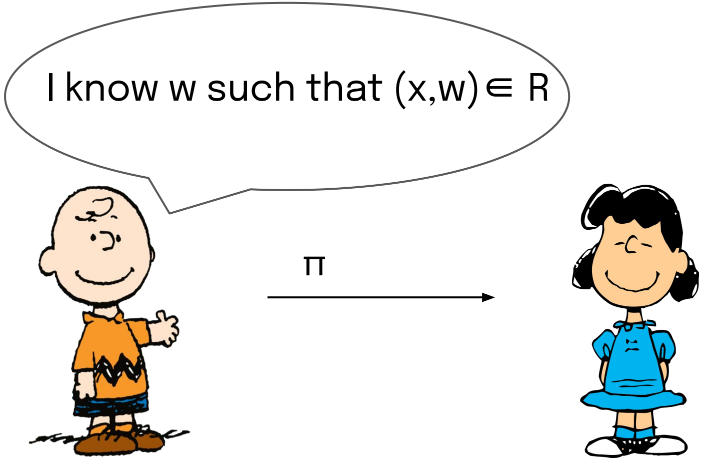
Real-world Universal zkSNARKs are non-malleable

Antonio Faonio, Dario Fiore, **Luigi Russo**

zkSNARKs

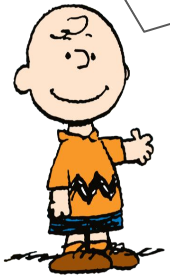


Input: R, x, w

Input: x, R
Output: accept/reject

zkSNARKs

I know w such that $(x,w) \in R$



$\xrightarrow{\pi}$

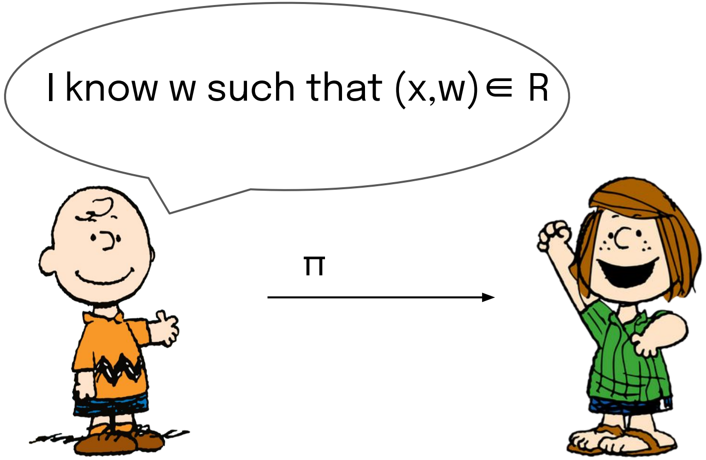
Input: R, x, w

Input: x, R
Output: accept/reject

Knowledge Soundness

If Verifier accepts, Prover “knows” w

zkSNARKs



Input: R, x, w

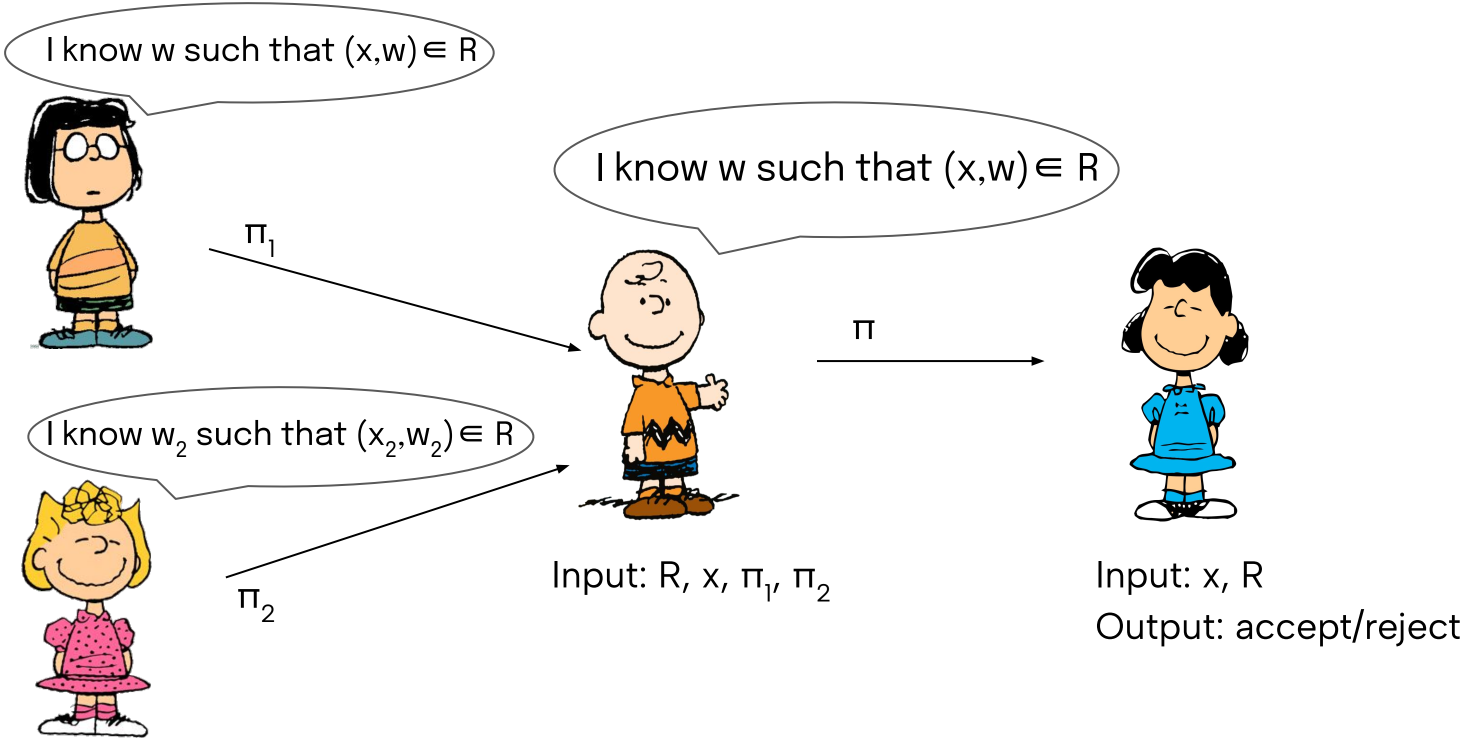
Input: x, R, …
Output: w

Knowledge Soundness

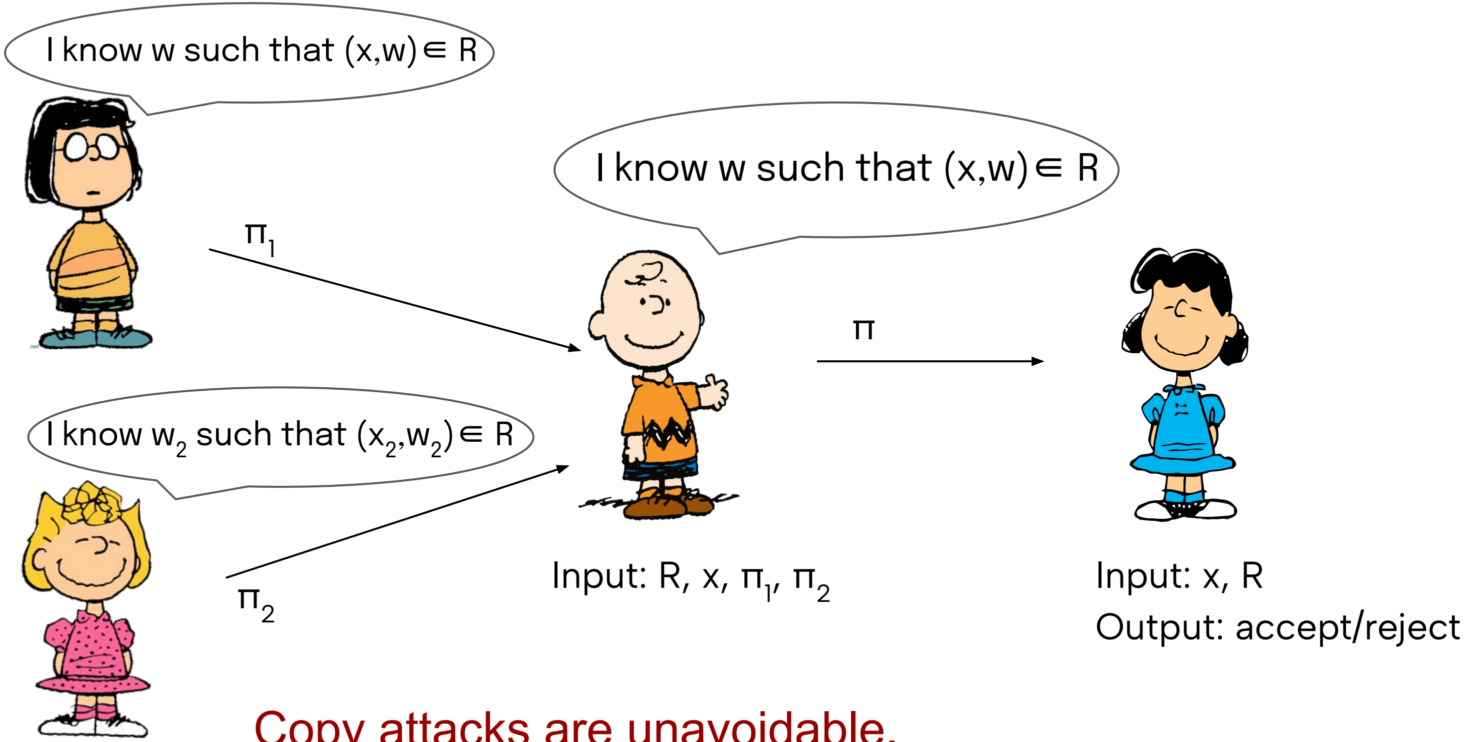
If Verifier accepts, Prover “knows" w

⇒ Extractor outputs w

zkSNARKs (in the wild)



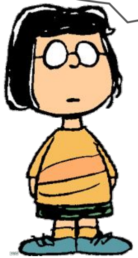
zkSNARKs (in the wild)



Copy attacks are unavoidable.

zkSNARKs (in the wild)

I know w such that $(x,w) \in R$

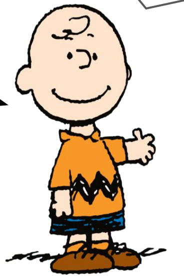


π_1

I know w_2 such that $(x_2,w_2) \in R$

π_2

I know w such that $(x,w) \in R$



Input: R, x, π_1, π_2

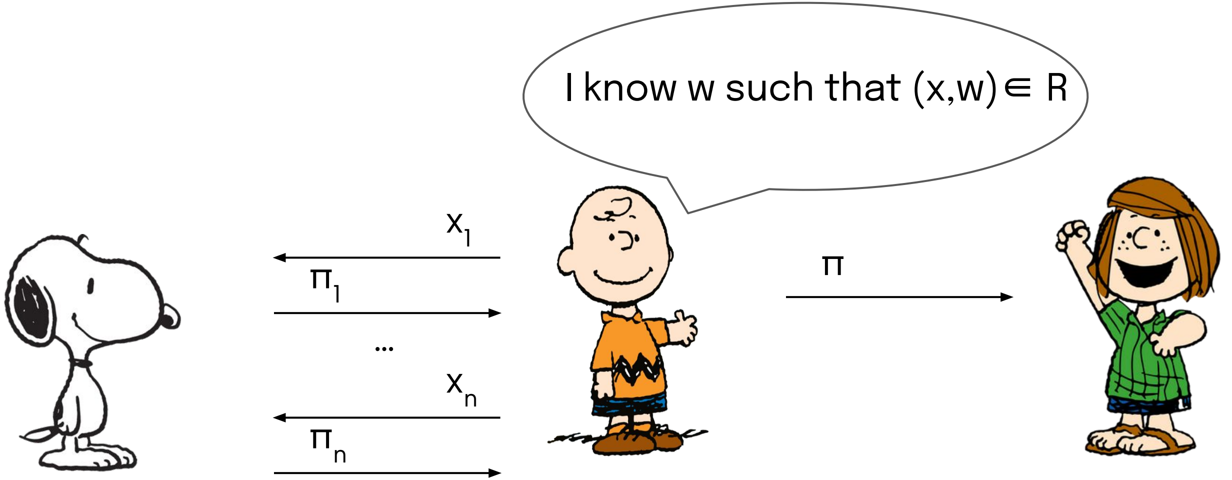
π

Input: x, R

Output: accept/reject

Copy attacks are unavoidable. What about malleability attacks?

Simulation Extractability [Sah99]



Input: R, x, π_1, ..., π_n

Input: x, R
Output: w

If Verifier accepts, Prover “knows" w ⇒ Extractor outputs w

The state of SE-zkSNARKs

	[GOP+22]	[GKK+22]	[DG23]	[FFK+23]	[KPT23]	[Lib24]
Bulletproofs	✓		✓			
Spartan			✓			
Sonic		✓				
PLONK		✓		✓	✓	
Marlin		✓		✓	✓	
Lunar				✓	✓	
Basilisk				✓		
HyperPlonk						✓

Why am I here?

Not enough to convince the reviewers

Two important gaps

Variants of zkSNARKS

Previous work can only argue for SE of small variants of protocols like Marlin and Lunar

Theory vs Implementation

Common optimizations (linearization trick) applied to zkSNARKs escape the SE security analysis in previous work

The state of SE-zkSNARKs - Closeup

	[GKK+22]	[FFK+23]	[KPT23]	[Lib24]
PLONK				
Marlin				
Lunar				
Basilisk				
HyperPlonk				

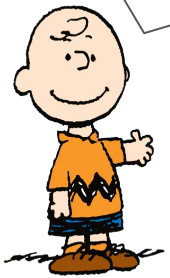
The state of SE-zkSNARKs - Revisited

	[GKK+22]	[FFK+23]	[KPT23]	[Lib24]	**Our work**
PLONK					
Marlin					
Lunar					
Basilisk					
HyperPlonk					

Anatomy of zkSNARKs

Interactive Proofs

I know w such that $(x,w) \in R$



m_1 →

← ρ_1

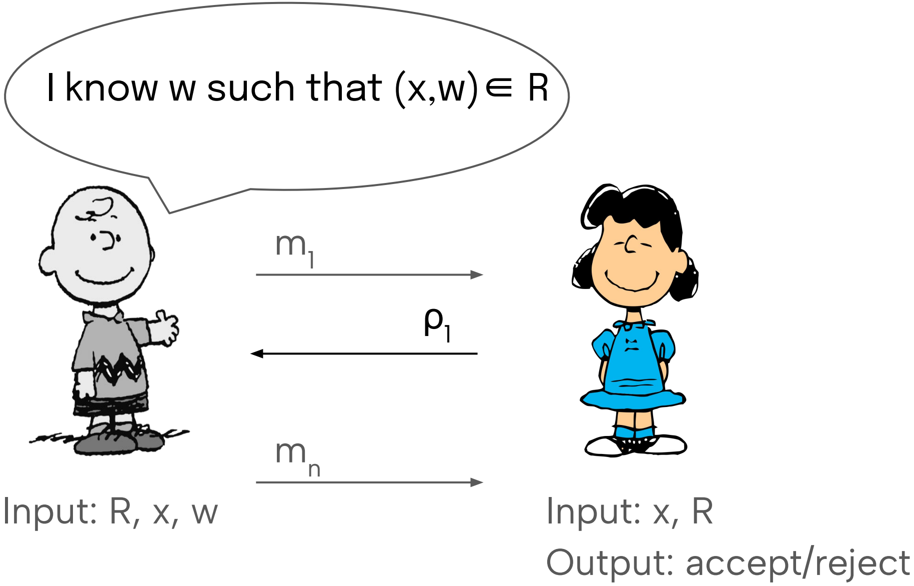
m_n →

Input: R, x, w

Input: x, R

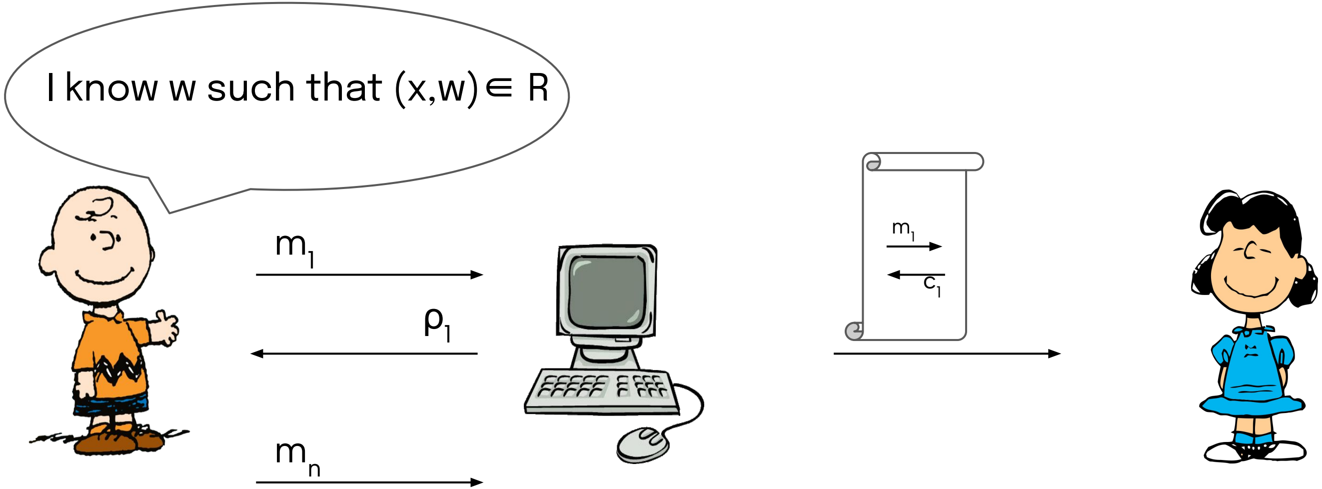
Output: accept/reject

Verifier messages



- **Public coin**
- Private coin

Non-Interactive Proofs



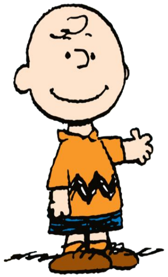
Input: R, x, w

Input: C
Output: accept/reject

If the verifier sends public random coins, apply FS transform

PIOP - Prover messages

$a(X)$	3	4	1	5	2	2	0	5
$b(X)$	2	1	8	0	3	1	4	1
$c(X)$	6	4	8	0	6	2	0	5

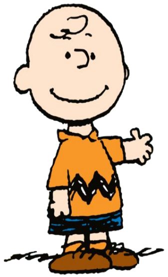


PIOP - Verifier queries

a(X)	3	4	1	5	2	2	0	5
b(X)	2	1	8	0	3	1	4	1
c(X)	6	4	8	0	6	2	0	5

ρ

$$a(\rho)b(\rho) = c(\rho)$$



a(ρ), b(ρ), c(ρ)

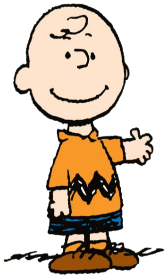
Polynomial Commitment

a(X)

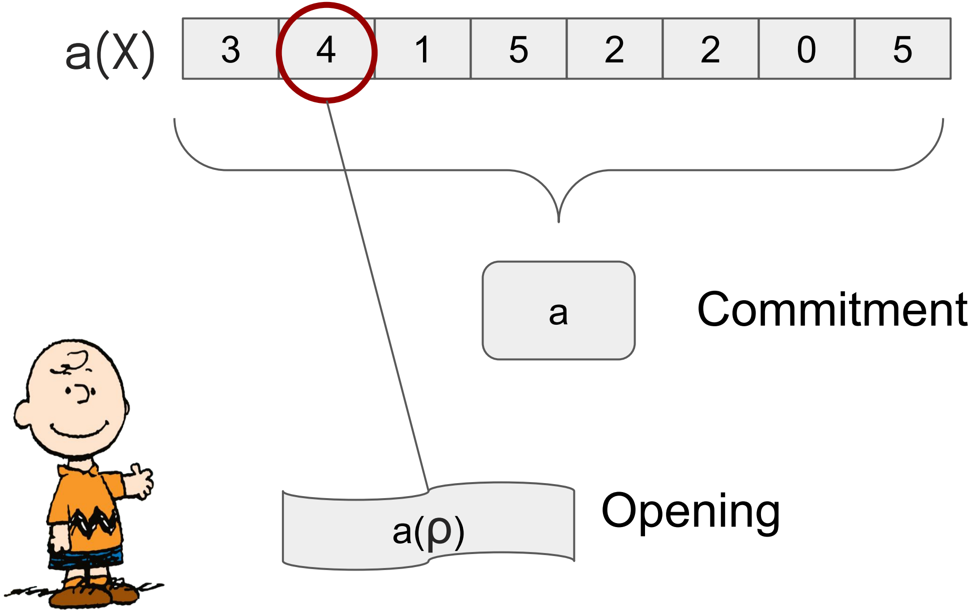
3	4	1	5	2	2	0	5

a

Commitment

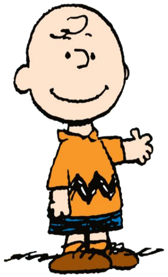


Polynomial Commitment



From PIOP to SNARK

a(X)	3	4	1	5	2	2	0	5
b(X)	2	1	8	0	3	1	4	1
c(X)	6	4	8	0	6	2	0	5



a

b

c

a(ρ), b(ρ), c(ρ)

a(ρ)

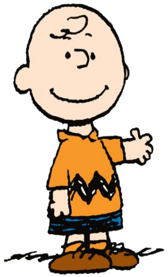
b(ρ)

c(ρ)

$$a(\rho)b(\rho) = c(\rho)$$

Batch optimizations

- If the polynomial commitment is “bacthable”, give only one evaluation proof



a b c

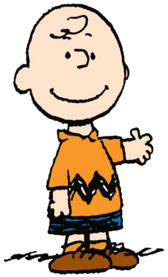
a(ρ), b(ρ), c(ρ)

a(ρ), b(ρ), c(ρ)

$$a(\rho)b(\rho) = c(\rho)$$

Batch optimizations

- If the polynomial commitment is “bacthable”, give only one evaluation proof
- If the polynomial commitment is homomorphic, we can do better



a b c

a(ρ), b(ρ), c(ρ)

a(ρ), b(ρ), c(ρ)

How?

$$a(\rho)b(\rho) = c(\rho)$$

Linearization trick
(Maller’s optimization)

Linearization trick

$$a(\rho)b(\rho) = c(\rho)$$

“Naive” approach: a(ρ), b(ρ), c(ρ)

a(ρ), b(ρ), c(ρ)

Linearization trick

$$a(\rho)b(\rho) = c(\rho)$$

1 $b(\rho) = z$

z b(ρ)

"Naive" approach: a(ρ), b(ρ), c(ρ) a(ρ), b(ρ), c(ρ)

Linearization trick

$$a(\rho)b(\rho) = c(\rho)$$

1 $b(\rho) = z$

z b(ρ)

2 $L(X) := a(X)z - c(X)$

$L(\rho) = 0$

L(ρ)

"Naive" approach: a(ρ), b(ρ), c(ρ) a(ρ), b(ρ), c(ρ)

Linearization trick

$$a(\rho)b(\rho) = c(\rho)$$

1 $b(\rho) = z$

z b(ρ)

2 $L(X) := a(X)z - c(X)$

$L(\rho) = 0$

L(ρ)

Optimization:	b(ρ)	b(ρ), L(ρ)
"Naive" approach:	a(ρ), b(ρ), c(ρ)	a(ρ), b(ρ), c(ρ)

KS of Linearization trick

$$a(\rho)b(\rho) = c(\rho)$$

a b c

a(X) b(X) c(X)

b(ρ)

b(ρ), L(ρ)

KS of Linearization trick [LPS23]

$$a(\rho)b(\rho) = c(\rho)$$

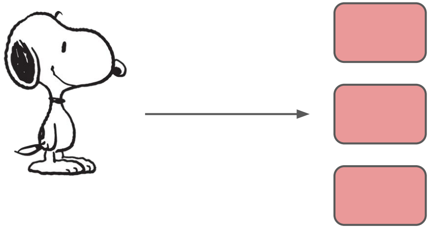
a → a(X)

b → b(X)

c → c(X)

b(ρ)

b(ρ), L(ρ)



KS of Linearization trick [LPS23]

$$a(\rho)b(\rho) = c(\rho)$$

a	b	c
?	b(X)	?

b(ρ) b(ρ), L(ρ)

Simple attack: $b(X) := 1$ a $=$ c $\Rightarrow L(X) := 0$

KS of Linearization trick

$$\sum_i a_i(\rho)b_i(\rho) = y$$

$(a_i(\rho))_i$

$(a_i(\rho))_i$, $L(\rho)$

KS of Linearization trick

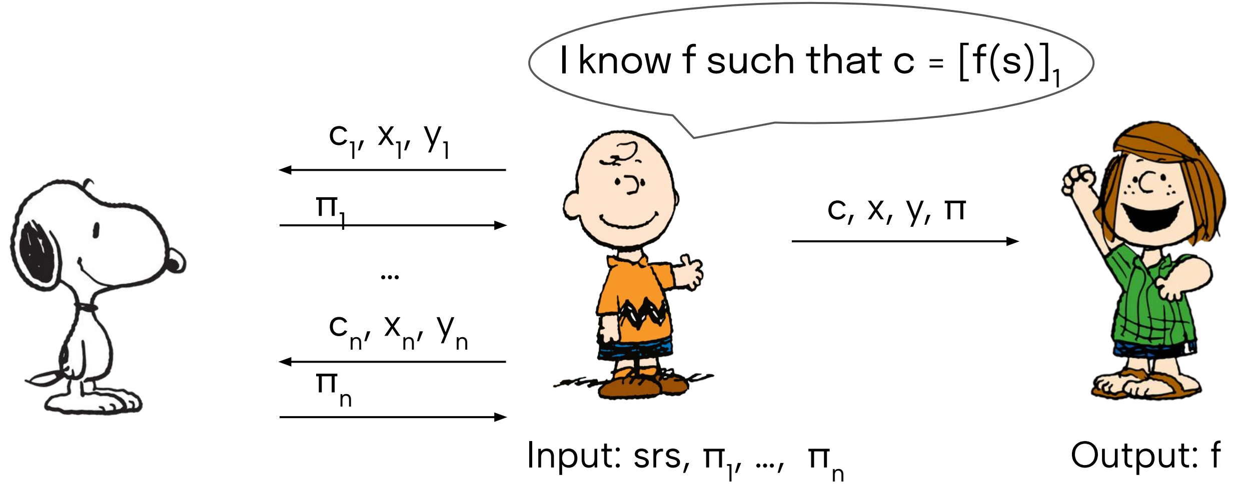
$$\sum_i a_i(\rho) b_i(\rho) = y$$

$(a_i(\rho))_i$ $\quad$ $(a_i(\rho))_i, L(\rho)$

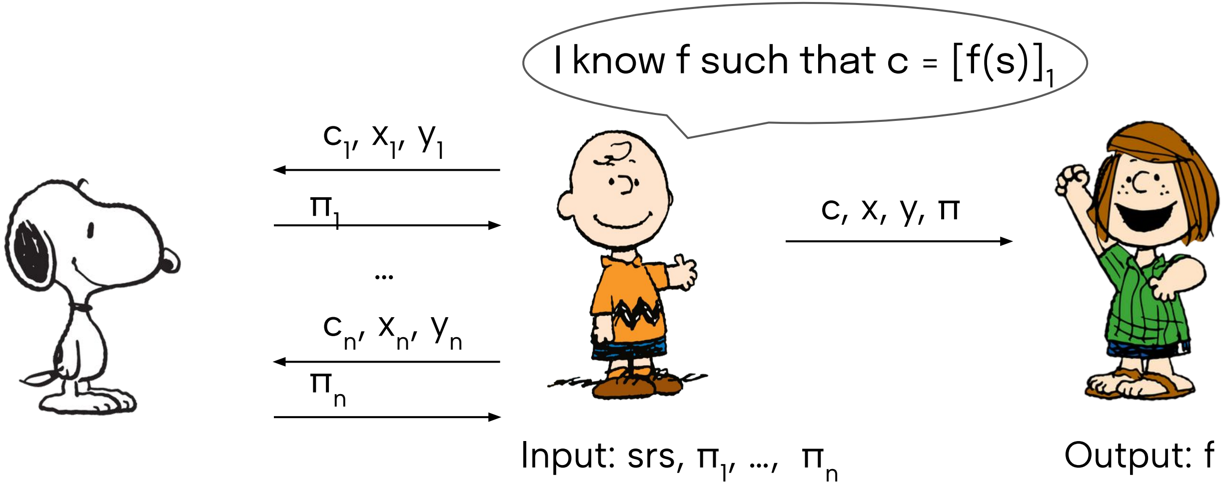
Linearization trick is KS if $a_i(X)$ are linearly independent polynomials

KZG-based schemes

Is KZG simulation-extractable?



KZG is (sort of) SE [FFK+23]



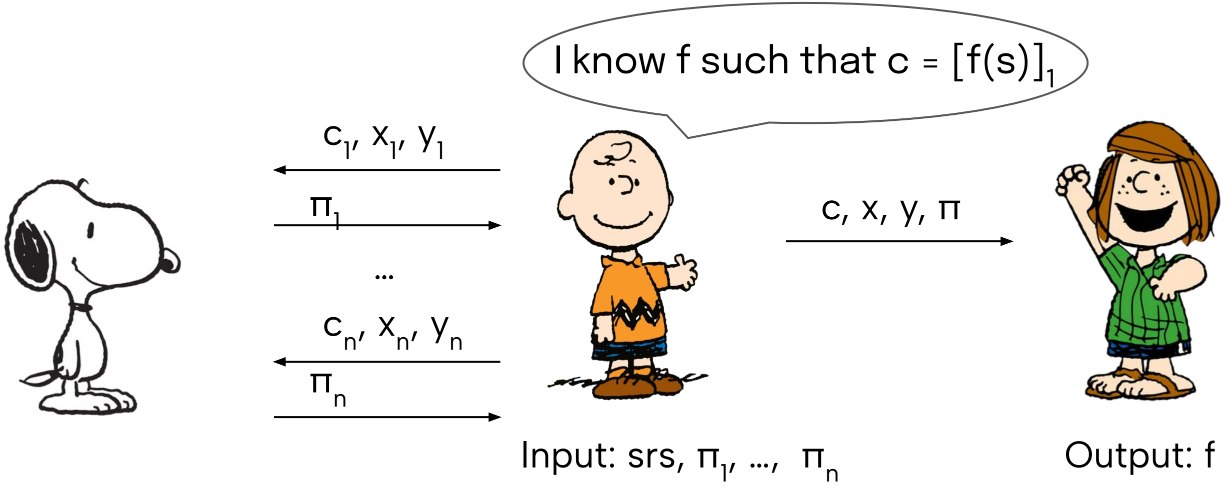
Simulation Constraints

- Algebraic Check
- Point check
- Commitment Check

Extraction Constraints

- Hash Check

KZG is (sort of) SE - Revisited



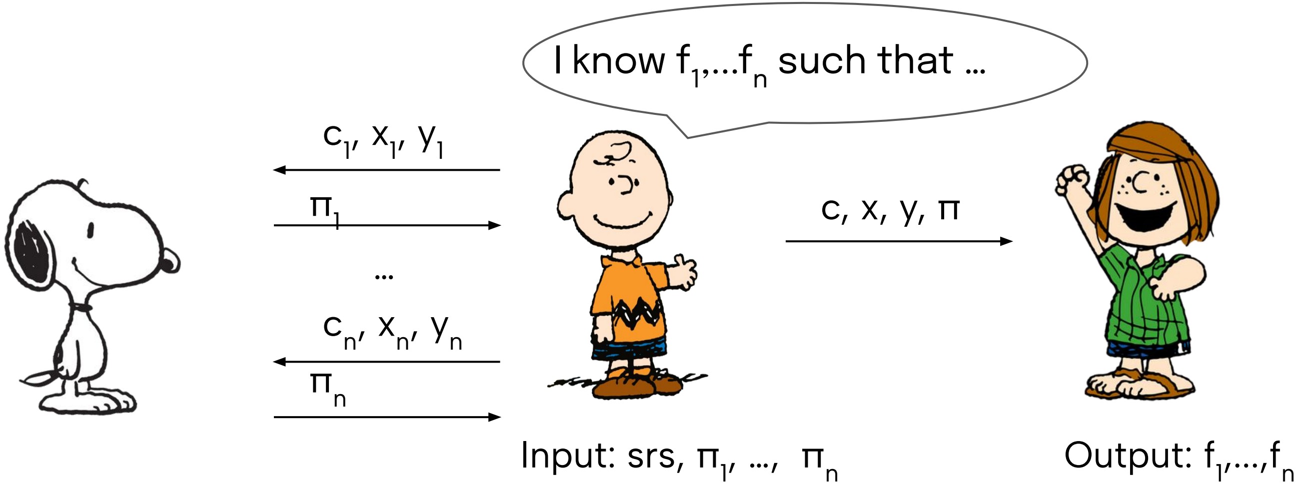
Simulation Constraints

- Algebraic Check
- ~~Point check~~
- ~~Commitment Check~~

Extraction Constraints

- Hash Check w/ lin. com.

KZG+Lin. Trick is (sort of) SE



Simulation Constraints

- Algebraic Check

Extraction Constraints

- Hash Check w/ lin. com.
- Linear Independence?

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments $$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Simple attack: ask proof for (c,x=0,y=0) -> π=c/s

$$a(\rho)b(\rho) = c(\rho)$$

a=c/s

b=[s]

c

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Simple attack: ask proof for (c,x=0,y=0) -> π=c/s

$$a(\rho)b(\rho) = c(\rho)$$

a=c/s

b=[s]

c

“High degree” linear independence is required

SE zkSNARKs

PIOP + “sort of ”SE PCOM + FS = SE zkSNARK

PIOP requirements
polynomials are evaluated on
(a function of) the last random coin

SE zkSNARKs

PIOP + “sort of ”SE PCOM + FS = SE zkSNARK

PIOP requirements

polynomials are evaluated on
(a function of) the last random coin

Linearization trick

“high degree” linear independence

SE zkSNARKs

PIOP + "sort of "SE PCOM + FS = SE zkSNARK

	PLONK
PIOP requirements polynomials are evaluated on (a function of) the last random coin	✓
Linearization trick "high degree" linear independence	✓

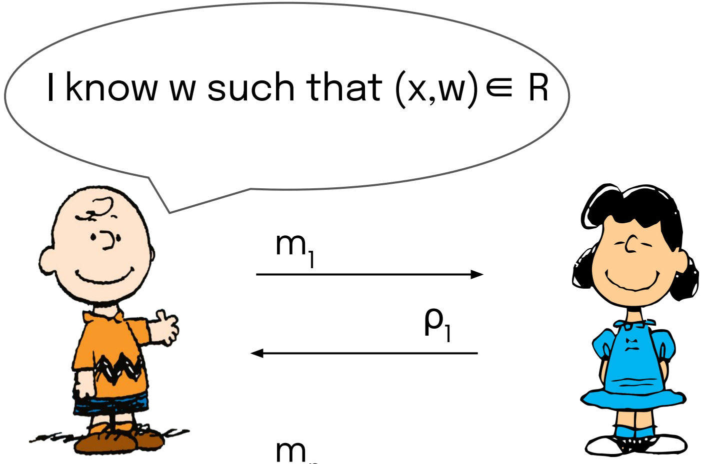
SE zkSNARKs

PIOP + "sort of" SE PCOM + FS = SE zkSNARK

	PLONK	Marlin
PIOP requirements polynomials are evaluated on (a function of) the last random coin	✓	
Linearization trick "high degree" linear independence	✓	✓

Delegation Phase

PIOP with a Delegation Phase



Input: R, x, w

Input: x, R
Output: accept/reject

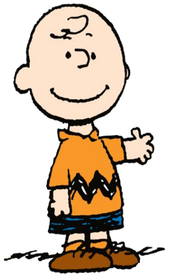
PIOP with a Delegation Phase

Input: R, x, w

$\xrightarrow{m_1}$

$\xleftarrow{\rho_1}$

$\xrightarrow{m_{n-2}}$



Input: R, x

$\xrightarrow{m_{n-1}}$

$\xleftarrow{\rho_{n-1}}$

$\xrightarrow{m_n}$

Input: x, R

Output: accept/reject

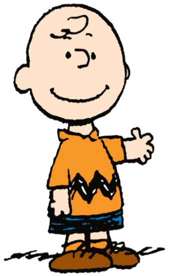
PIOP with a Delegation Phase

Input: R, x, w

$\xrightarrow{m_1}$

$\xleftarrow{\rho_1}$

$\xrightarrow{m_{n-2}}$



Input: R, x

$\xrightarrow{m_{n-1}}$

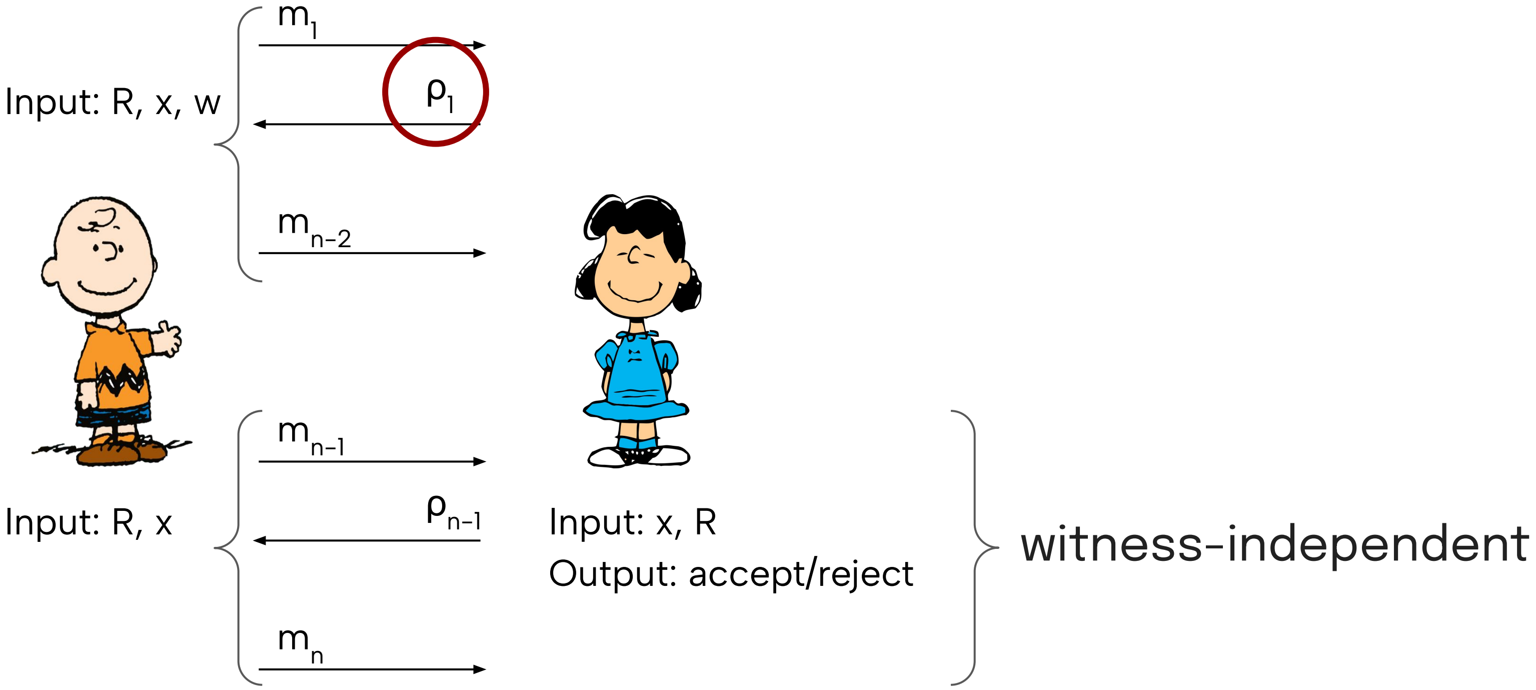
$\xleftarrow{\rho_{n-1}}$

$\xrightarrow{m_n}$

Input: x, R
Output: accept/reject

witness-independent

PIOP with a Delegation Phase



SE zkSNARKs

PIOP + φ-SE PCOM + FS = SE zkSNARK

	PLONK	Marlin
PIOP requirements		
polynomials are evaluated on (a function of) the last random coin	✓	✓*
Linearization trick		
"high degree" linear independence	✓	✓

Marlin is simulation-extractable but…

Delegation vs Uniqueness

Compilation must preserve the uniqueness of the delegation phase

- \+ KZG commitments
- \- hiding KZG
- \- FRI

Thank you

Real-world Universal zkSNARKs are non-malleable

Antonio Faonio[1], Dario Fiore[2], and Luigi Russo[1]

[1] EURECOM, Sophia Antipolis, France {faonio,russol}@eurecom.fr
[2] IMDEA Software Institute, Madrid, Spain dario.fiore@imdea.org

Abstract. Simulation extractability is a strong security notion of zkSNARKs that guarantees that an attacker who produces a valid proof must know the corresponding witness, even if the attacker had prior access to proofs generated by other users. Notably, simulation extractability implies that proofs are non-malleable and is of fundamental importance for applications of zkSNARKs in distributed systems. In this work, we study sufficient and necessary conditions for constructing simulation-extractable universal zkSNARKs via the popular design approach based on compiling polynomial interactive oracle proofs (PIOP). Our main result is the first security proof that popular universal zkSNARKs, such as PLONK and Marlin, *as deployed in the real world*, are simulation-extractable. Our result fills a gap left from previous work (Faonio et al. TCC'23, and Kohlweiss et al. TCC'23) which could only prove the simulation extractability of the "textbook" versions of these schemes and does not capture their optimized variants, with all the popular optimization tricks in place, that are eventually implemented and deployed in software libraries.

ia.cr/2024/721

References

[DG23] Quang Dao and Paul Grubbs. Spartan and bulletproofs are simulation-extractable (for free!). EUROCRYPT 2023

[FFK+23] Antonio Faonio, Dario Fiore, Markulf Kohlweiss, Luigi Russo, and Michal Zajac. From polynomial IOP and commitments to non-malleable zkSNARKs. TCC 2023

[GKK+22] Chaya Ganesh, Hamidreza Khoshakhlagh, Markulf Kohlweiss, Anca Nitulescu, and Michal Zajac. What makes fiat-shamir zksnarks (updatable SRS) simulation extractable? SCN 2022

[GOP+22] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). EUROCRYPT 2022

[KPT23] Markulf Kohlweiss, Mahak Pancholi, and Akira Takahashi. How to compile polynomial IOP into simulation-extractable SNARKs: A modular approach. TCC 2023

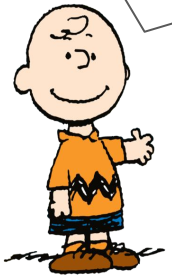
[Lib24] Benoit Libert. Simulation-Extractable KZG Polynomial Commitments and Applications to HyperPlonk. PKC 202

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. FOCS 1999

Additional notes

zkSNARKs

I know w such that $(x,w) \in R$



$\xrightarrow{\pi}$

Input: R, x, w

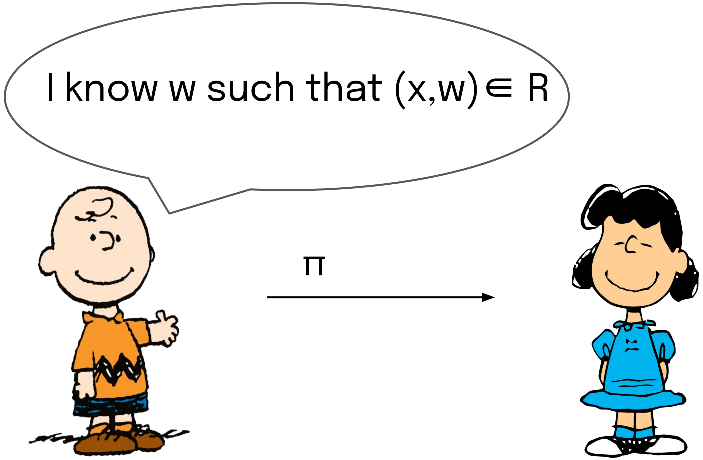
Input: x, R
Output: accept/reject

Zero-Knowledge

Verifier learns nothing

besides that $(x,w) \in R$

zkSNARKs



Input: R, x, w

Input: x, R
Output: accept/reject

- Short and efficient to verify
- Non-Interactive
- Efficient to generate

KZG Polynomial Commitment

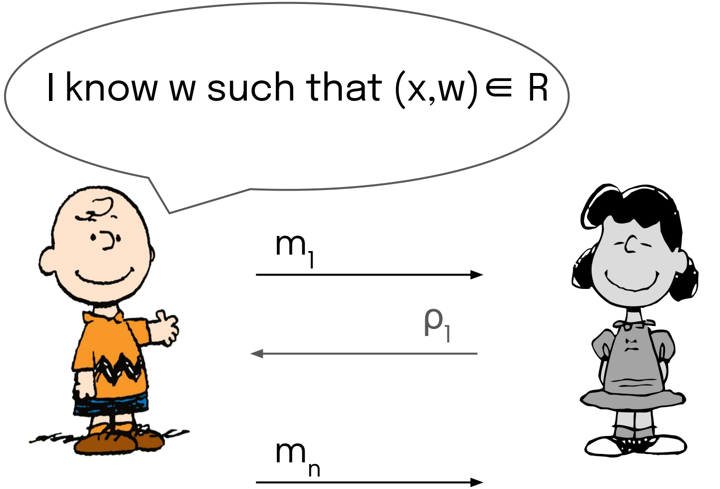
$$\mathrm{srs} \leftarrow \left(\left[1,\ s, \ldots, s^d\right]_1,\ [1, s]_2\right)$$

$$\mathrm{Com}(p) \rightarrow [p(s)]_1$$

$$\mathrm{Open}(p, x, y) \rightarrow \left[\frac{p(s) - p(x)}{s - x}\right]_1$$

$$\mathrm{Verify}(C, x, y, \pi) \rightarrow 1 \iff e(C - [y]_1,\ [1]_2) = e([\pi]_1, [s - x]_2)$$

Prover messages



Input: R, x, w

Input: x, R
Output: accept/reject

- Univariate Polynomials
- Multivariate Polynomials
- …

From PIOP to zkSNARKs - Something old

PIOP + **KS** PCOM + FS = **KS** zkSNARK

PIOP + **SE** PCOM + FS = **SE** zkSNARK

From PIOP to zkSNARKs - Something new

PIOP + KS PCOM + FS = KS zkSNARK

PIOP + SE PCOM + FS = SE zkSNARK

PIOP + SE PCOM **+ Lin. Trick** + FS = **Real World** SE zkSNARK

Lin. Trick - SE attack

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Simple attack: ask proof for (c,x=0,y=0)

$$a(\rho)b(\rho) = c(\rho)$$

a=c/s

b=[s]

c

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Simple attack: ask proof for (c,x=0,y=0)

$$b(\rho) = \rho$$

a=c/s

b=[s]

c

$$e(b - [\rho]_1, [1]_2) = e([1]_1, [s - \rho]_2)$$

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Simple attack: ask proof for (c,x=0,y=0)

$$b(\rho) = \rho$$
$$a(\rho)\rho = c(\rho)$$

a=c/s

b=[s]

c

$$e(b - [\rho]_1, [1]_2) = e([1]_1, [s - \rho]_2)$$
$$e(a\rho - c, [1]_2) = e([c/s]_1, [s - \rho]_2)$$

Linear independence is not enough for SE

Commitments vs Proofs

KZG proofs are in fact commitments

$$e(c - [y]_1, [1]_2) = e(\pi, [s - x]_2)$$

Simple attack: ask proof for (c,x=0,y=0)

$$a(\rho)b(\rho) = c(\rho)$$

a=c/s

b=[s]

c

$$e(b + \zeta(a\rho - c) - [\rho]_1, [1]_2) = e([1 + \zeta(c/s)]_1, [s - \rho]_2)$$

Credits

All images used in this presentation are being used for educational purposes in accordance with the principles of fair use. The copyright of these images remains with their respective owners. No infringement is intended.