Zerocash, Bitcoin, and Transparent Computational Integrity

Eli Ben-Sasson, Technion

Based on joint works with Iddo Ben-Tov, Alessandro Chiesa, Michael Forbes, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Ynon Horesh, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Nick Spooner, Eran Tromer, Madars Virza

January 2017

Overview

- Computational integrity and privacy (CIP) — motivation
- Importance of transparency
- A pair of new transparent CI(P) systems

Overview

- Computational integrity and privacy (CIP) — motivation
- Importance of transparency
- A pair of new transparent CI(P) systems

Brief bio:

- Professor at Technion — Israel Institute of Technology
- started w/ theoretical CS related to efficient proofs ("moon math")

Overview

- Computational integrity and privacy (CIP) — motivation
- Importance of transparency
- A pair of new transparent CI(P) systems

Brief bio:

- Professor at Technion — Israel Institute of Technology
- started w/ theoretical CS related to efficient proofs ("moon math")
- noticed (circa 2008) proofs efficient enough to reduce to practice
- traveled to Bitcoin 2013 to "show and tell" ...

Overview

- Computational integrity and privacy (CIP) — motivation
- Importance of transparency
- A pair of new transparent CI(P) systems

Brief bio:

- Professor at Technion — Israel Institute of Technology
- started w/ theoretical CS related to efficient proofs ("moon math")
- noticed (circa 2008) proofs efficient enough to reduce to practice
- traveled to Bitcoin 2013 to "show and tell" ...
- ... Zerocash co-author, ZCash co-founder & scientist

Overview

- Computational integrity and privacy (CIP) — motivation
- Importance of transparency
- A pair of new transparent CI(P) systems

Brief bio:

- Professor at Technion — Israel Institute of Technology
- started w/ theoretical CS related to efficient proofs ("moon math")
- noticed (circa 2008) proofs efficient enough to reduce to practice
- traveled to Bitcoin 2013 to "show and tell" ...
- ...Zerocash co-author, ZCash co-founder & scientist
- also interested in *CRowd-based INteractive Curation (CROINC)*
- early childhood development tracker: Baby.CROINC.org

Computational Integrity and Privacy (CIP)

A party executing program P on mix of public/private data ...

- may have *incentive* to *misreport* the output (*integrity problem*),

Computational Integrity and Privacy (CIP)

A party executing program P on mix of public/private data ...

- may have *incentive* to *misreport* the output (*integrity problem*),
- may wish to preserve privacy of parts of the data (*privacy problem*)

Computational Integrity and Privacy (CIP)

A party executing program P on mix of public/private data ...

- may have *incentive* to *misreport* the output (*integrity problem*),
- may wish to preserve privacy of parts of the data (*privacy problem*)
- examples: tax reporting, proof of (fractional) reserve, shielded payments, ...

Computational Integrity and Privacy (CIP)

A party executing program P on mix of public/private data . . .

- may have *incentive* to *misreport* the output (*integrity problem*),
- may wish to preserve privacy of parts of the data (*privacy problem*)
- examples: tax reporting, proof of (fractional) reserve, shielded payments, . . .
- the CIP problem: guaranteeing correct output reported, without compromising privacy

Computational Integrity and Privacy (CIP)

A party executing program P on mix of public/private data ...

- may have *incentive* to *misreport* the output (*integrity problem*),
- may wish to preserve privacy of parts of the data (*privacy problem*)
- examples: tax reporting, proof of (fractional) reserve, shielded payments, ...
- the CIP problem: guaranteeing correct output reported, without compromising privacy
- Zero knowledge proofs/arguments **[GMR88]** use randomness, interaction and cryptography to solve both CI and P in an astonishingly efficient way;

Computational Integrity and Privacy (CIP)

A party executing program P on mix of public/private data ...

- may have *incentive* to *misreport* the output (*integrity problem*),
- may wish to preserve privacy of parts of the data (*privacy problem*)
- examples: tax reporting, proof of (fractional) reserve, shielded payments, ...
- the CIP problem: guaranteeing correct output reported, without compromising privacy
- Zero knowledge proofs/arguments **[GMR88]** use randomness, interaction and cryptography to solve both CI and P in an astonishingly efficient way;
- protocols solving CIP are also known as protocols for *checking* **[BFL91]**, *certifying* **[M94]**, *delegating* **[GKR08]**, and *verifying* **[GGP10]**, computations

IP, ZK, MIP, PCP, NIZK, CS [circa 1990]

Definition (Computational Integrity (CI))

is the language of quadruples $(M, \mathcal{T}, x_{\text{in}}, x_{\text{out}})$ such that nondeterministic machine M, on input x_{in} reaches output x_{out} after $\mathcal{T}$ cycles, $\mathcal{T}$ in binary.

IP, ZK, MIP, PCP, NIZK, CS [circa 1990]

Definition (Computational Integrity (CI))

is the language of quadruples $(M, \mathcal{T}, x_{\text{in}}, x_{\text{out}})$ such that nondeterministic machine M, on input x_{in} reaches output x_{out} after $\mathcal{T}$ cycles, $\mathcal{T}$ in binary.

Lemma

CI is NEXP-complete

IP, ZK, MIP, PCP, NIZK, CS [circa 1990]

Definition (Computational Integrity (CI))

is the language of quadruples $(M, \mathcal{T}, x_{\text{in}}, x_{\text{out}})$ such that nondeterministic machine M, on input x_{in} reaches output x_{out} after $\mathcal{T}$ cycles, $\mathcal{T}$ in binary.

Lemma

CI is NEXP-complete

Definition (proof system)

An proof system S for L is a pair $\mathsf{S} = (\mathsf{V}, \mathsf{P})$ satisfying

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

IP, ZK, MIP, PCP, NIZK, CS [circa 1990]

Definition (Computational Integrity (CI))

is the language of quadruples $(M, \mathcal{T}, x_{\text{in}}, x_{\text{out}})$ such that nondeterministic machine M, on input x_{in} reaches output x_{out} after $\mathcal{T}$ cycles, $\mathcal{T}$ in binary.

Lemma

CI is NEXP-complete

Definition (argument system)

An argument system S for L is a pair $\mathsf{S} = (\mathsf{V}, \mathsf{P})$ satisfying

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

IP, ZK, MIP, PCP, NIZK, CS [circa 1990]

Definition (Computational Integrity (CI))

is the language of quadruples $(M, \mathcal{T}, x_{\text{in}}, x_{\text{out}})$ such that nondeterministic machine M, on input x_{in} reaches output x_{out} after $\mathcal{T}$ cycles, $\mathcal{T}$ in binary.

Lemma

CI is NEXP-complete

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

CI *has an argument system* $\mathsf{S} = (\mathsf{V}, \mathsf{P})$ *that is*

- **noninteractive:** *Prover sends a single message (requires setup/RO)*
- **succinct:** *Verifier run-time* $\text{poly}(n, \log \mathcal{T})$*; this bounds proof length*
- **transparent:** *Setup+verifier queries are public random coins*
- **zero knowledge:** *proof preserves privacy of nondeterministic witness*

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, ...
  - TPs have served societies for millenia
  - TPs *want* to stay such, so are not incentivized to pay for crypto CIP
  - Cryptographic CIP seems computationally costly compared to TP model (encryption suffices)

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, . . .
  - TPs have served societies for millenia
  - TPs *want* to stay such, so are not incentivized to pay for crypto CIP
  - Cryptographic CIP seems computationally costly compared to TP model (encryption suffices)
  - but considering the costs of *manual* CIP (audits, legislation, regulation), cryptographic CIP is cheap!

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, ...
- Enter Bitcoin !
  - decentralized, "In Crypto we trust"
  - huge incentive to compromise integrity (1BTC $>$ 1,000\$ (1/1/2017))
  - privacy crucial for fungibility and business-adoption

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, ...
- Enter Bitcoin !
- Zerocash, ZCash
  - Bitcoin Conference (5/2013), offered using CIP to improve Bitcoin
    - blockchain compression
    - proof of reserve ("I own 1,000 BTC")
    - improved privacy

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, ...
- Enter Bitcoin !
- Zerocash, ZCash
  - Bitcoin Conference (5/2013), offered using CIP to improve Bitcoin
    - blockchain compression
    - proof of reserve ("I own 1,000 BTC")
    - improved privacy
  - Zerocoin (5/2013): RSA-accumulator decentralized mix **[MGGR13]**

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, . . .
- Enter Bitcoin !
- Zerocash, ZCash
  - Bitcoin Conference (5/2013), offered using CIP to improve Bitcoin
    - blockchain compression
    - proof of reserve ("I own 1,000 BTC")
    - improved privacy
  - Zerocoin (5/2013): RSA-accumulator decentralized mix **[MGGR13]**
  - Zerocash **[B,Chiesa,Garman,Green,Miers,Tromer,Virza 14]**
    - first Decentralized Anonymous Payment (DAP) system
    - hides payer, payee and payment amount
    - uses KOE-based zkSNARKs **[GGPR13,BCGTV13]**

Who needs cryptographic CIP?

- Trusted parties (TP)? banks, Google, Facebook, Visa, PayPal, ...
- Enter Bitcoin !
- Zerocash, ZCash

Overview

- Computational integrity and privacy (CIP) — motivation ✓
- Importance of transparency
- A pair of new transparent CI(P) systems

CIP — desirable features

- universality — arbitrary programs (NP/NEXP complete)
- scalabilty — efficient prover running-time

CIP — desirable features

- universality — arbitrary programs (NP/NEXP complete)
- scalabilty — efficient prover running-time
  - any code you write, I can prove it was executed correctly

CIP — desirable features

- universality — arbitrary programs (NP/NEXP complete)
- scalabilty — efficient prover running-time
  - any code you write, I can prove it was executed correctly
- succinctness — short proofs, fast verification
  - compression of computation (like checking the whole blockchain), without compromising integrity and privacy
  - "heavy-weight proving" done only once, only by prover

CIP — desirable features

- universality — arbitrary programs (NP/NEXP complete)
- scalabilty — efficient prover running-time
  - any code you write, I can prove it was executed correctly
- succinctness — short proofs, fast verification
  - compression of computation (like checking the whole blockchain), without compromising integrity and privacy
  - "heavy-weight proving" done only once, only by prover
- Zerocash/ZCash uses (zkSNARKs) that achieve the above
  - based on bilinear pairings **[G10,GGP10,L12]** and Quadratic Arithmetic Programs (QAP) **[GGPR13]**
  - these zkSNARKs require *non-transparent* setup phase
  - if setup compromised, leaks a forgery-trapdoor

CIP — desirable features

- universality — arbitrary programs (NP/NEXP complete)
- scalabilty — efficient prover running-time
  - any code you write, I can prove it was executed correctly[1]
- succinctness — short proofs, fast verification
  - compression of computation (like checking the whole blockchain), without compromising integrity and privacy
  - "heavy-weight proving" done only once, only by prover
- Zerocash/ZCash uses (zkSNARKs) that achieve the above
  - based on bilinear pairings **[G10,GGP10,L12]** and Quadratic Arithmetic Programs (QAP) **[GGPR13]**
  - these zkSNARKs require *non-transparent* setup phase
  - if setup compromised, leaks a forgery-trapdoor

[1] fine print: assuming I don't know the trapdoor

CIP — desirable features

- universality — arbitrary programs (NP/NEXP complete)
- scalabilty — efficient prover running-time
  - any code you write, I can prove it was executed correctly[1]
- succinctness — short proofs, fast verification
  - compression of computation (like checking the whole blockchain), without compromising integrity and privacy
  - "heavy-weight proving" done only once, only by prover
- Zerocash/ZCash uses (zkSNARKs) that achieve the above
  - based on bilinear pairings **[G10,GGP10,L12]** and Quadratic Arithmetic Programs (QAP) **[GGPR13]**
  - these zkSNARKs require *non-transparent* setup phase
  - if setup compromised, leaks a forgery-trapdoor
- Definition: A CIP system is *transparent* if setup and all verifier queries are public random coins (Arthur-Merlin protocol)

[1] fine print: assuming I don't know the trapdoor

Transparency important for

- Ongoing public trust in integrity of the system
  - even one trapdoor leak could completely ruin integrity
  - increased value $\Rightarrow$ increased incentive to attack/corrupt
  - what if powerful entity/agency asks for the trapdoor?

Transparency important for

- Ongoing public trust in integrity of the system
  - even one trapdoor leak could completely ruin integrity
  - increased value $\Rightarrow$ increased incentive to attack/corrupt
  - what if powerful entity/agency asks for the trapdoor?
- Collaborative creation of CIP software
  - crypto-currencies use decentralized code development
  - who generates keys for a non-transparent CI?
  - with code proliferation, should you trust the setup?

Transparency important for

- Ongoing public trust in integrity of the system
  - even one trapdoor leak could completely ruin integrity
  - increased value $\Rightarrow$ increased incentive to attack/corrupt
  - what if powerful entity/agency asks for the trapdoor?
- Collaborative creation of CIP software
  - crypto-currencies use decentralized code development
  - who generates keys for a non-transparent CI?
  - with code proliferation, should you trust the setup?
- Transparent auditing of central private registries
  - registries maintained by governments have huge impact on citizen rights and liabilities
  - many registries contain private data, so privacy prevents public auditing
  - cryptographic CIP can enhance trust in registry management
  - public trust demands transparent CIP

Overview

- Computational integrity and privacy (CIP) — motivation ✓
- Importance of transparency ✓
- A pair of new transparent CI(P) systems

A pair of novel transparent CIP

1. SCI (Scalable Comptutional Integrity)
   - Joint work with Iddo Ben-Tov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer and Madars Virza
   - To appear in Eurocrypt 2017
   - universal, succinct, scalable, transparent, post-quantum secure
2. SCIP (Scalable Computational Integrity and Privacy)
   - Joint work with Iddo Ben-Tov, Yinon Horesh and Michael Riabzev
   - work in progress
   - ZK, universal, succinct, scalable, transparent, post-quantum secure

A pair of novel transparent CIP

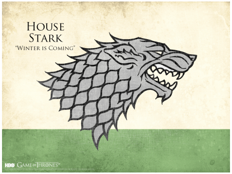
1. SCI (Scalable Comptutional Integrity)
   - Joint work with Iddo Ben-Tov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer and Madars Virza
   - To appear in Eurocrypt 2017
   - universal, succinct, scalable, transparent, post-quantum secure
2. SCIP (Scalable Computational Integrity and Privacy)
   - Joint work with Iddo Ben-Tov, Yinon Horesh and Michael Riabzev
   - work in progress
   - ZK, universal, succinct, scalable, transparent, post-quantum secure

Given popularity of SNARKs . . .

A pair of novel transparent CIP

1. SCI (Scalable Comptutional Integrity)
   - Joint work with Iddo Ben-Tov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer and Madars Virza
   - To appear in Eurocrypt 2017
   - universal, succinct, scalable, transparent, post-quantum secure
2. STARK (Succinct Transparent ARgument of Knowledge)
   - Joint work with Iddo Ben-Tov, Yinon Horesh and Michael Riabzev
   - work in progress
   - ZK, universal, succinct, scalable, transparent, post-quantum secure

Given popularity of SNARKs . . .



Comparison with prior CIP implementations

- Linear PCP (LPCP) [IKO07]
  - Use additively homomorphic encryption to (i) hide queries + (ii) eliminate need for low-degree testing
  - Implementations: pepper, ginger, ... [SBW11,SVP+12,SMBW12]

Comparison with prior CIP implementations

- Linear PCP (LPCP) [IKO07]
- MPC-in-head (MPCh) [IKOS07]
  - Prover commits to MPC transcript, then opens one party's view
  - Implementation: ZKBoo [GMO16]

Comparison with prior CIP implementations

- Linear PCP (LPCP) **[IKO07]**
- MPC-in-head (MPCh) **[IKOS07]**
- Proofs for muggles (IP) **[GKR08]**
  - Scaling-down of IP=PSPACE to case of poly-bounded prover, works for uniform log-space PTIME
  - Implementations: **[CTY11,CMT12,T13]**, allspice **[VSBW13]**

Comparison with prior CIP implementations

- Linear PCP (LPCP) **[IKO07]**
- MPC-in-head (MPCh) **[IKOS07]**
- Proofs for muggles (IP) **[GKR08]**
- Pairing-based/Knowledge of Exponent (KOE) **[G10,GGP10,L12,GGPR13]**
  - succinct proofs (<300 bytes) after setup, which is non-Arthur-Merlin
  - Implementations: Pinocchio **[PGHR13]**, SNARKs for C **[BCGTV13]**, Zaatar **[SBVBPW13]**, Buffet **[WSHRBW15]**

Comparison with prior CIP implementations

- Linear PCP (LPCP) **[IKO07]**
- MPC-in-head (MPCh) **[IKOS07]**
- Proofs for muggles (IP) **[GKR08]**
- Pairing-based/Knowledge of Exponent (KOE) **[G10,GGP10,L12,GGPR13]**
- Discrete-logarithm problem (DLP) based **[G11,S11]**
  - succinct proofs, public (Arthur-Merlin) setup, verification-time$> \mathcal{T}$
  - Implementation: **[BCCGP16]**

Comparison with prior CIP implementations

- Linear PCP (LPCP) [IKO07]
- MPC-in-head (MPCh) [IKOS07]
- Proofs for muggles (IP) [GKR08]
- Pairing-based/Knowledge of Exponent (KOE) [G10,GGP10,L12,GGPR13]
- Discrete-logarithm problem (DLP) based [G11,S11]
- Incrementally verifiable computation (IVC) [V08,BCCT13]
  - Prover runs verifier on each prior "chunk" of computation
  - Implementation: [BCTV14] (KOE based)

Comparison of implemented CIP

- **UN universal:** works for any language in NP
- **SC scalable:** prover runtime quasilinaer in $\mathcal{T}$
- **NI noninteractive:** after setup/common reference string
- **SU succinct:** verifier time poly(log $\mathcal{T}$, $|x|$)
- **TR transparent:** setup+queries are merely public random coins
- **ZK:** zero knowledge
- **PQ:** post-quantum resistant

	UN	SC	NI	SU	TR	ZK	PQ
LPCP **[IKO07]**	+	-	-	±	-	+	-
MPCh **[IKOS07]**	+	-	+	-	+	+	+
IP **[GKR08]**	-	-	-	+	+	-	+
KOE **[GGPR13]**	+	+	+	±	-	+	-
DLP **[BCCGP16]**	+	+	+	-	+	+	-
IVC **[V08]**	+	+	+	+	-	+	-

Comparison of implemented CIP

- **UN universal:** works for any language in NP
- **SC scalable:** prover runtime quasilinaer in $\mathcal{T}$
- **NI noninteractive:** after setup/common reference string
- **SU succinct:** verifier time poly(log $\mathcal{T}$, $|x|$)
- **TR transparent:** setup+queries are merely public random coins
- **ZK:** zero knowledge
- **PQ:** post-quantum resistant

	UN	SC	NI	SU	TR	ZK	PQ
LPCP **[IKO07]**	+	-	-	±	-	+	-
MPCh **[IKOS07]**	+	-	+	-	+	+	+
IP **[GKR08]**	-	-	-	+	+	-	+
KOE **[GGPR13]**	+	+	+	±	-	+	-
DLP **[BCCGP16]**	+	+	+	-	+	+	-
IVC **[V08]**	+	+	+	+	-	+	-
SCI **[BBC+17]**	+	+	+	+	+	-	+

Comparison of implemented CIP

- **UN universal:** works for any language in NP
- **SC scalable:** prover runtime quasilinaer in $\mathcal{T}$
- **NI noninteractive:** after setup/common reference string
- **SU succinct:** verifier time poly(log $\mathcal{T}$, $|x|$)
- **TR transparent:** setup+queries are merely public random coins
- **ZK:** zero knowledge
- **PQ:** post-quantum resistant

	UN	SC	NI	SU	TR	ZK	PQ
LPCP **[IKO07]**	+	-	-	±	-	+	-
MPCh **[IKOS07]**	+	-	+	-	+	+	+
IP **[GKR08]**	-	-	-	+	+	-	+
KOE **[GGPR13]**	+	+	+	±	-	+	-
DLP **[BCCGP16]**	+	+	+	-	+	+	-
IVC **[V08]**	+	+	+	+	-	+	-
SCI **[BBC+17]**	+	+	+	+	+	-	+
STARK **[BBHT17]**	+	+	+	+	+	+	+

SCI vs. other CIP implementations

Table: Execution of same TinyRAM program for 2^{16} cycles; 80-bit security level; machine w/ 32 AMD Opteron cores, clock rate 3.2 GHz, 512 GB RAM.

		KOE	IVC	DLP	SCI	STARK
Ver. setup	**time**	$\sim$ 28 min	$\sim$ 10 sec	$\sim$ 0.7 sec	<0.01 sec	<0.01 sec
	key len	$\sim$ 18.9 GB	43 MB	154 MB	16 bytes	16 bytes
Prov	**time**	$\sim$ 18 min	4.2 days	$\sim$ 8 min	$\sim$ 41 min	6.7 min
	memory	$\sim$ 216 GB	2.9 GB	$\sim$ 1 TB	$\sim$ 135 GB	$\sim$ 131 GB
Ver. dec.	**time**	$<$ 10 ms	$\sim$ 25 ms	$\sim$ 1.7 min	$\sim$ 0.5 sec	$\sim$ 0.1 sec
	comm comp	230 bytes	374 bytes	8.8KB	$\sim$ 42.5 MB	1.8 MB
V. total	**time**	$\sim$ 28 min	$\sim$ 10 sec	1.7 min	$\sim$ 0.5 sec	$\sim$ 0.1 sec
	comm comp	$\sim$ 18.9 GB	43 MB	$\sim$ 154 MB	$\sim$ 42.5 MB	$\sim$ 1.8 MB

Overview

- Computational integrity and privacy (CIP) — motivation ✓
- Importance of transparency ✓
- A pair of new transparent CI(P) systems
  - SCI performance [BBCGGHPRSTV16]
  - STARK performance [BBHT17]

SCI — Succinct Computational Integrity

- First assembly-code-to-PCP* reduction, including RS-proximity testing and PCPP composition

SCI — Succinct Computational Integrity

- First assembly-code-to-PCP* reduction, including RS-proximity testing and PCPP composition
  (* Use IOPP instead of PCPP to save space, similar "code complexity"/security/soundness)

SCI — Succinct Computational Integrity

- First assembly-code-to-PCP* reduction, including RS-proximity testing and PCPP composition
  (* Use IOPP instead of PCPP to save space, similar "code complexity"/security/soundness)
- Joint work with Iddo Ben-Tov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer and Madars Virza [Eurocrypt17]

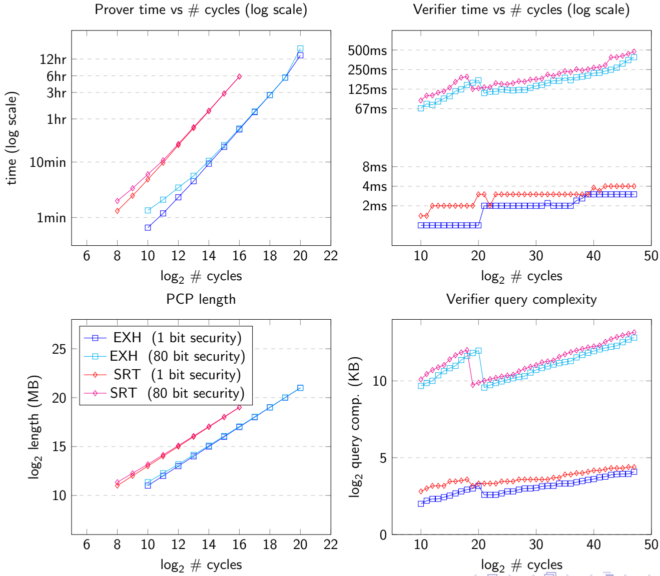
SCI — Succinct Computational Integrity

- First assembly-code-to-PCP* reduction, including RS-proximity testing and PCPP composition
  (* Use IOPP instead of PCPP to save space, similar "code complexity"/security/soundness)
- Joint work with Iddo Ben-Tov, Alessandro Chiesa, Ariel Gabizon, Daniel Genkin, Matan Hamilis, Evgenya Pergament, Michael Riabzev, Mark Silberstein, Eran Tromer and Madars Virza **[Eurocrypt17]**
- Large scale effort
  - Started summer 2010
  - More than $1M$ euro over first 6 years (thanks to European Research Council!!)
  - Mostly for programmers: Ohad Barta, Lior Greenblatt, Shaul Kfir, Gil Timnat, Arnon Yogev

SCI executed programs

- CI statement: "no subset of input array A sums to target t"
- Two different programs
  1. EXH — exhaustive search
     - running time $\mathcal{T} \sim 2^{|A|}$
     - memory $O(1)$
  2. SRT — sorted search
     - Sort each half of A increasingly, then "merge"
     - running time $\mathcal{T} \sim 2^{|A|/2}$
     - random access memory consumption $2^{|A|/2}$

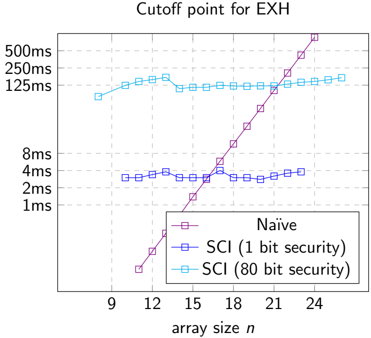
SCI numbers

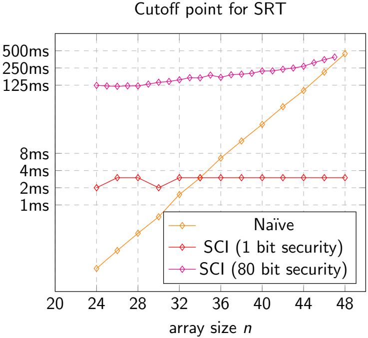


Prover time vs # cycles (log scale)
time (log scale)
12hr
6hr
3hr
1hr
10min
1min
6
8
10
12
14
16
18
20
22
log2 # cycles
Verifier time vs # cycles (log scale)
500ms
250ms
125ms
67ms
8ms
4ms
2ms
10
20
30
40
50
log2 # cycles
PCP length
log2 length (MB)
25
20
15
10
EXH (1 bit security)
EXH (80 bit security)
SRT (1 bit security)
SRT (80 bit security)
log2 # cycles
Verifier query complexity
log2 query comp. (KB)
10
5
0
log2 # cycles

SCI break-even point [SVPBBW12,SMBW12]

- Def: minimal n_0 for which naïve re-execution $>$ SCI-verification.
- For EXH at 80-bit security $n_{\text{EXH}} = 22$
- For SRT at 80-bit security $n_{\text{SRT}} = 48$
- $n_{\text{SRT}} > n_{\text{EXH}}$ because SRT is quadratically faster





STARK [BBHT17]

- first implementation of a succinct, universal, transparent, scalable zk proof system;

STARK [BBHT17]

- first implementation of a succinct, universal, transparent, scalable zk proof system;
- also first implemented post-quantum secure succinct zk proof system

STARK [BBHT17]

- first implementation of a succinct, universal, transparent, scalable zk proof system;
- also first implemented post-quantum secure succinct zk proof system
- uses recent scalable "duplex IOP" BCGV16, along with
  1. new "biased RS-IOPP" protocol

STARK [BBHT17]

- first implementation of a succinct, universal, transparent, scalable zk proof system;
- also first implemented post-quantum secure succinct zk proof system
- uses recent scalable "duplex IOP" BCGV16, along with
    1. new "biased RS-IOPP" protocol
        - ★ better concrete soundness and security than prior state of the art
        - ★ shorter proofs due to better "packaging" of proof parts into a Merkle tree
        - ★ more efficient to compute (single FFT followed by fully parallelizable local computations)

STARK [BBHT17]

- first implementation of a succinct, universal, transparent, scalable zk proof system;
- also first implemented post-quantum secure succinct zk proof system
- uses recent scalable "duplex IOP" BCGV16, along with
  1. new "biased RS-IOPP" protocol
     - better concrete soundness and security than prior state of the art
     - shorter proofs due to better "packaging" of proof parts into a Merkle tree
     - more efficient to compute (single FFT followed by fully parallelizable local computations)
  2. new IOP reduction from Algebraic constraint satisfaction to Reed-Solomon proximity testing
     - higher soundness retention
     - simpler proofs, of lower-degree

STARK [BBHT17]

- first implementation of a succinct, universal, transparent, scalable zk proof system;
- also first implemented post-quantum secure succinct zk proof system
- uses recent scalable "duplex IOP" BCGV16, along with
  1. new "biased RS-IOPP" protocol
     - better concrete soundness and security than prior state of the art
     - shorter proofs due to better "packaging" of proof parts into a Merkle tree
     - more efficient to compute (single FFT followed by fully parallelizable local computations)
  2. new IOP reduction from Algebraic constraint satisfaction to Reed-Solomon proximity testing
     - higher soundness retention
     - simpler proofs, of lower-degree
  3. improved concrete arithmetization of cryptographic primitives (AES)

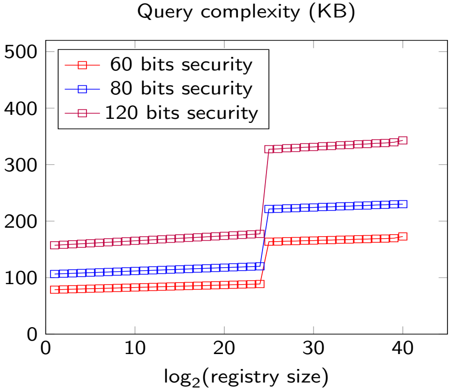
STARK black-list non-membership [BBHT17]

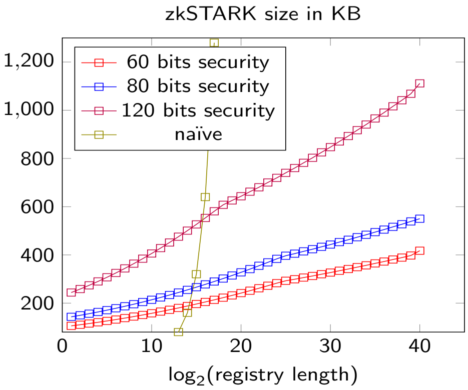
- CIP statement: y does not appear in private black-list, with public hash commitment r
- Useful for banks (FACTA compliance), airlines (anti-terrorism compliance), etc.

STARK black-list non-membership [BBHT17]

- CIP statement: y does not appear in private black-list, with public hash commitment r
- Useful for banks (FACTA compliance), airlines (anti-terrorism compliance), etc.
- Formally
  - Inputs: element y and public commitment r (hash of black-list)
  - Statement: $\exists D \ \mathrm{comm}(D) = r$ and $y \notin D$
  - pseudo-code: If $y \in D$ or $\mathrm{comm}(D) \neq r$ reject, else accept
  - D is private (nondeterministic witness), $|D| = 2^h$
  - comm is either Merkle tree or hash chain
  - Hash function is Davies-Meyer hash + AES160

STARK estimated proof length [BBHT17]





Disclaimer: work in progress, hence numbers may change

SCI vs. other CIP implementations

Table: Execution of same TinyRAM program for 2^{16} cycles; 80-bit security level; machine w/ 32 AMD Opteron cores, clock rate 3.2 GHz, 512 GB RAM.

		KOE	IVC	DLP	SCI	STARK
Ver. setup	**time**	$\sim$ 28 min	$\sim$ 10 sec	$\sim$ 0.7 sec	$<$0.01 sec	$<$0.01 sec
	key len	$\sim$ 18.9 GB	43 MB	154 MB	16 bytes	16 bytes
Prov	**time**	$\sim$ 18 min	4.2 days	$\sim$ 8 min	$\sim$ 41 min	6.7 min
	memory	$\sim$ 216 GB	2.9 GB	$\sim$ 1 TB	$\sim$ 135 GB	$\sim$ 131 GB
Ver. dec.	**time**	$<$ 10 ms	$\sim$ 25 ms	$\sim$ 1.7 min	$\sim$ 0.5 sec	$\sim$ 0.1 sec
	comm comp	230 bytes	374 bytes	8.8KB	$\sim$ 42.5 MB	1.8 MB
V. total	**time**	$\sim$ 28 min	$\sim$ 10 sec	1.7 min	$\sim$ 0.5 sec	$\sim$ 0.1 sec
	comm comp	$\sim$ 18.9 GB	43 MB	$\sim$ 154 MB	$\sim$ 42.5 MB	$\sim$ 1.8 MB

STARK vs. SNARK

- Main advantages of STARK over SNARK are transparency and scalability
  - both due to reliance on proven mathematics (PCPs) which lead to "lighter" crypto assumptions (hash+Fiat Shamir)
- Main advantage of SNARK of STARK is shorter proofs

STARK vs. SNARK

- Main advantages of STARK over SNARK are transparency and scalability
  - both due to reliance on proven mathematics (PCPs) which lead to "lighter" crypto assumptions (hash+Fiat Shamir)
- Main advantage of SNARK of STARK is shorter proofs
- Assuming STARK proofs don't get shorter, to use in a crypto-currency:
  - users send tx to a "tx-aggregator"
  - tx-aggregator checks and aggregates many transactions
  - generates single STARK for all of them (say, 2^{20})
  - broadcasts UTXO diff file + STARK
  - this improves the crypto-currency scalability
  - transparency implies: don't trust aggregator, trust the proof.

Concluding remarks

- Computational integrity+privacy (CIP)
  - crucial for long-term viability of decentralized blockchains
  - potentially useful even for trusted parties (Government, Banks, etc.)
- CIP systems for blockchains require universality, transparency, succinctness, scalability, and privacy (post-quantum security also helpful)
- STARK delivers all; SCI delivers all but privacy

Concluding remarks

- Computational integrity+privacy (CIP)
  - crucial for long-term viability of decentralized blockchains
  - potentially useful even for trusted parties (Government, Banks, etc.)
- CIP systems for blockchains require universality, transparency, succinctness, scalability, and privacy (post-quantum security also helpful)
- STARK delivers all; SCI delivers all but privacy
- "theory-to-practice" research, like this, leads to new models, new questions, and new applications

Concluding remarks

- Computational integrity+privacy (CIP)
  - crucial for long-term viability of decentralized blockchains
  - potentially useful even for trusted parties (Government, Banks, etc.)
- CIP systems for blockchains require universality, transparency, succinctness, scalability, and privacy (post-quantum security also helpful)
- STARK delivers all; SCI delivers all but privacy
- "theory-to-practice" research, like this, leads to new models, new questions, and new applications
- want to hear more?
  - Ethereum meetup this Sunday Jan 29, 6pm, Institute for the Future: *more details*
  - Berkeley CS Theory Seminar, Monday Feb 6, 4pm, Wozniak Lounge: *moon math*
  - Stanford Security Seminar, Tuesday Feb 7, 4:15pm, Gates 463: *moon math+engineering*