Succinct Zero-Knowledge Batch Proofs for Set Accumulators

Matteo Campanelli

Protocol Labs

Dario Fiore

IMDEA Software Institute

Semin Han

Hanyang University

Jihye Kim

Kookmin University

Dimitris Kolonelos

IMDEA Software Institute & Universidad Politécnica de Madrid

Hyunok Oh

Hanyang University
Zkrypto Inc.

HANYANG UNIVERSITY

institute imdea software

Zkrypto
ZeroKnowledgeProof

POLITÉCNICA

KMU KOOKMIN UNIVERSITY

Protocol Labs

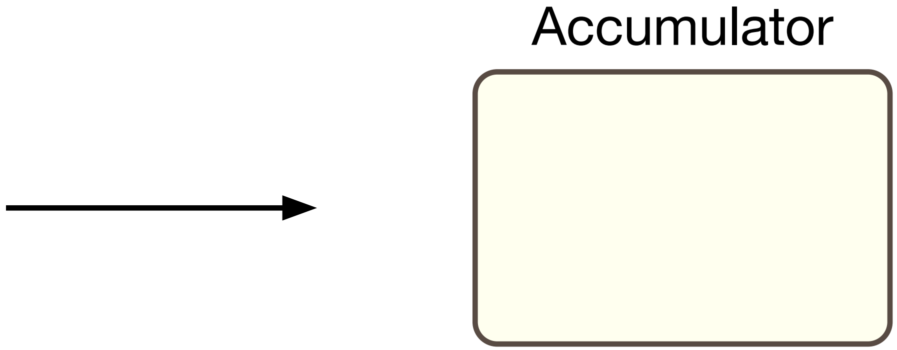
Contents

1. Intro

2. Main idea

3. Evaluation

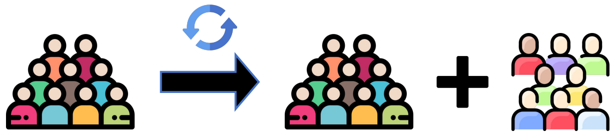
❖ Set accumulator



Set Accumulator » Solution for proving some information of large set

Set membership

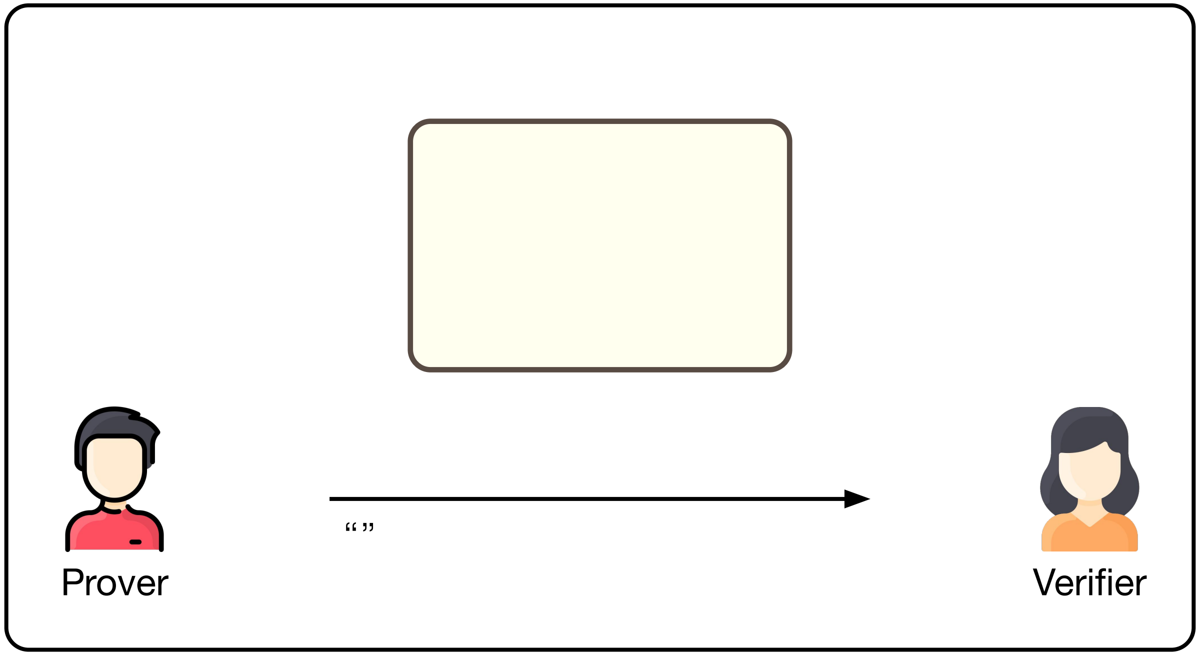
Set updates



1. Intro

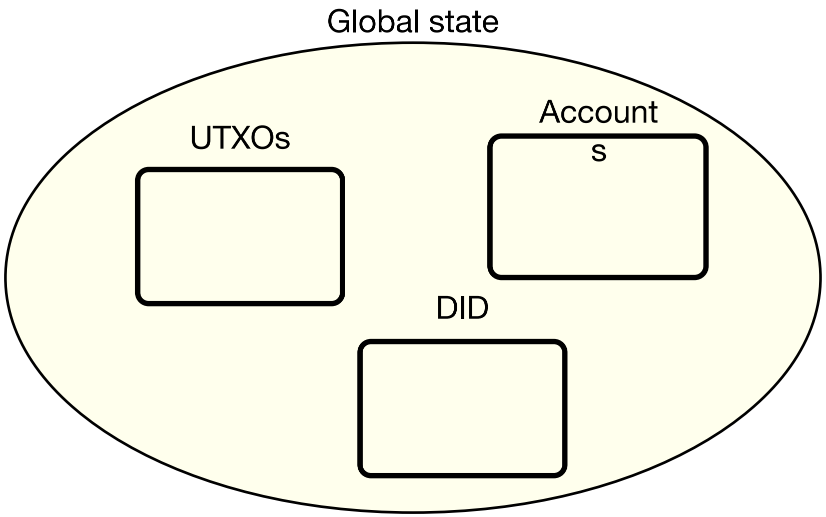
❖ Set membership

- Proving membership where element is in set :
- Batch membership: proving membership for batch elements
- Additional property is also proven with membership



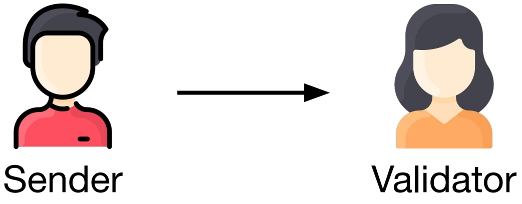
❖ Set membership in blockchain

- In blockchain, set membership is used to prove UTXO, DID, accounts,



✓**Local**: ,

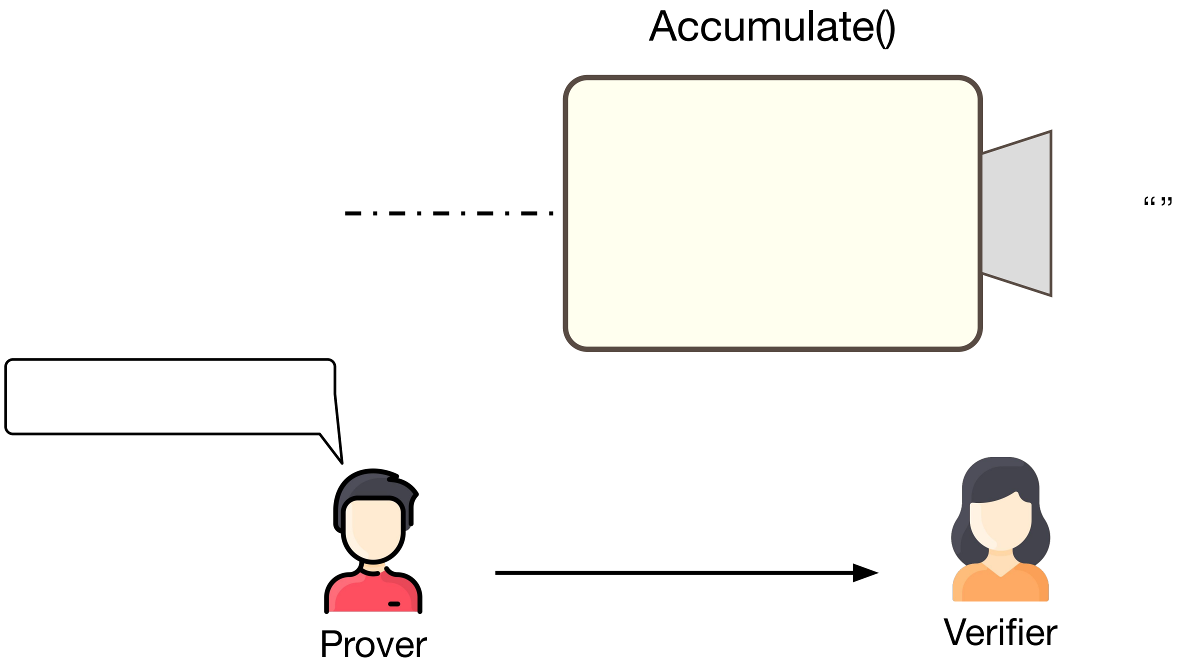
✓**Global**:



✓**Local**: Signature on is valid

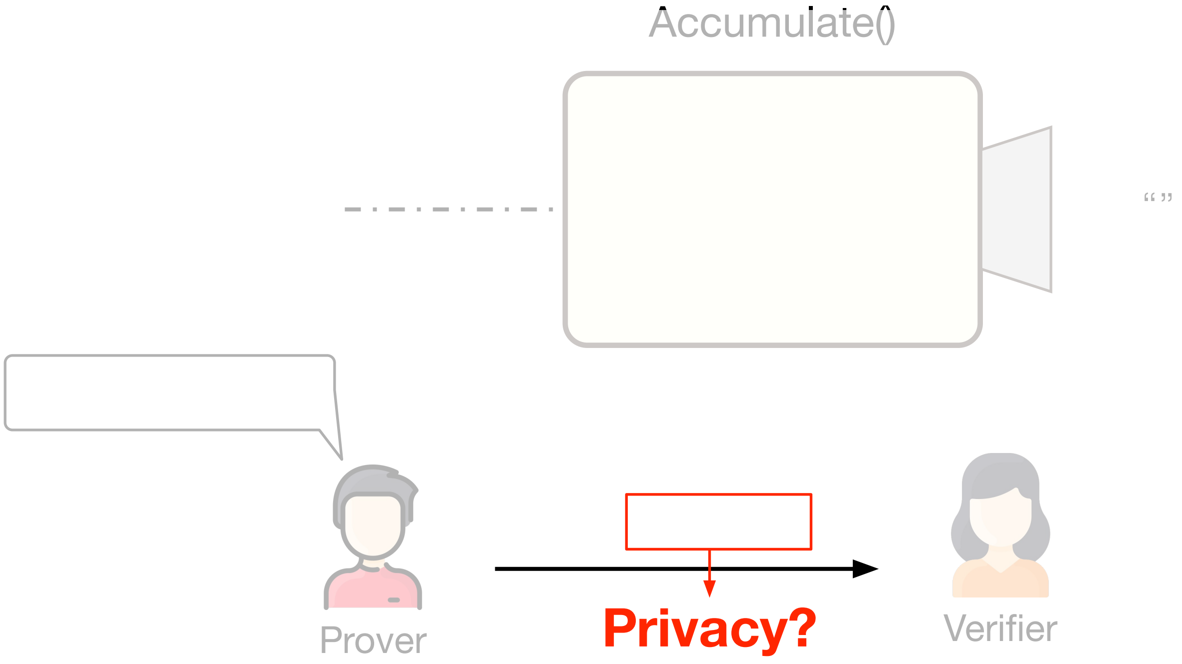
✓**Global**: is consistent with global state

❖ Set membership with accumulator



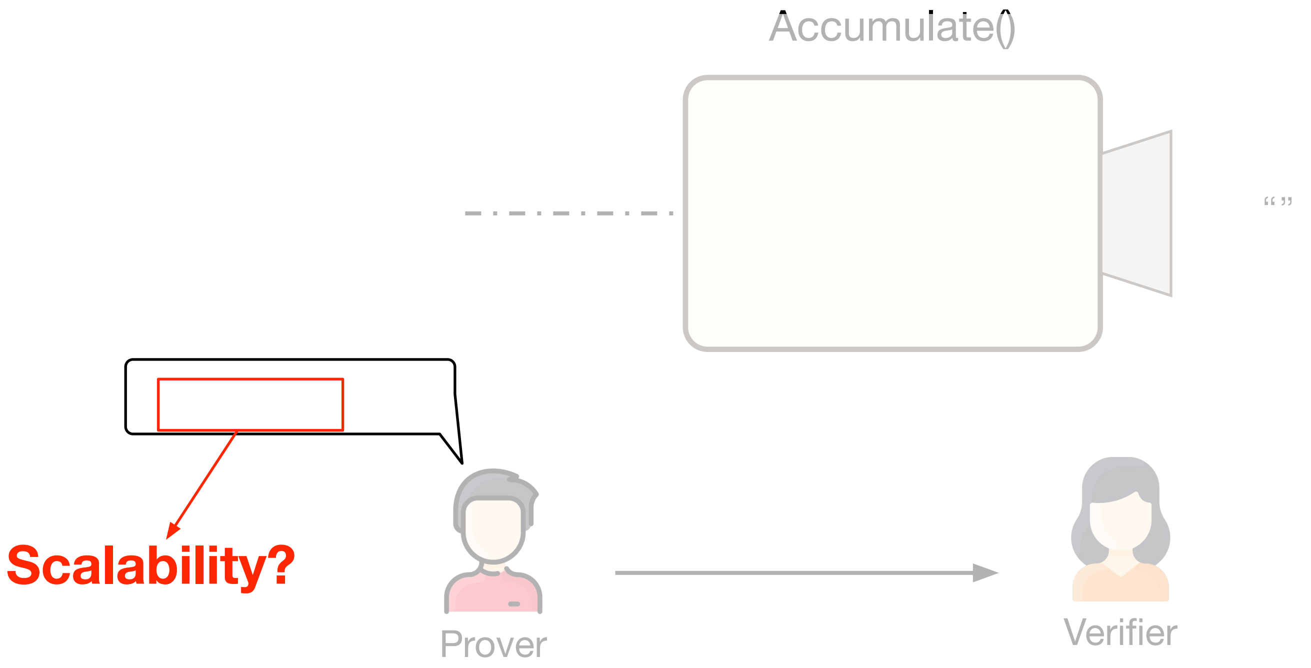
- Prover generates a short proof that an element is a valid member of set
- Verifier checks the proof with an element and
- It is hard to convince verifier that is a valid member of where

❖ Set membership with accumulator



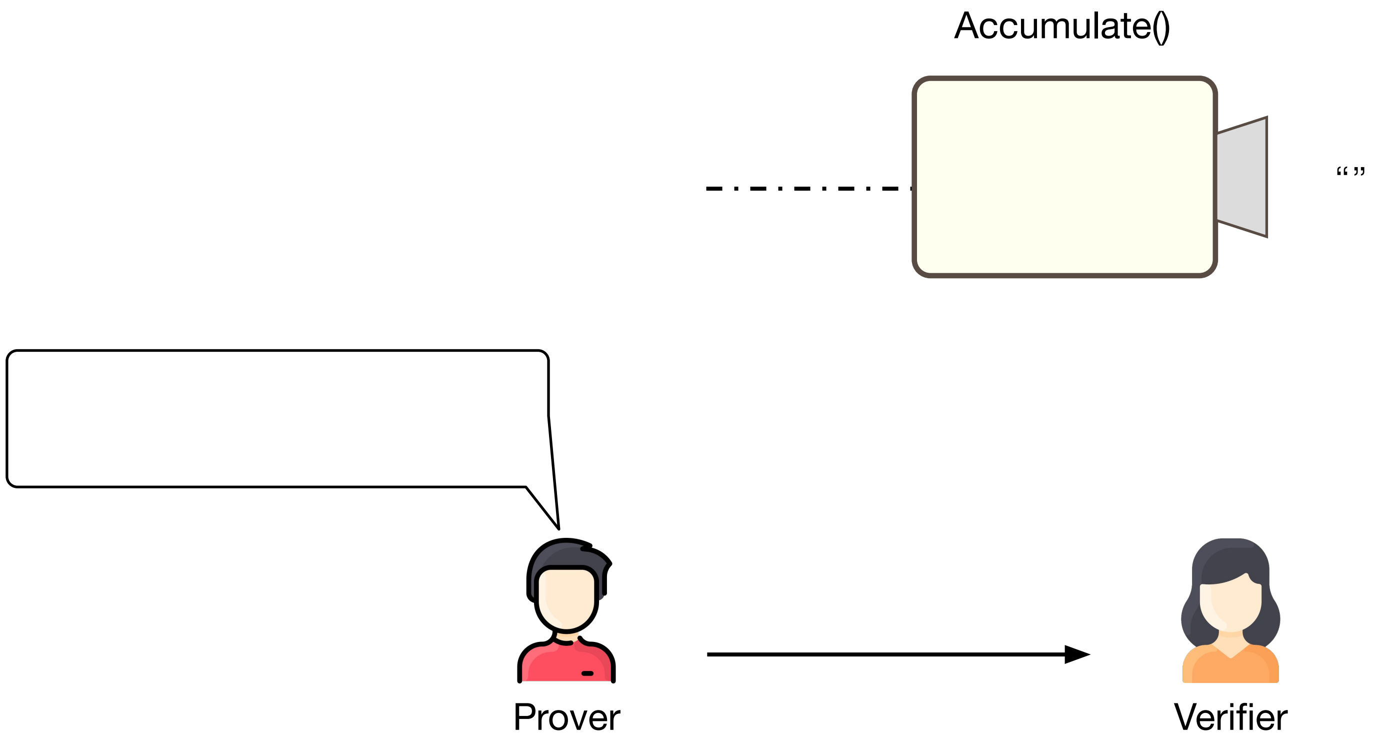
- Prover generates a short proof that an element is a valid member of set
- Verifier checks the proof with an element and
- It is hard to convince verifier that is a valid member of where

❖ Set membership with accumulator



- Prover generates a short proof that an element is a valid member of set
- Verifier checks the proof with an element and
- It is hard to convince verifier that is a valid member of where

❖ Accumulator + zk-SNARKs

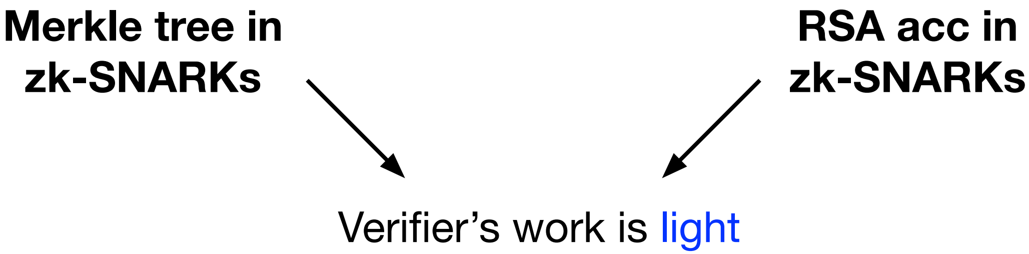


is a valid member subset of set

Additional property for

❖ Existing solutions

Zero-knowledge set membership: Set size: , Batch size:



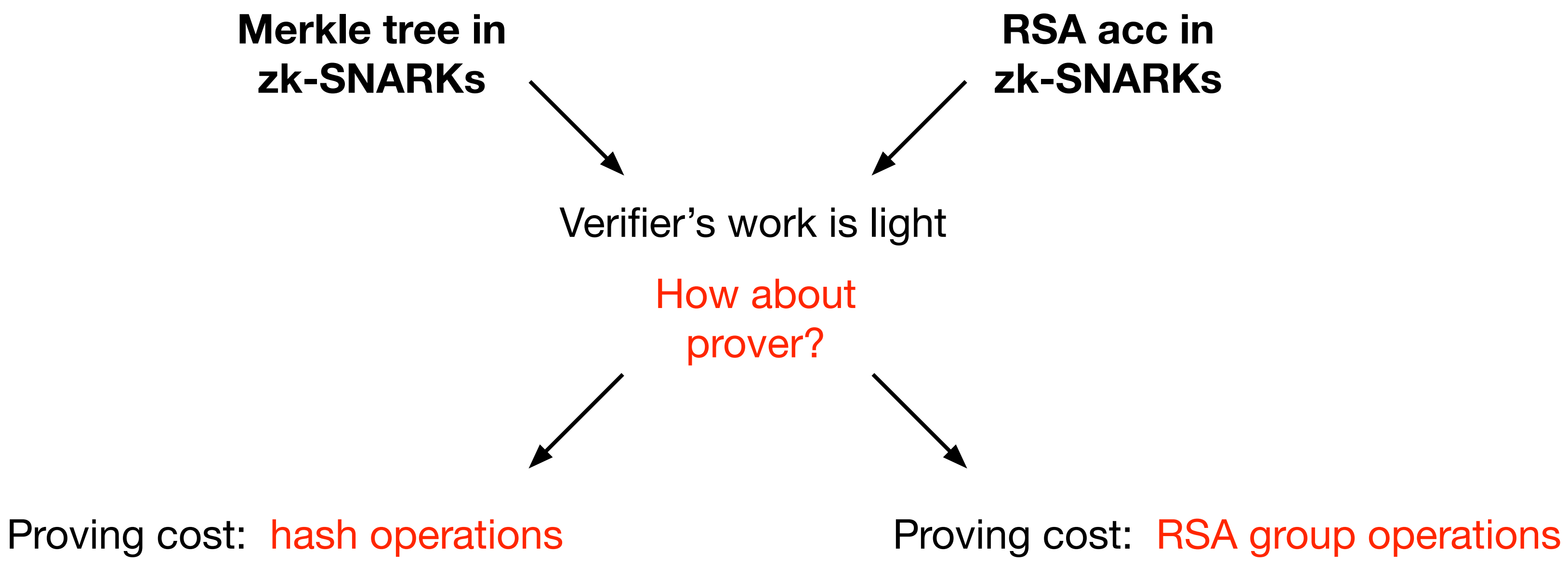
Merkle tree in zk-SNARKs

RSA acc in zk-SNARKs

Verifier’s work is light

❖ Existing solutions

Zero-knowledge set membership: Set size: , Batch size:



Merkle tree in zk-SNARKs

RSA acc in zk-SNARKs

Verifier's work is light

How about prover?

Proving cost: hash operations

Proving cost: RSA group operations

No scalable solution for proving batch membership proofs

❖ Existing solutions

	Batch	zk	Proving time
Merkle Trees in zk-SNARKs	No	Yes	✗
SNARK-friendly MTs[1] in zk-SNARKs	No	Yes	-
RSA accumulators in SNARK[2]	Yes	No	✓
RSA accumulators in SNARK[3]	No	Yes	✓
Ours	Yes	Yes	✓✓

1) Poseidon: A new hash function for zero-knowledge proof systems, Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger, Usenix Security 2021
2) Scaling Verifiable Computation Using Efficient Set Accumulators, Alex Ozdemir, Read S. Wahby, Barry Whitehat, and Dan Boneh, Usenix Security 2020
3) Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular, Daniel Benarroch, Matteo Campanelli, Dario Fiore, Kobi Gurkan, and Dimitris Colonels, Conference on Financial

❖ Our work

HARiSA:
elements-Hiding Argument for RSA accumulators

-A **new randomization method** for RSA accumulator witness

-A new way to prove the accumulator verification without encoding RSA group operations in the circuit

B-INS-ARiSA:
Batch-INSertion Argument for RSA accumulators

-Succinct proofs for batch updates (=>MultiSwap)

-Scaling down our techniques for set-membership

Implementation/Evaluation

-HARiSA vs Merkle tree(Poseidon) : 14~33x faster than Merkle tree prover

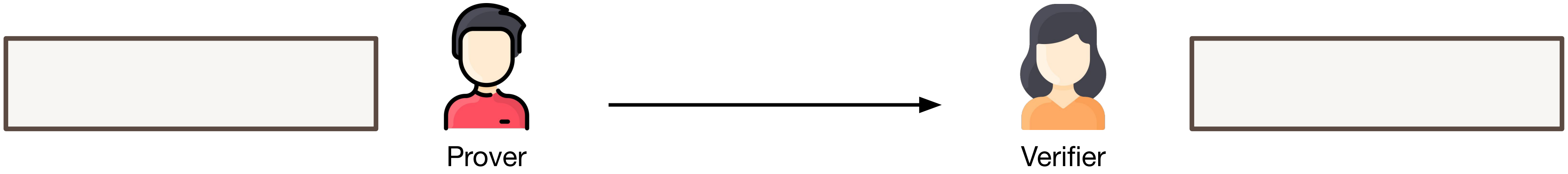
-MultiSwap for Set updates: B-INS-ARiSA vs MerkleSwap vs [OWWB]

2. Main idea

❖ HARiSA: From RSA accumulator

Given random group element

Goal: Prove that

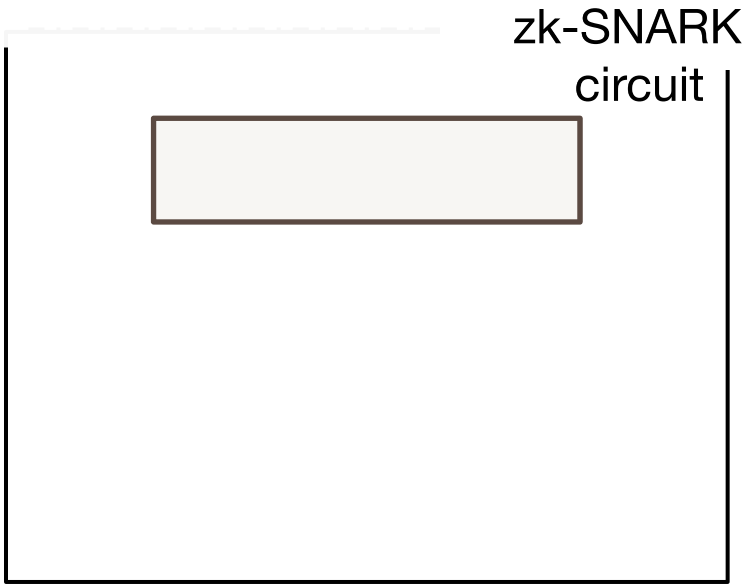


2. Main idea

❖ HARiSA: From RSA accumulator

Given random group element

Goal: Prove that



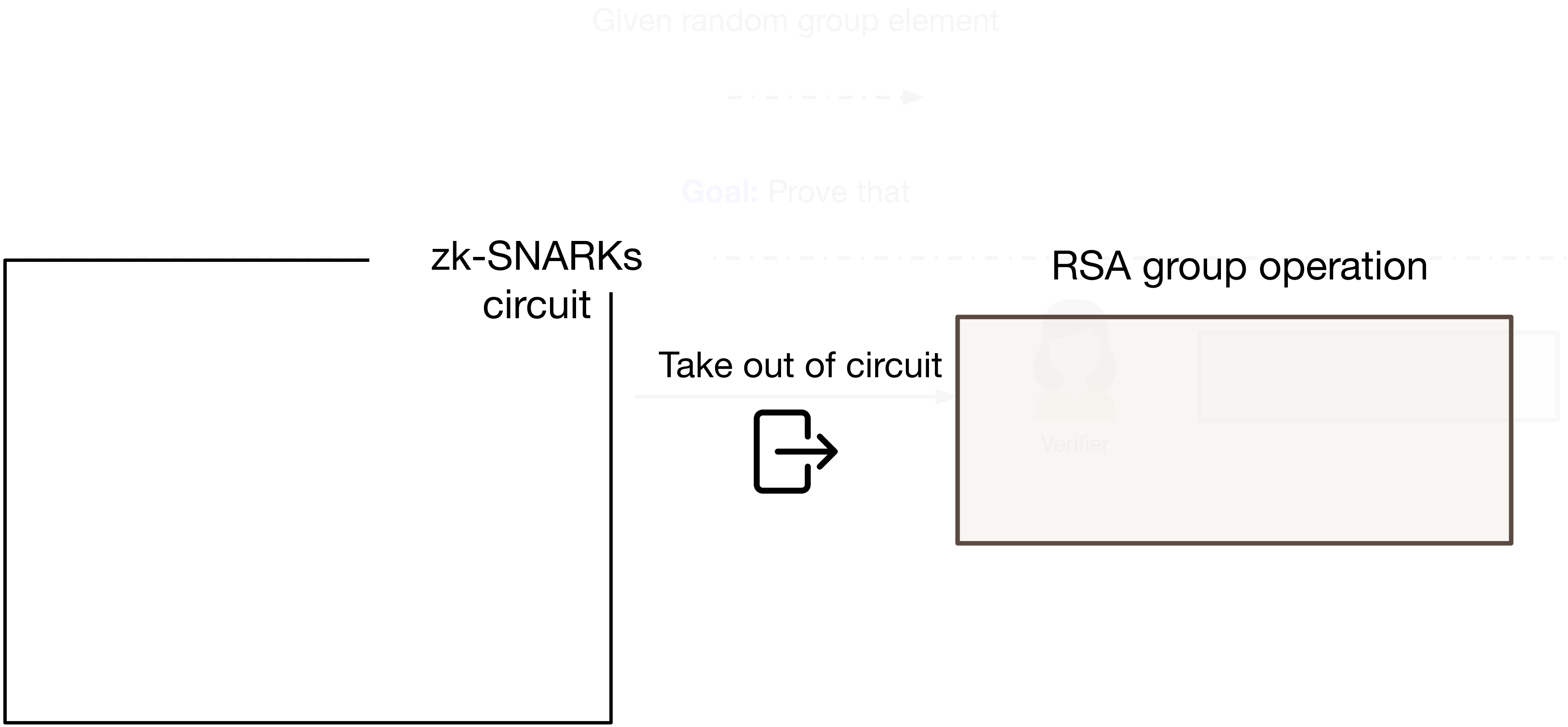
RSA group operation, …

Verifier

Proving overhead is quite large

2. Main idea

❖ HARiSA: From RSA accumulator



Given random group element

Goal: Prove that

zk-SNARKs circuit

Take out of circuit

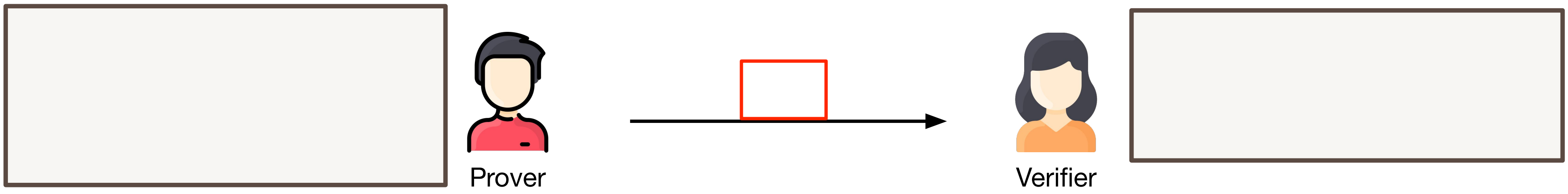
RSA group operation

Verifier

❖ HARiSA: From RSA accumulator

Given random group element

Goal: Prove that with hiding



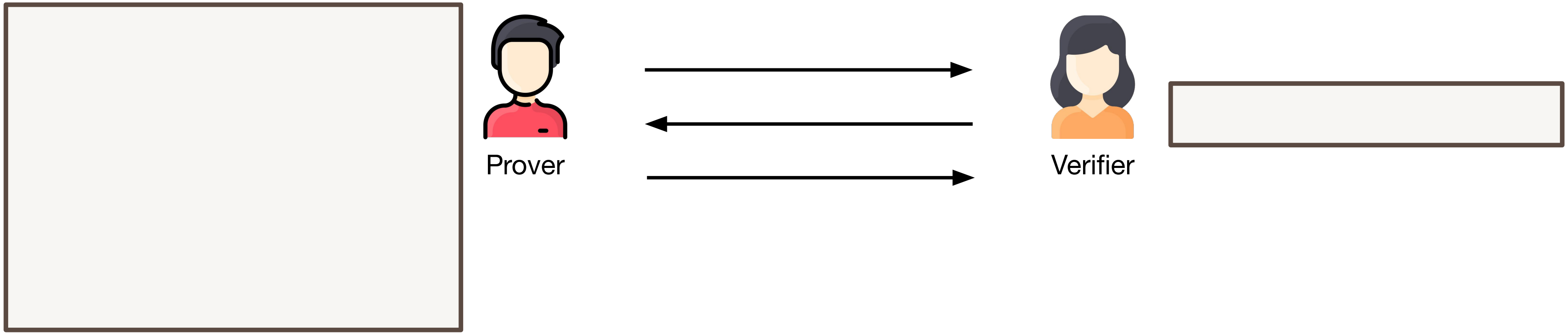
How to obtain **privacy**?

2. Main idea

❖ HARiSA: -protocol

Given random group element

Goal: Prove that with hiding

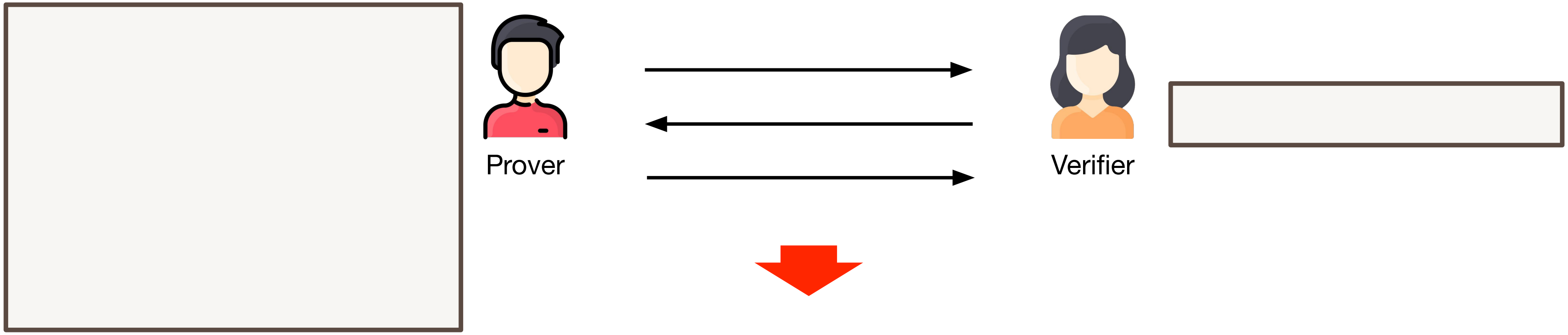


2. Main idea

❖ HARiSA: -protocol

Given random group element

Goal: Prove that with hiding



How can we link to zk-SNARKs?

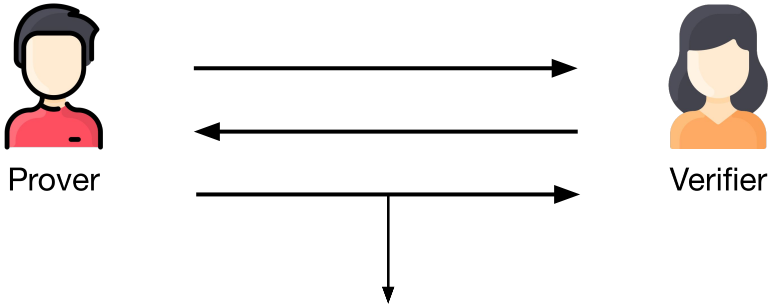
❖ HARiSA: -protocol

Given random group element

[redacted]

Goal: Prove that with hiding

[redacted]

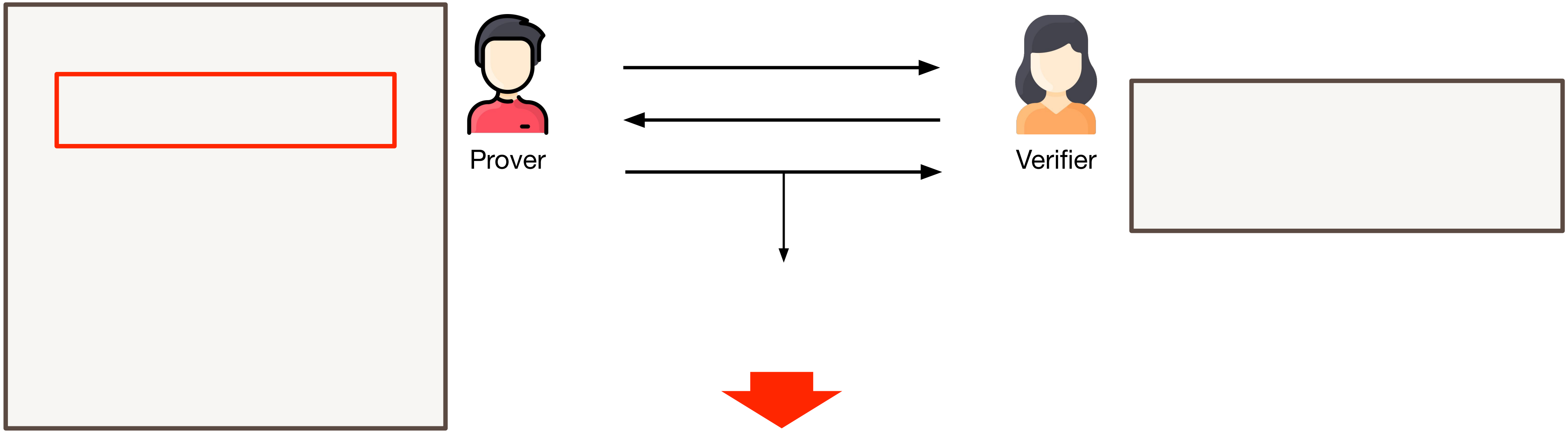


2. Main idea

❖ HARiSA: -protocol

Given random group element

Goal: Prove that with hiding

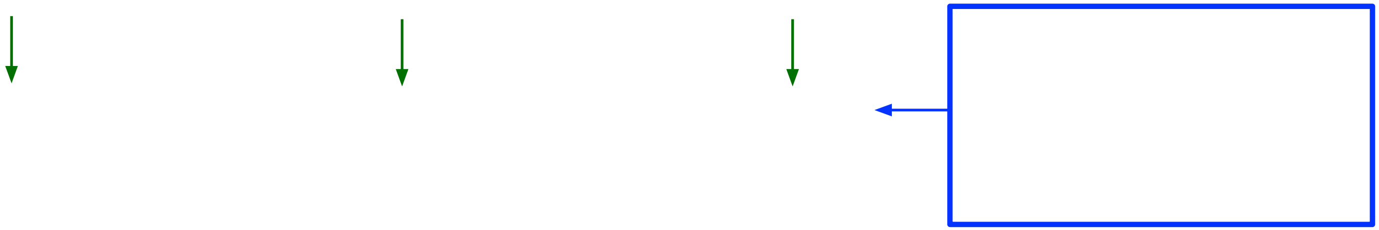


Witness still **leaks information** about

❖ HARiSA: New randomization technique for hiding witness

New randomization method for an RSA accumulator witness

Let be the first numbers:



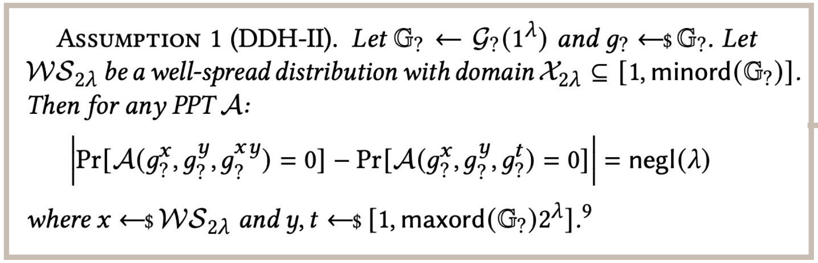
2. Main idea

❖ HARiSA: New randomization technique for hiding witness

New randomization method for an RSA accumulator witness

Let be the first numbers:

Zero-Knowledge :



Assumption 1 (DDH-II). *Let* $\mathbb{G}_? \leftarrow \mathcal{G}_?(1^\lambda)$ *and* $g_? \leftarrow_\$ \mathbb{G}_?$*. Let* $\mathcal{WS}_{2\lambda}$ *be a well-spread distribution with domain* $\mathcal{X}_{2\lambda} \subseteq [1, \mathrm{minord}(\mathbb{G}_?)]$*. Then for any PPT* $\mathcal{A}$*:*

$$\left|\Pr[\mathcal{A}(g_?^x, g_?^y, g_?^{xy}) = 0] - \Pr[\mathcal{A}(g_?^x, g_?^y, g_?^t) = 0]\right| = \mathrm{negl}(\lambda)$$

where $x \leftarrow_\$ \mathcal{WS}_{2\lambda}$ *and* $y, t \leftarrow_\$ [1, \mathrm{maxord}(\mathbb{G}_?)2^\lambda]$.[9]

is a random integer over a range of

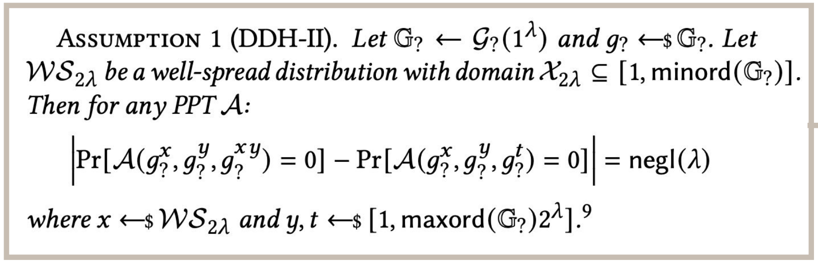
:distribution of is well-spread

❖ HARiSA: New randomization technique for hiding witness

New randomization method for an RSA accumulator witness

Let be the first numbers:

Zero-Knowledge :



ASSUMPTION 1 (DDH-II). *Let* $\mathbb{G}_? \leftarrow \mathcal{G}_?(1^\lambda)$ *and* $g_? \leftarrow_\$ \mathbb{G}_?$. *Let* $\mathcal{WS}_{2\lambda}$ *be a well-spread distribution with domain* $\mathcal{X}_{2\lambda} \subseteq [1, \mathrm{minord}(\mathbb{G}_?)]$. *Then for any PPT* $\mathcal{A}$*:*

$$\left|\Pr[\mathcal{A}(g_?^x, g_?^y, g_?^{xy}) = 0] - \Pr[\mathcal{A}(g_?^x, g_?^y, g_?^t) = 0]\right| = \mathrm{negl}(\lambda)$$

where $x \leftarrow_\$ \mathcal{WS}_{2\lambda}$ *and* $y, t \leftarrow_\$ [1, \mathrm{maxord}(\mathbb{G}_?)2^\lambda]$.[9]

is computationally indistinguishable

❖ HARiSA: New randomization technique for hiding

witness

Given random group element

Goal: Prove that with hiding

Prover
Verifier

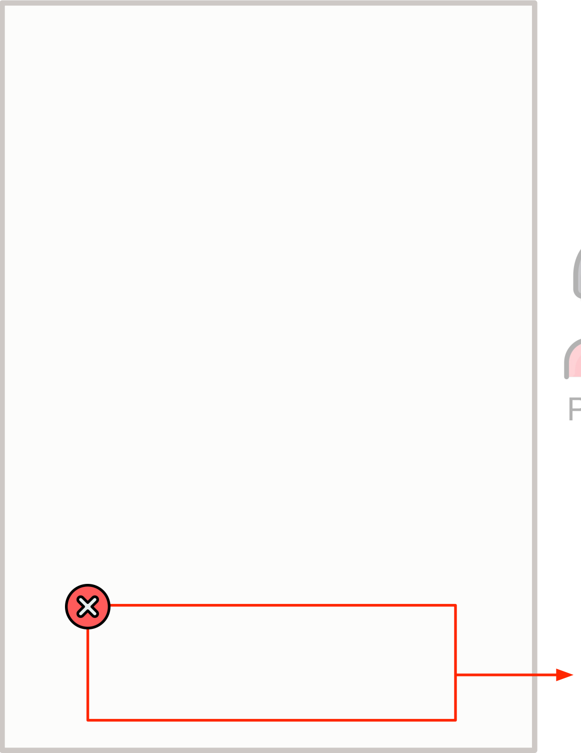
2. Main idea

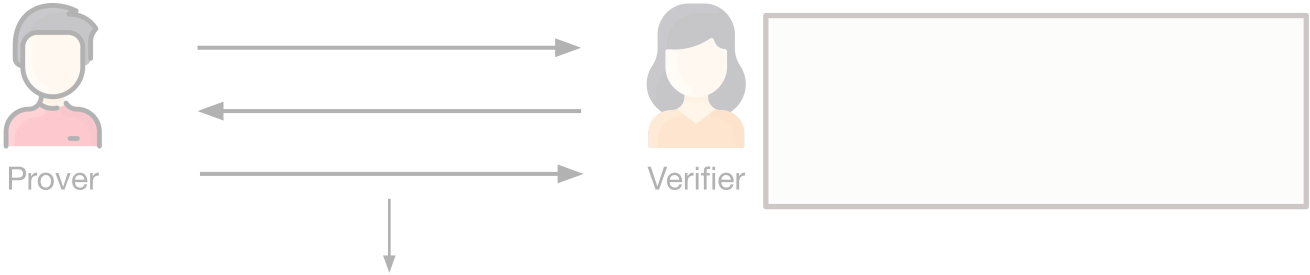
❖ HARiSA: New randomization technique for hiding

witness

Given random group element

Goal: Prove that with hiding



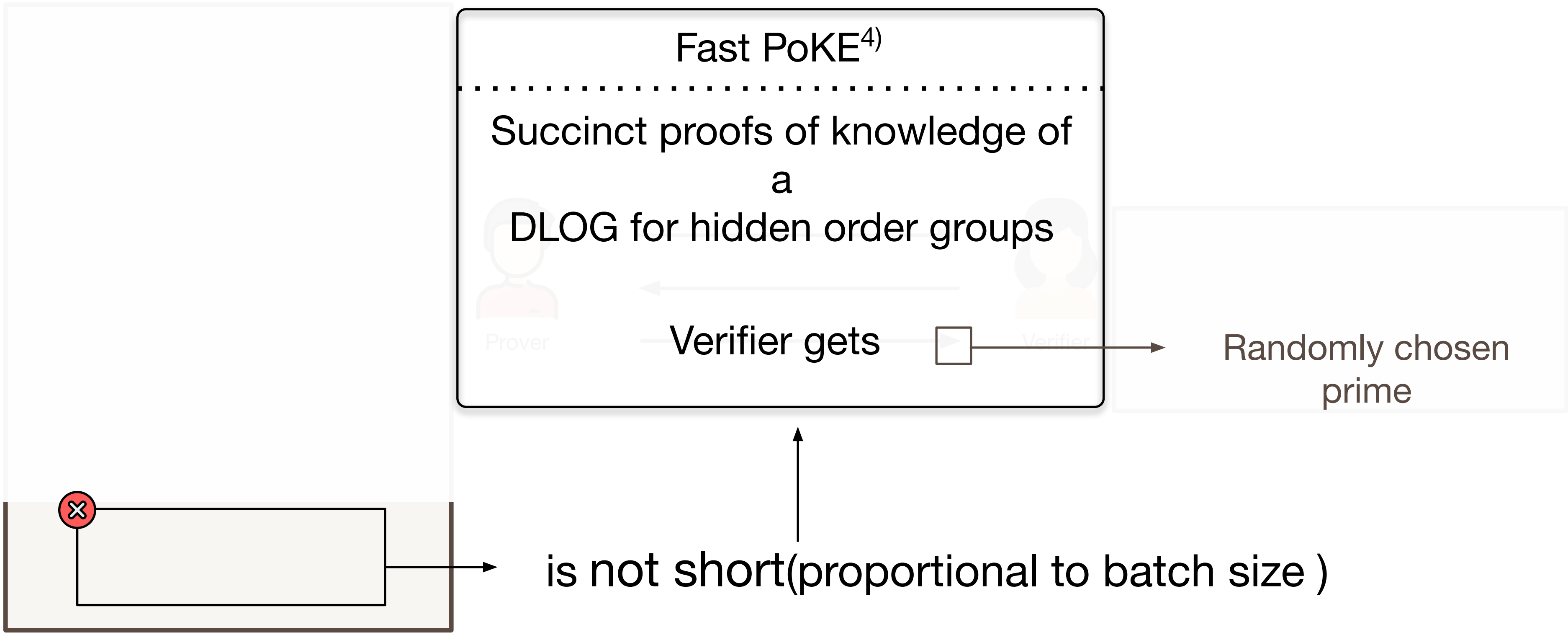


is **not short**(proportional to batch size)

❖ HARiSA: Succinctness with Proof of Knowledge Exponent(PoKE)

Given random group element

Goal: Prove that with hiding



Fast PoKE[4]

Succinct proofs of knowledge of a DLOG for hidden order groups

Verifier gets □

Randomly chosen prime

is not short(proportional to batch size)

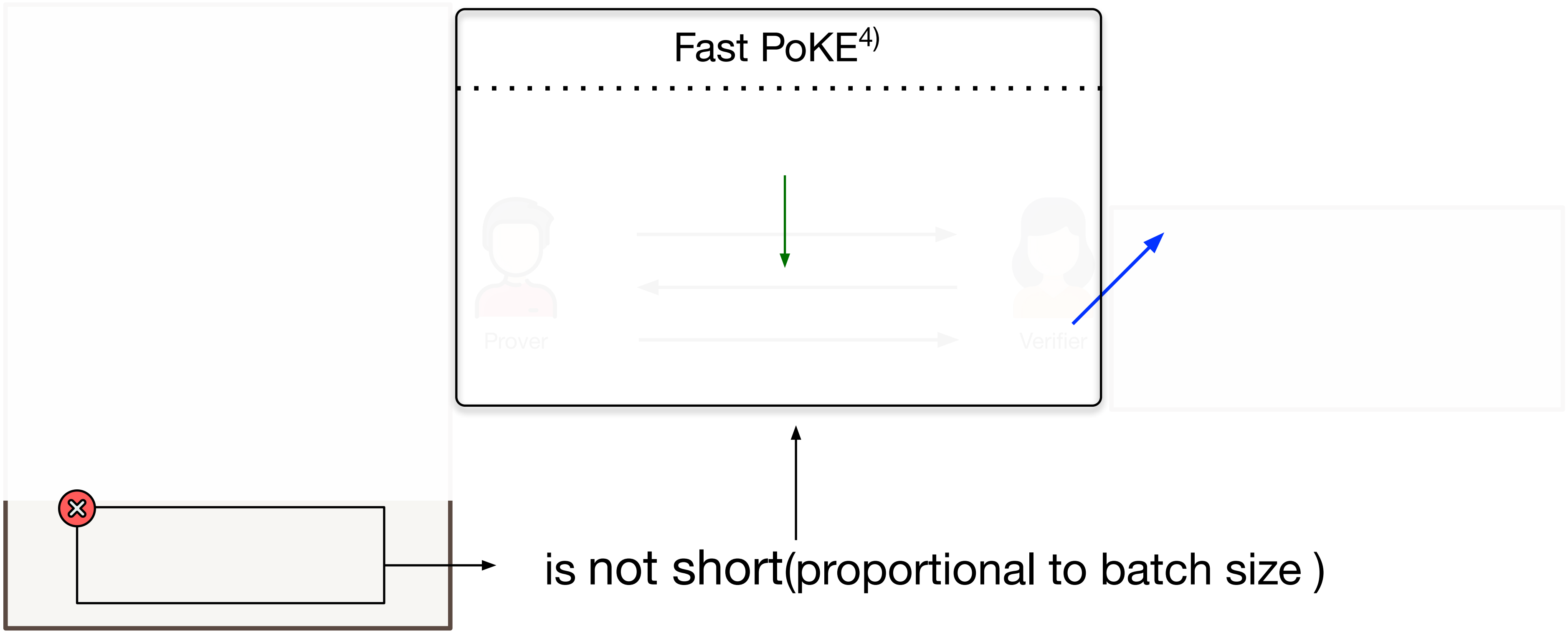
2) Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains, Dan Boneh, Benedikt Bünz, Ben Fisch, CRYPTO 2019

2. Main idea

❖ HARiSA: Succinctness with Proof of Knowledge Exponent(PoKE)

Given random group element

Goal: Prove that with hiding



is not short(proportional to batch size)

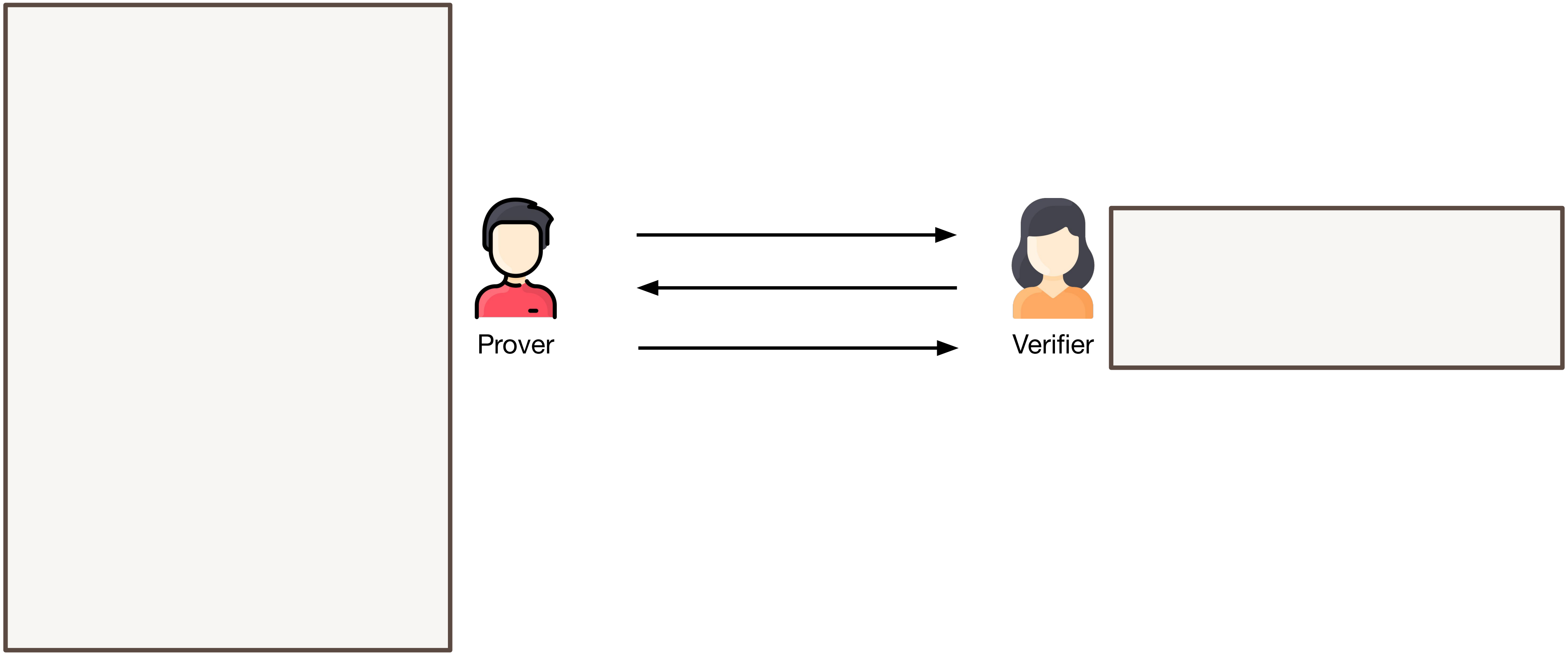
4) Batching Techniques for Accumulators with Applications to IOPs and Stateless Blockchains, Dan Boneh, Benedikt Bünz, Ben Fisch, CRYPTO 2019

2. Main idea

❖ HARiSA

Given random group element

Goal: Prove that with hiding



2. Main idea

❖ HARiSA

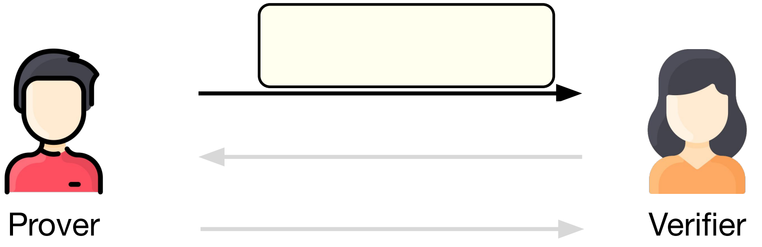
Given random group element

⇢

Goal: Prove that with hiding

✓Succinctness

✓Privacy



✓Link to SNARK

3. Evaluation

❖ Instantiation/Implementation

Instantiation

: LegoGroth16 [CFQ19] using BLS12-381 Curve

Hidden order group: 2048-bit RSA Group

Hash functions: Poseidon hash function

Implementation

C++ based on libsnark + Java based on Snark

Evaluations

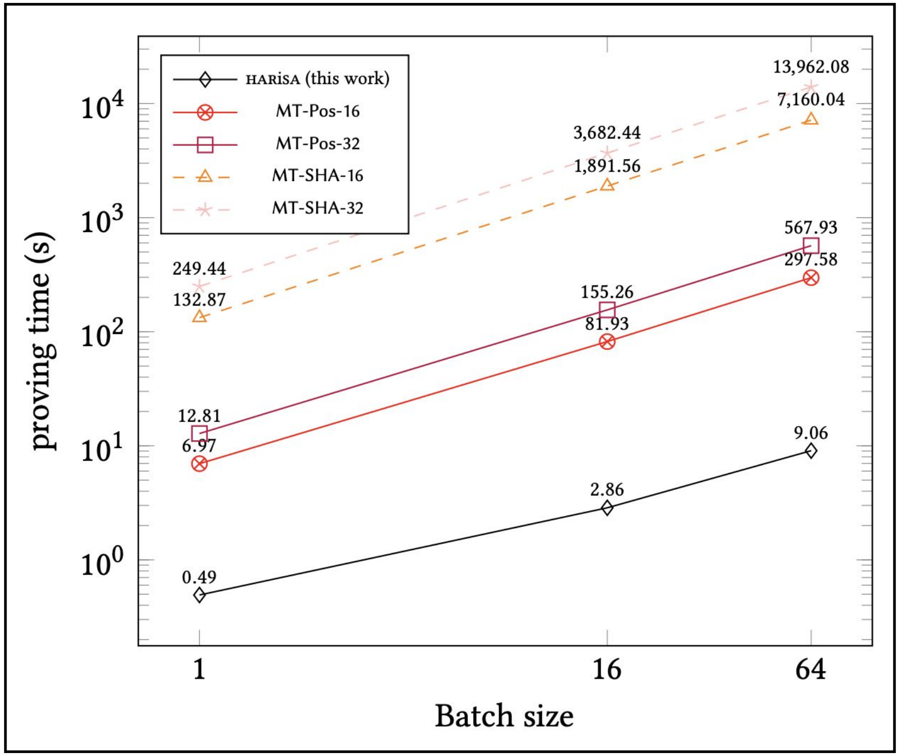
Batch membership: HARiSA vs Merkle Tree(SHA-256, Poseidon)

MultiSwap: B-INS-ARiSA vs MerkleSwap vs [OWWB][2)]

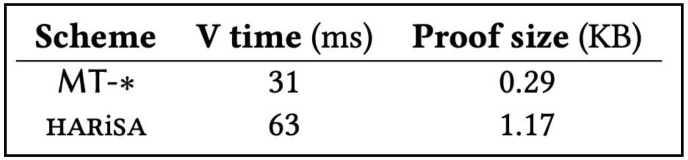
2) Scaling Verifiable Computation Using Efficient Set Accumulators, Alex Ozdemir, Read S. Wahby, Barry Whitehat, and Dan Boneh, Usenix Security 2020

3. Evaluation

❖ Evaluation for Batch Membership



General purpose batch membership

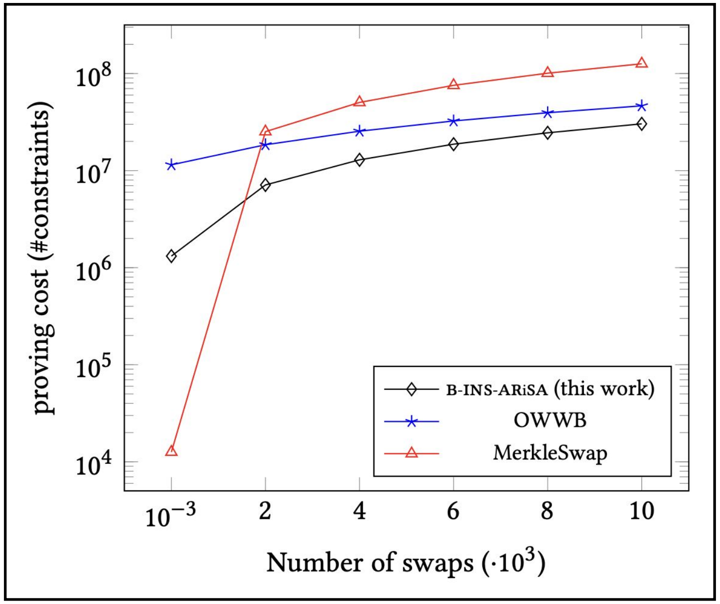


Scheme	V time (ms)	Proof size (KB)
MT-∗	31	0.29
HARiSA	63	1.17

14~33x faster than MT-Pos-16

5x smaller of CRS, **less** RAM consumption

❖ Evaluation for Batch Updates



Benchmark for batch updates

Proving costs

Verification time/Proof size

Proof size: B-INS-ARiSA: 1.4KB , MerkleSwap/OWWB: 288B

Ver time: B-INS-ARiSA: 120ms, MerkleSwap/OWWB: 30ms

Conclusion

Summary

Scalable solution for proving zero-knowledge batch membership succinctly

New techniques for RSA accumulator + zk-SNARKs

Applying our technique to **batch updates**

Evaluation

Batch membership: Much faster proving time than Merkle tree

MultiSwap: Surpass Merkle tree over 140 swaps

More in paper

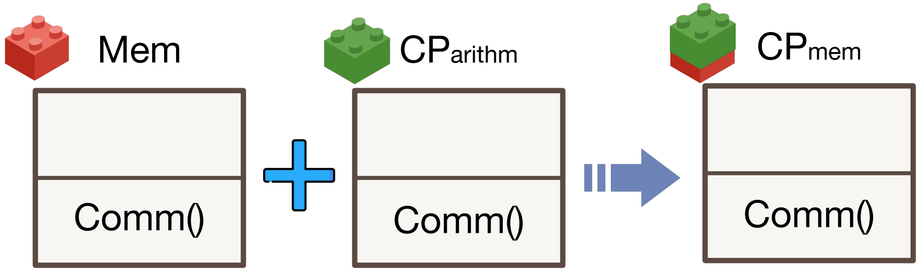
DID application: Much faster proving time than Merkle tree on the realistic scenario

Full security proofs

Discussion

❖ Commitment

Commit-and-Prove[1]



Commit-and-Pro
ve
Enc-and-Prove

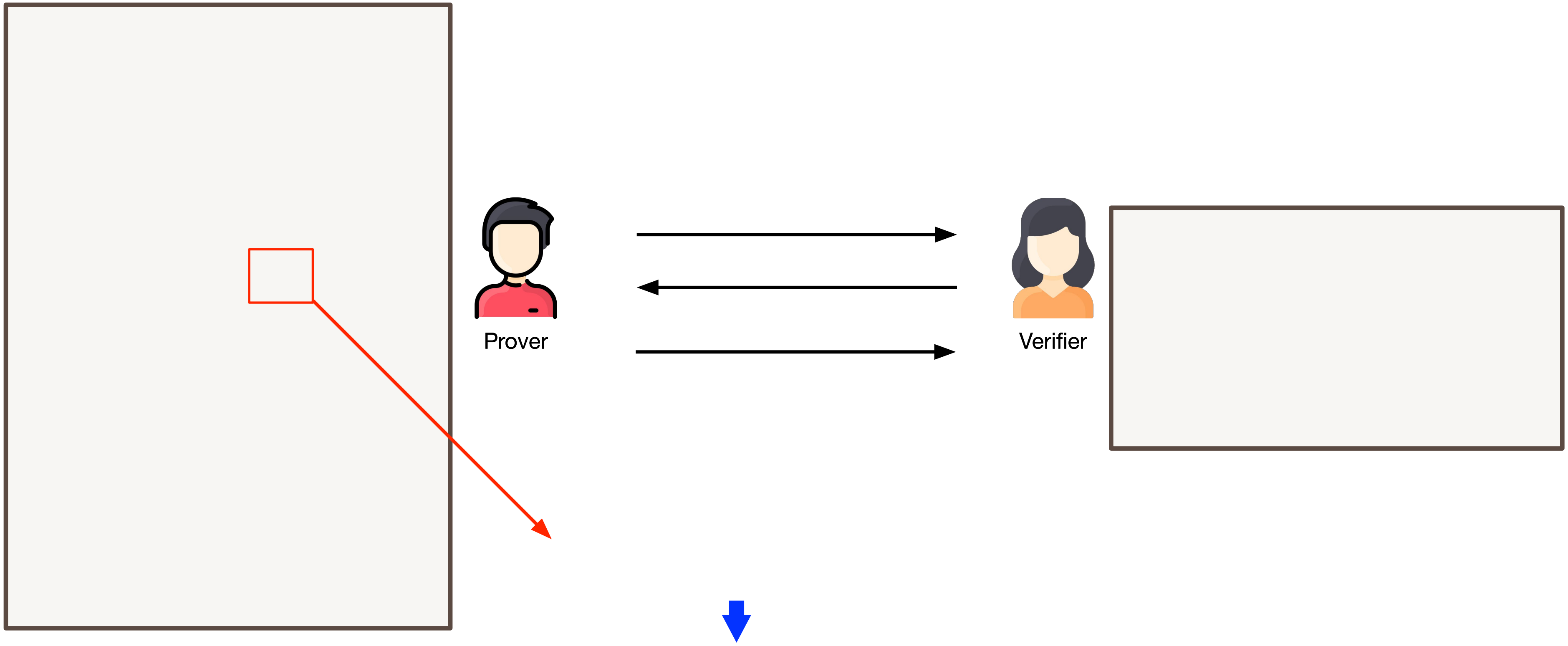
Harisa
:

[1] LegoSNARK: Modular Design and Composition of Succinct Zero-Knowledge Proofs, Matteo Campanelli, Dario Fiore, and Anaïs *Querol*, ACM CCS 2019

Thank you for listening

Appendix

❖ Witness aggregation

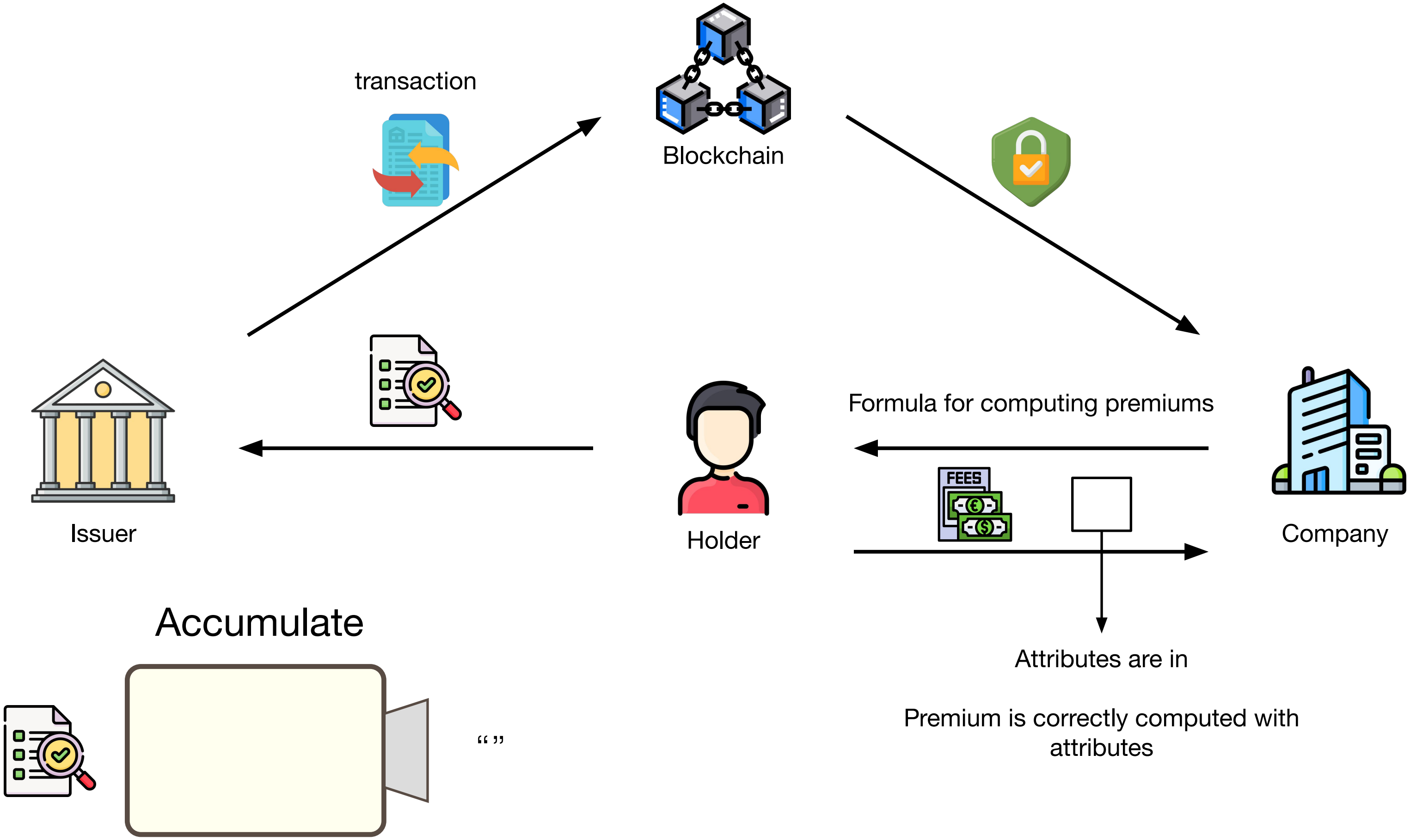


UTXO-like settings: Users hold **precomputed witness** and **update** it

Aggregation with Shamir’s trick GCD computation for batch size=

Appendix

❖ Scenario for DID application



transaction

Blockchain

Issuer

Holder

Company

Formula for computing premiums

Attributes are in

Premium is correctly computed with attributes

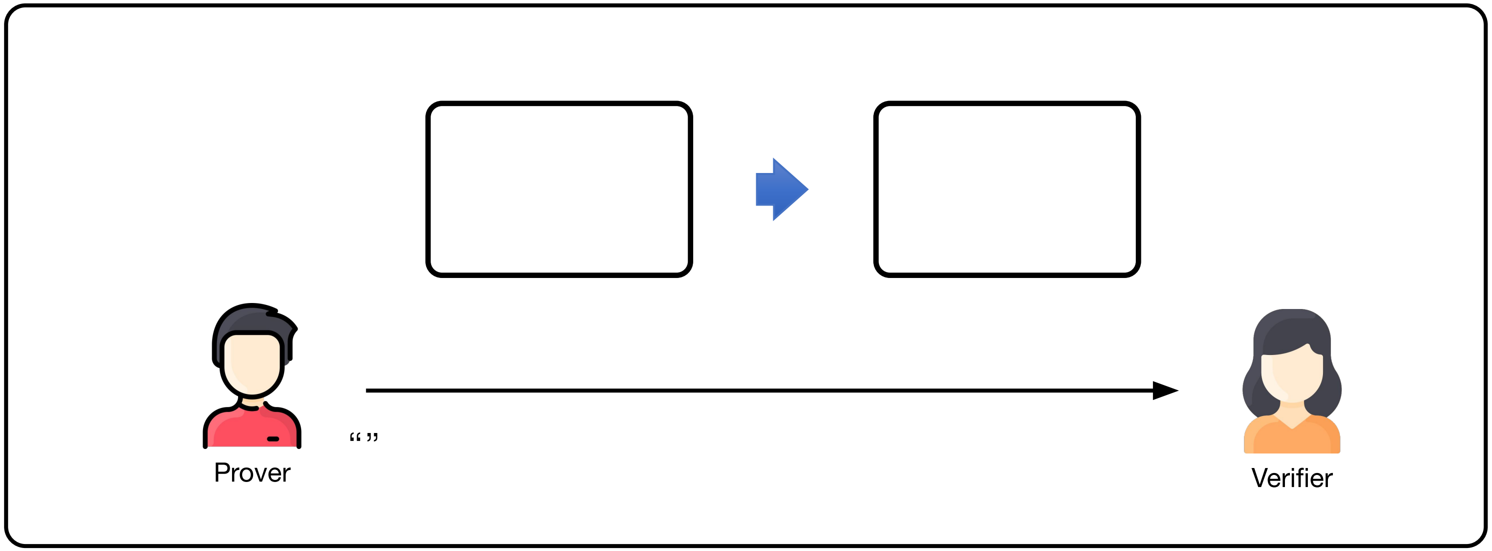
Accumulate

“ ”

Appendix

❖ **Set updates**

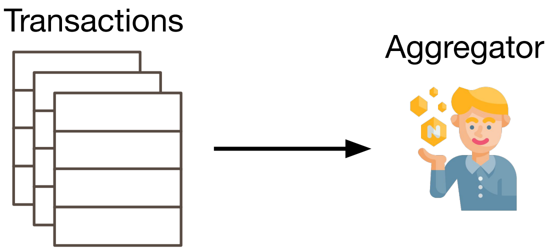
- Proving updates where updated set is from removing element and adding from/to existing set :
- Batch updates: proving updates for batch elements
- Additional property is also proven with updates proof



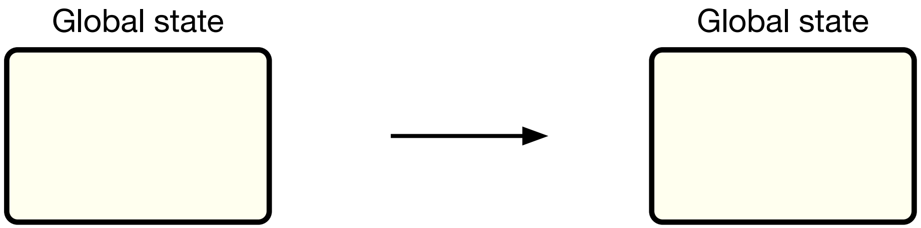
Appendix

❖ Set updates in blockchain

- In blockchain, set updates can be used in zk-rollup



- An aggregator collects transactions
- Proves that those transactions are valid
- Updates global state to
- Proves the correctness of new global state

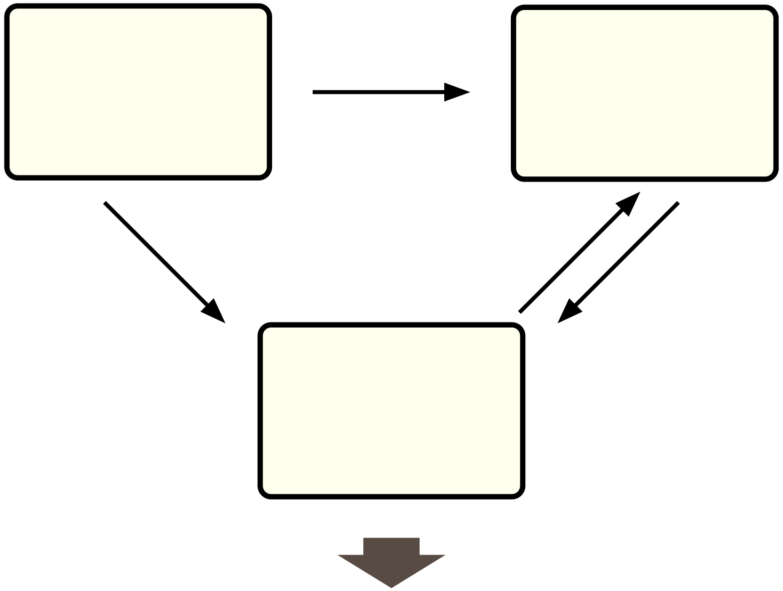


Multiswap[1]
)

[1] Scaling Verifiable Computation Using Efficient Set Accumulators, Alex Ozdemir, Riad S. Whaby, Barry Whitehat, and Dan Boneh, USENIX Security 2020

Appendix

❖ **Insertion to Multiswap**



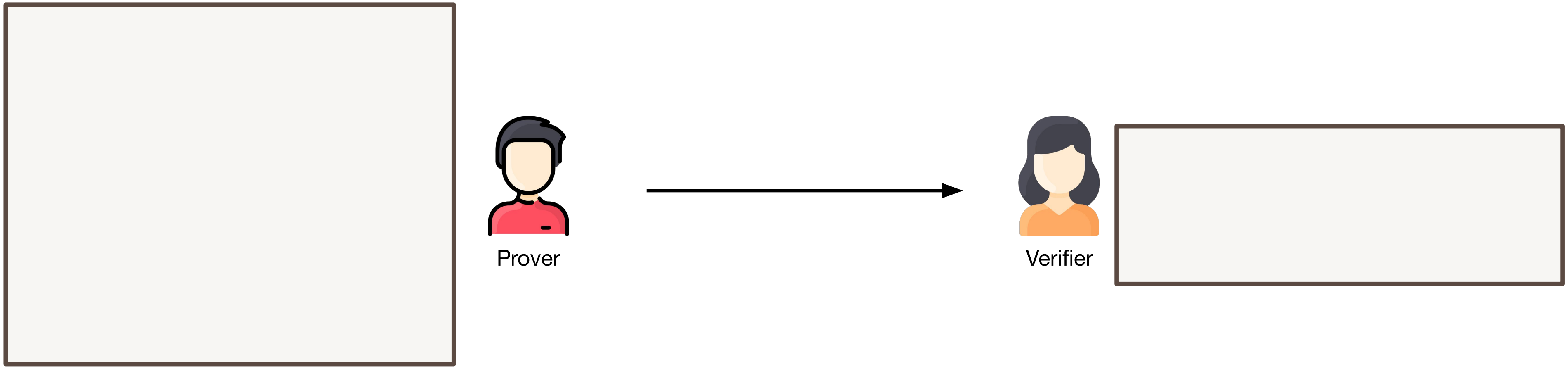
Appendix

- **B-INS-ARiSA**

↓

Scaling down HARiSA

Goal: Prove that

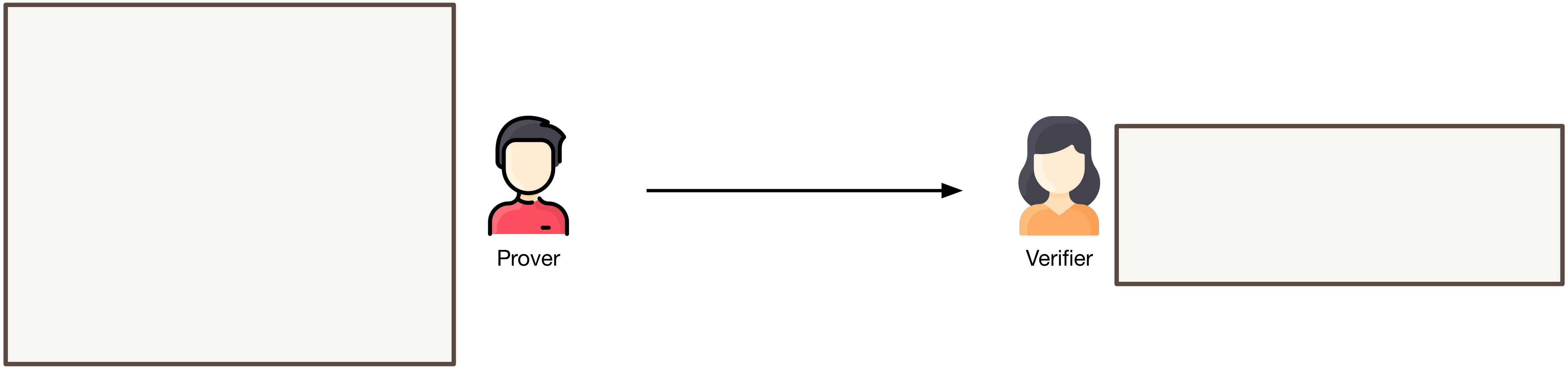


Appendix

❖ **B-INS-ARiSA**

Goal: Prove that

Are the updated elements in correct domain?



Appendix

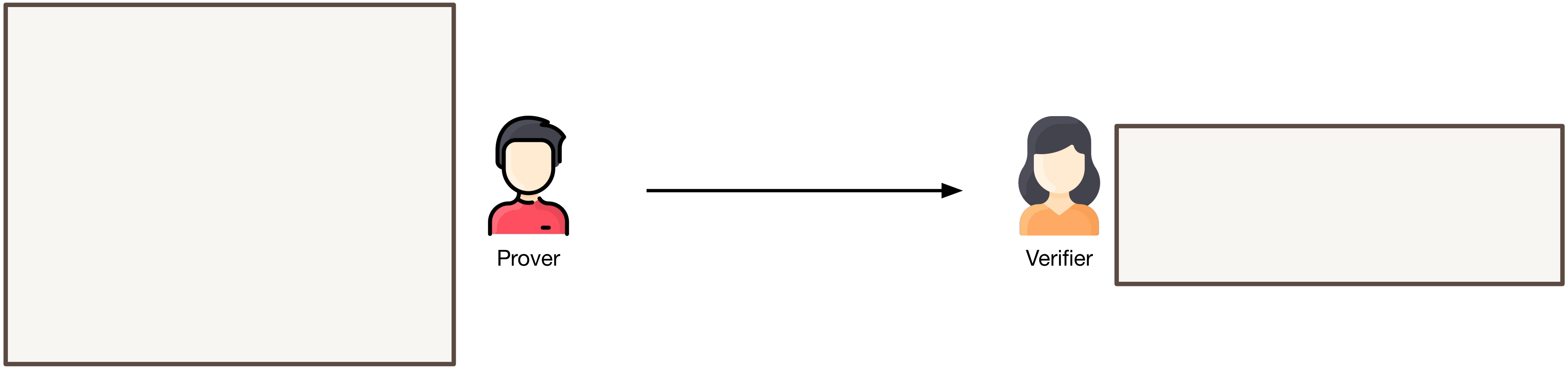
❖ **B-INS-ARiSA**

Goal: Prove that

Division intractable hash

: 2,048 bits offset

→ : Collision-resistant hash function



Appendix

❖ **B-INS-ARiSA**

Goal: Prove that