PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations

Assa Naveh, Eran Tromer

TEL AVIV UNIVERSITY אוניברסיטת תל אביב

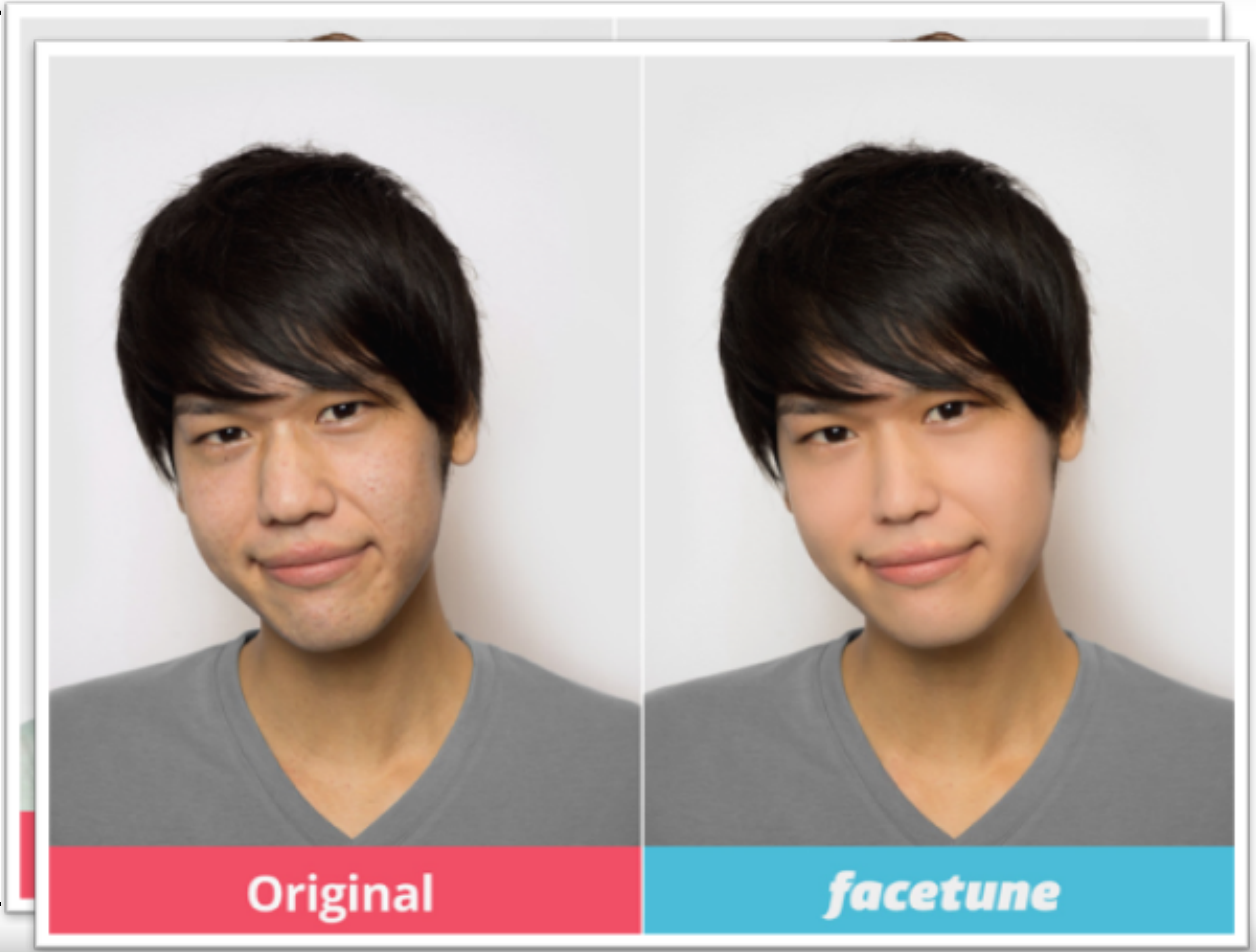
CHECK POINT INSTITUTE FOR INFORMATION SECURITY (CPIIS) · BROADCOM · THE LEONA M. AND HARRY B. HELMSLEY CHARITABLE TRUST · i-CORE ALGO

Photographic images as evidence of physical scenes

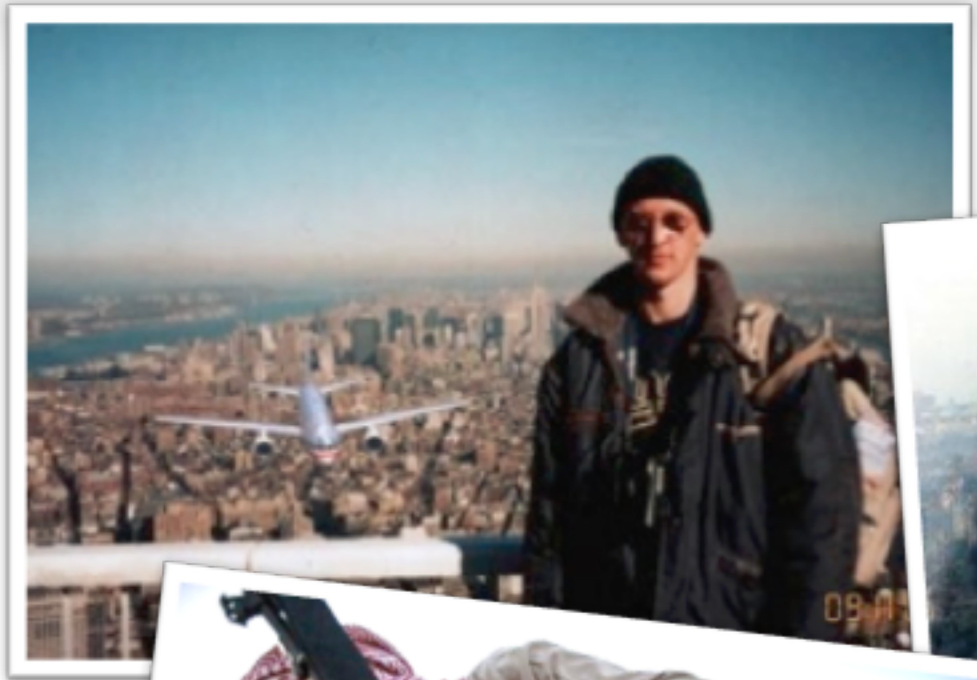
- Legal
- Journalism
- Politics
- Social
- Shaming/extortion
- Dating
- Cats

⋮

Trustworthy evidence?



More photographic “evidence”



DARPA

REUTERS

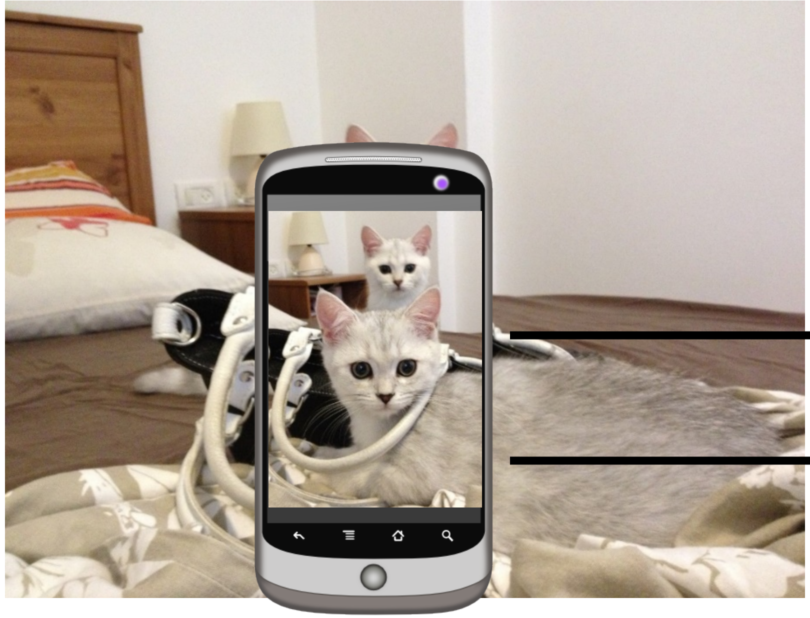
Image authentication scheme

Creator

Prover

Viewer

Verifier



image

proof

Naive solution?

- Camera signs images
  - Implemented by Nikon and Canon
  - (Very badly)

- Problem:
  Any change to the image will invalidate the signature

- Some changes may be considered legitimate:
  crop, rotate, jpeg-compress, resize, grayscale…

Goals

- Enable <u>any specified set</u> of permissible transformations
- Support a <u>sequence</u> of permissible transformation.
- Soundness
- Succintness
  - proofs are short and can be quickly verified.
- Proofs should be <u>zero-knowledge</u>
  - e.g., reveal no information about removed details.

Use-case: dating sites

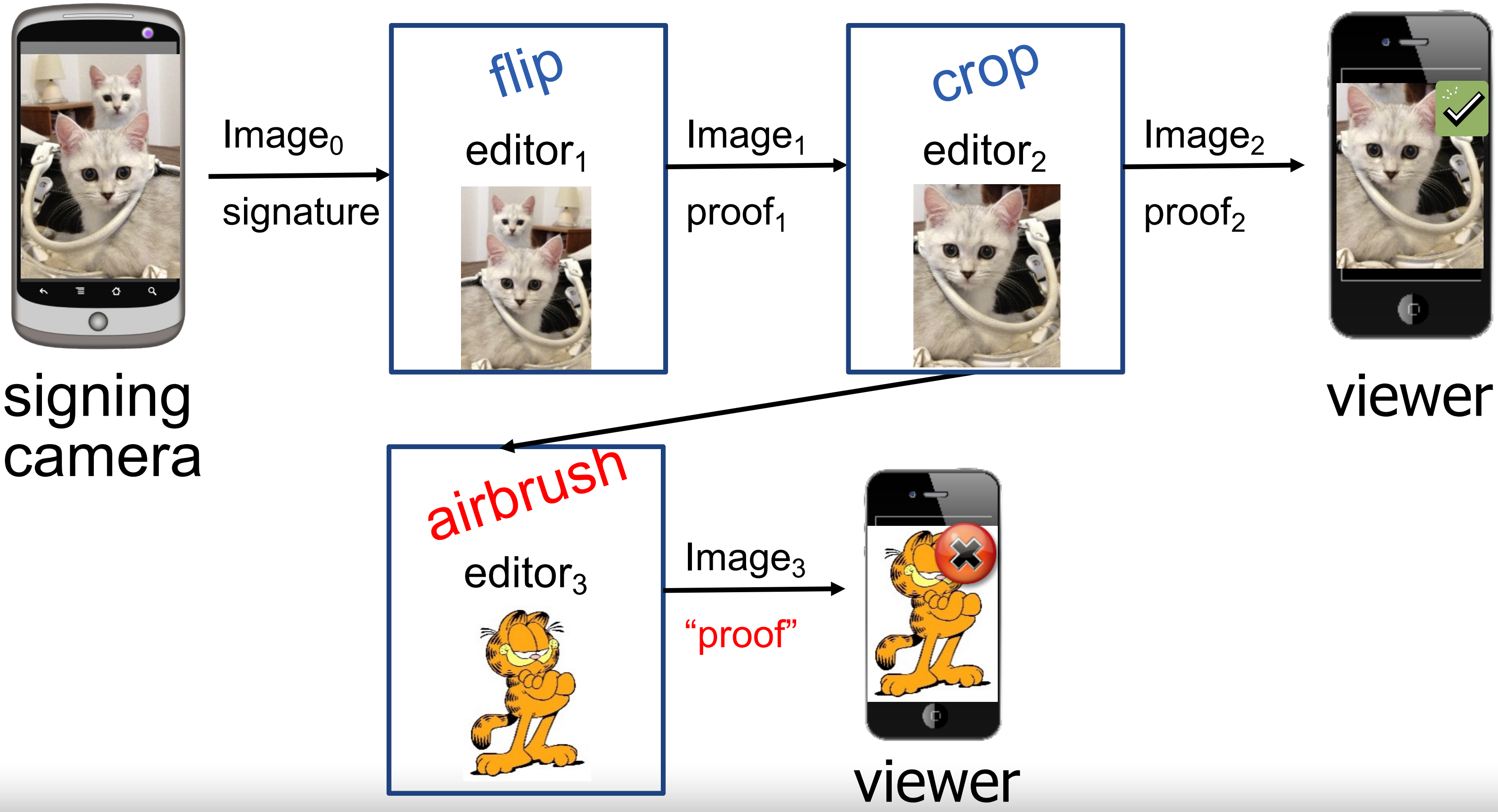
- Photos of users should reflect how they really look.
- They may go through some permissible editing, like cropping
- Cropped parts are not leaked (zero knowledge)

Image authentication: prior works

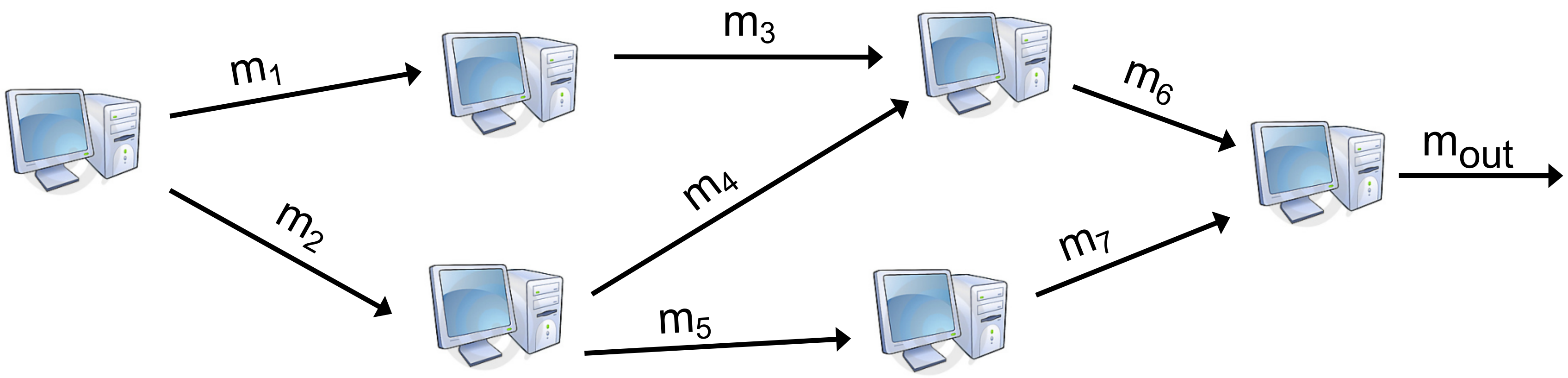
		transformations									
Paper	**Approach**	**JPEG**	**Rotate**	**Crop**	**Scale**	**Bright., contrast**	**Flip**	**Flexible spec**	**Small error**	**ZK**	**Size overhead**
	Digital signature								✓		O(1)
[Schneider Chang 1996], [Lin Chang 2001]	robust hash (JPEG coefficients)	✓									O(img)
[Fridrich Goljan 2000]	robust hash	✓				✓					O(1)
[Sun Sun Yao 2002]	semifragile watermark (SVD)	✓									O(img)
[Seo Haitsma Kalker Yoo 2004]	robust hash (Radon)	✓	✓	some							O(img)
[Feng Liu 2008]	robust hash (Bayesian)	✓			✓			some			O(1)
	⋮										
[Walsh 2012]	TPM + Nexus OS	✓	✓	✓	✓	✓	✓	✓	✓		O(edits)
This work	proof-carrying data	✓	✓	✓	✓	✓	✓	✓	✓	✓	O(1)

PhotoProof

Setup stage: system administrator defines permissible transformations and distribute keys.

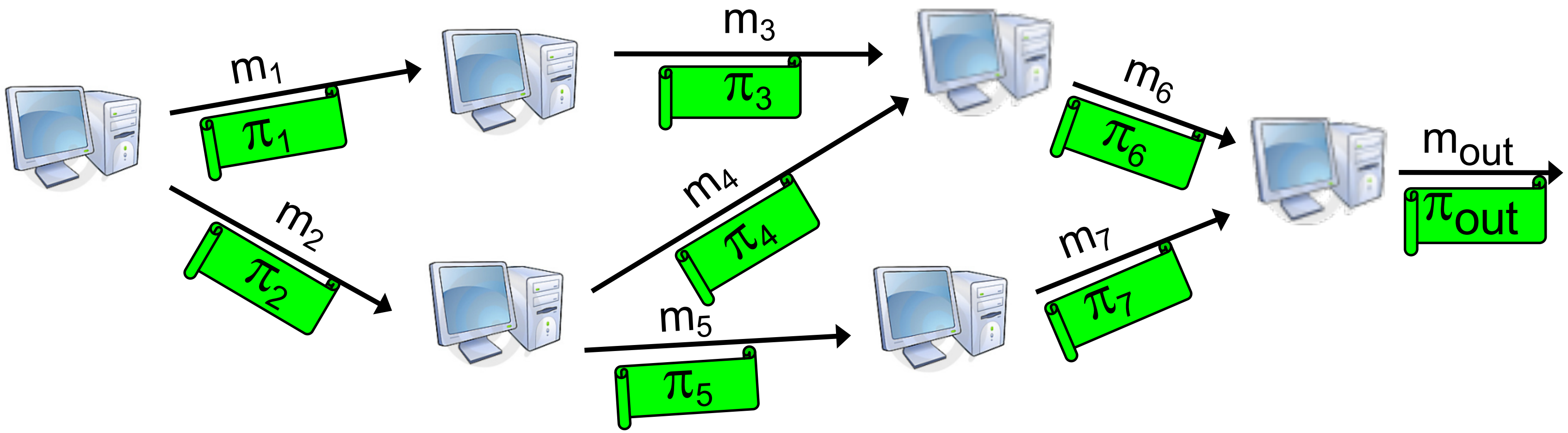


Approach:
Proof-Carrying Data [Chiesa Tromer 2012]



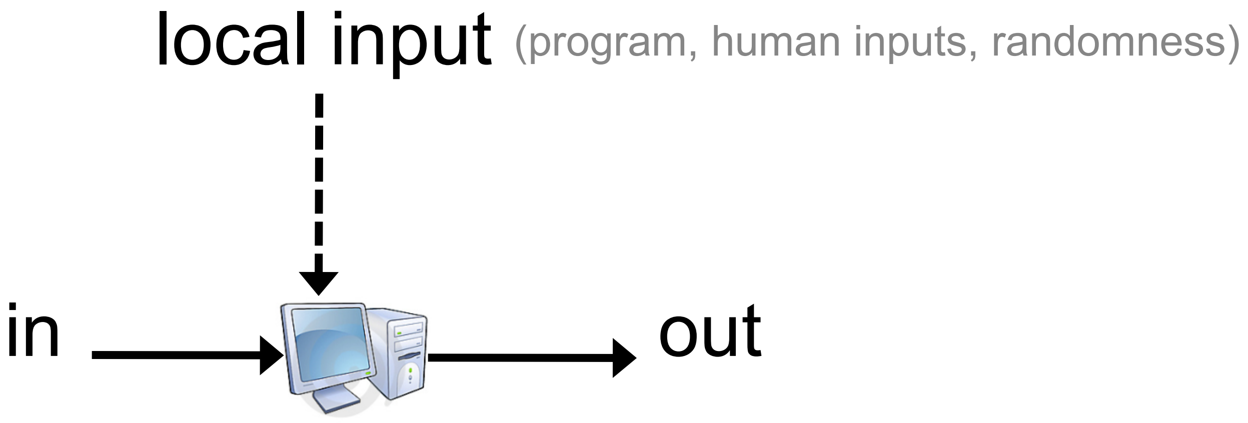
- Diverse network, containing untrustworthy parties and unreliable components.

Integrity via Proof-Carrying Data



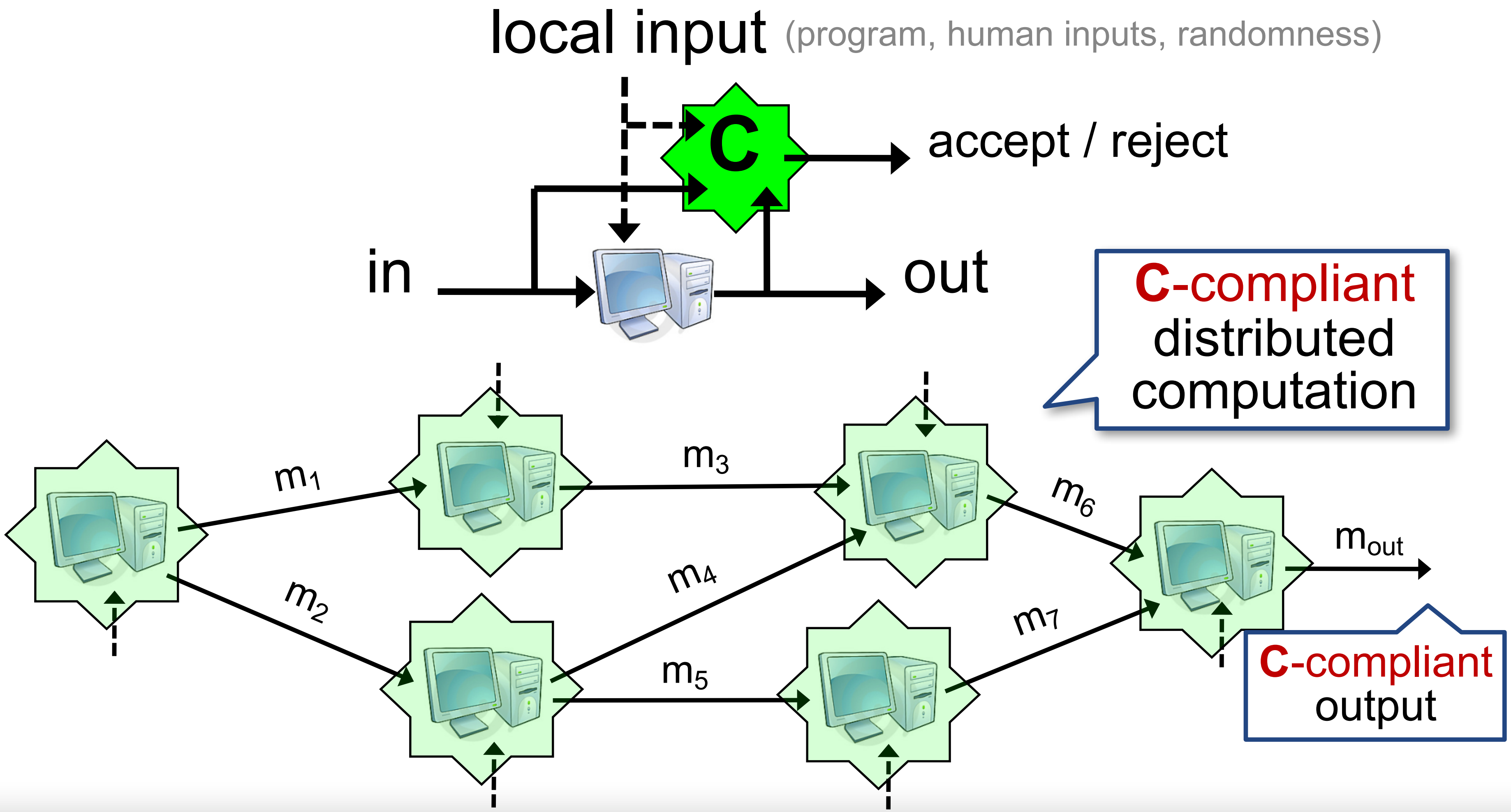
- Every message is augmented with a proof attesting to its compliance with a prescribed policy.
- Compliance can express any property that can be verified by locally checking every node.
- Proofs can be verified efficiently and retroactively.

C-compliance

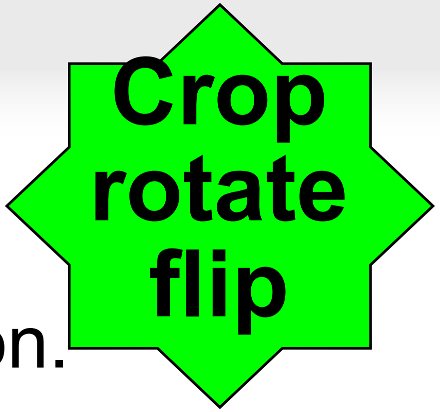


System administrator specifies his notion of **correctness** via a **compliance predicate** **C**(incoming, local inputs, outgoing) that must be locally fulfilled at every node.

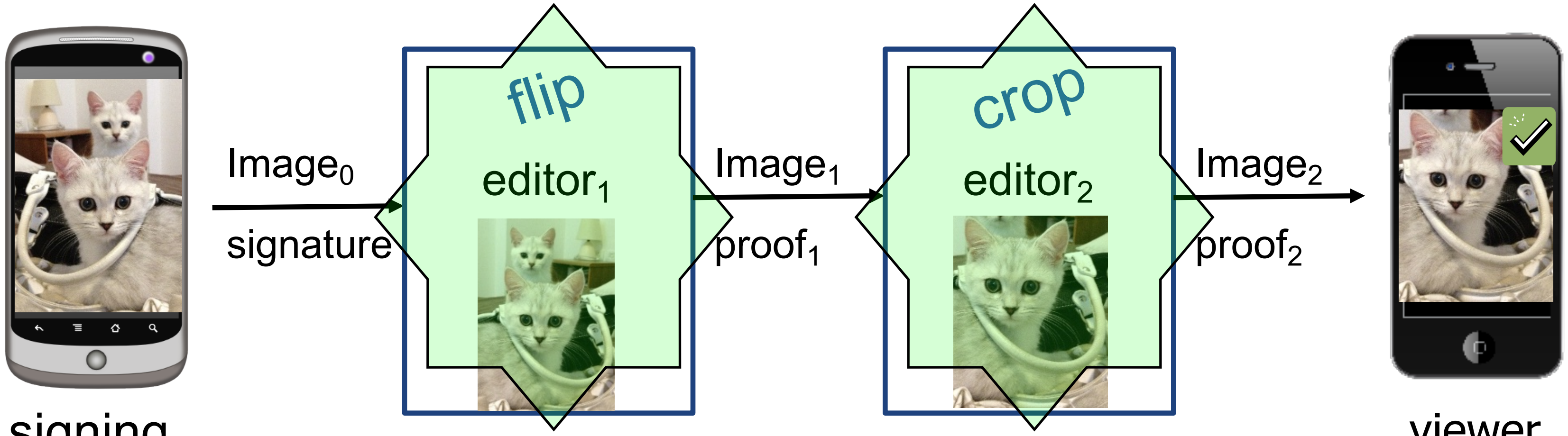
C-compliance



PhotoProof: IA Scheme from PCD

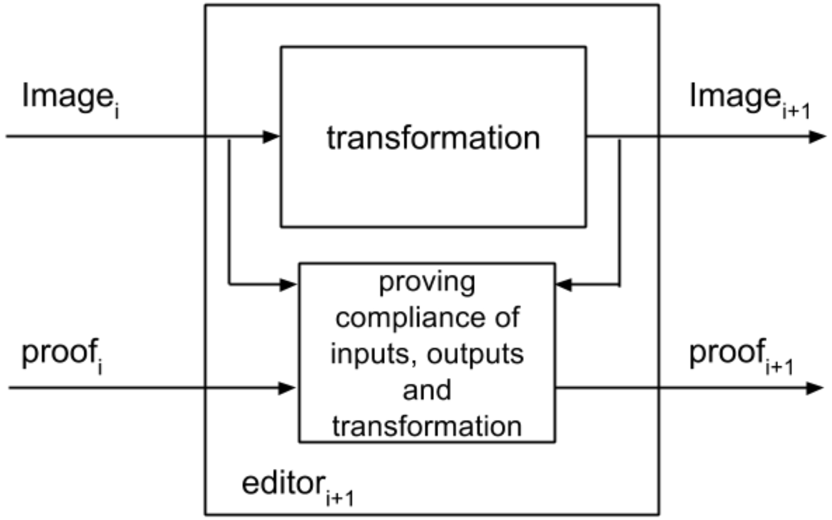


- The main idea:
  - Think of an image "life cycle" as a distributed computation.
  - Express local permissible edit as a compliance predicate.
  - Use PCD to enforce compliance of the image's history from signed to current version.



PhotoProof:
IA Scheme from PCD

- The main idea:
  - Think of an image “life cycle” as a distributed computation.
  - Express local permissible edit as a compliance predicate.
  - Use PCD to enforce compliance of the image’s history from signed to current version.



➢ An authentic image is a signed image which went through permissible edits exclusively.

PhotoProof prototype

- PCD implementation:
  `libsnark` library by SCIPR Lab
  based on [Valiant 08] [Chiesa Tromer 2010] [Parno Gentry Howell Raykova 2013] [Bitansky Canetti Chiesa Tromer 2013] [Ben-Sasson Chiesa Tromer Virza 2014] …
- Supports RGB images
- Prototype’s supported transformations:
  - identity
  - rectangular crop
  - horizontal and vertical flip
  - transpose
  - brightness/contrast adjustments
  - free rotation

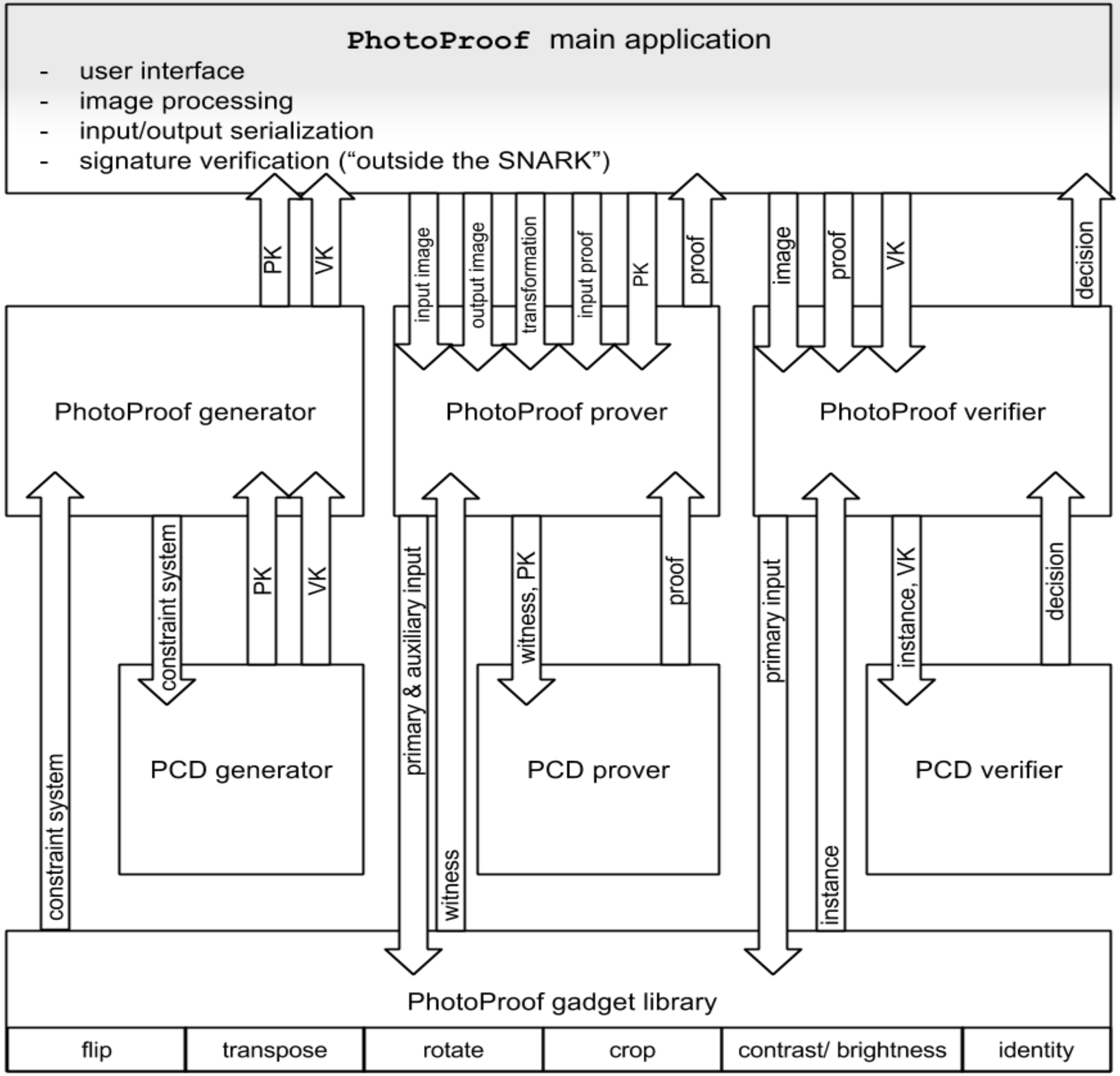
System architecture

1st level, Python

2nd level, C++

3rd level, C++ libsnark's PCD

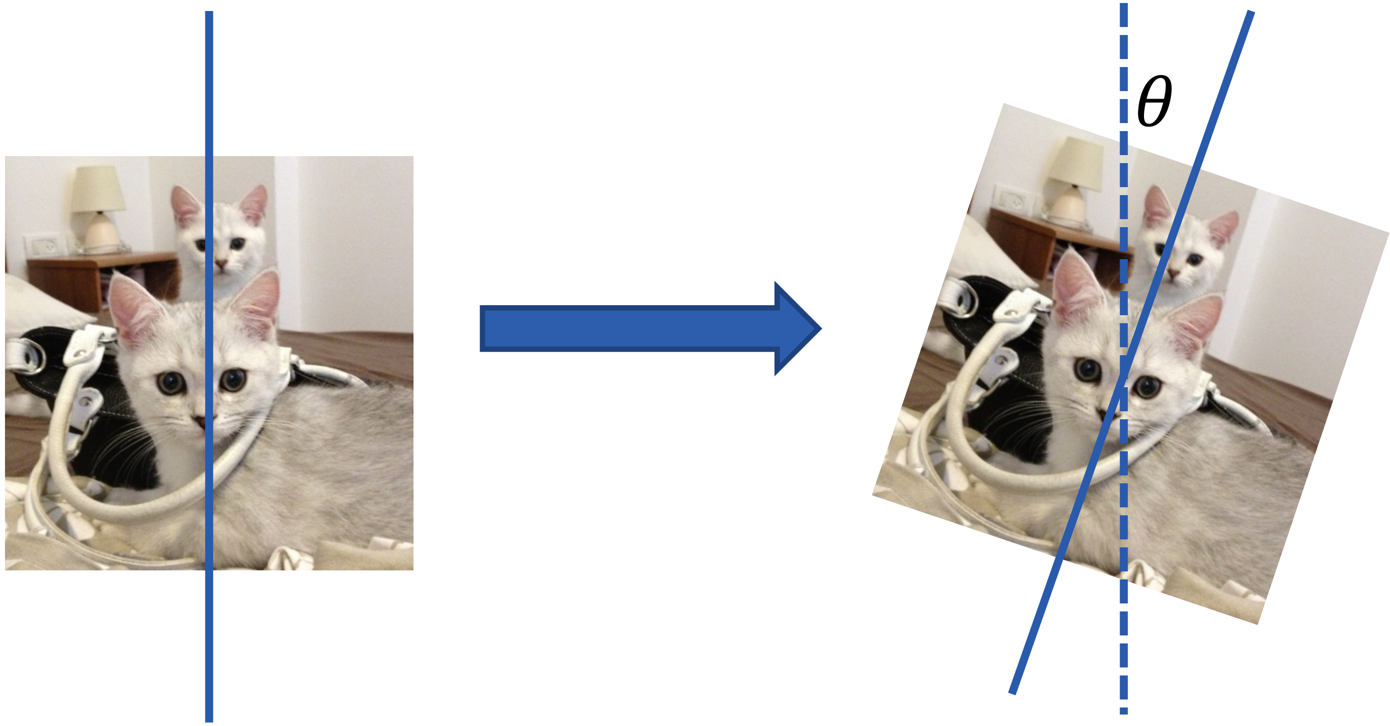
4th level C++ code Gadget construction



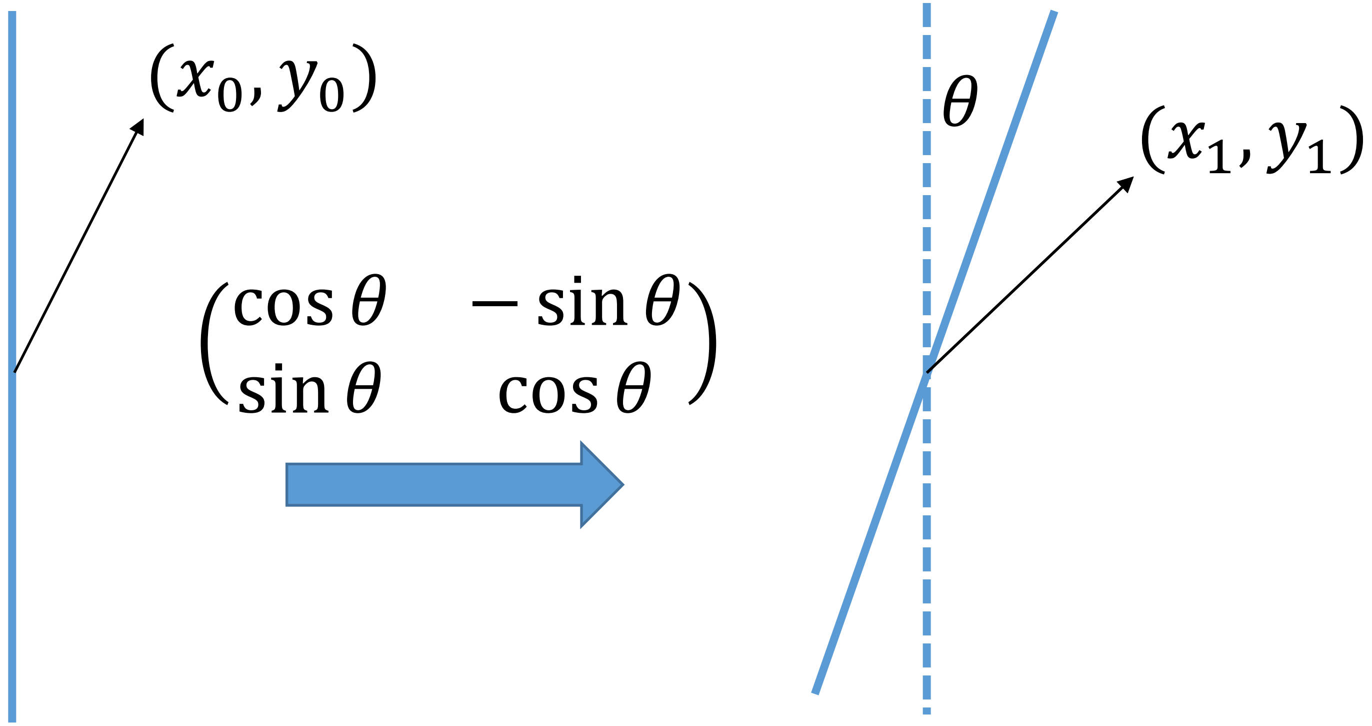
Challenges

- Formalization and analysis
  - Raises deep problems in definition of PCD
- For our compliance predicate, we need to express image transformations as arithmetic circuits.
- Example: rotate transform.

Case Study: Rotate

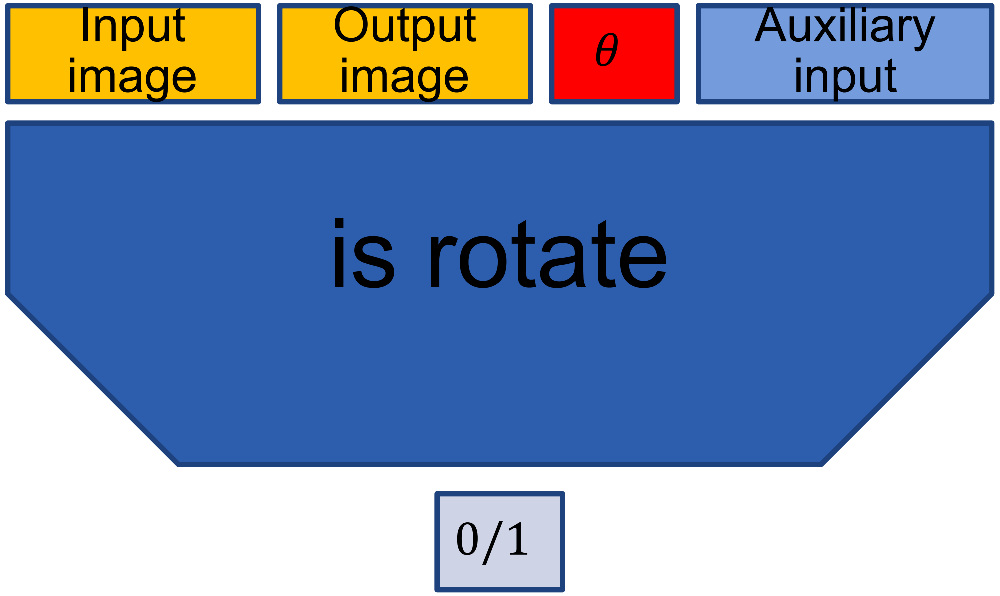


Case Study: Rotate



Case Study: Rotate

- We need to design an arithmetic circuit that checks rotation:

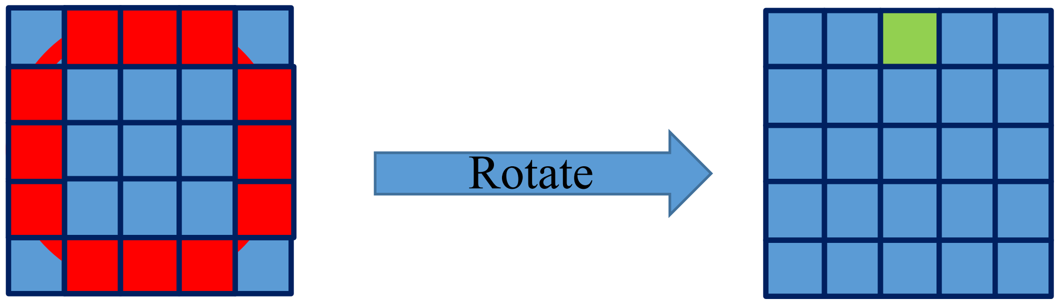


- The size of the circuit affects the PCD running time.

Case Study: Rotate

Images are assumed to be of fixed (maximal) size - $N \times N$

Naively, each output pixel depends on $O(N)$ input pixels, which sums to $O(N^3)$ circuit size.



Case Study: Rotate
Circuit design

To perform the rotation, we use *rotation by shearing* [Paeth 1986]:

The rotation matrix can be decomposed to 3 shearing operations

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$$

where $a = -\tan\frac{\theta}{2}$ and $b = \sin\theta$.

Shears can be efficiently performed with barrel-shifters with $O(NlogN)$ gates.

Case Study: Rotate
Nondeterminism

Our circuit can make use of any nondeterministic **advice** that can help it reach a decision.

We can use this to avoid calculating the trigonometric functions.

Instead we can give it a and b and only check they are consistent with some angle.

To do these, the prover provides $c = \sin\frac{\theta}{2}$ and $d = \cos\frac{\theta}{2}$ as **advice**, and the compliance predicate checks that:

$$c^2 + d^2 = 1$$
$$da = c$$
$$2cd = b$$

Challenges and dilemmas in formalizing

- How to combine the signature scheme in the IA definition?
  - require a signature scheme or, more generally, an "originality decider"?
- Who edits the image – inside P or in a separate algorithm?
- Who generates the secret signing key?
  - in reality cameras will already have signing keys.
- How to "convert" digital signatures to PCD proofs?
  - we do this in a user-transparent way.
- What is the correct proof-of-knowledge notion and how to prove it?

Performance

N	#Π	Generator (s)	Prover (s)	Verifier (s)	pk (MB)	vk (MB)	Proof size (kB)
16	171,815	16.9	15.9	0.09	144.4	2.7	2.67
32	706,959	32.9	30.7	0.1	255.5		
64	2,966,167	83.2	91.1	0.14	635.1		
128	12,531,999	367	423	0.5	2601.4		

PhotoProof prototype running times and key sizes for $N \times N$ images.
Average and (normalized) standard deviation (σ) are over 10 iterations each. #Π is the size of the generated compliance predicate.
Tests were ran on 4-core 3.4GHz Intel i7-4770 desktop with 32GB RAM.

Additional/future capabilities

- PhotoProof plugin for GIMP/PhotoShop
- Protect metadata to authenticate geo-tags, face tags, copyright messages etc.
- Include (and protect) the image edit-history in the metadata to avoid creating unwanted changes from many small permissible edits.
- Losslessly embed proofs in images.
- Use certificate chains to allow for multiple signing keys and revocation.

PhotoProof: discussion

- Longstanding open problem solved (modulo performance)
- Demonstrates the power of Proof-Carrying Data in tracking and enforcing authenticity for digital media

Open/problems:

- Performance
- Attacks at the sensor/scene level
- Analogously enforcing provenance in:
  - text (e.g., tracking citations)
  - audio (e.g., proving authenticity of a recording)
  - databases (e.g., tracking use of sensitive or unreliable information)