Lecture 22: SNARGs *for* NP

Notes by Yael Kalai

MIT - 6.5620
Lecture 21 (November 26, 2025)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Succinct Interactive Arguments for NP.
- SNARGs for NP.
- Secret Sharing.

Recap

Last lecture we saw the GKR protocol for bounded depth computations.

Overcoming the Low-Depth Restriction in the GKR protocol

We can use the GKR blueprint to construct doubly efficient protocols for any computation where the verifier's runtime and the number of rounds *do not grow* with the depth of the computation! Rather they grow only poly-logarithmically with the size. The basic idea is to "flatten" the circuit. Namely, suppose the prover wishes to prove to the verifier that $C(x) = y$ where C is of size S. The idea is for the prover and verifier to consider a new *shallow* circuit C' that takes as input an S-bit string, which corresponds to the values of all the wires of C, and checks that all the gates of C are satisfied. Importantly, note that C' is very shallow and is of depth $O(\log S)$. So, the idea is to run the GKR protocol on the shallow circuit C'.

Assume that S is the number of wires in C.

The problem is that the verifier does not know the input to C' since he cannot compute all the gates of C on its own – this is precisely the work we are trying to offload to the prover! As a result, the verifier cannot verify the GKR protocol, since to verify it the verifier needs to compute on its own a random point in the multi-linear extension of the input to C'.

Cryptography to the rescue

To overcome this problem we have the prover "commit" to the multilinear extension $\tilde{V}_0$ of the input to C', using a special commitment which is referred to as a "polynomial commitment." Then the prover and verifier run the GKR protocol on the flattened circuit C', at the end at which the verifier needs to check the validity of $\tilde{V}_0(z_0) = v_0$. This is done by the prover "opening" the commitment at the point z_0.

A more detailed description follows.

1. The prover does the following:
   (a) Compute the string V_0 which corresponds to all the wires in $C(x)$.
   (b) Compute $\tilde{V}_0$ which is the multilinear extension of V_0.
   (c) Send to the verifier a succinct "polynomial commitment"of $\tilde{V}_0$

   A polynomial commitment is a succinct and binding commitment of $\tilde{V}_0$ that allows the prover to "open" to any evaluation of $\tilde{V}_0$ succinctly.
2. The prover and verifier run the interactive GKR protocol w.r.t. C' on input V_0.

   To verify this protocol the verifier needs to check a claim of the form $\tilde{V}_0(z) = t$ for given $z \in \mathbb{F}^m$ and $t \in \mathbb{F}$.
3. The prover will "open" the polynomial commitment of $\tilde{V}_0$ at the point z.

We will not elaborate on how polynomial commitments are constructed. In the PSet, there is a polynomial commitment construction for a univariate polynomial.

Succinct Interactive Proofs for NP

Note that the above protocol gives a succinct interactive proof for NP.

1. First the prover sends a polynomial commitment to the multilinear extension of the NP witness w, which is denoted by $\widetilde{W} : \mathbb{F}^m \to \mathbb{F}$.
2. Run the GKR protocol with respect to the verification circuit that has the input x hardwired, and on input w outputs 1 if and only if w is a valid witness corresponding to x.

   To verify the GKR interactive proof the verifier needs to check the value of $\widetilde{W}$ at a single point $z \in \mathbb{F}^m$.
3. The prover will send an opening to the polynomial commitment at point z.

*Succinct Non-Interactive Arguments (*SNARG*s) for* NP

Finally, we note that since the GKR protocol is a public-coin protocol (i.e., a protocol where all the verifier's messages are random bits (corresponding to random field elements), we can eliminate interaction from these protocols by using the Fiat-Shamir paradigm! As a result we are able to convert any NP witness w into a succinct cryptographic witness π, where as opposed to w, a succinct witness π exists even for instances x that are not in the language, but are hard to find (under some hardness assumption on the Fiat-Shamir hash function and assuming the polynomial commitments are indeed binding).

Recall that this paradigm replaces the random messages of the verifier with the hash value of the transcript so far.

Secret Sharing

Intuitively, a secret sharing scheme allows a dealer to share a secret s among n parties $P_1, \ldots, P_n$ such that any *authorized* subset of parties can use all their shares to reconstruct the secret, while any other (non-authorized) subset learns nothing about the secret from their shares. Secret sharing has some direct applications, where we need to distribute a secret to several parties/servers, in order to distribute the required trust, as well as to allow reconstruction even if some of the parties fail.

Additionally, secret sharing is a useful tool in many larger cryptographic systems, notably, *secure multi-party computation* which will be the topic of the upcoming lectures.

In a secret sharing scheme we want to ensure that no information whatsoever is leaked to any unauthorized subset of parties. For example, simply giving some of the bits of the secret to each party certainly reveals information.

Secret sharing can be defined with respect to any access structure that specifies the set of authorized subsets, as long as that access structure is monotone (namely, if a subset is authorized, any larger subset should also be authorized). We will focus on a common access structure which is t-out-of-n or threshold secret sharing, where authorized subsets are all those of size at least t, while sets of size less than t are not authorize.

Definition

Definition 1. A t-out-of-n secret sharing scheme scheme over message space $\mathcal{M}$ consists of a pair of efficient algorithms $(\mathsf{Share}, \mathsf{Reconstruct})$ such that:

- Share is a randomized algorithm that takes as input a message $m \in \mathcal{M}$ and outputs a n-tuple of shares $(s_1, ..., s_n)$.

- Reconstruct is a deterministic algorithm that given a t-tuple of shares $\{(i, s_i)\}_{i \in I}$ for $|I| = t$, outputs a message $m \in \mathcal{M}$.

The following two properties are required to be satisfied.

Correctness: For every $m \in \mathcal{M}$ and every $I \subseteq \{1, \ldots, n\}$ of size t,

$$\Pr_{(s_1,\ldots,s_n) \leftarrow \mathsf{Share}(m)} [\mathsf{Reconstruct}(\{(i, s_i)\}_{i \in I}) = m] = 1$$

Security: For every $m, m' \in \mathcal{M}$ and for every $I \subseteq [n]$ such that $|I| < t$,

$$(s_i)_{i \in I} \equiv (s'_i)_{i \in I}$$

where $(s_i)_{i \in [n]} \leftarrow \mathsf{Share}(\mathsf{m})$ and $(s'_i)_{i \in [n]} \leftarrow \mathsf{Share}(\mathsf{m}')$.

n-out-of-n Secret Sharing Scheme

Suppose n parties wish to share a secret $m \in \{0,1\}^\ell$.

Share(m): Choose at random $s_1, \ldots, s_n \in \{0,1\}^\ell$ such that $\oplus_{i=1}^n s_i = m$, and output $(s_1, \ldots, s_n)$.

Reconstruct$(s_1, \ldots, s_n)$ outputs $\oplus_{i=1}^n s_i$.

Note that the correctness property follows from the construction, and the security property follows from the fact that an equivalent way to compute Share(m) is to choose at random $\{s_i\}_{i \neq j}$ and set $s_j = m \oplus (\oplus_{i \neq j} s_i)$, which implies that any set of shares that excludes at least one share, are randomly distributed, independently of m.

t-out-of-n Secret Sharing Scheme

Take 1: For simplicity, suppose that $t = 2$. Then to share a secret m, for every pair of distinct parties (i, j) in $\{1, \ldots, n\}$ generate 2-out-of-2 shares of m. Then give each party all the shares generated for that party. Note that each party has $n - 1$ shares.

Problem: If we use this approach to construct a t-out-of-n secret sharing scheme this will result with shares of size $\binom{n}{t-1}$.

Shamir's idea [1]: Use polynomials! Suppose we wish to share a message $m \in \{0,1\}$. If we want to share ℓ bits we will use the single-bit secret sharing scheme ℓ times, one for each bit. Choose a prime p such that $p > n$.

Share(m): Choose a random degree $t - 1$ polynomial $f : \mathsf{GF}[p] \to \mathsf{GF}[p]$ such that $f(0) = m$. This can be done by choosing $a_1, \ldots, a_{t-1}$

Recall that $\mathsf{GF}[p]$ denotes the field with elements $\{0, 1, \ldots, p-1\}$ where addition and multiplication are done modulo p.

at random in $\mathsf{GF}[p]$, setting $a_0 = m$ and letting

$$f = \sum_{i=0}^{t-1} a_i x^i.$$

For every $i \in [n]$ let $s_i = f(i)$ and output $(s_1, \ldots, s_n)$.

The shares can be the value of f on any n distinct points in the field $\mathsf{GF}[p]$ as long as none of these points is 0.

$\mathsf{Reconstruct}(\{(\alpha_j, s_{\alpha_j})\}_{j \in J})$: Solve t linear equations in t variables. The variables are $a_0, a_1, \ldots, a_{t-1}$ and each player P_{α_j} holds a linear questions:

$$\sum a_i (\alpha_j)^i = s_j$$

We note that these t linear equations are always independent (this is a Vandermonde matrix, which is known to be invertible).

We emphasize that these t players can jointly recover not only the secret m, but also the entire degree $t-1$ polynomial that the dealer chose, by computing:

$$f(x) = \sum_{i=1}^{t} f_i(x) \cdot s_i$$

where f_i is the unique degree $t-1$ polynomial that satisfies that $f_i(x) = 1$ if $x = \alpha_i$ and $f_i(x) = 0$ for every $x \in \{\alpha_j\}_{j \in [t] \setminus \{i\}}$. Namely,

$$f_i(x) = \prod_{j \in [t] \setminus \{i\}} \frac{\alpha_j - x}{\alpha_j - \alpha_i}$$

Connection to Reed-Solomon Codes

Shamir's secret-sharing scheme is very similar to the Reed-Solomon error-correcting codes (1960), which is a beautiful coding scheme! In a Reed-Solomon code a message $m = (m_0, m_1, \ldots, m_{t-1}) \in \{0,1\}^t$ is viewed as the unique degree $t-1$ polynomial f_m such that $f_m(i) = m_i$ for every $i \in \{0, 1, \ldots, t-1\}$. The codeword corresponding to the message m is the evaluation of the polynomial f corresponding to m on all the points in the field:

$$\mathsf{ECC}(m) = (f_m(0), f_m(1), \ldots, f_m(p-1)).$$

What we have seen above is that this codeword can be uniquely decoded even if all but t of the coordinates in the codeword were erased.

It is also known how to decode if less than $\frac{p-t+1}{2}$ of the coordinates were maliciously corrupted, and this is known to be optimal! Decoding from malicious corruptions is slightly trickier than decoding from erasure, but it is not too hard (and is explained beautifully

The main disadvantage of the Reed-Solomon ECC is that it is over a large alphabet. Often people want an ECC over the binary alphabet. For the application of cryptography and secret-sharing this is good enough.

in these Lecture notes by Anup Rao). This means that in Shamir's secret sharing scheme if a secret is shared among n parties and then the n parties jointly try to decode, even if some of the parties try to foil the outcome and give malicious shares, still the parties will be able to jointly recover the secret, as long as less than $\frac{n-t+1}{2}$ of the parties are corrupted.

Remark. In the next two classes we will see how to use Shamir's secret-sharing scheme to do *secure multi-party computation*, where a set of parties wish to jointly compute a function of their secret inputs (such as the average of their salaries) without revealing their secret inputs. Shamir's secret sharing scheme will be used as a key ingredient. Typically, in the setting of secure multi-party computation, two types of adversarial behaviors are considered: The first is honest-but-curious, where the parties are assumed to follow the protocol honestly but they are curious and are trying to learn anything they can about the secret inputs of the other players (by possibly colluding and sharing information). The second is adversarial where a malicious party can deviate from the protocol in arbitrary ways.

We will see a protocol for the honest-but-curious setting. For that setting Shamir's secret sharing scheme is used and security with erasures is all we need. For the malicious case, which we will not cover in this class, we use the fact that Shamir's secret sharing scheme has the stronger property that one can decode even if some of the shares are faulty.

References

[1] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.