Stanford University
Topics in Cryptography (CS 355) **Lecture #11**
Instructor: Lior Rotem **May 9, 2023**

Disclaimer *These lecture notes are aimed to serve as a supplementary resource, and are not written as a replacement for attending the in-person lecture. Some material might appear here in a more sketched form than in the lecture, and vice versa. These notes probably contain typos and might even contain errors. If you find any, please let me know.*

Outline *In the last couple of lectures we covered interactive and non-interactive proof systems. We introduced zero-knowledge as a desirable property that makes such protocols non-trivial to construct. In this lecute, we introduce yet another desirable property that is highly non-trivial to obtain: succinctness. We define what it means for an argument system to be succinct, and present a construction based on PCPs. Then, to set up the group for the next lecture, in which we will see a more efficient construction, we define and construct polynomial commitment schemes.*

1 Succinct Arguments

In the last few lectures, we encountered the magical notion of zero-knowledge (ZK) proofs and arguments, and saw that any NP language has a ZK proof. We also saw how to make public-coin proofs non-interactive using the Fiat-Shamir transform. Today, we will take the magic one step further and consider **succinct** non-interactive arguments (SNARGs) for NP. Before we define what we mean by succinct, let us consider an example.

Say that we want to outsource some heavy computation to a cloud service provider. We send the input x to the server, which responds with the output y. But can we really trust this output? The server provider is incentivized to minimize the computational work of its machines. What we can do, is ask that the server appends a non-interactive proof π that the computation was performed correctly. The computation is actually polynomial time, so a proof exists. But what have we gained here? Since the computation is polynomial-time, the server can just send an empty proof, and to verify, we can redo the computation. But this defeats the purpose of outsourcing it in the first place! Alternatively, the server can send the trace of the computation, but this runs into similar problems. What we want, is a proof π which is much shorter than the trace of the computation, and the verifier's work is greatly diminished relative to performing the entire computation.

More formally, let R be an NP relation, and let T denote the runtime of the associated NP verifier.[1] Let (P, V) be an interactive protocol for R. In this lecture we say that (P, V) is *succinct* if when the protocol is executed on input $(x, w) \in R$:

1. The total communication is $O(|x|) + o(|w| + T)$.
2. V's runtime is $O(|x|) + o(|w| + T)$.

[1] For concreteness, we focus on uniform algorithms. The definition readily extends to cover other models of computation as well.

2 Succinct Non-interactive Arguments (SNARGs) from PCPs

The construction of SNARGs for NP that we will see today relies on the PCP theorem, one of the highest-regarded accomplishments in computational complexity of the past few decades. Roughly, the PCP theorem states that any NP language $\mathcal{L}$ has a *probabilistically checkable proof* (PCP). This means that if $x \in \mathcal{L}$, then the prover can compute a long proof $\pi \in \{0,1\}^*$ asserting that this is indeed the case. The magic is in how the proof is verified. The verifier does not have to read the entire proof π, it just has to read a few of the bits of π! "Few" can be as little as 3!

Completeness of a PCP says that if $x \in \mathcal{L}$ then the verifier accepts with probability 1. Soundness guarantees that even a malicious prover cannot convince a verifier to accept with a probability greater than 1/2. What do we mean by that? We consider an ideal world, in which the prover first decides on π. Then, the verifier chooses three locations $i_1, i_2, i_3 \in \{1, \ldots, |\pi|\}$. The prover then *must* respond with $\pi[i_1], \pi[i_2], \pi[i_3]$ and is not allowed to change π after seeing i_1, i_2, i_3. Then, if $x \notin \mathcal{L}$, with probability at least 1/2 over the choice of i_1, i_2, i_3, the verifier will reject. Soundness can be amplified using parallel repetition.

We will not see how to construct PCPs in this course, but we will now see how to use them to construct a SNARG, following ideas by Kilian (1992) and Micali (1994).

From PCPs to SNARGs. The basic idea is to have the prover P compute a PCP proof π and use a Merkle Tree to commit to π. Then, P can can open $\pi[i_1], \pi[i_2], \pi[i_3]$. On joint input x, the protocol (P, V) is proceeds as follows:

1. P computes a PCP proof π for proving $x \in \mathcal{L}$. It then computes $c \leftarrow \text{MerkleCommit}(\pi)$ and sends c to V.[2]
2. V chooses three indices i_1, i_2, i_3 to query according to the PCP theorem. V sends i_1, i_2, i_3 to P.
3. P responds with $\pi[i_1], \pi[i_2], \pi[i_3]$ together with local openings of c with respect to these locations.
4. V accepts iff all of the local openings are valid with respect to c, and the PCP verifier accepts $(\pi[i_1], \pi[i_2], \pi[i_3])$.

Correctness follows from the correctness of the Merkle Tree commitment scheme and the PCP theorem.

We will not prove soundness, but the intuition is the following. If the Merkle Tree is built using a collision-resistant hash function, then once c is fixed, P can only open the indices i_1, i_2, i_3 to a single value each (we stress again that this is just intuition, not a proof!). So we can rely on the soundness of the PCP protocol.

Making the argument non-interactive and knowledge sound. After parallel repetition, the argument has negligible soundness error, so we can rely on the Fiat-Shamir transform to make

[2]If one uses a keyed hash function to compute the Merkle Tree, then an additional round of communication is needed, in which V samples the hash function and sends it to P. Alternatively, one can assume a keyless hash function modeled as a random oracle.

the protocol non-interactive, resulting in a SNARG. The SNARG could be made into a SNARK (i.e., an argument of knowledge), but we will not be concerened with knowledge soundness today.

Concrete efficiency. The SNARG that we just saw works in theory, but it is not a realistic approach towards implementing SNARGs in practice. The problem is that the PCP theorem gives us a proof that is asymptotically reasonable, but requires too many computational resources to actually construct in practice.

A broader perspective. The above succinct argument can be seen in more generic terms as being constructed in two steps. The first step is to construct a succinct interactive proof system for NP in an idealized world. This part is information-theoretic and requires no cryptography. In our case, this first step was a proof in the PCP model. The second step is to compile this proof system into a succinct argument in the standard model using cryptographic tools. In the example that we saw, this compilation is done using *vector commitments* (recall HW1). We used Merkle Trees as a specific instantiation of vector commitments, but other options also work.

Better efficiency via Poly-IOPs. This two-step view suggests a path towards improving the efficiency of SNARGs: perhaps we can assume a "more idealized" world, in which the verifier has more power. This can lead to more efficient proofs in this world. We will see a specific example of such an approach: instead of constructing a PCP, we will use an idealized object called *Polynomial Interactive Oracle Proofs (Poly-IOPs).* This generalizes PCPs in two respects. First, we allow for added interaction. The proof proceeds in rounds, where in each round, the prover fixes some string that the verifier can then "locally" query. The second generalization is that the verifier is not restricted to querying local bits of the proof strings. Instead, we interpret the proof strings as polynomials, and the verifier can query for evaluations of these polynomials at random points. We will see this model in more detail in the next lecture.

Compiling a Poly-IOP to a SNARG in the standard model requires heavier cryptographic machinery than just vector commitments. The added interaction does not pose a major problem, and we can still apply the Fiat-Shamir transform. But now we need a commitment scheme that allows the verifier to query for evaluation of committed polynomials, and not just individual entries of a committed string.

3 Polynomial Commitments

A polynomial commitment scheme is a generalization of vector commitment schemes. Informally, it allows the committer to commit to some polynomial f over a finite field $\mathbb{F}$. We will be interested in the case where $\mathbb{F} = \mathbb{F}_p$ for some prime p. More formally, a polynomial commitment scheme is a tuple of 4 algorithms:

- $\mathit{Setup}(d) \to pp$. // d is a bound on the degree of f[3]
- $\mathit{Commit}(pp, f) \to c$.

[3]Formally, the Setup algorithm should take in the security parameter. Looking ahead, we will fix a cryptographic group over which the final scheme will be defined, and assume that this group encodes the security parameter. Setup will take the description of the group as input, even if we do not explicitly note that fact.

- *Open*$(pp, f, x) \to \pi$.
- *Verify*$(pp, c, x, y, \pi) \to 0/1$.

Correcntess essentially requires an honestly-generated commitment and opening to pass verification. That is, for every $\lambda \in \mathbb{N}$, every polynomial $f \in \mathbb{F}[X]$, and every $x \in \mathbb{F}$, it holds that

$$\Pr\left[\mathit{Verify}(pp, c, x, y, \pi) = 1 \ : \ \begin{array}{c} pp \xleftarrow{\$} \mathit{Setup}(1^\lambda) \\ c \xleftarrow{\$} \mathit{Commit}(pp, f) \\ \pi \leftarrow \mathit{Open}(pp, f, x) \end{array} \right] = 1.$$

The biding property for polynomial commitments is called **evaluation binding**. Informally, it requires that no PPT adversary can produce a commitment c and open it at a point $x \in \mathbb{F}$ to two distinct values y and y'. More formally, for any PPT adversary A there exists a negligible function negl such that:

$$\Pr\left[\begin{array}{c} \mathit{Verify}(pp, c, x, y, \pi) = 1 \\ \wedge \mathit{Verify}(pp, c, x, y', \pi') = 1 \\ \wedge y \neq y' \end{array} \ : \ \begin{array}{c} pp \xleftarrow{\$} \mathit{Setup}(1^\lambda) \\ (c, x, y, \pi, y', \pi') \xleftarrow{\$} A(pp) \end{array} \right] \leq \mathsf{negl}(\lambda).$$

One can also define **extractability**, which informally means that π is an argument of knowledge for the underlying polynomial f. This is necessary if we want to prove that the resulting SNARG is actually a SNARK (a succinct non-interactive argument *of knowledge*). However, it is trickier to define and to prove, and we will not consider this property today.

An inefficient construction from vector commitments. A vector commitment scheme trivially yields the following polynomial commitment scheme: To commit to a polynomial f, just commit to the vector $\mathbf{v} = (f(0), f(1), \ldots, f(p-1))$. An opening at point $x \in \mathbb{F}_p$ is then just an opening of the $(x-1)$th location of the vector $\mathbf{v}$. This works, but the scheme is inefficient if the p is large. The power of Poly-IOPs over PCPs will come from working over a large field (and hence, evaluating a polynomial at a certain point might encode more information than querying a bit in a PCP string). Hence, the above suggested polynomial commitment scheme will be too inefficient for us.

A first idea: Commit to coefficients. The first idea is to use Pedersen Commitments to commit to the coefficients of f. It is parameterized by a group $\mathbb{G}$ or order p generated by g and is defined as follows:

- *Setup*$(d) \to pp = (g, h)$ where $h \xleftarrow{\$} \mathbb{G}$.
- *Commit*$((g, h), f) \to (c = (c_i)_i, \mathsf{st} = (r_i)_i)$, where $r_i \xleftarrow{\$} \mathbb{Z}_p$ and $c_i \leftarrow g^{r_i} \cdot h^{f_i}$ for every $i \in \{0, \ldots, d\}$.[4]
- *Open*$(pp, f, x, \mathsf{st}) \to \pi$, where $\pi \leftarrow \sum_{i=0}^{d} r_i x^i$.

[4]Note that the *Commit* algorithm also outputs a secret state st that will be used by the *Open* algorithm. This generalizes the definition for polynomial commitments that we saw before, and will not be needed by the more efficient scheme we will shortly.

- *Verify*(pp, c, x, y, π) outputs 1 iff $g^\pi \cdot h^y = \prod_i c_i^{x^i}$.

Observe that the scheme is correct since for $y = f(x)$ it holds that

$$\prod_i c_i^{x^i} = \prod_i g^{r_i x^i} \cdot h^{f_i x^i} = g^\pi \cdot h^y.$$

Evaluation binding can be shown based on the discrete log assumption, similarly to the binding property of standard Pedersen Commitments. We will not see this here.

The problem with this approach is efficiency. The size of the commitment and the verification time both grow linearly with the degree of f. This is not enough to construct a SNARG from Poly-IOPs, and we can do better, as we will now see.

KZG commitments (or: pairings to the rescue, again). We consider an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$. Denote $g_T = e(g_1, g_2)$. The scheme is defined as follows:

- *Setup*(d): Sample $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and set $v \leftarrow g_2^\alpha$ and $u_i \leftarrow g_1^{\alpha^i}$ for every $i \in \{0, \dots, d\}$. Output $pp = ((u_i)_i, v)$.
- *Commit*$(((u_i)_i, v), f)$: Output $c \leftarrow g_1^{f(\alpha)} = \prod_i u_i^{f_i}$.
- *Open*(pp, f, x): Let $h(X) = \frac{f(X)-f(x)}{X-x}$. Output $\pi \leftarrow g_1^{h(\alpha)}$ (computed as above).
- *Verify*(pp, c, x, y, π) Output 1 iff $e(\pi, v/g_2^x) = e(c/g_1^y, g_2)$.

Correctness holds since

$$e(\pi, v/g_2^x) = e(g_1^{h(\alpha)}, g_2^{\alpha - x}) = e\left(g_1^{\frac{f(\alpha)-f(x)}{\alpha - x}}, g_2^{\alpha - x}\right) = g_T^{f(\alpha)-y} = e(g_1^{f(\alpha)-y}, g_2) = e(c/g_1^y, g_2)$$

Evaluation binding is proven via a reduction from the t-Bilinear Strong Diffie-Hellman (t-BSDH) assumption. The assumption states that given $(g_1, g_1^\alpha, \dots, g_1^{\alpha^t}, g_2^\alpha)$ it is hard to come up with a constant $\gamma \neq \alpha$ and $g_T^{\frac{1}{\alpha+\gamma}}$.

Consider an adversary A that breaks evaluation binding. This means that it outputs (c, x, y, π, y', π') such that *Verify*$(pp, c, x, y, \pi) = 1$, *Verify*$(pp, c, x, y', \pi') = 1$ and $y \neq y'$. From the validity of the first verification, we obtain that

$$e(\pi, v/g_2^x) = e(c/g_1^y, g_2) \implies \pi^{\alpha - x} \cdot g_1^y = c.$$

Similarly, from the second verification, we get that

$$(\pi')^{\alpha - x} \cdot g_1^{y'} = c.$$

This implies that

$$(\pi/\pi')^{\frac{1}{y'-y}} = g_1^{\frac{1}{\alpha - x}}$$

and so the reduction can just output $\gamma = -x$ and the group element $e((\pi/\pi')^{\frac{1}{y'-y}}, g_2)$.