Foundations and Applications of Zero-Knowledge Proofs

4 Sept 2024

zkSNARKs for Blockchains

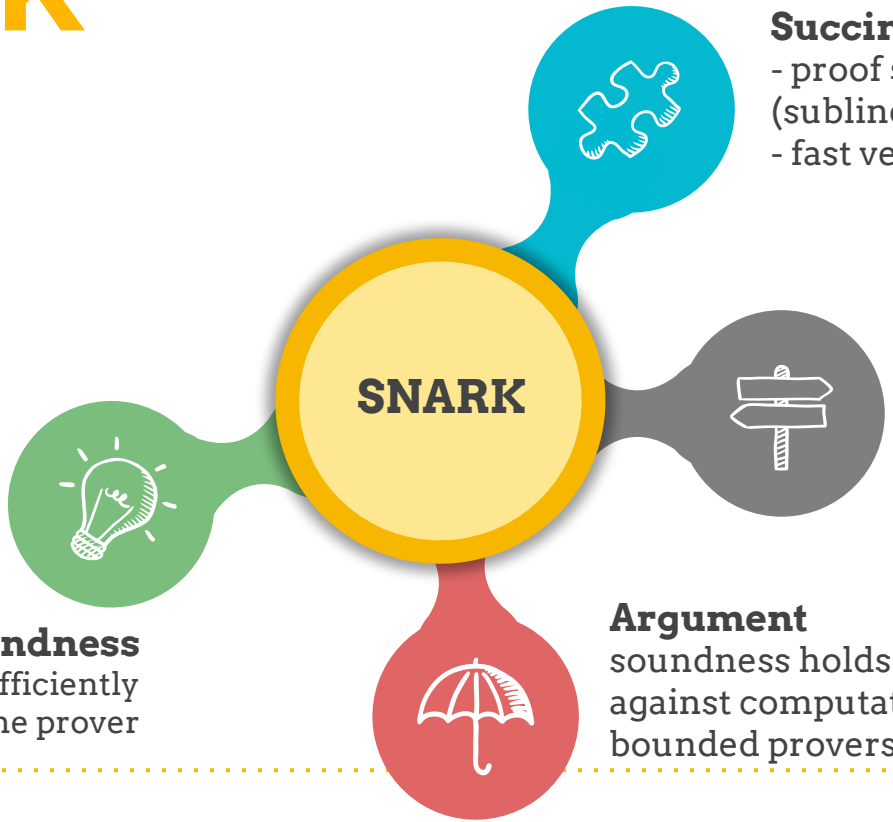
Anca Nitulescu

INPUT | OUTPUT

SNARKs

Background

SNARK



Succinctness
- proof size independent (sublinear) of NP witness size
- fast verification

Non-Interactivity
no exchange between prover and verifier

Argument
soundness holds only against computationally bounded provers

Knowledge Soundness
a witness can be efficiently extracted from the prover

zk-SNARK

Zero-Knowledge
does not leak anything about the witness

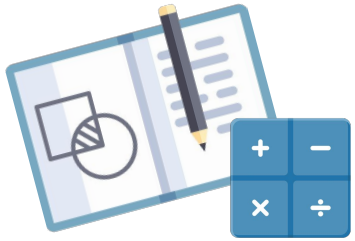
Succinctness
- proof size independent (sublinear) of NP witness size
- fast verification

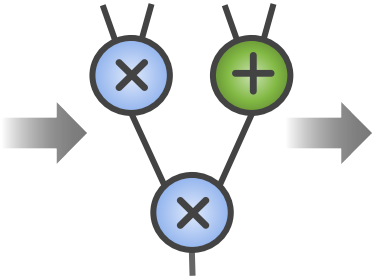
Non-Interactivity
no exchange between prover and verifier

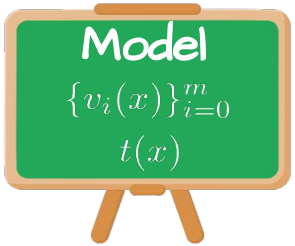
Knowledge Soundness
a witness can be efficiently extracted from the prover

Argument
soundness holds only against computationally bounded provers

Building SNARKs







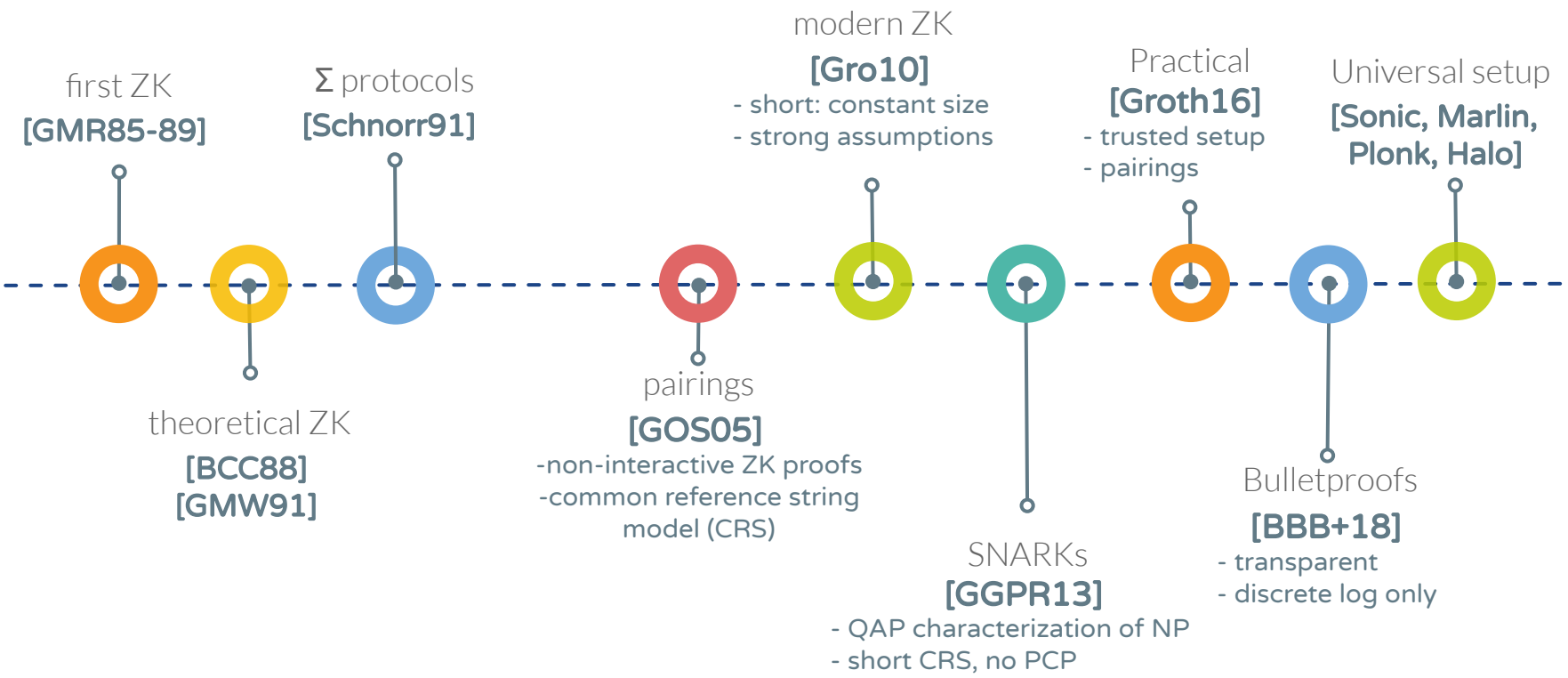
Target Statement
∠,R(y,w)=1

Circuit SAT
NP-complete

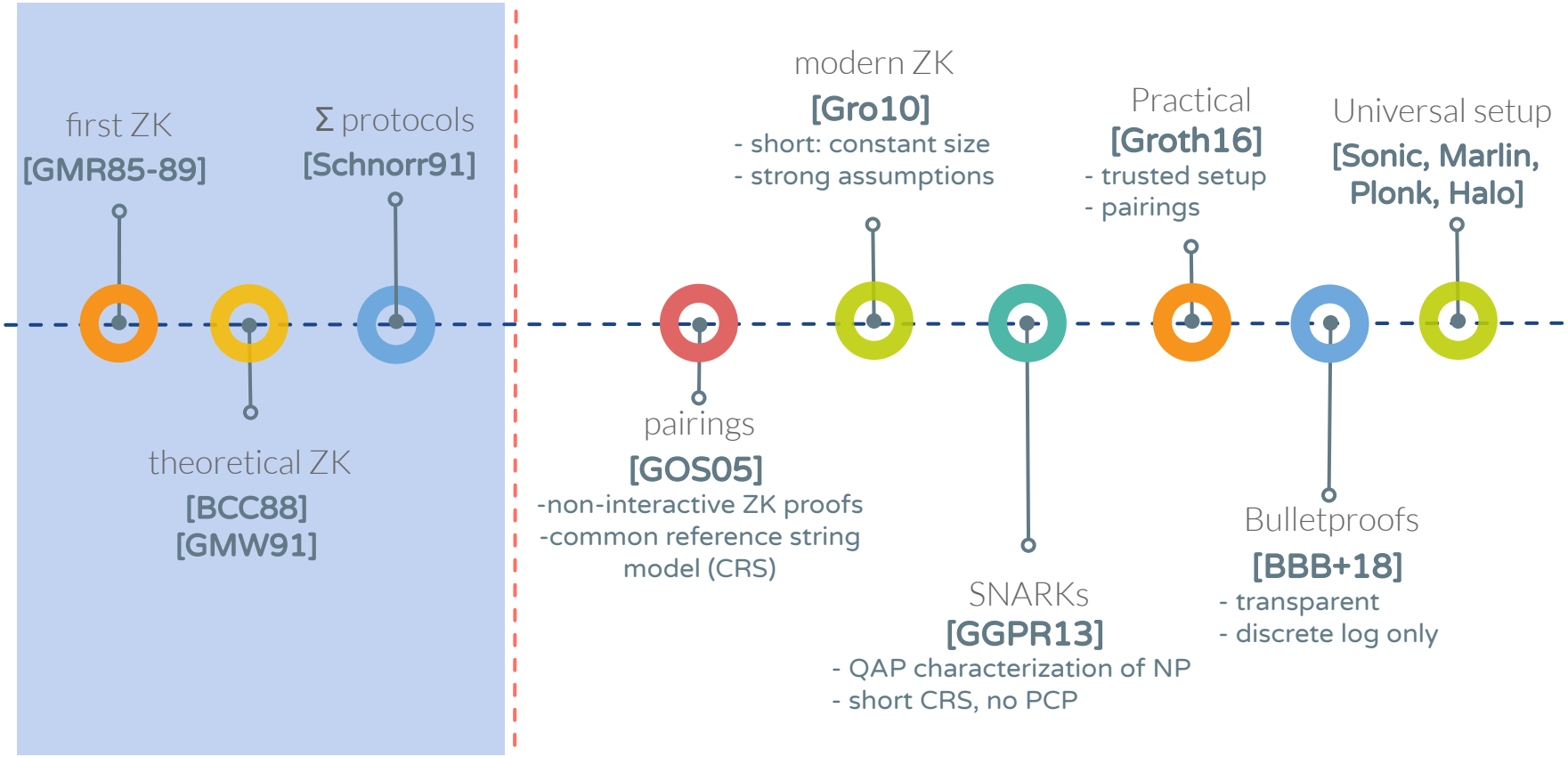
Computational Model
(Polynomials)

Cryptographic Compilation

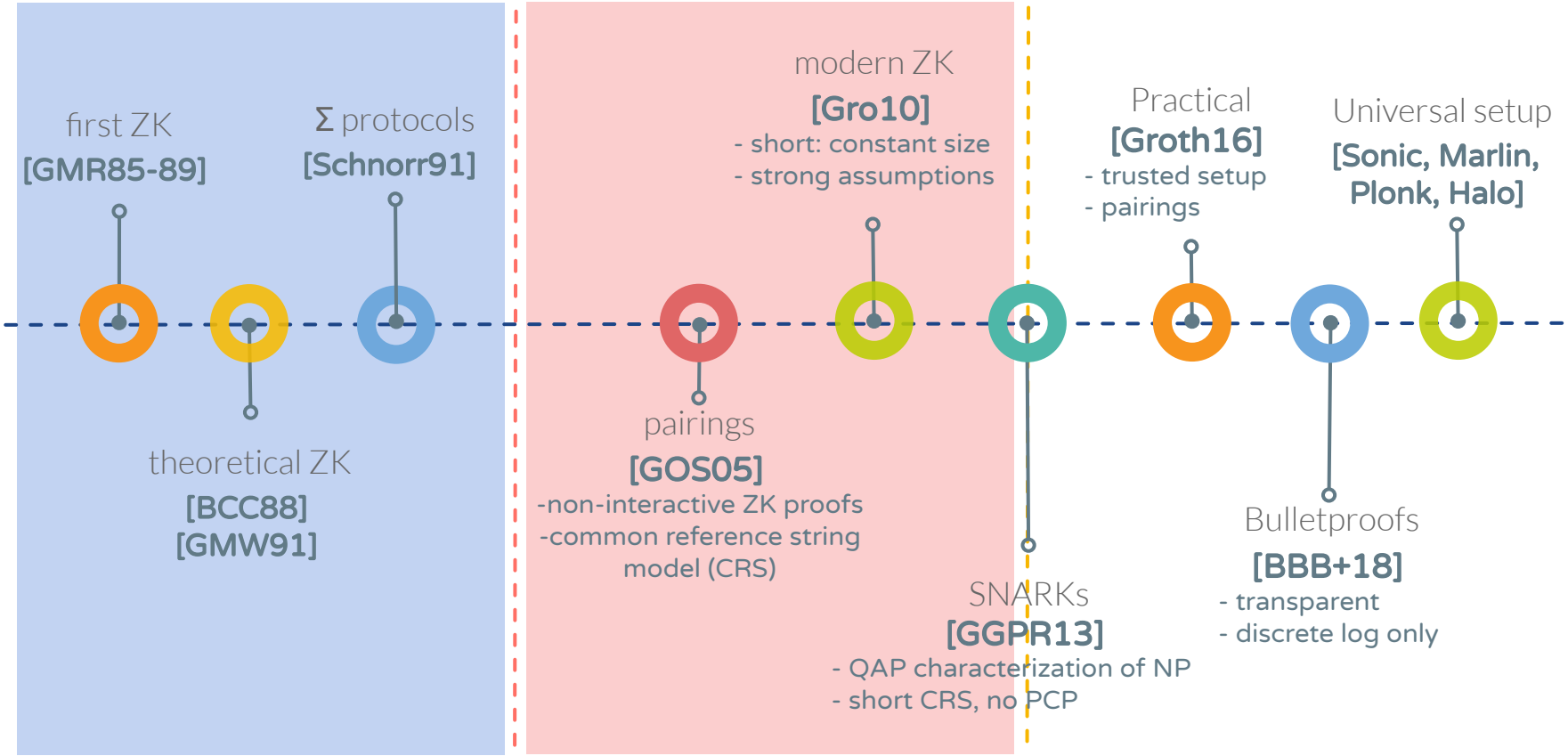
Non-Interactive ZK Proofs



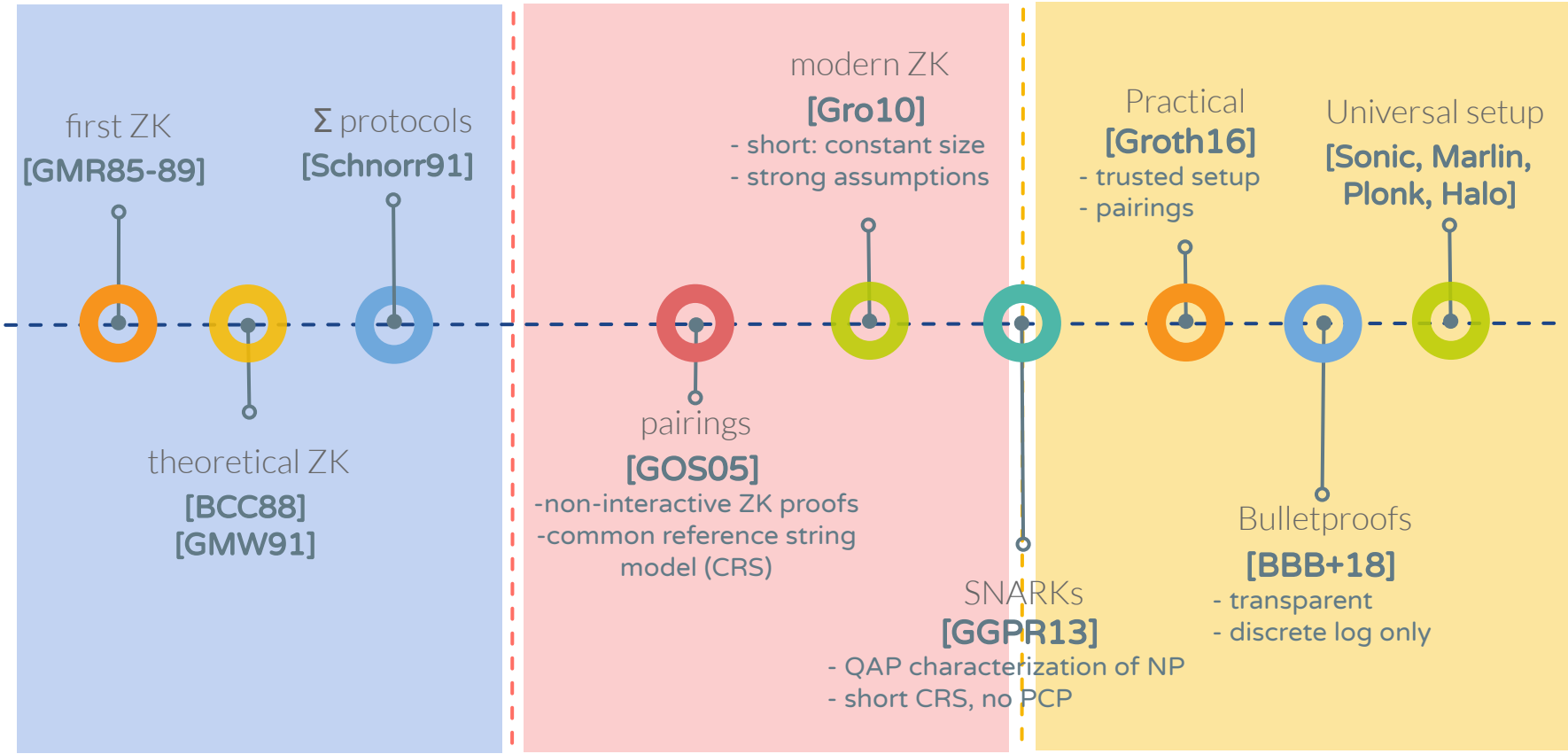
Non-Interactive ZK Proofs



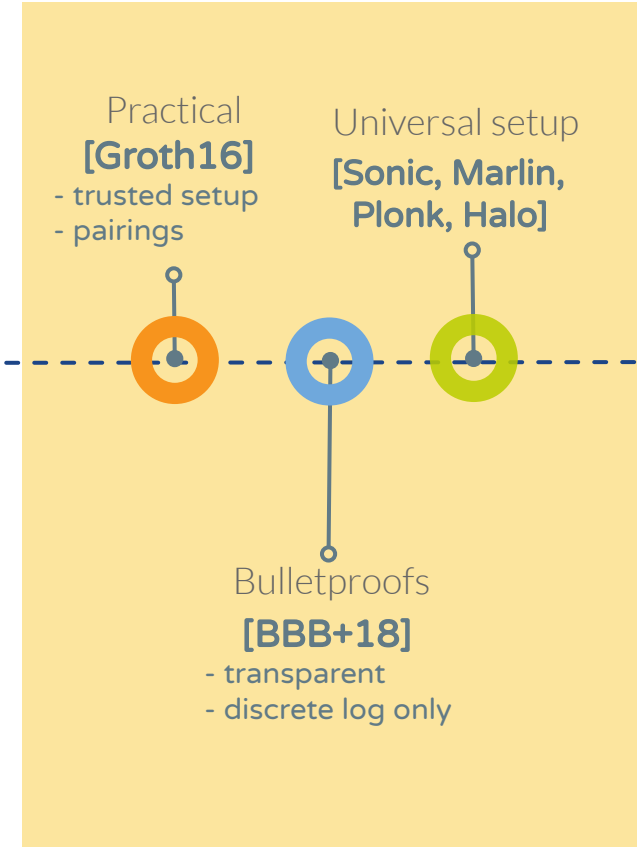
Non-Interactive ZK Proofs



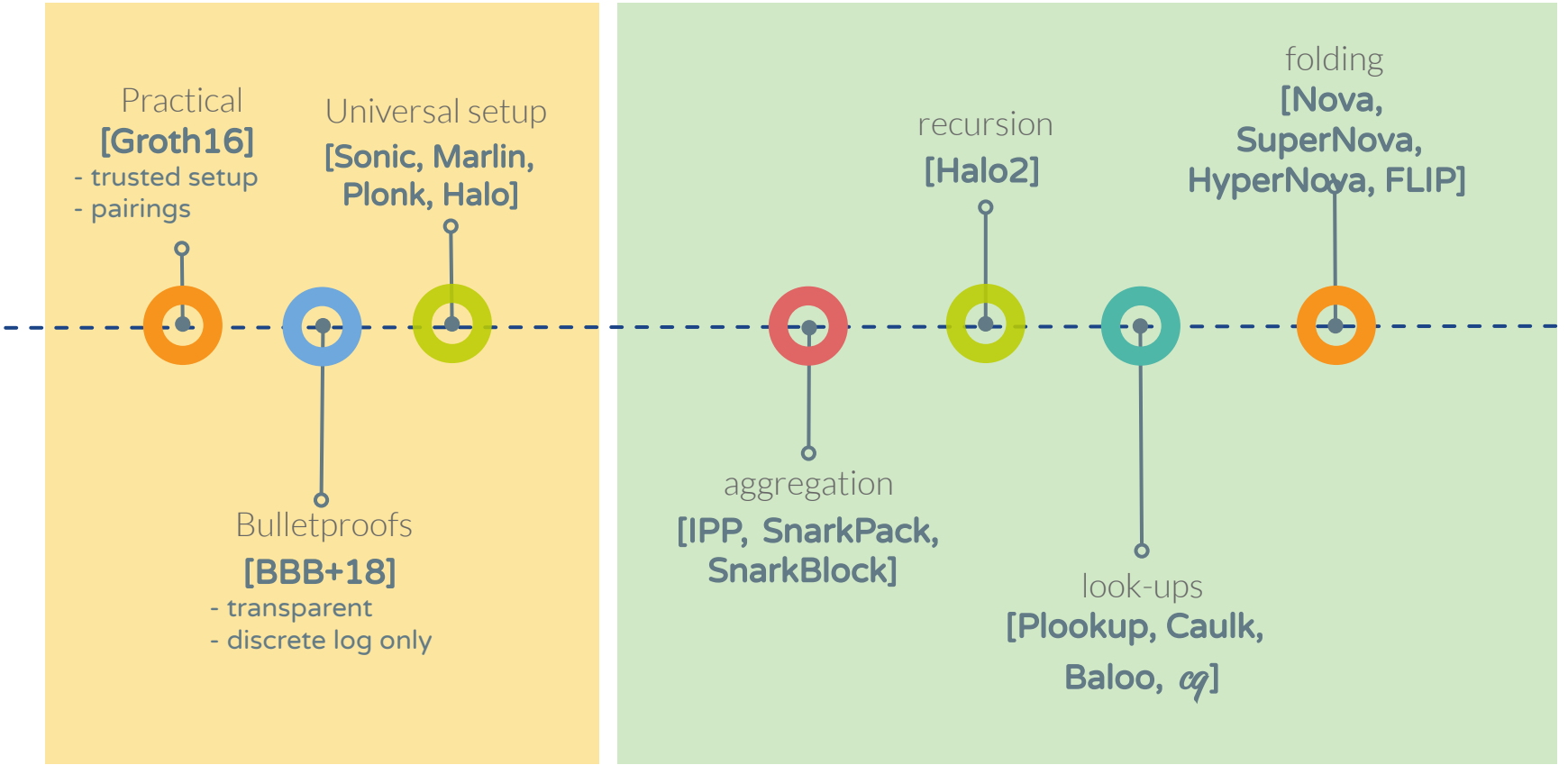
Non-Interactive ZK Proofs



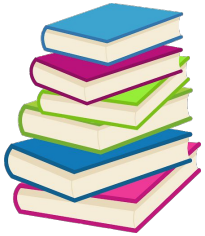
Recent zk-SNARK research



Recent zk-SNARK research



References



[GMR85-89] **The Knowledge Complexity Of Interactive Proof Systems,** by Shafi Goldwasser, Silvio Micali, and Charles Rackoff

[GMW91] **How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design,** by Oded Goldreich, Silvio Micali, and Avi Wigderson

[BCC88] **Minimum Disclosure Proofs of Knowledge**, by Gilles Brassard, David Chaum, and Claude Crépeau

[Sch91] **Efficient signature generation by smart cards,** by Claus-Peter Schnorr, Journal of Cryptology, 1991.

[GOS06] **Perfect Non-interactive Zero Knowledge for NP**, Jens Groth, Rafail Ostrovsky, and Amit Sahai, EUROCRYPT 2006.

[Groth10] **Short pairing-based non-interactive zero-knowledge arguments**, Jens Groth, ASIACRYPT 2010.

[KZG10] **Constant-Size Commitments to Polynomials and Their Applications**, by A. Kate, G. Zaverucha, I. Goldberg, in ASIACRYPT'10.

[GGPRR13] **Quadratic span programs and succinct NIZKs without PCPs**, by R. Gennaro, C. Gentry, Bryan Parno, and Mariana Raykova.

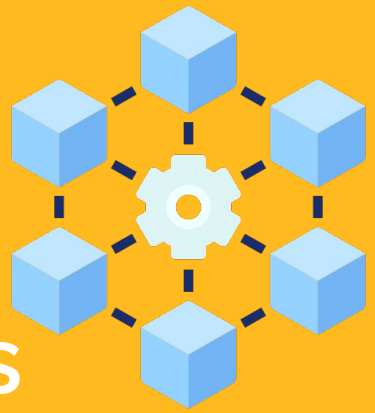
[BBB+18] **Bulletproofs: Short Proofs for Confidential Transactions and More**, by B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwel

[Groth16] **On the Size of Pairing-based Non-interactive Arguments**, Jens Groth, EUROCRYPT 2016.

[GWC19] **PlonK: Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge,** Ariel Gabizon, Zachary J. Williamson, Oana Ciobotaru

Applications

to descentralized systems

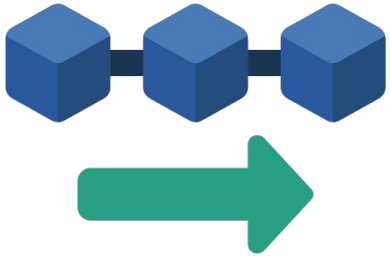


Crypto currencies

Distributed Ledger

Alice:	20 Coins
Bob:	55 Coins
Carol:	0 Coins
...	
Zoe:	17 Coins

tx: Alice → Bob, **5 Coins**

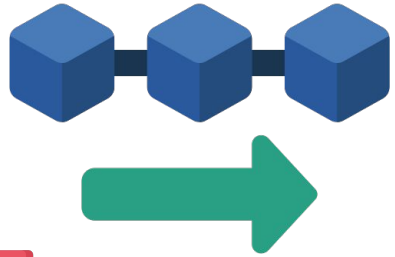


Crypto currencies

Distributed Ledger

Alice:	20 Coins
Bob:	55 Coins
Carol:	0 Coins
...	
Zoe:	17 Coins

tx: Alice → Bob, **5 Coins**



New State

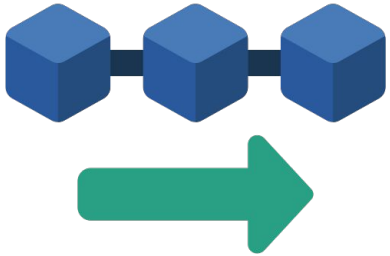
Alice:	15 Coins
Bob:	60 Coins
Carol:	0 Coins
...	
Zoe:	17 Coins

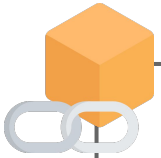
Crypto currencies

Distributed Ledger

Alice:	20 Coins
Bob:	55 Coins
Carol:	0 Coins
...	
Zoe:	17 Coins

tx: Alice → Bob, **5 Coins**



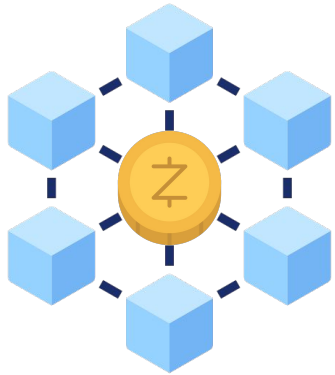


- inclusive accountability:
  anyone can see and check validity of actions
- all transactions are public
- consensus: multiple nodes check a new block

ZK Proofs **in Distributed Systems**

Privacy:

- Cryptocurrencies that guarantee privacy TX on public Ledger
- **Zero Knowledge** Proof: the TX is valid without revealing more
    - ZCash, Aleo, Aztec, Midnight



Compliance:

- Prove that a private TX is compliant with banking laws
    - Esspresso Systems
- Prove that an exchange is solvent
  e.g. a company has more assets than obligations to its customers
- **Zero-Knowledge** to protect customers private data

ZK **Proofs in Distributed Systems**

Distributed storage:

- Peer-to-peer network for storage
- Allows anyone to store and retrieve data on the internet
- built-in economic incentives ensure that files are stored reliably and continuously
- **Proof of Knowledge** of the stored file
    - Filecoin

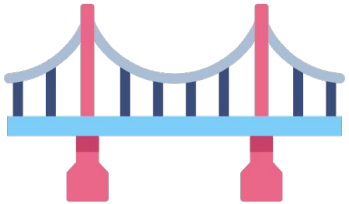
ZK **Proofs in Distributed Systems**

Distributed storage:

- Peer-to-peer network for storage
- Allows anyone to store and retrieve data on the internet
- built-in economic incentives ensure that files are stored reliably and continuously
- **Proof of Knowledge** of the stored file
    - Filecoin

Bridging Blockchains:

- Transfer of assets from a chain to another
- Source chain proves to target chain that the asset is locked-up
- No need for Zero Knowledge
- Need for **Proof of Knowledge**



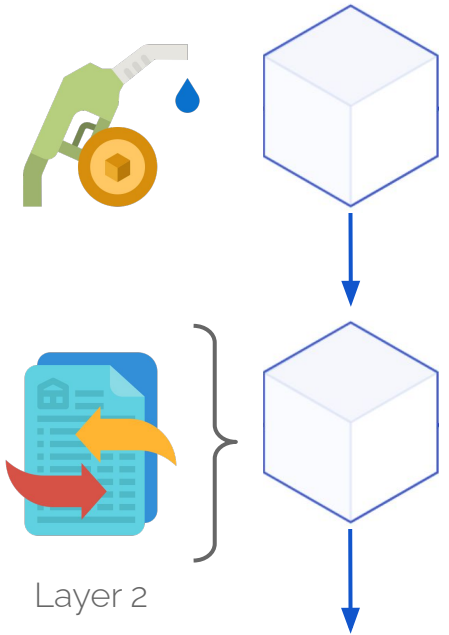
ZK **Proofs in Distributed Systems**

Rollups = Outsourced Computation

- Scalability tool
- Off-chain services (Layer 2) process a batch of TX (validates TX) and proves the work
- Main chain (Layer 1) verifies efficiently the work of an off-chain service
- Reduces the cost of transactions

- **No need** for Zero Knowledge
- Need for **Succinctness & Proof of Knowledge**

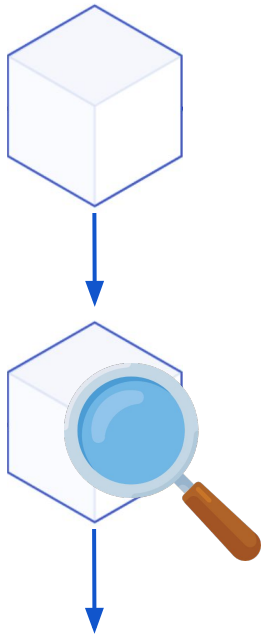
No zk: ZKSync, Loopring, Linea by Consensys, ZKSpace
With zk: Aztec, Polygon Hermez



ZK **Proofs Key Requirements**

Key Properties for Proofs in Blockchains

- **non-interactivity**
- publicly verifiable
- **succinctness for proofs**
- **efficiency for verifier**



Answer: SNARKs

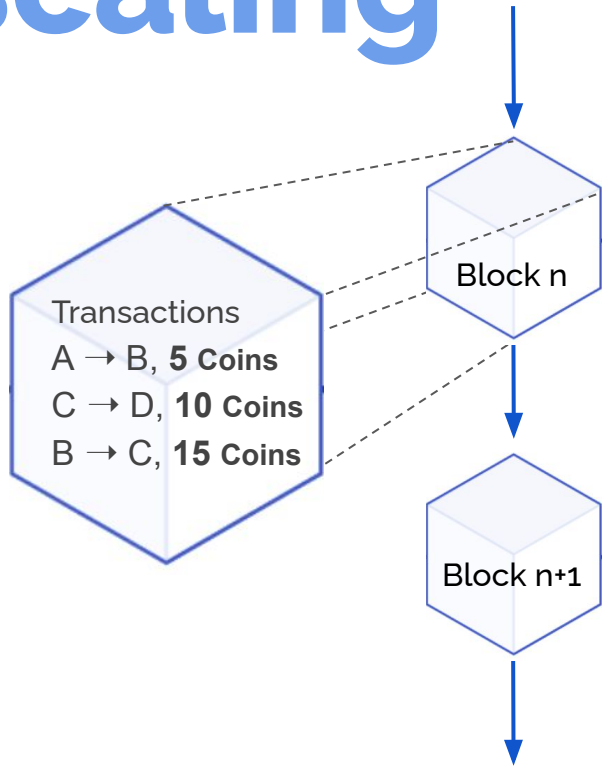
Succinct Non-Interactive Arguments of Knowledge

What is left?

Blockchain scaling

ZK Roll-ups

- instead of posting all transaction data on-chain
- execute **tx** using off-chain computation
- a summary of the state changes is published to the chain and verified



Blockchain scaling

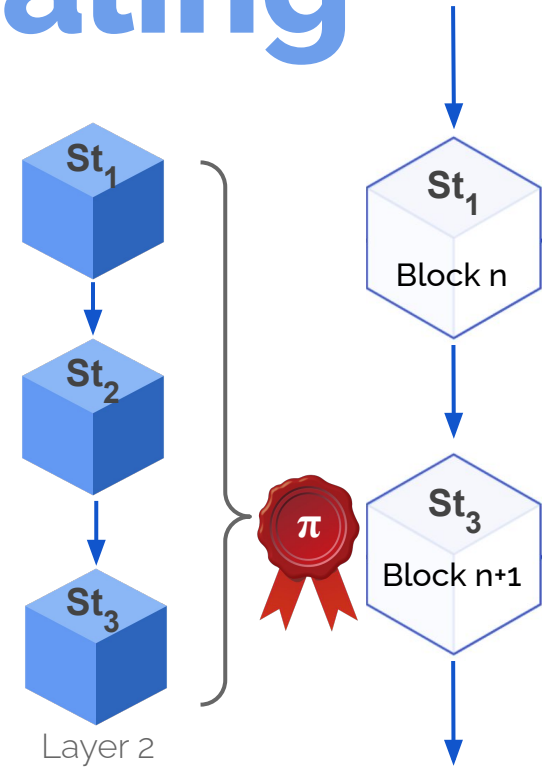
ZK Roll-ups

- instead of posting all transaction data on-chain
- execute **tx** using off-chain computation
- a summary of the state changes is published to the chain and verified

SNARK π proves the correctness of the changes

Statement: state changes proposed by the layer 2 are correct,

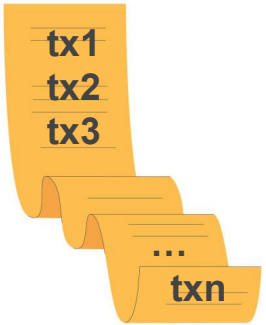
i.e. are the result of the execution of the given batch of transactions

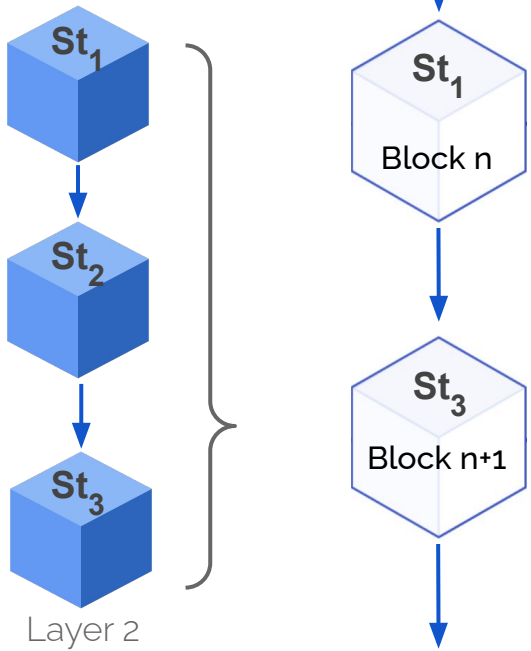


Blockchain scaling

ZK Roll-ups

Prove multiple instances:
- correctness of the execution of batch of **tx**



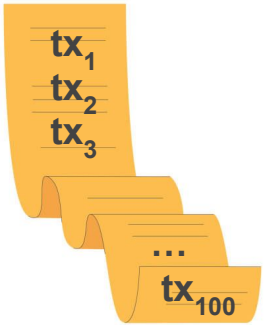


Blockchain scaling

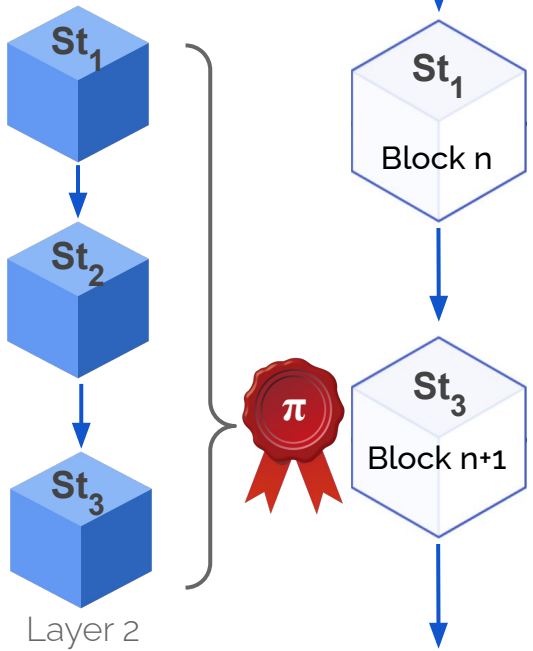
ZK Roll-ups

Example 100 **Tx**

Naively: generate state transition proof once all 100 Tx are submitted → **long delay**



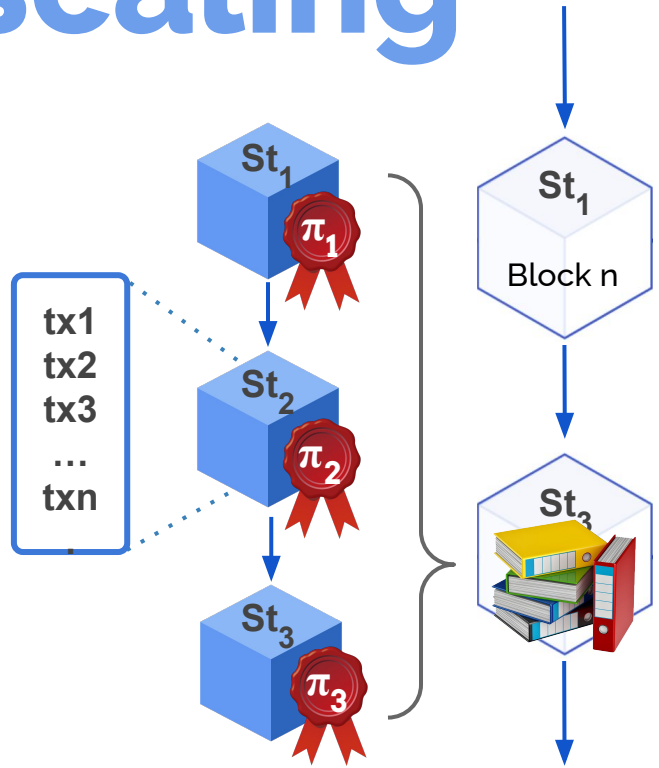
Proving is slow



Blockchain scaling

ZK Roll-ups

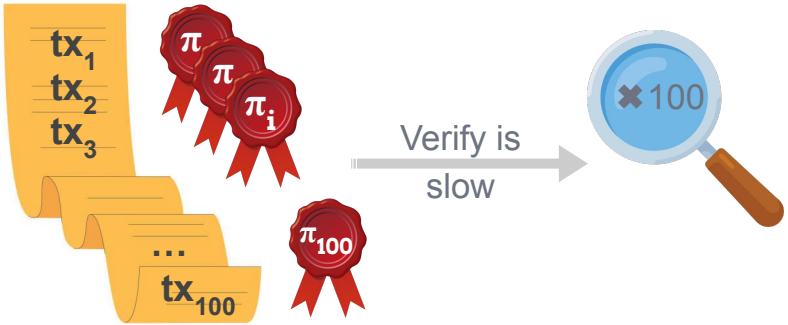
Periodically provide proofs for valid transaction batches

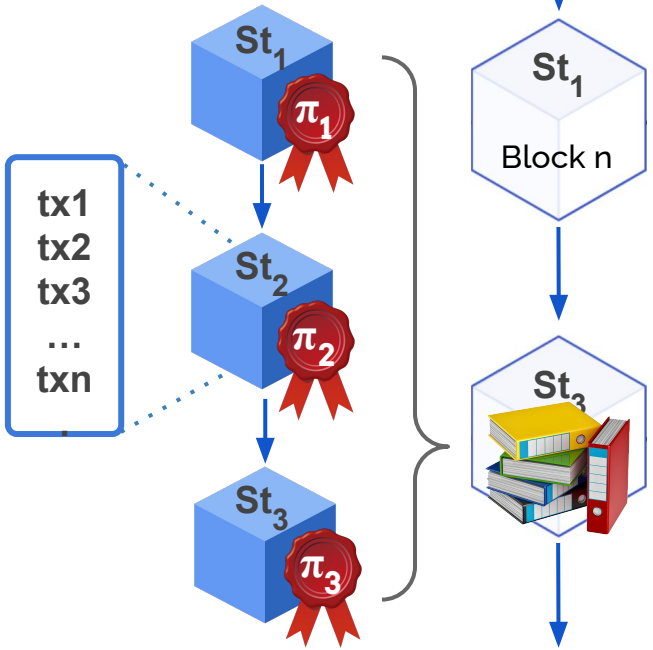


Blockchain scaling

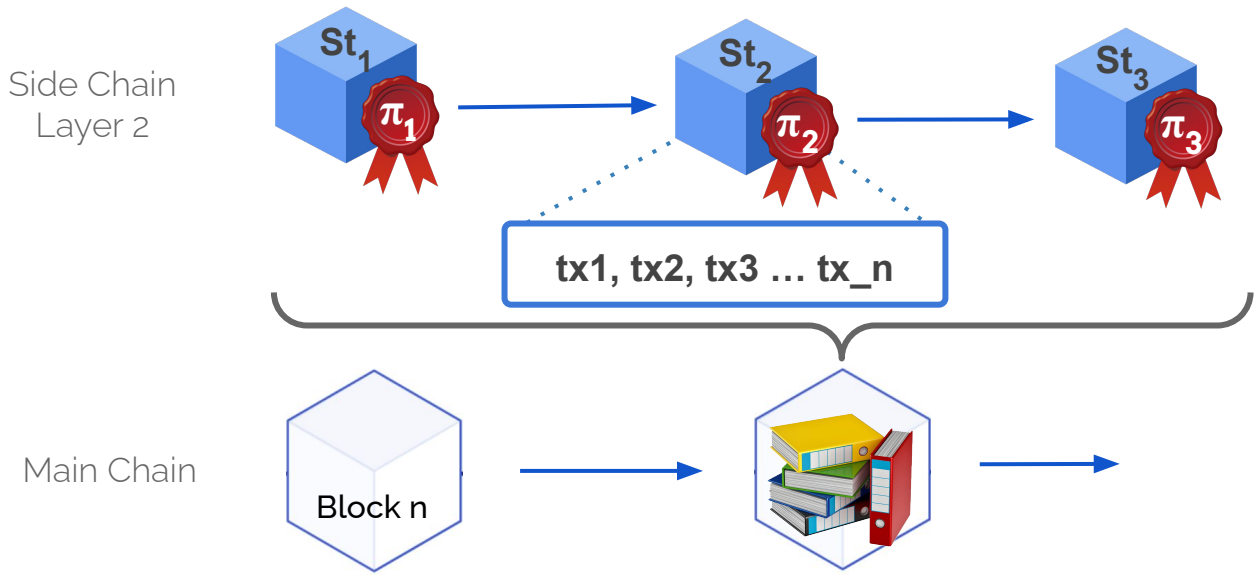
ZK Roll-ups

Periodically provide proofs for valid transaction batches

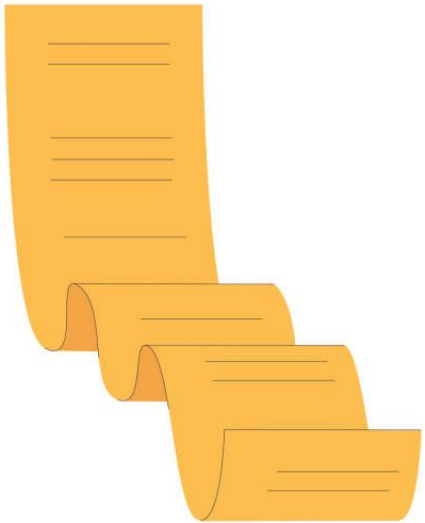




Prove & scale ?

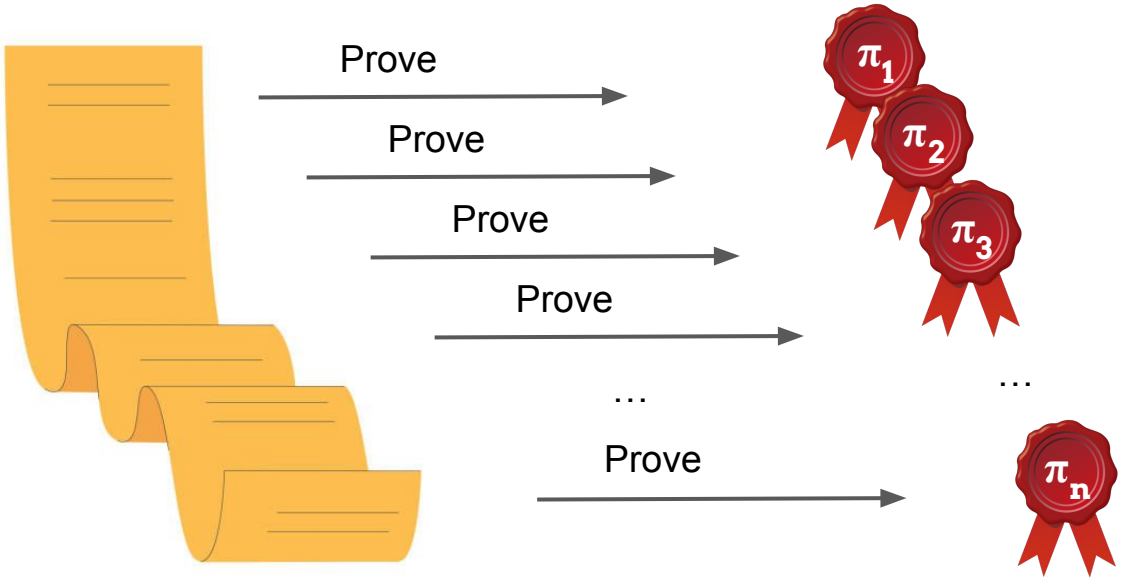

Side Chain
Layer 2
St1
π1
St2
π2
St3
π3
tx1, tx2, tx3 … tx_n
Main Chain
Block n

In Brief



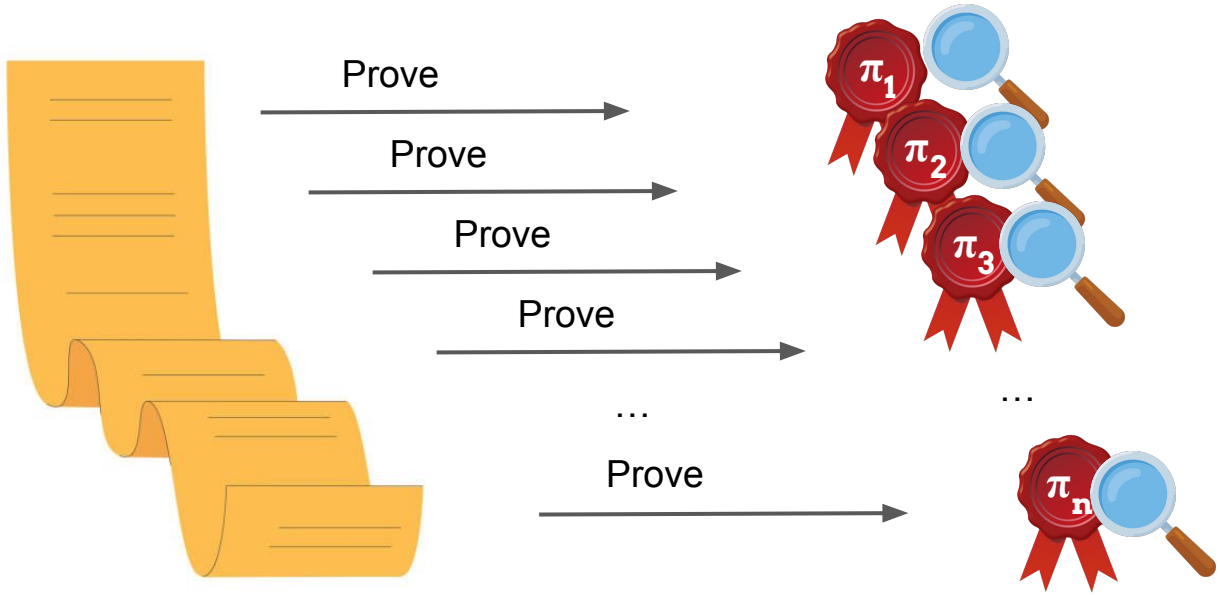
Multiple instances to prove

Naively



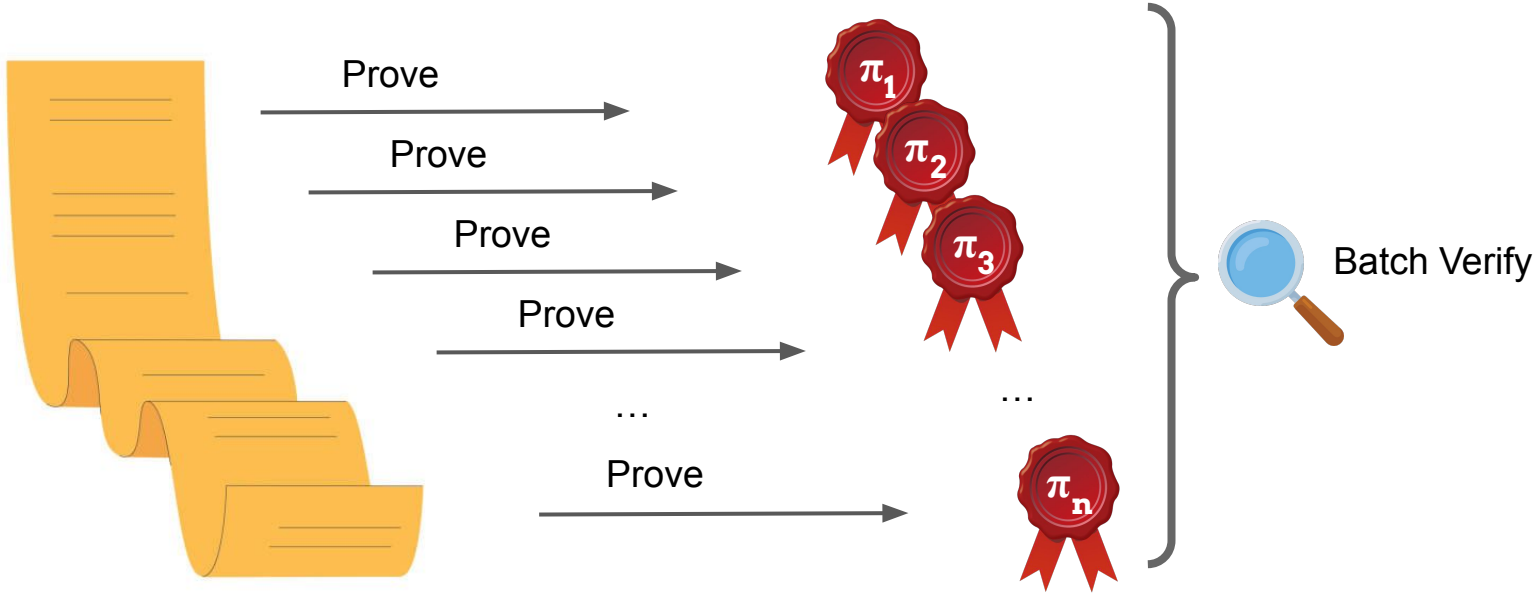
Multiple instances to prove

Naively



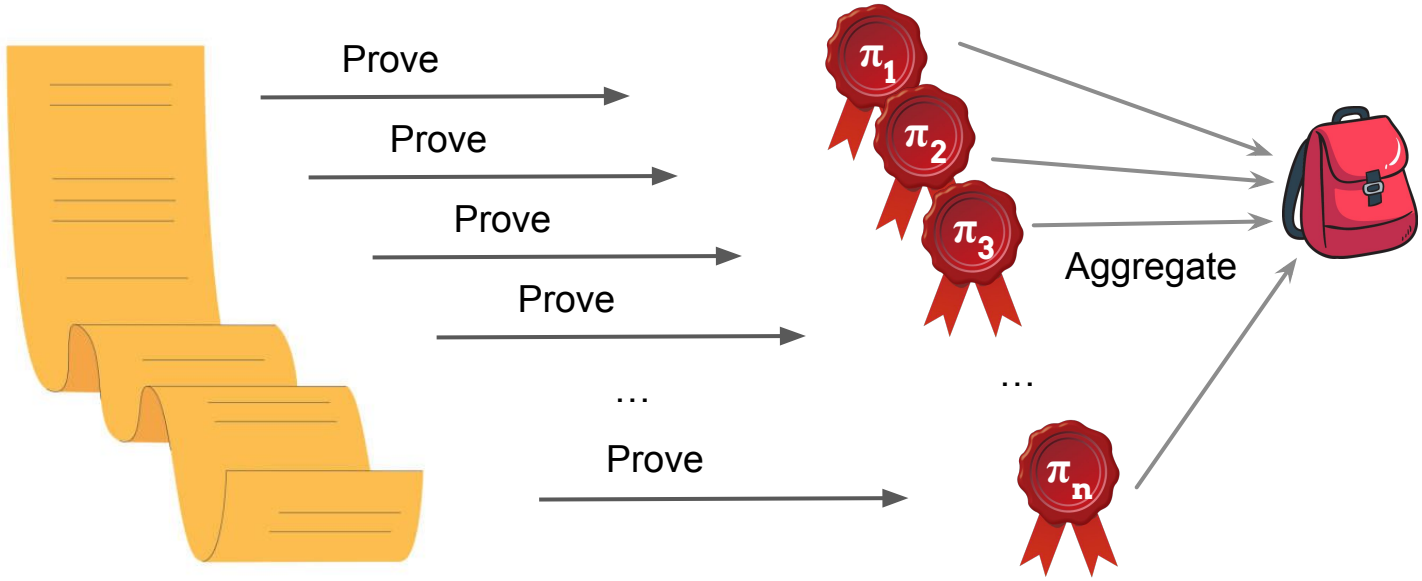
Multiple instances to prove

Batch Verify



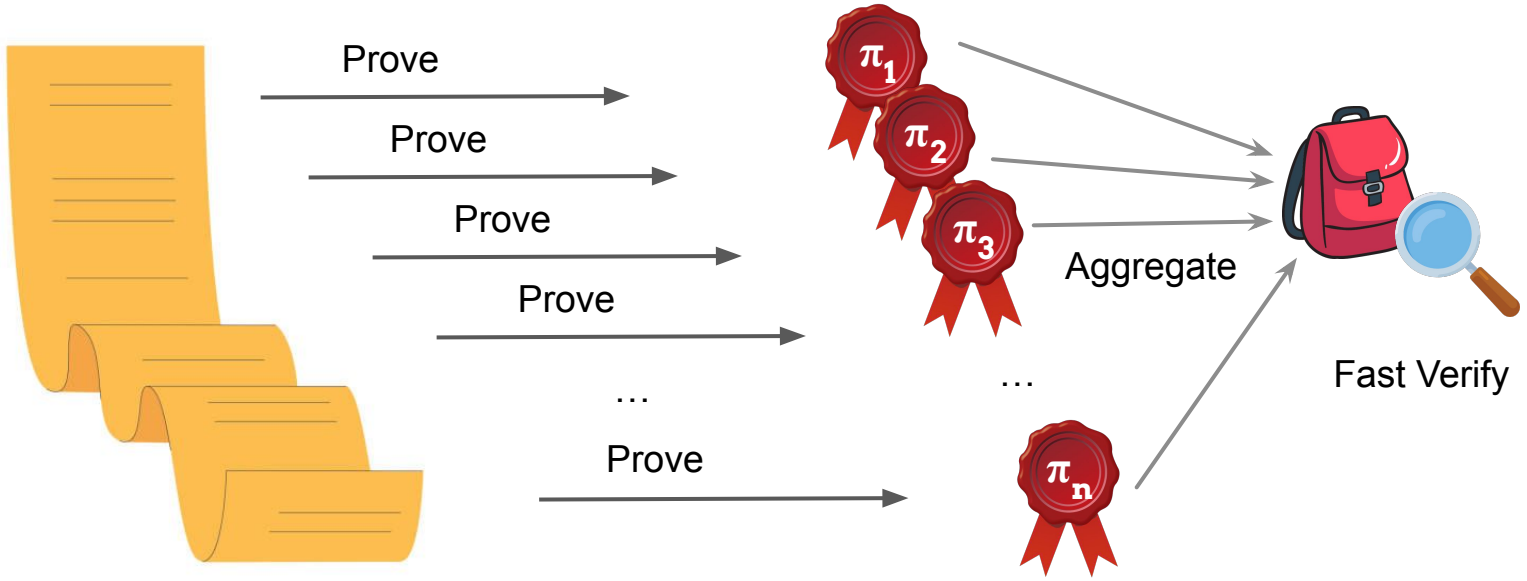
Multiple instances to prove

Aggregation



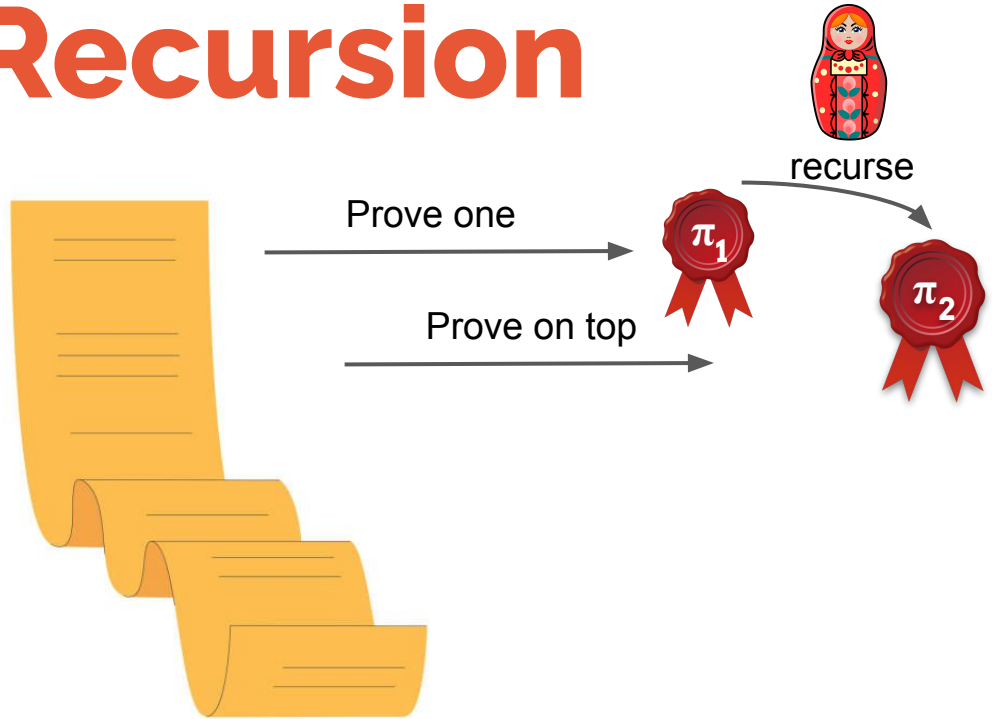
Multiple instances to prove

Aggregation



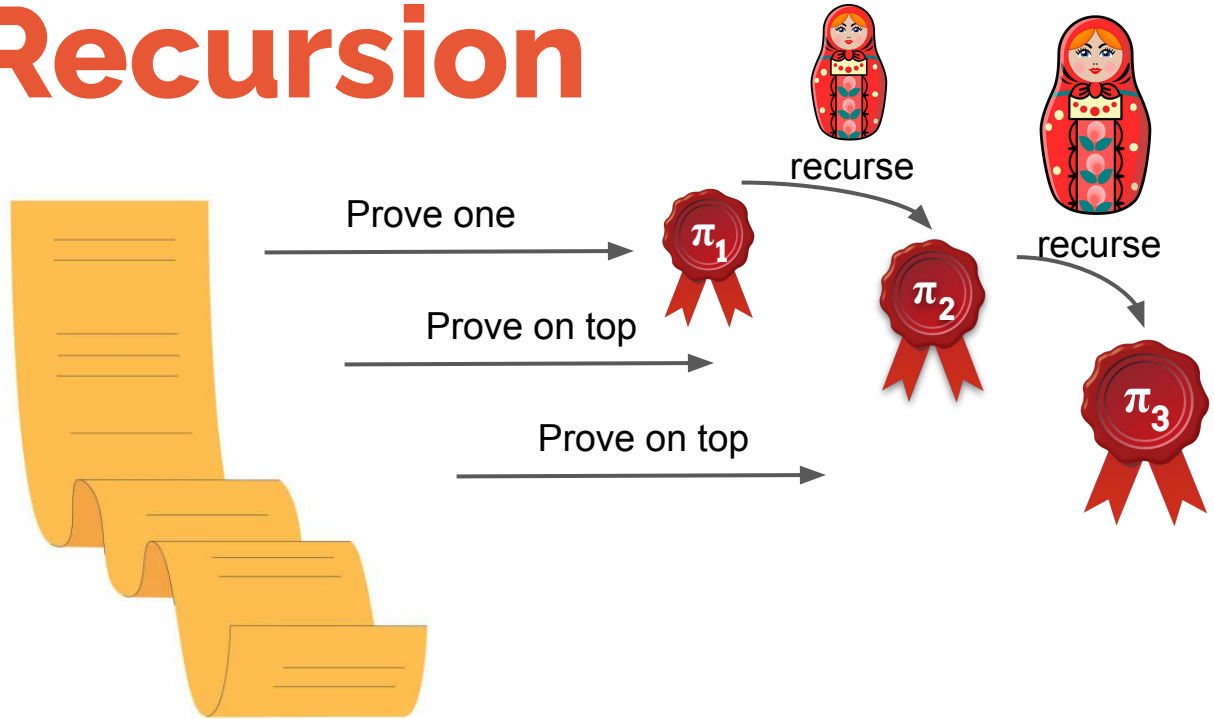
Multiple instances to prove

Recursion



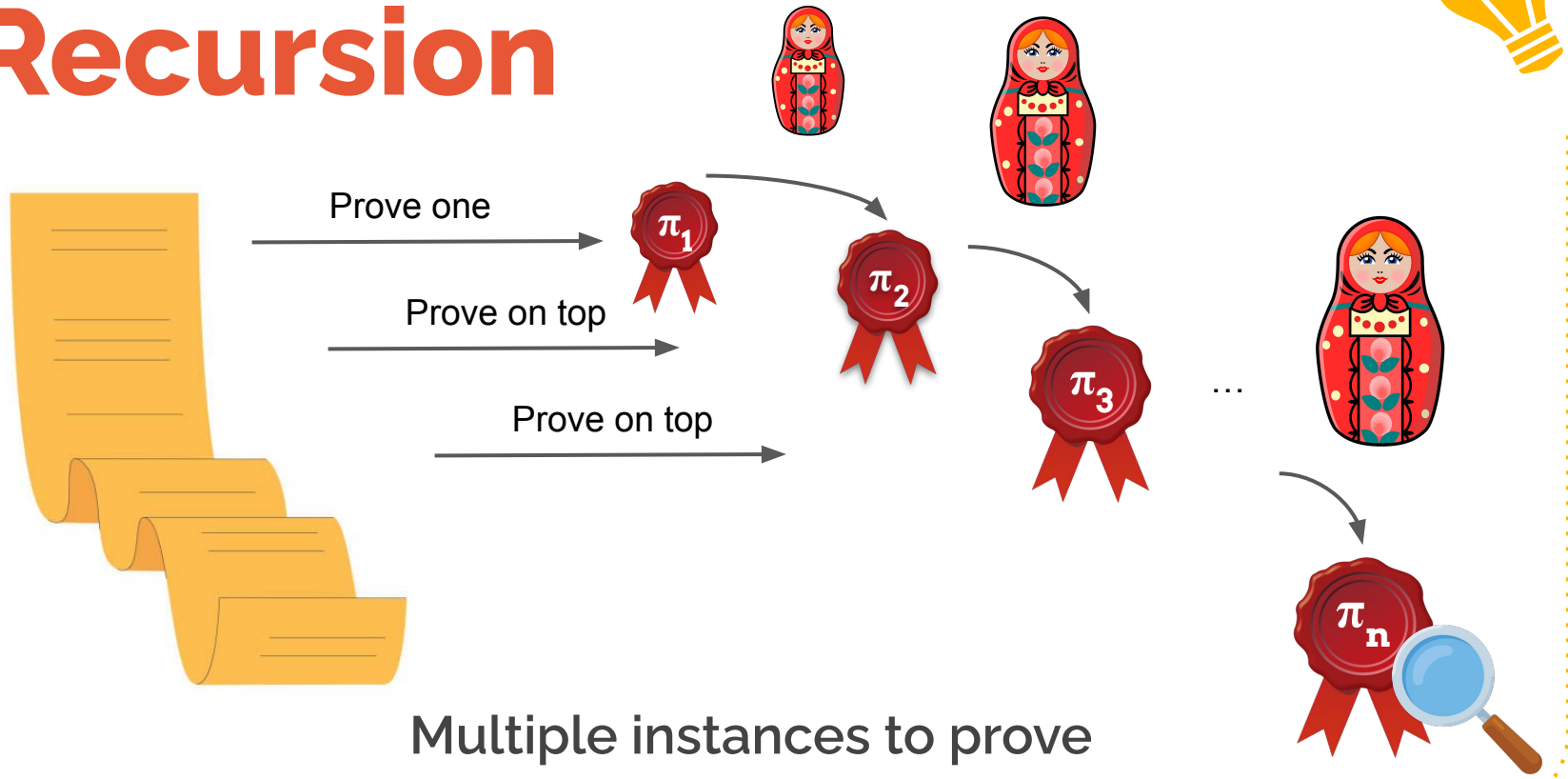
Multiple instances to prove

Recursion



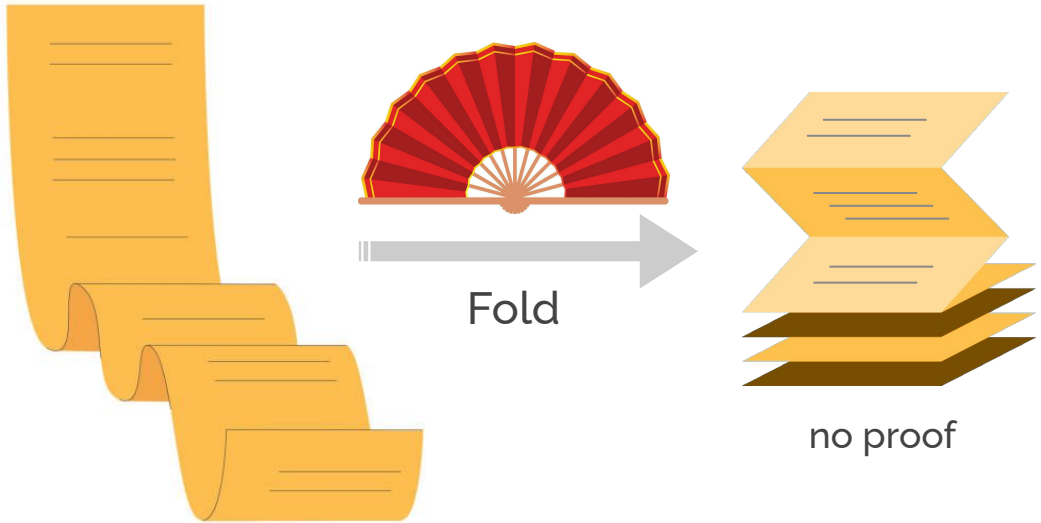
Multiple instances to prove

Recursion



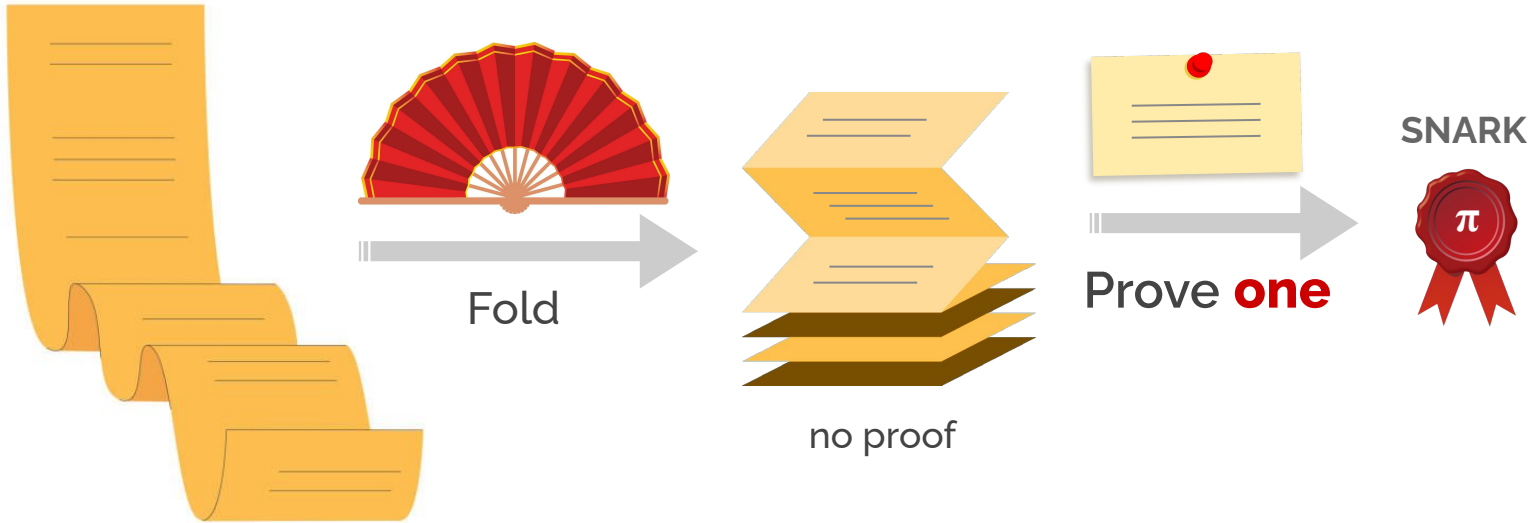
Multiple instances to prove

Folding



Multiple instances to prove

Folding



Multiple instances to prove

Prove many instances

Aggregation

Recursion

Folding

Proof Size

Verifier Time

Prover Time

SNARK Aggregation

Prove many instances

Aggregation

Proof Size

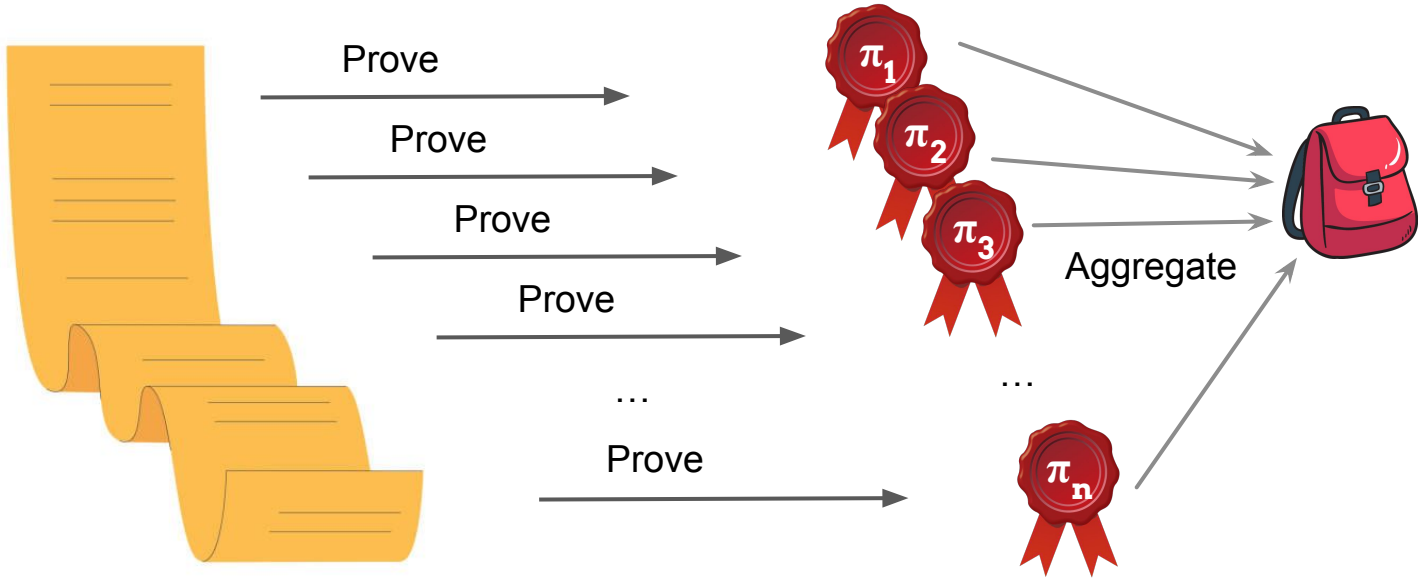
Verifier Time

Prover Time

Recursion

Folding

Aggregation



Multiple instances to prove

Distributed storage

Storage Providers

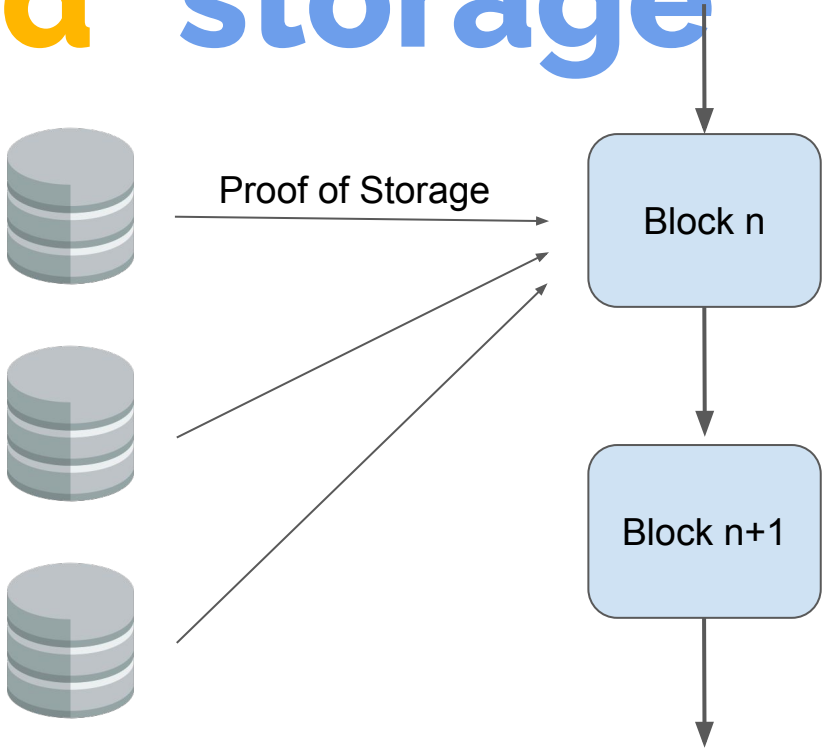
- onboard storage capacity
- earn block rewards
- regularly prove the storage

= **Provers**

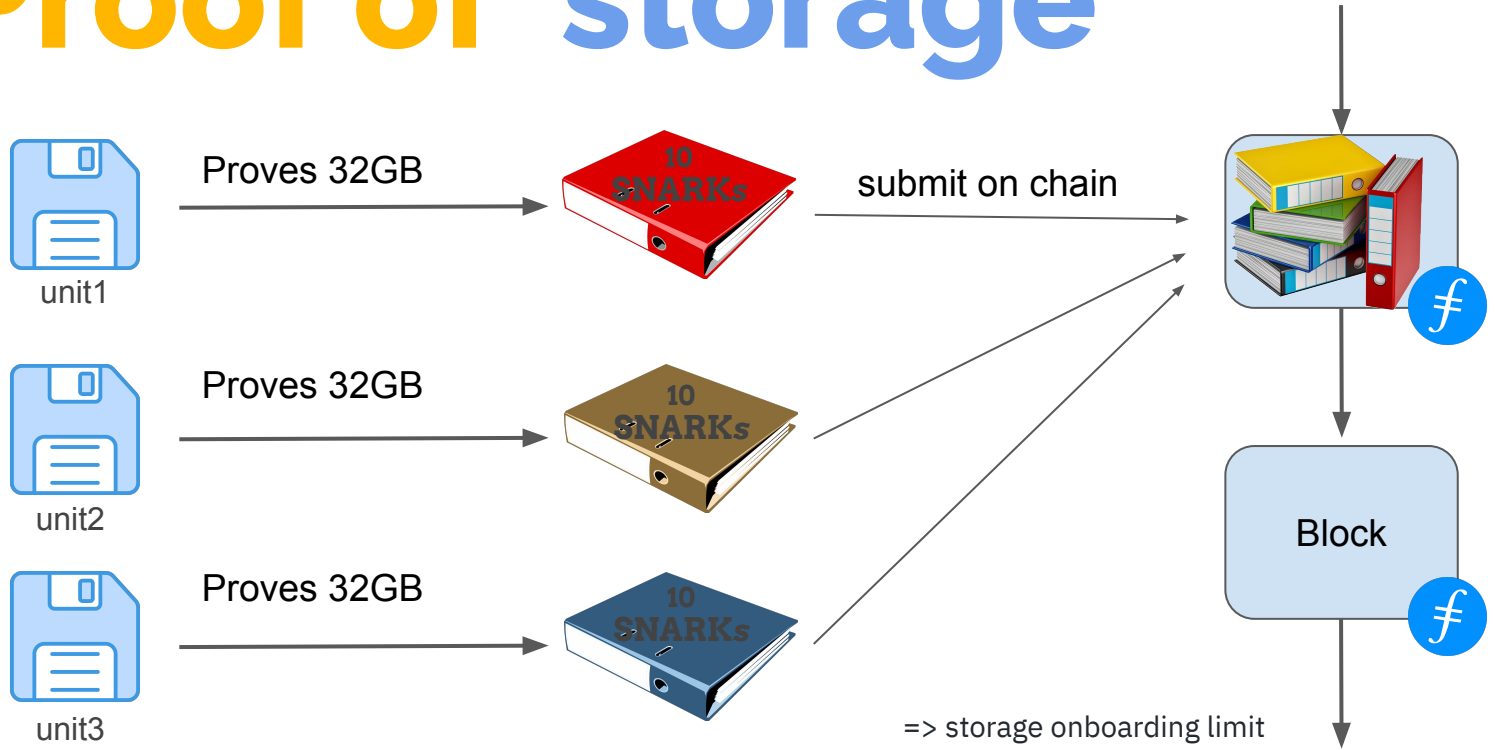
Nodes in network

- ensure data is being stored, maintained, and secured
- need to check proofs of space

= **Verifiers**



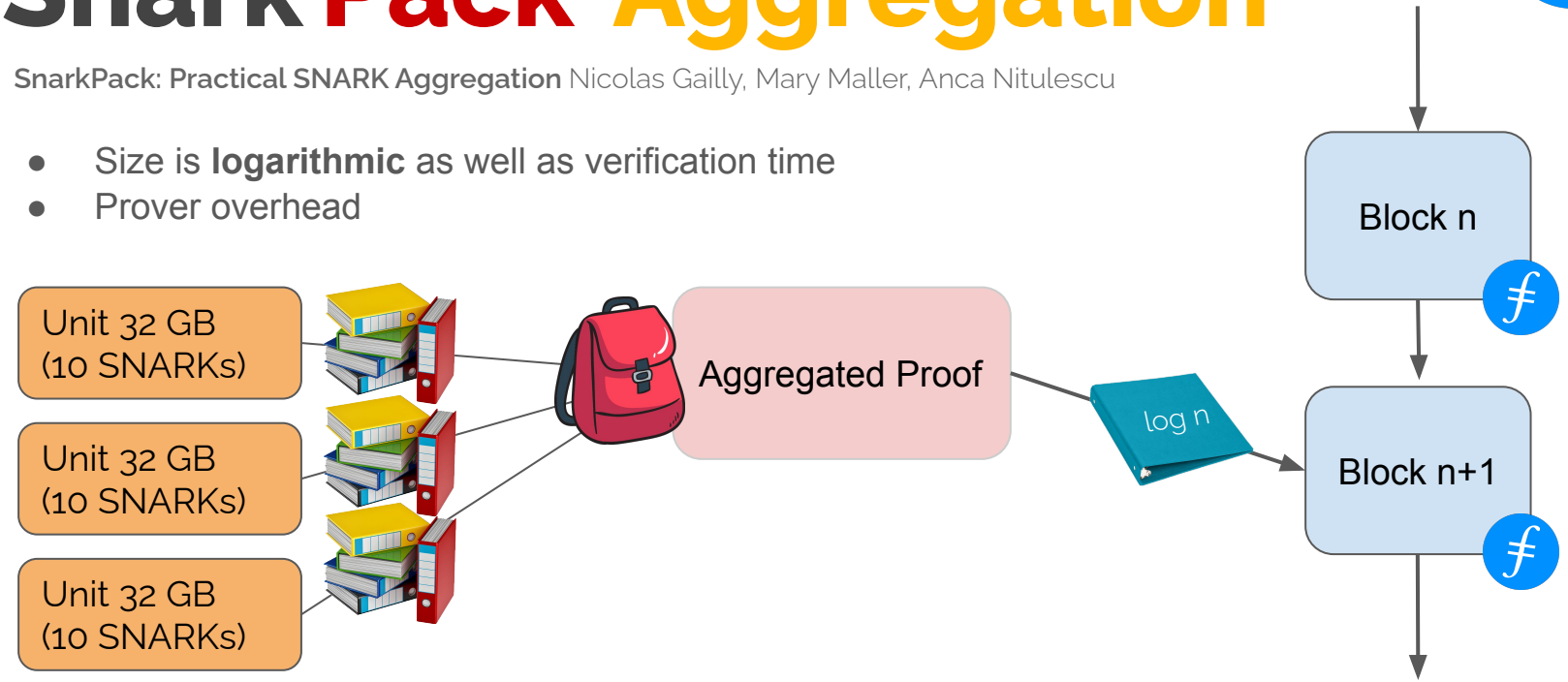
Proof of storage



Snark Pack Aggregation

SnarkPack: Practical SNARK Aggregation Nicolas Gailly, Mary Maller, Anca Nitulescu

- Size is **logarithmic** as well as verification time
- Prover overhead



Bilinear Groups

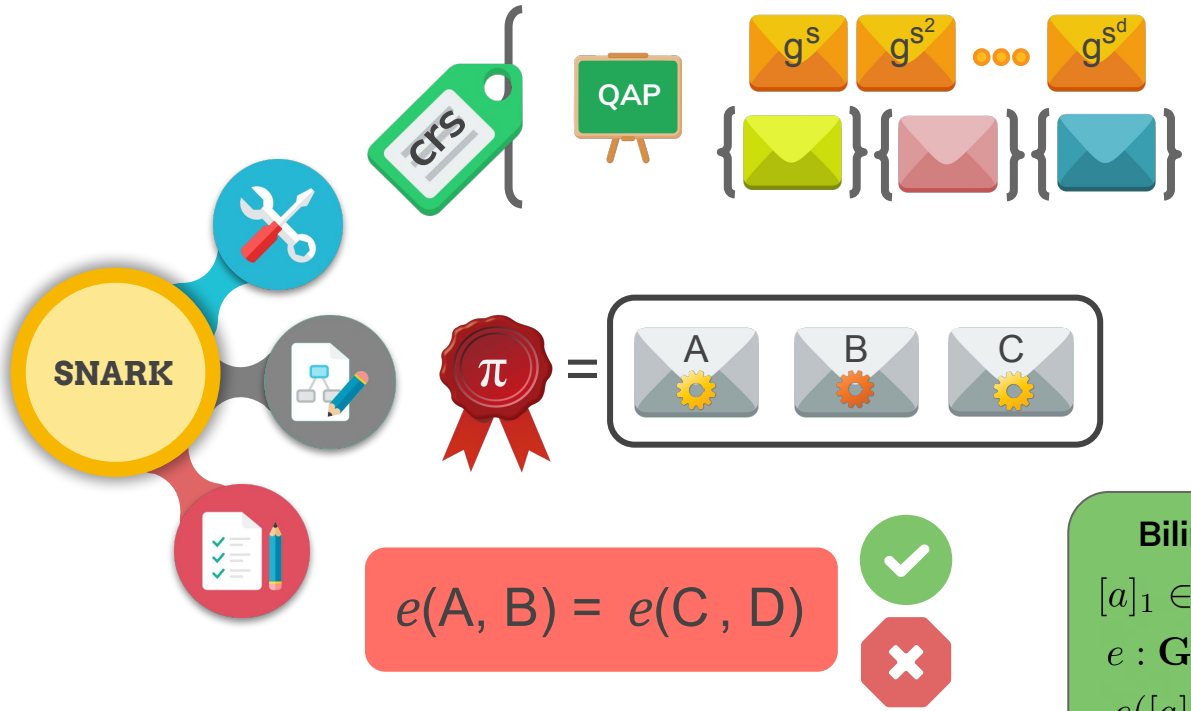
Bilinear Groups

$$[a]_1 \in \mathbf{G}_1, [b]_2 \in \mathbf{G}_2$$

$$e : \mathbf{G}_1 \times \mathbf{G}_2 \to \mathbf{G}_T$$

$$e([a]_1, [b]_2) = [ab]_T$$

Groth 16



Bilinear Groups

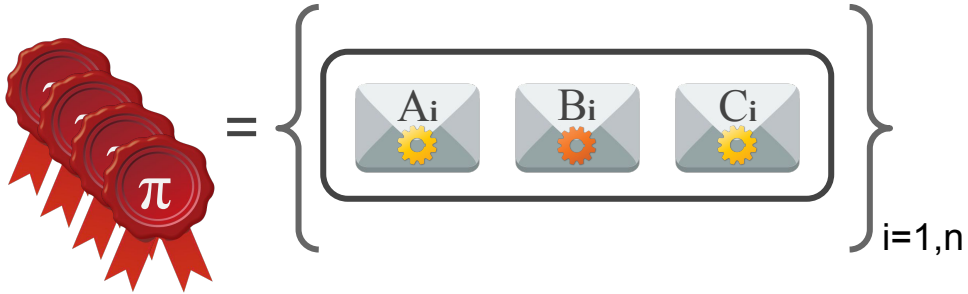
$[a]_1 \in \mathbf{G}_1, [b]_2 \in \mathbf{G}_2$

$e : \mathbf{G}_1 \times \mathbf{G}_2 \rightarrow \mathbf{G}_T$

$e([a]_1, [b]_2) = [ab]_T$

Many SNARKs

Proofs



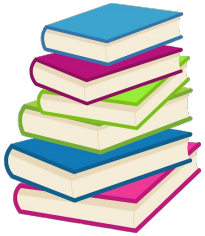
Verification ($D = g^d$)

$e(A_1, B_1) = e(C_1, D)$

$e(A_2, B_2) = e(C_2, D)$

...

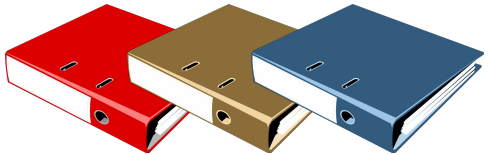
$e(A_n, B_n) = e(C_n, D)$



Verify many SNARKs

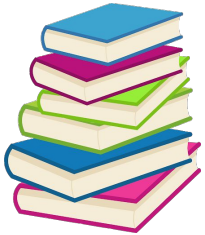
Batch Verification

Proof Size



Verification Time

SNARK Batching



Verification

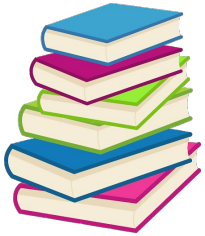
$e(A_1, B_1) = e(C_1, D)$

$e(A_2, B_2) = e(C_2, D)$

...

$e(A_n, B_n) = e(C_n, D)$

SNARK Batching



Verification

$e(A_1, B_1) = e(C_1, D)$

$e(A_2, B_2) = e(C_2, D)$

…

$e(A_n, B_n) = e(C_n, D)$

$r \leftarrow \$$

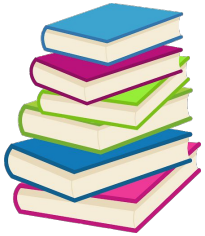
$r \times e(A_1, B_1) = e(C_1, D)$

$r^2 \times e(A_2, B_2) = e(C_2, D)$

…

$r^n \times e(A_n, B_n) = e(C_n, D)$

SNARK Batching



Verification

$e(A_1, B_1) = e(C_1, D)$

$e(A_2, B_2) = e(C_2, D)$

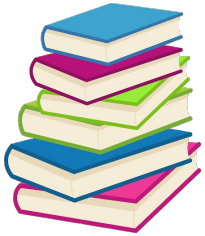
...

$e(A_n, B_n) = e(C_n, D)$

$r \leftarrow \$$

Batch Verification

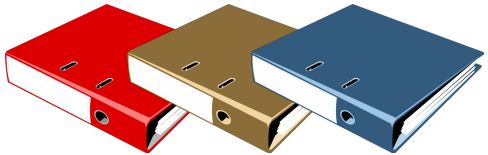
$$\sum r^i e(A_i, B_i) = \sum r^i e(C_i, D)$$



Verify many SNARKs

Batch Verification

Proof Size



Verification Time

Aggregation

Proof Size

Verification Time

SNARK Aggregation

SnarkPack: Practical SNARK Aggregation Nicolas Gailly, Mary Maller, Anca Nitulescu

Batch Verification

$$\sum e(A_i, r^i B_i) = \sum e(r^i C_i, D)$$

$$\Leftrightarrow$$

$$\sum e(A_i, r^i B_i) = e(\sum r^i C_i, D)$$

Bilinear Groups

$$[a]_1 \in \mathbb{G}_1, [b]_2 \in \mathbb{G}_2$$

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$$

$$e([a]_1, [b]_1) = [ab]_T$$

SNARK Aggregation

SnarkPack: Practical SNARK Aggregation Nicolas Gailly, Mary Maller, Anca Nitulescu

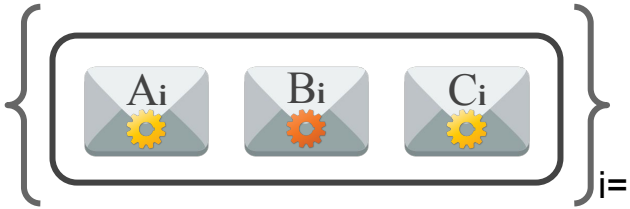
Batch Verification

$$\sum e(A_i, r^i B_i) = \sum e(r^i C_i, D)$$

$$\Leftrightarrow$$

$$\sum e(A_i, r^i B_i) = e(\sum r^i C_i, D)$$

Aggregation



SNARK Aggregation

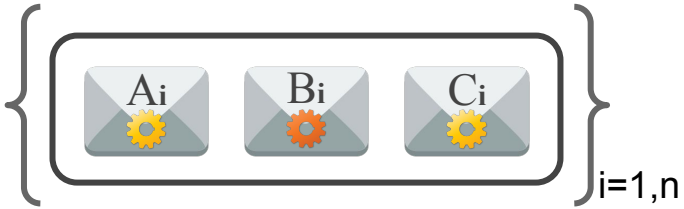
Batch Verification

$$\sum e(A_i, r^i B_i) = \sum e(r^i C_i, D)$$

$$\Leftrightarrow$$

$$\sum e(A_i, r^i B_i) = e(\sum r^i C_i, D)$$

Aggregation



$$Z_{AB} = \sum e(A_i, r^i B_i)$$

$$Z_C = \sum r^i C_i$$

SNARK Aggregation

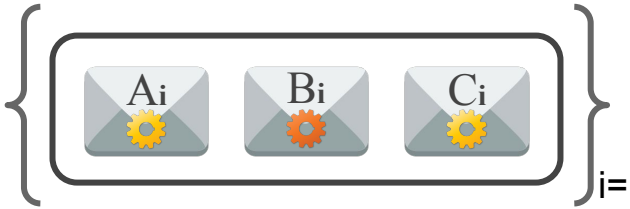
Batch Verification

$$\sum e(A_i, r^i B_i) = \sum e(r^i C_i, D)$$

$$\Leftrightarrow$$

$$Z_{AB} = e(Z_C, D)$$

Aggregation



$$Z_{AB} = \sum e(A_i, r^i B_i)$$

$$Z_C = \sum r^i C_i$$

SNARK Aggregation

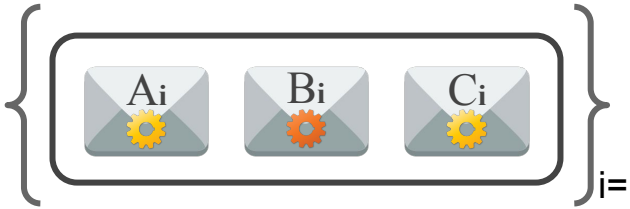
Batch Verification

$$\sum e(A_i, r^i B_i) = \sum e(r^i C_i, D)$$

$$\Leftrightarrow$$

$$Z_{AB} = e(Z_C, D)$$

Aggregation



$$Z_{AB} = \sum e(A_i, r^i B_i)$$

$$Z_C = \sum r^i C_i$$

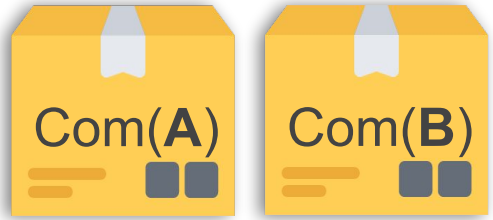
Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

$\langle \mathbf{A}, \mathbf{r} \rangle = \sum r_i A_i$

$\langle \mathbf{A}, \mathbf{B} \rangle = \sum e(A_i, B_i)$

$A_i \in \mathbb{G}_1, B_i \in \mathbb{G}_2, r_i \in \mathbb{Z}_q$



Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

$$\langle \mathbf{A}, \mathbf{r} \rangle = \sum r_i A_i = \sum [a_i, r_i]_1$$

$$\langle \mathbf{A}, \mathbf{B} \rangle = \sum e(A_i, B_i) = [\sum a_i b_i]_T$$

$$A_i \in \mathbb{G}_1, B_i \in \mathbb{G}_2, r_i \in \mathbb{Z}_q$$

Tools: MIPP & TIPP

Proofs for Inner Pairing Products and Applications - Bünz, Maller, Mishra, Tyagi, Vesely

$$\langle \mathbf{C}, \mathbf{r} \rangle = \sum r_i A_i$$

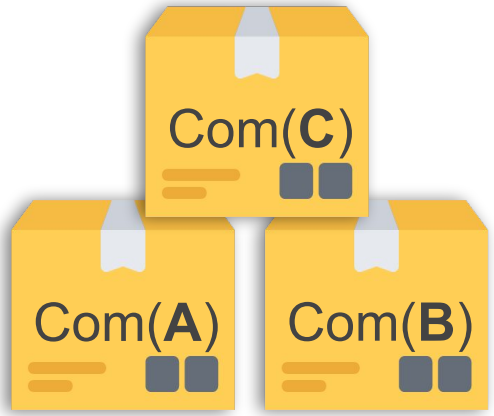
$$\langle \mathbf{A}, \mathbf{rB} \rangle = \sum e(A_i, B_i)$$

$$Z_C = \langle \mathbf{C}, \mathbf{r} \rangle$$

$$Z_{AB} = \langle \mathbf{A}, \mathbf{rB} \rangle$$

Aggregation

MIPP & TIPP Strategy



$Z_C = \langle C, r \rangle$

$Z_{AB} = \langle A, rB \rangle$

TIPP Folding Strategy

$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$$

$$\mathbf{A} = (A_1, A_2, \ldots A_n)$$

$$\mathbf{B} = (B_1, B_2 \ldots B_n)$$

$\mathbf{A}_{left}$ $\mathbf{A}_{right}$ $\mathbf{B}_{left}$ $\mathbf{B}_{right}$

Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$$

$$\mathbf{A} = (A_1, A_2, \ldots A_n) \qquad \mathbf{B} = (B_1, B_2 \ldots B_n)$$

$\mathbf{A}_{left}$ $\mathbf{A}_{right}$ $\mathbf{B}_{left}$ $\mathbf{B}_{right}$

$\mathbf{A'} = (A'_1, \ldots A'_{n/2})$ $\mathbf{B'} = (B'_1, \ldots B'_{n/2})$

Split & Collapse

$$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$$

$$\mathbf{A} = (A_1, A_2, \ldots A_n)$$

$$\mathbf{B} = (B_1, B_2 \ldots B_n)$$

$$\mathbf{A'} = (A'_1, \ldots A'_{n/2})$$

$$\mathbf{B'} = (B'_1, \ldots B'_{n/2})$$

Split & Collapse

$\langle \mathbf{A}, \mathbf{B} \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$

$\mathbf{A} = (A_1, A_2, \ldots A_n)$

$\mathbf{B} = (B_1, B_2 \ldots B_n)$

$\langle \mathbf{A'}, \mathbf{B'} \rangle$

$\mathbf{A'} = (A'_1, \ldots A'_{n/2})$

$\mathbf{B'} = (B'_1, \ldots B'_{n/2})$

Split & Collapse

Split $A = (A_{left}, A_{right})$

$B = (B_{left}, B_{right})$

$Z'_{A,B} = \langle A_{half}, B_{half} \rangle$

$L = \langle A_{right}, B_{left} \rangle \qquad R = \langle A_{left}, B_{right} \rangle$

$Z'_{A,B} = L + \langle A, B \rangle + Rx$

$\langle A, B \rangle = e(A_1, B_1) + e(A_2, B_2) \ldots + e(A_n, B_n)$

Linear Verification

$Z_{AB}=\langle A, B\rangle$

$A=(A_1, A_2, \ldots A_n)$ → $A'=A_{left}A_{right}$ → ... → A_{final}

Linear Verification

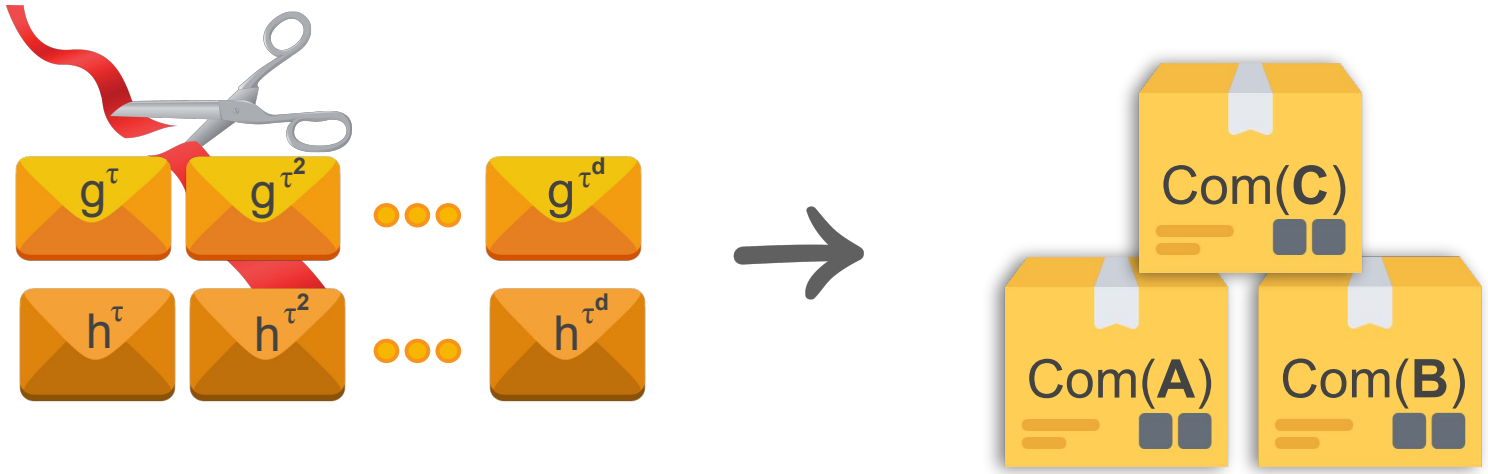
$Z_{AB}=\langle A, B\rangle$

$A=(A_1, A_2, \ldots A_n)$

$A'=A_{left}A_{right}$

A_{final}

Structured Setup



Vector Commitment

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$ 

$= [A(\tau)]_1 = A$

$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

Vector Commitment

KeyGen(λ, n) $\rightarrow$ ck: $[1]_1, [\tau]_1, [\tau^2]_1, \ldots [\tau^{n-1}]_1$

Commit(ck, **a**) $\rightarrow$  $= [A(\tau)]_1 = A$

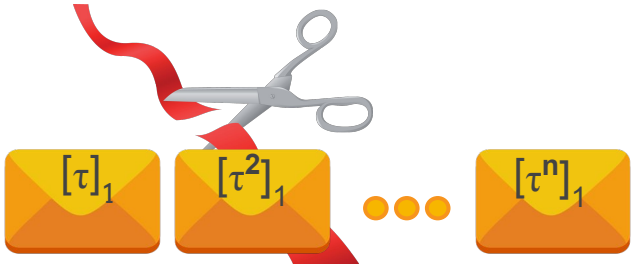
$A(X) = a_1 + a_2X + a_3X^2 + \ldots a_nX^{n-1}$

Target-Group Commitment

A=  ... $\rightarrow$

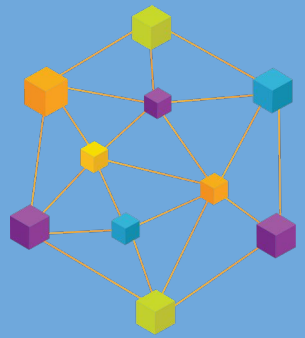
$Com(\mathbf{A}) = e(A_1,[\tau]_2)e(A_2,[\tau^2]_2)\ldots e(A_n,[\tau^n]_2)$

Structured Setup



Logarithmic verification for folding commitment key **ck**

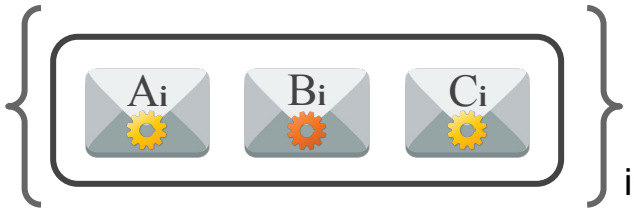
Fast verification ck_{final}: $g_\alpha(X) = \prod_{i=1}^{\mu} \left(1 + \alpha_{\mu+1-i}^{-2} X^{2^{i-1}}\right)$



Performance

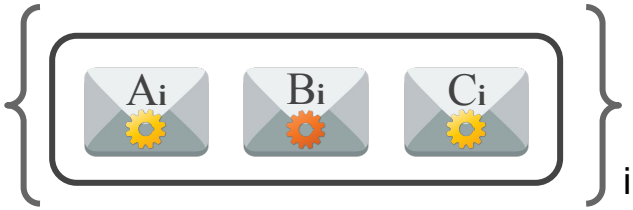
SNARK Aggregation

Aggregation



SNARK Aggregation

Aggregation



→

Com(**A, B**)

Com(**C**)

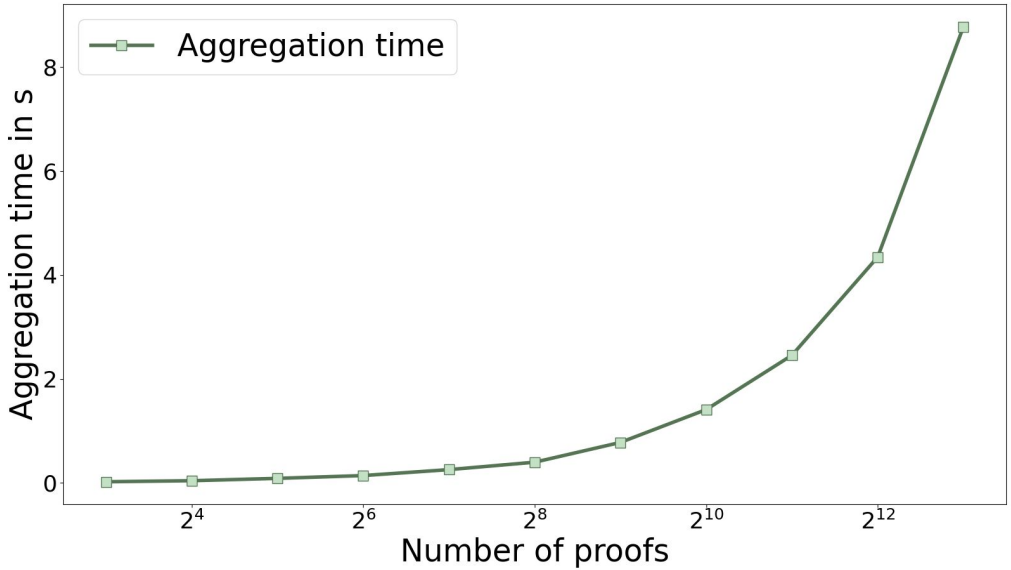
MIPP: $\langle \mathbf{C}, \mathbf{r} \rangle = \prod C_i^{r_i}$

TIPP: $\langle \mathbf{A}, \mathbf{B}^{\mathbf{r}} \rangle = \prod e(A_i, B_i^{r_i})$

→

log n

Aggregation

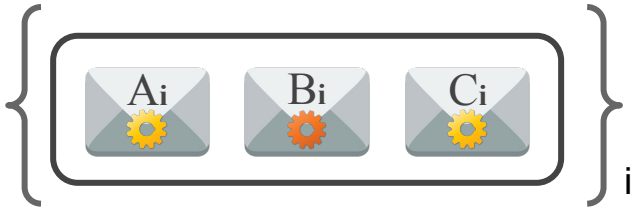


- 8.7s for 8 192 proofs
- Relies heavily on parallelism

- 2^{17} proofs in ~2mn

Proof size

Aggregation



→ Com(**A, B**) Com(**C**)

MIPP: $\langle \mathbf{C}, \mathbf{r} \rangle = \prod C_i^{r_i}$

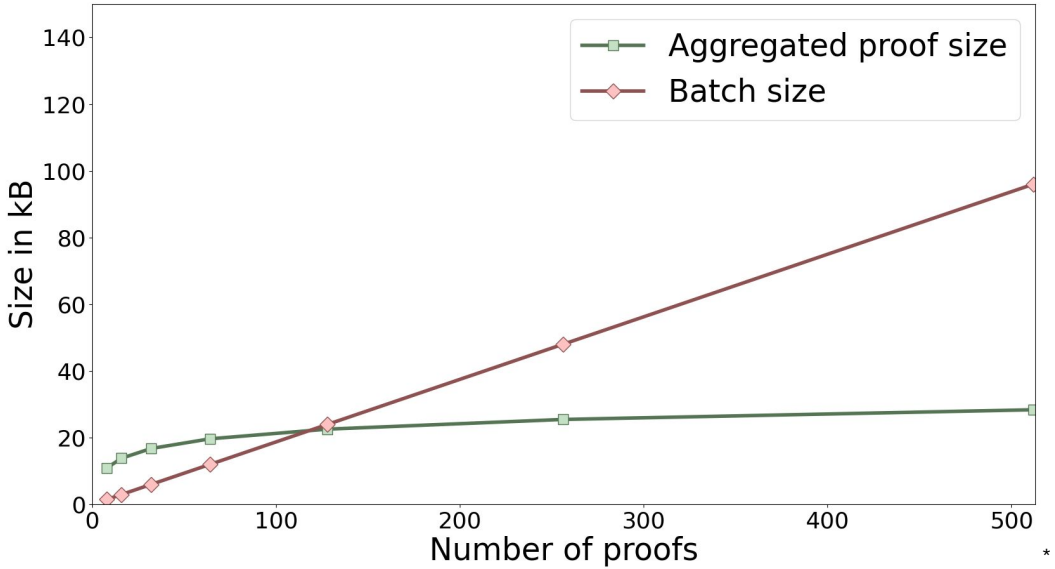
TIPP: $\langle \mathbf{A}, \mathbf{B}^{\mathbf{r}} \rangle = \prod e(A_i, B_i^{r_i})$

→

log n $Z_C = \langle \mathbf{C}, \mathbf{r} \rangle$

$Z_{AB} = \langle \mathbf{A}, \mathbf{B}^{\mathbf{r}} \rangle$

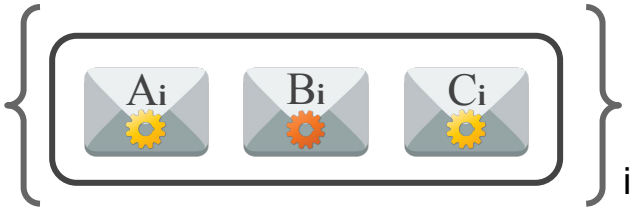
Proof size



- Use **compression** of target group points
  - based on Torus compression
  - credits - RELIC library implementation
- Turnover at 128 proofs
  - 23kB for aggregated
  - 24kB for “all proofs”

Verification

Aggregation



$Z_C = \langle C, r \rangle$

$Z_{AB} = \langle A, B^r \rangle$

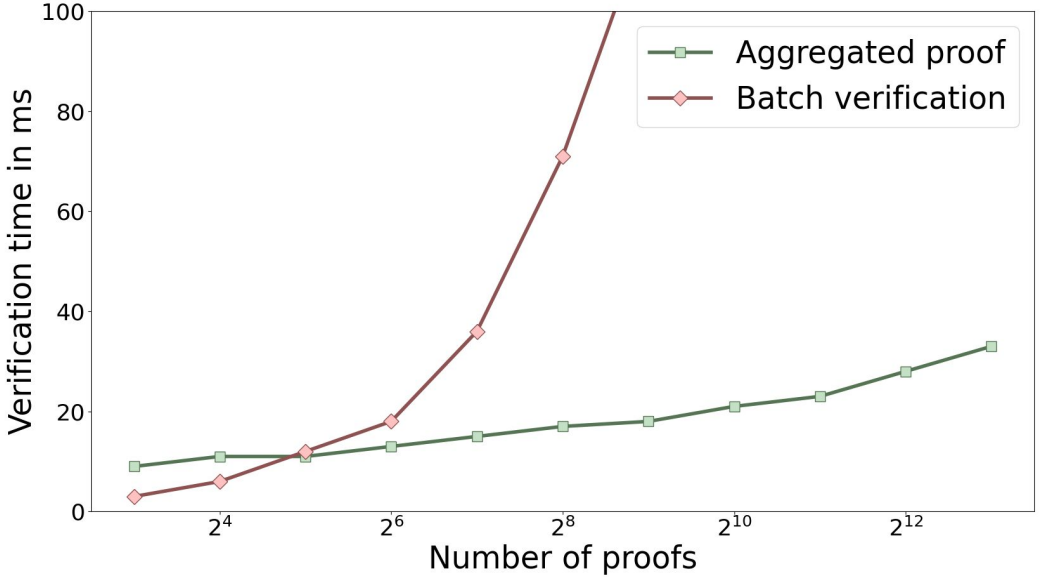
Verification

MIPP + TIPP

ck_{final}

$Z_{AB} = e(Z_C, D)$

Verifier Time



- Verifying aggregate proofs becomes **faster** from **32 proofs**
- **8192 proofs** in **33ms**
  - "ratio" of 0.004 ms per proof
- Optimizations:
  - Relies heavily on parallelism
  - MIPP/TIPP combined
  - Batching for pairing checks

SNARK Recursion

Prove many instances

Snark Pack

Recursion

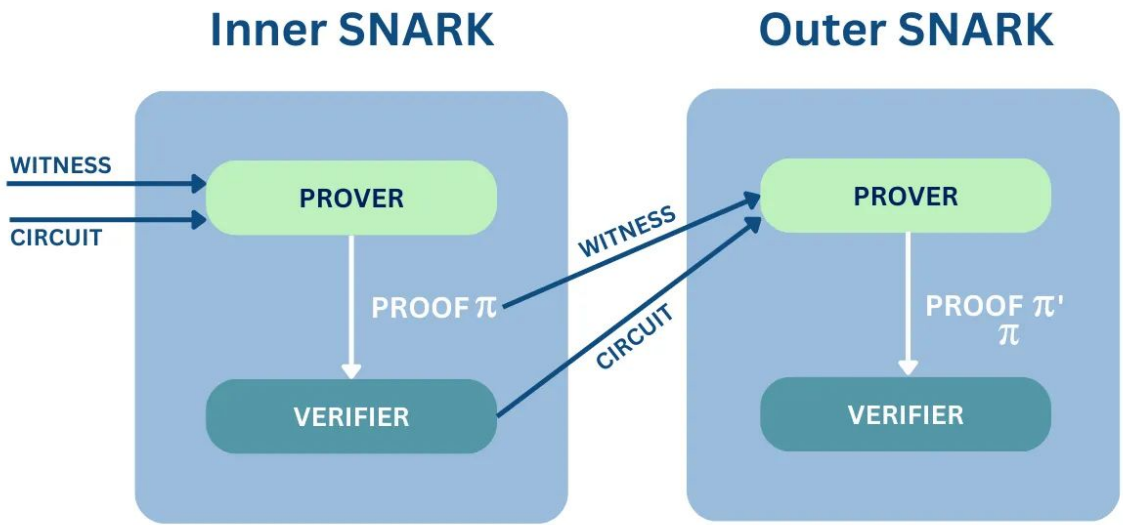
Folding

Proof Size

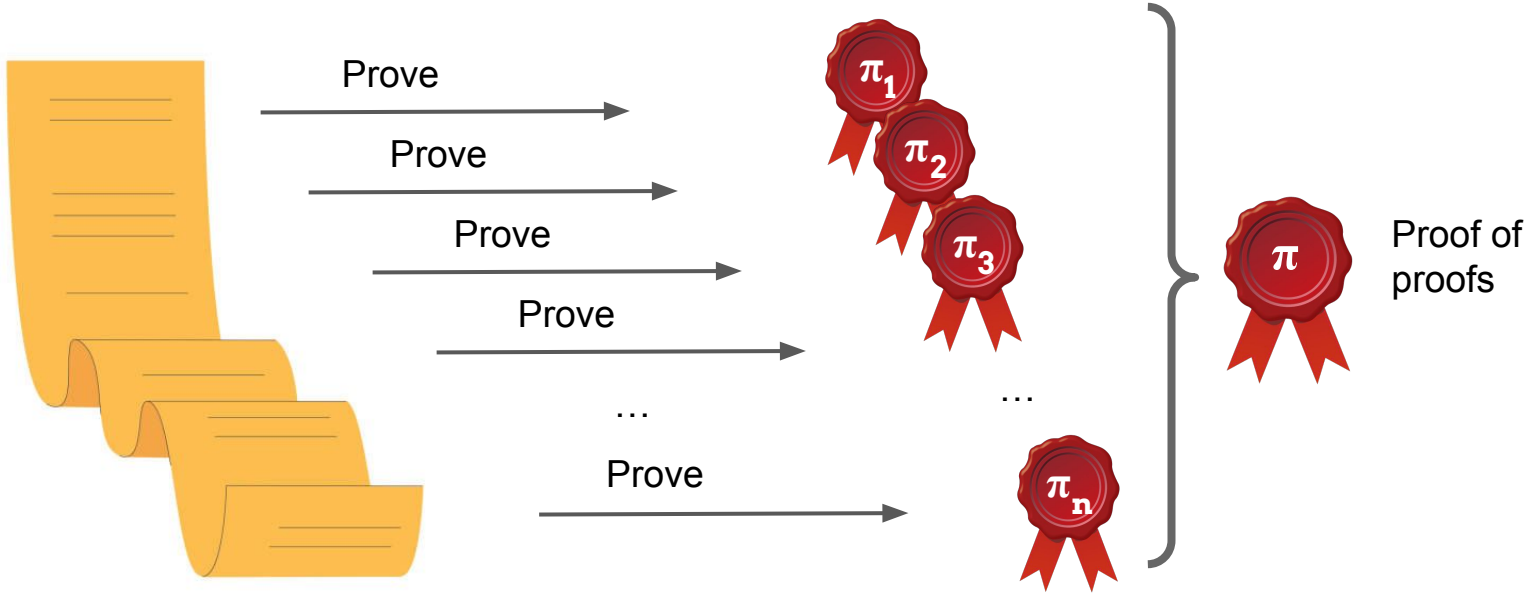
Verifier Time

Prover Time

Recursion

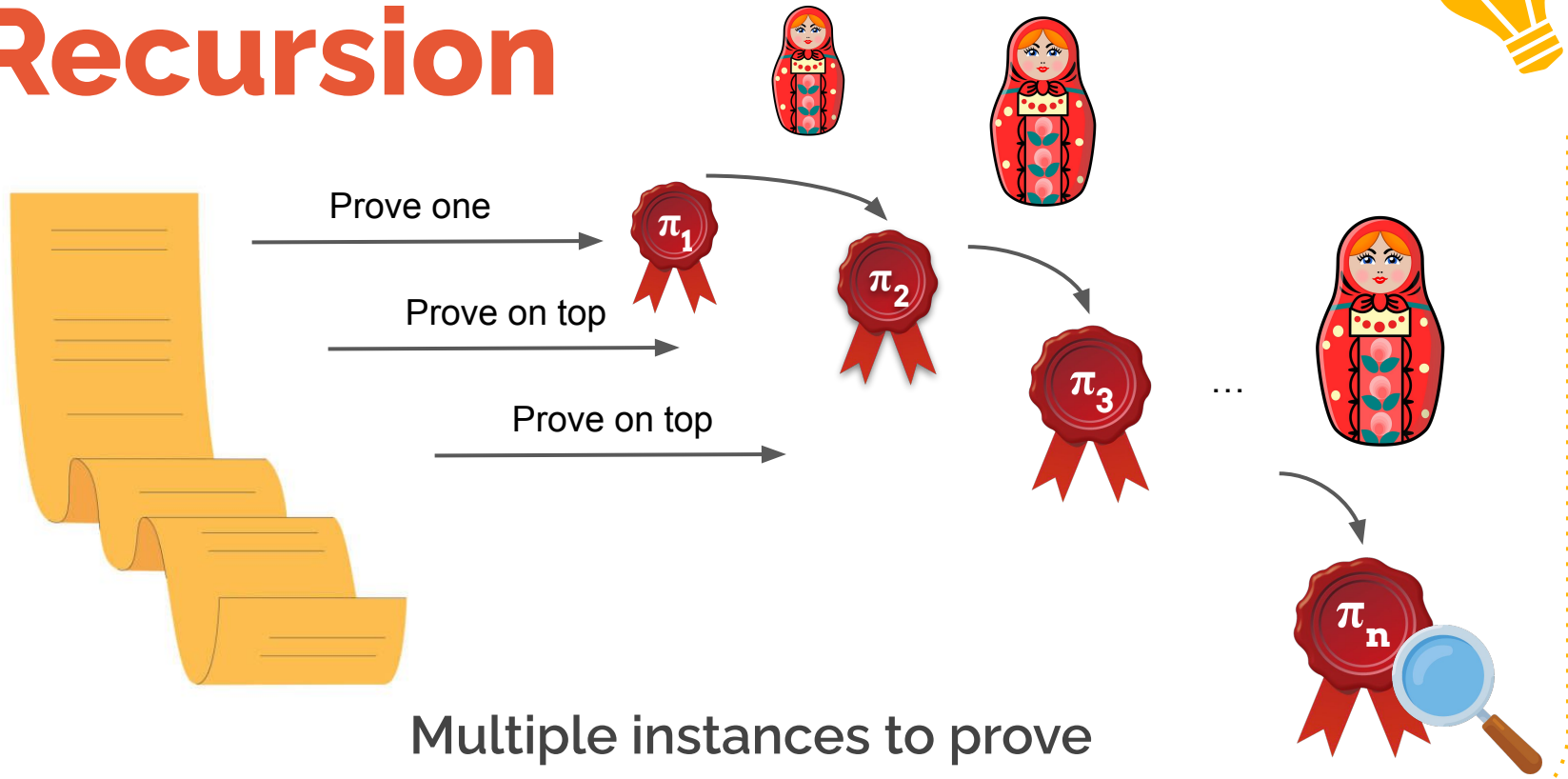

Inner SNARK
Outer SNARK
WITNESS
CIRCUIT
PROVER
PROOF π
VERIFIER
WITNESS
CIRCUIT
PROVER
PROOF π'
π
VERIFIER

Proof of proofs



Multiple instances to prove

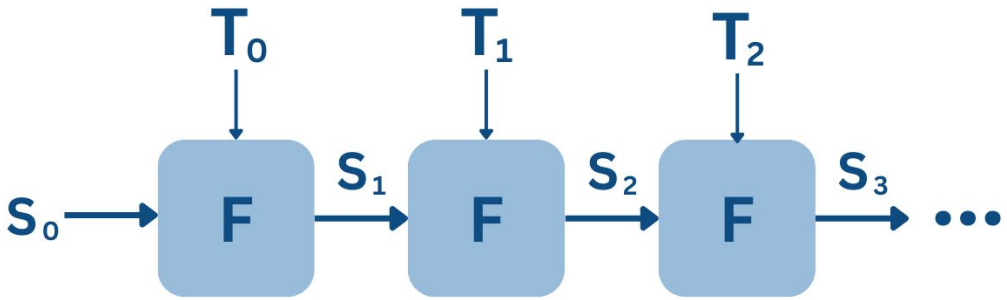
Recursion



Multiple instances to prove

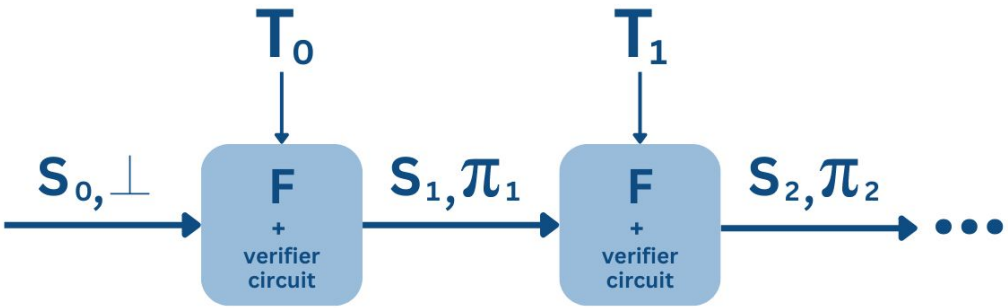
IVC

Incremental Verifiable Computation

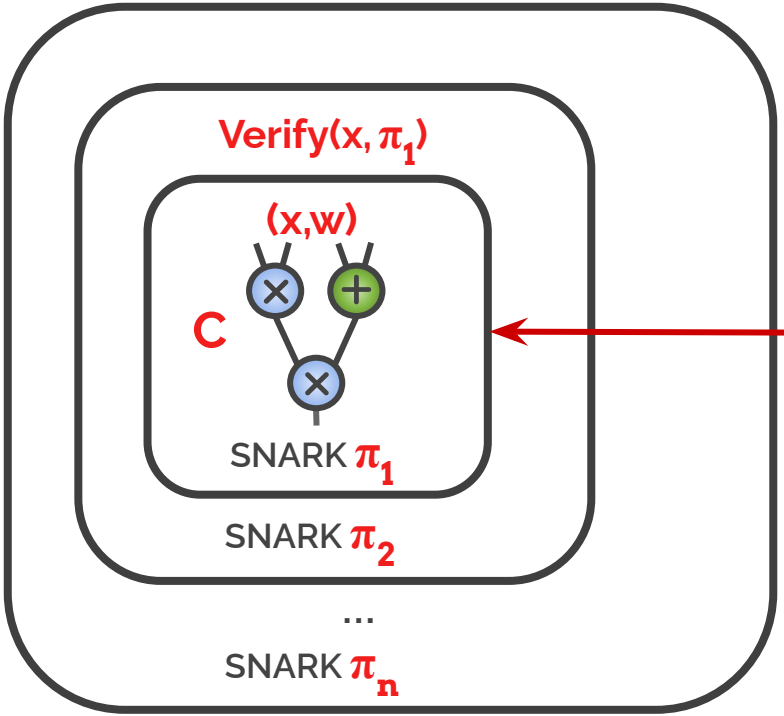


IVC

Incremental Verifiable Computation



Proof Recursion

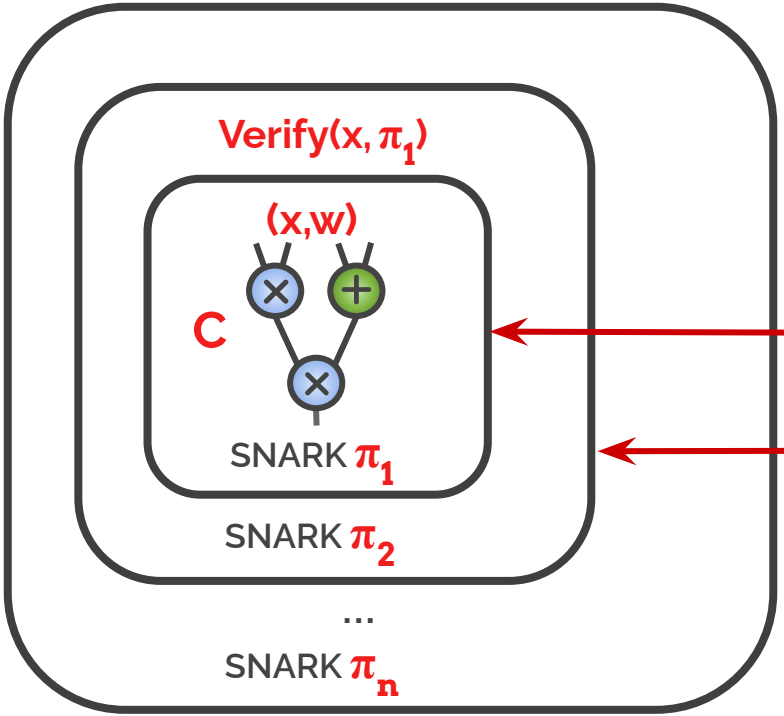


Verification Runtime

$|\text{Verify}| = 2|C_{\text{SNARK}}|$

$|C_{\text{Verify}_1}| = 2|C_{\text{SNARK}}|$

Proof Recursion



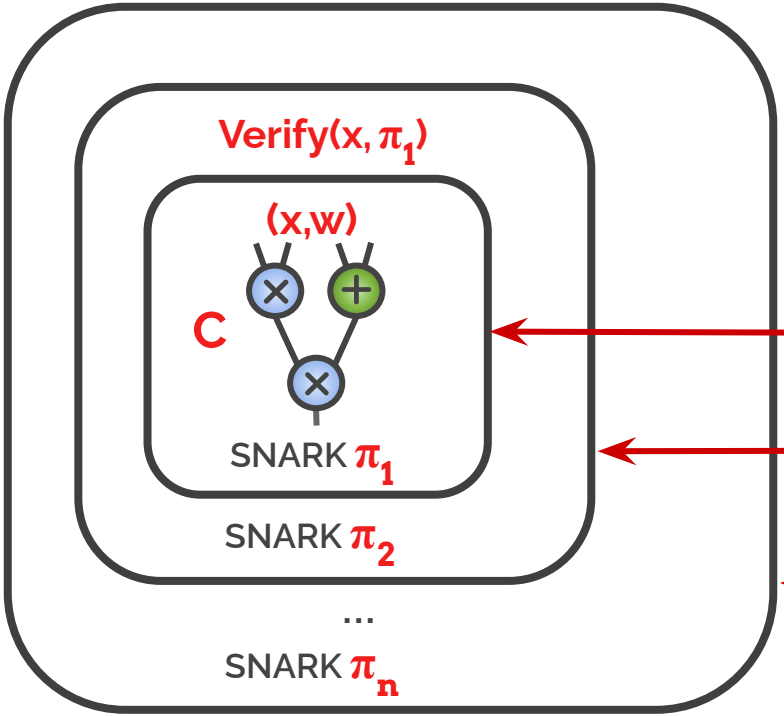
Verification Runtime

$|\text{Verify}| = 2|C_{\text{SNARK}}|$

$|C_{\text{Verify}_1}| = 2|C_{\text{SNARK}}|$

$|C_{\text{Verify}_2}| = 2|C_{\text{Verify}_1}| = 2^2|C_{\text{SNARK}}|$

Proof Recursion



Verification Runtime

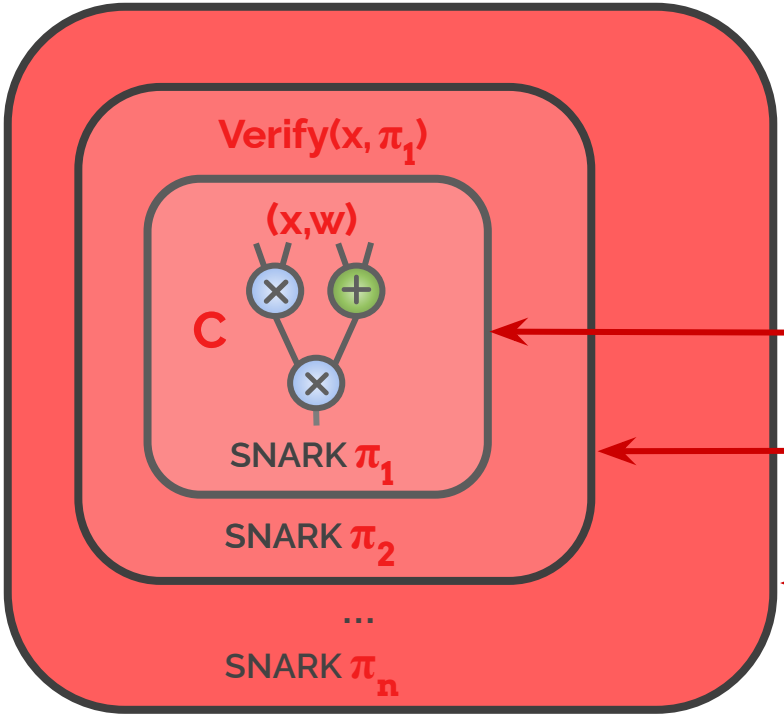
$|\text{Verify}| = 2|C_{\text{SNARK}}|$

$|C_{\text{Verify}_1}| = 2|C_{\text{SNARK}}|$

$|C_{\text{Verify}_2}| = 2|C_{\text{Verify}_1}| = 2^2|C_{\text{SNARK}}|$

$|C_{\text{Verify}_3}| = 2|C_{\text{Verify}_2}| = 2^3|C_{\text{SNARK}}|$

Proof Recursion



Needs Sublinear Verification!

$|C_{Verify_1}| = 2|C_{SNARK}|$

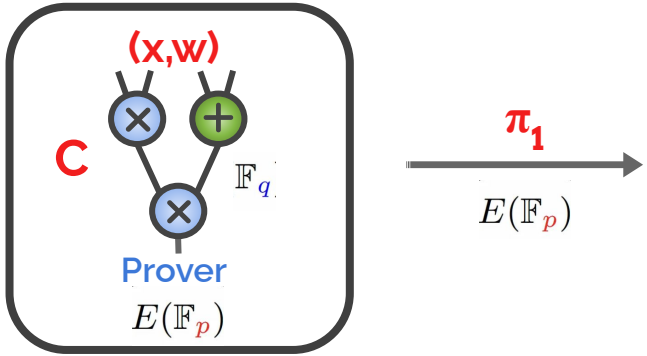
$|C_{Verify_2}| = 2|C_{Verify_1}| = 2^2|C_{SNARK}|$

$|C_{Verify_3}| = 2|C_{Verify_2}| = 2^3|C_{SNARK}|$

Proof Recursion

Verification Arithmetization

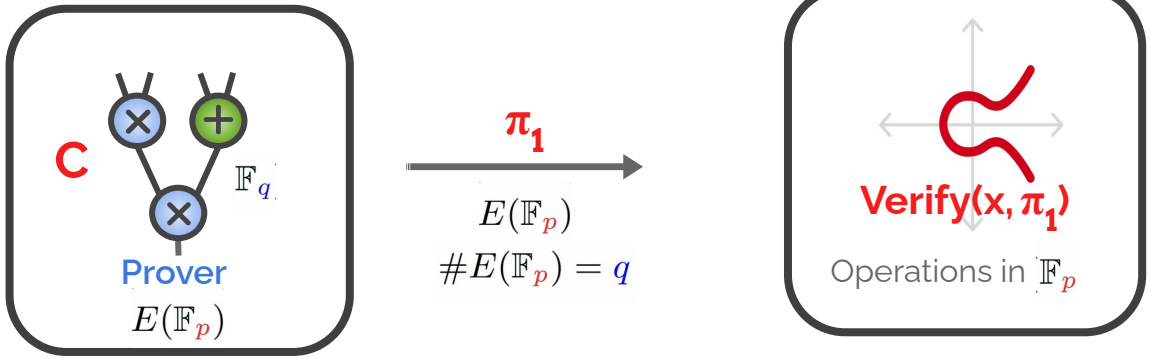
Verify = curve operations



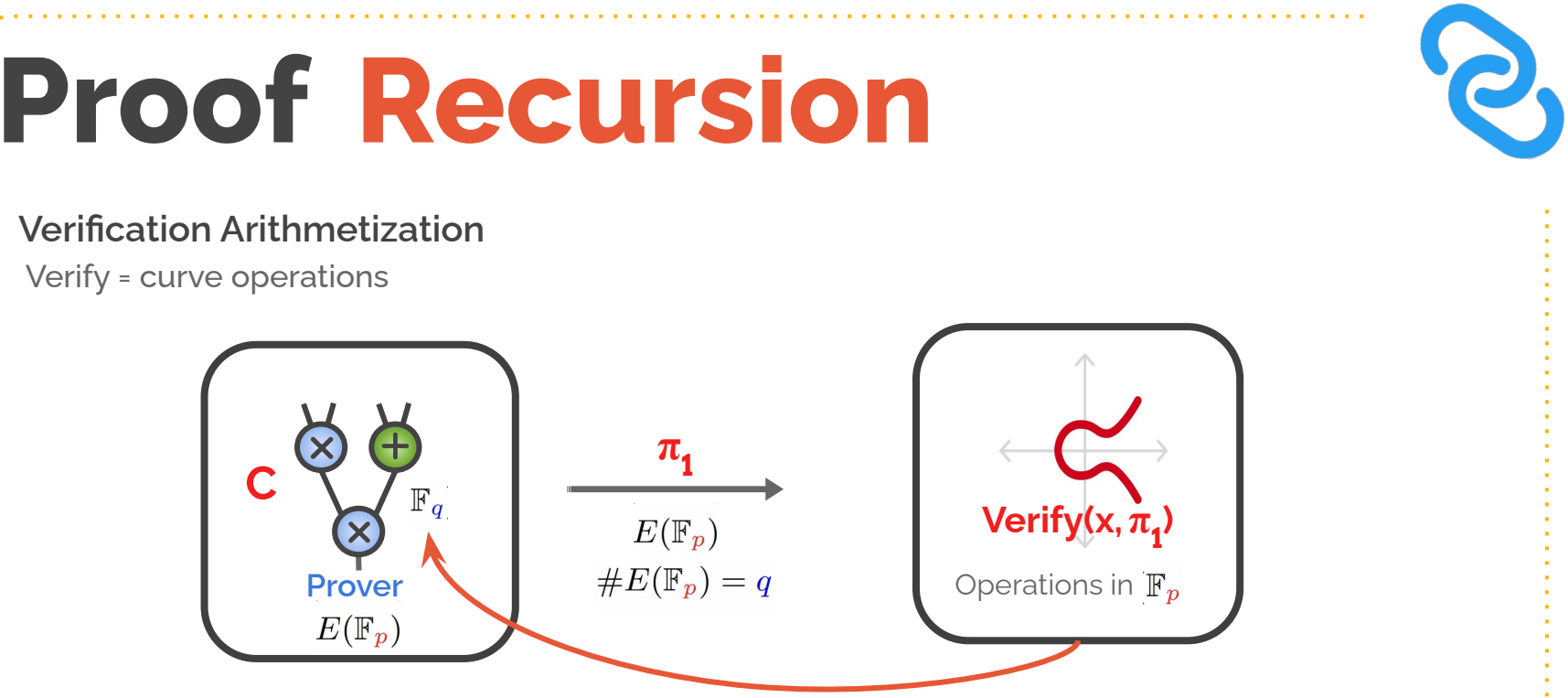
Proof Recursion

Verification Arithmetization

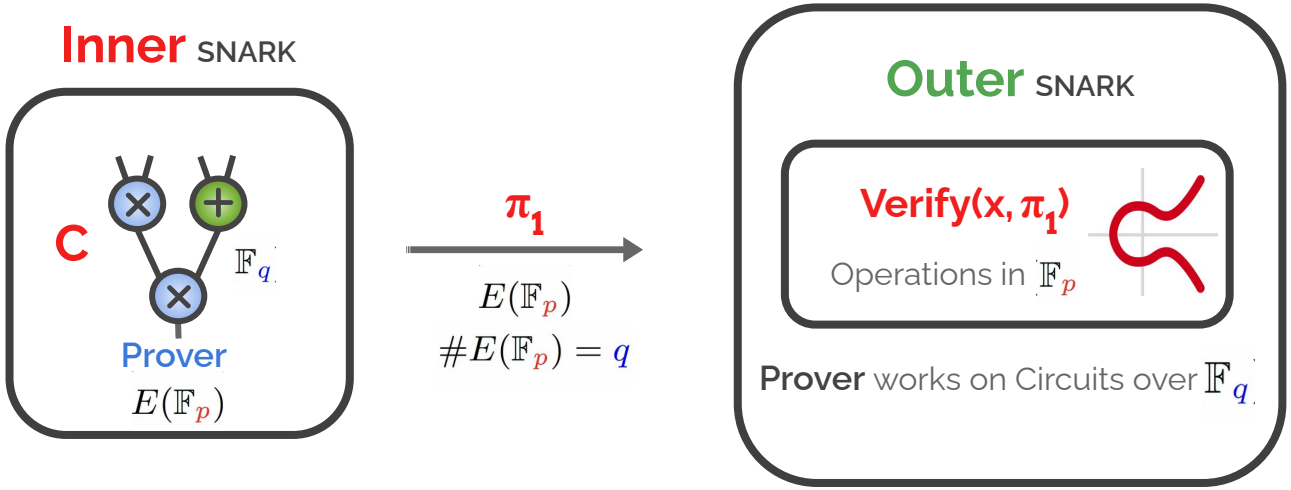
Verify = curve operations



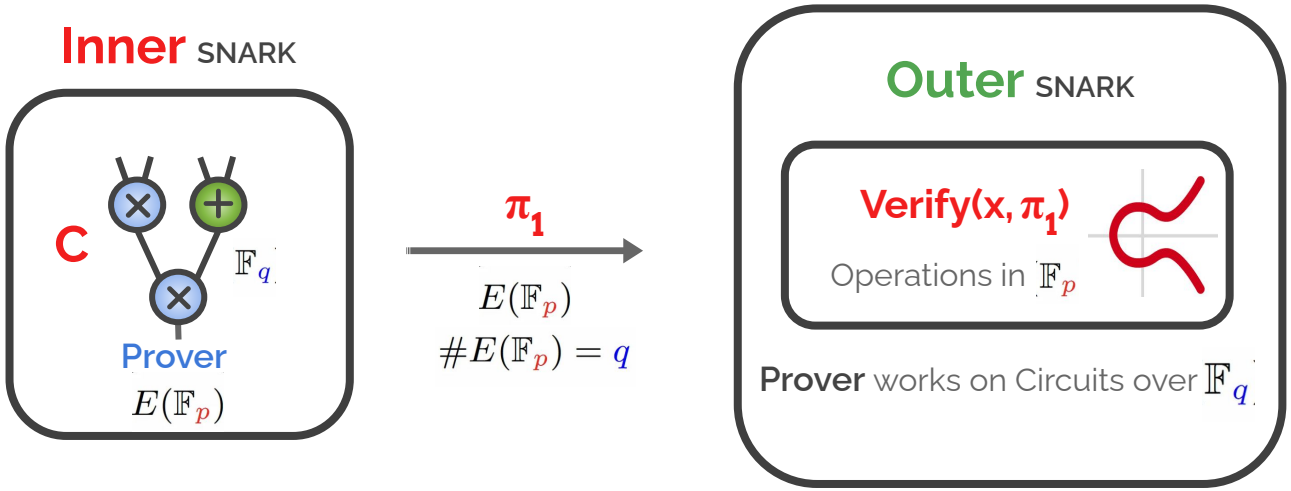
Proof Recursion



Proof Recursion



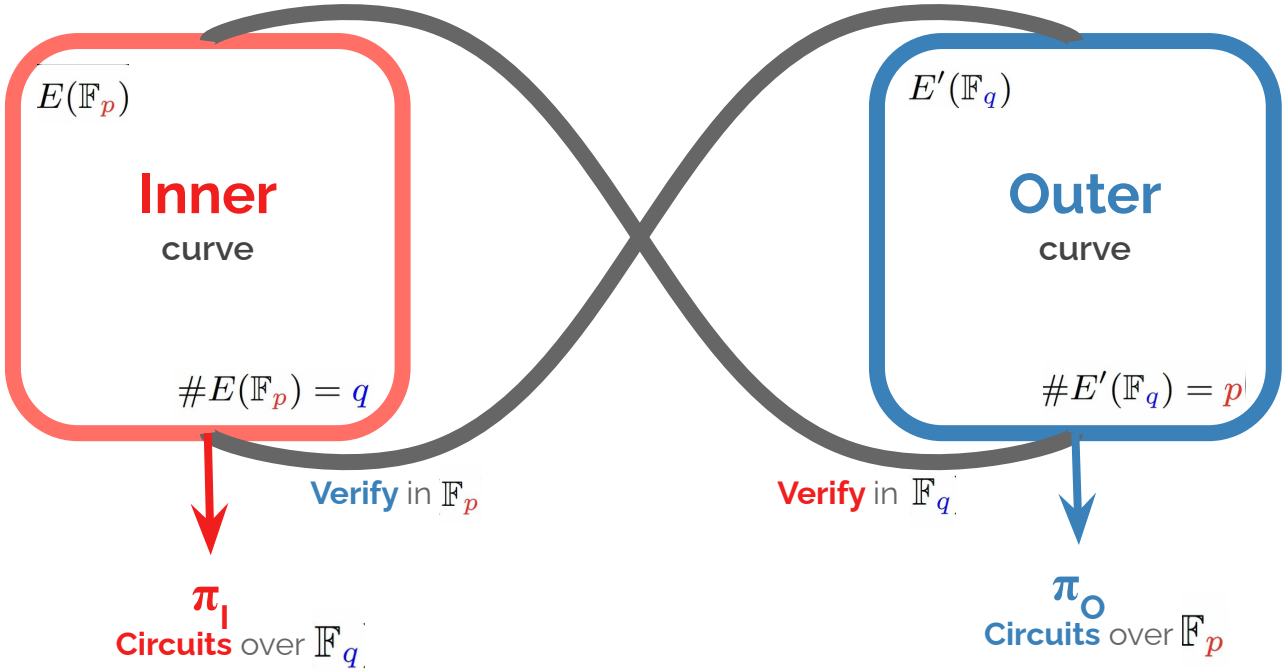
Proof Recursion



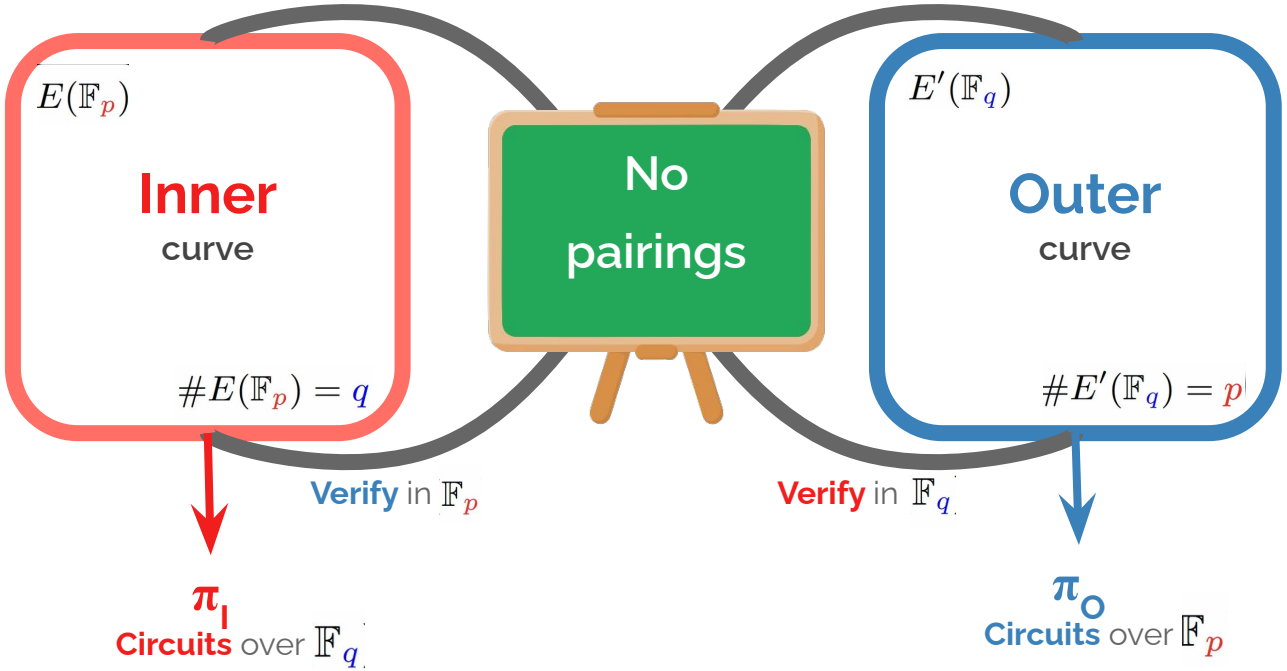
p = q

Dlog is easy to solve :(

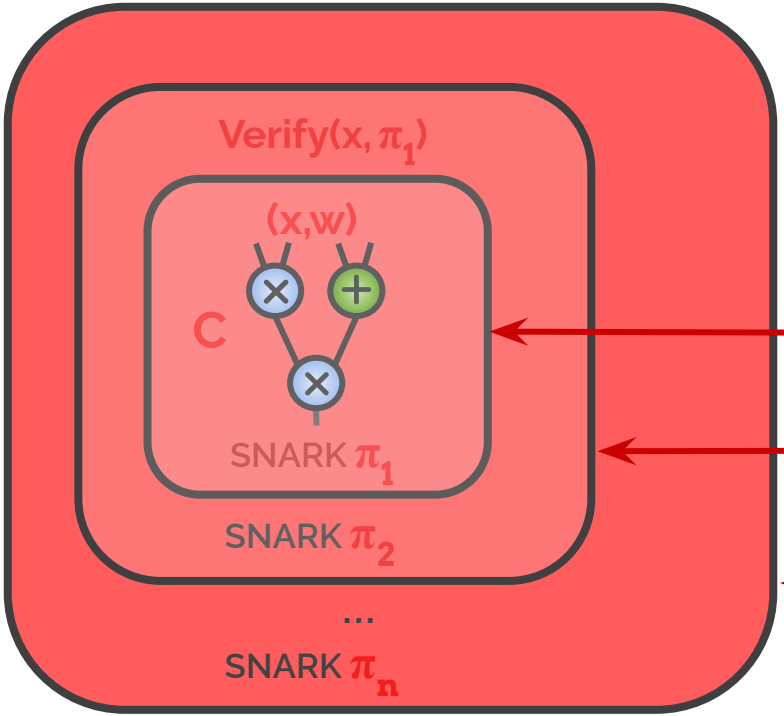
Cycles of Curves



Cycles of Curves



Linear Verfication!!!

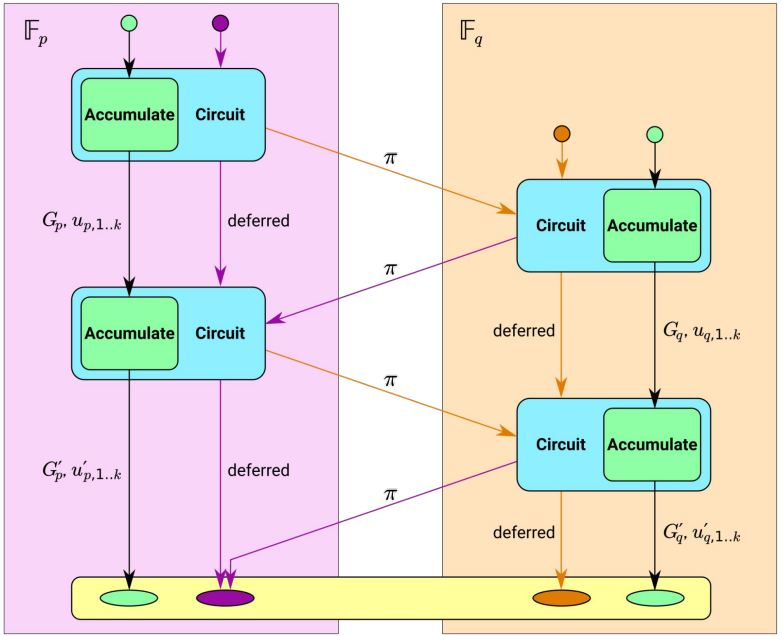


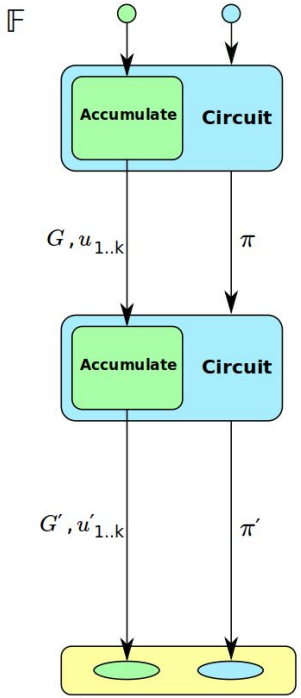
$|C_{Verify_1}| = 2|C_{SNARK}|$

$|C_{Verify_2}| = 2|C_{Verify_1}| = 2^2|C_{SNARK}|$

$|C_{Verify_3}| = 2|C_{Verify_2}| = 2^3|C_{SNARK}|$

Halo Roadmap





Source: [The Halo2 Book]

Thanks

https://www.di.ens.fr/~nitulesc/

Credits

Special thanks to all those who made and released these resources for free:

- Presentation template by SlidesCarnival
- Clip arts by Iconfinder, Flaticon and juicy_fish
- Illustrations by Disneyclips
- Figures from Blog post