Interactive Oracle Proofs

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

- *(zk)SNARK*
- *Elements of building a SNARK*
- *Interactive Oracle Proof (IOP)*
- *Building blocks*
- *PLONK*
- *Computational trace*
- *Constructing polynomials*
- *Summary*
- *Literature and Links*
- *Q&A and discussion*

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

Computation: arithmetic circuit : *C(x, w) -> F*

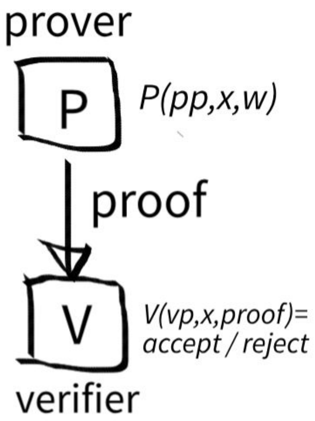
- *x* public input
- *w* private input, witness
- high level computation
- arithmetic circuit
- polynomials

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

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

Properties:

- *Succinct:*
- *Complete:*
- *Knowledge sound:*
- *Zero knowledge*



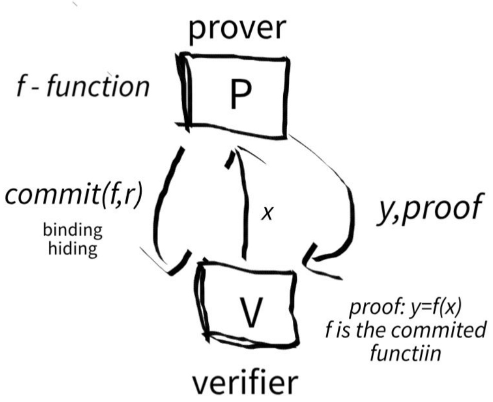
Elements of building a SNARK

Elements of a SNARK:

- *Polynomial commitment*
- *Interactive Oracle Proof*
- *Fiat-Shamir*

Functional commitment:

- 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



Interactive Oracle Proof (IOP)

Proving that $C(x, w) = 0$

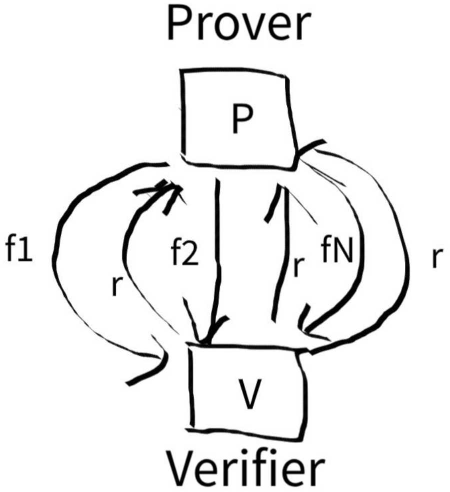
Committing $f_0, f_1, \ldots f_N$ functions

Committed functions can be evaluated at chosen points with the help of functional commitments

Repeated steps, n times:
- Prover sends committed functions
- Verifier responses with random values

Final step: verification:
- functions / function commitments
- public input
- opening functions at certain points

Properties:
- complete
- knowledge sound
- zero knowledge



Building blocks

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

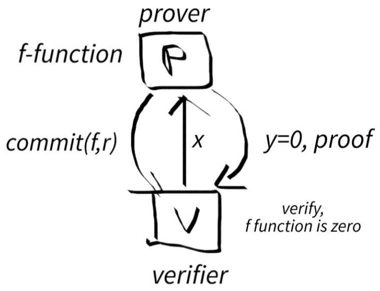
Testing if two ***polynomials are equal***:

- 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

Testing if a polynomial is ***zero on a set:*** quotation polynomial

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.



PLONK - constructing computational trace

Constructing a polynomial IOP

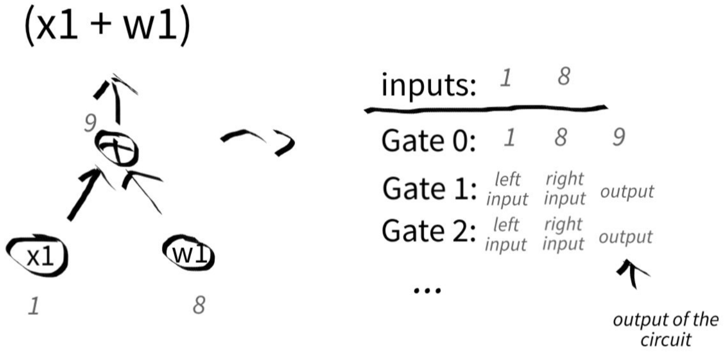
for a *C(x,w) = 0* circuit

Computational trace
- table
- first row is the input
- for each gate, there is a left input, right input and output

Using computational trace instead of arithmetic circuit

Proving if the computational trace is correct

Proving if the output is zero


(x1 + w1)
9
+
x1
w1
1
8
inputs: 1 8
Gate 0: 1 8 9
Gate 1: left input right input output
Gate 2: left input right input output
...
output of the circuit

PLONK - constructing polynomial

Computational trace to polynomial:

Constraints to the polynomial

Encoding all inputs:

- *P(w to the -i)* = *i* for all *i* input

Encoding all wires:

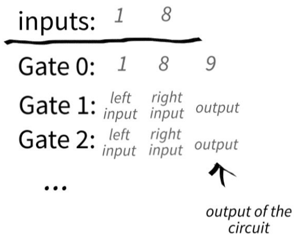
- *P(w to the 3l)* = left input of the *l* gate
- *P(w to the 3l+ 1)* = right input of the *l* gate
- *P(w to the 3l+ 2)* = output of the *l* gate

Prover can use the Fast Fourier Transformation to construct coefficients of the polynomial

$P(w^{-1})=1, \quad P(w^{-2})=8$

$P(w^{0})=1, \quad P(w^{1})=8, \quad P(w^{2})=9$

$P(w^{3})=\text{left}, \quad P(w^{4})=\text{right}, \quad P(w^{5})=\text{output}$


inputs: 1 8
Gate 0: 1 8 9
Gate 1: left input right input output
Gate 2: left input right input output
...
output of the circuit

PLONK - Proving validity

Proving validity of the created polynomial:

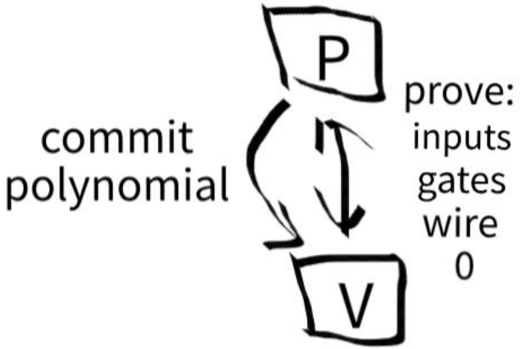
- All ***inputs*** are encoded correctly by the polynomial

- ***Gates*** are evaluated correctly

- ***Wiring*** is implemented correctly

- **Output** of the last gate is ***0***

Proving that the last gate is evaluated to 0, opening the polynomial at the *P(w to the 3l+ 2)* = and testing if that is rero



PLONK - Proving validity of gates and inputs

Proving correct input:

Additional *V* polynomial for encoding all inputs

- *V(w to the -i)* = *i* for all *i* input

Prove that the *V* input polynomial and the *P* committed polynomial agree on all input points: Zero on a set test *P-V*

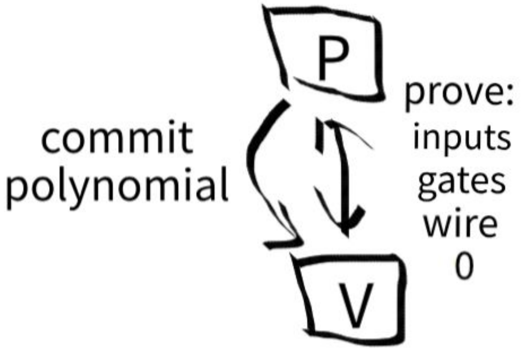
Proving correctness of gates:

Selector polynomial *S*:

- *S(w to the 3l)* = *1* if it is an addition gate
- *S(w to the 3l)* = *0* if it is a multiplication gate

Combining with the *P* polynomial and using the zero set test

Independent from the input values



PLONK - Proving validity of wiring

Proving correct wiring:

Equalities for the wiring constraint

Proving wiring constraints

W wiring polynomial:

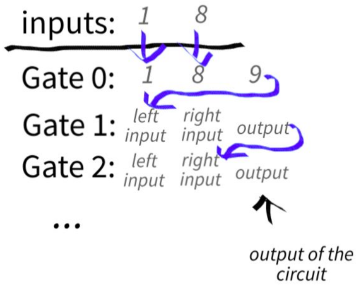
Rotation of coefficients

Independent from the input values

Can be computed at the setup

$P(w^{-1})=P(w^{0})$ $P(w^{-2})=P(w^{1})$

$P(w^{3})=\ldots$


inputs: 1 8
Gate 0: 1 8 9
Gate 1: left input right input output
Gate 2: left input right input output
...
output of the circuit

PLONK - Summary

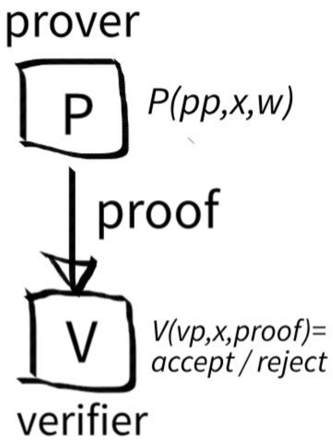
Setup / preprocessing:

- compute wiring polynomial
- compute selector polynomial

Prover:

- build P polynomial for the computation trace and commits
- prove inputs
- prove gates
- prove wiring
- prove output is zero

Using the Fiat Shamir transformation to make it non-interactive



Challenge

Quiz:

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

Links, Resources, Literature

A guide to Zero Knowledge Proofs (Part 2)
https://medium.com/@Luca_Franceschini/a-guide-to-zero-knowledge-proofs-part-2-7904dee9758d

Interactive Oracle Proofs
https://www.iacr.org/archive/tcc2016b/99850156/99850156.pdf

What is PLONK
https://medium.com/@Luca_Franceschini/what-is-plonk-29c56f326cf6

Plonk Interactive Oracle Proofs (IOP)
https://hackmd.io/@0xsachink/ByuqZfD63

Proofs, Arguments, and Zero-Knowledge, Chapter 4
https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.pdf

Happy Hunting for the SNARK :)

Q & A

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