zkSNARKs for Virtual Machines are Non-Malleable

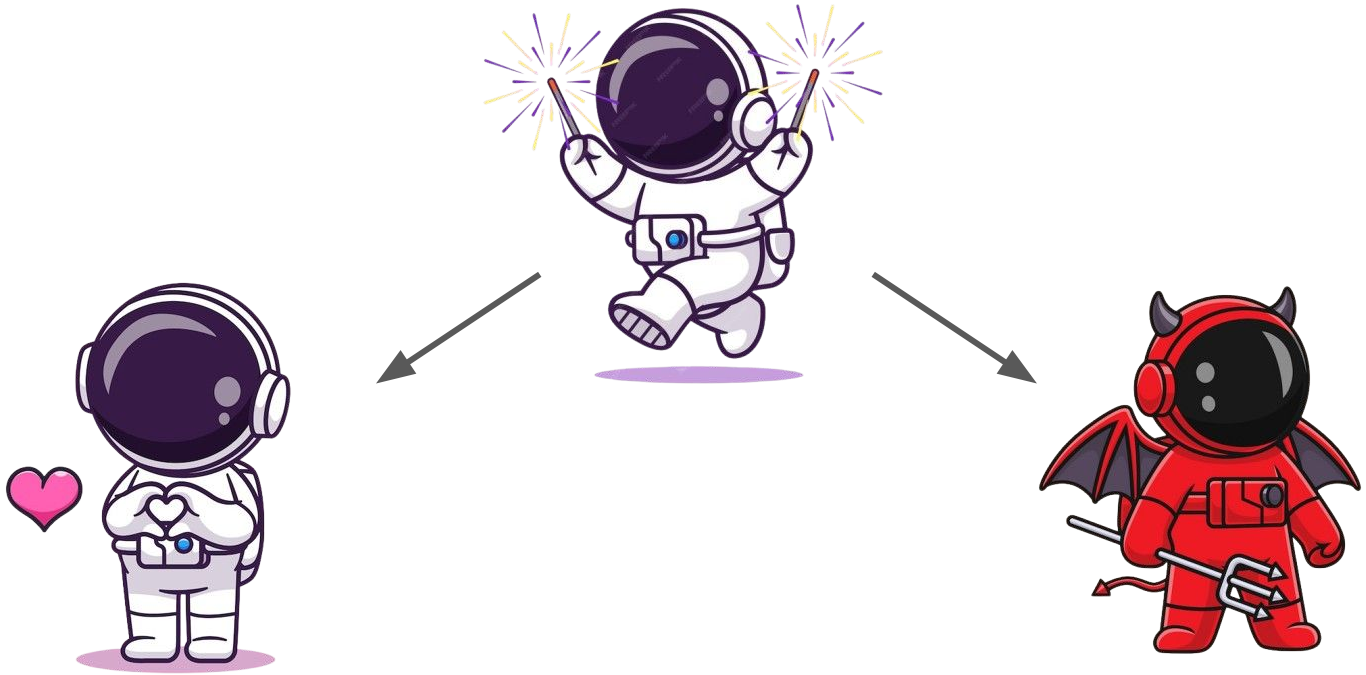
Matteo Campanelli, Antonio Faonio, **Luigi Russo**

Malleability in Cryptography

Métamorphose de Narcisse, Salvador Dalí (1937)

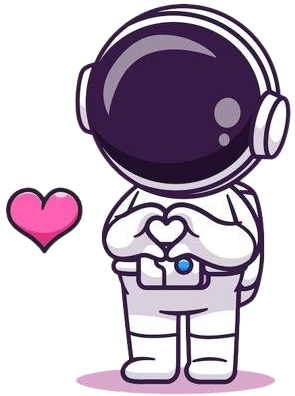
"Efficiently transforming a cryptographic object into a new valid one"

The double face of Malleability



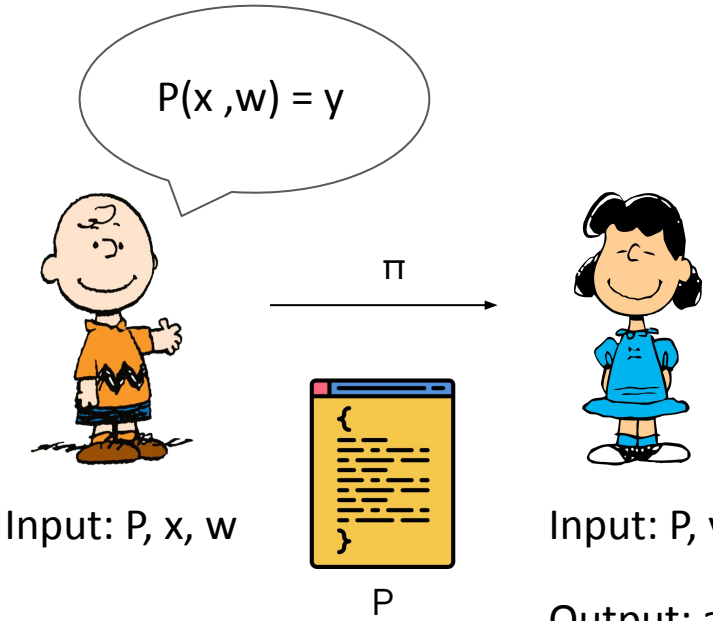
The double face of Malleability: (zk)SNARKs

The previous talk



This talk

Proofs of program execution



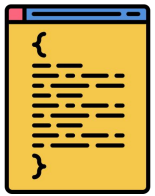
zkSNARK

- Zero-knowledge
- Non-Interactive
- Succinct
- Knowledge-sound

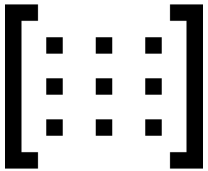
zkSNARKs: frontends and backends

Frontend
Converts program to a mathematical IR

Backend
Proves that the IR is satisfied



C++, Rust, Java

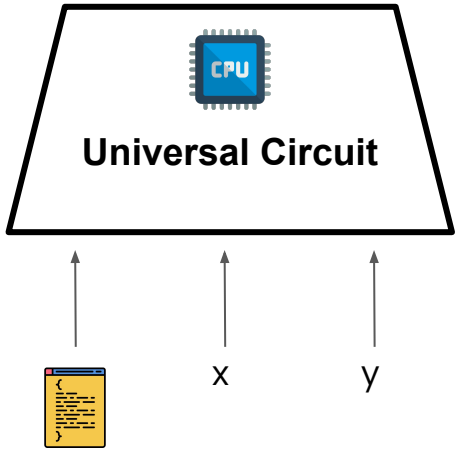


R1CS, Plonkish,CCS

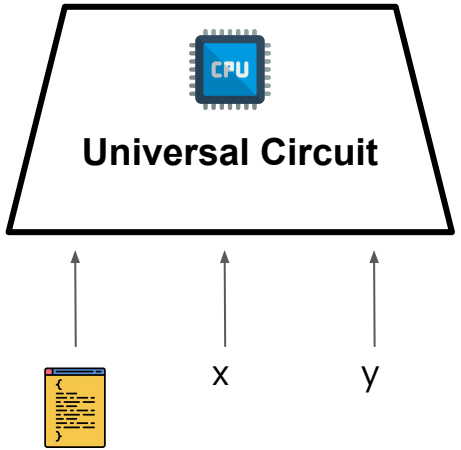
Groth16, Plonk, Spartan

Frontend: zkVM

Frontend: zkVM



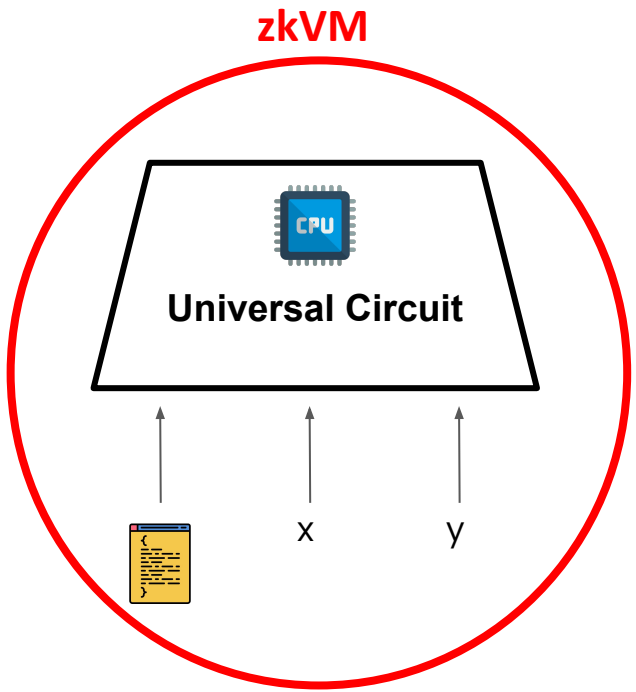
Frontend: zkVM



+ Auditing and formal verification on one circuit

+ Re-use existing languages and tooling

Frontend: zkVM



+ Auditing and formal verification on one circuit

+ Re-use existing languages and tooling

Applications & Use Cases

Generic
- Proof of solvency
- Image provenance
- Content moderation

Blockchain-related
- Private transactions
- Private smart contracts
- ZK-Rollups

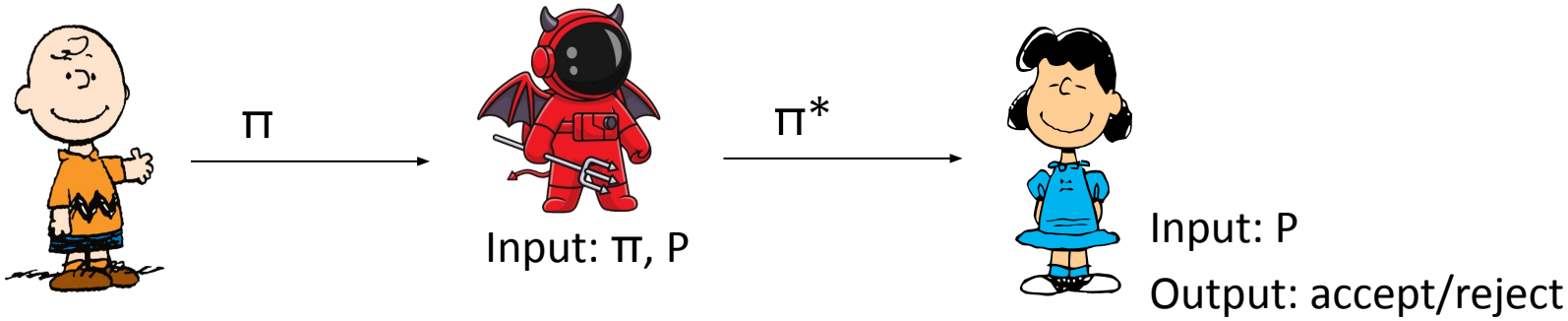
Applications & Use Cases

Generic
- Proof of solvency
- Image provenance
- Content moderation

Blockchain-related
- Private transactions
- Private smart contracts
- ZK-Rollups

Can old proofs be useful to the adversary?

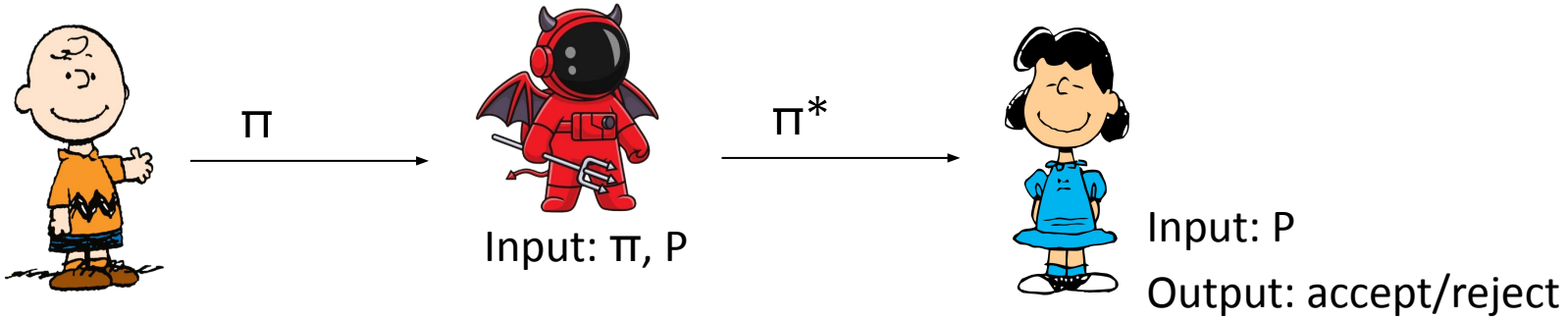
Malleability of zkSNARKs



Malleability attack

Modify an existing proof into a new proof without knowing the witness

Malleability of zkSNARKs

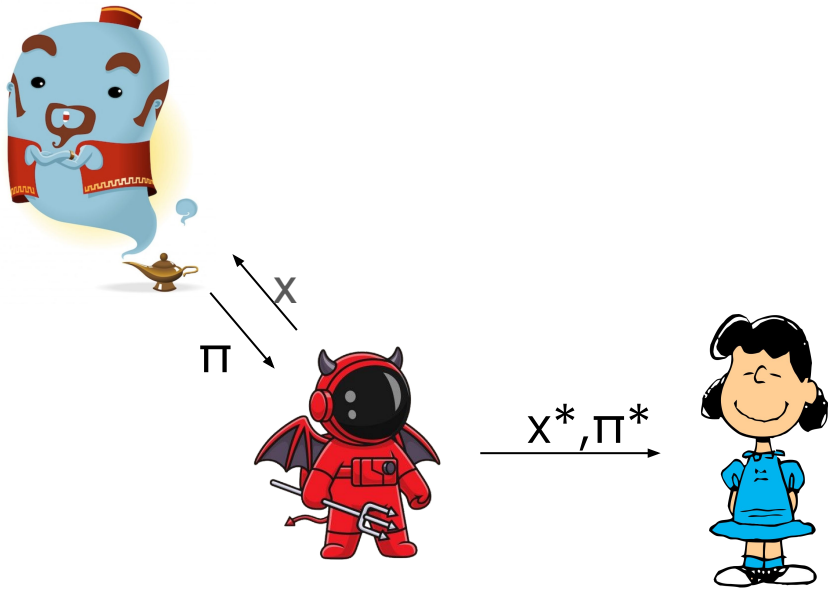


Malleability attack

Modify an existing proof into a new proof without knowing the witness

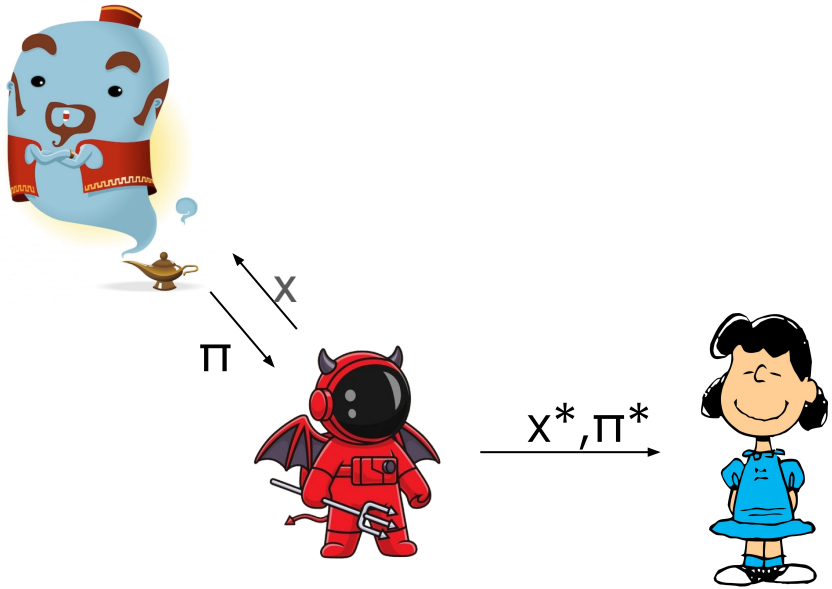
Not ruled out by zero-knowledge and knowledge soundness

Non-Malleability of zkSNARKs



Non-Malleability of zkSNARKs

= Simulation Extractability



Non-Malleability of existing zkSNARKs

The complexity of a zkVM

A universal circuit is large
It must be able to execute any operation at each step, e.g. RISC-V has 50 operations

```
switch(instruction) {
  case ADD: {...}
  case XOR: {...}
  ...
  case SHIFT: {...}
}
```

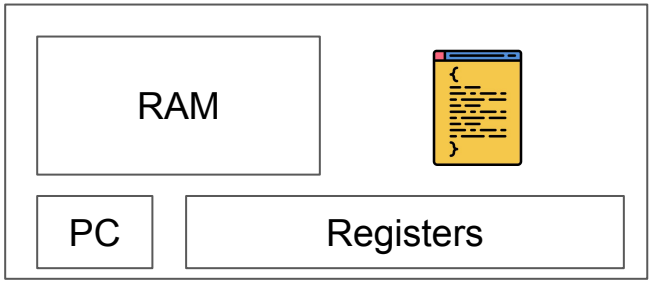
The complexity of a zkVM

A universal circuit is large
It must be able to execute any operation at each step, e.g. RISC-V has 50 operations

```
switch(instruction) {
  case ADD: {...}
  case XOR: {...}
  ...
  case SHIFT: {...}
}
```

It's not only about instructions
The zkSNARK-Prover proves that:

- The memory is consistent throughout the entire computation
- The fetch and decode are correctly executed



Our Results

1. A “natural” (and truly zero-knowledge) zkVM, inspired by Jolt

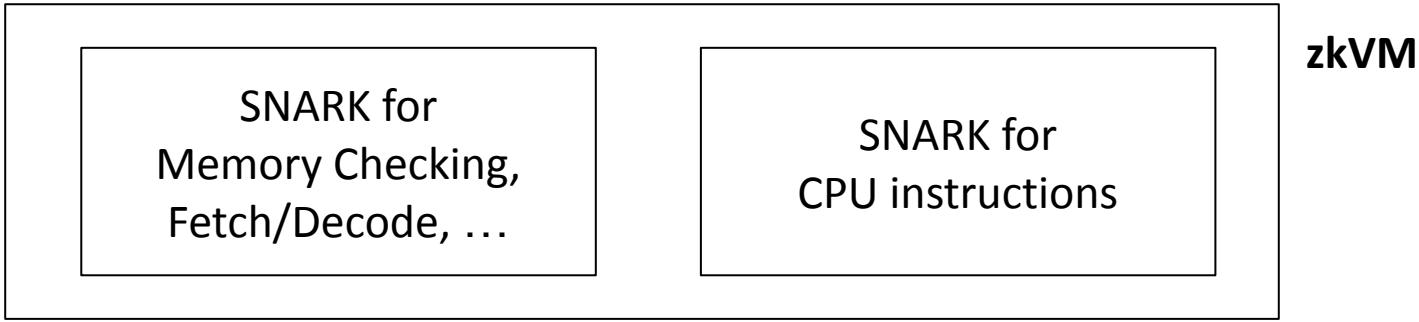
2. A framework for the non-malleability of “composable” zkSNARKs

Our Results

1. A “natural” (and truly zero-knowledge) zkVM, inspired by Jolt

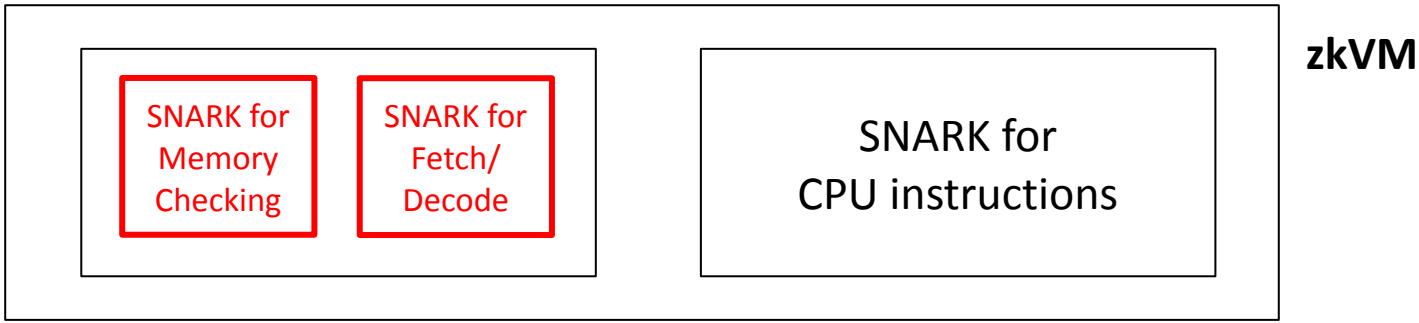
2. A framework for the non-malleability of “composable” zkSNARKs

Joltish: a modular zkVM

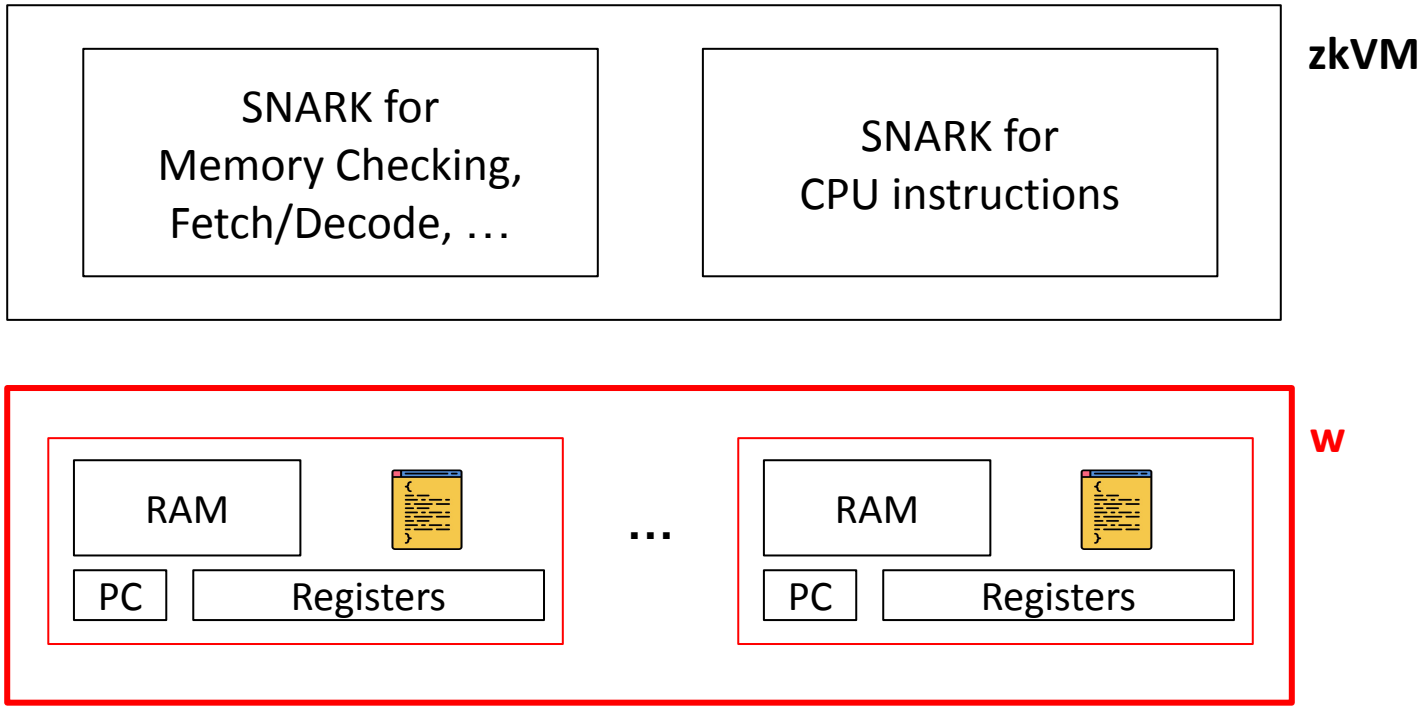


- Many zkVM's designs are modular (it's just natural)
- **Jolt** [AST24] is the first zkVM based on *the lookup-singularity*
- Our Joltish: inspired by Jolt, but not quite Jolt

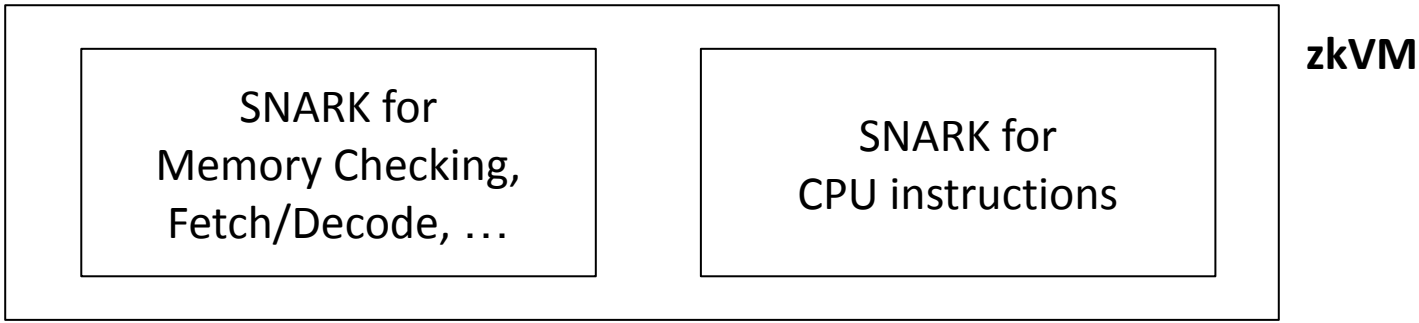
zk Jolt from Joltish



Joltish: a modular zkVM

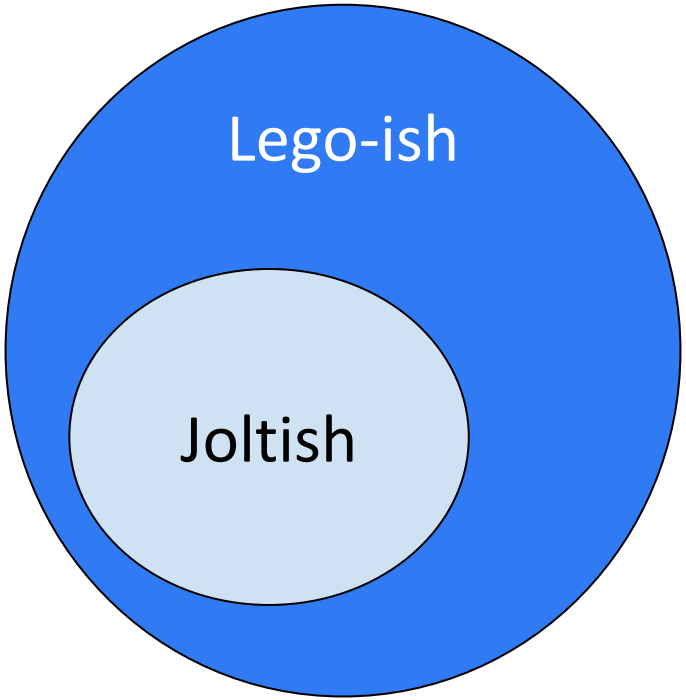


Joltish: a modular zkVM

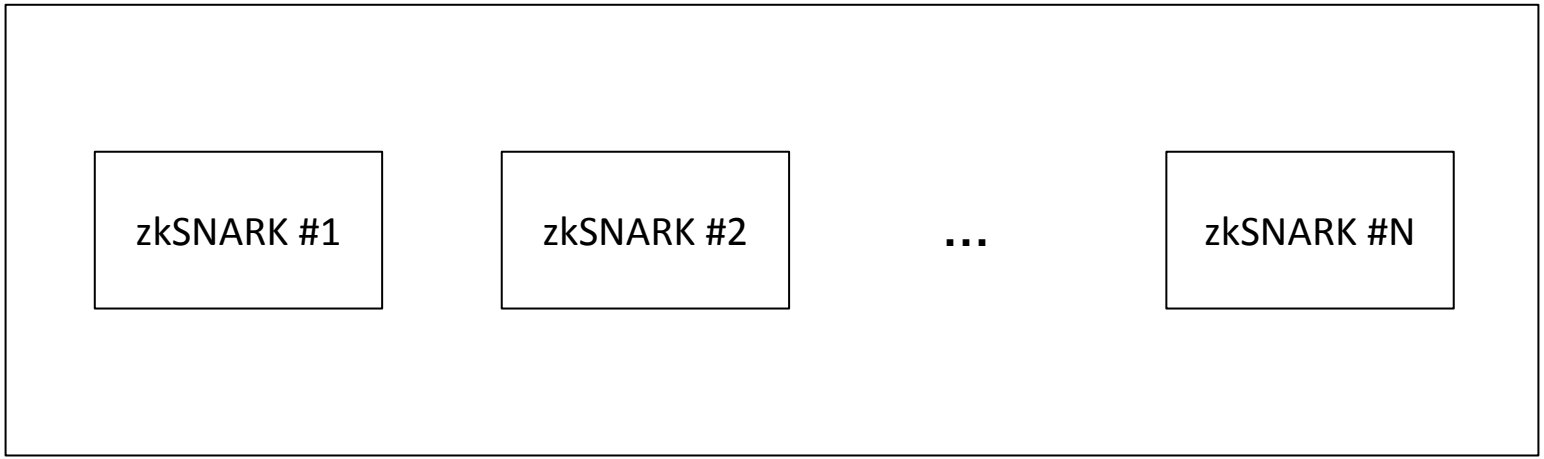


What are the conditions for the non-malleability of Joltish?

From *Joltish* to LEGO-ISH

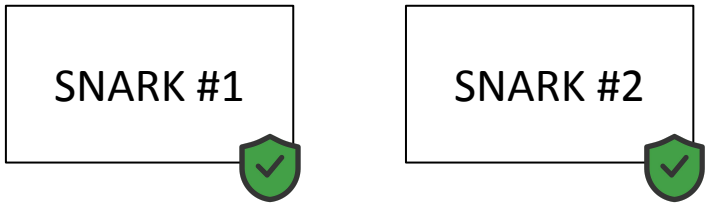


Lego-ish: a modular zkSNARK

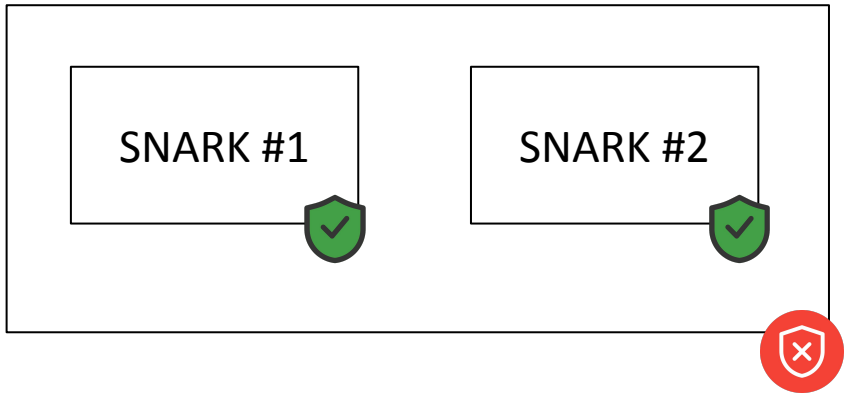


What are the conditions for the non-malleability of modular zkSNARKs?

Non-Malleability challenges



Non-Malleability challenges



Copy & Paste attacks
Composition of non-malleable SNARKs may be malleable

Our Results

1. A “natural” (and truly zero-knowledge) zkVM, inspired by Jolt

2. A framework for the non-malleability of “composable” zkSNARKs

Commit-and-Prove Relations

A commit-and-prove relation (c, x; w) in R iff:

- P(x,w) = 1
- c = Commit(w)

Commit-and-Prove Relations

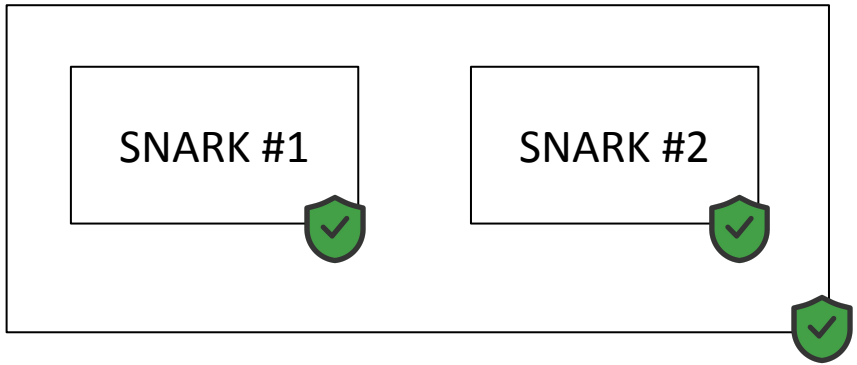
A commit-and-prove relation (c, x; w) in R iff:

- P(x,w) = 1
- c = Commit(w)

Conjunction of R_1 and R_2 with shared witness w:

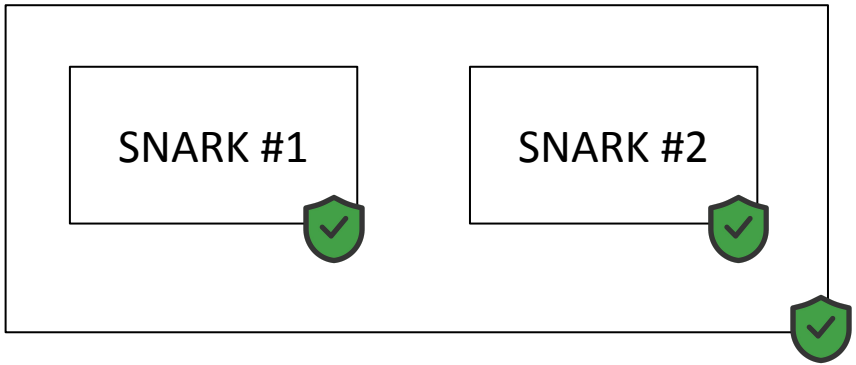
- Instance (c, x_1, x_2)
- P1(x_1, w) = 1 AND P2(x_2, w) = 1 AND c = Commit(w)

When is Conjunction Non-Malleable?



2 (slightly different) Non-Malleable Compositions

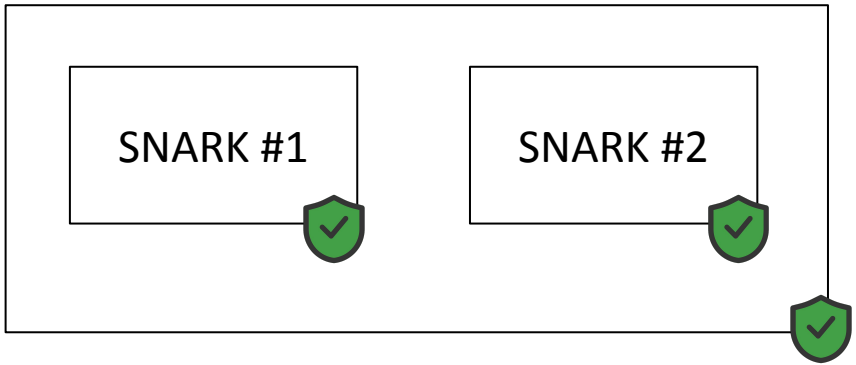
When is Conjunction Non-Malleable? **(1)**



If SNARK #1 AND SNARK #2 are **both**:

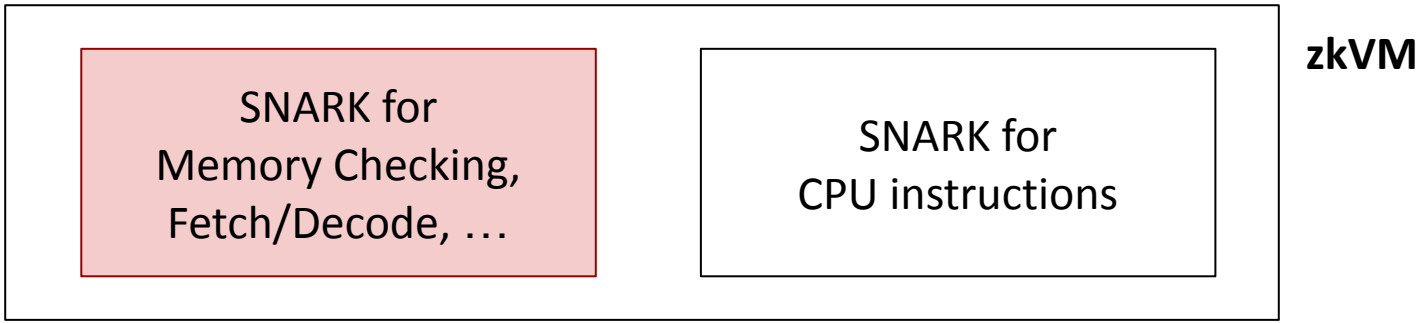
Trapdoorless zero-knowledge and non-malleable

When is Conjunction Non-Malleable? **(2)**



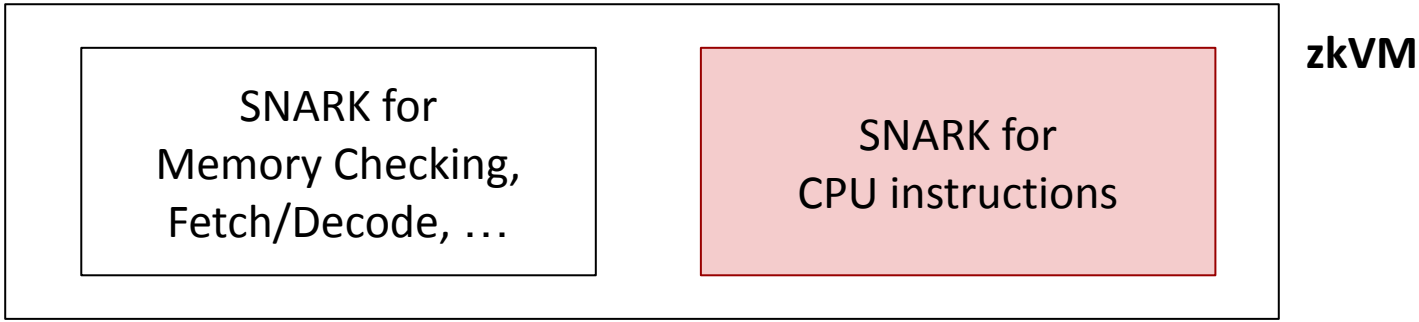
- SNARK #1 is witness-indistinguishable,
  efficient witness-computable (*easy to find w for x1*)
- SNARK #2 is Trapdoorless zero-knowledge, non-malleable SoK

On the Non-Malleability of Joltish



- Witness-indistinguishable SNARK for Memory Checking, Fetch/Decode, …

On the Non-Malleability of Joltish



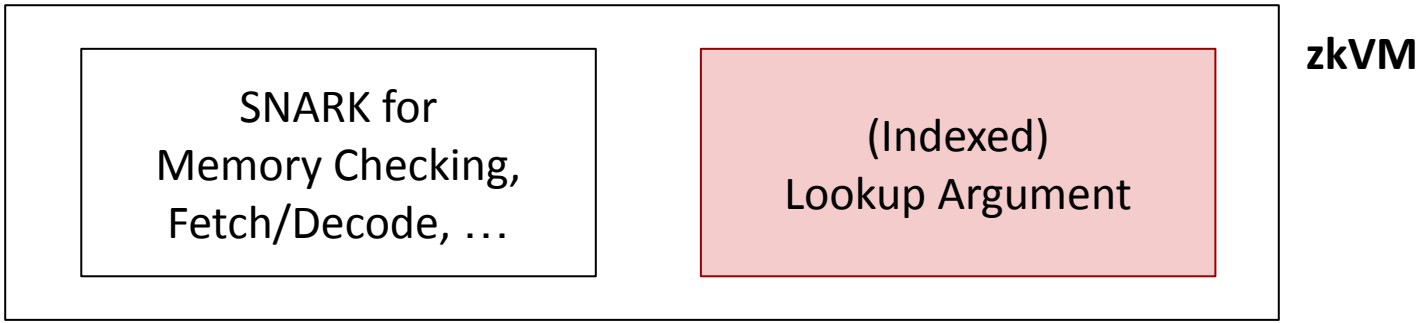
- Witness-indistinguishable SNARK for Memory Checking, Fetch/Decode, …

- Non-Malleable zkSNARK for CPU instructions

On the Non-Malleability of Joltish

- Statement: (P,x,y) s.t. P(x)=y on RISC-V architecture.
- ZK-Simulator for (P,x,y) s.t. P(x)=y'≠ y on RISC-V architecture
  1) Hack the instructions set s.t. P(x)=y
  2) Run P(x)=y on hacked architecture to obtain program trace W
  3) W good trace for (P,x,y) for SNARK#1 => Run Prover#1
  4) W not good witness for P(x)=y for SNARK#2 => Simulated Proof

The Lookup Singularity based zkVM



- Witness-indistinguishable SNARK for Memory Checking, Fetch/Decode, …

- Non-Malleable Lookup Argument

Lookup Arguments

- Lookup Arguments prove that committed vector F is sub-vector of big table T
  - |F| << |T|
  - prover complexity is proportional to |F|

Lookup Arguments

- Lookup Arguments prove that committed vector F is sub-vector of big table T
  - |F| << |T|
  - prover complexity is proportional to |F|

- Lookup Argument in Jolt is Lasso [STW24]
  - It can handle very big table as big as truth tables of all RISC-V instructions

zk-Lasso

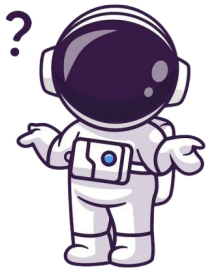
- Define a zero-knowledge version of Lasso
- Prove Sim-Extractability: based on the framework of [FKMV12]

  (Trapdoorless ZK + Unique Response + Special Soundness => Sim Ext)

- We extend and improve over [DG23]
  - (Lasso is based on Spartan)

Future Work

- Non-Malleability w/o simulation extractability?
- Non-Malleability for composition of Reduction-of-Knowledge: very natural!
- Non-Malleability of other Lookup Arguments?



Thank you

ia.cr/2024/1551

References

[AST24] Arasu Arun, Srinath Setty, Justin Thaler. Jolt: SNARKs for Virtual Machines via Lookups. EUROCRYPT 2024

[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

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the Non-malleability of the Fiat-Shamir Transform. INDOCRYPT 2012

[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 2024

[STW24] Srinath Setty, Justin Thaler, Riad Wahby. Unlocking the lookup singularity with Lasso. EUROCRYPT 2024

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.