Practical Instances of MPC-in-the-Head

Carmit Hazay
Faculty of Engineering, Bar-Ilan University
Practical Instances of Zero-Knowledge Proofs
Taxonomy of Proofs

1. P vs NP
2. Interactive vs Non-interactive
3. Trusted setup vs No setup (transparent)
4. ZK vs (only) Soundness
5. Succinct vs Non-succinct
6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Taxonomy of Proofs

1. P vs NP

2. Interactive vs Non-interactive

3. Trusted setup vs No setup (transparent)

4. ZK vs (only) Soundness

5. Succinct vs Non-succinct

6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Taxonomy of Proofs

1. P vs NP

2. Interactive vs Non-interactive

3. Trusted setup vs No setup (transparent)

4. ZK vs (only) Soundness

5. Succinct vs Non-succinct

6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Taxonomy of Proofs

1. P vs NP

2. Interactive vs Non-interactive

3. Trusted setup vs No setup (transparent)

4. ZK vs (only) Soundness

5. Succinct vs Non-succinct

6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Taxonomy of Proofs

1. P vs NP
2. Interactive vs Non-interactive
3. Trusted setup vs No setup (transparent)
4. ZK vs (only) Soundness
5. Succinct vs Non-succinct
6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Taxonomy of Proofs

1. P vs NP

2. Interactive vs Non-interactive

3. Trusted setup vs No setup (transparent)

4. ZK vs (only) Soundness

5. Succinct vs Non-succinct

6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Taxonomy of Proofs

1. P vs NP
2. Interactive vs Non-interactive
3. Trusted setup vs No setup (transparent)
4. ZK vs (only) Soundness
5. Succinct vs Non-succinct
6. Public-Key Crypto vs (only) Symmetric-Key Crypto
Prior Approaches to “Practical” ZK

1. **Probabilistically Checkable Proofs (PCPs)**
   [BFLS91, Kil92, Mic94, ALMSS98, AS98, DL08, GKR08, GLR11, CMT12, BC12, DFH12, BCCT12, IMS12, Tha13, VSBW13], Interactive PCPs [KR08], Interactive Oracle PCPs [BCGT13, BCS16, RRR16, BCR16, BCGGHPST17, BBHR17, ZGKPP17-18, WTSTW18]

2. **Linear PCPs** [IKO07, Gro10, GGPR13, BCIOP13, Gro10, Lip12, SMBW12, Lip13, PGHR13, BCGTV13, FLZ13, SBBPW13, Lip14, DFGK14, KPPST14, ZPK14, CFHKKNPZ15, WSRBW15, BCTV14, BBFR15, Groth16, FFGKOP16, BFS16, BISW17, GM17, BBBPWM18]

3. **Multiparty Computation**
   [IKOS07, GMO15, CDGORSZ17, AHIV17, KKW18]
**ZKBoo:** Faster Zero-Knowledge for Boolean Circuits [GMO15]

Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives (ZKB++) [CDGORRSZ17]
Zero-Knowledge Proofs - A Reminder

- Goal: ZK proof for an NP-relation $R(x, w)$

- Towards using MPC:
  - Define n-party functionality
    \[ g(x; w_1, \ldots, w_n) = R(x, w_1 \oplus \ldots \oplus w_n) \]
  - Use OT-based MPC
    - Security in semi-honest model
    - Simple consistency check for dishonest majority when $n > 2$
Zero-Knowledge from 3-Party GMW [IKOS07,GMO15]

Use 3-party GMW protocol $\pi^{\text{OT}}$ for
$$g(x; w_1, w_2, w_3) = R(x, w_1 \oplus w_2 \oplus w_3)$$

commit to views $V_1, V_2, V_3$

random $i, j$

open views $V_i, V_j$

accept iff output=1

&

$V_i, V_j$ are consistent

Soundness error $\leq 2/3$
Prior Approached to “Practical” ZK

1. **Probabilistically Checkable Proofs (PCPs)**
   - [BFLS91, Kil92, Mic94, ALMSS98, AS98, DL08, GKR08, GLR11, CMT12, BC12, DFH12, BCCT12, IMS12, Tha13, VSBW13],
   - Interactive PCPs [KR08], Interactive Oracle PCPs [BCGT13, BCS16, RRR16, BCGRS16, BBCGGHPRSTV17, BBHR17, ZGKPP17-18, WTSTW18]

2. **Linear PCPs**
   - [IKO07, Gro10, GGPR13, BCIOP13, Gro10, Lip12, SMBW12, Lip13, PGHR13, BCGTV13, FLZ13, SBBPW13, Lip14, DFGK14, KPPSST14, ZPK14, CFHKKNPZ15, WSRBW15, BCTV14, BBFR15, Groth16, FFGKOP16, BFS16, BISW17, GM17, BBBPWM18]

3. **Multiparty Computation**
   - [IKOS07, GMO15, CDGORRSZ17, AHIV17, KKW18]
Ligero: Lightweight Sublinear Arguments Without a Trusted Setup
[AHIV17]
Main Result

Sublinear ZK arguments without trusted setup

- Simple, concretely efficient
- Symmetric-crypto only (e.g., SHA256)
- Post-Quantum Secure

First “sublinear” arguments for NP that avoid both complex PCP machinery and public-key crypto
Main Result

Sublinear ZK arguments without trusted setup

Concretely:

- **40-bit security**: comm. is \(0.5\sqrt{|C|}\) kb in the Boolean case
- **80-bit security**: Non-interactive via Fiat-Shamir
- Can handle Boolean or arithmetic circuits
- Prover computation: Merkle Tree \((O(\sqrt{|C|}) \text{ leaves})\) + \(O\left(\sqrt{|C|}\right)\) FFT’s of \(O\left(\sqrt{|C|}\right)\) evaluations
Eg, SHA256 certification with 40-bit security:
i.e. For statement $y$, prover proves knowledge of $x$ such that $\text{SHA256}(x) = y$

<table>
<thead>
<tr>
<th></th>
<th>Linear PCP [Pinocchio]</th>
<th>ZKBoo/++ [CDGORRSZ17]</th>
<th>Ligero</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communication</td>
<td>~ bytes</td>
<td>200 KB</td>
<td>34 KB</td>
</tr>
<tr>
<td>Prover time</td>
<td>mins</td>
<td>~33ms</td>
<td>140ms</td>
</tr>
<tr>
<td>Verif. time</td>
<td>&lt;10ms</td>
<td>~38ms</td>
<td>60ms</td>
</tr>
<tr>
<td>Asymptotic Communication</td>
<td>~ bytes</td>
<td>$O(</td>
<td>C</td>
</tr>
<tr>
<td>Trusted Setup</td>
<td>YES</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Amortization</td>
<td>NA</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>
Proof Schematic
\[ a \cdot b \geq X \cdot \#gates \]

Boolean: \( X = 2 \), AND/XOR

Arithmetic: \( X = 3 \), AND
Prover

Verifier

Root( )

\( f_1, f_2, f_3, \ldots \)

\( i_1, i_2, i_3, \ldots \)

Row-wise

\( O(b) \)
Prover

Verifier

Root(□)

\[ f_1, f_2, f_3, \ldots \]

\[ i_1, i_2, i_3, \ldots \]

O(b)
Proof Length: $O(b + \kappa \cdot a)$
Computation: $O(a)$ FFTs of $O(b)$
High-level approach: use **MPC in the head** [IKOS07]
- Transform Honest-majority MPC to ZK
- Optimized and implemented in [GMO16,CDGORRSZ17]

Can the communication be sublinear?
Communication complexity of (i.t.) MPC > circuit size

Key insight: Communication per party can be sublinear [DI06,IPS09]
High level approach: use **MPC in the head** [IKOS07]

- Transform Honest-majority MPC to ZK
- Optimized and implemented in [GMO16,CDGORRSZ17]

**Key insight:** Communication per party can be sublinear [DI06,IPS09]
Idea 1: Shamir Secret Sharing [S79]

Pick a random $t$-degree polynomial $p$ such that $p(0)$ is secret
Distribute $p(1), \ldots, p(n)$
$t$ shares do not reveal the secrets
Pick a random \( t+\ell \)-degree polynomial \( p \) such that \( p(0), p(-1), \ldots, p(-\ell) \) are secrets

Distribute \( p(1), \ldots, p(n) \)

\( t+\ell \) shares do not reveal the secrets
Idea 2: IPCP for Testing Interleaved RS Codes
Idea 2: IPCP for Testing Interleaved RS Codes
Idea 2: IPCP for Testing Interleaved RS Codes

\[ z(x) = \sum_{i} f_i p_i(x) \]

\[ f_1, f_2, f_3, \ldots \]

Prover

Verifier

\[ i_1, i_2, i_3, \ldots \]
Idea 2: IPCP for Testing Interleaved RS Codes

\[ z(x) = \sum_i f_i p_i(x) \]

Check
- \( z(x) \) is of degree \( t+\ell \)
- \( z(i) = \sum_i f_i p_i(i) \)
Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures [KKW18]
Use MPC-in-the-head in the **preprocessing model**
- Check consistency of preprocessing using cut-and-choose

MPC-in-the-head can be instantiated with dishonest majority protocols
- Semi-honest instances for generating correlated randomness
- Implies two versions of 5/3 rounds

Improves on Ligero proof size for circuits containing $\approx 300$-$100,000$ AND gates
Removing Interaction via the Fiat-Shamir Transform

Analysis can be extended to any constant round public-coin protocol [BCS16]
The signature scheme:

PK: \( y = \text{PRF}_k(0^k) \) where PRF is a block cipher

\( \text{Sig}(m) \): a proof for \((y,k)\) on a challenge \(H(a,m)\)

Based on symmetric-key primitives
Extensions to ring and group signatures
Practical Instances of Arithmetic Two-Party Computation
LevioSA: Lightweight Secure Arithmetic Computation from Any Passively Secure OLE [HIMV]
How is a function represented?
Classically, Boolean circuits [Yao86, GMW87,…]
Arithmetic Computation

- Many computations are done over an arbitrary field $\mathbb{F}$

- Notable examples:
  - SHA-256
  - Threshold cryptography [BF97, Gil99…]
  - Machine learning [LP00,…, JVC18, MR18, WCG18]
  - Pattern matching [HL08, HT10, …, KRT17]
  - Even BMR garbling [LPSY15,…]
This Talk

- Two-party
- Active security
- Arithmetic circuits

Goal: reduce communication and computation concrete costs of securely evaluating an arithmetic circuit over a field $\mathbb{F}$ with active security
Atomic Building-Block

- Oblivious linear evaluation (OLE)
  - A generalization of oblivious-transfer

\[ a, b \in \mathbb{F} \]

\[ ax + b \]
Prior Approaches to Practical Arithmetic 2PC

1. 2PC in the OLE-hybrid [IPS09, DGNNR17]
   • Black-box calls to OLE
   - 22 calls to active OLE

2. 2PC in the OT-hybrid [Gil99, KOS16, FPY18]
   • Black-box calls to OT
   - $6 \log(|\mathbb{F}|)$ calls to active OT

3. 2PC based on semi-homomorphic encryption [BDOZ11, DPSZ12, KPR18]
   - Based on concrete (non-standard) assumptions
Main Result

**Theorem 1:** Actively secure 2PC that makes $O(1)$ invocations of any passive OLE implementation per multiplication

For “nice” circuits our communication overhead is 2

**Theorem 2:** Actively secure OLE protocol that makes 2 invocations of any passive OLE implementation

[DGNNR17]: 22 black-box calls to any active OLE

[GN17]: active OLE from 2 calls to specific passive OLE
The Combined Protocol

1. Parties emulate the servers using passive 2PC
   a) Server’s inputs are shared between the parties
2. Enforce honest majority among servers by ”watching”
   a) Alice obtains Bob’s shares for k-out-of-n servers
   b) Bob obtains Alice’s shares for k-out-of-n servers
Active OLE from Passive OLE

\[ a_1, a_2, \ldots, a_m \]
\[ b_1, b_2, \ldots, b_m \]
\[ A_1, A_2, \ldots, A_n \]
\[ B_1, B_2, \ldots, B_n \]
\[ x_1, x_2, \ldots, x_m \]
\[ X_1, X_2, \ldots, X_n \]
Active OLE from Passive OLE

\[ A_1, B_1, X_1 \]
\[ A_2, B_2, X_2 \]
\[ \ldots \]
\[ A_n, B_n, X_n \]

\[ a_1, a_2, \ldots, a_m \]
\[ b_1, b_2, \ldots, b_m \]
\[ A_1, A_2, \ldots, A_n \]
\[ B_1, B_2, \ldots, B_n \]
\[ x_1, x_2, \ldots, x_m \]
\[ X_1, X_2, \ldots, X_n \]
Active OLE from Passive OLE

\[ A_1 X_1 + B_1 \]
\[ A_2 X_2 + B_2 \]
\[ \ldots \]
\[ A_n X_n + B_n \]

\[ a_1, a_2, \ldots, a_m \]
\[ b_1, b_2, \ldots, b_m \]
\[ A_1, A_2, \ldots, A_n \]
\[ B_1, B_2, \ldots, B_n \]

\[ x_1, x_2, \ldots, x_m \]
\[ C_1, C_2, \ldots, C_n \]
\[ c_1, c_2, \ldots, c_m \]

\[ X_1, X_2, \ldots, X_n \]
Black-Box Based on Any Passive OLE

1. More flexibility
   • Use any existing approach to passive OLE (e.g., lattice-based, group-based, code-based, etc.)
     • Does not need “ZK friendliness”
     • Off-the-shelf software/hardware implementation

2. Bonus feature
   “error-correct” weak implementations of passive OLE efficiently [in progress]
   • Constant correctness error
   • Constant privacy error
Summary

MPC-in-the-head is a useful tool for designing practical protocols

• Highly flexible and can be instantiated with different building blocks

• Optimized protocols achieve better parameters

• Much to explore in the context of concrete efficiency