Lattice-Based SNARGs and Their Application to More Efficient Obfuscation

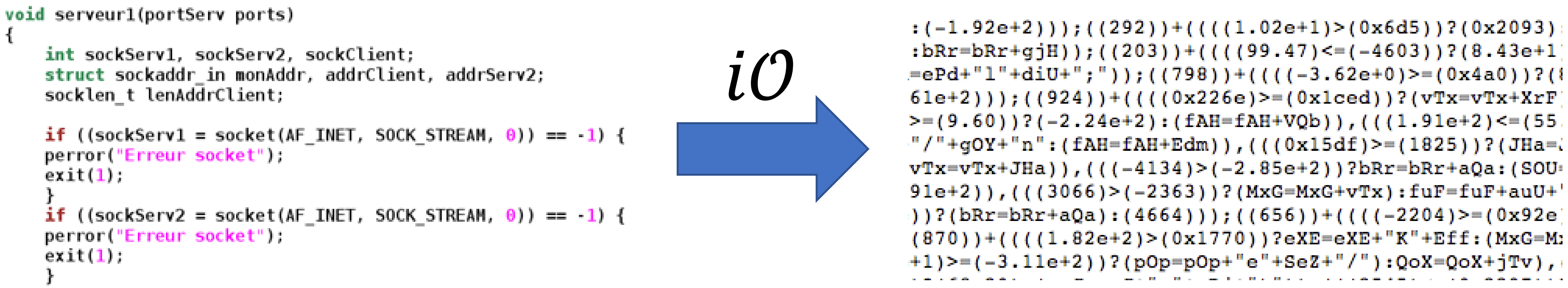
Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu

Program Obfuscation [BGIRSVY01, GGHRSW13]

Indistinguishability obfuscation ($i\mathcal{O}$) has emerged as a “central hub for cryptography” [BGIRSVY01, GGHRSW13]

[GGHRSW13, SW14, BZ14, BST14, GGHR14, GHRW14, BP15, CHNVW15, CLTV15, GP15, GPS16, BPW16 …]

Takes a program as input and “scrambles” it



Program Obfuscation [BGIRSVY01, GGHRSW13]

Indistinguishability obfuscation ($i\mathcal{O}$) has emerged as a “central hub for cryptography” [BGIRSVY01, GGHRSW13]

[GGHRSW13, SW14, BZ14, BST14, GGHR14, GHRW14, BP15, CHNVW15, CLTV15, GP15, GPS16, BPW16 …]

Many applications, yet extremely far from practical

The “Alien” Challenge: If we had to iO-obfuscate AES to save the planet from alien annihilation, can we do it?

Program Obfuscation [BGIRSVY01, GGHRSW13]

Indistinguishability obfuscation ($i\mathcal{O}$) has emerged as a "central hub for cryptography" [BGIRSVY01, GGHRSW13]

[GGHRSW13, SW14, BZ14, BST14, GGHR14, GHRW14, BP15, CHNVW15, CLTV15, GP15, GPS16, BPW16 …]

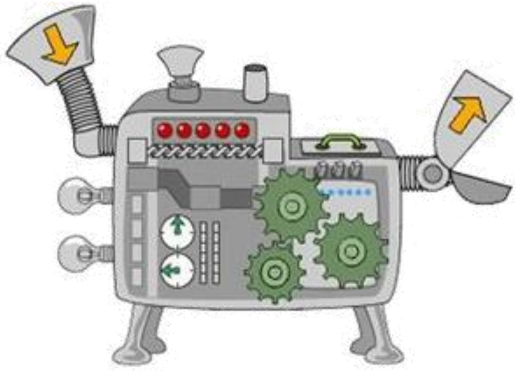
Many applications, yet extremely far from practical

Not just engineering challenges – fundamental theoretical challenges

Polynomial-time, but constant factors are $\geq 2^{100}$

Our Goal

Obtain an "obfuscation-complete" primitive with an emphasis on <u>concrete efficiency</u>



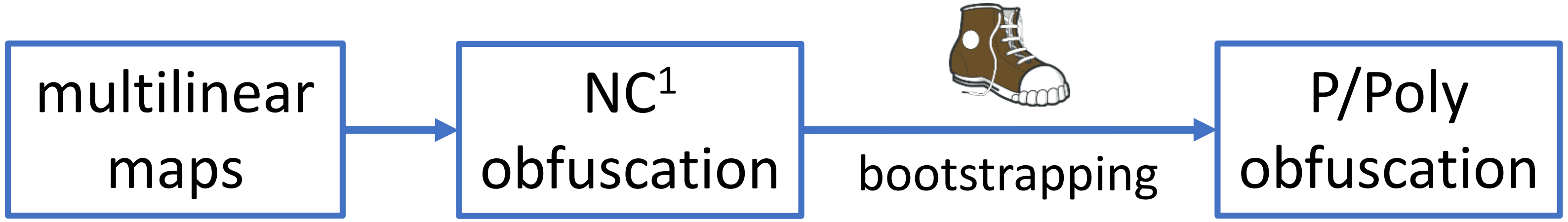
- Functionality whose (ideal) obfuscation can be used to obfuscate arbitrary circuits
- Obfuscated primitive should need to invoked <u>once</u> for function evaluation
- **Our setting:** obfuscate <u>FHE decryption</u> and <u>SNARG verification</u>

Concurrently: improve the asymptotic efficiency of SNARGs

How (Im)Practical is Obfuscation?

Existing constructions rely on multilinear maps [BS04, GGH13, CLT13, GGH15]

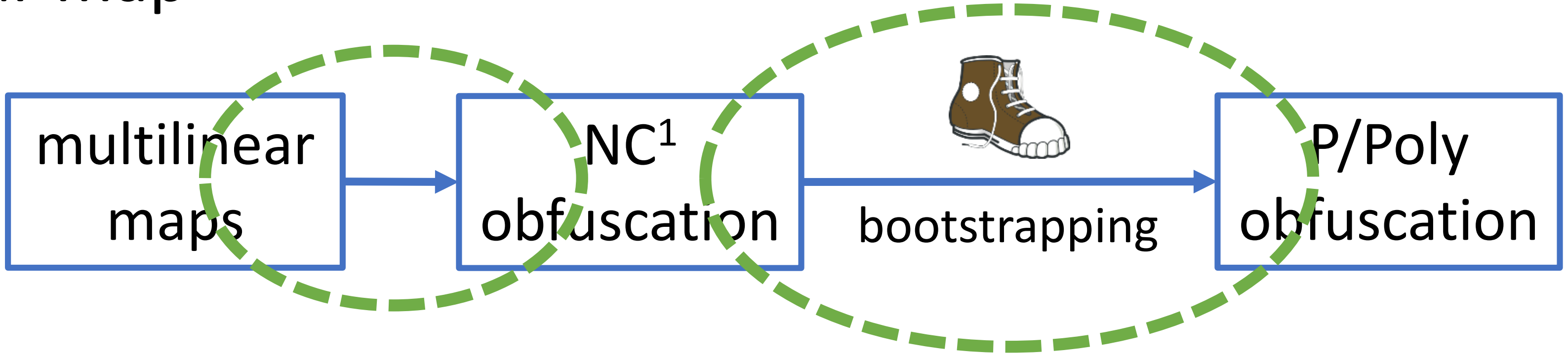
- Bootstrapping: [GGHRSW13, BR14, App14]



  - For AES, requires $\gg 2^{100}$ levels of multinearity and $\gg 2^{100}$ encodings
- Direct obfuscation of circuits: [Zim15, AB15]
  - For AES, already require $\gg 2^{100}$ levels of multilinearity
- Non-Black Box: [Lin16a, LV16, Lin16b, AS17, LT17]
  - Only requires constant-degree multilinear maps (e.g., 3-linear maps [LT17])
  - Multilinear maps are complex, so non-black box use of the multilinear maps will be difficult to implement

How (Im)Practical is Obfuscation?

Focus of this work will be on candidates that make black-box use of multilinear map

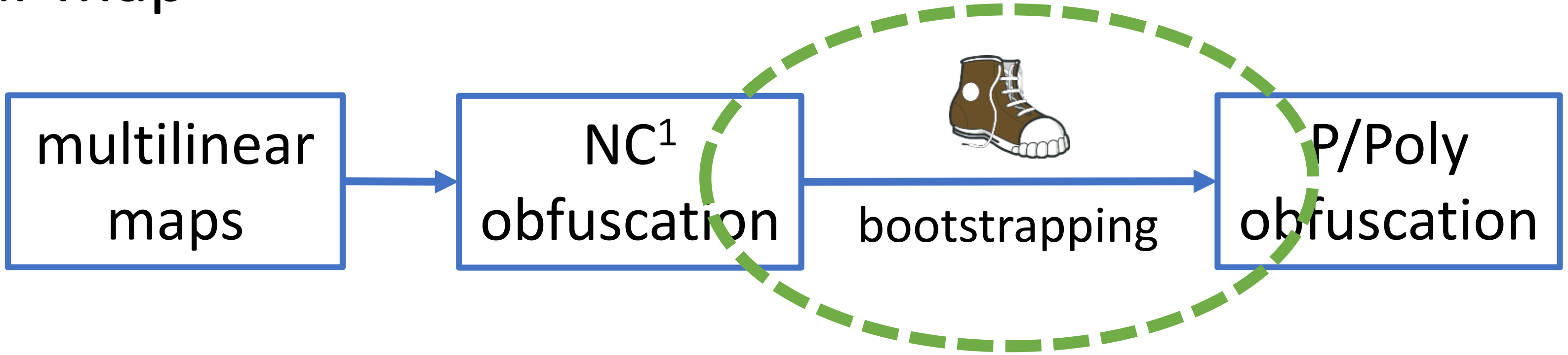


prior works have focused on improving the efficiency of obfuscation for NC^1 (branching programs) [AGIS14, BMSZ16]

our goal: improve efficiency of **bootstrapping**

How (Im)Practical is Obfuscation?

Focus of this work will be on candidates that make black-box use of multilinear map



- Obfuscated program does two things: <u>FHE decryption</u> and <u>proof verification</u> (of correct evaluation)
- NC^1 obfuscator works on *branching programs*, so need primitives with <u>short</u> branching programs (e.g., computing an inner products over a small field)
- FHE decryption is (rounded) inner product [BV11, BGV12, Bra12, GSW13, AP14, DM15, …], so just need a SNARG with simple verification

Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG)
that can be verified by a <u>short</u> branching program

Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG)
that can be verified by a <u>short</u> branching progr

Succinct non-interactive arguments (SNARG) for NP relation [GW11]

- $\mathrm{Setup}(1^{\lambda}) \to (\sigma, \tau)$: outputs common reference string (CRS) σ and verification state τ
- $\mathrm{Prove}(\sigma, x, w) \to \pi$: on input the CRS σ, the statement x *and* the witness w, outputs a proof π
- $\mathrm{Verify}(\tau, x, \pi) \to 0/1$: on input the verification state τ, the statement x, decides if the proof π is valid

Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG)
that can be verified by a <u>short</u> branching progr

Succinct non-interactive arguments (SNARG) for NP relation [GW11]

- Must satisfy usual notions of completeness and computational soundness
- Succinctness: proof size and verifier run-time should be polylogarithmic in the circuit size (for circuit satisfiability)
  - Verifier run-time: $\text{poly}(\lambda + |x| + \log\,|C|)$
  - Proof size: $\text{poly}(\lambda + \log\,|C|)$

Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG) that can be ... hort branc...

Verification state τ must be secret

Allow Setup algorithm to run in time $\mathrm{poly}(\lambda + |C|)$

Main result: new designated-verifier SNARGs in the preprocessing model with the following properties:

Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG) that can be verified by a <u>short</u> branching program

proofs have size $\tilde{O}(\lambda)$

prover complexity is $\tilde{O}(|C|)$

ult: new designate the preprocessing model with the properties:

- Quasi-optimal succinctness
- Quasi-optimal prover complexity

<u>first</u> SNARG that is "quasi-optimal"

Asymptotics based on achieving $\mathrm{negl}(\lambda)$ soundness error against provers of size 2^λ

Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG) that can be verified by a short branching program

Main result: new designated-verifier SNARGs in the preprocessing model with the following properties:

- Quasi-optimal succinctness
- Quasi-optimal prover complexity
- Post-quantum security
- Works over polynomial-size fields

first SNARG that is "quasi-optimal"

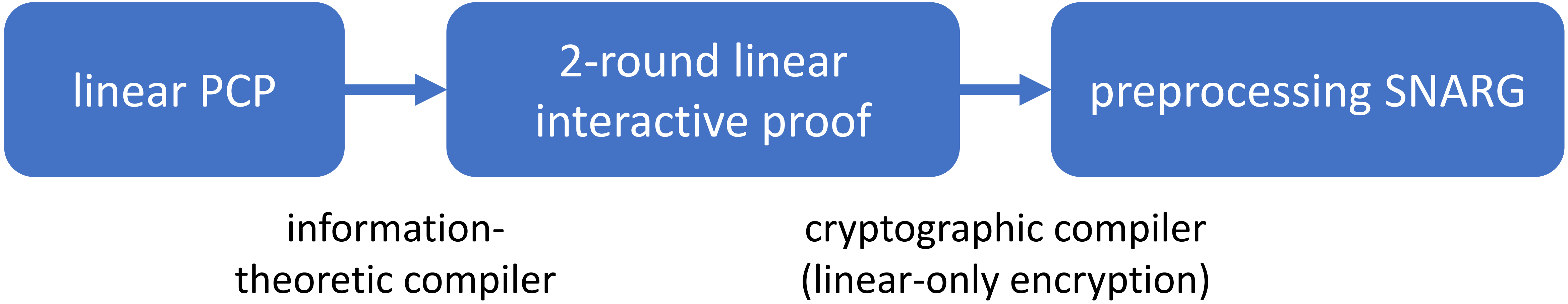
New SNARG candidates are lattice-based

- Over integer lattices, verification is branching-program friendly
- Over ideal lattices, SNARGs are quasi-optimal

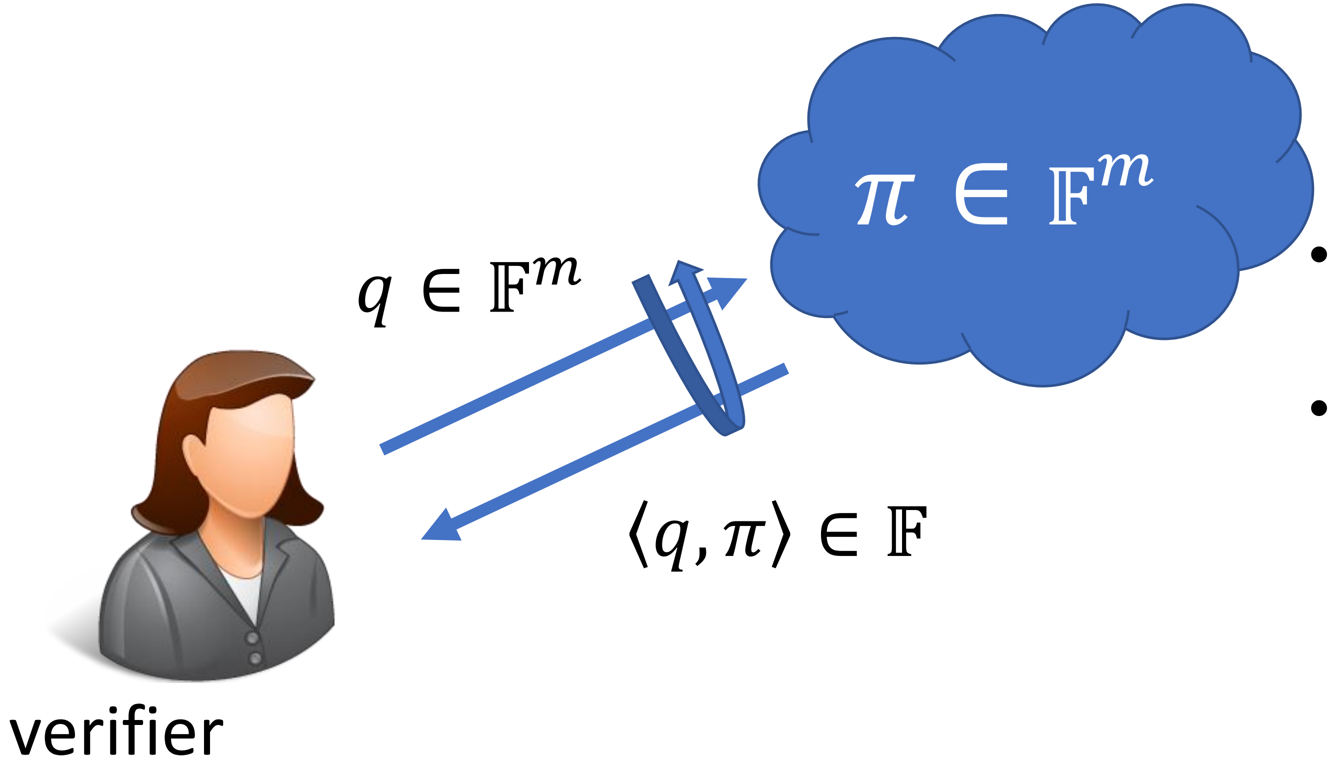
Branching-Program-Friendly SNARGs

Goal: construct a succinct non-interactive argument (SNARG)
that can be verified by a <u>short</u> branching program

Starting point: preprocessing SNARGs from [BCIOP13]

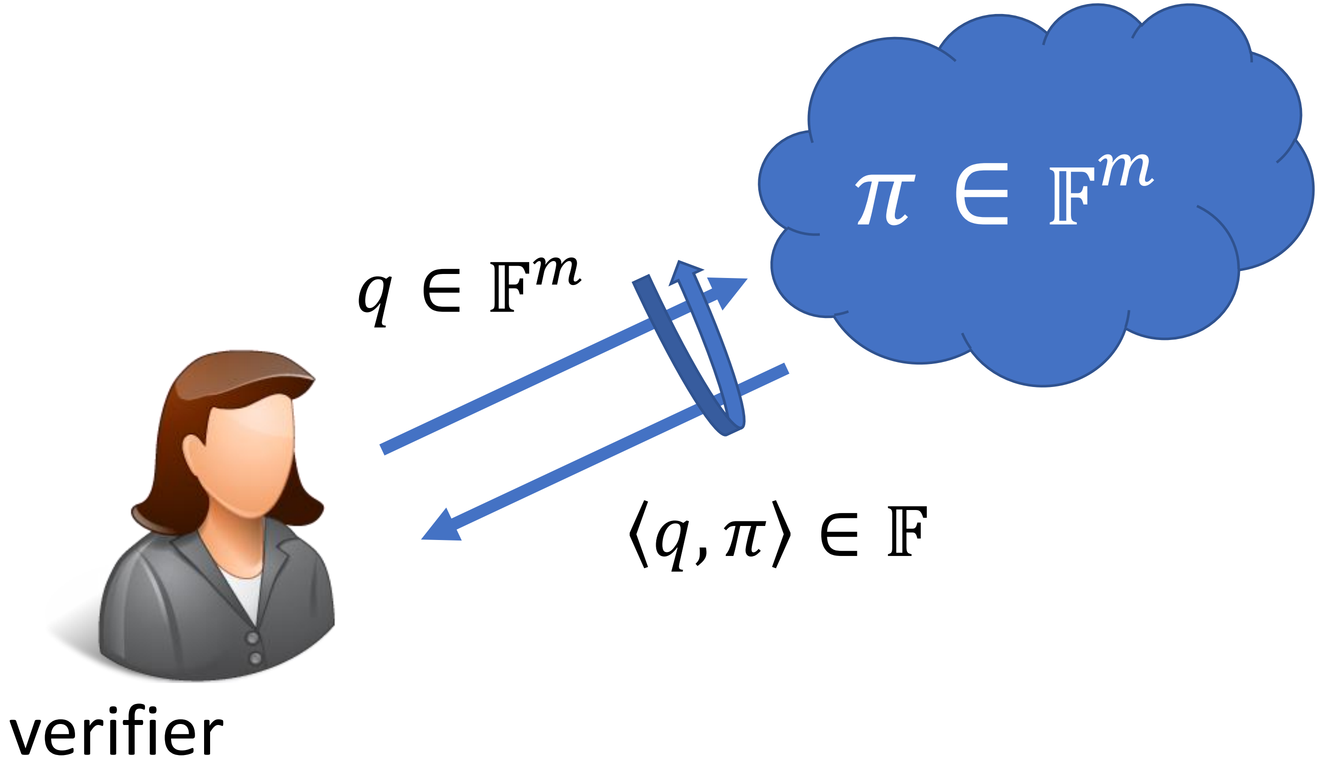


Linear PCPs (LPCPs) [IKO07]



- 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]
  - 3-query LPCP based on quadratic span programs: $m = O(|C|)$ [GGPR13]

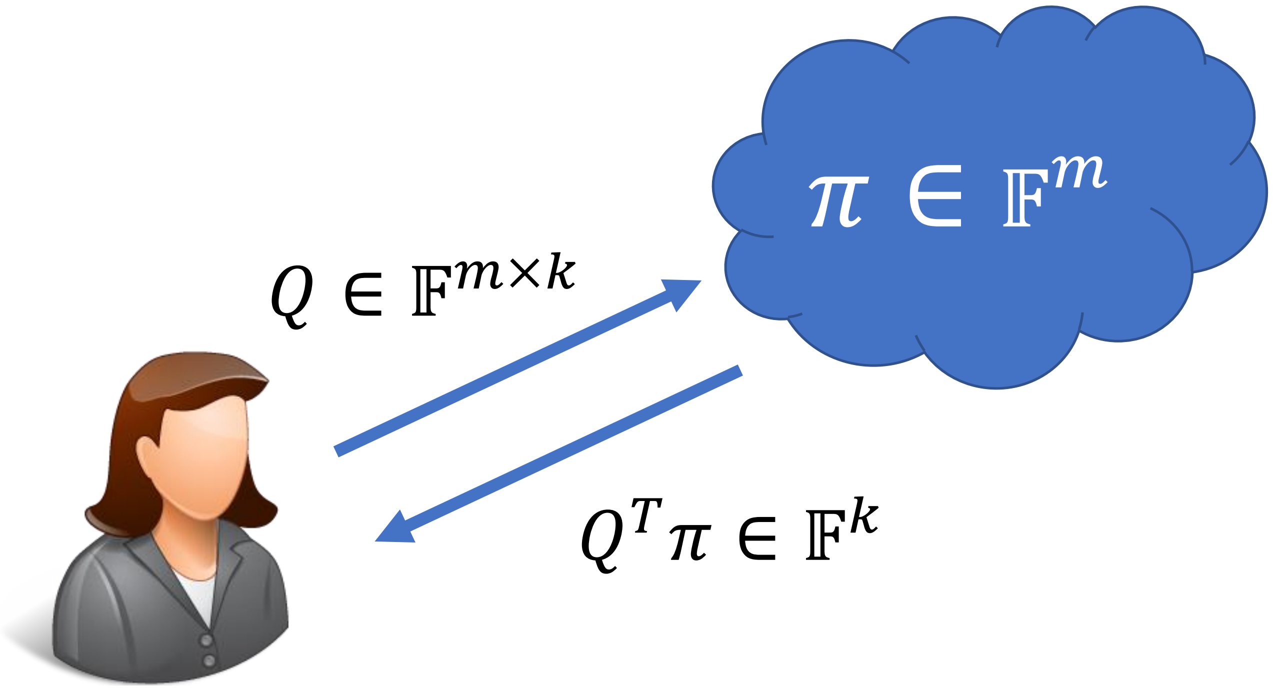
Linear PCPs (LPCPs) [IKO07]



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

Linear PCPs (LPCPs) [IKO07]

Equivalent view (if verifier is oblivious):



verifier

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix} \in \mathbb{F}^{m \times k}$$

pack all queries into single matrix

From Linear PCPs to Preprocessing SNARGs [BCIOP13]

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

$$Q = \underbrace{\begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}}_{\text{part of the CRS}}$$

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

Two problems:

- Malicious prover can choose π based on queries
- Malicious prover can apply different π to the different columns of Q

From Linear PCPs to Preprocessing SNARGs [BCIOP13]

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

$$Q = \underbrace{\begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}}_{\text{part of the CRS}}$$

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

Two problems:

- Malicious prover can choose π based on queries
- Malicious prover can apply different π to the different columns of Q

From Linear PCPs to Preprocessing SNARGs [BCIOP13]

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

$$\underbrace{Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}}_{\text{part of the CRS}}$$

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

Step 1: Encrypt elements of Q using additively homomorphic encryption scheme

- Prover homomorphically computes $Q^T\pi$
- Verifier decrypts encrypted response vector and performs LPCP verification

From Linear PCPs to Preprocessing SNARGs [BCIOP13]

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

$$Q = \underbrace{\begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}}_{\text{part of the CRS}}$$

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

Two problems:

- Malicious prover can choose π based on queries
- Malicious prover can apply different π to the different columns of Q

From Linear PCPs to Preprocessing SNARGs

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

$$Q = \underbrace{\begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}}_{\text{part of the CRS}}$$

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

Step 2: Conjecture that the encryption scheme only supports a limited subset of homomorphic operations (linear-only vector encryption)

Linear-Only Vector Encryption

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

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

$\vdots$

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

plaintext space is a *vector* space

Linear-Only Vector Encryption

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

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

$\vdots$

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

$\sum_{i \in [n]} \alpha_i v_i \in \mathbb{F}^k$

plaintext space is a *vector* space

encryption scheme is semantically-secure and additively homomorphic

Linear-Only Vector Encryption

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

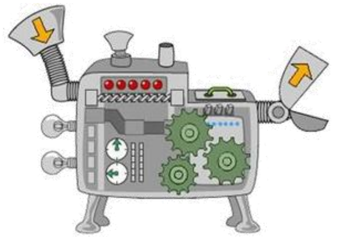
$v_2 \in \mathbb{F}^k$

$\vdots$

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

adversary

ct



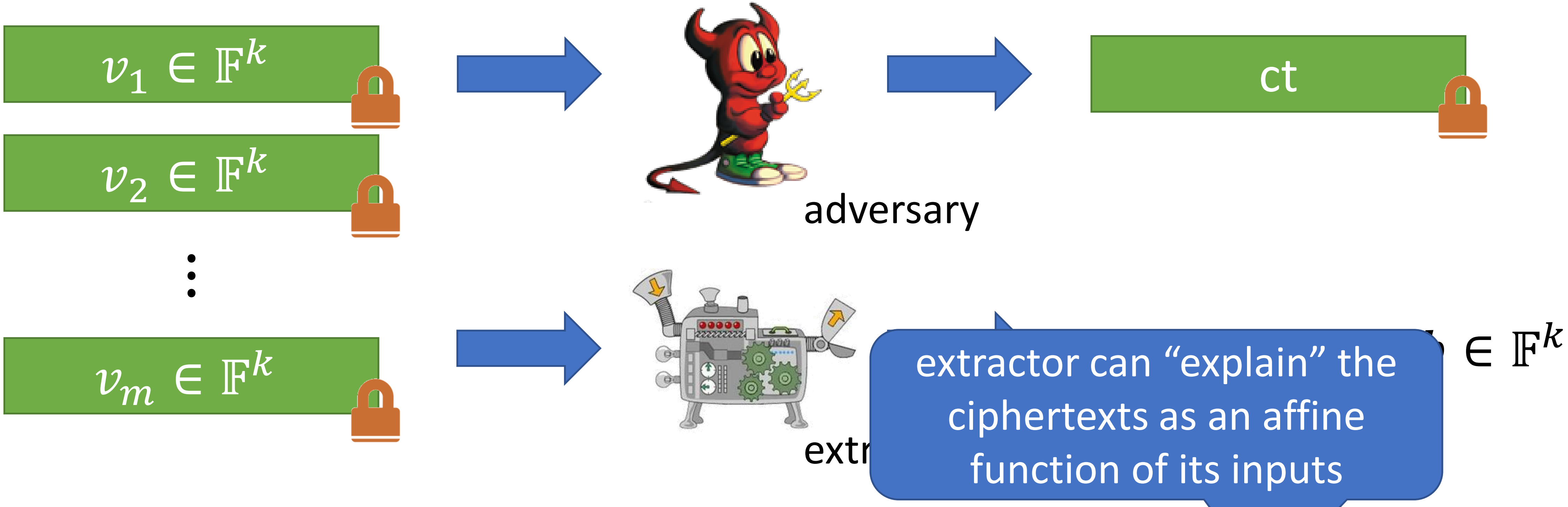
extractor

$\alpha_1, \dots, \alpha_m \in \mathbb{F}, b \in \mathbb{F}^k$

For all adversaries, there is an efficient extractor such that if ct is valid, then the extractor is able to produce a vector of coefficients $(\alpha_1, \dots, \alpha_m) \in \mathbb{F}^m$ and $b \in \mathbb{F}^k$ such that $\text{Decrypt}(\text{sk}, \text{ct}) = \sum_{i \in [n]} \alpha_i v_i + b$

Weaker property also suffices. [See paper for details.]

Linear-Only Vector Encryption



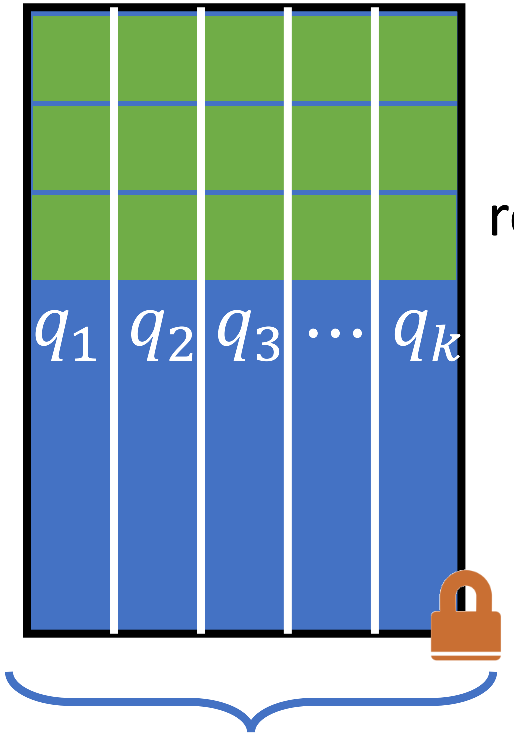
For all adversaries, there is an efficient extractor such that if ct is valid, then the extractor is able to produce a vector of coefficients $(\alpha_1, \ldots, \alpha_m) \in \mathbb{F}^m$ and $b \in \mathbb{F}^k$ such that $\mathrm{Decrypt}(\mathrm{sk}, \mathrm{ct}) = \sum_{i \in [n]} \alpha_i v_i + b$

Weaker property also suffices. [See paper for details.]

From Linear PCPs to Preprocessing SNARGs

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

$Q =$



encrypt row by row

part of the CRS

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

Step 2: Conjecture that the encryption scheme only supports a limited subset of homomorphic operations (linear-only vector encryption)

Linear-only vector encryption ⇒ all prover strategies can be explained by (π, b) as $Q^T\pi + b$

[See paper for full details.]

Comparison with [BCIOP13]

Our construction

linear PCP → preprocessing SNARG

Preprocessing SNARGs from [BCIOP13]:

linear PCP → 2-round linear interactive proof → preprocessing SNARG

Comparison with [BCIOP13]

Our construction

ssing SNARG

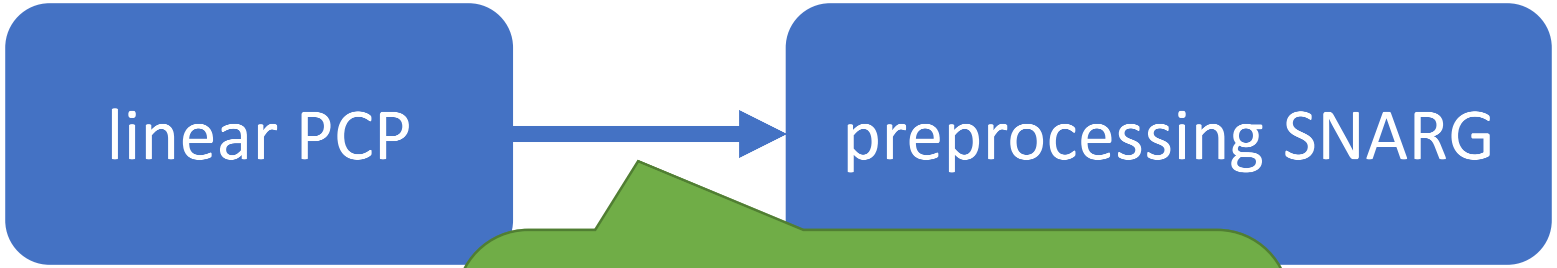
introduce additional consistency check to force prover to apply consistent linear function – soundness only over a large field

Preprocessi

linear PCP → 2-round linear interactive proof → preprocessing SNARG

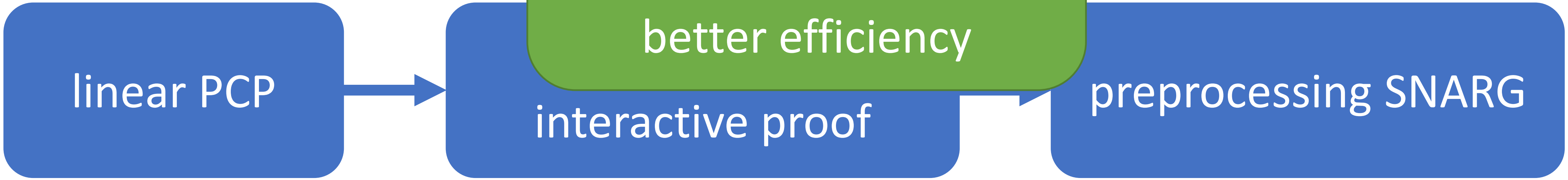
Comparison with [BCIOP13]

Our construction



stronger cryptographic assumption, but enables new constructions with better efficiency

Preprocessing SNARGs fr



Instantiating Linear-Only Vector Encryption

Conjecture: Regev-based encryption (specifically, the [PVW08] variant) is a linear-only vector encryption scheme.

Proof verification essentially consists of computing a rounded matrix-vector product

Obfuscation-friendly!

Concrete Comparisons

Construction	Public vs. Designated	Prover Complexity	Proof Size	Assumption
CS Proofs [Mic00]	Public	$\tilde{O}(\lvert C\rvert + \lambda^2)$	$\tilde{O}(\lambda^2)$	Random Oracle
Groth [Gro10]	Public	$\tilde{O}(\lvert C\rvert^2\lambda + \lvert C\rvert\lambda^2)$	$\tilde{O}(\lambda)$	Knowledge of Exponent
GGPR [GGPR12]	Public	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	
BCIOP (Pairing) [BCIOP13]	Public	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	Linear-Only Encryption
BCIOP (LWE) [BCIOP13]	Designated	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	
Our Construction (LWE)	Designated	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption [See paper.]
Our Construction (RLWE)	Designated	$\tilde{O}(\lvert C\rvert)$	$\tilde{O}(\lambda)$	

Only $\mathrm{negl}(\lambda)$-soundness (instead of $2^{-\lambda}$-soundness) against 2^{λ}-bounded provers

Concrete Comparisons

Construction	Public vs. Designated	Prover Complexity	Proof Size	Assumption
CS Proofs [Mic00]	Public	$\tilde{O}(\lvert C\rvert + \lambda^2)$	$\tilde{O}(\lambda^2)$	Random Oracle
Groth [Gro10]	Public	$\tilde{O}(\lvert C\rvert^2\lambda + \lvert C\rvert\lambda^2)$	$\tilde{O}(\lambda)$	Knowledge of Exponent
GGPR [GGPR12]	Public	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	
BCIOP (Pairing) [BCIOP13]	Public	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	Linear-Only Encryption
BCIOP (LWE) [BCIOP13]	Designated	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	
Our Construction (LWE)	Designated	$\tilde{O}(\lvert C\rvert\lambda)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption [See paper.]
Our Construction (RLWE)	Designated	$\tilde{O}(\lvert C\rvert)$	$\tilde{O}(\lambda)$	

Post-quantum resistant!

Back to Obfuscation…

For bootstrapping obfuscation…

- Obfuscate FHE decryption and SNARG verification
- Degree of multilinearity: $\approx 2^{12}$
- Number of encodings: $\approx 2^{44}$

Many optimizations. [See paper for details.]

Still infeasible, but much, much better than 2^{100} for previous black-box constructions!

Looking into obfuscation gave us new insights into constructing better SNARGs:

- More direct framework of building SNARGs from linear PCPs
- Quasi-succinct construction from standard lattices
- Quasi-optimal construction from ideal lattices [See paper.]

Open Problems

Publicly-verifiable SNARGs from lattice-based assumptions?

Concrete efficiency of new lattice-based SNARGs?

Thank you!

`http://eprint.iacr.org/2017/240`