Succinct Arguments From Linear Interactive Proofs

Nir Bitansky  Alessandro Chiesa  Yuval Ishai

Rafail Ostrovsky  Omer Paneth
Verifying NP Computations Fast

\[
\exists w \text{ s.t. } M(x, w) \text{ accepts in at most } T \text{ steps?}
\]

\[
(M, x, T)
\]

\[
\begin{align*}
P(w) &\quad \vdash \quad V \\
\text{efficiency: } \quad \text{TIME}(P) &= T^2 \\
\text{ TIME}(V) &= (|M| + |x| + \log T)^2
\end{align*}
\]

- completeness: \((M, x, T) \in L_U \rightarrow \Pr[\langle P(w), V_{(M,x,T)} \rangle = 1] = 1\)
- soundness: \((M, x, T) \not\in L_U \rightarrow \Pr[\langle P^*, V_{(M,x,T)} \rangle = 1] = \text{small for every } P^*\)
Verifying NP Computations Fast

\[ \exists \, w \text{ s.t. } M(x, w) \text{ accepts in at most } T \text{ steps?} \]

\[ (M, x, T) \]

\[ P(w) \quad : \quad V \quad 0/1 \]

efficiency:

\[ \text{TIME}(P) = k^3 T^2 \]
\[ \text{TIME}(V) = k^4 (|M| + |x| + \log T)^2 \]

- completeness: \((M, x, T) \in L_U \rightarrow \Pr[\langle P(w), V(M,x,T) \rangle = 1] = 1\)

- soundness: \((M, x, T) \notin L_U \rightarrow \Pr[\langle P^*, V(M,x,T) \rangle = 1] = \text{negl}(k)\)

for every poly\((k)\)-size \(P^*\)

[BHZ, GH, GVW, Wee]
Verifying NP Computations Fast

\[ (M, x, T) \]

\[ \exists w \text{ s.t. } M(x, w) \text{ accepts in at most } T \text{ steps?} \]

\[ P(w) \quad \vdash \quad V \quad 0/1 \]

**SUCCINCT ARGUMENT**

**efficiency:**

\[ \text{TIME}(P) = k^3 T^2 \]
\[ \text{TIME}(V) = k^4 (|M| + |x| + \log T)^2 \]

- completeness: \((M, x, T) \in L_U \rightarrow \Pr[\langle P(w), V_{(M,x,T)} \rangle = 1] = 1\]
- soundness: 
  \((M, x, T) \notin L_U \rightarrow \Pr[\langle P^*, V_{(M,x,T)} \rangle = 1] = \text{negl}(k)\)
  for every poly\((k)\)-size \(P^*\)

[BHZ, GH, GVW, Wee]
WHAT KINDS OF SUCCINCT ARGUMENTS ARE THERE?
[Kilian]  
tools: PCP system + collision-resistant hashing  
1 offline message  
3 online messages  

[Micali]  
apply Fiat-Shamir paradigm in the Random Oracle model  
1 non-interactive & publicly-verifiable message
CAN WE REDUCE
# ONLINE MESSAGES
W/O RANDOM ORACLES?
Succinct Non-Interactive Arguments (SNARGs)
Succinct Non-Interactive Arguments (SNARGs)

\[ G \]

proving key \( \sigma \)

verification key \( \tau \)

have privately-verifiable constructions under relatively-clean (albeit non-falsifiable) assumptions

[DL] [Mie] [BCCTa] [DFH] [GLR] [BC]
Probabilistic Checking & Succinct Arguments

A VERY PRODUCTIVE PARADIGM

**Step 1:** information-theoretic probabilistic checking, in a model where the prover is restricted in some form

**Step 2:** use cryptography to force the restriction

**EXAMPLES**

- Step 1 = design a PCP
- Step 2 = force prover to commit to a PCP

- Step 1 = design a no-signaling MIP
- Step 2 = force prover to act as no-signaling provers

- Step 1 = design an MIP
- Step 2 = force prover to act as non-communicating provers

...
TODAY: Preprocessing SNARGs

\[ (1^k, T) \]

\[ G \]

proving key \( \sigma \)
verification key \( \tau \)

statement \( (M, x, T) \)
proof \( \pi \)

\[ w \]

\[ P_{\sigma} \]

\[ V_{\tau} \]
Preprocessing SNARGs

- setup work is amortized over MANY proofs

- can obtain public verifiability [Groth,Lipmaa,GGPR]

- lead to constructions w/o expensive preprocessing [BCCTb]
  (provided the SNARG has a natural POK)

step 1: reduce CircuitSAT to algebraic satisfaction problem
step 2: use crypto to succinctly verify the latter

**surprising**: do not seem to rely on probabilistic checking!
OUR CONTRIBUTIONS
THIS WORK

give a general recipe to construct preprocessing SNARGs;
the recipe is a new instantiation of the paradigm

Specifically:

**Step 1:** (information-theoretic)
- design a 2-message linear interactive proof (LIP)

**Step 2:** (cryptographic)
- force prover to act as a linear function

results for Step 1: constructions of succinct LIPs
results for Step 2: compilers for private and public cases

- simpler and more efficient preprocessing SNARGs
- re-interpret previous constructions from new perspective
Linear PCPs

A PCP where the proof oracle is a linear function. Previously used in another instantiation of paradigm:

[IKO]

linear PCP
+ linearity testing

strong
linear PCP
+ function commitment

4-msg NP argument
with small communication
Linear Interactive Proofs (LIPs)

The prover is \textbf{algebraically bounded}: specifically, linear.

\[ P_{x,w} \in \mathbb{F}^{k \times m} \]

\[ q_1, \ldots, q_m \in \mathbb{F} \]

\[ a_1, \ldots, a_k \in \mathbb{F} \]

\[ V(x) \]

(in both completeness and soundness!)

We are interested in LIPs that are \textit{input oblivious}:

\[ V = (Q, D) \text{ s.t.} \]

\[ q_1, \ldots, q_m \]

\[ a_1, \ldots, a_k \]
PRIVATE VERIFICATION

PCP

compiler

Step 1 (information-theoretic)

succinct linear IP

Step 2 (cryptographic)

linear PCP

compiler

compiler

privately-verifiable preprocessing SNARG

[ALMSS]

[GGPR]
Step 2: From LIP To pp SNARG

$(pk, sk) \leftarrow \text{Gen}(1^k)$

$G(1^k, T)$

proving key
\[ \sigma = (pk, c_1, \ldots, c_m) \]

verification key
\[ \tau = (sk, u) \]

$V(\tau, x, \pi)$

$D_x(\cdot, u)$

$P(\sigma, x, w)$

$\text{HomEval}_{pk}(P_{x,w}, c)$

$\hat{c}_1 \leftarrow \text{Dec}_{sk}$

$\vdots$

$\hat{c}_k \leftarrow \text{Dec}_{sk}$

$\hat{r} \leftarrow \text{Dec}_{sk}$
Step 2: From LIP To pp SNARG
(private verification)

Linear Targeted Malleability ($\sim [BSW]$)
encryption scheme that ONLY allows
$F$-linear homomorphic operations
(e.g., knowledge variant of Paillier)

non-falsifiable assumption
(somewhat justified by [GW])

proving key
$\sigma = (pk, c_1, \ldots, c_m)$

verification key
$\tau = (sk, u)$
PUBLIC VERIFICATION

Step 1 (information-theoretic)
Step 2 (cryptographic)

compiler

linear PCP

linear IP

compiler

publicly-verifiable preprocessing SNARG

[ALMSS]

[GGPR]
Step 2: From LIP To pp SNARG (sketch)

\[ G(1^k, T) \xrightarrow{\text{Gen}(1^k)} (pk, \mathcal{\square}) \]

- Proving key: \( \sigma = (pk, c_1, \ldots, c_m) \)
- Verification key: \( \tau = (\mathcal{\square}, \mathcal{\square}) \)

\[ \mathcal{Q} \]

- Encrypted values: \( \text{Enc}_{pk}(Q) \)
- Random value: \( r \)

\[ \text{P}(\sigma, x, w) \xrightarrow{\text{HomEval}_{pk}(P_{x,w}, c)} \text{(same)} \]

\[ \text{V}(\tau, x, \pi) \]

- Decryption: \( \text{Dec}_{sk}(\cdot, u) \)
Step 2: From LIP To pp SNARG (sketch)

\[ G(1^k, T) \]

\[
\begin{array}{ccc}
\text{(pk, )} & \leftarrow & \text{Gen}(1^k) \\
\odot & \vdots & \\
\text{Enc}_\text{pk} & \odot & q_m \\
\odot & \vdots & \\
\text{Enc}_\text{pk} & \odot & \text{Q} \\
\odot & \vdots & \\
\text{Q} & \odot & r \\
\end{array}
\]

proving key \( \sigma = (\text{pk}, c_1, \ldots, c_m) \)

verification key \( \tau = (\text{pk}, \text{Q}) \)

-----------------------------------------------

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

\[ \text{HomEval}_{\text{pk}}(P_{x, w}, c) \]

(same)

\[ V(\tau, x, \pi) \]

what we have:

\[ \hat{c} \quad D_x(\cdot;\cdot) \quad \text{Enc}_{\text{pk}}(u) \]
Step 2: From LIP To pp SNARG (sketch)

\[ G(1^k, T) \xrightarrow{\sigma = (pk, c_1, \ldots, c_m)} \text{Gen}(1^k) \]

- Proving key: \( \sigma = (pk, c_1, \ldots, c_m) \)
- Verification key: \( \tau = (\square, \square) \)

\[ Q \xrightarrow{\text{Enc}_{pk}(u)} \]

\[ P(\sigma, x, w) \xrightarrow{\text{HomEval}_{pk}(P_{x,w}, c)} \]

\( \text{(same)} \)

\[ \hat{c}_1 \rightarrow \hat{c}_k \]

\[ V(\tau, x, \pi) \]

Need to test roots:
\[ \text{HomEval}_{pk}(D_x, \hat{c}, \text{Enc}_{pk}(u)) \in \text{Enc}_{pk}(0)? \]
Step 2: From LIP To pp SNARG (sketch)

\[ G(1^k, T) \xrightarrow{(pk, \mathbb{G})} Gen(1^k) \]

\[
\begin{array}{c}
  c_1 \\
  \vdots \\
  c_m
\end{array} \xrightarrow{\mathbb{G}} 
\]

\[
\begin{array}{c}
  Enc_{pk}(\mathbb{G})
\end{array} \xrightarrow{\mathbb{G}} 
\]

\[
\begin{array}{c}
  Q
\end{array} \xrightarrow{\mathbb{G}} 
\]

\[
\begin{array}{c}
  r
\end{array} \xrightarrow{\mathbb{G}} 
\]

\[
\begin{array}{c}
  u
\end{array} \xrightarrow{\mathbb{G}} 
\]

\[
\begin{array}{c}
  Enc_{pk}(u)
\end{array} \xrightarrow{\mathbb{G}} 
\]

proving key \[ \sigma = (pk, c_1, \ldots, c_m) \]

verification key \[ \tau = (\mathbb{G}, \mathbb{G}) \]

---

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

\[ \text{HomEval}_{pk}(P_{x,w}, c) \]

\[ \text{(same)} \]

\[ V(\tau, x, \pi) \]

\[ \text{Test}_{pk}(D_x, \hat{c}, Enc_{pk}(u)) \]
Publicly-Verifiable pp SNARGs

???
linear
interactive
proof

compiler

publicly-verifiable
preprocessing SNARG

uses bilinear maps + KEA
gives us Encoding with:
- test quadratic predicates
- certain one-way hardness
- almost-linear homomorph.

query/decision algorithms are low-degree polynomials

???

= 

a balancing act:
Summary

- A simple and motivated recipe

- PCP

- Linear PCP

- Compiler

- Linear IP

- Compiler

- Privately-verifiable preprocessing SNARG

- Low-degree linear PCP

- Compiler

- Low-degree linear IP

- Compiler

- Publicly-verifiable preprocessing SNARG

- See paper for more (including ZK generic transformation)
THANKS!
http://eprint.iacr.org/2012/718
THANKS!
http://eprint.iacr.org/2012/718
FOLLOWING ARE OLD SLIDES
Falsifiable Assumptions

An assumption is *falsifiable* if a challenger can efficiently test, via an interactive protocol, whether an efficient adversary breaks it.

**Example:** $\text{DL} \equiv \max_A \Pr[A(C) = \text{win}] \leq \text{negl}$

![Interactive Protocol Diagram]

**Other examples:**
DDH, RSA, LWE, QR, ...

**Unexamples (i.e., non-falsifiable):**
- $(P, V)$ is ZK  (not a game: requires a simulator)
- knowledge of exponent: given random $(g, h) \in G \times G$
  can’t efficiently generate $(g^\beta, h^\beta)$
  without “knowing” $\beta$

[Naor 03] [GW11] [Dam91] [HT98]
non-falsifiable assumptions are not all equally strong/complex

By investigating such assumptions and their power, we may:

- identify “nice” NF assumptions
- discover entirely new constructions
Bilinear Techniques & Preprocessing

coming from a line of work on NIZKs [Groth, GOS, AF] seeking to minimize group elements in a NI proof

very different construction approach

<table>
<thead>
<tr>
<th>supported functions</th>
<th># messages</th>
<th>offline work is cheap?</th>
<th>secure w. verifier oracle?</th>
<th>publicly verifiable?</th>
<th>main assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Groth]</td>
<td>NP</td>
<td>1</td>
<td>NO</td>
<td>YES</td>
<td>“KEA”</td>
</tr>
<tr>
<td>[Lipmaa]</td>
<td>NP</td>
<td>1</td>
<td>NO</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>[GGPR]</td>
<td>NP</td>
<td>1</td>
<td>NO</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

the verifier must preprocess the circuit to make a CRS
1) Bootstrapping SNARKs: A New Path To the Holy Grail

[Bitansky, Canetti, C, Tromer]
Bootstrapping SNARKs

provided a SNARG has a natural proof of knowledge
(all known ones do)
its efficiency properties can be improved to “optimal”

<table>
<thead>
<tr>
<th>supported functions</th>
<th># messages</th>
<th>offline work is cheap?</th>
<th>secure w. verifier oracle?</th>
<th>publicly verifiable?</th>
<th>main assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[BCCTb]</td>
<td>NP</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>NP</td>
<td>1</td>
<td>YES</td>
<td>X</td>
<td>X + CRH</td>
</tr>
</tbody>
</table>

ADDITIONAL EFFICIENCY BONUSES

● complexity preservation:
  prover in $T \cdot \text{poly}(k)$ time for $T$ time computations
  $S \cdot \text{poly}(k)$ space $S$ space

● transformation does not invoke the PCP theorem
The Old Path

Preprocessing (pv) SNARK

Kilian then FS w/ "RO HASH"

PCP

Machine computations

[Groth]
[Lipmaa]
[GGPR] w/ "KEA"
A New Path

Kilian then FS w/ "RO HASH"

preprocessing (pv) SNARK

bootstrapping [BccTb]

PCP

machine computations

[Groth] [Lipmaa] [GGPR] w/ "KEA"
A New & Better Path

any SNARK (no matter how weak)

Kilian then FS w/ "RO HASH"

preprocessing (pv) SNARK

bootstrapping [BCTb]

PCP

machine computations

[Groth]
[Lipmaa]
[GGPR]

w/ "KEA"
2) A General Technique For Making Preprocessing SNARKs

[Bitansky, C, Ishai, Ostrovsky, Paneth]
How Make Preprocessing SNARKs?

<table>
<thead>
<tr>
<th>supported functions</th>
<th># messages</th>
<th>offline work is cheap?</th>
<th>secure w. verifier oracle?</th>
<th>publicly verifiable?</th>
<th>main assumption</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Groth]</td>
<td>NP</td>
<td>1</td>
<td>NO</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>[Lipmaa]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[GGPR]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[BCIOP]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>“KEA”</td>
</tr>
</tbody>
</table>

**General technique to make preprocessing SNARKs:**

Step 1 = design a 2-message linear interactive proof (LIP)
Step 2 = force prover to act as a linear function
A New & Better Path

any SNARK
(no matter how crappy)

Kilian then FS
w/ “RO HASH”

preprocessing (pv) SNARK

bootstrapping

PCP

{BCIOP}

LIP

linear targeted malleability

machine computations
1) on bootstrapping SNARKs
[Bitansky, Canetti, C, Tromer]

2) on making preprocessing SNARKs
[Bitansky, C, Ishai, Ostrovsky, Paneth]
ON
BOOTSTRAPPING SNARKs
Theorem
Suppose CRHs exist.
Then there are efficient $T_1$ and $T_2$ such that:

\[
\text{any SNARK} \xrightarrow{T_1} \text{complexity-preserving SNARK} \xrightarrow{T_2} \text{complexity-preserving PCD}
\]

\text{complexity-preserving} = \begin{aligned} &\text{- no preprocessing} \\ &\text{- prover has quasi-optimal time \& space complexity} \end{aligned}
high-level intuition with no abstraction layers for

Theorem’ (removing preprocessing)
Suppose CRHs exist.
Then there is an efficient $T$ such that:

preprocessing publicly-verifiable SNARK

$T$

publicly-verifiable SNARK

WHAT DO WE DO?? 😞
The Core Idea: bootstrap the SNARK

Main Observation
only need to budget for small computations...

as small as SNARK verification (plus a bit more)

⇒ inefficiencies are ``localized''
and thus become inexpensive!
ON MAKING PREPROCESSING SNARKs
Designing Efficient Arguments

**Step 1:** information-theoretic probabilistic checking, in a model where the prover is restricted in some form

**Step 2:** use cryptography to “implement” the model

**EXAMPLES**

Step 1 = design a PCP
Step 2 = force prover to commit to a PCP

Step 1 = design a nsMIP
Step 2 = force prover to act as no-signaling provers

Step 1 = design an MIP
Step 2 = force prover to act as non-communicating provers
Designing Efficient Arguments

**Step 1:** information-theoretic probabilistic checking, in a model where the prover is restricted in some form

**Step 2:** use cryptography to "implement" the model

### A New Example

**Step 1** = design a 2-message linear interactive proof
**Step 2** = force prover to act as a linear function

**Q1:** how to design LIPs with suitable efficiency?
**Q2:** how to use crypto to make a prover linear?
Linear Interactive Proofs (LIPs)

The prover is **algebraically bounded**: specifically, linear.

\[
\exists \, \Pi \in \mathbb{F}^{k \times m}, \, b \in \mathbb{F}^k \text{ s.t. } a = \Pi q + b \\
(\Pi, b \text{ depend on } x, w)
\]

- **succinct**: \(k = O(1)\)
- **input oblivious**: \(V = (Q, D)\) s.t.

\[
q_1, \ldots, q_m \quad \text{r} \in_R \mathbb{F}^*
\]

\[
a_1, \ldots, a_k \quad \text{u} \in \mathbb{F}^*
\]
Step 2: Making Provers Linear

WARM UP: from LIP to privately-verifiable pp SNARK

TOOLS:

LIP \((P, (Q, D)))\)

\[
P(x, w) \quad q_1, \ldots, q_m \quad a_1, \ldots, a_k \quad r \in_R \mathbb{F}^* \quad u \in \mathbb{F}^*
\]

\&

“crypto”
Step 2: Making Provers Linear

WARM UP: from LIP to privately-verifiable pp SNARK

\[(1^k) \equiv (pk, sk) \leftarrow \text{Gen}(1^k)\]

proving key \(\sigma = (pk, c_1, \ldots, c_m)\)

verification key \(\tau = (sk, u)\)

\[P(\sigma, x, w) = \text{HomEval}_{pk}(P(x, w), c)\]

\[\hat{c}_1 \rightarrow \hat{c}_k\]

\[\text{Dec}_{sk} \rightarrow a_1 \rightarrow \cdots \rightarrow a_k \rightarrow D_x(\cdot, u)\]
Step 2: Making Provers Linear

WARM UP: from LIP to privately-verifiable pp SNARK

Linear Targeted Malleability (\sim [BSW])

encryption scheme that ONLY allows 
\(\mathbb{F}\) -additive homomorphic operations 
(e.g., Paillier)

proving key 
\[ \sigma = (pk, c_1, \ldots, c_m) \]

verification key 
\[ \tau = (sk, u) \]

\(P(\sigma, x, w) = \text{HomEval}_{pk}(P(x, w), c)\)

\(\hat{c}_1 \to \hat{c}_k \)

\[ V(\tau, x, \pi) \]

\(\text{Dec}_{sk} \)

\(a_1 \to a_k \)

\(D_x(\cdot, u) \)
Step 2: Making Provers Linear

What happens if we want **public verifiability**?

Being able to test properties of the prover’s answers implies that we must **give up** semantic security.

(1) What notion of security should $\text{Enc}_{pk}(q_i)$ satisfy?

In particular, security must be preserved even given certain **leakage** on the queries.

(2) What kinds of LIPs then suffice?
Step 2: Making Provers Linear

(1) What notion of security should Enc_{pk}(q_i) satisfy?

(1) $\Delta$-power OW

$$s \leftrightarrow A(pk, Enc_{pk}(s), Enc_{pk}(s^2), \ldots , Enc_{pk}(s^\Delta))$$

$$p^* \leftrightarrow A(pk, Enc_{pk}(p_1(s)), \ldots , Enc_{pk}(p_\ell(s)))$$

$p^*(s) = 0, p^* \neq 0$ \hspace{1cm} $p_1, \ldots , p_\ell$ of degree $\Delta$

(2) What kinds of LIPs then suffice?

(2) **LIPs with low-degree verifiers**

**Def:** an LIP $(P, (Q, D))$ has degree $(d_Q, d_D)$ if

i) $Q(r)$ has total degree at most $d_Q$

ii) $D_x(u, a)$ has total degree at most $d_D$

$(Q$ and $D_x$ are multivalued multivariate polynomials over $\mathbb{F})$
Step 2: Making Provers Linear

**SKETCH**

\[(1^k) \equiv (pk, \red{\Box}) \leftarrow \text{Gen}(1^k) \]

```plaintext
proving key
\sigma = (pk, c_1, \ldots, c_m)
```

```
verification key
\tau = (\red{\Box}, \red{\Box})
```

\[P(\sigma, x, w) = \text{HomEval}_{pk}(P(x, w), c)\]

\[\hat{c}_1 \quad \vdots \quad \hat{c}_k\]

\[V(\tau, x, \pi)\]

\[\text{Dec}_{sk} \quad a_1 \quad a_k \quad \text{Dec}_{sk} \quad D_x(\cdot, u)\]
Step 2: Making Provers Linear

**SKETCH**

\[(1^k) \equiv (pk, \text{red}) \leftarrow \text{Gen}(1^k)\]

\[Q \quad r \in_R \mathbb{F}^* \]

proving key
\[\sigma = (pk, c_1, ..., c_m)\]

verification key
\[\tau = (\text{red}, \text{red})\]

\[P(\sigma, x, w) = \text{HomEval}_{pk}(P(x, w), c)\]

\[\hat{c} \quad D_x(\cdot, \cdot) \text{AEnc}_{pk}(u)\]

\[V(\tau, x, \pi)\]
Step 2: Making Provers Linear

**SKETCH**

\[ G(1^k) \equiv (\text{pk}, \text{c}_1, \ldots, \text{c}_m) \]

proving key

\[ \sigma = (\text{pk}, c_1, \ldots, c_m) \]

verification key

\[ \tau = (\text{pk}, \text{c}) \]

**P(σ, x, w) = HomEval_{pk}(P(x, w), c)\]

\[ \hat{c}_1 \rightarrow \cdots \rightarrow \hat{c}_k \]

\[ V(\tau, x, \pi) = \text{HomEval}_{pk}(D_x, \hat{c}, AEnc_{pk}(u)) \subseteq \text{Enc}_{pk}(0) \]
Step 2: Making Provers Linear

SKETCH

\[(1^k) \equiv (\text{pk}, \text{SK}) \leftarrow \text{Gen}(1^k)\]

proving key
\[\sigma = (\text{pk}, c_1, \ldots, c_m)\]

verification key
\[\tau = (\text{pk}, \text{vk})\]

- test root with bilinear map so need \(d_D = 2\)
- similar linear TM assumption

\[P(\sigma, x, w) = \text{HomEval}_{\text{pk}}(P(x, w), c)\]

\[V(\tau, x, \pi) \quad \text{Test}_{\text{pk}}(D_x, \hat{c}, \text{AEnc}_{\text{pk}}(u))\]

“tests quadratic roots”
publicly-verifiable pp SNARK

Step 2

LIP with degree \((\text{poly}(k), 2)\)

Step 1

???
publicly-verifiable pp SNARK

Step 2

LIP with degree \((\text{poly}(k),2)\)

Step 1

Linear PCP with degree \((\text{poly}(k),2)\)
Linear PCPs (LPCPs)

A PCP in which (honest and dishonest) proofs are $\mathbb{F}$-linear.

$$\pi : \mathbb{F}^m \rightarrow \mathbb{F}$$

$q_1, \ldots, q_k \in \mathbb{F}^m$

$a_1, \ldots, a_k \in \mathbb{F}$

$V(x)$

($\pi$ depends on $x, w$)
Linear PCPs (LPCPs)

A PCP in which (honest and dishonest) proofs are \(\mathbb{F}\)-linear.

Similarly, input oblivious: \(V = (Q, D)\) s.t.

and degrees \((d_Q, d_D)\).
Linear PCPs (LPCPs)

A PCP in which (honest and dishonest) proofs are $\mathbb{F}$-linear.

Two technical notes:

1. *linear PCP* in [IKO, SMBW, SVP+, SBV+] does not restrict oracle to be linear in dishonest case

2. not the same as *linear PCPP* in [BSHLM09, Mei12]; there it is a proximity tester for the kernels of linear circuits
From LPCPs To LIPs

Given a $k$-query $m$-length LPCP, how to construct an LIP of similar efficiency?

Why isn’t an LPCP already an LIP? **Consistency.**
From LPCPs To LIPs

Given a $k$-query $m$-length LPCP, how to construct an LIP of similar efficiency?

\[ \pi \in \mathbb{F}^m \]

\[ q_1, \ldots, q_k \in \mathbb{F}^m \]

\[ a_1, \ldots, a_k \in \mathbb{F} \]

\[ V(x) \]

\[ q_{k+1} = \sum_{i=1}^{k} \alpha_i q_i \text{ where } \alpha_1, \ldots, \alpha_k \in \mathbb{F} \]

consistency check

\[ q'_1, \ldots, q'_{(k+1)m} \in \mathbb{F} \]

\[ a'_1, \ldots, a'_{k+1} \in \mathbb{F} \]

\[ V(x) \]

preserves algebraic properties!
publicly-verifiable pp SNARK

Step 2

LIP with degree \((\text{poly}(k),2)\)

Step 1

Linear PCP with degree \((\text{poly}(k),2)\)

Step 0

???
publicly-verifiable pp SNARK

Step 2

LIP with degree (poly(k),2)

Step 1

Linear PCP with degree (poly(k),2)

Step 0

designing LPCPs for NP with O(1) queries is easy!
publicly-verifiable pp SNARK

Step 2

LIP with degree \((\text{poly}(k), 2)\)

Step 1

Linear PCP with degree \((\text{poly}(k), 2)\)

Step 0

\[ m = O(|C|^2) \]
\[ k = 3 \]

\[ m = O(|C|) \]
\[ k = 3 \text{ (or 4)} \]

system of \(O(|C|)\) quadratic equations over \(\mathbb{F}\)
Two Known Paths To The Holy Grail

- “CS proof”
- Preprocessing SNARK
- PCP
- LIP
- LPCP
- Machine computations
THANKS!
THANKS!