Succinct Non-Interactive Arguments (SNARGs) for NP

Notes by Yael Kalai

MIT - 6.5610
Lecture 13 (March 18, 2024)

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

Outline

- The Fiat-Shamir Paradigm
- SNARGs for low-depth computations
- Overcoming the low-depth restriction
- Succinct hash with local opening

Last class we presented the GKR protocol which is a doubly efficient interactive proof for bounded depth computations. Recall that the GKR protocol consists of a sequence of (pairs of) Sumcheck protocols, which are public coin protocols, which means that the verifier simply sends truly random bits in each round.

Next, we will see how to eliminate interaction from the GKR protocol. More specifically, we will show a general technique for eliminating interaction from any public-coin protocol.

Eliminating interaction via the Fiat-Shamir Paradigm

In 1986, Fiat and Shamir [1] introduced a paradigm for eliminating interaction from public coin protocols. Specifically, they showed how to take any interactive identification scheme (where a prover proves via a public-coin interactive proof that he knows the secret key corresponding to a given public key) and turn it into a (non-interactive) digital signature scheme.

We will learn about digital signature schemes next class, stay tuned!

This transformation is known as the Fiat-Shamir (FS) paradigm, and it turns out that this paradigm is general enough to eliminate interaction from any *public-coin* interactive protocol. Loosely speaking, this is done by replacing the messages of the verifier with the hash of the transcript thus far. In other words, in the transformed non-interactive protocol, the prover generates each random message that

was supposed to be sent from the verifier by hashing the transcript of the protocol so far.

Let us demonstrate the Fiat-Shamir (FS) paradigm for the special case of interactive proofs that consist of only three messages, which we denote by (α, β, γ), where β is the random message sent by the verifier, and α and γ are the first and third messages that are sent by the prover.

The FS paradigm proposes to use a hash function H to eliminate interaction from this protocol by instructing the prover to generate the entire transcript (α, β, γ) on its own, by setting $\beta = H(\alpha, x)$ where x the instance. More generally, suppose the transcript is of the form

$$(\alpha_1, \beta_1, \ldots, \alpha_\ell, \beta_\ell, \alpha_{\ell+1})$$

where the messages α_i are chosen by the prover and the messages β_i are random messages chosen by the verifier. The FS paradigm eliminates interaction from this protocol by having the prover generate this entire transcript on its own subject to

$$\beta_i = H(\alpha_1, \beta_1, \ldots, \alpha_i). \tag{1}$$

The verifier will then check that the given transcript is accepting and that Equation (1) holds for every verifier's message.

Security of the Fiat-Shamir Paradigm

This paradigm is extremely simple and elegant. It is also prevalent in practice. But the question is: *is it secure?* Namely, if we start with a sound interactive proof does the non-interactive proof obtained by applying the FS paradigm still sound? Clearly there are hash functions for which the FS paradigm is insecure! the question is: *How do we choose a hash function H for which the FS paradigm is secure, and does one even exists?* In practice, off the shelve hash functions are used, such as SHA-256 or SHA-3.

The Insecurity of the FS Paradigm. We first note that the FS paradigm is not always secure. For example, it is not secure (no matter which hash function H is used) if applied to the 3-message zero-knowledge (ZK) protocol for 3COL. Recall that in this protocol the prover commits to a (randomized) coloring of the graph, the verifier chooses a random edge and then prover opens the commitment on this edge. This is repeated sequentially many times to reduce the soundness error.

The FS paradigm completely fails when applied to this protocol. This is due to a "rejection sampling attack" which is basically the

"attack" used to argue that this protocol is ZK. Namely, the cheating prover will do the following:

1. Guess the edge (u, v) that will be outputted by H.
2. Generate a commitment to an arbitrary coloring of the graph, such that the nodes u and v are colored with distinct (and legal) colors. Denote this commitment message by α.
3. Compute $\beta = H(x, \alpha)$, where x is the graph.
4. If $\beta = (u, v)$ then the prover produces a valid opening of the nodes u and v, and wins!
5. If $\beta \neq (u, v)$ go back to (1) and try again.

The hiding property of the commitment scheme implies that on expectation after about $|E|$ tries (where E is the number of edges) the cheating prover will succeed. Once he succeeds he will move to simulating the next sequential execution in a similar way.

The reason this protocol renders the FS paradigm to be insecure is because the 3-message version does not have negligible soundness, which allows for a rejection sampling attack. This begs the following question: *Is the FS-paradigm sound when applied to constant round protocols with negligible soundness?*

The Security of the FS Paradigm It turns out that the FS paradigm is sound in the Random Oracle Model (ROM) when applied to any constant round proof with negligible soundness. In the ROM we model H as a *truly random* function that all parties (including the adversary) have black-box access to. In this model, Pointcheval and Stern [2] proved that the FS paradigm is secure when applied to any constant round interactive proof with negligible soundness. Intuitively, security follows since in the ROM, there is no difference if the prover interacts with a random function or a verifier that produces random messages. The only thing a cheating prover can do with a hash function is "reject sample," but this is not helpful in the constant round setting with negligible soundness.

In particular, if we run the 3COL ZK protocol *in parallel* many times, and apply the FS paradigm to this parallel repeated protocol, then the new non-interactive protocol is sound in the ROM.

Security of the FS paradigm when applied to the GKR protocol

The GKR protocol is not constant round! So why should the FS paradigm be secure when applied to it? The reason is that it has a special soundness guarantee which is called *round-by-round soundness*, which means that if you start with a false claim, after every round you reduce the claim to another false claim with overwhelming probability (assuming the field $\mathbb{F}$ is of super-polynomial size or if the

GKR protocol is repeated in parallel). Indeed the FS-GKR protocol, which is the FS paradigm applied to the GKR protocol is known to be secure in the ROM. Moreover, recently it was proven to be secure under the LWE assumption for a specific choice of H (which is not efficient enough for practice). When people implement the FS-GKR protocol they still do so with off-the-shelve hash functions which is guaranteed to be sound only in the ROM, but it is good to know that if this scheme will be broken it is not due to a flaw in the paradigm but rather due to the bad choice of the hash function.

Succinct Non-Interactive Arguments (SNARGs)

We emphasize that the FS-GKR protocol no longer has statistical soundness. Namely, an all powerful cheating prover can cheat and convince the verifier to accept a false statement. Intuitively, the reason is that the GKR interactive protocol does not have soundness 0, rather there is a small (negligible) probability that the verifier accepts a false claim. So, a cheating prover can "interact" with the hash function exponentially many times until they are successful.

Therefore FS-GKR is not a proof, since a proof is assumed to have statistical soundness. Protocols that only have computational soundness (i.e., soundness holds only against computationally bounded cheating provers) is called an *argument*. With this terminology, note that the FS-GKR protocol is a succinct non-interactive argument (SNARG) for bounded depth computations (assuming all parties agree on a hash function).

Overcoming the Low-Depth Restriction

We can use the GKR blueprint to construct SNARGs where the proof size and the runtime of the verifier do not grow linearly the depth, rather they grow only poly-logarithmically with the depth and size. The basic idea to get around the depth restriction is to "flatten" the circuit. Namely, suppose the prover wishes to prove to the verifier that $C(x) = y$ where C is of depth D and size S (think of D as being large and the verifier cannot run in time D). 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 low-degree extension of the input to C'.

Cryptography to the rescue

To overcome this problem we have the prover compute the low-degree extension of the input to C', which is the values of all the wires of C. Denote these values by V_0 and denote by $\tilde{V}_0$ its low degree extension. The flattend GKR protocol proceeds as 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 low-degree extension of V_0.
   (c) Send to the verifier a hash of $\tilde{V}_0$.
2. The prover and verifier run the interactive GKR protocol (or the non-interactive FS 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 the hash of $\tilde{V}_0$ at the point $\tilde{z}$.

The question is how can the hash of $\tilde{V}_0$ be opened *succinctly*? Note that we need a collision resistant hash function $H : \{0,1\}^N \to \{0,1\}^\lambda$ that can be opened to a single location succinctly, where the opening should not grow with N. We can do this using Merkle Hash.

Merkle Hash

Given any collision resistant hash function $h : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$ one can construct a collision resistant hash function $H : \{0,1\}^N \to \{0,1\}^\lambda$, where N which may be very large, such that one can "open" to any bit of the preimage using an opening of size $O(\lambda \cdot \log N)$. H is constructed from h via a tree construction, as follows:

$$H(b_1, \ldots, b_N) = h(H(b_1, \ldots, b_{N/2}), H(b_{N/2+1}, \ldots, b_N))$$

Assume without loss of generality that N is a power of two (i.e., is of the form 2^k for some $k \in \mathbb{N}$. This can be always be achieved by padding.

The output of H needs to include the depth of the tree in order to be collision resistant.

Remark. We note that while FS GKR was proven to be secure under LWE (for a specific choice of a hash function for FS), such a result is not known for squashed GKR. Squashed GKR is only known to be secure in the ROM (with appropriate choices of parameters).

Remark. Computing a Merkle hash takes time which is at least linear in the input. Thus, computing the Merkle hash of $\tilde{V}_0$ takes time at least $|\mathbb{F}|^m$, which limits the size of $\mathbb{F}$ to be relatively small. In practice people use a primitive called *polynomial commitments* that allow one to commit to a low-degree polynomial in time that is sublinear in the field size and still open locally.

SNARGs for NP

Finally, note that the above protocol gives a SNARG for NP. The SNARG consists of:

1. A Merkle-Hash (or a polynomial commitment) to the low-degree extension of the witness w. Denote the low-degree extension of w by $\widetilde{W} : \mathbb{F}^m \to \mathbb{F}$.
2. A FS GKR proof that the verification circuit corresponding to the instance x is accepting on input w.

   Recall that to verify this proof the verifier needs to check the value of $\widetilde{W}$ at a single point $z \in \mathbb{F}^m$.
3. An opening of the Merkle-hash (or the polynomial commitment) at point z.

References

[1] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

[2] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.