Verifiable Compression of Images

Yue Zhang[1], Mingxun Zhou[2], and Benedikt Bünz[3]

[1,3]New York University
[2]The Hong Kong University of Science and Technology

Abstract

AI image generation has made image misinformation a serious concern. To address this issue, the Coalition for Content Provenance and Authenticity (C2PA) standard adopts digital signatures to attest that an image originates from an authorized source, such as an attested camera or authorized AI provider. In practice, however, raw images are rarely published directly: photos are typically compressed before publication, which invalidates any C2PA signature. Recent works have proposed the use of zero-knowledge proofs (zk-SNARKs) to prove that only allowed edits were applied to a C2PA-signed original image. Unfortunately, prior works only support simple edits (e.g. cropping, blurring, and resizing) and do not support lossy image compression like JPEG, which is ubiquitous. The purpose of compression is to save communication. Therefore, unlike other edits, it cannot be verified directly, requires small proof sizes, and is unsuitable to outsourcing.

To address these limitations, we present SPEG, the first practical proof system that supports the full image-transmission pipeline on personal devices. We present two protocols that support JPEG compression, while proving validity of the C2PA signature on the original image. The two modes are incomparable: the first is compatible with an arbitrary hashing algorithm (we use Poseidon for efficiency), whereas the second is significantly faster but requires the C2PA to use a polynomial commitment (e.g., KZH) instead of a conventional hash. Our key optimizations are handling the non-algebraic JPEG Encoding outside the proving circuit, and avoiding range checks in our floating-point arithmetic. We can prove the JPEG compression of an FHD(1080p) image in 47s in the Poseidon mode and in 2s in the fast mode. The fastest prior work (VerITAS) which only handles simple resizing, requires 227s on the same consumer hardware.

Additionally, we identify a security gap in VIMz (PETS 25) that enables forgery of proofs for unauthorized images and provide a fix. We also prove that we can securely use the popular powers-of-tau SRS with the polynomial commitment scheme KZH.

1 Introduction

The rapid progress of generative AI has fundamentally changed the trust model of digital images. Modern text-to-image and image-editing models can synthesize photorealistic scenes, alter facial expressions, insert or remove objects, and produce visually plausible evidence of events that never occurred [1]. Deepfake techniques [2] make it increasingly difficult for ordinary users, journalists, and downstream automated systems to distinguish authentic camera-captured content from manipulated or fully AI-generated media. This creates a security problem: when an image is used as evidence for formal information, the Verifier needs more than visual plausibility. The Verifier needs cryptographic assurance about the image source.

{yz11727,bb}@nyu.edu, mingxunz@ust.hk

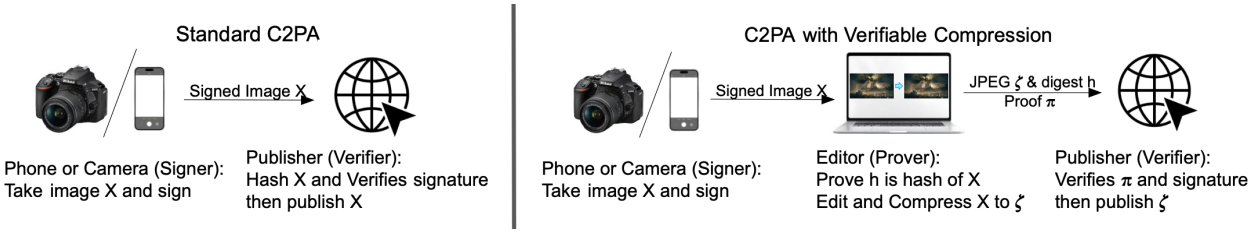


Figure 1: Standard C2PA versus version with verifiable compression. Standard C2PA (left) authenticates only byte-identical delivery. SPEG (right) authenticates a published JPEG that was derived from a signed source image.

The Coalition for Content Provenance and Authenticity (C2PA) standardizes this form of provenance [3] using digital signatures [4]. An attested camera[1], or an authorized image provider, computes a cryptographic digest of the image and signs this digest together with relevant metadata. A downstream Verifier can then check the signature on the signed content. Leica, Nikon, and Sony have all developed cameras with C2PA credentials [5] [6] [7], and AI companies like Google [8] and OpenAI [9] also support C2PA attestation for images they generated.

However, end-to-end image transmission almost always requires compression, most commonly JPEG. Once compression changes the bytes, the original digest no longer matches, so the C2PA signature cannot be verified directly. A camera also cannot sign a single compressed output for all workflows: many transformations occur before export from RAW or other high-fidelity formats, and publishers often choose the quality factor, subsampling mode, or target file size for each distribution channel. The Verifier therefore needs to authenticate the final JPEG as a valid derivative of the signed source image.

Recent zero-knowledge systems address byte mismatch by proving that a published image was derived from a signed source through an approved transformation. Existing systems [10] [11] [12] [13] [14] support basic edits including cropping, blurring, and resizing. Resizing can naturally serve as a strawman form of image compression, since reducing the number of pixels also reduces the file size. However, resizing does not preserve perceptual quality and provides poor rate–distortion tradeoffs. Figure 2 illustrates this difference.

Thus, practical image-transmission workflows still rely on JPEG compression, and existing proof systems still fall short of this goal. The Haar Wavelet Transform [15] can also serve as an image compression algorithm. However, it often introduces visible blocking artifacts and provides poor edge sharpness. For these reasons, JPEG remains the dominant format for real-world image compression [16].

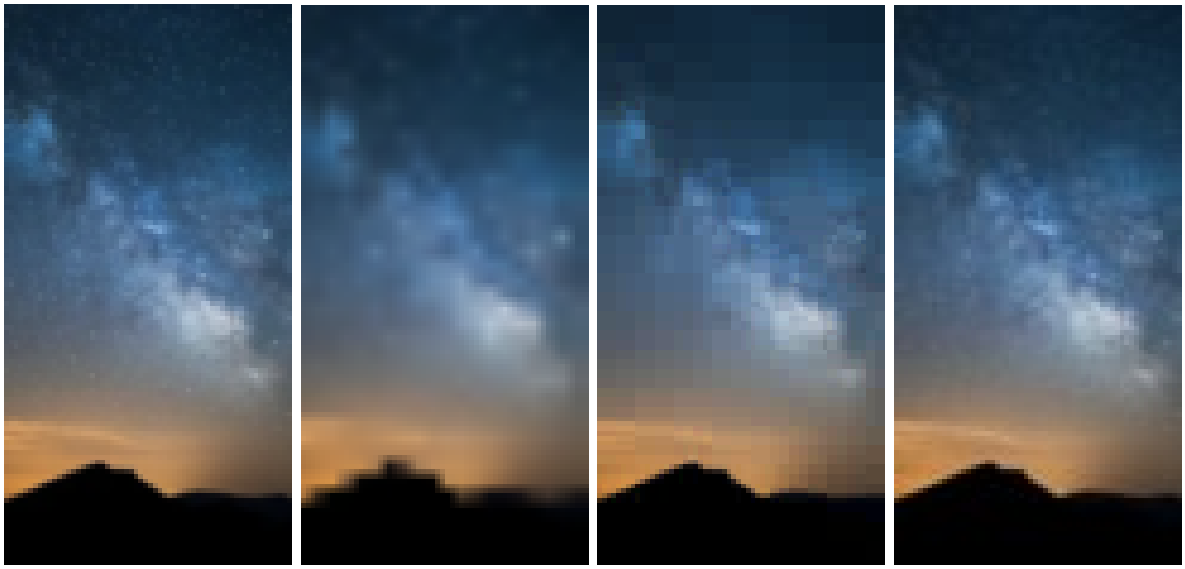
This leads to the following technical question:

Research Question: *Can we build a proof system for the C2PA-style authenticity of JPEG-compressed images that achieves practical Prover cost, low memory usage, and a proof small enough for end-to-end transmission?*

1.1 Our Contributions

Main observation. Our key observation is that JPEG compression can be divided into two parts: *Lossify* and *Encoding*. Lossify determines which image information survives compression and therefore must be authenticated. Encoding represents the resulting quantized coefficients with

[1] More specifically a secure enclave on the camera.



(a) Original (b) Resizing (c) Haar wavelet (d) JPEG

Figure 2: Resizing (1,908 bytes), Haar Wavelet Transform (1,902 bytes), and JPEG (1,851 bytes) all compress the Original image (19,027 bytes) with approximately 10% compression rate, but they produce different perceptual quality.

compact byte stream. SPEG exploits this structure by moving the non-algebraic Encoding stage outside the circuit and reusing the Verifier's standard JPEG Decoding workflow, which substantially reduces circuit complexity. We also use an efficient integer mapping to represent real-valued JPEG arithmetic. This avoids the extra range checks required by the remainder-based floating-point mappings used in existing systems, further reducing circuit complexity.

Technical contribution: SPEG protocol. Based on this observation, we design SPEG, the first proof system that supports proving JPEG compression of authenticated images and therefore supports end-to-end image transmission. The protocol has three stages. First, a camera or phone captures an image and signs on the digest. Second, the Prover with the authenticated source image proves knowledge of the original image that hashes to the claimed digest and compresses to the JPEG output. Third, the Verifier verifies the proof and checks the signature on the digest under the public key. The main technical challenge is that JPEG's Encoding stage is not circuit-friendly. SPEG addresses this problem by safely moving the Encoding stage outside the circuit, which, together with a more efficient Mapping for floating-point arithmetic, yields a deterministic circuit that is practical for generic SNARKs. These designs leverage the Verifier's ordinary decoding workflow and thus introduce no extra operational overhead.

Efficient proving calculation of hash. Proving the correctness of hash computation is expensive for large images, leads to impractical memory usage and unreasonable proof size. SPEG addresses this challenge with two modes.

SPEG-POSEIDON. This mode explicitly proves Hash computation inside circuit, so it directly fits the C2PA workflows. The structure is compatible with an arbitrary hashing algorithm; our implementation uses Poseidon for efficiency. We adopt folding scheme to reduce proof size and memory usage compared with a monolithic SNARK. We also extend it to an extra mode SPEG-SHA256, which supports the standard SHA-256 hash function used by C2PA.

SPEG-COMMIT. This fast mode requires the C2PA workflow to use a polynomial commitment

Table 1: Comparison of the three SPEG modes.

Mode	Binding primitive	Primitive verification	Input Range Checks	Prover speed
SPEG-SHA256	SHA-256	Explicitly in circuit	Explicitly in circuit	614.63 μs/px
SPEG-Poseidon	Poseidon	Explicitly in circuit	Explicitly in circuit	22.73 μs/px
SPEG-Commit	KZH commit	Implicitly by opening	Implicitly at capture	0.62 μs/px

Note. Performance is measured on an Apple M1 Pro chip using 8 threads. For an HD image of size 1280×720, i.e. approximately one million pixels, the Prover times are less than 567 s, 21 s, and 1 s, respectively.

as the authentication primitive instead of a traditional hash. This design allows the Verifier to check image binding implicitly, which significantly simplifies the circuit. We then use a GKR-based proof system [17] for optimal Prover time and pair it with KZH commitments [18] to keep the proof size trivial.

We fully implement [2] all modes for images up to 10 million pixels under personal-device-level settings. Our system reduces Prover time for hash computation by more than an order of magnitude over existing work while keeping proofs small enough for practical image transmission.

We summarize three SPEG modes in Table 1.

Additional contributions.

- We identify a security gap in VIMz [13]. We describe the vulnerability, present a concrete forgery attack, and provide the corresponding fix.
- We prove that the KZH polynomial commitment scheme [18] can be used with KZG SRS [19].

1.2 Related Work

Existing proof systems. PhotoProof [10] initiates the use of zk-SNARKs for authenticated photo editing, but its Prover cost is too high and it supports only tiny images. Recent works like zkIMG [11] and VerITAS [12] express the proof of a valid signature and the proof of a claimed image operation algebraically, encode the full computation as a constraint system, and prove it with a general-purpose SNARK, zkIMG adopts Halo2 [20] based on [21] inner product argument (IPA), and VerITAS uses Plonky2 [22] system. This monolithic approach does not scale to realistic image sizes. It requires impractical memory resources, and for JPEG it would also produce proof overhead that is too large for a compression-oriented transmission setting. VerITAS further introduces a commitment-hash design in which the correctness of the commitment computation is implicitly verified with permutation argument [23]; however, its treatment of floating-point arithmetic still relies on range checks, which prevent the use of GKR-style protocols to obtain optimal Prover time.

VIMz [13] and Eva [14] address the scalability limitations of monolithic SNARKs by adopting NOVA folding schemes [24] compiled with Pedersen commitment with Bulletproofs [21] IPA. In these systems, the Prover processes the image incrementally, typically one subimage or one row at a time, and folds each step into an accumulated proof state. Eva additionally supports partial H.264 pipeline, which compresses video with high compression rate. However, its core mechanism relies on inter-frame prediction, and its per-frame transformation is much simpler than JPEG. Its performance also mainly benefits from GPU acceleration and usage of Griffin hash function [25]. These design choices are largely orthogonal to the problem setting. This approach reduces memory usage and proof size, but folding introduces a new Prover bottleneck through additional

[2] Our code is available at https://github.com/hanbunko223/VerImgCmp.git

commitments. In particular, these systems inherit the cost of Nova cross-term commitments, which remains impractical for our image-transmission setting. More importantly, none of these systems supports authenticated JPEG compression.

We compare our protocol with recent proof systems in Table 2.

Table 2: Comparison with representative proof systems for authenticated image transformations.

System	Supported Compression	GPU requirement	FP range checks	Prover time	Prover RAM	Proof size
ZK-IMG [11]	Resize	No	PLONKish lookup	$O(P \log P)$	$O(P)$	$O(\log P)$
VerITAS [12]	Resize	No	PLONKish lookup	$O(P \log P)$	$O(P)$	$O(\log^2 P)$
VIMz [13]	Resize	No	Bit decomposition	$O(P)$	$O(N)$	$O(\log^2 N)$
Eva [14]	Partial H.264	CUDA required	LogUp	$O(P)$	$O(1)$	$O(1)$
SPEG-Poseidon	**JPEG**	**No**	**Free**	**O(P)**	**O(1)**	**O(1)**

Asymptotic complexity is measured with respect to the image size $P = M \times N$, where M and N denote the image height and width, respectively. FP denotes floating-point arithmetic. Here, "Free" means that our JPEG circuit avoids the extra range checks introduced by FP arithmetic. The comparison with other SPEG modes appears later in Section 8.

Alternative cryptographic designs. A different cryptographic direction is to use redactable signatures [26], and more generally homomorphic signatures [27] [28], which allow a party to transform a message-signature pair (m, σ) into a signature on a function of the original message denoted by $((f(m), f), \sigma')$, for a transformation function f. These approaches are attractive only when the allowed computation is simple, fixed in advance, and algebraically well structured. They are therefore impractical for complex transformations such as JPEG compression.

Watermarking [29] is another approach to mitigating misinformation. A watermark is a pattern embedded in an invisible manner and can be used to verify that an image was generated by a specified generation model. zkGPT [30] can also serve as an approach by proving that the image was generated by the model. These approaches study provenance from the perspective of AI generation and are of independent interest. However, the absence of such evidence does not imply that the image was not AI-generated. By contrast, C2PA is designed to provide positive provenance for authorized content, such guarantee is better aligned with goal of authenticating images.

2 Technical Overview

Overview of SPEG. SPEG provides proofs for authenticated JPEG compressed images. The prover aims to establish that a public JPEG image is obtained by applying JPEG compression to an authenticated original image captured by an attested camera. Authentication is provided by a digital signature over the digest of the original image. To establish that the image is properly authenticated, it suffices for the proof system to prove that the digest is correctly computed from the original image, then have the verifier check the signature on this digest outside of the system.

Accordingly, the authentication of a public JPEG image is established by combining the following three components:

- A signature verification algorithm that verifies the signature on the digest h.
- A proof of Hash showing that the digest h is correctly computed from the original image X.
- A zero-knowledge proof showing that the public JPEG file ζ was correctly obtained by applying JPEG compression with parameters par to the original image X;

Avoiding non-algebraic part. One major bottleneck in JPEG compression is the circuit-unfriendly nature of JPEG Encoding. SPEG avoids this cost by exploiting a structural property of JPEG: the compression pipeline naturally decomposes into two qualitatively different parts. The lossy part of the algorithm arises from quantization, together with optional downsampling. These are the Lossify stages that discard information and therefore must be proven by the Prover.

By contrast, JPEG Encoding part is lossless. Its role is not to change the image content, but only to represent the already quantized coefficients more compactly, and the Verifier can recover the information by decoding the JPEG object. Thus, only the Lossify part must be proved by the Prover inside the circuit. This allows us to place the circuit-unfriendly JPEG Encoding stage outside the circuit. Concretely, the Prover outputs the compressed JPEG object ζ, the Verifier performs JPEG Decoding outside the circuit, and the recovered quantized coefficients are used as public input to the proof.

Notice that this design imposes no extra overhead on the Verifier, since JPEG Decoding is already part of the Verifier's standard workflow for displaying the image.

Minimizing commitment cost. Proving authenticity requires showing that a digest h was computed correctly from the witness image X. This is the shared cost across all authenticated image-transformation systems; it dominates the cost and accounts for the majority of Prover time. The challenge is to obtain a proof system with small proof size, unlike prior monolithic SNARK approaches, while keeping Prover time practical, unlike prior folding-based approaches.

Our key insight is to optimize for witness weight rather than only witness length. Commitment cost depends on how many bits the committed witness elements carry. Commitments to full-field elements can therefore be expensive and account for most commitment time.

This perspective guides two design choices. First, SPEG uses NeutronNova [31] instead of Nova, avoiding Nova's extra commitment to a full-field cross-term vector. Second, SPEG uses bit decomposition for 8-bit range checks rather than lookup arguments. Lookup arguments are effective in many generic settings, but they commit to full-field elements; in our image-authentication setting, simple bit decomposition has lower witness weight and is substantially lighter.

Proving floating-point calculation. The next obstacle is the floating-point arithmetic used in JPEG transformations. To model real-valued computation with a deterministic circuit, existing systems typically map real numbers to integers using quotient-and-remainder relations and then range-check the remainder. Although conceptually simple, the range checks for the remainder are expensive.

To avoid this cost, SPEG uses a more efficient Mapping inspired by the quantization technique in zkCNN [32]. A real value a is mapped directly to an integer q for representation inside the circuit. However, simply applying the zkCNN strategy is insufficient for JPEG because the coefficient matrix after Lossify must remain sparse enough for effective compression. A naive mapping strategy that leaves small nonzero remainders does not preserve the intended compression rate.

SPEG resolves this issue by modifying the quantization part of Lossify stage without hurting the compression effect. Standard JPEG Lossify part divides the coefficient by an entry of a quantization matrix Q, which would require correct division and mapping. Instead, SPEG uses an approximate inverse matrix $R \approx 1/Q$ and maps this inverse into the integer domain. High-frequency positions that should be discarded under the target compression ratio are mapped to zero. These designs preserve the required sparsity pattern and yield an efficient deterministic circuit without extra range checks for remainders.

Polynomial commitment as hash. Inspired by VerITAS [12], our SPEG-Commit mode replaces the in-circuit hash computation with a polynomial commitment that serves the same binding role as a hash digest. In particular, SPEG-Commit offloads part of the computation from the prover to the signer: the signer generates a polynomial commitment to the original image, while

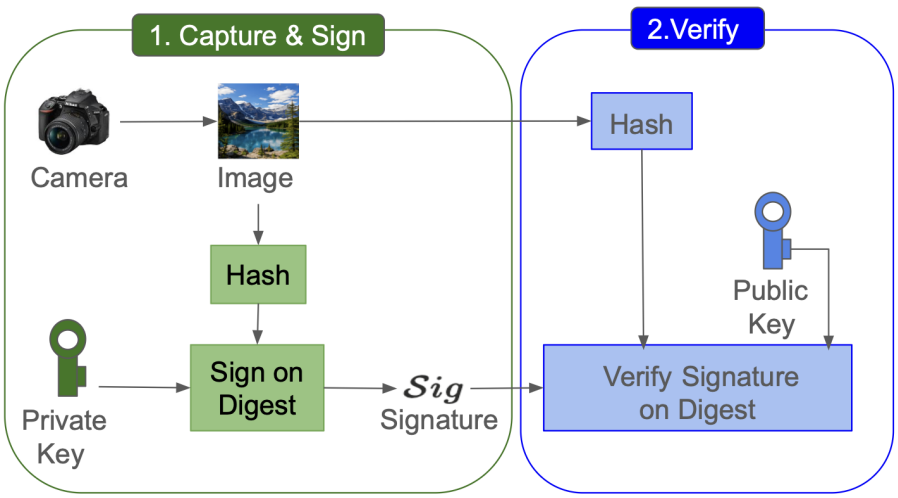


Figure 3: Standard C2PA workflow. The Signer signs on a digest, and the Verifier checks the signature on the digest.

the prover no longer needs to prove the expensive hash computation inside the circuit. Verifier checks commitment correctness implicitly by opening the commitment. This trade-off substantially reduces prover cost and circuit complexity at the expense of additional computation for the signer. We therefore target SPEG-Commit at settings where the signer has comparatively greater computational resources, such as generative-AI service providers or high-performance cameras.

Unlike VerITAS which uses PLONK for range checks, our floating-point arithmetic design with efficient circuit enables SPEG-Commit to use GKR, an interactive proof system with optimal Prover time. We adopt the proof system from zkCNN [32] and replace Hyrax [33] with KZH polynomial commitments [18] to obtain a trivial proof size.

3 Preliminaries

3.1 Notations

We write λ for the security parameter and $\mathsf{negl}(\lambda)$ for a negligible function. Let $\mathbb{F}_q$ be a finite field of order q. We write vectors as lowercase letters with arrows, such as $\vec{v}$, and matrices as uppercase letters, such as X.

We specially use $\odot$ to denote Hadamard Product.

3.2 C2PA standard

A Coalition for Content Provenance and Authenticity (C2PA) signer binds the captured image and its metadata to a digest computed with a specified hash function, then signs on the digest using a digital signature scheme [4]. The Verifier recomputes the digest on the delivered file and checks the signature. Figure 3 shows the basic workflow.

3.3 JPEG Compression

JPEG compression [34] has two conceptual parts. The first part transforms the image into a coefficient representation in which visually important information is concentrated in a small number

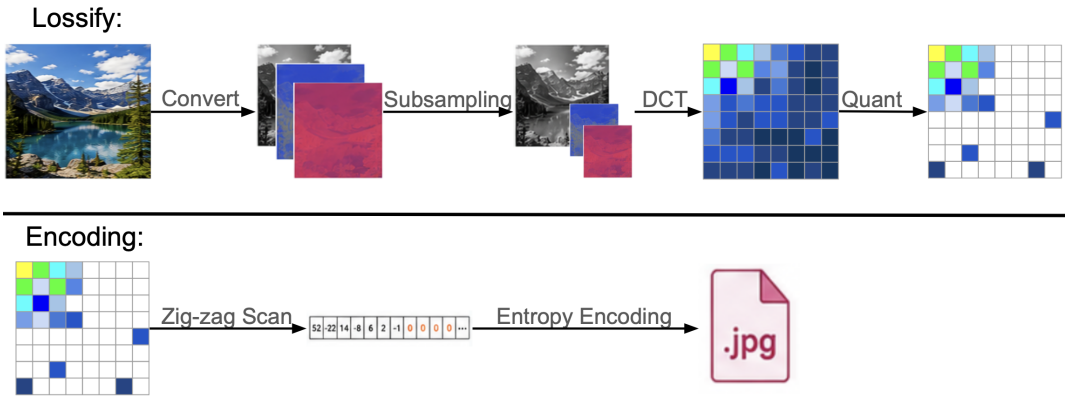


Figure 4: JPEG Encoding pipeline: colorspace conversion, optional subsampling, 8×8 DCT, quantization, then entropy coding. We prove the first stages in SNARK; decoding handles the rest.

of low-frequency entries and less important information is pushed into high-frequency entries. JPEG then discards part of that high-frequency information through quantization. The second part encodes the resulting sparse coefficients efficiently as a byte stream. Figure 4 sketches this pipeline.

3.3.1 Lossify

Algorithm 1 JPEG_Lossify$(X; par = \{s, D, Q\})$

Input: Original image X, subsampling mode s=`{'Full', 'Subsample'}`, DCT matrix D, Quant tables Q
Output: JPEG bitstream ζ
$(X^Y, X^{Cb}, X^{Cr}) \leftarrow X$; $(Q^Y, Q^{Cb}, Q^{Cr}) \leftarrow Q$;
if s==`'Subsample'` **then**
    $X'^{Cb} = \{X^{Cb}[2i, 2j]\}_{i\in[0,H/2),j\in[0,<W/2)}$
    $X'^{Cr} = \{X^{Cr}[2i, 2j]\}_{i\in[0,H/2),j\in[0,<W/2)}$
    $X^{Cb} = X'^{Cb}$; $X^{Cr} = X'^{Cr}$
end if
$\zeta \leftarrow \varepsilon$
$\text{PREVDC}_Y \leftarrow 0$, $\text{PREVDC}_{Cb} \leftarrow 0$, $\text{PREVDC}_{Cr} \leftarrow 0$
for $c \in \{Y, Cb, Cr\}$ **do**
    $M \leftarrow X^c - 128$
    for all 8×8 blocks B in M **do**
        $T \leftarrow D \cdot B \cdot D^\top$
        $Z \leftarrow T/Q^c$ element-wise
        $\vec{a} \leftarrow \text{ZIGZAGSCAN}(Z)$
        $\zeta \leftarrow \zeta \,\|\, \text{JPEG_ENCODE}(\vec{a}, \text{PREVDC}_c)$
        $\text{PREVDC}_c \leftarrow \vec{a}_0$
    end for
end for
return ζ

The first part is Lossify. For a given parameter set par, the encoder converts RGB pixels to YCbCr, optionally subsamples the chroma channels, splits each channel into 8×8 blocks, applies a discrete cosine transform (DCT), and quantizes each coefficient using the corresponding luminance or chrominance table.

The DCT is the core mathematical step in JPEG. It converts each 8×8 block from the spatial domain to the frequency domain. In the spatial domain, the block is represented by pixel intensities at fixed positions. In the frequency domain, the same block is represented by its coefficients on 64 cosine basis patterns with different horizontal and vertical frequencies. The upper-left coefficient captures the lowest-frequency component, corresponding to slow intensity changes across the block. Coefficients toward the lower-right corner capture higher-frequency components, such as sharp edges, fine textures, and local variation. In this way, the DCT separates image content by spatial frequency.

This change of basis is important because natural images tend to concentrate most visually important energy in the low-frequency coefficients. Human vision is much less sensitive to small errors in high-frequency components than to errors in low-frequency structure such as shading, contours, and large shapes. JPEG exploits this perceptual asymmetry. After applying the transform $T = D \cdot B \cdot D^{\top}$ to a block B, the encoder divides each coefficient by an entry of a quantization table and rounds the result. High-frequency entries are quantized more aggressively and often become zero. In other words, each block is represented as a weighted combination of the 64 cosine basis images, and JPEG keeps the coarse low-frequency combination more accurately than the fine high-frequency one.

The Lossify stage is the source of JPEG's information loss, and therefore must be proved by the Prover inside the circuit, as shown in Algorithm 1.

3.3.2 Encoding

After quantization, JPEG does not immediately emit the 8×8 coefficient matrix in its natural two-dimensional order. Instead, it first applies a zig-zag scan that lists the coefficients from low frequency to high frequency. This ordering is important because quantization usually creates long runs of trailing zeros in the high-frequency region. JPEG then encodes the first coefficient, the DC term, differentially relative to the previous block's DC term in the same channel. The remaining 63 AC coefficients are encoded by run-length compression of zeros together with the bit-length and value bits of each non-zero coefficient.

Algorithm 2 shows this entropy-coding step in simplified form. The exact wire format uses Huffman tables, but the logical structure is the same: DC coefficients are delta-coded, and AC coefficients are represented by zero runs interleaved with non-zero values. This stage is lossless. It does not change the quantized coefficients; it only produces a shorter representation of them. We denote by $\mathsf{DecCoeff}(\zeta; \mathsf{par})$ the coefficient tensor recovered by standard JPEG Decoding before inverse quantization and inverse DCT. This is the public object that our Verifier compares against the circuit output.

3.4 zk-SNARKs

Zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) [35] let a Prover convince a Verifier that a statement is true without revealing the underlying witness. We write an NP relation as $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$, where $x \in \mathcal{X}$ is the public instance and $w \in \mathcal{W}$ is the private witness. The associated language is $L_{\mathcal{R}} = \{x : \exists w \text{ such that } (x, w) \in \mathcal{R}\}$.

Algorithm 2 JPEG_Encoding($\vec{a}$, PREVDC)

Input: vector $\vec{a}$, PREVDC for DC encoding
Output: encoded bitstring ξ
$\xi \leftarrow \varepsilon$
$\Delta \leftarrow \vec{a}_0 - \text{PREVDC}$
$\xi \,\|\, \text{HUFFMAN ENCODE}(\Delta)$
run $\leftarrow 0$
for $k = 1$ to 63 **do**
    if $\vec{a}_k = 0$ **then**
        RUN $\leftarrow$ RUN $+ 1$
    else
        $\xi \,\|\, \text{HUFFMAN ENCODE}(\text{run})$
        $\xi \,\|\, \text{HUFFMAN ENCODE}(\vec{a}_k)$
        RUN $\leftarrow 0$
    end if
end for
if RUN > 0 **then**
    $\xi \,\|$ end-of-block marker
end if
return ξ

A zk-SNARK for $\mathcal{R}$ is a tuple of probabilistic polynomial-time algorithms $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$, for instance-witness pair $(x, w) \in \mathcal{R}$:

- $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R})$ outputs public parameters;
- $\pi \leftarrow \mathsf{Prove}(\mathsf{pk}, x, w)$ generates proof of $(x, w) \in \mathcal{R}$;
- $b \leftarrow \mathsf{Verify}(\mathsf{vk}, x, \pi)$ verifies the proof.

The zk-SNARKs must be complete, knowledge sound, zero-knowledge, non-interactive, and succinct. Appendix A.1 gives the formal definitions.

3.5 IVC/Folding Scheme

Incrementally verifiable computation (IVC) lets a Prover certify a long computation as a sequence of small steps. At step i, the Prover maintains a state z_i and a proof that z_i was obtained by correctly applying the step function from the initial state. A folding scheme compresses multiple proof obligations into a single accumulated obligation, so the Prover can continue the computation without carrying a proof whose size grows with the number of steps. A final decider proof can then certify the accumulated relation succinctly.

3.6 Polynomial Commitments

A polynomial commitment lets a committer publish a short commitment to a polynomial f and later prove claimed evaluations of the form $y = f(r)$ without sending the full polynomial. A polynomial commitment is *binding* if, after the commitment is fixed, no efficient adversary can open it to two different values at the same point, except with negligible probability. Appendix A.2 gives the formal definitions.

4 Threat Model

We consider a provenance setting in which a trusted camera is equipped with a secret signing key. Each time the camera captures a photo, it signs the RAW image together with associated metadata such as timestamp, location, device information, and other provenance-relevant attributes. The corresponding public verification key is available to Verifiers and adversaries.

The adversary may access the camera as an ordinary user and may observe or request signatures on legitimately captured images. However, we assume that the camera's signing key is securely stored and cannot be extracted. This is the standard trust assumption in C2PA-style authenticated capture: the camera is trusted to sign what it captures, and the cryptographic task is to preserve this authenticity through later transformations rather than to secure the signer against full device compromise.

The goal of our system is to prove that a received JPEG image was obtained by applying a claimed compression procedure, with declared JPEG parameters, to an authorized original image that was captured and signed by the camera. The Verifier should therefore be convinced of two facts: the original image was authenticated by the camera, and the public JPEG image is a valid compression of that authenticated image under the claimed parameters.

4.1 Out-of-scope Attacks

Secret-key extraction. If an adversary extracts the camera's signing key, it can sign arbitrary images and produce valid provenance records for them. This breaks the fundamental trust assumption of authenticated capture and cannot be prevented by our proof system. Secure hardware is therefore often adopted to protect signing keys; for example, Leica uses a special chipset [36], while Google relies on trusted execution environments (TEEs) [37] [38]. Unfortunately, such mechanisms remain vulnerable to attack: secure chips have been targeted by a range of attacks [39, 40], and TEEs are subject to known vulnerabilities [41, 42].

Picture-of-picture attacks. Our system guarantees that the authenticated source image was captured and signed by the trusted camera. It does not attempt to reason about the physical scene in front of the sensor. For example, an adversary may display an image on a screen and photograph that screen with the trusted camera. The resulting image is genuinely captured by the camera and therefore falls within the intended authentication semantics. Detecting such picture-of-picture attacks is a separate forensic or sensor-authentication problem and is orthogonal to our cryptographic protocol.

Semantic changes under allowed compression. Our system proves correctness of the claimed compression operation. It does not guarantee that compression preserves every possible semantic interpretation of the image. For example, aggressive compression parameters may remove visually subtle details, and different viewers or downstream machine-learning systems may interpret the compressed image differently. Such effects are inherent to lossy compression. Our guarantee is therefore operational rather than semantic: the public JPEG is proven to be the result of applying the declared JPEG compression procedure to an authenticated source image.

5 SPEG-Poseidon

5.1 Problem Formulation

Recall that authenticating a public JPEG image requires establishing both that the original image is properly signed and that the public JPEG image is obtained by compressing that original image.

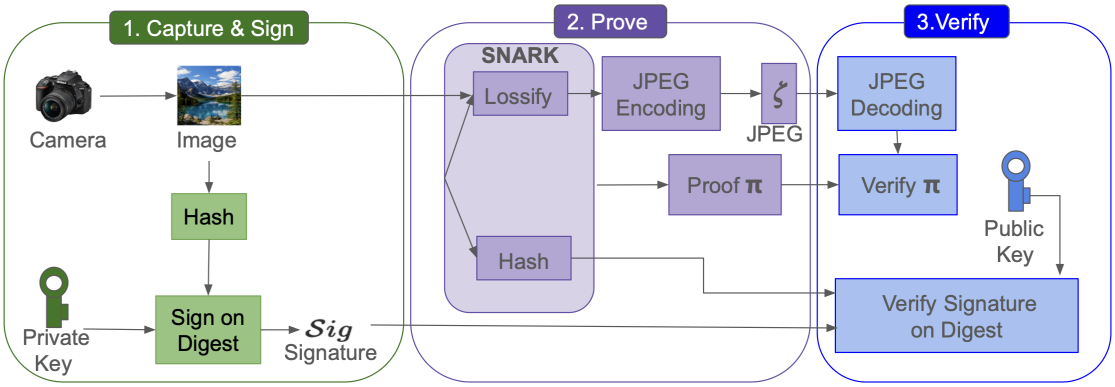


Figure 5: End-to-end workflow of SPEG. The signer authenticates the source image once, the Prover compresses the image and generates a proof, and the Verifier decodes the JPEG and checks the proof against the recovered coefficients.

Since the verifier can check the signature on the public digest outside the proof system, it suffices for the proof system to prove that the digest is correctly computed from the original image and that the public JPEG image is correctly obtained by applying JPEG compression to the same original image.

The SPEG-Poseidon proof system is an interaction between the editor, who acts as the Prover, and the publisher, who acts as the Verifier. The Prover sends the compressed JPEG image, digest, signature, and succinct zero-knowledge proof to the Verifier. The secret witness is the original image X. The public statement includes the JPEG parameters par, the compressed image ζ, and the digest h. We use Enc and Dec to denote JPEG Encoding and JPEG Decoding.

Formally, the JPEG compression proof is defined over the following instance-witness relation:

$$\mathcal{R}_{\mathsf{JPEG}} = \{((\mathsf{h}, \mathsf{par}, \zeta); X) : \quad \mathsf{Enc}(\mathsf{Lossify}_{\mathsf{par}}(X)) = \zeta, \mathsf{H}(X) = \mathsf{h}\}. \tag{1}$$

Leverage Verifier for JPEG Encoding. Recall that JPEG Encoding is lossless, which is a invertible process encoding the information into a shorter bitstream, and the Verifier can recover the information by decoding the JPEG object i.e. get $\mathsf{Dec}(\zeta)$ on their own.

Formally, the instance-witness relation becomes:

$$\mathcal{R}_{\mathsf{SPEG}} = \{((\mathsf{h}, \mathsf{par}, \mathsf{Dec}(\zeta)); X) : \quad \mathsf{Lossify}_{\mathsf{par}}(X) = \mathsf{Dec}(\zeta), \mathsf{H}(X) = \mathsf{h}\}. \tag{2}$$

Theorem 1. $\forall((\mathsf{h}, \mathsf{par}, \zeta, Z = \mathsf{Dec}(\zeta)); X),\ ((\mathsf{h}, \mathsf{par}, \zeta); X) \in \mathcal{R}_{\mathsf{JPEG}} \Leftrightarrow ((\mathsf{h}, \mathsf{par}, Z); X) \in \mathcal{R}_{\mathsf{SPEG}}$.

Proof. Given the $\mathsf{Enc}/\mathsf{Dec}$ process is invertible, the Theorem follows immediately from the definition. □

Thus, we can safely offload JPEG Encoding stage to the Verifier's verification part. Concretely, the Prover outputs the compressed JPEG object ζ, the Verifier performs JPEG Decoding outside the circuit and recovers quantized coefficients, which then serves as public input to the proof system.

Notice that this design does not impose extra overhead on the Verifier. In any realistic workflow, the Verifier, or more generally the downstream consumer such as a publisher, must decode ζ anyway in order to render the image. SPEG therefore aligns the proof system with the Verifier's existing workflow, and we reuse the decoding step that is already required in practice. In short, we exploit

the Verifier's natural decoding workflow and thereby avoid proving JPEG Encoding inside the circuit.

We discuss the proof as two parts:

- **Proof of Compression.** A proof that the original image X, after the Lossify part of JPEG compression with parameters par, transforms to the public coefficient representation recovered from ζ.
- **Proof of Hash.** A proof that the original image X, under the specified hash function, hashes to the digest h. This is the more expensive part, and SPEG therefore supports two approaches for it.

5.2 Proof of Compression

Mapping for JPEG Lossify. The arithmetic used in JPEG transformation involves real numbers. A straightforward implementation, and the one adopted by several existing systems, is to prove floating-point arithmetic by introducing quotient and remainder constraints and then range-checking the remainder. Although conceptually simple, this approach is expensive: range checks become one of the dominant contributors to Prover time.

To avoid this cost, SPEG does not prove floating-point arithmetic directly. Instead, it follows the general quantization principle used in zero-knowledge systems for machine learning. A real value a is represented by an integer q together with a scale factor L and zero point Z, so that $a = L(q - Z)$. We choose $L = 2^{-E}$ and track the scale throughout the computation. For addition, if $a_1 = L(q_1 - Z_1)$ and $a_2 = L(q_2 - Z_2)$, then $a_1 + a_2 = L(q_1 + q_2 - Z_1 - Z_2)$. For multiplication, we represent the product by $L_3(q_3 - Z_3) = L_1(q_1 - Z_1) \cdot L_2(q_2 - Z_2)$. Because image pixels are 8-bit values and the computation trace is fixed in advance, the scale parameters can be chosen so that no field overflow occurs.

However, applying this mapping strategy directly, as in prior zero-knowledge systems for machine learning, is insufficient for JPEG. Compression additionally requires that the coefficient matrix after Trans be sparse enough to compress effectively. A mapping strategy that merely keeps small non-zero remainders will often fail to preserve the sparsity pattern that JPEG relies on. Thus, a naive integer mapping does not preserve the compression behavior of the original floating-point computation.

SPEG resolves this problem by modifying the quantization stage without hurting the compression effect. Standard JPEG quantization divides each DCT coefficient by the corresponding entry of a quantization matrix Q, which would normally require range checks to enforce correct division and rounding. Instead, SPEG uses an approximate inverse matrix $R \approx 1/Q$ and maps this inverse matrix into the integer domain. High-frequency positions are then zeroed out according to the target compression ratio.

Moreover, SPEG removes the final bit-decomposition-based rescaling step by again exploiting the separation between the arithmetic circuit and the JPEG Decoding workflow. Since entropy encoding and decoding are handled outside the circuit, the system can use a customized mapping-back procedure during decoding. The resulting circuit is lightweight, integer-based, and free of both range checks and bit-decomposition-heavy rescaling.

This lightweight transformation circuit is also what later enables the fast commitment-based mode in Section 6. In SPEG-Poseidon itself, the same mapping substantially reduces Prover cost while preserving the exact public coefficient tensor recovered from the JPEG decoder.

With this mapping in place, the compression circuit is described at Circuit 3.

Circuit 3 $\mathcal{F}^{\text{Lossify}}(\mathbf{A}; s, D, R_Y, R_C)$

Input: $\mathbf{A} = (\mathbf{A}^Y, \mathbf{A}^{Cb}, \mathbf{A}^{Cr})$, subsampling mode s, DCT matrix D, mapped quantization tables R_Y, R_C
Output: quantized coefficient blocks $(\mathbf{C}^Y, \mathbf{C}^{Cb}, \mathbf{C}^{Cr})$
$\mathbf{R}^Y \leftarrow R_Y$, $\mathbf{R}^{Cb} \leftarrow R_C$, $\mathbf{R}^{Cr} \leftarrow R_C$
if s=='1' **then** ▷ use '1' to denote 'subsample'
    subsample $\mathbf{A}^{Cb}$ and $\mathbf{A}^{Cr}$
end if
for $c \in \{Y, Cb, Cr\}$ **do**
    $\mathbf{B}^c \leftarrow D \cdot \mathbf{A}^c \cdot D^\top$
    $\mathbf{C}^c \leftarrow \mathbf{B}^c \odot \mathbf{R}^c$
end for
return $(\mathbf{C}^Y, \mathbf{C}^{Cb}, \mathbf{C}^{Cr})$

Circuit 4 $\mathcal{F}^{\text{Hash}}(\mathbf{X})$

Input: image chunk $\mathbf{X}$
Output: packed Poseidon digest $\mathsf{Poseidon}(\mathbf{h})$
for $k \in [0, 128)$ **do**
    for $i \in [0, 10)$ **do**
        $\ell \leftarrow 10k + i$
        RangeCheck$(\mathbf{X}^\ell_Y, 255)$
        RangeCheck$(\mathbf{X}^\ell_{Cb}, 255)$
        RangeCheck$(\mathbf{X}^\ell_{Cr}, 255)$
        $t_i \leftarrow \mathbf{X}^\ell_Y + 2^8 \mathbf{X}^\ell_{Cb} + 2^{16} \mathbf{X}^\ell_{Cr}$
    end for
    $h_k \leftarrow \sum_{i=0}^{9} 2^{24i} t_i$
end for
return $\mathsf{Poseidon}(\mathbf{h})$

5.3 Proof of Hash

As discussed above, even a circuit-friendly hash such as Poseidon remains costly on very large images. Following the same high-level strategy as VerITAS, which uses a linear dimension-reduction layer before hashing, we reduce the dimensionality of the image first and then apply Poseidon. Concretely, SPEG-Poseidon uses a Pack-Poseidon hash.

Packing as dimension reductor. Packing works as follows. We operate over a 254-bit field. Each channel value lies in $[0, 255]$ and therefore occupies 8 bits. One RGB or YCbCr pixel therefore occupies 24 bits. It follows that ten pixels fit into one 240-bit field element. On valid 8-bit inputs, this packing is a bijection. Combined with the collision-resistance of Poseidon, this means that the Pack-Poseidon construction remains collision resistant on valid images.

With packing, the hash circuit is described at Circuit 4. RangeCheck denotes a standard bit-decomposition range check, which represents each channel value as eight Boolean bits and enforces that their binary reconstruction equals the original value. In our implementation we use only sponge-2 and sponge-8 Poseidon instances for efficiency.

Notice that this Pack-Poseidon hash is collision-resistant only when the corresponding range checks are enforced; omitting these range checks enables the attack described in 5.5.

Minimizing commitment cost. Our key insight is to prioritize witness weight. A viewpoint largely absent from prior works is that commitment cost depends on the total number of bits car-

Circuit 6 $\mathcal{F}^{\text{Main}}(h_i, (X_{bi+j})_{j=0}^{b-1})$

Input: running state h_i, witness batch $(X_{bi+j})_{j=0}^{b-1}$, public coefficients $(Z_{bi+j})_{j=0}^{b-1}$, s, D, R_Y, R_C
Output: next running state h_{i+1}
for $j \in [0, b-1]$ **do**
    $\mathbf{C} \leftarrow \mathcal{F}^{\text{Lossify}}(X_{bi+j}; s, D, R_Y, R_C)$
    for $(p, q) \in [0, 8)^2$ **do**
        enforce $\mathbf{C}[p, q] = Z_{bi+j}[p, q]$
    end for
end for
for $k \in [0, 8)$ **do**
    $r_k \leftarrow \mathcal{F}^{\text{Hash}}((X_{bi+j}[k, \cdot])_{j=0}^{b-1})$
end for
$h_{i+1} \leftarrow \mathsf{Poseidon}([h_i, \mathsf{Poseidon}((r_k)_{k=0}^{7})])$
return h_{i+1}

ried by the witness elements instead of the length of the witness vector. This distinction matters in folding-based proof systems. Existing proof systems for image operations typically adopt Nova because it has mature implementations and strong practical optimizations. However, Nova introduces an additional commitment to a cross-term vector. This vector has the same length as the witness, but its entries are full-field elements. In image-transmission settings, this cross-term can therefore dominate Prover cost. SPEG instead adopts the NeutronNova folding scheme [31], which folds a zero relation and avoids committing to the cross-term vector, and instantiates the decider layer with Spartan [43]. This largely reduces commitment cost and improves Prover efficiency.

The same witness-weight perspective also changes our choice of range-check mechanism. Existing systems often use lookup arguments for range checks because they are efficient in many generic settings. For example, Eva [14] integrates LogUp [44] with Nova using the Protostar [45] design. With μ lookup queries and table size ν, Eva only requires $\mu + 2\nu$ additional commitments, but these commitments are to full-field elements. Bit decomposition requires 8μ commitments for 8-bit values, but each commitment is only to a single-bit value. In our image-authentication setting, this latter choice has much lower witness weight. Since μ is comparable to the number of image channels, bit decomposition substantially reduces Prover time.

5.4 Putting Everything Together

With these two proof components in place, we encode the step circuit as a rank-1 constraint system (R1CS) and construct a proof system for JPEG compression. The Prover computes the public JPEG object and proves both the correctness of the hash computation and the correctness of JPEG Lossify. The Verifier decodes the JPEG object, checks the proof, and then follows the C2PA workflow to verify the signature on the hash.

The full protocol is shown at Protocol 5.

5.5 Security Gap of VIMz

VIMz [13] uses the same idea to pack several pixels into a field element, but such Pack-Poseidon hash is secure i.e. collision-resistant only when the input channels are constrained to lie in $[0, 255]$.

Protocol 5 SPEG-POSEIDON

Function $\mathcal{K}_\Sigma(1^\lambda)$
Input: security parameter 1^λ
Output: signing key pair $(\mathsf{sk}_\Sigma, \mathsf{vk}_\Sigma)$
return $(\mathsf{sk}_\Sigma, \mathsf{vk}_\Sigma) \leftarrow \mathsf{Sig}.\mathcal{K}(1^\lambda)$ ▷ setup signing keys
Function $\mathcal{K}_\Pi(1^\lambda)$
Input: security parameter 1^λ
Output: public proving and verification keys $(\mathsf{pk}_\Pi, \mathsf{vk}_\Pi)$
$\mathsf{pp} \leftarrow \mathsf{NeutronNovaIVC.G}(1^\lambda)$ ▷ public parameters for folding
$(\mathsf{pk}_\Phi, \mathsf{vk}_\Phi) \leftarrow \mathsf{NeutronNovaIVC}.\mathcal{K}(\mathsf{pp}, \mathcal{F}^{\mathsf{Main}})$
$(\mathsf{pk}_{\mathsf{spartan}}, \mathsf{vk}_{\mathsf{spartan}}) \leftarrow \mathsf{Spartan}.\mathcal{K}(1^\lambda, \mathcal{F}^{\mathsf{Decider}})$
return $\mathsf{pk}_\Pi := (\mathsf{pk}_\Phi, \mathsf{pk}_{\mathsf{spartan}})$, $\mathsf{vk}_\Pi := (\mathsf{vk}_\Phi, \mathsf{vk}_{\mathsf{spartan}})$ ▷ proving and verification keys
Function $\mathcal{S}_\Sigma(\mathsf{sk}_\Sigma, \mathbf{X}, \mathsf{meta})$
Input: signing key sk_Σ, source image $\mathbf{X}$, metadata meta
Output: digest h and signature σ
$h \leftarrow \mathsf{Hash}(\mathbf{X})$
$\sigma \leftarrow \mathsf{Sig}.\mathcal{S}(\mathsf{sk}_\Sigma, (h, \mathsf{meta}))$
return (h, σ) ▷ sign authenticated digest and metadata
Function $\mathcal{P}(\mathsf{pk}_\Pi, \mathbf{X}, h, \mathsf{par})$
Input: proving key pk_Π, source image $\mathbf{X}$, signed digest h, JPEG parameters par
Output: compressed JPEG ζ and proof π
$\mathbf{Z} \leftarrow \mathsf{Lossify}_{\mathsf{par}}(\mathbf{X})$ ▷ compute JPEG coefficients
$\zeta \leftarrow \mathsf{Encode}(\mathbf{Z}, \mathsf{par})$
Partition $\mathbf{X}$ into private batches $(X_j)_{j=0}^{k-1}$
Partition $\mathbf{Z}$ into public batches $(Z_j)_{j=0}^{k-1}$
$h_0 \leftarrow 0$ ▷ initialize incremental state
Initialize $\pi_0 := ((\mathbb{U}_\perp, \mathbb{W}_\perp), (\mathfrak{u}_0, \mathfrak{w}_0), r_0)$
for $j \in [0, k)$ **do**
    $h_{j+1} \leftarrow \mathcal{F}^{\mathsf{Main}}(h_j, X_j; Z_j)$
    $\pi_{j+1} \leftarrow \mathsf{NeutronNovaIVC.P}(\mathsf{pk}_\Phi, (j, h_j, h_{j+1}, Z_j), X_j, \pi_j)$
end for
Enforce $h_k = h$
Parse $\big((\mathbb{U}_k, \mathbb{W}_k), (\mathfrak{u}_k, \mathfrak{w}_k), r_k\big) := \pi_k$
$\big(\pi_k^{\mathsf{fold}}, (\mathbb{U}_{k+1}, \mathbb{W}_{k+1})\big) \leftarrow \mathsf{NeutronNovaFS.P}(\mathsf{pk}_\Phi, (\mathbb{U}_k, \mathbb{W}_k), (\mathfrak{u}_k, \mathfrak{w}_k))$
$\mathbf{x} := (k, h_0, h, (Z_j)_{j=0}^{k-1}, r_k, \mathfrak{u}_k, \mathbb{U}_k, \pi_k^{\mathsf{fold}})$
$\mathbf{w} := \mathbb{W}_{k+1}$
$\pi_{\mathsf{spartan}} \leftarrow \mathsf{Spartan.P}(\mathsf{pk}_{\mathsf{spartan}}, \mathbf{x}, \mathbf{w})$ ▷ final decider proof
return $\zeta,\ h,\ \pi := (\pi_{\mathsf{spartan}}, \mathbb{U}_k, \mathfrak{u}_k, r_k, \pi_k^{\mathsf{fold}})$
Function $\mathcal{V}(\mathsf{vk}_\Pi, \mathsf{vk}_\Sigma, \zeta, h, \sigma, \mathsf{meta}, \mathsf{par}, \pi)$
Input: $\mathsf{vk}_\Pi, \mathsf{vk}_\Sigma, \zeta, h, \sigma, \mathsf{meta}, \mathsf{par}, \pi$
Output: verification bit
if $\mathsf{Sig}.\mathcal{V}(\mathsf{vk}_\Sigma, (h, \mathsf{meta}), \sigma) = 0$ **then**
    return 0
end if
Parse $(\pi_{\mathsf{spartan}}, \mathbb{U}_k, \mathfrak{u}_k, r_k, \pi_k^{\mathsf{fold}}) := \pi$
Decode ζ to obtain $\mathbf{Z}^V := \mathsf{Decode}(\zeta, \mathsf{par})$
Partition $\mathbf{Z}^V$ into public batches $(Z_j^V)_{j=0}^{k-1}$
$h_0 \leftarrow 0$
$\mathbf{x}^V := (k, h_0, h, (Z_j^V)_{j=0}^{k-1}, r_k, \mathfrak{u}_k, \mathbb{U}_k, \pi_k^{\mathsf{fold}})$
return $\mathsf{Spartan.V}(\mathsf{vk}_{\mathsf{spartan}}, \mathbf{x}^V, \pi_{\mathsf{spartan}})$



(a) Signed image (b) Adversarial image

Figure 6: Illustration of the attack. The adversarially generated image collides with the signed original image, despite having different semantic content.

In other words, range checks on the input channels are necessary for security. VIMz does not perform these range checks, and this omission opens the door to a concrete attack.

The adversary starts from an honestly signed image X and computes its packed field representation. It then searches for a file X' whose channel values may exceed the legal $(0, 255]$ range, subject to two constraints: $\mathsf{Packing}(X') = \mathsf{Packing}(X)$, and the JPEG-compressed output of X' maps back into the $(0, 255]$ range. This second constraint is feasible as JPEG compression is lossy. The search can be implemented as a constrained optimization problem. Such an X' can be found with different semantic meaning, X and X' produce the same packed field elements and therefore the same digest, breaking collision-resistant. Figure 6 gives a concrete example.

5.6 Extension to Standard Hash

We further support an additional mode called SPEG-SHA256.

Some deployments may require strict compatibility with an existing byte-level hash such as SHA-256. In this variant, the signer authenticates the canonical byte representation directly and the proof includes a SHA-256 gadget. However, SHA-256 is inefficient to represent as an arithmetic circuit because it relies on non-linear bitwise operations. As a result, SPEG-SHA256 provides C2PA compatibility at a substantially higher proving cost.

6 SPEG-Commit

SPEG-Poseidon is designed for C2PA-style deployments, where the Verifier expects a standard hash as digest. We next describe our alternate approach for proving the binding, namely SPEG-Commit. SPEG-Commit is appropriate when compatibility of C2PA with a standard hash is not required, Prover time is the primary constraint, and the signer can afford more computation power. In this mode, the commitment binds the source image, so the proof only needs to certify the JPEG arithmetic relation.

We write the original image X row by row and channel by channel as a vector $\vec{x}$ of length $m = 3 \cdot \#\text{pixels}$, interpolate a univariate polynomial $W = \mathsf{poly}(\vec{x})$ of degree $m-1$ with $W(\omega^i) = \vec{x}_i$ for $i = 1, ..., m$, and commit to W with a polynomial commitment scheme to obtain a commitment com.

The key observation is that a polynomial commitment is binding, thus can serve as hash. The Verifier therefore wants to check that com is indeed a commitment to the polynomial derived from the original image, formally, we aim to construct a zk-SNARK for instance-witness relation:

$$\begin{aligned}\mathcal{R}_{\mathsf{commit}} = \{((\mathsf{pp}, \mathsf{com}, \mathsf{par}, \mathsf{Dec}(\zeta)); (W, r)) : \\ \mathsf{Lossify}_{\mathsf{par}}(X) = \mathsf{Dec}(\zeta), \mathsf{PCS.commit}(\mathsf{pp}, W, r) = \mathsf{com}\}.\end{aligned} \tag{3}$$

A strawman approach for such relation is to have the circuit for both proof of compression and proof of commitment; however, proving the correctness of polynomial commitment computation is far more complex than proving correctness of hash computation. Inspired by VerITAS [12], the polynomial commitment can be implicitly verified:

6.1 The Proof System

VerITAS uses a PLONKish design for range checks. In our setting, the mapping design already removes the floating-point range checks required by the remainder-based mappings used in prior systems. The remaining input range checks serve a different purpose: VerITAS range-checks inputs for lattice-hash security, while SPEG-Poseidon range-checks packed image channels to preserve the binding of packing. In SPEG-Commit, the binding primitive is a polynomial commitment rather than a hash over packed inputs. Given an enrolled source image, the binding property prevents the Prover from finding a different image that opens to the same commitment com. Therefore, the proof does not need to range-check the input image pixels inside the circuit. The valid 8-bit input range is enforced implicitly at capture or enrollment time, since images produced by the camera already lie in the valid 8-bit range.

Moreover, recall that the correctness of the polynomial commitment is checked implicitly through commitment openings. Therefore, the proof system only needs to provide the proof of compression. This range-check-free circuit enables the use of GKR-style protocols with optimal Prover time. Combined with the efficient circuit design, our protocol achieves very fast proving with optimal Prover time. Appendix A.3 reviews the relevant background on GKR.

We follow the framework of zkCNN [32], which uses argument system of vSQL [46] based on generalized GKR. The Prover evaluates the blockwise JPEG transformation and proves the resulting layered arithmetic computation representing Compression with a GKR-style backend. The correctness of commitment can be verified by the polynomial commitment opening instead of inside the circuit.

Concretely, the Prover commits to the polynomial that interpolates the original image X and sends the commitment to the Verifier. The Prover and Verifier then run sumchecks for the matrix-multiplication and Hadamard-product and generalized GKR protocol. These protocols reduce correctness of the JPEG computation to a single evaluation claim on the multilinear extension of the input, which the Verifier checks by opening the received polynomial commitment.

The full protocol is given in Algorithm 7 in Appendix B.

6.2 KZH Polynomial Commitment Scheme

One caveat of a direct GKR-based design is proof size. In the original zkCNN setting, the final proof uses Hyrax, and the proof size grows with the image dimension. That growth is not suitable for our setting, since JPEG compression is only useful when the proof remains small enough that it does not erase the transmission benefit of compression. We address this issue by replacing Hyrax with the KZH [18] polynomial commitment scheme.

KZH is a novel polynomial commitment scheme. We use it as the commitment layer in the fast mode and rely on its small proof size and efficient openings.

As mentioned in KZH, when the SRS is initialized with the popular powers of τ i.e. when $G^{(\vec{i})}$ is initialized with the $\tau^{i \times m} \times g$ instead, and $\tau^{(\vec{j})}$ is equal to τ^j, the KZH commitment serves as both a univariate KZG commitment and as a multilinear KZH commitment. We call this KZH with KZG SRS. An important practical advantage of KZH arises when its SRS is instantiated using the widely deployed powers-of-(τ) structure. Since powers-of-(τ) ceremonies have already been performed for many KZG-based systems, compatible reference strings are already available in existing deployments. KZH can therefore reuse an existing KZG SRS rather than requiring a new trusted setup dedicated specifically to KZH. This substantially lowers the deployment barrier, allowing users to instantiate KZH directly from an existing setup instead of conducting an additional setup ceremony solely for the KZH commitment scheme. We give the formal definition and the security proof of this modification of KZH at Appendix C.

7 Security Analysis of SPEG

We formally prove the security of both SPEG modes, with standard SNARK security model [47].

Theorem 2. *Assume completeness, knowledge soundness, and zero knowledge property of NeutronNova IVC and Spartan, an unforgeable signature scheme, and collision-resistance of Poseidon. Protocol 5* SPEG-POSEIDON *is a complete, knowledge-sound, and zero-knowledge proof system for C2PA with verifiable compression of images.*

Proof. We prove Theorem 2 with:
Completeness follows immediately by definition.
Knowledge Soundness. Consider any adversary that outputs an accepting tuple $(\zeta, h, \sigma, \mathsf{meta}, \mathsf{par}, \pi)$. The Verifier's acceptance implies a valid signature on (h, meta) and an accepting Spartan proof for the decider instance built from h and $Z = \mathsf{Decode}(\zeta, \mathsf{par})$. Spartan knowledge soundness extracts a witness for the decider circuit, and NeutronNova IVC knowledge soundness opens this witness into the per-step image batches used in the folded computation. Combining the batches gives an image X such that $\mathsf{Lossify}_{\mathsf{par}}(X) = Z$ and $\mathsf{Hash}(X) = h$.
Zero Knowledge. The simulator first performs the public checks that the Verifier performs: it verifies the signature and decodes ζ to obtain Z. It then uses the NeutronNova simulator for the folded transcript and the Spartan simulator for the decider proof. A standard hybrid argument replaces the real folding transcript and the real Spartan proof with their simulated counterparts, and each replacement is indistinguishable by the zero-knowledge property of the corresponding system. Since signature verification and JPEG Decoding are public deterministic checks, they do not reveal the witness image. □

Theorem 3. *Protocol 7* SPEG-COMMIT *is complete, knowledge sound, and zero knowledge.*

Proof. The GKR protocol is proved complete and knowledge sound, and made zero knowledge using similar arguments as zkCNN [32]. We then prove security of the polynomial commitment KZH with KZG SRS in Appendix C. □

8 Evaluation

This section answers the following three questions:

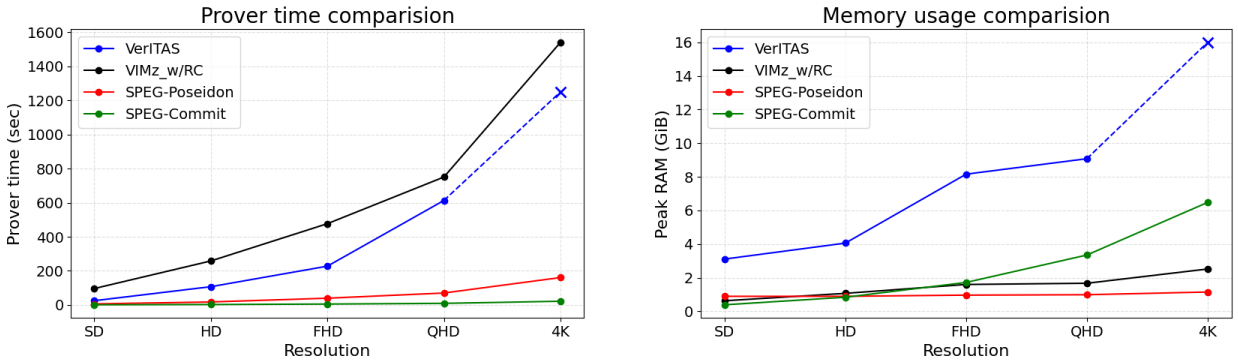


Figure 7: Performance comparison with existing proof systems across resolutions.

- How does SPEG compare with related proof systems?
- How do the two modes, SPEG-Poseidon and SPEG-Commit, perform in practice?
- What are the main sources of cost in the protocol?

8.1 Experimental Setup

We implemented SPEG[3] in Rust over the Pasta cycle (Pallas and Vesta) [48] and ran all experiments on a MacBook Pro with an Apple M1 Pro chip, 8 CPU threads, 16 GB of memory. Unless noted otherwise, numbers are averages over five runs. We evaluate resolutions from SD to 4K and use standard JPEG parameter sets, including common chroma-subsampling configurations and quantization tables.

8.2 Comparison with Existing Proof Systems

Figure 7 compare SPEG against VerITAS and VIMz on the shared task of proving authenticity-related computation over large images. We do not include Eva in this comparison because it targets a different video-specific pipeline and relies on CUDA acceleration.

The main result is that the gap widens with image size. At 4K, SPEG-Poseidon requires roughly 165 s of Prover time, compared with about 1250 s for VerITAS and 1540 s for VIMz. The same trend appears in memory usage: SPEG-Poseidon stays close to 1 GiB across the measured resolutions, whereas VerITAS runs out of memory (more than 16 GiB) at 4K and VIMz grows to about 2.5 GiB. This behavior matches the design intuition of controlling witness weight and avoiding cross-term commitments.

8.3 Comparison across SPEG Modes

Table 3 compares the three SPEG modes and highlights their central trade-off: from SPEG-SHA256 to SPEG-Poseidon to SPEG-Commit, prover time is progressively reduced by offloading more computation to the signer. SPEG-SHA256 most closely follows the standard C2PA workflow by using circuit-unfriendly SHA-256; the signer only computes a SHA-256 digest, but the prover must prove the corresponding hash computation inside the circuit, resulting in the highest proving cost. SPEG-Poseidon replaces SHA-256 with our Pack-Poseidon hash, deviating

[3]https://github.com/hanbunko223/VerImgCmp.git

Table 3: Comparison across SPEG modes at 1M pixels i.e. HD resolution (1280×720)

Mode	Prover Time (s)	RAM (GiB)	Proof Size (kB)	Verifier Time (s)
SPEG-SHA256	566.44	7.75	37.62	11.80
SPEG-Poseidon	20.95	1.06	28.5	0.53
SPEG-Commit	0.82	0.84	15.17	0.40

Table 4: Performance metrics of SPEG-Commit for different image resolutions

Image Size (Pixels)	Prover Time (s)	RAM (GiB)	Proof Size (kB)	Verifier Time (s)
SD (300K)	0.33	0.39	14.30	0.17
HD (1M)	0.82	0.84	15.17	0.40
FHD (2M)	1.68	1.72	17.05	0.80
QHD (4M)	2.89	3.35	18.92	1.48
4K (10M)	6.16	6.48	20.80	2.95

from the standard C2PA hash workflow but substantially simplifying the circuit and improving prover performance; in this mode, the signer computes the Pack-Poseidon hash instead. Finally, SPEG-Commit further offloads computation to the signer by replacing the hash with a polynomial commitment that serves the same binding role. This yields the lowest prover cost among the three modes, at the expense of requiring the signer to compute the more expensive polynomial commitment.

Table 4 reports the end-to-end cost of the fast mode SPEG-Commit across image resolutions. The main observation is that Prover time and memory grow with image size, but the proof remains small enough for practical transmission. Proof size is only 20.80 kB for 4K image, while Verifier time remains modest relative to the cost of proving.

Note that SPEG-Commit takes only 0.62 µs/px on a personal device with 8 threads, which is even faster than 2.6 µs/px that Eva [14] reports, the latest proof system, using 24 CPU threads and GPU acceleration.

8.4 Ablation Study

We focus on how each technique improves Prover performance in the main mode SPEG-Poseidon. Our starting point is a direct implementation that writes the circuit in Circom and uses the VIMz proving code. This baseline requires more than 600 s of Prover time, and the ablation below isolates the contribution of each optimization.

Mapping. The first major reduction comes from our Mapping for floating-point arithmetic. In the baseline, range checks for floating-point-style division consume a large fraction of the proving cost. Replacing this path with the mapped integer formulation reduces Prover time from more than 600 s to 394.8 s. This is a reduction of at least 34%.

NeutronNova. Replacing the previous Nova-based design with NeutronNova removes the cross-term commitment, which is particularly expensive for large image witnesses. This change reduces Prover time from 394.8 s to 308 s, a further 22.0% improvement.

Packing. Earlier versions used bit operations to realize the packing step, introduced unnecessary cost in the witness and arithmetic path. Rewriting packing with direct multiplications reduces Prover time from 308 s to 140 s, leads to a further 54.5% reduction.

Table 5: Per-step R1CS constraint breakdown in SPEG-Poseidon (10240 pixels per step)

	$\mathcal{F}^{\text{main}}$			Augmentation	
Component	range checks	Lossify	Hash	fold into running instance	compute output
number	276480	92160	79955	4380	1114
percentage	60.0%	20.3%	17.6%	1.0%	0.2%
total			454089		

8.5 Breakdown of Constraint

Table 5 shows the per-step R1CS constraint breakdown in SPEG-Poseidon.

The breakdown of R1CS constraints in SPEG-Poseidon follows the same two-part structure as the proof itself. The hash component contributes the largest share of constraints, which is consistent with the discussion in Section 5: even with packing, Poseidon-based authentication remains the dominant cost at realistic image sizes. The JPEG component is smaller because JPEG Encoding is kept outside the circuit and the mapping design removes floating-point range checks.

9 Extensions and Conclusion

We presented SPEG with two incomparable modes. Our key innovations are an efficient JPEG circuit design, which yields the first practical proof system for verifiable image compression, and a GKR-compatible construction that combines with polynomial commitment hash to further produce a fast mode. We also identify a security gap in VIMz [13] and prove security for KZH with a KZG SRS.

Several directions remain open. Sumcheck protocols for approximate computation [49] may provide a cleaner way to handle the mismatch between floating-point arithmetic and arithmetic circuits. Future work may also incorporate advances in folding schemes and polynomial commitment schemes to improve efficiency or obtain post-quantum security.

References

[1] *Government leads global fight against deepfake threats*, link, 2026.

[2] E. Strickland, *Facebook AI Launches Its Deepfake Detection Challenge*, link, 2019.

[3] *C2PA Specifications*, link, 2026.

[4] D. Boneh and V. Shoup, "A graduate course in applied cryptography," 2023. [Online]. Available: https://toc.cryptobook.us

[5] Leica, *Content Credentials*, link, 2026.

[6] Nikon, *Nikon Authenticity Service*, link, 2026.

[7] Sony, *Sony Electronics Launches Industry First(i) Video-Compatible Camera Authenticity Solution for News Organizations and Broadcasters*, link, 2025.

[8] Google, *Content Credentials*, link, 2026.

[9] OpenAI, *Advancing content provenance for a safer, more transparent AI ecosystem*, link, 2026.

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

[11] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun, "Zk-img: Attested images via zero-knowledge proofs to fight disinformation," 2022. [Online]. Available: https://arxiv.org/abs/2211.04775

[12] T. Datta, B. Chen, and D. Boneh, "VerITAS: Verifying image transformations at scale," Cryptology ePrint Archive, Paper 2024/1066, 2024. [Online]. Available: https://eprint.iacr.org/2024/1066

[13] S. Dziembowski, S. Ebrahimi, and P. Hassanizadeh, "VIMz: Private proofs of image manipulation using folding-based zkSNARKs," Cryptology ePrint Archive, Paper 2024/1063, 2024. [Online]. Available: https://eprint.iacr.org/2024/1063

[14] C. Zhang, X. Yang, D. Oswald, M. Ryan, and P. Jovanovic, "Eva: Efficient privacy-preserving proof of authenticity for lossily encoded videos," Cryptology ePrint Archive, Paper 2024/1436, 2024. [Online]. Available: https://eprint.iacr.org/2024/1436

[15] K. H. Talukder and K. Harada, "Haar wavelet based approach for image compression and quality assessment of compressed image," 2010. [Online]. Available: https://arxiv.org/abs/1010.4084

[16] B. Dornauer and M. Felderer, "Web image formats: Assessment of their real-world-usage and performance across popular web browsers," 2023. [Online]. Available: https://arxiv.org/abs/2310.00788

[17] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, "Delegating computation: Interactive proofs for muggles," *J. ACM*, vol. 62, no. 4, Sep. 2015. [Online]. Available: https://doi.org/10.1145/2699436

[18] G. Kadianakis, A. Zapico, H. Hafezi, and B. Bünz, "KZH-fold: Accountable voting from sublinear accumulation," Cryptology ePrint Archive, Paper 2025/144, 2025. [Online]. Available: https://eprint.iacr.org/2025/144

[19] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 177–194.

[20] Zcash, "The halo2 zero-knowledge proving system," 2022. [Online]. Available: https://github.com/zcash/halo2.git

[21] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," Cryptology ePrint Archive, Paper 2017/1066, 2017. [Online]. Available: https://eprint.iacr.org/2017/1066

[22] P. Zero, "Plonky2, a snark implementation based on techniques from plonk and fri," 2021. [Online]. Available: https://github.com/0xPolygonZero/plonky2.git

[23] B. Chen, B. Bünz, D. Boneh, and Z. Zhang, "HyperPlonk: Plonk with linear-time prover and high-degree custom gates," Cryptology ePrint Archive, Paper 2022/1355, 2022. [Online]. Available: https://eprint.iacr.org/2022/1355

[24] A. Kothapalli, S. Setty, and I. Tzialla, "Nova: Recursive zero-knowledge arguments from folding schemes," Cryptology ePrint Archive, Paper 2021/370, 2021. [Online]. Available: https://eprint.iacr.org/2021/370

[25] L. Grassi, Y. Hao, C. Rechberger, M. Schofnegger, R. Walch, and Q. Wang, "Horst meets fluid-SPN: Griffin for zero-knowledge applications," Cryptology ePrint Archive, Paper 2022/403, 2022. [Online]. Available: https://eprint.iacr.org/2022/403

[26] D. Derler, H. C. Pöhls, K. Samelin, and D. Slamanig, "A general framework for redactable signatures and new constructions," Cryptology ePrint Archive, Paper 2015/1059, 2015. [Online]. Available: https://eprint.iacr.org/2015/1059

[27] D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," Cryptology ePrint Archive, Paper 2011/018, 2011. [Online]. Available: https://eprint.iacr.org/2011/018

[28] S. Gorbunov, V. Vaikuntanathan, and D. Wichs, "Leveled fully homomorphic signatures from standard lattices," Cryptology ePrint Archive, Paper 2014/897, 2014. [Online]. Available: https://eprint.iacr.org/2014/897

[29] M. Christ and S. Gunn, "Pseudorandom error-correcting codes," 2024. [Online]. Available: https://arxiv.org/abs/2402.09370

[30] W. Qu, Y. Sun, X. Liu, T. Lu, Y. Guo, K. Chen, and J. Zhang, "zkgpt: an efficient non-interactive zero-knowledge proof framework for llm inference," in *Proceedings of the 34th USENIX Conference on Security Symposium*, ser. SEC '25. USENIX Association, 2025.

[31] A. Kothapalli and S. Setty, "NeutronNova: Folding everything that reduces to zero-check," Cryptology ePrint Archive, Paper 2024/1606, 2024. [Online]. Available: https://eprint.iacr.org/2024/1606

[32] T. Liu, X. Xie, and Y. Zhang, "zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy," Cryptology ePrint Archive, Paper 2021/673, 2021. [Online]. Available: https://eprint.iacr.org/2021/673

[33] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, "Doubly-efficient zksnarks without trusted setup," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 926–943.

[34] W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*. Springer Science & Business Media, 1992.

[35] N. Bitansky, R. Canetti, A. Chiesa, and E. Tromer, "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 326–349. [Online]. Available: https://doi.org/10.1145/2090236.2090263

[36] Leica, *New: Leica M11-P*, link, 2023.

[37] E. Lynch, *How Pixel and Android are bringing a new level of trust to your images with C2PA Content Credentials*, link, 2025.

[38] Google, *Encrypted zone*, link, 2026.

[39] T. Roche, V. Lomné, C. Mutschler, and L. Imbert, "A side journey to titan," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 231–248. [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/roche

[40] H. N. Jacob, C. Werling, R. Buhren, and J.-P. Seifert, "faultpm: Exposing amd ftpms' deepest secrets," 2023. [Online]. Available: https://arxiv.org/abs/2304.14717

[41] J. Chuang, A. Seto, N. Berrios, S. van Schaik, C. Garman, and D. Genkin, "Tee.fail: Breaking trusted execution environments via ddr5 memory bus interposition," in *47th IEEE Symposium on Security and Privacy (IEEE S&P '26)*. IEEE Computer Society, 2026. [Online]. Available: https://tee.fail

[42] J. De Meulemeester, L. Wilke, D. Oswald, T. Eisenbarth, I. Verbauwhede, and J. Van Bulck, "Badram: Practical memory aliasing attacks on trusted execution environments," in *2025 IEEE Symposium on Security and Privacy (SP)*, 2025, pp. 4117–4135.

[43] S. Setty, "Spartan: Efficient and general-purpose zkSNARKs without trusted setup," Cryptology ePrint Archive, Paper 2019/550, 2019. [Online]. Available: https://eprint.iacr.org/2019/550

[44] U. Haböck, "Multivariate lookups based on logarithmic derivatives," Cryptology ePrint Archive, Paper 2022/1530, 2022. [Online]. Available: https://eprint.iacr.org/2022/1530

[45] B. Bünz and B. Chen, "ProtoStar: Generic efficient accumulation/folding for special sound protocols," Cryptology ePrint Archive, Paper 2023/620, 2023. [Online]. Available: https://eprint.iacr.org/2023/620

[46] Y. Zhang, D. Genkin, J. Katz, D. Papadopoulos, and C. Papamanthou, "vSQL: Verifying arbitrary SQL queries over dynamic outsourced databases," Cryptology ePrint Archive, Paper 2017/1145, 2017. [Online]. Available: https://eprint.iacr.org/2017/1145

[47] B. Bünz, B. Fisch, and A. Szepieniec, "Transparent SNARKs from DARK compilers," Cryptology ePrint Archive, Paper 2019/1229, 2019. [Online]. Available: https://eprint.iacr.org/2019/1229

[48] "The pasta curves for halo 2 and beyond," 2020. [Online]. Available: https://electriccoin.co/blog/the-pasta-curves-for-halo-2-and-beyond/

[49] D. Bitan, Z. DeStefano, S. Goldwasser, Y. Ishai, Y. T. Kalai, and J. Thaler, "Sum-check protocol for approximate computations," Cryptology ePrint Archive, Paper 2025/2152, 2025. [Online]. Available: https://eprint.iacr.org/2025/2152

[50] J. Thaler, "Time-optimal interactive proofs for circuit evaluation," 2017. [Online]. Available: https://arxiv.org/abs/1304.3812

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

A Additional Preliminaries

A.1 zk-SNARK

Let $\mathcal{R} \subseteq \mathcal{X} \times \mathcal{W}$ be an NP relation, where $x \in \mathcal{X}$ is the public instance and $w \in \mathcal{W}$ is the witness. The associated language is $L_{\mathcal{R}} = \{x : \exists w \text{ such that } (x, w) \in \mathcal{R}\}$. A preprocessing zk-SNARK for $\mathcal{R}$ is a tuple of PPT algorithms

$$\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$$

with syntax

$$\begin{aligned}(\mathsf{pk}, \mathsf{vk}) &\leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}),\\ \pi &\leftarrow \mathsf{Prove}(\mathsf{pk}, x, w),\\ b &\leftarrow \mathsf{Verify}(\mathsf{vk}, x, \pi).\end{aligned}$$

Here π is a proof and $b \in \{0, 1\}$ is the Verifier's decision. All probabilities below are taken over the randomness of the relevant algorithms.

- **Completeness.** Honest proofs for true statements verify. For every $(x, w) \in \mathcal{R}$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{vk}, x, \pi) = 1\\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}),\\ \pi \leftarrow \mathsf{Prove}(\mathsf{pk}, x, w)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

- **Knowledge soundness.** Any efficient Prover that produces an accepting proof must know a valid witness. For every PPT adversary $\mathcal{A}$, there exists a PPT extractor $\mathcal{E}^{\mathcal{A}}$ such that

$$\Pr\left[\begin{array}{c}(x, w) \in \mathcal{R}\\ (\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}),\\ (x, \pi) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{vk}),\\ w \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{pk}, \mathsf{vk}, x)\end{array}\right]$$

is at least

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{vk},x,\pi)=1\\ (\mathsf{pk},\mathsf{vk})\leftarrow\mathsf{Setup}(1^\lambda,\mathcal{R}),\\ (x,\pi)\leftarrow\mathcal{A}(\mathsf{pk},\mathsf{vk})\end{array}\right]-\mathsf{negl}(\lambda).$$

- **Zero knowledge.** The proof reveals no information about the witness beyond the truth of the statement. Formally, there exists a PPT simulator $\mathsf{Sim}=(\mathsf{SimSetup},\mathsf{SimProve})$ such that, for every PPT distinguisher $\mathcal{D}$ and every $(x,w)\in\mathcal{R}$,

$$\begin{aligned}\mathsf{Adv}^{\mathsf{zk}}_{\Pi,\mathcal{D}}(\lambda)&=|\Pr[\mathsf{Real}=1]-\Pr[\mathsf{Ideal}=1]|\\&\leq\mathsf{negl}(\lambda).\end{aligned}$$

  In the real experiment, $(\mathsf{pk},\mathsf{vk})\leftarrow\mathsf{Setup}(1^\lambda,\mathcal{R})$, $\pi\leftarrow\mathsf{Prove}(\mathsf{pk},x,w)$, and the output is $\mathcal{D}(\mathsf{pk},\mathsf{vk},x,\pi)$. In the ideal experiment, $(\mathsf{pk},\mathsf{vk},\tau)\leftarrow\mathsf{SimSetup}(1^\lambda,\mathcal{R})$, $\pi\leftarrow\mathsf{SimProve}(\tau,x)$, and the output is $\mathcal{D}(\mathsf{pk},\mathsf{vk},x,\pi)$.

- **Non-interactivity.** The Prover sends a single proof π. Verification depends only on (vk,x,π) and requires no further Prover interaction.

- **Succinctness.** The proof and Verifier are short relative to the witness and the proved computation. There exists a polynomial p such that $|\pi|\leq p(\lambda,\log|\mathcal{R}|)$ and Verify runs in time $p(\lambda,|x|,\log|\mathcal{R}|)$, independent of the witness size except through the public instance and relation description.

A.2 Polynomial Commitment Scheme

A polynomial commitment scheme lets a Prover commit to a polynomial and later open the commitment at selected evaluation points. Let $\mathcal{P}_{n,D}\subseteq\mathbb{F}[X_1,\ldots,X_n]$ be a public class of admissible n-variate polynomials, where D denotes the relevant degree, size, or sparsity bound. A polynomial commitment scheme for $\mathcal{P}_{n,D}$ is a tuple of PPT algorithms

$$\mathsf{PC}=(\mathsf{Setup},\mathsf{Commit},\mathsf{Open},\mathsf{Verify})$$

with the following syntax:

$$\begin{aligned}\mathsf{pp}&\leftarrow\mathsf{Setup}(1^\lambda,n,D),\\(C,\mathsf{st})&\leftarrow\mathsf{Commit}(\mathsf{pp},f;r),\\\pi&\leftarrow\mathsf{Open}(\mathsf{pp},f,\mathsf{st},\vec{x}),\\b&\leftarrow\mathsf{Verify}(\mathsf{pp},C,\vec{x},y,\pi).\end{aligned}$$

Here $f\in\mathcal{P}_{n,D}$, r is commitment randomness, C is the commitment, st is opening state, $\vec{x}\in\mathbb{F}^n$ is an evaluation point, $y\in\mathbb{F}$ is the claimed value, and π is an evaluation proof.

A polynomial commitment scheme satisfies the following properties.

- **Completeness.** Honest openings verify. For every $f\in\mathcal{P}_{n,D}$ and every $\vec{x}\in\mathbb{F}^n$,

$$\Pr\left[\begin{array}{c}\mathsf{Verify}(\mathsf{pp},C,\vec{x},f(\vec{x}),\pi)=1\\ \mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda,n,D),\\ (C,\mathsf{st})\leftarrow\mathsf{Commit}(\mathsf{pp},f;r),\\ \pi\leftarrow\mathsf{Open}(\mathsf{pp},f,\mathsf{st},\vec{x})\end{array}\right]=1.$$

- **Binding.** Once a commitment is fixed, an adversary cannot open it to two different values at the same point. For every PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} y_0 \neq y_1 \\ \wedge\ \mathsf{Verify}(\mathsf{pp}, C, \vec{x}, y_0, \pi_0) = 1 \\ \wedge\ \mathsf{Verify}(\mathsf{pp}, C, \vec{x}, y_1, \pi_1) = 1 \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, D), \\ (C, \vec{x}, y_0, y_1, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

- **Knowledge soundness.** Any Prover that convinces the Verifier of an evaluation knows a polynomial consistent with the commitment and the claimed value. Let

$$\mathcal{R}_{\mathsf{PC}} = \{((\mathsf{pp}, C, \vec{x}, y); (f, r, \mathsf{st})) :\ f \in \mathcal{P}_{n,D}, \\ (C, \mathsf{st}) = \mathsf{Commit}(\mathsf{pp}, f; r)\ \wedge\ y = f(\vec{x})\}.$$

  The scheme is knowledge sound if, for every PPT adversary $\mathcal{A}$ that outputs an accepting tuple $(C, \vec{x}, y, \pi)$, there exists a PPT extractor $\mathcal{E}$ such that

$$\Pr\left[\begin{array}{c} ((\mathsf{pp}, C, \vec{x}, y); (f, r, \mathsf{st})) \in \mathcal{R}_{\mathsf{PC}} \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, D), \\ (C, \vec{x}, y, \pi) \leftarrow \mathcal{A}(\mathsf{pp}), \\ (f, r, \mathsf{st}) \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{pp}, C, \vec{x}, y) \end{array}\right]$$

  is at least

$$\Pr\left[\begin{array}{c} \mathsf{Verify}(\mathsf{pp}, C, \vec{x}, y, \pi) = 1 \\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, n, D), \\ (C, \vec{x}, y, \pi) \leftarrow \mathcal{A}(\mathsf{pp}) \end{array}\right] - \mathsf{negl}(\lambda).$$

- **Succinctness.** The commitment and evaluation proof are short compared with the polynomial representation. Concretely, $|C|$ and $|\pi|$ are $\mathsf{poly}(\lambda, n, \log D)$, and the Verifier runs in time $\mathsf{poly}(\lambda, n, \log D)$ plus the time needed to read the public point and value. The Prover may run in time nearly linear in the size of the committed polynomial.

A.3 Interactive Proof

This appendix recalls the algebraic tools used by the fast mode.

A.3.1 Multilinear Extensions

Let $\mathbb{F}$ be a finite field. For a function $V : \{0,1\}^\ell \to \mathbb{F}$, the multilinear extension is the unique polynomial $\widetilde{V} : \mathbb{F}^\ell \to \mathbb{F}$ that agrees with V on the Boolean hypercube and has degree at most one in each variable. We use the equality polynomial

$$\widetilde{\beta}(z, b) = \prod_{i=1}^{\ell} ((1 - z_i)(1 - b_i) + z_i b_i).$$

For Boolean z and b, this polynomial is 1 when $z = b$ and 0 otherwise. Therefore

$$\widetilde{V}(z) = \sum_{b \in \{0,1\}^\ell} \widetilde{\beta}(z, b) V(b).$$

We view vectors, matrices, and circuit layers as functions over Boolean index sets. Their low-degree extensions are the objects manipulated by sum-check and GKR protocols.

A.3.2 Sum Checks

The sum-check protocol [50] proves a claim of the form

$$H = \sum_{b \in \{0,1\}^\ell} f(b)$$

for a low-degree polynomial $f : \mathbb{F}^\ell \to \mathbb{F}$. In round t, after challenges $r_1, \ldots, r_{t-1}$ have been fixed, the Prover sends the univariate polynomial

$$g_t(X) = \sum_{b_{t+1},\ldots,b_\ell \in \{0,1\}} f(r_1, \ldots, r_{t-1}, X, b_{t+1}, \ldots, b_\ell).$$

The Verifier checks that $g_t(0) + g_t(1)$ equals the previous round's claim, samples a fresh challenge r_t, and updates the claim to $g_t(r_t)$. After ℓ rounds, the Verifier checks the final value against $f(r_1, \ldots, r_\ell)$. For multilinear f, the protocol is perfectly complete and has soundness error at most $\ell/|\mathbb{F}|$.

A.3.3 GKR

The GKR protocol [17] proves correct evaluation of a fan-in-two layered arithmetic circuit. Let the circuit have layers $V_0, \ldots, V_d$, where V_0 is the input layer and V_d is the output layer. Each gate in layer i is either an addition or multiplication gate whose children lie in layer $i-1$. Let $\widetilde{\mathsf{add}}_i$ and $\widetilde{\mathsf{mul}}_i$ denote the multilinear extensions of the wiring predicates. For a point r_i, the claimed value $\widetilde{V}_i(r_i)$ can be written as $\sum_{u,v,w} \widetilde{\beta}(r_i, u)\Big(\widetilde{\mathsf{add}}_i(u,v,w)(V_{i-1}(v) + V_{i-1}(w)) + \widetilde{\mathsf{mul}}_i(u,v,w)V_{i-1}(v)V_{i-1}(w)\Big)$, where the sum ranges over Boolean gate and wire indices. GKR applies sum-check to this identity. The result is one or two evaluation claims about $\widetilde{V}_{i-1}$ at random points. The Verifier combines the claims with a random linear combination and repeats the same reduction for the next layer. This recursive reduction is the central efficiency gain: the Prover streams layer values to answer sum-check messages, while the Verifier only evaluates low-degree wiring predicates and keeps a constant number of field claims per layer. At the input layer, the final claim is checked directly against the public input or against an opening of a committed input polynomial. For a circuit of size S and depth d, GKR has near-linear Prover time, Verifier work polylogarithmic in S plus input access, and communication proportional to d up to logarithmic factors.

B SPEG-Commit

This part gives the full protocol for SPEG-Commit.

Protocol 7 SPEG-COMMIT

Protocol setting. Let $\mathcal{C}$ be the arithmetic circuit denoting $\mathsf{JPEG.Lossify}$ with public parameters DCT matrix D and Quantization matrix Q. Let $D_3 = D$ and $D_2 = D^\top$. Let $X_1 = X$ be the source image, let X_i be the output of the i-th layer of $\mathcal{C}$.

Function SPEG-COMMIT.$\mathsf{KeyGen}(1^\lambda)$: **return** $\mathsf{pp} \leftarrow \mathsf{KZH.KeyGen}(1^\lambda)$

Function SPEG-COMMIT.$\mathsf{Commit}(X, \mathsf{pp}, r_X)$

1: View X as an array and define its multilinear extension $\widetilde{X}$.

2: $\mathsf{com}_X \leftarrow \mathsf{KZH.Commit}(\widetilde{X}, \mathsf{pp}, r_X)$. Send com_X to $\mathcal{V}$.

Protocol $\langle$SPEG-COMMIT.$\mathsf{Prove}(X, r_X)$, SPEG-COMMIT.$\mathsf{Verify}(\mathsf{com}_X)\rangle(\zeta, D, Q, \mathsf{pp})$

3: **(1)** The Prover $\mathcal{P}$ evaluates C to compute the intermediate layer values X_2, X_3, and X_4.

4: **(2)** The Verifier $\mathcal{V}$ calculates $Z \leftarrow \mathsf{JPEG.Decode}(\zeta)$ and defines the multilinear extension of $Z = X_4$ as $\widetilde{X}_4$.

$\mathcal{V}$ chooses a random $r^{(4)}$ and sends it to $\mathcal{P}$. Both parties compute $\widetilde{X}_4(r^{(4)})$.

5: **(3)** For $i = 4, 3, 2$, with current claim $\widetilde{X}_i(r^{(i)})$, run the corresponding layer check:

6: -If layer i is a DCT layer, the parties run sumcheck protocol for matrix multiplication in [50] on $\widetilde{X}_i(x, y) = \sum_z \widetilde{D}_i(x, z) \cdot \widetilde{X}_{i-1}(z, y)$. At the end, $\mathcal{V}$ receives $\widetilde{X}_{i-1}(r^{(i-1)})$.

7: -If layer i is a Quantization layer, the parties run sumcheck for Hadamard product in [32]. At the end, $\mathcal{V}$ receives $\widetilde{X}_{i-1}(r^{(i-1)})$.

8: **(4)** At the input layer, the Verifier has received $\widetilde{X}_1(r^{(1)})$, set $\widetilde{\mathsf{in}}(r_{\mathsf{in}}) = \widetilde{X}_1(r^{(1)})$.

9: **(5)** $\mathcal{P}$ and $\mathcal{V}$ run $\langle \mathsf{KZH.Open}(\widetilde{X}, r_X), \mathsf{KZH.Verify}(\mathsf{com}_X)\rangle(r^{(1)}, \mathsf{pp})$.

$\mathcal{V}$ checks that $\widetilde{\mathsf{in}}(r_{\mathsf{in}}) = \widetilde{X}(r^{(1)})$. If all verifications of the polynomial commitment and sumchecks pass, the Verifier outputs 1. Otherwise, the Verifier outputs 0.

C KZH with KZG SRS

This appendix records the assumptions and the KZH with KZG SRS used in the fast mode.

Recall that powers-of-(τ) setup ceremonies have already been performed for many KZG-based systems, and compatible reference strings therefore already exist in practice. Reusing these existing SRSs, rather than requiring users to run a new trusted setup specifically for KZH, substantially lowers the deployment barrier.

For integer t, let $B_t = \{0,1\}^t$. For vector $\vec{u} = (u_0, \ldots, u_{t-1}) \in B_t$, define $\mathrm{digit}(\vec{u}) = \sum_{k=0}^{t-1} u_k 2^k$. For $\vec{i} \in B_\mu$ and $\vec{j} \in B_\nu$, we use $\vec{i}|\vec{j}$ for concatenation so that $\mathrm{digit}(\vec{i}|\vec{j}) = \mathrm{digit}(\vec{i})2^\nu + \mathrm{digit}(\vec{j})$.

C.1 Assumptions

Definition 1 ((Q-Discrete Log) [51])**.** *Fix an integer $Q \geq 1$. Let $[a]_1$ and $[a]_2$ denote the canonical encodings of $a \in \mathbb{F}_p$ in $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Sample $x \leftarrow \mathbb{F}_p$ uniformly. The Q-Discrete Log problem is: given $[1]_1, [x]_1, \ldots, [x^Q]_1, [1]_2, [x]_2, \ldots, [x^Q]_2$, output x.*

We assume that Q-Discrete Log is hard, namely that every efficient adversary succeeds with probability at most $\mathrm{negl}(\lambda)$. In the proof below, let $n = 2^\mu$, $m = 2^\nu$, $N = mn$, and $L = 2N$. The largest exponent appearing in the pairing equations is at most $Q = 2L + m - 1 = 4N + m - 1$. Thus, the proof uses the Q-Discrete Log assumption with $Q = 4N + m - 1$.

Algorithm 8 KZH with KZG SRS

function SETUP($1^\lambda, \mu, \nu$)
  Input: security parameter and index lengths μ, ν
  Output: a structured reference string srs
  Let $n = 2^\mu$, $m = 2^\nu$, $L = 2^{\mu+\nu+1}$
  Sample generators $G \leftarrow^{\$} \mathbb{G}_1$, $V \leftarrow^{\$} \mathbb{G}_2$
  Sample single trapdoor $\rho \leftarrow^{\$} \mathbb{F}_p$
  Generators $\{G^{\vec{j}} = \rho^{\mathrm{digit}(\vec{j})} G\}_{\vec{j} \in B_\nu}$, $\{\tau^{\vec{i}} = \rho^{\mathrm{digit}(\vec{i})2^\nu}\}_{\vec{i} \in B_\mu}$
  $\{H^{\vec{i},\vec{j}} \leftarrow \tau^{\vec{i}} \times G^{\vec{j}} = \rho^{\mathrm{digit}(\vec{i})2^\nu} \times \rho^{\mathrm{digit}(\vec{j})} G = \rho^{\mathrm{digit}(\vec{i}|\vec{j})} G\}_{\vec{i} \in B_\mu, \vec{j} \in B_\nu}$;
  $\{H^{\vec{j}} \leftarrow \rho^L \times G^{\vec{j}} = \rho^{L+\mathrm{digit}(\vec{j})} G\}_{\vec{j} \in B_\nu}$;
  $\{V^{\vec{i}} \leftarrow \tau^{\vec{i}} \times V = \rho^{\mathrm{digit}(\vec{i})2^\nu} V\}_{\vec{i} \in B_\mu}$;
  $V' \leftarrow \rho^L \times V$;
  return $srs = (\{G^{\vec{j}}, H^{\vec{i},\vec{j}}, H^{\vec{j}}, V^{\vec{i}}, V'\}_{\vec{i} \in B_\mu, \vec{j} \in B_\nu})$
end function

function COMMIT(srs, f)
  Input: srs and a function $f : B_\mu \times B_\nu \to \mathbb{F}_p$
  Output: a commitment C and cached row commitments aux
  $C \leftarrow \sum_{\vec{x} \in B_\mu} \sum_{\vec{y} \in B_\nu} f(\vec{x}, \vec{y}) \cdot H^{\vec{x},\vec{y}}$
  for all $\vec{x} \in B_\mu$ **do**
    $D^{\vec{x}} \leftarrow \sum_{\vec{y} \in B_\nu} f(\vec{x}, \vec{y}) \cdot H^{\vec{y}}$
  end for
  return C and cache $aux = (\{D^{\vec{x}}\}_{\vec{x} \in B_\mu})$
end function

function OPEN($\mathsf{srs}, f, \vec{x}_0, \vec{y}_0, \mathsf{aux}$)
  Input: srs, f, a row index $\vec{x}_0$, a point $\vec{y}_0$, and aux
  Output: the opened value z_0 and an opening proof π
  Define the opened row polynomials $f^*(\vec{Y}) \leftarrow f(\vec{x}_0, \vec{Y})$
  $z_0 \leftarrow f^*(\vec{y}_0)$
  return z_0 and $\pi = \big(f^*(\vec{Y}), \mathsf{aux}\big)$
end function

function VERIFY($\mathsf{srs}, C, \vec{x}_0, \vec{y}_0, \pi, z_0$)
  Input: srs, a commitment C, an opening point $(\vec{x}_0, \vec{y}_0)$, a proof π, and a claimed value z_0
  Output: accept iff all checks pass
  Parse π as $\big(f^*(\vec{Y}), (D^{\vec{x}})_{\vec{x} \in B_\mu}\big)$
  Check $e(C, V') = \sum_{\vec{x} \in B_\mu} e(D^{\vec{x}}, V^{\vec{x}})$
  Check $\sum_{\vec{y} \in B_\nu} f^*(\vec{y}) \cdot H^{\vec{y}} = \sum_{\vec{x} \in B_\mu} \mathsf{eq}(\vec{x}, \vec{x}_0) \cdot D^{\vec{x}}$
  Check $z_0 = f^*(\vec{y}_0)$
  return accept iff all three checks succeed
end function

C.2 KZH with a KZG SRS

We now describe the KZH with KZG SRS variant used in SPEG-Commit. The key point is that one structured reference string simultaneously supports a multilinear commitment for the image matrix and a KZG-style row opening. Let $n = 2^\mu$ and $m = 2^\nu$. The committed object is a function $f : B_\mu \times B_\nu \to \mathbb{F}_p$. The first index selects a row and the second selects a position within that row.

Theorem 4. *In AGM under Q-Discrete Log with $Q = 4N + m - 1$, the protocol 8 KZH with KZG SRS is a complete and knowledge sound polynomial commitment scheme.*

Proof. We prove Theorem 4 with:
Completeness follows immediately by definition.
Knowledge Soundness. Under AGM, we are given C and $D^{\vec{x}}$ along with
$C = \sum_{\vec{x}\in B_\mu} \sum_{\vec{y}\in B_\nu} c_{\vec{x},\vec{y}} \times H^{\vec{x},\vec{y}} + \sum_{\vec{y}\in B_\nu} \hat{c}_{\vec{y}} \times H^{\vec{y}}$
$D^{\vec{x}} = \sum_{\vec{y}\in B_\nu} d^{\vec{x}}_{\vec{y}} \times H^{\vec{y}} + \sum_{\vec{a}\in B_\mu} \sum_{\vec{b}\in B_\nu} \hat{d}^{\vec{x}}_{\vec{a},\vec{b}} \times H^{\vec{a},\vec{b}}$.

The first summation in $D^{\vec{x}}$ is the intended row-opening basis, while the second summation represents possible cheating terms using the old matrix basis.

As the output of $\mathcal{V}$ is acc, we have the following:
1. $e(C, V') = \sum_{\vec{x}\in B_\mu} e(D^{\vec{x}}, V^{\vec{x}})$.
$LHS = e(\sum_{\vec{x}\in B_\mu} \sum_{\vec{y}\in B_\nu} c_{\vec{x},\vec{y}} \times H^{\vec{x},\vec{y}} + \sum_{\vec{y}\in B_\nu} \hat{c}_{\vec{y}} \times H^{\vec{y}}, \rho^L \times V)$
$= \sum_{\vec{x}\in B_\mu} \sum_{\vec{y}\in B_\nu} c_{\vec{x},\vec{y}} \times \rho^{L+\mathrm{digit}(\vec{x}|\vec{y})} \times e(G, V) + \sum_{\vec{y}\in B_\nu} \hat{c}_{\vec{y}} \times \rho^{2L+\mathrm{digit}(\vec{y})} \times e(G, V)$;
$RHS = \sum_{\vec{x}\in B_\mu} e(\sum_{\vec{y}\in B_\nu} d^{\vec{x}}_{\vec{y}} \times H^{\vec{y}} + \sum_{\vec{a}\in B_\mu} \sum_{\vec{b}\in B_\nu} \hat{d}^{\vec{x}}_{\vec{a},\vec{b}} \times H^{\vec{a},\vec{b}}, \rho^{\mathrm{digit}(\vec{x})2^\nu} \times V)$
$= \sum_{\vec{x}\in B_\mu} \sum_{\vec{y}\in B_\nu} d^{\vec{x}}_{\vec{y}} \times \rho^{L+\mathrm{digit}(\vec{x}|\vec{y})} \times e(G, V) + \sum_{\vec{x}\in B_\mu} \sum_{\vec{a}\in B_\mu} \sum_{\vec{b}\in B_\nu} \hat{d}^{\vec{x}}_{\vec{a},\vec{b}} \times \rho^{\mathrm{digit}(\vec{x})2^\nu+\mathrm{digit}(\vec{a}|\vec{b})} \times e(G, V)$.

We now compare powers of ρ by three disjoint intervals.
Interval $[\mathbf{0}, \mathbf{2N})$. The terms $\rho^{\mathrm{digit}(\vec{x})2^\nu+\mathrm{digit}(\vec{a}|\vec{b})} \times e(G, V)$ are old-times-old pairing products. Indeed, $0 \le \mathrm{digit}(\vec{x})2^\nu < N$ and $0 \le \mathrm{digit}(\vec{a}|\vec{b}) < N$, so all such exponents lie in $[0, 2N)$. Since $L = 2N$, these terms cannot collide with the intended shifted interval below. We do not need to prove that every coefficient $\hat{d}^{\vec{x}}_{\vec{a},\vec{b}}$ is individually zero here; they may only cancel with other old-times-old terms inside the same interval. The important point is that they cannot affect the coefficient comparison in the interval $[L, L + N)$.
Interval $[\mathbf{L}, \mathbf{L+N}) = [\mathbf{2N}, \mathbf{3N})$. The intended decomposition terms lie in this interval: $\rho^{L+\mathrm{digit}(\vec{x}|\vec{y})} \times e(G, V)$. No old-times-old term lies in this interval. Therefore, by Q-Discrete Log with $Q = 4N + m - 1$, the coefficients in this interval must match: $c_{\vec{x},\vec{y}} = d^{\vec{x}}_{\vec{y}}$ for all $\vec{x} \in B_\mu, \vec{y} \in B_\nu$.
Interval $[\mathbf{2L}, \mathbf{2L+m}) = [\mathbf{4N}, \mathbf{4N+m})$. The terms $\rho^{2L+\mathrm{digit}(\vec{y})} \times e(G, V)$ come only from the unwanted $\hat{c}_{\vec{y}} \times H^{\vec{y}}$ part of C. There is no matching term on the right-hand side. Hence, by Q-Discrete Log with $Q = 4N + m - 1$, $\hat{c}_{\vec{y}} = 0$ for all $\vec{y} \in B_\nu$.

We conclude that $C = \sum_{\vec{x}\in B_\mu} \sum_{\vec{y}\in B_\nu} c_{\vec{x},\vec{y}} \times H^{\vec{x},\vec{y}}$. Moreover, for every $\vec{x} \in B_\mu$, the coefficient of $H^{\vec{y}}$ in the shifted row-opening basis of $D^{\vec{x}}$ is $c_{\vec{x},\vec{y}}$. The possible remaining terms in $D^{\vec{x}}$ are exactly $\sum_{\vec{a}\in B_\mu} \sum_{\vec{b}\in B_\nu} \hat{d}^{\vec{x}}_{\vec{a},\vec{b}} \times H^{\vec{a},\vec{b}}$, which lie in the old matrix basis and cannot collide with the shifted row-opening basis because $L = 2N$.
2. $\sum_{\vec{y}\in B_\nu} f^*(\vec{y}) \times H^{\vec{y}} = \sum_{\vec{x}\in B_\mu} \mathsf{eq}(\vec{x}, \vec{x}_0) \times D^{\vec{x}}$
$LHS = \sum_{\vec{y}\in B_\nu} f^*(\vec{y}) \times \rho^{L+\mathrm{digit}(\vec{y})} G$
$RHS = \sum_{\vec{x}\in B_\mu} \mathsf{eq}(\vec{x}, \vec{x}_0) \times D^{\vec{x}}$
$= \sum_{\vec{x}\in B_\mu} \mathsf{eq}(\vec{x}, \vec{x}_0) \times \left(\sum_{\vec{y}\in B_\nu} d^{\vec{x}}_{\vec{y}} \times H^{\vec{y}} + \sum_{\vec{a}\in B_\mu} \sum_{\vec{b}\in B_\nu} \hat{d}^{\vec{x}}_{\vec{a},\vec{b}} \times H^{\vec{a},\vec{b}}\right)$

$= \sum_{\vec{x}\in B_\mu} \mathsf{eq}(\vec{x},\vec{x}_0) \times \sum_{\vec{y}\in B_\nu} c_{\vec{x},\vec{y}} \times H^{\vec{y}} + \sum_{\vec{x}\in B_\mu} \mathsf{eq}(\vec{x},\vec{x}_0) \times \sum_{\vec{a}\in B_\mu} \sum_{\vec{b}\in B_\nu} \hat{d}^{\vec{x}}_{\vec{a},\vec{b}} \times H^{\vec{a},\vec{b}}$, by part 1.

The first summation lies in the shifted row-opening basis interval $[L, L+m) = [2N, 2N+m)$, while the second summation lies in the old matrix-basis interval $[0, N)$. These two intervals are disjoint. Therefore, by Q-Discrete Log with $Q = 4N + m - 1$, the old-basis part in the selected linear combination must vanish, and the shifted coefficients must match. Hence, for every $\vec{y} \in B_\nu$, $f^*(\vec{y}) = \sum_{\vec{x}\in B_\mu} \mathsf{eq}(\vec{x},\vec{x}_0)\, c_{\vec{x},\vec{y}}$.

Finally, the verifier also checks $f^*(\vec{y}_0) = z_0$.

The extractor outputs $f(\vec{X},\vec{Y}) = \sum_{\vec{x}\in B_\mu} \sum_{\vec{y}\in B_\nu} c_{\vec{x},\vec{y}}, \mathsf{eq}(\vec{x},\vec{X}), \mathsf{eq}(\vec{y},\vec{Y})$. By the equation above, $f^*(\vec{Y}) = f(\vec{x}_0,\vec{Y})$, and by the final verifier check, $z_0 = f^*(\vec{y}_0) = f(\vec{x}_0,\vec{y}_0)$. Since C commits to the extracted f, the construction is knowledge sound. □