Polynomial Commitments

A learning group for ZK and SNARK application development

Daniel Szego
In: https://www.linkedin.com/in/daniel-szego/

Logistics: ZK Learning Group

Every month, third thursday in 2025, from 18 (CET)

One hour, presentation + short discussion

Different topics on zero knowledge proof,

- mostly from programmer and application developers perspective
- with some theory

Coordination:

- Discord channel: LF Decentralized Trust
  https://discord.com/channels/905194001349627914/1329201532628898036
- Meetup.com: https://www.meetup.com/lfdt-hungary/events/305634614/
- Repo with all the contents:https://github.com/LF-Decentralized-Trust-labs/

https://github.com/Daniel-Szego/zk-leraning-group

Quizzes and small programming challenges, LFDT merchs at the end

Logistics: Hunting for the SNARK

February - Introduction, Theory : Definitions and building blocks

March - Theory : Polynomial commitments

April - Theory : Interactive oracle proofs

May - Programming : Circom

June - Programming : Circom

July - Programming : Noir

August - Programming : Noir

September : Applications : Off-chain transaction

October : Applications : Proving solvency

November : Applications : Rollup

December : Wrap up, Applications

Subject to change based on community discussion ….

Agenda

- *zkSNARK*
- *Development flow*
- *Elements of building a SNARK*
- Commitments
- Functional commitments
- Types of functional commitment
- KZG10
- Protocols with functional commitment
- Fiat-Shamir transformation
- Quiz, literature, links and Q and A

(zk)SNARK - Succinct Non-interactive ARgument of Knowledge

Computation: arithmetic circuit : *C(x, w) -> F*
- *x* public input
- *w* private input, witness

Prover algorithm: *P (pp, x, w) -> proof*

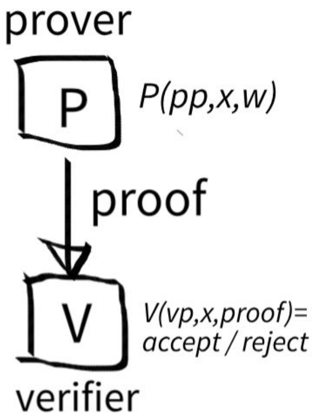
Verifier algorithm: *V (vp, x, proof) -> accept / reject*

Succinct*: small proof in size, easy / fast to verify*

Complete*:* if there is a proof, it is accepted by the prover by very high probability

Knowledge sound*:* If the verifier accepts, then the prover “knows” a w witness (secret input), that satisfies the computation. (malicious prover can not convince the verifier)

Zero knowledge *(optional):* the proof reveals nothing from the witness (secret input)



Development flow

High level language:

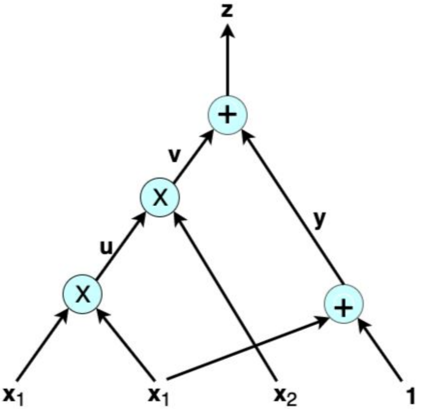
- Likie Cirom, Noir

Low level abstraction:

- Arithmetic circuit
- DAG: directed acyclic graphs
- Inputs, gates, constants, connections (wires)
- Nodes are labelled with “+” addition and “x” multiplication

Polynomials:

- R1CS, rank one constraint
- $a \times b = g$
- *a*,*b* and *g* are affine combination of variables
- examples: $w_1 \times (w_2 - w_3 + 1) = x_1$

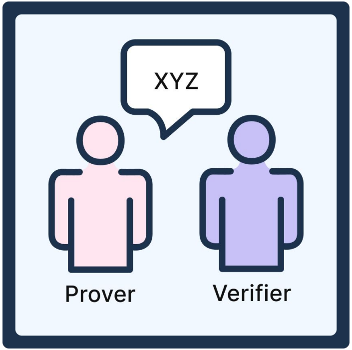


Elements of building a SNARK

Polynomial commitment scheme: get succinct interactive argument: Bind and commit a polynomial.

Interactive Oracle Proof (IOP): R1CS or circuit satisfiability. Evaluating (committed) polynomials in one or multiply points in multiply rounds, extend polynomial commitment to general circuit satisfiability.

Fiat-Shamir: transforming a interactive proofs to non-interactive:



Commitments

Commit to a chosen value (or chosen statement), keeping it hidden to others, reveal the committed value later:

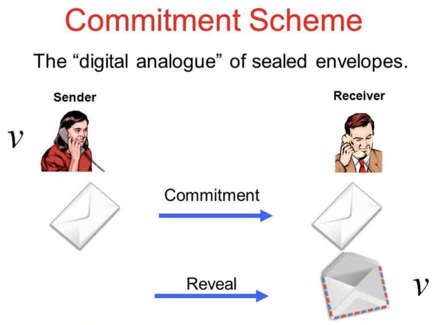
1. the ***commit*** phase during which a value is chosen and committed to
2. the reveal phase during which the value is revealed by the sender, then the receiver ***verifies*** its authenticity

Properties:

- ***hiding***: the committed information can not be revealed
- ***binding***: the prover can not fake the committed message

Example: commitment with hash function (hash commitment)

- commit: commitment=sha(message)
- verify: send message, verify if commitment=sha(message)

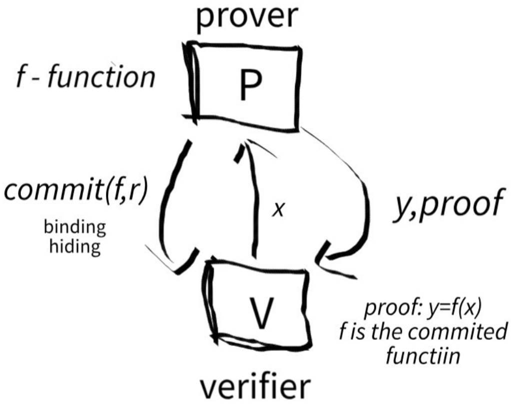


Link: https://en.wikipedia.org/wiki/Commitment_scheme

Functional commitment

Committing a function and opening at in a point:

- Having a family of functions *F*.
- Prover: choose an *f* function from the *F* function family
- Committing *f* function to the verifier, *com* commitment
- Verifier sends an *x* point of the function.
- Verifier sends *y* and proof that *y=f(x)* and that *f* is in the function family and that f is the committed function



Link: https://geometry.xyz/notebook/functional-commitments-zk-under-a-different-lens

Types of functional commitment

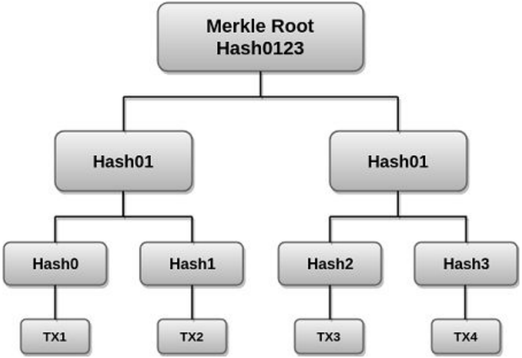
Different types of functional commitments:

- Polynomial commitment: univariate polynomial (at most degree *d*).
  - Bilinear groups: KZG 10
  - Hash functions: FRI
  - Elliptic curves: Bulletproof

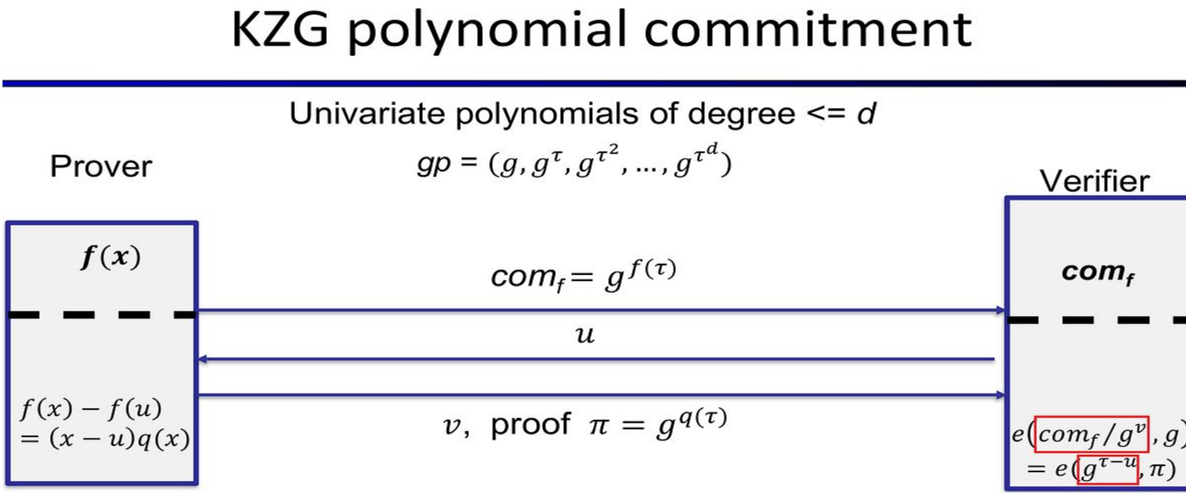
- Multilinear commitment: several variable but the degree is at most one.

- Vector commitments: committing a vector and opening it at a certain point / index. E.g. merkle tree.

- Inner product commitment



KZG10 system (Kate-Zaverucha-Goldberg)



Protocols with functional commitment

Testing if a ***polynomial is zero***:

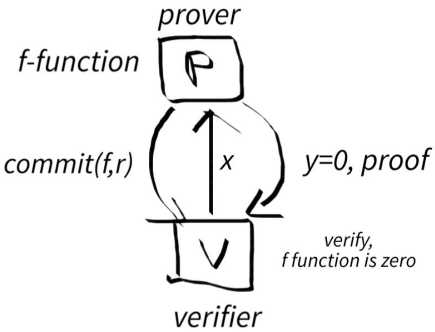
- If a truly randomly chosen point evaluates to 0, the whole polynomial is zero with a very high probability.
- *d/p* : *d* degree, *d* roots, *p* is the domain

Testing if two ***polynomials are equal***:

- if two polynomials *f* and *g* are equal in a randomly chosen *r* point *f(r)=g(r)*, then the the two polynomials are equal with a very high probability.
- *f(r)-g(r)* zero test

Sum check: sum of certain input values of a committed polynomial is a given value.

Prod check: product of certain input values of a committed polynomial is a given value.



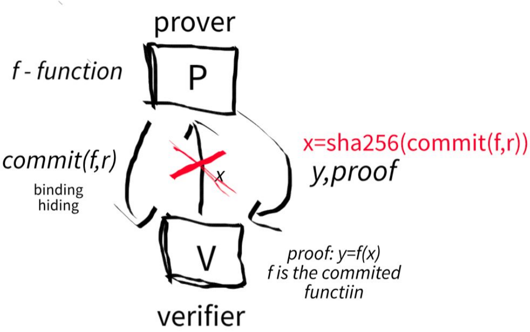
Fiat-Shamir transformation

Transforming interactive certain, random challenge, proofs to non-interactive. (***flip coin protocols***)

Proof schema:

- The prover first generates some random value, the commitment, and sends it to the verifier.
- The verifier responds with a challenge value generated uniformly at random.
- The prover computes the final proof based on both the commitment and challenge.

The prover can compute this random value themselves by using a random function, such as a cryptographic hash function.



Link: https://www.zkdocs.com/docs/zkdocs/protocol-primitives/fiat-shamir/

Challenge

Quiz:

Will be posted in the discord channel:
https://discord.com/channels/905194001349627914/1329201532628898036

Links, Resources, Literature

Commitment schemes:
https://medium.com/@icostan/commitment-schemes-8b523d48aa1e

Functional Commitments: ZK under a different lens
https://geometry.xyz/notebook/functional-commitments-zk-under-a-different-lens

Explaining KZG Commitment with Code Walkthrough
https://kaijuneer.medium.com/explaining-kzg-commitment-with-code-walkthrough-216638a620c9

Exploring Elliptic Curve Pairings
https://medium.com/@VitalikButerin/exploring-elliptic-curve-pairings-c73c1864e627

Fiat-Shamir transformation
https://www.zkdocs.com/docs/zkdocs/protocol-primitives/fiat-shamir/

Happy Hunting for the SNARK :)

Q & A

Daniel Szego
In: https://www.linkedin.com/in/daniel-szego/

Merkle tree - naive polynomial commitment