Lattice-Based Non-Interactive Argument Systems

David Wu
Stanford University

Based on joint works with Dan Boneh, Yuval Ishai, Sam Kim, and Amit Sahai
Proof Systems and Argument Systems

Language $\mathcal{L} \subseteq \{0,1\}^*$

Completeness:
$\forall x \in \mathcal{L} : \Pr[P, V](x) = \text{accept} = 1$
“Honest prover convinces honest verifier of true statements”

Soundness:
$\forall x \notin \mathcal{L}, \forall P^* : \Pr[P^*, V](x) = \text{accept} = 0$
“No prover can convince honest verifier of false statement”

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

Accept if $x \in \mathcal{L}$
Proof Systems and Argument Systems

Language $\mathcal{L} \subseteq \{0,1\}^*$

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

“Honest prover convinces honest verifier of true statements”

Soundness: $\forall x \notin \mathcal{L}, \forall P^* : \Pr[\langle P^*, V \rangle(x) = \text{accept}] = 0$

“No prover can convince honest verifier of false statement”

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

**NP** – the class of languages that are *efficiently verifiable*

A language $\mathcal{L}$ is in **NP** if there exists a polynomial-time verifier $R$ such that

$$x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)} \ R(x, w) = 1$$

**Statement**

**Witness**
**The Complexity Class NP**

**NP** – the class of languages that are *efficiently verifiable*

A language $\mathcal{L}$ is in **NP** if there exists a polynomial-time verifier $R$ such that

$$x \in \mathcal{L} \iff \exists w \in \{0,1\}^{\text{poly}(|x|)} \ R(x, w) = 1$$

In this talk, will focus on language of Boolean circuit satisfiability:

$$\mathcal{L}_C = \{x : C(x, w) = 1 \text{ for some } w\}$$
Non-Interactive Proof Systems for NP

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

NP languages have non-interactive proof systems

But what if we want other properties?
Non-Interactive Proof Systems for NP

Zero-Knowledge: The proof reveals nothing more about the statement $x$ other than $x \in \mathcal{L}_C$ [GMR85]

- Fundamental primitive to modern cryptography
- Important building block in many protocols (e.g., identification schemes, digital signatures, multiparty computation)

Succinctness: The proof is significantly shorter than $|C|$ (and correspondingly, $|w|$) [Kil92, Mic00, GW11]

- Natural complexity-theoretic question: what is the minimal communication complexity for proofs of NP statements?
- Numerous applications to delegating and verifying computations as well as privacy-preserving cryptocurrencies

But what if we want other properties?
The Landscape of Modern Cryptography

Cryptography is the study of hardness

[Slide inspired by Amit Sahai]
Which assumptions imply non-interactive zero-knowledge?

Which assumptions imply succinct non-interactive arguments?
The Landscape of Modern Cryptography

Which assumptions imply non-interactive zero-knowledge?

Which assumptions imply succinct non-interactive arguments?
This Work

Which assumptions imply non-interactive zero-knowledge?

* In a weaker preprocessing model

Which assumptions imply succinct non-interactive arguments?
This Work

Which assumptions imply non-interactive zero-knowledge?

Non-interactive zero-knowledge arguments from standard lattice assumptions in a preprocessing model [Kim-W; CRYPTO 2018]

Which assumptions imply succinct non-interactive arguments?

Succinct non-interactive arguments (SNARGs) from lattice-based assumptions [Boneh-Ishai-Sahai-W; EUROCRYPT 2017]

First construction of a quasi-optimal SNARG from lattice-based assumptions [Boneh-Ishai-Sahai-W; EUROCRYPT 2018]

Focus of this talk
Why Lattices?

(Conjectured) post-quantum resilience
Diversifying cryptographic assumptions
Enable new properties (e.g., quasi-optimality)
Succinct Non-Interactive Arguments
Succinct Non-Interactive Arguments (SNARGs)

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

Completeness:

"Honest prover convinces honest verifier of true statements"
Succinct Non-Interactive Arguments (SNARGs)

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

Completeness: \[ C(x, w) = 1 \Rightarrow \Pr[V(x, P(x, w)) = 1] = 1 \]

Soundness: “No efficient prover can convince honest verifier of false statement”
Succinct Non-Interactive Arguments (SNARGs)

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

Completeness: 
\[ C(x, w) = 1 \Rightarrow \Pr[V(x, P(x, w)) = 1] = 1 \]

Soundness: 
for all provers \( P^* \) of size \( 2^\lambda \) (\( \lambda \) is a security parameter),
\[ x \notin \mathcal{L}_C \Rightarrow \Pr[V(x, P^*(x)) = 1] \leq 2^{-\lambda} \]
Succinct Non-Interactive Arguments (SNARGs)

\[ L_C = \{ x : C(x, w) = 1 \text{ for some } w \} \]

Argument system is **succinct** if:
- Prover 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

\[ \pi = P(x, w) \]

\[ \text{accept if } V(x, \pi) = 1 \]
Succinct Non-Interactive Arguments (SNARGs)

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

Proof system is succinct if:
- Prover 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 unlikely to exist in the standard model [BP04, Wee05]
Succinct Non-Interactive Arguments (SNARGs)

Instantiation: “CS proofs” in the random oracle model

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

(prover)

(random oracle (\(\mathcal{RO}\))

(\(x, w\))

(\(x\))

(\(\pi\))

(\(P^{\mathcal{RO}}\))

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

(verifier)

Argument consists of a single message

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

accept if \(V^{\mathcal{RO}}(x, \pi) = 1\)
Succinct Non-Interactive Arguments (SNARGs)

Preprocessing SNARGs: allow “expensive” setup

common reference string (CRS)

Setup\((1^\lambda)\)

\(\sigma\)

\(\tau\)

verification state

prover

\((x, w)\)

verifier

can consider publicly-verifiable and secretly-verifiable SNARGs

\[\pi = P(\sigma, x, w)\]

Argument consists of a single message

accept if \(V(\tau, x, \pi) = 1\)
Complexity Metrics for SNARGs

**Soundness:** for all provers $P^*$ of size $2^\lambda$:

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

**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 $P^*$ of size $2^\lambda$:

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

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

- Quasi-optimal succinctness:
  $$|\pi| = \lambda \cdot \text{polylog}(\lambda, |C|) = \tilde{O}(\lambda)$$

- Quasi-optimal prover complexity:
  $$|P| = \tilde{O}(|C|) + \text{poly}(\lambda, \log|C|)$$
# Asymptotic Comparisons

<table>
<thead>
<tr>
<th>Construction</th>
<th>Prover Complexity</th>
<th>Proof Size</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Proofs [Mic94]</td>
<td>$\tilde{O}(</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>Groth [Gro16]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>Groth [Gro10]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>^2 +</td>
</tr>
<tr>
<td>GGPR [GGPR12]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>BCIOP (Pairing) [BCIOP13]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td><strong>This work</strong> (over integer lattices)</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td><strong>This work</strong> (over ideal lattices)</td>
<td>$\tilde{O}(</td>
<td>C</td>
<td>)$</td>
</tr>
</tbody>
</table>

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

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

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

In these instantiations, verifier is oblivious (queries independent of statement)

 verifier

\( q \in \mathbb{F}^m \)  \( \langle q, \pi \rangle \in \mathbb{F} \)

accept/reject

\( (x, w) \)
From Linear PCPs to SNARGs

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

\[ Q = \langle \pi, q_1 \rangle \langle \pi, q_2 \rangle \cdots \langle \pi, q_k \rangle \]

Prover constructs linear PCP \( \pi \) from \((x, w)\)

\( \pi \in \mathbb{F}^m \)

Prover computes responses to linear PCP queries

\[ \langle \pi, q_1 \rangle \quad \langle \pi, q_2 \rangle \quad \cdots \quad \langle \pi, q_k \rangle \]

SNARG proof

[BCIOP13]
From Linear PCPs to SNARGs

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

\[ Q = \langle \pi, q_1 \rangle \langle \pi, q_2 \rangle \cdots \langle \pi, q_k \rangle \]

Two issues:
- Malicious prover can choose \( \pi \) based on the queries
- Malicious prover can apply different \( \pi \) to each query

Prover computes responses to linear PCP queries

SNARG proof
From Linear PCPs to SNARGs

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

\[ Q = \langle \pi, q_1 \rangle \langle \pi, q_2 \rangle \langle \pi, q_3 \rangle \cdots \langle \pi, q_k \rangle \]

Prover computes responses to linear PCP queries

SNARG proof

Two issues:
- Malicious prover can choose \( \pi \) based on the queries
- Malicious prover can apply different \( \pi \) to each query
From Linear PCPs to SNARGs

Two issues:
- Malicious prover can choose $\pi$ based on the queries
- Malicious prover can apply different $\pi$ to each query

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

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

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
Two issues:

• Malicious prover can choose \( \pi \) based on the queries
• Malicious prover can apply different \( \pi \) 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

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

\[ Q = [q_1, q_2, q_3, \ldots, q_k] \]
Two issues:
• Malicious prover can choose $\pi$ based on the queries
• Malicious prover can apply different $\pi$ 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

\[ Q = [q_1, q_2, q_3, \ldots, q_k] \]

- 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

\[ \nu_1 \in \mathbb{F}^k \]
\[ \nu_2 \in \mathbb{F}^k \]
\[ \vdots \]
\[ \nu_m \in \mathbb{F}^k \]

plaintext space is a vector space
Linear-Only Vector Encryption

\[ v_1 \in \mathbb{F}^k \]
\[ v_2 \in \mathbb{F}^k \]
\[ \vdots \]
\[ v_m \in \mathbb{F}^k \]

plaintext space is a \textit{vector} space

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

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 $(\alpha_1, \ldots, \alpha_m) \in \mathbb{F}^m$ and $b \in \mathbb{F}^k$ such that $\text{Decrypt}(sk, ct) = \sum_{i \in [n]} \alpha_i v_i + b$

[Weaker property also suffices]
From Linear PCPs to SNARGs

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

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

Prover constructs linear PCP \( \pi \) from \((x, w)\)

\[ \langle \pi, q_1 \rangle \langle \pi, q_2 \rangle \cdots \langle \pi, q_k \rangle \]

Prover computes responses to linear PCP queries

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
**Conjecture**: Regev encryption (specifically, variant of the [PVW08] scheme) based on lattices is a linear-only vector encryption scheme.
Complexity of the Construction

Prover constructs linear PCP $\pi$ from $(x, w)$

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

Prover computes responses to linear PCP queries

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

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

$x, w \in \mathbb{F}^m$

SNARG proof
## Asymptotic Comparisons

<table>
<thead>
<tr>
<th>Construction</th>
<th>Prover Complexity</th>
<th>Proof Size</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Proofs [Mic94]</td>
<td>$\tilde{O}(</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>Groth [Gro16]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>Groth [Gro10]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>^2 +</td>
</tr>
<tr>
<td>GGPR [GGPR12]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>BCIOP (Pairing) [BCIOP13]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td><strong>This work</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(over integer lattices)</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
</tbody>
</table>

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

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

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

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

Prover constructs linear PCP $\pi$ from $(x, w)$

$$(x, w)$$

We pay $\Omega(\lambda)$ for each homomorphic operation. Can we reduce this?
Consider encryption scheme over a polynomial ring $R_p = \mathbb{Z}_p[x]/\Phi_\ell(x) \cong \mathbb{F}_p^\ell$

Plaintext space can be viewed as a vector of field elements

Homomorphic operations correspond to component-wise additions and scalar multiplications

Using RLWE-based encryption schemes, can encrypt $\ell = \tilde{O}(\lambda)$ field elements ($p = \text{poly}(\lambda)$) with ciphertexts of size $\tilde{O}(\lambda)$
Consider encryption scheme over a polynomial ring $R_p = \mathbb{Z}_p[x]/\Phi_\ell(x) \cong \mathbb{F}_p^\ell$

Plaintext space can be viewed as a vector of field elements

Homomorphic operations

Amortized cost of homomorphical operation on a single field element is polylog($\lambda$)

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 independent linear functions to each slot.

**Key idea:** Check multiple independent proofs in parallel.
Linear Multi-Prover Interactive Proofs (MIPs)

Verifier has oracle access to multiple linear proof oracles
[Proofs may be correlated]

Can convert linear MIP to preprocessing SNARG using linear-only (vector) encryption over rings
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 \( \pi_i \) is \( \text{polylog}(\lambda) \)

Then, linear MIP is quasi-optimal
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 $\pi_i$ is $\text{polylog}(\lambda)$

Then, linear MIP is quasi-optimal

Prover complexity:

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

Linear MIP size:

$O(\ell \cdot \text{polylog}(\lambda)) = \tilde{O}(\lambda)$
Quasi-Optimal Linear MIPs

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

(\(x, w\))

\(x'_1, x'_2, x'_3, \ldots, x'_n, w'_1, w'_2, w'_3, \ldots, w'_h\)

Statement-witness for \(C\)

Only depends on \(x\)

Statement-witness for \(f_1, \ldots, f_\ell\)

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 \(s/\ell\)

Boolean circuit \(C\) of size \(s\)
Robust Decomposition

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

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

Statement-witness for $C$ $(x, w)$ Encode

Statement-witness for $f_1, \ldots, f_\ell$
Robust Decomposition

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

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

Statement-witness for $C$ $(x, w)$

Statement-witness for $f_1, \ldots, f_\ell$

Encode $x'_1, x'_2, x'_3, \ldots, x'_n$ $w'_1, w'_2, w'_3, \ldots, w'_h$

Boolean circuit $C$ of size $s$
Robust Decomposition

Statement-witness for $C$
$(x, w)$ Encode

$x'_1 \ x'_2 \ x'_3 \ \ldots \ x'_n$

Statement-witness for $f_1, \ldots, f_\ell$

$f_1 \ f_2 \ \ldots \ f_\ell$

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 $\tilde{O}(s)$

Boolean circuit $C$ of size $s$
Robust Decomposition

Boolean circuit $C$ of size $s$

$f_1 \rightarrow \pi_1$

$f_2 \rightarrow \pi_2$

$\vdots$

$f_\ell \rightarrow \pi_\ell$

$(x, w)$：声明-证明（$C$的声明-证明）

$(x', w')$：声明-证明（$f_1, \ldots, f_\ell$的声明-证明）

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

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

Boolean circuit $C$ of size $s$

- $f_1$
- $f_2$
- $\vdots$
- $f_\ell$

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

Verifier invokes linear PCP verifier for each instance

Statement-witness for $C$

Statement-witness for $f_1, \ldots, f_\ell$

Encode

$(x, w)$

$(x', w')$
Robust Decomposition

Boolean circuit $C$ of size $s$

$\pi_1$: linear PCP that $f_1(x', \cdot)$ is satisfiable (instantiated over $\mathbb{F}_p$ where $p = \text{poly}(\lambda)$)

$\pi_2$: linear PCP that $f_2(x', \cdot)$ is satisfiable (instantiated over $\mathbb{F}_p$ where $p = \text{poly}(\lambda)$)

$\vdots$

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

**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 $2^{-\Omega(\lambda)}$
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 $1/poly(\lambda)$ 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 $\pi$ contains a copy of the witness:

$\pi_1$  
$w'_1 \quad w'_3$  
other components

$\pi_2$  
$w'_1 \quad w'_2$  
other components

$\pi_3$  
$w'_2 \quad w'_3$  
other components

**Goal**: check that assignments to $w'$ are consistent via linear queries to $\pi_i$

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 $w'$
Robust decomposition can be instantiated by combining “MPC-in-the-head” paradigm [IKOS07] with a robust MPC protocol with polylogarithmic overhead [DIK10]

- Checking satisfiability of $C$ corresponds to checking satisfiability of $f_1, ..., f_\ell$ (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

\[ C \]

\[ f_1 \quad f_2 \quad \cdots \quad f_\ell \]

- Checking satisfiability of \( C \) corresponds to checking satisfiability of \( f_1, \ldots, f_\ell \) (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
<table>
<thead>
<tr>
<th>Construction</th>
<th>Prover Complexity</th>
<th>Proof Size</th>
<th>Assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS Proofs [Mic94]</td>
<td>$\tilde{O}(</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>Groth [Gro10]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>Groth [Gro16]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>^2 +</td>
</tr>
<tr>
<td>GGPR [GGPR12]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>BCIOP (Pairing) [BCIOP13]</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>This work (over integer lattices)</td>
<td>$\tilde{O}(\lambda</td>
<td>C</td>
<td>)$</td>
</tr>
<tr>
<td>This work (over ideal lattices)</td>
<td>$\tilde{O}(</td>
<td>C</td>
<td>)$</td>
</tr>
</tbody>
</table>

For simplicity, we ignore low order terms $\text{poly}(\lambda, \log |C|)$ in the prover complexity.
A SNARG is quasi-optimal if it satisfies the following properties:

- Quasi-optimal succinctness: $|\pi| = \tilde{O}(\lambda)$
- Quasi-optimal prover complexity: $|P| = \tilde{O}(|C|) + \text{poly}(\lambda, \log|C|)$

New 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.
Which assumptions imply non-interactive zero-knowledge?

Which assumptions imply succinct non-interactive arguments?
**Summary**

*Which assumptions imply non-interactive zero-knowledge?*

*In a weaker preprocessing model*

*Which assumptions imply succinct non-interactive arguments?*
Acknowledgments

Special thanks to all of my amazing collaborators!