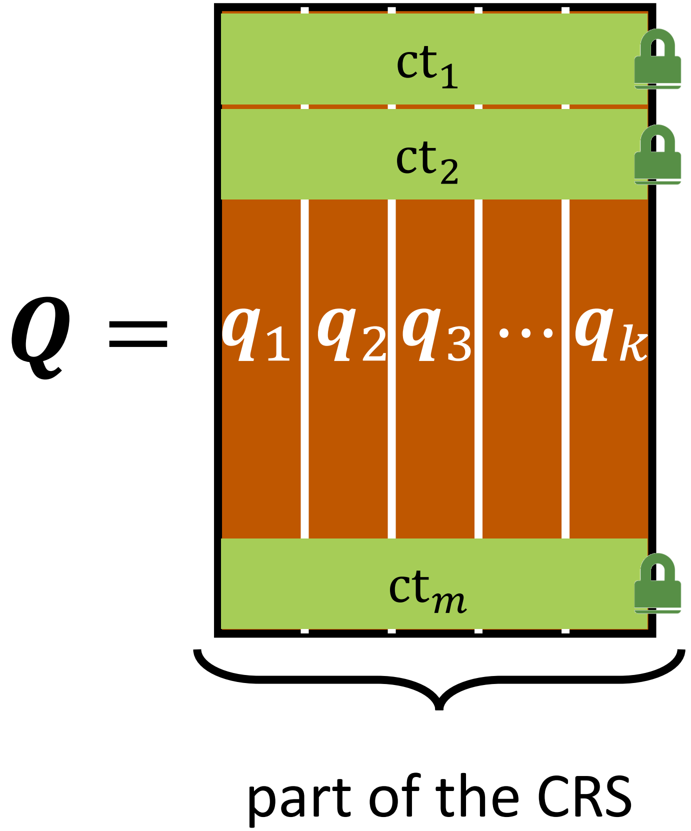
- Proofs are already 640 KB (and provide $\approx 15$ bits of provable soundness for verifying computations of size $2^{16}$)

**New techniques needed to reduce proof size**

# Revisiting the Bitansky et al. Compiler

[BISW17]

*Oblivious* verifier can "commit" to its queries ahead of time




Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

**Key idea:** Instead of encrypting each component of $\boldsymbol{Q}$ individually, encrypt rows instead

# Linear-Only Vector Encryption

[BISW17]

$v_1 \in \mathbb{F}^k$

$v_2 \in \mathbb{F}^k$

$\vdots$

$v_m \in \mathbb{F}^k$

plaintext space is a *vector* space

# Linear-Only Vector Encryption

[BISW17]

$\boldsymbol{v}_1 \in \mathbb{F}^k$

$\boldsymbol{v}_2 \in \mathbb{F}^k$

$\vdots$

$\boldsymbol{v}_m \in \mathbb{F}^k$

$$\sum_{i \in [m]} \alpha_i \boldsymbol{v}_i \in \mathbb{F}^k$$

supports homomorphic vector addition

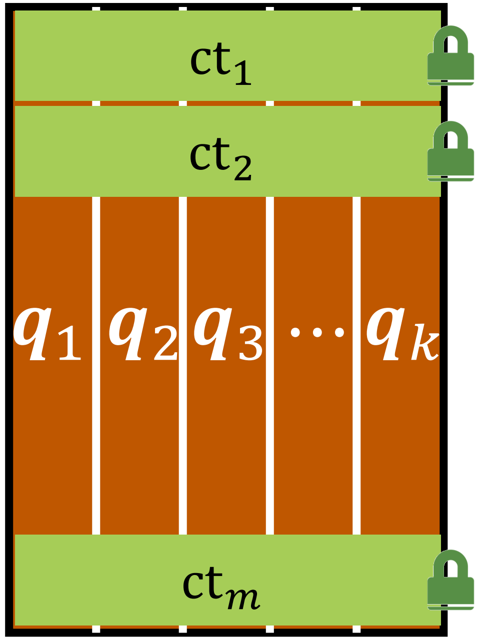
**Linear-only:** scheme only supports linear homomorphism

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13, BISW17]

common reference string

$\boldsymbol{Q} =$




Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

homomorphic evaluation

$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

SNARK proof

Verifier decrypts to learn $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$ and runs linear PCP decision procedure

# From Linear PCPs to Preprocessing SNARGs

[BCIOP13, BISW17]

common reference string

$\mathrm{ct}_1$

$\mathrm{ct}_2$

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 & \boldsymbol{q}_2 & \boldsymbol{q}_3 & \cdots & \boldsymbol{q}_k \end{bmatrix}$$

$\mathrm{ct}_m$

Honest prover takes $(x, w)$ and constructs linear PCP $\boldsymbol{\pi} \in \mathbb{F}^m$ and computes $\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

homomorphic evaluation

$\boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\pi}$

SNARK proof

- Proof is a single vector encryption ciphertext
- Allows direct compilation from linear PCPs to SNARKs (without extra linearity check from [BCIOP13])

# Candidate Linear-Only Vector Encryption

[BISW17]

**Conjecture:** Regev encryption is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$

$\mathrm{Encrypt}(\boldsymbol{s}, \mu \in \mathbb{Z}_p)$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} + pe + \mu)$$

$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, b)$ and output

$$(b - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} \bmod q) \bmod p$$

**Key observation:** the same vector $a \in \mathbb{Z}_q^n$ can be reused with many different secret keys

Amortized/vectorized variant of Regev encryption [PVW08]

# Candidate Linear-Only Vector Encryption

[BISW17]

**Conjecture:** Vectorized Regev encryption [PVW08] is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{s} \in \mathbb{Z}_q^n$

$\mathrm{Encrypt}(\boldsymbol{s}, \mu \in \mathbb{Z}_p)$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} + pe + \mu)$$

$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, b)$ and output

$$\left(b - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} \bmod q\right) \bmod p$$

# Candidate Linear-Only Vector Encryption

[BISW17]

**Conjecture:** Vectorized Regev encryption [PVW08] is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{S} \in \mathbb{Z}_q^{n \times k}$

$\mathrm{Encrypt}(\boldsymbol{s}, \mu \in \mathbb{Z}_p)$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $e \leftarrow \chi$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{s}^{\mathrm{T}} \boldsymbol{a} + pe + \mu)$$

$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, b)$ and output

$$\left(b - \boldsymbol{s}^{\mathrm{T}} \boldsymbol{a} \bmod q\right) \bmod p$$

# Candidate Linear-Only Vector Encryption

[BISW17]

**Conjecture:** Vectorized Regev encryption [PVW08] is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{S} \in \mathbb{Z}_q^{n \times k}$

$\mathrm{Encrypt}(\boldsymbol{S}, \boldsymbol{\mu} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{k}})$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $\boldsymbol{e} \leftarrow \chi^k$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a} + p\boldsymbol{e} + \boldsymbol{\mu})$$

$\mathrm{Decrypt}(\boldsymbol{s}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, b)$ and output

$$\left(b - \boldsymbol{s}^{\mathrm{T}}\boldsymbol{a} \bmod q\right) \bmod p$$

# Candidate Linear-Only Vector Encryption

[BISW17]

**Conjecture:** Vectorized Regev encryption [PVW08] is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{S} \in \mathbb{Z}_q^{n \times k}$

$\mathrm{Encrypt}(\boldsymbol{S}, \boldsymbol{\mu} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{k}})$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $\boldsymbol{e} \leftarrow \chi^k$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a} + p\boldsymbol{e} + \boldsymbol{\mu})$$

$\mathrm{Decrypt}(\boldsymbol{S}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{v})$ and output

$$\left(\boldsymbol{v} - \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a} \bmod q\right) \bmod p$$

$$|\mathrm{ct}| = (n + k)\log q$$

Would be $k(n+1)\log q$ using vanilla Regev

Ciphertext size is *additive* in the vector dimension

# Candidate Linear-Only Vector Encryption

[BISW17]

**Conjecture:** Vectorized Regev encryption [PVW08] is linear-only

$\mathrm{KeyGen}(1^\lambda)$: Outputs a secret key $\boldsymbol{S} \in \mathbb{Z}_q^{n \times k}$

$\mathrm{Encrypt}(\boldsymbol{S}, \boldsymbol{\mu} \in \mathbb{Z}_{\boldsymbol{p}}^{\boldsymbol{k}})$: Sample random $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$, error $\boldsymbol{e} \leftarrow \chi^k$ and output

$$\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a} + p\boldsymbol{e} + \boldsymbol{\mu})$$

$\mathrm{Decrypt}(\boldsymbol{S}, \mathrm{ct})$: Write $\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{v})$ and output

$$(\boldsymbol{v} - \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a} \bmod \mathrm{q}) \bmod p$$

Can use modulus switching [BV11, BGV12] to reduce ciphertext size <u>after</u> homomorphic evaluation: $(n+k)\log q \to (n+k)\log q'$

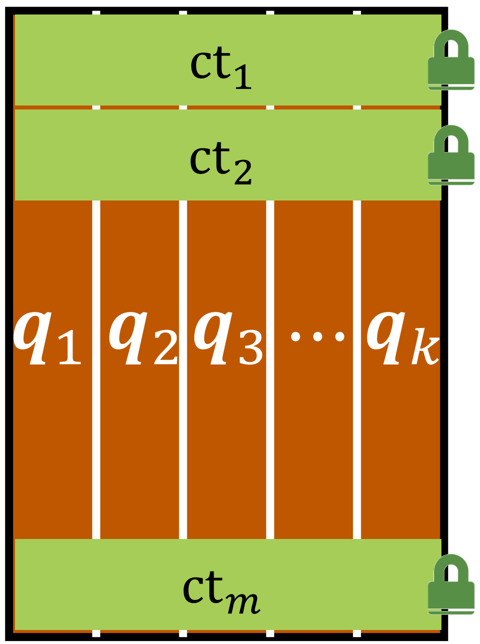
$|\mathrm{ct}| = (n +$

Ciphertext size is <u>*additive*</u> in the vector dimension

# Lattice-Based zkSNARKs using Vector Encryption

[BISW17, ISW21]

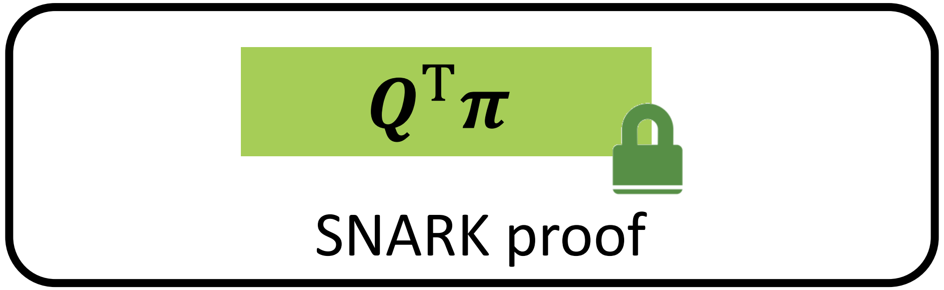
$\boldsymbol{Q} =$




common reference string

homomorphic evaluation

linear combinations of length $m$ over $\mathbb{F}_p$




Using quadratic arithmetic programs (for verifying circuit $C$):

- $k = 4$
- $m = O(|C|)$
- soundness $\approx \frac{2|C|}{|\mathbb{F}_p|} = \frac{2|C|}{p}$

# Lattice-Based zkSNARKs using Vector Encryption

[BISW17, ISW21]

**Previously techniques to achieve small soundness:**

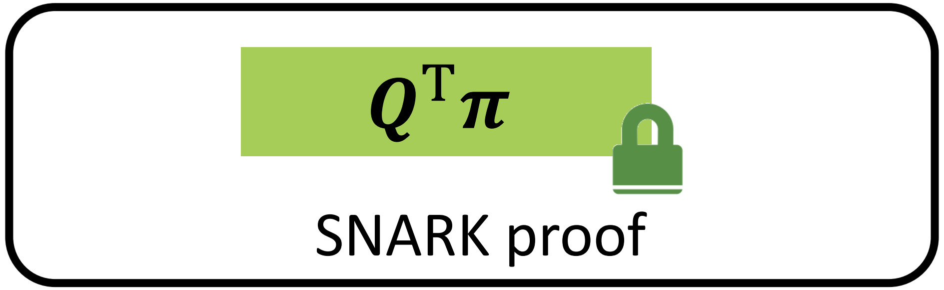
1. Use large $p$ (to ensure LPCP soundness); or
2. Use small $p$ and parallel repetition to amplify soundness

**Our approach:** parallel repetition of LPCP to amplify soundness:

- Define LPCP to be $t$ independent sets of queries
- Accept only if all $t$ sets accept
- Requires $kt$ LPCP queries and provides soundness $\left(\frac{|C|}{2p}\right)^t$

With vanilla [BCIOP13], same proof size as parallel repetition

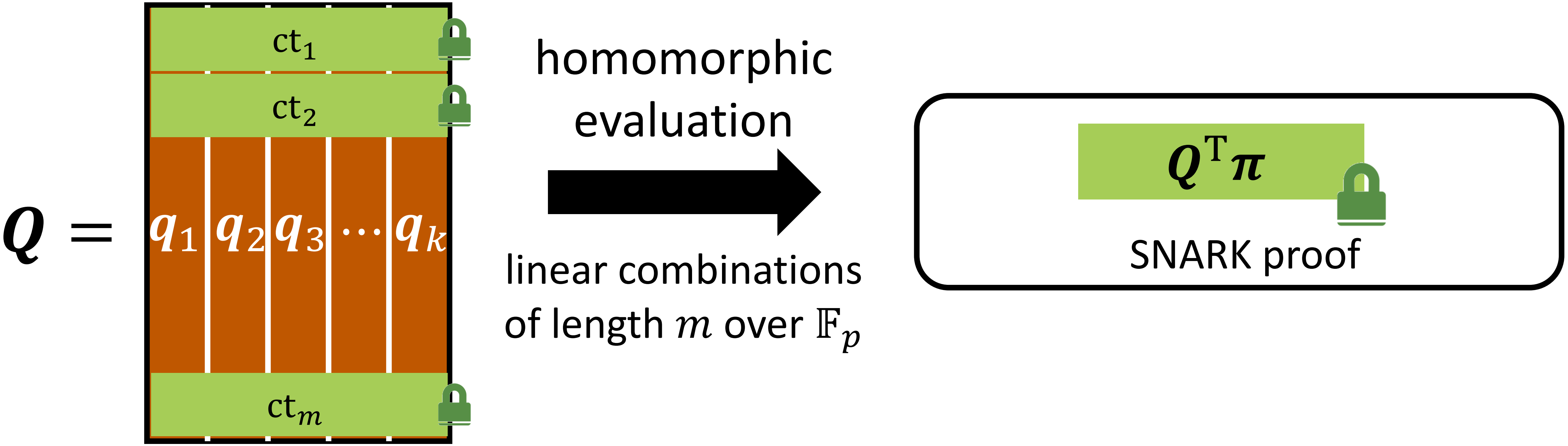
With vector encryption, proof is always a single vector encryption ciphertext and |ct| is *additive* in vector dimension (<u>not</u> multiplicative)




Setting $p \approx 2^{28}$, proof size is 29 KB (with a CRS of size 2.7 GB) for verifying circuit of size $2^{20}$

# Further Compression via Extensions Fields

[ISW21]




*Can we further reduce $p$?*

**Recall:** Noise growth in ciphertexts scales with

- Length $m$ of linear combination
- Magnitude of coefficients in linear combination $p$

Soundness of linear PCP: $\frac{2|C|}{|\mathbb{F}|}$

**Idea:** use an extension field of small <u>characteristic</u>

# Further Compression via Extensions Fields

[ISW21]

$(x, w)$ → $\pi \in \mathbb{F}^m$

linear PCP

Suppose $\mathbb{F} = \mathbb{F}_{p^k}$ where $k > 1$

Can still instantiate using quadratic arithmetic programs

Two approaches to compile to a SNARK:

- Compile LPCP over $\mathbb{F}_{p^k}$ to a LPCP over $\mathbb{F}_p$, apply linear-only vector encryption over $\mathbb{F}_p$

  Recall that $\mathbb{F}_{p^k} \cong \mathbb{F}_p^k$; field operations in $\mathbb{F}_{p^k}$ are linear transformations over $\mathbb{F}_p^k$

  Transformation increases number of queries and query dimension by $k$

- Apply linear-only vector encryption over $\mathbb{F}_{p^k}$

  Work over a polynomial ring $R = \mathbb{Z}[x]/\Phi_m(x)$ where $m$ is chosen so that $R/pR \cong \mathbb{F}_{p^k}$

  Consider Regev encryption over $R$ (using module lattices)

# Further Compression via Extensions Fields

[ISW21]

$(x, w)$ → $\pi \in \mathbb{F}^m$

linear PCP

Suppose $\mathbb{F} = \mathbb{F}_{p^k}$ where $k > 1$

Can still instantiate using quadratic arithmetic programs

Two approaches to compile to a SNARK:

- Compile LPCP over $\mathbb{F}_{p^k}$ to a LPCP over $\mathbb{F}$

  Recall that $\mathbb{F}_{p^k} \cong \mathbb{F}_p^k$; field operations in $\mathbb{F}_p$

  Transformation increases number of querie

- Apply linear-only vector encryption ove

  Work over a polynomial ring $R = \mathbb{Z}[x]/\Phi_m$

  Consider Regev encryption over $R$ (using mo

**In both settings:** coefficients of prover's linear combination have magnitude $\approx p$ while field has size $p^k$

# Further Compression via Extensions Fields

[ISW21]

$(x, w) \longrightarrow$ $\pi \in \mathbb{F}^m$

linear PCP

**This work:** consider <u>quadratic</u> extension fields

- $R = \mathbb{Z}[x]/(x^2 + 1)$ and set $p = 3 \bmod 4$ so $R_p = R/pR \cong \mathbb{F}_{p^2}$
- Choose ciphertext modulus $q$ to be a power of 2
  - All arithmetic operations can be implemented using 128-bit arithmetic
  - Low degree means polynomial arithmetic only slightly more expensive

# Further Compression via Extensions Fields

[ISW21]

$(x, w) \longrightarrow \boxed{\pi \in \mathbb{F}^m}$

linear PCP

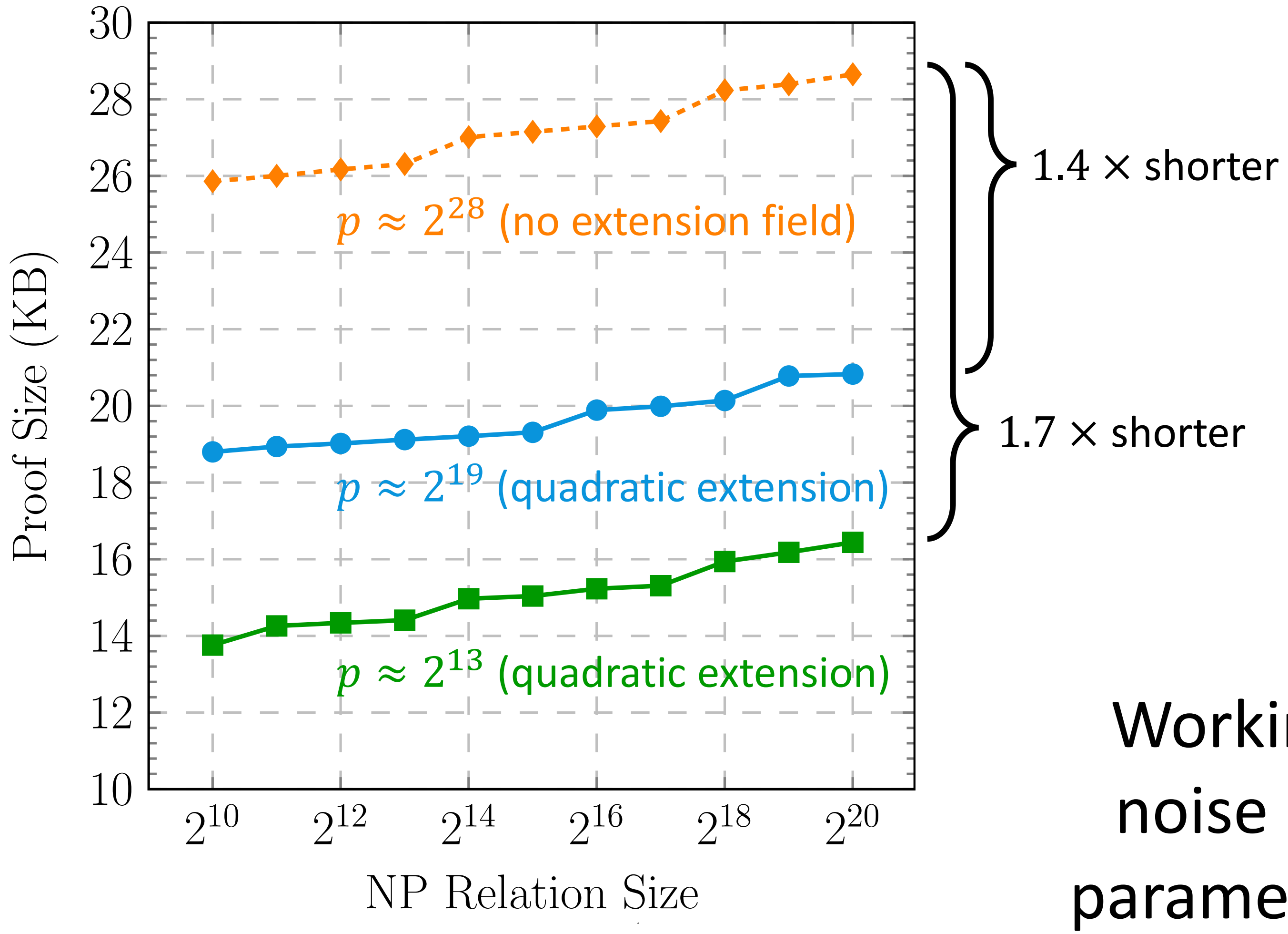
**This work:** consider <u>quadratic</u> extension fields

- $R = \mathbb{Z}[x]/(x^2+1)$ and set $p = 3 \bmod 4$ so
- Choose ciphertext modulus $q$ to be a power of 2
  - All arithmetic operations can be implemented using 128-bit arithmetic
  - Low degree means polynomial arithmetic only slightly more expensive
- Choose $p = 2^t \pm 1$ so $\mathbb{F}_{p^2}$ has $2^{t+1}$-th roots of unity (for efficient implementation of LPCP prover)

Higher-degree extension makes polynomial arithmetic more costly (or need non-power-of-two modulus to exploit FFTs)

# Further Compression via Extensions Fields

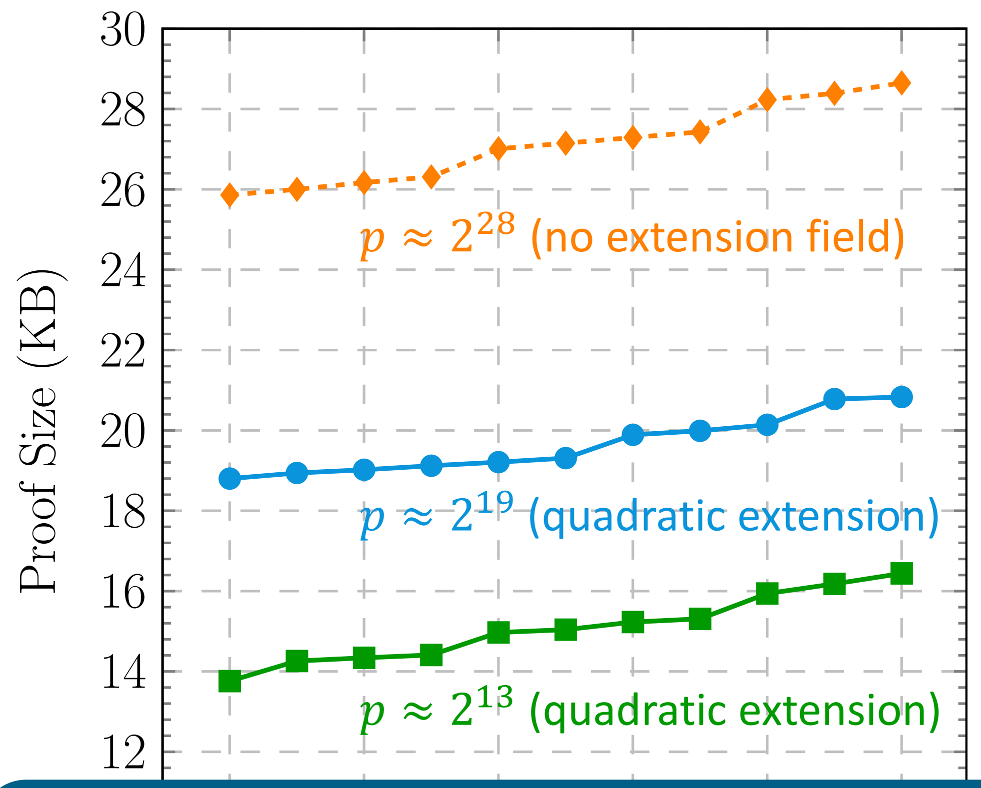
[ISW21]

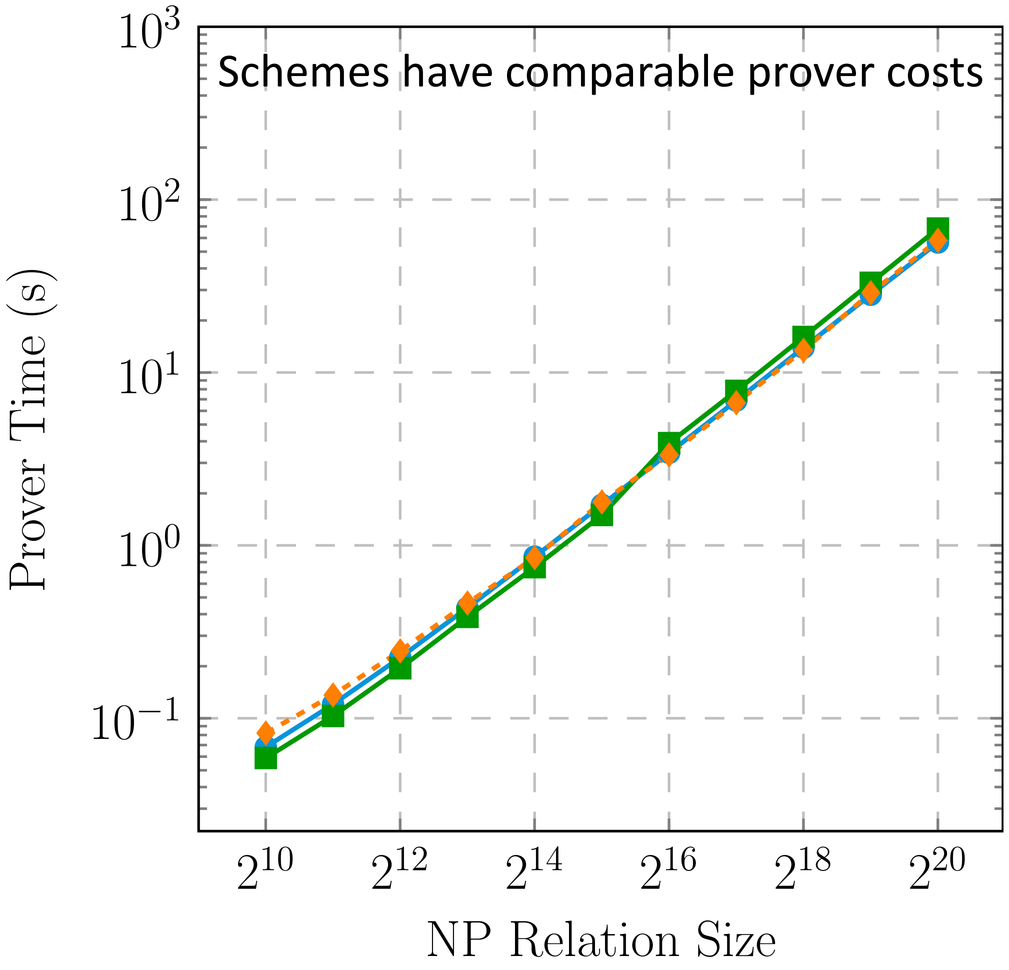



Working over extension field reduces noise accumulation ⇒ smaller lattice parameters ⇒ concretely shorter proofs

# Further Compression via Extensions Fields

[ISW21]

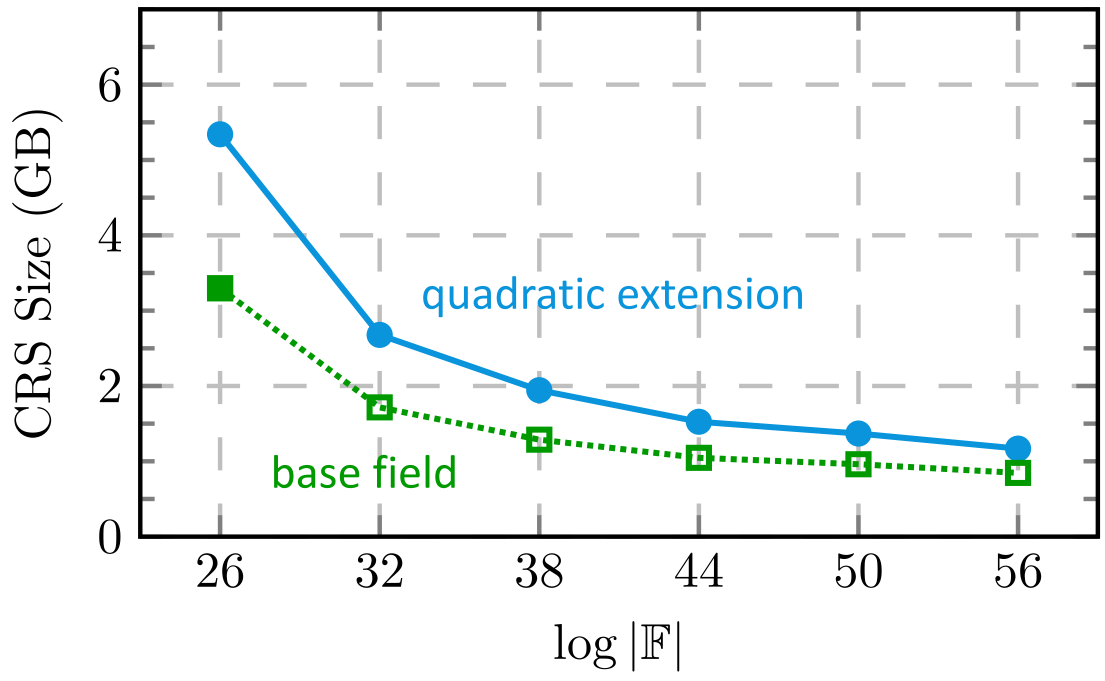


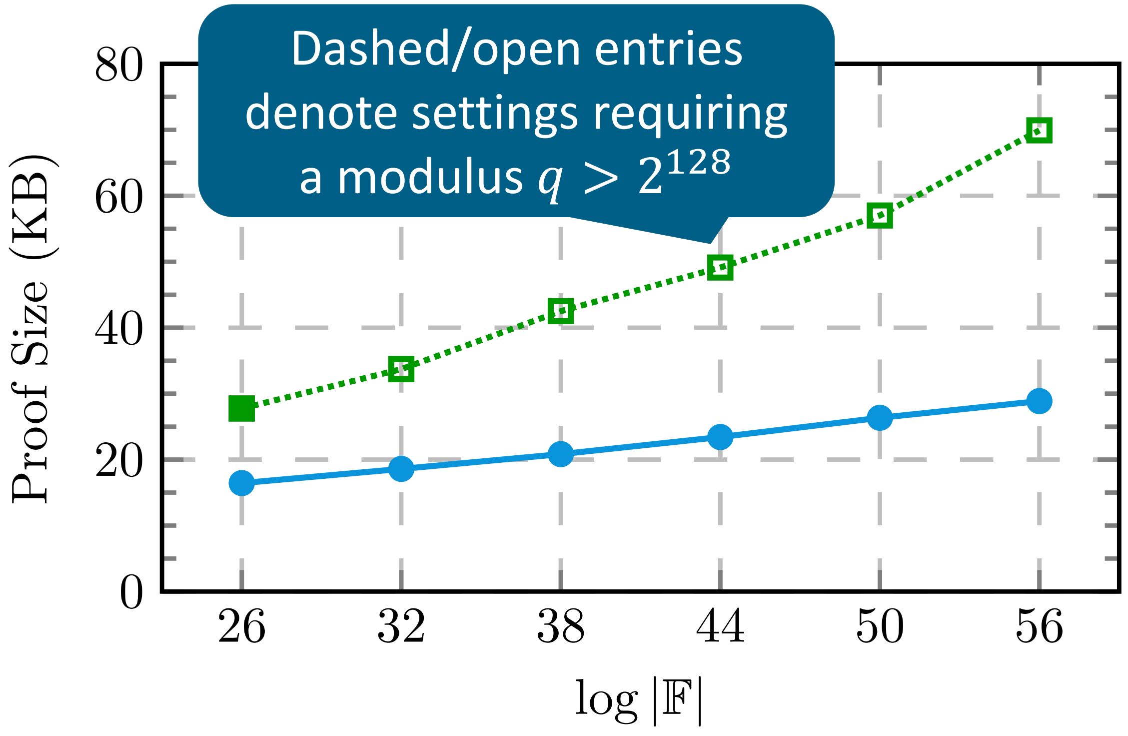




- Slightly more expensive homomorphic operations over extension field, but smaller lattice parameters
- Smaller field ⇒ more LPCP queries for soundness amplification ⇒ higher prover cost

# Effect of Field Size

[ISW21]




Using the extension field increases CRS size but decreases proof size

- CRS consists of “compressed” ciphertexts where random component is derived from a PRF (i.e., $\mathrm{ct} = (\boldsymbol{a}, \boldsymbol{v})$ where $\boldsymbol{a}$ is random and $v = \boldsymbol{S}^{\mathrm{T}}\boldsymbol{a} + p\boldsymbol{e} + \boldsymbol{\mu}$)
- Proof consists of full ciphertexts

[see paper for more microbenchmarks]

# Concrete Comparison with zkSNARKs

[ISW21]

| Construction | Size: CRS | Size: Proof | Time: Setup | Time: Prover | Time: Verifier | Assumption |
|---|---|---|---|---|---|---|
| [Gro16] | 199 MB | 128 bytes | 72 s | 79 s | 3.4 ms | Pairings |
| | | | | | | ***Pre-Quantum*** |
| Ligero [AHIV17] | – | 14 MB | – | 38 s | 22 s | Random Oracle |
| Aurora [BCR$^+$19] | – | 169 KB | – | 304 s | 6.3 s | Random Oracle |
| Fractal [COS20] | 11 GB | 215 KB | 116 s | 184 s | 9.5 ms | Random Oracle |
| **This work** | **5.3 GB** | **16.4 KB** | **2240 s** | **68 s** | **1.2 ms** | **Lattices** |
| **This work** | **1.9 GB** | **20.8 KB** | **877 s** | **56 s** | **0.4 ms** | **Lattices** |
| | | | | | | ***Post-Quantum*** |

All benchmarks collected on same hardware for verifying NP relation of size $2^{20}$

# Concrete Comparison with zkSNARKs

[ISW21]

| | Size | | Time | | | |
|---|---|---|---|---|---|---|
| **Construction** | **CRS** | **Proof** | **Setup** | **Prover** | **Verifier** | **Assumption** |
| [Gro16] | 199 MB | 128 bytes | | | | |
| Ligero [AHIV17] | – | 14 MB | | | | |
| Aurora [BCR+19] | – | 169 KB | | | | |
| Fractal [COS20] | 11 GB | 215 KB | | | | |
| **This work** | 5.3 GB | **16.4 KB** | | | | |
| **This work** | 1.9 GB | **20.8 KB** | 877 s | 56 s | 0.4 ms | Lattices |
| | | | | | | *Post-Quantum* |

Over 10.3× shorter than other post-quantum SNARKs

Still over 131× longer than pairing-based SNARKs

Over 42× shorter than previous lattice-based SNARKs [GMNO18] (based on reported numbers for verifying circuit of size $2^{16}$)

All benchmarks collected on same hardware for verifying NP relation of size $2^{20}$

# Concrete Comparison with zkSNARKs

[ISW21]

| | Size | | Time | | | |
|---|---|---|---|---|---|---|
| **Construction** | **CRS** | **Proof** | **Setup** | **Prover** | **Verifier** | **Assumption** |
| [Gro16] | | | | 79 s | 3.4 ms | Pairings *Pre-Quantum* |
| Ligero [AHIV17] | | | | 38 s | 22 s | |
| Aurora [BCR+19] | | | | 304 s | 6.3 s | |
| Fractal [COS20] | | | | 184 s | 9.5 ms | |
| **This work** | | | | **68 s** | **1.2 ms** | |
| **This work** | | | | **56 s** | **0.4 ms** | Lattices *Post-Quantum* |

Prover cost is essentially cost of LPCP prover and computing a linear combination

1.2× faster than pairing-based SNARKs

Slower than schemes like Ligero based on MPC-in-the-head (which does not have succinct verification)

If we consider restricted computations, can have much faster provers (e.g., ethSTARK [BBHR19])

All benchmarks collected on same hardware for verifying NP relation of size $2^{20}$

# Concrete Comparison with zkSNARKs

[ISW21]

| Construction | Size: CRS | Size: Proof | Time: Setup | Time: Prover | Time: Verifier | Assumption |
|---|---|---|---|---|---|---|
| [Gro16] | | | | | 3.4 ms | Pairings *Pre-Quantum* |
| Ligero [AHIV17] | | | | | 22 s | Random Oracle |
| Aurora [BCR+19] | | | | | 6.3 s | Random Oracle |
| Fractal [COS20] | | | | | 9.5 ms | Random Oracle |
| **This work** | | | | | **1.2 ms** | Lattices |
| **This work** | | | | | **0.4 ms** | Lattices *Post-Quantum* |

Lattice-based SNARKs have very lightweight verification: computing a matrix-vector product ($\approx$ 200,000 integer multiplications) and rounding

Well-suited for lightweight or energy-constrained devices

All benchmarks collected on same hardware for verifying NP relation of size $2^{20}$

# Concrete Comparison with zkSNARKs

[ISW21]

| | Size | | Time | | | |
|---|---|---|---|---|---|---|
| **Construction** | **CRS** | **Proof** | **Setup** | **Prover** | **Verifier** | **Assumption** |
| [Gro16] | 199 MB | 128 bytes | 72 s | | | |
| Ligero [AHIV17] | – | 14 MB | – | | | |
| Aurora [BCR+19] | – | 169 KB | – | | | |
| Fractal [COS20] | 11 GB | 215 KB | 116 s | | | |
| **This work** | **5.3 GB** | **16.4 KB** | **2240 s** | | | |
| **This work** | **1.9 GB** | **20.8 KB** | **877 s** | | | |

Limitations of lattice-based SNARKs:

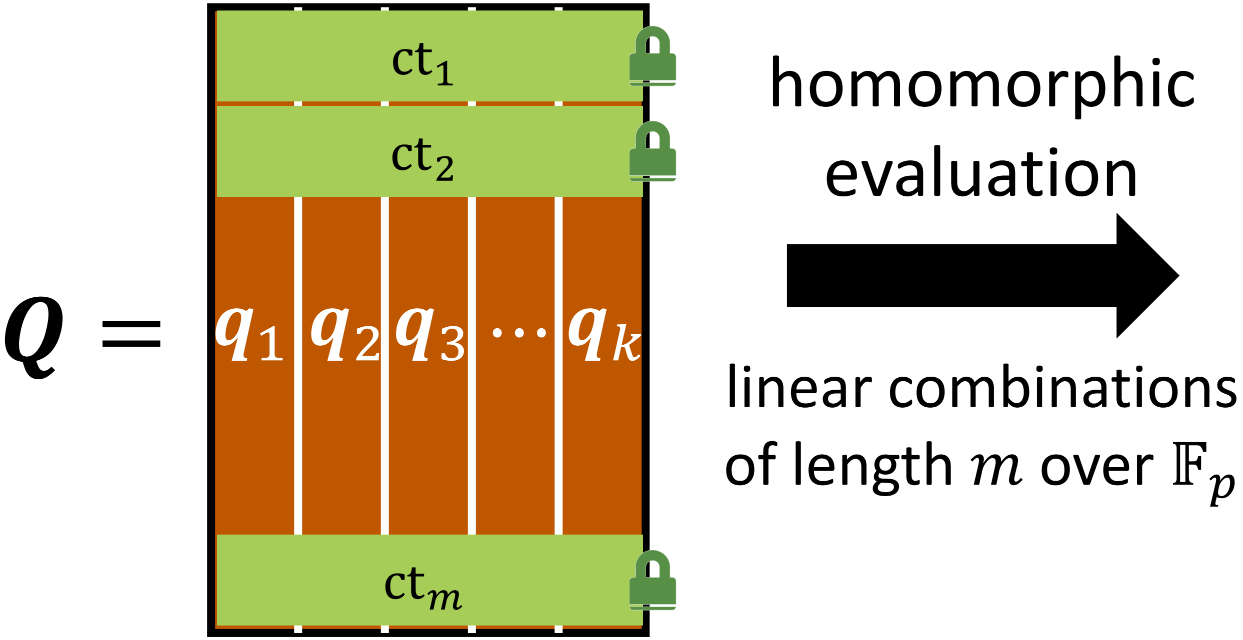
- Resulting construction is designated-verifier (other schemes are publicly-verifiable)
- Require expensive trusted setup (need to encrypt large number of vectors)
- Resulting CRS is large (lattice ciphertexts still large, even with compression)

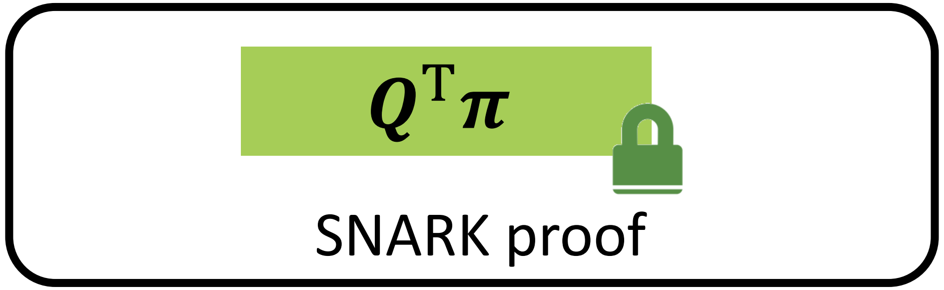
All benchmarks collected on same hardware for verifying NP relation of size $2^{20}$

# Summary

Directly compile linear PCPs to SNARKs using linear-only vector encryption

Instantiate linear-only vector encryption from vectorized Regev encryption




Work over extension fields for better concrete efficiency

[see paper for further optimizations]

# Open Problems

Concretely-efficient **publicly-verifiable** SNARKs from lattices

Constructions with short proofs but expensive verifiers are known from lattices [BBC+18, BLNS20]

Concretely-efficient designated-verifier SNARKs with **reusable soundness** from lattices

**Thank you!**

`https://eprint.iacr.org/2021/977`

`https://github.com/lattice-based-zkSNARKs/lattice-zksnark`