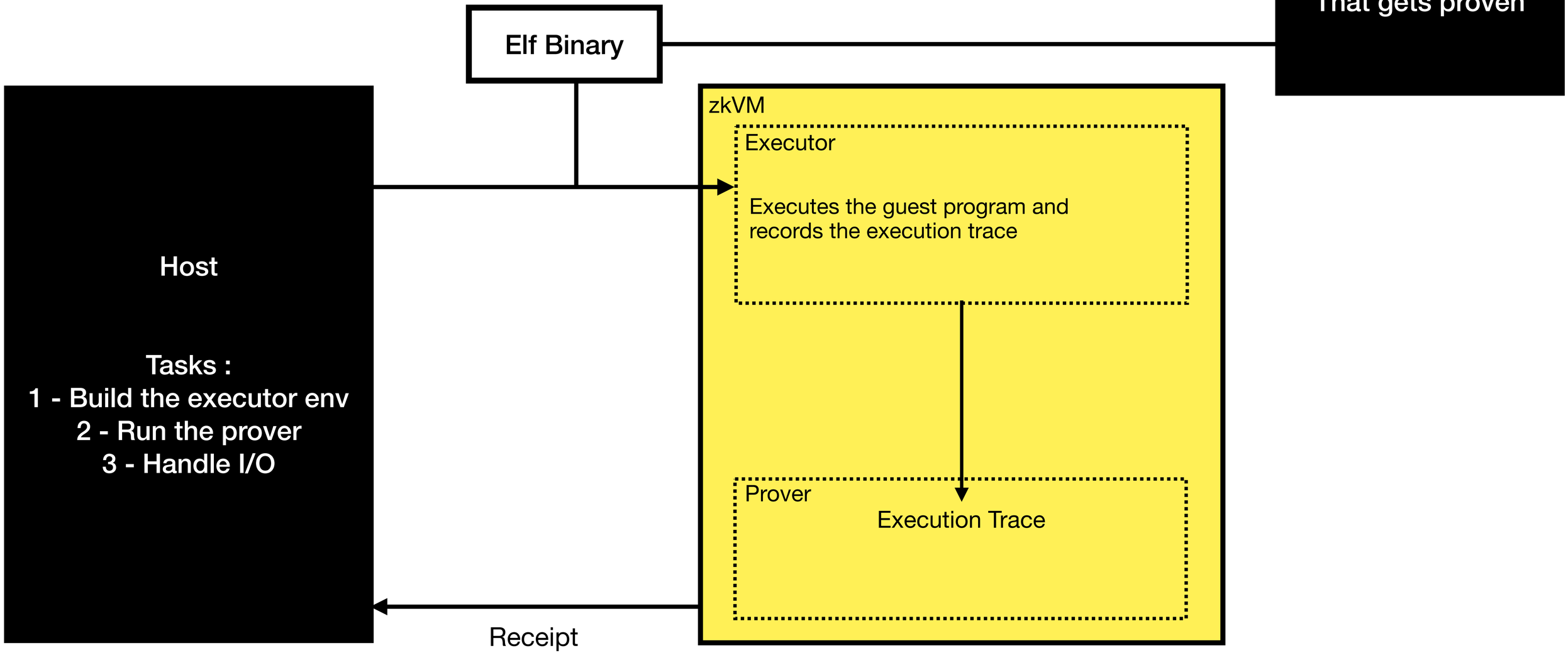
Implementation of verifiable programs

Starkware , risczero

Joseph Boulos

Risc zero

Program architecture



The receipt

Returned by the zkVM , contains proof and journal

- cryptographic proof that a specific program executed correctly within the RISC Zero zero-knowledge virtual machine (zkVM)
- **Journal**: Contains the public outputs of the program.
- **Seal**: An opaque blob that cryptographically attests to the validity of the receipt.

Image ID: A cryptographic identifier derived from the program's binary, ensuring the receipt corresponds to the expected code.

Verifiable Image transformations

Risc Zero implementation

- Claim : I know an image img1, such that this public image img2 is the result of t1 , t2 .., ti applied to img1
- Straight forward approach of proving it : release img1 and let everyone check for themselves by applying the transformations and comparing the result with the public image
- But sometimes , the initial image should be kept private (public image is a blurred version for covering sensitive data etc..)

Different ways of handling the problem

Prover side

Host :

- build the executor env by providing :

1 - the image

2 - the list of transformations

Guest :

1 - read image and transformations provided by host

2 - apply transformations to the image

3 - write transformed image back to the host

4 - commit transformed image along with the transformations to the journal

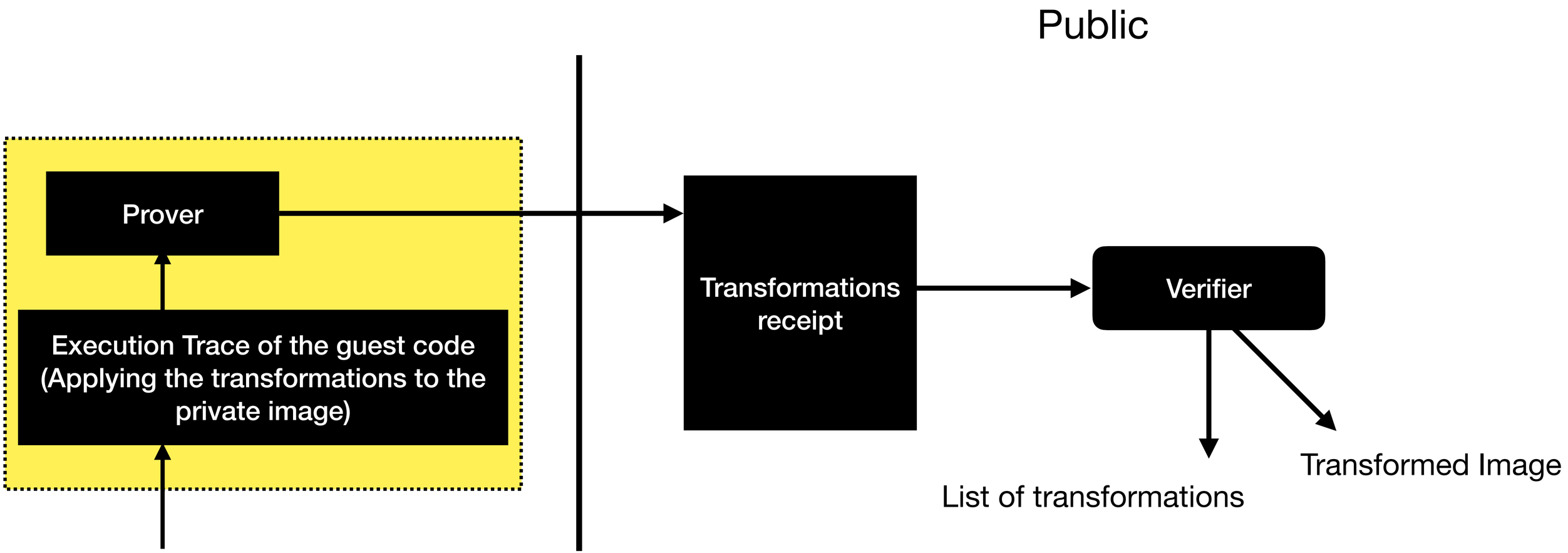
Verifier side

Has access to the receipt and image ID of the guest code :

- extracts journal
- Compares image and transformations with the ones mentioned in the claim

Example

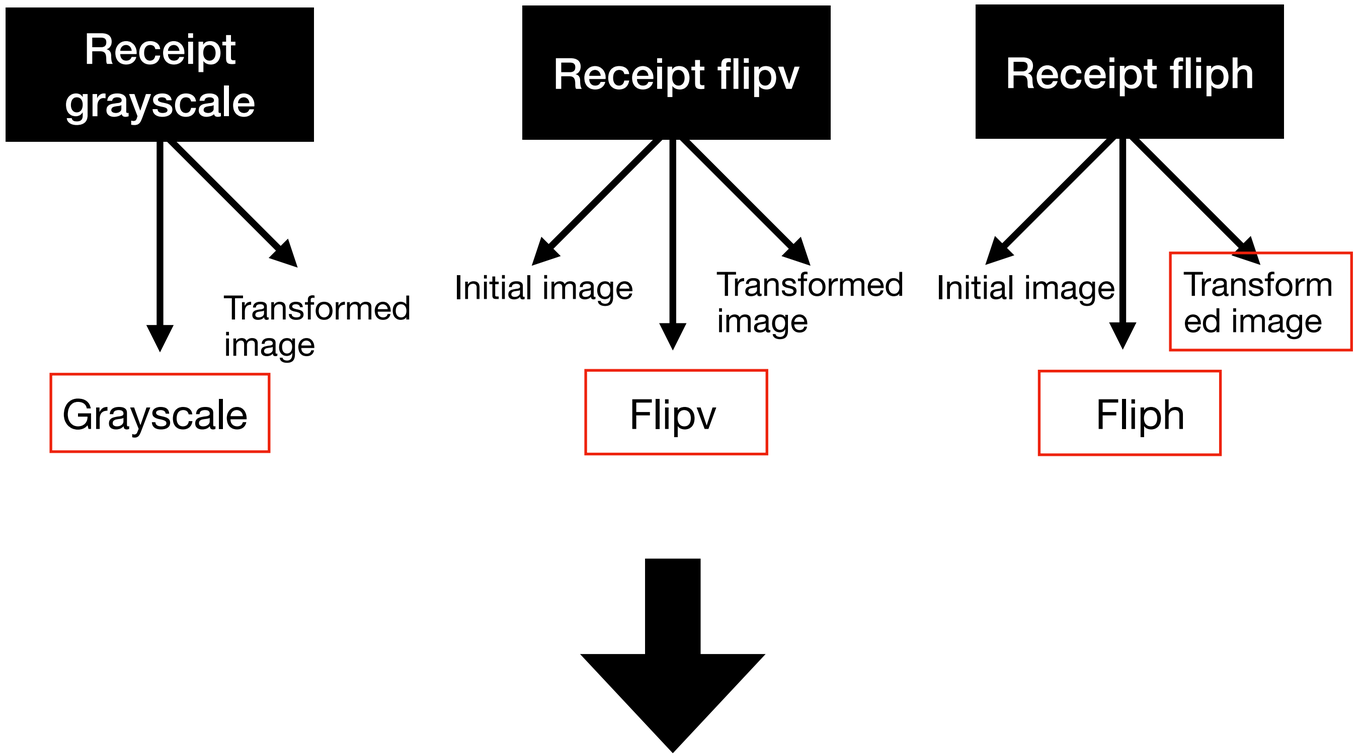
I want to prove that I know a private image img1 such that this public version img2 is the result of a grayscale , flipv , fliph applied to img1



Using proof composition —chained mode

Prover side :

Aggregate receipts and add them as assumptions*



Final published receipt

Load initial receipt

Verify it , and extract journal (using ZK_IMG_METHOD_ID)

Store transformed image

Add transformation to transformation set

For every other receipt :

- verify receipt_i
- Extract journal
- Compare initial_image to transformed image
- If != panic else :
    - Add transformation to transformation set and update initial image to actual transformed image

Commit transformed image and transformations set

Using proof composition —parallel mode


Receipt


Receipt


Receipt


Receipt

Proof on the validity of the receipts , and the image reconstruction

Single receipt

Starknet

Using off chain Cairo for writing verifiable programs