Scalable Transparent ARguments-of-Knowledge

Michael Riabzev

Department of Computer Science, Technion

DIMACS Workshop on Outsourcing Computation Securely

Joint work with Eli Ben-Sasson, Iddo Bentov, and Yinon Horesh
Talk outline

- Our result
- Novel theory review (Low degree testing)
- Concrete implementation performance review
Our result

Features

A peak under the hood
  Improvements
  Novel low-degree test
  Measurements

Summary
Today I will tell you about STARK:

- “Scalable Transparent ARGument of Knowledge”
- New construction (theory+implementation\(^1\)) featuring:
  - Perfect witness-indistinguishability
  - Publicly verifiable
  - No trusted-setup
  - Universal
  - Succinct verification
- And additionally:
  - Post-quantum secure
  - Scalable prover (quasi-linear)

\(^1\)Proof-of-concept in C++
Our result

Features

A peak under the hood
Improvements
Novel low-degree test
Measurements

Summary
Computational model

Interactive Oracle Proofs (IOP)[BCS16, RRR16]²:

- A generalization of IP[GMR89] and PCP[BFL91, AS98]
- Verifier interacts with the Prover
- Prover’s messages too big for the verifier to read entirely
  - Also known as oracles

²also known as PCIP in [RRR16]
Our result

Features

A peak under the hood

Summary

Computational model

Interactive Oracle Proofs (IOP)[BCS16, RRR16]$^2$:

- A generalization of IP[GMR89] and PCP[BFL91, AS98]
- Verifier interacts with the Prover
- Prover’s messages too big for the verifier to read entirely
  - Also known as oracles

Realistic argument-system:

- Using Merkle trees [Kil92, Kil95, Mic00, BCS16]
- Noninteractive system: Fiat-Shamir heuristic

\[ \text{also known as PCIP in [RRR16]} \]
Cryptographic assumption

- Inner protocol (IOP):  
  - Provably sound\(^3\)  
  - Provably perfect zero-knowledge
- Compilation to (noninteractive) argument system:  
  - Using the random oracle model
- Implementation:  
  - Simulating a random-oracle using a hash-function

\(^3\)Implementation uses security conjectures to improve concrete efficiency.
Our result

Features

A peak under the hood

   Improvements
   Novel low-degree test
   Measurements

Summary
Our result

Features

A peak under the hood

Improvements

Novel low-degree test

Measurements

Summary
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

- Privacy — witness indistinguishability based on [BCGV16]
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

- Privacy — witness indistinguishability based on [BCGV16]
- Arithmetization — optimized for interactive systems
  - Disclaimer: RAM usage introduces $\sim 8T \log T$ additive overhead to witness size
  - in addition to $O(T)$ witness size when no RAM is used
  - Derived from SCI
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

- Privacy — witness indistinguishability based on [BCGV16]
- Arithmetization — optimized for interactive systems
  - Disclaimer: RAM usage introduces \( \sim 8T \log T \) additive overhead to witness size
  - in addition to \( O(T) \) witness size when no RAM is used
  - Derived from SCI
- Low degree test — optimized for interactive systems
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

• Privacy — witness indistinguishability based on [BCGV16]
• Arithmetization — optimized for interactive systems
  • Disclaimer: RAM usage introduces $\sim 8T \log T$ additive overhead to witness size
  • in addition to $O(T)$ witness size when no RAM is used
  • Derived from SCI

• Low degree test — optimized for interactive systems
• Hash-tree commitment — optimization based on queries pattern
  • Reducing communication complexity
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

- Privacy — witness indistinguishability based on [BCGV16]
- Arithmetization — optimized for interactive systems
  - Disclaimer: RAM usage introduces $\sim 8T \log T$ additive overhead to witness size
  - in addition to $O(T)$ witness size when no RAM is used
  - Derived from SCI
- Low degree test — optimized for interactive systems
- Hash-tree commitment — optimization based on queries patter
  - Reducing communication complexity
- System — code optimizations
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

- Privacy — witness indistinguishability based on [BCGV16]
- Arithmetization — optimized for interactive systems
  - Disclaimer: RAM usage introduces $\sim 8T \log T$ additive overhead to witness size
  - in addition to $O(T)$ witness size when no RAM is used
  - Derived from SCI
- Low degree test — optimized for interactive systems
- Hash-tree commitment — optimization based on queries patter
  - Reducing communication complexity
- System — code optimizations
STARK (this work) introduces improvements over SCI [BBCGGHPRSTV17] in several aspects: (Ben-Sasson, Bentov, Chiesa, Gabizon, Genkin, Hamilis, Pergament, R, Silberstein, Tromer, Virza)

- Privacy — witness indistinguishability based on [BCGV16]
- Arithmetization — optimized for interactive systems
  - Disclaimer: RAM usage introduces $\sim 8T \log T$ additive overhead to witness size
  - in addition to $O(T)$ witness size when no RAM is used
  - Derived from SCI
- Low degree test — optimized for interactive systems
- Hash-tree commitment — optimization based on queries patter
  - Reducing communication complexity
- System — code optimizations

In this talk we focus on the novel low-degree test
<table>
<thead>
<tr>
<th>Our result</th>
<th>Features</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A peak under the hood</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Improvements</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Novel low-degree test</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Measurements</td>
<td></td>
</tr>
</tbody>
</table>

Our result

Features

A peak under the hood

Improvements

Novel low-degree test

Measurements

Summary
IOPP novel low-degree test

Theorem ([BBHR17])

Given oracle access to an evaluation $f : S \rightarrow \mathbb{F}_2^n$ over $\mathbb{F}_2$ linear subspace $S \subset \mathbb{F}_2^n$, there is an IOPP protocol to verify $f$ is close to degree $d < \frac{|S|}{3}$, with the following properties:

- **Total proof size** $< \frac{|S|}{2}$.
- **Round complexity** $\frac{\log |S|}{2}$.
- **Prover complexity** $< 4|S|$ arithmetic operations over $\mathbb{F}_2^n$.
  - Highly parallelizable.
- **Query complexity is** $2 \log |S|$.
- **Soundness:**
  \[
  \Pr[\text{Reject} | \text{dist}(f, C) = \delta] \geq \min \left( \delta, \frac{1}{4} - \frac{3d}{4|S|} \right) - 3 \frac{|S|}{|\mathbb{F}_2^n|}.
  \]
  - Close to $\delta$ in the unique-decoding-radius.
  - Shown to be tight there.
Low-degree testing in the Interactive-Oracle-Proof model

- **Redundancy addition:** Prover transforms univariate polynomial $p(x)$ into a bivariate polynomial $Q(x, y)$

- **Invariant:** $\deg_y(Q) = \deg(p)/4$

- **Verification:** Verifier chooses random $x_0$ and verifies $q(y) = Q(x_0, y)$ is low-degree
  - By repeating the test recursively
  - Until $\deg(q)$ is small enough
Our result: Features

A peak under the hood

Summary

---

**Low-degree testing — more details**

The transformation $T : \mathbb{F}[x] \rightarrow \mathbb{F}[x, y]$ is basically a biased version of [?] :

- $p(x) \in \mathbb{F}[x]$ is evaluated over $V = \text{Span}\{b_1, b_2, \ldots, b_n\}$

- Define:
  - $V_0 := \text{Span}\{b_1, b_2\}$
  - $V_1 := \text{Span}\{b_3, \ldots, b_n\}$
  - $Z_{V_0}(x) := \prod_{\alpha \in V_0} (x - \alpha)$

- $T(p) = Q(x, y)$ where $Q(x, y) := p(x) \mod (y - Z_{V_0}(x))$

- Features:
  - $\forall x : Q(x, Z_{V_0}(x)) = p(x)$
  - $\deg_x(Q) < 4$
  - $\deg_y(Q) = \deg(p)/4$
Low-degree testing — advantages of interactivity

- Deeper recursion is possible due to provers adaptivity
- ‘Lightweight’ prover algorithm
- Better soundness:
  - Rows are low degree by definition
  - Any column can be queried
<table>
<thead>
<tr>
<th>Our result</th>
<th>Features</th>
<th>A peak under the hood</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Improvements</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Novel low-degree test</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Measurements</td>
<td></td>
</tr>
</tbody>
</table>

**Summary**
Benchmark: Forensics DNA blacklist

- FBI has the forensics DB
- 🕵️‍♂️ knows hash digest of the DB
  - Davies-Meyer-AES160
- FBI provide Andy’s DNA profiling\(^4\) result with an integrity proof
- The program verified:

```python
def prog(database):
    currHash = 0
    for currEntry in database:
        if currEntry matches Andy\'s DNA:
            REJECT
            currHash = Hash(currEntry, currVal)
        if currHash == expectedHash:
            ACCEPT
        else:
            REJECT
```

\(^4\) Based on https://www.fbi.gov/services/laboratory/biometric-analysis/codis
Machine specifications:
**Prover:** CPU: 4 X AMD Opteron(tm) Processor 6328 (32 cores total, 3.2GHz), RAM: 512GB
**Verifier:** CPU: Intel(R) Core(TM) i7-4600 2.1GHz, RAM: 12GB, Circuit: runtime simulated for long inputs
**Security:** Security level: 80 bits (Probability of cheating $< 2^{-80}$)

Conclusions: Prover asymptotic behaviour as predicted; Proving is $\sim \times 50K$ slower than program execution

Conclusions: Verifier asymptotic behaviour as predicted; Speedup achieved only for a few generated arguments
Comparison to other approaches

**Machine specifications:**
*CPU:* 4 X AMD Opteron(tm) Processor 6328 (32 cores total, 3.2GHz), *RAM:* 512GB

**Benchmark:**
Executing subset-sum solver for 64K TinyRAM steps (9 elements — exhaustive algorithm).

<table>
<thead>
<tr>
<th>Prover (mins)</th>
<th>Verifier (mSec)</th>
<th>Comm. (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>4.2 days</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>28min</td>
<td>452K</td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>42M</td>
</tr>
<tr>
<td>41</td>
<td>9</td>
<td>19G</td>
</tr>
<tr>
<td>4.2 days</td>
<td>28min</td>
<td>43M</td>
</tr>
</tbody>
</table>

Comparison to other systems - lower is better (log scale)

- **STARK**
- **SCI**[BBCGGHPRSTV17] — based on IOP.
- **IVC**[BCTV14] — Incrementally Verifiable Computation based on KOE. Setup required (succinct).

Fastest prover;
Verification ∼ fastest so far;
CC lowest; Argument ∼ ×1K longer “best”
<table>
<thead>
<tr>
<th>Our result</th>
<th>Features</th>
<th>A peak under the hood</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Improvements</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Novel low-degree test</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Measurements</td>
<td></td>
</tr>
</tbody>
</table>

Summary
Summary

STARK Introduction:

New low-degree test:

Concrete measurements:
Summary

STARK Introduction:

New low-degree test:

Concrete measurements:

THANK YOU
Sanjeev Arora and Shmuel Safra.
Probabilistic checking of proofs: a new characterization of NP.

Preliminary version in FOCS ’92.

Eli Ben-Sasson, Alessandro Chiesa, Ariel Gabizon, and Madars Virza.
Quasilinear-size zero knowledge from linear-algebraic PCPs.

Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner.
Interactive oracle proofs.

László Babai, Lance Fortnow, and Carsten Lund.
Non-deterministic exponential time has two-prover interactive protocols.

Preliminary version appeared in FOCS ’90.

Shafi Goldwasser, Silvio Micali, and Charles Rackoff.
The knowledge complexity of interactive proof systems.
Preliminary version appeared in STOC ’85.

Joe Kilian.
A note on efficient zero-knowledge proofs and arguments.

Joe Kilian.
Improved efficient arguments.
Silvio Micali.

Computationally sound proofs.
Preliminary version appeared in FOCS '94.

Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum.
Constant-round interactive proofs for delegating computation.