Towards Trustless Provenance: A Privacy-Preserving Framework for On-chain Media Verification

Piotr Mikołajczyk
Aleph Zero Foundation /
Jagiellonian University, Doctoral School of Exact and Natural Sciences
piotr.mikolajczyk@doctoral.uj.edu.pl

Parisa Hassanizadeh*
IPPT PAN / Zero Savvy
parisaa.hassanizadeh@gmail.com

Shahriar Ebrahimi*
Zero Savvy
sh.ebrahimi92@gmail.com

Abstract

As generative models continue to evolve, verifying the authenticity, provenance, and integrity of digital media has become increasingly critical—particularly for domains like journalism, digital art, and scientific documentation. In this work, we present a decentralized verifiable media ecosystem for managing, verifying, and transacting authentic digital media using zero-knowledge proofs (ZKPs). Building on VIMz (Dziembowski et al., PETS'25), we extend the framework in three key directions. First, we generalize the model to support arbitrary image regions to achieve selective transformations support such as redaction and regional blurring—features commonly required in privacy-preserving applications. Second, we introduce performance optimizations that yield up to an 18% improvement in off-chain proof generation, and enhance the framework to support cost-efficient on-chain verification. Third, we design and implement a modular smart contract architecture to support a wide range of decentralized media applications. As a flagship use case, we develop a decentralized media marketplace that enables permissionless licensing, ownership transfer, and verifiable attribution. In this setting, users can share transformed media—such as cropped, blurred, or resized previews—alongside ZKPs that prove derivation from a signed original, eliminating the need to trust the seller. Unlike prior fair exchange protocols, which rely on trusted descriptions or encrypted payload delivery, our system enables verifiable public previews and origin-bound proofs without revealing the full content. This approach unlocks new applications beyond marketplaces, including automated infringement dispute resolution and photography contests with verifiable criteria.

Keywords

zkSNARKs, Media Provenance, Decentralized Marketplace.

1 Introduction

Every day, we encounter a wide range of information: photos, videos, and recordings. Verifying the authenticity of such content is often extremely time-consuming or outright infeasible. Despite significant efforts and resources dedicated to fact-checking—through initiatives like Snopes [36], PolitiFact [37], the Reuters Fact Check Team [61], or the Google News Initiative [46]—these methods remain insufficient. Moreover, false claims can rapidly gain traction and mislead large audiences before they are debunked.

Visual media, including photos and videos, present an especially challenging problem. As highly visual beings, humans are particularly susceptible to manipulation in this format, making it a great target for abuse. We identify three primary sources of concern: (1) The rise of tools for generating hyper-realistic visual content. Face swapping, deepfakes, and similar techniques are now widely accessible—often for free—and can run without high-end hardware. These tools have already been used in scams, propaganda, defamation, and political disinformation [41, 63]. (2) The reuse of genuine images or videos in misleading contexts [24, 58]. (3) Subtle (AI-assisted) edits that alter the semantic meaning of an image by adding, removing, or modifying elements [26, 47].

These problems are not confined to social media or news platforms; they extend to legal evidence, political communication, and even medical documentation.

1.1 Existing Solutions and Their Limitations

A variety of countermeasures have been developed in response to the growing concerns around media authenticity. Fact-checking platforms, collaborative journalism efforts, and detection algorithms based on machine learning (ML) have all been actively explored in recent years [5, 31, 36, 37, 44, 46, 50, 61, 62, 65, 67]. Likewise, technology standards such as C2PA [12] aim to formalize provenance metadata and trace edit history. These approaches definitely deserve recognition for their ambition and utility. However, some of their limitations are still significant.

Manual fact-checking is inherently slow and does not scale. On the other hand, ML models—despite potential concerns regarding their accuracy—function as black boxes and do not provide verifiable evidence for their outputs. Their predictions do not explain *why* a piece of media is deemed fake, nor do they offer cryptographic guarantees that can be independently audited or preserved. Moreover, these tools address only one side of the problem: identifying forgeries. In contrast, provenance-based approaches attempt to prove authenticity—a fundamentally stronger guarantee, especially in settings where trust must be established proactively.

Efforts like C2PA take a different path by focusing on metadata provenance based on digital signatures. While in general well-designed, they rely on trusted software and hardware environments [12]. This introduces new attack vectors: compromised editors[9, 14, 15], forged signatures, or invalid chains of trust [42, 53]. Moreover, real-world usage is limited: only few tools support metadata-preserving workflows, and many common operations (e.g., screenshots or compression) strip metadata entirely.

A different line of work aims to change the trust-based approach with cryptographic primitives and blockchain protocols. These methods often include mathematical proofs for verifying image transformations (e.g., Photoproof [52], VerITAS [16], VIMz [18]) or authenticated marketplaces[19, 55].

*The author was affiliated with IDEAS NCBR partially during the work on this paper.

These systems show clear advantages: they remove reliance on trusted software, reduce ambiguity, and offer composable guarantees. However, in practice, they are still infeasible to adopt widely. Proof systems often require considerable computation overhead, do not scale well to high-resolution content, or support only limited types of edits. Marketplaces on the other hand, are typically narrowly scoped, focusing on the licensing of individual assets rather than on their broader provenance or content history.

1.2 Our Approach

This work proposes a complementary approach. Building on the foundations of earlier systems (specifically VIMz [18]), we present an architecture that enables not just isolated verification or trading, but a complete ecosystem for handling authentic media and their derivatives. To this end, we extend the VIMz framework in mutiple directions. First, we enhance the model to support traversal over arbitrary image regions, enabling selective transformations—such as redaction and localized blurring—that are essential in privacy-preserving applications. Second, we introduce circuit-level optimizations targeting the commitment computation within the ZK system, resulting in up to an 18% improvement in off-chain proof generation performance compared to the original VIMz protocol.

Most importantly, while the authors in [18] outlined potential on-chain applications, these ideas remained largely conceptual and theoretical. We take the next step by realizing a fully working prototype. Importantly, we do not restrict ourselves to marketplace interactions. Our system supports workflows around attribution, licensing, publishing, contest submissions, collaborative editing and more. It is designed to function autonomously, transparently within public infrastructure and verifiable logic at every layer.

Furthermore, our framework can serve as a foundation for other systems. For example, FairSwap [19] and similar protocols could integrate our provenance guarantees into their logic, or layer payment enforcement on top of our license tracking and verification components. In this way, our work contributes not only a toolset but also a building block for broader applications.

1.3 Contributions Summary

Our main contributions are as follows:

- We generalize the VIMz model to operate over arbitrary image regions, rather than being limited to row-based traversal. This enhancement increases the protocol's flexibility, enabling more precise and customizable transformations.
- As a proof of concept, we introduce *redaction*, a transformation operating on square blocks, useful in legal, journalistic, and privacy-sensitive applications.
- We enhance prover-side efficiency by optimizing the internal circuit architecture. In particular, we evaluate alternative strategies for performing commitment computations within the ZK circuit and achieve up to 18% improvement in proof generation time.
- We integrate the VIMz protocol with the Sonobe library [1], enabling efficient on-chain verification of high-resolution image transformations.
- We design and implement a modular smart contract architecture that supports on-chain provenance tracking and media licensing. The architecture accommodates diverse use cases, including content marketplaces, verifiable provenance, and privacy-preserving media sharing. Additionally, we analyze the security properties of the system and provide a detailed evaluation of the on-chain gas costs associated with each phase of the protocol.

The rest of the paper is organized as follows. Section 2 briefly recalls the fundamental cryptographic concepts relevant to our work. In Section 3, we redefine the VIMz protocol along with our proposed generalized extension of it. Section 4 presents the design of an on-chain ecosystem for authenticated media. Sections 5 and 6 analyze the system's security and describe its implementation details respectively. Finally, Section 7 reviews related work in the context of verifiable media and fair exchange protocols, and Section 8 concludes with a summary and discussion of future directions.

2 Background

This section provides an overview of the main concepts employed to build VIMz. We generally borrow the mathematical representations from [40].

2.1 SNARKs

A **[zk]SNARK** ([Zero-Knowledge] Succinct Non-Interactive Argument of Knowledge) is a cryptographic proof system defined by the following properties:

- **[Optional]Zero-Knowledge**: The proof does not leak any information beyond the validity of the statement.
- **Succinctness**: The proof is short and can be verified in time significantly less than what would be required to recompute the original statement.
- **Non-Interactivity**: The protocol requires only a single message from the prover to the verifier, eliminating the need for back-and-forth communication.
- **Argument of Knowledge**: The proof ensures that, if the verifier accepts, then a computationally bounded prover (modeled as a probabilistic polynomial-time (PPT) algorithm) must possess a valid witness (i.e., secret input) corresponding to the public statement.

Definition 2.1. Formally, a zkSNARK is a tuple of polynomial-time algorithms $(\mathsf{Gen}, \mathsf{Prove}, \mathsf{Verify})$ and a deterministic encoder $\mathcal{K}$ defined as follows:

- $\mathsf{Gen}(1^\lambda) \to pp$: A key generation algorithm that takes a security parameter λ and outputs a public parameter pp.
- $\mathcal{K}(pp, s) \to (\mathsf{pk}, \mathsf{vk})$: A deterministic key generation algorithm that takes a public parameter pp and a statement s for defining a specific relation R, and outputs a proving key pk and a verification key vk.
- $\mathsf{Prove}(pp, \mathsf{pk}, x, w) \to \pi$: A proving algorithm that takes the proving key pk, public input x, and witness w, and outputs a proof π, indicating that (x, w) indeed satisfy the relation R through the statement s.
- $\mathsf{Verify}(pp, \mathsf{vk}, x, \pi) \to \{0, 1\}$: A verification algorithm that takes the verification key vk, public input x, and proof π, and returns a boolean indicating whether the proof is valid.

We consider a proof system to be a zk-SNARK if it satisfies the following three properties (Formal definitions in Appendix A).

- **Completeness**: If the prover is honest and possesses a valid witness, then the verifier should accept the proof.
- **Knowledge Soundness**: A malicious prover should not be able to convince the verifier of a false statement.
- **Zero-Knowledge**: A proof is said to be zero-knowledge if it conveys no information beyond the validity of the underlying statement.

2.2 Folding-based zk-SNARKs

Folding schemes are a class of cryptographic primitives designed to enable incrementally verifiable computation (IVC). They allow the verification of computations that result from repeated applications of a function. More precisely, given a function f and an initial input z_0, a folding scheme enables the generation of a proof Π_i attesting that $z_i = f(z_{i-1}) = f^i(z_0) = f(f^{i-1}(z_0))$, assuming the availability of a prior proof Π_{i-1} asserting $z_{i-1} = f^{i-1}(z_0)$.

In such a system, the successful verification of each step ensures, with overwhelming probability $(1-\text{negl}(\lambda))$, that the following property holds for all $i \in [n]$:

$$z_i = f^i(z_0).$$

Nova [40] was the first folding scheme to realize this approach. It enables a prover to incrementally construct proofs of correct execution for sequential computations of the form $y = F^l(x)$, where F denotes the computation, x is the initial input, and $l > 0$ indicates the number of function applications, or computational steps.

The `Nova-rust` library [49] provides a complete implementation of the Nova proving system and integrates with Spartan [64] to verify the compressed output of IVC, enabling the construction of compact SNARK proofs. The original `vimz` [18] was built using `Nova-rust` along with the `nova-scotia` library [33], which bridges Nova with the Circom [8] frontend. In our extension, we incorporate support for `Sonobe` [1] to further reduce final proof size, thereby enhancing suitability for on-chain verification.

3 Generalization of the VIMz

This section revisits and generalizes the original VIMz protocol [18]. We begin by identifying and formalizing the general use case that serves as the primary motivation for this work. We then recall the underlying trust assumptions and adversary model. Next, we introduce the *regionalizer*, which extends the protocol's flexibility. We redefine the VIMz protocol by presenting a formal definition, followed by examples of concrete instantiations and conclude with a discussion on how the protocol preserves privacy.

3.1 Motivation

The primary objective of the VIMz protocol is to enable secure and sound proofs that a given image results from applying a specific transformation to another image. Crucially, the protocol must support privacy-preserving features, such as hiding the transformation parameters or the source image itself, exposing only a public commitment. In some cases, it may also be necessary to prove properties of the original image, such as its capture time or author, in the spirit of the C2PA initiative [12].

Table 1: Terminology and notation

Notation	Description
α, β	Pixel matrices representing the *original* and *transformed* image, respectively.
f	Transformation function (e.g., grayscale, crop).
ψ	Parameters of the transformation f applied to α.
meta	Metadata associated with α, such as capture time or location.
device	Verified device (identified by a public key) used to capture α.
sig	Digital signature produced by *device* over $(\alpha, \textit{meta})$.
Φ	Predicate that must be satisfied by $(\alpha, \textit{meta})$ (e.g., capture time or author constraints).
C	Commitment function that is binding and hiding.
$C(\gamma), C_\gamma$	Actual and claimed commitment to data γ.
$\mathcal{D}$	Public set of approved device public keys.
$\mathfrak{R}$	Regionalizer - an algorithm that splits an image into regions for transformation.
r_i	i-th region obtained from applying `Split` to image.
$\mathcal{R}_{tran/auth}$	Subrelations capturing image transformation and authenticity verification, respectively.
$\mathcal{R}_{\text{VIMz}}$	The full VIMz relation: $\mathcal{R}_{tran} \wedge \mathcal{R}_{auth}$.

The motivating statement behind our construction can be summarized as follows: *"c_β is a commitment to an image obtained by applying a transformation f with hidden parameters to a source image with commitment c_α, where the source image was captured by a verified device and satisfies property Φ."* The only public information is the commitments, the transformation f, and the property Φ.[1]

Several real-world use cases illustrate the relevance of this model. For example, digital news platforms could use such a protocol to prove that a cover image was produced by cropping and sharpening an original photograph taken by the article's author, at a verifiable time and place. This would enhance reader trust without revealing the original image or author identity. Similarly, in online marketplaces for real estate or vehicles, sellers could be restricted to submitting images captured by verified devices and only minimally edited (e.g., cropped or resized). Additionally, requirements such as recency of capture or geographical location could be enforced without disclosing the seller's identity or precise location, thereby preventing fraud while maintaining privacy.

Finally, although we primarily refer to verified devices such as cameras or smartphones, the concept generalizes to any verifiable source of images, including generative AI models. Moreover, while our focus is on still images, extending the protocol to other media types such as video or audio remains an interesting direction for future work.

3.2 High-level Overview of the Protocol

Table 1 summarizes the key symbols and notational conventions used throughout this section. These definitions will be referenced in the formalization of the protocol below.

Let *device* be a verified device, meaning that: (1) it belongs to a publicly known set of admissible devices $\mathcal{D}$; (2) it can asymmetrically sign arbitrary data and the signature can be verified by anyone using a well-known public key. For simplicity, we identify devices with their public keys and assume that $\mathcal{D}$ is a set of public keys.

Let α be the original image captured by *device*, accompanied by metadata *meta* (such as capture date, location, etc.). Let *sig* denote the signature generated by *device* over the tuple $(\alpha, meta)$. Let Φ be a predicate specifying a required property of $(\alpha, meta)$, such as verifying the author's identity. Let f be a transformation function applied to α with parameters ψ, yielding a new image $\beta = f(\alpha, \psi)$. Let C denote a binding and hiding commitment scheme, and let $C(x)$ represent a commitment to data x.

The VIMz protocol enables proving the following statement using public inputs $C_\alpha, C_\beta, f, C_\psi, \Phi$, and $\mathcal{D}$:

$$\exists_{\alpha,\beta,\psi,device,meta,sig} \text{ such that } \begin{cases} C_\gamma = C(\gamma) \text{ for } \gamma \in \{\alpha, \beta, \psi\} \\ \beta = f(\alpha, \psi) \\ \Phi(\alpha, meta) \text{ is satisfied} \\ sig = device(\alpha, meta) \\ device \in \mathcal{D} \end{cases} \tag{1}$$

Moreover, the protocol is zero-knowledge, meaning that the proof reveals no information about the prover's private data, i.e., $(\alpha, \beta, \psi, device, meta, sig)$. To clarify the structure of the proof, we decompose the main relation $\mathcal{R}_{\text{VIMz}}(\alpha, \beta, \psi, device, meta, sig;\ C_\alpha, C_\beta, f, C_\psi, \Phi, \mathcal{D})$ into two simpler subrelations:

$$\mathcal{R}_{tran} = \{(\alpha, \beta, \psi;\ C_\alpha, C_\beta, f, C_\psi) : \begin{cases} C_\gamma = C(\gamma) \text{ for } \gamma \in \{\alpha, \beta, \psi\} \\ \beta = f(\alpha, \psi) \end{cases} \} \tag{2}$$

$$\mathcal{R}_{auth} = \{(\alpha, device, meta, sig;\ C_\alpha, \Phi, \mathcal{D}) : \begin{cases} C_\alpha = C(\alpha) \\ \Phi(\alpha, meta) \text{ is satisfied} \\ sig = device(\alpha, meta) \\ device \in \mathcal{D} \end{cases} \} \tag{3}$$

The second relation, $\mathcal{R}_{auth}$, is relatively straightforward and can be constructed from several independent components:

- Firstly, the substatements $C_\alpha = C(\alpha)$, $\Phi(\alpha, meta)$, and $sig = device(\alpha, meta)$ can each be proven using a standard ZKP system such as Groth16 [32] or Spartan [64], assuming that both the commitment scheme and the signature scheme are SNARK-friendly. For instance, Poseidon[30] can be used as the hash function for commitments, and ElGamal[21] for digital signatures.
- The condition $device \in \mathcal{D}$ can be proven via a standard ZK membership proof, using either Merkle tree inclusion[48] or accumulators[45].

[1]While it is technically possible to also hide the transformation itself (by encoding all admissible transformations into a disjunctive circuit) we do not pursue this approach for efficiency reasons and thus focus on the more constrained version above.

The primary challenge lies in proving the first relation, $\mathcal{R}_{tran}$. Establishing that β results from applying f with parameters ψ to α in a single monolithic circuit is typically highly inefficient, especially for images of high-quality resolution.

This is where the VIMz protocol provides a key advantage: it leverages a folding scheme to decompose the transformation into multiple independent operations over subregions of the image. In this approach, the prover iterates over each subregion and proves that the transformation applied to that region satisfies the expected constraints. These per-region proofs are then folded into a single compressed zkSNARK, which verifies that all local constraints across the image hold globally.

A key benefit of this design is that the memory footprint during proof generation is bounded by the size of a single subregion, rather than the entire image. This makes the protocol more scalable for high-resolution media. Additionally, the folding proof itself is significantly more efficient to compute than a monolithic zkSNARK over the full image, resulting in shorter proof generation times.

However, a notable limitation of the original VIMz protocol is its reliance on a row-by-row traversal, which is suboptimal for many transformation types, such as redaction or selective area blurring. In the remainder of this section, we present an extension of the protocol that supports arbitrary traversals, enabling more expressive and transformation-specific partitioning strategies.

3.3 Regionalizer

To formally capture the idea of splitting an image into regions, we introduce the concept of a *regionalizer scheme*.

Definition 3.1 (regionalizer)**.** Let α be an image and C a commitment scheme. A regionalizer (scheme) $\mathfrak{R}$ of order n is a triple of algorithms:

- `Split`, which converts a single image into a sequence of n *regions*: $\texttt{Split}(\alpha) = [r_i]_{i \in \{1,\ldots,n\}}$
- `Restore`, which reconstructs the original image from its regions: $\texttt{Restore}(\texttt{Split}(\alpha)) = \alpha$
- `Commit`, which returns a commitment to the image computed by sequentially folding commitments to the regions: $\texttt{Commit}(\alpha) = C(\ldots C(C(r_1), C(r_2)) \ldots, C(r_n))$

The essential component is the `Split` operation. The purpose of `Restore` is to ensure that the split is lossless. The function `Commit` serves as a convenient shorthand for aggregating regional commitments.

In the context of image transformations, several natural choices for $\mathfrak{R}$ exist:

- **Row-by-row regionalizer** ($\mathfrak{R}_{\text{row}}$): splits an image of height n into n rows.
- **Column-by-column regionalizer** ($\mathfrak{R}_{\text{col}}$): splits an image of width n into n columns.
- **Block regionalizer** ($\mathfrak{R}_{\text{block}}$): splits the image into n disjoint blocks of equal size.
- **Pixel regionalizer** ($\mathfrak{R}_{\text{pixel}}$): splits the image into n individual pixels.
- **Kernel regionalizer** ($\mathfrak{R}_{\text{kernel}}$): splits the image into n strips of k adjacent rows, typically used for $k \times k$ kernel-based transformations.

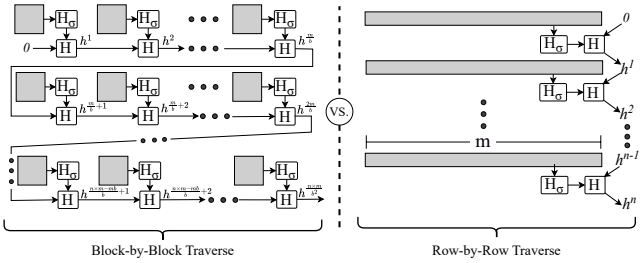

Figure 1: Comparison between `Commit` in row-by-row and block-based regionalizers.

For a visual comparison between the two that currently are used in our implementation, we provide Figure 1.

It is worth noting that `Split` may enrich each region with auxiliary information, such as padding (for convolution operations) or absolute position within the image (e.g., pixel coordinates), which is useful in transformations like cropping.

3.4 Folding-based Transformation

As discussed earlier, the core idea behind VIMz protocol is to split the image into regions and apply the transformation to each region independently. Depending on the transformation type, different regionalizers may be required for input and output images.

For example, the *grayscale* transformation in the original VIMz uses $\mathfrak{R}_{\text{row}}$ of order 720 (for HD resolution) to split both the source and the transformed image. For *resizing* from HD to SD, it uses a regionalizer of order 240 that groups rows in sets of 3 for the original image, and another regionalizer of the same order that groups rows in sets of 2 for the transformed image. For the most recent redaction transformation, we introduced a block-based regionalizer.

We now describe how to implement the relation $\mathcal{R}_{tran}$ in detail.

Algorithm 1: VIMz protocol: $\mathcal{R}_{tran}$

Private inputs: α, β, ψ
Public inputs: $n, \mathfrak{R}_\alpha, \mathfrak{R}_\beta, C_\alpha, C_\beta, f, C_\psi$

$[r_i^\alpha]_{i\in\{1,\dots,n\}} \leftarrow \mathfrak{R}_\alpha.\texttt{Split}(\alpha)$
$[r_i^\beta]_{i\in\{1,\dots,n\}} \leftarrow \mathfrak{R}_\beta.\texttt{Split}(\beta)$
for $i = 1 \rightarrow n$ **do**
  Assert $r_i^\beta = f(r_i^\alpha, \psi)$
Assert $\mathfrak{R}_\alpha.\texttt{Commit}(\alpha) = C_\alpha$
Assert $\mathfrak{R}_\beta.\texttt{Commit}(\beta) = C_\beta$
Assert $C_\psi = C(\psi)$

The for-loop and (unfolded) image commitment checks in Algorithm 1 are implemented using a folding scheme. The IVC state z_i updated at each step consists of a triple: the accumulated commitments of processed regions for both images (acc_α, acc_β) and the transformation parameters ψ.

We assume the following property holds to ensure the correctness of the relation $\beta = f(\alpha, \psi)$:

$$f(\alpha,\psi) = \mathfrak{R}_\beta.\texttt{Restore}\left([f(r_i,\psi)]_{r_i\in\mathfrak{R}_\alpha.\texttt{Split}(\alpha)}\right) \tag{4}$$

This assumption guarantees that applying the transformation f independently to each region and then restoring the output yields the same result as applying f to the entire image at once.

We emphasize that the modifications introduced in this section are purely structural and do not affect the core soundness, ZK, or security guarantees of the original VIMz protocol. All security properties are inherited directly from the original construction, as the underlying cryptographic assumptions, folding scheme, and proof systems remain unchanged. The purpose of this reformulation is to generalize and formalize the protocol interface, without altering its computational or adversarial model. This extension enables precise and selective control over transformations, as opposed to applying a global transformation to the entire image. This feature makes it possible to support additional operations, such as redaction. In particular, each subrelation maintains compatibility with existing proof strategies and circuit implementations from the original design.

4 On-chain Ecosystem

A central element of our ecosystem design is an effective protocol for verifiable image transformations. In particular, we assume that users can generate zero-knowledge proofs of valid image manipulations, which can be efficiently verified on-chain. For this purpose, our implementation employs the VIMz [18] protocol, as described in the previous section. However, we emphasize that the ecosystem abstracts away from any specific underlying mechanism and can be adapted to alternative protocols, such as MicroNova [69].

Compared to existing approaches, our design offers several notable advantages. The original C2PA [12] initiative relies on trusted software (e.g., Adobe Photoshop) and hardware-based infrastructure (e.g., Intel SGX). The protocol's security and integrity fundamentally depend on the trustworthiness of these components. However, both software and TEE-based solutions are known to suffer from critical vulnerabilities [9, 14, 15, 38, 53, 68]. Likewise, blockchain-oriented systems such as Numbers Protocol [57], while leveraging decentralization, still depend on off-chain trusted agents.

In contrast, our design eliminates the need to trust editors by replacing their role (as signers) with verifiable computation. This enables a trustless provenance record of media from the moment it is captured (e.g., by an authenticated camera) or generated (e.g., by a verifiable AI model). Furthermore, by integrating proof verification directly on-chain, our system inherits the transparency and trust guarantees of the underlying blockchain, further reinforcing the integrity and auditability of the provenance trail.

While the present work is an academic proof of concept, we have structured it to reflect real-world use cases as closely as possible. Although some simplifications are included, the design explicitly addresses practical challenges related to digital content licensing, attribution, provenance, and copyright. Overall overview of the architecture is presented in Figure 2. The remainder of this section details the ecosystem's design. We address security and implementation considerations in the subsequent chapters.

4.1 Note on Image Storage

An important aspect of any image authenticity system is how images are stored and made accessible for verification or consumption. In blockchain-based systems, this presents a non-trivial challenge.

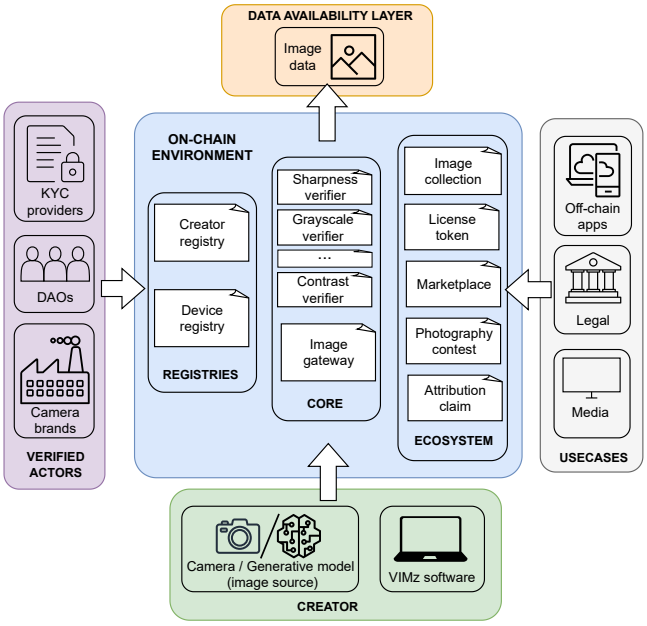

Figure 2: System architecture for the VIMz-powered ecosystem. Verified Actors (left) maintain and populate on-chain registries of creators and devices. At the bottom, these registered creators capture images that are cryptographically signed by their devices and upload the full image data to the off-chain Data Availability Layer. The signed images and metadata pass through the on-chain core - verifiers, an image gateway, and registry lookups to establish authenticated provenance. Finally, on the right, user-facing applications, legal authorities, and media consumers ("Use Cases") leverage the on-chain ecosystem services to trustlessly discover, license, verify, and consume digital assets.

Typically, storing large media files directly on Ethereum Virtual Machine (EVM) ledgers is either impractical or infeasible due to block size limitations and the high cost of on-chain storage.

Consequently, our design assumes the existence of an external data availability layer responsible for storing full-resolution image data and capable of producing verifiable storage proofs. These proofs can be seamlessly integrated into our system. In practice, this allows us to offload the complexity of data storage and retrieval to a third-party services such as IPFS, while interacting solely with short image commitments (hashes).

We also acknowledge that it would be entirely feasible to launch a dedicated EVM-compatible blockchain tailored to the needs of our ecosystem. Such a specialized ledger could natively support large-scale media storage while directly offering availability guarantees. In practice, this could take the form of a use case-specific sidechain or a Layer-2 solution operating atop a general-purpose blockchain such as Ethereum or Polygon, thereby inheriting its security properties.

4.2 Base Concepts, Actors, and Intuitions

As noted earlier, our design is conceptually aligned with the C2PA initiative, where digital media originate from verified devices. Industry leaders such as Leica [43] and Canon [10] already offer camera models compatible with the C2PA standard, enabling the capture of images with embedded metadata. Each original image is accompanied by additional information, typically including photographer identity, camera model, timestamp, and geographic coordinates. The camera cryptographically signs this metadata along with the image itself.

To enable robust authenticity and provenance verification, our ecosystem maintains two core public registries:

- *Device Registry*: a decentralized registry of verified camera manufacturers and their devices. Each authorized brand is responsible for registering the public keys of its devices. These keys are then used to verify signatures and ensure that all images originate from authenticated sources.
- *Creator Registry*: a decentralized registry of verified photographers and other content creators. Supported by a Know Your Customer (KYC) process, it ensures that only verified actors can register original images and their transformations in the system.

These components would typically be governed by decentralized autonomous organizations (DAOs), such as the C2PA committee [12], to ensure fair, transparent, and secure oversight.

Furthermore, our design is flexible enough to accommodate media sources beyond traditional cameras. For example, the device registry could be extended to include generative AI models such as Imagen [29] or DALL-E [54]. Watermarking techniques may also be integrated as complementary or alternative mechanisms for verifying authenticity such as [66].

4.3 License Terms and Edition Policies

Upon registration, every image is assigned three independent attributes. The first attribute defines ownership. By default, two options are supported: either the image has a single owner (represented by a blockchain address), or it is designated as a public good with no owner. However, this model can be extended to accommodate alternative ownership structures. For instance, privacy-enhancing features can be introduced by shielding the owner's identity using zkSNARK-based techniques, commonly applied to financial privacy [13, 28, 59]. The second attribute specifies the edition policy—that is, who is authorized to register edited versions of the image. Three policies are supported:

- *Sealed Policy* – no manipulations are allowed; the image is considered final and immutable.
- *Owner-Only Policy* – only the current owner is permitted to register derivative content (exclusive control).
- *Free Policy* – any verified creator may freely create and register editions.

The third attribute governs commercial usage. The creator can specify whether the image may be used for commercial purposes. If an ownerless image is marked as suitable for commercial use, it may be used freely by anyone. Otherwise, use is subject to compensation in accordance with license terms set by the owner (Section 4.5).

Additionally, the creator may extend the license structure with supplementary terms such as attribution requirements, or credit lines. To simplify system maintenance and ensure consistency, the ownership, edition policy, and commercial usage settings are shared across the original image and all of its registered editions. That is, the same terms apply uniformly across the entire derivation tree.

4.4 Image Gateway Contract

The central component of our ecosystem is the *Image Gateway* smart contract. It facilitates core system operations, including image registration, provenance tracking, and integrity verification. When a new original image is registered, the gateway contract verifies that both the creator and the device are authenticated via their respective registries (see Algorithm 2). While the default interface assumes that identities are passed as public information, a privacy-preserving API can also be supported. In this variant, registry membership is proven using a zero-knowledge argument, such as a Merkle tree inclusion proof [48]. The contract also checks that the image metadata is correctly signed by the device. At this stage, the initial license terms, edition policy, and ownership settings are also recorded.

For edited images, the gateway performs on-chain verification of transformation proofs. In addition, it enforces the applicable edition policy through supplementary checks (see Algorithm 3). The gateway contract also serves as the primary data layer for other components of the ecosystem. It exposes methods for retrieving edition histories, querying and transferring ownership, and accessing license terms.

Algorithm 2: Registering a new image on-chain

Data: h_{img}, π_{avail}, *img_meta*, *license*, pk_{owner}, pk_{device}, sig_{device}

Assert $pk_{owner} \in verified_creators$
Assert $pk_{device} \in verified_devices$
Assert $h_{img} \notin images$
Assert verify_data_availability(h_{img}, π_{avail})
$img_data \leftarrow \langle pk_{owner}, h_{img}, img_meta \rangle$
$pk'_{device} \leftarrow$verify_and_recover_pk(sig_{device}, img_data)
Assert $pk'_{device} == pk_{device}$
Store $images[h_{img}] \leftarrow \langle img_meta, pk_{owner}, pk_{device} \rangle$
Store $rights[h_{img}] \leftarrow \langle pk_{owner}, license \rangle$

4.5 Ownership Trading and Temporal Licensing

Our ecosystem supports multiple options for asset monetization. First, image ownership can be traded through the marketplace smart contract. In its simplest form, the current owner posts a public offer specifying a price. Once a buyer locks the required amount of tokens, ownership is transferred and the payment is forwarded to the seller. However, this approach is vulnerable to issues such as front-running and full transparency, which exposes both parties involved in the transaction. To address these concerns, established techniques from the DeFi space can be employed, such as commit–reveal schemes or shielding trades using zkSNARKs. The model can also be extended to support more complex mechanisms, such as blind auctions.

Algorithm 3: Registering an edited image on-chain

Data: h_α, h_β, π_{avail}, f_T, T_{params}, π_{SNARK}, pk_{editor}

Assert $pk_{editor} \in verified_creators$
Assert $h_\alpha \in images$
Assert $h_\beta \notin images$
Assert verify_data_availability(h_β, π_{avail})
$\langle pk_{owner}, license \rangle \leftarrow rights[h_\alpha]$
Assert edition_license_compatibility(*license*, pk_{owner}, pk_{editor}, f_T)
$vk \leftarrow$ retrieve_SNARK_key(f_T)
Assert verify_proof(vk, π_{SNARK}, h_α, h_β, T_{params})
Store $images[h_\beta] \leftarrow \langle pk_{editor}, f_T, T_{params} \rangle$
Store $rights[h_\beta] \leftarrow rights[h_\alpha]$

Another supported feature is temporal licensing. For assets marked as suitable for commercial use, the owner may define a license pricing scheme. In its basic form, this involves posting a marketplace offer that specifies the cost of usage over a blocktime-based period. Anyone can then acquire a temporary right to commercially exploit the image without requiring a transfer of ownership.

We also support bundling multiple assets into collections. This enables license pricing and acquisition to be defined collectively, which aligns with how online image databases often manage gallery-level access. The full diagram of interactions between system actors and components is depicted in Figure 3.

4.6 Attribution Claim and Infringement Report

We also introduce a basic mechanism to enhance intellectual property and copyright protection. To this end, the ecosystem is extended with an *Attribution Claim* smart contract. This component supports the detection of unauthorized content usage and violations of edition policy. It functions as a bounty-based reporting system, allowing asset owners to fund valid infringement claims. Anyone who identifies a legal violation may submit a report on-chain and, following a successful resolution, receive a reward. In cases involving derivative works of protected assets, the reporter may attach a corresponding proof of infringement.

In practice, such a component would typically be integrated with a legal authority or enforcement agency. Upon a successful and transparent on-chain dispute resolution, the associated legal entity could initiate formal proceedings against the offender.

4.7 Blockchain-Based Photography Contests

As a practical application of the described ecosystem, we propose a fully transparent, blockchain-powered photography contest implemented as a smart contract (see Figure 4). Organizers can define various requirements for submissions. For example, an image may be required to have been captured within the last year, with only cropping and grayscale transformations permitted. Participants must first register their work—including the original image and

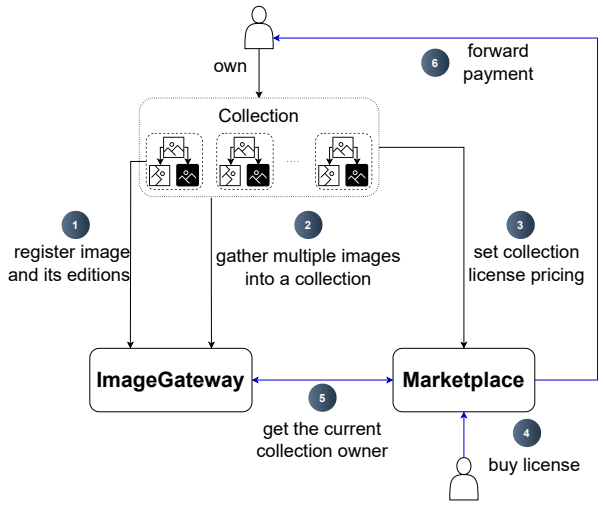

Figure 3: On-chain licensing workflow: originals and their editions are registered (1) in the `ImageGateway`; the owner groups (2) selected images into a `Collection` and assigns (3) a single licence price; a buyer acquires (4) a time-limited licence via the `Marketplace`, which queries (5) the `ImageGateway` to confirm the current owner and forwards (6) the payment. Licensing an individual image is also supported.

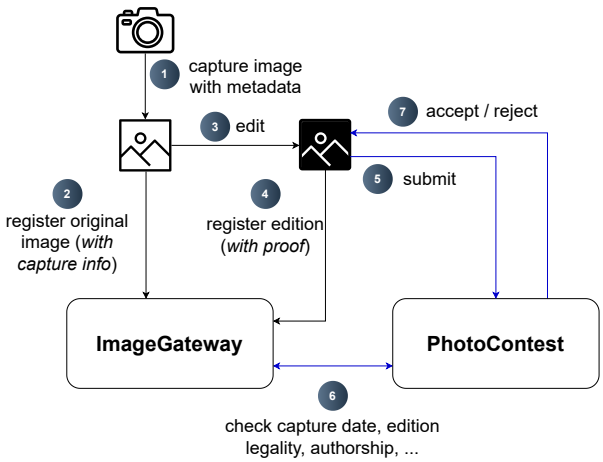

Figure 4: Architecture of the blockchain-based photography contest. Original photos and edits are registered via `ImageGateway`, while the `PhotoContest` contract automatically verifies things like capture date, permitted transformations, geolocation, and related constraints on-chain before judging happens.

any editing steps—through the gateway contract. Only then can a submission to the contest be made.

Submissions are automatically verified against the contest rules. Thanks to the information stored in the gateway contract—including image metadata and full provenance—we can guarantee that all conditions are satisfied. The contest jury can subsequently review verified submissions and declare a winner, ensuring transparency and fairness throughout the process. Upon completion, rewards can be distributed on-chain, enabling seamless, verifiable, and efficient contest management.

We note that this solution has the potential to significantly reduce the operational costs commonly associated with traditional contests. Such events typically require the formation of a dedicated committee responsible for manual submission verification. For instance, prominent competitions like World Press Photo employ external research teams to conduct fact-checking [56]. By automating at least part of this process through transparent on-chain verification, administrative overhead can be reduced and organizational efficiency improved.

5 Security Analysis

The ecosystem proposed in the previous section is already non-trivial. Given its integration within a complex blockchain environment, it is important to consider a broad range of security threats and their potential impact.

Data Availability. As described in Section 4.1, we fully outsource the problem of image storage and retrieval to an external layer. Our system relies heavily on either trust in this component or the availability proofs it can provide. Failures or censorship at this layer can have critical consequences for both security and usability. Such disruptions may not only prevent the addition of new assets but may also make already registered media inaccessible. We note that running a dedicated ledger with native, low-cost storage support might be the most straightforward and robust solution. If this is not feasible, other mitigations can be considered, such as incentivized replication or the use of multiple, redundant data availability layers.

Trusted-Entity Key Compromise. The integrity of the creator and device registries relies primarily on security of signing keys held by their maintainers. Any compromise of these keys—or malicious behavior by a maintainer—can introduce serious vulnerabilities, such as incorrectly verifying a bad actor. A particularly damaging example would be admission of a forged device capable of signing arbitrary data. Such a breach would fundamentally undermine the authenticity guarantees on which the entire system is built and must therefore be considered a critical threat. Recommended mitigations include the use of multisignature governance schemes, such as DAO-based approval processes, as well as regular key rotation policies.

Front-Running and MEV. The marketplace component presents the most attractive target for miner extractable value (MEV). For example, an adversary monitoring the mempool could front-run a pending purchase transaction in order to acquire the asset and resell it immediately at a profit. While our proof-of-concept implementation adopts a simple model that is vulnerable to such attacks, any practical deployment should incorporate protective mechanisms

such as commit–reveal schemes or shielded transaction environments. These also offer the added benefit of enhancing participant privacy.

Blockchain, Storage Spamming, and Denial-of-Service. Our design delegates service availability and denial-of-service (DoS) resilience to the underlying blockchain infrastructure. By relying on a decentralized network and standard security precautions for RPC nodes, the system benefits from inherent resistance to typical network-based attacks. Attempts to inflate storage or disrupt the system through transaction spam would require either substantial computational effort (e.g., for generating SNARKs) or significant financial cost. In addition, per-asset bounds on bids and auctions naturally limit the impact of spam attacks on the marketplace. Overall, such vectors are considered low risk in the context of our system.

Timestamps. All time-sensitive logic—such as KYC expiry, contest submission windows, and license durations—is based on block height or block timestamps. Since block production is neither fully deterministic nor precisely timed, minor timestamp manipulation is theoretically possible. For instance, block producers could attempt to gain an advantage in edge-case scenarios. However, in practice, most blockchains operate with relatively low block intervals, typically measured in seconds. This results in only a negligible margin for manipulation. For use cases requiring finer temporal granularity, one can opt for chains with shorter block times or integrate trusted off-chain time oracles.

Reentrancy Attacks. In our implementation, we conservatively mark all functions that modify contract state and perform external calls as non-reentrant. This precaution prevents malicious contracts from exploiting the ecosystem through double-spending or state corruption vulnerabilities. By adhering to the "checks-effects-interactions" pattern and employing a reentrancy guard, we eliminate this class of attack.

Key Loss. If any actor—such as a camera manufacturer, registry maintainer, verified creator, or asset owner—loses their signing key, it may impact system functionality. While such an event does not constitute a critical threat to security or integrity, it can degrade usability and impose economic costs on the affected party. For example, victim may lose the ability to register new images or claim accrued revenue. We therefore recommend that real-world participants adopt best practices for key management to mitigate the risk of single points of failure. This includes techniques such as social recovery, account abstraction, and use of hardware wallets.

Privacy, Censorship and Inappropriate Content. Since our design operates solely on image hashes (commitments), we cannot inspect or filter the underlying image content. Concerns such as privacy violations, child exploitation, or graphic violence must be addressed at the data availability layer. Only at that level can advanced moderation pipelines and content policies be effectively applied. We intentionally defer responsibility for filtering and censorship to the ecosystem deployer.

Hash Collisions and Key Forgery. Assuming the implementation employs standard cryptographic primitives—such as collision-resistant hash functions (e.g., Poseidon [30] or SHA-3) and secure signature schemes (e.g., ECDSA or EdDSA)—the risk of accidental hash collisions or signature forgery is considered negligible.

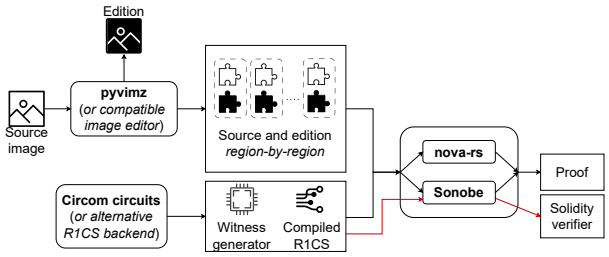

Figure 5: Overview of the VIMz architecture. The pipeline begins with a source image and applies a supported transformation, producing both the edited image and folding-compatible data via `pyvimz`. The folded proof is produced using either `nova-rs`[49] or `Sonobe`[1], based on Circom[8] circuits and can be verified on-chain using a Solidity verifier.

6 Implementation

In this section, we describe the implementation and software architecture that realize the on-chain ecosystem introduced in the previous chapters. The full technical pipeline is presented in Figure 5. While the solution builds upon an earlier version of the VIMz codebase [18], it has undergone significant modifications. These include major structural changes, key integrations, and substantial extensions to functionality. All modifications were developed specifically in the context of this work. The current implementation is available as an open-source project.

6.1 Circom Circuits Optimization

As in the previous version, the supported image transformations are represented as arithmetic circuits over BN128 using the Circom language [8, 34]. A major improvement introduced in this work is the optimization of region hashing. Since this accounts for a large portion of the computation, the changes, described below, resulted in significant reductions in circuit size and improvements in overall proving time.

We evaluated three hashing strategies:

- *LinearFoldHasher*: the baseline approach used in the original VIMz implementation, hashes the image pixel array entry-by-entry (two field elements at a time).
- *WindowFoldHasher*: processes the image data in fixed-size chunks, allowing multiple field elements to be hashed simultaneously.
- *MerkleFoldHasher*: constructs a k-ary recursive hash tree over disjoint data blocks.

WindowFoldHasher was selected as the preferred method due to its favorable trade-off between implementation complexity and efficiency. Compared to the baseline, it reduced the number of wires by up to 78% and improved proving time by up to almost 18% across supported image operations. While *MerkleFoldHasher*

achieved comparable performance to *WindowFoldHasher*, it introduced greater complexity in circuit templating. Table 2 presents detailed benchmark results.

We also performed a substantial refactor and modernization of the circuits codebase. New Circom features—such as anonymous components and buses—were adopted, resulting in more modular, readable, and robust code. Additionally, we introduced a new image transformation: redaction (see Listing 1). Unlike the other operations, redaction is applied to image blocks rather than pixel rows. Table 3 presents proof generation time for redaction, using 'nova-snark' [49] backend. We note that while using bigger block sizes reduces significantly proving time, it yields a less useful and practical operation.

Table 2: Comparison of array hashing strategies for HD images. L = Linear folding, W = Windowed folding (size 8). ΔW denotes the relative reduction in wire count when using W instead of L. Runtime was measured for 10 folding steps.

Transform	L [s]	W [s]	Speedup [%]	ΔW [%]
Blur	17.5	14.6	16.57	28.51
Brightness	16.2	14.8	8.64	14.21
Contrast	16.4	14.9	9.15	14.21
Crop	23.6	21.1	10.59	5.08
Grayscale	11.0	9.63	12.45	28.90
Hash	6.29	5.69	9.54	78.00
Resize	17.3	14.2	17.92	28.97
Sharpness	19.4	16.8	13.40	23.64

```
template RedactHash(blockSize) {
    input  IVCState() step_in;
    output IVCState() step_out;

    signal input block [blockSize];
    signal input redact;

    signal prev_redact_hash <== step_in.tran_hash;
    signal block_hash <== ArrayHasher(blockSize)(block);

    component selector = Mux1();
    selector.c[0] <== PairHasher()(prev_redact_hash,
        block_hash);
    selector.c[1] <== PairHasher()(prev_redact_hash, 0);
    selector.s <== redact;

    step_out.orig_hash <== PairHasher()(step_in.orig_hash
        , block_hash);
    step_out.tran_hash <== selector.out;
}
```

Listing 1: Circom implementation of the redaction circuit. This circuit operates on image blocks rather than individual rows and conditionally accumulates the transformation hash depending on whether a block is marked for redaction.

Table 3: Folding times for an HD-resolution image redaction transformations using Nova-SNARK backend on AWS EC2 `c7a.8xlarge`, measured across different block sizes. Mem = maximum RAM usage during proof generation.

Block Size	# Regions	Folding time (s)	Mem (MB)
10×10	9216	3866	266
20×20	2304	978	263
40×40	576	266	367
80×80	144	85	690

6.2 Proof Generation Backends

Once Circom R1CS circuits are compiled, the resulting artifacts (witness generator and R1CS description) must be passed to a framework capable of executing a folding-based (IVC) protocol and generating the final SNARK proof. The original VIMz implementation [18] used the `nova-snark` [49] Rust library developed by Microsoft, in combination with the `Nova-Scotia` middleware [33], which enabled integration with Circom-generated circuits. This setup achieved state-of-the-art performance [18].

The most significant advancement since then has been integration with the Sonobe library [1], a Rust-based folding schemes framework developed by the 0xPARC [2] and Privacy Scaling Explorations (PSE) [25] teams. Sonobe introduces several notable capabilities:

- A unified interface supporting multiple modern folding schemes, including Nova [40], HyperNova [39], and ProtoGalaxy [20].
- An automated pipeline for generating Solidity verifier contracts, enabling efficient on-chain proof validation.
- Compatibility with several popular circuit frontends, including Arkworks [4] (native), Circom [8, 34], Noir [6], and Noname [70].

Despite these advantages, Sonobe is still an emerging project and currently does not match the performance of the previous setup. In particular, proof generation with Sonobe requires significantly more computational resources and longer runtimes. The current bottleneck lies in the conversion from the folded proof to the final SNARK, handled by the so-called *decider*, which remains highly unoptimized. This step alone can require approximately 36 GB of RAM for a typical image transformation circuit.

Although operations such as Solidity verifier generation and trusted setup could (and arguably should) be outsourced to a trusted and powerful actor, the current Sonobe interface does not offer a clean separation between these one-time setup steps and the main proof computation. As a result, during development we had to repeat these expensive steps for each proof generation.

On an AWS EC2 `c7a.8xlarge` instance, generating a Solidity verifier takes under 2 minutes. However, producing a zkSNARK for an HD-resolution image transformation currently requires up to 36 GB of RAM and takes approximately 25 minutes. While this is feasible only on high-end personal machines or dedicated servers, we expect that modest improvements to Sonobe could yield substantial performance gains. In the meantime, a practical mitigation is to outsource proof generation to an external server.

Looking ahead, further efficiency gains may be achieved by migrating from Nova to more advanced folding schemes supported by Sonobe. Another promising direction is adopting alternative circuit frontends such as the natively supported Arkworks stack. These directions are already under evaluation and planned for future development. Our current implementation includes a unified toolchain that supports both `nova-snark` [49] and Sonobe [1] as proof generation backends.

6.3 Image Preprocessing and Tooling

The backends discussed above cannot operate directly on PNG files or raw binary image data. Instead, they require input formatted specifically for performing folding operations over finite field elements. To address this, we extended and refactored the legacy image preprocessing scripts into a standalone auxiliary Python toolbox called `pyvimz`. This tool serves two primary purposes:

- It provides a simple command-line (CLI) image editor. In particular, it can apply any of the supported transformations to an image and convert both the original and edited versions into a folding-friendly format.
- It enables hashing images in a circuit-compatible manner. As a reminder, the VIMz protocol computes accumulated hashes for both the source and edited images to ensure integrity, as is shown in Figure 1. `pyvimz` uses the same Circom hash implementation as the proof generation backend to ensure consistency.

For the redaction transformation, we also provide a simple graphical tool (GUI) implemented in TypeScript. It allows users to manually select regions of the image to redact—an interaction that is significantly more user-friendly than command-line input. Like `pyvimz`, this tool also converts images into a folding-compatible format that can be consumed by the backend.

6.4 Smart Contracts

A newly introduced component of the codebase is the on-chain realization of the ecosystem, implemented through a suite of Solidity smart contracts. These contracts fall into two main categories: verification utilities and application-layer components.

Verifier Contracts. Each supported image transformation is associated with a dedicated Solidity verifier. These verifiers are generated automatically by the Sonobe [1] toolchain from the corresponding Circom [34] circuit. The deployed bytecode size typically ranges from 25 kB to 27.5 kB [2]. Deploying a single verifier contract consumes up to 3.5 million gas. As of May 2025, this corresponds to an approximate cost of $0.01 on the Aleph Zero network. Verification of an HD-resolution image transformation proof requires no more than 850,000 gas, making it practical and cost-effective.

Application-Layer Contracts. The application-layer suite—including the image gateway, marketplace, and photography contest contracts—was implemented manually in Solidity (version 0.8.26). Where applicable, we adhered to established standards and best practices to ensure code quality and security. For example, the `ImageCollection` contract is implemented as an ERC721 [22] NFT, and the `LicenseToken` contract conforms to the ERC165 [60], ERC721 [22], and ERC4907 [3] interfaces. Where relevant, access control and reentrancy guards were applied to ensure contract safety.

Comprehensive documentation and source code for all contracts are publicly available in the GitHub repository[3]. Table 4 contains several measurements for the representative on-chain components.

Table 4: Contract deployment and execution costs. Size: deployed bytecode size. Deploy: gas cost of deployment (in thousands). Call: upper-bound for gas usage for representative function calls, e.g., registration or verification.

Contract	Size [kB]	Deploy [k]	Call [k]
Verifier	27.3	3,237	848 (verify)
DeviceRegistry	3.5	439	48 (register)
CreatorRegistry	3.2	399	136 (register)
ImageGateway	10.8	1,415	1,019 (register)
Marketplace	10.4	1,201	208 (licensing)
PhotoContest	8.4	1,076	69 (submit)

6.5 SDK and Developer Tooling

For testing and demonstration purposes, we provide a complete Python SDK that enables both local and remote interaction with the on-chain ecosystem. It supports deploying and calling contracts on locally spawned nodes (e.g., Foundry's `anvil`) as well as public blockchains such as Aleph Zero [27].

In addition, the SDK includes a set of illustrative scenarios that demonstrate common usage patterns described in previous sections, including image registration, edition tracking, license minting, and photography contest workflows. Each script reports associated gas costs for both performance evaluation and result reproducibility.

7 Related Work

In this section, we review recent advancements in FairSwap-style protocols. Although our protocol does not strictly belong to this category, it shares certain similarities. For instance, both systems aim to enable trustless exchange of digital assets. However, our system extends beyond asset swapping to support applications such as photography contests and fact-checking platforms, where on-chain verification of media provenance is required. Unlike FairSwap-based protocols that typically focus on private exchanges between two parties, our protocol enables public verification through ZKPs and supports visibility-preserving mechanisms such as redacted or transformed previews. Furthermore, because our design is grounded in recent developments in image authenticity verification, we also examine related literature in that area.

7.1 FairSwap Protocols

FairSwap [19] is an efficient protocol for the fair exchange of digital goods using smart contracts. It enables a seller to exchange a

[2] This slightly exceeds the size limits imposed by some popular EVM chains such as Ethereum. However, these contracts remain deployable on several Layer 2 platforms that support larger contract sizes, including the Aleph Zero blockchain. Ongoing improvements to the Sonobe code generation pipeline may help reduce contract size in future releases.

[3] https://github.com/zero-savvy/vimz

digital asset for payment, guaranteeing that the buyer pays only upon receiving the correct item, and the seller receives payment only if they deliver the agreed content. Historically, such fairness was proven impossible without a Trusted Third Party (TTP), but FairSwap circumvents this via blockchain-based arbiters acting as decentralized judges.

The primary goals of FairSwap are efficiency and fairness, minimizing on-chain computation. However, several challenges remain:

- **Passive Party Fairness:** In multi-party settings such as marketplaces involving owners, facilitators, and buyers, fairness is typically ensured only for active participants. TEDX [7] addresses this by separating distribution and settlement, using incentive mechanisms and state channels to support passive roles.
- **Scalability:** Multi-round interactions and computational demands limit scalability for high-throughput digital marketplaces [7].
- **Complex Predicates and Large Data:** FairSwap's optimistic approach may impose high workloads on arbiters, especially for complex or large digital assets.
- **Privacy:** FairSwap does not inherently preserve user or transaction privacy.

zkMarket [55] builds a privacy-preserving fair exchange system leveraging zk-SNARKs and a novel MatPRG pseudorandom generator. It addresses inefficiencies in previous ZKCP approaches and supports anonymous transactions. However, zkMarket lacks support for verifying content transformations, provenance, or partially visible media.

FairDEx [23] targets the exchange of unique digital goods where public descriptions (e.g., hashes) are unavailable. It uses a sampling-based protocol to jointly construct a description and reduce on-chain costs using symmetric encryption and key publication. This is suitable for customized or private digital assets.

Advertisement-Based Fair Exchange [51] combines an advertising phase with a fair exchange mechanism, particularly for confidential digital assets like NFTs. It allows public previews and asset commitments using zk-SNARKs, without requiring off-chain communication. Verification is performed once per asset, addressing challenges such as denial-of-service or constant seller availability.

While FairSwap provides a foundational primitive for fair exchange, real-world marketplace implementations require broader considerations. Protocols such as TEDX, zkMarket, FairDEx, and advertisement-based approaches extend or generalize FairSwap concepts to support scalability, privacy, public verifiability, and diverse participant roles.

Our system continues along this route by emphasizing verifiable media integrity, publicly shareable transformations, and authenticity inheritance. These features enable not only fair exchange but also novel applications such as automated licensing, contest validation, and infringement resolution. To support such use cases effectively, on-chain verification must remain efficient and scalable. In this regard, we explore and incorporate recent techniques aimed at minimizing the cost and complexity of proof verification on the blockchain.

7.2 Media Verification Protocols

The foundational idea of using ZKPs to verify the originality of edited images was introduced by [52], employing Proof-Carrying Data (PCD) [11]. Since then, verifiable computation and proof systems have advanced significantly, making them efficient enough for real-world applications such as on-chain marketplaces with verifiable media authenticity.

VIMz [18], leveraging folding-based SNARKs [49], proposes a practical solution for generating proofs of authenticity for high-resolution edited images—up to 8K—on commodity hardware. Veritas [16] introduces a scalable approach for AI-generated image verification using lattice-based hashing for image commitment. However the commitment performance is the drawback and results in longer proving times and higher memory usage compared to the VIMz. The system presented in [17] reduces prover complexity by tiling the image and generating separate proofs per tile. However, this increases both proof size and verification time linearly and limits support for certain transformations due to the lack of context between adjacent tiles.

A comparison of these approaches is provided in Table 8 of the VIMz paper. According to those results, generating a proof for an 8K image resized to 2048×1365 takes approximately 49, 76, and 56 minutes for VIMz, Veritas [16], and [17], respectively. Verification times (for the same proof) are reported as 0.3 seconds for VIMz, 198 seconds (or 2.2 seconds in an optimized variant) for Veritas, and 107 seconds for [17].

8 Conclusion and Future Work

In this work, we address the problem of building robust, verifiable, public, and trustless systems for working with authenticated images and their editions. We provide an approach complementary to existing techniques such as C2PA [12], aimed at protecting users from media manipulation and fake news.

We extended the original VIMz protocol [18] in several directions. We optimized the arithmetic circuits, which form the core of the protocol, improving proof generation times. Further, we generalized the protocol to operate over arbitrary image regions and, as an example, introduced a new block-based transformation. Most importantly, we designed and implemented a complete on-chain, trustless ecosystem, which includes both core verification and registration components, as well as practical, user-facing applications.

In terms of future work, we observe a significant performance penalty when using the `Sonobe` [1] backend. We expect that future improvements to the `Sonobe` interface and internals will lead to substantial performance gains. Additionally, we plan to evaluate performance across different supported folding schemes and circuit frontends. We also plan to integrate accelerator libraries such as `ICICLE` [35] to further enhance performance by leveraging hardware acceleration on CPU and GPU.

Acknowledgments

The authors used ChatGPT-4 to revise the writing throughout the entire paper. The tool was employed to correct typographical and grammatical errors, improve phrasing, and enhance sentence flow. In addition, GitHub Copilot was used as a coding assistant for

generating documentation, unit tests, and auxiliary scripts during software development.

This work was conducted as part of Piotr Mikołajczyk's industrial PhD program, supported by the Jagiellonian University and the Polish Ministry of Science and Higher Education.

References

[1] 0xPARC and PSE. 2025. Sonobe: zkSNARK Verifier for Solidity. https://github.com/privacy-scaling-explorations/sonobe. Accessed: 2025-04-30.

[2] 0xPARC Foundation. 2025. 0xPARC offical website. https://0xparc.org/. Accessed: 2025-05-26.

[3] Anders, Lance, and Shrug. 2018. *ERC-4907: Rental NFT, an Extension of EIP-721.* Ethereum Improvement Proposal. Ethereum Foundation. https://eips.ethereum.org/EIPS/eip-4907 EIP-4907.

[4] arkworks contributors. 2022. `arkworks` *zkSNARK ecosystem.* https://arkworks.rs

[5] Zahra Nazemi Ashani, Iszuanie Syfidza Che Ilias, Keng Yap Ng, Muhammad Rezal Kamel Ariffin, Ahmad Dahari Jarno, and Nor Zarina Zamri. 2024. Comparative Analysis of Deepfake Image Detection Method Using VGG16, VGG19 and ResNet50. *Journal of Advanced Research in Applied Sciences and Engineering Technology* 47, 1 (Jun. 2024), 16–28. https://doi.org/10.37934/araset.47.1.1628

[6] Aztec. 2025. Noir language offical website. https://aztec.network/noir. Accessed: 2025-05-28.

[7] Prabal Banerjee, Dushyant Behl, Palanivel Kodeswaran, Chaitanya Kumar, Sushmita Ruj, Sayandeep Sen, and Dhinakaran Vinayagamurthy. 2023. Accelerated Verifiable Fair Digital Exchange. *Distributed Ledger Technologies: Research and Practice* 2, 3 (2023), 1–24.

[8] Marta Bellés-Muñoz, Miguel Isabel, Jose Luis Muñoz-Tapia, Albert Rubio, and Jordi Baylina. 2023. Circom: A Circuit Description Language for Building Zero-Knowledge Applications. *IEEE Transactions on Dependable and Secure Computing* 20, 6 (2023), 4733–4751. https://doi.org/10.1109/TDSC.2022.3232813

[9] Adobe Security Bulletin. 2025. APSB25-40. https://helpx.adobe.com/security/products/photoshop/apsb25-40.html. Accessed: 2025-04-30.

[10] Canon. 2024. Canon Launches Flagship EOS R1 and Advanced EOS R5 Mark II Mirrorless Cameras. https://www.canon.co.uk/press-centre/press-releases/2024/07/canon-launches-flagship-eos-r1-and-advanced-eos-r5-mark-ii-mirrorless-cameras/. Accessed: 2025-05-22.

[11] Alessandro Chiesa and Eran Tromer. 2010. Proof-Carrying Data and Hearsay Arguments from Signature Cards. , 310–331 pages.

[12] Coalition for Content Provenance and Authenticity (C2PA). 2025. C2PA official website. https://c2pa.org/. Accessed: 2025-05-22.

[13] Radiant Commons. 2025. Penumbra official website. https://penumbra.zone/. Accessed: 2025-05-25.

[14] National Vulnerability Database. 2025. CVE-2025-30324. https://nvd.nist.gov/vuln/detail/. Accessed: 2025-04-30.

[15] National Vulnerability Database. 2025. CVE-2025-30325. https://nvd.nist.gov/vuln/detail/CVE-2025-30325. Accessed: 2025-04-30.

[16] Trisha Datta, Binyi Chen, and Dan Boneh. 2024. VerITAS: Verifying Image Transformations at Scale. Cryptology ePrint Archive, Paper 2024/1066. https://eprint.iacr.org/2024/1066

[17] Pierpaolo Della Monica, Ivan Visconti, Andrea Vitaletti, and Marco Zecchini. 2024. Trust Nobody: Privacy-Preserving Proofs for Edited Photos with Your Laptop. Cryptology ePrint Archive, Paper 2024/1074. https://eprint.iacr.org/2024/1074

[18] Stefan Dziembowski, Shahriar Ebrahimi, and Parisa Hassanizadeh. 2024. VIMz: Private Proofs of Image Manipulation using Folding-based zkSNARKs. Cryptology ePrint Archive, Paper 2024/1063. https://eprint.iacr.org/2024/1063

[19] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. 2018. FairSwap: How to fairly exchange digital goods. Cryptology ePrint Archive, Paper 2018/740. https://doi.org/10.1145/3243734.3243857

[20] Liam Eagen and Ariel Gabizon. 2023. ProtoGalaxy: Efficient ProtoStar-style folding of multiple instances. Cryptology ePrint Archive, Paper 2023/1106. https://eprint.iacr.org/2023/1106

[21] T. Elgamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 4 (1985), 469–472. https://doi.org/10.1109/TIT.1985.1057074

[22] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. 2018. *ERC-721: Non-Fungible Token Standard.* Ethereum Improvement Proposal. Ethereum Foundation. https://eips.ethereum.org/EIPS/eip-721 EIP-721.

[23] Oguzhan Ersoy, Ziya Alper Genç, Zekeriya Erkin, and Mauro Conti. 2021. Practical Exchange for Unique Digital Goods. , 49-58 pages. https://api.semanticscholar.org/CorpusID:239037447

[24] Dan Evon. 2020. Did Italy Dig Mass Graves for Coronavirus Victims? https://www.snopes.com/fact-check/mass-graves-italy-coronavirus/. Accessed: 2025-05-22.

[25] Privacy + Scaling Explorations. 2025. PSE offical website. https://pse.dev/. Accessed: 2025-05-26.

[26] Robert Farley. 2015. Obama-Rouhani Photo is Not Real. https://www.factcheck.org/2015/07/obama-rouhani-photo-is-not-real/. Accessed: 2025-05-22.

[27] Aleph Zero Foundation. 2025. Aleph Zero official website. https://alephzero.org/. Accessed: 2025-05-25.

[28] Aleph Zero Foundation and Cardinal Cryptography. 2025. Common official website. https://common.fi/. Accessed: 2025-05-25.

[29] Google. 2025. Imagen official website. https://deepmind.google/models/imagen/. Accessed: 2025-05-25.

[30] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. 2019. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. Cryptology ePrint Archive, Paper 2019/458. https://eprint.iacr.org/2019/458

[31] Lucas Graves. 2018. Understanding the Promise and Limits of Automated Fact-Checking. https://reutersinstitute.politics.ox.ac.uk/our-research/understanding-promise-and-limits-automated-fact-checking. Accessed: 2025-05-22.

[32] Jens Groth. 2016. On the Size of Pairing-Based Non-interactive Arguments. In *Advances in Cryptology – EUROCRYPT 2016*, Marc Fischlin and Jean-Sébastien Coron (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 305–326.

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

[34] iden3. 2024. CircomLib. https://github.com/iden3/circomlib/. Accessed: 2024-02-07.

[35] Karthik Inbasekar, Yuval Shekel, and Michael Asa. 2024. ICICLE v2: Polynomial API for Coding ZK Provers to Run on Specialized Hardware. *Cryptology ePrint Archive* (2024).

[36] Snopes Media Group Inc. 2025. Snopes website. https://www.snopes.com/. Accessed: 2025-05-31.

[37] Poynter Institute. 2025. PolitiFact website. https://www.politifact.com/. Accessed: 2025-05-31.

[38] Jihoon Kim, Hyerean Jang, and Youngjoo Shin. 2025. A Survey of Side-Channel Attacks on Branch Prediction Units. https://doi.org/10.1145/3734218 Just Accepted.

[39] Abhiram Kothapalli and Srinath Setty. 2023. HyperNova: Recursive arguments for customizable constraint systems. Cryptology ePrint Archive, Paper 2023/573. https://eprint.iacr.org/2023/573

[40] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. 2021. Nova: Recursive Zero-Knowledge Arguments from Folding Schemes. Cryptology ePrint Archive, Paper 2021/370. https://eprint.iacr.org/2021/370

[41] Digital Forensic Research Lab. 2022. Russian War Report: Hacked News Program and Deepfake Video Spread False Zelenskyy Claims. https://www.atlanticcouncil.org/blogs/new-atlanticist/russian-war-report-hacked-news-program-and-deepfake-video-spread-false-zelenskyy-claims/. Accessed: 2025-05-22.

[42] Ben Lapid and Avishai Wool. 2018. Cache-Attacks on the ARM TrustZone implementations of AES-256 and AES-256-GCM via GPU-based analysis. Cryptology ePrint Archive, Paper 2018/621. https://eprint.iacr.org/2018/621

[43] Leica. 2025. Leica and Content Credentials: Trusted Metadata in Photography. https://leica-camera.com/en-GB/photography/content-credentials. Accessed: 2025-05-22.

[44] Yuezun Li, Xin Yang, Pu Sun, Honggang Qi, and Siwei Lyu. 2020. Celeb-DF: A Large-scale Challenging Dataset for DeepFake Forensics. arXiv:1909.12962 [cs.CR] https://arxiv.org/abs/1909.12962

[45] Matteo Loporchio, Anna Bernasconi, Damiano Di Francesco Maesa, and Laura Ricci. 2023. A survey of set accumulators for blockchain systems. *Computer Science Review* 49 (2023), 100570. https://doi.org/10.1016/j.cosrev.2023.100570

[46] Olivia Ma and Brandon Feldman. 2022. How Google and YouTube Are Investing in Fact-Checking. https://blog.google/outreach-initiatives/google-news-initiative/how-google-and-youtube-are-investing-in-fact-checking/. Accessed: 2025-05-22.

[47] Bill McCarthy. 2024. Photo of Charlottesville Rally Was Doctored. https://factcheck.afp.com/doc.afp.com.34QE8QC. Accessed: 2025-05-22.

[48] Ralph C. Merkle. 1980. Protocols for Public Key Cryptosystems. , 122-122 pages. https://doi.org/10.1109/SP.1980.10006

[49] Microsoft. 2024. Nova High-speed recursive arguments from folding schemes. https://github.com/microsoft/Nova. Accessed: 2024-01-03.

[50] Yisroel Mirsky and Wenke Lee. 2021. The Creation and Detection of Deepfakes: A Survey. *Comput. Surveys* 54, 1 (Jan. 2021), 1–41. https://doi.org/10.1145/3425780

[51] Pierpaolo Della Monica, Ivan Visconti, Andrea Vitaletti, and Marco Zecchini. 2025. Public Channel-Based Fair Exchange Protocols with Advertising. arXiv:2503.10411 [cs.CR] https://arxiv.org/abs/2503.10411

[52] Assa Naveh and Eran Tromer. 2016. PhotoProof: Cryptographic Image Authentication for Any Set of Permissible Transformations. , 255-271 pages. https://doi.org/10.1109/SP.2016.23

[53] Alexander Nilsson, Pegah Nikbakht Bideh, and Joakim Brorsson. 2020. A Survey of Published Attacks on Intel SGX. arXiv:2006.13598 [cs.CR] https://arxiv.org/abs/2006.13598

[54] OpenAI. 2025. DALL·E 3 official website. https://openai.com/index/dall-e-3/. Accessed: 2025-05-25.

[55] Seongho Park, Seungwoo Kim, Semin Han, Kyeongtae Lee, Jihye Kim, and Hyunok Oh. 2024. zkMarket : Privacy-preserving Digital Data Trade System via Blockchain. Cryptology ePrint Archive, Paper 2024/1775. https://eprint.iacr.org/2024/1775

[56] World Press Photo. 2025. World Press Photo 2025 Verification Process. https://www.worldpressphoto.org/contest/2025/verification-process. Accessed: 2025-05-25.

[57] Numbers Protocol. 2023. Numbers Protocol Whitepaper. https://ipfs-pin.numbersprotocol.io/ipfs/bafybeifu5srb5jlstk4eepsg5xaqm3dq5ty2tjetzwb6t5nqb4rrubxola. Accessed: 2024-12-30.

[58] Josh Raab. 2014. Did Ukraine Use Fake Images to Win Sympathy? https://time.com/3810444/ukraine-fake-images-claim/. Accessed: 2025-05-22.

[59] Railgun. 2025. Railgun official website. https://www.railgun.org/. Accessed: 2025-05-25.

[60] Christian Reitwießner, Nick Johnson, Fabian Vogelsteller, Jordi Baylina, Konrad Feldmeier, and William Entriken. 2018. *ERC-165: Standard Interface Detection.* Ethereum Improvement Proposal. Ethereum Foundation. https://eips.ethereum.org/EIPS/eip-165 EIP-721.

[61] Reuters. 2020. Fact Check: Staged Prank Among Posts with Photographs of Demonstrators with Nazi Flags. https://www.reuters.com/article/world/fact-check-staged-prank-among-post-with-photographs-of-demonstrators-with-nazi-idUSKBN23M2TH/. Accessed: 2025-05-22.

[62] Mike Schroepfer. 2019. Deepfake Detection Challenge. https://ai.meta.com/blog/deepfake-detection-challenge/. Accessed: 2025-05-22.

[63] Alex Seitz-Wald. 2024. Telecom Company Agrees to 1 Million Fine over Biden Deepfake. https://www.nbcnews.com/politics/2024-election/telecom-company-agrees-1-million-fine-biden-deepfake-rcna167564. Accessed: 2025-05-22.

[64] Srinath Setty. 2019. Spartan: Efficient and general-purpose zkSNARKs without trusted setup. Cryptology ePrint Archive, Paper 2019/550. https://eprint.iacr.org/2019/550 https://eprint.iacr.org/2019/550.

[65] Fazeela Siddiqui, Jiachen Yang, Shuai Xiao, and Muhammad Fahad. 2025. Enhanced deepfake detection with DenseNet and Cross-ViT. *Expert Systems with Applications* 267 (2025), 126150. https://doi.org/10.1016/j.eswa.2024.126150

[66] Shree Singhi, Aayan Yadav, Aayush Gupta, Shariar Ebrahimi, and Parisa Hassanizadeh. 2025. Provenance Detection for AI-Generated Images: Combining Perceptual Hashing, Homomorphic Encryption, and AI Detection Models. *arXiv e-prints* (2025), arXiv–2503.

[67] Luisa Verdoliva. 2020. Media Forensics and DeepFakes: An Overview. *IEEE Journal of Selected Topics in Signal Processing* 14, 5 (2020), 910–932. https://doi.org/10.1109/JSTSP.2020.3002101

[68] Zhiyuan Zhang, Mingtian Tao, Sioli O'Connell, Chitchanok Chuengsatiansup, Daniel Genkin, and Yuval Yarom. 2023. BunnyHop: Exploiting the Instruction Prefetcher. In *32nd USENIX Security Symposium (USENIX Security 23).* USENIX Association, Anaheim, CA, 7321–7337. https://www.usenix.org/conference/usenixsecurity23/presentation/zhang-zhiyuan-bunnyhop

[69] Jiaxing Zhao, Srinath Setty, Weidong Cui, and Greg Zaverucha. 2024. MicroNova: Folding-based arguments with efficient (on-chain) verification. Cryptology ePrint Archive, Paper 2024/2099. https://eprint.iacr.org/2024/2099

[70] zkSecurity. 2025. NoName language offical website. https://zksecurity.github.io/noname/. Accessed: 2025-05-28.

A Formal Definition of SNARKs

Formally, the properties of [zk]SNARKs are as follows. A non-interactive argument for $\mathcal{R}$ $\Pi = (\mathcal{G}, \mathcal{P}, \mathcal{V}, \mathcal{S})$ is a SNARK if it satisfies the following three properties:

- **Completeness**: If the prover is honest and possesses a valid witness, then the verifier should accept the proof. Formally, for any probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$:

$$Pr\left[\begin{array}{l|l} \mathsf{Verify}(pp, vk, u, \pi) = 1 & \begin{array}{l} pp \leftarrow \mathsf{Gen}(1^\lambda) \\ (s, (u, w))) \leftarrow \mathcal{A}(pp) \\ (pp, s, u, w) \in \mathcal{R} \\ (pk, vk) \leftarrow \mathcal{K}(pp, s) \\ \pi \leftarrow \mathsf{Prove}(pk, u, w) \end{array} \end{array}\right] = 1$$

- **Knowledge Soundness**: A malicious prover (i.e., an adversary) should not be able to convince the verifier of a false statement. More formally, for every PPT adversary $\mathcal{A}$, there exists an extractor $\mathcal{E}$ such that, for any random tape ρ, the following holds:

$$Pr\left[\begin{array}{l|l} \mathsf{Verify}(pp, vk, u, \pi) = 1, & \begin{array}{l} pp \leftarrow \mathsf{Gen}(1^\lambda) \\ (s, (u, w))) \leftarrow \mathcal{A}(pp) \\ (pp, s, u, w) \notin \mathcal{R} & (pk, vk) \leftarrow \mathcal{K}(pp, s) \\ w \leftarrow \mathcal{E}(pp, \rho) \end{array} \end{array}\right] = \mathrm{negl}(\lambda)$$

- **Zero-Knowledge**: There must exist a PPT simulator $\mathcal{S}$ such that for all PPT adversaries $\mathcal{A}$ following distributions are indistinguishable:

$$\mathcal{D}_1 = \left\{ (pp, s, u, \pi) \;\middle|\; \begin{array}{l} pp \leftarrow \mathcal{G}(1^\lambda) \\ (s, (u, w))) \leftarrow \mathcal{A}(pp) \\ (pp, s, u, w) \in \mathcal{R} \\ (pk, vk) \leftarrow \mathcal{K}(pp, s) \\ \pi \leftarrow \mathcal{P}(pk, u, w) \end{array} \right\}$$

$$\mathcal{D}_2 = \left\{ (pp, s, u, \pi) \;\middle|\; \begin{array}{l} (pp, \rho) \leftarrow \mathcal{S}(1^\lambda) \\ (s, (u, w))) \leftarrow \mathcal{A}(pp) \\ (pp, s, u, w) \in \mathcal{R} \\ (pk, vk) \leftarrow \mathcal{K}(pp, s) \\ \pi \leftarrow \mathcal{S}(pp, u, \rho) \end{array} \right\}$$