Delegation with (nearly) optimal time/space overhead

Justin Holmgren
MIT

Ron Rothblum
MIT
Verifiable Computation
Verifiable Computation
Verifiable Computation

M(x) = y

"M(x) = y"
Verifiable Computation

\[ M(x) = ? \text{, challenge} \]

\[ \text{“} M(x) = y \text{” , proof} \]
Verifiable Computation

\[ M(x) = ? \text{, challenge} \]

\[ "M(x) = y" \text{, proof} \]

accept?
Verifiable Computation

M(x) = ?, challenge

“M(x) = y”, proof

Complexity << evaluating M(x)

accept?
Verifiable Computation

M(x) = ?, challenge

“M(x) = y”, proof

accept?

Complexity ~evaluating M(x)

Complexity << evaluating M(x)
Figure 5. Prover overhead normalized to native execution cost for two computations. Prover overheads are generally enormous.

Walfish, Blumberg ’15
“An additional bottleneck is memory: the prover must materialize a transcript of a computation's execution.”

Walfish, Blumberg ’15
Verifiable Computation

Our focus:
- Prover efficiency
- Computational assumptions
Prior Work
## Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>No-Signaling PCP [KRR14, KP15, BHK16]</td>
<td>RAM, PIR</td>
<td>poly$(T)$</td>
<td>poly$(T')$</td>
</tr>
</tbody>
</table>
Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCP</strong> [KRR14, KP15, BHK16]</td>
<td>RAM</td>
<td>PIR</td>
<td>$T^{60},?$</td>
</tr>
</tbody>
</table>
## Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCP</strong> [KRR14, KP15, BHK16]</td>
<td>RAM, PIR</td>
<td>$T^3?$</td>
<td>$T^3?$</td>
</tr>
</tbody>
</table>
## Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCP</strong></td>
<td>RAM, PIR</td>
<td>$T^3?$</td>
<td>$T^3?$</td>
</tr>
<tr>
<td>[KRR14, KP15, BHK16]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>SNARKs</strong></td>
<td>RAM, Non-Falsifiable</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td>[BC12, BCCT12, …]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Succinct Garbling</strong></td>
<td>RAM, Obfuscation</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td>[GHRW14, KLW15, CH15, CCCLLZ15]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCP</strong> [KRR14, KP15, BHK16]</td>
<td>RAM, PIR</td>
<td>$T^3?$</td>
<td>$T^3?$</td>
</tr>
<tr>
<td><strong>SNARKs</strong> [BC12, BCCT12, …]</td>
<td>RAM, Non-Falsifiable</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td><strong>Succinct Garbling</strong> [GHRW14, KLW15, CH15, CCCLLZ15]</td>
<td>RAM, Obfuscation</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td>[this work]</td>
<td>TM, “Slightly” Homomorphic Encryption</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S + \text{poly}(\kappa)$</td>
</tr>
</tbody>
</table>
## Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCP</strong> [KRR14, KP15, BHK16]</td>
<td>RAM, PIR</td>
<td>$T^3?$</td>
<td>$T^3?$</td>
</tr>
<tr>
<td><strong>SNARKs</strong> [BC12, BCCT12, …]</td>
<td>RAM, Non-Falsifiable</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td><strong>Succinct Garbling</strong> [GHRW14, KLW15, CH15, CCCLLZ15]</td>
<td>RAM, Obfuscation</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td>[this work]</td>
<td>TM, “Slightly” Homomorphic Encryption</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S + \text{poly}(\kappa)$</td>
</tr>
</tbody>
</table>

- Extends to (cache-efficient) RAM
## Prior Work

<table>
<thead>
<tr>
<th>Model</th>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCP</strong>&lt;br&gt;[KRR14, KP15, BHK16]</td>
<td>RAM, PIR</td>
<td>$T^3$?</td>
<td>$T^3$?</td>
</tr>
<tr>
<td><strong>SNARKs</strong>&lt;br&gt;[BC12, BCCT12, …]</td>
<td>RAM, Non-Falsifiable</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td><strong>Succinct Garbling</strong>&lt;br&gt;[GHRW14, KKW15, CH15, CCCLLZ15]</td>
<td>RAM, Obfuscation</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td>[this work]</td>
<td>TM, “Slightly” Homomorphic Encryption</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
<td>$S + \text{poly}(\kappa)$</td>
</tr>
</tbody>
</table>

Extends to (cache-efficient) RAM
Probabilistically Checkable Proofs
Probabilistically Checkable Proofs

Proof string $\pi$: $\pi_1, \pi_2, \ldots, \pi_L$

Verifier

Input $x$
Probabilistically Checkable Proofs

Proof string $\pi$

Verifier

Input $x$
Probabilistically Checkable Proofs

Proof string \( \pi \)

\[
\begin{array}{c}
\pi_1 \\
\pi_2 \\
\vdots \\
\pi_L
\end{array}
\]

\( i_1 \rightarrow \pi_1 \)
\( i_2 \rightarrow \pi_2 \)
\( i_3 \rightarrow \pi_L \)

\( x \in \mathcal{L} \implies \text{exists convincing proof} \)

Verifier

Input \( x \)
Probabilistically Checkable Proofs

Proof string $\pi$  

Proof string $\pi$  

$\pi_1 \pi_2 \ldots \pi_L$

$x \in \mathcal{L} \implies$ exists convincing proof

$x \notin \mathcal{L} \implies$ every proof convinces with low probability

Verifier

Input $x$
Probabilistically Checkable Proofs

Proof string $\pi$:

$$\pi_1 \pi_2 \ldots \pi_L$$

$\exists$ convincing proof

$x \in \mathcal{L}$ $\implies$ every proof convinces with low probability

$x \notin \mathcal{L}$ $\implies$ not a standard-model delegation scheme

Verifier

Input $x$
PCP-based Delegation
PCP-based Delegation

PCP proof $\pi$

PCP verifier
PCP-based Delegation

PCP proof $\pi$

PCP verifier

independent PIR queries

$\pi_{i_1}, \ldots, \pi_{i_k}$
PCP-based Delegation

[Biehl-Meyer-Wetzel 98]

PCP proof $\pi$

PCP verifier

Independent PIR queries

$\pi i_1, \ldots, \pi i_k$

$i_1, \ldots, i_k$
PCP-based Delegation

Not sound in general

\[ i_1, \ldots, i_k \]

\[ i_1, \ldots, i_k \]

\[ \pi i_1, \ldots, \pi i_k \]

PCP proof $\pi$

PCP verifier

[Biehl-Meyer-Wetzel 98]

[Dwork-Langberg-Naor-Nissim-Reingold 01]
PCP-based Delegation

[PCP proof $\pi$]

independent PIR queries

- $i_1, \ldots, i_k$

PCP verifier

- $\pi_{i_1}, \ldots, \pi_{i_k}$

• Not sound in general
  [Dwork-Langberg-Naor-Nissim-Reingold 01]

• Sound if the PCP is no-signaling sound
  [Kalai-Raz-Rothblum 14]
PCP-based Delegation
[Biehl-Meyer-Wetzel 98]

- Not sound in general
  [Dwork-Langberg-Naor-Nissim-Reingold 01]
- Sound if the PCP is no-signaling sound
  [Kalai-Raz-Rothblum 14]
PCP-based Delegation

[Biehl-Meyer-Wetzel 98]

- Not sound in general
  [Dwork-Langberg-Naor-Nissim-Reingold 01]
- Sound if the PCP is no-signaling sound
  [Kalai-Raz-Rothblum 14]

PCP proof $\pi$

general computations!

PCP verifier

no precomputation!

independent PIR queries

$\pi_i_1, \ldots, \pi_i_k$

$i_1, \ldots, i_k$
Observation 0

PCP proof $\pi$

PCP verifier

independent PIR queries

$i_1, \ldots, i_k$

$\pi i_1, \ldots, \pi i_k$
Observation 0

PCP proof $\pi$

FHE ciphertexts

independent PIR queries

$\pi i_1, \ldots, \pi i_k$

PCP verifier

$i_1, \ldots, i_k$
Observation 0

• If PIR = FHE, just need efficient “random-access” to PCP.
Observation 0

If PIR = FHE, just need efficient "random-access" to PCP.

No-Signaling PCP with efficient prover

PCP verifier

PCP proof \pi

WANTED

$$$$ reward
Our Technical Contributions
Our Technical Contributions

1. Simpler and direct NS-PCP (essentially BFLS) for any language \( \mathcal{L} \in \text{TISP}(T, S) \)
Our Technical Contributions

1. Simpler and direct NS-PCP (essentially BFLS) for any language $\mathcal{L} \in \text{TISP}(T, S)$

Remove major component of KRR, namely “augmented circuit”
Our Technical Contributions

Remove major component of KRR, namely “augmented circuit”

1. **Simpler** and **direct** NS-PCP (essentially BFLS) for any language \( \mathcal{L} \in \text{TISP}(T, S) \)

2. Super-efficient prover: Any symbol computable in time: \( \tilde{O}(T) \) space: \( S + \text{polylog}(T) \)
Our Technical Contributions

Remove major component of KRR, namely “augmented circuit”

1  **Simpler** and **direct** NS-PCP (essentially BFLS) for any language \( \mathcal{L} \in \text{TISP}(T, S) \)

2  **Super-efficient prover:** Any symbol computable in time: \( \tilde{O}(T) \) space: \( S + \text{polylog}(T) \)

2’ **Limited efficiency loss under FHE**
Our Technical Contributions

1. Simpler and direct NS-PCP (essentially BFLS) for any language $\mathcal{L} \in \text{TISP}(T, S)$

2. Super-efficient prover: Any symbol computable in time: $\tilde{O}(T)$, space: $S + \text{polylog}(T)$

2'. Limited efficiency loss under FHE
   time: $T \cdot \text{poly}(\lambda)$
Our Technical Contributions

1. **Simpler and direct** NS-PCP (essentially BFLS) for any language $\mathcal{L} \in \text{TISP}(T, S)$

2. **Super-efficient prover:** Any symbol computable in time: $\tilde{O}(T)$ space: $S + \text{polylog}(T)$

2'. **Limited efficiency loss under FHE**
   - time: $T \cdot \text{poly}(\lambda)$
   - space: $S + \text{poly}(\lambda)$
Our Technical Contributions

1. **Simpler** and **direct** NS-PCP (essentially BFLS) for any language \( \mathcal{L} \in \text{TISP}(T, S) \)

2. **Super-efficient prover**: Any symbol computable in \( \tilde{O}(T) \) time and \( S + \text{poly}(\lambda) \) space in \( \mathcal{L} \)

2'. **Limited efficiency loss under FHE**
   time: \( T \cdot \text{poly}(\lambda) \)    space: \( S + \text{poly}(\lambda) \)

Remove major component of KRR, namely “augmented circuit”

BFLS already known to be complexity-preserving? [BC12, BTVW14]
Our Technical Contributions

1. Simpler and direct NS-PCP (essentially BFLS) for any language $L \in \text{TISP}(T, S)$

2. Super-efficient prover: Any symbol computable in $\tilde{O}(T)$ time and $S + \text{poly}(\lambda)$ space:

   - time: $\tilde{O}(T)$
   - space: $S + \text{poly}(\lambda)$

2'. Limited efficiency loss under FHE
   - time: $T \cdot \text{poly}(\lambda)$
   - space: $S + \text{poly}(\lambda)$

Remove major component of KRR, namely “augmented circuit”

BFLS already known to be complexity-preserving? [BC12, BTVW14]
Our Technical Contributions

1. Simpler and direct NS-PCP (essentially BFLS) for any language $\mathcal{L} \in \text{TISP}(T, S)$

2. Super-efficient prover: Any symbol computable in $\tilde{O}(T)$ time: $S + \text{poly}(\lambda)$ space:

2'. Limited efficiency loss under FHE time: $T \cdot \text{poly}(\lambda)$ space: $S + \text{poly}(\lambda)$

Remove major component of KRR, namely “augmented circuit”

BFLS already known to be complexity-preserving? [BC12, BTVW14] with non-deterministic computations
Talk Outline
Talk Outline

NOT proving NS-soundness of BFLS for deterministic circuits
Talk Outline

NOT proving NS-soundness of BFLS for deterministic circuits

Part 1: Turing / RAM Machines (non-succinct) deterministic circuits
Talk Outline

NOT proving NS-soundness of BFLS for deterministic circuits

**Part 1:** Turing / RAM Machines → (non-succinct) deterministic circuits

**Part 2:** (part of) BFLS prover efficiency despite non-succinctness.
Turing Machines as Circuits

TM Configuration

tape
Turing Machines as Circuits

TM Configuration

tape
Turing Machines as Circuits

TM Configuration

tape
Turing Machines as Circuits

TM Configuration

tape

:
Turing Machines as Circuits

TM Configuration

Transcript / Circuit

Config_{T-1}

Config_1

Config_0
Turing Machines as Circuits

TM Configuration

Transcript / Circuit

Config_{T-1}

Config_{i_1}

Config_{i_0}
RAM Machines as Circuits

Configuration:

● ● ● ● ●
RAM Machines as Circuits

Configuration:
(diameter log S)

leaves = memory
RAM Machines as Circuits

Configuration:
(diameter log S)

leaves = memory
RAM Machines as Circuits

Configuration:
(diameter log S)

leaves = memory
RAM Machines as Circuits

Configuration:
(diameter $\log S$)

leaves = memory
RAM Machines as Circuits

Configuration:
(diameter log S)

leaves = memory

Important for BFLS:
Graph is “regular”!
RAM Machines as Circuits

Configuration:
(diameter $\log S$)

Transcript / Circuit:

leaves = memory

Important for BFLS:
Graph is “regular”!
RAM Machines as Circuits

Configuration:
(diameter log S)
leaves = memory

Important for BFLS:
Graph is “regular”!
RAM Machines as Circuits

Configuration:
(diameter log S)

leaves = memory

Transcript / Circuit:

Important for BFLS:
Graph is “regular”!
RAM Machines as Circuits

Configuration:
(diameter log S)

leaves = memory

Important for BFLS:
Graph is “regular”!

no Merkle trees!

no routing networks!

Transcript / Circuit:
The PCP (BFLS) Part 1: Multilinear extension
The PCP (BFLS) Part 1: Multilinear extension

Let $f : \{0, 1\}^m \rightarrow \mathbb{F}$ be any function.
The PCP (BFLS) Part 1: Multilinear extension

Let $f : \{0, 1\}^m \to \mathbb{F}$ be any function.
The PCP (BFLS) Part 1: Multilinear extension

Let \( f : \{0, 1\}^m \rightarrow \mathbb{F} \) be any function.
The PCP (BFLS) Part 1: Multilinear extension

Let \( f : \{0, 1\}^m \rightarrow \mathbb{F} \) be any function.

\[
\hat{f}(x) = \sum_{x \in \{0,1\}^m} f(x) \cdot \mathbf{1}_x(x)
\]
The PCP (BFLS) Part 1:
Multilinear extension

Let $f : \{0, 1\}^m \rightarrow \mathbb{F}$ be any function.

Let $\hat{f} : \mathbb{F}^m \rightarrow \mathbb{F}$ be any function.

\[
\hat{f}(\mathbf{x}) = \sum_{\mathbf{x} \in \{0, 1\}^m} f(\mathbf{x}) \cdot \mathbf{1}_x(\mathbf{x})
\]
Let $f : \{0, 1\}^m \to \mathbb{F}$ be any function.

\[
\hat{f}(x) = \sum_{x \in \{0, 1\}^m} f(x) \cdot \mathbf{1}_x(x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$
1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x) \cdot \hat{1}_{y, x}(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]
1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y,x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[ \hat{C}(y, x) = \sum_{y,x} C(y, x) \]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} \mathcal{C}(y, x)$$
1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\( \hat{C}(y, x) = \sum_{y, x} C(y, x) \)
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]
1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y,x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y,x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$

was 3, now 0
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y,x} C(y, x)$$

was 3, now 0

-3
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \to \{0, 1\} \)

\[ \hat{C}(y, x) = \sum_{y, x} C(y, x) \]

was 3, now 0
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]

was 3, now 0

\[
\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}
\]
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]

was 3, now 0
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y,x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x)$$

- $\hat{C}$ was 3, now 0
- $\hat{C}$ was 1, now 2

+1

sum

sum
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \to \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y,x} C(y, x)
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C}: \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y,x} C(y, x)$$
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x)
\]

\[
\sum_{x, y} C(x, y)
\]

was 3, now 0

\[
\hat{C}(x) = \sum_{y} \hat{C}(y, x)
\]

\[
\hat{C}(y) = \sum_{x} \hat{C}(x, y)
\]

\[
\hat{C}(x) \cdot \hat{C}(y) = \hat{C}(y) \cdot \hat{C}(x)
\]

\[
\hat{C}(x) = \hat{1}
\]
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y,x} C(y, x)$$

$$\sum_{x,y} C(x, y)$$

- was 3, now 0
- was 1, now 2

implicit enumeration of $\square$
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x) \cdot \hat{1}_{y,x}(y, x)
\]

\[
\sum_{x, y} C(x, y)
\]

was 3, now 0

\( \hat{1}_{y,x}(y, x) \)

was 1, now 2

implicit enumeration of \( \square \)
Prover Efficiency

1. Evaluating extension of transcript $\hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\}$

$$\hat{C}(y, x) = \sum_{y, x} C(y, x) \cdot \hat{1}_{y,x}(y, x)$$

$$\sum_{x, y} C(x, y)$$

was 3, now 0

was 1, now 2

implicit enumeration of $\square$
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x) \cdot \hat{1}_{y,x}(y, x)
\]

Coefficients structured; all is still well

\[
\sum_{x, y} C(x, y)
\]

was 3, now 0

\( \hat{C} \):

\[
\hat{C}(y, x) = X_y \cdot x
\]

was 1, now 2

\[
\hat{C}(y, x) = \hat{1}_y \cdot x
\]

implicit enumeration of \( \square \)
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[
\hat{C}(y, x) = \sum_{y, x} C(y, x) \cdot \hat{1}_{y, x}(y, x)
\]

Coefficients structured; all is still well

\[
\sum_{x, y} C(x, y)
\]

was 3, now 0

was 1, now 2

implicit enumeration of \( \square \)
Prover Efficiency

1. Evaluating extension of transcript \( \hat{C} : \{0, 1\}^{t+s} \rightarrow \{0, 1\} \)

\[ \hat{C}(y, x) = \sum_{y,x} C(y, x) \cdot \hat{1}_{y,x}(y, x) \]

Coefficients structured; all is still well

\[ \sum_{x,y} C(x, y) \]

\( \vdots \) was 3, now 0

\( \vdots \) was 1, now 2

implicit enumeration of \( \square \)
Additional Challenges
Additional Challenges

- Other (sum-check) polynomials
Additional Challenges

- Other (sum-check) polynomials
- Getting rid of KRR’s augmented circuit
Additional Challenges

• Other (sum-check) polynomials
• Getting rid of KRR’s augmented circuit
• Prover efficiency under somewhat homomorphic encryption
Additional Challenges

- Other (sum-check) polynomials
- Getting rid of KRR’s augmented circuit
- Prover efficiency under somewhat homomorphic encryption
  - Low multiplicative degree, $O(1)$ field operations per step
Additional Challenges

• Other (sum-check) polynomials

• Getting rid of KRR’s augmented circuit

• Prover efficiency under somewhat homomorphic encryption
  • Low multiplicative degree, \( O(1) \) field operations per step
  • Space stays \( S + \text{poly}(\kappa) \), not \( S \cdot \text{poly}(\kappa) \)
## Summary

<table>
<thead>
<tr>
<th>Assumptions</th>
<th>Prover Time</th>
<th>Prover Space</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>No-Signaling PCPs [KRR, ...]</strong></td>
<td>PIR</td>
<td>$\geq T^3 S^3$</td>
</tr>
<tr>
<td><strong>SNARKs [BC,BCCT, ...]</strong></td>
<td>Non-Falsifiable</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td><strong>Succinct Garbling [GHRW, KLW, ...]</strong></td>
<td>Obfuscation/ multilinear maps</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td>[this work]</td>
<td>“Slightly”</td>
<td>$T \cdot \text{poly}(\kappa)$</td>
</tr>
<tr>
<td></td>
<td>Homomorphic Encryption</td>
<td></td>
</tr>
</tbody>
</table>
Open Questions

• How does this compare in practice? What are the remaining bottlenecks?

• Can PCP query complexity be reduced?

• Is there an FHE scheme which is extra efficient for our prover?

• Efficiently evaluate low-degree arithmetic circuits (large fields)
Open Questions

• How does this compare in practice? What are the remaining bottlenecks?

• Can PCP query complexity be reduced?

• Is there an FHE scheme which is extra efficient for our prover?

• Efficiently evaluate low-degree arithmetic circuits (large fields)

  low “asymmetric” degree (GSW) even better