COSC 544 Probabilistic Proof Systems 10/24/17

Linear PCPs and Succinct Arguments

Lecturer: Justin Thaler

1 Interactive Arguments From "Long", Structured PCPs

1.1 Overview

Ishai, Kushilevitz, and Ostrovsky [IKO07] (IKO) put forth an ingenious two-step methodology for developing argument systems for arithmetic circuit satisfiability, without the use of "short" PCPs (which, as mentioned in the previous lecture, tend to be quite complicated). The basic idea is the following. Why is the prover inefficient if one instantiates Kilian's argument system with a PCP of superpolynomial size $L = n^{\omega(1)}$? The problem is that $\mathcal{P}$ has to materialize the full proof π in order to compute its Merkle-hash and thereby "commit" to π. Materializing a proof of superpolynomial length clearly takes superpolynomial time.

But IKO [IKO07] show that if π is highly structured in a manner made precise below, then there is a way for $\mathcal{P}$ to commit to π *without* materializing all of it.This enables them to use structured PCPs of exponential size, which turn out to be much simpler than "short" PCPs for circuit satisfiability (by short, we mean of length quasilinear in the circuit size). The commitment protocol of [IKO07] is based on any semantically secure, additively-homomorphic cryptosystem, such as the Goldwasser-Micali cryptosystem [GM84].

The downside of this approach is that in order for $\mathcal{P}$ the commit to a proof π of size L, the verifier must first send $\mathcal{P}$ a message of size $\log L$. If L is exponential in the circuit size $S = |\mathcal{C}|$, then even $\log(L)$ is *linear* in the circuit size, so even writing down the verifier's message takes time $\Omega(S)$. Compare this to the MIPs and PCPs from previous lectures, where the verifier's total runtime was $O(n + \text{polylog}(S))$.

However, these downsides are offset by the ability of the verifier to amortize this large cost when outsourcing the evaluation of $\mathcal{C}$ on many separate inputs. In addition, the communication in the reverse direction, from prover to verifier, is *very* small (a constant number of field elements) and the verifier's online verification phase is especially fast as a consequence ($\mathcal{V}$ performs just $O(1)$ field or group operations to check $\mathcal{P}$'s final message).

1.2 Linear PCPs

The type of structure that IKO [IKO07] exploit is linearity. Specifically, in a *linear PCP*, the proof is interpreted as a function mapping $\mathbb{F}^v \to \mathbb{F}$ for some integer $v > 0$. A linear PCP is one in which the "honest" proof is a linear function π. That is, π should satisfy that for any two queries $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{F}^v$, $\pi(\mathbf{q}_1 + \mathbf{q}_2) = \pi(\mathbf{q}_1) + \pi(\mathbf{q}_2)$. This is the same as requiring that π be a v-variate polynomial of total degree 1, with constant term equal to 0. Note that in a linear PCP, soundness should even against "cheating" proofs that are non-linear. So the only difference between a linear PCP and a PCP is that in a linear PCP, the honest proof is guaranteed to have special structure.

The arguments of IKO [IKO07] conceptually proceeds in the same two steps as Kilian's approach based on short PCPs. In the first step, the prover commits to the proof, but unlike in Kilian's approach, here the prover can leverage the linearity of the proof to commit to it without ever materializing the whole thing.

In the second step, the argument system verifier simulates the linear PCP verifier asks the prover to reveal certain locations of the proof, which the prover does using the reveal phase of the commitment protocol.

Hence, in order to give an efficient argument system, IKO [IKO07] had to do two things. First, they gave a commit/reveal protocol for linear functions. Second, they gave a linear PCP for arithmetic circuit satisfiability.

Unfortunately, when applied to circuits of size S, their linear PCP had length $|\mathbb{F}|^{O(S^2)}$. It takes $O(S^2)$ time for the verifier to even *write down* a query into a proof of this length. A subsequent refinement of Gennaro, Gentry, Parno, and Raykova (GGPR) [GGPR13] give a linear PCP[1] of length $|\mathbb{F}|^{O(S)}$. GGPR's linear PCP has formed the theoretical foundation for many of the argument systems that have been implemented in the research literature and deployed in commercial settings.

In this lecture, we cover only IKO's commit/reveal protocol for linear functions. Next lecture we will see IKO's linear PCP of length $|\mathbb{F}|^{O(S^2)}$, and the following lecture we will see GGPR's linear PCP of length $|\mathbb{F}|^{O(S)}$.

1.3 Committing to a Linear PCP without Materializing It

Let π be a linear function $\mathbb{F}^v \to \mathbb{F}$. This section sketches the technique of IKO [IKO07] for allowing the prover to first commit to π in a "commit phase" and then answer a series of queries k queries $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(k)} \in \mathbb{F}^v$ in a "reveal phase". Roughly speaking, the security guarantee is that at the end of the commit phase, there is *some* function π' (which may not be linear) such that, if the verifier's checks in the protocol all pass and $\mathcal{P}$ cannot break the cryptosystem used in the protocol, then the prover's answers in the reveal phase are all consistent with π'.[2]

In more detail, the protocol uses a semantically secure homomorphic cryptosystem. Roughly speaking, semantic security guarantees that, given a ciphertext $c = \text{Enc}(m)$ of a plaintext m, any probabilistic polynomial time algorithm cannot "learn anything" about m. A cryptosystem is (additively) homomorphic if it is possible, for any pair of plain texts (m_1, m_2), it is possible to efficiently compute the encryption of $m_1 + m_2$ from the encryptions of m_1 and m_2 individually.

In the commit phase, the verifier chooses a vector $\mathbf{r} = (r_1, \ldots, r_v) \in \mathbb{F}^v$ at random, encrypts each entry of $\mathbf{r}$ and sends all v encryptions to the prover. Since π is linear, there is some vector $\mathbf{d} = (d_1, \ldots, d_v) \in \mathbb{F}^v$ such that $\pi(\mathbf{q}) = \sum_{i=1}^{v} d_i \cdot q_i = \langle \mathbf{d}, \mathbf{q} \rangle$ for all queries $\mathbf{q} = (q_1, \ldots, q_v)$. Hence, using the homomorphism property of the encryption scheme, the prover can efficiently compute the encryption of $\pi(\mathbf{r})$ from the encryptions of the individual entries of $\mathbf{r}$. The prover sends this encryption to the verifier, who decrypts it to obtain (what is claimed to be) $s = \pi(\mathbf{r})$.[3]

In the reveal phase, the verifier picks k field elements $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$ at random, and keeps them secret. The verifier then sends the prover the queries $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(k)}$ in the clear, as well as $\mathbf{q}^* = \mathbf{r} + \sum_{i=1}^{k} \alpha_i \cdot \mathbf{q}^{(i)}$ (throughout, we can assume that none of the queries are the all-zeros vector, since any linear function π evaluates to 0 on the all-zeros vector). The prover returns claimed answers $a^{(1)}, \ldots, a^{(k)}, a^* \in \mathbb{F}$, which are

[1] The observation that GGPR's protocol is actually a linear PCP as defined by IKO was made in later work [BCI+13, SBV+13].

[2] This section actually sketches a refinement of IKO's commitment/reveal protocol, due to Setty et al. [SMBW12]. The original protocol of IKO guaranteed that for each query i, there is a separate function π_i to which $\mathcal{P}$ was committed. Setty et al. [SMBW12] tweaked the protocol of IKO in a way that both reduced costs and guaranteed that $\mathcal{P}$ was committed to answering all k queries using a single function π' (which is possibly non-linear).

[3] This is kind of amazing. Using the homomorphism property of Enc and the linearity of π, the honest prover has managed to send to the verifier an encoding of $\pi(\mathbf{r})$, even though the prover has *no idea what* $\mathbf{r}$ *is* (this is roughly what semantic security of Enc guarantees). Moreover, the prover can compute $\text{Enc}(\pi(\mathbf{r}))$ from $\text{Enc}(r_1), \ldots, \text{Enc}(r_v)$ in time $O(v)$. This is far less time the $\Omega(|\mathbb{F}|^v)$ time required to evaluate π at all points, which would be required if the prover were to build a Merkle tree with the evaluations of π as the leaves.

supposed to equal $\pi(\mathbf{q}^{(1)}),\ldots,\pi(\mathbf{q}^{(k)}),\pi(\mathbf{q}^*)$. The verifier checks that $a^* = s + \sum_{i=1}^{k} \alpha_i \cdot a^{(i)}$, accepting the answers as valid if so, and rejecting otherwise.

Clearly the verifier's check will pass if the prover is honest. The proof of binding roughly argues that the only way for $\mathcal{P}$ to pass the verifier's tests, if $\mathcal{P}$ does not answer all queries using a single function, is to know the α_i's, in the sense that one can efficiently compute the α_i's given access to such a prover. But if the prover "knows" the α_i's, then the prover must be able to solve for $\mathbf{r}$, since $\mathcal{V}$ reveals $\mathbf{q}^* = \mathbf{r} + \sum_{i=1}^{k} \alpha_i \cdot \mathbf{q}^{(i)}$ to the prover. But this would contradict the semantic security of the underlying cryptosystem, which guarantees that the prover learned nothing about $\mathbf{r}$ from the encryptions of $\mathbf{r}$'s entries.

1.4 Detailed Presentation of Binding Property When $k = 1$

We present the main idea of the proof of binding, in the case the $k = 1$.

What does it mean for the prover *not* to be bound to a fixed function after the commitment phase of the protocol? It means that there are at least two runs of the reveal protocol, where in the first run, the verifier sends queries $\mathbf{q}_1$ and $\mathbf{q}^* = \mathbf{r} + \alpha \cdot \mathbf{q}_1$, and the prover responds with answers a_1 and a^*, while in the second run the verifier sends queries $\mathbf{q}_1$ and $\hat{\mathbf{q}} = \mathbf{r} + \alpha' \cdot \mathbf{q}_1$, and the prover responds with answers $a_1' \neq a_1$ and $\hat{a}$. That is, in two different runs of the reveal protocol, the prover responded to the same query $\mathbf{q}_1$ with two different answers, and managed to pass the verifier's checks.

We will argue that in this case, the prover must *know* α *and* α'. But this breaks the semantic security of the encryption scheme. Roughly speaking, this is because if the prover really learned nothing about $\mathbf{r}$ from $\mathrm{Enc}(r_1),\ldots,\mathrm{Enc}(r_v)$ as promised by semantic security of Enc, then it should be impossible for the prover to determine α with probability any better than random guessing, even given $\mathbf{q}_1$, $\mathbf{q}^* = \mathbf{r} + \alpha \cdot q_1$, and $\hat{\mathbf{q}} = \mathbf{r} + \alpha' \cdot \mathbf{q}_1$. This is because, without knowing $\mathbf{r}$, all that the equations

$$\mathbf{q}^* = \mathbf{r} + \alpha q_1 \tag{1}$$

and

$$\hat{\mathbf{q}} = \mathbf{r} + \alpha' \cdot \mathbf{q}_1 \tag{2}$$

tell the prover about α and α' is that they are two field elements satisfying $\mathbf{q}^* - \hat{\mathbf{q}} = (\alpha - \alpha')\mathbf{q}_1$. That is, for every pair $\alpha, \alpha' \in \mathbb{F}$ satisfying Equations (1) and (2) for $\mathbf{r}$, and any $c \in \mathbb{F}$, the pair $\alpha + c, \alpha' + c$ also satisfy both equations when $\mathbf{r}$ is replaced by $\mathbf{r} + c\mathbf{q}_1$. So without knowing anything about $\mathbf{r}$, Equations (1) and (2) reveal no information whatsoever about α itself.

Showing that the prover must know α and α'. Recall that s is the decryption of the value sent by the prover in the commit phase, which is claimed to be $\mathrm{Enc}(\pi(r))$. The prover doesn't know s, but the verifier does. Even though the prover does not know s, if the verifier's checks in the two runs of the reveal phase pass, then the prover does know that:

$$a^* = s + \alpha a_1, \tag{3}$$

and

$$\hat{a} = s + \alpha' a_1, \tag{4}$$

Subtracting these two equations means that the prover knows that

$$(a^* - \hat{a}) = \alpha a_1 - \alpha' a_1', \tag{5}$$

Similarly, even though the prover doesn't know r, the prover does know that Equations (1) and (2) hold. Subtracting those two equations implies that $\mathbf{q}^* - \hat{\mathbf{q}} = (\alpha - \alpha')\mathbf{q}_1$.

In particular, since we have assumed $\mathbf{q}_1$ is not the all-zeros vector, if we let j denote any non-zero coordinate of $\mathbf{q}_1$, then:

$$\mathbf{q}^*_j - \hat{\mathbf{q}}_j = (\alpha - \alpha')\mathbf{q}_{1,j} \tag{6}$$

Since $a_1 \neq a'_1$, Equations (5) and (6) express α and α' via two linearly independent equations in two unknowns, and these have a unique solution. Hence, the prover can solve these two equations for α and α' as claimed.

References

[BCI+13] Nir Bitansky, Alessandro Chiesa, Yuval Ishai, Rafail Ostrovsky, and Omer Paneth. Succinct non-interactive arguments via linear interactive proofs. In *TCC*, pages 315–333, 2013.

[GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct NIZKs without PCPs. In *EUROCRYPT*, pages 626–645, 2013.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[IKO07] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. Efficient arguments without short pcps. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007), 13-16 June 2007, San Diego, California, USA*, pages 278–291. IEEE Computer Society, 2007.

[SBV+13] Srinath T. V. Setty, Benjamin Braun, Victor Vu, Andrew J. Blumberg, Bryan Parno, and Michael Walfish. Resolving the conflict between generality and plausibility in verified computation. In Zdenek Hanzálek, Hermann Härtig, Miguel Castro, and M. Frans Kaashoek, editors, *EuroSys*, pages 71–84. ACM, 2013.

[SMBW12] Srinath T. V. Setty, Richard McPherson, Andrew J. Blumberg, and Michael Walfish. Making argument systems for outsourced computation practical (sometimes). In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.