Lattice-Based Succinct Non-Interactive Arguments

David Wu
Stanford University

based on joint works with Dan Boneh, Yuval Ishai, and Amit Sahai

Proof Systems and Argument Systems

[GMR85]

$x \in \{0,1\}^*$

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

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

prover

verifier

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

"Honest prover convinces honest verifier of true statements"

Soundness: $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V \rangle(x) = \text{accept}] \leq \varepsilon$

"No prover can convince honest verifier of false statement"

Proof Systems and Argument Systems

[GMR85]

$x \in \{0,1\}^*$

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

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

prover

verifier

In an argument system, we relax soundness to only consider computationally-bounded (i.e., polynomial-time) provers P^*

Completeness:

“Honest p… …vinces honest verifier of true statements”

Soundness: $\forall x \notin \mathcal{L},\ \forall P^* : \Pr[\langle P^*, V\rangle(x) = \text{accept}] \leq \varepsilon$

“No prover can convince honest verifier of false statement”

Succinct Arguments

[Kil92]

$x \in \{0,1\}^*$

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

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

prover

verifier

Argument system is **succinct** if:

- Communication is $\text{poly}(\lambda + \log|C|)$
- V can be implemented by a circuit of size $\text{poly}(\lambda + |x| + \log|C|)$

Verifier complexity significantly smaller than classic NP verifier

Succinct Non-Interactive Arguments (SNARGs)

[Mic94, GW11]

$x \in \{0,1\}^*$

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

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

prover

$\pi = P(x,w)$

verifier

Argument system is **succinct** if:

- Communication is $\text{poly}(\lambda + \log|C|)$
- V can be implemented by a circuit of size $\text{poly}(\lambda + |x| + \log|C|)$

For general NP languages, succinct non-interactive arguments are <u>unlikely</u> to exist in the standard model [BP04, Wee05]

Succinct Non-Interactive Arguments (SNARGs)

[Mic94, 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)

[Mic94, GW11]

Preprocessing SNARGs: allow "expensive" setup

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

Can consider publicly-verifiable and secretly-verifiable SNARGs

common reference string (CRS) σ

verification state τ

prover

(x, w)

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

Argument consists of a single message

verifier

x

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

Complexity Metrics for SNARGs

Soundness: for all provers $P^\star$ of size 2^λ:

$$x \notin \mathcal{L}_C \Longrightarrow \Pr\big[V\big(x, P^*(x)\big) = 1\big] \leq 2^{-\lambda}$$

How short can the proofs be?

$$|\pi| = \Omega(\lambda)$$

Even in the designated-verifier setting

How much work is needed to generate the proof?

$$|P| = \Omega(|C|)$$

Quasi-Optimal SNARGs

Soundness: for all provers $P^\star$ of size 2^λ:

$$x \notin \mathcal{L}_C \Longrightarrow \Pr\left[V\left(x, P^*(x)\right) = 1\right] \leq 2^{-\lambda}$$

A SNARG (for Boolean circuit satisfiability) is <u>quasi-optimal</u> if it satisfies the following properties:

- Quasi-optimal succinctness:

$$|\pi| = \lambda \cdot \mathrm{polylog}(\lambda, |C|) = \tilde{O}(\lambda)$$

- Quasi-optimal prover complexity:

$$|P| = \tilde{O}(|C|) + \mathrm{poly}(\lambda, \log|C|)$$

Asymptotic Comparisons

Construction	Prover Complexity	Proof Size	Assumption
CS Proofs [Mic94]	$\tilde{O}(\|C\|)$	$\tilde{O}(\lambda^2)$	Random Oracle
Groth [Gro16]	$\tilde{O}(\lambda\|C\|)$	$\tilde{O}(\lambda)$	Generic Group
Groth [Gro10]	$\tilde{O}(\lambda\|C\|^2 + \|C\|\lambda^2)$	$\tilde{O}(\lambda)$	Knowledge of Exponent
GGPR [GGPR12]	$\tilde{O}(\lambda\|C\|)$	$\tilde{O}(\lambda)$	
BCIOP (Pairing) [BCIOP13]	$\tilde{O}(\lambda\|C\|)$	$\tilde{O}(\lambda)$	Linear-Only Encryption
BISW (integer lattices) [BISW17]	$\tilde{O}(\lambda\|C\|)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption
BISW (ideal lattices) [BISW18]	$\tilde{O}(\|C\|)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption

For simplicity, we ignore low order terms $\text{poly}(\lambda, \log|C|)$ in the prover complexity

Constructing (Quasi-Optimal) SNARGs

New framework for building preprocessing SNARGs (following [BCIOP13]):

Step 1 (information-theoretic):

- Identify useful information-theoretic building block (linear PCPs and linear MIPs)

Step 2 (cryptographic):

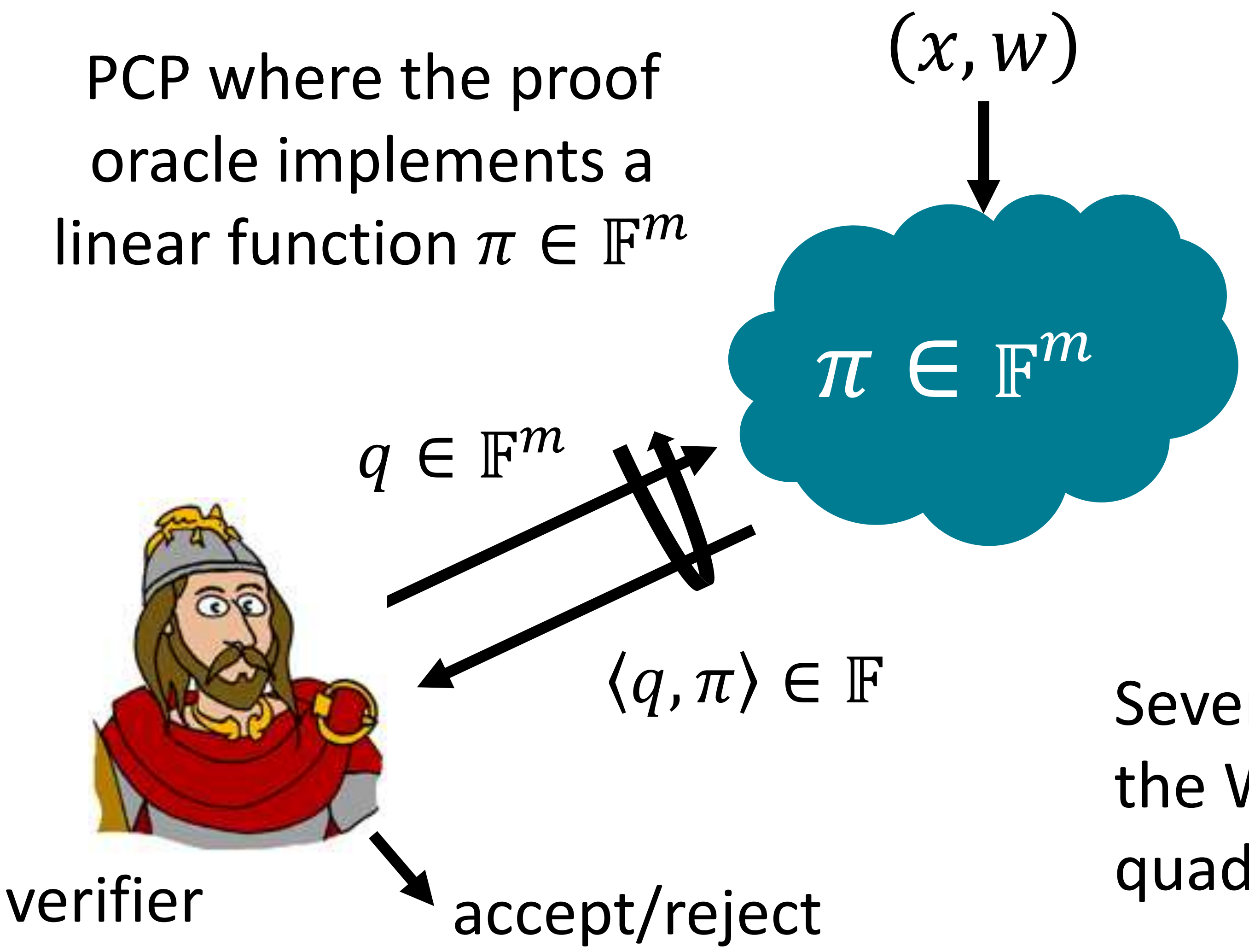
- Use cryptographic primitives to compile information-theoretic building block into a preprocessing SNARG

Instantiating our framework yields new lattice-based SNARG candidates

Linear PCPs

[IKO07]

PCP where the proof oracle implements a linear function $\pi \in \mathbb{F}^m$



In these instantiations, verifier is <u>oblivious</u> (queries independent of statement)

Several possible instantiations: based on the Walsh-Hadamard code [ALMSS92] or quadratic span programs [GGPR13]

From Linear PCPs to SNARGs

[BCIOP13]

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

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}$$

part of the CRS

Prover constructs linear PCP π from (x, w)

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

Prover computes responses to linear PCP queries

$$\langle \pi, q_1 \rangle \; \langle \pi, q_2 \rangle \; \cdots \; \langle \pi, q_k \rangle$$

SNARG proof

From Linear PCPs to SNARGs

[BCIOP13]

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

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}$$

part of the CRS

Two issues:

- Malicious prover can choose π based on the queries
- Malicious prover can apply different π to each query

Prover computes responses to linear PCP queries

$\langle \pi, q_1 \rangle$	$\langle \pi, q_2 \rangle$	$\cdots$	$\langle \pi, q_k \rangle$

SNARG proof

From Linear PCPs to SNARGs

[BCIOP13]

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

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}$$

part of the CRS

Two issues:

- Malicious prover can choose π based on the queries
- Malicious prover can apply different π to each query

Prover computes responses to linear PCP queries

$\langle \pi, q_1 \rangle$	$\langle \pi, q_2 \rangle$	$\cdots$	$\langle \pi, q_k \rangle$

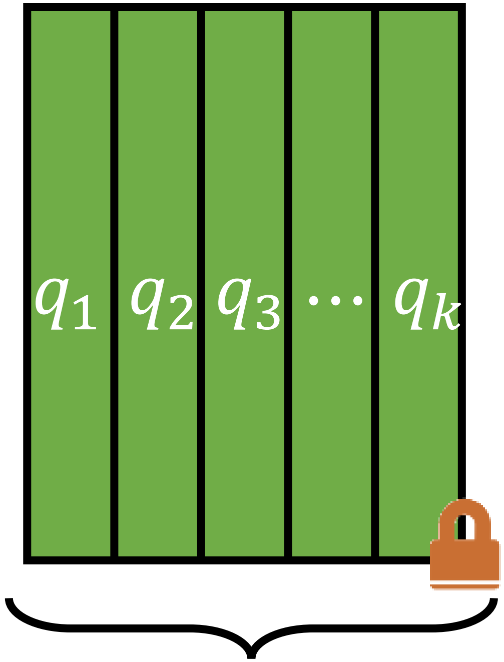
SNARG proof

From Linear PCPs to SNARGs

[BCIOP13]

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

$Q =$



part of the CRS

Two issues:

- Malicious prover can choose π based on the queries
- Malicious prover can apply different π to each query

Step 1: Verifier encrypts its queries using an additively homomorphic encryption scheme

- Prover homomorphically computes $Q^T\pi$
- Verifier decrypts encrypted response vector and applies linear PCP verification

From Linear PCPs to SNARGs

[BCIOP13]

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

$$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}$$

part of the CRS

Two issues:

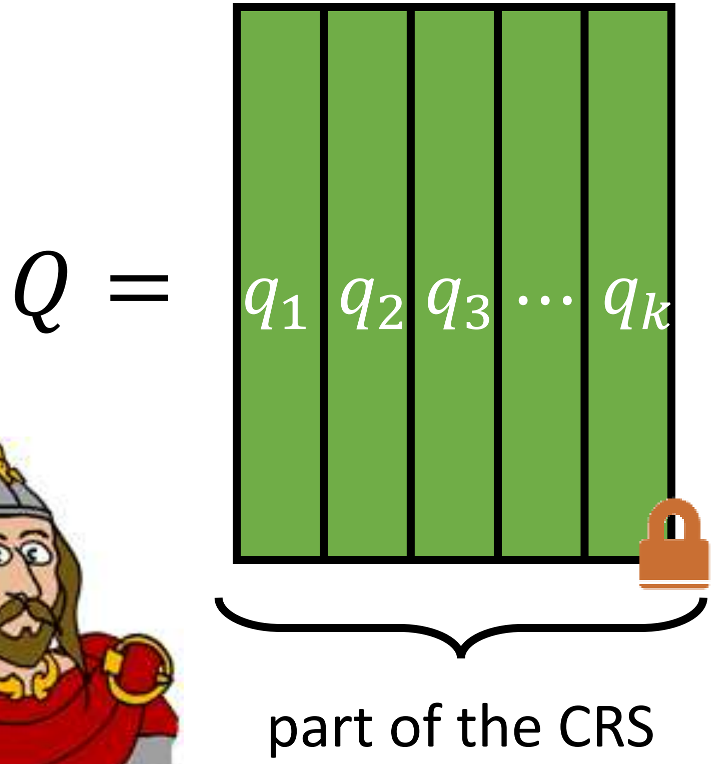
- Malicious prover can choose π based on the queries
- Malicious prover can apply different π to each query

Step 1: Verifier encrypts its queries using an additively homomorphic encryption scheme

- Prover homomorphically computes $Q^T \pi$
- Verifier decrypts encrypted response vector and applies linear PCP verification

From Linear PCPs to SNARGs

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



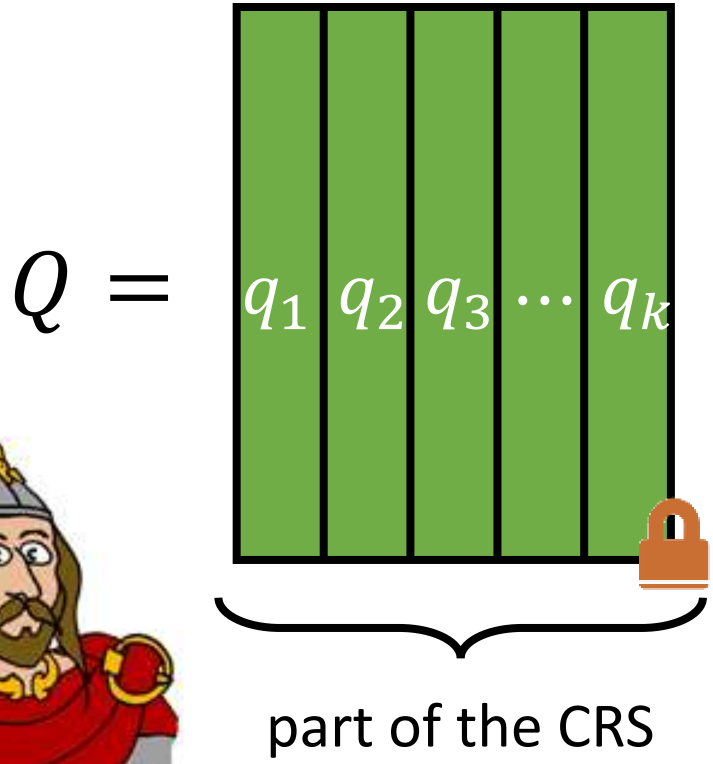
Two issues:

- Malicious prover can choose π based on the queries
- Malicious prover can apply different π to each query

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

From Linear PCPs to SNARGs

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



- Differs from [BCIOP13] compiler which relies on additional consistency checks to build a preprocessing SNARG
- Using linear-only vector encryption allows for efficient instantiation from lattices (resulting SNARG satisfies quasi-optimal succinctness)

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

Linear-Only Vector Encryption

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

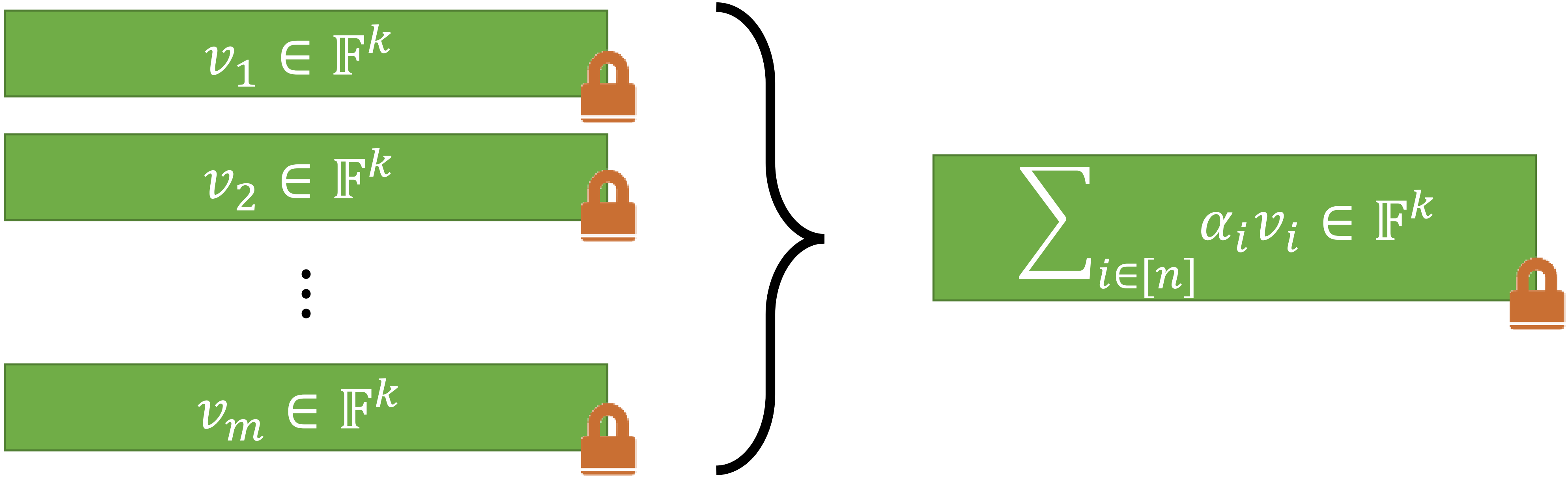
$v_2 \in \mathbb{F}^k$

$\vdots$

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

plaintext space is a *vector* space

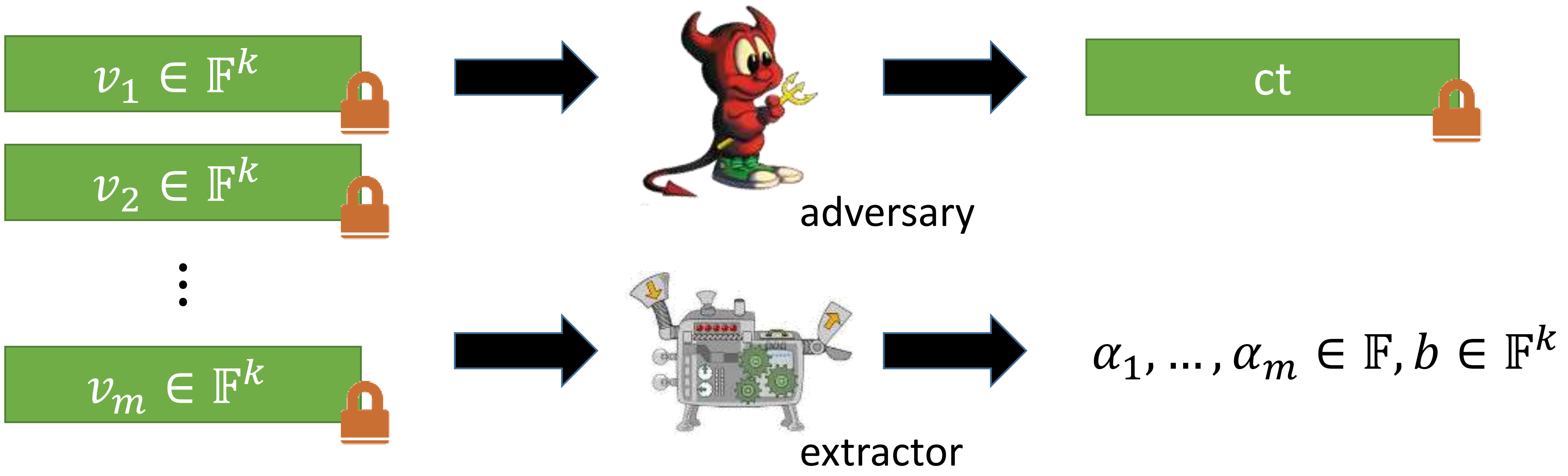
Linear-Only Vector Encryption



plaintext space is a *vector* space

encryption scheme is semantically-secure and additively homomorphic

Linear-Only Vector Encryption



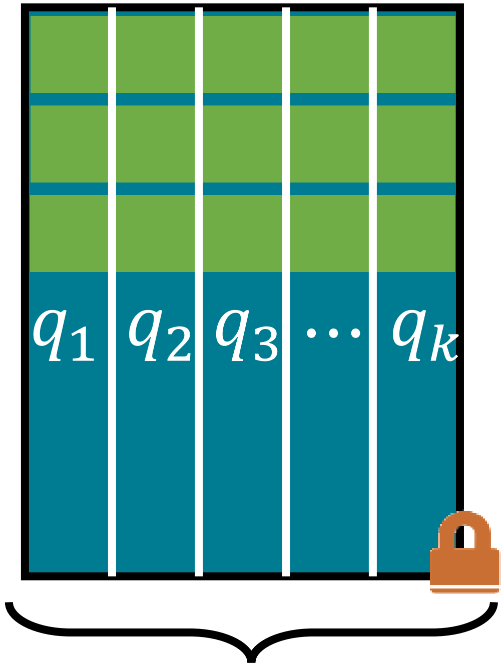
For all adversaries, there is an efficient extractor such that if ct is valid, then the extractor is able to produce a vector of coefficients $(\alpha_1, \dots, \alpha_m) \in \mathbb{F}^m$ and $b \in \mathbb{F}^k$ such that $\text{Decrypt}(\text{sk}, \text{ct}) = \sum_{i \in [n]} \alpha_i v_i + b$

[Weaker property also suffices]

From Linear PCPs to SNARGs

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

$Q =$



encrypt row by row

part of the CRS

Linear-only vector encryption ensures that all prover strategies can be explained by a *linear* function ⇒ can appeal to soundness of underlying linear PCP to argue soundness

Prover computes responses to linear PCP queries

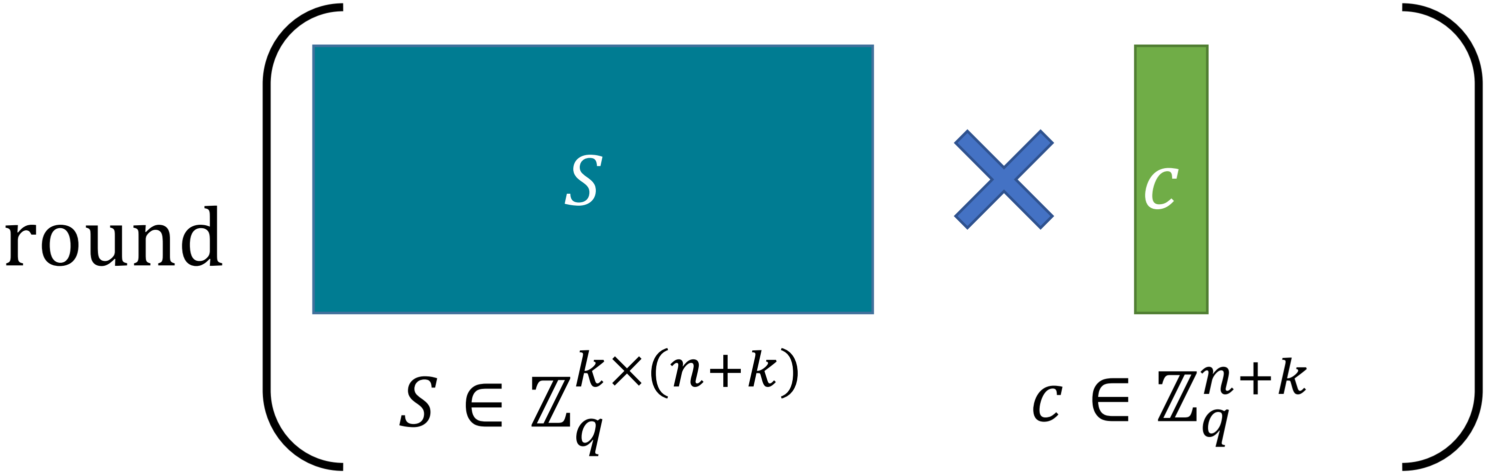
$\langle \pi, q_1 \rangle$	$\langle \pi, q_2 \rangle$	$\cdots$	$\langle \pi, q_k \rangle$

SNARG proof

Instantiating Linear-Only Vector Encryption

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

PVW decryption (for plaintexts with dimension k):



Each row of S can be viewed as an independent Regev secret key

Complexity of the Construction

Evaluating inner product requires $\Omega(|C|)$ homomorphic operations; prover complexity:
$\Omega(\lambda) \cdot \Omega(|C|) = \Omega(\lambda|C|)$

$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}$

Prover constructs linear PCP π from (x, w)

(x, w)

$\downarrow$

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

Proof consists of a single ciphertext: total length $O(\lambda)$ bits

Prover computes responses to linear PCP queries

$\langle \pi, q_1 \rangle$	$\langle \pi, q_2 \rangle$	$\cdots$	$\langle \pi, q_k \rangle$

SNARG proof

Asymptotic Comparisons

Construction	Prover Complexity	Proof Size	Assumption
CS Proofs [Mic94]	$\tilde{O}(\lvert C\rvert)$	$\tilde{O}(\lambda^2)$	Random Oracle
Groth [Gro16]	$\tilde{O}(\lambda\lvert C\rvert)$	$\tilde{O}(\lambda)$	Generic Group
Groth [Gro10]	$\tilde{O}(\lambda\lvert C\rvert^2 + \lvert C\rvert\lambda^2)$	$\tilde{O}(\lambda)$	Knowledge of Exponent
GGPR [GGPR12]	$\tilde{O}(\lambda\lvert C\rvert)$	$\tilde{O}(\lambda)$	
BCIOP (Pairing) [BCIOP13]	$\tilde{O}(\lambda\lvert C\rvert)$	$\tilde{O}(\lambda)$	Linear-Only Encryption
BISW (integer lattices) [BISW17]	$\tilde{O}(\lambda\lvert C\rvert)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption

For simplicity, we ignore low order terms $\text{poly}(\lambda, \log\lvert C\rvert)$ in the prover complexity

Towards Quasi-Optimality

Evaluating inner product requires $\Omega(|C|)$ homomorphic operations; prover complexity:

$\Omega(\lambda) \cdot \Omega(|C|) = \Omega(\lambda|C|)$

Prover constructs linear PCP π from (x, w)

(x, w)

ow

$Q = \begin{bmatrix} q_1 & q_2 & q_3 & \cdots & q_k \end{bmatrix}$

We pay $\Omega(\lambda)$ for each homomorphic operation. Can we reduce this?

Pr

to li

Proof consists of a <u>constant</u> number of ciphertexts: total length $O(\lambda)$ bits

$\langle \pi, q_1 \rangle$	$\langle \pi, q_2 \rangle$	$\cdots$	$\langle \pi, q_k \rangle$

SNARG proof

Linear-Only Encryption over Rings

Consider encryption scheme over a polynomial ring $R_p = \mathbb{Z}_p[x]/\Phi_d(x) \cong \mathbb{F}_p^\ell$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_\ell \end{pmatrix} + \begin{pmatrix} x_1' \\ x_2' \\ x_3' \\ \vdots \\ x_\ell' \end{pmatrix} = \begin{pmatrix} x_1 + x_1{}' \\ x_2 + x_2' \\ x_3 + x_3' \\ \vdots \\ x_\ell + x_\ell' \end{pmatrix}$$

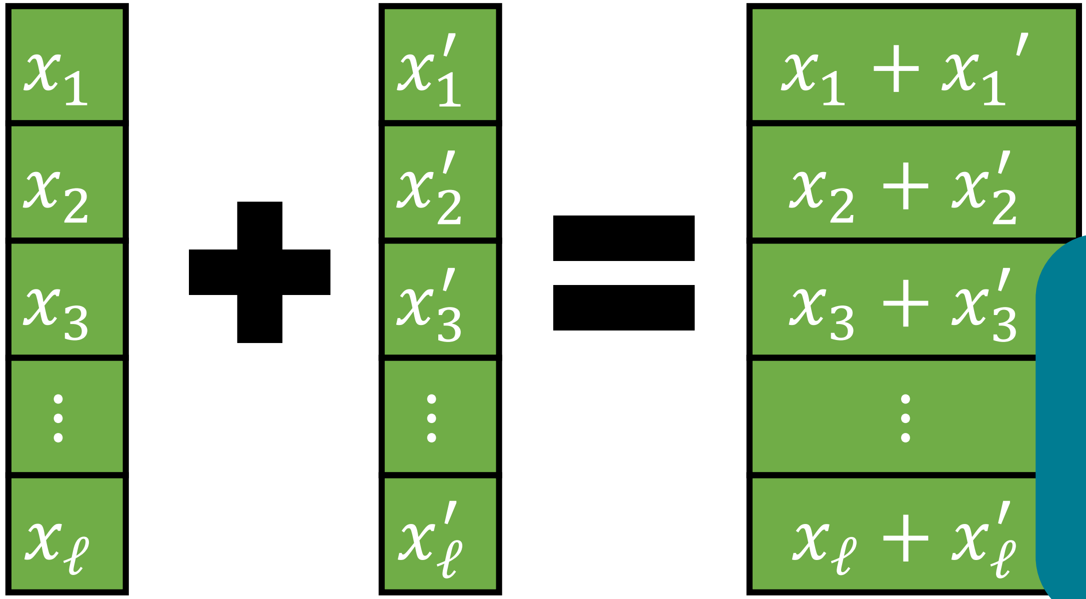
Homomorphic operations correspond to <u>component-wise</u> additions and scalar multiplications

Plaintext space can be viewed as a vector of field elements

Using RLWE-based encryption schemes, can encrypt $\ell = \tilde{O}(\lambda)$ field elements ($p = \text{poly}(\lambda)$) with ciphertexts of size $\tilde{O}(\lambda)$

Linear-Only Encryption over Rings

Consider encryption scheme over a polynomial ring $R_p = \mathbb{Z}_p[x]/\Phi_d(x) \cong \mathbb{F}_p^\ell$



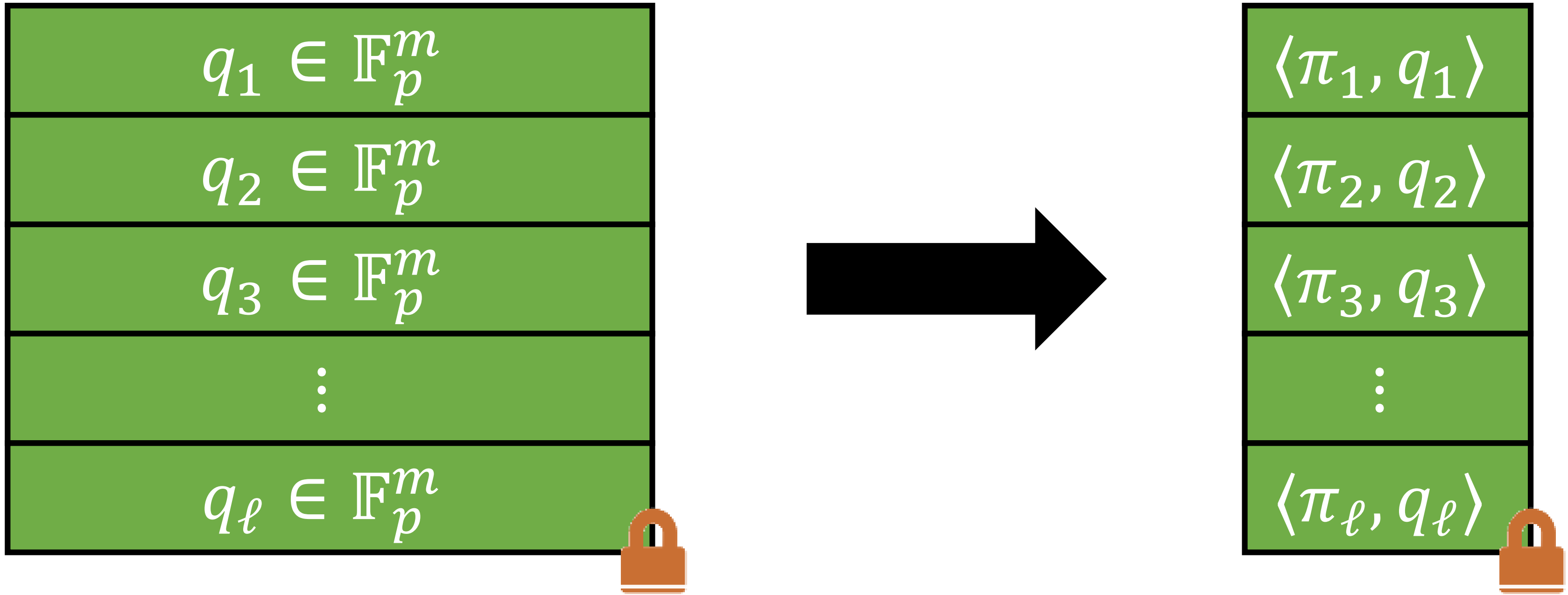
Homomorphic operations

Amortized cost of homomorphic operation on a single field element is $\text{polylog}(\lambda)$

Plaintext space can be viewed as a vector of field elements

Using RLWE-based encryption schemes, can encrypt $\ell = \tilde{O}(\lambda)$ field elements ($p = \text{poly}(\lambda)$) with ciphertexts of size $\tilde{O}(\lambda)$

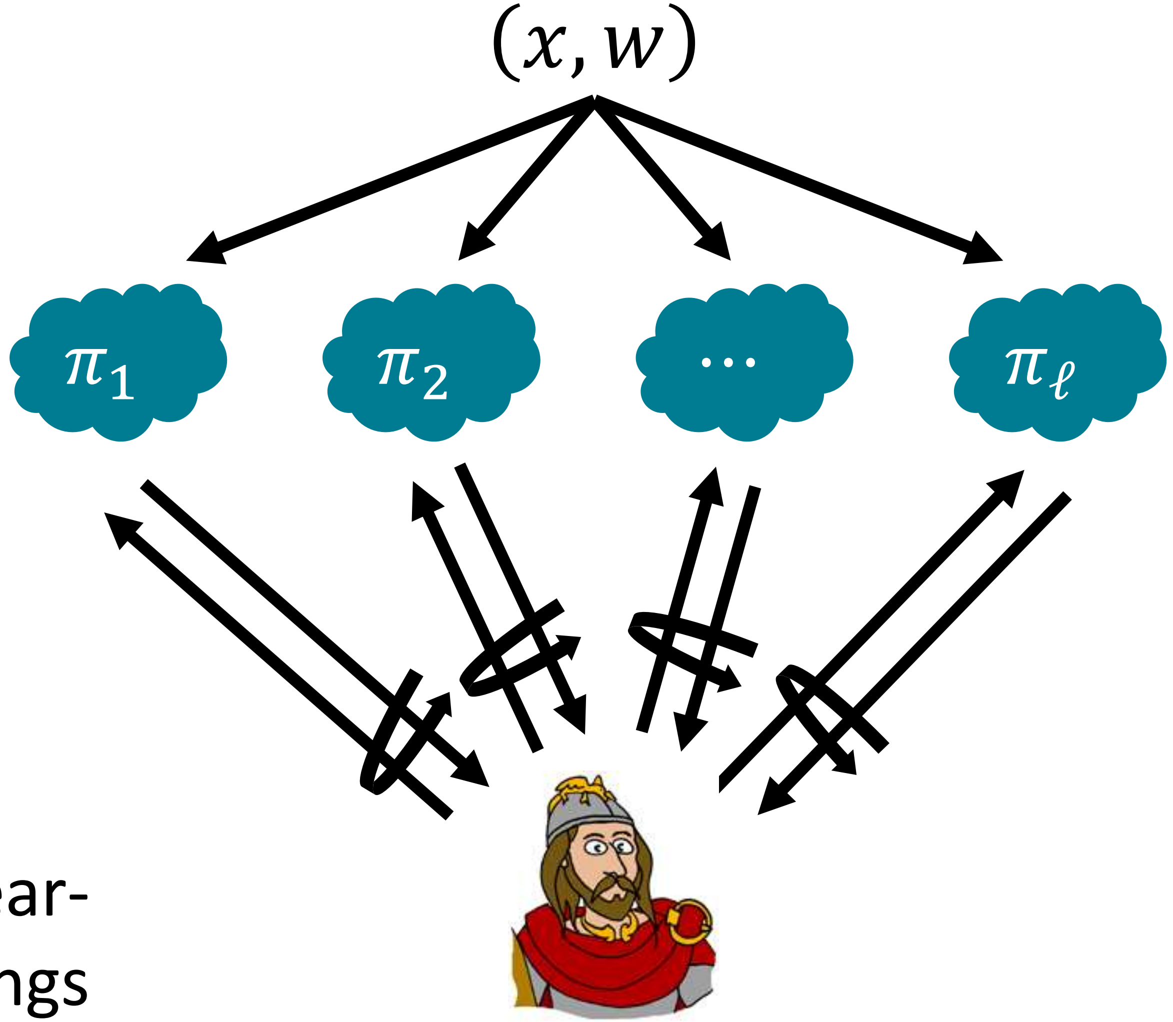
Linear-Only Encryption over Rings



Given encrypted set of query vectors, prover can homomorphically apply <u>independent</u> linear functions to each slot

Key idea: Check <u>multiple</u> independent proofs in <u>parallel</u>

Linear Multi-Prover Interactive Proofs (MIPs)

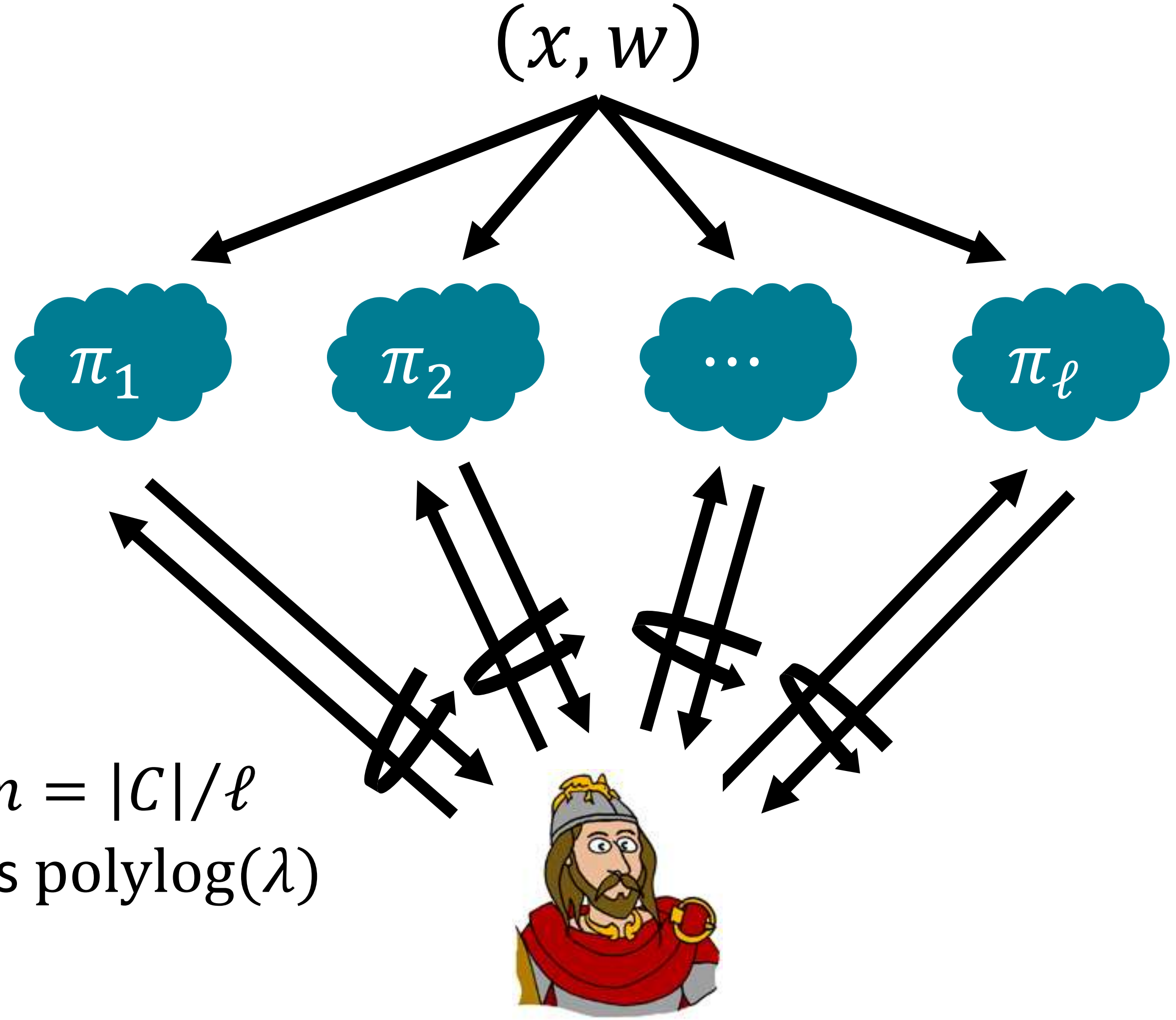


Verifier has oracle access to <u>multiple</u> linear proof oracles

[Proofs may be correlated]

Can convert linear MIP to preprocessing SNARG using linear-only (vector) encryption over rings

Linear Multi-Prover Interactive Proofs (MIPs)



Suppose

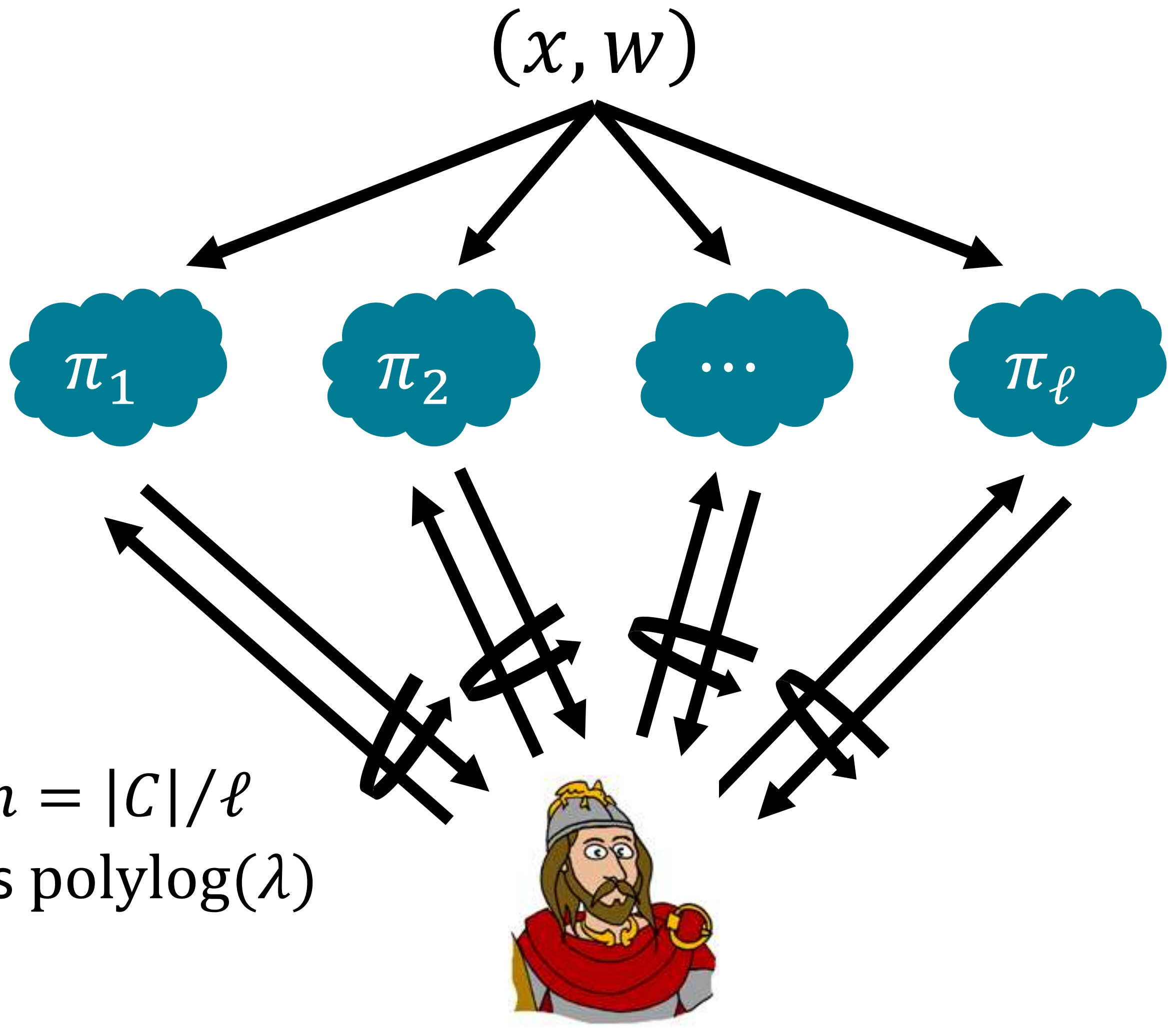
- Number of provers $\ell = \tilde{O}(\lambda)$
- Proofs $\pi_1, \ldots, \pi_\ell \in \mathbb{F}_p^m$ where $m = |C|/\ell$
- Number of queries to each π_i is $\text{polylog}(\lambda)$

Then, linear MIP is quasi-optimal

Linear Multi-Prover Interactive Proofs (MIPs)

Prover complexity:
$\tilde{O}(\ell m) = \tilde{O}(|C|)$

Linear MIP size:
$O(\ell \cdot \mathrm{polylog}(\lambda)) = \tilde{O}(\lambda)$



Suppose

- Number of provers $\ell = \tilde{O}(\lambda)$
- Proofs $\pi_1, \ldots, \pi_\ell \in \mathbb{F}_p^m$ where $m = |C|/\ell$
- Number of queries to each π_i is $\mathrm{polylog}(\lambda)$

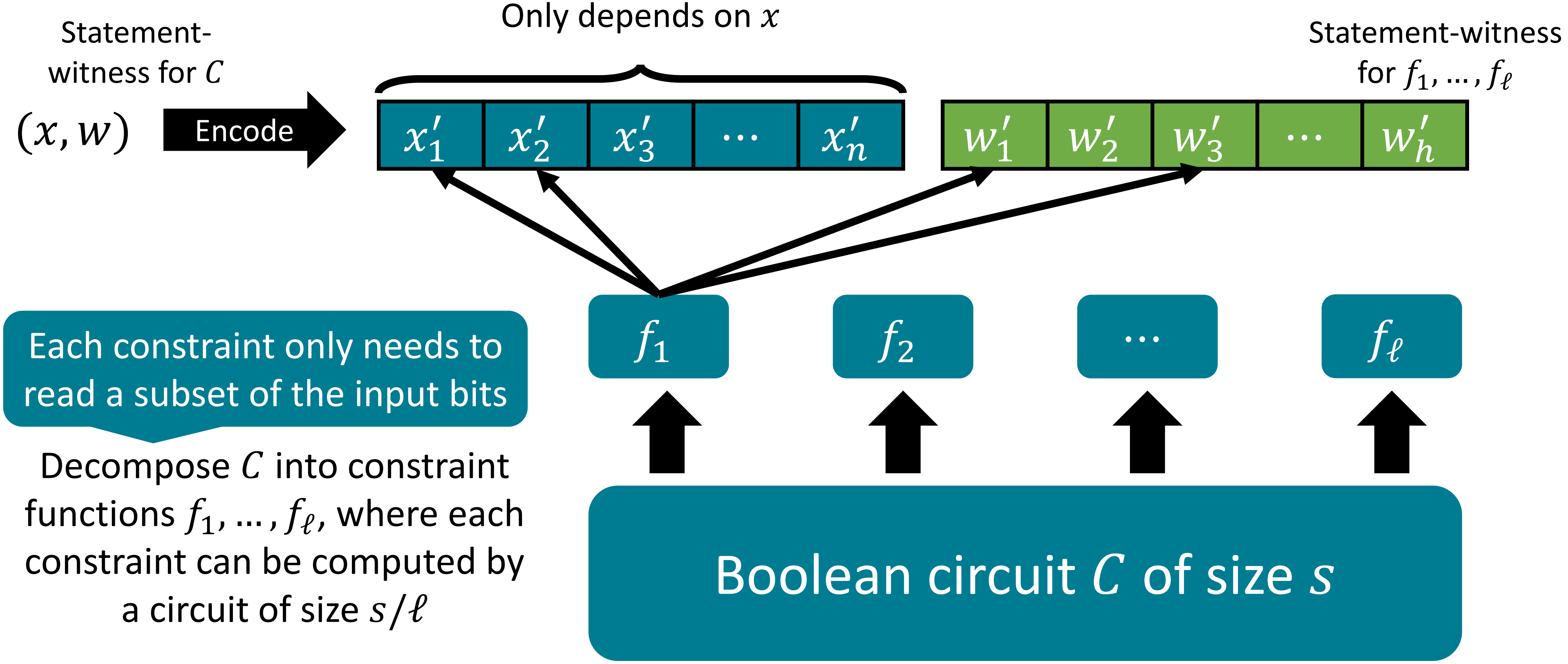
Then, linear MIP is quasi-optimal

Quasi-Optimal Linear MIPs

This work: Construction of a quasi-optimal linear MIP for Boolean circuit satisfiability

Robust Decomposition → Consistency Check → Quasi-Optimal Linear MIP

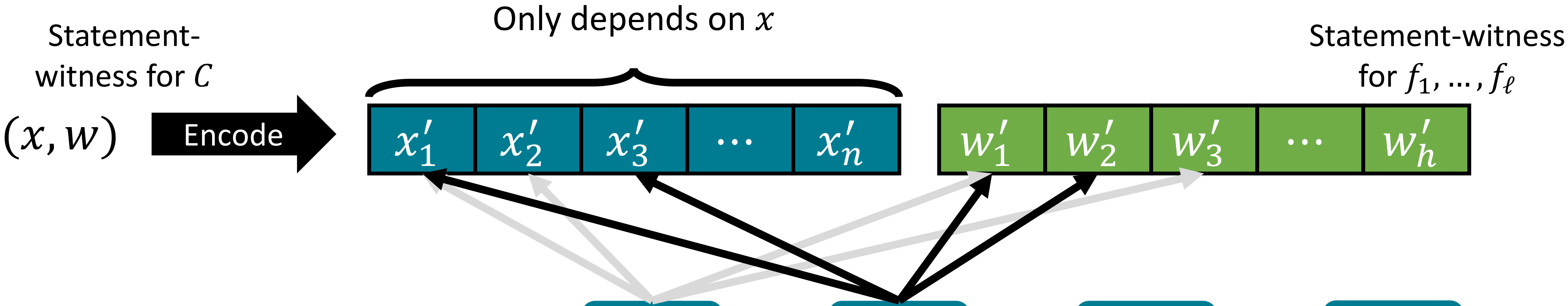
Robust Decomposition



Each constraint only needs to read a subset of the input bits

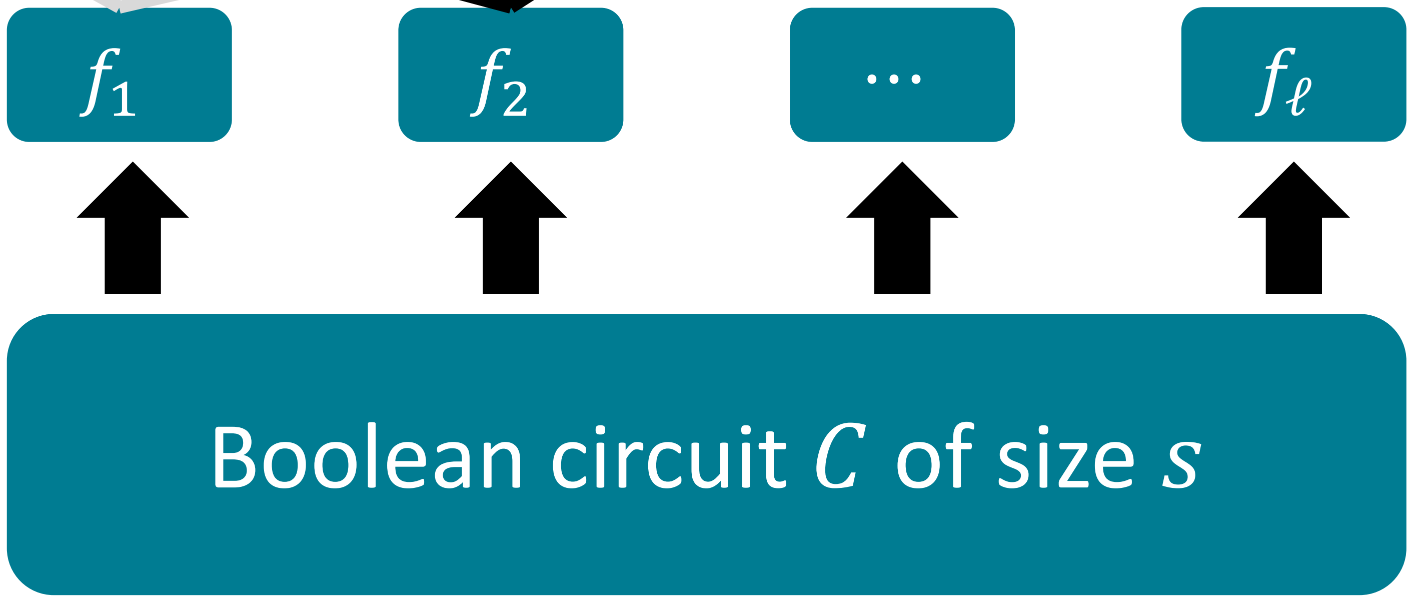
Decompose C into constraint functions $f_1, \dots, f_\ell$, where each constraint can be computed by a circuit of size s/ℓ

Robust Decomposition

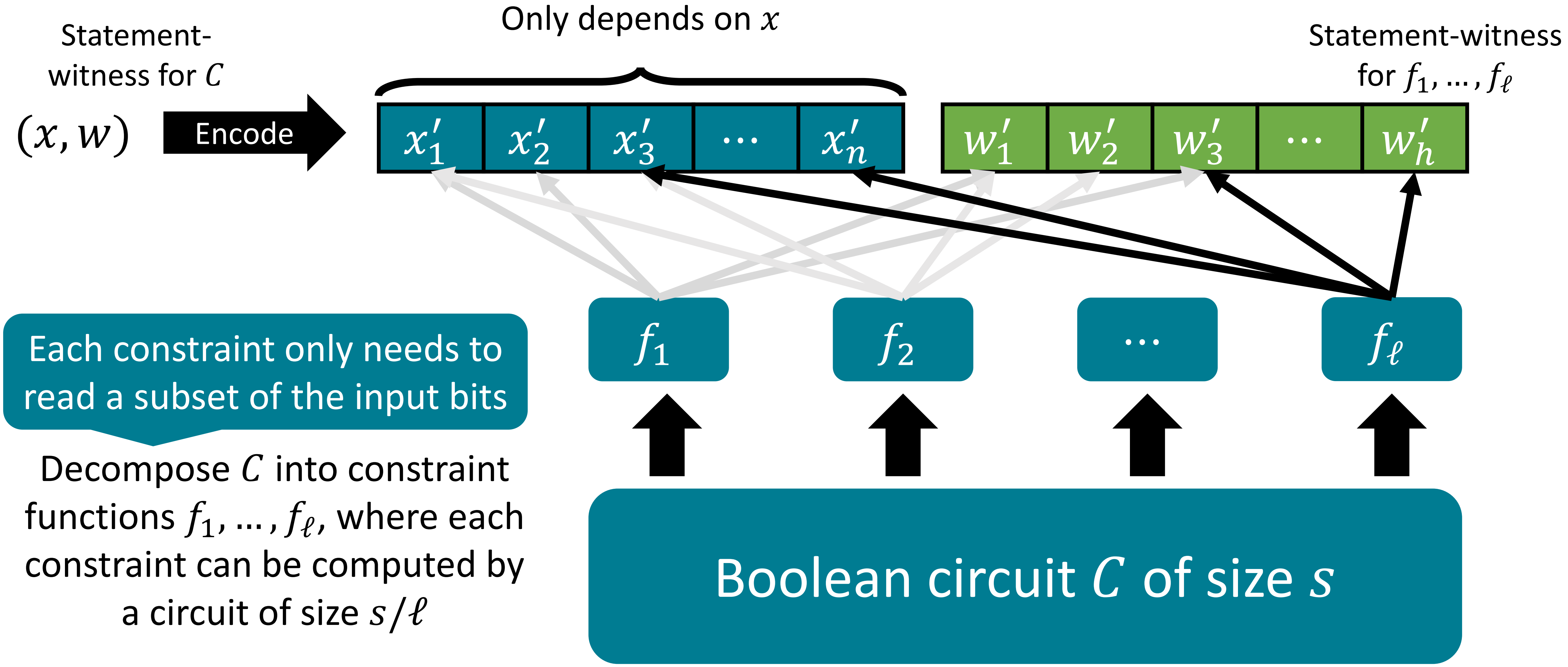


Each constraint only needs to read a subset of the input bits

Decompose C into constraint functions $f_1, \ldots, f_\ell$, where each constraint can be computed by a circuit of size s/ℓ

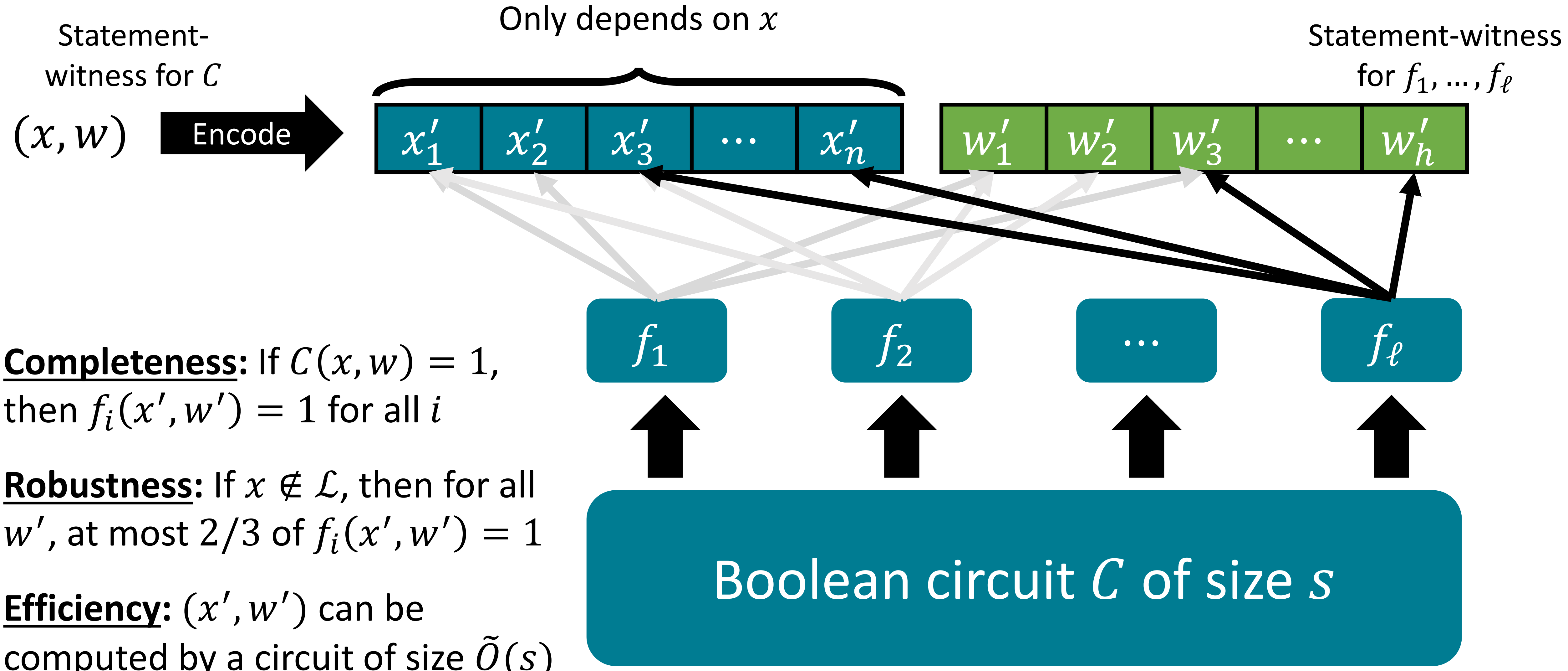


Robust Decomposition



Decompose C into constraint functions $f_1, \ldots, f_\ell$, where each constraint can be computed by a circuit of size s/ℓ

Robust Decomposition



Completeness: If $C(x, w) = 1$, then $f_i(x', w') = 1$ for all i

Robustness: If $x \notin \mathcal{L}$, then for all w', at most 2/3 of $f_i(x', w') = 1$

Efficiency: (x', w') can be computed by a circuit of size $\tilde{O}(s)$

Robust Decomposition

Boolean circuit C of size s

f_1 → π_1

f_2 → π_2

$\vdots$ → $\vdots$

f_ℓ → π_ℓ

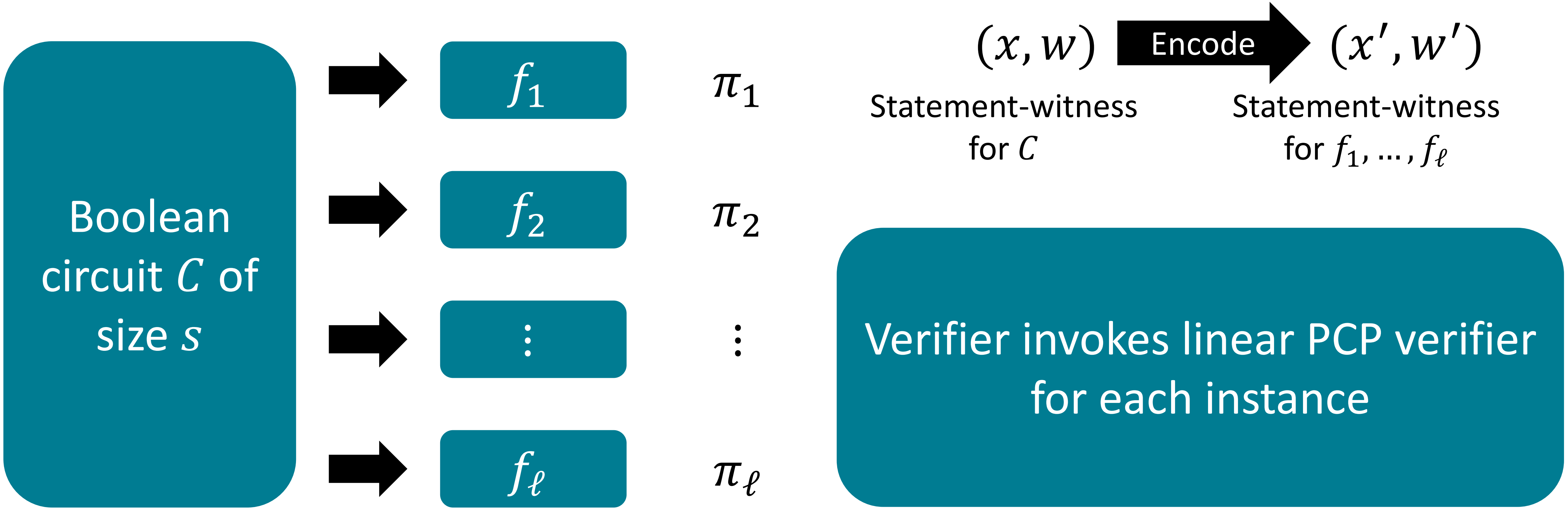
(x, w) **Encode** → (x', w')

Statement-witness for C → Statement-witness for $f_1, \dots, f_\ell$

Using linear PCP based on QSPs [GGPR13], $|\pi_i| = O(|C|/\ell)$ and provides soundness $1/\text{poly}(\lambda)$

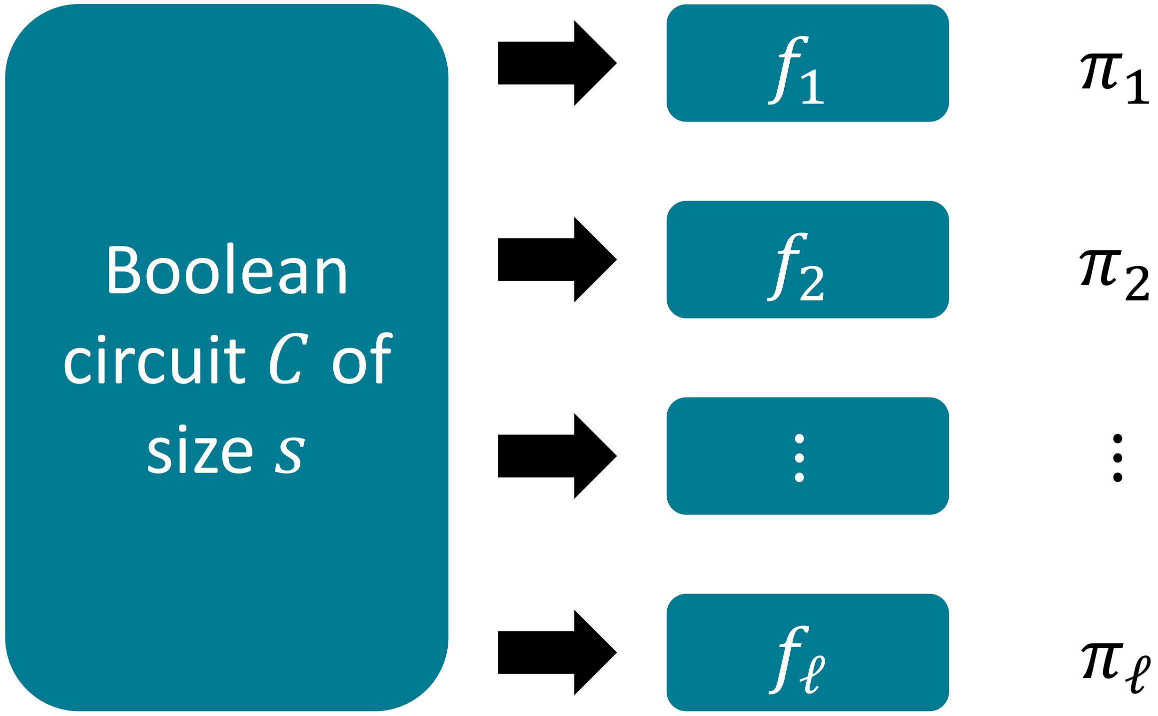
π_i: linear PCP that $f_i(x', \cdot)$ is satisfiable
(instantiated over $\mathbb{F}_p$ where $p = \text{poly}(\lambda)$)

Robust Decomposition



π_i: linear PCP that $f_i(x', \cdot)$ is satisfiable
(instantiated over $\mathbb{F}_p$ where $p = \text{poly}(\lambda)$)

Robust Decomposition



Completeness: Follows by completeness of decomposition and linear PCPs

Soundness: Each linear PCP provides $1/\text{poly}(\lambda)$ soundness and for false statement, at least $1/3$ of the statements are false, so if $\ell = \Omega(\lambda)$, verifier accepts with probability $2^{-\Omega(\lambda)}$

π_i: linear PCP that $f_i(x', \cdot)$ is satisfiable
(instantiated over $\mathbb{F}_p$ where $p = \text{poly}(\lambda)$)

Robust Decomposition

Robustness: If $x \notin \mathcal{L}$, then for all w', at most $2/3$ of $f_i(x', w') = 1$

For false x, no single w' can simultaneously satisfy $f_i(x', \cdot)$; however, all of the $f_i(x', \cdot)$ could individually be satisfiable

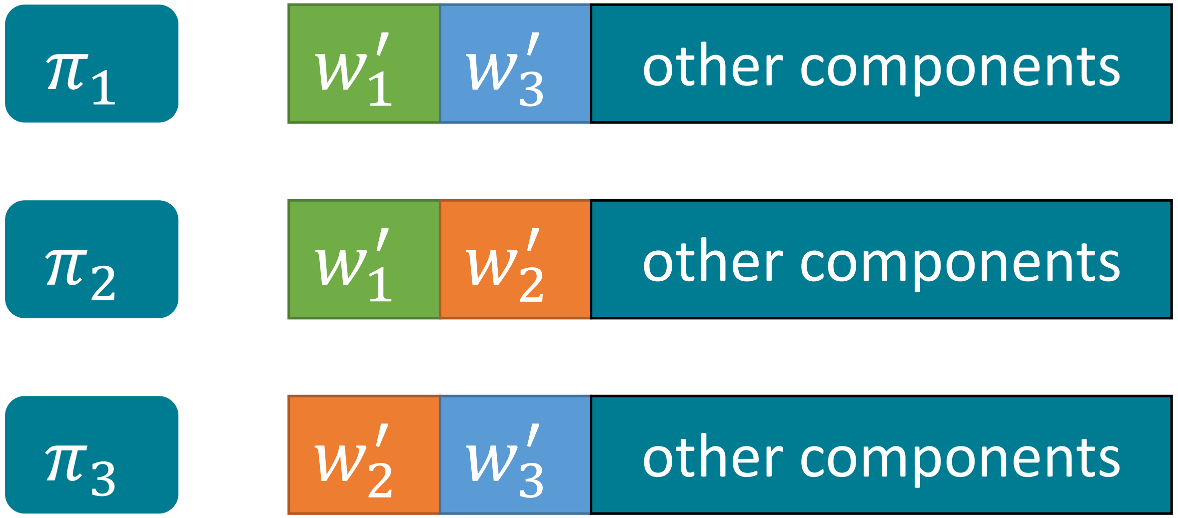
Completeness: Follows by completeness of decomposition and linear PCPs

Soundness: Each linear PCP provides $1/\text{poly}(\lambda)$ soundness and for false statement, at least $1/3$ of the statements are false, so if $\ell = \Omega(\lambda)$, verifier accepts with probability $2^{-\Omega(\lambda)}$

Problematic however if prover uses different (x', w') to construct proofs for different f_i's

Consistency Checking

Require that linear PCPs are systematic: linear PCP π contains a copy of the witness:



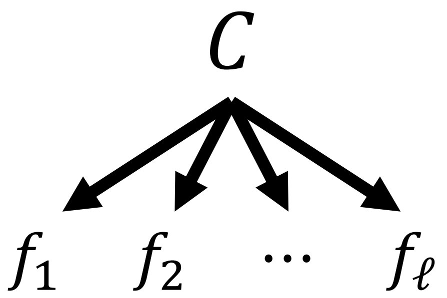
Goal: check that assignments to w' are consistent via linear queries to π_i

First few components of proof correspond to witness associated with the statement

→ Each proof induces an assignment to a few bits of the common witness w'

Quasi-Optimal Linear MIP

Robust Decomposition

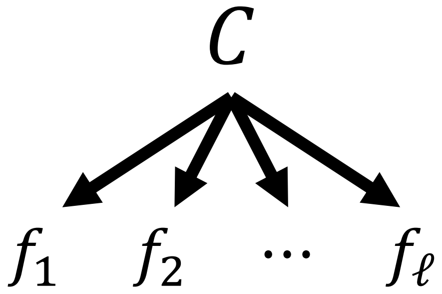


- Checking satisfiability of C corresponds to checking satisfiability of $f_1, \dots, f_\ell$ (each of which can be checked by a circuit of size $|C|/\ell$)
- For a false statement, no single witness can simultaneously satisfy more than a constant fraction of f_i

Robust decomposition can be instantiated by combining "MPC-in-the-head" paradigm [IKOS07] with a robust MPC protocol with polylogarithmic overhead [DIK10]

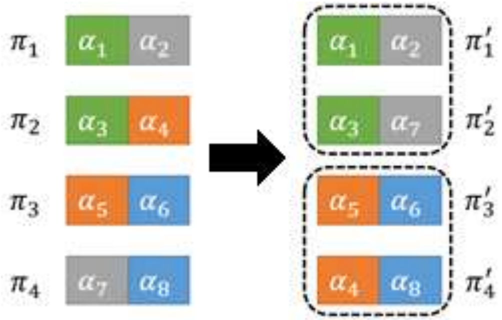
Quasi-Optimal Linear MIP

Robust Decomposition



- Checking satisfiability of C corresponds to checking satisfiability of $f_1, \dots, f_\ell$ (each of which can be checked by a circuit of size $|C|/\ell$)
- For a false statement, no single witness can simultaneously satisfy more than a constant fraction of f_i

Consistency Check



- Check that consistent witness is used to prove satisfiability of each f_i
- Relies on pairwise consistency checks and permuting the entries to obtain a "nice" replication structure

Asymptotic Comparisons

Construction	Prover Complexity	Proof Size	Assumption
CS Proofs [Mic94]	$\tilde{O}(\lvert C \rvert)$	$\tilde{O}(\lambda^2)$	Random Oracle
Groth [Gro16]	$\tilde{O}(\lambda \lvert C \rvert)$	$\tilde{O}(\lambda)$	Generic Group
Groth [Gro10]	$\tilde{O}(\lambda \lvert C \rvert^2 + \lvert C \rvert \lambda^2)$	$\tilde{O}(\lambda)$	Knowledge of Exponent
GGPR [GGPR12]	$\tilde{O}(\lambda \lvert C \rvert)$	$\tilde{O}(\lambda)$	
BCIOP (Pairing) [BCIOP13]	$\tilde{O}(\lambda \lvert C \rvert)$	$\tilde{O}(\lambda)$	Linear-Only Encryption
BISW (integer lattices) [BISW17]	$\tilde{O}(\lambda \lvert C \rvert)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption
BISW (ideal lattices) [BISW18]	$\tilde{O}(\lvert C \rvert)$	$\tilde{O}(\lambda)$	Linear-Only Vector Encryption

For simplicity, we ignore low order terms $\text{poly}(\lambda, \log \lvert C \rvert)$ in the prover complexity

Conclusions

Introduced framework for building SNARGs by combining linear PCPs (and linear MIPs) with linear-only vector encryption

Framework yields first quasi-optimal SNARG by combining quasi-optimal linear MIP with linear-only vector encryption

- Construction of a quasi-optimal linear MIP possible by combining robust decomposition and consistency check

Open Problems

Publicly-verifiable SNARGs from lattices

Quasi-optimal zero-knowledge SNARGs

Concrete efficiency of lattice-based SNARGs

Thank you!

https://cs.stanford.edu/~dwu4/snargs-project.html