Image Authenticity:
Slightly Homomorphic Digital Signatures and Privacy Preserving Folding Schemes

Simon Erfurth

Universitat Rovira i Virgili, 20/2 2025

simon@serfurth.dk
www.serfurth.dk
@serfurth.dk

SDU DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

DDC digital democracy centre

Agenda

Agenda

1. Motivation: Tracing the provenance of images.

Agenda

1. Motivation: Tracing the provenance of images.

STRENGTHENING AUTHENTICITY AND MITIGATING MISINFORMATION

SIMON ERFURTH

DECEMBER 2024

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

digital democracy centre

Agenda

1. Motivation: Tracing the provenance of images.
2. A solution supporting JPEG compression.

STRENGTHENING AUTHENTICITY AND MITIGATING MISINFORMATION

SIMON ERFURTH

DECEMBER 2024

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

digital democracy centre

Agenda

1. Motivation: Tracing the provenance of images.
2. A solution supporting JPEG compression.

STRENGTHENING AUTHENTICITY AND MITIGATING MISINFORMATION

SLIGHTLY HOMOMORPHIC DIGITAL SIGNATURES

SIMON ERFURTH

DECEMBER 2024

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

digital democracy centre

Agenda

1. Motivation: Tracing the provenance of images.
2. A solution supporting JPEG compression.
3. Towards a general solution from SNARKs.

STRENGTHENING AUTHENTICITY
AND MITIGATING MISINFORMATION

SLIGHTLY HOMOMORPHIC DIGITAL SIGNATURES

SIMON ERFURTH

DECEMBER 2024

DEPARTMENT OF MATHEMATICS
AND COMPUTER SCIENCE

digital democracy centre

Agenda

1. Motivation: Tracing the provenance of images.
2. A solution supporting JPEG compression.
3. Towards a general solution from SNARKs.

STRENGTHENING AUTHENTICITY AND MITIGATING MISINFORMATION

SLIGHTLY HOMOMORPHIC DIGITAL SIGNATURES AND PRIVACY PRESERVING FOLDING SCHEMES

SIMON ERFURTH

DECEMBER 2024

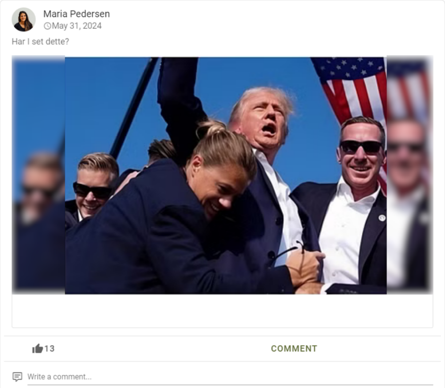
SDU
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

DDC
digital democracy centre

Motivation: Tracing the provenance of images

Strengthening Authenticity and Mitigating Misinformation

Image Provenance



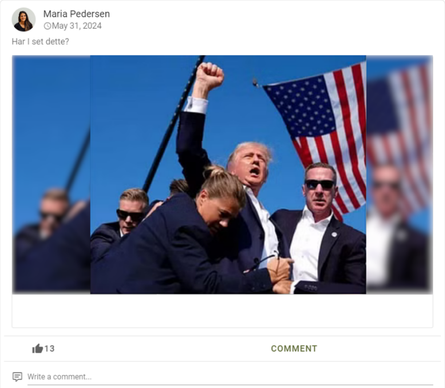


Image Provenance

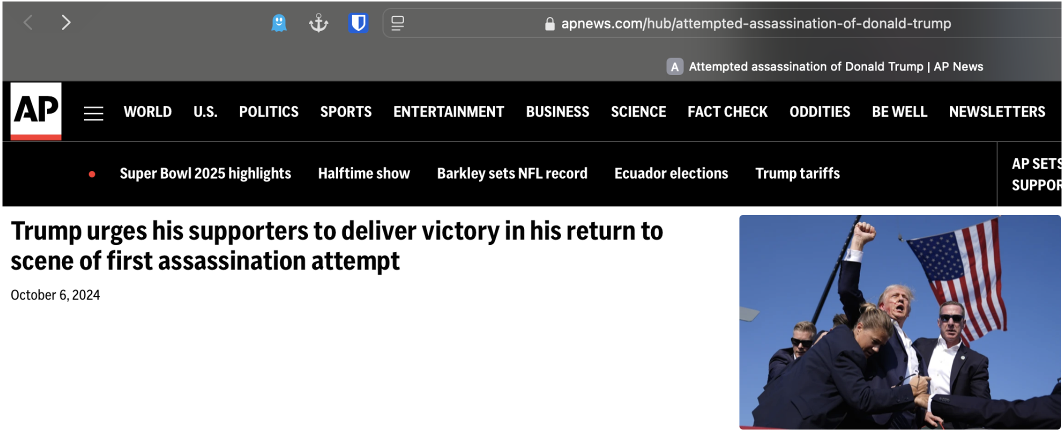
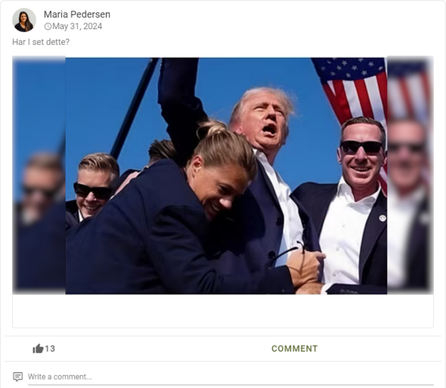


Image Provenance



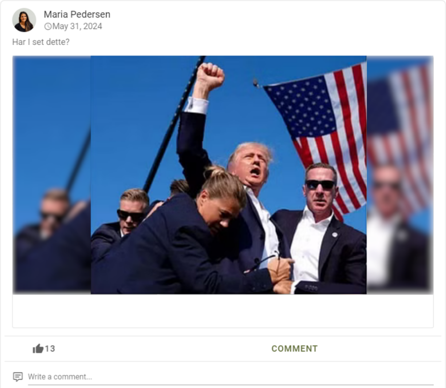
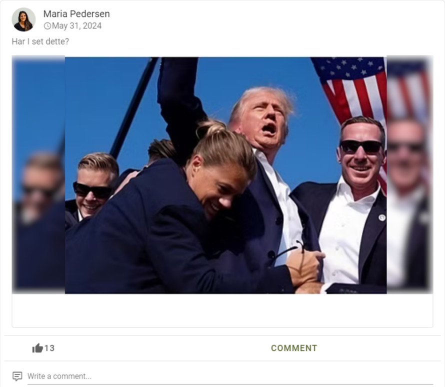


Image Provenance



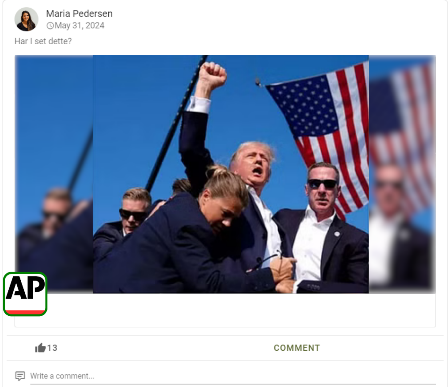


Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

But...

[PLZ+09]

Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

But...

[PLZ+09]

Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

But...

[PLZ+09]

Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

But...

[PLZ+09]

Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

But...

[PLZ+09]

Image Provanance from Digital Signatures

A straight forward solution?

Picture provider signs a signature for the image.

But...

[PLZ+09]

Standard digital signatures do not work with (lossy) transformations.

A solution supporting JPEG compression

Slightly Homomorphic Digital Signatures

Slightly Homomorphic Digital Signatures

P-Homomorphic Signature Scheme

Slightly Homomorphic Digital Signatures

P-Homomorphic Signature Scheme

For message space $\mathcal{M}$, fix predicate $P\colon \mathcal{M} \times \mathcal{P}(\mathcal{M}) \to \{0, 1\}$.

Slightly Homomorphic Digital Signatures

P-Homomorphic Signature Scheme

For message space $\mathcal{M}$, fix predicate $P\colon \mathcal{M} \times \mathcal{P}(\mathcal{M}) \to \{0,1\}$.
$(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ regular signature scheme

Slightly Homomorphic Digital Signatures

P-Homomorphic Signature Scheme

For message space $\mathcal{M}$, fix predicate $P \colon \mathcal{M} \times \mathcal{P}(\mathcal{M}) \to \{0,1\}$.
$(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ regular signature scheme with additional property that

- If $P(m, \{m_1, \ldots, m_i\}) = 1$:
- then *anyone* can extract a signature for m signed with sk from signatures for $m_1, \ldots, m_i$ all signed with sk.

Slightly Homomorphic Digital Signatures

P-Homomorphic Signature Scheme

For message space $\mathcal{M}$, fix predicate $P\colon \mathcal{M} \times \mathcal{P}(\mathcal{M}) \to \{0,1\}$.
(KeyGen, Sign, Verify) regular signature scheme with additional property that

- If $P(m, \{m_1, \ldots, m_i\}) = 1$:
- then *anyone* can extract a signature for m signed with sk from signatures for $m_1, \ldots, m_i$ all signed with sk.

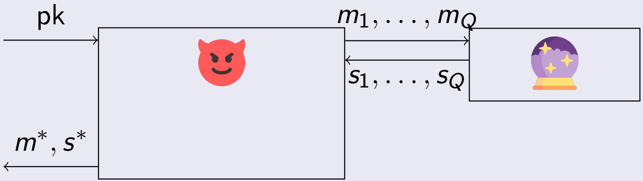
For JPEG compression: Predicate $P(m', M)$ returns 1 if and only if $|M| = 1$ and m' a compression of $m \in M$..

Slightly Homomorphic Digital Signatures

Unforgeability

EUF-CMA

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$



if $(\mathsf{Ver}_{\mathsf{pk}}(m^*, s^*) = 1)$ and $(\forall M \subseteq \{m_1, \ldots, m_Q\} \colon P(m^*, M) = 0)$:

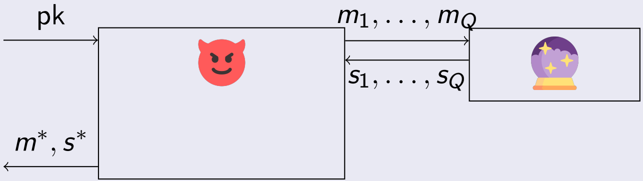
$\mathsf{Adv}^{\mathcal{O}_{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}}(\mathsf{pk})$ wins

Slightly Homomorphic Digital Signatures

Unforgeability

EUF-CMA

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}()$



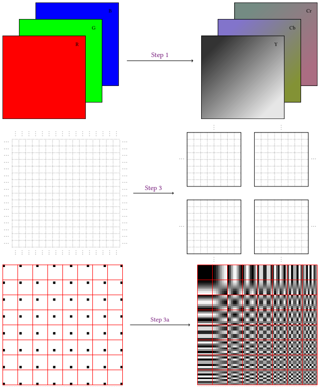
if $(\mathsf{Ver}_{\mathsf{pk}}(m^*, s^*) = 1)$ and $(\forall M \subseteq \{m_1, \ldots, m_Q\} \colon P(m^*, M) = 0)$:

$\quad \mathsf{Adv}^{\mathcal{O}_{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}}(\mathsf{pk})$ wins

Towards our construction: JPEG Compression

Towards our construction: JPEG Compression

1. RGB to YCbCr color space
2. Optional: Down sample
3. Split into 8×8
   a. Discrete cosine transformation
   b. Quantization
4. Encode using entropy encoder

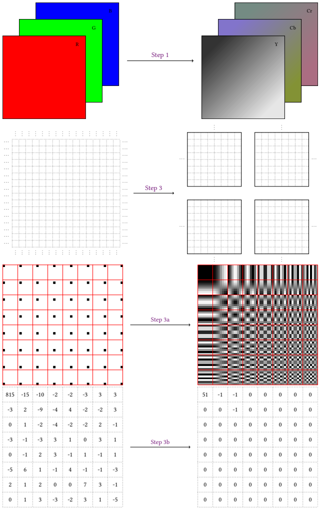


815	-15	-10	-2	-2	-3	3	3
-3	2	-9	-4	4	-2	-2	3
0	1	-2	-4	-2	-2	2	-1
-3	-1	-3	3	1	0	3	1
0	-1	2	3	-1	1	-1	1
-5	6	1	-1	4	-1	-1	-3
2	1	2	0	0	7	3	-1
0	1	3	-3	-2	3	1	-5

51	-1	-1	0	0	0	0	0
0	0	-1	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Towards our construction: JPEG Compression

1. RGB to YCbCr color space
2. Optional: Down sample
3. Split into 8×8
   a. Discrete cosine transformation
   b. Quantization
4. Encode using entropy encoder



$$\begin{array}{cccccccc} 815 & -15 & -10 & -2 & -2 & -3 & 3 & 3 \\ -3 & 2 & -9 & -4 & 4 & -2 & -2 & 3 \\ 0 & 1 & -2 & -4 & -2 & -2 & 2 & -1 \\ -3 & -1 & -3 & 3 & 1 & 0 & 3 & 1 \\ 0 & -1 & 2 & 3 & -1 & 1 & -1 & 1 \\ -5 & 6 & 1 & -1 & 4 & -1 & -1 & -3 \\ 2 & 1 & 2 & 0 & 0 & 7 & 3 & -1 \\ 0 & 1 & 3 & -3 & -2 & 3 & 1 & -5 \end{array} \xrightarrow{\text{Step 3b}} \begin{array}{cccccccc} 51 & -1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Construction: 1 Byte

Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3 xxx$$

Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0}{2^3}$$
$$\rightsquigarrow b_7 b_6 b_5 b_4 b_3 xxx$$

Idea: Provide something instead of xxx?

Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3 xxx$$

Idea: Provide something instead of xxx?

$b = \quad b_7 \quad b_6 \quad b_5 \quad b_4 \quad b_3 \quad b_2 \quad b_1 \quad b_0$

Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\,b_6\,b_5\,b_4\,b_3\,b_2\,b_1\,b_0}{2^3}$$

$$\rightsquigarrow b_7\,b_6\,b_5\,b_4\,b_3 xxx$$

Idea: Provide something instead of xxx?

$h_0 = H(b_0)$

$$b = \quad b_7 \quad b_6 \quad b_5 \quad b_4 \quad b_3 \quad b_2 \quad b_1 \quad b_0$$

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3 xxx$$

Idea: Provide something instead of xxx?

$h_1 = H(h_0 \| b_1) \leftarrow h_0 = H(b_0)$

$b = \quad b_7 \quad b_6 \quad b_5 \quad b_4 \quad b_3 \quad b_2 \quad b_1 \quad b_0$

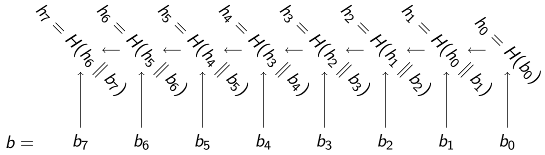
Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3 xxx$$

Idea: Provide something instead of xxx?



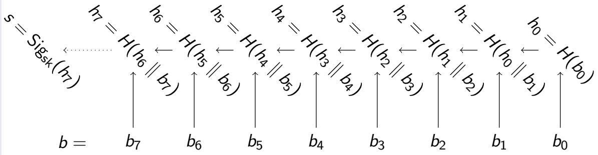
Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3 xxx$$

Idea: Provide something instead of xxx?



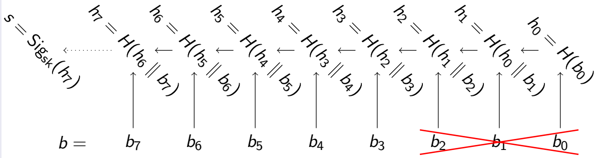
Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3\, xxx$$

Idea: Provide something instead of xxx?



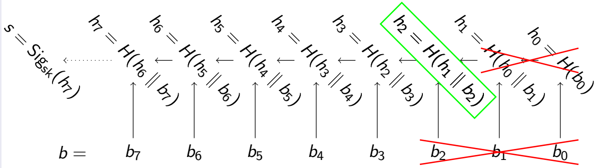
Construction: 1 Byte

Observation: Quantization with 2^n is truncation

$$\frac{b_7\, b_6\, b_5\, b_4\, b_3\, b_2\, b_1\, b_0}{2^3}$$

$$\rightsquigarrow b_7\, b_6\, b_5\, b_4\, b_3 xxx$$

Idea: Provide something instead of xxx?



Construction: 1 Byte

Observation: Quantization with 2^n is truncation

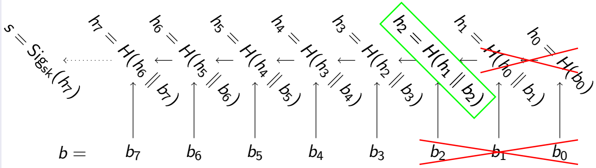
$$\frac{b_7\,b_6\,b_5\,b_4\,b_3\,b_2\,b_1\,b_0}{2^3}$$

$$\rightsquigarrow b_7\,b_6\,b_5\,b_4\,b_3 xxx$$

Idea: Provide something instead of *xxx*?

However...

... it is super inefficient!



Construction: Images

But...

- ...each of the 64 DCT coefficients are truncated the same in every 8×8 block
- and images generally have many pixels/blocks

Construction: Images

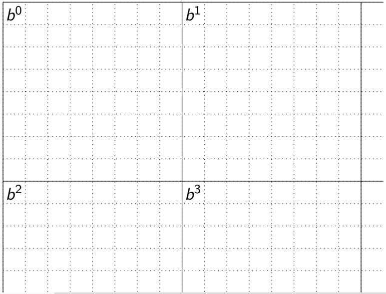
But...

- ...each of the 64 DCT coefficients are truncated the same in every 8×8 block
- and images generally have many pixels/blocks

Our solution

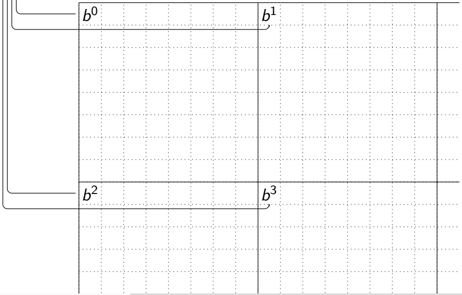
- Generate signature by “combining” matching DCT coefficients and then generate hash chains and hash ends together.
- Compress using quantization table with powers of two and update signature to include relevant nodes.

Construction: Images

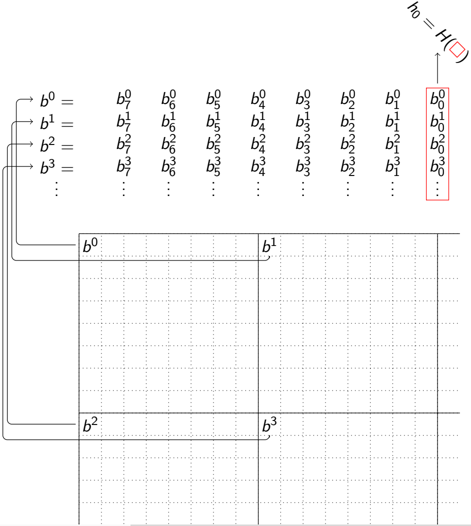


Construction: Images

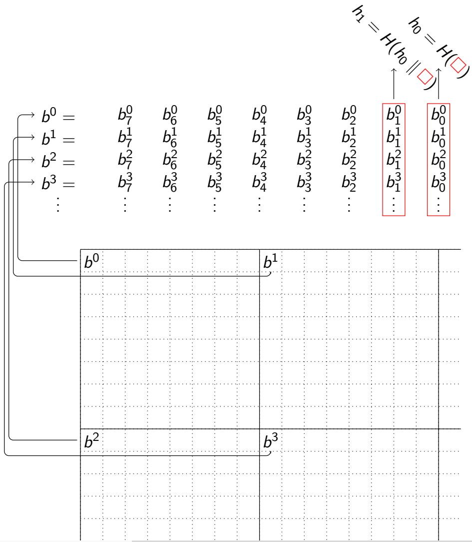
$$\begin{array}{lcccccccc}
b^0 = & b^0_7 & b^0_6 & b^0_5 & b^0_4 & b^0_3 & b^0_2 & b^0_1 & b^0_0 \\
b^1 = & b^1_7 & b^1_6 & b^1_5 & b^1_4 & b^1_3 & b^1_2 & b^1_1 & b^1_0 \\
b^2 = & b^2_7 & b^2_6 & b^2_5 & b^2_4 & b^2_3 & b^2_2 & b^2_1 & b^2_0 \\
b^3 = & b^3_7 & b^3_6 & b^3_5 & b^3_4 & b^3_3 & b^3_2 & b^3_1 & b^3_0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}$$



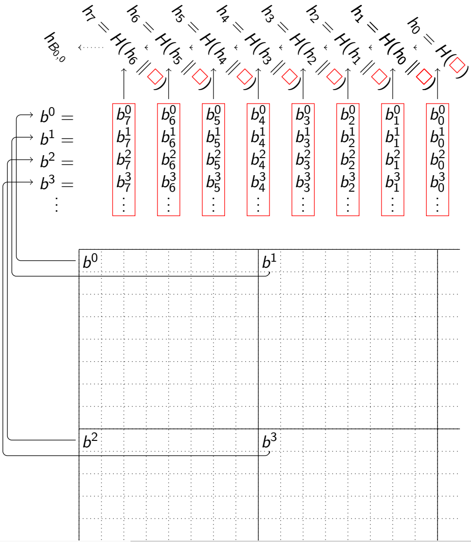
Construction: Images



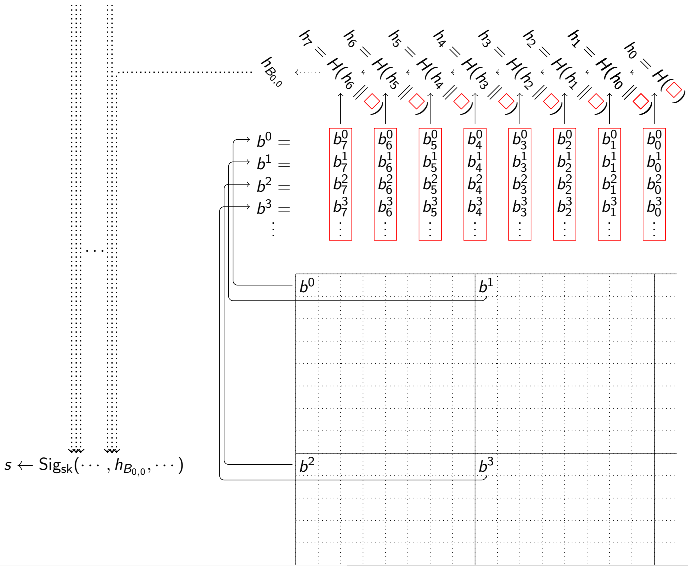
Construction: Images



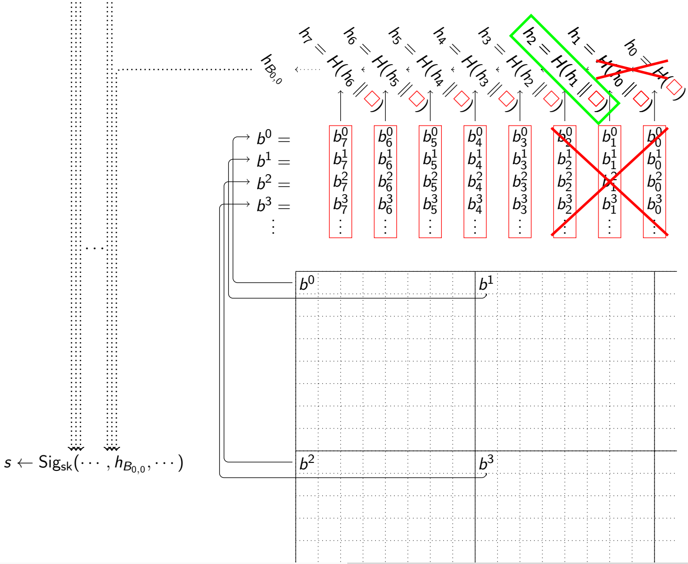
Construction: Images



Construction: Images



Construction: Images



Construction: Summary

Digital signature scheme allowing JPEG compression

Construction: Summary

Digital signature scheme allowing JPEG compression

Let H be a cryptographic hash function and $\mathsf{DS} = (\mathsf{KeyGen}^{\mathsf{DS}}, \mathsf{Sign}^{\mathsf{DS}}, \mathsf{Verify}^{\mathsf{DS}})$ a standard digital signature scheme.

Construction: Summary

Digital signature scheme allowing JPEG compression

Let H be a cryptographic hash function and $\mathsf{DS} = (\mathsf{KeyGen}^{\mathsf{DS}}, \mathsf{Sign}^{\mathsf{DS}}, \mathsf{Verify}^{\mathsf{DS}})$ a standard digital signature scheme.

- $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{KeyGen}(1^\lambda)$: Identical to $\mathsf{KeyGen}^{\mathsf{DS}}(1^\lambda)$.

Construction: Summary

Digital signature scheme allowing JPEG compression

Let H be a cryptographic hash function and $\mathsf{DS} = (\mathsf{KeyGen}^{\mathsf{DS}}, \mathsf{Sign}^{\mathsf{DS}}, \mathsf{Verify}^{\mathsf{DS}})$ a standard digital signature scheme.

- $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{KeyGen}(1^\lambda)$: Identical to $\mathsf{KeyGen}^{\mathsf{DS}}(1^\lambda)$.
- $s \leftarrow \mathsf{Sign}_{\mathsf{sk}}(i)$: Compute the chains of hashes, sign the ends using $\mathsf{Sign}^{\mathsf{DS}}$.

Construction: Summary

Digital signature scheme allowing JPEG compression

Let H be a cryptographic hash function and $\mathsf{DS} = (\mathsf{KeyGen}^{\mathsf{DS}}, \mathsf{Sign}^{\mathsf{DS}}, \mathsf{Verify}^{\mathsf{DS}})$ a standard digital signature scheme.

- $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{KeyGen}(1^\lambda)$: Identical to $\mathsf{KeyGen}^{\mathsf{DS}}(1^\lambda)$.
- $s \leftarrow \mathsf{Sign}_{\mathsf{sk}}(i)$: Compute the chains of hashes, sign the ends using $\mathsf{Sign}^{\mathsf{DS}}$.
- $i', s' \leftarrow \mathsf{Compress}(i, s, P)$: Compress i using quantization tables P, compute chains of hashes and extract relevant ones. Add these to s.

Construction: Summary

Digital signature scheme allowing JPEG compression

Let H be a cryptographic hash function and $\mathsf{DS} = (\mathsf{KeyGen}^{\mathsf{DS}}, \mathsf{Sign}^{\mathsf{DS}}, \mathsf{Verify}^{\mathsf{DS}})$ a standard digital signature scheme.

- $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{KeyGen}(1^\lambda)$: Identical to $\mathsf{KeyGen}^{\mathsf{DS}}(1^\lambda)$.
- $s \leftarrow \mathsf{Sign}_{\mathsf{sk}}(i)$: Compute the chains of hashes, sign the ends using $\mathsf{Sign}^{\mathsf{DS}}$.
- $i', s' \leftarrow \mathsf{Compress}(i, s, P)$: Compress i using quantization tables P, compute chains of hashes and extract relevant ones. Add these to s.
- $0/1 \leftarrow \mathsf{Verify}_{\mathsf{pk}}(i, s)$: Use i and hashes in s, if any, to find chain ends. Verify with $\mathsf{Verify}^{\mathsf{DS}}$.

Security and Performance Claims

Security and Performance Claims

Unforgeability

Constructed signature scheme allowing image compression is EUF-CMA.

Security and Performance Claims

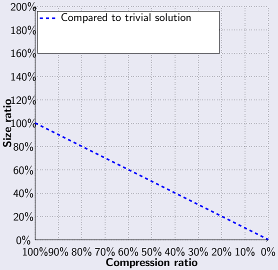
Unforgeability

Constructed signature scheme allowing image compression is EUF-CMA.

Performance: Signature Size

Uncompressed image 2 MB

- - - $|S| = 512$ bits.



Security and Performance Claims

Unforgeability

Constructed signature scheme allowing image compression is EUF-CMA.

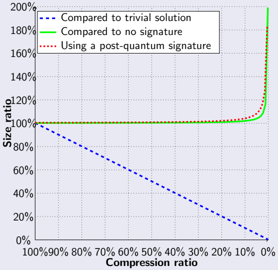
Performance: Signature Size

Uncompressed image 2 MB

- - - $|S| = 512$ bits.

— $|S| = 512$ bits, $|H| = 256$ bits.

······ $|S| = 36760$ bits, $|H| = 256$ bits.



Security and Performance Claims

Unforgeability

Constructed signature scheme allowing image compression is EUF-CMA.

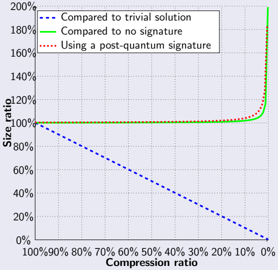
Performance: Signature Size

Uncompressed image 2 MB

$|S| = 512$ bits.

$|S| = 512$ bits,
$|H| = 256$ bits.

$|S| = 36760$ bits,
$|H| = 256$ bits.



Visual Fidelity

Security and Performance Claims

Unforgeability

Constructed signature scheme allowing image compression is EUF-CMA.

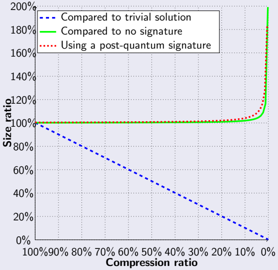
Performance: Signature Size

Uncompressed image 2 MB

$|S| = 512$ bits.

$|S| = 512$ bits,
$|H| = 256$ bits.

$|S| = 36760$ bits,
$|H| = 256$ bits.



Visual Fidelity

Uncompressed

Classic (QF50)

Powers of two

Security and Performance Claims

Unforgeability

Constructed signature scheme allowing image compression is EUF-CMA.

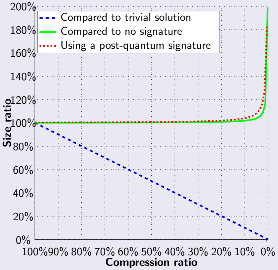
Performance: Signature Size

Uncompressed image 2 MB

- - - $|S| = 512$ bits.

— $|S| = 512$ bits, $|H| = 256$ bits.

······ $|S| = 36760$ bits, $|H| = 256$ bits.



Visual Fidelity

Uncompressed

Classic (QF50)

Powers of two

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	16.0 kB	0.960	0.978	77.526	29.749
	Unmodified	15.1 kB	0.959	0.978	78.900	29.655
	[JWL11] (64)	10.4 kB	0.914	0.936	109.016	28.021
	[JWL11] (32)	20.5 kB	0.961	0.976	46.071	31.706
QF50	**Our tables**	25.4 kB	0.979	0.991	45.831	32.008
	Unmodified	24.4 kB	0.979	0.991	45.910	31.988
	[JWL11] (32)	20.5 kB	0.961	0.976	46.071	31.706
	[JWL11] (16)	36.5 kB	0.983	0.992	18.451	35.605
QF80	**Our tables**	43.9 kB	0.990	0.997	20.256	35.402
	Unmodified	43.4 kB	0.991	0.997	20.432	35.364
	[JWL11] (16)	36.5 kB	0.983	0.992	18.451	35.605
	[JWL11] (8)	60.4 kB	0.993	0.997	7.532	39.439

Average over all images in [PLZ+09].

Towards a general solution from SNARKs

Privacy Preserving Folding Schemes

Image Provenance for Other Transformations

[PLZ+09]

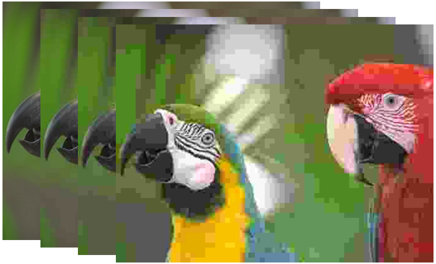


Image Provenance for Other Transformations

[PLZ+09]

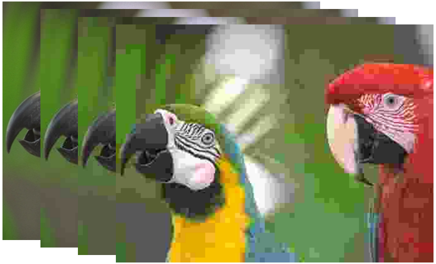


Image Provenance for Other Transformations

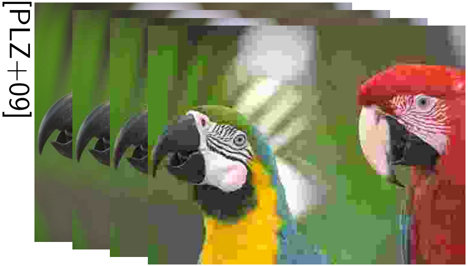


Image Provenance for Other Transformations

[PLZ+09]

Slightly homomorphic signatures support other transformations, but are not practical [JWL11]

Image Provenance for Other Transformations

[PLZ+09]

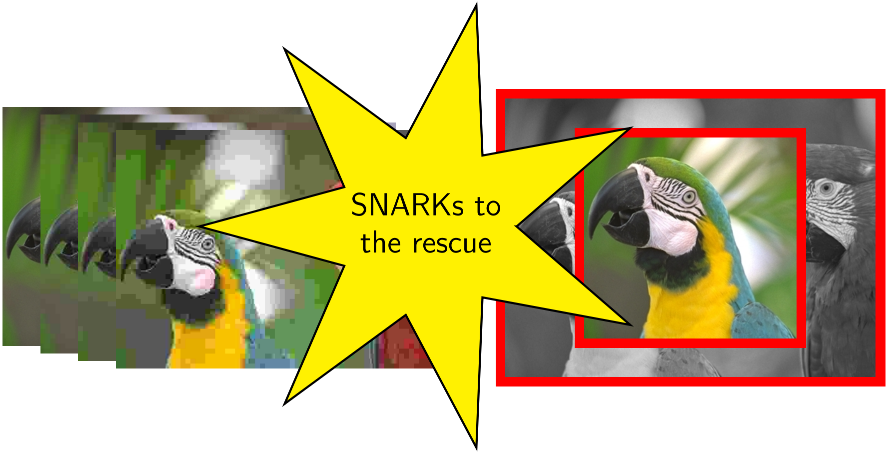


Image Provenance from SNARKs [NT16; DB23; DCB25; DEH25; MVVZ25]

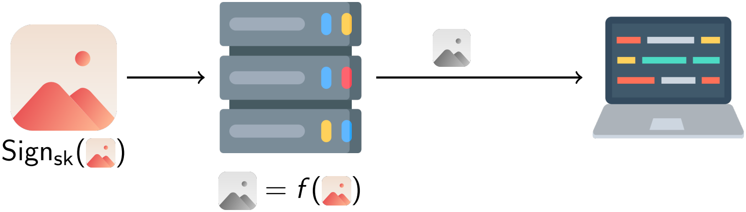
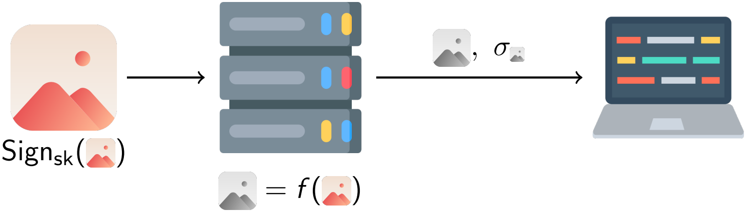


Image Provenance from SNARKs [NT16; DB23; DCB25; DEH25; MVVZ25]



$$\sigma = \begin{cases} \text{I know such that:} \\ 1.\ \text{Sign}_{sk}(\ )\ \text{is valid for} \\ 2.\ \text{is result of } f(\ ) \\ 3.\ \text{Metadata}(\ ) = \text{Metadata}(\ ) \end{cases}$$

Image Provenance from SNARKs [NT16; DB23; DCB25; DEH25; MVVZ25]

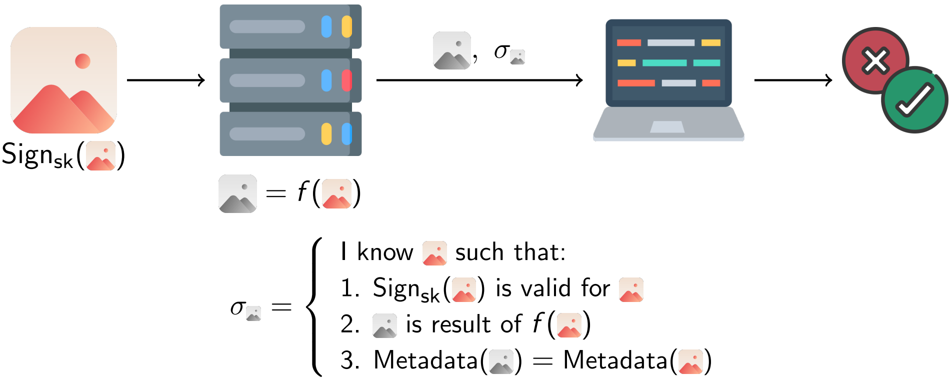
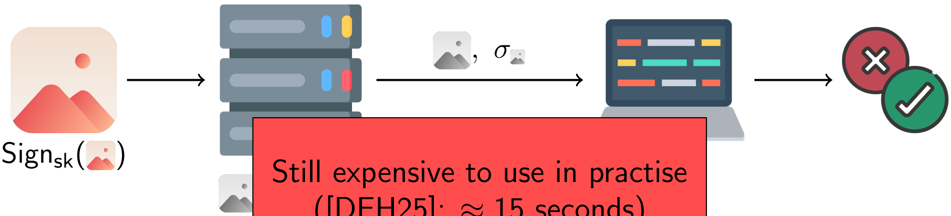


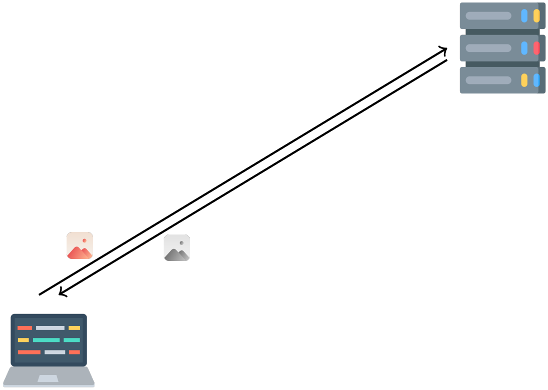
Image Provenance from SNARKs [NT16; DB23; DCB25; DEH25; MVVZ25]



Still expensive to use in practise ([DEH25]: $\approx$ 15 seconds)

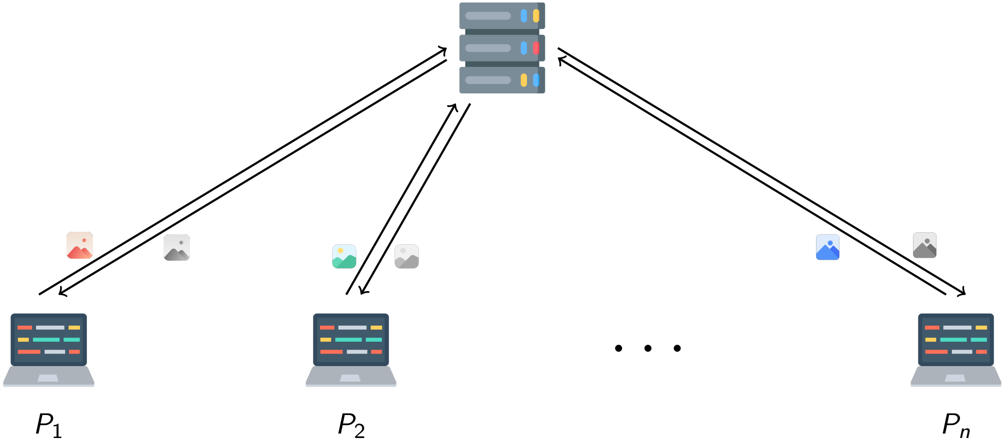
$\sigma =$

1. $\mathsf{Sign}_{sk}(\ )$ is valid for
2. is result of $f(\ )$
3. $\mathsf{Metadata}(\ ) = \mathsf{Metadata}(\ )$

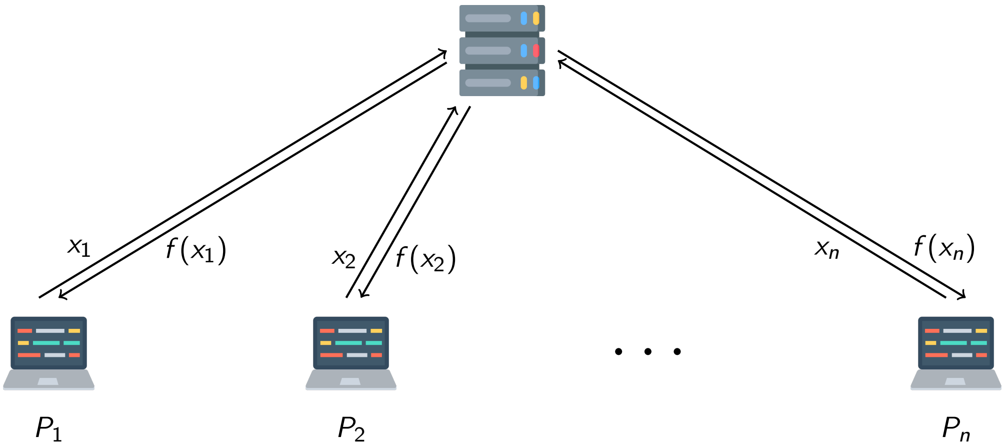


P_1

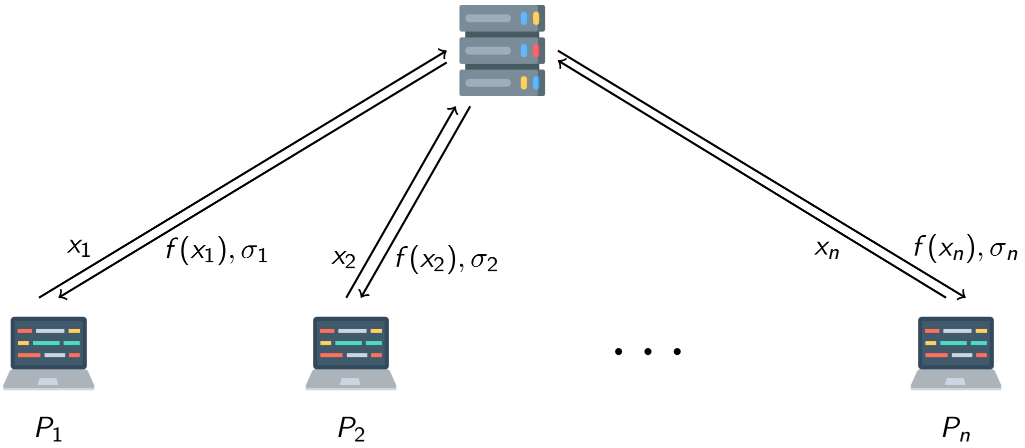
Restating the Problem: Computation as a Service



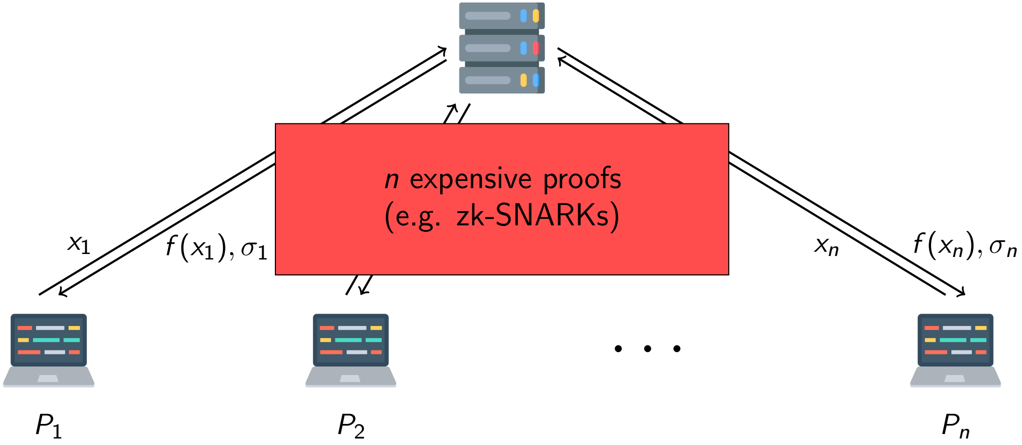
Restating the Problem: Computation as a Service



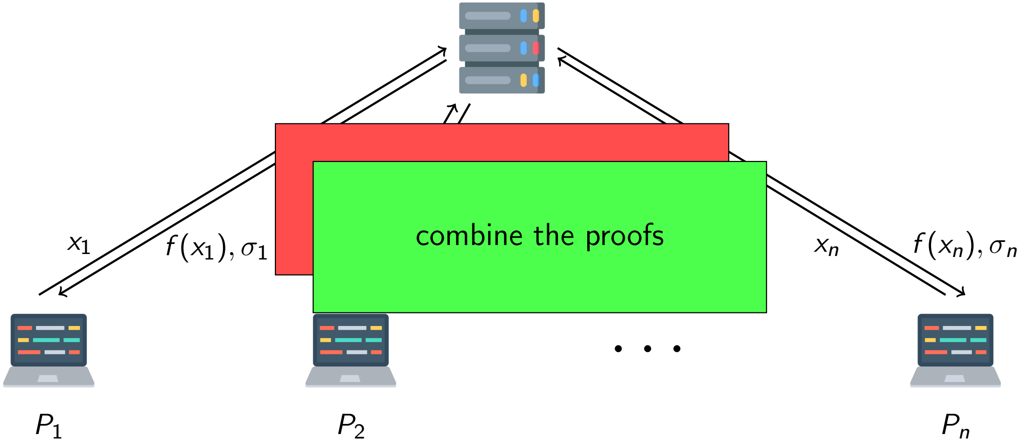
Restating the Problem: Verifiable Computation as a Service



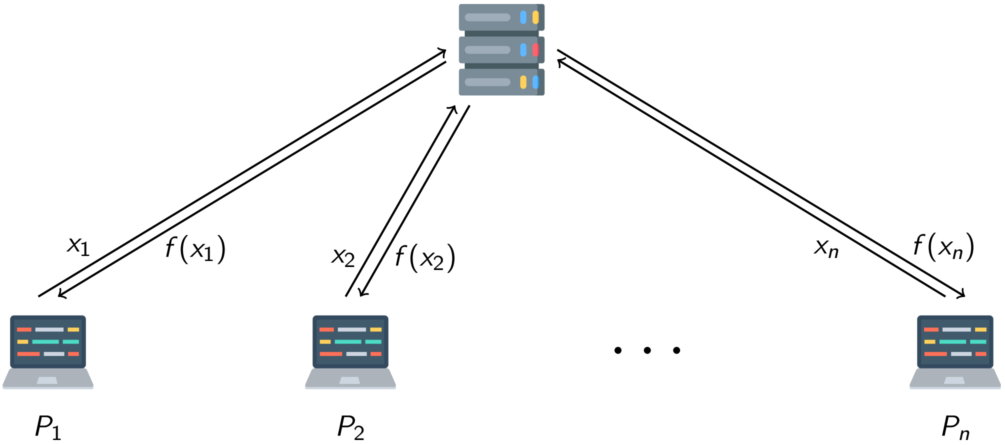
Restating the Problem: Verifiable Computation as a Service



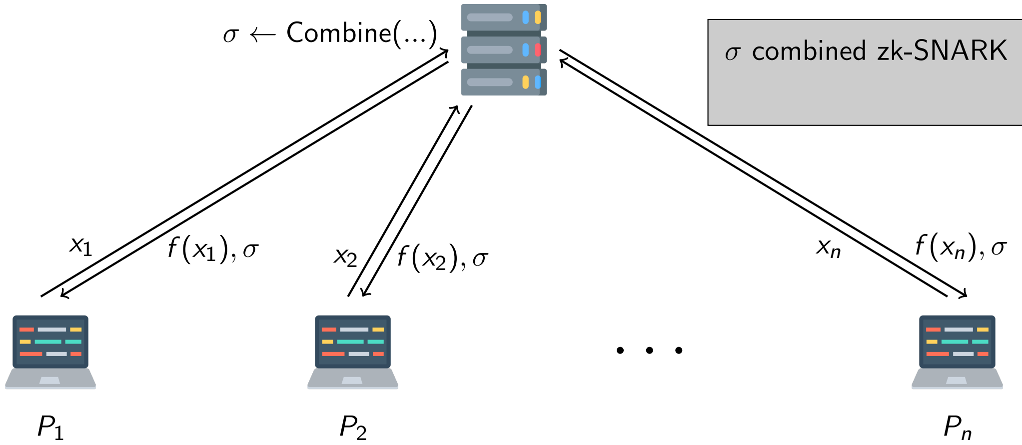
Restating the Problem: Verifiable Computation as a Service



Motivating Example: Verifiable Computation as a Service

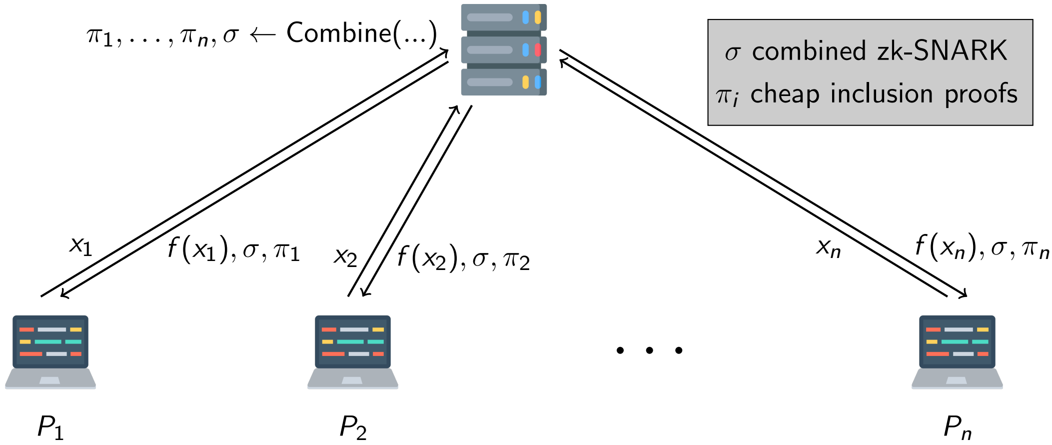


Motivating Example: Verifiable Computation as a Service



σ combined zk-SNARK

Motivating Example: Verifiable Computation as a Service



σ combined zk-SNARK

π_i cheap inclusion proofs

Folding Scheme with Selective Verification [KST22]

Folding Scheme

For NP-language $\mathcal{L}$ with relation

$$\mathcal{R} = \{(x, v) \mid v \text{ is a proof that } x \in \mathcal{L}\},$$

folding scheme FS which

Folding Scheme with Selective Verification [KST22]

Folding Scheme

For NP-language $\mathcal{L}$ with relation

$$\mathcal{R} = \{(x, v) \mid v \text{ is a proof that } x \in \mathcal{L}\},$$

folding scheme FS which

- Combines instances:
  $\mathsf{Fold}\colon ((x_1, v_1), (x_2, v_2)) \to (x, v, \pi)$
  $(x, v) \in \mathcal{R} \iff (x_1, v_1), (x_2, v_2) \in \mathcal{R}$

Folding Scheme with Selective Verification [KST22]

Folding Scheme

For NP-language $\mathcal{L}$ with relation

$$\mathcal{R} = \{(x, v) \mid v \text{ is a proof that } x \in \mathcal{L}\},$$

folding scheme FS which

- Combines instances:
  $\mathsf{Fold}\colon ((x_1, v_1), (x_2, v_2)) \to (x, v, \pi)$
  $(x, v) \in \mathcal{R} \iff (x_1, v_1), (x_2, v_2) \in \mathcal{R}$
- Check statement inclusion
  $\mathsf{FoldVerify}\colon (x_1, x_2, x, \pi) \to 0/1$
  1 if π is proof that x_1 and x_2 were folded into x

Folding Scheme with Selective Verification [KST22]

Folding Scheme

For NP-language $\mathcal{L}$ with relation

$$\mathcal{R} = \{(x, v) \mid v \text{ is a proof that } x \in \mathcal{L}\},$$

folding scheme FS which

- Combines instances:
  $\mathsf{Fold}\colon ((x_1, v_1), (x_2, v_2)) \to (x, v, \pi)$
  $(x, v) \in \mathcal{R} \iff (x_1, v_1), (x_2, v_2) \in \mathcal{R}$
- Check statement inclusion
  $\mathsf{FoldVerify}\colon (x_1, x_2, x, \pi) \to 0/1$
  1 if π is proof that x_1 and x_2 were folded into x

Example

For $A \in \mathbb{F}^{n\times m}$; $\mathcal{L}_A = \{x \mid \exists v\colon Av = x\}$.

Folding Scheme with Selective Verification [KST22]

Folding Scheme

For NP-language $\mathcal{L}$ with relation

$$\mathcal{R} = \{(x, v) \mid v \text{ is a proof that } x \in \mathcal{L}\},$$

folding scheme FS which

- Combines instances:
  $\mathsf{Fold}\colon ((x_1, v_1), (x_2, v_2)) \to (x, v, \pi)$
  $(x, v) \in \mathcal{R} \iff (x_1, v_1), (x_2, v_2) \in \mathcal{R}$
- Check statement inclusion
  $\mathsf{FoldVerify}\colon (x_1, x_2, x, \pi) \to 0/1$
  1 if π is proof that x_1 and x_2 were folded into x

Example

For $A \in \mathbb{F}^{n\times m}$; $\mathcal{L}_A = \{x \mid \exists v\colon Av = x\}$.

- $\mathsf{Fold}((x_1, v_1), (x_2, v_2))$:
  $\rho \leftarrow_\$ \mathbb{F}; \pi = \rho;$
  $$x = x_1 + \rho x_2; \qquad v = v_1 + \rho v_2.$$

Folding Scheme with Selective Verification [KST22]

Folding Scheme

For NP-language $\mathcal{L}$ with relation

$\mathcal{R} = \{(x, v) \mid v \text{ is a proof that } x \in \mathcal{L}\}$,

folding scheme FS which

- Combines instances:
  $\mathsf{Fold}\colon ((x_1, v_1), (x_2, v_2)) \to (x, v, \pi)$
  $(x, v) \in \mathcal{R} \iff (x_1, v_1), (x_2, v_2) \in \mathcal{R}$
- Check statement inclusion
  $\mathsf{FoldVerify}\colon (x_1, x_2, x, \pi) \to 0/1$
  1 if π is proof that x_1 and x_2 were folded into x

Example

For $A \in \mathbb{F}^{n\times m}$; $\mathcal{L}_A = \{x \mid \exists v\colon Av = x\}$.

- $\mathsf{Fold}((x_1, v_1), (x_2, v_2))$:
  $\rho \leftarrow_\$ \mathbb{F}; \pi = \rho;$
  $$x = x_1 + \rho x_2; \qquad v = v_1 + \rho v_2.$$
- $\mathsf{FoldVerify}(x_1, x_2, x, \pi)$: check that
  $$x = x_1 + \rho x_2.$$

Folding Scheme: Security

Folding Scheme: Security

- **Completeness**: No Adv. can output input to Fold in $\mathcal{R}$, which gives output not in $\mathcal{R}$ (or invalid folding proof).

Folding Scheme with Selective Verification

Folding Scheme: Security

- **Completeness**: No Adv. can output input to Fold in $\mathcal{R}$, which gives output not in $\mathcal{R}$ (or invalid folding proof).
- **Knowledge Soundness**: From Adv. giving x_1, x_2, x, v, π where $(x, v) \in \mathcal{R}$ and π is accepted, we can extract witness for x_1, x_2.

Folding Scheme with Selective Verification

Folding Scheme: Security

- **Completeness**: No Adv. can output input to Fold in $\mathcal{R}$, which gives output not in $\mathcal{R}$ (or invalid folding proof).
- **Knowledge Soundness**: From Adv. giving x_1, x_2, x, v, π where $(x, v) \in \mathcal{R}$ and π is accepted, we can extract witness for x_1, x_2.

Example

Folding Scheme with Selective Verification

Folding Scheme: Security

- **Completeness**: No Adv. can output input to Fold in $\mathcal{R}$, which gives output not in $\mathcal{R}$ (or invalid folding proof).
- **Knowledge Soundness**: From Adv. giving x_1, x_2, x, v, π where $(x, v) \in \mathcal{R}$ and π is accepted, we can extract witness for x_1, x_2.

Example

- **Completeness**: $(x_1, v_1), (x_2, v_2) \in \mathcal{R}$ then

$$\begin{aligned} Av =& A(v_1 + \rho v_2) = Av_1 + \rho A v_2 \\ =& x_1 + \rho x_2 = x \end{aligned}$$

Folding Scheme with Selective Verification

Folding Scheme: Security

- **Completeness**: No Adv. can output input to Fold in $\mathcal{R}$, which gives output not in $\mathcal{R}$ (or invalid folding proof).
- **Knowledge Soundness**: From Adv. giving x_1, x_2, x, v, π where $(x, v) \in \mathcal{R}$ and π is accepted, we can extract witness for x_1, x_2.

Example

- **Completeness**: $(x_1, v_1), (x_2, v_2) \in \mathcal{R}$ then

$$\begin{aligned} Av =& A(v_1 + \rho v_2) = Av_1 + \rho A v_2 \\ =& x_1 + \rho x_2 = x \end{aligned}$$

- **Knowledge Soundness**: Run to get $x, v, \pi = \rho$ and $x', v', \pi' = \rho'$ for same input.

$$\begin{aligned} v =& v_1 + \rho v_2 \\ v' =& v_1 + \rho' v_2 \\ \Rightarrow v_2 =& (\rho' - \rho)^{-1}(v' - v) \end{aligned}$$

Folding Scheme with Selective Verification

From 2-folding to 4-folding

Folding Scheme with Selective Verification

From 2-folding to 4-folding

Output of Fold is in $\mathcal{R} \Rightarrow$ **Bootstrapping**

From 2-folding to 4-folding

Output of Fold is in $\mathcal{R}$ $\Rightarrow$ **Bootstrapping**

(x_1, v_1) (x_2, v_2) (x_3, v_3) (x_4, v_4)

Folding Scheme with Selective Verification

From 2-folding to 4-folding

Output of Fold is in $\mathcal{R}$ $\Rightarrow$ **Bootstrapping**

$$(x', v', \pi')$$

Fold

$$(x_1, v_1) \qquad (x_2, v_2) \qquad (x_3, v_3) \qquad (x_4, v_4)$$

Folding Scheme with Selective Verification

From 2-folding to 4-folding

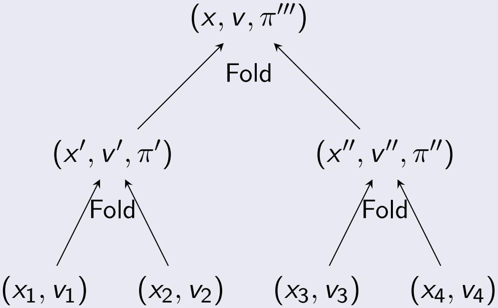
Output of Fold is in $\mathcal{R}$ $\Rightarrow$ **Bootstrapping**

$$
\begin{tikzcd}
 & (x', v', \pi') & & & (x'', v'', \pi'') & \\
(x_1, v_1) \arrow[ur, "\mathsf{Fold}"] & & (x_2, v_2) \arrow[ul] & (x_3, v_3) \arrow[ur, "\mathsf{Fold}"] & & (x_4, v_4) \arrow[ul]
\end{tikzcd}
$$

Folding Scheme with Selective Verification

From 2-folding to 4-folding

Output of Fold is in $\mathcal{R}$ $\Rightarrow$ **Bootstrapping**



Folding Scheme with Selective Verification

From 2-folding to 4-folding

Output of Fold is in $\mathcal{R}$ $\Rightarrow$ **Bootstrapping**

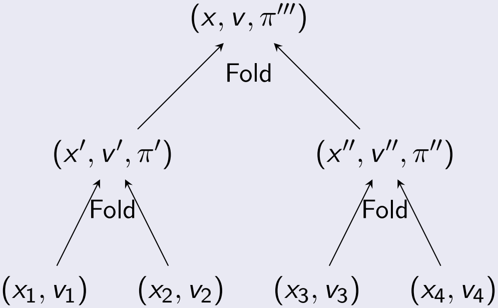
$$
\begin{tikzcd}
 & & & (x, v, \pi''') & & & \\
 & (x', v', \pi') \arrow[urr, "\text{Fold}"] & & & & (x'', v'', \pi'') \arrow[ull] & \\
(x_1, v_1) \arrow[ur, "\text{Fold}"] & & (x_2, v_2) \arrow[ul] & & (x_3, v_3) \arrow[ur, "\text{Fold}"] & & (x_4, v_4) \arrow[ul]
\end{tikzcd}
$$

$$\pi = (\pi', \pi'', \pi''')$$

Folding Scheme with Selective Verification

From 2-folding to 4-folding

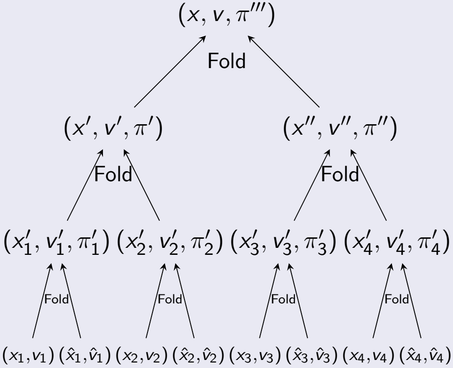
Output of Fold is in $\mathcal{R}$ $\Rightarrow$ **Bootstrapping**



$$\pi = (\pi', \pi'', \pi''')$$

From 2-folding to n-folding

Binary tree with n leaves



Folding Scheme **with Selective Verification** [RZ23]

But...

$|\pi|$ and number of x_i's needed scales linearly in n.

Folding Scheme **with Selective Verification** [RZ23]

But…

$|\pi|$ and number of x_i's needed scales linearly in n.

Idea

Generate n proofs π_i, each containing $O(\log n)$ folding proofs and statements.

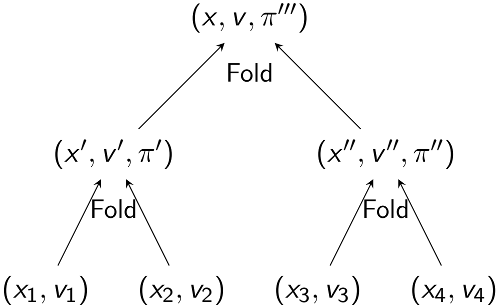
Folding Scheme **with Selective Verification** [RZ23]

But...

$|\pi|$ and number of x_i's needed scales linearly in n.

Idea

Generate n proofs π_i, each containing $O(\log n)$ folding proofs and statements.



Folding Scheme **with Selective Verification** [RZ23]

But...

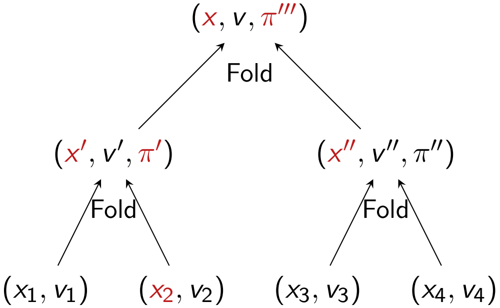
$|\pi|$ and number of x_i's needed scales linearly in n.

Idea

Generate n proofs π_i, each containing $O(\log n)$ folding proofs and statements.

Example

- $\pi_1 = \{x_2, x', \pi', x'', x, \pi'''\}$



Folding Scheme **with Selective Verification** [RZ23]

But...

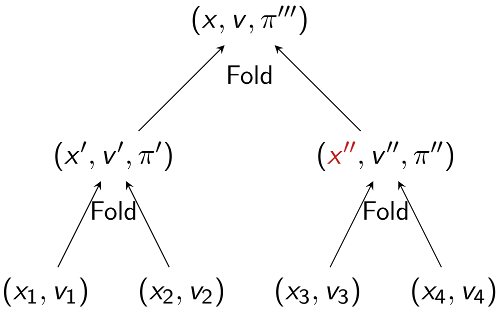
$|\pi|$ and number of x_i's needed scales linearly in n.

Idea

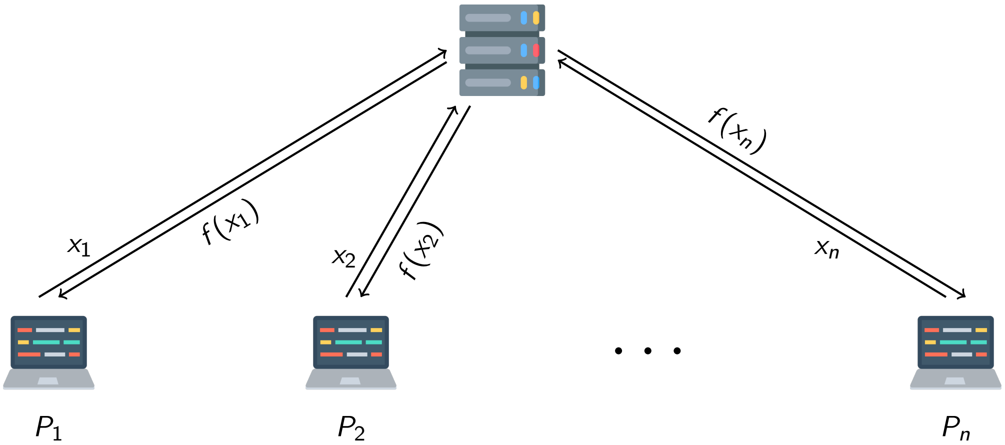
Generate n proofs π_i, each containing $O(\log n)$ folding proofs and statements.

Example

- $\pi_1 = \{x_2, x', \pi', x'', x, \pi'''\}$
- $\pi_2 = \{x_1, x', \pi', x'', x, \pi'''\}$
- $\pi_3 = \{x_4, x'', \pi'', x', x, \pi'''\}$
- $\pi_4 = \{x_3, x'', \pi'', x', x, \pi'''\}$



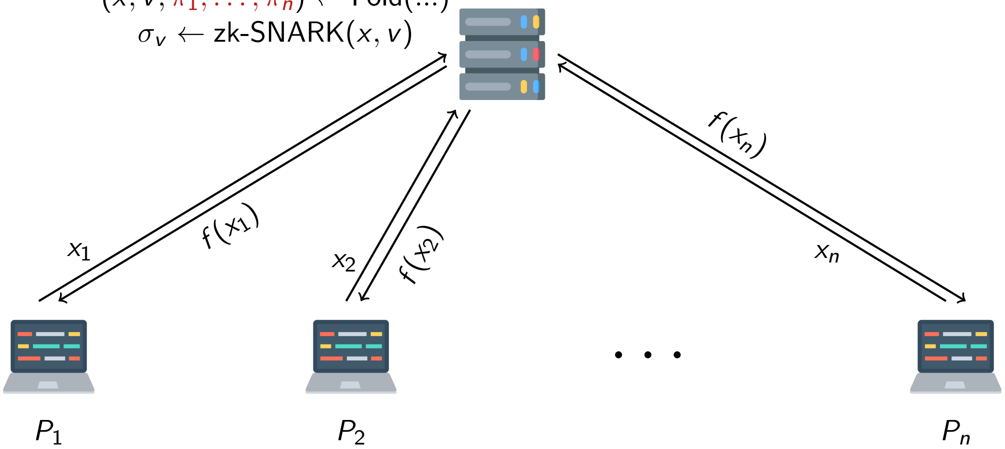
Motivating Example: Verifiable Computation as a Service



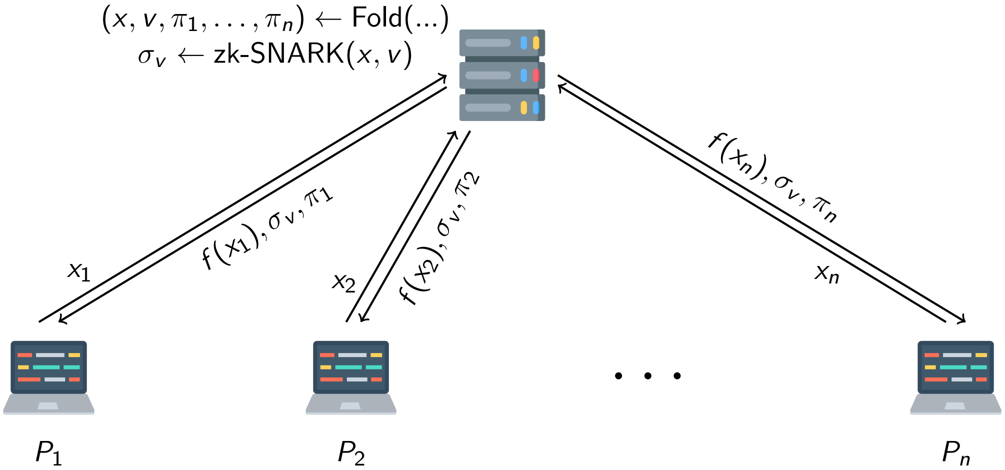
Motivating Example: Verifiable Computation as a Service

$(x, v, \pi_1, \ldots, \pi_n) \leftarrow \text{Fold}(\ldots)$

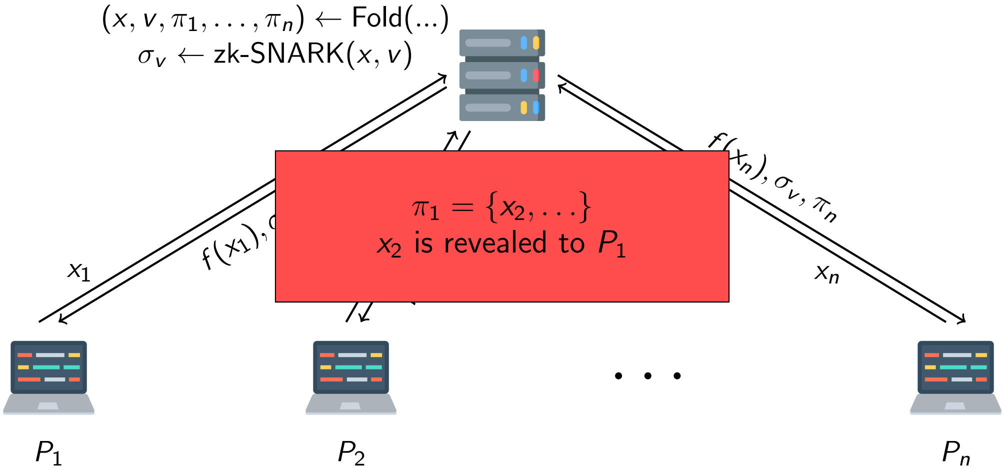
$\sigma_v \leftarrow \text{zk-SNARK}(x, v)$



Motivating Example: Verifiable Computation as a Service



Motivating Example: Verifiable Computation as a Service



$\pi_1 = \{x_2, \ldots\}$

x_2 is revealed to P_1

Privacy Preserving Folding Scheme [B**E**24]

Idea

Folding scheme hiding others' statements.

Privacy Preserving Folding Scheme [BE24]

Idea

Folding scheme hiding others' statements.

NP-statement hider

Hide each instance (x, v) as another instance (x', v') and generate certificate c that x' hides x. More on these later

Privacy Preserving Folding Scheme [BE24]

Idea

Folding scheme hiding others' statements.

NP-statement hider

Hide each instance (x, v) as another instance (x', v') and generate certificate c that x' hides x. More on these later

(x_1, v_1) (x_2, v_2) (x_3, v_3) (x_4, v_4)

Privacy Preserving Folding Scheme [BE24]

Idea

Folding scheme hiding others' statements.

NP-statement hider

Hide each instance (x, v) as another instance (x', v') and generate certificate c that x' hides x. More on these later

Example

- $\pi_1 = \{x'_1, c_1 \quad\quad\quad\quad\quad\quad \}$
- $\pi_2 = \{x'_2, c_2 \quad\quad\quad\quad\quad\quad \}$
- $\pi_3 = \{x'_3, c_3 \quad\quad\quad\quad\quad\quad \}$
- $\pi_4 = \{x'_4, c_4 \quad\quad\quad\quad\quad\quad \}$

$$
\begin{tikzcd}
(x'_1, v'_1, c_1) & (x'_2, v'_2, c_2) & (x'_3, v'_3, c_3) & (x'_4, v'_4, c_4) \\
(x_1, v_1) \arrow[u, "\text{Hide}"] & (x_2, v_2) \arrow[u, "\text{Hide}"] & (x_3, v_3) \arrow[u, "\text{Hide}"] & (x_4, v_4) \arrow[u, "\text{Hide}"]
\end{tikzcd}
$$

Privacy Preserving Folding Scheme [BE24]

Idea

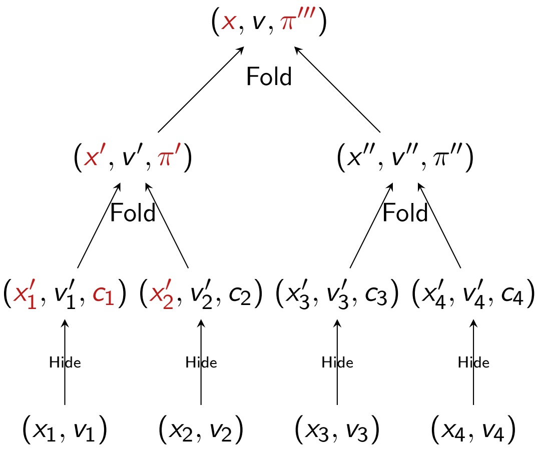
Folding scheme hiding others' statements.

NP-statement hider

Hide each instance (x, v) as another instance (x', v') and generate certificate c that x' hides x. More on these later

Example

- $\pi_1 = \{x_1', c_1, x_2', x', \pi', x, \pi'''\}$
- $\pi_2 = \{x_2', c_2, x_1', x', \pi', x, \pi'''\}$
- $\pi_3 = \{x_3', c_3, x_4', x'', \pi'', x, \pi'''\}$
- $\pi_4 = \{x_4', c_4, x_3', x'', \pi'', x, \pi'''\}$



Privacy Preserving Folding Scheme [B**E**24]

Security of Privacy Preserving FS

IND-CMA flavor:

Privacy Preserving Folding Scheme [BE24]

Security of Privacy Preserving FS

IND-CMA flavor:

1. Adv choose input with 2 options for entry j

$$(x_1, v_1) \quad \begin{matrix} ({x_2}^0, v_2^0) \\ ({x_2}^1, v_2^1) \end{matrix} \quad (x_3, v_3) \quad (x_4, v_4)$$

Privacy Preserving Folding Scheme [BE24]

Security of Privacy Preserving FS

IND-CMA flavor:

1. Adv choose input with 2 options for entry j
2. Entry j chosen at random

$$(x_1, v_1) \quad \begin{matrix} ({x_2}^0, v_2^0) \\ ({x_2}^1, v_2^1) \end{matrix} \quad (x_3, v_3) \quad (x_4, v_4)$$

$$b \leftarrow_\$ \{0, 1\}$$

Privacy Preserving Folding Scheme [B**E**24]

Security of Privacy Preserving FS

IND-CMA flavor:

1. Adv choose input with 2 options for entry j
2. Entry j chosen at random

$$(x_1, v_1) \quad (x_2^b, v_2^b) \quad (x_3, v_3) \quad (x_4, v_4)$$

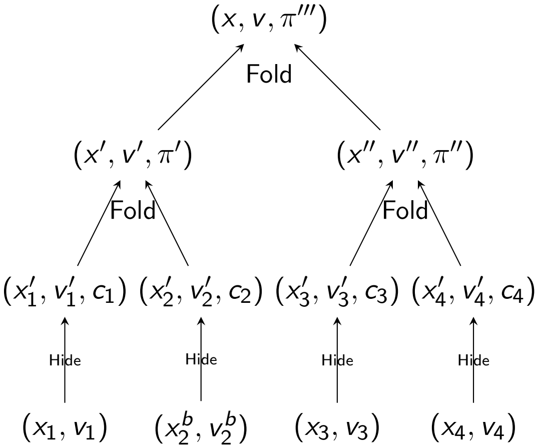
$$b \leftarrow_{\$} \{0, 1\}$$

Privacy Preserving Folding Scheme [BE24]

Security of Privacy Preserving FS

IND-CMA flavor:

1. Adv choose input with 2 options for entry j
2. Entry j chosen at random
3. Everything is hidden and folded



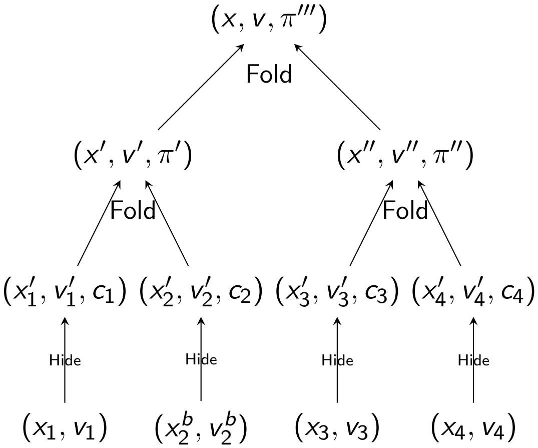
$$b \leftarrow_{\$} \{0, 1\}$$

Privacy Preserving Folding Scheme [BE24]

Security of Privacy Preserving FS

IND-CMA flavor:

1. Adv choose input with 2 options for entry j
2. Entry j chosen at random
3. Everything is hidden and folded
4. Adv chooses index ℓ and receives π_ℓ



$$\mathrm{Adv} \longleftarrow \pi_1$$

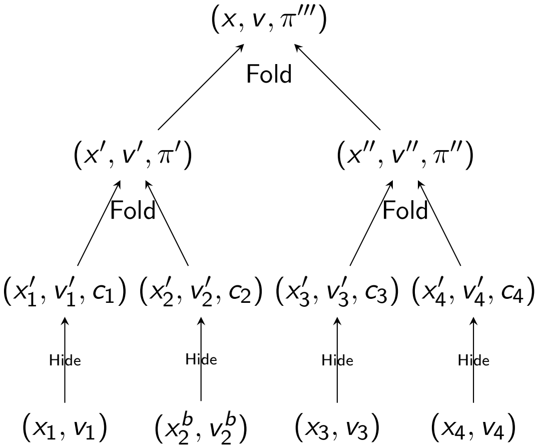
$$b \leftarrow_{\$} \{0, 1\}$$

Privacy Preserving Folding Scheme [BE24]

Security of Privacy Preserving FS

IND-CMA flavor:

1. Adv choose input with 2 options for entry j
2. Entry j chosen at random
3. Everything is hidden and folded
4. Adv chooses index ℓ and receives π_ℓ
5. Guess which (x_j, v_j) was used



$$b' \longleftarrow \text{Adv} \longleftarrow \pi_1$$

$$b \leftarrow_{\$} \{0, 1\}$$

Win if $b' = b$

Privacy Preserving Folding Scheme [BE24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

Privacy Preserving Folding Scheme [B**E**24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness

Privacy Preserving Folding Scheme [B**E**24]

NP-statement hider

$$(x, v) \longrightarrow \mathsf{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

Privacy Preserving Folding Scheme [BE24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

  (x_0, v_0)
  (x_1, v_1)

Privacy Preserving Folding Scheme [B**E**24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

$$(x_0, v_0)$$
$$(x_1, v_1)$$
$$b \leftarrow_{\$} \{0, 1\}$$

Privacy Preserving Folding Scheme [BE24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

$$(x_b, v_b)$$

$$b \leftarrow_{\$} \{0, 1\}$$

Privacy Preserving Folding Scheme [B**E**24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

$$(x_b, v_b) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

$$b \leftarrow_{\$} \{0, 1\}$$

Privacy Preserving Folding Scheme [B**E**24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

$$(x_b, v_b) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

$$b \leftarrow_{\$} \{0, 1\}$$

$$(x', v') \rightarrow \text{Adv}$$

Privacy Preserving Folding Scheme [BE24]

NP-statement hider

$$(x, v) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

$$(x_b, v_b) \longrightarrow \text{Hide} \longrightarrow (x', v', c)$$

$$b \leftarrow_{\$} \{0, 1\}$$

$$(x', v') \rightarrow \text{Adv} \longrightarrow b'$$

Privacy Preserving Folding Scheme [BE24]

NP-statement hider

$$(x, v) \longrightarrow \mathsf{Hide} \longrightarrow (x', v', c)$$

- Completeness
- Knowledge Soundness
- Hiding (IND-CMA)

$$(x_b, v_b) \longrightarrow \mathsf{Hide} \longrightarrow (x', v', c)$$

$$b \leftarrow_\$ \{0, 1\}$$

$$(x', v') \rightarrow \mathsf{Adv} \longrightarrow b'$$

Claim

Composing a Folding Scheme with an NP-statement hider gives a Privacy Preserving Folding Scheme.

NP-statement hider: Example

Folding with random instance

To hide (x, v):

1. Generate random instance (x_r, v_r).
2. Fold:
$$(x', v', \pi) \leftarrow \mathrm{Fold}((x, v), (x_r, v_r))$$
3. Output $(x', v', c = (\pi, x_r))$.

NP-statement hider: Example

Folding with random instance

To hide (x, v):

1. Generate random instance (x_r, v_r).
2. Fold:
$$(x', v', \pi) \leftarrow \mathrm{Fold}((x, v), (x_r, v_r))$$
3. Output $(x', v', c = (\pi, x_r))$.

Recall

$$\mathcal{L}_A = \{x \mid \exists v \colon Av = x\}$$

$\mathrm{Fold}((x_1, v_1), (x_2, v_2))$: $\rho \leftarrow_{\$} \mathbb{F}; \pi = \rho;$

$$x = x_1 + \rho x_2; \qquad v = v_1 + \rho v_2.$$

Example

NP-statement hider: Example

Folding with random instance

To hide (x, v):

1. Generate random instance (x_r, v_r).
2. Fold:
$$(x', v', \pi) \leftarrow \mathsf{Fold}((x, v), (x_r, v_r))$$
3. Output $(x', v', c = (\pi, x_r))$.

Recall

$$\mathcal{L}_A = \{x \mid \exists v \colon Av = x\}$$

$\mathsf{Fold}((x_1, v_1), (x_2, v_2))$: $\rho \leftarrow_\$ \mathbb{F}; \pi = \rho;$
$$x = x_1 + \rho x_2; \qquad v = v_1 + \rho v_2.$$

Example

1. Generate random instance in $\mathcal{R}$ as
$$v_r \leftarrow_\$ \mathbb{F}^m; \qquad x_r = Av_r.$$

NP-statement hider: Example

Folding with random instance

To hide (x, v):

1. Generate random instance (x_r, v_r).
2. Fold:
$$(x', v', \pi) \leftarrow \mathsf{Fold}((x, v), (x_r, v_r))$$
3. Output $(x', v', c = (\pi, x_r))$.

Recall

$$\mathcal{L}_A = \{x \mid \exists v \colon Av = x\}$$

$\mathsf{Fold}((x_1, v_1), (x_2, v_2))$: $\rho \leftarrow_\$ \mathbb{F}; \pi = \rho;$

$$x = x_1 + \rho x_2; \qquad v = v_1 + \rho v_2.$$

Example

1. Generate random instance in $\mathcal{R}$ as
$$v_r \leftarrow_\$ \mathbb{F}^m; \qquad x_r = Av_r.$$
2. Hide by folding
$\mathsf{Hide}((x, v), (x_r, v_r))$:
$$\rho \leftarrow_\$ \mathbb{F}$$
$$x' = x_1 + \rho x_r$$
$$v' = v_1 + \rho v_r.$$

NP-statement hider: Example

Folding with random instance

To hide (x, v):

1. Generate random instance (x_r, v_r).
2. Fold:
$$(x', v', \pi) \leftarrow \mathsf{Fold}((x, v), (x_r, v_r))$$
3. Output $(x', v', c = (\pi, x_r))$.

Recall

$$\mathcal{L}_A = \{x \mid \exists v\colon Av = x\}$$

$\mathsf{Fold}((x_1, v_1), (x_2, v_2))$: $\rho \leftarrow_\$ \mathbb{F}; \pi = \rho;$
$$x = x_1 + \rho x_2; \qquad v = v_1 + \rho v_2.$$

Example

1. Generate random instance in $\mathcal{R}$ as
$$v_r \leftarrow_\$ \mathbb{F}^m; \qquad x_r = Av_r.$$
2. Hide by folding
$\mathsf{Hide}((x, v), (x_r, v_r))$:
$$\rho \leftarrow_\$ \mathbb{F}$$
$$x' = x_1 + \rho x_r$$
$$v' = v_1 + \rho v_r.$$
3. Output $(x', v', c = (\rho, x_r))$.

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

Assume (x, v) hides (x_1, v_1) using (x_r, v_r):

$$x = x_1 + \rho x_r$$
$$v = v_1 + \rho v_r.$$

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

Assume (x, v) hides (x_1, v_1) using (x_r, v_r):

$$x = x_1 + \rho x_r$$
$$v = v_1 + \rho v_r.$$

(x, v) is equally likely to hide (x_2, v_2) if there is $(x'_r, v'_r) \in \mathcal{R}_A$ such that:

$$x_2 + \rho' x'_r = x_1 + \rho x_r$$
$$v_2 + \rho' v'_r = v_1 + \rho v_r.$$

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

Assume (x, v) hides (x_1, v_1) using (x_r, v_r):

$$x = x_1 + \rho x_r$$
$$v = v_1 + \rho v_r.$$

(x, v) is equally likely to hide (x_2, v_2) if there is $(x'_r, v'_r) \in \mathcal{R}_A$ such that:

$$x_2 + \rho' x'_r = x_1 + \rho x_r$$
$$v_2 + \rho' v'_r = v_1 + \rho v_r.$$

So we must have

$$x'_r = (\rho')^{-1}(x_1 + \rho x_r - x_2)$$
$$v'_r = (\rho')^{-1}(v_1 + \rho v_r - v_2)$$

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

Assume (x, v) hides (x_1, v_1) using (x_r, v_r):

$$x = x_1 + \rho x_r$$
$$v = v_1 + \rho v_r.$$

(x, v) is equally likely to hide (x_2, v_2) if there is $(x'_r, v'_r) \in \mathcal{R}_A$ such that:

$$x_2 + \rho' x'_r = x_1 + \rho x_r$$
$$v_2 + \rho' v'_r = v_1 + \rho v_r.$$

So we must have

$$x'_r = (\rho')^{-1}(x_1 + \rho x_r - x_2)$$
$$v'_r = (\rho')^{-1}(v_1 + \rho v_r - v_2)$$

But is this in $\mathcal{R}_A$?

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

Assume (x, v) hides (x_1, v_1) using (x_r, v_r):

$$x = x_1 + \rho x_r$$
$$v = v_1 + \rho v_r.$$

(x, v) is equally likely to hide (x_2, v_2) if there is $(x_r', v_r') \in \mathcal{R}_A$ such that:

$$x_2 + \rho' x_r' = x_1 + \rho x_r$$
$$v_2 + \rho' v_r' = v_1 + \rho v_r.$$

So we must have

$$x_r' = (\rho')^{-1}(x_1 + \rho x_r - x_2)$$
$$v_r' = (\rho')^{-1}(v_1 + \rho v_r - v_2)$$

But is this in $\mathcal{R}_A$?

$$\begin{aligned} Av_r' &= A(\rho')^{-1}(v_1 + \rho v_r - v_2) \\ &= (\rho')^{-1}(Av_1 + \rho Av_r - Av_2) \\ &= (\rho')^{-1}(x_1 \rho x_r - x_2) \\ &= x_r' \end{aligned}$$

NP-statement hider: Example

Example is secure

Can Adv distinguish if (x, v) hides (x_1, v_1) or (x_2, v_2)?

Assume (x, v) hides (x_1, v_1) using (x_r, v_r):

$$x = x_1 + \rho x_r$$
$$v = v_1 + \rho v_r.$$

Theorem

There is a privacy preserving folding scheme for $\mathcal{L}_A = \mathrm{Im}(A)$.

(x, v) is equally likely to hide (x_2, v_2) if there is $(x'_r, v'_r) \in \mathcal{R}_A$ such that:

$$x_2 + \rho' x'_r = x_1 + \rho x_r$$
$$v_2 + \rho' v'_r = v_1 + \rho v_r.$$

So we must have

$$x'_r = (\rho')^{-1}(x_1 + \rho x_r - x_2)$$
$$v'_r = (\rho')^{-1}(v_1 + \rho v_r - v_2)$$

But is this in $\mathcal{R}_A$?

$$\begin{aligned} A v'_r &= A(\rho')^{-1}(v_1 + \rho v_r - v_2) \\ &= (\rho')^{-1}(A v_1 + \rho A v_r - A v_2) \\ &= (\rho')^{-1}(x_1 \rho x_r - x_2) \\ &= x'_r \end{aligned}$$

In general...

NP-statement hider

If there is a folding scheme for $\mathcal{L}$, $\mathcal{R}$ supports efficient random sampling, and for any three instances $(x_1, v_1), (x_2, v_2), (x, v) \in \mathcal{R}$ there are equally many ways to fold (x_1, v_1) into (x, v) as there is to fold (x_2, v_2) into (x, v), then there is an NP-statement hider for $\mathcal{L}$.

In general...

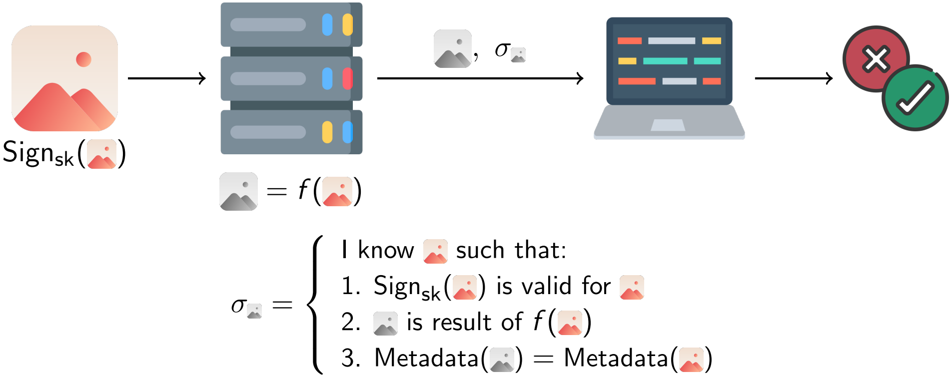
NP-statement hider

If there is a folding scheme for $\mathcal{L}$, $\mathcal{R}$ supports efficient random sampling, and for any three instances $(x_1, v_1), (x_2, v_2), (x, v) \in \mathcal{R}$ there are equally many ways to fold (x_1, v_1) into (x, v) as there is to fold (x_2, v_2) into (x, v), then there is an NP-statement hider for $\mathcal{L}$.

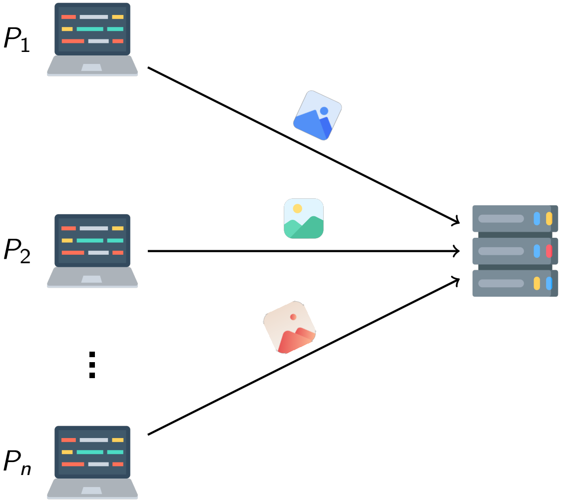
Privacy Preserving Folding Scheme

As above: There is a Privacy Preserving Folding Scheme for $\mathcal{L}$.

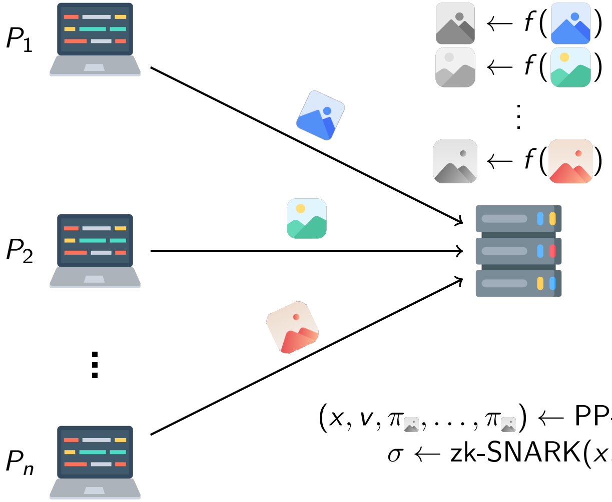
Returning to Images



Returning to Images

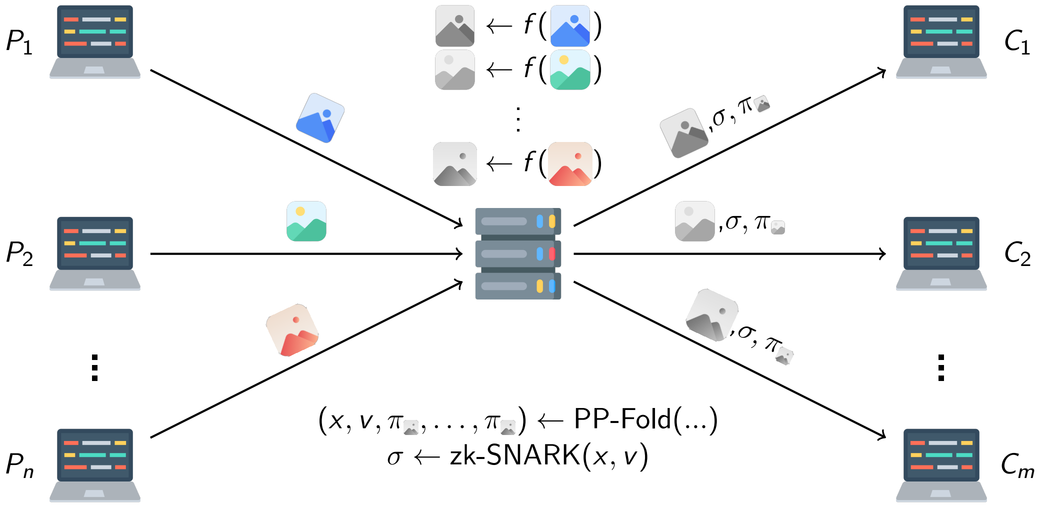


Returning to Images



$$(x, v, \pi, \ldots, \pi) \leftarrow \text{PP-Fold}(\ldots)$$
$$\sigma \leftarrow \text{zk-SNARK}(x, v)$$

Returning to Images



Thank you for listening.

Questions?

References I

[BE24] Joan Boyar and Simon Erfurth. "Folding Schemes with Privacy Preserving Selective Verification". In: *IACR Communications in Cryptology* 1.4 (2024). DOI: 10.62056/a0iv4fe-3.

[DB23] Trisha Datta and Dan Boneh. *Using ZK Proofs to Fight Disinformation*. https://medium.com/@boneh/using-zk-proofs-to-fight-disinformation-17e7d57fe52f. 2023. (Visited on 11/28/2023).

[DCB25] Trisha Datta, Binyi Chen, and Dan Boneh. "VerITAS: Verifying Image Transformations at Scale". In: *IEEE Symposium on Security and Privacy - S&P 2025*. IEEE Computer Society, 2025. DOI: 10.1109/SP61157.2025.00097.

References II

[DEH25] Stefan Dziembowski, Shahriar Ebrahimi, and Parisa Hassanizadeh. "VIMz: Verifiable Image Manipulation using Folding-based zkSNARKs". In: *Proc. Priv. Enhancing Technol.* 2025.2 (2025). URL: https://eprint.iacr.org/2024/1063.

[JWL11] Rob Johnson, Leif Walsh, and Michael Lamb. "Homomorphic Signatures for Digital Photographs". In: *Financial Cryptography and Data Security - FC 2011*. Vol. 7035. Lecture Notes in Computer Science. Springer, 2011, pp. 141–157. DOI: 10.1007/978-3-642-27576-0_12.

[KST22] Abhiram Kothapalli, Srinath T. V. Setty, and Ioanna Tzialla. "Nova: Recursive Zero-Knowledge Arguments from Folding Schemes". In: *Advances in Cryptology - CRYPTO 2022*. Vol. 13510. Lecture Notes in Computer Science. Springer, 2022, pp. 359–388. DOI: 10.1007/978-3-031-15985-5_13.

References III

[MVVZ25] Pierpaolo Della Monica, Ivan Visconti, Andrea Vitaletti, and Marco Zecchini. "Trust Nobody: Privacy-Preserving Proofs for Edited Photos with Your Laptop". In: *IEEE Symposium on Security and Privacy - S&P 2025*. IEEE Computer Society, 2025. DOI: 10.1109/SP61157.2025.00014.

[NT16] Assa Naveh and Eran Tromer. "PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations". In: *IEEE Symposium on Security and Privacy - S&P 2016*. IEEE Computer Society, 2016, pp. 255–271. DOI: 10.1109/SP.2016.23.

References IV

[PLZ+09] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, Marco Carli, and Federica Battisti. "TID2008 — A Database for Evaluation of Full-Reference Visual Quality Assessment Metrics". In: *Advances of Modern Radioelectronics* 10.4 (2009), pp. 30–45. URL: https://www.ponomarenko.info/papers/mre2009tid.pdf.

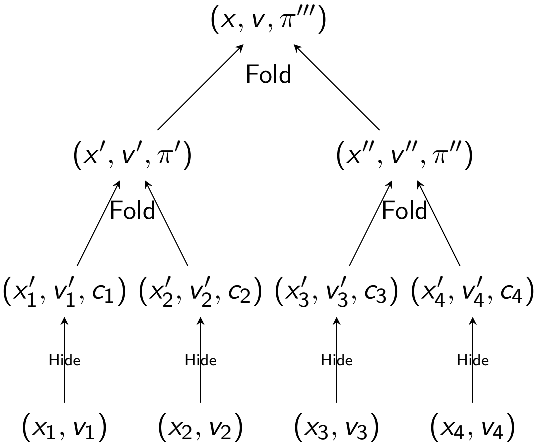
[RZ23] Carla Ràfols and Alexandros Zacharakis. "Folding Schemes with Selective Verification". In: *Progress in Cryptology - LATINCRYPT 2023*. Vol. 14168. Lecture Notes in Computer Science. Springer, 2023, pp. 229–248. DOI: 10.1007/978-3-031-44469-2_12.

Made using icons from flaticon.com.

Privacy Preserving Folding Scheme [B**E**24]

$$
\begin{tikzcd}
 & & & (x, v, \pi''') & & & \\
 & (x', v', \pi') \arrow[urr, "\mathrm{Fold}"] & & & & (x'', v'', \pi'') \arrow[ull] & \\
(x_1', v_1', c_1) \arrow[ur, "\mathrm{Fold}"] & & (x_2', v_2', c_2) \arrow[ul] & & (x_3', v_3', c_3) \arrow[ur, "\mathrm{Fold}"] & & (x_4', v_4', c_4) \arrow[ul] \\
(x_1, v_1) \arrow[u, "\mathrm{Hide}"] & & (x_2, v_2) \arrow[u, "\mathrm{Hide}"] & & (x_3, v_3) \arrow[u, "\mathrm{Hide}"] & & (x_4, v_4) \arrow[u, "\mathrm{Hide}"]
\end{tikzcd}
$$

Privacy Preserving Folding Scheme [BE24]



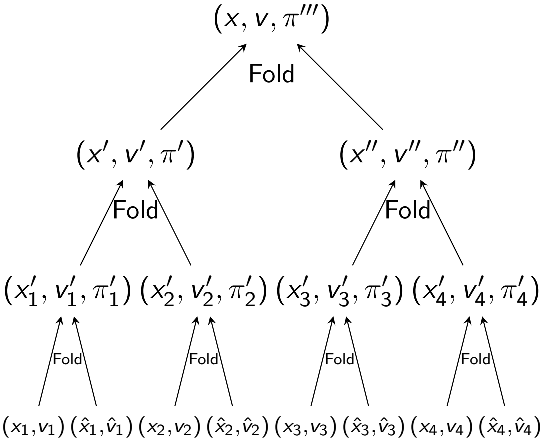


Image Signature Scheme Computation

	Computation Time	**Signature Size**

Image Signature Scheme Computation

	Computation Time	**Signature Size**
Key generation	Same as $\mathsf{KeyGen}^{\mathsf{DS}}$	—

Image Signature Scheme Computation

	Computation Time	**Signature Size**
Key generation	Same as $\mathsf{KeyGen}^{\mathsf{DS}}$	—
Signing	1025 hashes and time of $\mathsf{Sign}^{\mathsf{DS}}$	$\lvert S \rvert$

Image Signature Scheme Computation

	Computation Time	**Signature Size**
Key generation	Same as $\mathsf{KeyGen}^{\mathsf{DS}}$	—
Signing	1025 hashes and time of $\mathsf{Sign}^{\mathsf{DS}}$	$\lvert S \rvert$
Compression	1025 hashes	$128\lvert H \rvert + \lvert S \rvert$

Image Signature Scheme Computation

	Computation Time	**Signature Size**
Key generation	Same as $\text{KeyGen}^{\text{DS}}$	—
Signing	1025 hashes and time of Sign^{DS}	$\lvert S \rvert$
Compression	1025 hashes	$128\lvert H \rvert + \lvert S \rvert$
Verification	1025 hashes and time of $\text{Verify}^{\text{DS}}$	—

Image Signature Scheme Final Construction Step

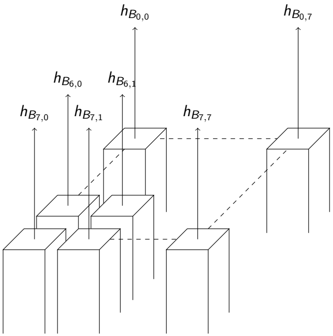


Image Signature Scheme Final Construction Step

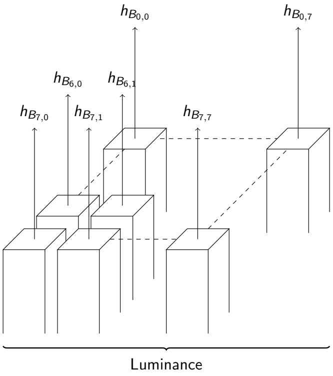
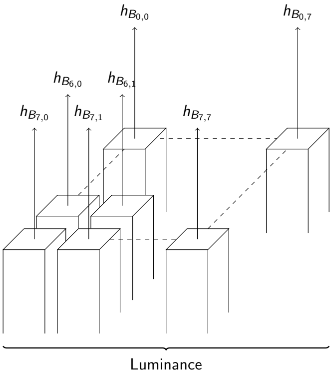


Image Signature Scheme Final Construction Step



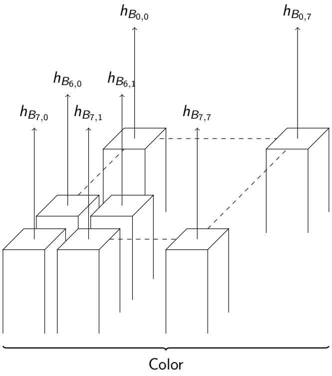


Image Signature Scheme Final Construction Step

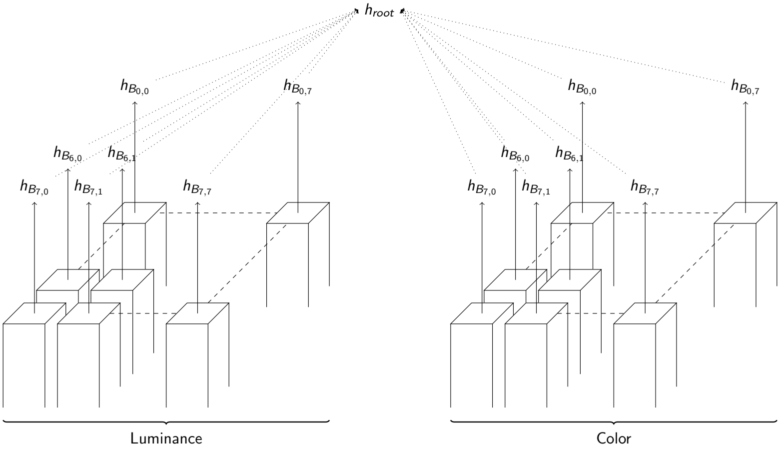
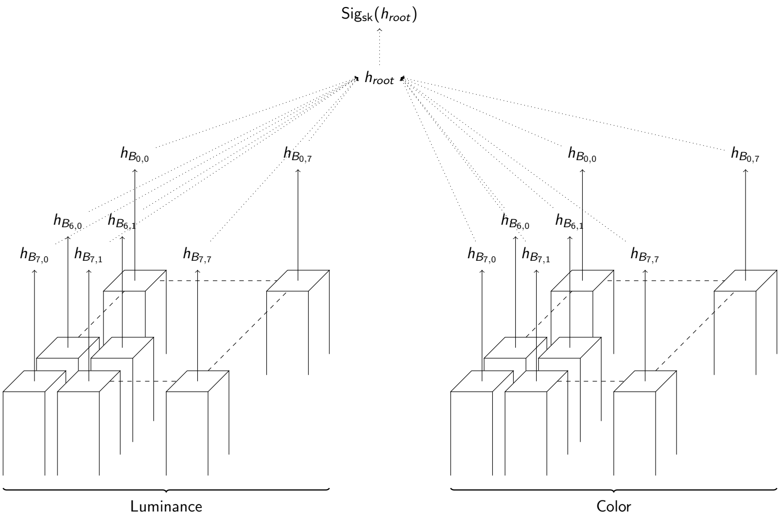


Image Signature Scheme Final Construction Step



Proof of Security

Proof of Security

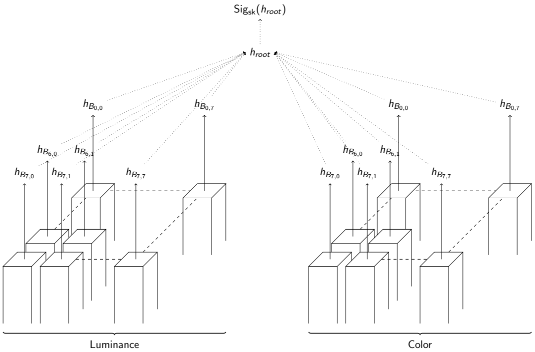
If $\forall k:$ i^*'s h_{root} is different from i_k's h_{root}: Forgery against DS.

Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.

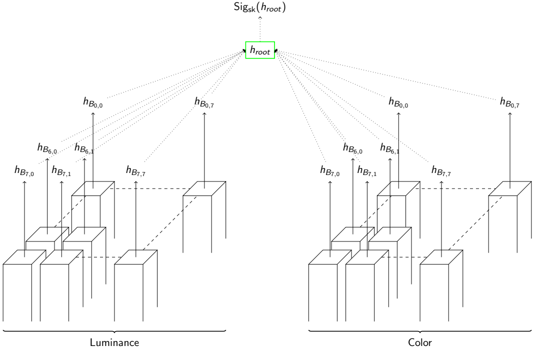
Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.



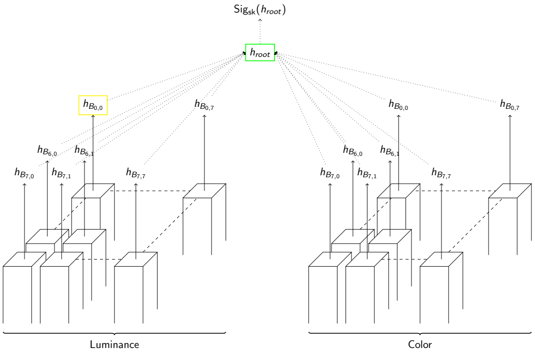
Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.



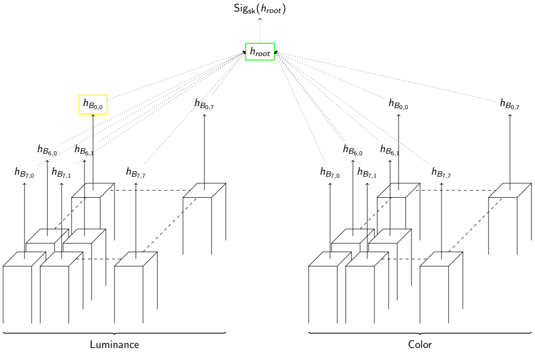
Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.



Proof of Security

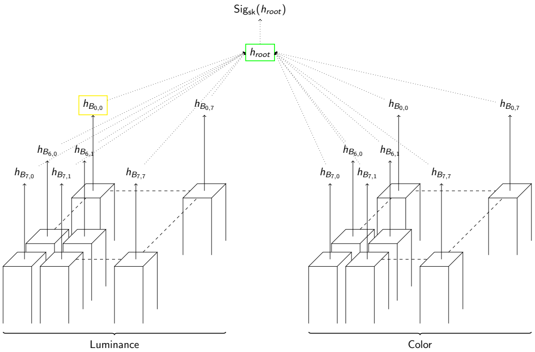
Else let k be such that i^* and i_k have the same h_{root}.



If $h_{B_{0,0}}$ is different for i^* and i^k: Collision for H:

$$H(\cdots, h^*_{B_{0,0}}, \cdots) = h_{root} = H(\cdots, h^k_{B_{0,0}}, \cdots).$$

Else let k be such that i^* and i_k have the same h_{root}.



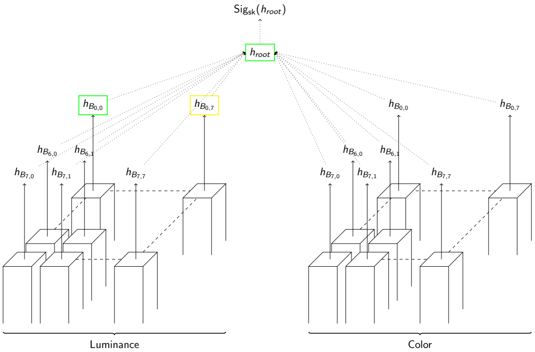
If $h_{B_{0,0}}$ is different for i^* and i^k: Collision for H:

$$H(\cdots, h^*_{B_{0,0}}, \cdots) = h_{root} = H(\cdots, h^k_{B_{0,0}}, \cdots).$$

Else move on.

Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.



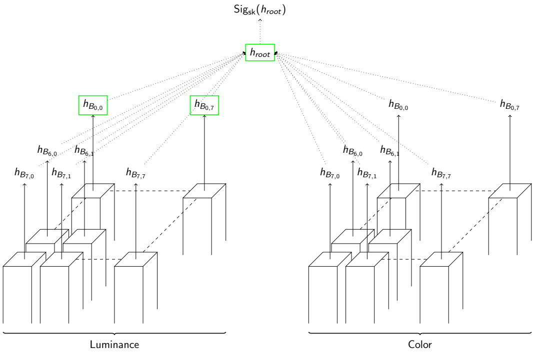
If $h_{B_{0,0}}$ is different for i^* and i^k: Collision for H:

$$H(\cdots, h^*_{B_{0,0}}, \cdots) = h_{root} = H(\cdots, h^k_{B_{0,0}}, \cdots).$$

Else move on.

Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.



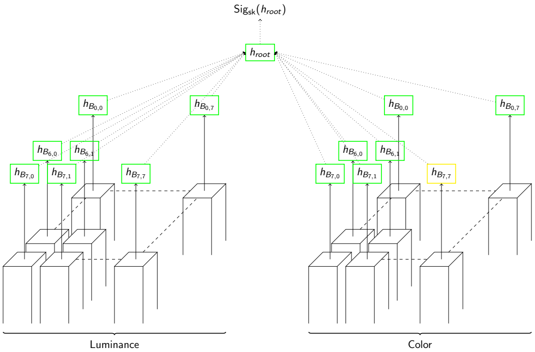
If $h_{B_{0,0}}$ is different for i^* and i^k: Collision for H:

$$H(\cdots, h^*_{B_{0,0}}, \cdots) = h_{root} = H(\cdots, h^k_{B_{0,0}}, \cdots).$$

Else move on.

Proof of Security

Else let k be such that i^* and i_k have the same h_{root}.



If $h_{B_{0,0}}$ is different for i^* and i^k: Collision for H:

$$H(\cdots, h^*_{B_{0,0}}, \cdots) = h_{root} = H(\cdots, h^k_{B_{0,0}}, \cdots).$$

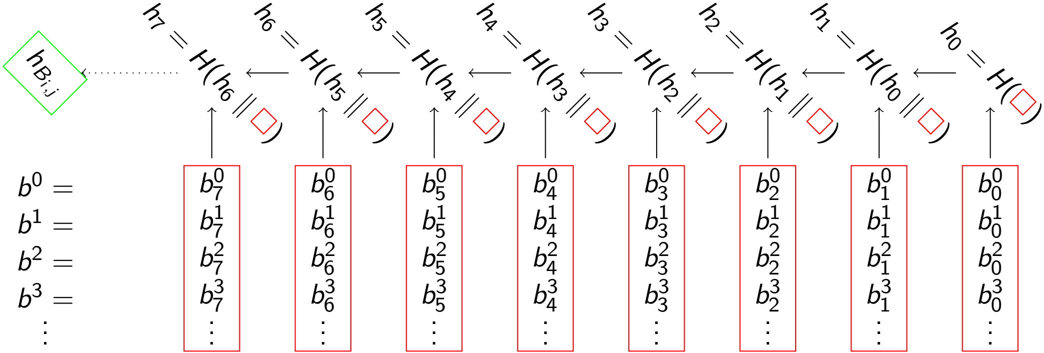
Else move on.

Proof of Security

Either we have found $h^*_{B_{i,j}} \neq h^k_{B_{i,j}}$ and a collision to H.

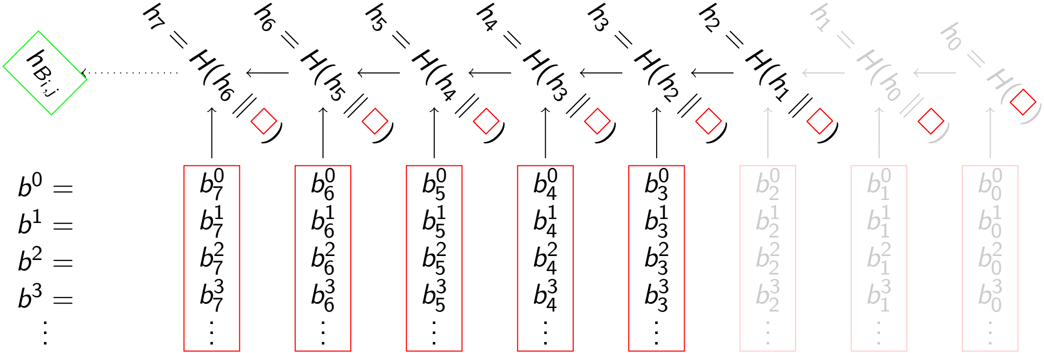
Proof of Security

Or we go one level deeper for each $h_{B_{i,j}}$.



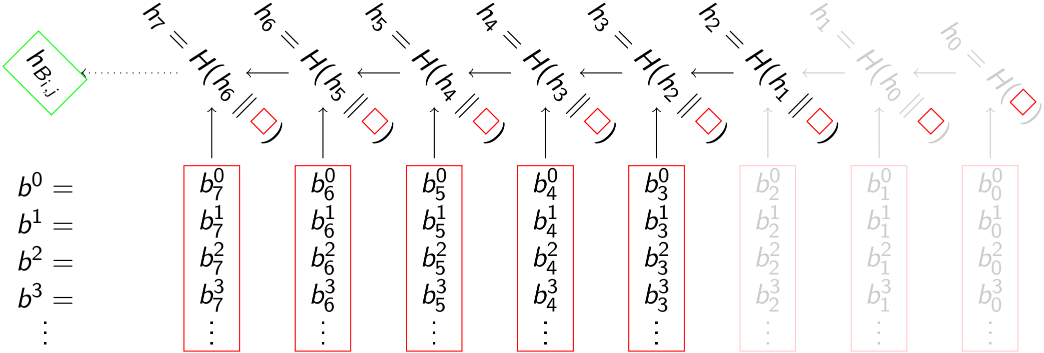
Proof of Security

Or we go one level deeper for each $h_{B_{i,j}}$.



Proof of Security

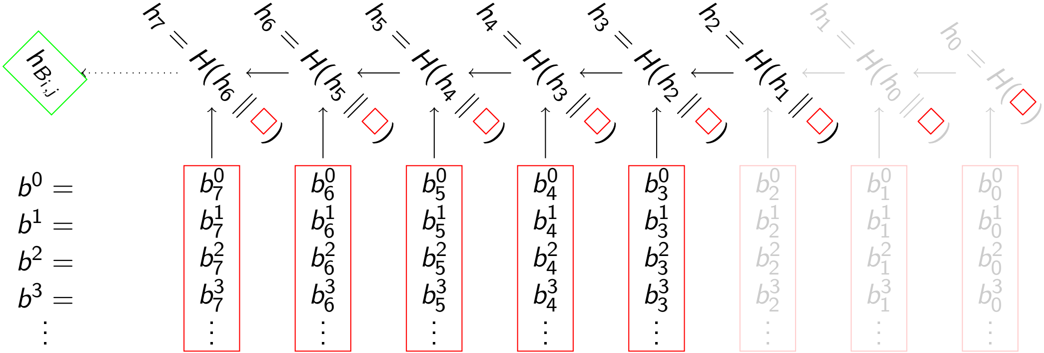
Or we go one level deeper for each $h_{B_{i,j}}$.



Since $i^* \notin \mathrm{CSpan}(i_k, s_k)$ we are guaranteed tho eventually find a difference

Proof of Security

Or we go one level deeper for each $h_{B_{i,j}}$.



Since $i^* \notin \text{CSpan}(i_k, s_k)$ we are guaranteed tho eventually find a difference, and hence a collision for H. $\square$