Computational Integrity with a Public Random String from Quasi-Linear PCPs

Michael Riabzev
Technion - Israel Institute of Technology

EUROCRYPT 2017

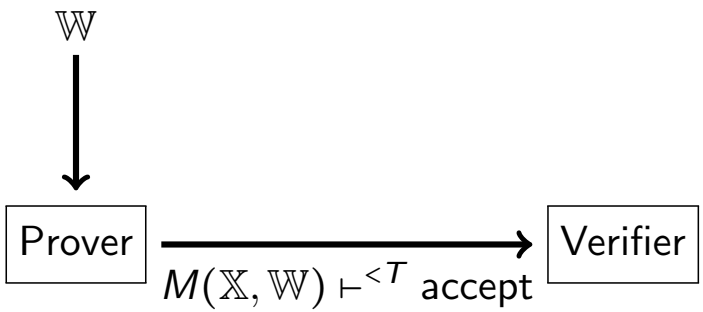
Joint work with Eli Ben-Sasson, Iddo Ben-Tov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Mark Silberstein, Eran Tromer and Madars Virza

Talk outline

Motivation

Definition (Computational-integrity[1](CI))

The language of triples $(M, \mathbb{X}, T)$ such that: Nondeterministic machine M accepts $\mathbb{X}$, within at most T steps (T is binary).



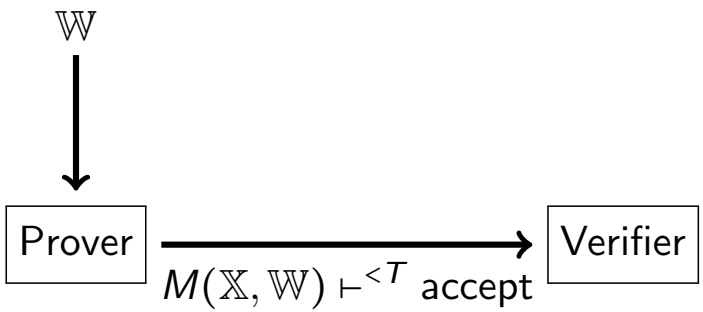
Goal: Practical CI system implementation (POC)
Take home message: Practical solutions without trusted-setup are achievable

[1]This problem also known as *checking* [BFLS91],*certifying* [Mic00],*delegating* [GKR08],and *verifying* [GGP10] (computations).

Our result

Today I will tell you about SCI:

- "Scalable Computational Integrity"
- First implementation[2] of a theoretical construction that achieves all of the below:
  - Publicly verifiable
  - No trusted-setup
  - Universal
  - Succinct verification



[2] Proof-of-concept in C++

Other approaches

- Designated-verifier/trusted-setup systems [IKO07, GGPR13, PGHR13, BCG+13, BCG+14, CFH+15, ...]
  - ☺Tiny proofs (hundreds of bytes)
  - ☺Very efficient verification (milliseconds)
  - ☹Designated-verifier...
  - ☹...or require a trusted-setup

3

Other approaches

- Designated-verifier/trusted-setup systems [IKO07, GGPR13, PGHR13, BCG+13, BCG+14, CFH+15, ...]
  - ☺Tiny proofs (hundreds of bytes)
  - ☺Very efficient verification (milliseconds)
  - ☹Designated-verifier...
  - ☹...or require a trusted-setup
- Non-universal systems [GKR08, RRR16, ...]
  - ☺No cryptographic assumptions
  - ☹Restricted class of programs

3

Other approaches

- Designated-verifier/trusted-setup systems [IKO07, GGPR13, PGHR13, BCG+13, BCG+14, CFH+15, ...]
  - ☺Tiny proofs (hundreds of bytes)
  - ☺Very efficient verification (milliseconds)
  - ☹Designated-verifier...
  - ☹...or require a trusted-setup
- Non-universal systems [GKR08, RRR16, ...]
  - ☺No cryptographic assumptions
  - ☹Restricted class of programs
- Non-succinct systems [Gro11, GMO16, ...][3]
  - ☺Efficient prover
  - ☹Verification time ~ program execution time

[3] Succinct communication-complexity in [Gro11]

Background

- Uses classical approach (PCP)
  [BM88, GMR89, BFL91, BGKW88, FLS99, BFLS91, AS98, ALM+92, Kil92, Mic00, . . .]
- With recent asymptotic improvements
  [BGH+05, BS08, BCS16]
- And concrete (non-asymptotic) constructions
  [BCGT13, CA15]

Cryptographic assumption

- Inner protocol (IOP[BCS16, RRR16][4]):
  - Provably sound[5].

[4] also known as PCIP in [RRR16]

[5] Implementation uses security conjectures to improve concrete efficiency.

Cryptographic assumption

- Inner protocol (IOP[BCS16, RRR16][4]):
  - Provably sound[5].
- Compilation to argument system:
  - Using the random oracle model.
  - Non-interactive using Fiat-Shamir heuristic.

[4] also known as PCIP in [RRR16]

[5] Implementation uses security conjectures to improve concrete efficiency.

Cryptographic assumption

- Inner protocol (IOP[BCS16, RRR16][4]):
  - Provably sound[5].
- Compilation to argument system:
  - Using the random oracle model.
  - Non-interactive using Fiat-Shamir heuristic.
- Implementation:
  - Treating the hash-function as a random-oracle.

[4] also known as PCIP in [RRR16]

[5] Implementation uses security conjectures to improve concrete efficiency.

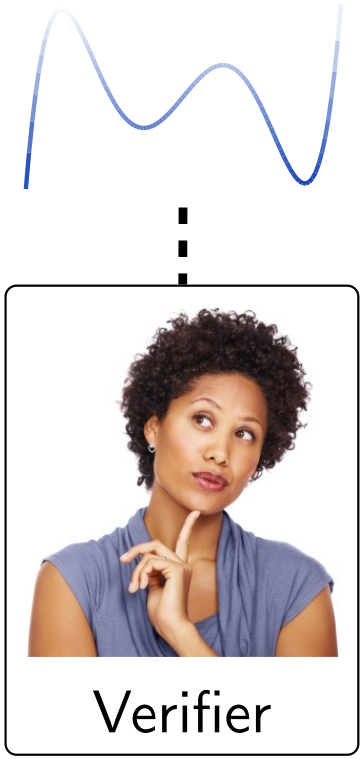
Protocol overview (based on [Kil92])

1. Prover constructs a proof for the CI claim
   - Proof is too big to be sent to verifier
   - Only Merkle commitment is passed to verifier
   - Interaction with verifier used to reduce load on prover
     - Formalized in [BCGRS17], to be presented in ICALP 2017
   - Time complexity $\tilde{O}(T)$
2. Verifier draws polylog(T) random queries to proof, sends them to prover
3. Prover answers queries
   - Merkle paths added for integrity with commitment
4. Verifier decides whether to accept
   - False-rejection impossible
   - False-acceptance with probability $< 2^{-80}$

Protocol overview (based on [Kil92])

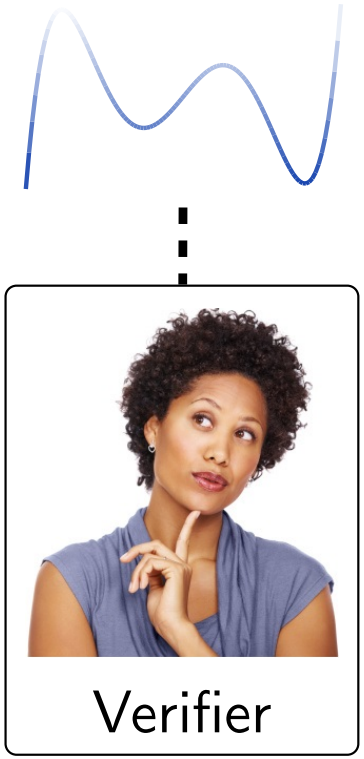
1. Prover constructs a proof for the CI claim
   - Proof is too big to be sent to verifier
   - Only Merkle commitment is passed to verifier
   - Interaction with verifier used to reduce load on prover
     - Formalized in [BCGRS17], to be presented in ICALP 2017
   - Time complexity $\tilde{O}(T)$
2. Verifier draws polylog(T) random queries to proof, sends them to prover
3. Prover answers queries
   - Merkle paths added for integrity with commitment
4. Verifier decides whether to accept
   - False-rejection impossible
   - False-acceptance with probability $< 2^{-80}$

Low-degree testing definition (informal)



Low-degree testing definition (informal)

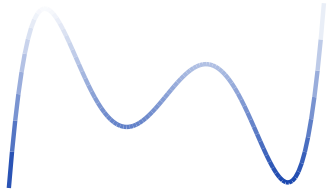
I wonder if this polynomial is of degree $< 2^n$. Too bad my time complexity is only poly (n) ☹



Low-degree testing definition (informal)

I wonder if this polynomial is of degree $< 2^n$. Too bad my time complexity is only poly (n) ☹

Of course it is low degree!



Prover

Verifier

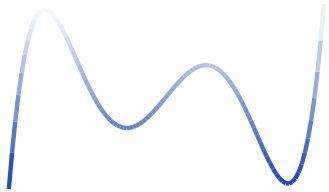
Low-degree testing definition (informal)

I wonder if this polynomial is of degree $< 2^n$. Too bad my time complexity is only poly (n) ☹

Of course it is low degree!

I don't know you, why would I trust you?

Prover



Verifier

Low-degree testing definition (informal)

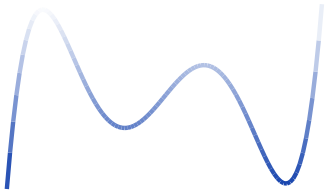
I wonder if this polynomial is of degree $< 2^n$. Too bad my time complexity is only poly (n) ☹

Of course it is low degree!

I don't know you, why would I trust you?

Don't trust—Verify! Here is a proof oracle! (PCPP)

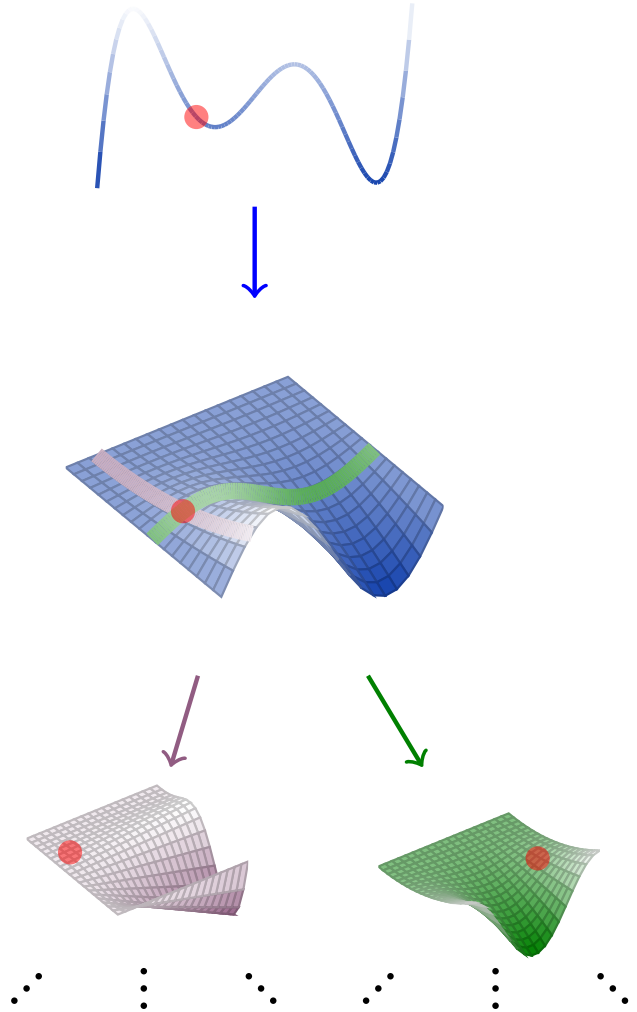
Prover



Verifier

Low-degree testing

- Low-degree testing is common in classical CI solutions
- SCI is the first system implementing succinct low-degree testing
  - Based on [BS08]
- In contrast: Trusted-setup systems use public-key cryptography that enforces low-degree polynomials
  - Using homomorphic encryption



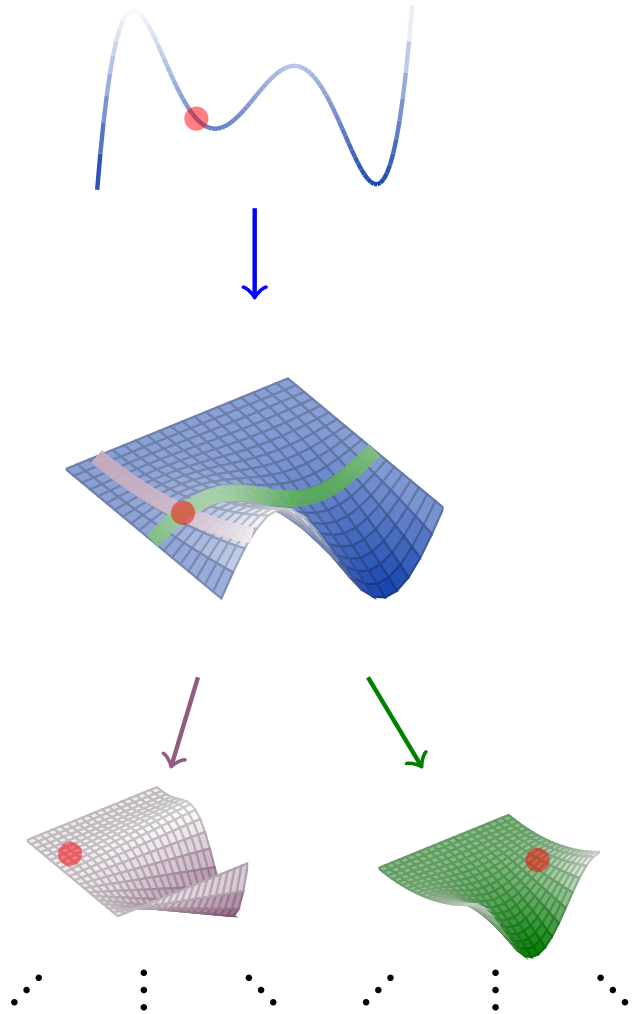
Low-degree testing — the [BS08] test

The [BS08] test:

Prover algorithm:

- Given a candidate $f : \mathbb{F} \to \mathbb{F}$ claimed to be of degree d
- The prover constructs $Q : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ s.t.

$$\deg_x(Q), \deg_y(Q) < \sqrt{d} \iff \deg(f) < d$$



Low-degree testing — the [BS08] test

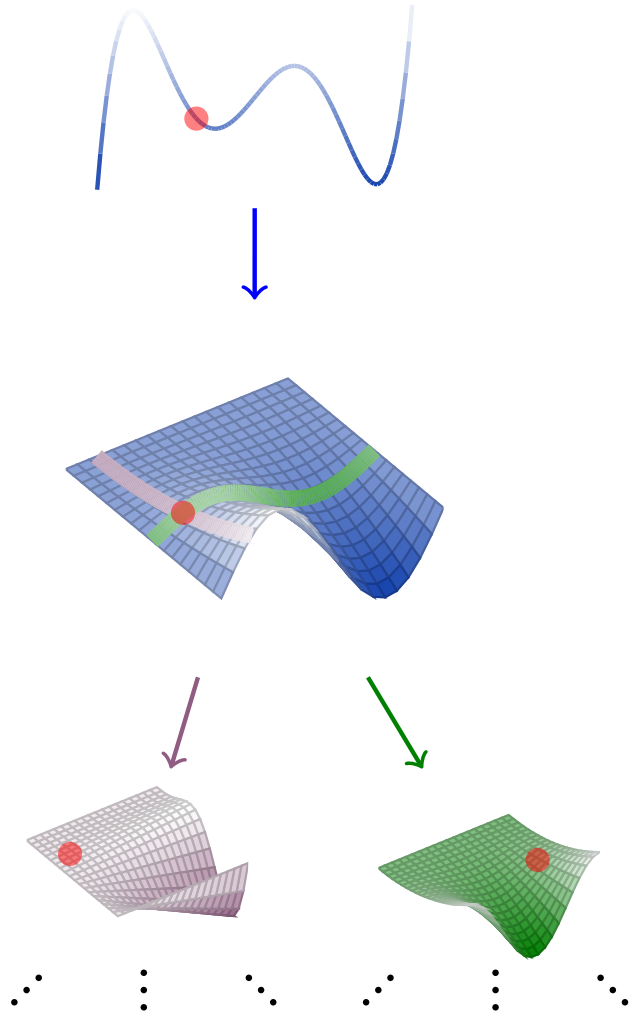
The [BS08] test:

Prover algorithm:

- Given a candidate $f : \mathbb{F} \to \mathbb{F}$ claimed to be of degree d
- The prover constructs $Q : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ s.t.

  $$\deg_x(Q), \deg_y(Q) < \sqrt{d} \iff \deg(f) < d$$

- Repeated recursively for Q's restrictions to rows and columns
  - Until degree small enough
  - Resulting in a proofs-tree



Low-degree testing — the [BS08] test

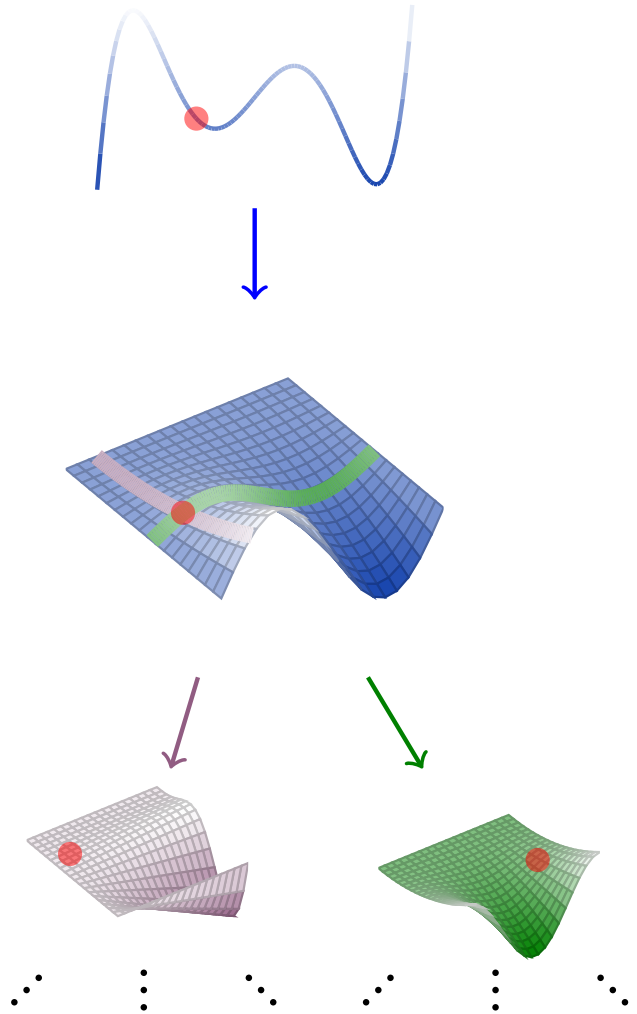
The [BS08] test:

Prover algorithm:

- Given a candidate $f : \mathbb{F} \to \mathbb{F}$ claimed to be of degree d
- The prover constructs $Q : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$ s.t.

$$\deg_x(Q), \deg_y(Q) < \sqrt{d} \iff \deg(f) < d$$

- Repeated recursively for Q's restrictions to rows and columns
  - Until degree small enough
  - Resulting in a proofs-tree

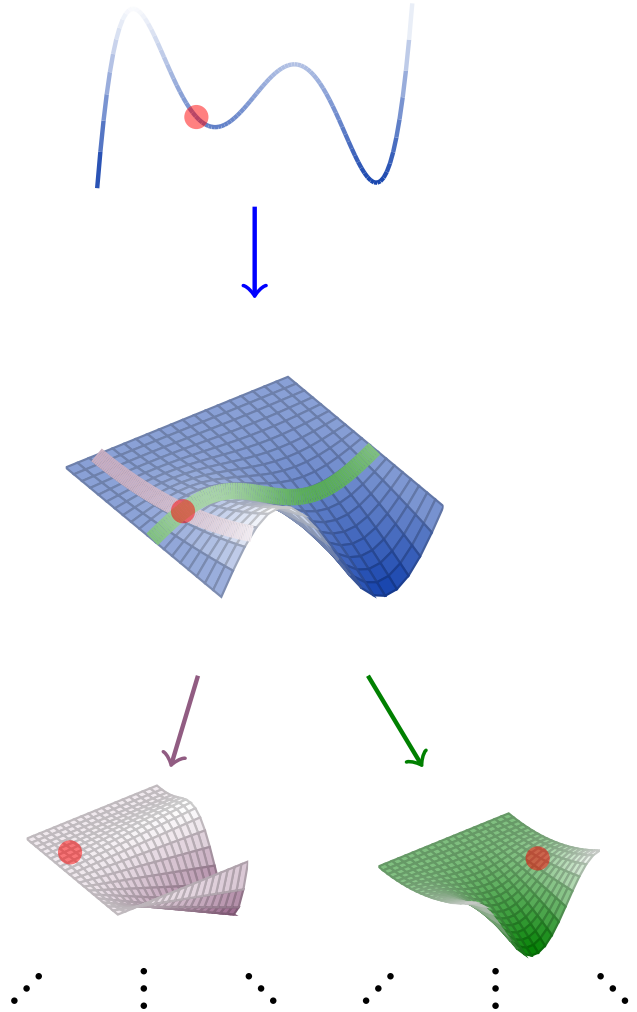


Verifier algorithm:

Verifier tests a small random fraction of leafs and consistency over their paths to the root

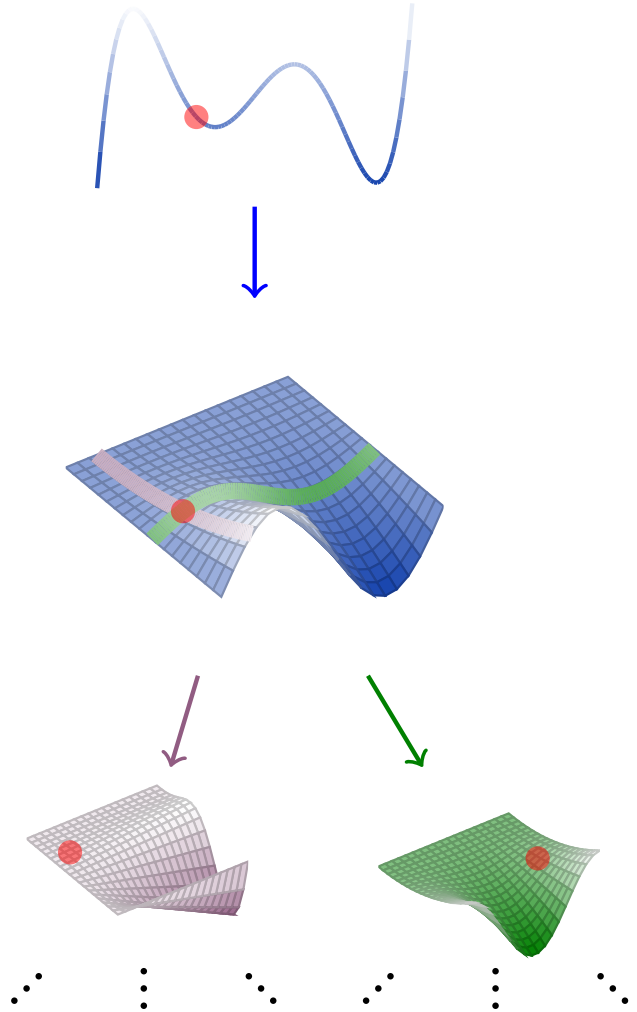
Low-degree testing — the [BS08] test

- Observation: most subproofs never accessed by verifier
- In the PCP model, queries are not known in advance, thus prover must construct the entire proofs-tree
  - Results in proof size $\Omega(2^n \cdot n)$
- Problem: too expensive for practical implementations



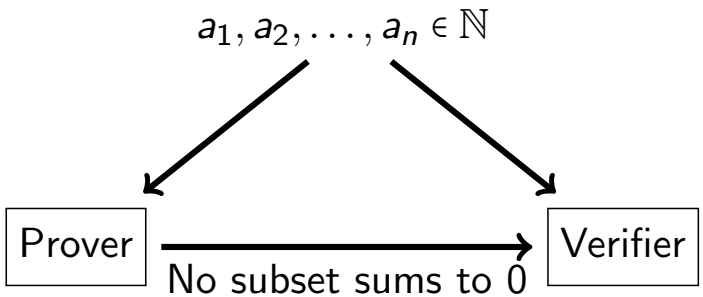
Low-degree testing — SCI solution

- SCI solves the problem by interaction
- The verifier guides the prover to construct subproofs only if accessed
- In our construction prover learns a subproof is accessed only after it's path to root is unchangeable
  - Soundness preserved
  - Proof length reduced to $O(2^n)$
- Formal method description in [BCGRS17] (ICALP)



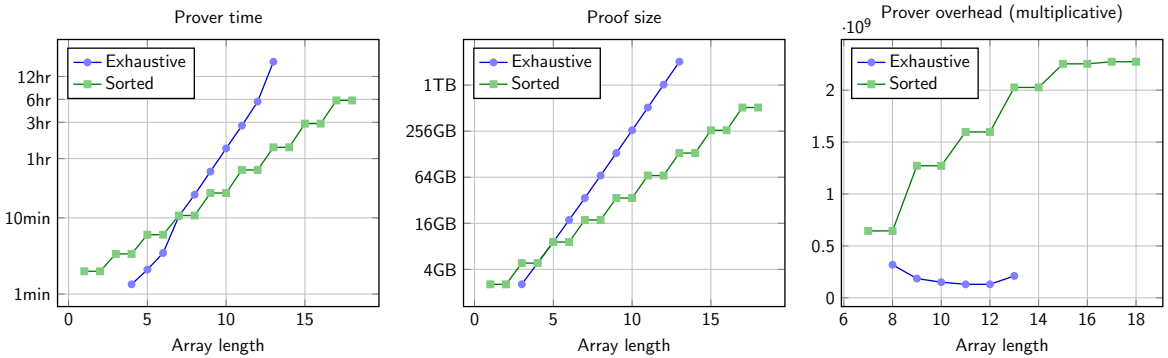
Benchmark — Subset Sum

- CI claim: "no nonempty subset of $\vec{a}$ sums to 0"
  - co-NP hard problem
- Two implementations in TinyRAM[6]:
  - Exhaustive: $\Theta(2^n)$-time, no RAM
  - Sorting: $\Theta(2^{n/2})$ time and space
    - RAM usage increases proof by $\times 2 \log(\text{exec-length}) = O(n)$

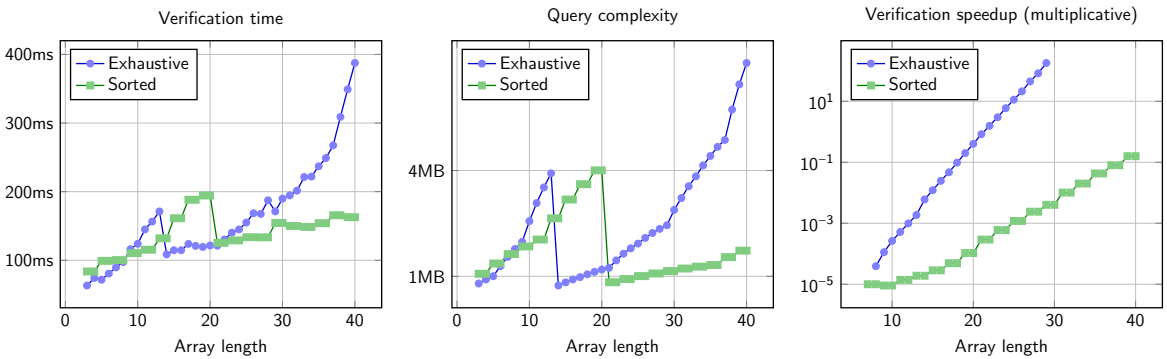


[6] Turing-complete assembly.

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 $< 2^{-80}$)



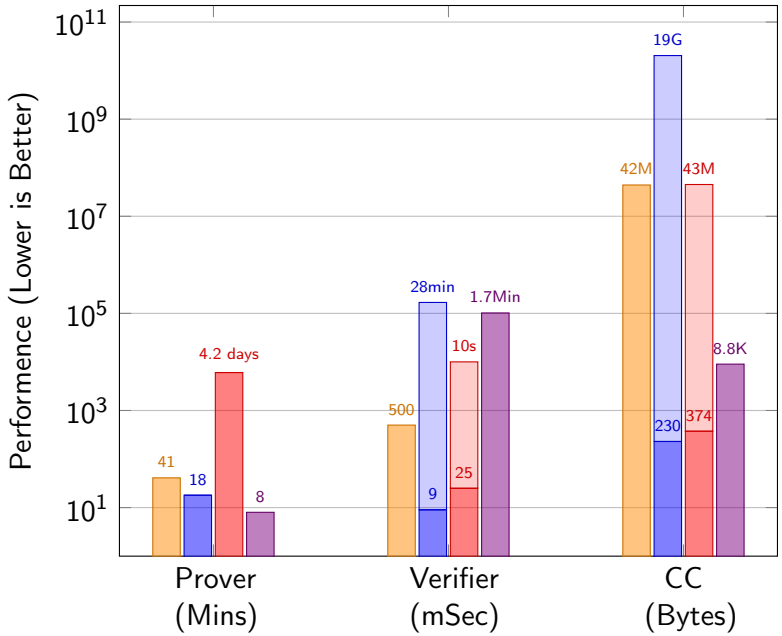
Conclusions: Prover asymptotic behaviour as predicted;
Proving is $\sim \times 10^9$ slower than program execution



Conclusions: Verifier asymptotic behaviour as predicted;
Succinct only for very long program executions

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).



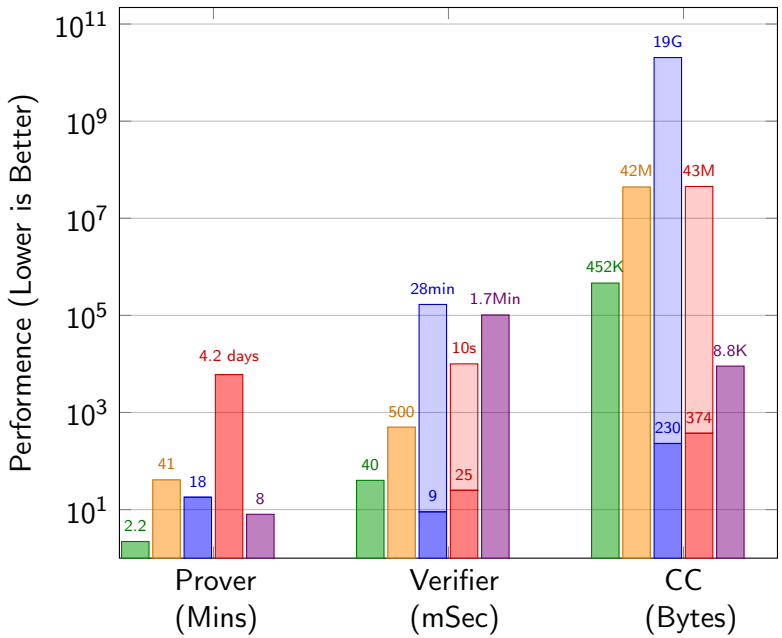
- **SCI** - our system.
- **KOE**[BCG+13] - zkSNARK based on Knowledge Of Exponent hardness. **Non-succinct setup required.**
- **IVC**[BCTV14] - Incrementally Verifiable Computation based on KOE. **Setup required (succinct).**
- **DLP**[Gro11] - Publicly-verifiable succinct CC but **non-succinct verification**. Based on hardness of DLOG[7].

Highlights: competitive prover;
Verification succinct but slow;
Communication succinct but high

[7] Extrapolated from [Gro11, Table 2]

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).



Highlights: Fastest prover;
Verification ~ fastest so far;
Communication greatly improved

- **SCI** - our system.
- **KOE**[BCG+13] - zkSNARK based on Knowledge Of Exponent hardness. **Non-succinct setup required.**
- **IVC**[BCTV14] - Incrementally Verifiable Computation based on KOE. **Setup required (succinct).**
- **DLP**[Gro11] - Publicly-verifiable succinct CC but **non-succinct verification**. Based on hardness of DLOG[7].
- **Follow-up (in-progress)** [BBHR17]
  - Same approach as SCI
  - Guaranties privacy (ZK)
  - Introduces new theory
  - Prover overhead ~ $\times 10^6$
  - Practical succinctness in-reach

[7] Extrapolated from [Gro11, Table 2]

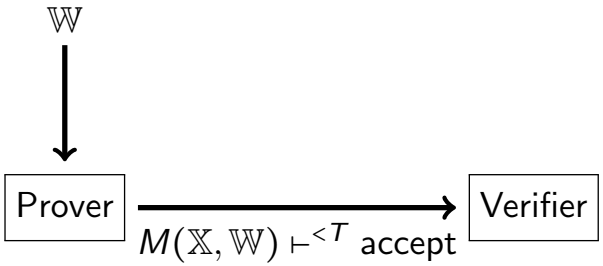
Acknowledgment

Research supported by:

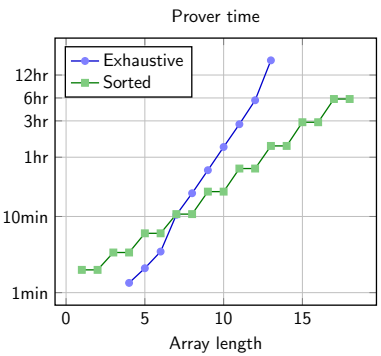
Programmers:

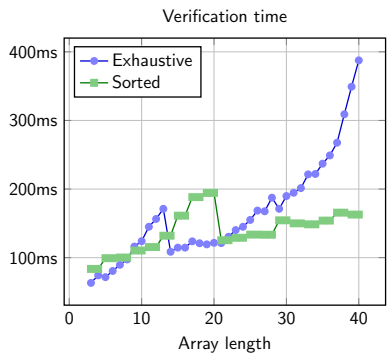
- Ohad Barta
- Lior Greenblatt
- Shaul Kfir
- Gil Timnat
- Arnon Yogev

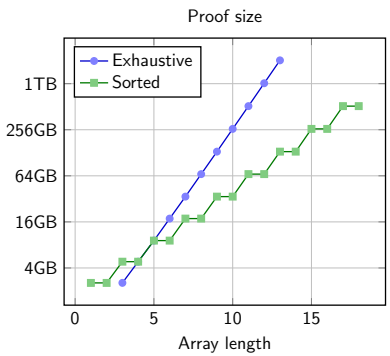
SCI Introduction:

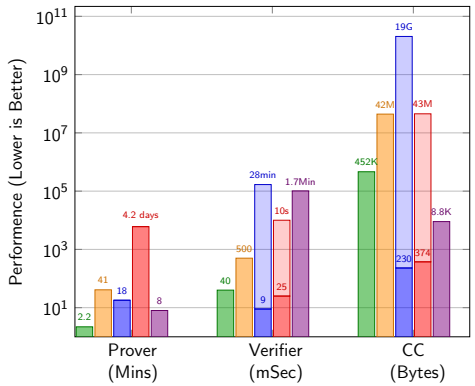


Measurements:

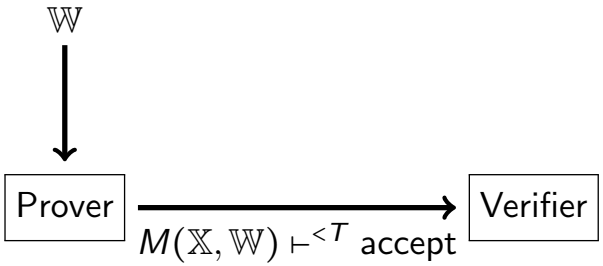




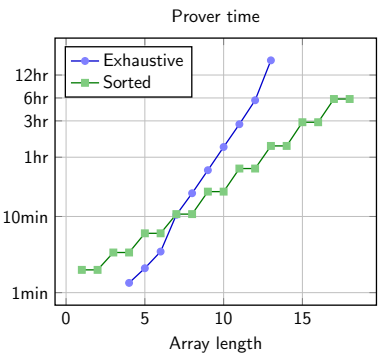


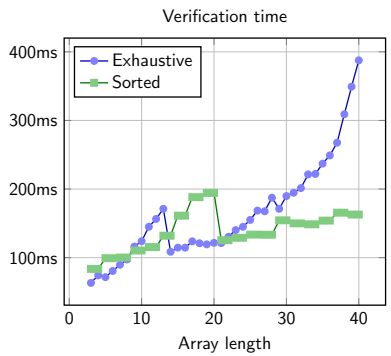


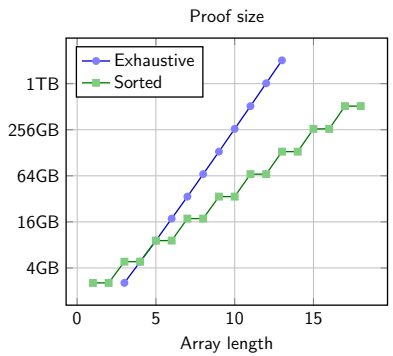
SCI Introduction:

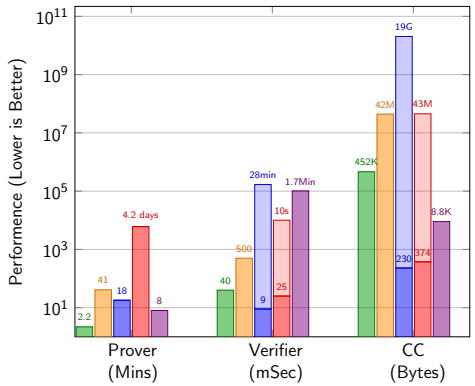


Measurements:









Questions?


Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy.
Proof verification and hardness of approximation problems.
In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 14–23, 1992.


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

Journal of the ACM, 45(1):70–122, 1998.
Preliminary version in FOCS '92.


Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza.
SNARKs for C: Verifying program executions succinctly and in zero knowledge.
In *Proceedings of the 33rd Annual International Cryptology Conference*, CRYPTO '13, pages 90–108, 2013.


Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza.
Zerocash: Decentralized anonymous payments from bitcoin.
In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474, 2014.


Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, and Eran Tromer.
Fast reductions from RAMs to delegatable succinct constraint satisfaction problems.
In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*, ITCS '13, pages 401–414, 2013.


Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner.
Interactive oracle proofs.
In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 31–60, 2016.


Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza.
Scalable zero knowledge via cycles of elliptic curves.
In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II*, pages 276–294, 2014.


László Babai, Lance Fortnow, and Carsten Lund.
Non-deterministic exponential time has two-prover interactive protocols.
Computational Complexity, 1:3–40, 1991.
Preliminary version appeared in FOCS '90.


László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy.
Checking computations in polylogarithmic time.
In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, STOC '91, pages 21–32, 1991.


Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil Vadhan.
Short PCPs verifiable in polylogarithmic time.
In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, CCC '05, pages 120–134, 2005.


Michael Ben-Or, Shafi Goldwasser, Joe Kilian, and Avi Wigderson.
Multi-prover interactive proofs: how to remove intractability assumptions.
In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, STOC '88, pages 113–131, 1988.


László Babai and Shlomo Moran.
Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes.
Journal of Computer and System Sciences, 36(2):254–276, 1988.


Eli Ben-Sasson and Madhu Sudan.

Short PCPs with polylog query complexity.
SIAM Journal on Computing, 38(2):551–607, 2008.
Preliminary version appeared in STOC '05.


Alessandro Chiesa and Zeyuan Allen Zhu.
Shorter arithmetization of nondeterministic computations.
Theor. Comput. Sci., 600:107–131, 2015.


Craig Costello, Cédric Fournet, Jon Howell, Markulf Kohlweiss, Benjamin Kreuter, Michael Naehrig, Bryan Parno, and Samee Zahur.
Geppetto: Versatile verifiable computation.
In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 253–270, 2015.


Uriel Feige, Dror Lapidot, and Adi Shamir.
Multiple noninteractive zero knowledge proofs under general assumptions.
SIAM Journal on Computing, 29(1):1–28, 1999.


Rosario Gennaro, Craig Gentry, and Bryan Parno.

Non-interactive verifiable computing: outsourcing computation to untrusted workers.
In *Proceedings of the 30th Annual International Cryptology Conference*, CRYPTO '10, pages 465–482, 2010.


Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova.
Quadratic span programs and succinct NIZKs without PCPs.
In *Proceedings of the 32nd Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT '13, pages 626–645, 2013.


Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum.
Delegating computation: Interactive proofs for Muggles.
In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, 2008.


Irene Giacomelli, Jesper Madsen, and Claudio Orlandi.
Zkboo: Faster zero-knowledge for boolean circuits.

In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 1069–1083, 2016.


Shafi Goldwasser, Silvio Micali, and Charles Rackoff.
The knowledge complexity of interactive proof systems.
SIAM Journal on Computing, 18(1):186–208, 1989.
Preliminary version appeared in STOC '85.


Jens Groth.
Efficient zero-knowledge arguments from two-tiered homomorphic commitments.
In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 431–448, 2011.


Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky.
Efficient arguments without short PCPs.
In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 278–291, 2007.


Joe Kilian.
A note on efficient zero-knowledge proofs and arguments.
In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC '92, pages 723–732, 1992.


Silvio Micali.
Computationally sound proofs.
SIAM Journal on Computing, 30(4):1253–1298, 2000.
Preliminary version appeared in FOCS '94.


Brian Parno, Craig Gentry, Jon Howell, and Mariana Raykova.
Pinocchio: Nearly practical verifiable computation.
In *Proceedings of the 34th IEEE Symposium on Security and Privacy*, Oakland '13, pages 238–252, 2013.


Omer Reingold, Guy N. Rothblum, and Ron D. Rothblum.
Constant-round interactive proofs for delegating computation.
In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 49–62, 2016.