VeriTAS

Verifying Image Provenance at Scale

Trisha Datta **Binyi Chen** Dan Boneh

Stanford University

Image Provenance Verification

Verify Image Metadata

- Who
- When
- Where

Image Provenance Verification

Verify Image Metadata

- Who
- When
- Where

Why Important?

- Fight disinformation
- Copyright protection
- Regulation compliance

These look like prizewinning photos. They're AI fakes.

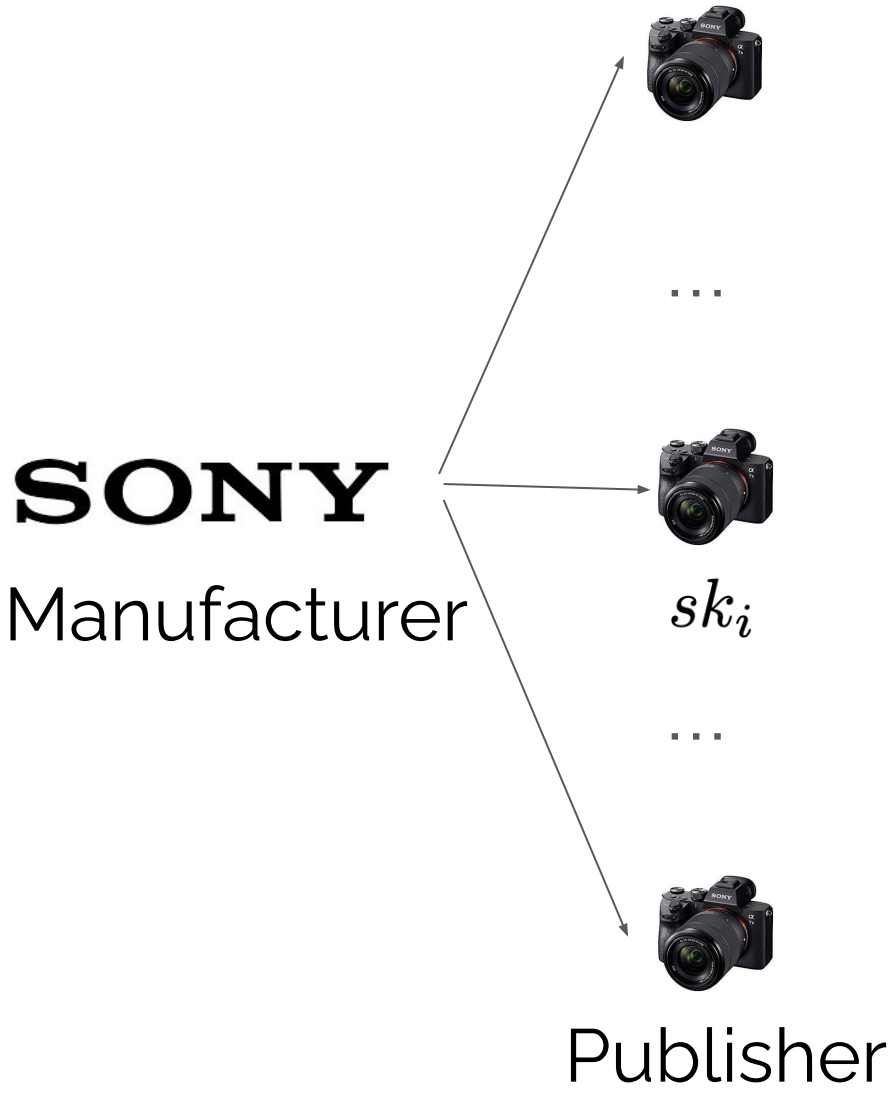
Artificially generated images of real-world news events proliferate on stock image sites, blurring truth and fiction

November 23, 2023

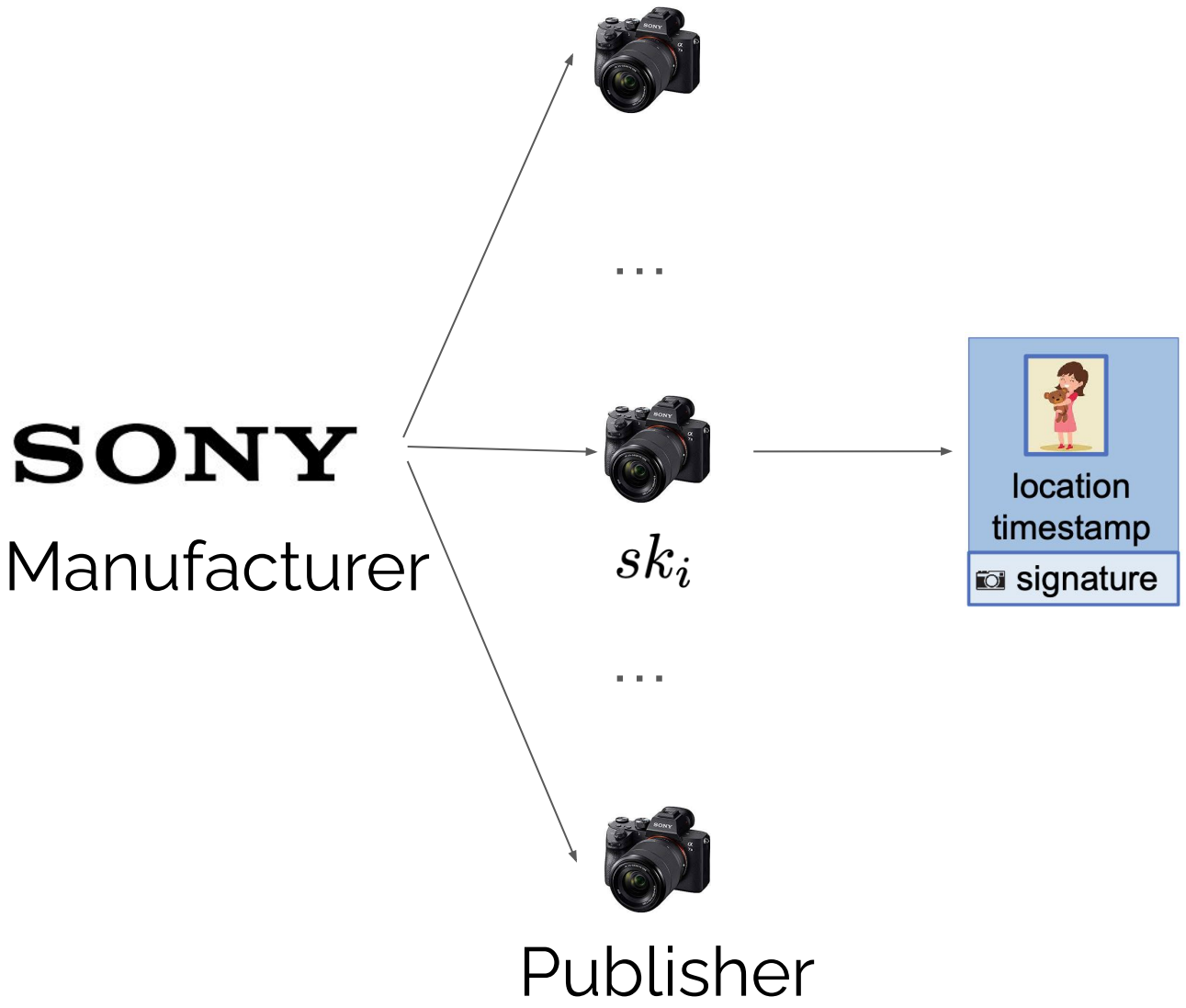
By Will Oremus and Pranshu Verma

C2PA Standard (Simplified)

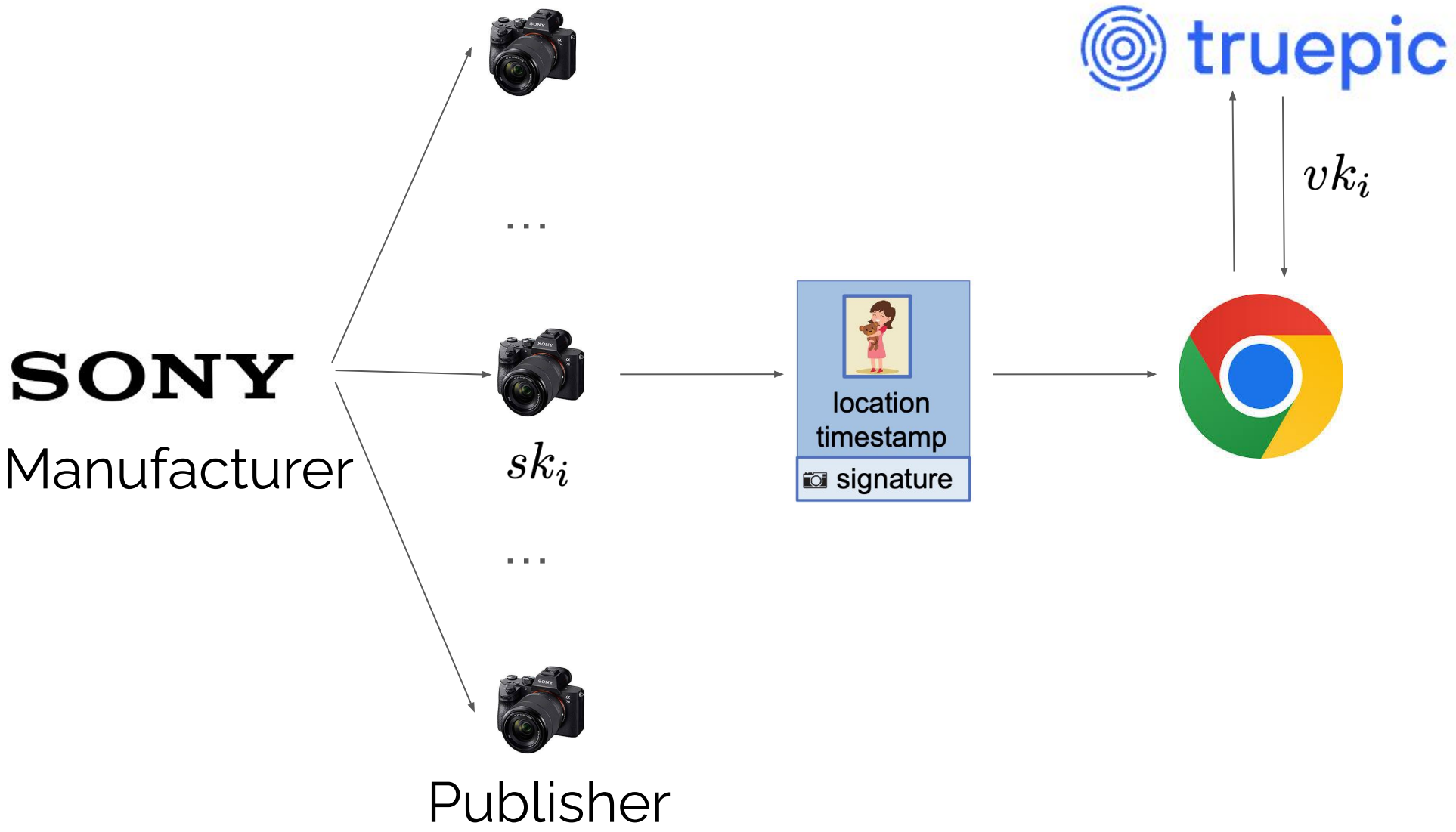
C2PA Standard (Simplified)



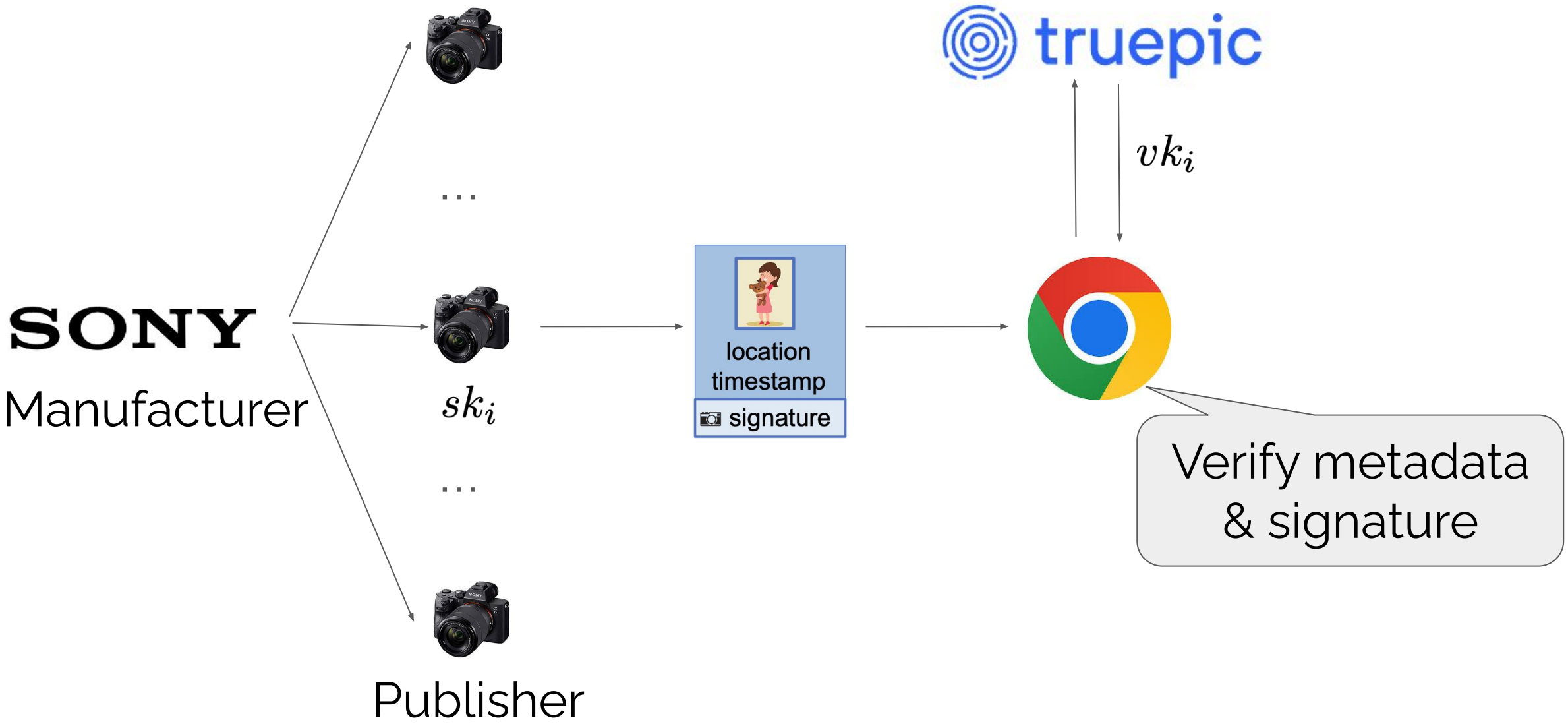
C2PA Standard (Simplified)



C2PA Standard (Simplified)

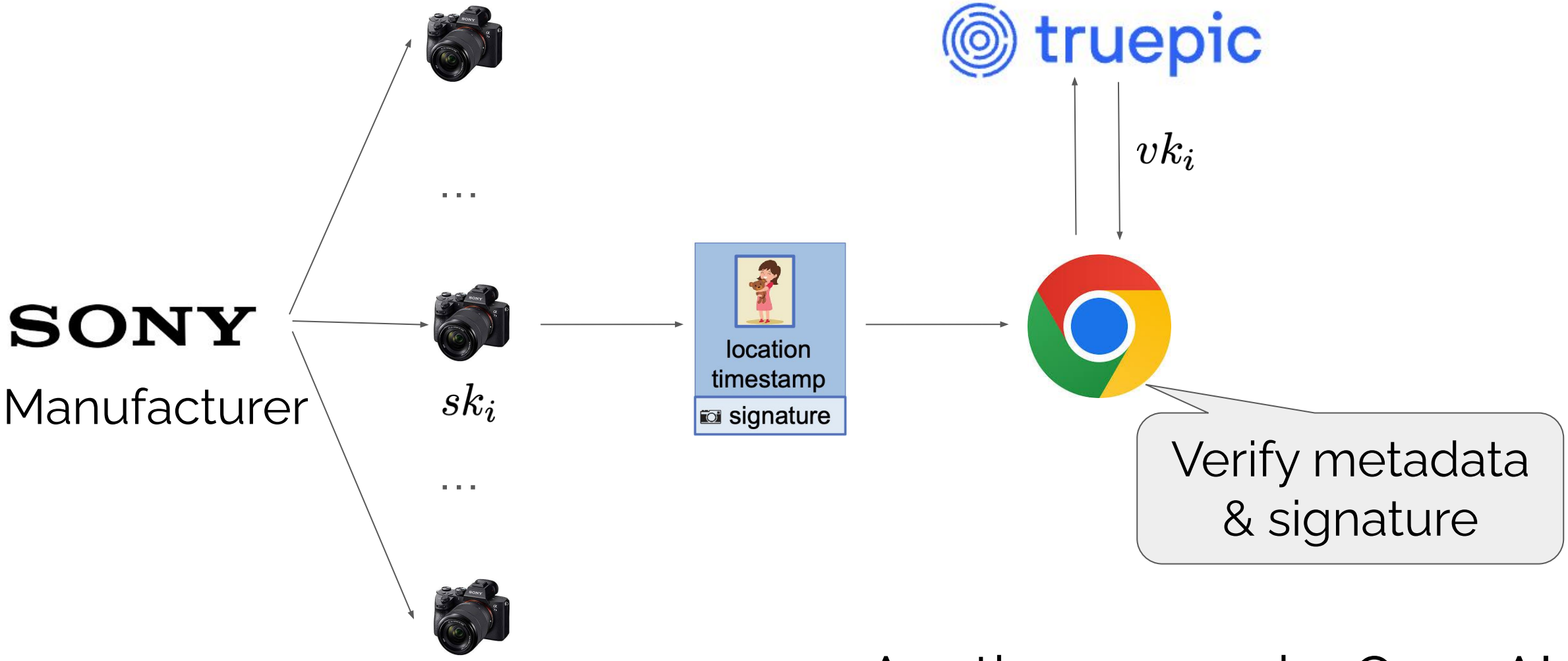


C2PA Standard (Simplified)



C2PA Standard

Leica camera has built-in defense against misleading AI, costs $9,125



Another example: OpenAI

[https://help.openai.com/en/articles/8912793-c2pa-in-dall-e-3]

Challenge: Verifying Edited Images

Issue: Publishers process photos before publication

(e.g. cropping, blurring, etc.)

Publisher has no signature for it

Challenge: Verifying Edited Images

Issue: Publishers process photos before publication
(e.g. cropping, blurring, etc.)

C2PA's solution:

Publisher

C2PA-approved
Editing App

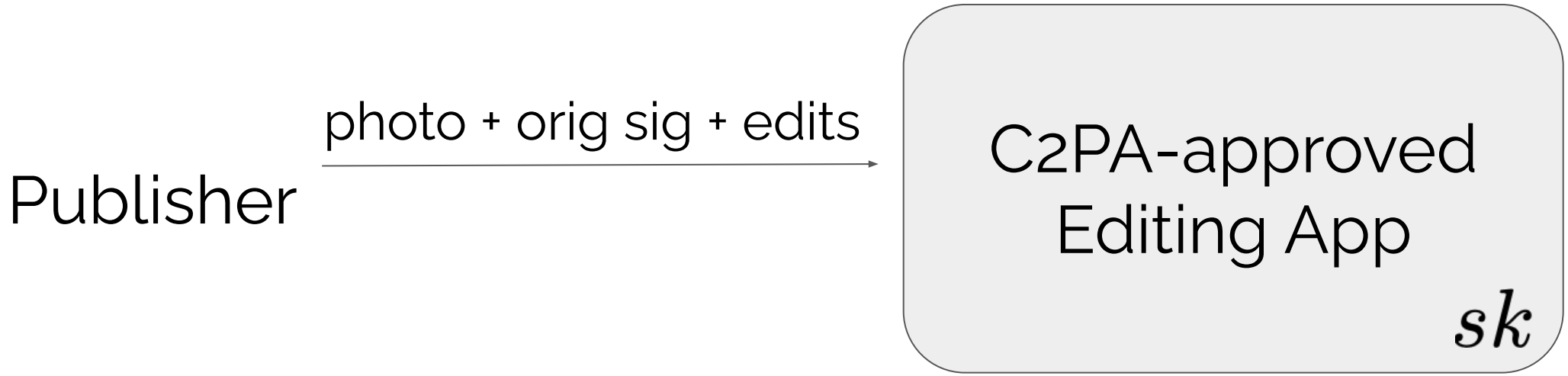
sk

Challenge: Verifying Edited Images

Issue: Publishers process photos before publication

(e.g. cropping, blurring, etc.)

C2PA's solution:

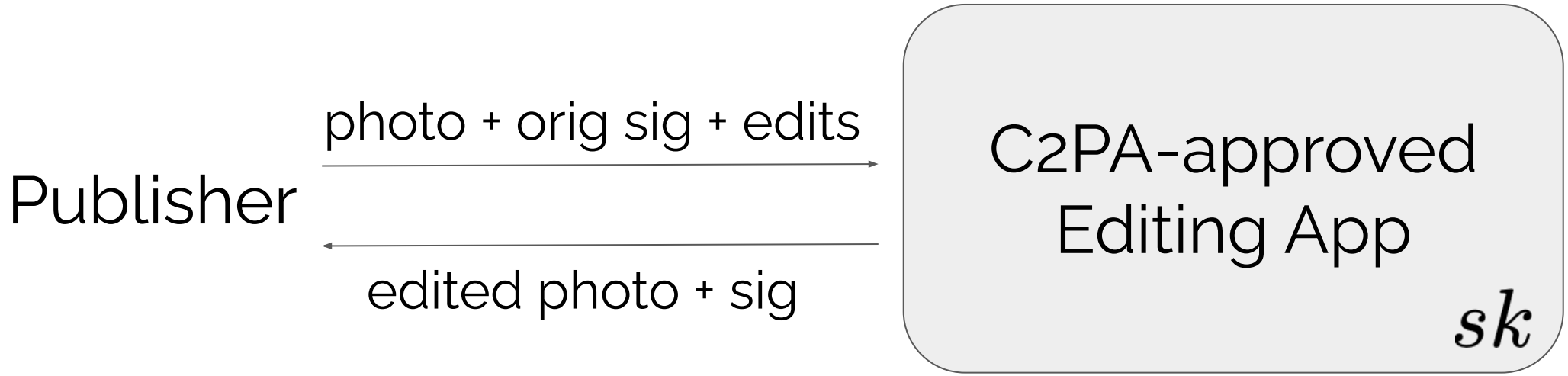


Challenge: Verifying Edited Images

Issue: Publishers process photos before publication

(e.g. cropping, blurring, etc.)

C2PA's solution:

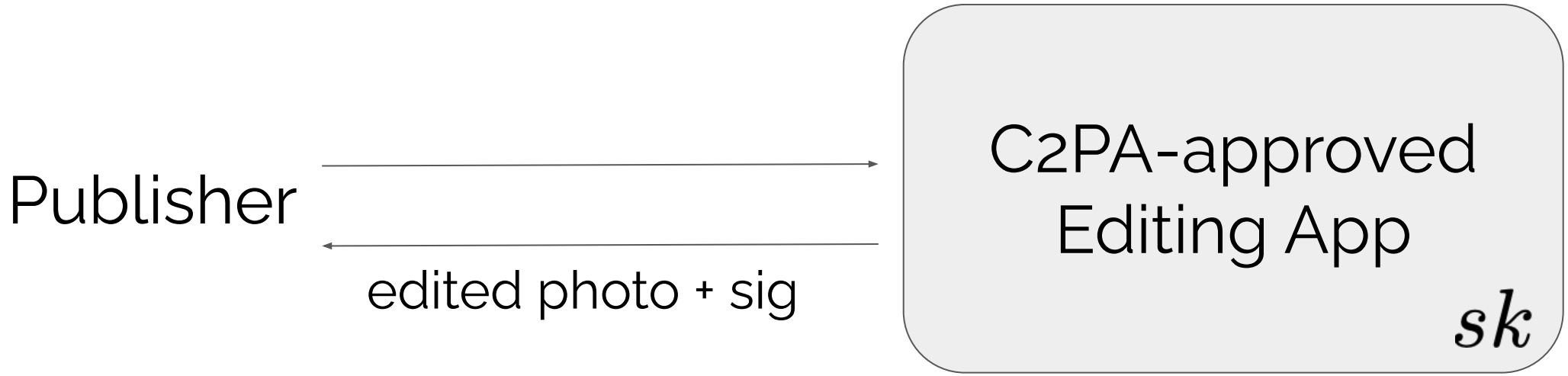


Challenge: Verifying Edited Images

Issue: Publishers process photos before publication

(e.g. cropping, blurring, etc.)

C2PA's solution:



Issue: Must trust the editing application

Our Goals

- Original photos properly signed

Our Goals

- Original photos properly signed
- Only permissible edits were made

Our Goals

- Original photos properly signed
- Only permissible edits were made
- Same metadata as the original

Our Goals

- Original photos properly signed
- Only permissible edits were made
- Same metadata as the original

Glass-to-glass security:

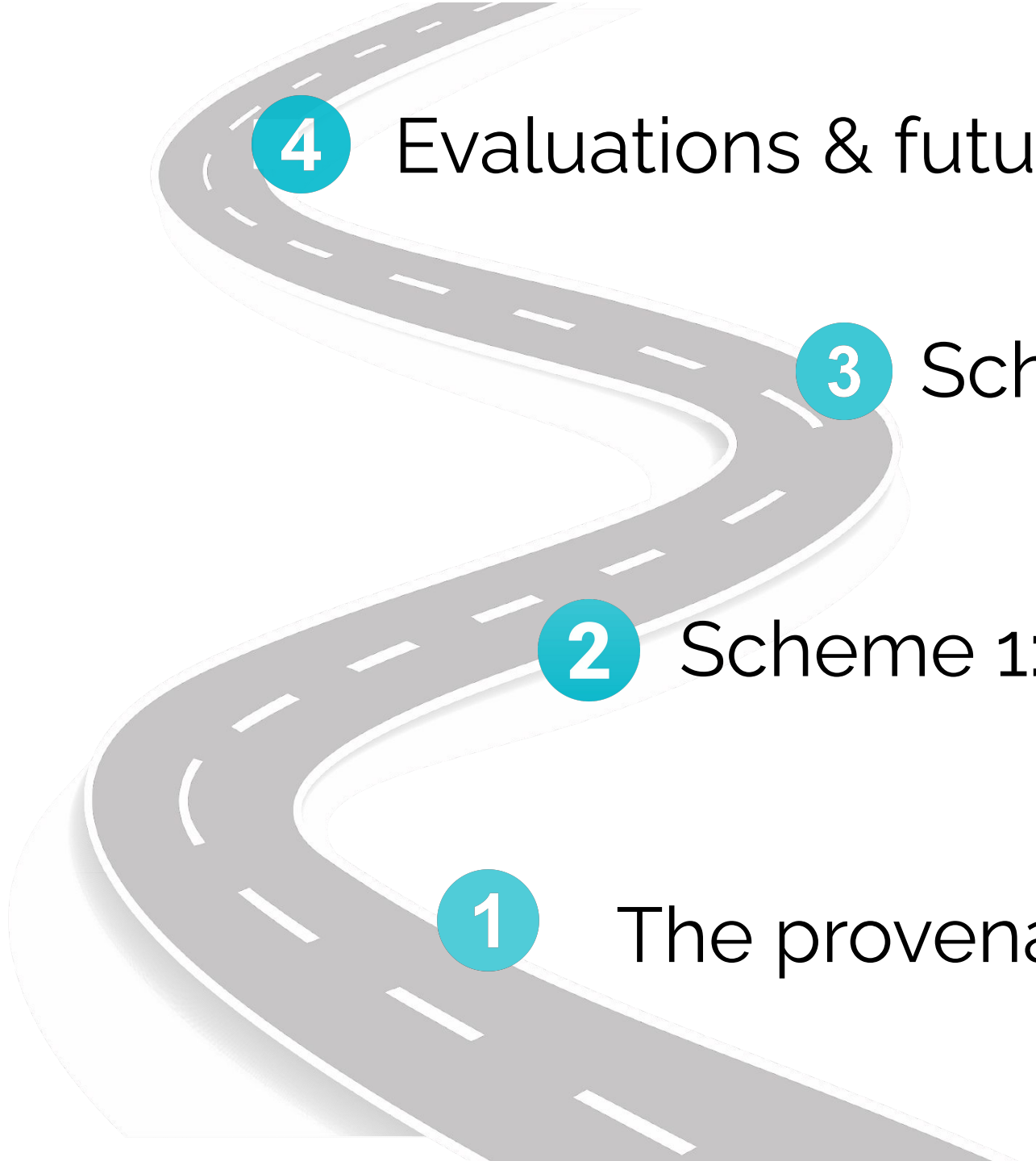
- No intermediate trust from camera to user screen

Our Results

- A zkSNARK-based image provenance system
- Verifying provenance of edited images
- No intermediate trust

Our Results

- A zkSNARK-based image provenance system
- Verifying provenance of edited images
- No intermediate trust
- The 1st system that supports >90MB images



4 Evaluations & future directions

3 Scheme 2: Faster proving

2 Scheme 1: Lightweight signing

1 The provenance framework

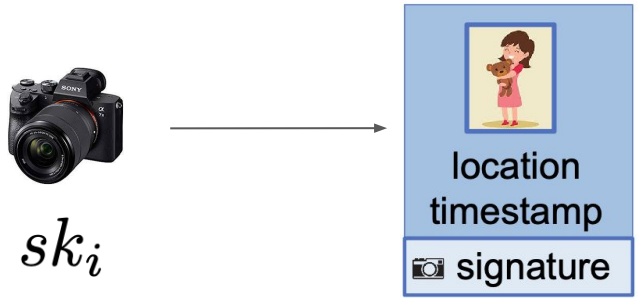
4 Evaluations & future directions

3 Scheme 2: Faster proving

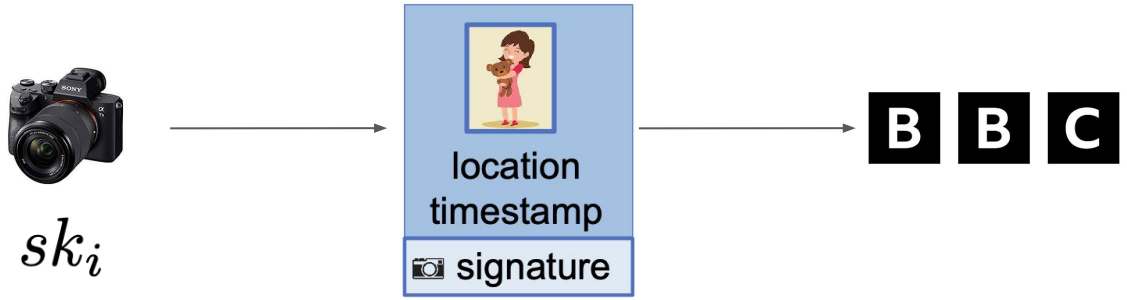
2 Scheme 1: Lightweight signing

1 The provenance framework

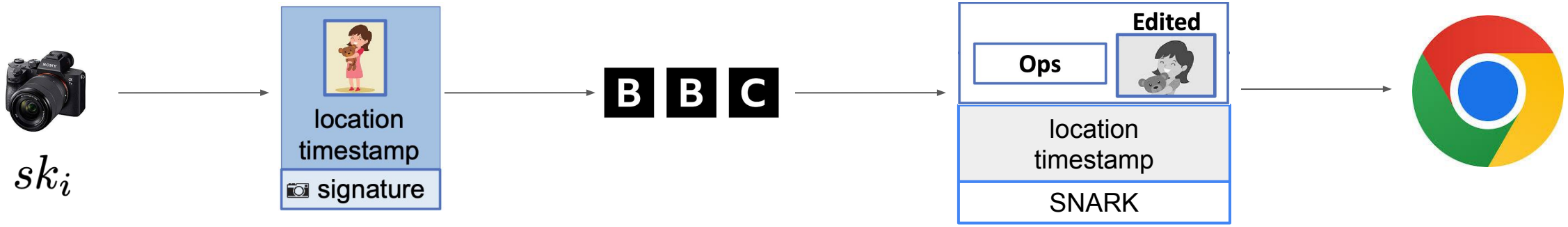
The Provenance Framework



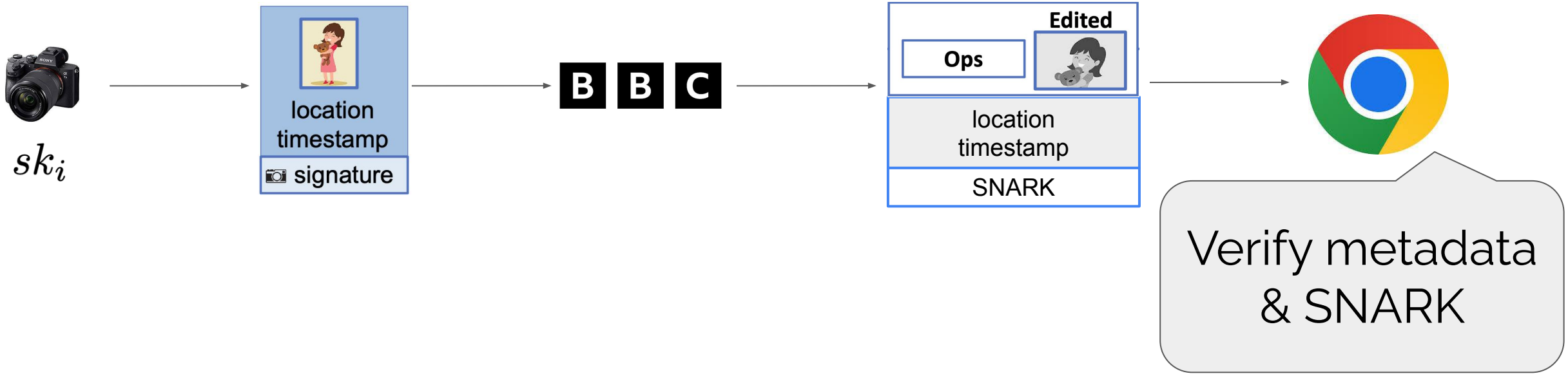
The Provenance Framework



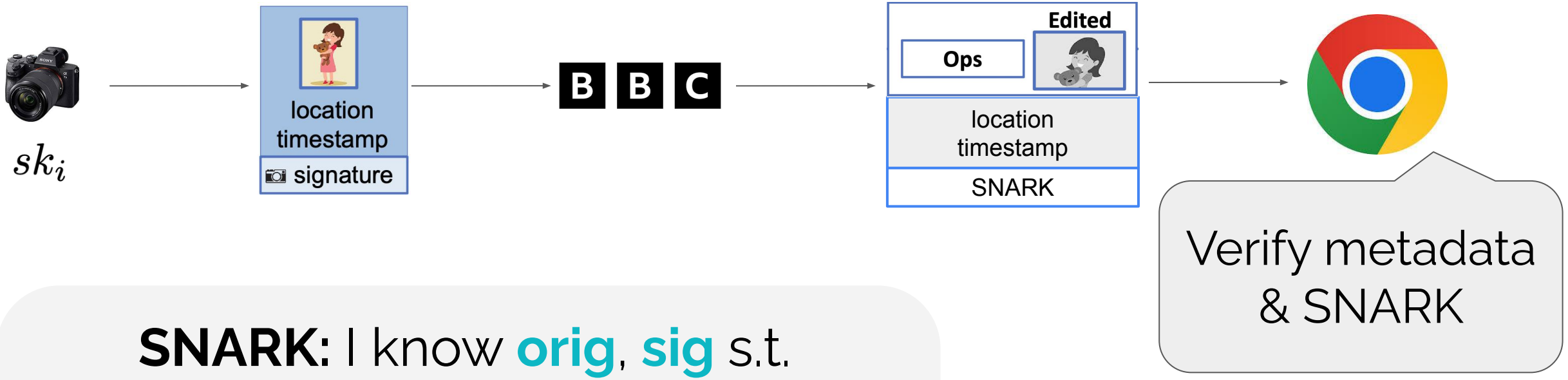
The Provenance Framework



The Provenance Framework



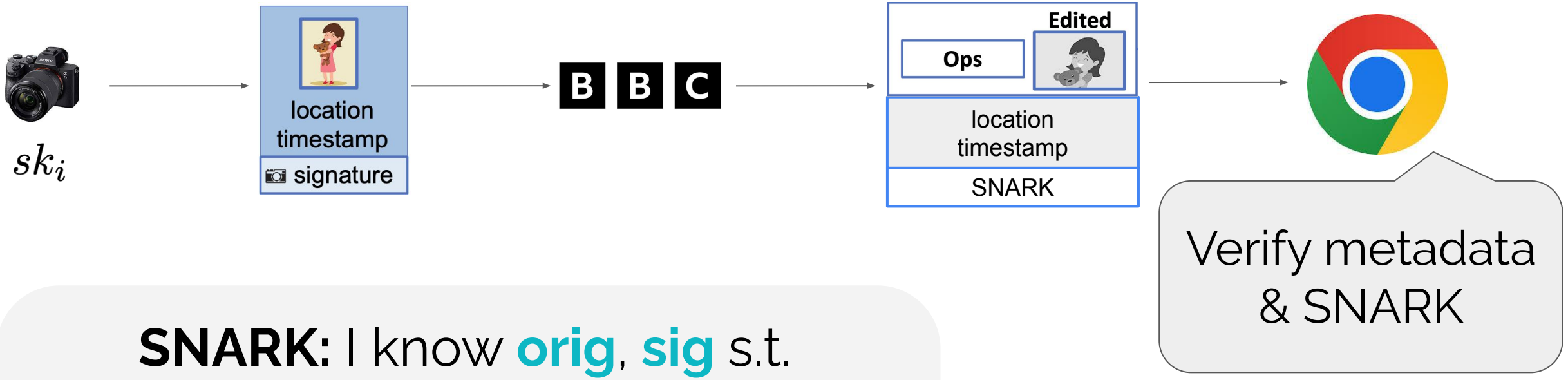
The Provenance Framework



SNARK: I know **orig**, **sig** s.t.

- **sig** is valid on hash(**orig**) w.r.t. **vk**
- **Edited** = **Ops**(**orig**)
- **Edited**.metadata = **orig**.metadata

The Provenance Framework



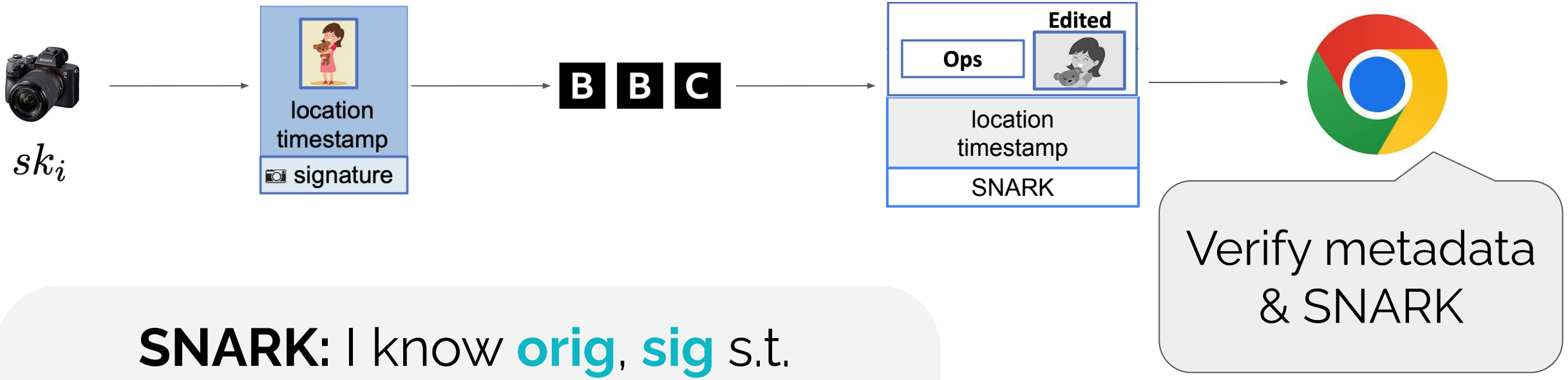
SNARK: I know **orig**, **sig** s.t.

- **sig** is valid on hash(**orig**) w.r.t. **vk**
- **Edited** = **Ops**(**orig**)
- **Edited**.metadata = **orig**.metadata

Hash proving is hard

orig > 90MB

The Provenance Framework



SNARK: I know **orig**, **sig** s.t.

- **sig** is valid on hash(**orig**) w.r.t. **vk**
- **Edited** = **Ops**(**orig**)
- **Edited**.metadata = **orig**.metadata

Hash proving is hard

orig > 90MB

PhotoProof (2016):
a few mins for 128 x 128 pixel image

4 Evaluations & future directions

3 Scheme 2: Faster proving

2 Scheme 1: Lightweight signing

1 The provenance framework

Hash Proving for Weak Signers

Scheme 1: Lightweight signing + ZK-friendly hash

	Speed	ZK-friendly	Output length
SHA-256			
Poseidon			
Ajtai			

Hash Proving for Weak Signers

Scheme 1: Lightweight signing + ZK-friendly hash

	Speed	ZK-friendly	Output length
SHA-256	✓	✗	✓
Poseidon	✗ / ✓	✓ ?	✓
Ajtai			

Hash Proving for Weak Signers

Scheme 1: Lightweight signing + ZK-friendly hash

	Speed	ZK-friendly	Output length
SHA-256	✓	✗	✓
Poseidon	✗ / ✓	✓?	✓
Ajtai	✓	✓	✗

Hash Proving for Weak Signers

Scheme 1: Lightweight signing + ZK-friendly hash

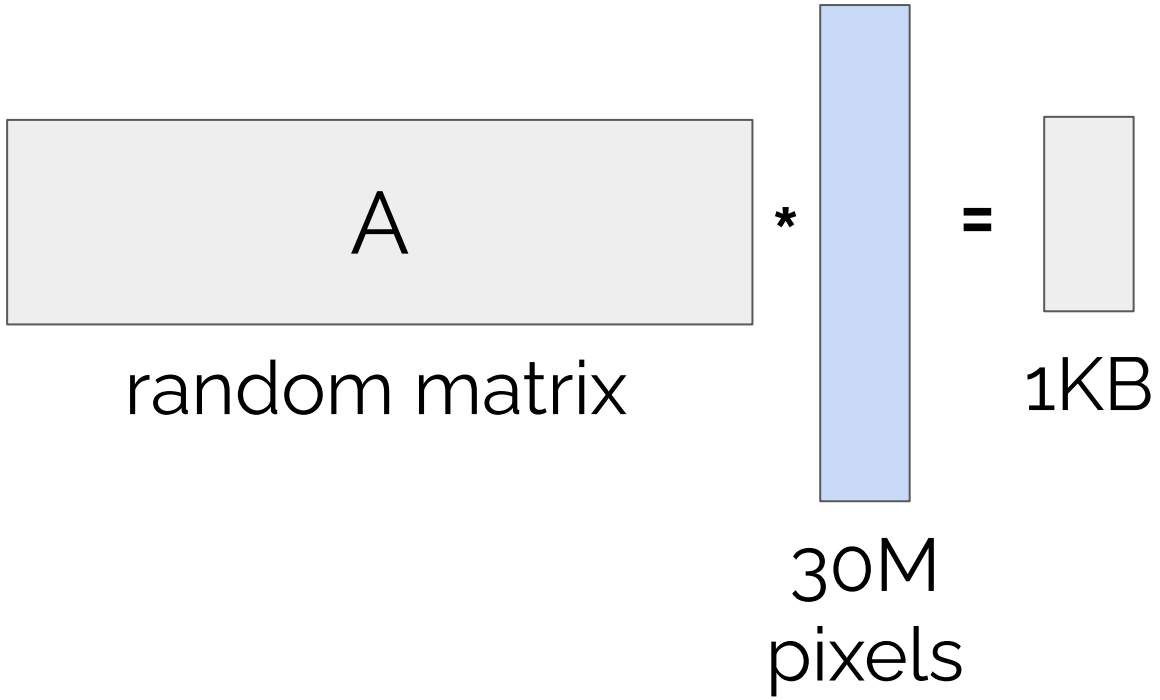
	Speed	ZK-friendly	Output length
SHA-256	✓	✗	✓
Poseidon	✗ / ✓	✓ ?	✓
Ajtai	✓	✓	✗

Our choice: Compose Ajtai with Poseidon

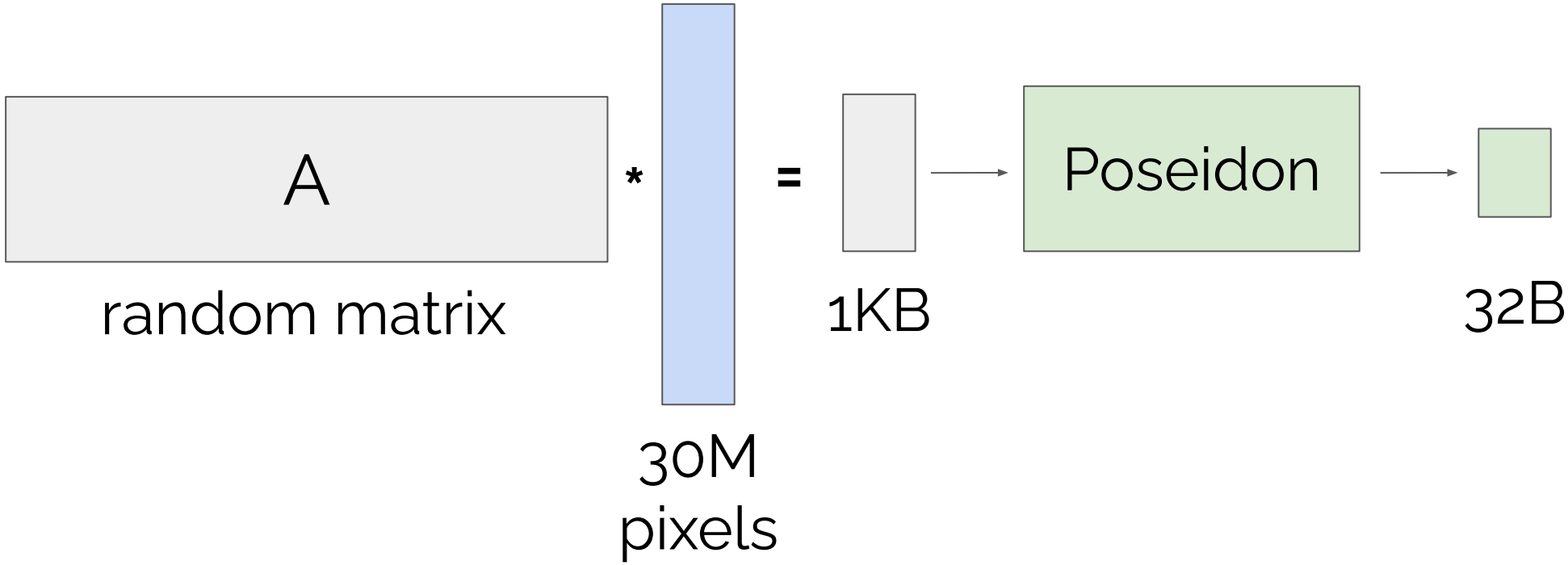
Hash Proving for Weak Signers

30M
pixels

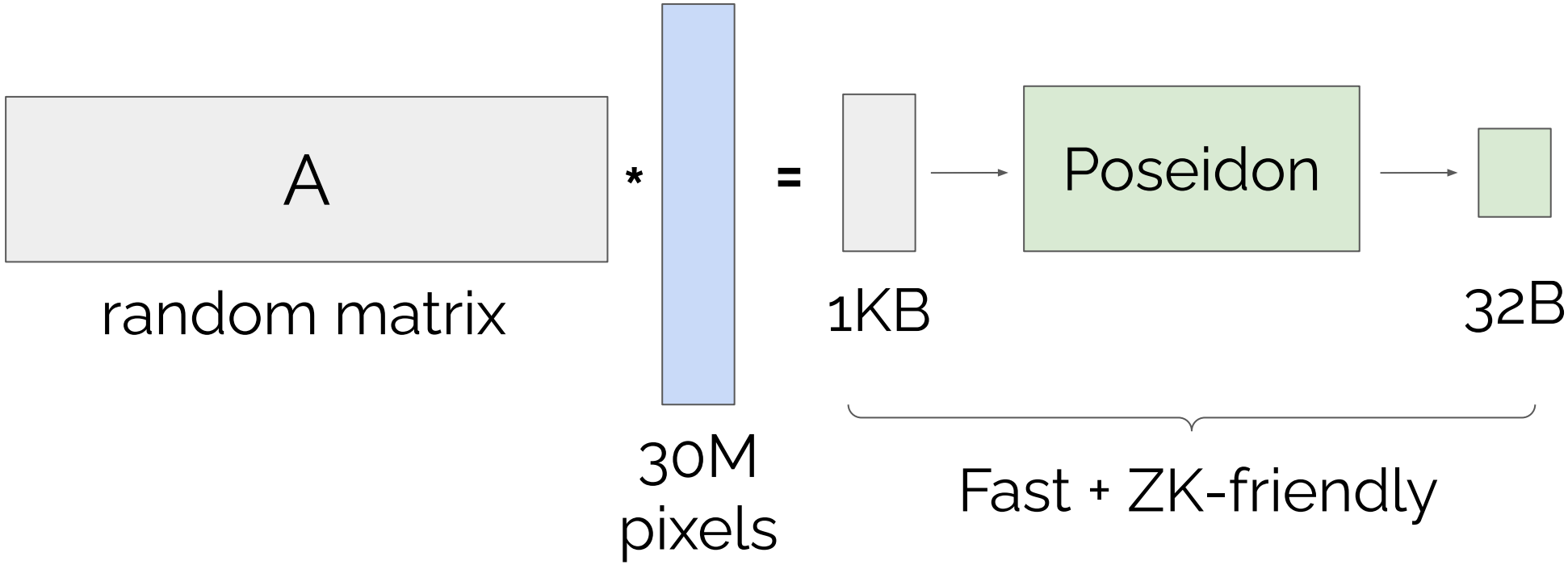
Hash Proving for Weak Signers



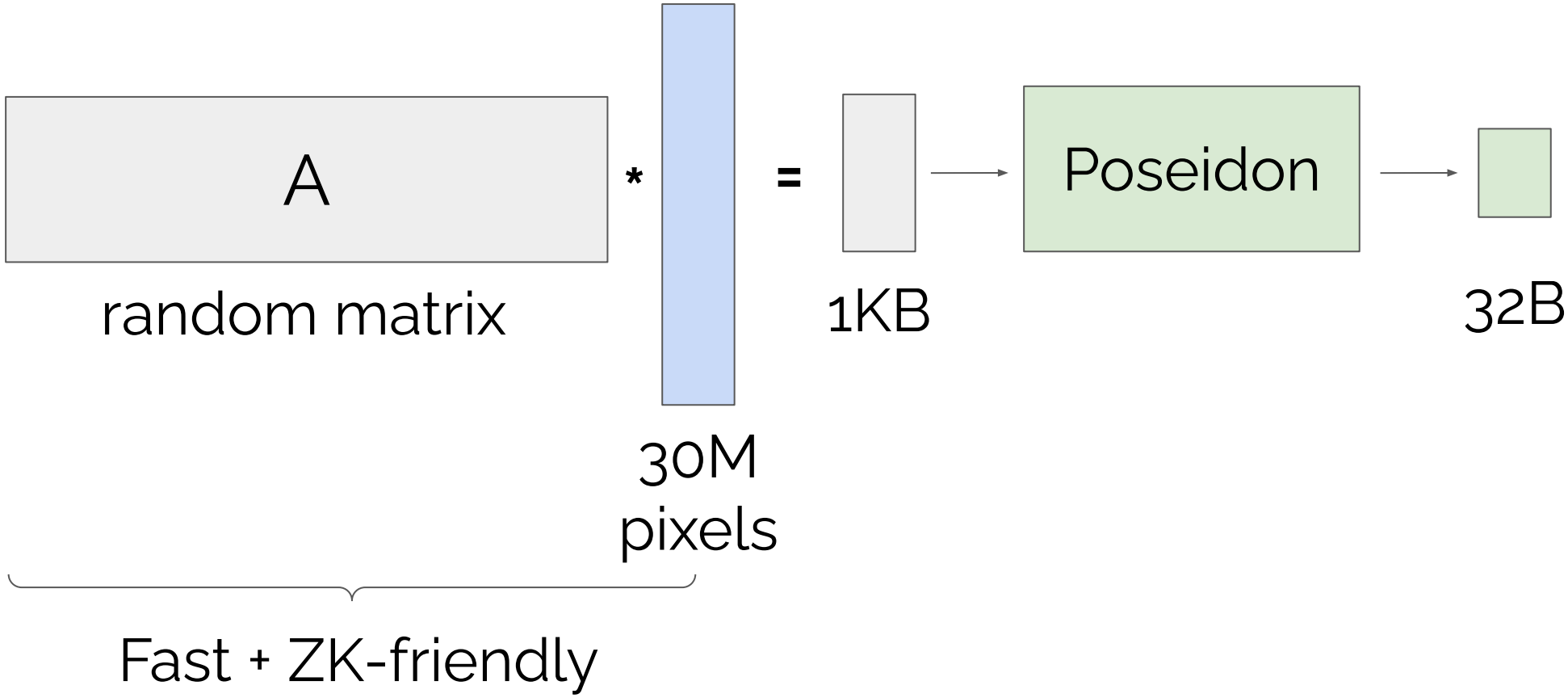
Hash Proving for Weak Signers



Hash Proving for Weak Signers

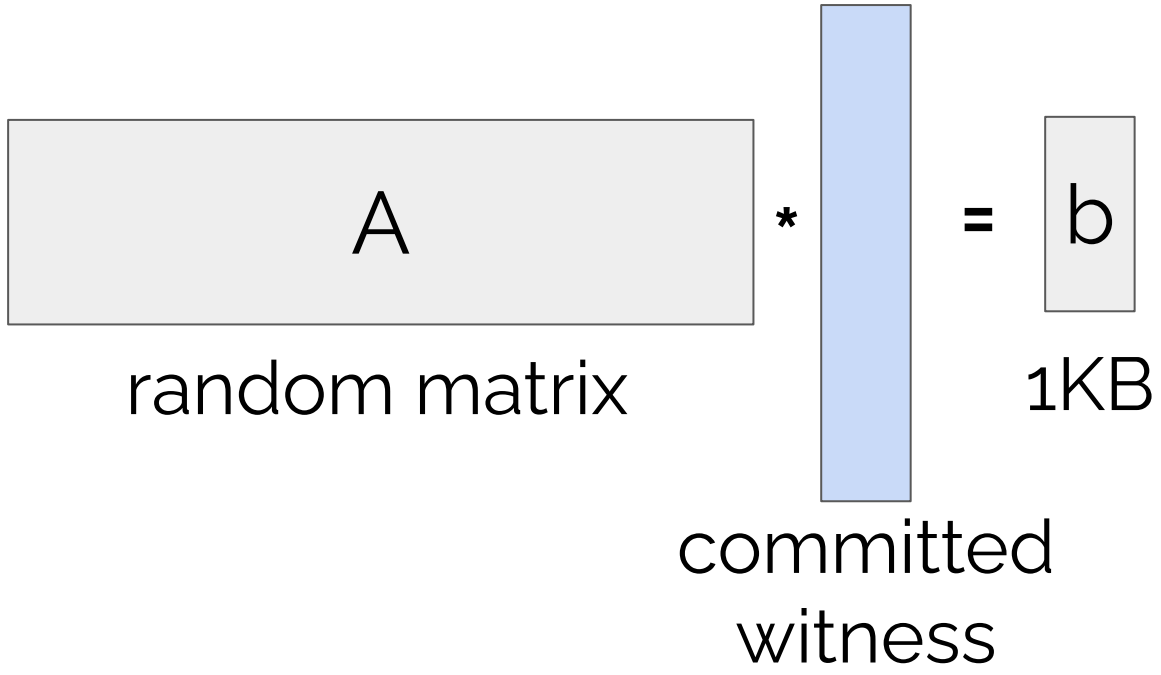


Hash Proving for Weak Signers



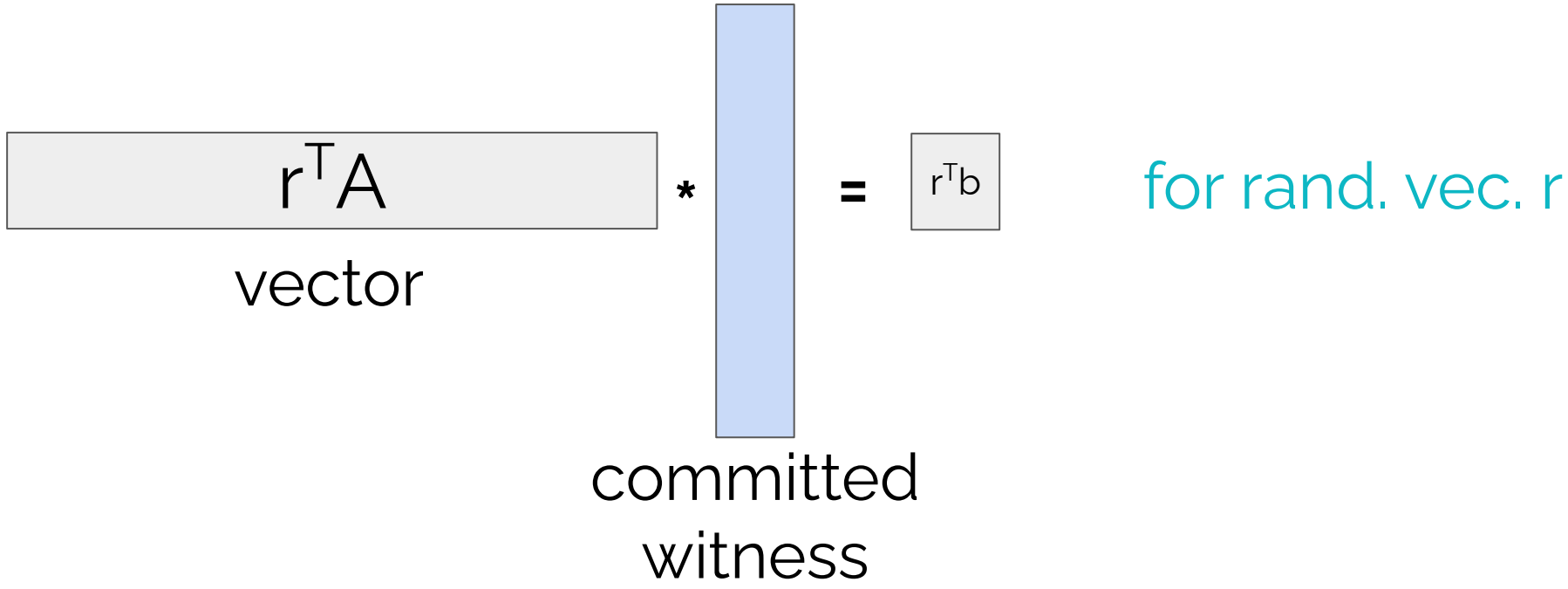
Hash Proving for Weak Signers

Freivald's algorithm: Prove



Hash Proving for Weak Signers

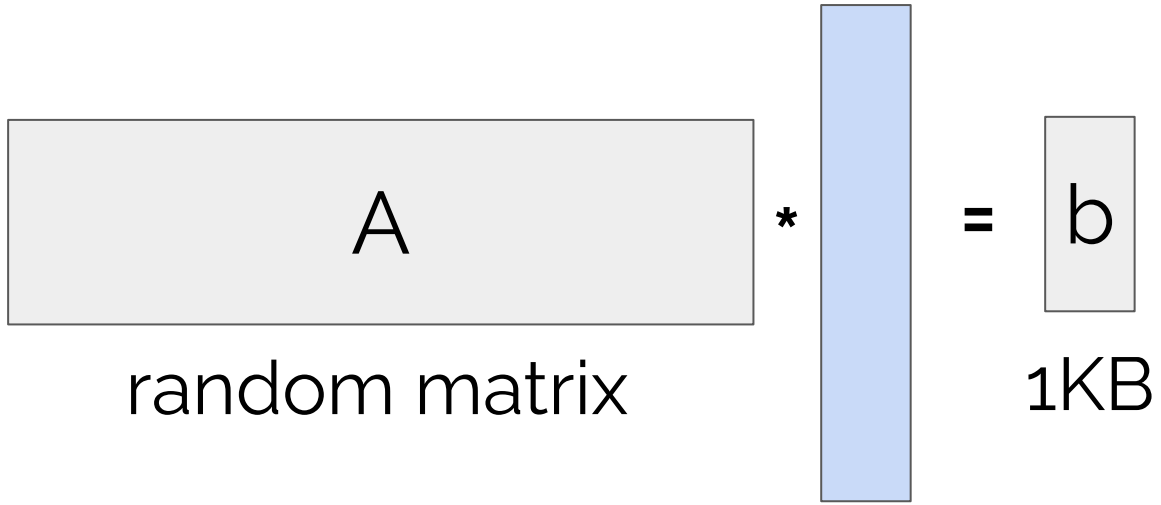
Freivald's algorithm: Sufficient to prove



for rand. vec. r

Hash Proving for Weak Signers

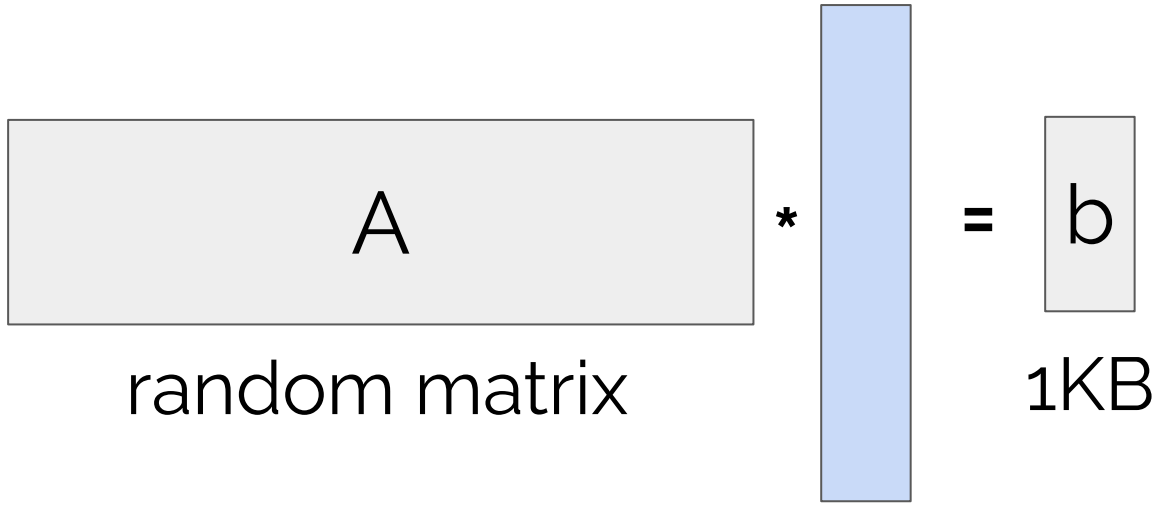
Ajtai binding commitment



Binding for low-norm input

Hash Proving for Weak Signers

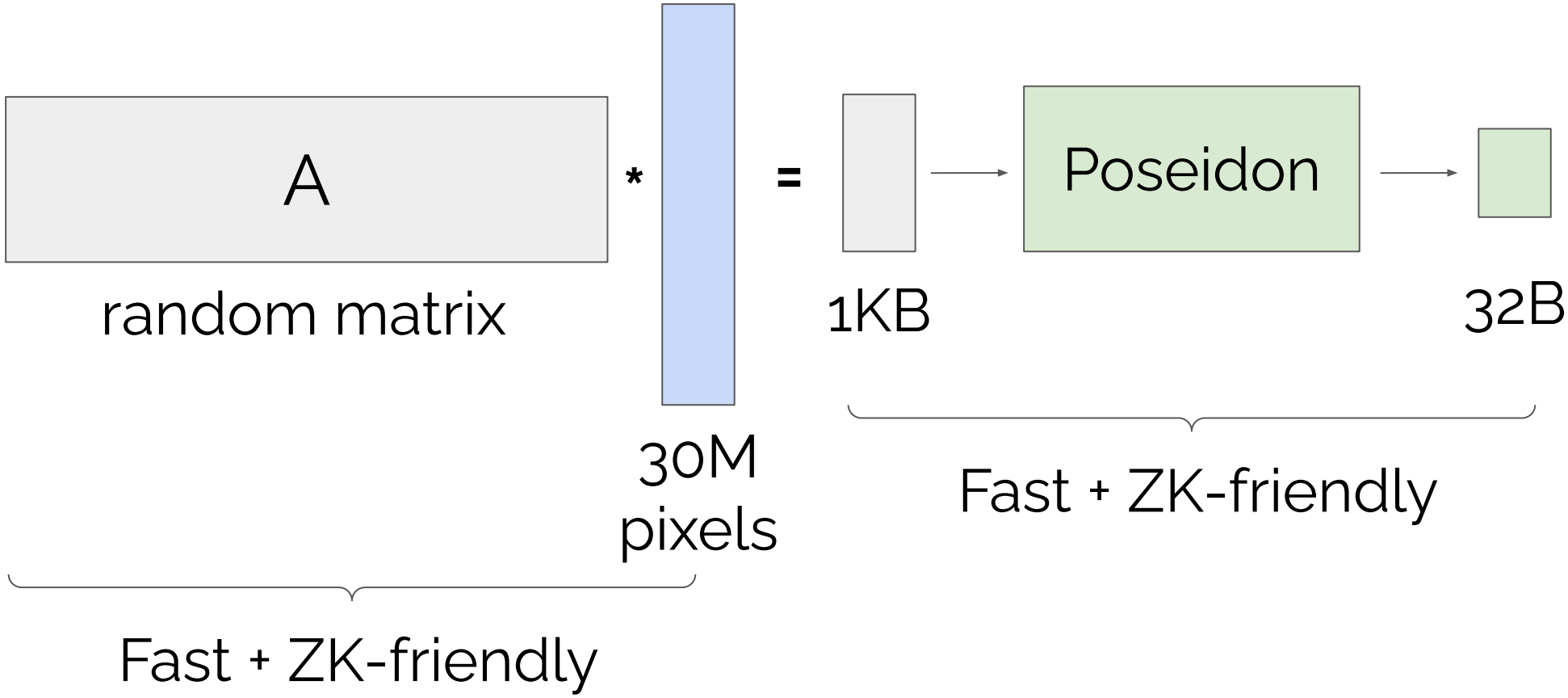
Ajtai binding commitment



Binding for low-norm input

Challenge: Range-check image pixels

Hash Proving for Weak Signers



Challenge: Range-check image pixels

Range-check Pixels [BCGJM'18, GWC'19, GW'20]

Prove $\vec{x} \in \{0, 1, \ldots, n-1\}^m$

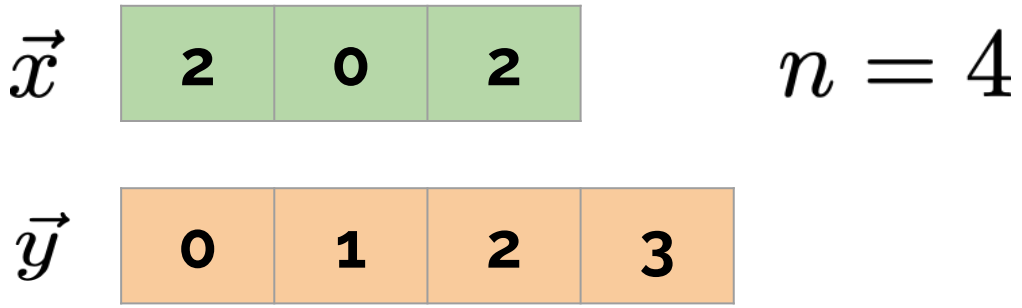
$\vec{x}$ | 2 | 0 | 2 |

$n = 4$

Range-check Pixels [BCGJM'18, GWC'19, GW'20]

Prove $\vec{x} \in \{0, 1, \ldots, n-1\}^m$

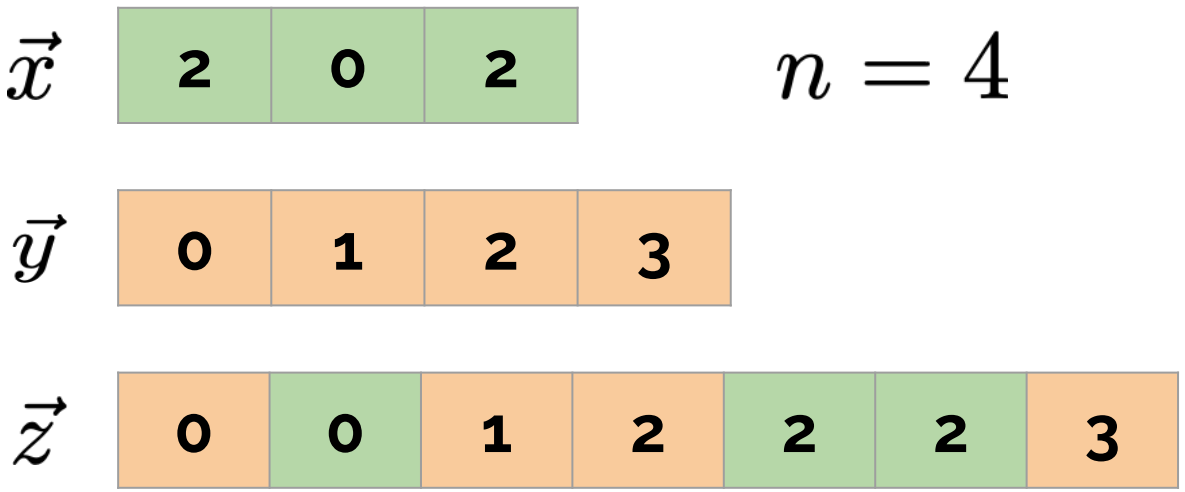
- $\vec{y} := [0, 1, \ldots, n-1]$



Range-check Pixels [BCGJM'18, GWC'19, GW'20]

Prove $\vec{x} \in \{0, 1, \ldots, n-1\}^m$

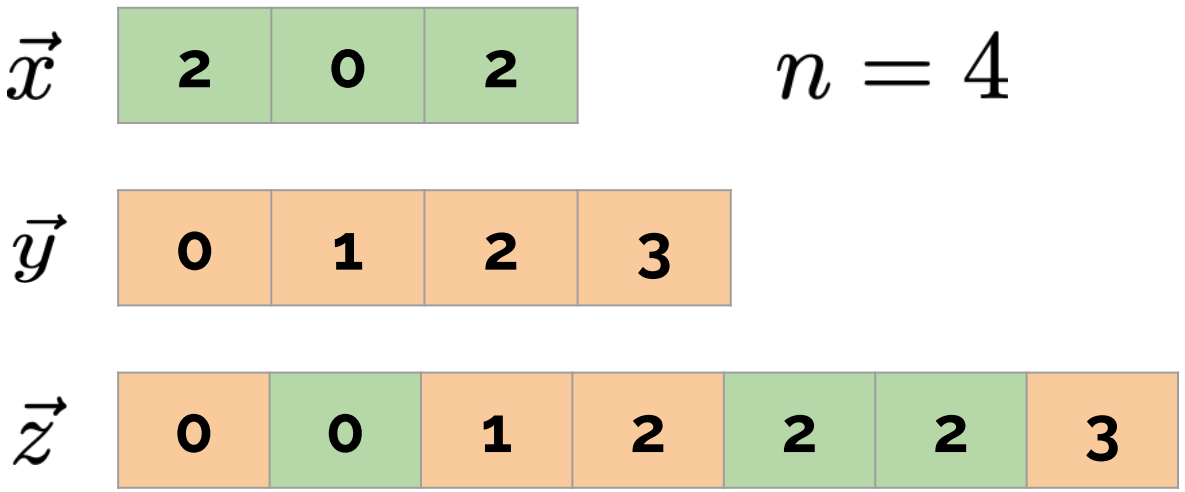
- $\vec{y} := [0, 1, \ldots, n-1]$
- $\vec{z} := \mathrm{sort}(\vec{x} || \vec{y})$



Range-check Pixels [BCGJM'18, GWC'19, GW'20]

Prove $\vec{x} \in \{0, 1, \ldots, n-1\}^m$

- $\vec{y} := [0, 1, \ldots, n-1]$
- $\vec{z} := \text{sort}(\vec{x}||\vec{y})$



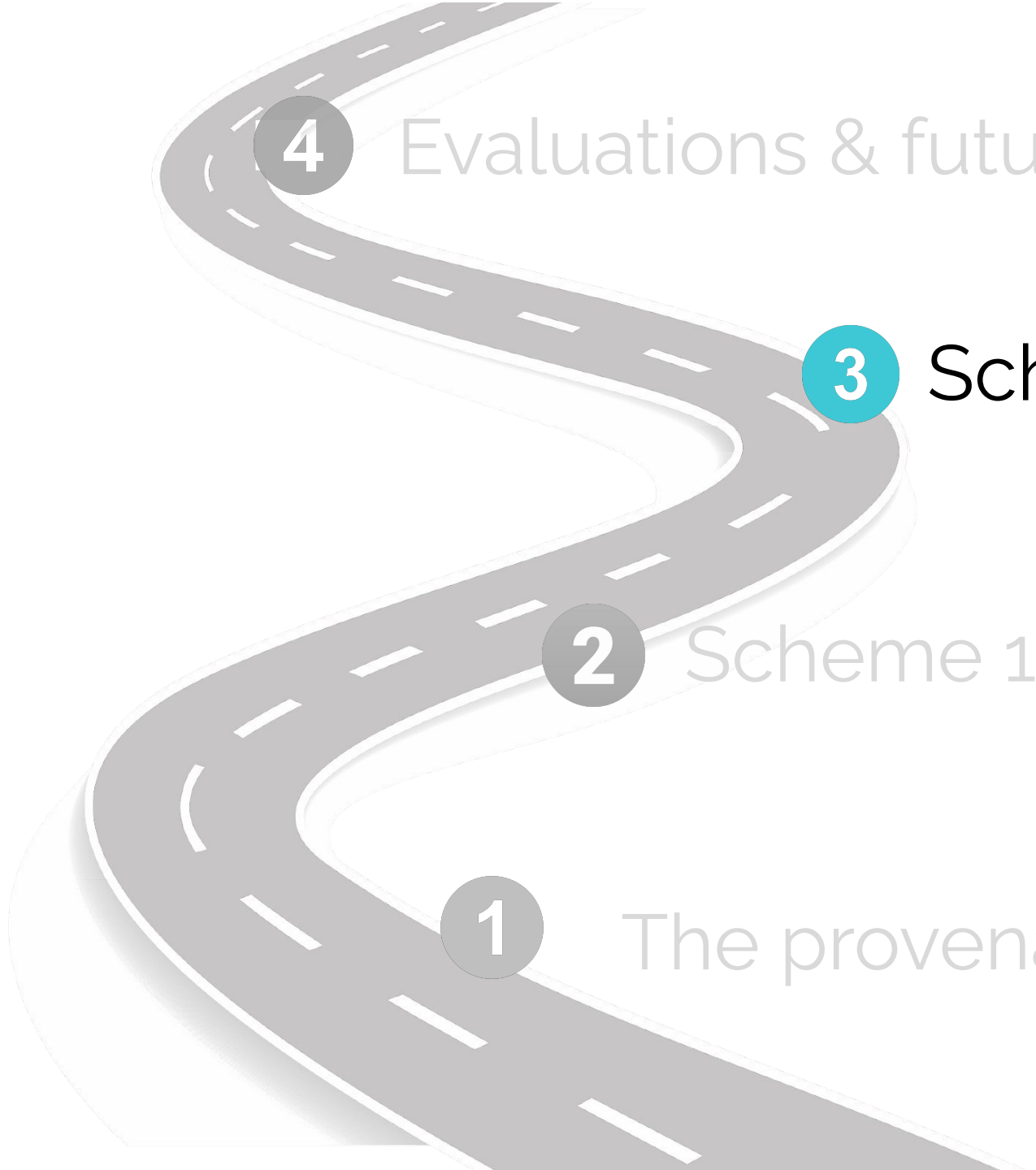
Check

- $\vec{z}$ is a permutation of $(\vec{x}||\vec{y})$
- $\vec{z}[i+1] - \vec{z}[i] \in \{0, 1\}$

Summary

- A lightweight & ZK-friendly signing scheme
- Fit for camera use cases
- Limitation: Hash proving is still the bottleneck

10 mins vs 1 min



4 Evaluations & future directions

3 Scheme 2: Faster proving

2 Scheme 1: Lightweight signing

1 The provenance framework

Removing Hash Proving

Scheme 2: Move hashing out from the SNARK circuit

(5~10x proving speedup)

Removing Hash Proving

Scheme 2: Move hashing out from the SNARK circuit

(5~10x proving speedup)

Building block: Polynomial commitment scheme (PCS)

Removing Hash Proving

Scheme 2: Move hashing out from the SNARK circuit

(5~10x proving speedup)

Building block: Polynomial commitment scheme (PCS)

- Commit(pp, f(X), r) ⟶ "short" C_f
- EvalOpen(pp, z, y; f, r) ⟶ proof π
  - f(z) = y
  - C_f= Commit(pp, f(X), r)

Removing Hash Proving

Scheme 2: Move hashing out from the SNARK circuit

(5~10x proving speedup)

Building block: Polynomial commitment scheme (PCS)

- Commit(pp, f(X), r) ⟶ "short" C_f
- EvalOpen(pp, z, y; f, r) ⟶ proof π
  - f(z) = y
  - C_f= Commit(pp, f(X), r)

👁 Serve as CRHF

Removing Hash Proving

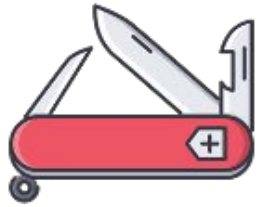
Scheme 2: Move hashing out from the SNARK circuit

(5~10x proving speedup)

Building block: Polynomial commitment scheme (PCS)

- Commit(pp, f(X), r) ⟶ "short" C_f
- EvalOpen(pp, z, y; f, r) ⟶ proof π
  - f(z) = y
  - C_f= Commit(pp, f(X), r)

- Serve as CRHF



- Commit & open witness

A Commit-and-Prove Scheme

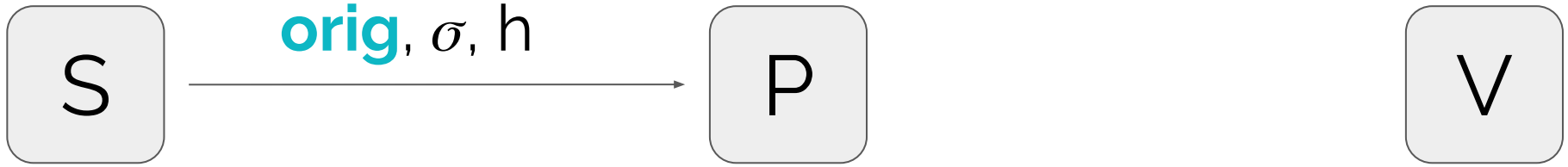
S

P

V

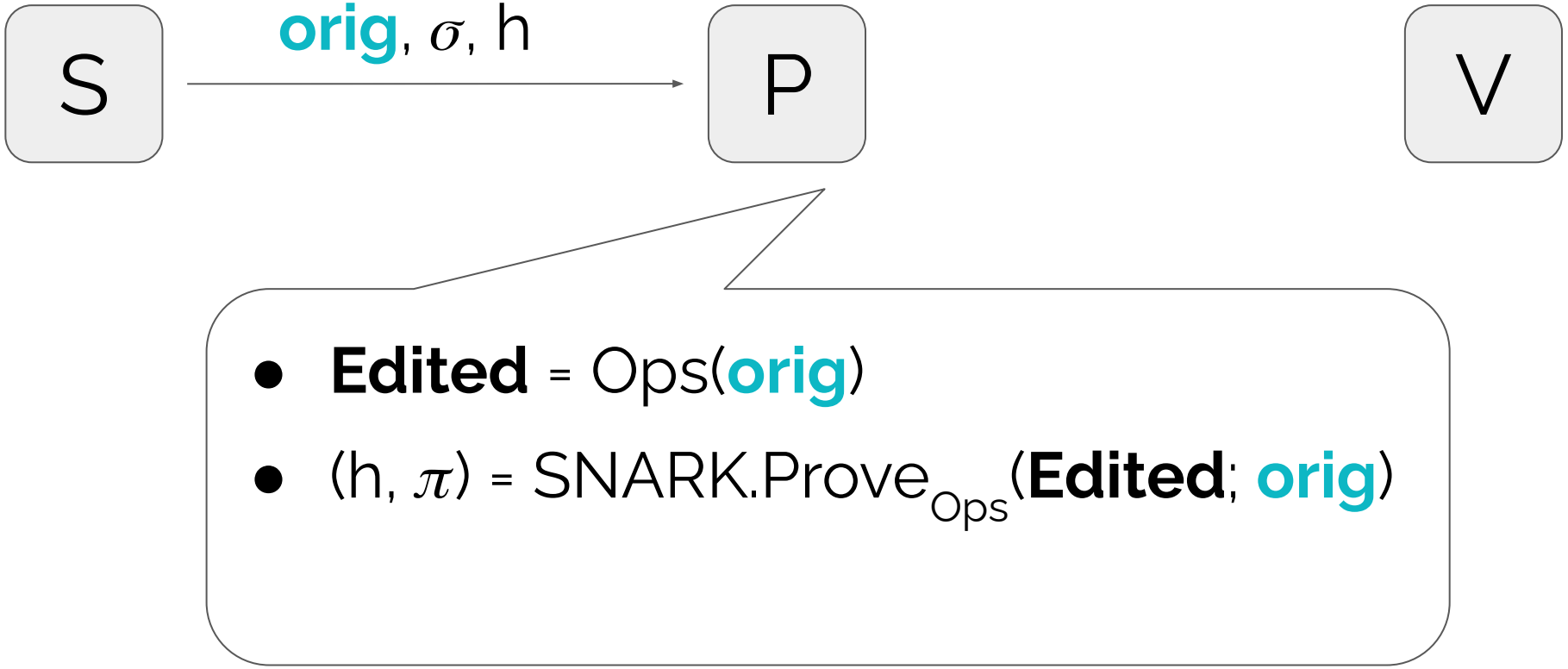
- Create photo **orig**
- h = PCS.Commit(orig)
- σ = Sign(sk, h)

A Commit-and-Prove Scheme

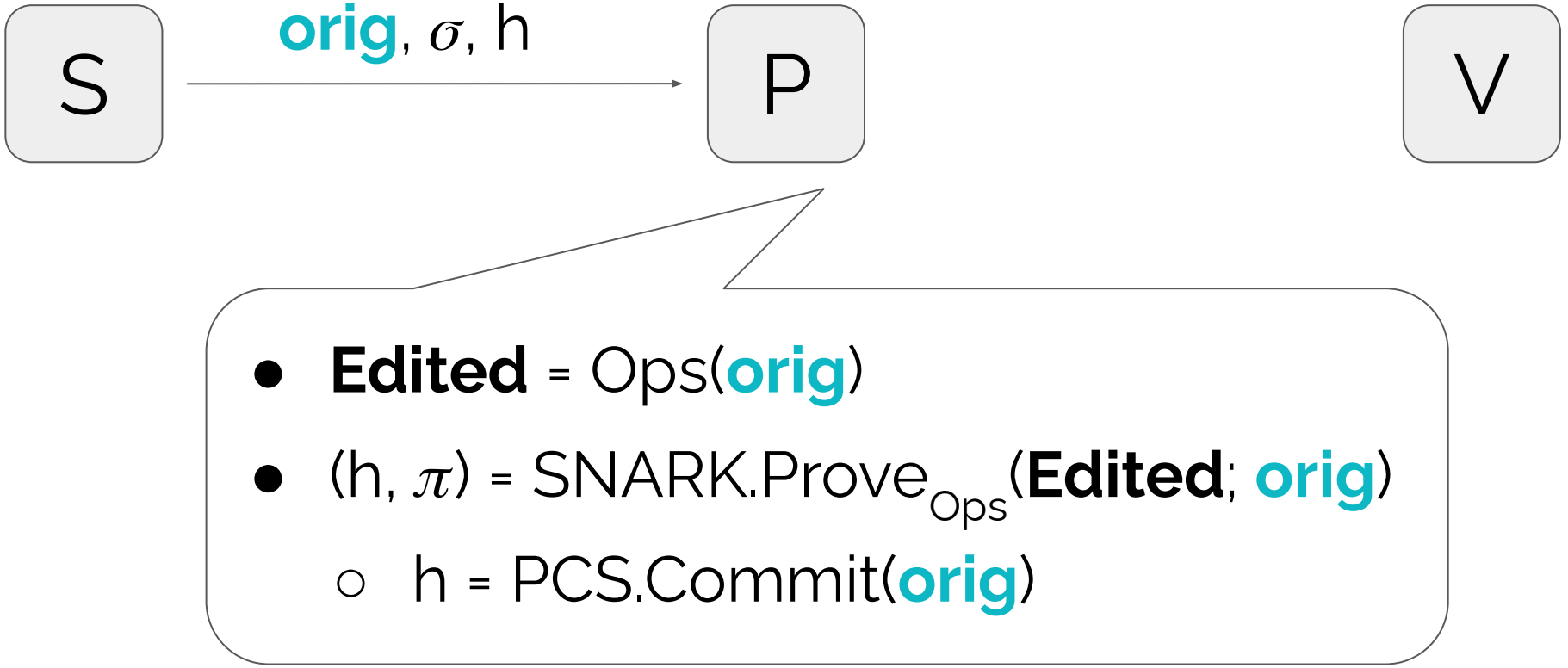


- Create photo **orig**
- h = PCS.Commit(orig)
- σ = Sign(sk, h)

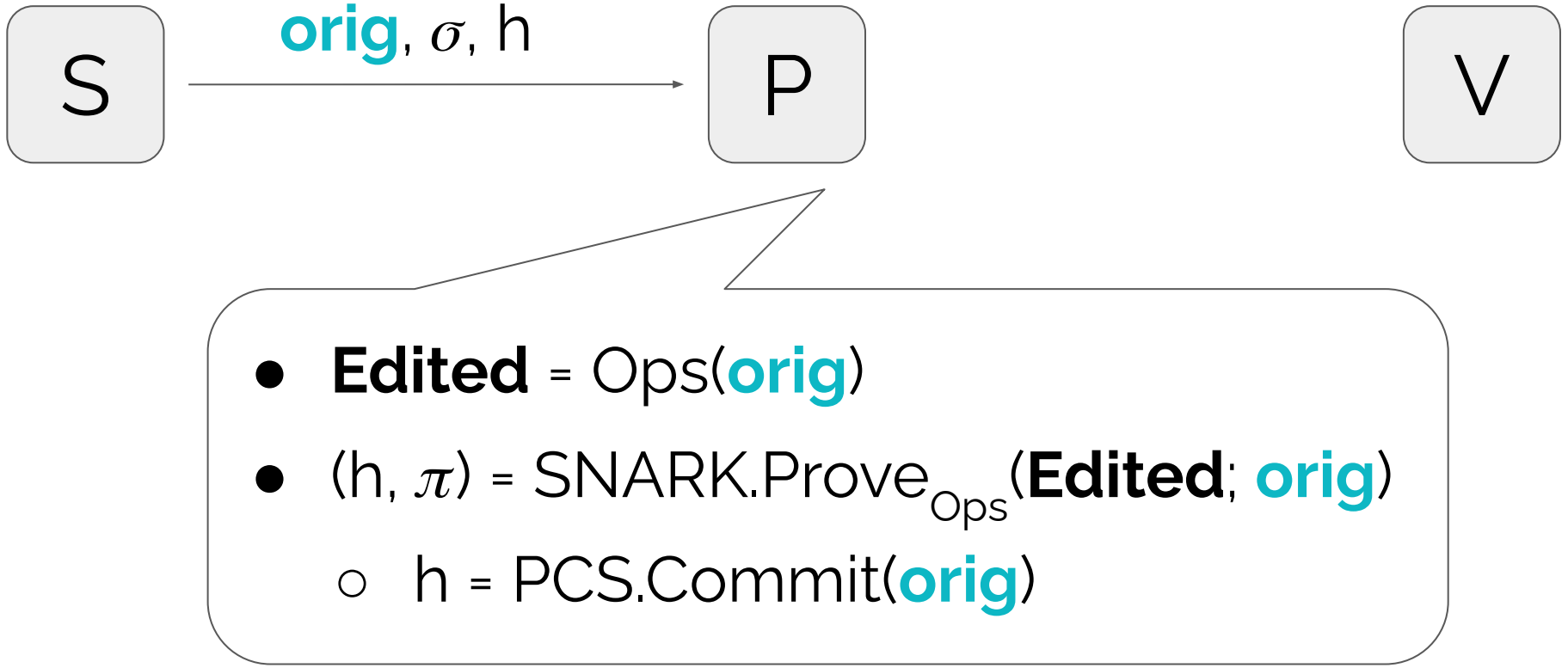
A Commit-and-Prove Scheme



A Commit-and-Prove Scheme

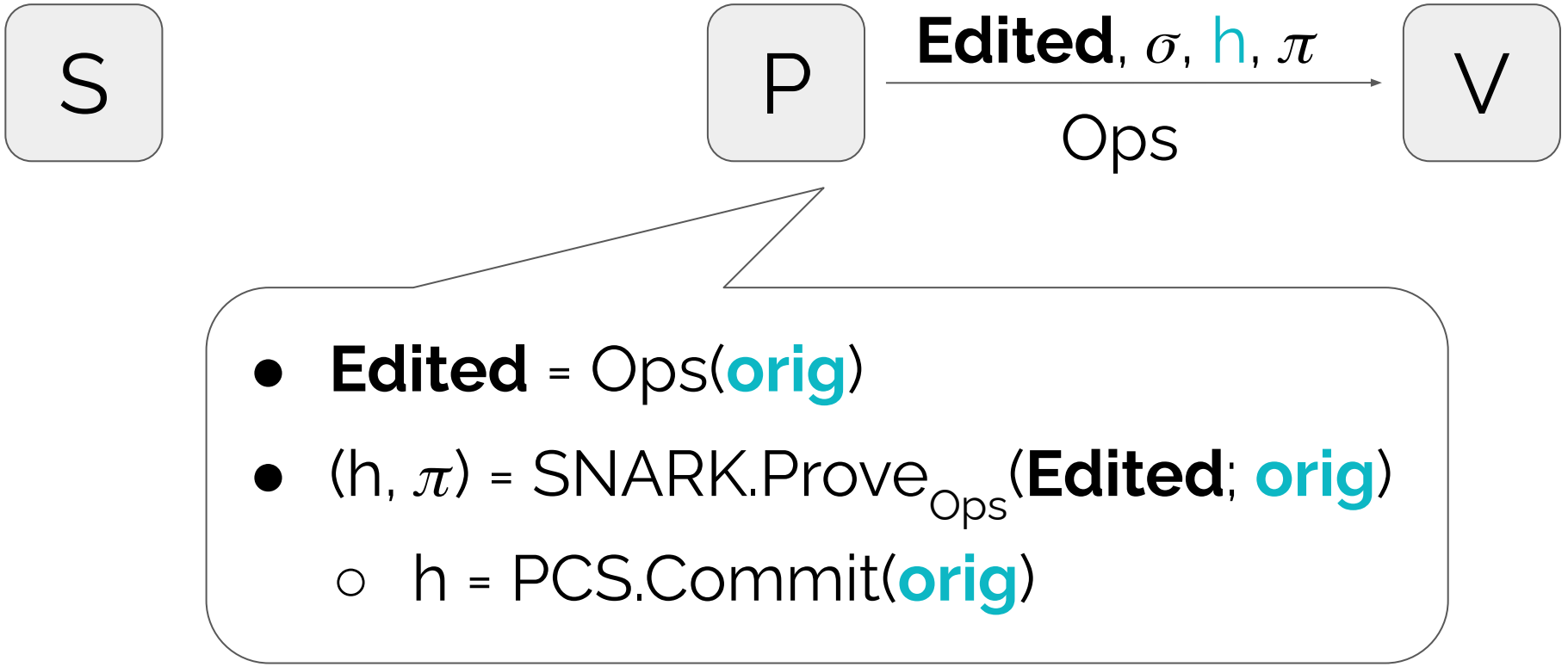


A Commit-and-Prove Scheme



Advantage: Prove the image editing function only

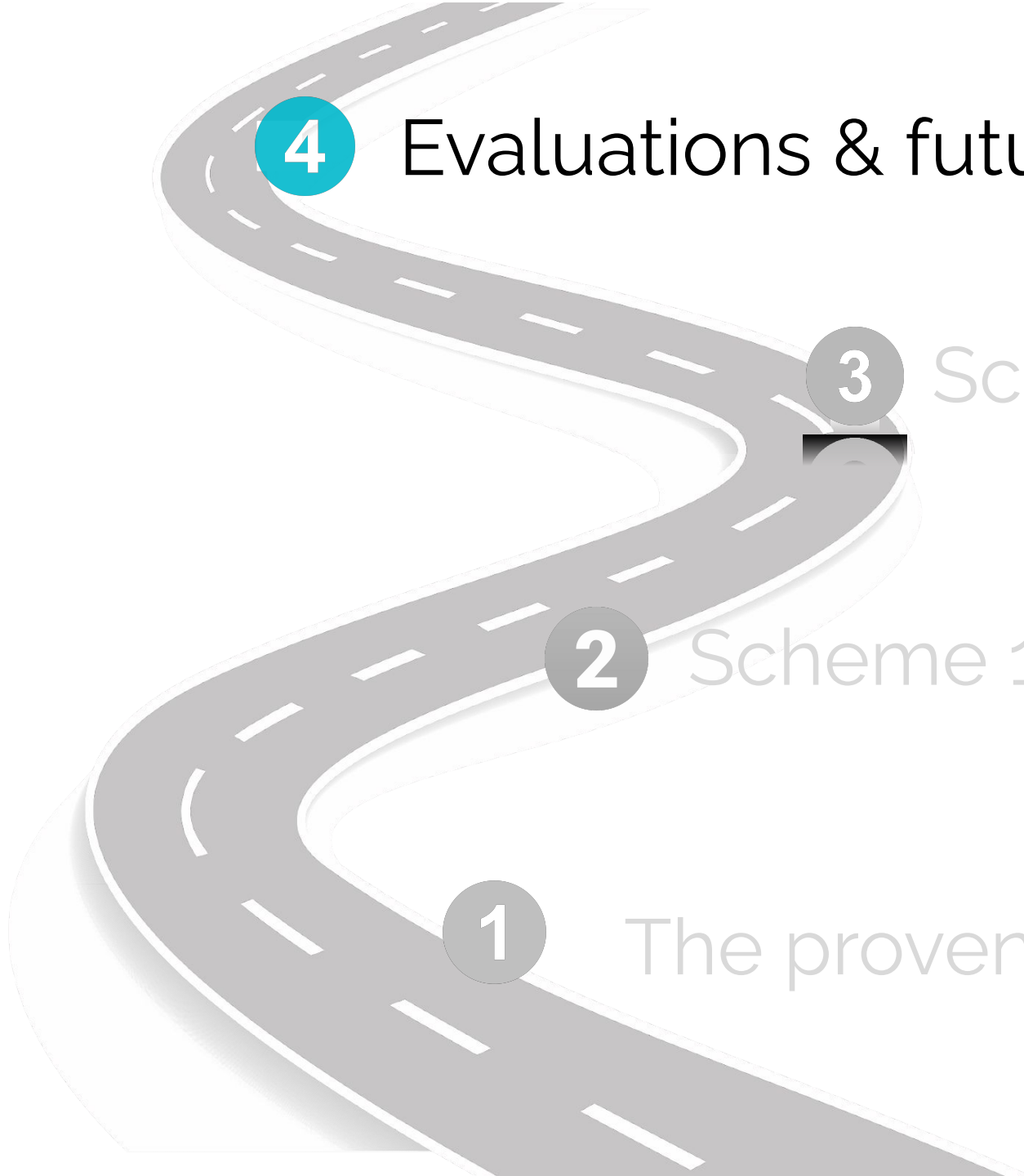
A Commit-and-Prove Scheme



A Commit-and-Prove Scheme

Summary

- A scheme with 5-10x faster proof generation
- Fit for powerful signers (e.g. OpenAI)
- Limitation: Signing is more heavyweight



4 Evaluations & future directions

3 Scheme 2: Faster proving

2 Scheme 1: Lightweight signing

1 The provenance framework

Evaluation for 30MP Photos

Photo editing proof:

- 0.93 ~ 4.41 mins / $0.13 on AWS

Lattice hash proof:

- 10.25 mins / $0.41 on AWS (>5x faster than Poseidon)
- ~0.7s (optimized) verification

Plain hashing:

Hashing Scheme	Time (s)	Memory (GB)
SHA256	1.71	0.003
Lattice (64 bit)	4.24	0.003
FRI-PCS	19.84	18.90

Summary & Future Directions

A zkSNARK-based image provenance system

- Supports >90MB images
- **Mode 1:** Lightweight hashing + fast proof generation
- **Mode 2:** Further 5-10x proof generation acceleration

Summary & Future Directions

A zkSNARK-based image provenance system

- Supports >90MB images
- **Mode 1:** Lightweight hashing + fast proof generation
- **Mode 2:** Further 5-10x proof generation acceleration

Future directions

- Lightweight signing without hash proof
- Multi-hop photo editing
- Video transformation


THANKS