zkIPV: Zero-Knowledge Proofs for Image Provenance Verification

Master's Thesis

Bibek Ghosh

July 11, 2025

Advisors: Prof. Debrup Chakraborty, Capt. Manish Khanna

Cryptology and Security Research Unit (CSRU)
Indian Statistical Institute, Kolkata

Declaration

I, Bibek Ghosh (Roll No: CrS2304), hereby declare that, this report entitled "zkIPV: Zero-Knowledge Proofs for Image Provenance Verification" submitted to Indian Statistical Institute, Kolkata towards the fulfilment of the requirements for the degree of Master of Technology in CSRU, is an original work carried out by me under the supervision of Captain Ritesh Wahi, Lt. Cdr. Keval Krishan and Prof. Debrup Chakraborty and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.

Bibek Ghosh
Kolkata - 700 108
July 2024

Acknowledgements

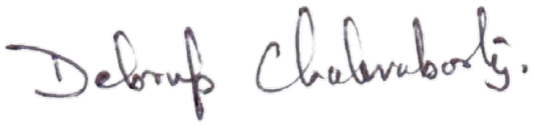
I would like to express my deepest gratitude to my supervisors Prof. Debrup Chakraborty of the Cryptology and Security Research Unit, Indian Statistical Institute, Kolkata and Capt. Ritesh Wahi from WESEE, Indian Navy, for their invaluable guidance, continuous support, and encouragement throughout the course of this project. Their insights and mentorship have been instrumental in shaping the direction and quality of this work.

I am sincerely thankful to my friends for their unwavering support, thoughtful feedback, and motivation at every stage of this journey.

Finally, I am grateful to the Indian Statistical Institute, Kolkata, for providing the essential resources, facilities, and academic environment that enabled me to carry out this work to the best of my abilities.

Certificate

This is to certify that the work contained in this project report entitled **"zkIPV: Zero-Knowledge Proofs for Image Provenance Verification"** submitted by **Bibek Ghosh (Roll No. CrS2304)** to the **Indian Statistical Institute, Kolkata** towards the fulfilment of the requirements for the degree of **Master of Technology in CRS** has been carried out by him under my supervision and that it has not been submitted elsewhere for the award of any degree.



Prof. Debrup Chakraborty
Associate Professor
Cryptology and Security Research Unit
Indian Statistical Institute
Kolkata-700108, INDIA.

Abstract

Recent advances in generative AI have significantly improved the ability to create photorealistic synthetic images, including so-called deepfakes, raising concerns about misinformation and the erosion of trust in digital media. Ensuring the integrity and authenticity of images, especially in sensitive domains like journalism, is thus increasingly critical. Existing solutions such as the C2PA (Content Provenance and Authenticity) framework provide origin verification through camera-generated digital signatures, but fail to account for image modifications that invalidate these signatures. To address this limitation, we propose a zero-knowledge approach to verifiable image editing that preserves both integrity and privacy.

This system, ZK-IPV, introduces a practical framework for transforming high-resolution images using zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs). ZK-IPV enables developers to specify permissible image transformations, which are then automatically compiled into zk-SNARK circuits. These circuits verify that edits conform to approved operations while concealing the original image content. Furthermore, ZK-IPV supports composable transformations and efficient hashing within proofs, enabling scalable verification pipelines even on commodity hardware.

We also formalize the protocol in which an editor can prove that a publicly shared image is derived from a signed original through an authorized transformation, without revealing the original image. This is achieved by demonstrating knowledge of a valid signature and original image such that the verified transformation results in the shared output. Our approach thus extends the C2PA framework with privacy-preserving guarantees and post-edit authentication, contributing to the broader goal of trustworthy digital content verification.

Keywords: Deepfakes, C2PA, zk-SNARKs, Recursive proof, Verifiable image editing, Composable transformations

Contents

Chapter 1

Introduction

1.1 Motivation

The widespread availability of generative AI tools has significantly complicated efforts to verify the authenticity and origin of digital media. With the capability to generate highly realistic synthetic content now accessible to everyday users through consumer-grade GPUs and open-source software such as DeepFaceLab'[3], discerning real from fake has become increasingly difficult. Moreover, commercial AI models like Midjourney [4], DALL·E [2], and Stability AI [9] are available via low-cost subscriptions, enabling the easy creation of synthetic images that can convincingly mimic authentic media.

This growing ease of content manipulation has led to real-world controversies. A well-known example occurred in 2023 [28], when an artist withdrew from the Sony World Photography Award after revealing that their submission was entirely AI-generated. Such incidents underscore the urgency of developing reliable mechanisms for verifying the authenticity and provenance of digital content.

In response, major tech companies and research institutions have ramped up their efforts to detect deepfakes and other AI-generated media [37, 40, 47]. Many of these initiatives focus on training models to identify visual inconsistencies or forensic traces typical of synthetic generation. Others explore embedding invisible digital watermarks to signal an image's origin [49, 30, 45]. However, even the most advanced deepfake detection systems today demonstrate a performance gap when compared to the rapidly evolving generative models, making them unreliable for high-stakes or real-world deployment.

An alternative approach involves embedding cryptographically signed metadata—such as capture time, location, and device information—directly into digital images. The Coalition for Content Provenance and Authenticity (C2PA) [7] has spearheaded this strategy, proposing a standardized frame-

work for tracking and verifying the entire lifecycle of an image. Under the C2PA protocol, a camera can sign captured media at the point of creation, and any subsequent edits performed through approved software are also logged and re-signed, forming a verifiable chain of custody. Tools such as Truepic [10] and applications like Adobe Photoshop have begun integrating these features, and camera manufacturers like Leica [36] and Sony [48] are developing devices with secure signing capabilities built in, often utilizing Trusted Execution Environments (TEEs) [31] to protect private keys.

However, this reliance on trusted software and hardware introduces new vulnerabilities [39, 41, 42, 43, 46]. If the editing software or TEE is compromised, so too is the integrity of the media it processes. Furthermore, the privacy of the content creator may be at risk, as current implementations often require disclosure of sensitive metadata to third parties. Another complication arises from the need to re-sign images after even minor, permissible edits—such as cropping or resizing—which invalidates the original signature.The Associated Press published a list of acceptable edits [12] that do not fundamentally alter the content of the photo.

To address these limitations, recent work has explored cryptographic techniques such as Succinct Non-Interactive Arguments of Knowledge (SNARKs) [33, 44, 23]. These allow for the creation of short, publicly verifiable proofs that certain transformations have been applied to an image without exposing the original data or requiring trust in any intermediary party. Unlike the C2PA model, SNARK-based approaches eliminate the need for trusted software during the verification process.

In this thesis, we propose the concept of *glass-to-glass* security, which aims to ensure the authenticity of digital media from the moment it is captured (via the camera lens) to its final display on a user's screen. This model seeks to provide strong guarantees of content integrity without relying on external software, human verifiers, or publishers, enabling end-users to independently assess whether (i) an image was originally captured by a compliant device, (ii) only allowable modifications were made, and (iii) critical metadata remains intact.

1.2 Objectives and Contributions

In this thesis, we introduce **zkIPV** (*ero-Knowledge Proofs for Image Provenance Verification*), a system that leverages succinct zero-knowledge arguments (zk-SNARKs) [17] to prove that an image has undergone authorized modifications while preserving its authenticity. By employing zero-knowledge proofs, zkIPV enables public verification of image provenance without revealing sensitive information removed during edits. This approach ad-

dresses key shortcomings in current signature-based systems like C2PA, particularly the need to trust intermediaries or software in the editing pipeline.

Zero-knowledge proofs allow a verifier to confirm that an edited image originates from a signed original and that only allowable transformations—such as cropping or blurring—were applied. These proofs remain succinct and efficient, enabling integration with browsers or news readers for automatic verification.

Previous work, including PhotoProof [44] and later optimization efforts, demonstrated the theoretical feasibility of such proof systems, but were limited in scale. Their systems operated on images far smaller than typical photos from modern high-resolution cameras, which commonly exceed 30 megapixels (MP). zkIPV addresses this gap by producing zk-SNARKs for full-resolution images, achieving practical performance on data sizes upwards of 90MB.

The core functionality of zkIPV is defined by a protocol between a *prover* (e.g., an editor) and a *verifier* (e.g., an end user). Given a publicly available image x, and a known editing function f, the prover demonstrates that there exists an original image w and a digital signature σ such that:

$$f(w) = x \quad \text{and} \quad \text{SigVerify}(vk, w, \sigma) = 1$$

where vk is the public verification key corresponding to the signer.

A major technical hurdle in this process is constructing zk-SNARK circuits capable of verifying the digital signature on a large image file. In traditional schemes, proving the correctness of hashing such a large input—particularly using standard cryptographic hashes like SHA-256—is computationally prohibitive in zk circuits due to their complex non-linear operations.

To tackle this, zkIPV introduces two signing modes based on the computational capability of the signer:

Mode 1: Lightweight Signing for Resource-Constrained Devices

For environments such as digital cameras with limited compute power, we use a lattice-based collision-resistant hash function to compress the image into a digest. This intermediate result is then further reduced with a SNARK-friendly hash like Poseidon before being signed. This layered approach enables efficient proof generation within zk-SNARKs, even for large image data, as lattice-based operations are significantly more SNARK-compatible than traditional cryptographic hash functions.

Mode 2: Optimized Proving for High-Power Signers

For cases where the signer is a powerful entity (e.g., a cloud provider or an AI company), we utilize polynomial commitments over the image data, which are then signed. This allows us to restructure the proof so that the zk-SNARK only verifies the transformation function without needing to re-verify the signature inside the circuit. This significantly reduces the proving overhead for editors.

Implementation and Performance

We implemented zkIPV in two main parts: the signature proof system and the editing proof logic using the Plonky2 framework. Furthermore, when leveraging the powerful signer mode, editors can prove image edits in under five minutes, making the system feasible for real-world newsrooms.

While our work focuses on photo editing, the underlying cryptographic techniques are more broadly applicable to any scenario requiring verifiable transformation of signed data—such as authenticated medical or financial records. Alternative designs, such as redactable or homomorphic signatures, offer potential benefits but lack the generality or practicality needed for complex edits like resizing, contrast adjustment, or selective blurring. zkIPV, by contrast, supports these transformations while preserving user privacy and verification integrity.

Alternative Approaches

An alternate path to enabling verifiable edits involves redactable or homomorphic signatures [32, 25, 20, 29]. These allow transformations such that a signature on the original data can be converted into a signature on the edited version. While redactable signatures are efficient for basic transformations like cropping, they cannot accommodate complex edits such as blurring, resizing, or tone adjustments. zkIPV supports such edits—for example, bilinear interpolation used for resizing in Adobe Photoshop [11]—which go beyond the capabilities of redactable or homomorphic signatures. Moreover, some transformations require information not known at signing time (e.g., resize dimensions), making pre-signing impractical. For these reasons, zkIPV offers a more general and practical solution.

1.3 Organizations

In Chapter 2, we present the necessary background to understand our work, focusing particularly on cryptographic image provenance techniques, the C2PA standard, and succinct zero-knowledge proofs (zk-SNARKs). In Chapter 3, we present a unified view of zero-knowledge proof (ZKP) systems and

their use in secure image provenance. We survey general-purpose ZKPs such as Groth16, Halo2, and Plonky2, comparing them by proof size, proving time, verification time, and scalability. Our analysis identifies trade-offs relevant to high-resolution image workflows. We then outline the core components needed to construct a ZKP system for verifying authorized image transformations. These include the cryptographic assumptions, witness structure (original image and digital signature), and circuit designs for transformations like cropping, blurring, and resizing. We also evaluate the computational costs of encoding these operations in zk-SNARK circuits. Finally, we describe our prototype implementation, supporting two modes: one for constrained signers (e.g., cameras) and another for powerful platforms (e.g., OpenAI). It features a custom hashing scheme for large images and integrates signature verification with transformation logic. We walk through the full workflow—from image capture to proof generation and verification—demonstrating the system's practicality and scalability.

In Chapter 4, we discuss complete implementation considerations, including memory bottlenecks, usability trade-offs, and limitations of our current approach. We also explore how our design generalizes to other signed data domains, such as verifiable health or financial records.

Chapter 5 reviews related work, covering previous efforts in zero-knowledge photo editing, provenance standards. We highlight where our contributions fit within and extend this landscape.

Finally, in Chapter 6, we summarize our main contributions, reflect on the practical feasibility of zk-SNARKs for image provenance at scale, and propose directions for future research, including optimizations, broader deployment strategies, and support for additional transformation types.

Chapter 2

Background

In this chapter, we introduce the theoretical background of our work with a particular focus on the zero-knowledge proofs.

Terminology used here

Notation	Description
α, β	Pixel matrices of the original and transformed images, respectively.
$\alpha^{\{R\|G\|B\}}$	Red, Green, and Blue color planes of image α.
α_i, β_i	i-th row of images α and β.
$\alpha_{i,j}$	Pixel value at row i, column j of image α.
H	Poseidon hash function, $H : \mathbb{Z}_p^2 \to \mathbb{Z}_p$.
H_σ	Hash value of an entire image row with n pixels: $H_\sigma : \mathbb{Z}_p^n \to \mathbb{Z}_p$
f_T	Transformation function: $\beta \leftarrow f_T(\alpha, U_{\text{in}})$.
D_1, D_2	Kernel matrices used in convolution operations.
$\mathcal{O}_p$	Merkle proof checker $\mathcal{O}_p : \mathbb{Z}_q \to \{0,1\}$, takes as input a Merkle path, root, and additional values to open a commitment at a leaf.
`Vsig`	Signature verification function: $\texttt{Vsig} : \mathbb{Z}_q^3 \to \{0,1\}$, which verifies a signature against a message and public key.

2.1 Digital Signatures

A digital signature scheme $\mathcal{S}$ is a triple of efficient algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ such that:

- $\mathsf{KGen}(1^\lambda) \to (sk, vk)$, where sk is the secret signing key and vk is the public verification key.
- $\mathsf{Sign}(sk, m) \to \sigma$, where σ is a signature on message m.
- $\mathsf{Vf}(vk, m, \sigma) \to 0/1$, where 0 implies rejection and 1 implies acceptance.

We say that a signature scheme is secure if it is existentially unforgeable under a chosen message attack [21]. Digital signatures in practice are implemented as a two step process: first hash the data using a collision-resistant hash and then sign the hash.

For a signature scheme SIG to be existentially unforgeable under a chosen message attack, every efficient adversary $\mathcal{A}$ with access to the verification key vk, and a signing oracle that for messages of its choice, should not be able to produce a signature on a new message m^* for which $\mathcal{A}$ has not queried the signing oracle. The advantage of the adversary $\mathcal{A}$ in the corresponding security game with security parameter λ is $\mathbf{Adv}^{\text{euf}}_{\text{SIG}}(\lambda)$.

2.2 Commitment Schemes

A commitment scheme enables a party to commit to a value $x \in \mathcal{X}$ by producing a commitment string com. The commitment should be hiding and binding. More precisely, a commitment scheme $\mathcal{C} = (setup, commit)$ is a pair of PPT algorithms:

- $setup(1^\lambda) \to pp$, where pp are public parameters for the scheme
- $commit(pp, x, r) \to com$, where com is a commitment to a message $x \in \mathcal{X}$ with randomness $r \in \mathcal{R}_C$

To open the commitment com, the committer reveals x and r and the verifier accepts if $commit(x, r) = com$. In some cases the setup algorithm is trivial in which case we say that the commitment scheme is just the algorithm $commit(x, r) \to com$.

Polynomial commitments. A polynomial commitment scheme [34] lets a prover commit to a polynomial $f \in \mathbb{F}[X]$ of bounded degree d. Additionally, the committer can provide an evaluation proof for the committed polynomial at any point $x \in \mathbb{F}$. More precisely, a polynomial commitment scheme $\mathcal{C}$ is a tuple of four efficient algorithms $\mathcal{C} = (setup, commit, open, \mathsf{Vf})$ such that:

- $setup(1^\lambda, d) \to pp$, where pp are public parameters to commit to a polynomial of degree at most d.
- $commit(pp, f, r) \to com$, where com is a commitment to a polynomial $f \in \mathbb{F}[X]$ of degree at most d using randomness $r \in \mathcal{R}_C$.

- $open(pp, f, x, r) \rightarrow (\pi, y)$, where π is an opening proof that proves that $f(x) = y$.
- $\mathsf{Vf}(pp, com, x, y, \pi) \rightarrow 0/1$, where 0 implies rejection and 1 implies acceptance.

A polynomial commitment scheme must be correct, evaluation binding, and optionally hiding. We defer these definitions to Appendix A.2. The KZG polynomial commitment scheme [34] is built from pairings. Another polynomial commitment scheme is built from the Fast Reed Solomon IOP of Proximity (FRI IOPP) protocol [14] using a collision resistant hash function.

We define the hiding and binding security property for commitment schemes below:

- **Hiding:** for every PPT adversary $\mathcal{A}$, there exists a negligible function $\nu(\cdot)$:

$$\Pr\left[b' \neq b : \begin{array}{l} \mathrm{pp} \leftarrow \mathrm{setup}(1^\lambda) \\ (x_0, x_1) \leftarrow \mathcal{A}(\mathrm{pp}) \\ b \leftarrow \{0,1\} \\ r \leftarrow \mathcal{R}_c \\ c \leftarrow \mathrm{commit}(\mathrm{pp}, x_b, r) \\ b' \leftarrow \mathcal{A}(c) \end{array}\right] \leq \frac{1}{2} + \nu(\lambda) \tag{2.1}$$

- **Binding:** for every PPT adversary $\mathcal{A}$ the following probability is negligible

$$\mathbf{Adv}^{\mathrm{bind}}_{\mathcal{A}}(\lambda) = \Pr[\mathrm{commit}(\mathrm{pp}, x, r) = \mathrm{commit}(\mathrm{pp}, x', r')] \tag{2.2}$$

where $\mathrm{pp} \leftarrow \mathrm{setup}(1^\lambda)$ and $(x, x', r, r') \leftarrow \mathcal{A}(\mathrm{pp})$.

We define correctness, evaluation binding, and hiding for a polynomial commitment scheme below.

- **Correctness:** for all $\lambda, d \in \mathbb{N}$, all $f \in \mathbb{F}[X]$ of degree at most d, the following probability is 1:

$$\Pr\left[\mathrm{Vf}(\mathrm{pp}, \mathrm{com}, x, y, \pi) = 1 : \begin{array}{l} \mathrm{pp} \leftarrow \mathrm{setup}(1^\lambda, d) \\ r \leftarrow \mathcal{R}_c \\ \mathrm{com} \leftarrow \mathrm{commit}(\mathrm{pp}, f, r) \\ (x, y) \leftarrow \mathrm{open}(\mathrm{pp}, f, x, \pi) \end{array}\right] \tag{2.3}$$

- **Evaluation Binding:** it is not possible to open a committed polynomial to two different values at one point. That is, for every PPT adversary $\mathcal{A}$ and for all $\lambda, d \in \mathbb{N}$, the following function is negligible

$$\Pr\left[\begin{array}{ll} \mathrm{Vf}(\mathrm{pp}, \mathrm{com}, x, y, \pi) = 1 & \mathrm{pp} \leftarrow \mathrm{setup}(1^\lambda, d) \\ \mathrm{Vf}(\mathrm{pp}, \mathrm{com}, x, y', \pi') = 1 \ : & (\mathrm{com}, x, y, \pi, \\ y \neq y' & y', \pi') \leftarrow \mathcal{A}(\mathrm{pp}, d) \end{array}\right] \tag{2.4}$$

2.3 SNARKs

A succinct non-interactive argument (SNARG) is a triple of algorithms (G, P, V) that works as follows. The generator G, on input the security parameter k and a time bound B, samples a reference string σ and a corresponding verification state τ. The honest prover $P(\sigma, y, w)$ produces a proof π for the statement $y = (M, x, t)$ given a valid w, provided that $t \leq B$; then $V(\tau, y, \pi)$ deterministically verifies π.

The SNARG is *adaptive* if the prover may choose the statement after seeing σ; otherwise, it is *non-adaptive*. The SNARG is *fully-succinct* if G runs "fast", otherwise it is of the preprocessing kind.

Definition 4.1

A triple of algorithms (G, P, V), where G is probabilistic and V is deterministic, is a SNARG for the relation $\mathcal{R}_U$ if the following conditions are satisfied:

1. **Completeness**: For every large enough security parameter $k \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance-witness pair $(y, w) = ((M, x, t), w) \in \mathcal{R}_U$ with $t \leq B$,

$$\Pr\left[V(\tau, y, \pi) = 1 \mid (\sigma, \tau) \leftarrow G(1^k, B),\ \pi \leftarrow P(\sigma, y, w)\right] = 1.$$

2. **Soundness (depending on which notion is considered)**:

   - **Non-adaptive**: For every polynomial-size prover P^*, every large enough security parameter $k \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance $y = (M, x, t) \notin \mathcal{L}_U$,

$$\Pr\left[V(\tau, y, \pi) = 1 \mid (\sigma, \tau) \leftarrow G(1^k, B),\ \pi \leftarrow P^*(\sigma, y)\right] \leq \mathrm{negl}(k).$$

   - **Adaptive**: For every polynomial-size prover P^*, every large enough security parameter $k \in \mathbb{N}$, and every time bound $B \in \mathbb{N}$,

$$\Pr\left[V(\tau, y, \pi) = 1 \wedge y \notin \mathcal{L}_U \mid (\sigma, \tau) \leftarrow G(1^k, B),\ (y, \pi) \leftarrow P^*(\sigma)\right] \leq \mathrm{negl}(k).$$

3. **Efficiency**: There exists a universal polynomial p such that for every large enough security parameter $k \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance $y = (M, x, t)$ with $t \leq B$:
   - $G(1^k, B)$ runs in time $p(k + B)$ for fully-succinct SNARG, or $p(k + \log B)$ for preprocessing SNARG.
   - $P(\sigma, y, w)$ runs in time $p(k + |M| + |x| + t + \log B)$ for fully-succinct, or $p(k + |M| + |x| + B)$ for preprocessing.
   - $V(\tau, y, \pi)$ runs in time $p(k + |M| + |x| + \log B)$.
   - An honestly generated proof π has size $p(k + \log B)$.

Definition 4.2 (Complexity-Preserving SNARG)

A triple of algorithms (G, P, V) is a *complexity-preserving SNARG* if it is a SNARG where efficiency is replaced by:

Complexity-preserving efficiency: There exists a universal polynomial p such that for every large enough security parameter $k \in \mathbb{N}$, every time bound $B \in \mathbb{N}$, and every instance $y = (M, x, t)$ with $t \leq B$:

- $G(1^k, B)$ runs in time $p(k + \log B)$.
- $P(\sigma, y, w)$ runs in time $(|M| + |x| + t) \cdot p(k + \log B)$.
- $P(\sigma, y, w)$ runs in space $(|M| + |x| + s) \cdot p(k + \log B)$.
- $V(\tau, y, \pi)$ runs in time $(|M| + |x| + \log t) \cdot p(k + \log B)$.
- An honestly generated proof π has size $p(k + \log B)$.

Definition 4.3 (SNARK)

A *SNARK* is a SNARG where adaptive soundness is replaced by:

Adaptive proof of knowledge: For every polynomial-size prover P^* there exists a polynomial-size extractor E_{P^*} such that for every large enough $k \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, and every time bound $B \in \mathbb{N}$:

$$\Pr\left[V(\tau, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}_U \mid (\sigma, \tau) \leftarrow G(1^k, B), (y, \pi) \leftarrow P^*(z, \sigma), w \leftarrow \mathsf{E}_{P^*}(z, \sigma)\right] \leq \mathrm{negl}(k).$$

Verifier privacy: If σ is public, we call the SNARK *publicly-verifiable* (pvSNARK). Otherwise, it is a *designated-verifier* SNARK (dvSNARK).

Universal vs. NP SNARKs: The SNARKs in Definition 4.3 are for the universal relation $\mathcal{R}_U$ (*universal SNARKs*). In this work, we use and construct SNARKs for NP.

Definition 4.4 (SNARK for NP)

A SNARK for NP is defined as in Definition 4.3, except that the adaptive proof of knowledge is weakened:

Adaptive proof of knowledge for NP: For every polynomial-size prover P^*, there exists a polynomial-size extractor E_{P^*} such that for every large enough $k \in \mathbb{N}$, every auxiliary input $z \in \{0,1\}^{\mathrm{poly}(k)}$, every time bound $B \in \mathbb{N}$, and every constant $c > 0$:

$$\Pr\Big[\, V_c(\tau, y, \pi) = 1 \wedge (y, w) \notin \mathcal{R}_c \;\Big|\; (\sigma, \tau) \leftarrow G(1^k, B),\ (y, \pi) \leftarrow P^*(z, \sigma),\ w \leftarrow \mathsf{E}_{P^*}(z, \sigma) \Big] \leq \mathrm{negl}(k).$$

2.4 Recursive ZKP

One of the latest advancements in efficient zero-knowledge (zk) proof generation is *recursive proofs*. A recursive zk proof is a proof that verifies some zk proofs inside of its circuit. The prover proves that they verified some inner proofs:

$$P(x_1, \pi_1, x_2, \pi_2, \ldots) \to \pi.$$

The proving circuit implements the constraints of zk proof verifiers for the inner proofs. When the verifier verifies the outer proof π, the inner proofs $\pi_1, \pi_2, \ldots$ are also verified:

$$V(x, \pi) \to \text{true} \Rightarrow \pi_1, \pi_2, \ldots \text{ are true.}$$

Recursive proofs offer the significant advantage of enabling parallel proof generation. This allows the total proving work to be distributed among multiple computers, rather than relying on a single device. Such an approach results in considerable performance improvements for proving multiple circuits or a single circuit that can be divided into small parts. Since the prover carries out work proportional to the circuit size, breaking a large circuit into smaller components will yield performance gains.

Recursive proof composition was first introduced in [18] and later realized in practice using cycles of elliptic curves [16]. Subsequent research, such as Halo [22] and Nova [35], has continued to enhance recursion speed and verification cost. Plonky2 [1], the most recent implementation, employs techniques from PLONK [27] and FRI [15].

In our work, we utilize Plonky2 to implement `zkTree` due to its rapid recursion speed. Tree structures are frequently employed in recursive ZKPs to generate the final proof that is verified on-chain. The concept of sharing verification costs by recursively verifying proofs within a tree was first introduced by [5]. zkEVMs [6] also utilize recursion trees for their specific use cases.

2.5 Arithmetic Circuits and Circuit Satisfiability

Arithmetic circuits are an NP language, where each instance of an arithmetic circuit is composed of *gates* and *wires*, and is defined over a finite field. Gates may be either input gates or operations (additions and multiplications) over the finite field. Wires carry values between these gates.

Formally, an arithmetic circuit C over a finite field $\mathbb{F}_p$ is an acyclic directed graph. Every node with indegree zero is an input gate (which, in our context, can be either public or private), and represents either a variable or a field element in $\mathbb{F}_p$. All other gates perform operations: either modular additions or modular multiplications.

Edges in an arithmetic circuit may be labeled with field elements, thereby constraining the values the corresponding wires are allowed to carry.

The **circuit satisfiability problem** is the problem of finding an assignment $s = [s_0, \ldots, s_{n-1}]$ to the input gates of a circuit C such that all constraints (as enforced by the labeled wires in C) are satisfied. Such an assignment s is called a *valid solution* of the circuit C with respect to its constraints.

It is possible to prove knowledge of a valid solution to a circuit in *zero-knowledge* using zk-SNARKs. This principle forms the foundation for many of the proof systems that we will explore throughout this document.

2.6 Custom Gates

Following prior work like TurboPLONK [26], Plonky2 makes extensive use of custom gates. To illustrate the model, suppose we are designing a gate for (field) division, $q = x/y$, or equivalently, $(qy = x) \wedge (y \neq 0)$.

To enforce $y \neq 0$, we ask the prover to supply a purported inverse, $i = 1/y$. We then enforce the following constraints:

$$qy = x,$$
$$yi = 1.$$

More concretely, we map these variables to wire polynomials. Suppose we assign:

$$w_1(x), w_2(x), w_3(x), w_4(x)$$

to represent x, y, q, and i, respectively. Then the constraints become:

$$w_3(x)w_2(x) - w_1(x) = 0, \tag{2.5}$$
$$w_2(x)w_4(x) - 1 = 0. \tag{2.6}$$

Note that these constraints should only be enforced on rows of our trace that correspond to division gates. A simple solution is to preprocess a filter polynomial $d(x)$, defined by:

$$d(g^i) = \begin{cases} 1 & \text{if the } i\text{th gate is a division gate,} \\ 0 & \text{otherwise.} \end{cases}$$

We use $d(x)$ to filter our division constraints, yielding:

$$d(x)(w_3(x)w_2(x) - w_1(x)) = 0, \tag{2.7}$$
$$d(x)(w_2(x)w_4(x) - 1) = 0. \tag{2.8}$$

2.7 Filtering Constraints

If we have k custom gates, introducing k individual filter polynomials would substantially increase the cost of our polynomial opening protocol. We can achieve the same effect with fewer polynomials by batching multiple filters together—a technique inspired by [38].

Suppose our circuit uses n custom gates and we fix a canonical ordering. Let $f(x)$ be a polynomial defined by its evaluations on g^i as follows:

$$f(g^i) = j \quad \text{if the } j\text{th gate is used in the } i\text{th row.}$$

Then, the jth gate's constraints can be filtered using:

$$f_j(x) = \prod_{\substack{0 \le k < n \\ k \neq j}} (f(x) - k),$$

which evaluates to non-zero on g^i if and only if the jth gate is used in the ith row.

Note that $f_j(x)$ has degree $n - 1$. Thus, filtering constraints with $f_j(x)$ can render some gates' constraint polynomials unacceptably high in degree. This issue can be mitigated by partitioning the gates into different subsets and defining distinct filter polynomials for each subset.

2.8 PLONKY2

Here we describe the interactive variant of the Plonky2 protocol. The non-interactive variant is obtained by applying the Fiat-Shamir transform.

Let r be our repetition parameter. This is the number of times we repeat certain checks, particularly those with soundness error $y/|\mathbb{F}_p|$, where y is some "small" quantity such as our circuit degree or constraint count.

We use $\text{Com}(\vec{p})$ to denote a commitment to a vector of polynomials. Concretely, it represents the cap of a Merkle tree, where each leaf corresponds to a single point $x \in H$, and contains the evaluations $p_1(x), \ldots, p_k(x)$.

2.8.1 Preprocessing

1. P, V construct the circuit and compute $\vec{C}$, a set of polynomials which encode the constants configured for each gate, as well as $\vec{\sigma}$, which encode the "extended" permutation described in [**?**].
2. P, V construct a Merkle tree containing $\vec{C}, \vec{\sigma}$.
3. P stores this Merkle tree as its proving key.
4. V stores $\text{Com}(\vec{C}, \vec{\sigma})$ as its verification key.

2.8.2 Main Protocol

1. P generates a witness $\vec{w}$ and sends $\text{Com}(\vec{w})$, a commitment to each wire polynomial $w(x)$.
2. V samples $\beta_1, \ldots, \beta_r, \gamma_1, \ldots, \gamma_r \in \mathbb{F}_p$, the randomness used in our permutation argument.
3. P sends $\text{Com}(\vec{Z}, \vec{\pi})$, a commitment to each permutation polynomial $Z(x)$ and partial product polynomial $\pi(x)$.
4. V samples $\alpha_1, \ldots, \alpha_r \in \mathbb{F}_p$, the randomness used to combine constraints.
5. P sends $\text{Com}(\vec{q})$, a commitment to each quotient polynomial $q(x)$. In particular,
$$q_i(x) = \frac{C_i(x)}{Z_H(x)},$$
where $C_i(x)$ is a combined constraint (see Section 2.6), and $Z_H(x) = x^n - 1$ is the vanishing polynomial on H.
6. V samples $\zeta \in \mathbb{F}_p(\phi)$, the random point at which polynomial identities are to be tested.
7. P sends $\vec{P}(\zeta)$, the purported evaluation of each polynomial at ζ, where $\vec{P} = (\vec{C}, \vec{\sigma}, \vec{w}, \vec{Z}, \vec{\pi}, \vec{q})$.
8. To verify these openings, P and V engage in a batch FRI protocol on the following list of purported codewords:
$$B = \left\{ \frac{p_i(x) - p_i(\zeta)}{x - \zeta} \mid p_i \in \vec{P} \right\}.$$

9. V uses $\vec{P}(\zeta)$ to infer the evaluation of the combined constraint polynomials $c_1(\zeta), \ldots, c_r(\zeta)$, and asserts that each $c_i(\zeta) = q_i(x)Z_H(x)$.

2.8.3 FRI Protocol

Our FRI protocol is nearly identical to that in [**?**], with minor differences such as the use of Merkle caps in place of Merkle roots.

1. V samples $\alpha \in \mathbb{F}_p(\phi)$ to reduce the batch B into a single purported codeword:
$$h_0(x) = \sum_{i=0}^{|B|-1} \alpha^i B_i(x).$$
2. P sends $\mathrm{Com}(h_0)$.
3. P and V carry out the FRI commit phase. For each reduction arity ℓ_i:
   a) V samples $\beta \in \mathbb{F}_p(\phi)$.
   b) P rewrites $h_i(x)$ as:
   $$h_i(x) = \sum_{j=0}^{\ell_i-1} x^j h_{i,j}(x^{\ell_i}),$$
   where $h_{i,j}$ contains the coefficients of $h_i(x)$ whose exponents are congruent to $j \mod \ell_i$.
   c) P sends $\mathrm{Com}(h_{i+1})$, a commitment to the polynomial:
   $$h_{i+1}(x) = \sum_{j=0}^{\ell_i-1} \beta^j h_{i,j}(x).$$
4. V samples $\tau \in \mathbb{F}_{4p}$.
5. P performs grinding [**?**] and sends the proof-of-work witness μ.
6. V asserts that $H(\tau, \mu)$ contains at least the required number of leading zero bits in its binary encoding.
7. V samples num_query_rounds random indices $q_1, \ldots, q_k \in \{0, \ldots, n-1\}$.
8. For each query index q_i:
   a) P sends the evaluations $\vec{P}(x)$ and each $h_i(x)$, where x is the ith point in the coset over which codewords are defined.
   b) V performs a series of consistency checks at x: first between $\vec{P}(x)$ and $h_0(x)$, and then between each pair $(h_i(x), h_{i+1}(x))$. See [**?**] for further details.

Chapter 3

Design of zkIPV

3.1 Selection of framework

Here we benchmark some of the most widely used zero-knowledge (ZK) proof systems to identify the most suitable framework for our application. The goal is to compare their performance in terms of proving and verification time, particularly for operations relevant to image provenance, such as Poseidon hashing. This evaluation guides our selection of an efficient and scalable ZK backend. All benchmarks (except web benchmarks) were run on GCP N2 instances.

Protocol	Platform Details	Prover Time (s)	Verifier Time (s)
Groth16	snarkjs	0.910267	0.787 781
Groth16	Chrome (Catalina)	1.1015	0.3255
Groth16	Firefox (Catalina)	1.375	0.445
Halo2	Native	0.13675	0.002 262 3
Plonky2	Native	0.0000020804	0.000 001 204

Table 3.1: Unified benchmark results for Groth16, Halo2, and Plonky2.

Among the evaluated zero-knowledge proof systems, Plonky2 demonstrated the best performance in terms of prover and verifier times—especially when applied to Poseidon hashing. With prover times in the order of microseconds and similarly low verifier times, Plonky2 stands out as the most computationally efficient framework.

Given our use case of verifying authorized image transformations, where many Poseidon hashes are computed as part of the circuit logic (e.g., hashing pixel blocks or metadata), the efficiency of the hashing operation becomes a critical performance factor. Poseidon is a SNARK-friendly hash function optimized for recursive proof systems, and Plonky2 is specifically designed

to support such use cases with minimal overhead.

Moreover, Plonky2 supports fast recursive composition, enabling future scalability improvements where multiple proofs (e.g., per image region or step in an editing pipeline) can be aggregated efficiently. This property is particularly relevant for high-resolution images, where computation and proof sizes grow rapidly.

For these reasons, and particularly because Plonky2 offers the fastest Poseidon hash implementation among the tested frameworks, we chose Plonky2 as the core ZK proving system for our image provenance pipeline.

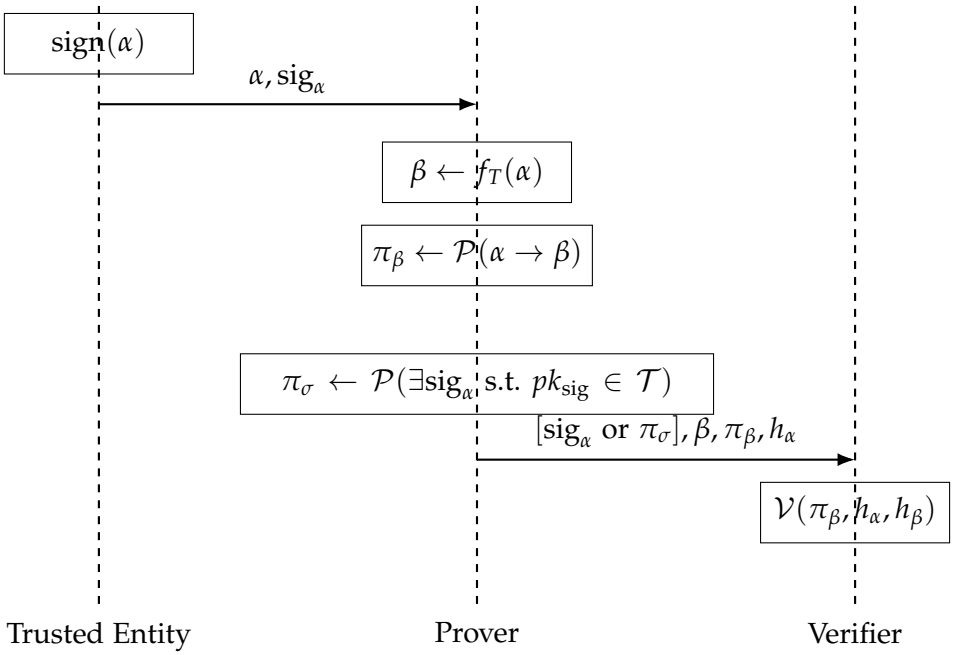


Figure 3.1: Protocol diagram for privacy-preserving verification.

3.2 Threat Model

In the C2PA setting, every camera is equipped with an embedded certified signing key for a secure signature scheme. The secret signing key is generated on the camera and certified by a C2PA certificate authority at manufacturing time.

Every time a camera takes a photo, it signs the raw RGB values of the photo's pixels and relevant metadata (e.g., location, timestamp, exposure time). We assume that the adversary has access to the camera and the camera's public key. However, the C2PA assumes that the attacker:

- Cannot extract the signing key from the camera.
- Cannot cause the camera to sign an image that was not captured by the camera's optical hardware.

In this setting, the only root of trust is the camera and its signing key. The editor of a photo is not trusted in any way. The verifier wants to ensure that the received edited photo is the result of applying an acceptable transformation to a C2PA-signed image.

While C2PA is an important step towards image provenance, it is not a complete solution and must be combined with other defenses. Specific attacks on C2PA are out of scope for this paper, as our primary focus is on securing the image editing pipeline.

Nevertheless, for completeness, we describe a few potential attacks on C2PA and how they might be addressed. These attacks are considered in the C2PA documentation.

The adversary might extract the C2PA signing key from some deployed camera. For instance, the Leica camera implements a hardware trusted execution environment (TEE) to protect the key and make extraction harder (though not impossible).

If a key is extracted, the standard includes a revocation mechanism that alerts all verifiers to revoke a compromised C2PA certificate.

An attacker might display an AI-generated image on a screen and then take a picture of the screen using a C2PA camera. The result is a properly signed image of a fake event.

This is a known challenge. One defense is to require the verifying client to run a picture-of-picture detector. For instance, the focal length in the signed metadata may indicate a close-up of a screen, which is inconsistent with the purported scene.

Other detection strategies have been suggested, but this may result in a cat-and-mouse game between attackers and defenders.

The list of allowed transformations (e.g., by the Associated Press) may unintentionally allow altering the semantic meaning of an image. For example, cropping out a person from a photo could be used to falsely imply they were not present.

C2PA may pose privacy risks since the signature on a photo can identify the camera that took it. One mitigation strategy is to use group signatures [13, 19], which can anonymize the origin while preserving provenance.

As explained above, these attacks are considered out of scope. We operate within the C2PA threat model, which assumes protection against the above threats. Our focus is on securing the image editing pipeline.

3.3 Design of zkIPV

3.3.1 System Overview

We present zkIPV as an interaction between a news organization (prover) and a web browser (verifier). The system ensures that every photo displayed in a news article is accompanied by comprehensive metadata including location, timestamp, focal length, and other relevant information. Additionally, each photo must include a detailed description of the edits that were made to the original photo, along with a succinct zero-knowledge proof that verifies the authenticity of these modifications.

The public statement in our system consists of three key components:

- The published edited photo, denoted as x
- The edits performed to the original photo, represented by function f
- The camera's public key, denoted as vk

The secret witness contains two critical elements:

- The original photo, denoted as w
- The camera's signature σ on the original photo w

3.3.2 Mathematical Framework

Our primary objective is to design an efficient proof system for the instance-witness relation:

$$\mathcal{R} := \{(vk, f, x); (w, \sigma) : f(w) = x \wedge \mathcal{V}_f(vk, w, \sigma) = 1\} \tag{3.1}$$

While equation (1) places the verification key vk in the public statement, we note that photographer privacy can be enhanced by moving vk and its certificate to the secret witness. For clarity of presentation, we utilize relation (1) and discuss the more private variant in Section 9 of the extended work.

3.3.3 Verification Protocol

A web browser will only accept a photo's provenance if the photo x is accompanied by a triple (π, vk, f) where:

1. π is a valid zero-knowledge proof for relation $\mathcal{R}$
2. vk is a properly certified C2PA verification key
3. The function f, which encodes the list of edits, is "acceptable" as defined by the Associated Press standards

We can thus conceptualize zkIPV as enforcing a whitelist of allowable edits, ensuring that only legitimate modifications are permitted while maintaining the integrity of the original photographic evidence.

3.3.4 Zero-Knowledge SNARK Properties

zkIPV relies on all the fundamental properties of a zero-knowledge Succinct Non-Interactive Argument of Knowledge (zk-SNARK):

- **Completeness and Knowledge-Soundness**: These properties ensure that the web browser does not need to trust the editor, thereby preserving end-to-end security of the signature verification process.
- **Non-Interactivity**: This property means that the news organization does not need to interact with any web browser and can instead publish a single proof along with the news article that any verifier can independently check.
- **Zero-Knowledge**: This property ensures that the proof does not reveal information that was cropped or blurred in the original photo, maintaining privacy of sensitive content.
- **Succinctness**: This property ensures that the proof can be quickly verified by the browser within a few seconds, making the system practical for real-world deployment.

3.4 Proof System Architecture

We now turn to designing a proof system for the relation $\mathcal{R}$ from equation (3.1). The proof system consists of two main components:

3.4.1 Image Transformation Proof

First, we require a proof that $f(w) = x$, for an image transformation function f. This component verifies that the published image x is indeed the result of applying the declared edits f to the original image w.

3.4.2 Signature Validity Proof

Second, we need a proof that σ is a valid signature on w. This component is significantly more complex and represents a major technical challenge in our system design. We present our comprehensive approach in the following section.

3.4.3 Witness Consistency

We must also ensure that the secret witness w used to generate both proofs are identical. This consistency requirement is crucial for the security of the entire system and is achieved through careful cryptographic techniques.

3.5 Proving Knowledge of a Valid Signature

3.5.1 Traditional Signature Verification

When verifying a signature σ on some data w, the standard verification process involves two steps:

1. Compute $h \leftarrow H(w)$, where H is a collision-resistant hash function
2. Verify that σ is a valid signature on h

When the data being verified is large, as in the case of a high-resolution photograph, the majority of computational time is spent on computing the hash h in step (1). The same computational bottleneck applies when proving knowledge of a valid signature within a zero-knowledge framework.

3.5.2 The Hashing Challenge

The primary challenge is to design an efficient SNARK circuit that can verify that the hash h of a large amount of data w is computed correctly. Once the verifier has a valid hash h, proving knowledge of an ECDSA signature on h can be accomplished using existing circuits.

Ideally, we would prefer to use a standard hash function such as SHA-256. Unfortunately, proving that we have honestly applied SHA-256 to a 30-megapixel (MP) witness is practically infeasible. This infeasibility stems from the fact that SHA-256 consists primarily of non-algebraic operations (such as logical operations), and proving non-algebraic constraints in a zk-SNARK is extremely time-consuming.

3.5.3 Our Approach

We propose two solutions to address this fundamental challenge:

Solution 1: Poseidon Hash Construction

Our first solution, While there exist SNARK-friendly hash functions like Poseidon, proving that we have honestly applied Poseidon to a 30 MP witness is also quite challenging in practice due to the sheer size of the input data, thus a efficient method to prove the instance-witness relation would be proving it recursively.

$$\mathcal{R}_{hash} := \{h; w : h = H(w)\} \tag{3.2}$$

Solution 2: Polynomial Commitment Hash

Our second solution employs a polynomial commitment scheme as the collision-resistant hash function H. Computing this hash function on the image w requires more computational resources than our first solution. However, once the hash value is computed, incorporating it into a SNARK proof requires no additional work. This approach is particularly suitable when the original photo signer has sufficient computing power to compute a polynomial commitment to the original photo.

3.6 Polynomial Commitment Hash

3.6.1 High-Level Approach

Our second approach is specifically designed for scenarios where the signer has substantial computational power, such as the owner of a generative AI model. As before, we encode the original image w as three vectors $v_r, v_g, v_b \in [0, 255]^m$.

For simplicity in this presentation, we consider only one such vector, denoted by v, which can be thought of as any of the three color channel vectors.

3.6.2 Polynomial Interpolation

In our second approach (mode 2), the signer interpolates a univariate polynomial W of degree at most $m - 1$ such that $W(\omega^i) = v_i$ for $i = 1, \ldots, m$. We denote this polynomial by $\text{poly}(w)$, namely:

$$W := \text{poly}(w) \tag{3.3}$$

3.6.3 Commitment Generation

Next, the signer computes a succinct polynomial commitment to $W(X)$ as:

$$\text{com} \leftarrow \text{PCS.commit}(pp, W, r) \tag{3.4}$$

where r is the commitment randomness chosen by the signer, and pp represents the public parameters of the polynomial commitment scheme.

The signer then signs com to obtain a signature σ on com. The signer sends the image w along with $(vk, \text{com}, \sigma, r)$ to the news editor, where vk is the signer's certified public key.

3.6.4 Editor's Workflow

The news editor creates an edited image $x := f(w)$ by applying the permitted editing function f to the original image w. Because the polynomial commitment is computationally binding, it suffices for the editor to construct a zero-knowledge proof for the following instance-witness relation:

$$\mathcal{R}_{simple} := \{(pp, f, x, \text{com}); (w, r) : f(w) = x \wedge \text{com} = \text{PCS.commit}(pp, \text{poly}(w), r)\} \tag{3.5}$$

3.6.5 Efficient Proof Construction

A naive approach to prove relation (7) would require the SNARK circuit to verify both that $f(w) = x$ and that com is a polynomial commitment to $\text{poly}(w)$. However, proving the latter would be far more expensive than proving that the editor has correctly hashed w

3.6.6 Key Insight: Modified PLONK

The key insight is that for the PLONK proof system, building a proof for $\mathcal{R}_{simple}$ requires no more work for the editor than simply proving that $f(w) = x$. This means that the SNARK circuit never needs to hash w, which greatly reduces the computational burden for the editor.

To achieve this reduction in work, zkIPV modifies PLONK's permutation argument. Let $C(x, w)$ be a circuit that outputs 0 if and only if $f(w) = x$.

3.6.7 Modified PLONK Protocol

Remarkably, if an editor wants to build a proof for $\mathcal{R}_{simple}$, it can simply use C as the circuit provided to the PLONK system. However, this requires modifying the PLONK prover to indirectly prove that com is a valid commitment to $\text{poly}(w)$.

Computation Trace Construction

First, the editor builds the computation trace $T(X)$ for $C(x, w)$ just as the standard PLONK prover would. The standard PLONK prover would then construct a proof π that $T(X)$ is a valid computation with respect to the gate constraints and copy constraints specified by C.

Extended Permutation Argument

Our editor must additionally prove that com is a valid commitment to $\text{poly}(w)$. Some entries in $T(\Omega)$ correspond to the witness w, as illustrated in

Figure 4. Thus, proving that com is a valid commitment to $\text{poly}(w) = W(X)$ is equivalent to proving that the witness elements represented by entries of $W(\Omega)$ are equal to the corresponding witness entries in $T(\Omega)$.

The editor can prove this equality by extending the PLONK permutation argument. The permutation argument proves that the vector $T(\Omega)$ is equal to the vector $T(\tau(\Omega))$, where τ is a polynomial that implements a permutation of Ω.

Constraint Extension

The standard PLONK prover defines τ to capture copy constraints within the circuit C and then uses the permutation argument to prove that the computation trace respects the circuit wiring. In our scheme, the editor extends the PLONK permutation argument to prove that all entries in $T(\Omega)$ that correspond to a witness element are equal to the corresponding witness element in $W(\Omega)$.

More specifically, the editor extends τ to a new permutation τ' that captures additional copy constraints between $T(X)$ and $W(X)$. The standard PLONK copy constraints are represented by black edges in Figure 4, while the thick red edges represent the extended copy constraints between $T(\Omega)$ and $W(\Omega)$.

3.6.8 Verification Protocol

The news editor uses this extended protocol to efficiently construct a proof that $C(x, w) = 0$ and that com is a valid commitment to $\text{poly}(w)$. It uses both $T(X)$ and $W(X)$ to build a PLONK proof with respect to the permutation τ', which includes the new copy constraints.

The result is a proof π that (pp, f, x, com) is a valid instance of $\mathcal{R}_{simple}$.

3.6.9 Final Verification

The final data sent to the verifier (the news reader) is (x, f) along with the proof $\pi' := (vk, \text{com}, \sigma, \pi)$. The verifier checks that:

1. $\mathcal{V}_f(vk, \text{com}, \sigma)$ accepts (i.e., σ is a valid signature on com)
2. π is a valid proof that (pp, f, x, com) is a valid instance of $\mathcal{R}_{simple}$

We note that because the polynomial commitment is computationally hiding, the commitment com to w reveals nothing to the verifier about w beyond what can be inferred from the edited image x.

3.6.10 Complexity Analysis

Augmenting the permutation argument to operate over two polynomials T and W in this manner does not significantly change the proving time

compared to simply proving that $f(w) = x$ for a public value x and witness w. This efficiency is achieved because the additional copy constraints are far fewer than the copy constraints required for the circuit C.

3.7 Summary and Trade-offs

This polynomial commitment method results in a massive reduction in computational work for the editor, because the zk-SNARK circuit is now much simpler than the circuit required previously. In particular, the circuit does not need to hash the large original image w.

While this approach saves significant work for the editor, it creates additional computational burden for the signer because computing a polynomial commitment to w is more costly than computing a poseidon hash of w.

The choice between these two approaches depends on the specific deployment scenario and the computational resources available to different parties in the system. For scenarios where editors have limited computational power but signers have substantial resources, the polynomial commitment approach is preferable. Conversely, when computational resources are more evenly distributed, the lattice + Poseidon hash approach may be more appropriate.

3.8 Conclusion

The zkIPV system represents a significant advancement in photo provenance verification, providing a practical framework for ensuring the integrity of digital images while preserving privacy and enabling efficient verification. The two-pronged approach to signature verification addresses different deployment scenarios and computational constraints, making the system adaptable to various real-world applications.

The innovative use of zero-knowledge proofs in this context not only solves the technical challenges of large-scale image verification but also establishes a new paradigm for trustworthy digital media in an era of increasing concern about image manipulation and misinformation.

Chapter 4

Implementing zkIPV

4.1 Image Cropping

In this section, we implement a zero-knowledge proof system using the `Plonky2` framework to verify that a cropped portion of an image corresponds to the original image. The original image is treated as a private input, while the cropped portion and its hash are public inputs. The circuit enforces consistency of the crop and commits to the original image using a Poseidon hash, enabling succinct and efficient proof generation.

Overview

Let the original image be represented as a 2D RGB image of resolution 1280×720, and let the cropped image be the upper 1280×540 portion of it. We encode pixel values into field elements using a custom packing scheme and construct a STARK-friendly circuit with the following constraints:

- The original image is serialized and packed into a vector of field elements $\mathbf{W} \in \mathbb{F}^N$, where $N = \frac{1280 \cdot 720}{3}$.
- The cropped image consists of the first $M = \frac{1280 \cdot 540}{3}$ elements of $\mathbf{W}$.
- The circuit exposes:
  - The Poseidon hash $H = \texttt{Poseidon}(\mathbf{W})$ as a *public output*.
  - The cropped image values $\mathbf{C} = \mathbf{W}_{0..M}$ as *public inputs*.
- The circuit enforces: $\forall i < M,\ \mathbf{W}[i] = \mathbf{C}[i]$.

This design ensures that a verifier can check the cropped image's correctness and confirm it is derived from a committed full image, without revealing the full image content.

Field Encoding of RGB Pixels

Each RGB pixel is represented using 3 bytes. To fit efficiently into a field element of the Goldilocks field $\mathbb{F}_{2^{64}-2^{32}+1}$, three RGB pixels (9 bytes) are packed into a single field element by stripping the least significant bit (retaining 7 bits per byte): For each 3-pixel chunk (9 bytes), we strip the lowest bit of each byte, yielding 63 bits. These 63 bits are concatenated and converted into a 64-bit integer. The integer is interpreted as a canonical field element. This tradeoff allows efficient packing without overflow, while maintaining approximate visual fidelity.

Circuit Logic and Constraints

The circuit is constructed using Plonky2's layered API:

1. Allocate virtual targets for each packed pixel from the full image.
2. Allocate separate virtual targets for each cropped pixel.
3. Enforce equality constraints between the cropped pixel targets and the corresponding prefix of the full image pixel targets:

$$\texttt{connect}(\texttt{crop\_targets}[i], \texttt{pixel\_targets}[i]) \quad \forall i < M$$

4. Hash the full image pixel vector using Poseidon: `hash_n_to_hash_no_pad`.
5. Register hash output and cropped pixels as public inputs.

Proof System Workflow

Algorithm 1 Proof Generation for Cropped Image Consistency

Input: Original image $\mathcal{I}$, crop size (1280×540)
Load and pack image into $\mathbf{W} \in \mathbb{F}^N$
Extract $\mathbf{C} \leftarrow \mathbf{W}_{0..M}$
Initialize circuit builder over Goldilocks field
for each index $i < N$ **do**
    Add virtual target $t_i \leftarrow \mathbf{W}[i]$
for each index $i < M$ **do**
    Add target $c_i \leftarrow \mathbf{C}[i]$
    Register c_i as public input
    Enforce constraint: $t_i = c_i$
Compute $H \leftarrow \texttt{PoseidonHash}(\mathbf{W})$
Register H as public output
Assign witness values from packed image
Generate zero-knowledge proof

The use of a Poseidon hash binds the cropped image to a committed full image. While the cropped image is public, the original image remains hidden unless voluntarily revealed.

4.2 Grayscale Operation

In this section, we present the circuit that verifies the correctness of a grayscale transformation over a color image using the Plonky2 SNARK framework. The goal is to enable a prover to convince a verifier that a grayscale image has been computed correctly from a private RGB image using a fixed linear weighting scheme, without revealing the original RGB image itself.

We use a weighted average technique where grayscale values are derived using a weighted sum of red, green, and blue channels, using the equation:

$$\text{Gray} = \frac{5 \cdot R + 9 \cdot G + 2 \cdot B}{16}$$

This is implemented within a Plonky2 circuit, where each grayscale output is calculated from packed RGB values using fixed constants. The proof does not require zero-knowledge (ZK flag is disabled), and we hash the entire RGB input using Poseidon hash and register the hash as an additional public input to commit to the image.

Algorithm 2 Grayscale Image Proof Circuit (Pseudocode)

Private Input: RGB pixel values in field form (packed 5 pixels per field)
Public Input: Grayscaled values in field form (packed 5 pixels per field)
Constants: $w_r = 5$, $w_g = 9$, $w_b = 2$
for $i = 0$ to N (number of packed pixel groups) **do**
    $R_i \leftarrow$ add_virtual_target()
    $G_i \leftarrow$ add_virtual_target()
    $B_i \leftarrow$ add_virtual_target()
    $r_{\text{scaled}} \leftarrow w_r \cdot R_i$
    $g_{\text{scaled}} \leftarrow w_g \cdot G_i$
    $b_{\text{scaled}} \leftarrow w_b \cdot B_i$
    $\text{gray}_i \leftarrow r_{\text{scaled}} + g_{\text{scaled}} + b_{\text{scaled}}$
    Register gray_i as public input
    Build the circuit and generate the proof

The circuit is configured with the standard Plonky2 recursion parameters using the PoseidonGoldilocks configuration. RGB inputs are packed into field elements by storing every 5 RGB pixels (15 bytes) using 12 bits per

channel to avoid overflow in a 64-bit field element. After proof generation, the public grayscale values are extracted and converted back into grayscale pixel format to reconstruct the output image. The proof is then verified using the Plonky2 verifier.

4.3 Blur Operation

In this section, we describe the implementation of a blur operation applied to a 2D image and how a zero-knowledge proof is generated using the Plonky2 framework to prove that the transformation was applied correctly.

The blur is implemented as a 3×3 average filter, where each pixel in the blurred region is the rounded average of its 8-connected neighborhood (including itself). To enable verifiability, the original image is provided as a private witness, and the blurred output (or unblurred remainder of the image) is exposed as public input. Additionally, a Poseidon hash of the original image is exposed publicly to allow verification of image integrity without revealing the image itself.

To ensure correctness of the average operation in zero-knowledge, the circuit checks that the result $x_{i,j}$ satisfies:$(\sum_{(u,v)\in N(i,j)} w_{u,v}) + 4 = 9 \cdot x_{i,j} + r$

Where: - $N(i,j)$ are the 9 neighboring pixel values (including the center), - $r \in [0,8]$ is the remainder.

This is enforced using Plonky2's `range_check` constraint, validating the bounded range of r.

Algorithm 3 Blur Circuit Construction in Plonky2

Private Input: Private image matrix $w_{i,j}$ of size $H \times W$
Public Input: Blurred output $x_{i,j}$ in region $H_b \times W_b$, Poseidon hash of w
for $i \leftarrow 0$ to $H - 1$ **do**
    for $j \leftarrow 0$ to $W - 1$ **do**
        if i, j within blur region **then**
            Collect 3x3 neighbors of $w_{i,j}$ into list N
            $s \leftarrow \sum N$
            $s' \leftarrow s + 4$
            Create virtual target $x_{i,j}$
            $z \leftarrow 9 \cdot x_{i,j}$
            $r \leftarrow s' - z$
            Enforce $r \in [0, 8]$ using range checks
        else
            Register $w_{i,j}$ as public input
Compute Poseidon hash of all $w_{i,j}$
Register Poseidon output as public input
Build the circuit and generate the proof with zero knowledge configuration

The proof verifies that:

- The blurred region is correctly averaged.
- The unblurred portion matches the public input.
- The full image corresponds to the public Poseidon hash.

The circuit supports partial disclosure of the image—exposing only the blurred region or border pixels while maintaining full integrity via the hash. This design allows privacy-preserving image transformations with succinct and efficient proofs.

4.4 Image Resize Proof with Bilinear Interpolation

In this section, we implement a zero-knowledge circuit for verifying a bilinear resize operation on an image using the Plonky2 proof system. The goal is to allow a prover to demonstrate that a resized image was correctly derived from a larger source image, without revealing the full source image. A Poseidon hash commitment of the original image is used as a public input to bind the resized image to its source.

We assume that the image is downsampled (or upsampled) using bilinear interpolation. For each pixel in the resized image, the proof enforces that

its value is a weighted sum of four neighboring pixels from the original image. The weights are determined by the interpolation ratios derived from the pixel's position in the resized image.

The circuit treats each pixel value as a field element. For a resized image of dimensions $H_{new} \times W_{new}$, the circuit performs the following for each pixel (i, j):

- Allocate four virtual targets representing the four neighboring source pixels: a, b, c, d.
- Multiply each pixel value by its corresponding bilinear weight (precomputed constants).
- Sum the four weighted values to obtain the interpolated result.
- Register this value as a public input.

Poseidon hashing of the original (packed) image is done similarly as in the cropping circuit (see Section **??**), and the resulting digest is added as a public input to bind the private witness to a committed source image.

The circuit also supports verification of rounding via a remainder term stored privately. This ensures that integer truncation from floating-point interpolation is faithfully proven.

Pseudocode: Resize Proof Circuit Construction

Algorithm 4 Resize Circuit via Bilinear Interpolation in Plonky2

Input: Resized dimensions (H_{new}, W_{new}), original image hash H, target image T
Private: Original image pixels P, rounding remainders R
for $i = 0$ to $H_{new} - 1$ **do**
    for $j = 0$ to $W_{new} - 1$ **do**
        Compute lower and upper bounds (x_l, x_h, y_l, y_h) from interpolation
        Allocate four virtual targets: $a = P[y_l][x_l]$, $b = P[y_l][x_h]$, $c = P[y_h][x_l]$, $d = P[y_h][x_h]$
        Compute bilinear weights: $\alpha, \beta, \gamma, \delta$
        Multiply: $w_1 = a \cdot \alpha$, $w_2 = b \cdot \beta$, $w_3 = c \cdot \gamma$, $w_4 = d \cdot \delta$
        Sum: $s = w_1 + w_2 + w_3 + w_4$
        Register s as public output
        Enforce: $s = T[i][j] \cdot \texttt{denominator} + R[i][j]$
Compute Poseidon hash of original image
Register hash digest as public input

4.5 Sharpness Enhancement

In this section, we describe the implementation of the *sharpness enhancement* operation on an image and how a zero-knowledge proof is constructed using the Plonky2 framework.

Overview. Sharpness enhancement is performed by applying a convolutional kernel over the image, which amplifies edges and fine details. This is achieved using a standard sharpening filter applied in a sliding-window manner. The original (unfiltered) image is kept private, and only the sharpened image and a commitment (Poseidon hash) to the original image are revealed as public inputs.

Constraints. For each pixel (excluding the borders), the following constraint is enforced:

$$\texttt{sharpened}[i] = 5 \cdot \texttt{orig}[i] - \texttt{orig}[i-1] - \texttt{orig}[i+1] - \texttt{orig}[i-w] - \texttt{orig}[i+w]$$

where w is the image width and pixel access is restricted to ensure boundary safety.

Pseudocode. The implementation can be abstractly described as follows:

Algorithm 5 Proving Sharpness Enhancement with Poseidon Hash Commitment

1: Load image and convert to packed field vector
2: Allocate `orig_pixels` as private input targets
3: Allocate `sharpened_pixels` as public input targets
4: **for** each pixel i (excluding border) **do**
5: Compute $v = 5 \cdot \texttt{orig}[i] - \texttt{orig}[i-1] - \texttt{orig}[i+1] - \texttt{orig}[i-w] - \texttt{orig}[i+w]$
6: Enforce constraint: $\texttt{sharpened}[i] = v$
7: Compute Poseidon hash of `orig_pixels`
8: Register the hash as public output
9: Build circuit, assign witness, generate proof

Verification. The verifier checks that:

- The provided sharpened image matches the computation from the original (private) image.
- The Poseidon hash matches the committed original image.

This ensures the integrity of the sharpness operation while preserving the privacy of the raw image data.

4.6 Brightness Adjustment

In this subsection, we describe the construction of a zero-knowledge circuit to prove that an image was modified by applying a brightness adjustment operation — either increasing or decreasing its pixel intensities — and that the modified image corresponds to a committed original image.

Operation: Brightness adjustment involves modifying each RGB channel of a pixel by adding or subtracting a constant scalar value $b \in \mathbb{Z}$, applied element-wise with clipping at 0 and 255.

Circuit Overview: The circuit accepts as:

- **Private input:** The full original image, packed into field elements.
- **Public input:** The brightness-adjusted image (also packed) and the Poseidon hash commitment to the original image.

Packing: Pixels are packed three RGB values at a time (9 bytes) into a single 63-bit field element, similar to the crop operation. Packing reduces the number of gates and increases performance.

Clipping: Since clipping to the $[0, 255]$ range is non-linear and would require costly logic in zero-knowledge, we restrict the brightness scalar to a safe range where overflow or underflow does not occur, i.e., $b \in [-100, 100]$.

Pseudocode: Brightness Adjustment Proof

Algorithm 6 Prove Brightness Adjustment Correctness

Input: Original image $I \in \mathbb{F}^n$, brightness scalar b, packed as private input
Input: Brightness-adjusted image $I' \in \mathbb{F}^n$, packed as public input
Input: Poseidon hash $H = \texttt{PoseidonHash}(I)$, as public input
Step 1: Recompute $H^* \leftarrow \texttt{PoseidonHash}(I)$
Step 2: `assert` $H^* = H$
for $i = 0$ to n **do**
    $I'_i \leftarrow I_i + b$
    `assert` I'_i equals public input

4.7 Contrast Adjustment

This subsection describes the construction of a zero-knowledge proof circuit that verifies the correctness of a contrast adjustment applied to an im-

age. The goal is to prove, without revealing the original image, that a given contrast-adjusted subregion corresponds correctly to a committed full image using the Poseidon hash function.

The prover begins by loading a raw RGB image of dimensions 1280×720, which is then packed into field elements. Each field element encodes three RGB pixels by preserving the most significant 7 bits per channel to fit within a 63-bit range. The original image is hashed using Poseidon to produce a succinct public commitment.

The contrast adjustment operation is implemented using a fixed contrast factor $\alpha > 1$, applied as follows:

$$\text{new\_value} = \text{clamp}(128 + \alpha \cdot (\text{original\_value} - 128), 0, 255)$$

This adjustment is performed off-circuit to generate the expected result, which is then verified inside the circuit by checking that each adjusted pixel in the output subregion matches the transformation applied to the corresponding pixel in the original committed image.

The circuit takes as:

- **Private inputs**: The packed field elements representing the full image.
- **Public inputs**: (1) The Poseidon hash of the full image, and (2) the contrast-adjusted field elements for a fixed subregion (e.g., top 540 rows).

Each element in the contrast-adjusted subregion is connected to its corresponding original input pixel via the inverse of the contrast formula, ensuring consistency with the transformation. The proof is generated over this constraint system, and the verifier can efficiently check that the adjusted region is correctly derived from the committed image, without learning anything about the rest of the image.

Pseudocode Overview:

4.8 JPEG Compression

JPEG Compression Overview

JPEG (Joint Photographic Experts Group) is a widely used image compression standard that significantly reduces image file sizes by applying lossy compression. It achieves this by exploiting the limitations of the human visual system—removing details that are less perceptible while preserving those that are visually important.

Algorithm 7 Contrast Adjustment ZK Proof

1: Load image I of size 1280×720
2: Pack I into field elements f_i for circuit
3: Select top half rows: $\texttt{crop}(I, 0, 0, 1280, 540)$
4: Apply contrast: $O = \texttt{adjust\_contrast}(\texttt{crop}, \alpha)$
5: Compute Poseidon hash $H = \texttt{PoseidonHash}(f_i)$
6: Initialize circuit builder
7: Add private targets for original image fields f_i
8: Add public inputs:
- Poseidon hash H
- Packed contrast-adjusted output O

9: **for** each contrast-adjusted target o_j **do**
10: Connect o_j to contrast formula applied on corresponding f_i
11: Generate and compress proof π
12: Verify proof using public inputs (H, O)

Compression Pipeline

Algorithm 8 JPEG Compression Pipeline

1: **Input:** RGB image of resolution $W \times H$
2: Convert the image from **RGB** to **YCbCr** color space
3: Downsample the **Cb** and **Cr** chrominance channels (e.g., 4:2:0)
4: **for all** channels (Y, Cb, Cr) **do**
5: Divide image into 8×8 blocks
6: **for all** blocks **do**
7: Subtract 128 from each pixel value
8: Apply **2D Discrete Cosine Transform (DCT)**
9: Quantize DCT coefficients using standard quantization tables
10: Serialize quantized blocks into final binary format (.jpg)
11: **Output:** Compressed JPEG image

Color Space Conversion

JPEG first converts images from the RGB color space to YCbCr to separate luminance (Y) from chrominance (Cb, Cr). The transformation is defined as:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

Downsampling

Because human vision is more sensitive to brightness than to color, the chrominance channels (Cb, Cr) can be downsampled (commonly in 4:2:0 format), reducing their resolution by a factor of four without noticeably impacting visual quality. The luminance channel (Y) remains at full resolution.

Block Division and DCT

Each channel is divided into 8×8 pixel blocks. For each block:

- Subtract 128 from each pixel to center values around zero.
- Apply a 2D Discrete Cosine Transform (DCT) to convert the spatial-domain block into frequency components.

Quantization

High-frequency DCT coefficients are less important visually and are more aggressively quantized. Quantization is performed using standard tables. For example:

$$Q^Y = \begin{bmatrix} 16 & 11 & 10 & \cdots \\ 12 & 12 & 14 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad Q^{CbCr} = \begin{bmatrix} 17 & 18 & 24 & \cdots \\ 21 & 23 & 26 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}$$

Quantized coefficient calculation:

$$C'_{u,v} = \text{round}\left(\frac{C_{u,v}}{Q_{u,v}}\right)$$

This introduces controlled loss, reducing storage while preserving important information.

Poseidon Hash Commitment

After quantization, the JPEG image is processed block-wise to convert each 8×8 coefficient matrix into a sequence of field elements. This step enables cryptographic hashing within a SNARK-compatible environment.

Algorithm 9 Hash Commitment of JPEG Data

Input: Quantized DCT coefficients for all blocks
for all blocks **do**
    Pack each 64-length coefficient array into field elements
Flatten all packed blocks into one field element vector
Compute **Poseidon hash** on the complete vector
Output: Image hash $h \in \mathbb{F}_p$

Chapter 5

Performance

5.1 Performance Results

In this section, we evaluate proof generation time, verification time, and proof size for the hash relation and for the image editing relations. We ran our timing experiments on png images on a virtual machine with 64 GB of RAM and 12 CPU cores. When considering what is a reasonable amount of time to generate and verify proofs, it is important to remember that proof generation only needs to happen once per photo, while proof verification needs to be performed by every browser that accesses the article. This means that while proof generation needs to be fast, proof verification needs to be very fast.

5.1.1 Hash Proof Generation Results

Below we represent the Prover memory needs and Verifier time for Poseidon hash generation for zkIPV.

Image Size (KP)	Peak Memory (GB)	Proving Time (s)	Verify Time (s)
1	1.14	0.75	0.087
10	1.52	0.95	0.156
100	1.81	1.27	0.170
1,000	2.38	3.52	0.180
10,000	15.4	35.41	0.72
30,000	26.2	400.36	0.75

5.1.2 Photo Edit Proof Generation Results

To evaluate the Plonky2-based implementation for image editing proofs, we used 33 MP input images, consistent with high-end camera outputs (e.g.,

Sony Alpha series). For resizing and cropping, we targeted a standard publishing resolution of 2048×1365. For grayscale and blurring, we applied edits directly on the resized image.

Table 5.1: Cropping Proof Generation Timing

Original Size	Cropped Size	Setup Time (min)	Proof Gen Time (min)
6632 x 4976	2048 x 1365	2.34	1.87

Table 5.2: Resizing Proof Generation Timing

Original Size	Resized Size	Setup Time (min)	Proof Gen Time (min)
6632 x 4976	2048 x 1365	5.71	3.25

Table 5.3: Grayscale Conversion Timing

Image Size	Setup Time (min)	Proof Gen Time (min)
2048 x 1365	1.25	1.10

Table 5.4: Blurring Proof Generation Timing

Image Size	Blur Region	Setup Time (min)	Proof Gen Time (min)
2048 x 1365	529 x 529	1.67	1.23

Table 5.5: Brightness adjustment Timing

Image Size	Setup Time (min)	Proof Gen Time (min)
2048 x 1365	3.4	3.18

Table 5.6: Contrast adjustment Timing

Image Size	Setup Time (min)	Proof Gen Time (min)
2048 x 1365	3.99	2.87

Table 5.7: Sharpness Enhancement Timing

Image Size	Setup Time (min)	Proof Gen Time (min)
2048 x 1365	5.75	3.85

Table 5.8: Jpeg Conversion Timing

Image Size	Setup Time (min)	Proof Gen Time (min)
2048 x 1365	20.63	12.81

Above results shows the setup and proof generation times for different editing operations. Overall, setup and proof generation take only a few minutes. Since proofs are generated once by the news organization, these times are practical for real-world use.

Verification takes approximately 2 seconds in a browser. Plonky2 proofs are about 100–200 KB, which is negligible compared to edited photos (typically around 8 MB). Moreover, Plonky2 proofs can be further compressed via a constant-sized zkSNARK (e.g., Groth16 or PLONK) that attests to the correctness of Plonky2 verification.

Hashing Scheme Comparison

Table 5.9 compares the time and memory usage of different hashing and polynomial commitment schemes when applied to a 30 MP image.

Table 5.9: Performance comparison of hash functions for zero-knowledge proof systems

Hash Function	Proof Time (ms)	Verify Time (ms)	Prover Memory (MB)
SHA-256	1470	472	158
Poseidon	1507	1329	163
MiMC	1520	1280	190
Pedersen	1552	1275	533
Keccak-256	11933	1426	1601

Table 5.9 shows that computing polynomial commitments (FRI-PCS and KZG-PCS) takes significantly longer and uses more memory than simple hashes (SHA256 and lattice-based). This makes it infeasible for computationally constrained signers like cameras to compute these commitments.

In particular, for a camera to compute a KZG commitment on a 30 MP photo, it would require access to a large structured reference string (SRS) and significant storage and RAM to handle polynomials of degree up to 3×10^7. Similarly, FRI-PCS requires approximately 20 GB of RAM to efficiently perform FFTs, which far exceeds typical camera hardware capacity.

5.2 Related Work

The rationale for employing VC in the context of media provenance is to generate secure proofs of authentic image refinements without requiring trust in the prover. By opening a commitment that binds the image to its original source (e.g., using a hash), we prove the authenticity without revealing the original image. Additionally, each proof can compute a new commitment for the transformed image, allowing for chained proofs and reduced veri-

fication complexity, albeit with a significant increase in prover complexity. Even without commitment generation for the transformed image, the high prover complexity in most of the related work posed a serious challenge for practical adoption.

Photoproof [44] pioneered this idea by demonstrating the usability of cryptographic proofs in general for verification of operations like crop, flip, or adjustments to contrast and brightness. However, their underlying proof system was not efficient enough and as a result, their experiments were limited to low resolution images of only 128×128, which is not practical in real world. A subsequent study in 2022, ZK-IMG [33] utilized the Halo2, which is a more efficient method of achieving SNARKs based on Plonky2 [1] proof system. As a result, ZK-IMG was able to achieve higher efficiency and was able to generate complete proofs in HD resolution (1280×720). The main drawback of works like [33] is that the entire original and transformed images are given to the proof system, such as Halo2, at once. This means that in practice, the prover must execute a large amount of arithmetic in one round of proving, resulting in very high memory complexity. In addition to lower memory consumption, zkIPV achieves up to 3× faster proving times compared to [24].

Chapter 6

Discussion and Future Work

6.1 Discussion and Future Work

This paper showcases the practicality of generating proofs for valid image manipulations using folding-based zkSNARKs. We present zkIPV, an open-source platform that can efficiently operate on consumer-level hardware. It can prove the transformations on 8K (33MP, i.e., 100MB) images under 10 minutes, reaching to a peak memory of only 30 GB, while proof size is just around 10 KB and verification time is under 1 second. zkIPV's succinct proofs enable the development of a privacy-preserving, trustless marketplace for authentic media. Appendix F provides further design details for such a marketplace. As a complete proof system, zkIPV proves the integrity of both the original and edited images, as well as the correctness of the transformation without revealing intermediate images within a chain of edits—only the final result is disclosed. We further introduce a privacy-preserving option to the protocol, where the original image owner's identity remains confidential, while still enabling a trustless authenticity check of the signer.

zkIPV, similar to C2PA and other VC-based methods, assumes a trustworthy signature on the original image. However, zkIPV extends C2PA by eliminating the need for trusted editors, enabling easier integration into existing frameworks, while guaranteeing editor and original signer's privacy. As zkIPV is built on recursion schemes, it is particularly suited for recursion-friendly transformations, where the final value of the resulting image is determined by neighboring pixel values. Other types of image manipulation like affine transformations or rotations, which involve shifting pixel positions, can be expensive or incompatible with zkIPV's tiled commitment scheme. Thus, a potential future direction is to explore efficient recursion-based mechanisms that support such transformations. Currently, zkIPV supports a limited set of recursion-friendly transformations, but this

can be expanded to include more configurability with additional hyperparameters, such as dynamic convolution kernels, to accommodate arbitrary global image effects.

Future work may also incorporate advancements in recursion-based, some of which have already been integrated into the underlying Nova protocol [8]. Finally, optimizing zkIPV for greater efficiency could make it more suitable for deployment on mobile devices.

6.1.1 Extensions and Limitations

The description of zkIPV given here does not protect the identity of the signer (the photographer). Indeed, the verification key (and signature in certain modes) are sent along with the ZK proof to the verifier. If the editor wants to hide the identity of the signer, then the editor could replace the verification key (and signature) by a public commitment to those values, and move the verification key and signature to the zk-SNARK secret witness. The zk-SNARK circuit would then verify that the commitment is a valid commitment to the verification key and signature, instead of directly using those values as public inputs. This fully hides the verification key and signature from the verifier and protects the identity of the signer.

This paper has primarily discussed how to prove edits for photos, but videos are also a major source of misinformation. The main challenge with videos is that once they are edited, they are stored in a lossy compressed format. Directly applying the techniques discussed here to videos would thus require us to prove statements about video compression in a SNARK, which is challenging due to the size of a video file. Another avenue for future research is exploring different kinds of range proofs. In our work, we used folding-based range proofs. More recent methods may lead to time and memory savings for the editor.

6.2 C2PA Compatible Marketplace

Assuming the original image remains untampered and is signed with a trusted and authorized key linked to a real or organizational entity[1], we propose an approach that eliminates the need for pre-registration of the original image before publishing the edited version. To achieve this, we force any editor to prove their knowledge of specific transformations performed on an authentic original source, resulting in the final refined image.

Figure 6.1 provides an overview of the proposed protocol. There are two potential scenarios for submitting a new edited version of an original source:

[1]We assume the existence of a trusted certificate authority or public key infrastructure that can validate the authenticity of signing keys.

1. **Sender has direct rights to the original image:** In this case the smart contract verifies the proof and finalizes the transaction accordingly.
2. **Sender lacks rights to the original image:** The sender must additionally provide a signed declaration from the entity holding the rights to the original image to confirm ownership transfer.

In the proposed model, each original content can have only one owner, implying that all edited versions of an original image belong to a single owner. In our ownership model, all edited versions of an image reference the original image, and the original image points to its owner. This structure ensures that the cost of ownership transfer remains independent of the number of edited versions, as only one storage field in the contract needs updating[2].

Another consideration in the protocol design is that the percentage of photos taken by artists that ultimately get published is typically low. Therefore, registering every unedited image to the blockchain before editing and finalizing is not a scalable solution. To address this, we propose a method allowing honest users to register commitments to an original image simultaneously with submitting the finalized edited version.

The protocol operates as follows:

1. **Original Image Creation:** A trusted entity (verified organization or capable camera) creates an original image α and signs it with their private key, producing signature sig_α.
2. **Image Editing:** An editor or photographer processes the original image α through a series of transformations T_F to produce the refined image β. The editor generates a zero-knowledge proof Π that demonstrates knowledge of the valid transformations without revealing the original image.
3. **Smart Contract Submission:** The editor submits to the smart contract:
   - Commitment outputs $C_{out} = h_\alpha, h_\beta, \ldots$ (hash values of original and refined images)
   - Zero-knowledge proof $\Pi = List(\alpha, T_F, C_{in}, C_{out})$
   - Signature sig_w and optionally sig_{owner} for ownership transfer
4. **Verification and Storage:** The smart contract verifies the proof and stores:
   - In `verified_originals`: mapping from h_α to $pubkey_{owner}$
   - In `verified_edits`: mapping from h_β to h_α

[2]This approach scales efficiently as the number of edits increases, maintaining constant-time ownership operations.

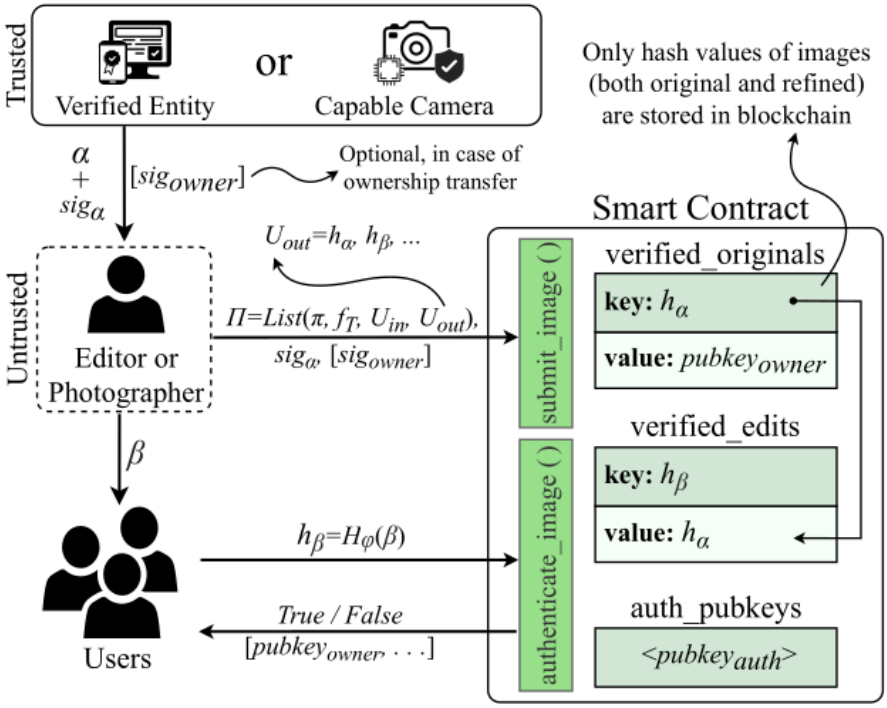


Figure 6.1: Overview of the C2PA Compatible Marketplace Protocol. The system enables trustless verification of image authenticity and ownership through smart contracts, eliminating the need for trusted third-party editors while maintaining compatibility with existing C2PA standards.

- In `auth_pubkeys`: authorized public keys

5. **Public Verification:** Users can verify the authenticity of image β by computing $h_\beta = H_0(\beta)$ and querying the smart contract, which returns the ownership information and verification status.

This approach provides several key advantages:

- **Trustless Operation:** No need to trust third-party editors or intermediaries
- **Privacy Preservation:** Original images and intermediate edits remain private
- **Scalability:** Only hash values are stored on-chain, minimizing storage costs
- **Efficient Ownership Transfer:** Ownership changes require updating only one storage field
- **C2PA Compatibility:** Integrates with existing content provenance standards

6.2.1 Key Innovations and Contributions

Our key innovations are two-fold: first, we introduce a new SNARK-friendly hashing method that reduces the hash proof generation time. We believe this SNARK-friendly method, may be of independent interest to those looking to create SNARKs that prove hashes of large amounts of data. Additionally, we introduce a polynomial commitment hash that completely eliminates the need for proving knowledge of a valid signature in the SNARK circuit. However, signing the unedited image using a polynomial commitment hash is more expensive than the poseidon hash scheme.

In summary, we have demonstrated how to use zk-SNARKs to enable practical provenance verification for realistically large edited images in online news articles. Our system uses signing keys embedded in cameras as the origin of trust, but rather than trusting a third-party application to digitally sign edited images, we propose to use zero-knowledge proofs to prove to a news reader that an edited published photo was taken when and where the article claims it was taken. We create proofs for 30 MP images, which is the size of images produced by actual cameras equipped with embedded signing keys. The bottleneck in image editing proof systems is proving knowledge of a valid signature on the unedited photo, which our innovations successfully address.

Bibliography

[1] plonky2. Available at: https://github.com/0xPolygonZero/plonky2.

[2] Dall•e. https://openai.com/research/dall-e, 2023. Accessed: 2023-10-04.

[3] Deepfacelab library. https://github.com/iperov/DeepFaceLab, 2023. Accessed: 2023-10-05.

[4] Midjourney. https://www.midjourney.com/home/, 2023. Accessed: 2023-10-04.

[5] Reducing the verification cost of a snark through hierarchical aggregation. https://ethresear.ch/t/reducing-the-verification-cost-of-a-snark-through-hierarchical-aggregation/5128, 2023. Accessed: 2025-07-10.

[6] Starkware. https://starkware.co/, 2023.

[7] Coalition for content provenance and authenticity (c2pa). https://c2pa.org/, 2024. Accessed: 2024-01-03.

[8] nova-scotia. https://github.com/nalinbhardwaj/Nova-Scotia, 2024. Accessed: 2024-01-03.

[9] Stability ai. https://stability.ai/home, 2024. Accessed: 2024-01-02.

[10] Truepic: Secure content transparency with c2pa. https://truepic.com/, 2024. Accessed: 2024-01-03.

[11] Adobe. Image size and resolution, 2024. Available at: https://helpx.adobe.com/photoshop/using/image-size-resolution.html.

[12] Associated Press. Visuals. *Associated Press*, 2022. Available at: https://www.ap.org/about/news-values-and-principles/telling-the-story/visuals.

[13] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In E. Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.

[14] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In I. Chatzigiannakis, C. Kaklamanis, D. Marx, and D. Sannella, editors, *ICALP 2018*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl, Jul 2018.

[15] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza, and N. P. Ward. Aurora: Transparent succinct arguments for r1cs. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.

[16] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Scalable zero knowledge via cycles of elliptic curves. *Journal of Cryptology*, 79:1102–1160, 2014.

[17] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In S. Goldwasser, editor, *ITCS 2012*, pages 326–349. ACM, Jan 2012.

[18] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. https://eprint.iacr.org/2012/095, 2012. Cryptology ePrint Archive.

[19] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, Aug 2004.

[20] D. Boneh and D. M. Freeman. Homomorphic signatures for polynomial functions. In K. G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 149–168. Springer, Heidelberg, May 2011.

[21] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography*. 2023. Available at: https://toc.cryptobook.us/book.pdf.

[22] Sean Bowe, Jack Grigg, and Daira Hopwood. Recursive proof composition without a trusted setup. https://eprint.iacr.org/2019/1021, 2019. Cryptology ePrint Archive.

[23] T. Datta and D. Boneh. Using zk proofs to fight disinformation. *Medium*, 2022. Available at: link.

[24] Trisha Datta, Binyi Chen, and Dan Boneh. Veritas: Verifying image transformations at scale. *Cryptology ePrint Archive*, 2024.

[25] D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig. A general framework for redactable signatures and new constructions. In S. Kwon and A. Yun, editors, *Information Security and Cryptology - ICISC 2015*, pages 3–19, Cham, 2016. Springer International Publishing.

[26] A. Gabizon and Z. J. Williamson. Plookup: A simplified polynomial protocol for lookup tables. *Cryptology ePrint Archive, Paper 2020/315*, 2020. Available at: https://eprint.iacr.org/2020/315.

[27] A. Gabizon, Z. J. Williamson, and O. Ciobotaru. Plonk: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *Cryptology ePrint Archive, Paper 2019/953*, 2019. Available at: https://eprint.iacr.org/2019/953.

[28] Paul Glynn. Sony world photography award 2023: Winner refuses award after revealing ai creation. https://www.bbc.com/news/entertainment-arts-65296763, 2023.

[29] S. Gorbunov, V. Vaikuntanathan, and D. Wichs. Leveled fully homomorphic signatures from standard lattices. In R. A. Servedio and R. Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, Jun 2015.

[30] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Deepfake detection by analyzing convolutional traces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 666–667, 2020.

[31] Intel Corporation. Intel® 64 and ia-32 architectures software developer manuals. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html, 2024.

[32] R. Johnson, D. Molnar, D. Song, and D. Wagner. Homomorphic signature schemes. In B. Preneel, editor, *Topics in Cryptology — CT-RSA 2002*, pages 244–262, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[33] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun. Zk-img: Attested images via zero-knowledge proofs to fight disinformation. *arXiv preprint arXiv:2211.04775*, 2022.

[34] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *Advances in Cryptology-ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings 16*, pages 177–194. Springer, 2010.

[35] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. Nova: Recursive zero-knowledge arguments from folding schemes. In *Annual International Cryptology Conference*, pages 359–388. Springer, 2022.

[36] Leica. Partnership for greater trust in digital photography: Leica and content authenticity initiative. *Leica*, 2022. Available at: link.

[37] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. Celeb-df: A large-scale challenging dataset for deepfake forensics. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3207–3216, 2020.

[38] Dmitry Lubarov. Adding zero knowledge to plonk-halo. https://mirprotocol.org/blog/Adding-zero-knowledge-to-Plonk-Halo, 2020. Accessed: 2025-07-10.

[39] Christian Mainka, Vladislav Mladenov, and Simon Rohlmann. Shadow attacks: Hiding and replacing content in signed pdfs. In *NDSS*, 2021.

[40] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)*, 54(1):1–41, 2021.

[41] Vladislav Mladenov, Christian Mainka, Karsten Meyer zu Selhausen, Martin Grothe, and Jörg Schwenk. 1 trillion dollar refund: How to spoof pdf signatures. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1–14, 2019.

[42] Jens Müller, Fabian Ising, Vladislav Mladenov, Christian Mainka, Sebastian Schinzel, and Jörg Schwenk. Practical decryption exfiltration: Breaking pdf encryption. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29, 2019.

[43] Jens Müller, Dominik Noss, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. Processing dangerous paths. In *Network and Distributed Systems Security Symposium*. NDSS, 2021.

[44] A. Naveh and E. Tromer. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 255–271, 2016.

[45] Amna Qureshi, David Megías, and Minoru Kuribayashi. Detecting deepfake videos using digital watermarking. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1786–1793. IEEE, 2021.

[46] Simon Rohlmann, Vladislav Mladenov, Christian Mainka, and Jörg Schwenk. Breaking the specification: Pdf certification. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1485–1501. IEEE, 2021.

[47] Mike Schroepfer. Creating a data set and a challenge for deepfakes. *Facebook artificial intelligence*, 5, 2019. Available at: https://ai.meta.com/blog/deepfake-detection-challenge/.

[48] Sony. Sony unlocks in-camera forgery-proof technology. *Sony*, 2022. Available at: https://www.sony.eu/presscentre/news/sony-unlocks-in-camera-forgery-proof-technology.

[49] Junke Wang, Zuxuan Wu, Wenhao Ouyang, Xintong Han, Jingjing Chen, Yu-Gang Jiang, and Ser-Nam Li. M2tr: Multi-modal multi-scale transformers for deepfake detection. In *Proceedings of the 2022 international conference on multimedia retrieval*, pages 615–623, 2022.