ZERO KNOWLEDGE PROOFS

by Marija Mikić

ZKP Course

Class 3: ZK SNARKs & ZK STARKs

What is a SNARK?

SNARK - Succinct Non-Interactive Argument of Knowledge;

SNARK: the proof is “short” and fast to “verify”;

You can find more information on these links:

Link 1 >
Link 2 >

What is a SNARK?

SNARK - Succinct Non-Interactive Argument of Knowledge;

SNARK: the proof is “short” and fast to “verify”;

Example: I know an *w* such that SHA256(*w*)=*y*, where *w* is 1 GB.

Proof: few kB; Time to verify: few ms;

Proof size: $O(\log|C|, \lambda)$ Time to verify: $O(|y|, \log|C|, \lambda)$

There’s More 👉

What is a SNARK?

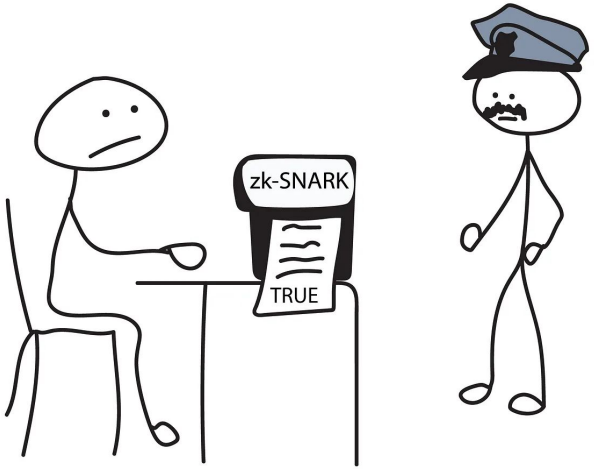
SNARK - Succinct Non-Interactive Argument of Knowledge;

SNARK: the proof is “short” and fast to “verify”;

Example: I know an *w* such that SHA256(*w*)=*y*, where *w* is 1 GB.

Proof: few kB; Time to verify: few ms;

ZK SNARK = Zero Knowledge + SNARK;



There’s More 👉

What is a STARK?

STARK - Scalable Transparent Arguments of Knowledge.

STARK: no trusted setup;

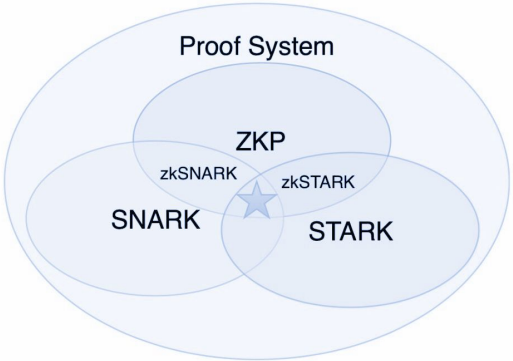
There's More 👉

What is a STARK?

STARK - Scalable Transparent Arguments of Knowledge.

STARK: no trusted setup;

ZK SNARK = Zero Knowledge + SNARK;

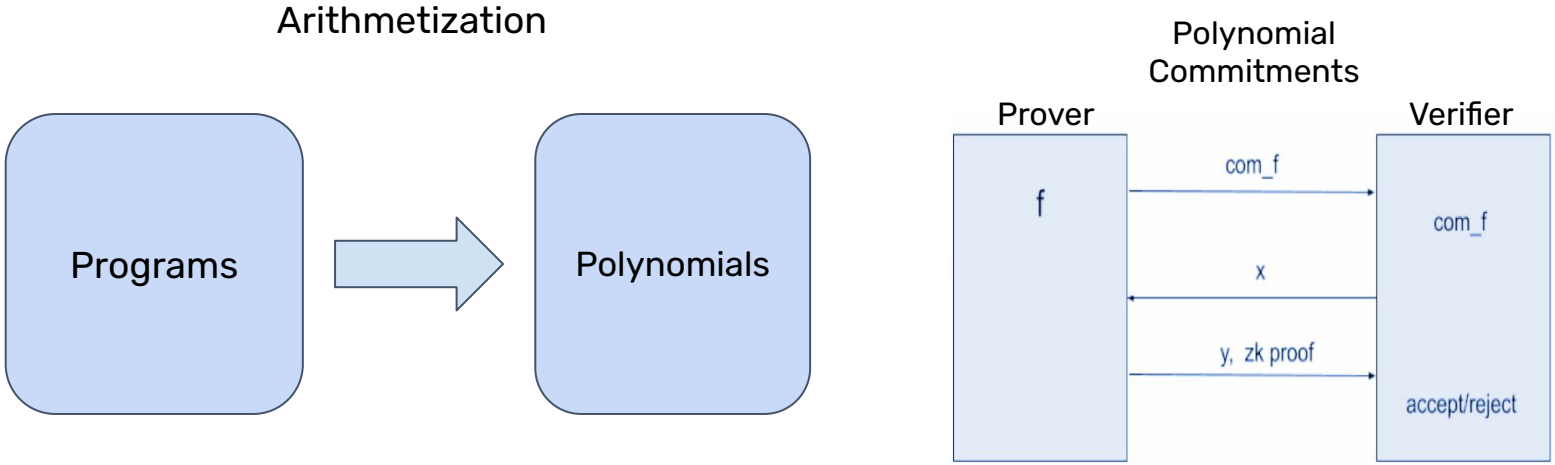


	Proof size (bytes)	Verification time (ms)
Groth 16	200	3
PLONK	400	6
STARK	80*1024	10

You can find more information on this link:

Link 1 >

Building zk SNARKs & zk STARKs



There's More 👉

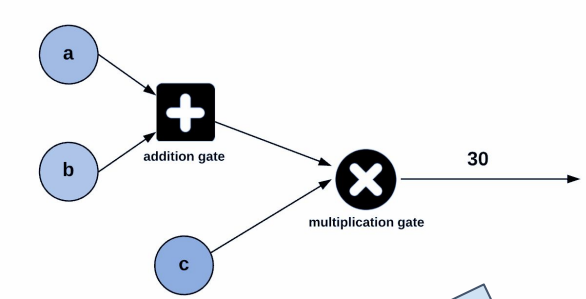
Arithmetization

R1CS - Rank-1 Constraint Systems

AIR -Algebraic Intermediate Representation

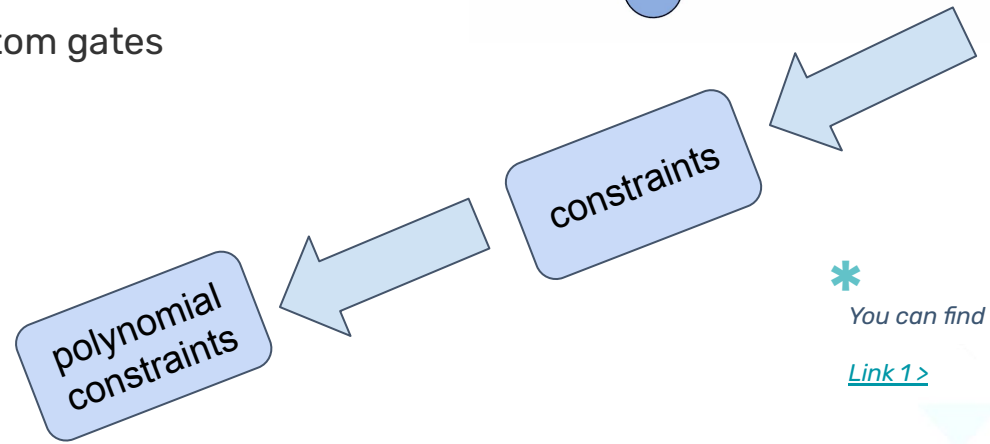
PLONK CG - PLONK custom gates

Arithmetic circuits



Execution trace

0	0	1	0	1	1
0	1	0	1	1	0
1	0	1	1	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	0	0
1	0	1	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	0	0	0
1	0	0	0	0	0

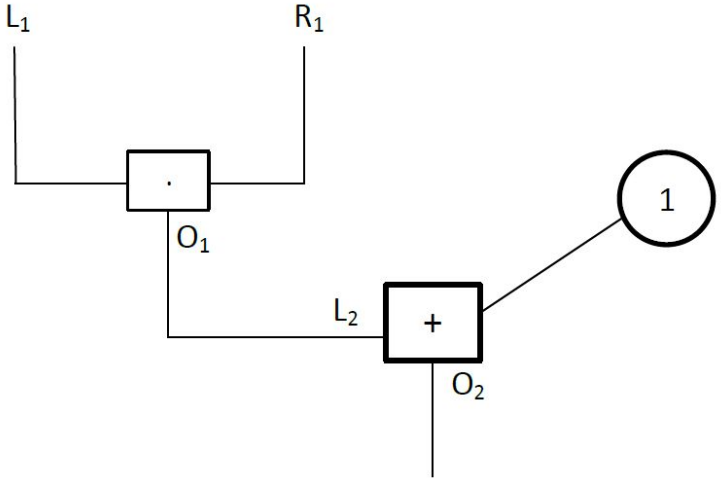


You can find more information on this link:

Link 1 >

Arithmetic circuits

Example 1: I know an *a* such that *a*a+1=b*, for given *b*.



Arithmetic circuit *C*: $F^n \to F$.

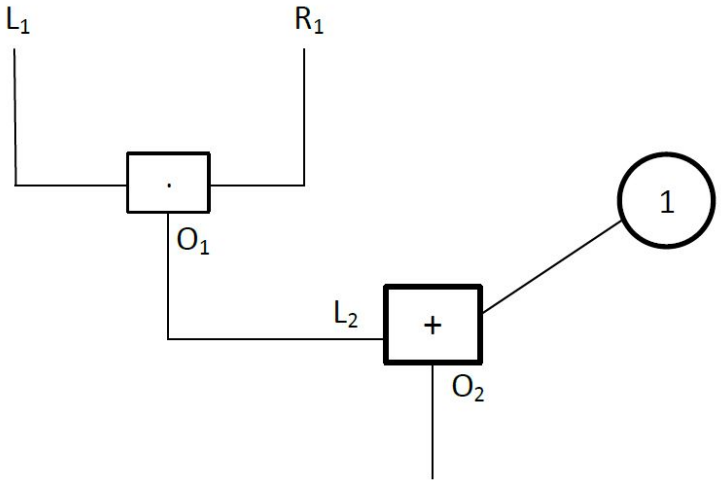
|*C*| - number of gates in *C*

Example 2: *C*_SHA256(*w*,*y*)=*y*-SHA256(*w*), |*C*| ≈ 20 000 gates

There's More 👉

Arithmetic circuits + system constraints

Example: I know an *a* such that *a*a+1=b*, for given *b*.



System constraints:

Gate constraints:

(1) $L_1 \cdot R_1 - O_1 = 0$

(2) $L_2 + 1 - O_2 = 0$

Copy constraints:

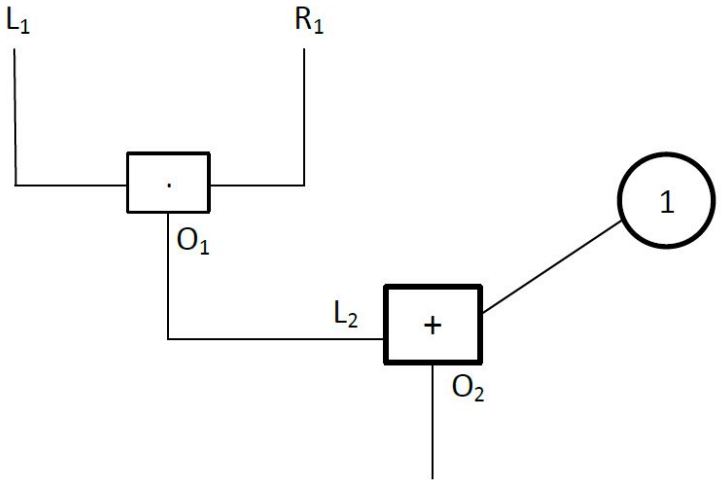
$L_1 = R_1$

$O_1 = L_2$

There's More 👉

Arithmetic circuits + system constraints

Example: I know an *a* such that *a*a+1=b*, for given *b*.



System constraints:

Gate constraints:

(1) $L_1 \cdot R_1 - O_1 = 0$

(2) $L_2 + 1 - O_2 = 0$

Copy constraints:

$L_1 = R_1$

$O_1 = L_2$

$$L_i \cdot q_{L_i} + R_i \cdot q_{R_i} + O_i \cdot q_{O_i} + q_{C_i} + L_i \cdot R_i \cdot q_{M_i} = 0.$$

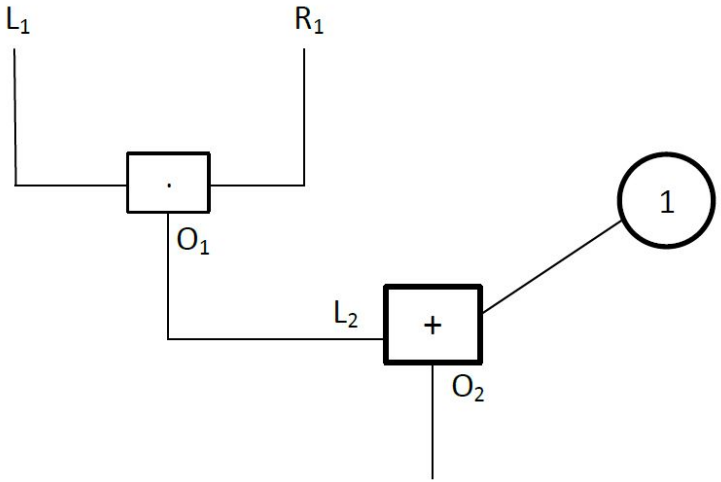
PLONK

* *You can find more information on this link:*

Link 1 >

Arithmetic circuits + polynomial constraints

Example: I know an *a* such that *a*a+1=b*, for given *b*.



System constraints:

Gate constraints:

(1) $L_1 \cdot R_1 - O_1 = 0$

(2) $L_2 + 1 - O_2 = 0$

Copy constraints:

$L_1 = R_1$

$O_1 = L_2$

$$L_i \cdot q_{L_i} + R_i \cdot q_{R_i} + O_i \cdot q_{O_i} + q_{C_i} + L_i \cdot R_i \cdot q_{M_i} = 0.$$

Define $L(x), R(x), O(x)$:

$$L(1) = L_1,\ L(2) = L_2,\ R(1) = R_1,\ R(2) = R_2,\ O(1) = O_1,\ O(2) = O_2.$$

$$f(x) = L(x) \cdot q_L(x) + R(x) \cdot q_R(x) + O(x) \cdot q_O(x) + q_C(x) + L(x) \cdot R(x) \cdot q_M(x).$$

Execution trace + constraints

Example: States for bill.

item	price	running total
Avocado	$4.98	$0.00
Apple	$7.98	$4.98
Milk	$3.45	$12.96
Bread	$2.65	$16.41
Brown Sugar	$1.40	$19.06
total	**$20.46**	$20.46

Zoom in:

Milk	$3.45	$12.96
Bread	$2.65	$16.41

There's More 👉

Execution trace + constraints

Example: States for bill.

Avocado	4.98	0
Apple	7.98	4.98
Milk	3.45	12.96
Bread	2.65	16.41
Brown Sugar	1.40	19.06
Total	20.46	20.46

Constraints:

1) $A_{0,2} = 0$ // We start the running total from 0.
2) $\forall\ 1 \le i \le 5 : A_{i,2} - A_{i-1,2} - A_{i-1,1} = 0$ // Each row's running total is correct.
3) $A_{5,1} - A_{5,2} = 0$ // The last running total is the total sum.

*

You can find more information on these links:

Link 1 >
Link 2 >

Thank you!