Arithmetization - what Algebra’s got to do with it ?

This article is based on a talk by Professor Eli Ben-Sasson, the co-founder and president of StarkWare, at the 13th Bar Ilan (BIU) Winter School on Cryptography. The BIU Winter School focused on recent advances in cryptography and blockchain technology. The lecture given by Professor Eli Ben-Sasson can be found [here](https://www.youtube.com/watch?v=fxxu8JyHYco).

This article aims to get an intuition on how polynomials lead to succinct verification. To convert a computational problem into polynomials, we use arithmetization (covered in the first part of this article) and polynomial commitment schemes to commit to polynomials with a certain cap on the degree of the polynomial (committed using PCS, which are discussed in the later part of this article).

Part I. Arithmetization

Arithmetization is a process of conversion of some computational problem into polynomial form. To gain a more intuitive understanding of arithmetization, we will review some toy problems that need solving. However, there are some terms whose definitions we can quickly revise.

**Polynomial -** a polynomial is a mathematical expression consisting of variables, coefficients, and arithmetic operations such as addition, subtraction, and multiplication. It comprises one or more terms, each with a variable raised to a non-negative integer exponent.

**Degree of a polynomial:** The degree of a polynomial is the highest exponent of the variable in any of its terms. It provides information about the complexity and behavior of the polynomial. For example, a polynomial of degree 2 is quadratic, while a polynomial of degree 3 is cubic.

**Field:** In mathematics, a field is a set of elements with two binary operations, addition and multiplication. It satisfies certain axioms, such as closure, associativity, commutativity, and the existence of inverses and identity elements. Examples of fields include real numbers, rational numbers, and complex numbers.

**Multiplicative group:** When we refer to “multiplicative group” below, we talk of “multiplicative group for roots of unity.” The multiplicative group for roots of unity is a mathematical structure that consists of a set of complex numbers that are solutions to the equation z^n = 1, where n is a positive integer.

Arithmetization Toy Problem

Let’s assume we have a field F and a set H (a subset of F). Alice, the verifier, provides any field *F* and a degree bound *d*. Bob, the prover, can send over the coefficients for some polynomial of degree *d-1* (a total of d coefficients) to another party, Tom. Tom is a trustworthy person, and Alice can query Tom for the value of the polynomial that Bob suggested at any point in the domain of the polynomial. Bob cannot directly transfer the coefficients of the polynomial to Alice (and neither can Tom)

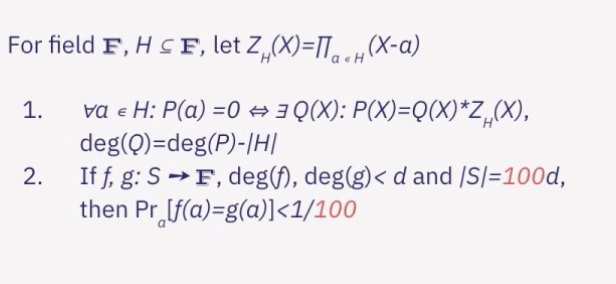
Now we need to revise some of the basic principles of algebra.

1. For a set H (a subset of field F), the vanishing polynomial ZH is defined as the polynomial that vanishes (or equates to zero) at every point in H. Mathematically, it can be expressed as:

ZH(x) = (x - h1)(x - h2)(x - h3)...(x - hn)

where h1, h2, h3, ..., hn are the elements of the set H.

1. If a polynomial P can be factored into two polynomials Q and Z then the degree of P is equal to the sum of degrees of Q and Z.
2. Suppose we have two polynomials ***f*** and ***g***, both bound by degree ***d*** and a sampling set ***S*** of size 100***d***. Now, if we sample a random point ***a*** from S, the probability of ***f(a)*** being equal to ***g(a)*** is 1/100 (Note: 100 is any arbitrary large number and can be replaced)

In mathematical terms the above facts can be written as:   


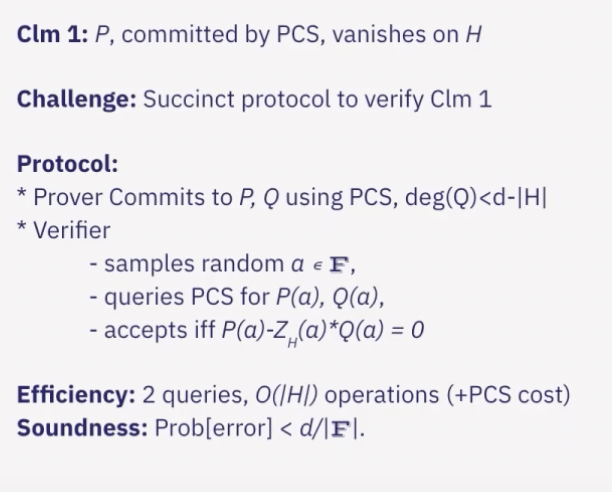
Assuming we have the ideal polynomial commitment scheme that Tom offers, let’s start with the problem. Bob, the prover, wants to prove to Alice that some polynomial ***P*** vanishes at a set ***H*** (is equal to 0 at each point in H). Assuming that H is very large (e.g., to a degree of 10 million), Alice does not want to query Tom for the value of ***P(a)*** at all points ***a*** in the set ***H*** - this would be a brute force method for solving this problem.

Hence, we need to derive such a verification protocol that:

* is succinct and does not require huge computational resources to verify the statement (hence brute force does not work)
* would reject Bob’s statement with very high probability if there is even one element a ∈ H s.t. ***P***(***a***)≠0.

To solve this problem, Bob needs to commit to Tom (share the coefficients of the polynomial) another polynomial ***Q*** s.t. ***P***(***X***) = ***Q***(***X***) \* ***ZH***(***X***), where ***ZH***(***X)*** is the vanishing polynomial for the set H.

The verifier, Alice, can now query any random element ***a*** from the field ***F***, and ask Tom for values of ***P(a)*** and ***Q(a)***. With these two values in hand, Alice can confirm if ***P(a)*** - ***Q(a)***\****ZH(a)*** = 0



The “magic” of this protocol lies in the fact that the probability of Alice not rejecting the claim even if Bob’s statement was false (probability of error in simpler terms) is extremely low and equal to ***d***/***|F|***, where |F| is the size of field F. Since the size of the field is generally magnitudes larger than the degree of polynomial the probability of error is drastically low.

However, there is still one shortcoming of this method. To verify the claim, Alice has to do ***O(d)*** operations - these come from calculating ZH(a) - and since d is to the magnitude of millions (or billions), this is not exactly a **succinct** solution.

To make the above proof and verification succinct, we may use the fact that if the set H is a multiplicative group (see Definitions above), ZH(X) = X|H|-1.

*Problem 2.*

Let us now take a look at a different problem in a similar context. Let us assume that Bob needs to prove that a polynomial P of degree bound d only attains values of {0,1} on all the elements of some set H.

In order to generate a proof for this, the following property is used that ***Y***\*(1-***Y***) is zero when ***Y*** = 0 or ***Y*** = 1. Similar to Problem 1, Bob will have to commit to a polynomial Q of degree bound 2***d***-|***H***| (s.t. ***Q*** \* ***ZH*** gives ***P***). Alice can verify if Bob’s statement is true or not by randomly sampling an element ***a*** from the field ***F***, and verifying if .

***P***(***a***)\*(1-***P***(***a***))-***ZH***(***a***)\****Q***(***a***) = 0

Assuming that the set H is a multiplicative group the verification cost for Alice is an O(log|H|) operation and she has to make two interactions with Tom (termed as PCS cost). The probability of erroneously accepting a false claim is 2***d***/|***F***|, which is generally a very small number.

Proof of the above. In order to understand it better, let us prove why the above equation works by contradiction. Assume that ***P*** is not {0,1} valued at all elements of set ***H***. In that case there is at least some element ***a*** in ***H*** such that ***P***(***a***)\*(1-***P***(***a***)) ≠0. In that case, ***P***(***a***)\*(1-***P***(***a***)) does not vanish on H and the above equation would be a non-zero polynomial. Since the polynomial is non-zero and is of degree 2***d***, there is a chance of 2***d***/|***F***| of Alice accidentally choosing such an element in F where the above polynomial has a root (it has 2d roots because degree is 2***d***).

Problem 3.

The next and last problem of the Arithmetization chapter that we will look at is that of consecutive computations. Very often in the blockchain world, we are interested in proving transactions that occurred one after the other. This means we must prove that the computations were done honestly and accurately and that all were noticed. The following is an example of such a problem statement.

Let us assume that Bob has a list of bits {a1, a2, … a|H|-1} and a sequence {b0, … b|H|-1} s.t. 1. b0 = b1 = 1

2. b|H|-1 = 42 mod p

3. bi = b3i-2+ai\*bi-1

How can Bob succinctly prove to Alice that the computations were done correctly ? The solution to this problem is not included in this piece but this should be attempted as a practice problem.

FRI and Polynomial Commitment Schemes (PCS)

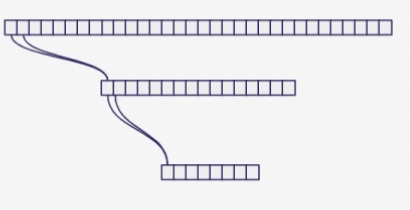
In all of the above problems we had an entity, Tom, that was acting as a “bridge” for Alice’s interaction with Bob. The role of this “bridge” in real-life implementations of cryptographic proofs is carried by *polynomial commitment schemes* (such as FRI, KZG etc.). Tom was acting as a “perfect” polynomial commitment scheme because it had the following properties:

* **Soundness**: Soundness refers to the property that a cryptographic proof system rejects invalid or false statements with high probability
* **Computational Hiding**: Computational hiding ensures that an attacker cannot gain any information about the polynomial being committed to, based solely on the commitment value
* **Efficient** Polynomial **Evaluation**: The scheme should allow for efficient evaluation of the committed polynomial at any given point without revealing the entire polynomial or requiring costly computations
* **Non-Interactive**: Ideally, the commitment scheme should be non-interactive, meaning that it does not require multiple rounds of interaction between the committer and verifier. Note that some interactive schemes can be converted to non-interactive by using Fiat-Shamir transform.
* **Succinctness**: The commitment scheme should generate commitment values that are relatively short, regardless of the size of the polynomial being committed to
* **Unpredictability**: The commitment values should appear random and unpredictable, even if the committed polynomials follow some specific patterns or structures. Since Tom did not reveal any details about the polynomial itself, unpredictability came by default.

A common PCS used in proof is a scheme built on top of the FRI protocol. FRI (Fast Reed Solomon IOP of Proximity) is a protocol that involves a pair of interactive algorithms. The protocol is designed to verify whether a certain polynomial is bound by a degree ***d***.

The protocol works in rounds of decreasing complexity such that the prover sends a message in linear time at each round, and the verifier replies with a random seed in logarithmic time. The verifier can query the prover’s message at random locations, ensuring that the prover cannot cheat.

The protocol works because, in each round, the prover commits to a polynomial of a certain degree d. The verifier provides some random seed, and the prover commits to a new polynomial of half the degree based on what the verifier provided (hence cheating becomes difficult for the prover). Iteratively going through the protocol, the probability of error decreases. On top of such a scheme, a PCS can be built and is commonly used today.



In each round of the FRI protocol, the degree of the polynomial to which the prover commits is halved.

The problem with commitment schemes today is that there is no ideal one. Some PCSs may have large proofs but logarithmic verification (FRI), while others may have smaller proofs but linear verification time. As more time passes, the most optimal methods will emerge, and more projects will start moving to those protocols.

Now that we have a better intuition of arithmetization and polynomial commitment schemes, learning more about SNARKs should become easier and more exciting.