Fully Linear PCPs and their Cryptographic Applications

Niv Gilboa – Ben-Gurion University
Based On


Goal

- Compute $f(x_1, \ldots, x_n)$
- Semi-honest adversary

- Prove correctness
- Com. = $o($circuit size$)$

$\Rightarrow$ Malicious com. $\approx$ semi-honest com.
Flavors of Malicious Security

• Security with abort
  • Incorrect execution
  • Who’s the bad guy? ➔
  • Abort

• Full security – guaranteed output delivery
  • Incorrect execution
  • What we want
  • What we actually get
Zero-knowledge proofs

[GMR89]

Complete. Honest $P$ convinces honest $V$.

Sound. Dishonest $P^*$ rarely fools honest $V$.

ZK. Dishonest $V^*$ learns only that $G \in 3\text{COL}$.

$\Rightarrow V^*$ learns nothing else about $G$
ZK for NP Statements (3-Colorability)

[GMW91]

Commit to colors

Prover $P$

Verifier $V$

Repeat $k$ times to improve prob.

Proof: polynomial size in input length
This talk

Zero-knowledge proofs on distributed data

Complete. Honest $P$ convinces honest $(V_1, V_2)$.

Sound. Dishonest $P^*$ rarely fools honest $(V_1, V_2)$.

Strong ZK. Dishonest $V_1^*$ (or $V_2^*$) learns only that $G_1 + G_2 \in 3\text{COL}$. $\Rightarrow V_1$ learns nothing else about $G_2$.
This talk
Zero-knowledge proofs on distributed data

\[ G_1 + G_2 \]

3-coloring of \( G_1 + G_2 \)

"\( G_1 + G_2 \) is 3-colorable"

**Verifier** \( V_1 \)

**Verifier** \( V_2 \)

**Prover** \( P \)

**3-round protocol** = As in other multiparty protocols

**Public coin** = Verifiers’ messages to prover are random strings

**More than two verifiers**
Special case

Zero-knowledge proofs on secret-shared data

Language $\mathcal{L} \subseteq \mathbb{F}^n$, for finite field $\mathbb{F}$.
Fully Linear PCP / IOP
Linear Probabilistically Checkable Proofs (PCPs) [IKO07]

Finite field $\mathbb{F}$, language $\mathcal{L} \subseteq \mathbb{F}^n$

**Linear PCP proof** is a vector $\pi$.

**Linear PCP verifier**
- takes $x$ as input,
- makes $O(1)$ linear queries to $\pi$.

Must satisfy notions of completeness, soundness, and zero knowledge.
**Fully linear probabilistically checkable proofs (PCPs)**

[This line of work]

Finite field $\mathbb{F}$, language $\mathcal{L} \subseteq \mathbb{F}^n$

**Fully linear PCP proof** is a vector $\pi$.

**Fully linear PCP verifier**
- takes $x$ as input,
- makes $O(1)$ linear queries to $(x \| \pi)$.

Must satisfy notions of completeness, soundness, and zero knowledge.

\[ x \in \mathbb{F}^n \quad \pi \in \mathbb{F}^m \]

query $q \in \mathbb{F}^{n+m}$

answer $a = \langle q, x \| \pi \rangle \in \mathbb{F}$

"$x \in \mathcal{L}$"
Fully linear IOPs
An interactive analogue of fully linear PCPs

Linear analogue + ZK of: [BCS16], [RRR16]

At the end of the interaction, verifier makes linear queries to
\[(x|\pi_1|\pi_2|...|\pi_t)\]
and accepts or rejects.

Naturally captures many existing proof protocols (GKR, ...)

Prover

Verifier

\[\pi_1 \in \mathbb{F}^m\]

\[\pi_2 \in \mathbb{F}^m\]

\[\text{challenge}_1\]

\[\text{challenge}_2\]

...
If language $\mathcal{L}$ has an efficient fully linear PCP/IOP, it has an efficient ZK proof on distributed data.

1. Generate FLPCP proof and split it using secret sharing.
If language $\mathcal{L}$ has an efficient fully linear PCP/IOP, it has an efficient ZK proof on distributed data.

2. Sample query vectors using common randomness.

Verifier $V_1$ 

\[
\begin{align*}
\text{Query } q &= 5 | 1 | 2 | 7 | 4 | 9 \\
\pi_1
\end{align*}
\]

\[
\begin{align*}
x_1 &\in \mathbb{F}^{n/2} \\
\pi_1
\end{align*}
\]

Verifier $V_2$ 

\[
\begin{align*}
x_2 &\in \mathbb{F}^{n/2} \\
\pi_2
\end{align*}
\]
If language $\mathcal{L}$ has an efficient fully linear PCP/IOP, it has an efficient ZK proof on distributed data.

3. Publish shares of query answers and reconstruct.

Verifier $V_1$

- $x_1 \in \mathbb{F}^{n/2}$
- $\pi_1$

$\langle q, x_1 \parallel \pi_1 \rangle \in \mathbb{F}$

$\langle q, x_1 \parallel \pi_1 \rangle = \langle q, x_1 \parallel (\pi_1 + \pi_2) \rangle$

Verifier $V_2$

- $x_2 \in \mathbb{F}^{n/2}$
- $\pi_2$

$\langle q, x_2 \parallel \pi_2 \rangle \in \mathbb{F}$

$\langle q, x_2 \parallel \pi_2 \rangle = \langle q, x \parallel \pi \rangle = \text{answer}$
If language $\mathcal{L}$ has an efficient fully linear PCP/IOP, it has an efficient ZK proof on distributed data.

4. Recover $O(1)$ query answers, run FLPCP verifier.

**Verifier $V_1$**

$x_1 \in \mathbb{F}^{n/2}$  
$\pi_1$

**Verifier $V_2$**

$x_2 \in \mathbb{F}^{n/2}$  
$\pi_2$

Communication: $|\text{proof}| + O(1)$
Selected results: New ZK proofs I

\( \mathbb{F} \) - finite field, \( \mathcal{L} \subseteq \mathbb{F}^n \)- language (\( n \ll |\mathbb{F}| \)), \( G: \mathbb{F}^L \rightarrow \mathbb{F} \) - algebraic gate

**Theorem.** If \( \mathcal{L} \) is recognized by a circuit \( \mathcal{C} \) that has \( M \) \( G \)-gates, and some addition gates, there is a public-coin ZK proof on distributed data for \( \mathcal{L} \) with:

- \( O(1) \) rounds and
- communication cost \( O(L + M(deg.G)) \). (elements of \( \mathbb{F} \))
Selected results: New ZK proofs II

**Theorem.** If $\mathcal{L}$ has a degree-two arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:
- $k$ rounds and
- communication cost $O(n^{O(1/k)})$.

(Improves: $\Omega(n)$ [BC17])

**Extensions to:**
- Rings $\mathbb{Z}_{2^k}$
- Degree $O(1)$ circuits
Constructions
Short proofs for structured circuits I

- Ideas similar to [LFKN92, AW09, GGPR13]
- Circuit over field $\mathbb{F}$:
  - Linear gates
  - “Large” algebraic G-gates
- Order gates
- Define Polynomials
  - $f_L$ – left inputs
  - $f_M$ – middle inputs
  - $f_R$ – right inputs
Short proofs for structured circuits II

- \( p = G(f_L, f_M, f_R) \) defines outputs
- \( p(1) \)
- \( p(2) \)
- \( p(3) \)
- \( p(\#G \text{ gates}) = C(x) \)
Short proofs for structured circuits II

- Prover sends \( p \)
- Length: \((\#G \text{ gates}) (\text{degree } G)\)
- Verifier checks
  - \( p(\#G \text{ gates})=0 \ (x \in \mathcal{L}) \)
  - \( p(r)=G(f_L,f_M,f_R)(r) \) for random \( r \)
- Verifier work requires
  - Interpolation
  - Evaluation \( \left\{\right. \) Linear! \( \left.\right\} \)
- ZK by extra randomization of \( f_L/f_M/f_R \)
Corollary

• $O(\sqrt{n})$ FL-PCP for any degree 2 circuit (Improves: Prio $\Omega(n)$ [BC17])
• $C(x)$ degree 2 $\Rightarrow C(x) = x^{-1}Ax$ for some matrix $A$
• $C(x) = \langle x, Ax \rangle$
• $C$ made up of
  • G gate – inner product on length $n^{1/2}$ inputs
  • Linear gates
• Proof size: $2n^{1/2}$
Reducing communication for simple languages

Let $\mathbb{F}$ be a finite field. Let $\mathcal{L} \subseteq \mathbb{F}^n$ be a language. $(n \ll \mathbb{F})$

**Theorem.** If $\mathcal{L}$ has a degree-two arithmetic circuit, there is a public-coin ZK proof on distributed data for $\mathcal{L}$ with:

- $O(\log n)$ rounds and
- communication cost $O(\log n)$.

- Uses our new FLPCP
- Idea: Recursively outsource the verifier’s work to the prover.
"Prove to me that the FLPCP verifier would have accepted $\pi_1$, using random coins $r$."

For circuits with “SIMD” structure, proof size shrinks: $O(|C|) \rightarrow O(\log |C|)$

Low-degree circuits have the necessary structure
Semi-Honest to Malicious MPC Compiler
Secure Multi-Party Computation (MPC)

What is the communication complexity of securely evaluating \( f \)?

- **HE:** \( \tilde{O}(|a, b, c| + |f(a, b, c)|) \) [G09,BGI16]
  - Based on heavy cryptographic tools
  - In practice: \((\alpha \cdot C)\) elements/party, small const \( \alpha \geq 2 \)
    - Long line of work improving \( \alpha \) in various settings

\[ C = (\text{Boolean/arithmetic}) \]

\[ \alpha \geq 2 \quad \text{Step 2: (standard) “active” security} \]

Lightweight \( \alpha = 1 \) 

\[ \text{Step 1: (weak) “passive” security} \]
Results

• Generic MPC: compiler from semi-honest to malicious
  • “Natural” protocols
  • Semi-honest majority
  • Any number of parties – secure with abort
  • Constant Number of parties – full security
  • In this talk – focus on 3PC
  • Sub-linear communication (in circuit size)
  • Soundness – $1/|F|$, but reduce by extension field

• Specific MPC functionalities
  • Even better communication!
# Comparison of 3PC Protocols

<table>
<thead>
<tr>
<th>The protocol</th>
<th># of elements sent per party per multiplication gate</th>
<th>Full security?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Boolean Circuits</td>
<td>Circuits over $\mathbb{F}_2^8$</td>
</tr>
<tr>
<td>Araki et al. [ABF+17]</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Chaudhari et al. [CCPS19]</td>
<td>7(offline)+4/3(online)</td>
<td>-</td>
</tr>
<tr>
<td>Chida et al. [CGH+18]</td>
<td>41</td>
<td>6</td>
</tr>
<tr>
<td>Eerikson et al. [EOP+19]</td>
<td>123</td>
<td>-</td>
</tr>
<tr>
<td>This work</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
3PC: Main Theorem

Given any **passive-secure 3PC** protocol with **“natural”** structure, then can achieve **active** security with \( +o(|C|)/\text{party extra comm} \).

**“Natural” 3PC protocol:**

**Input Shares of Adv:**
- Commit to his input

**Before final message:**
- Total random garbage

**If...** Some degree-2 relation holds on msgs

... then [robustly shared \( y \)] = \( C(a, b, c) \)

**Final round:** Robust shares of output
Natural Protocol – Example [AFLNO16]

Step 0: Represent $f$ as circuit

$X_{1,1}$
$X_{2,1}$
$X_{1,2}$
$X_{3,1}$
$X_{2,2}$
$X_{3,2}$

$+\quad *\quad +\quad *\quad f(x_1,x_2,x_3)$

$X_{1,1}$
$X_{2,1}$
$X_{1,2}$
$X_{3,1}$
$X_{2,2}$
$X_{3,2}$

$X_{1,1}$
$X_{2,1}$
$X_{1,2}$
$X_{3,1}$
$X_{2,2}$
$X_{3,2}$
Step 1: Secret Share inputs

- Party 1: $X$
- Party 2: $y$

Step 2: Secret Share zeros

- Party 1: $a+b+c=x$
- Party 2: $d+e+f=x$

- Party 1: $a, b$
- Party 2: $b, c$
- Party 3: $a, c$
- Party 1: $d, e$
- Party 2: $e, f$
- Party 3: $d, f$

Seed $k$

Long shared mask $_{12}$
Natural Protocol - Example

Step 3: distributed evaluation of every gate

$X \rightarrow a+b+c=x$

$y \rightarrow d+e+f=y$

$x \oplus y$

$x \cdot y$

Party 1: $a+d, b+e$

Party 2: $b+e, c+f$

Party 3: $a+d, c+f$

Mask
Re-share values
3PC “Passive” secure protocol

1. Secret share inputs
   (note: linear shares)

2. Generate $|C|$ sets of shares of 0

3. Gate-by-gate evaluation
   + : Locally on shares
   $x$ : Cross-terms $a_i b_j$ computable!
   Locally: Compute additive shares
   Compute, mask, & send share

4. Output gate: Exchange final shares

Comm Cost: 1 elmt/party/multiplication
Verifying Correct Execution

Party 1
- a, b
- d, e

Party 2
- b, c
- e, f

Party 3
- Shares of $af + cd + \text{masks}$
- Degree 2 function of shared input

$\text{mask}_{12}$
$\text{mask}_{13}$
$\text{mask}_{23}$
Collective 3PC Protocol

Protocol $\Pi'$ (without final message)

Final Message (robust shares, expect same message from two parties)

Each party proves in ZK their messages in $\Pi'$ were computed correctly

Fully secure – abort leads to identifying “good” party

Total extra communication:

$$|\text{proof}| + |\text{verifier comm}| = o(|C|)$$
3PC Summary

- Fully Linear PCPs: Proving on secret shared / committed / distributed data
- New (passive $\rightarrow$ active) security compiler for 3PC
- Concrete efficiency:
  - $2^{20}$ gate circuit
  - 0.5 Kbyte communication
  - 30 field operations per gate $\text{GF}(2^{47})$
  - Soundness $2^{-40}$

Protocols with a particular “natural” structure

(Standard) active security + $o(|C|)$ communication

(Weak) passive security
Extending to $n > 3$ Parties

- Challenge: **Malicious prover + verifier(s)**
  - Even defining soundness becomes non-trivial
  - Requires “robustness” of pieces of statement $x$

- Challenge: In MPC protocol with $n>3$, Prover no longer knows the full robust statement
  - Involves messages Prover wasn’t involved in

- Challenge: Replication based protocol inefficient for $\omega(1)$ servers

- Approach – Parties distribute role of prover. Stay tuned...
Thank You!
Applying Our Compiler to “Natural” Protocols

• 3 parties, 1 corruption (“3PC”)
  - Motivated setting: “Minimal” across MPC settings eg: [MRZ15,AFLN+16,ABFL+17,LN17,FLNW17,CGHI+18,GR018,NV18,EnOP+19]

• Comparison:
  - Over large field: $\alpha = 2$ [CGHIKLN18, NV18]
  - Over Boolean: $\alpha = 7$ [ABFLLNOWW17]
  - Any field or $\mathbb{Z}_N$: $\alpha = 1$ [This work]

• Constant $n \in O(1)$ parties, $t$ corruptions, $n = 2t + 1$
  - Over large field: $\alpha = 3$ [CGHIKLN18]
  - Over Boolean/$\mathbb{Z}_2^k$: $\alpha > 40$ [CGHIKLN18]
  - Any field or $\mathbb{Z}_N$: $\alpha = 3t/(2t + 1) \leq 1.5$ [This line of work]
  - Compiling eg, [AFLNO16,KKW18]