Concretely efficient Computational Integrity (CI) from PCPs

Eli Ben-Sasson, Technion

June 2017

PCP efficiency

- Recent asymptotic progress: short proofs, few queries, large soundness
  - Quasilinear PCPs, $O(1)$ queries, polylog verifier [BS05,D08,BGHSV05,Mie08]
  - Nearly-linear PCPs, 3 bit queries, soundness $1/2 - o(1)$ [MR10]
  - Linear-length PCPs, n^{ϵ} queries [BKKMS16]
  - LTCs approaching GV bound, $\log n^{\log \log n}$ queries [GKORS17]
  - Linear-length 2-round IOP, 3 queries, soundness $1/2 - \epsilon$ [BCGRS17]
- This talk is about *concrete*, i.e., *non-asymptotic* PCPs
  1. Why should we care? (Decentralized crypto-currencies, for example)
  2. How should we measure progress? (compression functions)
  3. What do we study? (new IOPs, soundness upper bounds)
  4. Measurements

Decentralized crypto-currency evangelism

- Decentralized crypto-currencies
  - Fiat, in Latin, is "It shall be"
  - Fiat Money (€, $, ...) managed by Trusted Party (TP)
  - Bitcoin: Decentralized Fiat Money; "In Crypto We Trust"
  - Innovation: TP-based "societal function" replaced by algorithms !!
  - Which TP-based systems next? Law? Government?
- Abolishing TP creates a problem: Computational Integrity (CI)
  - CI problem: is the reported output of a computation correct?
  - Bitcoin's solution: naïve verification by re-execution
  - This solution harms privacy, fungibility and hence, adoption
- Cyrptographic proofs (IP, PCP, IOP,...) solve CI with
  1. **Efficiency:** verifying proofs $\ll$ executing computation [BFL90, BFLS91]
  2. **Privacy:** ZK arguments [Kilian92, Micali94]
- Zerocash [BCGGMTV13]: zkSNARKs enhance privacy, fungibility
  - ZCash: crypto-currency, launched Nov. 2016
- Given zkSNARKs, what do PCP-based ones add?
  - **Transparency**: AM protocols, verifier messages are public randomness

Overview

1. Motivation ✓
2. Complexity measures for concrete proof systems
   - definitions
   - compression measures
3. Concrete soundness
4. Measurements

Proof systems – Definitions

Definition

A proof system S for $L \in NTIME(T(n))$ is a pair $\mathrm{S} = (\mathrm{V}, \mathrm{P})$ of randomized interactive algs, satisfying

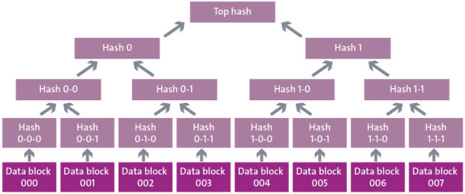
- **efficiency** V is randomized polynomial time; P unbounded
- **completeness** $x \in L \Rightarrow \Pr[\mathrm{V}(x) \leftrightarrow \mathrm{P}(x) \rightsquigarrow \text{accept}] = 1$
- **soundness** $x \notin L \Rightarrow \Pr[\mathrm{V}(x) \leftrightarrow \mathrm{P}(x) \rightsquigarrow \text{accept}] \leq 1/2$

Models of interactive systems

- IP [BM, GMR]: V, P send messages
- PCP [BFL]
  - P "sends" oracle π_1
  - V has random access to π_1
  - *query complexity*, denoted q, is # symbols read by V,
  - *proof length*, denoted ℓ, is $|\pi_1|$
- IOP/PCIP [BCS16,RRR16]
  - P "sends" oracle π_1
  - V sends randomness r_1
  - P "sends" oracle $\pi_2(r_1)$
  - V sends randomness r_2
  - ...
  - V has random access to $\pi_1, \ldots, \pi_r$
  - *query complexity* (q) is # symbols read by V from all oracles
  - *proof length* (ℓ) is $|\pi_1| + \ldots + |\pi_r|$
- IOPs offer results that are not known in PCP model
  - 2 rounds, perfect ZK for NP, scalable prover (run-time is $\tilde{O}(\mathcal{T} + \mathrm{k})$) [BCGV16]

The Kilian-Micali (KM) argument compiler

- 3 steps: (i) P *commits* oracle(s); (ii) V sends queries (public randomness); (iii) P opens commitments at relevant locations
- need *global* commitment c_π to π, *local* verfication of answers
- use hash $H : \{0,1\}^{2\lambda} \to \{0,1\}^{\lambda}$; λ is *security parameter*



- *global* commitment c_π is label of root
- *locally* verify answers by appending *authentication path* to c_π
- **Take-away: KM compiler increases answer size by $\lambda \cdot \log|\pi|$ bits**

The Kilian-Micali compiler

- 3 steps: (i) P *commits* oracle(s); (ii) V sends queries (public randomness); (iii) P opens commitments at relevant locations

Theorem ([BM88, GMR88, BFL88, BFL91 , BGKW88, FLS90, BFLS91, AS92, ALMSS92, K92, M94])

Each $L \in NEXP$ has an argument system $\mathsf{S} = (\mathsf{V}, \mathsf{P})$ *with*

- **scalable verifier:** *run-time* $poly(n, \log \mathcal{T})$*; this bounds proof length*
- **transparency:** *verifier messages are public random coins*
- **zero knowledge:** *proof preserves privacy of nondeterministic witness*
- *can be* **noninteractive** *assuming Random Oracle*

Lemma ([BCS16])

The KM compiler can be applied to a multi-round IOP, preserving soundness and ZK; assuming RO, can be noninteractive.

Overview

1. Motivation ✓
2. Complexity measures for concrete proof systems
   - definitions ✓
   - compression measures
3. Concrete soundness
4. Measurements

Concrete efficiency threshold [BCGT13]

- Tradeoff between prover complexity and verifier complexity
- How do we simultaneously improve both, for concrete inputs?
- Use complexity measures μ that penalize both complexities, like

$$\mu(n) = \frac{\ell(n)}{T(n)} \cdot q(n)$$

- Define the *concrete complexity threshold* as smallest n s.t.

$$\mu(n) < T(n)$$

- Now we can compare systems, measure progress ...
- Today: introduce complexity measures that have a concrete meaning

Compression ratio — PCP version

- Fix language $L \in NTIME(T(n))$ decided by M, and proof system S
- Let $w(n)$ denote witness size (for M)
- Let $q_\lambda(n)$ denote query complexity for soundness error $\leq 2^{-\lambda}$

Definition (Compression ratio and threshold)

The compression function of $L, M, \mathrm{S}, \lambda$ is witness/argument ratio,

$$C(n) = \frac{w(n)}{\lambda \cdot q_\lambda(n) \cdot \log \ell(n)}$$

and the compression threshold θ is minimal integer (if exists) s.t.

$$\forall n \geq \theta \quad C(n) \geq 1$$

Compression ratio — PCP version

Definition (Compression ratio and threshold)

The compression function of L, M, S, λ is witness/argument ratio,

$$C(n) = \frac{w(n)}{\lambda \cdot q_\lambda(n) \cdot \log \ell(n)}$$

and the compression threshold θ is minimal integer (if exists) s.t.

$$\forall n \geq \theta \quad C(n) \geq 1$$

Remarks

- higher $C(n)$ is better; lower θ is better
- $C(n)$ scales *logarithmically* with $\ell(n)$, but prover complexity scales super-linearly with $\ell(n)$
- doubly scalable systems have $C(n) \sim w(n)/\text{poly}(\log T(n))$; we care about concrete n

Compression ratio — IOP version

Definition (Compression ratio and threshold)

The compression function of $L, M, \mathsf{S}, \lambda$ is witness/argument ratio,

$$C(n) = \frac{w(n)}{\lambda \cdot q_\lambda(n) \cdot \log \ell(n)}$$

$$C(n) = \frac{w(n)}{\lambda \cdot \sum_{i=1}^{r} q_\lambda^i(n) \cdot \log \ell^i(n)}$$

and the compression threshold θ is minimal integer (if exists) s.t.

$$\forall n \geq \theta \quad C(n) \geq 1$$

$C(n)$ for IOP with proofs $\pi^1, \ldots, \pi^r$ and q_λ^i queries to π^i is ...

Which language to compress?

- the hash of a sequence $w_1, \ldots, w_n, w_i \in \{0,1\}^\lambda$ is

$$\mathcal{H}(w_1, \ldots, w_n) = \begin{cases} H(w_1 \| w_2) & n = 2 \\ \mathcal{H}(H(w_1 \| w_2), (w_3, \ldots, w_n)) & \text{otherwise} \end{cases}$$

- suggestion: study the compression function and threshold of

$$L_H = \{(x, n) \mid \exists w = (w_1, \ldots, w_n), \mathcal{H}(w) = x\}$$

- Why this language?
  - stepping stone towards aggregating and compressing proofs
  - required for incrementally verifiable computation [V08, BCCT13]
  - side question: which H minimizes threshold for a given proof system?

Proximity proof systems – Definitions

- Scalable PCPs use PCPs of Proximity (PCPP) as building block
- PCPPs used to verify proximity of a purported codeword to a code
- IOPP generalize PCPP exactly like IOP generalizes PCP

Definition (IOPP)

An r-round IOPP for a family of codes $\mathcal{C}$ with proximity parameter δ (say, $\delta = \delta_{\mathcal{C}}/3$) is an $(r+1)$-round IOP; the first oracle (π_0), is a purported codeword, and

- **efficiency** V is randomized polynomial time; P unbounded
- **completeness** $\pi_0 \in C \Rightarrow \Pr[\mathrm{V} \leftrightarrow \mathrm{P} \rightsquigarrow \text{accept}] = 1$
- **soundness** $\Delta(\pi_0, C) > \delta, \Rightarrow \Pr[\mathrm{V} \leftrightarrow \mathrm{P} \rightsquigarrow \text{accept}] \leq 1/2$

A 1-round IOPP is a PCPP; a 0-round IOPP is an LTC.

IOPP compression

Definition (Compression ratio and threshold)

The compression function of $\mathcal{C}, \mathrm{S}, \delta, \lambda$ is code-dim/argument ratio,

$$\Theta(k) = \frac{k}{\lambda \cdot \sum_{i=1}^{r} q_\lambda^i(n) \cdot \log \ell^i(n)}$$

and the compression threshold θ is minimal integer (if exists) s.t.

$$\forall k \geq \theta \quad \Theta(k) \geq 1$$

Remarks

- code compression is cleaner problem than language compression
- for "PCP-friendly" codes (Hadammard, RS, RM, ...) code compression needed for language compression
- compression meaningful for LTCs (0 rounds) and PCPPs (1 round)

LTC compression – examples

- Hadamard: $\ell^0 = 2^k$; 3-query tester rejects δ-far words w.p. $\geq \delta$
  - so $q^0_\lambda = 3\lambda / \log(1/1-\delta)$, and

$$\Theta(k) = \frac{k}{\lambda \cdot 3\lambda / \log(1/1-\delta) \cdot \log 2^k} = \frac{\log(1/1-\delta)}{3\lambda^2} > 1$$

  - Corollary: Hadamard PCP, with KM-compiler, cannot compress any L
- Bivariate RM, fractional degree 1/2, code rate = 1/4,
  - $\sqrt{k}$ query tester rejects δ-far words w.p. $\geq \delta$
  - so $q^0_\lambda = \sqrt{k}\lambda / \log(1/1-\delta)$, and

$$\Theta(k) = \frac{k}{\lambda \cdot \sqrt{k}\lambda / \log(1/1-\delta) \cdot \log 4k} = \frac{\log(1/1-\delta) \cdot \sqrt{k}}{\lambda^2 \log 4k} = c_{\delta,\lambda} \cdot \frac{\sqrt{k}}{\log 4k}$$

- compression threshold for $\lambda = 128$ and $\delta = 1/8$ is $\approx 2^{40}$ or 1 Tera.

PCPP compression – examples

- Hadamard: $\ell^0 = 2^k$; 3-query tester rejects δ-far words w.p. $\geq \delta$
  - Corollary: Hadamard PCP, with KM-compiler, cannot compress any L
- Bivariate RM, fractional degree 1/2, code rate = 1/4,
  - $\sqrt{k}$ query tester rejects δ-far words w.p. $\geq \delta$
  - $\Theta(k) = c_{\delta,\lambda} \cdot \frac{\sqrt{k}}{\log 4k}$, $\theta_{128} \approx 2^{40}$
- Quaslinear Reed Solomon (RS) PCPP [BS05]
  - recursive construction, uses bivariate RM
  - with 1 level of recursion has similar compression to RM
  - with 2 levels $q \sim k^{1/4}$, soundness $\sim 3\delta/64$, so $\Theta(k) = c'_{\delta,\lambda} k^{3/4}$ and $\ldots \theta_{128} = 2^{31}$ or 2 Mega

New: Biased RS (BRS) IOPP (submitted) [BBHR17]

Theorem (RS proximity w/ linear arithmetic complexity)

Rate-1/4 RS codes have a $\frac{\log k}{2}$-round IOPP with $q = 2\log n$; rejection prob. $\geq \delta - o(1)$ for $\delta < \delta_C/4$, and moreover

- *given π_0, prover has total arithmetic complexity $< 6 \cdot n$*
- *Verifier decision circuit has total arithmetic complexity $< 21 \log n$*
- *Length of ith oracle is $\ell^i(n) = n/4^i$*

Remarks

- first proximity proof w/ linear prover-side arithmetic complexity and non-trivial q
- soundness + q combination better than [BS05]
- low "code complexity", parallelizable, implemented in STARK (later)

$$\Theta(k) = \frac{k}{\lambda^2 \cdot \log(1/1-\delta) \cdot 4^{\binom{(\log 4k)/2}{2}}} > c_{\delta,\lambda} \cdot \frac{2k}{(\log k + 2)^2}$$

Compression — summary

- Hadamard: no compression threshold
- RM: $\Theta(k) \sim k^{1/2}/\log k$, $\theta_{128} \approx 2^{40}$
- 2-level [BS05]: $\Theta(k) \sim k^{3/4}/\log k$, $\theta_{128} \approx 2^{31}$
- BRS-IOPP: $\Theta(k) \sim k/\log k$, $\theta_{128} \approx 2^{26}$
- even if soundness 1/2 requires only 1 query, $\theta_{128} \geq \lambda^2 = 2^{14}$
- for better compression, need tests with high soundness

Overview

1. Motivation ✓
2. Complexity measures for concrete proof systems ✓
   - definitions ✓
   - compression measures ✓
3. Concrete soundness
4. Measurements

Improving concrete soundness

- soundness parameter s: probability of rejecting false claim
- some PCPs have tight lower bounds on soundness ...
  - [Håstad 00]: 3-bit-query PCP, test is CNF clause, $s \geq 7/8 - \epsilon$
  - [Moshkovitz-Raz 08]: $q = 3$, $s \geq 7/8 - o(1)$, nearly-linear pf-length
  - [Raz-Safra 96]: Plane-vs.-plane test of RM codes, $q = n^{\epsilon}$, great soundness
- ... but use *concretely* long proofs, have large compression threshold
- concrete soundness of scalable PCP/IOP systems not tight
- consider PCPPs for RS codes, distance $\delta_C = 1 - \rho_C$
  - PCPP soundness analysis breaks at unique decoding radius ($\delta < \delta_C/2 = (1 - \rho)/2$)
  - goals: soundness for list-decoding radius (1-$\sqrt{\rho}$), and even capacity $(1 - \rho)$
  - bottleneck is the Polischuk-Spielman (PS) bivariate test [PS94]
  - [CMS17]: First PS soundness beyond unique-decoding radius
- [BBGR16]: initiate study of soundness *upper bounds*
  - no known non-trivial upper bounds on soundness, for any δ, even up to capacity $(1 - \rho)$

Compression using soundness upper bounds

Theorem (RS proximity w/ linear arithmetic complexity)

Rate-ρ RS codes have a $\frac{\log k}{2}$-round IOPP with $q = 2\log n$; rejection prob. $\geq \delta - o(1)$ for $\delta < (1-\rho)/4$, and moreover

- *given π_0, prover has total arithmetic complexity $< 6 \cdot n$*
- *Verifier decision circuit has total arithmetic complexity $< 21 \log n$*
- *Length of ith oracle is $\ell^i(n) = n/4^i$*

Conjecture (RS proximity w/ linear arithmetic complexity, to capacity)

Rate-ρ RS codes have a $\frac{\log k}{2}$-round IOPP with $q = 2\log n$; rejection prob. $\geq \delta - o(1)$ for $\delta < 1-\rho$, and moreover

- *given π_0, prover has total arithmetic complexity $< 6 \cdot n$*
- *Verifier decision circuit has total arithmetic complexity $< 21 \log n$*
- *Length of ith oracle is $\ell^i(n) = n/4^i$*

Overview

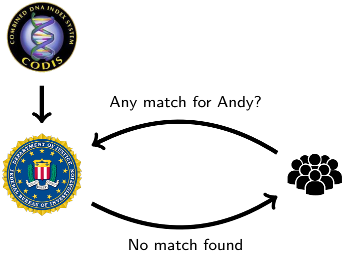
1. Motivation ✓
2. Complexity measures for concrete proof systems ✓
    - definitions ✓
    - compression measures ✓
3. Concrete soundness ✓
4. Measurements

Practical implementation [BBHR17]

- New implemented system – **(zk)STARK**
  - **Scalable**: quasilinear prover, polylog verifier
  - **Transparent**: AM protocol, verifier messages are public randomness
  - **ARgument of Knowledge**: can extract witness from "good" proof
  - Perfect ZK in IOP model [BCGV16, BCGRS17]; Computational ZK Kilian-Micali argument [BCS16]
  - "Post-quantum secure" – no number-theoretic assumptions
  - Uses BRS-IOPP (among other things)

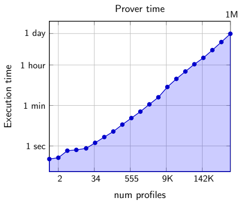
Practical zk-STARK benchmark: forensic DNA profile

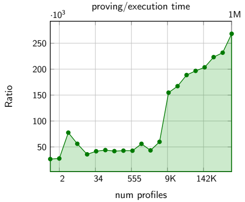
- FBI holds forensics DNA profile DB D
- knows $\mathcal{H}(D)$
  - Davies-Meyer-AES160
- FBI reports Andy's DNA profile match result, along with zk-STARK proof, $\lambda = 80$
- The program verified:

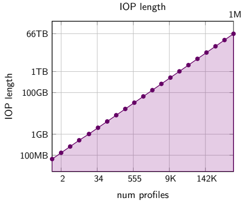



```
def prog(database):
    currHash = 0

    for currEntry in database:
        if currEntry matches AndysDNA:
            REJECT
        currHash = Hash(currEntry, currVal)

    if currHash == expectedHash : ACCEPT
    else : REJECT
```

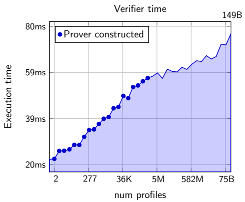
Machine specifications:
Prover: *CPU*: 4 X AMD Opteron(tm) Processor 6328 (32 cores total, 3.2GHz), *RAM*: 512GB
Verifier: *CPU*: Intel(R) Core(TM) i7-4600 2.1GHz, *RAM*: 12GB, *Circuit*: runtime simulated for long inputs
Security: *Security level*: 80 bits (Probability of cheating $< 2^{-80}$)

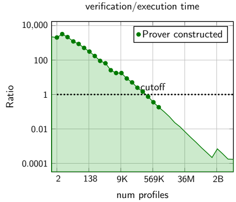


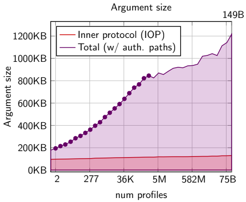




Conclusions: Prover asymptotic behaviour as predicted; Proving is $\sim \times 50K$ slower than program execution





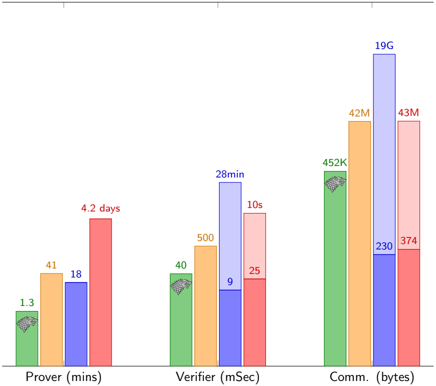


Conclusions: Verifier asymptotic behaviour as predicted; Speedup achieved only for a few generated arguments

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).

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



- **STARK**
- **SCI**[BBCGGHPRSTV17] — based on IOP.
- **KOE**[BCGTV13] — zkSNARK based on Knowledge Of Exponent hardness. **Non-succinct setup required.**
- **IVC**[BCTV14] — Incrementally Verifiable Computation based on KOE. **Setup required (succinct).**

Fastest prover; verifier nearly fastest; lowest total CC; argument $\sim \times 1K$ "best"

Concluding remarks

1. Motivation ✓
2. Complexity measures for concrete proof systems ✓
   - definitions ✓
   - compression measures ✓
3. Concrete soundness ✓
4. Measurements ✓

- attempting to implement "practical PCPs" led to new theory results
  - IOP model
  - scalable PZK for NEXP
  - RS proximity proofs with linear arith. comp.
  - ...
- and uncovered interesting theory questions
  - best compression ratio?
  - "proof-system friendly" crypto primitives?
  - soundness gaps for scalable PCPs?
  - concrete soundness beyond unique decoding radius?
- and lets us interact with new communities
  - crypto currencies