From PCP to ZK-STARK

Eli Ben-Sasson | Chief Scientist (East) | February 2019
Overview

1. Crypto proofs
   - PCP, IOP
   - STIK, STARK
   - FRI

2. Concrete Questions

zk-STARK | February 2019
Proofs of Computational Integrity (CI)

INTEGRITY

The quality of being honest
(Dictionary)
Proofs of Computational Integrity (CI)

**INTEGRITY**

The quality of being honest

(Dictionary)

**COMPUTATIONAL INTEGRITY**

The quality of a computation being executed honestly
Proofs of Computational Integrity (CI)

**INTEGRITY**

The quality of being honest
(Dictionary)

**COMPUTATIONAL INTEGRITY**

The quality of a computation being executed honestly

**CI Statement:** total=$138.16

**Prover:** Party producing proof
(here: Grocer)

**Verifier:** Party checking proof
(here: Customer)

Generic CI statement: Computation $C$, with public input $x$ and auxiliary private input $w$, reached output $y$ in $T$ steps
Proofs of Computational Integrity (CI)

**INTEGRITY**

The quality of being honest (Dictionary)

**COMPUTATIONAL INTEGRITY**

The quality of a computation being executed honestly

Grocery receipts are proofs of computational integrity

- Verification via naive re-execution of computation
- Proof is (i) deterministic, (ii) error free, (iii) one-shot (non-interactive)

Generic CI statement: Computation $C$, with public input $x$ and auxiliary private input $w$, reached output $y$ in $T$ steps
Proofs of Computational Integrity (CI)

INTEGRITY

The quality of being honest (Dictionary)

COMPUTATIONAL INTEGRITY

The quality of a computation being executed honestly

Grocery receipts are proofs of computational integrity

- Verification via naive re-execution of computation
- Proof is (i) deterministic, (ii) error free, (iii) one-shot (non-interactive)
- Modern CI proofs have (i) randomness, (ii) small error, (iii) interaction; in return, offer many benefits...

Generic CI statement: Computation $C$, with public input $x$ and auxiliary private input $w$, reached output $y$ in $T$ steps
Generic CI statement: Computation $\mathcal{C}$, with public input $x$ and auxiliary private input $w$, reached output $y$ in $T$ steps
Modern Computational Integrity proofs [GMR85]

**IP, ZK, CS, PCP, MIP, IPCP, LPCP, PCIP, IOP, ...**

**Privacy (Zero Knowledge, ZK):** Prover’s private inputs are shielded.

Generic CI statement: Computation $C$, with public input $x$ and auxiliary **private input $w$**, reached output $y$ in $T$ steps.
Modern Computational Integrity proofs [GMR85]

Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded

Scalability: for computation lasting $T$ cycles, proofs
- generated in $\sim T$ cycles (quasi-linear in $T$), and
- verified exponentially faster than $T$ ($\sim \log T$ cycles)

Generic CI statement: Computation $C$, with public input $x$ and auxiliary private input $w$, reached output $y$ in $T$ steps
Modern Computational Integrity proofs [GMR85]

Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded

Scalability: for computation lasting $T$ cycles, proofs
- generated in $\sim T$ cycles (quasi-linear in $T$), and
- verified exponentially faster than $T$ ($\sim \log T$ cycles)

Universality (Turing Completeness): apply to any computation $C$

Generic CI statement: Computation $C$, with public input $x$ and auxiliary private input $w$, reached output $y$ in $T$ steps
Modern Computational Integrity proofs [GMR85]

Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded

Scalability: for computation lasting T cycles, proofs
- generated in \( \sim T \) cycles (quasi-linear in T), and
- verified exponentially faster than T (\( \sim \log T \) cycles)

Universality (Turing Completeness): apply to any computation \( C \)

Transparency: All verifier messages are public random coins

Generic CI statement: Computation \( C \), with public input \( x \) and auxiliary private input \( w \), reached output \( y \) in \( T \) steps
Modern Computational Integrity proofs [GMR85]

IP, ZK, CS, PCP, MIP, IPCP, LPCP, PCIP, IOP, ...

- Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded
- Scalability: for computation lasting $T$ cycles, proofs
  - generated in $\sim T$ cycles (quasi-linear in $T$), and
  - verified exponentially faster than $T$ ($\sim \log T$ cycles)
- Universality (Turing Completeness): apply to any computation
- Transparency: All verifier messages are public random coins

Privacy examples

- Zcash shielded transaction: shield payer, payee and payment amount
- Paid taxes on all my 2018 transactions, without revealing them
- My crypto exchange is in the black, without showing my positions
- ...

\[ \deg(f(x) \mod Z_H(x)) < |H| - 1 \]

\[ (x - \beta^2)^z = a(w^2 + w) \land (z^2 = z) \sum_{h} f(h) = 0 \iff \deg(l) = 0 \]

zk-STARK | February 2019
Modern Computational Integrity proofs [GMR85]

Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded

Scalability: for computation lasting T cycles, proofs
- generated in ~ T cycles (quasi-linear in T), and
- verified exponentially faster than T (~ log T cycles)

Universality (Turing Completeness): apply to any computation

Transparency: All verifier messages are public random coins

Scalability examples

<table>
<thead>
<tr>
<th>Naïve computation time</th>
<th>Verifier time</th>
<th>Prover time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mega ($2^{20}$)</td>
<td>400</td>
<td>400-Mega</td>
</tr>
<tr>
<td>Giga ($2^{30}$)</td>
<td>900</td>
<td>900-Giga</td>
</tr>
<tr>
<td>Tera ($2^{40}$)</td>
<td>1600</td>
<td>1600-Tera</td>
</tr>
<tr>
<td>Peta ($2^{50}$)</td>
<td>2500</td>
<td>2500-Peta</td>
</tr>
</tbody>
</table>

Modern Computational Integrity proofs [GMR85]

Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded

Scalability: for computation lasting T cycles, proofs
- generated in ~ T cycles (quasi-linear in T), and
- verified exponentially faster than T (~ log T cycles)

Universality (Turing Completeness): apply to any computation

Transparency: All verifier messages are public random coins

Scalability examples

<table>
<thead>
<tr>
<th>Naïve computation time</th>
<th>Verifier time</th>
<th>Prover time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mega ($2^{20}$)</td>
<td>400</td>
<td>400-Mega</td>
</tr>
<tr>
<td>Giga ($2^{30}$)</td>
<td>900</td>
<td>900-Giga</td>
</tr>
<tr>
<td>Tera ($2^{40}$)</td>
<td>1600</td>
<td>1600-Tera</td>
</tr>
<tr>
<td>Peta ($2^{50}$)</td>
<td>2500</td>
<td>2500-Peta</td>
</tr>
</tbody>
</table>
Modern Computational Integrity proofs [GMR85]

**Privacy (Zero Knowledge, ZK):** Prover’s private inputs are shielded

**Scalability:** for computation lasting $T$ cycles, proofs
- generated in $\sim T$ cycles (quasi-linear in $T$), and
- verified exponentially faster than $T$ ($\sim \log T$ cycles)

**Universality (Turing Completeness):** apply to any computation

**Transparency:** All verifier messages are public random coins

### Scalability examples

**Proof scalability can solve blockchain scalability problems**
- Suppose computing latest Bitcoin state takes $1$Peta ($2^{50}$) steps
- A single prover spends $2500 \cdot 2^{50}$ steps, posts proof
- All other nodes verify exponentially faster, in $2500$ steps
Modern Computational Integrity proofs [GMR85]

Privacy (Zero Knowledge, ZK): Prover’s private inputs are shielded

Scalability: for computation lasting T cycles, proofs
  - generated in ~ T cycles (quasi-linear in T), and
  - verified exponentially faster than T (~ log T cycles)

Universality (Turing Completeness): apply to any computation

Transparency: All verifier messages are public random coins

**tl;dr my research: PCP-based proofs, concrete efficiency**

- 1995: ZK w/ scalable verifier was “galactic algorithm”
- 2018: scalable ZK realized in code for meaningful computation
- using **scalable PCPs** and **Interactive Oracle Proofs (IOPs)**
Modern Computational Integrity proofs [GMR85]

IP, ZK, CS, PCP, MIP, IPCP, LPCP, PCIP, IOP, ...

- **Privacy (Zero Knowledge, ZK):** Prover’s private inputs are shielded
- **Scalability:** for computation lasting T cycles, proofs
  - generated in ~ T cycles (quasi-linear in T), and
  - verified exponentially faster than T (~ log T cycles)
- **Universality (Turing Completeness):** apply to any computation
- **Transparency:** All verifier messages are public random coins

**tl;dr my research:** PCP-based proofs, concrete efficiency

Many flavors of proof systems

Variety of theoretical constructions (past 30 yrs)

PCP based, linear PCPs, elliptic curve+pairing based succinct NIZKs, proofs for muggles, quadratic span/arithmetic programs (QAP/QSP), interactive oracle proofs (IOP), ...
Many flavors of proof systems

Variety of theoretical constructions (past 30 yrs)

PCP based, linear PCPs, elliptic curve+pairing based succinct NIZKs, proofs for muggles, quadratic span/arithmetic programs (QAP/QSP), interactive oracle proofs (IOP), ...

...and implementations (past 5 yrs)

Pinocchio, libsnark, Zcash, Pepper, Buffet, ZKboo, Ligero, Bulletproofs, Hyrax, libstark, Aurora, ...

See zk.science
Overview

- 1: Cryptoproofs
- 2: PCP, IOP, STIK, STARK, FRI
- 3: Concrete Questions
zk-STARK definition [BBHR18]

An argument system is a zk-STARK if it satisfies:

zk **zero knowledge**: private inputs are shielded

S **Scalable**: proofs for CI of computation lasting $T$ cycles are
- generated in roughly $T$ cycles (quasi-linear in $T$), and
- verified exponentially faster than $T$ (roughly $\log T$ cycles)

T **Transparent**: verifier messages are random coins; no trusted setup

AR **Argument of Knowledge**: proof can be generated only by party knowing private input (formally: an efficient procedure can extract the secrets from a prover)
zk-STARK definition [BBHR18]

An argument system is a zk-STARK if it satisfies:

**zk**
- zero **knowledge**: private inputs are shielded

**S**
- Scalable: proofs for CI of computation lasting $T$ cycles are
  - generated in roughly $T$ cycles (quasi-linear in $T$), and
  - verified exponentially faster than $T$ (roughly $\log T$ cycles)

**T**
- Transparent: verifier messages are random coins; no trusted setup

**AR**
- Argument of Knowledge: proof can be generated only by party knowing private input (formally: an efficient procedure can extract the secrets from a prover)

- STARKs may be interactive (use blockchain as source of transparent randomness), gives shorter & safer proofs
- 1st STARK implementation: SCI-POC [BCG+16]; 1st zk-STARK: libstark [BBHR18]
"In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware" [Babai, Fortnow, Levin, 1991]

**Setup:** to prove $x \in L$ for some $L \in NTIME(T(n))$

- Verifier has oracle access to PCP $\pi$,
- Verifier runs in time $\text{poly}(n + \log (T(n)))$
- If $x \in L$ then exists $\pi$ accepted w.p. 1
- If $x \in L$ then all $\pi$ rejected w.p. $> \frac{1}{2}$
PCP and scalability [BFL, BFLS, AS, AL, K, M 1991-4]

1995

Proof activity time

Computation time

Naive:
- Verification = proving

PCP:
- Poly-logarithmic verification

\[ \kappa = \frac{T}{T_V} \]
PCP and scalability [BFL, BFLS, AS, ALMSS, K, M 1991-4]

1995

Proof activity time

\[ \kappa = \frac{T}{T_V} \]

Computation time

Verifier compute/unit of time

Naive:
- Verification = proving

PCP:
- Poly-logarithmic verification

Verifiable activity time

Naive verifier

Veriﬁer compute/unit of time

naive verifier
PCP and scalability [BFL, BFLS, AS, ALMSS, K, M 1991-4]

Naive:
- Verification = proving

PCP:
- Polynomial proving time
- Poly-logarithmic verification

\[ \kappa = \frac{T}{T_V} \]

Proof activity time

Computation time

Proofer compute/unit of time

Verifier compute/unit of time
PCP and scalability [BS, BGHSV, D, M, 2003-8]

Naive:
Verification = proving

PCP:
Quasi-linear proving
Poly-logarithmic verification

\[ \kappa = \frac{T}{T_V} \]
PCPs and polylogarithmic verification

“In this setup, a single reliable PC can monitor the operation of a herd of supercomputers working with possibly extremely powerful but unreliable software and untested hardware. “ [Babai, Fortnow, Levin, 1991]

**Setup:** to prove $x \in L$ for some $L \in NTIME(T(n))$

- Verifier has oracle access to PCP $\pi$,
- Verifier runs in time $\text{poly}(n + \log (T(n)))$
- If $x \in L$ then exists $\pi$ accepted w.p. 1
- If $x \in L$ then all $\pi$ rejected w.p. $> \frac{1}{2}$
Interactive Oracle Proofs (IOP) [RRR16, BCS16]

**IOP setup:** to prove $x \in L$ for some $L \in NTIME(T(n))$

- Verifier has oracle access to 1st oracle $\pi_0$,
- Verifier sends public randomness $r_0$
Interactive Oracle Proofs (IOP) [RRR16, BCS16]

**IOP setup:** to prove \( x \in L \) for some \( L \in \text{NTIME}(T(n)) \)

- Verifier has oracle access to 1\(^{st}\) oracle \( \pi_0 \),
- Verifier sends public randomness \( r_0 \)
- Verifier has oracle access to 2\(^{nd}\) oracle \( \pi_0 \)
- ...

\[ = 0 \iff \deg(f(x) \mod Z_H(x)) < |H| - 1 \]
Interactive Oracle Proofs (IOP) [RRR16, BCS16]

**IOP setup:** to prove $x \in L$ for some $L \in \text{NTIME}(T(n))$

- Verifier has oracle access to 1st oracle $\pi_0$,
- Verifier sends public randomness $r_0$
- Verifier has oracle access to 2nd oracle $\pi_0$
- ...
- Verifier runs in time $\text{poly}(n + \log(T(n)))$
- If $x \in L$ then $\exists \pi_0, \pi_1, \ldots, \pi_t$ accepted w.p. 1
- If $x \in L$ then all $\forall \pi$ rejected w.p. $> \frac{1}{2}$
Scalable Transparent ARGument of Knowledge (STARK)

An argument system is a zk-STARK if it satisfies:

zk **zero knowledge**: private inputs are shielded

S **Scalable**: proofs for CI of computation lasting $T$ cycles are
- generated in roughly $T$ cycles (quasi-linear in $T$), and
- verified exponentially faster than $T$ (roughly log $T$ cycles)

T **Transparent**: verifier messages are random coins; no trusted setup

AR **ARGument of Knowledge**: proof can be generated only by party knowing private input (formally: an efficient procedure can extract the secrets from a prover)
IOP tl;dr

With IOPs, can prove results that are yet unknown in PCP model

• 2-round IOP with perfect ZK, non-adaptive verifier for NP [BCGV16]
• Doubly-efficient, constant round, Interactive Proofs [RRR16]
• $O(1)$-round IOP with linear bit-length proofs and constant query comp [BCGRS17]
• Proximity protocols for Reed-Solomon with linear arithmetic complexity and logarithmic query complexity [BBHR18]
• ...
• Concretely efficient ZK-STARKs [BBC+16, BBHR18, ...]
PCP and scalability [BS, BGHSV, D, M, 2003-8]

2006

Proof activity time

Prover compute/unit of time

Verifier compute/unit of time

Computation time

\[ \kappa = \frac{T}{T_V} \]

Commitment (Merkle tree root)

\[ \nu(Z) = x - \beta^2 \]

\[ \nu(H) = a(w^2 + w) \]

\[ \nu(x) = \left( \sum_z \nu(f(z)) + \sum_z \nu(f(0)) \right)^n \]

\[ \nu(H) = 0 \iff \deg f = 0 \]

\[ \deg f(x) \mod Z_H(x) < |H| - 1 \]
IOP and Scalability [BBC+16, BBHR18]

\[ \kappa = \frac{T}{T_V} \]

Verdicts:
- \( \pi_0 \)
- \( \pi_1 \)
- \( \pi_2 \)

Commitment (Merkle tree root)

Proof activity time

Computation time

naive verifier prover
IOP and Scalability

12/2018

Proof activity time

Prover compute/unit of time

Computation time

Proof activity time (ms)

STARK Prover/Naive ratio ~ 35X

STARK Prover

Naïve

STARK verifier

ZOOM IN

#Pedersen hashes (log scale)

1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192

STARK prover

STARK verifier

Naïve

T480 laptop

i7-8550U CPU @ 1.80GHz

Quad-core

32 GB DDR4 RAM

zk-STARK | February 2019
**Scalable Transparent IOP of Knowledge (STIK) [BBHR18]**

**Definition:**

An IOP for $L \in NTIME(T(n))$ is said to be:

- **Scalable** if both of the following hold:
  - Proving time $T_P = \tilde{O}(T(n)) + poly(n) = T(n) \cdot \log^0 T(n) + poly(n)$
  - Verifying time $T_V = poly \log T(n)$

- **Transparent** if all verifier messages are public random coins (Arthur-Merlin protocols)

- **IOP of Knowledge** if there exists an extractor $E$ that extracts in time $poly T(n)$ a witness $w$ for membership of $x \in L$, from ``good'' prover $P_x$
**Strict STIK (arithmetic complexity)**

**Definition:**

An STIK for \( L \in \text{NTIME}(T(n)) \) is

- **Strictly Scalable** if both of the following hold:
  - Proving time \( T_P = O(T(n) \log T(n)) \)
  - Verifying time \( T_V = O(\log T(n)) + \tilde{O}(n) \)

**Thm [B, Chiesa, Goldberg, Gur, Riabzev, Spooner, 2019]:**

Every \( L \in \text{NTIME}(T(n)) \) has a strict (ZK)-STIK, where \( T_P, T_V \) are measured using arithmetic complexity over field of size \( O(T(n)) \)

**Question:** Strict STIK, Boolean complexity?
Interactive Oracle Proofs of Proximity (IOPP) [RRR16, BCS16]

**IOPP**: to prove oracle \( f \) close to code \( C \subset F^n \)

- Verifier sends public randomness \( r_0 \)
- Verifier has oracle access to 1\textsuperscript{st} oracle \( \pi_1 \)
- ...
- Verifier runs in time \( \text{poly}(n + \log (T(n))) \)
- If \( f \in C \) then \( \exists \pi_0, \pi_1, \ldots, \pi_t \) accepted w.p. 1
- Otherwise, \( \forall \pi \) rejected w.p. \( >s(\Delta(f, C)) \)
- \( s \) is soundness function, want to maximize it
Fast Reed-Solomon IOPP (FRI) [B, Bentov, Horesh, Riabzev 2018]

Definition: Reed-Solomon code (low deg polys)

\[ RS[F, S, \rho] = \{ f: S \to F \mid \deg(f) < \rho |S| \} \]

Thm [BBHR 2018]:

\( \forall S \subseteq F, S \) is a group of size \( N = 2^n \), the code \( RS[F, S, \rho] \) has a (fast) IOPP with

- \( T_p \leq 6 \cdot N \)
- \( T_v \leq 21 \cdot \log N \)
- \( s(\delta) \geq \min\{\delta_0, \delta\} \) for \( \delta_0 \approx \frac{1-\rho^4}{4} \) \([\text{BKS18}]\)
- \( \delta_0 \approx 1 - \rho^\frac{1}{3} \) \([\text{BGKS19}]\)

Question: Is \( s(\delta) \geq \delta - \left( \frac{|S|}{|F|} \right)^{O(1)} \) for \( \delta \) as large as \( 1 - \rho \)?
Overview

1. Cryptoproofs
2. PCP, IOP, STIK, STARK, FRI
3. Concrete Questions

zk-STARK | February 2019
Theory questions with practical impact

1. Strict STIK, Boolean complexity
   - $T_P = O(T(n) \log T(n))$ and $T_P = O(\log T(n))$
   - Approach: use AG codes over constant alphabets
   - Requires quasi-linear encoding time for AG codes

2. Better soundness analysis for FRI
   - Is $s(\delta) \geq \delta - \left(\frac{|S|}{|F|}\right)^{O(1)}$ for $\delta$ greater than $1 - \rho^\frac{1}{3}$?
   - Reaching Johnson bound $(1 - \sqrt{\rho})$? Beyond it?

3. Sliding scale conjecture for IOP and STIK?
   - Currently soundness error greater than rate ($\rho$)
   - Want soundness error closer to $\text{poly}(\frac{1}{|F|})$
   - Perhaps simpler to solve for IOPs than for PCPs?
Crypto-Security Questions

1. **STARK-friendly crypto primitives**
   • SHA2/3 STARK “cost” ≈ 10⁴
   • Pedersen STARK “cost” ≈ 10³
   • MiMC/Jarvis/Friday “cost” ≈ 10² [AGRRT16, AD18]
   • How low can you get? Algebraic security analysis?

2. **STARK/STIK security analysis**
   • Best efficient attack on FRI? on other PCPs/PCPPs? [BBGR16]

3. **STARK-friendly commitments and accumulators**
   • Replace Merkle trees with more efficient data structures? [LM18, BBF18]
   • With Merkle trees in RO model, do you need \( \lambda \) or \( 2\lambda \) bits RO to reach \( 2^{-\lambda} \) soundness error?

4. **How to formally verify a STIK/STARK constraint system?**
Questions?

we’re hiring: jobs@starkware.co
learn more: workshop@starkware.co