

AIPA: Anonymous Image Provenance Authentication in Online Social Networks via Unlinkable Pseudonym Certificates and zk-SNARKs

Linsheng Yu and Yanqing Yao

Abstract—Generative AI has made high-fidelity deepfakes easy to produce, and online social networks (OSNs) spread them widely across users and platforms. Publicly verifying an image’s provenance, i.e., where an image came from and what edits it has undergone, can help combat deepfakes. The Coalition for Content Provenance and Authenticity (C2PA) offers an industry standard, but depends on trusted software and hardware. Through digital signatures and zero-knowledge proofs, cryptographic image authentication eliminates the reliance on trusted editors. However, deploying such schemes in OSNs faces three problems: preserving signer privacy, maintaining provenance authenticity across edits, and ensuring efficient verification. To ensure signer anonymity in image provenance, we propose an unlinkable pseudonym certificate scheme (UPCS) based on anonymous credentials, which reduces online signing and verification to ordinary digital signature operations taking only 0.017 and 0.066 ms. Building on UPCS, zk-SNARKs, and hash chains, we propose an anonymous image provenance authentication (AIPA) scheme, ensuring the authenticity of provenance and editing history of an image as it propagates across mutually untrusted editors, while preserving signer privacy. To achieve efficient verification in OSNs, we introduce VIMz-Loua, a GPU-accelerated image proof system with a dedicated JPEG-compression circuit, whose proofs remain a constant 448 B and verify in 6–12 ms across image sizes. We formally prove the security of UPCS and AIPA, and implement an end-to-end workflow in a simulated OSN environment, where the full multi-edit provenance lineage is only 40.2 KB (6.5% of the published image), and is verified in 4.28 s.

Index Terms—Anonymous credentials, pseudonym certificates, online social networks, image provenance, zk-SNARKs.

I. INTRODUCTION

GENERATIVE AI (GenAI) for images has changed digital content creation. Yet its widespread adoption introduces significant security risks: GenAI enables mass production of high-fidelity fake faces, impersonations, and fabricated scenes, thereby facilitating deepfake generation and threatening individual rights, social trust, and public opinion [1], [2]. Online social networks (OSNs), with their large user bases and rapid content dissemination, amplify this threat by accelerating the spread of forged images across users and platforms [3], making it increasingly difficult to ascertain image provenance.

Many deepfake defenses have been proposed. Detection-based approaches employ models to distinguish forged content from authentic content [4], [5]. Digital watermarking embeds

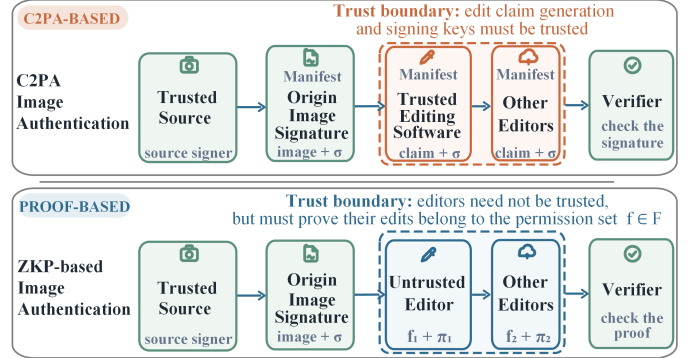


Fig. 1. Editing workflows and trust assumptions in C2PA (top) and ZKP-based image authentication (bottom).

imperceptible messages for traceability and copyright protection [6]. In contrast, the Coalition for Content Provenance and Authenticity (C2PA) [7] addresses a fundamentally different question: *where an image came from and what edits it has undergone*.

As shown in Fig. 1, when a C2PA-compliant device or service produces an image, it signs a manifest of provenance assertions, such as the device type, capture location, and timestamp, and embeds it in the image. Each subsequent edit generates a new signed manifest, recording the editing software, operations, and parameters, which is appended to the manifest store, forming a traceable provenance lineage. Verifying this lineage enables anyone to trace the image’s provenance and editing history, offering protection against deepfakes. C2PA has been widely adopted by industry leaders such as Sony [8], Google [9], and OpenAI [10].

However, this lineage is trustworthy only if each manifest records the edit truthfully and each signing key is protected. At the source this is reasonable: signers are usually authenticated devices or generative services. At edit time, however, the trustworthiness of each appended manifest depends on the editing software to sign it honestly. Yet this assumption is vulnerable to many attacks [11]–[13]. This risk is acute in OSNs, where images are edited and re-shared among users and platforms, so that the lineage spans many mutually untrusted editors. It is a significant challenge to ensure that each edit is recorded truthfully and completely.

Cryptographic image authentication, pioneered by Naveh and Tromer’s PhotoProof [14], eliminates the reliance on trusted editors through digital signatures and zero-knowledge

Linsheng Yu and Yanqing Yao are with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China, and also with the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, Beihang University, Beijing 100083, China (e-mail: linsygyu@163.com; yaoyq@buaa.edu.cn). Corresponding author: Yanqing Yao.

proofs (ZKPs). As in C2PA, the source device's signature on the original image acts as the root of trust (see Fig. 1). Unlike C2PA, any party can locally apply permissible edits in an untrusted environment and prove their validity and computational integrity. Besides, the zero-knowledge property ensures that the proof reveals nothing about the image beyond its authenticity. However, deploying cryptographic image authentication in OSNs faces three problems.

First, signer privacy is critical in OSNs: the signer's long-term public key propagates with the image as it is re-shared across platforms, enabling attackers to link signatures to the signer's identity and behavior. VIMz [15] mitigates this risk by proving signature validity in zero knowledge via Spartan-ECDSA [16], but incurs high proving overhead. VerITAS [17] employs a group signature scheme [18] with low overhead, yet its group manager can trace signers. A practical solution must therefore achieve low proving overhead while guaranteeing signer anonymity even against the certificate authority (CA).

A second problem is ensuring the authenticity and integrity of the editing history across mutually untrusted editors in OSNs. PhotoProof [14] provides this guarantee via proof-carrying data (PCD) [19], but even a single edit requires proving the entire circuit of permissible edits, imposing high computational overhead. Later work [20]–[22] uses zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) to make the proof of a single edit efficient and practical, but not for multi-edit scenarios. ZK-IMG [23] and VIMz [15] instead link a sequence of edits with hash chains, where each proof covers only the edit actually applied, so the proving cost depends only on the current edit rather than on the permissible set. However, their work focuses on a single editor, and neither extends a provenance lineage across mutually untrusted editors.

Finally, deploying image authentication in OSNs demands high verification efficiency. These platforms handle massive uploads, and a popular image may be viewed millions of times. Each view may trigger a provenance verification. Proofs must therefore be small and efficient to verify. At the same time, platforms routinely resize, filter, and compress uploaded images to reduce storage [24]–[26], and these operations must likewise be covered by the provenance lineage.

To preserve signer privacy, we propose an unlinkable pseudonym certificate scheme (UPCS), which decouples long-term identity from signing: a signer holds an anonymous credential [27], [28] at the identity layer and signs with one-time pseudonym certificates issued at the certificate layer. This decoupling reduces online signing and verification to ordinary digital signature operations, while providing signer anonymity, selective attribute disclosure, certificate preprocessing, batch issuance, and X.509 compatibility. And the signer remains unlinkable even to the issuer and CA. Beyond image provenance, UPCS serves as a general anonymous signing primitive for other applications such as the Internet of Vehicles [29]–[31].

To establish a provenance lineage with an authentic and complete editing history across mutually untrusted editors, we propose an anonymous image provenance authentication (AIPA) scheme that formalizes the hash-chain design [15], [23] in the multi-editor setting. Built on top of UPCS, AIPA anchors

the lineage to one-time pseudonym keys to preserve signer privacy. Crucially, the hash-chain structure decouples adjacent proofs, allowing heterogeneous zk-SNARK proving systems to be composed, such as a lightweight one on user devices and a prover-heavy but succinct one on OSN platforms. Furthermore, the AIPA scheme can establish image provenance in a broader class of distributed scenarios beyond OSNs.

Finally, we implement an end-to-end AIPA workflow that establishes image provenance in OSNs, covering source signing, user editing, platform processing, public verification, and redissemiation. Leveraging AIPA's composability of heterogeneous proving systems, we instantiate user-side proving with the lightweight VIMz [15] and platform-side proving with our introduced VIMz-Loua, a GPU-accelerated proof system with constant-size proofs and low verification overhead, built from VIMz [15] and the Loua backend proposed in Eva [21]. In particular, we design a dedicated circuit for JPEG compression to cover platform-side processing within the provenance lineage. We evaluate the full pipeline in a simulated OSN environment following the platform workflows described in [24].

The main contributions of this paper are as follows:

- We propose and formalize UPCS, an unlinkable pseudonym certificate scheme based on anonymous credentials. UPCS provides signer anonymity even against the issuer and CA, supports selective attribute disclosure, certificate preprocessing, batch issuance, and X.509 compatibility, which reduces online signing and verification to ordinary digital signature operations taking only 0.017 and 0.066 ms. We formally prove the security of UPCS.
- We propose and formalize AIPA, an anonymous image provenance authentication scheme based on UPCS, zk-SNARKs, and hash chains. AIPA establishes a publicly verifiable image provenance lineage across untrusted editors. It has two key properties: (i) signer anonymity via UPCS throughout the lineage, and (ii) composability of heterogeneous proving systems across editing steps. We formally prove the security of AIPA.
- We implement an end-to-end AIPA workflow for image dissemination in OSNs. On the platform side, we instantiate the prover with our GPU-accelerated VIMz-Loua, which produces constant-size proofs and includes dedicated circuits for platform operations such as JPEG compression. Each VIMz-Loua proof remains a constant 448 B and verify in 6–12 ms across image sizes. In a simulated OSN environment, the provenance lineage of an image is only 40.2 KB (6.5% of the published image size) and can be verified in 4.28 s.

The rest of this paper is organized as follows. Section II reviews related work. Section III reviews preliminaries. Section IV formalizes UPCS and presents its construction and security analysis. Section V defines AIPA and presents its construction and security analysis. Section VI describes the end-to-end OSN workflow. Section VII reports implementation and evaluation results, and Section VIII concludes.

II. RELATED WORK

We review related work on pseudonym certificates, anonymous credentials, and cryptographic image authentication.

A. Pseudonym Certificate and Anonymous Credentials

Pseudonym certificates, primarily used in Internet of Vehicles [29]–[31], decouple the long-term identity from the public key certificate used for signing. Verifiers cannot infer the signer’s identity or link signers via the certificate. However, the certificate issuer must first verify the requester’s legitimate identity before issuing a certificate, yet must remain unable to trace the legitimate requester afterward. This is a conflict. Existing solutions [29]–[31] adopt a separation of duties principle, delegating long-term identity verification and pseudonym certificate issuance to a registration authority and pseudonym certificate authority (PCA), respectively, thus protecting identity privacy. However, this design paradigm cannot fundamentally resist the security threats posed by collusion among authorities. We therefore propose UPCS, which uses anonymous credentials as long-term identities. When requesting a certificate, the PCA verifies the anonymous credential to confirm the user’s identity. Even if the credential issuer and the PCA collude, the anonymity of the identity remains unbroken, preventing linkage of users.

Anonymous credentials [27], [28] are an important privacy-preserving technique for identity protection, offering features such as issuer hiding [32], aggregate credentials [33], and post-quantum signatures [34]. In UPCS, we explore combining anonymous credentials as a long-term identity layer with pseudonym certificates to achieve an efficient anonymous signature system that remains compatible with existing public key infrastructure (PKI). Moreover, the design paradigm of UPCS is compatible with the aforementioned anonymous credential systems, without being tied to any specific implementation.

B. Cryptographic Image Authentication

Table I compares the related protocols. PhotoProof [14] first proposed and defined cryptographic image authentication, but has the high computational cost of prover. ZK-IMG [23] and zk-REAL [35] use the Halo2 proof system [36] to reduce the prover cost. VerITAS [17] uses the lattice hash and lookup arguments to reduce prover time. To lower the memory required for proof generation, Trust Nobody [20] and VIMz [15] use tiling and folding [37] techniques, respectively, with a peak memory of only a few gigabytes. HyperVerITAS [22] defines image transformations as affine functions and builds customized multilinear interactive oracle proofs, which significantly reduces the time of proof generation. Extending from image to video, Eva [21] introduces the first cryptographic video authentication, supporting H.264 compression proofs, with efficient Groth16 proofs [38]. Recently, Bai et al. [39] have aimed to reduce the memory overhead of video proofs. Zk-Cinema [40] proposed the two-sided Freivald’s algorithm, to accelerate proof generation.

More recently, Zhang et al. proposed SPEG [41], a proof system dedicated to JPEG compression that features efficient proof generation, low memory usage, and constant-size proofs (28.5 KB). By contrast, VIMz-Loua targets the compression setting on the OSN platform side: the platform applies uniform JPEG compression to every uploaded image and possesses abundant computing resources, yet it is highly sensitive to

TABLE I
COMPARISON OF CRYPTOGRAPHIC IMAGE AND VIDEO AUTHENTICATION PROTOCOLS.^a

Protocol	Format	Compression	Provertime	Prover RAM	Proof ^b size
PhotoProof [14]	Image	Lossless	$O(P^3 \log P)$	$O(P)$	$O(1)$ (2.67 KB) $O(\log P)$
ZK-IMG [23]	Image	Lossless	$O(P \log P)$	$O(P)$	$(\approx 10 \text{ KB})$ $O(\log^2 N)$
VIMz [15]	Image	Lossless	$O(P)$	$O(N)$	$(\approx 10 \text{ KB})$ $O(\log^2 P)$
VerITAS [17]	Image	Lossless	$O(P \log P)$	$O(P)$	$(\approx 100 \text{ KB})$ $O(\log P)$
HyperVerITAS [22]	Image	Lossless	$O(P)$	$O(P)$	$(\approx 50 \text{ KB})$
Trust Nobody [20]	Image	Lossless	$O(P \log(P/T))$	$O(P/T)$	$O(T)$ ($\approx 10 \text{ KB}$)
Eva [21]	Video	H.264	$O(P)$	$O(1)$	$O(1)$ (448 B)
SPEG [41]	Image	JPEG	$O(P)$	$O(1)$	$O(1)$ (28.5 KB)
VIMz-Loua	Image	JPEG	$O(P)$	$O(N)$	$O(1)$ (448 B)

^aAsymptotic complexity is measured in the pixel count $P = M \times N \times L$ (height $\times$ width $\times$ frames) and the tile count T (used by Trust Nobody [20]); ^bparenthesized values are concrete sizes from our measurements or the original papers.

the storage and transmission overhead of proofs. VIMz-Loua therefore supports JPEG compression proofs, accelerates proof generation with GPUs, and keeps proofs as small as 448 B.

III. PRELIMINARIES

We adapt the formal definitions of images, transformations, and editing relations from prior work [14], [15], [20], [23]. The underlying cryptographic primitives, i.e., digital signatures, hash functions, and zk-SNARKs, are formally defined in Appendices A and B of the supplementary material.

A. Images and Transformations

Image and Metadata. Let $\mathcal{I}_{N,M}$ denote the set of all possible $N \times M$ images. An image $I \in \mathcal{I}_{N,M}$ is a pixel matrix $I \in \{0, 1, \dots, 255\}^{3 \times N \times M}$ of size $N \times M$, where each pixel is represented by an RGB value ranging from 0 to 255. Metadata meta covers image information such as source types (e.g., a capture device or a generative model) and recording time.

Transformation. An image transformation f is a function $f : \mathcal{I}_{N,M} \times \{0, 1\}^* \rightarrow \mathcal{I}_{N',M'}$ that maps an image and an optional parameter to a transformed image (e.g., resize, crop, compression). Let $\mathcal{F} = \{f_1, f_2, \dots, f_{|\mathcal{F}|}\}$ be the set of permissible transformations.

Original image. Let $\Psi = (G_s, S_s, V_s)$ be a digital signature scheme. For an image tuple (I, meta) , a public key pk , and a signature σ , we say that (I, meta) is *original* with respect to pk if $V_s(\text{pk}, (I, \text{meta}), \sigma) = 1$. This guarantees both data integrity (the image and metadata have not been altered) and source authenticity (it originated from the claimed signer pk).

Authentic image. Given a permissible transformation set $\mathcal{F}$ and a public key pk , we say that $\hat{I}$ is *authentic* with respect to $(\mathcal{F}, \text{pk})$, if there exist an *original* image tuple (I_0, meta) and a signature σ such that $V_s(\text{pk}, (I_0, \text{meta}), \sigma) = 1$ and $\hat{I} = f_n \circ \dots \circ f_1(I_0)$, with each $f_i \in \mathcal{F}$, where $\circ$ denotes function composition. An authentic image $\hat{I}$ is obtained by applying the sequence of transformations $\vec{f} = [f_1, \dots, f_n]$ on an original image I_0 .

Authentic Provenance. For an authentic image $\hat{I}$, its authentic provenance is a tuple $e = (I_0, \text{meta}, \text{pk}, \sigma, \vec{f})$.

B. Cryptographic Image Authentication

Cryptographic image authentication [14] lets a verifier be convinced that a public edited image $\hat{I}$ was produced by $\vec{f} = [f_1, \dots, f_n]$ applied to an original image I_0 , without revealing I_0 . The statement is captured by the relation $\mathcal{R}$:

$$\mathcal{R} = \left\{ \left\{ (\text{pk}, \sigma, \vec{f}, \hat{I}, \text{meta}); (I_0) \right\} : \begin{array}{l} \hat{I} = f_n \circ \dots \circ f_1(I_0) \wedge \\ \Psi.V_s(\text{pk}, H(I_0 \parallel \text{meta}), \sigma) = 1 \end{array} \right\} \quad (1)$$

where V_s is the verification algorithm of Ψ and H is a collision-resistant hash function. The relation establishes that $\hat{I}$ is authentic (derived from I_0 via $\vec{f}$) and that (I_0, meta) is original with respect to pk .

Unlike $\mathcal{R}$, which treats the whole edit sequence as one monolithic statement, the hash-chain method [15], [23] decomposes it into per-step proofs: each transformation is asserted by its own zk-SNARK proof, and the hash values of intermediate images form a public hash chain. The full provenance relation $\mathcal{R}_{\vec{f}}$ and the per-step relation $\mathcal{R}_f$ are defined in Eq. (2) and Eq. (3):

$$\mathcal{R}_{\vec{f}} = \left\{ \left\{ (\text{pk}, \sigma, \vec{f}, \hat{I}, \text{HC}, \text{meta}); (I_0, \dots, I_n) \right\} : \begin{array}{l} \Psi.V_s(\text{pk}, H(h_0 \parallel \text{meta}), \sigma) = 1 \wedge h_0 = H(I_0) \\ \wedge \hat{I} = I_n \wedge \text{HC} = [h_0, \dots, h_n] \\ \wedge \forall i \in [1, n] : h_i = H(I_i) \wedge I_i = f_i(I_{i-1}) \end{array} \right\} \quad (2)$$

$$\mathcal{R}_f = \left\{ \left\{ ((h_{i-1}, h_i, f); (I_{i-1}, I_i)) : I_i = f(I_{i-1}) \right\} \wedge h_{i-1} = H(I_{i-1}) \wedge h_i = H(I_i) \right\} \quad (3)$$

Here, $\text{HC} = [h_0, \dots, h_n]$ is the public hash chain, where $h_i = H(I_i)$, and $I_0, \dots, I_n$ are the secret intermediate images. The hash chain is illustrated in Fig. 2. Starting from $h_0 = H(I_0)$, each step i produces $I_i = f_i(I_{i-1})$, computes $h_i = H(I_i)$, and generates π_i for (h_{i-1}, h_i, f_i) . Adjacent proofs π_i and π_{i+1} share h_i , forming a chain: tampering with any I_i would change h_i and invalidate both proofs.

Intuitively, π_1 attests that h_0 and h_1 are the hashes of images I_0 and I_1 , respectively, with $I_1 = f_1(I_0)$, while π_2 attests that h_1 and h_2 are the hashes of images I_1' and I_2 with $I_2 = f_2(I_1')$. The junction hash h_1 therefore refers both to the output image of the first step and to the input image of the second. Since H is collision-resistant, no efficient adversary can exhibit $I_1' \neq I_1$ such that $H(I_1') = H(I_1) = h_1$. Hence, except with negligible probability, $I_1' = I_1$, and $I_2 = f_2(f_1(I_0))$. By induction, the chain of proofs certifies $\hat{I} = f_n \circ \dots \circ f_1(I_0)$. Moreover, each π_i depends only on its own statement (h_{i-1}, h_i, f_i) and witness (I_{i-1}, I_i) , not on the proofs of the other steps. We observe that this independence enables different steps to be proved by different, mutually untrusted editors, possibly even with different zk-SNARK systems, as long as each step proves the same relation $\mathcal{R}_f$. Prior hash-chain-based schemes [15], [23] focused on a single editor and did not discuss this consequence, which AIPA (Definition 4) exploits to extend provenance across mutually untrusted editors.

Verification of $\mathcal{R}_{\vec{f}}$ thus reduces to two checks: (i) a signature binding the chain to the original image, and (ii) n independent instances of $\mathcal{R}_f$. Proving $\mathcal{R}_f$ for each step therefore suffices

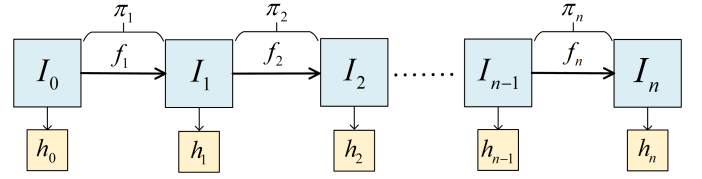


Fig. 2. The hash chain and proof chain. Each proof π_i attests to the per-step relation $\mathcal{R}_f$ of Eq. (3): (h_{i-1}, h_i, f_i) holds with witness (I_{i-1}, I_i) . The hash values $h_i = H(I_i)$ link adjacent transformations through the shared h_i between π_i and π_{i+1} .

to establish end-to-end authenticity, while the one-wayness of H and the zero-knowledge property of the underlying proof system ensure that all intermediate images remain hidden.

We denote by $\mathbf{IM} = (\text{Setup}, \text{Prove}, \text{Verify})$ a zk-SNARK proof system for $\mathcal{R}_f$:

- $\text{Setup}(1^\lambda, \mathcal{R}_f) \rightarrow (\text{pk}_f, \text{vk}_f)$: generates proving and verification keys for $\mathcal{R}_f$.
- $\text{Prove}(\text{pk}_f, (h_{i-1}, h_i, f), (I_{i-1}, I_i)) \rightarrow \pi_i$: produces a proof for a single transformation step.
- $\text{Verify}(\text{vk}_f, (h_{i-1}, h_i, f), \pi_i) \rightarrow \{0, 1\}$: verifies the proof.

C. Anonymous Credentials

An anonymous credential (AC) scheme allows a user to obtain a credential cred on an attribute set $M_n = \{m_j\}_{j=1}^n$ from an issuer, and later prove possession of cred while selectively disclosing only a subset $M_D = \{m_j\}_{j \in D}$ ($D \subseteq [1, n]$), without revealing the user's identity, the remaining attributes, or linking multiple presentations. The formal syntax and security model are given in the identity layer of UPCS (Definition 1 and Appendix C of the supplementary material).

IV. UNLINKABLE PSEUDONYM CERTIFICATE SCHEME

This section presents UPCS, the identity and certificate layer for our image provenance authentication scheme.

Pseudonym certificate. Unlike a conventional certificate that binds a public key to a real-world identity, a pseudonym certificate carries only auxiliary information aux (e.g., issuer identity and validity period) and a *one-time* public key pk_p , signed by the CA as σ_{CA} , without containing any long-term user identifier [29], [30]. Formally, $\text{cert} = (\text{pk}_p, \text{aux}, \sigma_{\text{CA}})$. Both the pseudonym certificate and the corresponding signing key pair must be one-time, being essential for certificate unlinkability.

High-level workflow. The UPCS system involves four participants: an Anonymous Credential Issuer (Issuer), a Certificate Authority (CA), a User, and a Certificate Verifier (CV). The protocol operates in two layers (Fig. 3):

- *Identity layer.* The User obtains a long-term anonymous credential $\{\text{cred}\}$ from the Issuer via the AC system [27], certifying the User's attributes without revealing a persistent identifier.
- *Certificate layer.* When signing a message m , the User generates a fresh one-time key pair $(\text{sk}_p, \text{pk}_p)$, anonymously presents $\{\text{cred}, \text{sk}_p\}$ to the CA, and receives a one-time pseudonym certificate $\{\text{cert}\}$ on pk_p . The User signs m with sk_p and presents $\{m, \sigma, \text{cert}\}$ to the CV,

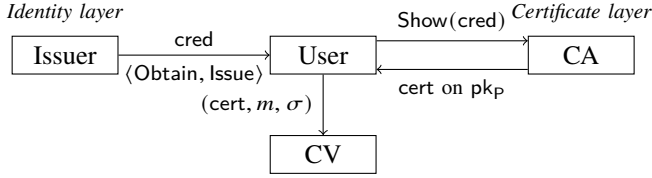


Fig. 3. The two-layer workflow of UPCS.

who verifies the CA's signature on cert and then (m, σ) under pk_P . The CV confirms message authenticity but cannot link cert to the User's long-term identity.

Threat model. For *identity privacy*, the Issuer and CA are malicious and may collude with each other and with arbitrary users, while the target user remains honest. For *certificate and credential unforgeability*, the Issuer and CA are honest, while an arbitrary subset of users may be corrupt.

We formalize UPCS in the AC framework [27], [28].

A. Syntax

For clarity, we present the syntax with a single Issuer and a single CA. Since each authority holds independent signing keys, extension to multiple parties is straightforward [27], [28].

A credential is issued on the full attribute set M_n , and the user may later disclose any subset $M_D \subseteq M_n$ while hiding the rest. The certificate layer employs a standard signature scheme $\Psi = (G_s, S_s, V_s)$.

Definition 1 (Unlinkable Pseudonym Certificate Scheme (UPCS)). Given Ψ , a UPCS consists of the following identity-layer and certificate-layer algorithms.

Identity layer.

- $IKeygen(1^\lambda, n) \rightarrow (sk_I, pk_I)$: outputs the Issuer's key pair.
- $UKeygen(pk_I) \rightarrow (sk_U, pk_U)$: outputs a User's long-term key pair.
- $\langle Obtain(sk_U, pk_I, M_n), Issue(pk_U, sk_I, M_n) \rangle$: To obtain an anonymous credential on sk_U and M_n , the user running Obtain, interacts with the issuer, running Issue. Finally, Obtain output a long-term credential cred or $\perp$.
- $\langle Show(sk_U, pk_I, M_n, M_D, cred), ACVerify(pk_I, M_D) \rangle$: To prove that a subset M_D of the attributes M_n has been certified, the user running Show interacts with the verifier running ACVerify. Finally, ACVerify output 1 or 0.

Certificate layer.

- $CAKeygen(1^\lambda) \rightarrow (sk_{CA}, pk_{CA})$: runs $\Psi.G_s(1^\lambda)$ and outputs the CA's signing key pair.
- $PseuKeyGen(1^\lambda) \rightarrow (sk_P, pk_P)$: runs $\Psi.G_s(1^\lambda)$ and outputs a fresh one-time signing key pair.
- $\langle CObtain(sk_U, pk_I, M_n, M_D, cred, pk_{CA}), CIssue(pk_I, pk_P, M_D, sk_{CA}) \rangle \rightarrow cert$: Interactive protocol between User and CA. The User anonymously proves possession of cred (via Show). The CA verifies (via ACVerify) and, if valid, issues $cert = (pk_P, aux, \sigma_{CA})$, where $\sigma_{CA} \leftarrow \Psi.S_s(sk_{CA}, pk_P || aux)$. CObtain and CIssue output cert on success, $\perp$ otherwise.
- $\langle CShow(sk_P, cert, m), CVerify(pk_{CA}, m) \rangle \rightarrow \{0, 1\}$: The User computes $\sigma \leftarrow \Psi.S_s(sk_P, m)$ and sends $(cert, m, \sigma)$.

CVerify outputs 1 iff $\Psi.V_s(pk_{CA}, pk_P || aux, \sigma_{CA}) = 1$ and $\Psi.V_s(pk_P, m, \sigma) = 1$.

B. Security Model

The security model follows the game-based framework of [27], [28], extended with certificate-layer oracles. Beyond *correctness*, UPCS must satisfy *certificate unforgeability* and *certificate unlinkability* (analogous to unforgeability and anonymity in anonymous credentials).

The following global lists are maintained: HU (honest users), CU (corrupt users), L_{cred} of issued credentials $(i, cred, M_n)$, L_{cert} of issued certificates $(i, pk_P, aux, \sigma_{CA})$, and L_P of certificates already presented (to prevent reuse), and L_{sk_P} recording the secret key sk_P corresponding to each entry in L_{cert} . Following [27], [28], we adopt standard identity-layer oracles for user registration (O_{HU} , O_{CU}), credential issuance under honest and corrupt participants (O_{ObtIss} , O_{Issue} , O_{Obtain}), and credential presentation (O_{Show}). Full definitions appear in Appendix C of the supplementary material.

The following certificate-layer oracles are specific to UPCS:

- $O_{CObtIss}(i, M_D)$: if $i \notin HU$, returns $\perp$; else, samples $(sk_P, pk_P) \leftarrow PseuKeyGen(1^\lambda)$, runs $\langle CObtain(sk_U^i, pk_I, M_n, M_D, cred, pk_{CA}) \rangle$, stores sk_P in L_{sk_P} , adds $(i, pk_P, aux, \sigma_{CA})$ to L_{cert} , and returns $(pk_P, cert)$.
- $O_{CIssue}(i, M_D, pk_P)$: if $i \notin CU$, returns $\perp$; else, runs $CIssue(pk_I, pk_P, M_D, sk_{CA})$ with $\mathcal{A}$ as the malicious user. If it succeeds, stores $(i, pk_P, aux, \sigma_{CA})$ in L_{cert} ; else returns $\perp$. Models an adversary playing a malicious user obtaining a certificate from an honest CA.
- $O_{CObtain}(i, M_D)$: if $i \notin HU$, returns $\perp$; else, samples $(sk_P, pk_P) \leftarrow PseuKeyGen(1^\lambda)$ and runs $CObtain(sk_U^i, pk_I, M_n, M_D, cred, pk_{CA})$ with $\mathcal{A}$ as the malicious CA. If $cert = \perp$, returns $\perp$; else stores sk_P in L_{sk_P} , adds $(i, pk_P, aux, \sigma_{CA})$ to L_{cert} , and returns $(pk_P, cert)$. Models an adversary impersonating the CA to issue a certificate to an honest user.
- $O_{CShow}(j, m)$: let $(i, pk_P, aux, \sigma_{CA})$ be the j -th entry in L_{cert} , and set $cert = (pk_P, aux, \sigma_{CA})$. If $i \notin HU$ or $cert \in L_P$, returns $\perp$; else retrieves sk_P from L_{sk_P} and runs $CShow(sk_P, cert, m)$ with $\mathcal{A}$ as a malicious verifier, and adds $cert$ to L_P to prevent reuse.

Correctness requires that ACVerify and CVerify output 1 for all honestly generated keys and honestly executed protocols, following the correctness of the underlying AC scheme (see Appendix C of the supplementary material). The *unforgeability* and *anonymity* of long-term credentials follow from the underlying AC scheme (also defined in Appendix C of the supplementary material) [27], and message-signature unforgeability follows from Ψ . We now define the two certificate-layer properties specific to UPCS, with the corresponding security experiments summarized in Fig. 4.

Definition 2 (Certificate Unforgeability). A UPCS system satisfies *certificate unforgeability* if, for any PPT adversary $\mathcal{A}$ with adaptive oracle access to $\mathcal{O} = \{O_{HU}, O_{CU}, O_{ObtIss}, O_{Issue}, O_{Show}, O_{CObtIss}, O_{CIssue}, O_{CShow}\}$, its advantage $\text{Adv}_{\mathcal{A}}^{\text{Cert-uf}}$ of

Certificate Unforgeability. $\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}(1^\lambda, n)$:

1. $(\text{sk}_1, \text{pk}_1, \text{sk}_{\text{CA}}, \text{pk}_{\text{CA}}) \leftarrow \text{IKeyGen}(1^\lambda, n), \text{CAKeyGen}(1^\lambda)$.
2. $\text{cert}^* = (\text{pk}_p^*, \text{aux}^*, \sigma_{\text{CA}}^*) \leftarrow \mathcal{A}^O(\text{pk}_1, \text{pk}_{\text{CA}})$.
3. Return 1 if and only if:
 - $\Psi.V_s(\text{pk}_{\text{CA}}, \text{pk}_p^* \parallel \text{aux}^*, \sigma_{\text{CA}}^*) = 1$;
 - $\nexists (\cdot, \text{pk}_p^*, \text{aux}^*, \cdot) \in \mathcal{L}_{\text{cert}}$.
4. Otherwise return 0.

Certificate Unlinkability. $\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-b}}(1^\lambda, n)$:

1. $(\text{sk}_1, \text{pk}_1, \text{sk}_{\text{CA}}, \text{pk}_{\text{CA}}) \leftarrow \text{IKeyGen}(1^\lambda, n), \text{CAKeyGen}(1^\lambda)$.
2. $(i_0, i_1, M_D) \leftarrow \mathcal{A}^O(\text{sk}_1, \text{sk}_{\text{CA}})$.
3. If $i_0 = i_1$, or $\{i_0, i_1\} \not\subseteq \text{HU}$, or for some $i \in \{i_0, i_1\}$ there is no entry $(i, \text{cred}, M_n) \in \mathcal{L}_{\text{cred}}$ with $M_D \subseteq M_n$, return 0.
4. Sample $b \leftarrow \{0, 1\}$.
5. $(\text{pk}_{p0}, \text{cert}_0) \leftarrow \mathcal{O}_{\text{CObtain}}(i_0, M_D)$;
 $(\text{pk}_{p1}, \text{cert}_1) \leftarrow \mathcal{O}_{\text{CObtain}}(i_b, M_D)$.
6. $b^* \leftarrow \mathcal{A}^O(\text{sk}_1, \text{sk}_{\text{CA}}, \text{cert}_0, \text{cert}_1)$.
7. If $i_0 \in \text{CU}$ or $i_1 \in \text{CU}$, return 0. Else return b^* .

Fig. 4. Security experiments for the UPCS.

playing the security game above is negligible in λ , where $\text{Adv}_{\mathcal{A}}^{\text{Cert-uf}}(1^\lambda, n) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}(1^\lambda, n) = 1]$.

Definition 3 (Certificate Unlinkability). A UPCS system satisfies *certificate unlinkability* if, for any PPT adversary $\mathcal{A}$ with adaptive oracle access to $\mathcal{O} = \{\mathcal{O}_{\text{HU}}, \mathcal{O}_{\text{CU}}, \mathcal{O}_{\text{Obtain}}, \mathcal{O}_{\text{Show}}, \mathcal{O}_{\text{CObtain}}, \mathcal{O}_{\text{CShow}}\}$, its advantage $\text{Adv}_{\mathcal{A}}^{\text{Cert-unl}}$ in the security game above is negligible in λ , where $\text{Adv}_{\mathcal{A}}^{\text{Cert-unl}}(1^\lambda, n) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-1}}(1^\lambda, n) = 1] - \Pr[\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-0}}(1^\lambda, n) = 1]|$.

C. Construction of UPCS

We instantiate the identity layer using the AC construction of [27], which can be made non-interactive via the Fiat–Shamir heuristic [42]. Let $\text{Gen}(1^\lambda)$ be a bilinear group generator that returns $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a type-3 pairing over groups of prime order p , and g_1, g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. The construction is given in Fig. 5.

D. Certificate Preprocessing and Batch Issuance

Consider a user who posts multiple signed images to an OSN over time. Our protocol design ensures that the user need not contact the CA for every image.

Preprocessing. The UPCS certificate layer decouples credential presentation from message signing through as follows:

- 1) *Certificate request.* When idle, the user repeatedly invokes $\langle \text{CObtain}, \text{CIssue} \rangle$, each time anonymously presenting credential cred to the CA and obtaining one pseudonym certificate cert_i on a fresh one-time public key $\text{pk}_{p,i}$, thereby collecting a pool of certificates $\{\text{cert}_i\}_{i=1}^k$.
- 2) *Local storage.* The user stores the certificates $\{\text{cert}_i\}_{i=1}^k$ together with the secret keys $\{\text{sk}_{p,i}\}_{i=1}^k$ locally.
- 3) *Online Message signing.* When the user later needs to sign a message m , they select an unused certificate cert and its key sk_p from the local pool, compute $\sigma \leftarrow \Psi.S_s(\text{sk}_p, m)$, and present (m, σ, cert) . The online signing cost is reduced to a single plain signature.

Since credential presentation dominates the cost, this shifts the expensive cryptographic work to the certificate-request phase.

Batch issuance. Preprocessing is further accelerated by batch issuance: the user performs credential presentation once

Unlinkable Pseudonym Certificate Scheme:

Identity layer (following [27]).

$\text{IKeyGen}(1^\lambda, n)$: sample $x, y_0, \dots, y_n \leftarrow \mathbb{Z}_p$, compute $X = g_1^x$, $Y_i = g_1^{y_i}$, $\tilde{Y}_i = g_2^{y_i}$, $Z_{i,j} = g_1^{y_i y_j}$ ($0 \leq i \neq j \leq n$), and output $\text{sk}_1 = (x, y_0, \dots, y_n)$ and $\text{pk}_1 = (X, \{Y_i, \tilde{Y}_i\}_{i=0}^n, \{Z_{i,j}\})$.

$\text{UKeyGen}(\text{pk}_1)$: sample $\text{sk}_U \leftarrow \mathbb{Z}_p$ and set $\text{pk}_U = g_2^{\text{sk}_U}$.

$\langle \text{Obtain}, \text{Issue} \rangle$: The user sends pk_U together with a Schnorr proof of knowledge of sk_U . The issuer samples $r \leftarrow \mathbb{Z}_p$ and returns $\text{cred} = (\tilde{\sigma}_1, \tilde{\sigma}_2) = (g_2^r, \text{pk}_U^{r y_0} \cdot g_2^{r(x + \sum_{i=1}^n y_i m_i)})$.

$\langle \text{Show}, \text{ACVerify} \rangle$ (disclosing M_D , let $D_0 = \{0\} \cup D$):

- User: sample $k, \rho, t \leftarrow \mathbb{Z}_p$, compute $\sigma'_1 = g_1^t \cdot \prod_{j \notin D} Y_j^{m_j}$, $\sigma'_2 = (\prod_{i \in D_0} Y_i)^t \cdot \prod_{i \in D_0, j \notin D} Z_{i,j}$, $\tilde{\sigma}'_1 = \tilde{\sigma}_1^\rho$, $\tilde{\sigma}'_2 = \tilde{\sigma}_2^\rho$. $(\tilde{\sigma}'_1)^t$, $C = e(Y_0^k, \tilde{\sigma}'_1)$, and send $(\sigma'_1, \sigma'_2, \tilde{\sigma}'_1, \tilde{\sigma}'_2, C)$.
- Verifier: sample $c \leftarrow \mathbb{Z}_p$ and send c .
- User: compute $s = k + c \cdot \text{sk}_U \bmod p$ and send s .
- Verifier: let $B = e(X \cdot \sigma'_1 \cdot \prod_{i \in D} Y_i^{m_i}, (\tilde{\sigma}'_1)^{-1})$ and accept iff $e(Y_0^s, \tilde{\sigma}'_1) \cdot C^{-1} = [B \cdot e(g_1, \tilde{\sigma}'_2)]^c$ and $e(\sigma'_1, \prod_{i \in D_0} \tilde{Y}_i) = e(\sigma'_2, g_2)$.

Certificate layer.

$\text{CAKeyGen}(1^\lambda)$: run $\Psi.G_s(1^\lambda)$ and output $(\text{sk}_{\text{CA}}, \text{pk}_{\text{CA}})$.

$\text{PseuKeyGen}(1^\lambda)$: run $\Psi.G_s(1^\lambda)$ and output a fresh one-time key pair $(\text{sk}_p, \text{pk}_p)$.

$\langle \text{CObtain}, \text{CIssue} \rangle$: The user runs Show with the CA as the verifier. On acceptance, the CA sets aux , computes $\sigma_{\text{CA}} \leftarrow \Psi.S_s(\text{sk}_{\text{CA}}, \text{pk}_p \parallel \text{aux})$, and returns $\text{cert} = (\text{pk}_p, \text{aux}, \sigma_{\text{CA}})$.

$\langle \text{CShow}, \text{CVerify} \rangle$: The user computes $\sigma \leftarrow \Psi.S_s(\text{sk}_p, m)$ and sends (cert, m, σ) . The verifier accepts iff $\Psi.V_s(\text{pk}_{\text{CA}}, \text{pk}_p \parallel \text{aux}, \sigma_{\text{CA}}) = 1$ and $\Psi.V_s(\text{pk}_p, m, \sigma) = 1$.

Fig. 5. The UPCS construction.

and requests k pseudonym certificates on k distinct one-time public keys in a single interaction with the CA. The CA verifies the presentation once, then signs each public key individually, amortizing the dominant credential cost over k certificates.

Privacy trade-off. Certificates obtained within the same batch are linkable to the same presentation session, but the user's long-term identity remains fully anonymous. Cross-batch unlinkability follows from the underlying AC scheme via fresh randomization per batch, making sessions mutually unlinkable. The user may adjust the batch size to balance efficiency against the desired unlinkability granularity.

Modularization. The separation of the identity and certificate layers also extends system functionality. The identity layer can support various anonymity properties through different anonymous credential techniques (e.g., issuer hiding, credential aggregation [33], and post-quantum instantiations [34]). Crucially, the certificate layer of UPCS can reuse existing PKI infrastructure and supports the X.509 standard, lowering the deployment barrier of the scheme in practice.

E. Security Analysis

The certificate unforgeability and certificate unlinkability of the construction are proven in Appendix D of the supplementary material.

V. ANONYMOUS IMAGE PROVENANCE AUTHENTICATION

The AIPA scheme establishes an end-to-end, publicly verifiable provenance lineage for an image as it is edited and re-shared across multiple mutually untrusted parties. It verifies the provenance authenticity of an image while keeping the identity of its source signer anonymous.

The scheme operates in four stages. First, the Authority generates the system keys via Gen_{IA} , publishing the proving and verification parameters. A signer holding a valid pseudonym certificate then invokes Sig_{IA} to sign the original image, anchoring the provenance lineage. Any editor may subsequently apply a permissible transformation and invoke Prove_{IA} to extend the lineage with a zk-SNARK proof. Finally, any verifier can independently audit the end-to-end provenance by running $\text{Verify}_{\text{IA}}$, while keeping the signer anonymous.

The scheme guarantees three properties: (i) *source anonymity*: images are authenticated without revealing the signer's long-term identity, inherited from UPCS; (ii) *editing integrity*: every transformation is bound to a publicly verifiable proof; and (iii) *image privacy*: intermediate images remain hidden throughout the editing pipeline.

A. Syntax

Definition 4 (Anonymous Image Provenance Authentication (AIPA) Scheme). Let UPCS be a UPCS scheme and $\Psi = (G_s, S_s, V_s)$ a digital signature scheme. An AIPA scheme is a tuple of polynomial-time algorithms $\text{AIPA} = (\text{Gen}_{\text{IA}}, \text{Sig}_{\text{IA}}, \text{Prove}_{\text{IA}}, \text{Verify}_{\text{IA}})$ defined over UPCS and Ψ .

The credential and certificate lifecycle is handled entirely by UPCS and not redefined here. In the image-signing context, we write $(\text{sk}_{\text{img}}, \text{pk}_{\text{img}})$ for the one-time pseudonym key pair $(\text{sk}_P, \text{pk}_P)$ of UPCS. The AIPA algorithms are:

- $\text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F}) \rightarrow (\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$: Run once by the Authority. Outputs the public proving key pk_{IA} for all $f \in \mathcal{F}$, the public verification key vk_{IA} (containing pk_I , pk_{CA} , and the verification keys for $\mathcal{F}$), and a system secret key $\text{sk}_{\text{sys}} = (\text{sk}_I, \text{sk}_{\text{CA}})$.
- $\text{Sig}_{\text{IA}}(\text{sk}_{\text{img}}, I_0, \text{meta}) \rightarrow (\pi_0, \sigma_{\text{img}})$: Run by the image signer. The input sk_{img} is the secret key corresponding to a pseudonym certificate cert on pk_{img} , previously obtained via UPCS. $\langle \text{CObtain}, \text{CIssue} \rangle$ (which builds on a long-term credential from UPCS. $\langle \text{Obtain}, \text{Issue} \rangle$). On input $(\text{sk}_{\text{img}}, I_0, \text{meta})$, outputs an image signature σ_{img} and the initial proof state π_0 .
- $\text{Prove}_{\text{IA}}(\text{pk}_{\text{IA}}, I_{\text{in}}, \pi_{\text{in}}, f) \rightarrow (I_{\text{out}}, \pi_{\text{out}})$: Run by any editor. Apply the transformation $I_{\text{out}} = f(I_{\text{in}})$, where $f \in \mathcal{F}$, and updates the proof state from π_{in} to π_{out} .
- $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \widehat{I}, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \pi) \rightarrow \{0, 1\}$: Run by any public verifier. cert and σ_{img} are verified through UPCS. CVerify . Returns 1 iff $\widehat{I}$ is authentic with respect to $(\mathcal{F}, \text{pk}_{\text{img}})$ and cert is a valid UPCS certificate for pk_{img} .

B. Security Model

The following properties must hold for the AIPA scheme, adapted from [14] and [20], with the corresponding security experiments summarized in Fig. 6.

Correctness. An AIPA scheme is *correct* if it is always possible to successfully prove the authenticity of an authentic image. That is, for every security parameter λ , every permissible transformation set $\mathcal{F}$, every original image tuple (I_0, meta) , and every sequence of permissible transformations $\vec{f} = [f_1, \dots, f_n] \in \mathcal{F}^n$, let $\widehat{I} = f_n \circ \dots \circ f_1(I_0)$. Then the following probability equals 1:

$$\Pr \left[\begin{array}{l} \text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \widehat{I}, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \pi) = 1 \\ (\pi_0, \sigma_{\text{img}}) \leftarrow \text{Sig}_{\text{IA}}(\text{sk}_{\text{img}}, I_0, \text{meta}), \\ (\widehat{I}, \pi) \leftarrow \text{prove}(I_0, \pi_0, \vec{f}) \end{array} \mid \begin{array}{l} (\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}}) \leftarrow \text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F}), \\ (\pi_0, \sigma_{\text{img}}) \leftarrow \text{Sig}_{\text{IA}}(\text{sk}_{\text{img}}, I_0, \text{meta}), \\ (\widehat{I}, \pi) \leftarrow \text{prove}(I_0, \pi_0, \vec{f}) \end{array} \right] = 1$$

Above, $\text{prove}(I_0, \pi_0, \vec{f})$ yields $(\widehat{I}, \pi)$ via the recurrence $(I_i, \pi_i) \leftarrow \text{Prove}_{\text{IA}}(\text{pk}_{\text{IA}}, I_{i-1}, \pi_{i-1}, f_i)$ for $1 \leq i \leq n$, starting from the original image I_0 . The credential cred , certificate cert , and signing key pair $(\text{sk}_{\text{img}}, \text{pk}_{\text{img}})$ are honestly obtained via UPCS, and their correctness follows from UPCS correctness.

Global Variables and Oracles. The adversary can access to all oracles of the underlying UPCS as defined in Section IV, denoted as $\mathcal{O}_{\text{UPCS}}$, which populate the global lists HU , CU , L_{cred} , L_{cert} , and L_P . We additionally define the following AIPA-specific lists and oracles. Let L_{sig} be a list of $(\text{pk}_{\text{img}}, \text{sk}_{\text{img}}, \text{cert})$ storing signing keys for each certificate issued via $\mathcal{O}_{\text{CObtIss}}$, and let S_{sig} be a list of $(\text{pk}_{\text{img}}, I_0, \text{meta}, \sigma_{\text{img}})$ recording every signature output by $\mathcal{O}_{\text{Sign}}$ below and every signature produced by $\mathcal{O}_{\text{CShow}}$ on the certificate of an honest user, recorded as $(\text{pk}_P, \perp, \perp, \sigma)$. Hence the third winning condition of $\text{Exp}_{\mathcal{A}}^{\text{Sig-uf}}$ excludes not only $\mathcal{O}_{\text{Sign}}$ outputs but also any signature σ_{img}^* that $\mathcal{A}$ obtained from $\mathcal{O}_{\text{CShow}}$ under pk_{img}^* .

- $\mathcal{O}_{\text{Sign}}(\text{pk}_{\text{img}}, I_0, \text{meta})$: if $(\text{pk}_{\text{img}}, \cdot, \cdot) \notin \text{L}_{\text{sig}}$, return $\perp$; else retrieve sk_{img} from L_{sig} , compute $(\pi_0, \sigma_{\text{img}}) \leftarrow \text{Sig}_{\text{IA}}(\text{sk}_{\text{img}}, I_0, \text{meta})$, adding $(\text{pk}_{\text{img}}, I_0, \text{meta}, \sigma_{\text{img}})$ to S_{sig} , and return $(\pi_0, \sigma_{\text{img}})$.

Definition 5 (Image Signature Unforgeability). An AIPA scheme satisfies *image signature unforgeability* if, for any PPT adversary $\mathcal{A}$ with adaptive oracle access to $\mathcal{O} = \mathcal{O}_{\text{UPCS}} \cup \mathcal{O}_{\text{Sign}}$, its advantage $\text{Adv}_{\mathcal{A}}^{\text{Sig-uf}}(1^\lambda) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{Sig-uf}}(1^\lambda, \mathcal{F}) = 1]$ is negligible in λ .

Definition 6 (Proof of Knowledge). An AIPA scheme satisfies *proof of knowledge* if for every PPT adversary $\mathcal{A}$ with adaptive oracle access to $\mathcal{O} = \mathcal{O}_{\text{UPCS}} \cup \mathcal{O}_{\text{Sign}}$, there exist a PPT extractor $\mathcal{E}$ and a negligible function negl such that $\Pr[\text{Exp}_{\mathcal{A}, \mathcal{E}}^{\text{PoK}}(1^\lambda, \mathcal{F}) = 1] \leq \text{negl}(\lambda)$.

In the PoK experiment, $D_{\mathcal{F}}$ is the procedure that returns 1 if and only if $\widehat{I}$ is authentic with respect to $(\mathcal{F}, \text{pk}_{\text{img}})$. Let $\vec{f} = [f_1, \dots, f_n]$. And $D_{\mathcal{F}}(\widehat{I}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \vec{f}, I_0, \text{meta})$ returns 1 iff all of the following hold:

- 1) $\Psi.V_s(\text{pk}_{\text{img}}, (I_0, \text{meta}), \sigma_{\text{img}}) = 1$;
- 2) $\widehat{I} = f_n \circ \dots \circ f_1(I_0)$, where $f_i \in \mathcal{F}$ for all $i \in [1, n]$.

Remark. In our PoK experiment, the extractor $\mathcal{E}$ receives the full system secret key sk_{sys} (including sk_I and sk_{CA}) generated by $\text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F})$. This corresponds to the *trivial case* discussed in Section 1.1 of the reference [43]. The extractor is given the oracle's secret state and can thus perfectly simulate all oracle queries internally. Consequently, the proof

Image Signature Unforgeability. $\text{Exp}_{\mathcal{A}}^{\text{Sig-uf}}(1^\lambda, \mathcal{F})$:

1. $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}}) \leftarrow \text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F})$.
2. $(\tilde{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*) \leftarrow \mathcal{A}^O(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}})$.
3. Return 1 iff:
 - $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \tilde{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*) = 1$;
 - $\exists (\text{pk}_{\text{img}}^*, \cdot, \cdot) \in \mathcal{L}_{\text{sig}}$;
 - $(\text{pk}_{\text{img}}^*, \cdot, \cdot, \sigma_{\text{img}}^*) \notin \mathcal{S}_{\text{sig}}$.

Proof of Knowledge. $\text{Exp}_{\mathcal{A}, \mathcal{E}}^{\text{PoK}}(1^\lambda, \mathcal{F})$:

1. $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}}) \leftarrow \text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F})$.
2. $(\tilde{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*) \leftarrow \mathcal{A}^O(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}})$.
3. $(I_0^*, \text{meta}^*, \tilde{f}^*) \leftarrow \mathcal{E}(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$.
4. Return 1 iff $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \tilde{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*) = 1$ and $D_{\mathcal{F}}(\tilde{T}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \tilde{f}^*, I_0^*, \text{meta}^*) = 0$.

Zero-Knowledge. $\text{Real}_{\mathcal{D}}^{\text{zk}}(1^\lambda, \mathcal{F})$:

1. $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}}) \leftarrow \text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F})$.
2. $(\text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \tilde{f}, \text{meta}, I_0, \dots, \tilde{T}) \leftarrow \mathcal{A}_1^O(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}})$.
3. Publicly derive the initial proof state π_0 from (I_0, meta) , then compute $(I_i, \pi_i) \leftarrow \text{Prove}_{\text{IA}}(\text{pk}_{\text{IA}}, I_{i-1}, \pi_{i-1}, f_i)$ for $i = 1, \dots, n$, and output $\pi = \pi_n$.
4. $b \leftarrow \mathcal{A}_2(\pi)$.
5. If $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \tilde{T}, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \pi) = 0$, return 0; else return b .

$\text{Ideal}_{\mathcal{D}}^{\text{zk}}(1^\lambda, \mathcal{F})$:

1. $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}}, \text{td}) \leftarrow \mathcal{S}_{\text{key}}(1^\lambda, \mathcal{F})$, where td is a trapdoor.
2. $(\text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \tilde{f}, \text{meta}, I_0, \dots, \tilde{T}) \leftarrow \mathcal{A}_1^O(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}})$.
3. Publicly derives, from $\mathcal{A}_1$'s output, the public instance x of the provenance relation (i.e., the public statements of all n editing steps), and runs $\pi \leftarrow \mathcal{S}_{\text{prove}}(\text{pk}_{\text{IA}}, \text{td}, x)$. (Note: $\mathcal{S}_{\text{prove}}$ receives only the instance x , not the witness $w = (I_0, \dots, I_{n-1})$.)
4. $b \leftarrow \mathcal{A}_2(\pi)$.
5. If $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \tilde{T}, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \pi) = 0$, return 0; else return b .

Fig. 6. Security experiments for the AIPA scheme.

of knowledge in our AIPA scheme reduces to standard SNARK extraction, without requiring the O-SNARK machinery.

Definition 7 (Zero-Knowledge). An AIPA scheme satisfies *zero-knowledge* if there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_{\text{key}}, \mathcal{S}_{\text{prove}})$ such that for every PPT distinguisher $\mathcal{D} = (\mathcal{A}_1, \mathcal{A}_2)$ with adaptive oracle access to $\mathcal{O} = \mathcal{O}_{\text{UPCS}} \cup \mathcal{O}_{\text{Sign}}$, and a negligible function negl such that $\text{Adv}_{\mathcal{D}}^{\text{zk}}(1^\lambda, \mathcal{F}) = |\Pr[\text{Real}_{\mathcal{D}}^{\text{zk}}(1^\lambda, \mathcal{F}) = 1] - \Pr[\text{Ideal}_{\mathcal{D}}^{\text{zk}}(1^\lambda, \mathcal{F}) = 1]| \leq \text{negl}(\lambda)$.

C. Construction of AIPA

This section presents a concrete instantiation of AIPA built on UPCS and a zk-SNARK-based proof system $\text{IM} = (\text{Setup}, \text{Prove}, \text{Verify})$ for private image manipulation. We first define hash functions and the proof state structure, then present the AIPA construction in Fig. 7.

Hash functions. Images are hashed by a ZKP-friendly hash H_{ZKP} (e.g., Griffin [44]), which incurs far fewer arithmetic circuit constraints than standard hashes. The image signature uses a standard hash H_0 (e.g., SHA-256).

Proof state structure. A proof state is a tuple $\pi = (\text{HC}, \Pi, \tilde{f}, \text{meta})$ whose components are defined as follows:

- **HC:** the hash chain, as defined in Eq. (2). It is a list $[h_0, \dots, h_n]$, where $h_i = H_{\text{ZKP}}(I_i)$. Adjacent steps are linked because h_i is shared between π_i and π_{i+1} , with $H_{\text{ZKP}}(\tilde{T}) = h_n$ anchoring the final image.

- Π : the set of zk-SNARK proofs $\{\pi_1, \dots, \pi_n\}$.
- $\tilde{f}$: the transformation sequence $[f_1, \dots, f_n]$.
- **meta:** the original image metadata.

The confidentiality of the intermediate images is guaranteed by the one-wayness of the hash function and the zero-knowledge property of the proofs.

D. Proof-system Modularity and Optimization

Proof-system modularity. We highlight a structural property of the AIPA construction: different proofs in Π can be produced by different zk-SNARK proving systems. As discussed in Section III-B, per-step proofs are independent and linked only by the shared hash h_i . And each proof only needs to assert its own relation $\mathcal{R}_f$. So the prover can use different zk-SNARK systems to prove this relation. The essential requirement is that hash values coincide exactly between different proving systems. The security analysis for multiple proving systems can be conducted analogously to the single-system case in Appendix E of the supplementary material, where each proof is reduced to its own system.

Proof parallelization and single-circuit. The scheme admits two optimizations. First, parallelization [15]: in the case of multiple edits, a prover with knowledge of all witnesses can generate all proofs in parallel to reduce proving time.

Second, single-circuit: for a fixed sequence $\tilde{f}$, one circuit enforces all transformations and outputs endpoint hashes h_0, h_n , replacing n proofs with one, reducing verification to a single check but increasing circuit size. Besides this approach requires the prover knows the witness for the full sequence.

Managing proof growth. The complete AIPA proof grows linearly with the number of editing steps, as each step carries HC, Π , and $\tilde{f}$. C2PA [7] exhibits the same behavior: each edit generates a new manifest, and the manifest store is embedded in and propagates with the image. To manage this growth, C2PA keeps small manifests embedded with the image but moves oversized manifests to external storage, retaining only a reference. AIPA can adopt a similar strategy: short provenance lineages can be embedded with the image, whereas long ones can be anchored to external storage, such as a blockchain [45], with only an index retained alongside the image.

E. Security Analysis

The formal security proofs of the AIPA scheme (image signature unforgeability, proof of knowledge, and zero-knowledge) are proven in Appendix E of the supplementary material.

VI. OUR INTEGRATED FRAMEWORK IN OSNs

In the OSN deployment scenario, as shown in Fig. 8, our integrated framework involves four types of entities: (1) The **Authority**, which serves dual roles as the Issuer and the CA of UPCS (Section IV), is responsible for system initialization and key distribution. (2) **Image Provenance Entities (IPEs)** are Authority-certified imaging devices or generative models that anchor the provenance lineage by producing the initial signature over original images. (3) **Users** act as holders, editors, and

Anonymous Image Provenance Authentication:

(1) **System Initialization** $\text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F}) \rightarrow (\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$. Run once by the Authority.

1) For each $f \in \mathcal{F}$, run $\text{IM.Setup}(1^\lambda, \mathcal{R}_f) \rightarrow (\text{pk}_f, \text{vk}_f)$, where $\mathcal{R}_f$ is defined in Eq. (3).

2) Choose the attribute upper bound n and generate the UPCS keys: $\text{UPCS.IKeygen}(1^\lambda, n) \rightarrow (\text{sk}_{\text{I}}, \text{pk}_{\text{I}})$, $\text{UPCS.CAKeygen}(1^\lambda) \rightarrow (\text{sk}_{\text{CA}}, \text{pk}_{\text{CA}})$.

Output $\text{pk}_{\text{IA}} = \{\text{pk}_f\}_{f \in \mathcal{F}}$, $\text{vk}_{\text{IA}} = (\{\text{vk}_f\}_{f \in \mathcal{F}}, \text{pk}_{\text{I}}, \text{pk}_{\text{CA}})$, and $\text{sk}_{\text{sys}} = (\text{sk}_{\text{I}}, \text{sk}_{\text{CA}})$.

(2) **Original Image Signing** $\text{Sig}_{\text{IA}}(\text{sk}_{\text{img}}, I_0, \text{meta}) \rightarrow (\pi_0, \sigma_{\text{img}})$. The image signer holds sk_{img} , the secret key corresponding to a pseudonym certificate cert on pk_{img} previously obtained via $\text{UPCS}(\text{CObtain}, \text{CIssue})$ (which in turn builds on a long-term credential previously obtained from the Authority via $\text{UPCS}(\text{Obtain}, \text{Issue})$). On input $(\text{sk}_{\text{img}}, I_0, \text{meta})$:

1) Compute $h_0 = \text{H}_{\text{ZKP}}(I_0)$ and $h_{\text{sig}} = \text{H}_0(h_0 \parallel \text{meta})$.

2) Compute the core signature $\sigma' \leftarrow \Psi.S_s(\text{sk}_{\text{img}}, h_{\text{sig}})$.

Output the full signature $\sigma_{\text{img}} = \sigma'$ and the initial proof state $\pi_0 = (\text{HC} = [h_0], \Pi = \emptyset, \vec{f} = [], \text{meta})$.

(3) **Edit Proving** $\text{Prove}_{\text{IA}}(\text{pk}_{\text{IA}}, I_{\text{in}}, \pi_{\text{in}}, f) \rightarrow (I_{\text{out}}, \pi_{\text{out}})$. Run by any editor. Let i be the current step index. Parse $\pi_{\text{in}} = (\text{HC}_{\text{in}}, \Pi_{\text{in}}, \vec{f}_{\text{in}}, \text{meta})$, where $\text{HC}_{\text{in}} = [h_0, \dots, h_{i-1}]$.

1) Retrieve $h_{i-1} = \text{H}_{\text{ZKP}}(I_{\text{in}})$ from HC_{in} , apply the transformation $I_{\text{out}} = f(I_{\text{in}})$ for $f \in \mathcal{F}$, and compute $h_i = \text{H}_{\text{ZKP}}(I_{\text{out}})$.

2) Generate a zk-SNARK proof for this step: $\pi_i \leftarrow \text{IM.Prove}(\text{pk}_f, (h_{i-1}, h_i, f), (I_{\text{in}}, I_{\text{out}}))$.

3) Extend the proof state: $\pi_{\text{out}} = (\text{HC}_{\text{in}} \cup [h_i], \Pi_{\text{in}} \cup \{\pi_i\}, \vec{f}_{\text{in}} \cup [f], \text{meta})$.

After n adjacent invocations with $\vec{f} = [f_1, \dots, f_n]$, the hash chain becomes $\text{HC} = [h_0, \dots, h_n]$, where $h_i = \text{H}_{\text{ZKP}}(I_i)$ and each adjacent step is linked by the shared hash value h_i .

(4) **Provenance Verification** $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \hat{I}, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \pi) \rightarrow \{0, 1\}$. Run by any public verifier. Parse $\pi = (\text{HC}, \Pi, \vec{f}, \text{meta})$

with $\text{HC} = [h_0, \dots, h_n]$, $\Pi = \{\pi_1, \dots, \pi_n\}$, $\vec{f} = [f_1, \dots, f_n]$, and $\sigma_{\text{img}} = \sigma'$, and check:

1) **Pseudonym certificate validity:** $\Psi.V_s(\text{pk}_{\text{CA}}, \text{pk}_{\text{img}} \parallel \text{aux}, \sigma_{\text{CA}}) = 1$, where σ_{CA} and aux are parsed from cert .

2) **Original image signature:** $\Psi.V_s(\text{pk}_{\text{img}}, \text{H}_0(h_0 \parallel \text{meta}), \sigma') = 1$.

3) **Proof chain and final image validity:** $\forall i \in [1, n] : \text{IM.Verify}(\text{vk}_{f_i}, (h_{i-1}, h_i, f_i), \pi_i) = 1$, and $\text{H}_{\text{ZKP}}(\hat{I}) = h_n$.

Return 1 iff all three checks pass.

Fig. 7. The construction of the AIPA scheme.

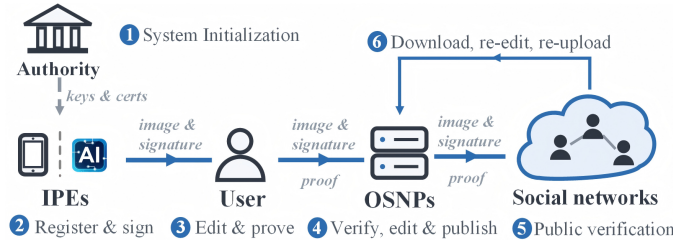


Fig. 8. Our integrated framework in OSNs.

verifiers of images; upon receiving the signed original image from the IPE, they may apply permissible edits with zero-knowledge proofs and upload the result to OSNPs. (4) **Online Social Network Platforms (OSNPs)** actively participate in the protocol by verifying provenance integrity, applying platform-specific processing (e.g., compression), and publishing images with complete, publicly verifiable provenance records.

Threat model. The Authority and OSNPs are *honest-but-curious*: they follow the protocol but may attempt to infer IPE identities or link pseudonym certificates. We assume each IPE (e.g., a trusted entity such as OpenAI [10], or a tamper-proof device such as the Sony Alpha 7 IV [8] or Google Pixel 10 [9]) securely stores its signing key and faithfully signs every original image at capture or generation, as in prior work [14], [15], [20]. In contrast, users are actively malicious.

A. AIPA in OSNs

We now describe the complete end-to-end pipeline, showing how each entity invokes the AIPA and UPCS components to achieve anonymous image provenance authentication in OSNs.

Phase 1: System Initialization (Authority). Run once by the Authority. $\text{AIPA.Gen}_{\text{IA}}(1^\lambda, \mathcal{F}) \rightarrow (\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$ generates the zk-SNARK proving and verification keys for all permissible transformations and sets up the UPCS identity and certificate layers. The Authority publishes pk_{IA} and vk_{IA} , and securely stores $\text{sk}_{\text{sys}} = (\text{sk}_{\text{I}}, \text{sk}_{\text{CA}})$.

Phase 2: Device Registration and Image Signing (IPE). Each IPE first performs a one-time registration: it generates a long-term key pair $(\text{sk}_{\text{U}}, \text{pk}_{\text{U}})$ via UPCS.UKeygen and obtains an anonymous credential cred from the Authority (as Issuer) via $\text{UPCS}(\text{Obtain}, \text{Issue})$, which certifies its device attributes without revealing any long-term identifier.

For each original image (I_0, meta) to be signed, the IPE generates a fresh one-time signing key $(\text{sk}_{\text{img}}, \text{pk}_{\text{img}})$ via UPCS.PseuKeyGen and requests a pseudonym certificate cert for pk_{img} from the Authority (as CA) by anonymously presenting cred via $\text{UPCS}(\text{CObtain}, \text{CIssue})$. It then invokes $\text{AIPA.Sig}_{\text{IA}}(\text{sk}_{\text{img}}, I_0, \text{meta})$ to produce the initial proof state π_0 and image signature σ_{img} . The signed image and provenance data $(I_0, \pi_0, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}})$ are transferred to the User.

Importantly, each certificate request presents an anonymous credential, keeping the IPE anonymous. Certificate unlinkability ensures that certificates on different images cannot be linked to each other. Moreover, UPCS supports preprocessing and batch issuance, allowing the IPE to obtain and store a pool of certificates in advance for future signing.

Phase 3: Image Editing (User). Upon receiving the signed image and provenance, the User may apply a sequence of permissible transformations $\vec{f} = [f_1, \dots, f_n]$ (e.g., cropping, filtering, or grayscale) according to their preferences. For

each f_i , the User invokes $\text{AIPA.Prove}_{\text{IA}}(\text{pk}_{\text{IA}}, I_{i-1}, \pi_{i-1}, f_i) \rightarrow (I_i, \pi_i)$, which applies the transformation and extends the proof state into π_i . After n steps, the User uploads the provenance tuple $(\hat{I} = I_n, \pi_n, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}})$ to the OSNP.

Phase 4: OSNP Publishing (OSNP). Upon receiving the provenance tuple, the OSNP first verifies the received provenance via $\text{AIPA.Verify}_{\text{IA}}$, rejecting any forged or tampered provenance tuple. Once verified, the OSNP applies platform-specific processing transformations $[g_1, \dots, g_m] \in \mathcal{F}$, such as JPEG compression at the platform's quality setting or downscaling to the platform's resolution limit, invoking $\text{AIPA.Prove}_{\text{IA}}$ for each and extend π_n to π' . Finally, it publishes the final image and publicly verifiable provenance $(\hat{I}' = I_{n+m}, \pi', \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}})$.

Phase 5: Public Verification. Any entity can independently audit the provenance authenticity of a published image by running $\text{AIPA.Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \hat{I}', \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \pi')$, without trusting the OSNP or editors. The unchanged metadata meta indicates the image's provenance, e.g., a specific camera or generative AI model. Thus, AIPA guarantees end-to-end provenance authenticity while preserving IPE anonymity.

Phase 6: Image Re-disseminated. Any entity on the OSNs may download a published image, apply further permissible transformations via $\text{AIPA.Prove}_{\text{IA}}$, and upload it to other OSNPs. The provenance lineage is extended transparently, and any subsequent verifier can audit the full history by running $\text{AIPA.Verify}_{\text{IA}}$, without trusting intermediate editors.

B. VIMz-Loua and JPEG Image Compression Proof

Unlike resource-constrained devices in VIMz [15], the prover on the OSN platform side possesses abundant memory and GPU resources and needs to perform nonlinear transformations such as JPEG compression that involve costly field operations including division and rounding. To this end, we integrate the proof backend Loua, proposed in Eva [21], which supports lookup tables and GPU acceleration, with VIMz [15], yielding VIMz-Loua adapted to the OSN platform environment. In particular, VIMz-Loua produces a Groth16 proof [38] with small proof size and low verification overhead.

The process of JPEG image compression is illustrated in Fig. 9. As observed by Eva [21] in video compression, entropy coding in JPEG compression is also lossless: the decoder can precisely recover the quantized coefficients Z through entropy decoding. Thus, the prover can encode both the RGB-to-quantized-coefficients and quantized-coefficients-to-RGB transformations as circuits. Because the quantized coefficients remain identical across entropy encoding and decoding, the two adjacent circuits share the same hash value, thereby preserving the continuity of the JPEG compression proof. If JPEG is the last transformation, only the forward RGB-to-quantized-coefficients circuit is needed. The verifier can decode the received JPEG file to retrieve the quantized coefficients for proof verification.

In VIMz-Loua, we design the JPEG image compression proving circuit based on the above settings. As the block-DCT-quantization is the core pipeline, involving the non-linear operation, we show how to accelerate proof generation

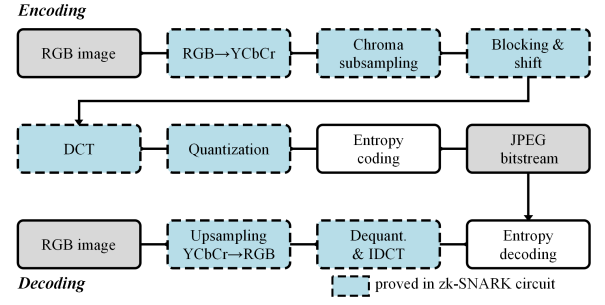


Fig. 9. The JPEG compression pipeline.

Algorithm 1 The proposed JPEG compression circuit.

```

1: Public:  $(h_{i-1}, h_i, Q)$ , where  $Q$  is the quantization table.
2: Witness:  $(I_{i-1}, Z)$ , where  $Z$  is the quantized coefficient stream recovered from the JPEG file by the verifier via entropy decoding.
3: Enforce  $h_{i-1} = \text{H}_{\text{ZKP}}(I_{i-1})$ .
4: Partition  $I_{i-1}$  into level-shifted  $8 \times 8$  blocks of YCbCr samples.
5: for each  $8 \times 8$  block  $B$  do
6:    $F \leftarrow$  two separable 1-D DCT passes over  $B$ ; // scaled by  $2^{13}$ 
7:    $F' \leftarrow \text{Shift}(F + 2^{n-1}, n)$  element-wise; // rounding shift
8:   for each coefficient  $v$  of  $F'$  do
9:      $s \leftarrow$  advice; Enforce  $s(1-s) = 0$ ; // sign
10:     $p \leftarrow ((s ? -v : v) + c) \cdot \rho$ ; //  $\rho, c$ : reciprocal/correction for division by  $Q$ 
11:     $q \leftarrow \text{Shift}(p, 15)$ ;
12:    Enforce  $(s ? -q : q) =$  advised coefficient in  $Z$ .
13:   end for
14: end for
15: Enforce  $h_i = \text{H}_{\text{ZKP}}(Z)$ .
16:
17: function  $\text{Shift}(y, n)$  // range check by lookup table
18: advice  $q$ ; Enforce  $0 \leq y - q \cdot 2^n < 2^n$ ;
19: return  $q$ . // returns  $q = \lfloor y/2^n \rfloor$ 

```

using the lookup table, as shown in Algorithm 1. The circuit takes (h_{i-1}, h_i, Q) as public inputs and (I_{i-1}, Z) as private witnesses, where the quantization table Q is embedded as circuit constants.

VII. IMPLEMENTATION AND EVALUATION

We present the performance of UPCS and AIPA separately.¹

A. The UPCS Implementation and Evaluation

We implement the UPCS construction in Rust. In the UPCS implementation, the anonymous credential identity layer, is instantiated over the BLS12-381 type-3 pairing curve using the Arkworks library. The certificate layer deploys X.509 certificates generated via Rcggen, with signatures instantiated using Ed25519. All algorithms are benchmarked on a Legion Y9000P machine equipped with an Intel i7-12700H processor (14 cores, 2.70 GHz) and 32.00 GB of RAM.

We compare UPCS with three privacy-preserving signature schemes: Spartan-ECDSA [16] (used in VIMz [15]) which proves in zero knowledge that an image signature verifies under a public key committed as a leaf in a Merkle tree, PoP [46] which proves that a message is signed by a ring member without revealing which one, and the group signature scheme [18] (recommended by VerITAS [17]) which allows

¹The source code is available at <https://github.com/Ivyforests/AIPA>.

a user to sign on behalf of a group. For fair comparison in anonymous message authentication, we align UPCS signing and verification with signature generation and verification in these schemes: UPCS certificate requests can be preprocessed and stored in advance, excluding their cost from the runtime of message authentication, whereas the comparison schemes lack such preprocessing and must generate their proofs or signatures in real time for every authentication.

TABLE II
PERFORMANCE COMPARISON OF UPCS AND ANONYMOUS SIGNING SCHEMES.^a

Scheme	Sign (ms)	Verify (ms)	Size (bytes)
Spartan-ECDSA [16] ($u = 2^{20}$)	2686.64	559.20	15903
PoP [46] ($u = 2^{14}$)	999.56	124.05	5332
Group signature [18]	3.37	3.66	336
UPCS (online signing)	0.017	0.066	362
UPCS (request, $n = 10$, $d = 1$)	9.356	–	1262
key generation	0.036	–	–
request prep.	3.47	–	964
CA issue	5.85	–	298

^aFor Spartan-ECDSA [16] and PoP [46], u denotes the anonymity set size. UPCS and the group signature [18] achieve constant overhead independent of u . The UPCS preprocessable certificate request overhead depends on the number of identity attributes n and the disclosed attributes d , but is independent of the number of users.

As shown in Table II, UPCS signing takes only 0.017 ms, outperforming Spartan-ECDSA [16] ($u = 2^{20}$) by roughly 158037 $\times$, PoP [46] ($u = 2^{14}$) by 58797 $\times$, and the group signature [18] by roughly 198 $\times$. Verification takes 0.066 ms, outperforming Spartan-ECDSA [16] by 8472 $\times$, PoP [46] by 1879 $\times$, and the group signature [18] by 55 $\times$. Moreover, the transmitted signature and X.509 certificate size is only 362 bytes, reducing communication by roughly 98% over Spartan-ECDSA [16] and 93% over PoP [46], while remaining comparable to the group signature [18] (336 bytes).

This improvement stems from a decoupled design: UPCS isolates heavy anonymous credential operations in the pre-processable certificate request phase, reducing online message signing and verification to lightweight Ed25519 signing and verification. As a result, signer anonymity is preserved without introducing expensive cryptographic overhead in the signing phase. In contrast, comparison schemes [16], [18], [46] must execute both expensive anonymous proofs and signature operations during message authentication, which constitutes their primary performance bottleneck. Consequently, UPCS is particularly well-suited to large-scale OSN deployments, offering strong anonymity with low latency and communication cost.

The certificate request phase (Table II) is lightweight: a single request incurs 9.356 ms of computation and 1262 bytes of communication. In contrast, Spartan-ECDSA [16] and PoP [46] require higher computational and communication overhead to achieve anonymity. The group signature scheme requires only 7.03 ms and 336 bytes per sign-and-verify operation, which is lower than the cost of a UPCS certificate request. Yet, UPCS is built on AC systems, enabling attribute-based authentication and selective disclosure of attributes. Besides, UPCS allows users to generate their own private keys, thus preventing even a malicious issuer from learning

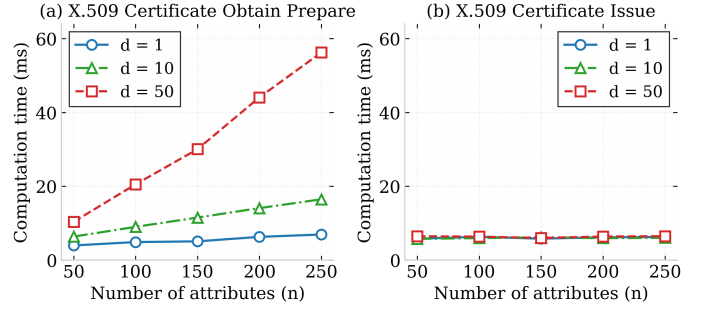


Fig. 10. Impact of the number of identity attributes n and disclosed attributes d on the X.509 certificate request phase of UPCS ($n \in \{50, 100, 150, 200, 250\}$, $d \in \{1, 10, 50\}$).

the user's identity. In contrast, group signatures rely on a group manager to distribute keys, which may introduce privacy concerns regarding traceability. Crucially, UPCS certificate requests can be preprocessed, while comparison schemes must repeat this expensive cryptographic overhead at signing time. As shown in Fig. 10 and [27], the obtain-prepare overhead grows linearly with n and increases with d , whereas the issue overhead is insensitive to both n and d . Online signing and verification are independent of both n and d .

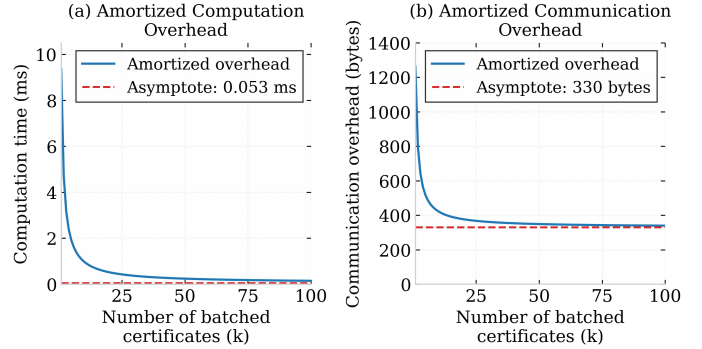


Fig. 11. Amortized per-certificate (a) computation and (b) communication overhead versus the number of batched certificates k . The dashed lines denote the asymptotic lower bounds (0.053 ms and 330 bytes, respectively).

UPCS supports batch certificate requests: a user obtains k certificates in one interaction with the CA, amortizing the credential presentation cost. For a batch of k certificates, the request is generated and verified only once (9.303 ms, 932 bytes), while per-certificate key generation, certificate signing, and the transmission of the public key and certificate recur per certificate (0.053 ms, 330 bytes). The amortized overhead is thus $0.053 + 9.303/k$ ms for computation and $330 + 932/k$ bytes for communication. Fig. 11 shows the per-certificate computation and communication approaching their asymptotic lower bounds as k grows. Same-batch certificates are linkable to the same session, but the user's identity remains anonymous, and cross-batch unlinkability is preserved. The batch size k thus is an adjustable privacy-efficiency trade-off.

B. The AIPA Implementation and Evaluation in a Simulated OSN Environment

We implemented the full AIPA pipeline (phases 1–5) in a simulated OSN environment, where the user side runs

TABLE III

PER-OPERATION OVERHEAD OF THE COMPLETE AIPA PIPELINE IN THE SIMULATED OSN ENVIRONMENT.

Phase (Entity)	Operation	Prover runtime (s)	Peak memory (GB)	Proof size (B)	Verifier runtime (ms)
1 (Authority)	Initialization	198.5 [†]	—	—	—
2 (IPE)	Issuance + sign	0.015 [*]	—	362 [‡]	—
3 (User)	Crop	1302.6	1.37	11401	859
VIMz [15]	Brightness	1544.1	3.28	12343	2341
	Contrast	1590.3	3.20	12341	2643
4 (OSNP)	Verification	—	—	36728 [‡]	3938
VIMz-Loua	Resize	283.2	25.0	968 [§]	10
	Sharpen	373.7	27.9	1096 [§]	7
	RGB-YCbCr	221.2	42.3	1032 [§]	5
	YCbCr-to-DCT-quantization	183.5	30.2	968 [§]	7
5 (Public)	Verification	—	—	41116 [‡]	4280

[†]Initialization time of the system; ^{*}credential issuance and signing time; [‡]image provenance size; [§]proof sizes include the 448 B VIMz-Loua proof plus auxiliary information.

VIMz [15] on a Legion Y9000P laptop (Intel i7-12700H, 32 GB RAM), while the OSN platform side runs GPU-accelerated VIMz-Loua on a server (12 vCPUs Intel Xeon Silver 4214R @ 2.40 GHz, 90 GB RAM, NVIDIA RTX 3080 Ti). Both sides use the BN254-Grumpkin cycle of curves. Thus, the continuity of the AIPA proofs from the two backends can be guaranteed.

We observed that WeChat Moments resizes and compresses uploaded images. For example, a 1.14 MB image of 2560×1920 pixels is processed into a 405 KB image of 1440×1080 pixels. However, WeChat Moments has not disclosed its specific procedures. We follow the processing pipeline described in [24].

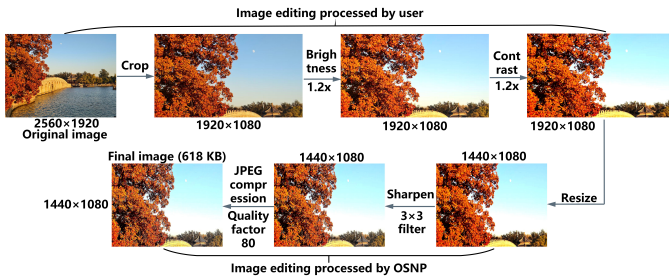


Fig. 12. The image editing workflow.

The editing chain of the user and OSNP is shown in Fig. 12. User-side edits comprise cropping (2560×1920 → 1920×1080) and 1.2× brightness and contrast adjustments. The platform side follows the work flow in [24]: resize to 1440×1080 (the display resolution of WeChat Moments), 3×3 sharpening filter, and JPEG compression (quality factor 80). Here, compression is the final edit. We design the RGB-to-quantization circuits for the JPEG compression, with two sub-circuits: RGB-to-YCbCr circuit and YCbCr-to-DCT-quantization circuit, since YCbCr requires 8×8 block processing. Our compression circuit replicates the workflow of the jpeg – encoder library in Rust.

TABLE IV

PERFORMANCE COMPARISON OF PROOF SYSTEMS FOR THE CROP (50% OF IMAGE) EDIT ON AN IMAGE OF SIZE 2^{22} PIXELS.[†]

Scheme	Prover runtime (s)	Verifier runtime (ms)	Peak memory (GB)	Proof size (B)
HyperVerITAS [22]*	342.9	46483	37.7	53856
VerITAS [17]*	421.9	20346	48.6	941411
VIMz [15]	997.6	205	1.0	10432
Trust nobody [20]	373.9	54280	2.5	16284
VIMz-Loua	331.5	12	21.06	448

[†]Evaluated using the benchmark of [22] on the same server as in Table III. Results for other image sizes are reported in Appendix F of the supplementary material.

*HyperVerITAS is instantiated with PST polynomial commitment; VerITAS is instantiated with FRI polynomial commitment.

Table III summarizes the overhead of the full pipeline. The Authority’s one-time initialization takes 198.5 s and is precomputable. On the IPE side, credential issuance and image signing take only 15 ms, due to the efficiency of UPCS.

In an OSN, generating an image editing proof is a one-time operation, whereas verifying the provenance of an image may occur up to millions of times. Thus, the performance of the verification side is very important. According to Table III, the final AIPA proof is only 40.2 KB, which is 6.5% of the 618 KB final image in our experiment and 9.9% of the 405 KB size of a comparable image in WeChat Moments, indicating that the proof size is within a reasonable range. The verification of the final AIPA proof takes only 4.28 s. This performance is achieved by using VIMz [15] as the user proof system, which features low memory usage, small proof size, and efficient verification. On the platform side, VIMz-Loua is adopted, leveraging GPU acceleration, constant proof size, and efficient verification. In particular, VIMz-Loua generates a total proof of only 3.97 KB with a verification time of 29 ms, making the verification side even more efficient than that of VIMz [15].

The user side takes an average of 1479 seconds to generate an edit proof. However, we note that when immediate publication is not required, proof generation can proceed asynchronously in the background before the image is uploaded. Moreover, provenance proofs need not be mandatory for all users. The platform may instead require them selectively from high-risk accounts, concentrating the proving cost where the risk of deepfake misuse is highest. Finally, as discussed in Section V-C, multiple edit proofs can be generated in parallel when the prover holds all witnesses, further reducing the proving time for multi-step edits.

According to Table IV, VIMz-Loua achieves the smallest verification overhead and proof size among the five systems, with verification taking only 12 milliseconds and proofs of just 448 bytes. Moreover, as shown in Fig. 13, the proof size of VIMz-Loua remains a constant 448 B ($O(1)$), and its verification time stays within 6–12 ms, as the input size grows from 2^{19} to 2^{23} pixels. Therefore, VIMz-Loua is well suited to verification-intensive scenarios such as OSNs.

VIII. CONCLUSION

We presented AIPA, an anonymous image provenance authentication scheme that establishes the provenance and editing

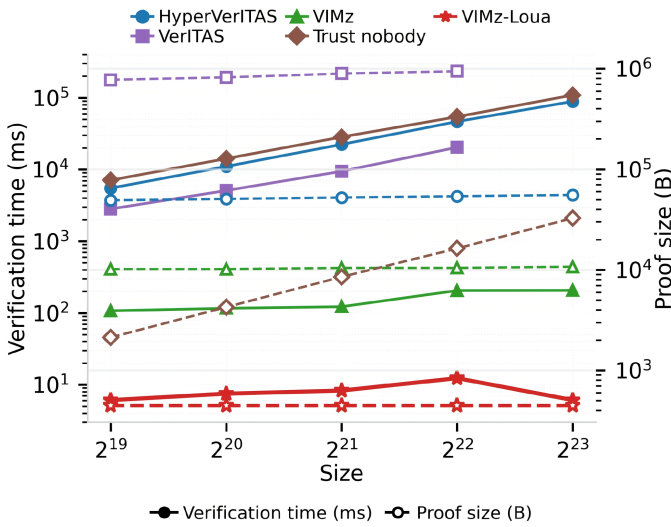


Fig. 13. Verification time and proof size of five proof systems across input sizes from 2^{19} to 2^{23} (log scale).

history of an image in OSNs, answering where an image came from and what edits it has undergone. To preserve the signer's anonymity, we built UPCS on anonymous credentials, which provides one-time unlinkable pseudonym certificates compatible with X.509, and applied it within AIPA. For the OSN platform side, we designed VIMz-Loua, a GPU-accelerated proof system for efficient verification and JPEG compression proofs, and implemented the complete end-to-end workflow in a simulated OSN environment. Our experiments show that UPCS signing takes only 0.017 ms and that the complete provenance lineage of an image in AIPA is only 40.2 KB, verifiable in 4.28 s.

Future work includes exploring certificate revocation in UPCS to trace and hold accountable users who upload illegal content, and employing proof aggregation [47] or multi-predicate PCD [48] technologies to achieve succinct proofs, making the proof size independent of the number of editing steps.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (Grant No. 62572025), Beijing Natural Science Foundation (Grant No. 4242024), Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing (Grant No. GJJ-25-xxx), and the Fundamental Research Funds for the Central Universities, China. The authors used DeepSeek and Kimi to assist with coding, and to proofread and polish the paper text, correcting spelling errors, grammatical errors, and awkward expressions.

REFERENCES

- [1] F. A. Croitoru, A. Hiji, V. Hondru, N.-C. Ristea, P. Irofti, M. Popescu, C. Rusu, R. Ionescu, F. Khan, and M. Shah, "Deepfake media generation and detection in the generative ai era: A survey and outlook," *ACM Computing Surveys*, vol. 58, no. 15, pp. 1–35, 2026.
- [2] S. Alanazi, S. Asif, A. Caird-Daley, and I. Moulitsas, "Unmasking deepfakes: a multidisciplinary examination of social impacts and regulatory responses," *Human-Intelligent Systems Integration*, vol. 7, no. 1, pp. 131–153, 2025.

- [3] H. Wu, J. Zhou, J. Tian, J. Liu, and Y. Qiao, "Robust image forgery detection against transmission over online social networks," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 443–456, 2022.
- [4] T. Wang, X. Liao, K. P. Chow, X. Lin, and Y. Wang, "Deepfake detection: A comprehensive survey from the reliability perspective," *ACM Computing Surveys*, vol. 57, no. 3, pp. 1–35, 2024.
- [5] G. Pei, J. Zhang, M. Hu, Z. Zhang, C. Wang, Y. Wu, G. Zhai, J. Yang, and D. Tao, "Deepfake generation and detection: A benchmark and survey," *ACM Computing Surveys*, vol. 58, no. 11, pp. 1–41, 2026.
- [6] H.-H. Nguyen-Le, V.-T. Tran, T. Nguyen, and N.-A. Le-Khac, "A survey on proactive deepfake defense: Disruption and watermarking," *ACM Computing Surveys*, vol. 58, no. 5, pp. 1–37, 2025.
- [7] C2PA, "Coalition for content provenance and authenticity (c2pa)," 2026, accessed: 2026-06-08. [Online]. Available: <https://c2pa.org>
- [8] Sony, "Sony unlocks in-camera forgery-detection technology," 2022, accessed: 2026-06-08. [Online]. Available: <https://www.sony.eu/presscentre/sony-unlocks-in-camera-forgery-proof-technology>
- [9] Google, "How pixel and android are bringing a new level of trust to your images with c2pa content credentials," 2025, accessed: 2026-06-08. [Online]. Available: <https://blog.google/security/pixel-android-trusted-images-c2pa-content-credentials>
- [10] OPEN AI, "Advancing content provenance for a safer, more transparent ai ecosystem," 2026, accessed: 2026-06-08. [Online]. Available: <https://openai.com/index/advancing-content-provenance/>
- [11] C. Mainka, V. Mladenov, and S. Rohmann, "Shadow attacks: Hiding and replacing content in signed pdfs." in *NDSS*, 2021.
- [12] S. Chen, Z. Lin, and Y. Zhang, "Controlled data races in enclaves: Attacks and detection," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4069–4086.
- [13] K. Murdock, D. Oswald, F. D. Garcia, J. Van Bulck, D. Gruss, and F. Piessens, "Plundervolt: Software-based fault injection attacks against intel sgx," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1466–1482.
- [14] A. Naveh and E. Tromer, "Photoproof: Cryptographic image authentication for any set of permissible transformations," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 255–271.
- [15] S. Dziembowski, S. Ebrahimi, and P. Hassanizadeh, "Vimz: Private proofs of image manipulation using folding-based zkSNARKs," in *Proceedings on Privacy Enhancing Technologies*, 2025.
- [16] Personaelabs, "Spartan-ecdsa: The fastest open-source method to verify ecdsa (secp256k1) signatures in zero-knowledge," 2023, accessed: 2026-07-15. [Online]. Available: <https://github.com/personaelabs/spartan-ecdsa>
- [17] T. Datta, B. Chen, and D. Boneh, "Veritas: Verifying image transformations at scale," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 4606–4623.
- [18] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Annual international cryptology conference*. Springer, 2004, pp. 41–55.
- [19] A. Chiesa and E. Tromer, "Proof-carrying data and hearsay arguments from signature cards," in *ICS*, vol. 10, 2010, pp. 310–331.
- [20] P. Della Monica, I. Visconti, A. Vitaletti, and M. Zecchini, "Trust nobody: Privacy-preserving proofs for edited photos with your laptop," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 4624–4642.
- [21] C. Zhang, X. Yang, D. Oswald, M. Ryan, and P. Jovanovic, "Eva: Efficient privacy-preserving proof of authenticity for lossily encoded videos," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 4643–4662.
- [22] G. Greiner, T. Mowery, and P. Soni, "Hyperveritas: Verifying image transformations at scale on boolean hypercubes," in *Proceedings on Privacy Enhancing Technologies*, 2026.
- [23] 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.
- [24] W. Sun, J. Zhou, Y. Li, M. Cheung, and J. She, "Robust high-capacity watermarking over online social network shared images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 3, pp. 1208–1221, 2020.
- [25] Z. Zhu, P. Wei, Z. Qian, S. Li, and X. Zhang, "Image sanitization in online social networks: A general framework for breaking robust information hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 6, pp. 3017–3029, 2022.
- [26] Z. Qu, Z. Xi, W. Lu, X. Luo, Q. Wang, and B. Li, "Df-rap: A robust adversarial perturbation for defending against deepfakes in real-world social network scenarios," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 3943–3957, 2024.

- [27] O. Sanders, “Efficient redactable signature and application to anonymous credentials,” in *IACR international conference on public-key cryptography*. Springer, 2020, pp. 628–656.
- [28] G. Fuchsbauer, C. Hanser, and D. Slamanig, “Structure-preserving signatures on equivalence classes and constant-size anonymous credentials,” *Journal of Cryptology*, vol. 32, no. 2, pp. 498–546, 2019.
- [29] B. Brecht, D. Theriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, “A security credential management system for v2x communications,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 12, pp. 3850–3871, 2018.
- [30] M. Khodaei, H. Jin, and P. Papadimitratos, “Secmace: Scalable and robust identity and credential management infrastructure in vehicular communication systems,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1430–1444, 2018.
- [31] S. Khan, M. Khan, M. A. Khan, F. Luo, M. Usman, A. Irshad, K. Wu, L. Wang, and M. A. Khan, “Pbatch: Pseudonym certificate batch authentication with generative ai-based cache for cooperative intelligent transportation systems,” *IEEE Transactions on Intelligent Transportation Systems*, 2025.
- [32] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin, “Issuer-hiding attribute-based credentials,” in *International Conference on Cryptology and Network Security*. Springer, 2021, pp. 158–178.
- [33] O. Mir, B. Bauer, S. Griffy, A. Lysyanskaya, and D. Slamanig, “Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials,” in *Proceedings of the 2023 ACM SIGSAC conference on computer and communications security*, 2023, pp. 30–44.
- [34] S. Argo, T. Güneysu, C. Jeudy, G. Land, A. Roux-Langlois, and O. Sanders, “Practical post-quantum signatures for privacy,” in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1523–1537.
- [35] A. Koyama, K. Toyoda, M. Fujimoto, and T. H. Tran, “zk-real: A zero-knowledge-based protocol for repeated image edit authenticity proof with lattice hashing,” *IEEE Open Journal of the Computer Society*, 2026.
- [36] Zcash, “The halo2 zero-knowledge proving system,” 2022, accessed: 2026-06-08. [Online]. Available: <https://github.com/zcash/halo2>
- [37] A. Kothapalli, S. Setty, and I. Tzialla, “Nova: Recursive zero-knowledge arguments from folding schemes,” in *Annual International Cryptology Conference*. Springer, 2022, pp. 359–388.
- [38] J. Groth, “On the size of pairing-based non-interactive arguments,” in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2016, pp. 305–326.
- [39] F. Bai, X. Yang, T. Shen, B. Gong, M. Waqas, and H. Alfuhaid, “Two-level iva: Scalable zero-knowledge proof for trustworthy video stream verification,” *IEEE Transactions on Circuits and Systems for Video Technology*, 2026.
- [40] A. Frolov, J. Guo, X. Zhao, T. Datta, D. Boneh, and I. Miers, “zk-cinema: Proving video provenance in zero knowledge,” *Cryptology ePrint Archive*, 2026.
- [41] Y. Zhang, M. Zhou, and B. Bünz, “Verifiable compression of images,” *Cryptology ePrint Archive*, 2026.
- [42] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Conference on the theory and application of cryptographic techniques*. Springer, 1986, pp. 186–194.
- [43] D. Fiore and A. Nitulescu, “On the (in) security of snarks in the presence of oracles,” in *Theory of Cryptography Conference*. Springer, 2016, pp. 108–138.
- [44] L. Grassi, Y. Hao, C. Rechberger, M. Schafneggler, R. Walch, and Q. Wang, “Horst meets fluid-spn: Griffin for zero-knowledge applications,” in *Annual international cryptology conference*. Springer, 2023, pp. 573–606.
- [45] P. Mikołajczyk, P. Hassanizadeh, and S. Ebrahimi, “Towards trustless provenance: A privacy-preserving framework for on-chain media verification,” *Cryptology ePrint Archive*, 2025.
- [46] A. P. Woo, A. Ozdemir, C. Sharp, T. Pornin, and P. Grubbs, “Efficient proofs of possession for legacy signatures,” in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 3291–3308.
- [47] N. Gailly, M. Maller, and A. Nitulescu, “Snarkpack: Practical snark aggregation,” in *International Conference on Financial Cryptography and Data Security*. Springer, 2022, pp. 203–229.
- [48] A. Chiesa, E. Tromer, and M. Virza, “Cluster computing in zero knowledge,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 371–403.

APPENDIX A

FORMAL DEFINITION OF THE DIGITAL SIGNATURE SCHEME AND THE HASH FUNCTION

Definition 1 (Digital Signature Scheme). A digital signature scheme is a triple of PPT algorithms $\Psi = (G_s, S_s, V_s)$:

- 1) $(pk, sk) \leftarrow G_s(1^\lambda)$: on input the security parameter λ , it outputs a key pair (pk, sk) .
- 2) $\sigma \leftarrow S_s(sk, m)$: on input the secret key sk and a message $m \in \mathcal{M}$, it outputs a signature σ .
- 3) $b \leftarrow V_s(pk, m, \sigma)$: it outputs $b = 1$ iff σ is a valid signature of m under pk .

Correctness. For every (pk, sk) generated by $G_s(1^\lambda)$ and every $m \in \mathcal{M}$, it holds with probability 1 that $V_s(pk, m, S_s(sk, m)) = 1$.

Existential Unforgeability under Chosen-Message Attacks (EUF-CMA). Ψ is EUF-CMA secure if no PPT adversary $\mathcal{A}$ wins the following experiment with non-negligible probability: given pk and access to a signing oracle $O_{\text{Sign}}^{\text{EUF}}$ that returns $S_s(sk, m)$ for any queried message m , $\mathcal{A}$ outputs a forgery (m^*, σ^*) , and succeeds iff $V_s(pk, m^*, \sigma^*) = 1$ and m^* was never submitted to the oracle. Formally, Ψ is EUF-CMA secure if, for all PPT adversaries $\mathcal{A}$, the advantage $\text{Adv}_{\Psi}^{\text{EUF-CMA}}(1^\lambda) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}(1^\lambda) = 1]$ is negligible.

Throughout this paper, Ψ serves as the building block of the certificate layer of UPCS and of image signing in AIPA, and it can be instantiated by any other EUF-CMA-secure signature scheme.

Definition 2 (Collision-Resistant Hash Function). A hash function $H : \mathcal{M} \rightarrow \mathcal{T}$ is an efficiently computable function mapping a (large) message space $\mathcal{M}$ to a (small) digest space $\mathcal{T}$, and we say that H is defined over $(\mathcal{M}, \mathcal{T})$. H is *collision resistant* if, for every adversary $\mathcal{A}$ that takes no input and outputs a pair $(m_0, m_1) \in \mathcal{M}^2$, the advantage

$$\text{CRadv}[\mathcal{A}, H] = \Pr[m_0 \neq m_1 \wedge H(m_0) = H(m_1)]$$

is negligible.

Definition 3 (One-Wayness). A hash function H defined over $(\mathcal{M}, \mathcal{T})$ is *one-way* if, for every adversary $\mathcal{A}$, the advantage

$$\text{OWadv}[\mathcal{A}, H] = \Pr[m' \leftarrow \mathcal{A}(H(m)) : H(m') = H(m)],$$

where $m \in \mathcal{M}$ is sampled uniformly at random and the probability is taken over m and the randomness of $\mathcal{A}$, is negligible.

In this paper, the hash functions H_0 and H_{ZKP} are assumed to be collision resistant and one-way.

APPENDIX B

FORMAL DEFINITION OF THE ZK-SNARK

Definition 4 (zk-SNARK). We adopt the definition of zk-SNARKs from Trust Nobody [20]. Let R be a polynomial relation, i.e., an efficiently checkable relation such that $R(x, w) = 1$ implies $|w| \leq \text{poly}(|x|)$. A zk-SNARK for R is a triple of PPT algorithms $\Sigma = (\text{Setup}, \text{Prove}, \text{Verify})$:

- 1) $(pk, vk) \leftarrow \text{Setup}(1^\lambda, R)$: on input the security parameter λ and the relation R , it outputs a proving key pk and a verification key vk .
- 2) $\pi \leftarrow \text{Prove}(pk, x, w)$: on input the proving key pk , an instance x , and a witness w with $R(x, w) = 1$, it outputs a proof π .
- 3) $b \leftarrow \text{Verify}(vk, x, \pi)$: on input the verification key vk , an instance x , and a proof π , it outputs a bit b , where $b = 1$ indicates acceptance.

Completeness. For all $\lambda \in \mathbb{N}$ and all (x, w) with $R(x, w) = 1$,

$$\Pr \left[\begin{array}{l} \text{Verify}(vk, x, \pi) = 1 : \\ (pk, vk) \leftarrow \text{Setup}(1^\lambda, R), \\ \pi \leftarrow \text{Prove}(pk, x, w) \end{array} \right] = 1.$$

Knowledge Soundness. Σ has knowledge soundness for the auxiliary input distribution Z if, for every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}$ such that

$$\Pr \left[\begin{array}{l} \text{Verify}(vk, x, \pi) = 1 \wedge R(x, w) = 0 : \\ (pk, vk) \leftarrow \text{Setup}(1^\lambda, R), \\ \text{aux}_Z \leftarrow Z(pk, vk), \\ (x, \pi) \leftarrow \mathcal{A}(pk, vk, \text{aux}_Z), \\ w \leftarrow \mathcal{E}(pk, vk, \text{aux}_Z) \end{array} \right] \leq \text{negl}(\lambda).$$

Zero Knowledge. Σ satisfies (composable) zero-knowledge if there exists a PPT simulator $(S_{\text{key}}, S_{\text{prove}})$ such that for all PPT adversaries $\mathcal{A}$:

Keys Indistinguishability:

$$\begin{aligned} \Pr[\mathcal{A}(pk, vk) = 1 : (pk, vk) \leftarrow \text{Setup}(1^\lambda, R)] \\ \approx \Pr[\mathcal{A}(pk, vk) = 1 : (pk, vk, td) \leftarrow S_{\text{key}}(1^\lambda, R)]. \end{aligned}$$

Proof Indistinguishability: for all (x, w) with $R(x, w) = 1$, where $(pk, vk, td) \leftarrow S_{\text{key}}(1^\lambda, R)$,

$$\begin{aligned} \Pr[\mathcal{A}(pk, vk, \pi) = 1 : \pi \leftarrow \text{Prove}(pk, x, w)] \\ \approx \Pr[\mathcal{A}(pk, vk, \pi) = 1 : \pi \leftarrow S_{\text{prove}}(pk, vk, td, x)]. \end{aligned}$$

Non-interactivity and Succinctness. The proof is non-interactive and publicly verifiable. The verifier runs in time $\text{poly}(\lambda + |x| + \log(|w|))$ and the proof size is $\text{poly}(\lambda + \log(|w|))$.

Instantiation. Throughout this paper, the proof system $\text{IM} = (\text{Setup}, \text{Prove}, \text{Verify})$ (Section V-C of the main paper) is an instantiation of the above zk-SNARK for the per-step transformation relation $\mathcal{R}_f$: its setup generates a key pair $(pk_f, vk_f) \leftarrow \text{IM.Setup}(1^\lambda, \mathcal{R}_f)$ for each transformation $f \in \mathcal{F}$.

APPENDIX C

SECURITY MODEL OF ANONYMOUS CREDENTIAL SCHEME

Following the anonymous credential scheme[27], we define the identity-layer oracles adopted in the UPCS security model. These oracles populate the global lists HU , CU , and L_{cred} defined in Section IV of the main paper.

- $O_{\text{HU}}(i)$: if $i \in \text{HU} \cup \text{CU}$, returns $\perp$; else runs $(sk_U, pk_U) \leftarrow \text{UKeygen}(pk_I)$, adds i to HU , returns pk_U .
- $O_{\text{CU}}(i, pk_U)$: if $i \in \text{CU}$, returns $\perp$; else if $i \notin \text{HU}$, registers a corrupt user with pk_U , adds i to CU ; else if $i \in \text{HU}$, returns sk_U and all $(\text{cred}, \text{cert})$ of i , then moves i to CU .

Unforgeability $\text{Exp}_{\mathcal{A}}^{\text{uf}}(1^\lambda, n):$

1. $(\text{sk}_I, \text{pk}_I) \leftarrow \text{IKeygen}(1^\lambda, n)$.
2. $(\text{M}_D^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{HU}}, \mathcal{O}_{\text{CU}}, \mathcal{O}_{\text{ObtIss}}, \mathcal{O}_{\text{Issue}}, \mathcal{O}_{\text{Show}}}(\text{pk}_I)$; $\mathcal{A}$ then engages in Show with the challenger (as verifier) on a disclosed set M_D^* of its choice.
3. Return 1 iff $\text{ACVerify}(\text{pk}_I, \text{M}_D^*) = 1$ and there is no $(\cdot, \cdot, \text{M}_n) \in \text{L}_{\text{cred}}$ with $\text{M}_D^* \subseteq \text{M}_n$.

Anonymity $\text{Exp}_{\mathcal{A}}^{\text{ano-b}}(1^\lambda, n):$

1. $(\text{sk}_I, \text{pk}_I) \leftarrow \text{IKeygen}(1^\lambda, n)$.
2. $(i_0, i_1, \text{M}_D) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{HU}}, \mathcal{O}_{\text{CU}}, \mathcal{O}_{\text{Obtain}}, \mathcal{O}_{\text{Show}}}(\text{sk}_I)$.
3. If $i_0 = i_1$, or $\{i_0, i_1\} \not\subseteq \text{HU}$, or for some $i \in \{i_0, i_1\}$ there is no entry $(i, \text{cred}, \text{M}_n) \in \text{L}_{\text{cred}}$ with $\text{M}_D \subseteq \text{M}_n$, return 0.
4. Sample $b \leftarrow \{0, 1\}$; $\mathcal{A}$ as the verifier interacts with $\text{Show}(\text{sk}_I^b, \text{pk}_I, \text{M}_n, \text{M}_D, \text{cred})$.
5. $b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\text{HU}}, \mathcal{O}_{\text{CU}}, \mathcal{O}_{\text{Obtain}}, \mathcal{O}_{\text{Show}}}(\text{sk}_I)$.
6. If $\mathcal{O}_{\text{CU}}$ was queried on i_{b^*} for some $b^* \in \{0, 1\}$, return 0.
7. Return b^* .

Fig. 1. Security experiments for the underlying AC scheme [27].

- $\mathcal{O}_{\text{ObtIss}}(i, \text{M}_n)$: if $i \notin \text{HU}$, returns $\perp$; else, runs $\langle \text{Obtain}(\text{sk}_I^i, \text{pk}_I, \text{M}_n), \text{Issue}(\text{pk}_I^i, \text{sk}_I, \text{M}_n) \rangle$. Outputs the resulting credential cred and adds $(i, \text{cred}, \text{M}_n)$ to L_{cred} .
- $\mathcal{O}_{\text{Issue}}(i, \text{M}_n)$: if $i \notin \text{CU}$, returns $\perp$; else, runs $\text{Issue}(\text{pk}_I^i, \text{sk}_I, \text{M}_n)$ with $\mathcal{A}$ as the malicious user. If it succeeds, stores $(i, \perp, \text{M}_n)$ in L_{cred} ; else returns $\perp$.
- $\mathcal{O}_{\text{Obtain}}(i, \text{M}_n)$: if $i \notin \text{HU}$, returns $\perp$; else, runs $\text{Obtain}(\text{sk}_I^i, \text{pk}_I, \text{M}_n)$ with $\mathcal{A}$ as the malicious issuer. Outputs the resulting credential cred . If $\text{cred} = \perp$, returns $\perp$; else stores $(i, \text{cred}, \text{M}_n)$ in L_{cred} .
- $\mathcal{O}_{\text{Show}}(j, \text{M}_D)$: let $(i, \text{cred}, \text{M}_n)$ be the j -th entry in L_{cred} . If $i \notin \text{HU}$ or $\text{M}_D \not\subseteq \text{M}_n$, returns $\perp$; else runs $\text{Show}(\text{sk}_I^i, \text{pk}_I, \text{M}_n, \text{M}_D, \text{cred})$ with $\mathcal{A}$ as a malicious verifier.

Definition 5 (Correctness). A credential cred honestly issued on M_n always verifies for any disclosed subset $\text{M}_D \subseteq \text{M}_n$ via ACVerify .

Definition 6 (Unforgeability). An AC scheme is *unforgeable* if $\text{Adv}_{\mathcal{A}}^{\text{uf}}(1^\lambda, n) = \Pr[\text{Exp}_{\mathcal{A}}^{\text{uf}}(1^\lambda, n) = 1]$ is negligible for any PPT adversary $\mathcal{A}$, where $\text{Exp}_{\mathcal{A}}^{\text{uf}}$ is defined in Fig. 1.

Definition 7 (Anonymity). An AC scheme is *anonymous* if $\text{Adv}_{\mathcal{A}}^{\text{ano}}(1^\lambda, n) = |\Pr[\text{Exp}_{\mathcal{A}}^{\text{ano-1}}(1^\lambda, n) = 1] - \Pr[\text{Exp}_{\mathcal{A}}^{\text{ano-0}}(1^\lambda, n) = 1]|$ is negligible for any PPT adversary $\mathcal{A}$, where the experiments $\text{Exp}_{\mathcal{A}}^{\text{ano-b}}$ are defined in Fig. 1.

APPENDIX D

SECURITY PROOFS OF UPCS

Theorem 1 (Certificate Unforgeability). *If Ψ is EUF-CMA secure, then the UPCS construction in Fig. 5 of the main paper satisfies certificate unforgeability (Definition 2 of the*

main paper). Concretely, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ against the EUF-CMA security of Ψ , such that

$$\text{Adv}_{\mathcal{A}}^{\text{Cert-uf}}(1^\lambda, n) \leq \text{Adv}_{\Psi}^{\text{EUF-CMA}}(1^\lambda).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ whose advantage $\text{Adv}_{\mathcal{A}}^{\text{Cert-uf}}$ of winning the certificate unforgeability experiment $\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ against the EUF-CMA security of Ψ . $\mathcal{B}$ receives a challenge public key pk^* and access to a signing oracle $\mathcal{O}_{\text{Sign}}^{\text{EUF}}$, and simulates the certificate unforgeability experiment for $\mathcal{A}$ as follows.

Setup. $\mathcal{B}$ runs $\text{IKeygen}(1^\lambda, n)$ itself to obtain $(\text{sk}_I, \text{pk}_I)$, sets $\text{pk}_{\text{CA}} \leftarrow \text{pk}^*$, and gives $(\text{pk}_I, \text{pk}_{\text{CA}})$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ maintains the lists HU , CU , L_{cred} , L_{cert} , L_P , and L_{skp} .

Answering oracles. All identity-layer oracles ($\mathcal{O}_{\text{HU}}$, $\mathcal{O}_{\text{CU}}$, $\mathcal{O}_{\text{ObtIss}}$, $\mathcal{O}_{\text{Issue}}$, $\mathcal{O}_{\text{Show}}$) are answered directly using sk_I . The certificate-layer oracles are answered as follows.

- $\mathcal{O}_{\text{CObtIss}}(i, \text{M}_D)$: as in the real experiment, except that the CA signature is obtained from the signing oracle: $\mathcal{B}$ samples $(\text{sk}_P, \text{pk}_P)$, runs the honest Show/ACVerify interaction itself, queries $\sigma_{\text{CA}} \leftarrow \mathcal{O}_{\text{Sign}}^{\text{EUF}}(\text{pk}_P \parallel \text{aux})$, records sk_P , adds $(i, \text{pk}_P, \text{aux}, \sigma_{\text{CA}})$ to L_{cert} , and returns $(\text{pk}_P, \text{cert})$.
- $\mathcal{O}_{\text{CIssue}}(i, \text{M}_D, \text{pk}_P)$: $\mathcal{B}$ runs $\text{ACVerify}(\text{pk}_I, \text{M}_D)$ with $\mathcal{A}$ (playing the malicious user); if it accepts, $\mathcal{B}$ queries $\sigma_{\text{CA}} \leftarrow \mathcal{O}_{\text{Sign}}^{\text{EUF}}(\text{pk}_P \parallel \text{aux})$, adds $(i, \text{pk}_P, \text{aux}, \sigma_{\text{CA}})$ to L_{cert} , and returns cert ; else it returns $\perp$.
- $\mathcal{O}_{\text{CShow}}(j, m)$: answered using sk_P from L_{skp} , enforcing the one-time-use list L_P .

Extraction. Eventually $\mathcal{A}$ outputs $\text{cert}^* = (\text{pk}_P^*, \text{aux}^*, \sigma_{\text{CA}}^*)$. If the winning conditions of the experiment $\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}$ do not hold, $\mathcal{B}$ aborts; otherwise $\mathcal{B}$ outputs $(\text{pk}_P^* \parallel \text{aux}^*, \sigma_{\text{CA}}^*)$ as its forgery.

Analysis. The simulation is perfect: the only difference from the real experiment is that CA signatures are produced by $\mathcal{O}_{\text{Sign}}^{\text{EUF}}$ under sk^* instead of sk_{CA} , which is identically distributed since $\text{pk}_{\text{CA}} = \text{pk}^*$. Every CA signature produced in $\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}$ corresponds to a signing query $\text{pk}_P \parallel \text{aux}$ that $\mathcal{B}$ made to $\mathcal{O}_{\text{Sign}}^{\text{EUF}}$, and every such query is recorded as an entry $(\cdot, \text{pk}_P, \text{aux}, \cdot) \in \text{L}_{\text{cert}}$. It holds for both $\mathcal{O}_{\text{CObtIss}}$ and $\mathcal{O}_{\text{CIssue}}$, the only two oracles in which the CA signs. Hence, whenever $\mathcal{A}$ wins, the second winning condition in $\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}$ guarantees that $\text{pk}_P^* \parallel \text{aux}^*$ was never queried to $\mathcal{O}_{\text{Sign}}^{\text{EUF}}$, while the first condition in $\text{Exp}_{\mathcal{A}}^{\text{Cert-uf}}$ guarantees $\Psi.V_s(\text{pk}^*, \text{pk}_P^* \parallel \text{aux}^*, \sigma_{\text{CA}}^*) = 1$, so $(\text{pk}_P^* \parallel \text{aux}^*, \sigma_{\text{CA}}^*)$ is a valid EUF-CMA forgery. Therefore $\mathcal{B}$ succeeds with exactly $\mathcal{A}$'s advantage, which proves the bound. This completes the proof of Theorem 1. $\square$

Theorem 2 (Certificate Unlinkability). *If the anonymous credential scheme of [27] is anonymous (Definition 7), then the UPCS construction in Fig. 5 of the main paper satisfies certificate unlinkability (Definition 3 of the main paper). Concretely, for any PPT adversary $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ against the anonymity of the underlying credential scheme, such that*

$$\text{Adv}_{\mathcal{A}}^{\text{Cert-unl}}(1^\lambda, n) \leq \text{Adv}_{\mathcal{B}}^{\text{ano}}(1^\lambda, n).$$

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ whose advantage $\text{Adv}_{\mathcal{A}}^{\text{Cert-unl}}$ of winning the certificate unlinkability experiment $\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-}b}$ is non-negligible. We construct a PPT adversary $\mathcal{B}$ against the anonymity of the underlying credential scheme. $\mathcal{B}$ is given sk_I and access to $O_{\text{HU}}, O_{\text{CU}}, O_{\text{Obtain}}, O_{\text{Show}}$. In addition, $\mathcal{B}$ simulates the certificate unlinkability experiment for $\mathcal{A}$ as follows.

Setup. $\mathcal{B}$ generates the CA key pair $(\text{sk}_{\text{CA}}, \text{pk}_{\text{CA}}) \leftarrow \Psi.G_s(1^\lambda)$ itself, and gives $\mathcal{A}$ the issuer keys $(\text{sk}_I, \text{pk}_I)$ received from its own challenger together with $(\text{sk}_{\text{CA}}, \text{pk}_{\text{CA}})$. Besides, $\mathcal{B}$ maintains L_{cert}, L_P , and L_{skp} . The identity-layer oracles $O_{\text{HU}}, O_{\text{CU}}, O_{\text{Obtain}}, O_{\text{Show}}$ which $\mathcal{A}$ can access to are forwarded to $\mathcal{B}$'s own oracles, which are defined identically.

Answering Certificate-layer oracles. $\mathcal{B}$ answers them using sk_{CA} and its credential oracles.

- $O_{\text{CObtain}}(i, M_D)$: $\mathcal{B}$ samples $(\text{sk}_P, \text{pk}_P)$ and relays the messages between $\mathcal{A}$ (playing the CA) and its own $O_{\text{Show}}(j, M_D)$, where j indexes a credential of user i covering M_D . $\mathcal{B}$ then records sk_P , adds the certificate $(i, \text{pk}_P, \text{aux}, \sigma_{\text{CA}})$ completed by $\mathcal{A}$ to L_{cert} , and returns $(\text{pk}_P, \text{cert})$.
- $O_{\text{CShow}}(j, m)$: answered using sk_P from L_{skp} , enforcing the check for the one-time-use list L_P .

Challenge. When $\mathcal{A}$ outputs (i_0, i_1, M_D) , $\mathcal{B}$ forwards it as its own challenge in $\text{Exp}_{\mathcal{A}}^{\text{ano-}b}$. The anonymity experiment performs the same validity checks as Step 3 of $\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-}b}$, so invalid choices make both experiments return 0. Then, $\mathcal{B}$ samples $(\text{sk}_{P0}, \text{pk}_{P0})$ and $(\text{sk}_{P1}, \text{pk}_{P1})$, produces cert_0 for user i_0 exactly as in its answer to O_{CObtain} above (relaying a Show of user i_0 with $\mathcal{A}$).

Critically, $\mathcal{B}$ produces cert_1 by relaying the challenge $\text{Show}(\text{sk}_{I0}^b, \text{pk}_I, M_n, M_D, \text{cred})$ with $\mathcal{A}$. And $\mathcal{A}$, playing the CA, completes both $\sigma_{\text{CA},0}$ and $\sigma_{\text{CA},1}$ itself. $\mathcal{B}$ then gives $(\text{cert}_0, \text{cert}_1)$ to $\mathcal{A}$, answers further oracle queries as before, and outputs $\mathcal{A}$'s guess b^* .

Analysis. All oracle answers are distributed identically to the real experiment. For the challenge certificates, cert_0 is always issued to i_0 , whereas cert_1 is issued to i_0 if $b = 0$ and to i_1 if $b = 1$, i.e., to the user i_b selected by the anonymity challenger. Hence $\mathcal{A}$'s view is distributed exactly as in $\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-}b}$ with the same bit b , and the simulation is perfect. Moreover, $\mathcal{A}$ corrupts i_0 or i_1 (via the forwarded O_{CU}) if and only if $\mathcal{B}$ does, so the final corruption checks coincide in both experiments. Therefore $\Pr[\text{Exp}_{\mathcal{B}}^{\text{ano-}b}(1^\lambda, n) = 1] = \Pr[\text{Exp}_{\mathcal{A}}^{\text{Cert-unl-}b}(1^\lambda, n) = 1]$ for $b \in \{0, 1\}$, i.e., $\mathcal{B}$'s anonymity advantage equals $\mathcal{A}$'s certificate unlinkability advantage, which proves the bound. This completes the proof of Theorem 2. $\square$

APPENDIX E SECURITY PROOFS OF THE AIPA SCHEME

Theorem 3 (Signature Unforgeability). *If Ψ is EUF-CMA secure and deterministic, and H_0 and H_{ZKP} are collision-resistant, then the AIPA construction satisfies image signature unforgeability (Definition 5 of the main paper). Concretely, for any PPT adversary $\mathcal{A}$ making at most q queries to O_{CObtss} ,*

$$\text{Adv}_{\mathcal{A}}^{\text{Sig-uf}}(1^\lambda) \leq q \cdot \text{Adv}_{\Psi}^{\text{EUF-CMA}}(1^\lambda)$$

Proof. Let $\mathcal{A}$ be a PPT adversary against $\text{Exp}_{\mathcal{A}}^{\text{Sig-uf}}$ and output a forgery $(\hat{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*)$. We construct a reduction algorithm $\mathcal{B}$ that uses $\mathcal{A}$ to break the EUF-CMA security of Ψ . Concretely, $\mathcal{B}$ receives a challenge public key pk^* from the EUF-CMA challenger of Ψ and has access to a signing oracle $O_{\text{Sign}}^{\text{EUF}}$.

Setup. Initially, $\mathcal{B}$ runs $\text{Gen}_{\text{IA}}(1^\lambda, \mathcal{F})$ to simulate the system key $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$. It picks a target index $i^* \in [1, q]$ uniformly at random. And $\mathcal{B}$ publishes $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}})$ to $\mathcal{A}$.

Answering UPCS oracles. $\mathcal{B}$ simulates all UPCS oracles O_{UPCS} using $\text{sk}_{\text{sys}} = (\text{sk}_I, \text{sk}_{\text{CA}})$ it knows. Especially, it answers O_{CObtss} queries as follows. For the i -th query to O_{CObtss} :

- If $i \neq i^*$: generate a fresh key pair $(\text{sk}_{\text{img}}, \text{pk}_{\text{img}}) \leftarrow \Psi.G_s(1^\lambda)$, issue cert via $\text{UPCS}(\langle \text{CObtain}, \text{Cissue} \rangle)$, record $(\text{pk}_{\text{img}}, \text{sk}_{\text{img}}, \text{cert})$ in L_{sig} , and return $(\text{pk}_{\text{img}}, \text{cert})$.
- If $i = i^*$: embed the challenge key by setting $\text{pk}_{\text{img}} \leftarrow \text{pk}^*$, issue cert for pk^* via $\text{UPCS}(\langle \text{CObtain}, \text{Cissue} \rangle)$, record $(\text{pk}^*, \perp, \text{cert})$ in L_{sig} , and return $(\text{pk}_{\text{img}}, \text{cert})$. Since $\mathcal{B}$ does not know sk^* , no sk^* is stored in L_{sig} . And $\mathcal{B}$, with knowledge of sk_{CA} , can sign $\sigma_{\text{CA}} \leftarrow \Psi.S_s(\text{sk}_{\text{CA}}, \text{pk}^* \parallel \text{aux})$ even if it does not know sk^* , where $\text{cert} = (\text{pk}^*, \text{aux}, \sigma_{\text{CA}})$. But it cannot sign images using this key and must forward requests to $O_{\text{Sign}}^{\text{EUF}}$.

Answering $O_{\text{CShow}}(j, m)$. Let $(i, \text{pk}_P, \text{aux}, \sigma_{\text{CA}})$ be the j -th entry of L_{cert} and set $\text{cert} = (\text{pk}_P, \text{aux}, \sigma_{\text{CA}})$. If $i \notin \text{HU}$ or $\text{cert} \in L_P$, return $\perp$ as in the real game. Otherwise, if $\text{pk}_P \neq \text{pk}^*$, $\mathcal{B}$ computes $\sigma \leftarrow \Psi.S_s(\text{sk}_P, m)$ with the recorded one-time key; if $\text{pk}_P = \text{pk}^*$ (the challenge certificate), $\mathcal{B}$ forwards the message by querying $\sigma \leftarrow O_{\text{Sign}}^{\text{EUF}}(m)$. In both cases $\mathcal{B}$ records $(\text{pk}_P, \perp, \perp, \sigma)$ in S_{sig} , adds cert to L_P , and returns (cert, m, σ) . Since the challenge certificate is one-time and can be presented at most once, $\mathcal{B}$ forwards at most one message to $O_{\text{Sign}}^{\text{EUF}}$ via this oracle.

Answering $O_{\text{Sign}}(\text{pk}_{\text{img}}, I_0, \text{meta})$. When receiving a signing query, $\mathcal{B}$ looks up pk_{img} in L_{sig} . If not found, return $\perp$. Otherwise:

- Compute $h_0 \leftarrow H_{\text{ZKP}}(I_0)$ and $h_{\text{sig}} \leftarrow H_0(h_0 \parallel \text{meta})$.
- If the stored $\text{sk}_{\text{img}} \neq \perp$: compute $\sigma' \leftarrow \Psi.S_s(\text{sk}_{\text{img}}, h_{\text{sig}})$ directly.
- If $\text{sk}_{\text{img}} = \perp$, it means that the key is embedded ($\text{pk}_{\text{img}} = \text{pk}^*$): query the external oracle $\sigma' \leftarrow O_{\text{Sign}}^{\text{EUF}}(h_{\text{sig}})$.
- Set $\sigma_{\text{img}} \leftarrow \sigma'$, add $(\text{pk}_{\text{img}}, I_0, \text{meta}, \sigma_{\text{img}})$ to S_{sig} , initialize π_0 , and return $(\pi_0, \sigma_{\text{img}})$.

Extraction. The adversary $\mathcal{A}$ is allowed adaptive access to these oracles and may employ Prove_{IA} to produce arbitrary valid editing proofs. Eventually, $\mathcal{A}$ outputs a forgery $(\hat{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*)$. And $\mathcal{B}$ checks the winning conditions of the experiment $\text{Exp}_{\mathcal{A}}^{\text{Sig-uf}}$:

- $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \hat{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*) = 1$;
- $\exists (\text{pk}_{\text{img}}^*, \cdot, \cdot) \in L_{\text{sig}}$;
- $(\text{pk}_{\text{img}}^*, \cdot, \cdot, \sigma_{\text{img}}^*) \notin S_{\text{sig}}$.

If any condition fails, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ checks whether $\text{pk}_{\text{img}}^* = \text{pk}^*$. If not, $\mathcal{B}$ aborts. In this case, $\mathcal{B}$ guess of i^* wrongly. If equality holds, $\mathcal{B}$ sets $\sigma'^* = \sigma_{\text{img}}^*$. Since V_{IA}

accepts, its step 2 guarantees

$$\Psi.V_s(\text{pk}^*, H_0(h_0^*||\text{meta}^*), \sigma'^*) = 1,$$

where h_0^* and meta^* are parsed from π^* , and $h_{\text{sig}}^* = H_0(h_0^*||\text{meta}^*)$.

We show that $(h_{\text{sig}}^*, \sigma'^*)$ is a valid EUF-CMA forgery against Ψ . Every message signed by $\mathcal{B}$ under pk^* arises either as $h_{\text{sig}} = H_0(h_0||\text{meta})$ of some O_{Sign} query on $(\text{pk}^*, I_0, \text{meta})$, or as the message m of an O_{CShow} query on the challenge certificate; in both cases the resulting signature is recorded in S_{sig} . Since H_{ZKP} , H_0 , and $\Psi.S_s$ are all *deterministic*, if some prior O_{Sign} query used (I_0, meta^*) with $H_{\text{ZKP}}(I_0) = h_0^*$ (unique up to H_{ZKP} collisions), or if h_{sig}^* was submitted to O_{CShow} , then σ_{img}^* would already appear as the last component of an entry $(\text{pk}_{\text{img}}^*, \cdot, \cdot, \cdot)$ in S_{sig} , contradicting the winning condition. Therefore h_{sig}^* was never submitted to $O_{\text{Sign}}^{\text{EUF}}$, neither via O_{Sign} nor via O_{CShow} , and $(h_{\text{sig}}^*, \sigma'^*)$ constitutes a valid EUF-CMA forgery against Ψ .

Success probability. Whenever $\mathcal{A}$ succeeds and $\text{pk}_{\text{img}}^* = \text{pk}^*$, $\mathcal{B}$ breaks the EUF-CMA security of Ψ . Since i^* is chosen uniformly from $[1, q]$ and is independent of $\mathcal{A}$'s view, $\Pr[\text{pk}_{\text{img}}^* = \text{pk}^* \mid \mathcal{A} \text{ succeeds}] \geq 1/q$. Thus,

$$\text{Adv}_{\mathcal{A}}^{\text{Sig-uf}}(1^\lambda) \leq q \cdot \text{Adv}_{\Psi}^{\text{EUF-CMA}}(1^\lambda)$$

Since q is polynomial in λ and $\text{Adv}_{\mathcal{A}}^{\text{Sig-uf}}(1^\lambda)$ is non-negligible, $\text{Adv}_{\Psi}^{\text{EUF-CMA}}(1^\lambda)$ is also non-negligible. This completes the proof of Theorem 3. $\square$

Theorem 4 (Proof of Knowledge). *If IM satisfies knowledge soundness (Appendix B) and H_{ZKP} is collision-resistant, then the AIPA construction satisfies proof of knowledge (Definition 6 of the main paper). Concretely, for any PPT adversary $\mathcal{A}$ making at most n editing steps,*

$$\text{Adv}_{\mathcal{A}}^{\text{PoK}}(1^\lambda, \mathcal{F}) \leq n \cdot \text{Adv}_{\text{IM}}^{\text{KS}}(\lambda) + \text{Adv}_{H_{\text{ZKP}}}^{\text{CR}}(\lambda).$$

Proof. For any PPT adversary $\mathcal{A}$ that performs image editing n times, we construct an AIPA extractor $\mathcal{E}$ as follows. Let $\mathcal{E}_{\text{IM}}^i$ denote the i -th IM extractor. According to the definition of $\text{Exp}_{\mathcal{A}, \mathcal{E}}^{\text{PoK}}$, the extractor $\mathcal{E}$ takes as input the system master key $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$ (where sk_{sys} contains sk_I and sk_{CA}). The workflow of $\mathcal{E}$ is as follows:

- $\mathcal{E}$ constructs a PPT adversary $\mathcal{B}_i$ against the knowledge soundness of IM for each step $i \in [1, n]$. $\mathcal{B}_i$ receives $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$ and simulates the entire AIPA experiment for $\mathcal{A}$, using sk_{sys} to answer all UPCS oracles and O_{Sign} . When $\mathcal{A}$ outputs a forgery $(\hat{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*)$ accepted by $\text{Verify}_{\text{IA}}$, $\mathcal{B}_i$ parses the proof state π^* into $(\text{HC}^*, \Pi^*, \vec{f}^*, \text{meta}^*)$, extracts the i -th statement $y_i^* = (h_{i-1}^*, h_i^*, f_i^*)$ and proof π_i^* from Π^* , and outputs (y_i^*, π_i^*) . Since $\text{Verify}_{\text{IA}}$ accepts, $\text{IM.Verify}(\text{vk}_{f_i}, y_i^*, \pi_i^*) = 1$ for all i .
- Since $\mathcal{B}_i$ on input $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$, which can be considered as the common reference string $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}})$ and auxiliary information sk_{sys} , and outputs (y_i^*, π_i^*) , it is a standard zk-SNARKs PoK adversary. Thus, $\mathcal{E}$ runs the IM extractor $\mathcal{E}_i$ on the same input $(\text{pk}_{\text{IA}}, \text{vk}_{\text{IA}}, \text{sk}_{\text{sys}})$ to

obtain $w_i = (I_{i-1}, I_i)$, for each $i \in [1, n]$. It then outputs $(I_0, \text{meta}^*, \vec{f}^*)$.

In the $\text{Exp}_{\mathcal{A}, \mathcal{E}}^{\text{PoK}}$, we define $\mathbf{W}$ as the adversary's winning event, namely that $\text{Verify}_{\text{IA}}(\text{vk}_{\text{IA}}, \hat{T}^*, \text{cert}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \pi^*) = 1$ and $D_{\mathcal{F}}(\hat{T}^*, \text{pk}_{\text{img}}^*, \sigma_{\text{img}}^*, \vec{f}^*, I_0^*, \text{meta}^*) = 0$. We partition $\mathbf{W}$ into two mutually exclusive events:

- **Event E:** All sub-extractors output a witness other than $\perp$. That is, for each edit step i , the extractor $\mathcal{E}_i$ of IM successfully outputs $(I_{\text{in},i}, I_{\text{out},i})$.
- **Event F:** Some sub-extractor fails. That is, there is a step j such that although IM.Verify accepts π_j^* , the extractor $\mathcal{E}_j$ outputs $\perp$ and fails to extract a valid (I_{j-1}, I_j) from it.

Thus, we have the probability decomposition:

$$\Pr[\mathbf{W}] = \Pr[\mathbf{W} \wedge \mathbf{E}] + \Pr[\mathbf{W} \wedge \mathbf{F}]$$

Case 1: $\mathbf{W} \wedge \mathbf{E}$. In this case, for each $i \in [1, n]$, the extracted witness (I_{i-1}, I_i) satisfies $\mathcal{R}_{f_i}$, i.e.: $I_i = f_i(I_{i-1})$, $h_{i-1}^* = H_{\text{ZKP}}(I_{i-1})$, and $h_i^* = H_{\text{ZKP}}(I_i)$. Let (I'_i, I_{i+1}) denote the witness extracted for step $i+1$. Since $\text{Verify}_{\text{IA}}$ accepts, adjacent proofs share the link hash $h_i^* = H_{\text{ZKP}}(I_i) = H_{\text{ZKP}}(I'_i)$ for all $i \in [1, n-1]$. If $I_i \neq I'_i$ for some i , then (I_i, I'_i) is an explicit collision of H_{ZKP} . Hence an adversary causing $\mathbf{W} \wedge \mathbf{E}$ directly yields a collision-finding adversary that runs $\mathcal{A}$, runs all sub-extractors, and outputs the first mismatching pair (I_i, I'_i) , contradicting the collision resistance of H_{ZKP} . Otherwise $I_i = I'_i$ for all i , and by induction we obtain $\hat{T}^* = I_n = f_n \circ \dots \circ f_1(I_0^*)$. Because the original signature is also valid (which is ensured by the verification in Step 2 of $\text{Verify}_{\text{IA}}$), $D_{\mathcal{F}}$ would output 1, contradicting the definition of $\mathbf{W}$. Hence, under the event $\mathbf{W} \wedge \mathbf{E}$, a collision of H_{ZKP} must be found, and thus

$$\Pr[\mathbf{W} \wedge \mathbf{E}] \leq \text{Adv}_{H_{\text{ZKP}}}^{\text{CR}}(\lambda)$$

Case 2: $\mathbf{W} \wedge \mathbf{F}$. There is some index j for which $\mathcal{E}_j$ outputs $\perp$ while IM.Verify accepts π_j^* . This constitutes a direct attack on the knowledge soundness property of IM. By the knowledge soundness of IM, there exists an extractor $\mathcal{E}_i$ such that the failure probability of $\mathcal{E}_i$ is at most $\text{Adv}_{\text{IM}}^{\text{KS}}(\lambda)$. Taking the union bound:

$$\Pr[\mathbf{W} \wedge \mathbf{F}] \leq \sum_{i=1}^n \text{Adv}_{\text{IM}, \mathcal{B}_i}^{\text{KS}}(\lambda) \leq n \cdot \text{Adv}_{\text{IM}}^{\text{KS}}(\lambda)$$

Thus, by combining the two cases, we obtain

$$\text{Adv}_{\mathcal{A}}^{\text{PoK}}(1^\lambda, \mathcal{F}) = \Pr[\mathbf{W}] \leq n \cdot \text{Adv}_{\text{IM}}^{\text{KS}}(\lambda) + \text{Adv}_{H_{\text{ZKP}}}^{\text{CR}}(\lambda)$$

As $n = \text{poly}(\lambda)$ and both terms on the right are negligible, it follows that $\text{Adv}_{\mathcal{A}}^{\text{PoK}}$ is negligible as well. This completes the proof of Theorem 4. $\square$

Theorem 5 (Zero-Knowledge). *If IM satisfies zero-knowledge (as defined in Appendix B) for every transformation relation $\mathcal{R}_f$ ($f \in \mathcal{F}$), then the AIPA construction satisfies zero-knowledge (Definition 7 of the main paper). Concretely, for every PPT distinguisher $\mathcal{D} = (\mathcal{A}_1, \mathcal{A}_2)$ making at most n editing steps,*

$$\text{Adv}_{\mathcal{D}}^{\text{ZK}}(1^\lambda, \mathcal{F}) \leq \sum_{f \in \mathcal{F}} \text{Adv}_{\text{IM}, f}^{\text{key-ind}}(\lambda) + \sum_{j=1}^n \text{Adv}_{\text{IM}, f_j}^{\text{prv-ind}}(\lambda)$$

where n is the length of the editing sequence $\vec{f}$.

Proof. We prove the theorem by a standard hybrid argument that gradually transforms $\text{Real}_{\mathcal{D}}^{\text{zk}}$ into $\text{Ideal}_{\mathcal{D}}^{\text{zk}}$, using a simulator $\mathcal{S} = (\mathcal{S}_{\text{key}}, \mathcal{S}_{\text{prove}})$ constructed below from the zero-knowledge simulator of IM. The transformation proceeds in two phases: first, the honest key generation is replaced by the simulated one, one transformation relation at a time ($|\mathcal{F}|$ hybrids). Second, starting from the resulting experiment, the proofs in the chain are replaced by simulated proofs one editing step at a time (n hybrids), until the entire proof state is simulated and the experiment coincides with $\text{Ideal}_{\mathcal{D}}^{\text{zk}}$.

The proof intuition is as follows. Suppose a distinguisher $\mathcal{D}$ distinguishes the $\text{Real}_{\mathcal{D}}^{\text{zk}}$ and $\text{Ideal}_{\mathcal{D}}^{\text{zk}}$ experiments with non-negligible advantage. Since the two experiments are connected by the above sequence of $|\mathcal{F}| + n$ hybrids, each of which modifies only a single key pair or a single proof, this advantage must already appear between some pair of adjacent hybrids; otherwise the total advantage would be a sum of $|\mathcal{F}| + n$ negligible gaps, hence negligible. Any distinguisher of two adjacent hybrids, however, immediately yields an adversary against the zero-knowledge property of IM: the two hybrids differ only in one key pair or in the proof π_j of one editing step, so by embedding the corresponding IM challenge (challenge keys or a challenge proof) at exactly this position, an adversary $\mathcal{B}$ transfers $\mathcal{D}$'s advantage to the keys-indistinguishability or proof-indistinguishability game without any loss. This contradicts the assumed security of IM.

The simulator. We construct $\mathcal{S} = (\mathcal{S}_{\text{key}}, \mathcal{S}_{\text{prove}})$ from the IM simulator $(\mathcal{S}_{\text{IM}}^{\text{key}}, \mathcal{S}_{\text{IM}}^{\text{prove}})$:

- $\mathcal{S}_{\text{key}}(1^\lambda, \mathcal{F})$: for each $f \in \mathcal{F}$, run $(\text{pk}_f, \text{vk}_f, \text{td}_f) \leftarrow \mathcal{S}_{\text{IM}}^{\text{key}}(1^\lambda, \mathcal{R}_f)$. Generate the UPCS keys honestly: $(\text{sk}_I, \text{pk}_I) \leftarrow \text{UPCS.IKeygen}(1^\lambda, n)$ and $(\text{sk}_{CA}, \text{pk}_{CA}) \leftarrow \text{UPCS.CAKeygen}(1^\lambda)$. Output $\text{pk}_{IA} = \{\text{pk}_f\}_{f \in \mathcal{F}}$, $\text{vk}_{IA} = (\{\text{vk}_f\}_{f \in \mathcal{F}}, \text{pk}_I, \text{pk}_{CA})$, $\text{sk}_{\text{sys}} = (\text{sk}_I, \text{sk}_{CA})$, and $\text{td} = \{\text{td}_f\}_{f \in \mathcal{F}}$.
- $\mathcal{S}_{\text{prove}}(\text{pk}_{IA}, \text{td}, x)$: parse $x = (\vec{I}, \text{cert}, \text{pk}_{\text{img}}, \sigma_{\text{img}}, \text{meta}, \vec{f}, \text{HC})$ with $\text{HC} = [h_0, \dots, h_n]$ and $\vec{f} = [f_1, \dots, f_n]$. For each $i \in [1, n]$, compute $\pi_i \leftarrow \mathcal{S}_{\text{IM}}^{\text{prove}}(\text{pk}_{f_i}, \text{td}_{f_i}, (h_{i-1}, h_i, f_i))$. Output $\pi = (\text{HC}, \Pi = \{\pi_1, \dots, \pi_n\}, \vec{f}, \text{meta})$.

Note that in the $\text{Ideal}_{\mathcal{D}}^{\text{zk}}$ experiment, x is derived publicly from $\mathcal{A}_1$'s output as the honest algorithms would (the chain is computed from I_0 and $\vec{f}$), so every statement (h_{i-1}, h_i, f_i) is satisfied by the witness (I_{i-1}, I_i) , although $\mathcal{S}_{\text{prove}}$ receives only x and never the witnesses.

Setup indistinguishability. We first replace the honest setup by the simulated one, one relation at a time: for each $f \in \mathcal{F}$ (in any fixed order), the generation of $(\text{pk}_f, \text{vk}_f)$ is replaced by $\mathcal{S}_{\text{IM}}^{\text{key}}(1^\lambda, \mathcal{R}_f)$. Each replacement is justified by a direct reduction to the keys indistinguishability of IM (Appendix B): the reduction embeds the challenge keys as the f -component of $(\text{pk}_{IA}, \text{vk}_{IA})$, generates all remaining keys itself, and answers all oracle queries using sk_{sys} ; all proofs are still generated honestly, which requires only the proving keys and the witnesses, both known to the reduction. The total advantage in this phase is bounded by $\sum_{f \in \mathcal{F}} \text{Adv}_{\text{IM}, f}^{\text{key-ind}}(\lambda)$,

where key-ind denotes key indistinguishability. We denote the resulting experiment by H_0 : it proceeds as $\text{Real}_{\mathcal{D}}^{\text{zk}}$, except that all transformation keys are generated by $\mathcal{S}_{\text{IM}}^{\text{key}}$.

Hybrids over the editing steps. Starting from H_0 , we define hybrids H_j for $j = 0, \dots, n$ over the composition of the final proof π . In H_j , the first j proofs are simulated: $\pi_i \leftarrow \mathcal{S}_{\text{IM}}^{\text{prove}}(\text{pk}_{f_i}, \text{td}_{f_i}, (h_{i-1}, h_i, f_i))$. The remaining proofs are generated honestly with real witnesses. Everything else is as in H_0 . Finally, H_n is exactly the $\text{Ideal}_{\mathcal{D}}^{\text{zk}}$ experiment: keys from $\mathcal{S}_{\text{key}}$, proof state from $\mathcal{S}_{\text{prove}}$.

Indistinguishability of adjacent hybrids. For every $j \in [1, n]$, H_{j-1} and H_j are computationally indistinguishable. Suppose a PPT distinguisher $\mathcal{D} = (\mathcal{A}_1, \mathcal{A}_2)$ distinguishes H_{j-1} from H_j with non-negligible advantage. We construct a PPT adversary $\mathcal{B}$ against the proof indistinguishability of IM for the relation $\mathcal{R}_{f_j}$:

- $\mathcal{B}$ receives $(\text{pk}^*, \text{vk}^*)$ for $\mathcal{R}_{f_j}$, generated by $\mathcal{S}_{\text{IM}}^{\text{key}}(1^\lambda, \mathcal{R}_{f_j})$ with the trapdoor withheld. It embeds $(\text{pk}^*, \text{vk}^*)$ as the f_j -component of $(\text{pk}_{IA}, \text{vk}_{IA})$, generates the keys of all other relations via $\mathcal{S}_{\text{IM}}^{\text{key}}$ itself and keeps their trapdoors, generates the UPCS keys, and answers all O_{UPCS} and O_{Sign} queries using sk_{sys} exactly as the experiment does.
- When $\mathcal{A}_1^O$ outputs the full provenance tuple, $\mathcal{B}$ derives the chain honestly (it holds all images) and computes $\text{HC} = [h_0, \dots, h_n]$. It submits $(x_j, w_j) = ((h_{j-1}, h_j, f_j), (I_{j-1}, I_j))$ to its challenger and receives a proof π^* , which is either $\text{IM.Prove}(\text{pk}^*, x_j, w_j)$ or $\mathcal{S}_{\text{IM}}^{\text{prove}}(\text{pk}^*, \text{td}^*, x_j)$.
- For $i \in [1, j-1]$, $\mathcal{B}$ computes $\pi_i \leftarrow \mathcal{S}_{\text{IM}}^{\text{prove}}(\text{pk}_{f_i}, \text{td}_{f_i}, (h_{i-1}, h_i, f_i))$ using its own trapdoors; it sets $\pi_j \leftarrow \pi^*$; for $i \in [j+1, n]$, it generates π_i honestly via IM.Prove with the real witnesses.
- $\mathcal{B}$ assembles $\pi = (\text{HC}, \Pi, \vec{f}, \text{meta})$, sends it to $\mathcal{A}_2$, and outputs whatever $\mathcal{A}_2$ outputs.

If π^* is a real proof, $\mathcal{B}$ reproduces H_{j-1} exactly; if π^* is simulated, $\mathcal{B}$ reproduces H_j exactly. Hence the advantage of $\mathcal{B}$ equals that of $\mathcal{D}$, which is negligible by the proof indistinguishability of IM for $\mathcal{R}_{f_j}$.

Advantage analysis. By the triangle inequality over the $|\mathcal{F}|$ setup replacements and the n step hybrids,

$$\text{Adv}_{\mathcal{D}}^{\text{zk}}(1^\lambda, \mathcal{F}) \leq \sum_{f \in \mathcal{F}} \text{Adv}_{\text{IM}, f}^{\text{key-ind}}(\lambda) + \sum_{j=1}^n \text{Adv}_{\text{IM}, f_j}^{\text{prv-ind}}(\lambda)$$

Since each term is negligible by assumption, $|\mathcal{F}|$ is a fixed constant, and $n = \text{poly}(\lambda)$, $\text{Adv}_{\mathcal{D}}^{\text{zk}}$ is negligible, which completes the proof of Theorem 5. $\square$

APPENDIX F

PERFORMANCE COMPARISON ON ADDITIONAL IMAGE SIZES

Table IV in the main paper reports the comparison at size 2^{22} . Tables I–IV report the remaining image sizes from 2^{19} to 2^{23} pixels.

TABLE I

PERFORMANCE COMPARISON OF PROOF SYSTEMS FOR THE CROP (50% OF IMAGE)
EDIT ON AN IMAGE OF SIZE 2^{19} PIXELS.

Scheme	Prover runtime (s)	Verifier runtime (ms)	Peak memory (GB)	Proof size (B)
HyperVerITAS[22]*	43.1	5478	4.7	49056
VerITAS[17]*	48.5	2811	6.5	774507
VIMz [15]	174.6	107	0.7	10144
Trust nobody [20]	48.8	7080	2.5	2124
VIMz-Loua	153.4	6	20.4	448

*HyperVerITAS is instantiated with PST polynomial commitment; VerITAS is instantiated with FRI polynomial commitment.

TABLE II

PERFORMANCE COMPARISON OF PROOF SYSTEMS FOR THE CROP (50% OF IMAGE)
EDIT ON AN IMAGE OF SIZE 2^{20} PIXELS.

Scheme	Prover runtime (s)	Verifier runtime (ms)	Peak memory (GB)	Proof size (B)
HyperVerITAS[22]*	86.0	11002	9.5	50656
VerITAS[17]*	105.7	5080	12.3	818775
VIMz [15]	341.1	116	0.7	10144
Trust nobody [20]	97.5	14160	2.5	4248
VIMz-Loua	211.4	7	20.5	448

*HyperVerITAS is instantiated with PST polynomial commitment; VerITAS is instantiated with FRI polynomial commitment.

TABLE III

PERFORMANCE COMPARISON OF PROOF SYSTEMS FOR THE CROP (50% OF IMAGE)
EDIT ON AN IMAGE OF SIZE 2^{21} PIXELS.

Scheme	Prover runtime (s)	Verifier runtime (ms)	Peak memory (GB)	Proof size (B)
HyperVerITAS[22]*	170.4	22328	19.1	52256
VerITAS[17]*	196.2	9429	24.8	892497
VIMz [15]	531.7	122	1.0	10432
Trust nobody [20]	195.1	28320	2.5	8496
VIMz-Loua	225.2	8	20.9	448

*HyperVerITAS is instantiated with PST polynomial commitment; VerITAS is instantiated with FRI polynomial commitment.

TABLE IV

PERFORMANCE COMPARISON OF PROOF SYSTEMS FOR THE CROP (50% OF IMAGE)
EDIT ON AN IMAGE OF SIZE 2^{23} PIXELS.

Scheme	Prover runtime (s)	Verifier runtime (ms)	Peak memory (GB)	Proof size (B)
HyperVerITAS[22]*	669.0	88682	74.8	55456
VerITAS[17]*, [†]	—	—	—	—
VIMz [15]	1824.8	206	1.9	10720
Trust nobody [20]	747.8	108560	2.5	32568
VIMz-Loua	376.1	6	22.2	448

*HyperVerITAS is instantiated with PST polynomial commitment; VerITAS is instantiated with FRI polynomial commitment. [†]VerITAS ran out of memory at size 2^{23} .