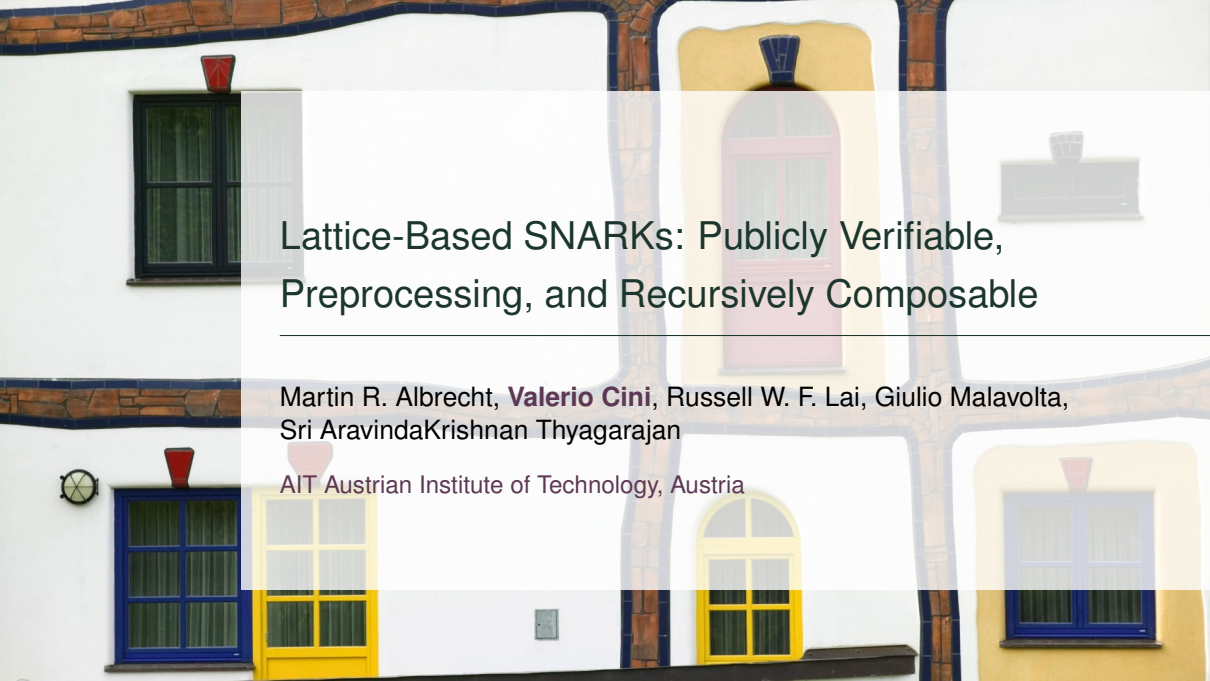

Lattice-Based SNARKs: Publicly Verifiable, Preprocessing, and Recursively Composable

Martin R. Albrecht, **Valerio Cini**, Russell W. F. Lai, Giulio Malavolta, Sri AravindaKrishnan Thyagarajan

AIT Austrian Institute of Technology, Austria

Agenda

- Succinct Non-Interactive Arguments of Knowledge (SNARK)
- Vector Commitments (VC) with Functional Openings
- Lattice-based VC with Openings to Polynomial Maps
- Further Results

Agenda

- Succinct Non-Interactive Arguments of Knowledge (SNARK)
- Vector Commitments (VC) with Functional Openings
- Lattice-based VC with Openings to Polynomial Maps
- Further Results

Succinct Non-Interactive Arguments of Knowledge (SNARK) for $\mathcal{L}$

- NP language $\mathcal{L} = \{\mathsf{stmt} : \exists\, \mathsf{wit},\ R(\mathsf{stmt}, \mathsf{wit}) = 1\}$
- SNARK for $\mathcal{L}$ is a tuple $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$	
$\pi \leftarrow \mathsf{Prove}(\mathsf{pp}, \mathsf{stmt}, \mathsf{wit})$	π	$\mathsf{pp}_{\mathsf{stmt}} \leftarrow \mathsf{Preproc}(\mathsf{pp}, \mathsf{stmt})$
		$b \leftarrow \mathsf{Verify}(\mathsf{pp}, \mathsf{stmt}, \pi)$
		return b

- Completeness: If $R(\mathsf{stmt}, \mathsf{wit}) = 1$ then $b = 1$
- Knowledge Soundness: If $\mathsf{Verify}(\mathsf{pp}, \mathsf{stmt}, \mathcal{A}(\mathsf{pp}, \mathsf{stmt})) = 1$, then $\mathcal{E}_{\mathcal{A}}(\mathsf{pp}, \mathsf{stmt}) \to \mathsf{wit}$
- Succinctness: $|\pi| = \mathsf{polylog}(|\mathsf{stmt}|)$
- Preprocessing: Extra $\mathsf{pp}_{\mathsf{stmt}} \leftarrow \mathsf{Preproc}(\mathsf{pp}, \mathsf{stmt})$ algorithm so that

$$\mathsf{Time}(\mathsf{Verify}(\mathsf{pp}_{\mathsf{stmt}}, \cdot)) = \mathsf{polylog}(|\mathsf{stmt}|)$$

Succinct Non-Interactive Arguments of Knowledge (SNARK) for $\mathcal{L}$

- NP language $\mathcal{L} = \{\text{stmt} : \exists\, \text{wit},\ R(\text{stmt}, \text{wit}) = 1\}$
- SNARK for $\mathcal{L}$ is a tuple $(\text{Setup}, \text{Prove}, \text{Verify})$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\text{pp} \leftarrow \text{Setup}(1^\lambda)$	
$\pi \leftarrow \text{Prove}(\text{pp}, \text{stmt}, \text{wit})$	π	$\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$
		$b \leftarrow \text{Verify}(\text{pp}, \text{stmt}, \pi)$
		return b

- Completeness: If $R(\text{stmt}, \text{wit}) = 1$ then $b = 1$
- Knowledge Soundness: If $\text{Verify}(\text{pp}, \text{stmt}, \mathcal{A}(\text{pp}, \text{stmt})) = 1$, then $\mathcal{E}_{\mathcal{A}}(\text{pp}, \text{stmt}) \rightarrow \text{wit}$
- Succinctness: $|\pi| = \text{polylog}(|\text{stmt}|)$
- Preprocessing: Extra $\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$ algorithm so that

$$\text{Time}(\text{Verify}(\text{pp}_{\text{stmt}}, \cdot)) = \text{polylog}(|\text{stmt}|)$$

Succinct Non-Interactive Arguments of Knowledge (SNARK) for $\mathcal{L}$

- NP language $\mathcal{L} = \{\text{stmt} : \exists\, \text{wit},\ R(\text{stmt}, \text{wit}) = 1\}$
- SNARK for $\mathcal{L}$ is a tuple $(\text{Setup}, \text{Prove}, \text{Verify})$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\text{pp} \leftarrow \text{Setup}(1^\lambda)$	
$\pi \leftarrow \text{Prove}(\text{pp}, \text{stmt}, \text{wit})$	π	$\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$
		$b \leftarrow \text{Verify}(\text{pp}, \text{stmt}, \pi)$
		return b

- Completeness: If $R(\text{stmt}, \text{wit}) = 1$ then $b = 1$
- Knowledge Soundness: If $\text{Verify}(\text{pp}, \text{stmt}, \mathcal{A}(\text{pp}, \text{stmt})) = 1$, then $\mathcal{E}_{\mathcal{A}}(\text{pp}, \text{stmt}) \rightarrow \text{wit}$
- Succinctness: $|\pi| = \text{polylog}(|\text{stmt}|)$
- Preprocessing: Extra $\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$ algorithm so that

$$\text{Time}(\text{Verify}(\text{pp}_{\text{stmt}}, \cdot)) = \text{polylog}(|\text{stmt}|)$$

Succinct Non-Interactive Arguments of Knowledge (SNARK) for $\mathcal{L}$

- NP language $\mathcal{L} = \{\text{stmt} : \exists\, \text{wit},\ R(\text{stmt}, \text{wit}) = 1\}$
- SNARK for $\mathcal{L}$ is a tuple $(\text{Setup}, \text{Prove}, \text{Verify})$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\text{pp} \leftarrow \text{Setup}(1^\lambda)$	
$\pi \leftarrow \text{Prove}(\text{pp}, \text{stmt}, \text{wit})$	π	$\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$
		$b \leftarrow \text{Verify}(\text{pp}, \text{stmt}, \pi)$
		return b

- Completeness: If $R(\text{stmt}, \text{wit}) = 1$ then $b = 1$
- Knowledge Soundness: If $\text{Verify}(\text{pp}, \text{stmt}, \mathcal{A}(\text{pp}, \text{stmt})) = 1$, then $\mathcal{E}_{\mathcal{A}}(\text{pp}, \text{stmt}) \rightarrow \text{wit}$
- Succinctness: $|\pi| = \text{polylog}(|\text{stmt}|)$
- Preprocessing: Extra $\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$ algorithm so that

$$\text{Time}(\text{Verify}(\text{pp}_{\text{stmt}}, \cdot)) = \text{polylog}(|\text{stmt}|)$$

Succinct Non-Interactive Arguments of Knowledge (SNARK) for $\mathcal{L}$

- NP language $\mathcal{L} = \{\text{stmt} : \exists\, \text{wit},\ R(\text{stmt}, \text{wit}) = 1\}$
- SNARK for $\mathcal{L}$ is a tuple $(\text{Setup}, \text{Prove}, \text{Verify})$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\text{pp} \leftarrow \text{Setup}(1^\lambda)$	
$\pi \leftarrow \text{Prove}(\text{pp}, \text{stmt}, \text{wit})$	π	$\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$
		$b \leftarrow \text{Verify}(\text{pp}, \text{stmt}, \pi)$
		return b

- Completeness: If $R(\text{stmt}, \text{wit}) = 1$ then $b = 1$
- Knowledge Soundness: If $\text{Verify}(\text{pp}, \text{stmt}, \mathcal{A}(\text{pp}, \text{stmt})) = 1$, then $\mathcal{E}_{\mathcal{A}}(\text{pp}, \text{stmt}) \rightarrow \text{wit}$
- Succinctness: $|\pi| = \text{polylog}(|\text{stmt}|)$
- Preprocessing: Extra $\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$ algorithm so that

$$\text{Time}(\text{Verify}(\text{pp}_{\text{stmt}}, \cdot)) = \text{polylog}(|\text{stmt}|)$$

Succinct Non-Interactive Arguments of Knowledge (SNARK) for $\mathcal{L}$

- NP language $\mathcal{L} = \{\text{stmt} : \exists\, \text{wit},\ R(\text{stmt}, \text{wit}) = 1\}$
- SNARK for $\mathcal{L}$ is a tuple $(\text{Setup}, \text{Prove}, \text{Verify})$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\text{pp} \leftarrow \text{Setup}(1^\lambda)$	
$\pi \leftarrow \text{Prove}(\text{pp}, \text{stmt}, \text{wit})$	π	$\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$
		$b \leftarrow \text{Verify}(\text{pp}_{\text{stmt}}, \pi)$
		return b

- Completeness: If $R(\text{stmt}, \text{wit}) = 1$ then $b = 1$
- Knowledge Soundness: If $\text{Verify}(\text{pp}, \text{stmt}, \mathcal{A}(\text{pp}, \text{stmt})) = 1$, then $\mathcal{E}_{\mathcal{A}}(\text{pp}, \text{stmt}) \rightarrow \text{wit}$
- Succinctness: $|\pi| = \text{polylog}(|\text{stmt}|)$
- Preprocessing: Extra $\text{pp}_{\text{stmt}} \leftarrow \text{Preproc}(\text{pp}, \text{stmt})$ algorithm so that

$$\text{Time}(\text{Verify}(\text{pp}_{\text{stmt}}, \cdot)) = \text{polylog}(|\text{stmt}|)$$

Publicly Verifiable SNARKs for NP

Comparison of *publicly verifiable* SNARKs for unstructured languages, e.g.

† Boolean or arithmetic circuit satisfiability (CircuitSAT)

† Rank-1 constraint satisfiability (R1CS)

Scheme	Preprocessing	Algebraic	Post-Quantum
Bulletproof (Group), Lakonia	✘	✘	✘
Sonic, Marlin, SuperSonic, Spartan, Kopis, Xiphos	✓	✘	✘
Groth16, GKMMM18	✓	✓	✘
Ligero, Aurora, Bulletproof (Lattice)	✘	✘	✓
Fractal, Brakedown, Shockwave	✓	✘	✓

Publicly Verifiable + Preprocessing + Algebraic + Structure-Preserving
$\Longrightarrow$ Recursive-Composition-Friendly

Lattice-based SNARKs for NP

Comparison of *lattice-based* SNARKs for unstructured languages, e.g.

- † Boolean or arithmetic circuit satisfiability (CircuitSAT)
- † Rank-1 constraint satisfiability (R1CS)

Scheme	Publicly Verifiable	Preprocessing	Post-Quantum
BCIOP13	✘	✓	✓
Bulletproof (Lattice)	✓	✘	✓

Question

How to construct SNARKs which are

- † Post-quantum secure
- † Publicly verifiable
- † Preprocessing
- † Algebraic
- † Structure-preserving

at the same time?

This Talk

Objective

Lattice-based SNARK:

$$\left.\begin{array}{c}\text{Publicly Verifiable}\\ \text{Preprocessing}\\ \text{Algebraic, Structure-Preserving}\end{array}\right\} \Longrightarrow \text{Recusive-Composition-Friendly}$$

Main (and Only) Ingredient

Lattice-based VC for constant-degree multivariate polynomials, e.g. $f(\mathbf{X}) = X_0X_1X_2 + 2X_1^2X_3 - X_2$

Caution

New (but natural) lattice-based (knowledge) assumptions, with justification

This Talk

Objective

Lattice-based SNARK:

$$\left.\begin{array}{c}\text{Publicly Verifiable}\\ \text{Preprocessing}\\ \text{Algebraic, Structure-Preserving}\end{array}\right\} \Longrightarrow \text{Recusive-Composition-Friendly}$$

Main (and Only) Ingredient

Lattice-based VC for constant-degree multivariate polynomials, e.g. $f(\mathbf{X}) = X_0X_1X_2 + 2X_1^2X_3 - X_2$

Caution

New (but natural) lattice-based (knowledge) assumptions, with justification

This Talk

Objective

Lattice-based SNARK:

$$\left.\begin{array}{c}\text{Publicly Verifiable}\\ \text{Preprocessing}\\ \text{Algebraic, Structure-Preserving}\end{array}\right\} \Longrightarrow \text{Recusive-Composition-Friendly}$$

Main (and Only) Ingredient

Lattice-based VC for constant-degree multivariate polynomials, e.g. $f(\mathbf{X}) = X_0X_1X_2 + 2X_1^2X_3 - X_2$

Caution

New (but natural) lattice-based (knowledge) assumptions, with justification

Agenda

- Succinct Non-Interactive Arguments of Knowledge (SNARK)
- Vector Commitments (VC) with Functional Openings
- Lattice-based VC with Openings to Polynomial Maps
- Further Results

VC with Functional Openings (a.k.a. Functional Commitments [LRY16])

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	com	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	π : $\mathbf{x}$ committed in com satisfies $f(\mathbf{x}) = y$	$\mathsf{pp}_{f,y} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, y))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}, \mathsf{com}, (f, y), \pi)$
		return b

- † Correctness: If $f(\mathbf{x}) = y$, then $b = 1$
- † Weak Binding: Cannot open to (f, y) and (f, y') with $y \neq y'$
- † Binding: Cannot open to inconsistent $(f_0, y_0), (f_1, y_1), (f_2, y_2), \ldots$
- † Extractable: If $\mathcal{A}$ can open to (f, y), $\mathcal{E}_\mathcal{A}$ can extract $\mathbf{x}$
- † Succinctness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|) \cdot |y|$
- † Compactness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|, |y|)$
- † Preprocessing

VC with Functional Openings (a.k.a. Functional Commitments [LRY16])

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	com	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	π : $\mathbf{x}$ committed in com satisfies $f(\mathbf{x}) = y$	$\mathsf{pp}_{f,y} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, y))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}, \mathsf{com}, (f, y), \pi)$
		return b

† Correctness: If $f(\mathbf{x}) = y$, then $b = 1$

† Weak Binding: Cannot open to (f, y) and (f, y') with $y \neq y'$

† Binding: Cannot open to inconsistent $(f_0, y_0), (f_1, y_1), (f_2, y_2), \ldots$

† Extractable: If $\mathcal{A}$ can open to (f, y), $\mathcal{E}_\mathcal{A}$ can extract $\mathbf{x}$

† Succinctness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|) \cdot |y|$

† Compactness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|, |y|)$

† Preprocessing

VC with Functional Openings (a.k.a. Functional Commitments [LRY16])

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	com	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	π : $\mathbf{x}$ committed in com satisfies $f(\mathbf{x}) = y$	$\mathsf{pp}_{f,y} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, y))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}, \mathsf{com}, (f, y), \pi)$
		return b

† Correctness: If $f(\mathbf{x}) = y$, then $b = 1$
† Weak Binding: Cannot open to (f, y) and (f, y') with $y \neq y'$
† Binding: Cannot open to inconsistent $(f_0, y_0), (f_1, y_1), (f_2, y_2), \ldots$
† Extractable: If $\mathcal{A}$ can open to (f, y), $\mathcal{E}_\mathcal{A}$ can extract $\mathbf{x}$
† Succinctness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|) \cdot |y|$
† Compactness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|, |y|)$
† Preprocessing

VC with Functional Openings (a.k.a. Functional Commitments [LRY16])

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	com	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	π : $\mathbf{x}$ committed in com satisfies $f(\mathbf{x}) = y$	$\mathsf{pp}_{f,y} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, y))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}, \mathsf{com}, (f, y), \pi)$
		return b

† Correctness: If $f(\mathbf{x}) = y$, then $b = 1$
† Weak Binding: Cannot open to (f, y) and (f, y') with $y \neq y'$
† Binding: Cannot open to inconsistent $(f_0, y_0), (f_1, y_1), (f_2, y_2), \ldots$
† Extractable: If $\mathcal{A}$ can open to (f, y), $\mathcal{E}_\mathcal{A}$ can extract $\mathbf{x}$
† Succinctness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|) \cdot |y|$
† Compactness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|, |y|)$
† Preprocessing

VC with Functional Openings (a.k.a. Functional Commitments [LRY16])

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	com	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	π : $\mathbf{x}$ committed in com satisfies $f(\mathbf{x}) = y$	$\mathsf{pp}_{f,y} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, y))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}_{f,y}, \mathsf{com}, \pi)$
		return b

† Correctness: If $f(\mathbf{x}) = y$, then $b = 1$
† Weak Binding: Cannot open to (f, y) and (f, y') with $y \neq y'$
† Binding: Cannot open to inconsistent $(f_0, y_0), (f_1, y_1), (f_2, y_2), \ldots$
† Extractable: If $\mathcal{A}$ can open to (f, y), $\mathcal{E}_\mathcal{A}$ can extract $\mathbf{x}$
† Succinctness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|) \cdot |y|$
† Compactness: $|\mathsf{com}|, |\pi| = \mathsf{polylog}(|\mathbf{x}|, |y|)$
† Preprocessing

VC Overview

What do we know about vector commitments (VC)?

- Openings to position functions $f(\mathbf{X}) = X_i$ or linear functions $f(\mathbf{X}) = \sum_i f_i \cdot X_i$
- Constructions over
  - unknown order groups, e.g. $\mathbb{Z}_N^*$, $\mathrm{Cl}(\mathcal{O})$ or pairing-friendly groups $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$
- Post-quantum: Merkle trees and recent work [PPS21] for positional openings VC based on SIS

VC Overview

What do we know about vector commitments (VC)?

- Openings to position functions $f(\mathbf{X}) = X_i$ or linear functions $f(\mathbf{X}) = \sum_i f_i \cdot X_i$
- Constructions over

  unknown order groups, e.g. $\mathbb{Z}_N^*, \mathrm{Cl}(\mathcal{O})$ or pairing-friendly groups $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$

- Post-quantum: Merkle trees and recent work [PPS21] for positional openings VC based on SIS

VC Overview

What do we know about vector commitments (VC)?

- Openings to position functions $f(\mathbf{X}) = X_i$ or linear functions $f(\mathbf{X}) = \sum_i f_i \cdot X_i$
- Constructions over

  unknown order groups, e.g. $\mathbb{Z}_N^*, \mathrm{Cl}(\mathcal{O})$ or pairing-friendly groups $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$

- Post-quantum: Merkle trees and recent work [PPS21] for positional openings VC based on SIS

VC Overview

What do we know about vector commitments (VC)?

- Openings to position functions $f(\mathbf{X}) = X_i$ or linear functions $f(\mathbf{X}) = \sum_i f_i \cdot X_i$
- Constructions over

  unknown order groups, e.g. $\mathbb{Z}_N^*, \mathrm{Cl}(\mathcal{O})$ or pairing-friendly groups $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$

- Post-quantum: Merkle trees and recent work [PPS21] for positional openings VC based on SIS

Question

Vector commitments which:

- are post-quantum secure and compact, or
- support openings to non-linear functions, e.g. quadratic polynomials?

SNARK from VC for Quadratic Functions

- Satisfiability of system of quadratic equations is NP-complete
- stmt $= (f, \mathbf{y})$ where f is a quadratic polynomial map
- wit $= \mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	com, π	$\mathsf{pp}_{f,\mathbf{y}} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, \mathbf{y}))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}_{f,\mathbf{y}}, \mathsf{com}, \pi)$
		return b

- VC Correctness $\implies$ SNARK Completeness
- VC Extractability $\implies$ SNARK Knowledge Soundness
- VC Compactness $\implies$ SNARK Succinctness
- VC Preprocessing $\implies$ SNARK Preprocessing

SNARK from VC for Quadratic Functions

- Satisfiability of system of quadratic equations is NP-complete
- stmt $= (f, \mathbf{y})$ where f is a quadratic polynomial map
- wit $= \mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	com, π	$\mathsf{pp}_{f,\mathbf{y}} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, \mathbf{y}))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}_{f,\mathbf{y}}, \mathsf{com}, \pi)$
		return b

- VC Correctness $\Longrightarrow$ SNARK Completeness
- VC Extractability $\Longrightarrow$ SNARK Knowledge Soundness
- VC Compactness $\Longrightarrow$ SNARK Succinctness
- VC Preprocessing $\Longrightarrow$ SNARK Preprocessing

SNARK from VC for Quadratic Functions

- Satisfiability of system of quadratic equations is NP-complete
- stmt $= (f, \mathbf{y})$ where f is a quadratic polynomial map
- wit $= \mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	com, π	$\mathsf{pp}_{f,\mathbf{y}} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, \mathbf{y}))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}_{f,\mathbf{y}}, \mathsf{com}, \pi)$
		return b

- VC Correctness $\implies$ SNARK Completeness
- VC Extractability $\implies$ SNARK Knowledge Soundness
- VC Compactness $\implies$ SNARK Succinctness
- VC Preprocessing $\implies$ SNARK Preprocessing

SNARK from VC for Quadratic Functions

- Satisfiability of system of quadratic equations is NP-complete
- stmt $= (f, \mathbf{y})$ where f is a quadratic polynomial map
- wit $= \mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	com, π	$\mathsf{pp}_{f,\mathbf{y}} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, \mathbf{y}))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}_{f,\mathbf{y}}, \mathsf{com}, \pi)$
		return b

- VC Correctness $\implies$ SNARK Completeness
- VC Extractability $\implies$ SNARK Knowledge Soundness
- VC Compactness $\implies$ SNARK Succinctness
- VC Preprocessing $\implies$ SNARK Preprocessing

SNARK from VC for Quadratic Functions

- Satisfiability of system of quadratic equations is NP-complete
- stmt $= (f, \mathbf{y})$ where f is a quadratic polynomial map
- wit $= \mathbf{x}$ such that $f(\mathbf{x}) = \mathbf{y}$

Prover $\mathcal{P}$		Verifier $\mathcal{V}$
$\mathsf{com} \leftarrow \mathsf{VC.Com}(\mathsf{pp}, \mathbf{x})$	$\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda)$	
$\pi \leftarrow \mathsf{VC.Open}(\mathsf{pp}, f, \mathbf{x})$	com, π	$\mathsf{pp}_{f,\mathbf{y}} \leftarrow \mathsf{VC.Preproc}(\mathsf{pp}, (f, \mathbf{y}))$
		$b \leftarrow \mathsf{VC.Verify}(\mathsf{pp}_{f,\mathbf{y}}, \mathsf{com}, \pi)$
		return b

- VC Correctness $\implies$ SNARK Completeness
- VC Extractability $\implies$ SNARK Knowledge Soundness
- VC Compactness $\implies$ SNARK Succinctness
- VC Preprocessing $\implies$ SNARK Preprocessing

Agenda

- Succinct Non-Interactive Arguments of Knowledge (SNARK)
- Vector Commitments (VC) with Functional Openings
- Lattice-based VC with Openings to Polynomial Maps
- Further Results

Roadmap

1. Translating pairing-based VC for linear functions
   $\Longrightarrow$ Lattice-based VC for linear functions from new k-R-ISIS assumption
2. Exploiting ring structure in lattices $\Longrightarrow$ VC with polynomial openings
3. New k-R-ISIS of knowledge assumption $\Longrightarrow$ Extractability
4. New proof aggregation trick $\Longrightarrow$ Compactness

Step	Functions	Security	Efficiency
1	Linear	Weak Binding	Succinctness
2	Polynomial	Weak Binding	Succinctness
3	Polynomial	Extractability	Succinctness
4	Polynomial	Extractability	Compactness

Roadmap

1. Translating pairing-based VC for linear functions
   $\Longrightarrow$ Lattice-based VC for linear functions from new k-R-ISIS assumption
2. Exploiting ring structure in lattices $\Longrightarrow$ VC with polynomial openings
3. New k-R-ISIS of knowledge assumption $\Longrightarrow$ Extractability
4. New proof aggregation trick $\Longrightarrow$ Compactness

Step	Functions	Security	Efficiency
1	Linear	Weak Binding	Succinctness
2	Polynomial	Weak Binding	Succinctness
3	Polynomial	Extractability	Succinctness
4	Polynomial	Extractability	Compactness

Pairing and Lattice Settings

Pairing Setting:

- Cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ of prime order q
- Generator $[1]_i$ for $\mathbb{G}_i$
- Group operation written additively:

$$[a]_i + [b]_i = [a + b]_i$$

- Pairing operation written multiplicatively:

$$[a]_1 \cdot [b]_2 = [a \cdot b]_t$$

- Extension to vector operations: e.g.

$$\langle [\mathbf{a}]_1, [\mathbf{b}]_2 \rangle = [\langle \mathbf{a}, \mathbf{b} \rangle]_t$$

Lattice Setting:

- Ring $\mathcal{R}$ equipped with norm $\|\cdot\| : \mathcal{R} \to \mathbb{R}$
- Prime modulus q
- Quotient ring $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$
- Random public wide matrix $\mathbf{A} \leftarrow\!\$\ \mathcal{R}_q^{\eta \times \ell}$
- Random target vector $\mathbf{t} \leftarrow\!\$\ \mathcal{R}_q^{\eta}$

Pairing and Lattice Settings

Pairing Setting:

- Cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ of prime order q
- Generator $[1]_i$ for $\mathbb{G}_i$
- Group operation written additively:

$$[a]_i + [b]_i = [a + b]_i$$

- Pairing operation written multiplicatively:

$$[a]_1 \cdot [b]_2 = [a \cdot b]_t$$

- Extension to vector operations: e.g.

$$\langle [\mathbf{a}]_1, [\mathbf{b}]_2 \rangle = [\langle \mathbf{a}, \mathbf{b} \rangle]_t$$

Lattice Setting:

- Ring $\mathcal{R}$ equipped with norm $\|\cdot\| : \mathcal{R} \to \mathbb{R}$
- Prime modulus q
- Quotient ring $\mathcal{R}_q := \mathcal{R}/q\mathcal{R}$
- Random public wide matrix $\mathbf{A} \leftarrow\!\!\$\ \mathcal{R}_q^{\eta \times \ell}$
- Random target vector $\mathbf{t} \leftarrow\!\!\$\ \mathcal{R}_q^{\eta}$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \mathrm{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \mathrm{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

General Construction Idea for Linear Functions (Implicit in Literature)

- † Fix random vector/point $\mathbf{v} = (v_i)_{i \in \mathbb{Z}_w}$, where $|\mathbf{v}| = |\mathbf{x}|$
- † Treat $\mathbf{v}$ as variables and define (Laurent) monomials/polynomials over them:
- † Define "target monomial", e.g. $\bar{v} = \prod_{i \in \mathbb{Z}_w} v_i$
- † Define "complement monomials" $\bar{v}_i$ such that $\bar{v}_i \cdot v_i = \bar{v}$
- † Encode $\mathbf{x} = (x_i)_{i \in \mathbb{Z}_w}$ as polynomial $p_{\mathbf{x}}(\mathbf{v})$
- † Encode $f(\mathbf{X}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot X_i$ as polynomial $p_f(\mathbf{v})$
- † Goal of encoding: $p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\ldots}_{\text{cross terms} \notin \text{Span}(\bar{v})}$

Example:

$$p_f(\mathbf{v}) = \sum_{i \in \mathbb{Z}_w} f_i \cdot \bar{v}_i \qquad p_{\mathbf{x}}(\mathbf{v}) = \sum_{j \in \mathbb{Z}_w} x_j \cdot v_j$$

$$p_f(\mathbf{v}) \cdot p_{\mathbf{x}}(\mathbf{v}) = f(\mathbf{x}) \cdot \bar{v} + \underbrace{\sum_{i \neq j} f_i \cdot x_j \cdot \bar{v}_i \cdot v_j}_{\text{cross terms} \notin \text{Span}(\bar{v})}$$

Pairing-based VC for Linear Functions

Construction:

- † Public parameters: $[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$
- † Commit $\mathbf{x}$: $\mathsf{com} = [c]_1 = [p_{\mathbf{x}}(\mathbf{v})]_1 = \sum_{j\in\mathbb{Z}_w} x_j \cdot [v_j]_1$
- † Open f:
  - ‡ Compute $[u]_2 = \sum_{i\neq j} f_i \cdot x_j \cdot [\bar{v}_i \cdot v_j]_2$
  - ‡ Output $\pi = [u]_2$
- † Verify $(\mathsf{com}, (f, y), \pi)$:
  - ‡ Compute $[p_f(\mathbf{v})]_2 = \sum_{i\in\mathbb{Z}_w} f_i \cdot [\bar{v}_i]_2$
  - ‡ Compute $[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t$
  - ‡ (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathsf{Span}(\bar{v})$)
  - ‡ Parse π as $[u]_2$
  - ‡ Check if $[1]_1 \cdot [u]_2 = [\Delta]_t$

Pairing-based VC for Linear Functions

Construction:

- Public parameters: $[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$
- Commit $\mathbf{x}$: $\mathsf{com} = [c]_1 = [p_{\mathbf{x}}(\mathbf{v})]_1 = \sum_{j\in\mathbb{Z}_w} x_j \cdot [v_j]_1$
- Open f:
  - Compute $[u]_2 = \sum_{i\neq j} f_i \cdot x_j \cdot [\bar{v}_i \cdot v_j]_2$
  - Output $\pi = [u]_2$
- Verify $(\mathsf{com}, (f, y), \pi)$:
  - Compute $[p_f(\mathbf{v})]_2 = \sum_{i\in\mathbb{Z}_w} f_i \cdot [\bar{v}_i]_2$
  - Compute $[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathsf{Span}(\bar{v})$)
  - Parse π as $[u]_2$
  - Check if $[1]_1 \cdot [u]_2 = [\Delta]_t$

Pairing-based VC for Linear Functions

Construction:

- † Public parameters: $[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$
- † Commit $\mathbf{x}$: $\mathsf{com} = [c]_1 = [p_{\mathbf{x}}(\mathbf{v})]_1 = \sum_{j\in\mathbb{Z}_w} x_j \cdot [v_j]_1$
- † Open f:
  - ‡ Compute $[u]_2 = \sum_{i\neq j} f_i \cdot x_j \cdot [\bar{v}_i \cdot v_j]_2$
  - ‡ Output $\pi = [u]_2$
- † Verify $(\mathsf{com}, (f, y), \pi)$:
  - ‡ Compute $[p_f(\mathbf{v})]_2 = \sum_{i\in\mathbb{Z}_w} f_i \cdot [\bar{v}_i]_2$
  - ‡ Compute $[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t$
  - ‡ (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathsf{Span}(\bar{v})$)
  - ‡ Parse π as $[u]_2$
  - ‡ Check if $[1]_1 \cdot [u]_2 = [\Delta]_t$

Pairing-based VC for Linear Functions

Construction:

- Public parameters: $[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$
- Commit $\mathbf{x}$: $\mathsf{com} = [c]_1 = [p_{\mathbf{x}}(\mathbf{v})]_1 = \sum_{j\in\mathbb{Z}_w} x_j \cdot [v_j]_1$
- Open f:
  - Compute $[u]_2 = \sum_{i\neq j} f_i \cdot x_j \cdot [\bar{v}_i \cdot v_j]_2$
  - Output $\pi = [u]_2$
- Verify $(\mathsf{com}, (f, y), \pi)$:
  - Compute $[p_f(\mathbf{v})]_2 = \sum_{i\in\mathbb{Z}_w} f_i \cdot [\bar{v}_i]_2$
  - Compute $[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathsf{Span}(\bar{v})$)
  - Parse π as $[u]_2$
  - Check if $[1]_1 \cdot [u]_2 = [\Delta]_t$

Pairing-based VC for Linear Functions

Construction:

- Public parameters: $[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$
- Commit $\mathbf{x}$: $\mathrm{com} = [c]_1 = [p_{\mathbf{x}}(\mathbf{v})]_1 = \sum_{j\in\mathbb{Z}_w} x_j \cdot [v_j]_1$
- Open f:
  - Compute $[u]_2 = \sum_{i\neq j} f_i \cdot x_j \cdot [\bar{v}_i \cdot v_j]_2$
  - Output $\pi = [u]_2$
- Verify $(\mathrm{com}, (f, y), \pi)$:
  - Compute $[p_f(\mathbf{v})]_2 = \sum_{i\in\mathbb{Z}_w} f_i \cdot [\bar{v}_i]_2$
  - Compute $[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathrm{Span}(\bar{v})$)
  - Parse π as $[u]_2$
  - Check if $[1]_1 \cdot [u]_2 = [\Delta]_t$

Pairing-based VC for Linear Functions

Construction:

- Public parameters: $[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$
- Commit $\mathbf{x}$: $\mathsf{com} = [c]_1 = [p_{\mathbf{x}}(\mathbf{v})]_1 = \sum_{j\in\mathbb{Z}_w} x_j \cdot [v_j]_1$
- Open f:
  - Compute $[u]_2 = \sum_{i\neq j} f_i \cdot x_j \cdot [\bar{v}_i \cdot v_j]_2$
  - Output $\pi = [u]_2$
- Verify $(\mathsf{com}, (f, y), \pi)$:
  - Compute $[p_f(\mathbf{v})]_2 = \sum_{i\in\mathbb{Z}_w} f_i \cdot [\bar{v}_i]_2$
  - Compute $[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathsf{Span}(\bar{v})$)
  - Parse π as $[u]_2$
  - Check if $[1]_1 \cdot [u]_2 = [\Delta]_t$

Translating Public Parameters

Pairing-based Scheme Public Parameters

$$[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$$

1. $[1]_1, [1]_2 \mapsto \mathbf{A}, \mathbf{t}$
2. Make $(v_j)_{j\in\mathbb{Z}_w}$ (and hence $\bar{\mathbf{v}} = (\bar{v}_i)_{j\in\mathbb{Z}_w}$ and $\bar{v} = \prod_{i\in\mathbb{Z}_w} v_i$) public
3. $([\bar{v}_i \cdot v_j]_2)_{i\neq j} \mapsto$ Short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$

Lattice-based Scheme Public Parameters

$$\mathbf{A}, \mathbf{t}, \bar{v}, (v_j)_{j\in\mathbb{Z}_w}, (\bar{v}_i)_{i\in\mathbb{Z}_w}, \text{short } \mathbf{u}_{i,j} \text{ such that } \mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$$

Translating Public Parameters

Pairing-based Scheme Public Parameters

$$[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$$

1. $[1]_1, [1]_2 \mapsto \mathbf{A}, \mathbf{t}$
2. Make $(v_j)_{j\in\mathbb{Z}_w}$ (and hence $\bar{\mathbf{v}} = (\bar{v}_i)_{i\in\mathbb{Z}_w}$ and $\bar{v} = \prod_{i\in\mathbb{Z}_w} v_i$) public
3. $([\bar{v}_i \cdot v_j]_2)_{i\neq j} \mapsto$ Short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$

Lattice-based Scheme Public Parameters

$$\mathbf{A}, \mathbf{t}, \bar{v}, (v_j)_{j\in\mathbb{Z}_w}, (\bar{v}_i)_{i\in\mathbb{Z}_w}, \text{short } \mathbf{u}_{i,j} \text{ such that } \mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$$

Translating Public Parameters

Pairing-based Scheme Public Parameters

$$[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$$

1. $[1]_1, [1]_2 \mapsto \mathbf{A}, \mathbf{t}$
2. Make $(v_j)_{j\in\mathbb{Z}_w}$ (and hence $\bar{\mathbf{v}} = (\bar{v}_i)_{i\in\mathbb{Z}_w}$ and $\bar{v} = \prod_{i\in\mathbb{Z}_w} v_i$) public
3. $([\bar{v}_i \cdot v_j]_2)_{i\neq j} \mapsto$ Short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$

Lattice-based Scheme Public Parameters

$$\mathbf{A}, \mathbf{t}, \bar{v}, (v_j)_{j\in\mathbb{Z}_w}, (\bar{v}_i)_{i\in\mathbb{Z}_w}, \text{short } \mathbf{u}_{i,j} \text{ such that } \mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$$

Translating Public Parameters

Pairing-based Scheme Public Parameters

$$[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$$

1. $[1]_1, [1]_2 \mapsto \mathbf{A}, \mathbf{t}$
2. Make $(v_j)_{j\in\mathbb{Z}_w}$ (and hence $\bar{\mathbf{v}} = (\bar{v}_i)_{i\in\mathbb{Z}_w}$ and $\bar{v} = \prod_{i\in\mathbb{Z}_w} v_i$) public
3. $([\bar{v}_i \cdot v_j]_2)_{i\neq j} \mapsto$ Short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$

Lattice-based Scheme Public Parameters

$$\mathbf{A}, \mathbf{t}, \bar{v}, (v_j)_{j\in\mathbb{Z}_w}, (\bar{v}_i)_{i\in\mathbb{Z}_w}, \text{short } \mathbf{u}_{i,j} \text{ such that } \mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$$

Translating Public Parameters

Pairing-based Scheme Public Parameters

$$[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$$

1. $[1]_1, [1]_2 \mapsto \mathbf{A}, \mathbf{t}$
2. Make $(v_j)_{j\in\mathbb{Z}_w}$ (and hence $\bar{\mathbf{v}} = (\bar{v}_i)_{i\in\mathbb{Z}_w}$ and $\bar{v} = \prod_{i\in\mathbb{Z}_w} v_i$) public
3. $([\bar{v}_i \cdot v_j]_2)_{i\neq j} \mapsto$ Short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$

Lattice-based Scheme Public Parameters

$$\mathbf{A}, \mathbf{t}, \bar{v}, (v_j)_{j\in\mathbb{Z}_w}, (\bar{v}_i)_{i\in\mathbb{Z}_w}, \text{short } \mathbf{u}_{i,j} \text{ such that } \mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$$

Translating Public Parameters

Pairing-based Scheme Public Parameters

$$[1]_1, [1]_2, [\bar{v}]_t, ([v_j]_1)_{j\in\mathbb{Z}_w}, ([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}, ([\bar{v}_i \cdot v_j]_2)_{i\neq j}$$

1. $[1]_1, [1]_2 \mapsto \mathbf{A}, \mathbf{t}$
2. Make $(v_j)_{j\in\mathbb{Z}_w}$ (and hence $\bar{\mathbf{v}} = (\bar{v}_i)_{i\in\mathbb{Z}_w}$ and $\bar{v} = \prod_{i\in\mathbb{Z}_w} v_i$) public
3. $([\bar{v}_i \cdot v_j]_2)_{i\neq j} \mapsto$ Short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$

Lattice-based Scheme Public Parameters

$$\mathbf{A}, \mathbf{t}, \bar{v}, (v_j)_{j\in\mathbb{Z}_w}, (\bar{v}_i)_{i\in\mathbb{Z}_w}, \text{short } \mathbf{u}_{i,j} \text{ such that } \mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$$

Translating Scheme

Construction:

- Public parameters: $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $(\mathbf{u}_{i,j})_{i\neq j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$
- Commit $\mathbf{x}$: $\mathsf{com} = c = p_{\mathbf{x}}(\mathbf{v}) \bmod q = \sum_{j\in\mathbb{Z}_w} x_j \cdot v_j \bmod q$
- Open f:
  - Compute $\mathbf{u} = \sum_{i\neq j} f_i \cdot x_j \cdot \mathbf{u}_{i,j}$
  - Output $\pi = \mathbf{u}$
- Verify $(\mathsf{com}, (f, y), \pi)$:
  - Compute $p_f(\mathbf{v}) = \sum_{i\in\mathbb{Z}_w} f_i \cdot \bar{v}_i \bmod q$
  - Compute $\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} \bmod q$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \in \mathrm{Span}(\bar{v})$)
  - Parse π as $\mathbf{u}$
  - Check if $\mathbf{A}\cdot\mathbf{u} = \Delta \cdot \mathbf{t} \bmod q$ and $\mathbf{u}$ is short

Translating Scheme

Construction:

- Public parameters: $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $(\mathbf{u}_{i,j})_{i\neq j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$
- Commit $\mathbf{x}$: $\mathrm{com} = c = p_{\mathbf{x}}(\mathbf{v}) \bmod q = \sum_{j\in\mathbb{Z}_w} x_j \cdot v_j \bmod q$
- Open f:
  - Compute $\mathbf{u} = \sum_{i\neq j} f_i \cdot x_j \cdot \mathbf{u}_{i,j}$
  - Output $\pi = \mathbf{u}$
- Verify $(\mathrm{com}, (f, y), \pi)$:
  - Compute $p_f(\mathbf{v}) = \sum_{i\in\mathbb{Z}_w} f_i \cdot \bar{v}_i \bmod q$
  - Compute $\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} \bmod q$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \in \mathrm{Span}(\bar{v})$)
  - Parse π as $\mathbf{u}$
  - Check if $\mathbf{A}\cdot\mathbf{u} = \Delta \cdot \mathbf{t} \bmod q$ and $\mathbf{u}$ is short

Translating Scheme

Construction:

- † Public parameters: $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $(\mathbf{u}_{i,j})_{i\neq j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$
- † Commit $\mathbf{x}$: $\mathrm{com} = c = p_{\mathbf{x}}(\mathbf{v}) \bmod q = \sum_{j\in\mathbb{Z}_w} x_j \cdot v_j \bmod q$
- † Open f:
  - ‡ Compute $\mathbf{u} = \sum_{i\neq j} f_i \cdot x_j \cdot \mathbf{u}_{i,j}$
  - ‡ Output $\pi = \mathbf{u}$
- † Verify $(\mathrm{com}, (f, y), \pi)$:
  - ‡ Compute $p_f(\mathbf{v}) = \sum_{i\in\mathbb{Z}_w} f_i \cdot \bar{v}_i \bmod q$
  - ‡ Compute $\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} \bmod q$
  - ‡ (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathrm{Span}(\bar{v})$)
  - ‡ Parse π as $\mathbf{u}$
  - ‡ Check if $\mathbf{A}\cdot\mathbf{u} = \Delta \cdot \mathbf{t} \bmod q$ and $\mathbf{u}$ is short

Translating Scheme

Construction:

- Public parameters: $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $(\mathbf{u}_{i,j})_{i\neq j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$
- Commit $\mathbf{x}$: $\mathrm{com} = c = p_{\mathbf{x}}(\mathbf{v}) \bmod q = \sum_{j\in\mathbb{Z}_w} x_j \cdot v_j \bmod q$
- Open f:
  - Compute $\mathbf{u} = \sum_{i\neq j} f_i \cdot x_j \cdot \mathbf{u}_{i,j}$
  - Output $\pi = \mathbf{u}$
- Verify $(\mathrm{com}, (f, y), \pi)$:
  - Compute $p_f(\mathbf{v}) = \sum_{i\in\mathbb{Z}_w} f_i \cdot \bar{v}_i \bmod q$
  - Compute $\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} \bmod q$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathrm{Span}(\bar{v})$)
  - Parse π as $\mathbf{u}$
  - Check if $\mathbf{A}\cdot\mathbf{u} = \Delta \cdot \mathbf{t} \bmod q$ and $\mathbf{u}$ is short

Translating Scheme

Construction:

- Public parameters: $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $(\mathbf{u}_{i,j})_{i\neq j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$
- Commit $\mathbf{x}$: $\mathrm{com} = c = p_{\mathbf{x}}(\mathbf{v}) \bmod q = \sum_{j\in\mathbb{Z}_w} x_j \cdot v_j \bmod q$
- Open f:
  - Compute $\mathbf{u} = \sum_{i\neq j} f_i \cdot x_j \cdot \mathbf{u}_{i,j}$
  - Output $\pi = \mathbf{u}$
- Verify $(\mathrm{com}, (f, y), \pi)$:
  - Compute $p_f(\mathbf{v}) = \sum_{i\in\mathbb{Z}_w} f_i \cdot \bar{v}_i \bmod q$
  - Compute $\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} \bmod q$
  - (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathrm{Span}(\bar{v})$)
  - Parse π as $\mathbf{u}$
  - Check if $\mathbf{A}\cdot\mathbf{u} = \Delta \cdot \mathbf{t} \bmod q$ and $\mathbf{u}$ is short

Translating Scheme

Construction:

- † Public parameters: $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $(\mathbf{u}_{i,j})_{i\neq j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$
- † Commit $\mathbf{x}$: $\mathsf{com} = c = p_{\mathbf{x}}(\mathbf{v}) \bmod q = \sum_{j\in\mathbb{Z}_w} x_j \cdot v_j \bmod q$
- † Open f:
  - ‡ Compute $\mathbf{u} = \sum_{i\neq j} f_i \cdot x_j \cdot \mathbf{u}_{i,j}$
  - ‡ Output $\pi = \mathbf{u}$
- † Verify $(\mathsf{com}, (f, y), \pi)$:
  - ‡ Compute $p_f(\mathbf{v}) = \sum_{i\in\mathbb{Z}_w} f_i \cdot \bar{v}_i \bmod q$
  - ‡ Compute $\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} \bmod q$
  - ‡ (Note: If $f(\mathbf{x}) = y$, then $\Delta \notin \mathrm{Span}(\bar{v})$)
  - ‡ Parse π as $\mathbf{u}$
  - ‡ Check if $\mathbf{A}\cdot\mathbf{u} = \Delta \cdot \mathbf{t} \bmod q$ and $\mathbf{u}$ is short

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j \in \mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i \in \mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i \neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j \in \mathbb{Z}_w}$, $(\bar{v}_i)_{i \in \mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A} \cdot \mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A} \cdot \mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- In English: Finding short preimage of small multiple of $\bar{v} \cdot \mathbf{t}$ is hard.
- Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- Why Plausible?
  1. Without hints $\mathbf{u}_{i,j}$ = R-SIS
  2. Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  3. $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i \neq j}) \implies$ Hints don't seem helpful

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j\in\mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i\neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A}\cdot\mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- † In English: Finding short preimage of small multiple of $\bar{v}\cdot\mathbf{t}$ is hard.
- † Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- † Why Plausible?
  - Without hints $\mathbf{u}_{i,j}$ = R-SIS
  - Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  - $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i\neq j}) \implies$ Hints don't seem helpful

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j\in\mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i\neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A}\cdot\mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- In English: Finding short preimage of small multiple of $\bar{v}\cdot\mathbf{t}$ is hard.
- Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- Why Plausible?
  - Without hints $\mathbf{u}_{i,j}$ = R-SIS
  - Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  - $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i\neq j}) \implies$ Hints don't seem helpful

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j\in\mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i\neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A}\cdot\mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- † In English: Finding short preimage of small multiple of $\bar{v}\cdot\mathbf{t}$ is hard.
- † Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- † Why Plausible?
  - ‡ Without hints $\mathbf{u}_{i,j}$ = R-SIS
  - ‡ Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  - ‡ $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i\neq j}) \implies$ Hints don't seem helpful

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j\in\mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i\neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A}\cdot\mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- In English: Finding short preimage of small multiple of $\bar{v}\cdot\mathbf{t}$ is hard.
- Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- Why Plausible?
  - Without hints $\mathbf{u}_{i,j}$ = R-SIS
  - Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  - $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i\neq j}) \implies$ Hints don't seem helpful

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j\in\mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i\neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A}\cdot\mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- In English: Finding short preimage of small multiple of $\bar{v}\cdot\mathbf{t}$ is hard.
- Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- Why Plausible?
  - Without hints $\mathbf{u}_{i,j}$ = R-SIS
  - Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  - $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i\neq j}) \implies$ Hints don't seem helpful

Translating Assumption

Pairing-based Assumption for Weak Binding

Given $[1]_1$, $[1]_2$, $[\bar{v}]_t$, $([v_j]_1)_{j\in\mathbb{Z}_w}$, $([\bar{v}_i]_2)_{i\in\mathbb{Z}_w}$, $([\bar{v}_i \cdot v_j]_2)_{i\neq j}$, finding $[\bar{v}]_2$ is hard.

Lattice-based Assumption for Weak Binding

Given $\mathbf{A}$, $\mathbf{t}$, $\bar{v}$, $(v_j)_{j\in\mathbb{Z}_w}$, $(\bar{v}_i)_{i\in\mathbb{Z}_w}$, short $\mathbf{u}_{i,j}$ such that $\mathbf{A}\cdot\mathbf{u}_{i,j} = \bar{v}_i \cdot v_j \cdot \mathbf{t} \bmod q$, finding short $(\mathbf{u}, s)$ such that $\mathbf{A}\cdot\mathbf{u} = s \cdot \bar{v} \cdot \mathbf{t} \bmod q$ is hard.

- In English: Finding short preimage of small multiple of $\bar{v}\cdot\mathbf{t}$ is hard.
- Member of new k-R-ISIS (k Ring Inhomogeneous Short Integer Solution) assumption family
- Why Plausible?
  - Without hints $\mathbf{u}_{i,j}$ = R-SIS
  - Seemingly only way to use hints $\mathbf{u}_{i,j}$: Short linear combination
  - $\bar{v} \notin \mathrm{Span}(\{\bar{v}_i \cdot v_j\}_{i\neq j}) \implies$ Hints don't seem helpful

Roadmap

1. Translating pairing-based VC for linear functions
   $\Longrightarrow$ Lattice-based VC for linear functions from new k-R-ISIS assumption
2. Exploiting ring structure in lattices $\Longrightarrow$ VC with polynomial openings
3. New k-R-ISIS of knowledge assumption $\Longrightarrow$ Extractability
4. New proof aggregation trick $\Longrightarrow$ Compactness

Step	Functions	Security	Efficiency
1	Linear	Weak Binding	Succinctness
2	Polynomial	Weak Binding	Succinctness
3	Polynomial	Extractability	Succinctness
4	Polynomial	Extractability	Compactness

From Linear Functions to Polynomials

Pairing-based Scheme

$$[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t = f(\underbrace{[\bar{v}_0]_2 \cdot [c]_1, \ldots, [\bar{v}_{w-1}]_2 \cdot [c]_1}_{\mathbb{G}_t \text{ elements}}) - y \cdot [\bar{v}]_t$$

Only $+$ allowed over $\mathbb{G}_t \implies f$ has to be linear

Lattice-based Scheme

$$\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} = f(\underbrace{\bar{v}_0 \cdot c, \ldots, \bar{v}_{w-1} \cdot c}_{\mathcal{R}_q \text{ elements}}) - y \cdot \bar{v} \bmod q$$

Both $+$ and $\times$ allowed over $\mathcal{R}_q \implies f$ can be non-linear

From Linear Functions to Polynomials

Pairing-based Scheme

$$[\Delta]_t = [p_f(\mathbf{v})]_2 \cdot \underbrace{[p_{\mathbf{x}}(\mathbf{v})]_1}_{[c]_1} - y \cdot [\bar{v}]_t = f(\underbrace{[\bar{v}_0]_2 \cdot [c]_1, \ldots, [\bar{v}_{w-1}]_2 \cdot [c]_1}_{\mathbb{G}_t \text{ elements}}) - y \cdot [\bar{v}]_t$$

Only $+$ allowed over $\mathbb{G}_t \implies f$ has to be linear

Lattice-based Scheme

$$\Delta = p_f(\mathbf{v}) \cdot \underbrace{p_{\mathbf{x}}(\mathbf{v})}_{c} - y \cdot \bar{v} = f(\underbrace{\bar{v}_0 \cdot c, \ldots, \bar{v}_{w-1} \cdot c}_{\mathcal{R}_q \text{ elements}}) - y \cdot \bar{v} \bmod q$$

Both $+$ and $\times$ allowed over $\mathcal{R}_q \implies f$ can be non-linear

Roadmap

1. Translating pairing-based VC for linear functions
   $\Longrightarrow$ Lattice-based VC for linear functions from new k-R-ISIS assumption
2. Exploiting ring structure in lattices $\Longrightarrow$ VC with polynomial openings
3. New k-R-ISIS of knowledge assumption $\Longrightarrow$ Extractability
4. New proof aggregation trick $\Longrightarrow$ Compactness

Step	Functions	Security	Efficiency
1	Linear	Weak Binding	Succinctness
2	Polynomial	Weak Binding	Succinctness
3	Polynomial	Extractability	Succinctness
4	Polynomial	Extractability	Compactness

Roadmap

1. Translating pairing-based VC for linear functions
   $\Longrightarrow$ Lattice-based VC for linear functions from new k-R-ISIS assumption
2. Exploiting ring structure in lattices $\Longrightarrow$ VC with polynomial openings
3. New k-R-ISIS of knowledge assumption $\Longrightarrow$ Extractability
4. New proof aggregation trick $\Longrightarrow$ Compactness

Step	Functions	Security	Efficiency
1	Linear	Weak Binding	Succinctness
2	Polynomial	Weak Binding	Succinctness
3	Polynomial	Extractability	Succinctness
4	Polynomial	Extractability	Compactness

Agenda

- † Succinct Non-Interactive Arguments of Knowledge (SNARK)
- † Vector Commitments (VC) with Functional Openings
- † Lattice-based VC with Openings to Polynomial Maps
- † Further Results

Further Results

Reductions.

- There exist hard instances of k-R-ISIS and k-M-ISIS,
- Relations between k-M-ISIS problems for different choices of parameters.

Applications. Languages natively supported by our SNARK construction:

- Aggregating GPV (Adaptor) Signatures
- Recursive SNARK Composition

Further Results

Reductions.

- There exist hard instances of k-R-ISIS and k-M-ISIS,
- Relations between k-M-ISIS problems for different choices of parameters.

Applications. Languages natively supported by our SNARK construction:

- Aggregating GPV (Adaptor) Signatures
- Recursive SNARK Composition

Thank you for your attention!

Questions?

(full version of the paper available on ePrint 2022/941)

Valerio Cini
AIT Austrian Institute of Technology, Austria
@cini_valerio

The template was a gift from Ivy K. Y. Woo to Russell W. F. Lai.