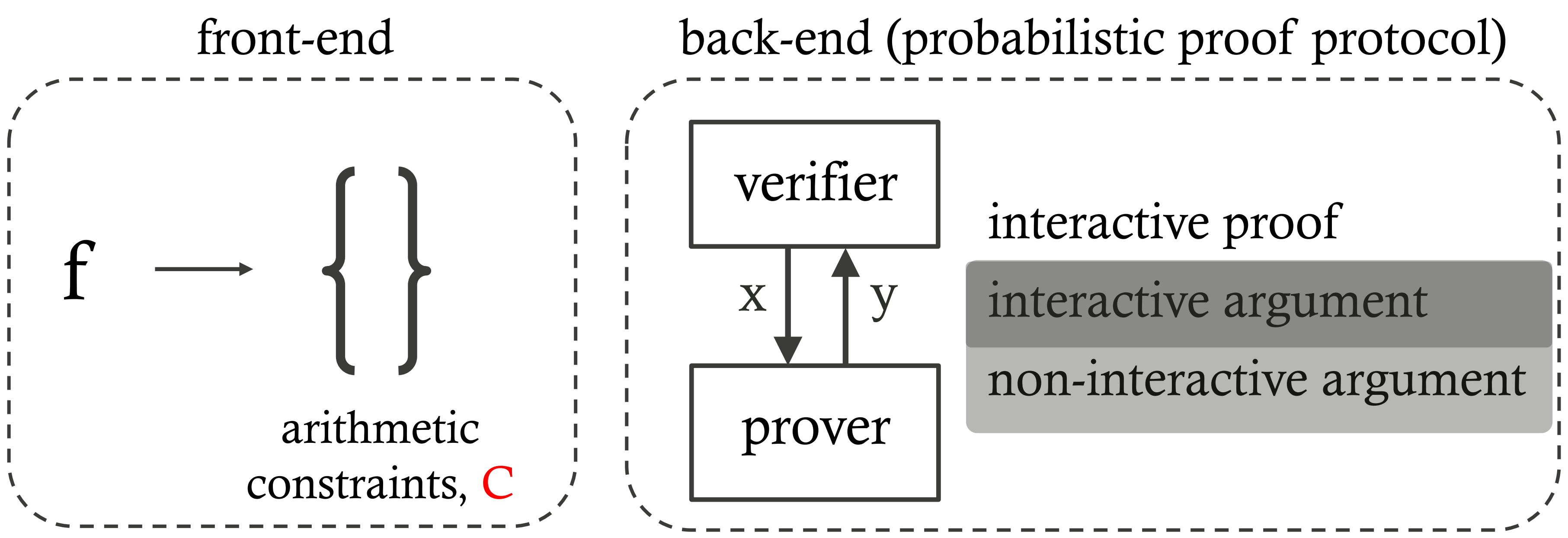
Interactive Arguments with Preprocessing

Bar-Ilan Winter School on Verifiable Computation
Class 5
January 4, 2016

Michael Walfish
Dept. of Computer Science, Courant Institute, NYU

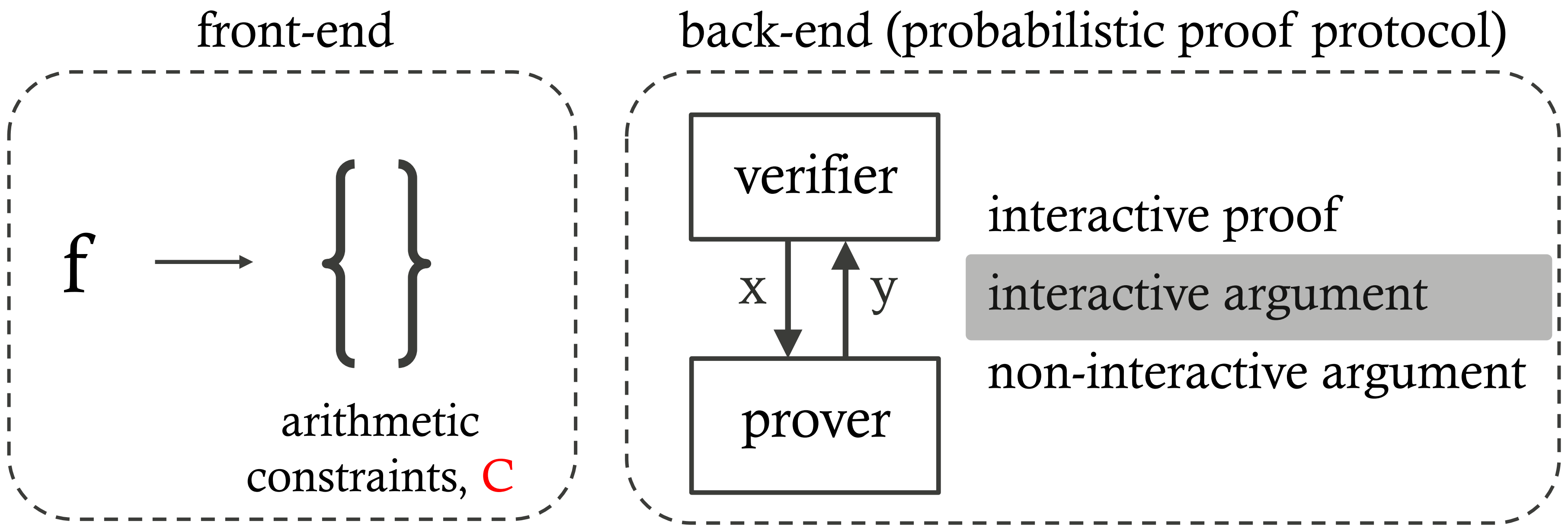


Next two sessions: foundations for (implemented) arguments

- Key ideas: linear PCPs, QAPs

This session: focus on interactive arguments

- Conceptually illuminating
- Based on standard assumptions
- Plan: linear PCPs, interactivity, the role of QAPs



Recall: by construction, C is satisfiable iff f executes correctly

More formally: C is degree-2 constraints over $\mathbb{F}$ and variables (X, Y, Z) s.t.

$\forall$ x,y: $\exists$ w s.t. $y=f(x,w) \Leftrightarrow C(X=x, Y=y)$ is satisfiable

Thus, as usual, P must convince V that constraints C’ are satisfiable.
(C’ is notation for C(X=x,Y=y).)

Goal: P convinces V that a set of constraints is satisfiable while:

- V's running time (possibly amortized) is less than executing f
- P's running time is quasilinear in the time to execute f
- mechanics are simple, concrete costs aren't crazy

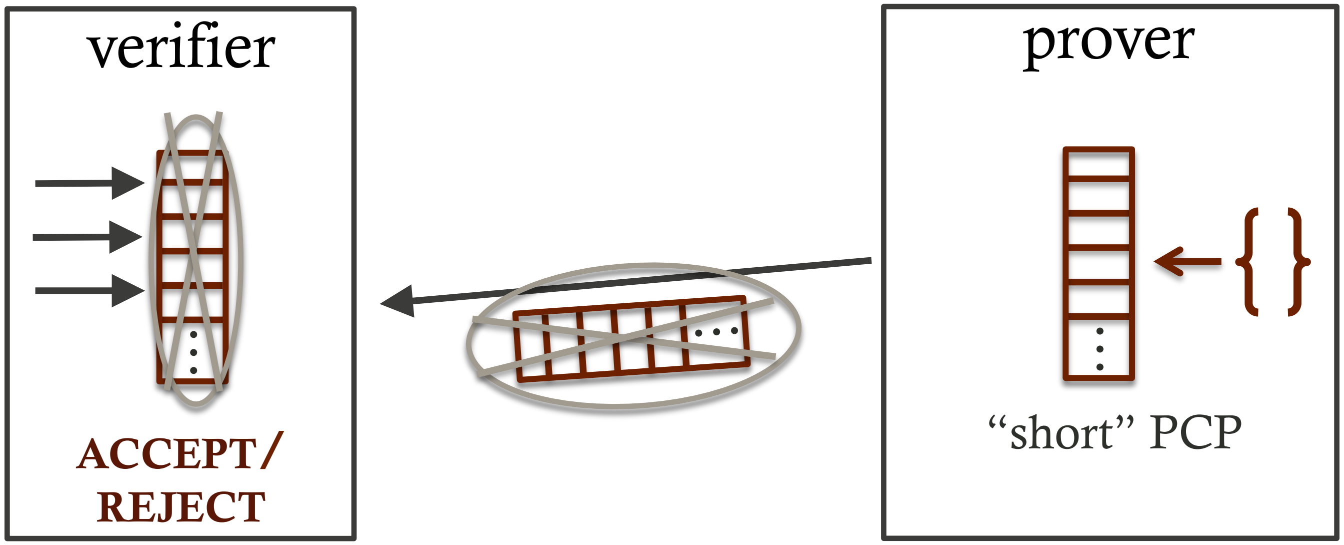
(Today, assume that P has an assignment, z. Tomorrow, we will describe how P derives z.)

Attempt 0: P sends z to V; V checks that z satisfies all constraints

- Doesn't meet the goal; z is same "size" as running time of f

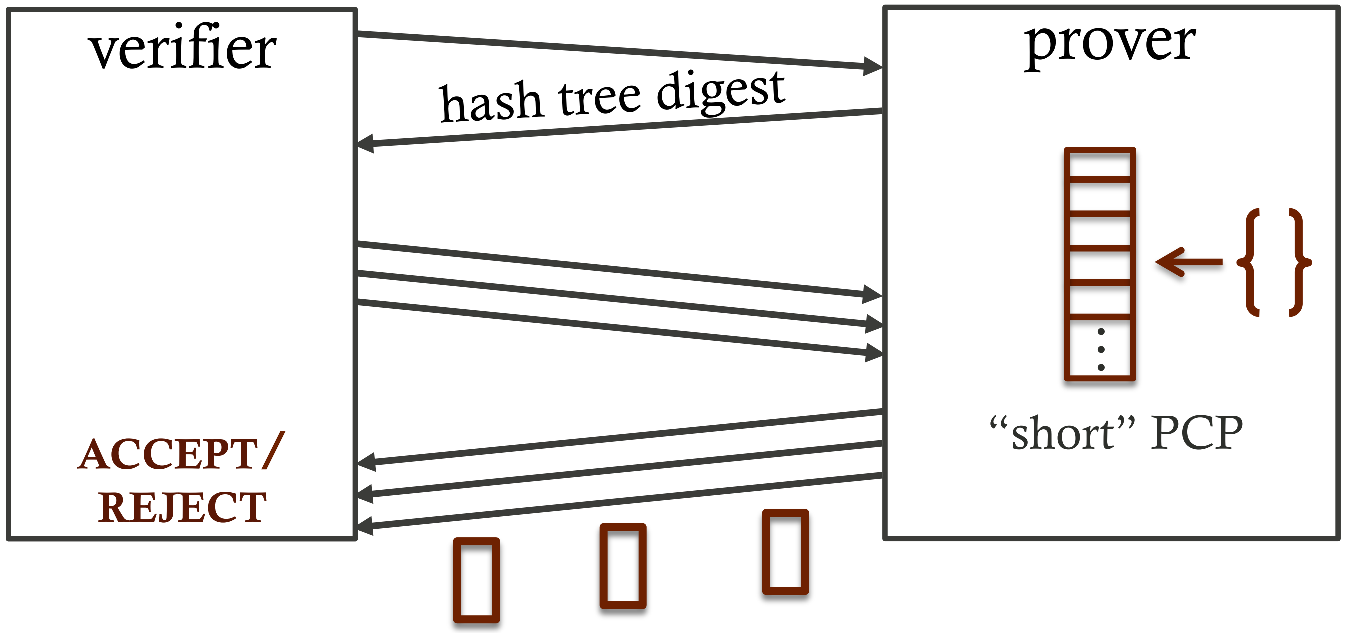
Attempt 1: Use PCPs that are asymptotically short

[ALMSS92, AS92] [BGHSV05, BGHSV06, Dinur07, BS08, Meir12, BCGT13]



This also doesn't meet the goal
(because |PCP| > running time of f).

Attempt 2: Use arguments or CS proofs
[Kilian92, Micali94, BG02]



But the constants seem too high …

Attempt 3: Use long PCPs interactively

Claim (informal): V can check the satisfiability of constraints C' by inspecting a long proof in a constant number of entries.

A more formal version of this claim is a "lite" PCP theorem: $\text{QuadConstraint}_{\mathbb{F}} \subset \text{PCP}(\text{poly}(n), \log|\mathbb{F}|)$, with soundness error 7/9.

Claim (informal): P can play the role of the long proof without materializing the proof explicitly.

More formally: Using additively homomorphic encryption, the PCP above implies an argument (with similar soundess) in which the V→P communication complexity is the length of the PCP queries, and the P→V communication is a constant number of field elements.

This requires preprocessing, or setup work, for V.

Claim (soft): The approach is simple, with good constants.

Claim: $\mathcal{V}$ can check the satisfiability of degree-2 constraints $\mathcal{C}'$ (over a field $\mathbb{F}$) by inspecting a long proof in a constant number of entries.

Consider a polynomial $Q(Z) \triangleq \sum_{j=1}^{m} \gamma_j \cdot Q_j(Z)$:

$Q(Z)$ is a “bellwether”:

- If $Z = z$ satisfies $\mathcal{C}'$, then …
- If $Z = z$ does not satisfy $\mathcal{C}'$, then …

$Q(z)$ can be evaluated with two queries to a correct proof oracle, π

$$Q(z) =$$

$$\pi_1 =$$

$$\pi_2 =$$

But what if π is adversarially constructed?

- π_1, π_2 might not be Hadamard codewords

Hadamard codeword $\longleftrightarrow$ linear function. So test for linearity [BLR90].

Defn. of δ-close, δ-far: functions g, h with the same domain are δ-close (δ-far) if they disagree on less (more) then a fraction δ of their domain.

But what if π is adversarially constructed?

- π_1, π_2 might not be (close to) Hadamard code words
- π_1, π_2 might be close to Hadamard codewords but inconsistent

Quad correction test:

But what if π is adversarially constructed?

- π_1, π_2 might not be (close to) Hadamard code words
- π_1, π_2 might be close to Hadamard codewords but inconsistent
- π_1, π_2 might be close to Hadamard codewords and consistent but encode a non-satisfying assignment.

Circuit test:

Putting together the guarantees of the tests:

- If $\mathcal{C}'$ is satisfiable, there are π_1, π_2 s.t. $\Pr\{\text{all tests pass}\} = 1$.
- If $\mathcal{C}'$ is not satisfiable, then for all π_1, π_2, $\Pr\{\text{all tests pass}\} \leq 7/9$.

There were 14 queries; each of length n or n^2 field elements. The mechanics were simple.

So our desired claim is established:

> *$\mathcal{V}$ can check the satisfiability of degree-2 constraints $\mathcal{C}'$ (over $\mathbb{F}$) by inspecting a long proof in a constant number of entries.*
>
> *More formally, $QuadConstraint_{\mathbb{F}} \subset \mathrm{PCP}(\mathrm{poly}(n), \log|\mathbb{F}|)$.*

This kind of PCP is known as a Hadamard PCP, or linear PCP [ALMSS92, IKO07].

Attempt 3: Use long PCPs interactively

Claim (informal): V can check the satisfiability of constraints C' by inspecting a long proof in a constant number of entries.

A more formal version of this claim is a "lite" PCP theorem: $\text{QuadConstraint}_{\mathbb{F}} \subset \text{PCP}(\text{poly}(n), \log|\mathbb{F}|)$, with soundness error 7/9.

Claim (informal): P can play the role of the long proof without materializing the proof explicitly.

More formally: Using additively homomorphic encryption, the PCP above implies an argument (with similar soundness) in which the V→P communication complexity is the length of the PCP queries, and the P→V communication is a constant number of field elements.

This requires preprocessing, or setup work, for V.

Claim (soft): The approach is simple, with good constants.

How can $\mathcal{P}$ play the role of an exponentially-sized linear PCP?

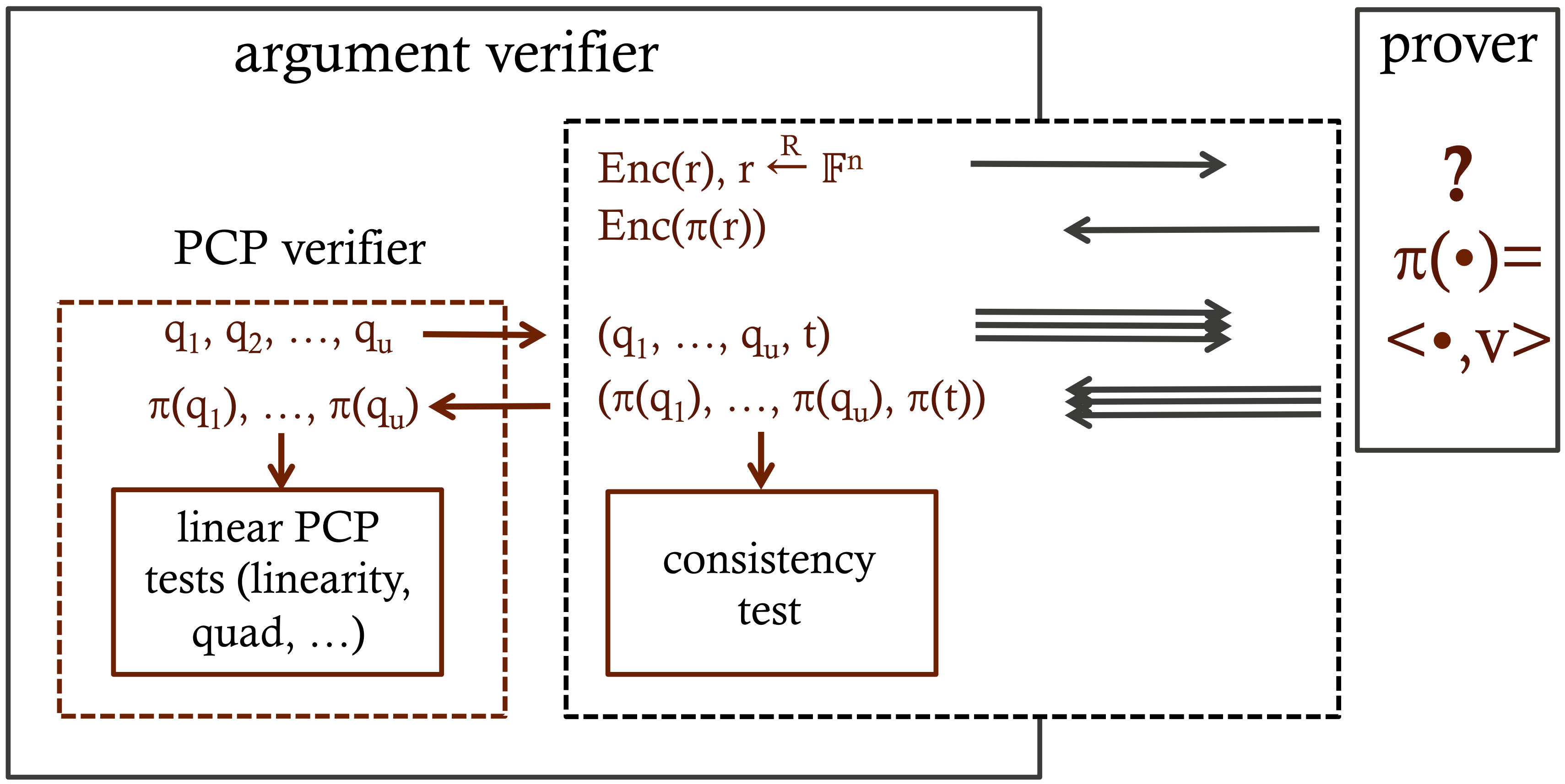
- Idea: represent linear PCP implicitly, as a linear function
- Idea: $\mathcal{V}$ uses additively homomorphic encryption to make $\mathcal{P}$ commit to a function of this form.

Informal guarantee of linear commitment primitive: $\mathcal{P}$ is bound to a fixed function.

This leads to our desired claim:

> *$\mathcal{P}$ can play the role of the linear PCP without materializing that PCP explicitly. More formally:*
>
> *Using additively homomorphic encryption, a linear PCP can be transformed into an argument (with negligibly more soundness error) in which the $\mathcal{V}$-to-$\mathcal{P}$ communication complexity equals the PCP query length and the $\mathcal{P}$-to-$\mathcal{V}$ communication is a constant number of field elements.*

[IKO07, SMBW12]

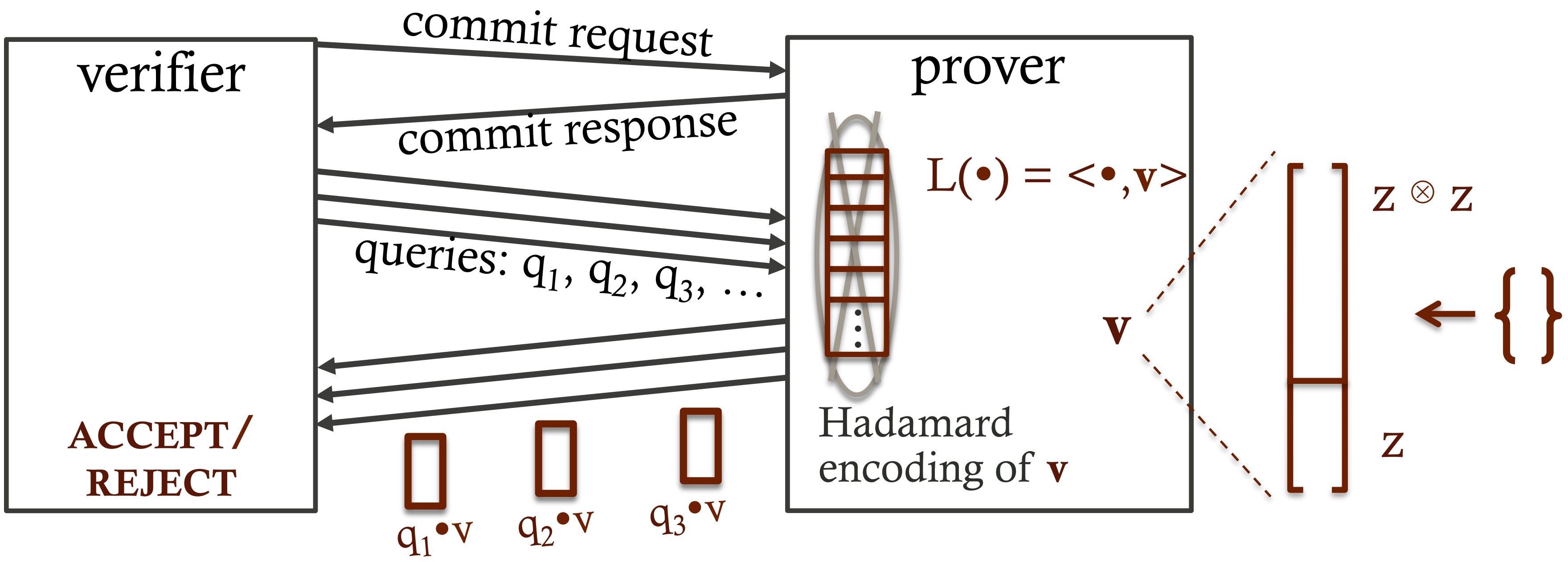

argument verifier
PCP verifier
$q_1, q_2, \ldots, q_u$
$\pi(q_1), \ldots, \pi(q_u)$
linear PCP tests (linearity, quad, …)
$\mathrm{Enc}(r),\ r \xleftarrow{R} \mathbb{F}^n$
$\mathrm{Enc}(\pi(r))$
$(q_1, \ldots, q_u, t)$
$(\pi(q_1), \ldots, \pi(q_u), \pi(t))$
consistency test
prover
?
$\pi(\bullet) = \langle \bullet, v \rangle$

consistency query: $t = r + \alpha_1 \bullet q_1 + \ldots + \alpha_u \bullet q_u\,,\quad \alpha_i \xleftarrow{R} \mathbb{F}$

consistency test: $\pi(t) \overset{?}{=} \pi(r) + \alpha_1 \bullet \pi\,(q_1) + \ldots + \alpha_u \bullet \pi\,(q_u)$

Attempt 3: Use long PCPs interactively (summary)

[IKO07, SMBW12]



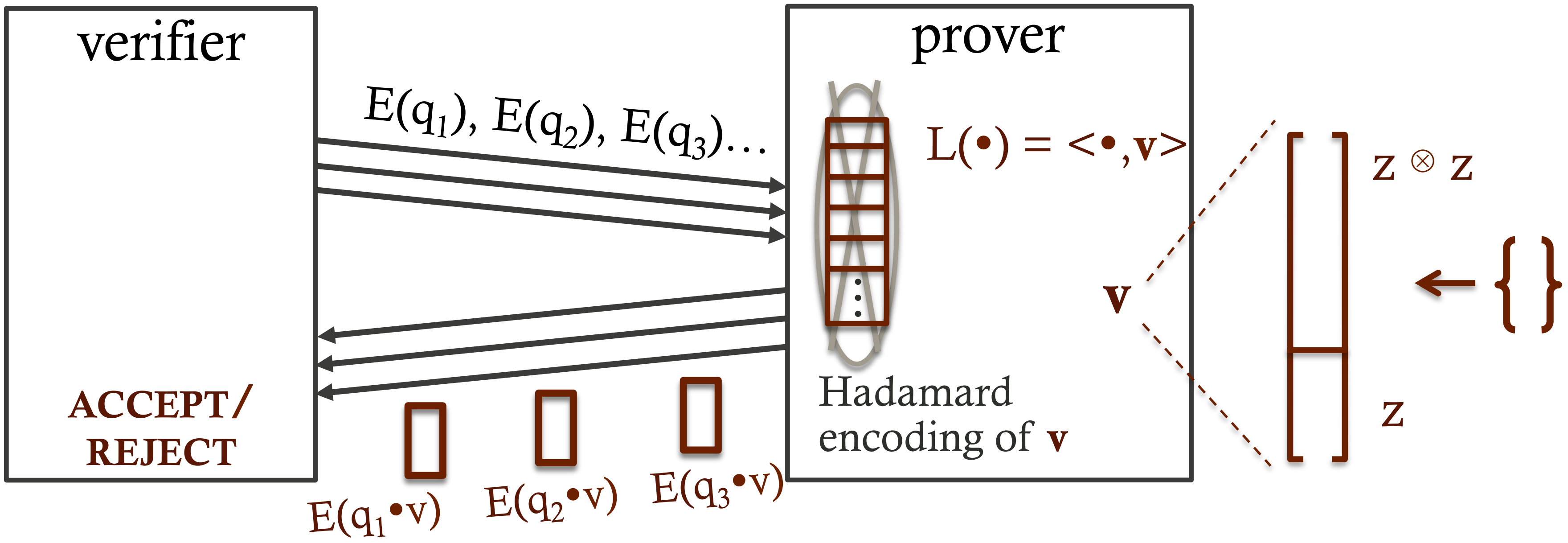
Achieves simplicity, with good constants …

… but pre-processing is required (because $|q_i| = |v|$)

… and prover’s work is quadratic; address that shortly

Attempt 4: Use long PCPs non-interactively [BCIOP13]

preview



Query process now happens "in the exponent"

… pre-processing still required (again because $|q_i|=|v|$)

… prover's work still quadratic; addressing that soon

Recap

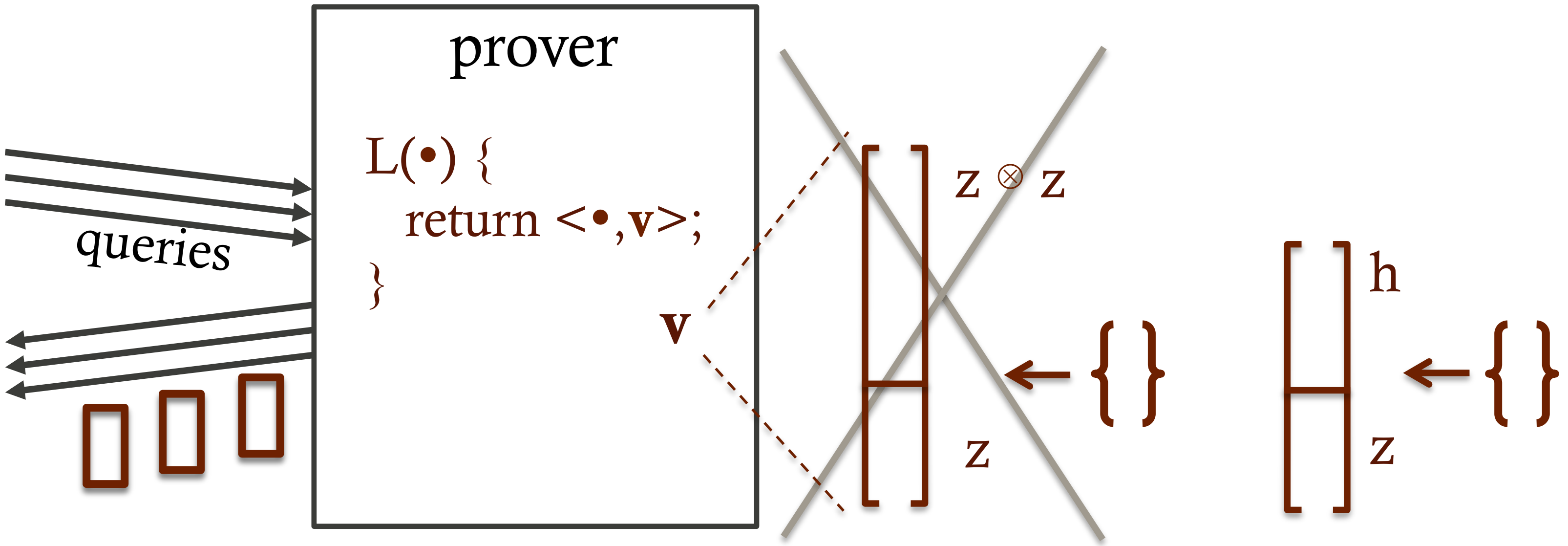
	efficient (short) PCPs	arguments, CS proofs	arguments w/ preprocessing	preview SNARGs w/ preprocessing
who	ALMSS92, AS92, BGSHV, Dinur, …	Kilian92, Micali94	IKO07, SMBW12, SVPBBW12	Groth10, GGPR12, BCIOP13, …
what	classical PCP	commit to PCP by hashing	commit to long PCP using linearity	encrypt queries to a long PCP
security	unconditional	CRHFs	linearly HE	knowledge-of-exponent
why/why not	not efficient for V	constants are unfavorable	simple	simple, non-interactive

(Thanks to Rafael Pass.)

Final attempt: apply linear query structure to GGPR's QAPs

[Groth10, Lipmaa12, GGPR12]

preview



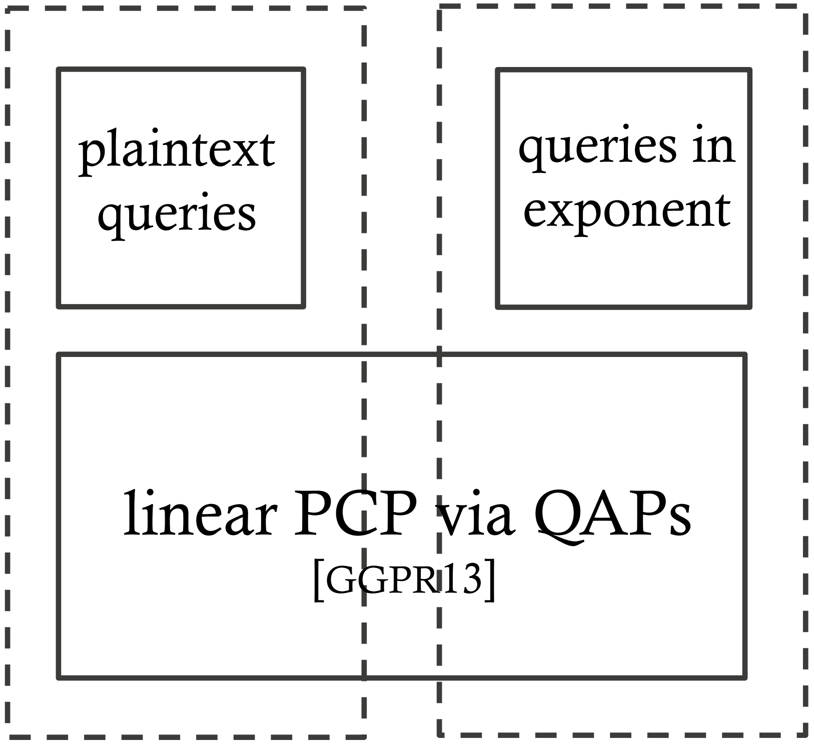
Addresses the issue of quadratic costs.

PCP structure implicit in GGPR. Made explicit in [BCIOP13, SBVBBW13].

Summary of published argument implementations

"Zaatar"
[SBVBBW13]

interactive argument
[IKO07]



"Pinocchio," "libsnark"
[PGHR13, BCTV14b]

SNARG, zk-SNARK with preprocessing
[Groth10, BCCT12, GGPR12]

preprocessing lowered to (high) constant
[BCCT13, BCTV14a, CTV15]

- standard assumptions
- amortize over batch
- interactive

- non-falsifiable assumptions
- amortize indefinitely
- non-interactive, ZK, …

All recent implementations are based on GGPR
SBVBPW13, PGHR13, BFRSBW13, BCGTV13, BCGGMTV14, BCTV14a, BCTV14b, FL14, KPPSST14, WSRBW15, CFHKKNPZ15, BBFR15, CTV15

Summary of key concepts

1. Linear (Hadamard) PCPs, to prove satisfiability

- Exponentially long but mechanically simple, with good constants

2. Linear PCPs can be transformed into argument protocols with preprocessing

- Interactive version: only standard assumptions
- Non-interactive version: better amortization and properties

3. QAPs lower quadratic costs to quasilinear, and fit into the linear PCP framework.

“As noted above, state-of-the-art PCP constructions leave little to be desired in terms of asymptotic efficiency. However, the practical motivation of verifying computations may call for a more refined efficiency analysis. From this point of view, we believe that our approach has potential to yield better efficiency, at least in some circumstances. … our approach does not inherently require the prover to compute a redundant encoding of its input. This suggests the possibility of designing PCPs that are optimized to make better use of the ‘implicit encoding feature of our approach.”

—Ishai, Kushilevitz, and Ostrovsky,
Efficient Arguments without Short PCPs, 2007

Notes for Attempt 1 (Use asymptotically short PCPs)

(This note relies heavily on [AB09, Ch. 11].)

Let us define the complexity class **PCP**. A language $L \in \mathbf{PCP}(r(n), q(n))$ if:

- **Efficiency.** There is a PPT (probabilistic polynomial-time algorithm) V_L that, on input $a \in \{0,1\}^n$, uses $O(r(n))$ random coins and inspects $O(q(n))$ locations in a string π, after which V_L outputs "accept" or "reject".
- **Completeness.** If $a \in L$, there exists a π s.t. $\Pr\{V_L^{\pi}(a) \text{ accepts}\} = 1$.
- **Soundness.** If $a \notin L$, then for all $\tilde{\pi}$, $\Pr\{V_L^{\tilde{\pi}}(a) \text{ accepts}\} < 1/2$.

The notation $V_L^{\pi}(\cdot)$ denotes V_L with random access to π. The probabilities are taken over the random coins of V_L.

The PCP theorem [ALMSS92,AS92] says: $\mathbf{NP} = \mathbf{PCP}(\log n, 1)$.

What happens if we apply this theory to $\text{QuadConstraint}_{\mathbb{F}}$?

Correct proofs are "short": asymptotically, they are bounded by $2^{O(\log n)} \cdot O(1)$ (because this is the space "swept out" by the verifier's coin flips).

But this length is still longer than n (which in our context, is the length of the execution trace of the computation), so transmitting such a proof to $\mathcal{V}$ would conflict with our efficiency goals.

Notes for Attempt 2 (Use arguments or CS proofs)

Informal and rough definition of an *argument*: an interactive proof (that is, a probabilistic verifier $\mathcal{V}$ and a prover $\mathcal{P}$) plus an assumption that any would-be prover is computationally limited [BCC86, GMR85, Kilian92, Micali94, BG02]. In particular, a dishonest $\mathcal{P}$ with unbounded computational ability can "win" (spuriously convince $\mathcal{V}$ that instances are in a language). In addition, in the current context, we want the protocol to require not much more of an honest $\mathcal{P}$ than T: the time required to decide the given instance's membership (in our context, this corresponds to running the computation f). We also want $\mathcal{V}$'s work to be substantially less than T. These properties were beautifully articulated by [Micali94] in the context of *CS proofs*.

Theorem (informal): Assuming CRHFs (collision-resistant hash functions), arguments (and CS proofs) exist.

The construction is "PCP + tree hash" [Kilian92, Micali94, BG02]. Specifically, $\mathcal{P}$ materializes a PCP, and commits to it using a hash tree [Merkle87]. Then, $\mathcal{V}$ asks $\mathcal{P}$ what values the PCP contains at particular locations; $\mathcal{P}$ is forced (by the CRHF assumption) to respond in a way that is consistent with the original commitment [Merkle87, BEGKN91]. Thus (except with negligible probability), $\mathcal{P}$ "acts like" the fixed proof string of the PCP model.

This results in a 4-message (2-round) scheme. Micali makes this approach non-interactive, in the Random Oracle model. Barak and Goldreich strengthen the analysis so that the construction can work with a CRHF that resists a stronger prover.

Notes for Attempt 2, continued

What happens if we apply this theory to $\text{QuadConstraint}_{\mathbb{F}}$?

This is a great idea in principle. In fact, it remains asymptotically the best approach, if we are limited to standard assumptions (in which case we use the interactive version) or we are comfortable with the Fiat-Shamir Heuristic and the Random Oracle Model (in which case we get a non-interactive version).

But in practice, it's rather costly because of the high constants and intricate constructions in asymptotically short PCP constructions. Indeed, despite intense interest, no experimental results from this approach have been reported.

Notes for Attempt 3 (Use long PCPs interactively)

This approach turns to a *long* (or linear) PCP (a concept that we are fleshing out in this class). This approach is worse in theory than the prior approach; yet, it's a simpler approach, and yields good constants. Perhaps for this reason, the ideas that underly it have been at the heart of published implementations of arguments.

The first claim that we are establishing is certainly not a strong PCP theorem. But the construction that we present highlights some important techniques. We are going through the construction in class, so here are just a few brief notes.

The constraint set $\mathcal{C}$ is over variables in the set X (corresponding to the inputs), the set Y (corresponding to the outputs), and the set Z (corresponding to "intermediate" variables and variables set non-deterministically). We label the constraints in $\mathcal{C}$ as $Q_1, \ldots, Q_m$. Notice: each Q_j is a degree-2 function of (X, Y, Z) and that, at a particular $(X{=}x, Y{=}y)$, the Q_j are functions only of $Z_1, \ldots, Z_n$. Denote $\mathcal{C}(X{=}x, Y{=}y)$ with $\mathcal{C}'$.

Let's notate $Q(z)$ as $Q^{(\gamma)}(z)$ to make clearer that it is a random variable. The reason that $Q^{(\gamma)}(z)$ is a bellwether is as follows:

- If $Z = z$ satisfies $\mathcal{C}'$, then $\Pr_\gamma\{Q^{(\gamma)}(z) = 0\} = 1$.
- If $Z = z$ does not satisfy $\mathcal{C}'$, then $\Pr_\gamma\{Q^{(\gamma)}(z) = 0\} \leq 1/|\mathbb{F}|$.

The second one holds because at least one constraint $Q_{j'}(z)$ is not equal to 0. So the whole sum is equal to 0 only if: $\gamma_{j'} = (-\sum_{j \neq j'} \gamma_j \cdot Q_j(z)) \cdot (Q_{j'}(z))^{-1}$, which has probability $1/|\mathbb{F}|$ (since $\gamma_{j'}$ is conceptually chosen after z).

Notes for Attempt 3, continued

There is a proof oracle such that $Q^{(\gamma)}(z)$ can be evaluated with two queries to that oracle (if it's correctly constructed). To see this, write

$$Q^{(\gamma)}(z) = \langle \lambda_2, z \otimes z \rangle + \langle \lambda_1, z \rangle + \lambda_0,$$

where $v \otimes w$ is the outer product of two vectors, meaning all pairs $v_i w_j$ (how do we know that $Q^{(\gamma)}(z)$ has this form?). Notice that $\lambda_0, \lambda_1, \lambda_2$ depend on γ and the structure of the $Q_j(\cdot)$. In addition, λ_0 depends on x, y (but we can arrange for λ_1 and λ_2 not to have such a dependence).

The proof oracle is then, for some z, two long tables:
(1) $\pi_1 = \langle u_1, z \rangle$ for all $u_1 \in \mathbb{F}^n$
(2) $\pi_2 = \langle u_2, z \otimes z \rangle$ for all $u_2 \in \mathbb{F}^{n^2}$
Notice that π_1, π_2 can be thought of as *linear functions*.

EXERCISE: Convince yourself that if π_1, π_2 are constructed this way, $Q^{(\gamma)}(z)$ can be evaluated with only one query each to π_1 and π_2.

Notes for Attempt 3, continued

But of course π might be constructed adversarially. To address this issue, $\mathcal{V}$ will perform three tests:

- Linearity tests of π_1 and π_2. This consists of choosing two elements a, b at random from the domain of a (purportedly) linear function π and checking whether $\pi(a) + \pi(b) = \pi(a + b)$. (In a linear function, this holds for all a, b in the domain.)
- Quadratic test (self-corrected). This consists of choosing two elements q_5, q_6 at random from the domain of π_1 and an element q_7 from the domain of π_2 and checking whether $\pi_1(q_5) \cdot \pi_1(q_6) = \pi_2(q_5 \otimes q_6 + q_7) - \pi_2(q_7)$.
- Circuit test (self-corrected). This is a modified form of our idealized check on the previous page. Choose random q_8 from the domain of π_1 and q_9 from the domain of π_2 and check whether $\pi_2(\lambda_2 + q_9) - \pi_2(q_9) + \pi_1(\lambda_1 + q_8) - \pi_1(q_8) + \lambda_0 = 0$.

Notes for Attempt 3, continued

The remaining step is to prove the completeness and soundness of this long PCP by analyzing the tests. Take the following as a given:

LEMMA ([BLR90, BGLR93, BCHKS96]): If no linear function is $(1/10)$-close to g, then $\Pr_{a,b}\{\text{linearity test passes}\} \leq 7/9$.

EXERCISE: Prove the following lemma:
If π_1 and π_2 are $(1/10)$-close to some linear functions $\tilde{\pi}_1$ and $\tilde{\pi}_2$ (respectively) and $\Pr\{\pi_1, \pi_2 \text{ pass the quadratic test}\} > 4 \cdot (1/10) + 2/|\mathbb{F}|$ (the probability is taken over the random choices made in the quadratic test), then there exists a vector w for which $\tilde{\pi}_1(\cdot) = \langle \cdot, w \rangle$ and $\tilde{\pi}_2(\cdot) = \langle \cdot, w \otimes w \rangle$.

EXERCISE: Prove the following lemma: If π_1 and π_2 are $(1/10)$-close to some linear functions $\tilde{\pi}_1$ and $\tilde{\pi}_2$ (respectively), if $\tilde{\pi}_1$ and $\tilde{\pi}_2$ are consistent (in the sense of encoding the same vector, z), and $\Pr\{\pi_1, \pi_2 \text{ pass the circuit test}\} > 4 \cdot (1/10) + 1/|\mathbb{F}|$ (the probability is taken over the random choices made in the circuit test), then z is a satisfying assignment.

EXERCISE: Prove completeness and soundness:

- If $\mathcal{C}'$ is satisfiable, there are π_1, π_2 s.t. $\Pr\{\text{all tests pass}\} = 1$.
- If $\mathcal{C}'$ is not satisfiable, then for all π_1, π_2, $\Pr\{\text{all tests pass}\} \leq 7/9$.

Notes on linear commitment primitive

In the security proof for this primitive, we imagine running the “commitment” phase once and the “decommit” phase twice. If $\mathcal{P}$ could, with non-negligible probability, produce different answers for the same query in the two different decommit phases, then $\mathcal{P}$ could break the semantic security of the additively homomorphic encryption scheme.

This in turn means that the prover is bound to a single function after the commitment phase. The details are given in [IKO07], and the compiler given there is improved in [SMBW12].

References

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ALMSS92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, May 1998. Prelim. version FOCS 1992.

[AS92] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, January 1998. Prelim. version FOCS 1992.

[Babai85] L. Babai. Trading group theory for randomness. In *STOC*, May 1985.

[BBFR15] M. Backes, M. Barbosa, D. Fiore, and R. M. Reischuk. ADSNARK: nearly practical and privacy-preserving proofs on authenticated data. In *IEEE Symposium on Security and Privacy*, May 2015.

[BCC86] G. Brassard, D. Chaum, and C. Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and Systems Sciences*, 37(2):156–189, October 1988. Prelim. versions: several papers in CRYPTO and FOCS 1986.

[BCCGLRT14] N. Bitansky, R. Canetti, A. Chiesa, S. Goldwasser, H. Lin, A. Rubinstein, and E. Tromer. The hunting of the SNARK. Cryptology ePrint Archive, Report 2014/580. 2014.

[BCCT12] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, January 2012.

[BCCT13] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. Recursive composition and bootstrapping for SNARKs and proof-carrying data. In *STOC*, June 2013.

[BCGGMTV14] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Decentralized anonymous payments from Bitcoin. In *IEEE Symposium on Security and Privacy*, May 2014.

[BCGT13] E. Ben-Sasson, A. Chiesa, D. Genkin, and E. Tromer. On the concrete-efficiency threshold of probabilistically-checkable proofs. In *STOC*, June 2013.

[BCGTV13] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. SNARKs for C: verifying program executions succinctly and in zero knowledge. In *CRYPTO*, August 2013.

[BCHKS96] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. *IEEE transactions on information theory*, 42(6):1781–1795, November 1996.

[BCIOP13] N. Bitansky, A. Chiesa, Y. Ishai, R. Ostrovsky, and O. Paneth. Succinct non-interactive arguments via linear interactive proofs. In *IACR TCC*, March 2013.

[BCTV14a] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Scalable zero knowledge via cycles of elliptic curves. In *CRYPTO*, August 2014.

[BCTV14b] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza. Succinct non-interactive zero knowledge for a von Neumann architecture. In *USENIX Security Symposium*, August 2014.

[BEGKN91] M. Blum, W. Evans, P. Gemmell, S. Kannan, and M. Naor. Checking the correctness of memories. In *FOCS*, October 1991.

[BF11] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In *Eurocrypt*, May 2011, pages 149–168.

[BFLS91] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *STOC*, May 1991.

[BFR13] M. Backes, D. Fiore, and R. M. Reischuk. Verifiable delegation of computation on outsourced data. In *ACM CCS*, November 2013.

[BFRSBW13] B. Braun, A. J. Feldman, Z. Ren, S. Setty, A. J. Blumberg, and M. Walfish. Verifying computations with state. In *SOSP*, November 2013.

[BG02] B. Barak and O. Goldreich. Universal arguments and their applications. *SIAM Journal on Computing*, 38(5):1661–1694, 2008. Prelim. version CCC 2002.

[BGHSV05] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Short PCPs verifiable in polylogarithmic time. In *Conference on Computational Complexity (CCC)*, 2005.

[BGHSV06] E. Ben-Sasson, O. Goldreich, P. Harsha, M. Sudan, and S. Vadhan. Robust PCPs of proximity, shorter PCPs and applications to coding. *SIAM Journal on Computing*, 36(4):889–974, December 2006.

[BGLR93] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximations. In *STOC*, 1993.

[BGV11] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable delegation of computation over large datasets. In *CRYPTO*, August 2011, pages 111–131.

[BLR90] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and Systems Sciences*, 47(3):549–595, December 1993. Prelim. version STOC 1990.

[Braun12] B. Braun. Compiling computations to constraints for verified computation. UT Austin Honors thesis HR-12-10. December 2012.

[BS08] E. Ben-Sasson and M. Sudan. Short PCPs with polylog query complexity. *SIAM Journal on Computing*, 38(2):551–607, May 2008.

[BZF11] M. Blanton, Y. Zhang, and K. Frikken. Secure and verifiable outsourcing of large-scale biometric computations. In *IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT)*, October 2011.

[CFHKKNPZ15] C. Costello, C. Fournet, J. Howell, M. Kohlweiss, B. Kreuter, M. Naehrig, B. Parno, and S. Zahur. Geppetto: versatile verifiable computation. In *IEEE Symposium on Security and Privacy*, May 2015.

[CL99] M. Castro and B. Liskov. Practical Byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002. Prelim. versions OSDI 1999, OSDI 2000.

[CMT12] G. Cormode, M. Mitzenmacher, and J. Thaler. Practical verified computation with streaming interactive proofs. In *ITCS*, January 2012.

[CRR11] R. Canetti, B. Riva, and G. Rothblum. Practical delegation of computation using multiple servers. In *ACM CCS*, October 2011.

[CT10] A. Chiesa and E. Tromer. Proof-carrying data and hearsay arguments from signature cards. In *ICS*, 2010.

[CTV15] A. Chiesa, E. Tromer, and M. Virza. Cluster computing in zero knowledge. In *Eurocrypt*, April 2015, pages 371–403.

[DFKP13] G. Danezis, C. Fournet, M. Kohlweiss, and B. Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *Workshop on Language Support for Privacy-enhancing Technologies*, November 2013.

[Din07] I. Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3), June 2007.

[FG12] D. Fiore and R. Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In *ACM CCS*, May 2012, pages 501–512.

[FGLSS91] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, March 1996. Prelim. version FOCS 1991.

[FGP14] D. Fiore, R. Gennaro, and V. Pastro. Efficiently verifiable computation on encrypted data. In *ACM CCS*, 2014.

[FL14] M. Fredrikson and B. Livshits. ZØ: an optimizing distributing zero-knowledge compiler. In *USENIX Security Symposium*, August 2014.

[Freivalds77] R. Freivalds. Probabilistic machines can use less running time. In *Proceedings of the IFIP Congress*, 1977, pages 839–842.

[GF95] A. M. Ghuloum and A. L. Fisher. Flattening and parallelizing irregular, recurrent loop nests. In *ACM PPoPP*, July 1995.

[GGP10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: outsourcing computation to untrusted workers. In *CRYPTO*, August 2010.

[GGPR13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *Eurocrypt*, 2013.

[GKR08] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. *Journal of the ACM*, 62(4):27:1–27:64, August 2015. Prelim. version STOC 2008.

[GLR11] S. Goldwasser, H. Lin, and A. Rubinstein. Delegation of computation without rejection problem from designated verifier CS-proofs. Cryptology ePrint Archive, Report 2011/456. 2011.

[GM01] P. Golle and I. Mironov. Uncheatable distributed computations. In *RSA Conference*, April 2001, pages 425–440.

[GMR85] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Prelim. version STOC 1985.

[Goldreich07] O. Goldreich. Probabilistic proof systems – a primer. *Foundations and trends in theoretical computer science*, 3(1):1–91, 2007.

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. New techniques for noninteractive zero-knowledge. *Journal of the ACM*, 59(3):11:1–11:35, June 2012. Prelim. versions CRYPTO 2006, Eurocrypt 2006.

[Groth10] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In *Asiacrypt*, 2010.

[GW11] C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, June 2011.

[HKD07] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: practical accountability for distributed systems. In *SOSP*, October 2007, pages 175–188.

[IKO07] Y. Ishai, E. Kushilevitz, and R. Ostrovsky. Efficient arguments without short PCPs. In *Conference on Computational Complexity (CCC)*, 2007.

[Kilian92] J. Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, May 1992.

[Knijnenburg98] P. M. W. Knijnenburg. Flattening: VLIW code generation for imperfectly nested loops. In *CPC98*, June 1998.

[KNP05] A. Kejariwal, A. Nicolau, and C. D. Polychronopoulos. Enhanced loop coalescing: a compiler technique for transforming non-uniform iteration spaces. In *ISHPC05/ALPS06*, September 2005.

[KP15] Y. T. Kalai and O. Paneth. Delegating RAM computations. Cryptology ePrint Archive, Report 2015/957. 2015.

[KPPSST14] A. E. Kosba, D. Papadopoulos, C. Papamanthou, M. F. Sayed, E. Shi, and N. Triandopoulos. TRUESET: faster verifiable set computations. In *USENIX Security Symposium*, August 2014.

[KR09] Y. T. Kalai and R. Raz. Probabilistically checkable arguments. In *CRYPTO*, 2009.

[KRR14] Y. T. Kalai, R. Raz, and R. Rothblum. How to delegate computations: the power of no-signaling proofs. In *STOC*, 2014.

[KSC09] G. O. Karame, M. Strasser, and S. Čapkun. Secure remote execution of sequential computations. In *International Conference on Information and Communications Security*, December 2009.

[KZMQCPPsS15] A. Kosba, Z. Zhao, A. Miller, Y. Qian, H. Chan, C. Papamanthou, R. Pass, abhi shelat, and E. Shi. How to use SNARKs in universally composable protocols. Cryptology ePrint Archive, Report 2015/1093. 2015.

[Lipmaa11] H. Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In *IACR TCC*, 2011.

[Meir12] O. Meir. Combinatorial PCPs with short proofs. In *Conference on Computational Complexity (CCC)*, 2012.

[Merkle87] R. C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, August 1987.

[Micali94] S. Micali. Computationally sound proofs. *SIAM Journal on Computing*, 30(4):1253–1298, 2000. Prelim. version FOCS 1994.

[MR97] D. Malkhi and M. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, October 1998. Prelim. version STOC 1997.

[MSG07] N. Michalakis, R. Soulé, and R. Grimm. Ensuring content integrity for untrusted peer-to-peer content distribution networks. In *Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[MWR99] F. Monrose, P. Wycko, and A. D. Rubin. Distributed execution with remote audit. In *Network and Distributed System Security Symposium (NDSS)*, February 1999.

[PGHR13] B. Parno, C. Gentry, J. Howell, and M. Raykova. Pinocchio: nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, May 2013.

[PMP11] B. Parno, J. M. McCune, and A. Perrig. *Bootstrapping trust in modern computers*. Springer, 2011.

[Polychron87] C. D. Polychronopoulos. Loop coalescing: a compiler transformation for parallel machines. In *ICPP*, August 1987.

[SBVBPW13] S. Setty, B. Braun, V. Vu, A. J. Blumberg, B. Parno, and M. Walfish. Resolving the conflict between generality and plausibility in verified computation. In *Eurosys*, April 2013.

[SBW11] S. Setty, A. J. Blumberg, and M. Walfish. Toward practical and unconditional verification of remote computations. In *Workshop on Hot Topics in Operating Systems (HotOS)*, May 2011.

[Sion05] R. Sion. Query execution assurance for outsourced databases. In *International Conference on Very Large Databases (VLDB)*, August 2005, pages 601–612.

[SLSPDK05] A. Seshadri, M. Luk, E. Shi, A. Perrig, L. van Doorn, and P. Khosla. Pioneer: verifying integrity and guaranteeing execution of code on legacy platforms. In *SOSP*, October 2005.

[SMBW12] S. Setty, R. McPherson, A. J. Blumberg, and M. Walfish. Making argument systems for outsourced computation practical (sometimes). In *Network and Distributed System Security Symposium (NDSS)*, February 2012.

[SSW10] A.-R. Sadeghi, T. Schneider, and M. Winandy. Token-based cloud computing: secure outsourcing of data and arbitrary computations with lower latency. In *TRUST*, June 2010.

[SVPBBW12] S. Setty, V. Vu, N. Panpalia, B. Braun, A. J. Blumberg, and M. Walfish. Taking proof-based verified computation a few steps closer to practicality. In *USENIX Security Symposium*, August 2012.

[Thaler13] J. Thaler. Time-optimal interactive proofs for circuit evaluation. In *CRYPTO*, August 2013.

[TRMP12] J. Thaler, M. Roberts, M. Mitzenmacher, and H. Pfister. Verifiable computation with massively parallel interactive proofs. In *USENIX HotCloud Workshop*, June 2012.

[VSBW13] V. Vu, S. Setty, A. J. Blumberg, and M. Walfish. A hybrid architecture for interactive verifiable computation. In *IEEE Symposium on Security and Privacy*, May 2013.

[WB15] M. Walfish and A. J. Blumberg. Verifying computations without reexecuting them: from theoretical possibility to near practicality. *Communications of the ACM*, 58(2):74–84, February 2015.

[WHGsW15] R. Wahby, M. Howald, S. Garg, abhi shelat, and M. Walfish. Verifiable ASICs. Preprint. December 2015.

[WSRBW15] R. S. Wahby, S. Setty, Z. Ren, A. J. Blumberg, and M. Walfish. Efficient RAM and control flow in verifiable outsourced computation. In *Network and Distributed System Security Symposium (NDSS)*, February 2015.

[YCFVEEGH08] B. Ylvisaker, A. Carroll, S. Friedman, B. Van Essen, C. Ebeling, D. Grossman, and S. Huack. Macah: a "C-level" language for programming kernels on coprocessor accelerators. Technical report. University of Washington, Department of CSE, 2008.