Lightweight Blockchain-Assisted Image Authentication for Low-Bandwidth IoT Networks

Tile-level inline tags, sparse Merkle manifests and batched ledger anchoring compared with full-image, block-level and feature-level authentication

[1]Pravat Satpathy, [2]Sonia Mohapatra, [3]Saswat Swain
[1]Researcher, [2]Researcher, [3]Ph.D Scholar
[1,2,3]Research and Development Division,
[1,2,3]Tekstein Solutions Private Limited, Bhubaneswar, Odisha, India
analytics.pravat@gmail.com, smctc98@gmail.com, saswat.mck@gmail.com (corresponding author: analytics.pravat@gmail.com)

***Abstract*—Camera-equipped Internet of Things (IoT) nodes used for environmental monitoring, wildlife observation, industrial inspection and disaster response must deliver images whose integrity can be proven later, yet they transmit over links of a few tens of kilobits per second that lose frames in bursts. The common practice of hashing the complete image and anchoring that digest on a blockchain makes verification all-or-nothing: one lost frame invalidates the image, and ledger records grow with every camera and every frame. This paper proposes LBIA, a lightweight blockchain-assisted image authentication scheme that combines tile-aligned JPEG coding, a 64-bit truncated HMAC tag carried inside every tile, a signed Merkle root delivered as a sparse manifest, and Merkle batching of image roots so that one ledger transaction covers K images. A variant, LBIA-P, adds a 64-bit feature-level digest per tile for provenance after archival transcoding. LBIA was compared with full-image hashing, block hashing and feature-level authentication in a MATLAB-language simulation, executed in GNU Octave, on 12 test images of 512 x 512 pixels with IEEE 802.15.4-class framing and Bernoulli and Gilbert loss channels. Tile-aligned coding reproduced the whole-image reconstruction exactly and cost 0.39% extra rate for 64 x 64 tiles. In one-shot delivery at 10% bursty loss, LBIA authenticated 73.0% of the image area while full-image hashing authenticated none; with link-layer retransmission at 50 kbit/s and 5% bursty loss, the first verified tile arrived after 0.13 s instead of 9.65 s, for 3.94% more bytes on air. Batching 64 images per transaction cut ledger growth from 282 to 4.41 bytes per image at the cost of a 97.5 s anchoring delay. The feature-level digest reached an area under the ROC curve of 0.850 but missed every local-blur manipulation, so it is positioned as a complement to cryptographic tags rather than a replacement. Hardware validation, key management and adversarial collisions on the feature digest are left for future work.**

***Index Terms*—Blockchain, image authentication, Internet of Things, JPEG, Merkle tree, message authentication code, packet loss, perceptual hashing, wireless multimedia sensor networks.**

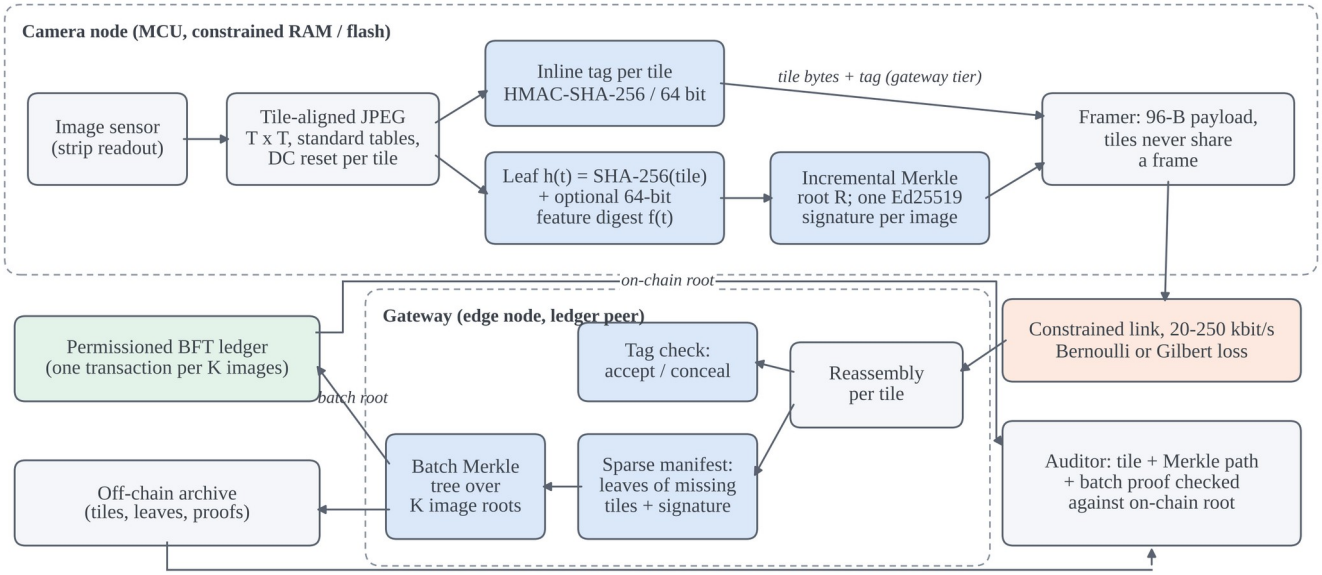


Fig. 1. Architecture of LBIA. Blue blocks are the authentication functions added to a conventional camera node and gateway; the gateway tier (inline tags) gives immediate, loss-tolerant verification, and the provenance tier (signed Merkle root, batch tree and ledger) gives public, after-the-fact auditability.

I. Introduction

Unattended cameras have become routine instruments in field science and in the operation of critical infrastructure. Camera traps document wildlife at a scale that now depends on automated image analysis [1], forest and river stations photograph sites that no road reaches [2], and the same class of hardware is mounted on pipelines, substations and the debris fields left after floods and cyclones. In most of these deployments the image is not an end in itself: it is evidence. A

ranger's prosecution, an insurance assessment or a relief allocation may later depend on showing that a particular frame was captured by a particular device at a particular time and has not been altered since.

The communication budget available to such devices is small. Sub-GHz and long-range links deliver from a few to a few hundred kilobits per second, and practical image transfer over them has been shown to suffer from packet loss once vegetation or terrain obstructs the path [3], [4]. Low-rate personal-area networks built on IEEE 802.15.4 [5] limit a frame to 127 bytes, so a modest 512 x 512 grey-scale JPEG of 30–40 kB becomes several hundred frames. Recent camera-trap work responds by shrinking the payload, either with progressive thumbnails [2] or with learned codecs adapted to each site [6], but it does not address how the receiver can trust what arrives.

Blockchain ledgers are attractive as a neutral, append-only record of what a device produced [7]. Existing image-integrity schemes typically compute a digest of each frame or image at the source and write it to a permissioned chain [8], [9], store Merkle roots of image features [10], or store perceptual hashes that tolerate benign processing [11], [12]. These designs were developed for networks in which the image arrives intact. On a lossy, narrow link they inherit three problems. First, a digest over the whole image cannot verify anything until every byte has arrived, so a single unrecovered frame destroys the evidential value of the entire picture. Second, per-block digests restore partial verifiability but multiply both the authentication bytes on air and the on-chain footprint, which matters because lightweight IoT ledgers are themselves capacity-limited [13], [14]. Third, perceptual digests survive recompression but are not cryptographic, so their detection power against small, deliberate edits has to be measured rather than assumed.

What is missing is a single, quantitative account of how compression, tiling, digest granularity, packet loss, storage and ledger cost interact for a constrained camera node. This paper provides that account and uses it to design a scheme that keeps the useful parts of each approach. The central question is: *given a fixed link rate and loss process, which authentication granularity maximises the fraction of an image that can be proven authentic, at what cost in bytes, delay, memory and ledger growth, and with what detection power against content manipulation?*

The main contributions are as follows.

1) A lightweight blockchain-assisted image authentication scheme, LBIA, in which every independently decodable JPEG tile carries its own 64-bit truncated HMAC tag, the sensor signs one Merkle root per image, the root and only the leaves of missing tiles travel in a sparse manifest, and the gateway anchors a Merkle tree of K image roots in a single ledger transaction. A variant, LBIA-P, adds a 64-bit feature-level digest per tile.

2) An analytical model with four propositions: bit-identical reconstruction under tile-aligned coding, the expected authenticated area under Bernoulli and Gilbert loss, the ledger-growth versus anchoring-delay trade-off of batching, and the frame width above which tile-wise processing needs less sender memory than full-image hashing. Each is checked numerically.

3) A reproducible MATLAB-language simulator, including a baseline JPEG encoder with standard tables validated against libjpeg, packet-level framing, two loss channels, five content attacks, two benign transcodings, and cost models for computation, memory, storage and ledger growth.

4) A comparison with full-image hashing, block hashing and feature-level authentication that reports the unfavourable results alongside the favourable ones, including cases in which the baseline is preferable.

Section II reviews related work. Section III states the problem and the system model. Section IV describes LBIA, Section V develops its mathematical model, and Section VI specifies the simulation. Results are discussed in Section VII and compared in Section VIII. Section IX summarises key outcomes and limitations, and Section X concludes.

II. Related Work

A. Blockchain-Anchored Image and Video Integrity

Danko *et al.* hashed video frames on a wireless IoT device before transmission and replicated the hashes across permissioned blockchain nodes so that later modification could be detected [8]. Li *et al.* divided sensing images into blocks, signed every block with time-period keys and verified the signatures at storage servers before committing them to a chain [9]. Khan and Byun encrypted industrial images and recorded cryptographic pixel values on a private permissioned ledger [15]. Chen *et al.* stored image features in a Merkle tree whose root was placed on chain, which allowed tampered regions to be located and partly restored, with the image itself kept in IPFS [10]. These works establish that ledgers can provide tamper evidence for images, but they assume reliable delivery and they record either one digest per image or one record per block, without analysing what either choice costs on a constrained link.

Saito recently proposed logging general sensor data with tamper-evident devices that sign readings, chain them with redundant hashes and anchor Merkle proofs on a blockchain service, explicitly so that records stay verifiable when some are lost [16]. LBIA shares the idea of loss-tolerant Merkle anchoring, but it is specific to compressed images: its unit of verification is an independently decodable JPEG tile, and its evaluation measures image quality, tile-level tamper localisation and the interaction with compression.

B. Perceptual Hashing and Semi-Fragile Authentication

Content-based authentication accepts benign processing such as recompression while rejecting semantic edits. Lin and Chang exploited invariant relations between DCT coefficients to separate JPEG compression from malicious manipulation [17]. Swaminathan *et al.* introduced keyed, randomised image hashes that resist both perceptual distortion and forgery [18], and later schemes select robust feature points to localise tampering [19]. A recent survey reports that robustness differs strongly across attack types, with rotation a persistent weakness [20]. Blockchain systems have adopted these hashes: White and Paul ran several OpenCV perceptual hashes on simulated ledger nodes [12], and Xu *et al.* combined a

contourlet-based perceptual hash with Hyperledger Fabric and IPFS for multi-band remote-sensing imagery [11]. Semi-fragile watermarking pursues the same goal by embedding the check inside the image [21]; for low-power sensor networks, Pal and Das Bit combined compression, a watermark and a short signature to separate sender forgery from in-transit tampering [22].

C. Image Transport over Constrained IoT Links

Measurements on LoRa show that packet reception depends on spreading factor, bandwidth, coding rate and packet length, and that image transfer fails under obstruction [3], [4]. Camera-trap networks therefore send progressively refined thumbnails [2] or learned latent codes [6]. None of these transport studies couples the delivery process with authentication.

D. Lightweight Ledgers for IoT

Public proof-of-work chains are unsuitable for sensor data, and IoT deployments favour permissioned ledgers with Byzantine-fault-tolerant ordering [23], [24] or purpose-built lightweight designs [25]. Surveys of consensus for resource-constrained IoT [14] and of lightweight blockchain for IoT [13] identify throughput, storage growth and the cost of participating nodes as the main limits, which motivates reducing the number and size of transactions per sensor record. Merkle aggregation, used at scale for certificate logs [26], is the standard tool for this.

TABLE I
COMPARISON OF RELATED IMAGE-INTEGRITY SCHEMES

Work, year	Approach and strength	Limitation for low-bandwidth IoT
Danko *et al.* [8], 2019	Per-frame video hash on device, permissioned chain; detects any later change	All-or-nothing per frame; loss not modelled
Chen *et al.* [10], 2019	Merkle tree of image features, root on chain, IPFS; localisation and recovery	Assumes intact delivery; no rate or loss analysis
Li *et al.* [9], 2020	Image blocks individually signed and verified by servers	One signature per block, heavy on air and on chain
Khan, Byun [15], 2020	Image encryption, pixel values on private chain	Large on-chain data; no constrained-link study
White, Paul [12], 2020	Several perceptual hashes on ledger nodes; tolerates processing	Weak under rotation; not cryptographic
Pal, Das Bit [22], 2023	Compression, watermark and short signature for WMSN; energy-aware	No ledger; watermark alters content
Xu *et al.* [11], 2023	NSCT perceptual hash, Fabric and IPFS; robust, localises	Heavy transform; per-image ledger records
Aberna *et al.* [21], 2024	Semi-fragile watermark with Swin Transformer; high fidelity	Too heavy for MCU-class nodes
Saito [16], 2025	Signed, hash-chained sensor logs, Merkle anchoring; tolerates lost records	Generic data; no image coding or quality study
This work (LBIA)	Tile-aligned JPEG, inline 64-bit tags, sparse manifest, K-image batches	Feature layer weak on blur; simulation only

Table I summarises the representative works. Taken together, they leave four gaps that this paper addresses: (i) the effect of authentication granularity on what can be verified when frames are lost has not been quantified; (ii) compression and authentication are designed separately, although tiling changes both the bit rate and the unit of verification; (iii) on-chain cost is reported, if at all, per image, without the batching that lightweight ledgers need; and (iv) perceptual digests are evaluated on robustness benchmarks rather than against the small, localised edits that matter for evidential imagery.

III. PROBLEM STATEMENT AND SYSTEM MODEL

A. Network and Data Model

A cluster of M camera nodes reports to one gateway over a single-hop, low-rate link of rate R. Each node captures an 8-bit grey-scale frame of W x H pixels, encodes it as baseline JPEG at quality factor Q, and transmits it in frames that carry at most $P = 96$ bytes of payload. The gateway is a ledger peer in a small permissioned network that orders transactions with a Byzantine-fault-tolerant protocol [23]. Images and proofs are kept in an off-chain archive; the ledger stores only commitments.

B. Threat Model and Trust Assumptions

The adversary can observe, drop, delay, inject and modify frames on the link, and can later modify images in the archive. Its goal is to make the gateway or an auditor accept content that the camera did not capture. The camera's signing key and the key it shares with its gateway are assumed uncompromised; physical attacks on the node are out of scope. At most f of $3f + 1$ ledger peers are Byzantine. The gateway is trusted to report verification outcomes honestly, but an auditor does not need to trust it, because the camera's signature and the on-chain batch root can be checked independently.

C. Engineering Constraints

Four constraints shape the design: link capacity (tens of kilobits per second), frame loss that is often bursty, node memory of tens to a few hundred kibibytes with flash used for store-and-forward, and ledger capacity, which must absorb records from every camera in the cluster. Energy is taken as proportional to bytes on air, which is the dominant term for low-rate radios.

D. Formal Objective

Let A in [0, 1] be the fraction of image area whose authenticity the gateway can establish, B_{air} the bytes on air per image, M_{s} the sender working memory and G the ledger bytes per image. The design problem is

$$\max E[A] \ \text{ s.t. } \ B_{\text{air}} \le (1 + \varepsilon) B_{\text{FH}}, \ M_{\text{s}} \le M_{\max}, \ G \le G_{\max}, \quad (1)$$

where B_{FH} is the on-air cost of the full-image hashing baseline and ε is the tolerated overhead. Detection power against content manipulation, verification delay and image quality are reported as secondary metrics.

IV. Proposed LBIA Scheme

A. Architecture Overview

Fig. 1 shows the pipeline. On the camera, a tile-aligned encoder produces N_{T} independently decodable entropy-coded segments. Each segment receives an inline tag for the gateway tier and a leaf digest for the provenance tier. The leaves are folded into an incremental Merkle tree whose root is signed once per image. At the gateway, every tile is checked as soon as its last frame arrives. The signed root, together with the leaves of any tiles that did not arrive, lets the gateway reconstruct and verify the image root; roots from K images are then combined into a batch tree whose root is the only datum written to the ledger.

B. Tile-Aligned JPEG Coding

The frame is divided into T x T tiles with T a multiple of 8, so tile edges coincide with DCT block edges. Every tile is coded as its own entropy-coded segment: its blocks are scanned in raster order, the DC predictor restarts at zero, and the segment is padded to a byte boundary. All tiles use the ITU-T T.81 Annex K tables [27] and one quantisation table, so the shared header is sent once per image (or not at all when both ends use the standard tables). The arrangement is equivalent to an abbreviated JPEG stream per tile; any decoder can reconstruct a tile on its own.

C. Inline Gateway-Tier Tags

For each tile t the camera computes

$$\tau_t = \text{Trunc}_{64}(\text{HMAC}_k(id \,\|\, t \,\|\, B_t)), \quad (2)$$

where k is the key shared with the gateway, id is the image identifier (camera, sequence number and capture time), B_t is the entropy-coded segment, and Trunc_{64} keeps the first 64 bits of the HMAC-SHA-256 output [28], [29]. The tag is appended to the last frame of the tile, and tiles never share a frame, so the loss of one frame affects exactly one tile.

D. Provenance Tier: Merkle Root, Signature and Sparse Manifest

The leaf of tile t is h_t = SHA-256(B_t). Leaves and inner nodes use the domain-separated construction of RFC 6962 [26], [30]:

$$\ell_t = H(0\text{x}00 \,\|\, h_t), \quad n = H(0\text{x}01 \,\|\, n_{\text{L}} \,\|\, n_{\text{R}}). \quad (3)$$

The camera keeps an incremental Merkle stack of at most $\lceil \log_2 N_{\text{T}} \rceil$ nodes and, after the last tile, signs

$$\sigma = \text{Sign}_{sk}(id \,\|\, Q \,\|\, T \,\|\, R_{\text{img}}) \quad (4)$$

with Ed25519 [31]. The sparse manifest carries σ and the leaves of tiles that the gateway reports as missing; in the loss-free case it is 80 bytes. Because the gateway can recompute the leaves of tiles it received, it can rebuild R_{img} and check σ regardless of how many tiles were lost.

E. Optional Feature-Level Layer (LBIA-P)

Archival storage is often reduced by recompressing old images, which breaks every cryptographic digest. LBIA-P therefore adds, inside the signed manifest, a 64-bit feature digest per tile: the decoded tile is reduced to a 32 x 32 thumbnail by block averaging, the 8 x 8 low-frequency DCT coefficients are computed, the 56 coefficients after DC in zig-zag order are compared with their median to give 56 sign bits, and an 8-bit tile mean is appended. A tile is declared modified if

$$s_t = \max(d_{\text{H}} / 56, \ |\mu_t - \mu'_t| / 32) > \theta, \quad (5)$$

where d_{H} is the Hamming distance between sign vectors, μ_t and μ'_t are the reference and received tile means, and θ is a threshold set from a target false-positive rate. Only the 8 x 8 low-frequency band is needed, which costs 10 240 multiply–accumulate operations per 64 x 64 tile.

F. Batched Ledger Anchoring

The gateway accumulates verified image roots and, when K have arrived, builds a batch Merkle tree and submits its root as one transaction. Each image keeps an inclusion proof of $\lceil \log_2 K \rceil$ hashes in the archive. An auditor verifies a single tile with the tile bytes, its path in the image tree, the camera's signature, the batch proof and the on-chain root, without contacting the gateway.

G. Gateway Decision Logic

A received tile is in one of three states. *Verified*: all its frames arrived and its tag matches; it is displayed and archived. *Missing*: one or more frames were lost; the region is concealed (mid-grey in this study) and, if the link allows, requested again. *Rejected*: frames arrived but the tag does not match; the tile is discarded and the event is logged as a possible attack. An image is anchored once its signature verifies, with the set of verified tiles recorded alongside the root, so that partial images remain auditable.

H. Algorithms and Complexity

Algorithm 1. LBIA sender (camera node)

Input: frame I (W x H), quality Q, tile side T, keys k, sk
Output: frames on air; signed manifest
stack ← empty Merkle stack; id ← (camera, seq, time)
for each tile row r (read T sensor lines) do
  for each tile t in row r do
    B_t ← EncodeSegment(tile, Q) // DC predictor reset
    τ_t ← $\mathrm{Trunc}_{64}(\mathrm{HMAC}_k(id \,\|\, t \,\|\, B_t))$
    h_t ← SHA-256(B_t); stack.push(h_t)
    if LBIA-P then f_t ← FeatureDigest(decoded tile)
    send Fragment($B_t \,\|\, \tau_t$) in frames of ≤ P bytes
  end for; release strip and acknowledged tiles
end for
R_{img} ← stack.root(); σ ← $\mathrm{Sign}_{sk}(id \,\|\, Q \,\|\, T \,\|\, R_{\mathrm{img}})$
send manifest(σ, $\{f_t\}$); on request send h_t of missing tiles

Algorithm 2. LBIA gateway verification and anchoring

Input: frames of image id; batch size K
Output: tile states; ledger transaction per K images
for each tile t whose frames are complete do
  if $\tau_t = \mathrm{Trunc}_{64}(\mathrm{HMAC}_k(id \,\|\, t \,\|\, B_t))$ then state ← Verified
  else state ← Rejected; log event
end for; tiles without complete frames ← Missing
request leaves h_t of Missing tiles; rebuild R_{img}
if $\mathrm{Verify}_{pk}(\sigma, id \,\|\, Q \,\|\, T \,\|\, R_{\mathrm{img}})$ then queue (id, R_{img}, states)
if queue holds K roots then
  R_{B} ← MerkleRoot(queue); submit Tx(R_{B}); store proofs
end if

Complexity. On the camera, coding is O(WH) as for any JPEG encoder. Authentication hashes every coded byte twice (HMAC and leaf) plus $N_{\mathrm{T}} - 1$ inner nodes, i.e. O($S + N_{\mathrm{T}}$) for an image of S bytes, and one signature. Working memory is O($TW + \log N_{\mathrm{T}}$) because tiles are released once acknowledged (Section V-F). The gateway performs O($S + N_{\mathrm{T}}$) hashing and one signature check per image and O(K) work per batch. An auditor needs O($\log N_{\mathrm{T}} + \log K$) hashes per tile.

V. Mathematical Modeling

A. Coding and Communication Cost

Let $Y_b(u, v)$ be the 2-D DCT of the level-shifted 8 x 8 block b. The quantised coefficients and the IJG quality scaling are

$$Z_b(u,v) = \mathrm{round}\left(Y_b(u,v) \,/\, q(u,v) \right), \tag{6}$$

$$q = \mathrm{clip}\left(\lfloor (q_0\, s(Q) + 50)/100 \rfloor, 1, 255 \right),$$

$$s(Q) = 5000/Q \text{ for } Q < 50, \;\; 200 - 2Q \text{ for } Q \geq 50, \tag{7}$$

where q_0 is the Annex K luminance table. The tiling rate overhead is

$$\Delta_T = \left(\Sigma_t \, |B_t| - |B_{\mathrm{full}}| \right) / |B_{\mathrm{full}}|. \tag{8}$$

With F frames of payload L_f, frame overhead O_{fr}, PHY overhead O_{phy} and ACK size O_{ack}, the bytes on air without loss are

$$B_{\mathrm{air}} = \Sigma_{f=1}^{F} \left(L_f + O_{\mathrm{fr}} + O_{\mathrm{phy}} \right) + F\, O_{\mathrm{ack}}. \tag{9}$$

A tile occupies $n_t = \lceil (|B_t| + 8)/P \rceil$ frames, whereas the full image occupies $N = \lceil (|B_{\mathrm{full}}| + 16)/P \rceil$ frames. Because each tile ends in a partly filled frame, tiling adds on average half a frame per tile; this, and not the authentication data, dominates overhead for small tiles.

Proposition 1 (reconstruction identity). If T is a multiple of 8 and all tiles share the quantisation table, the decoded image of the tiled stream equals that of the whole-image stream pixel for pixel.

Proof. Both streams quantise the same DCT blocks with the same table, so Z_b is identical for every block. Restarting the DC predictor and the byte padding change only the lossless entropy-coding stage, which the decoder inverts exactly. Dequantisation and inverse DCT therefore operate on identical inputs. □

B. Loss Channel and One-Shot Delivery

Frame loss is modelled by a Gilbert channel [32], [33] with a good state (no loss) and a bad state (loss). With mean burst length L_{b} and stationary loss probability p,

$$r = 1/L_{\mathrm{b}}, \quad q = r\,p\,/\,(1-p), \quad \pi_{\mathrm{G}} = 1 - p, \tag{10}$$

where r and q are the bad-to-good and good-to-bad transition probabilities. $L_{\mathrm{b}} = 1$ gives independent (Bernoulli) loss.

Proposition 2 (expected authenticated area). In one-shot delivery, the expected fraction of authenticated area under LBIA is

$$E[A] = (1/N_{\mathrm{T}})\, \Sigma_t \, \pi_{\mathrm{G}} \exp[(n_t - 1)\ln(1-q)] \quad \text{(Gilbert)},$$

$$E[A] = (1/N_{\mathrm{T}})\, \Sigma_t \exp[n_t \ln(1-p)] \quad \text{(Bernoulli)}, \tag{11}$$

while for full-image hashing it is at most $\pi_{\mathrm{G}}(1-q)^{N-1}$.

Proof. A tile is authenticated iff its n_t consecutive frames all arrive, since its tag travels with it. Under the Gilbert channel the first frame survives with probability π_{G}, and each subsequent frame survives iff the chain stays in the good state, with probability $1 - q$. Averaging over tiles gives (11). Full-image hashing needs all N data frames, and its signature frames add further conditions. □ Since $n_t << N$, the gap is exponential in $N - n_t$.

C. Retransmission and Verification Delay

With link-layer ARQ, frame j needs a_j attempts, a geometric variable under Bernoulli loss and a burst-correlated one under the Gilbert channel. Its completion time is

$$t_k = \Sigma_{j \leq k} \, a_j \left(8(L_j + O_{\mathrm{fr}} + O_{\mathrm{phy}} + O_{\mathrm{ack}}) / R + t_{\mathrm{turn}} \right). \tag{12}$$

A tile is verified at the completion time of its last frame; a full-image digest is verified at t_F. With a deadline D, the authenticated area is the fraction of tiles completed before D.

D. Security of the Two Tiers

If HMAC-SHA-256 is a pseudorandom function, an adversary that makes m forgery attempts against 64-bit tags succeeds with probability at most

$$\Pr[\mathrm{forge}] \leq m \cdot 2^{-64} + \varepsilon_{\mathrm{PRF}}, \tag{13}$$

which NIST guidance considers acceptable for truncated MACs when the number of verification attempts is bounded [34]; 802.15.4 link security offers the same 64-bit option. The provenance tier inherits the collision resistance of SHA-256 through the Merkle tree and the unforgeability of Ed25519. Neither tier protects against a compromised camera key, and the feature digest carries no cryptographic guarantee (Section IX).

E. Ledger Batching

Proposition 3 (batching trade-off). With images arriving at rate λ, transaction envelope B_{tx}, block interval T_{blk} and finality time T_{fin}, batching K roots gives

$$G(K) = (B_{tx} + 32)/K,$$
$$D(K) = (K - 1)/(2\lambda) + T_{blk}/2 + T_{fin}, \quad (14)$$

and the largest batch meeting an anchoring deadline D_{max} is $K^* = \lfloor 1 + 2\lambda(D_{max} - T_{blk}/2 - T_{fin}) \rfloor$. *Proof.* One transaction of B_{tx} + 32 bytes serves K images. The i-th image of a batch waits $(K - i)/\lambda$ for the batch to fill; averaging over i gives the first term, and the transaction then waits on average half a block interval plus finality. □

F. Sender Memory

Assume the camera keeps coded data until the gateway confirms authentication, as the end-to-end argument requires. With mean rate β bits per pixel, a streaming full-image encoder needs an 8-line strip plus the whole bitstream, while LBIA needs a T-line strip, two tile rows of bitstream in flight, the Merkle stack and two hash states:

$$M_{FH} = 8W + \beta WH/8 + 112,$$
$$M_{LBIA} = TW + \beta TW/4 + 32\lceil \log_2 N_T \rceil + 224. \quad (15)$$

Proposition 4 (memory crossover). For 4:3 frames ($H = 3W/4$), neglecting the logarithmic and constant terms, LBIA needs less memory than full-image hashing when

$$W > W^* = 32(\, T(1 + \beta/4) - 8\,) / (3\beta). \quad (16)$$

Proof. Substitute $H = 3W/4$ in (15) and solve $M_{LBIA} < M_{FH}$ for W. □ Block hashing and the feature-level baseline transmit their manifest first and must buffer the whole bitstream, so they never fall below full-image hashing.

G. Detection Metrics

A tile is a positive if at least 1/16 of its pixels were changed by an attack and a negative if it belongs to a benignly transcoded image. With *TP*, *FP*, *TN* and *FN* counted per tile, TPR = $TP/(TP + FN)$ and FPR = $FP/(FP + TN)$; the area under the ROC curve (AUC) is integrated over θ in (5). Image quality is measured by PSNR and by SSIM [35].

VI. Simulation Setup

A. Software and Hardware

All experiments are written in the MATLAB language and were executed in GNU Octave 8.4 [36] on a two-core x86-64 Linux host; they were not run in MATLAB itself in this study. SHA-256 is provided by Octave's built-in hash function and, in MATLAB, by the Java MessageDigest class, through one wrapper. Python was used only for plotting, for cross-checking the encoder with libjpeg, and for timing Ed25519 on the host. All random streams start from one global seed (20261016). Octave and MATLAB generate different random sequences from the same seed, so a MATLAB rerun will reproduce the statistics but not the individual Monte Carlo draws.

B. Dataset

Twelve public test images distributed with scikit-image (camera, astronaut, coffee, chelsea, rocket, moon, grass, gravel, brick, immunohistochemistry, hubble_deep_field and coins) were converted to 8-bit grey scale and resampled to 512 x 512 pixels. They span flat scenes (moon, 15.9 kB at $Q = 75$) to dense texture (grass, 78.4 kB). They are generic rather than field imagery, which is a limitation discussed in Section IX.

C. Encoder Validation

Octave's imwrite emits per-image optimised Huffman tables, which would distort tile-overhead comparisons, so a baseline encoder with the Annex K tables was implemented in MATLAB. Its streams, wrapped in JFIF headers, were decoded by libjpeg: the decoded pixels differed from the MATLAB reconstruction by at most 1 grey level (inverse-DCT rounding) for all 12 images at $Q = 50$, 75 and 90, and an isolated tile decoded on its own with the same bound. File sizes were within 1.08% of libjpeg's standard-table output (mean -0.42%); libjpeg's optimised tables would save a further 2.77% on average, equally for all schemes.

TABLE II
Simulation Parameters

Parameter	Value	Basis
Images	12, 512 x 512, 8-bit grey	S
JPEG quality Q	20–90 (nominal 75)	S
Tile side T	32, 64, 128 (nominal 64)	D
Payload / frame ovh. / PHY / ACK	96 / 25 / 6 / 11 B	S, A
Per-attempt turnaround t_{turn}	1 ms	A
Link rate R	20, 50, 250 kbit/s	S
Loss p; mean burst L_b	0–20%; 1 and 3 frames	A
Monte Carlo runs	500 per point and image	D
Deadline D	1.3 x loss-free FH time	D
Repetitions of FH/BH/FL auth. object	3 (one-shot)	D
Tag / leaf / signature / feature	8 / 32 / 64 / 8 B	S
Ledger tx envelope B_{tx}	250 B	A
T_{blk}, T_{fin}; ledger capacity	2 s, 2 s; 50 tx/s	A
Cameras; frame interval	20; 60 s ($\lambda = 1/3\ s^{-1}$)	A
MCU clock; SHA-256 cost	64 MHz; 20/50/100 cycles/B	A
Feature DCT cost	2 cycles per MAC	A
Attacks; trials	5 types; 10 per type and image	D
Benign transcoding	Q' = 50 and 30	D

S: standard or dataset; D: design choice; A: assumption, not measured.

D. Schemes Compared

Raw + hash: uncompressed frame, SHA-256 and signature (bandwidth reference only). *Full-image hash (FH)*: whole-image JPEG, one SHA-256 digest signed with Ed25519, one ledger record per image. *Block hash (BH)*: tiled JPEG, a signed manifest of 32-byte SHA-256 digests sent ahead of the tiles, all digests recorded on chain. *Feature-level (FL)*: tiled JPEG, a signed manifest of 64-bit feature digests sent ahead of the tiles. *LBIA* and *LBIA-P* as in Section IV. FH, BH and FL repeat their authentication object three times, at the start, middle and end of the image, in one-shot mode; LBIA's tags need no repetition.

E. Scenarios and Metrics

Two delivery regimes are simulated: one-shot (no feedback, as in unconfirmed LoRa uplinks or broadcast after a disaster)

and link-layer ARQ with unlimited retries and a deadline. Metrics are image quality (PSNR, SSIM), communication overhead (authentication bytes and bytes on air including framing, ACKs and retries), authentication accuracy (tile-level TPR, FPR and AUC under attacks and benign transcoding), verification delay (time to the first and to all verified tiles), and memory (sender working memory and store-and-forward capacity), plus ledger growth and anchoring delay.

VII. Results and Discussion

A. Rate–Distortion and Tiling Cost

Fig. 2(a) and (b) show the expected rate–distortion behaviour. Averaged over the images, $Q = 75$ costs 1.16 bits per pixel (38.0 kB, a 6.9:1 reduction from the raw 262 kB) at 37.72 dB PSNR and an SSIM of 0.959; $Q = 20$ halves the rate again to 0.44 bits per pixel at 31.98 dB. The spread across images is wider than the spread across quality factors, so the image content, not the codec setting, decides whether a frame fits a given link budget.

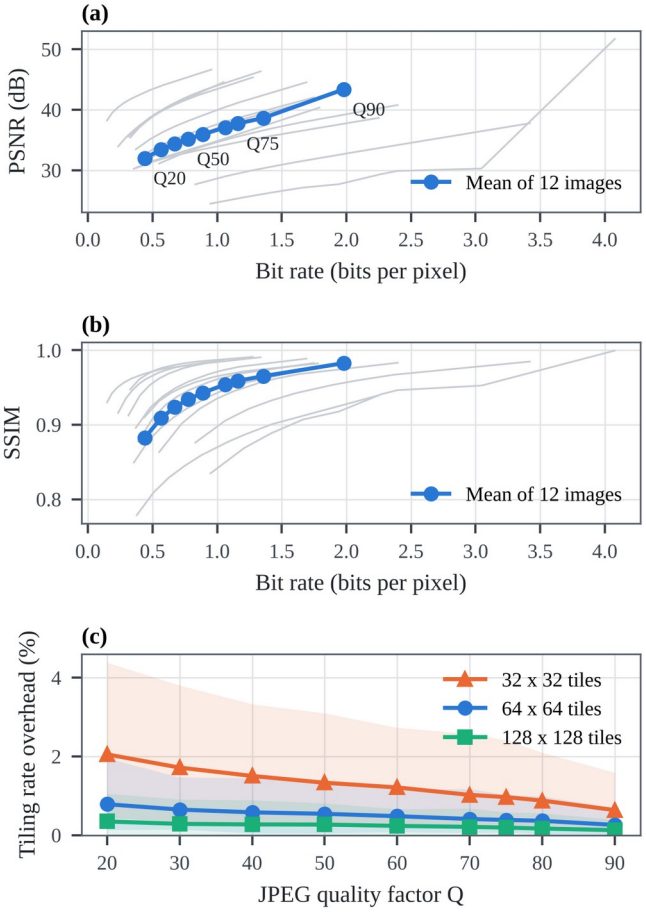


Fig. 2. Coding results over 12 images. (a) PSNR and (b) SSIM versus bit rate for Q = 20–90 (grey: individual images, blue: mean). (c) Rate overhead of tile-aligned coding relative to whole-image coding; shading spans the minimum and maximum over images.

Proposition 1 held in all 324 combinations of image, quality and tile size: tiled and whole-image streams decoded to identical pixels, so tiling has no quality cost. Its rate cost, Fig. 2(c), comes from restarting the DC predictor and padding each segment, and it shrinks as tiles grow and as quality rises. At $Q = 75$ the mean overhead was 0.97%, 0.39% and 0.19% for 32, 64 and 128-pixel tiles, with worst cases over all settings of 4.38%, 1.95% and 1.05%. On coding grounds alone, 64-pixel tiles are close to free.

B. Communication Overhead

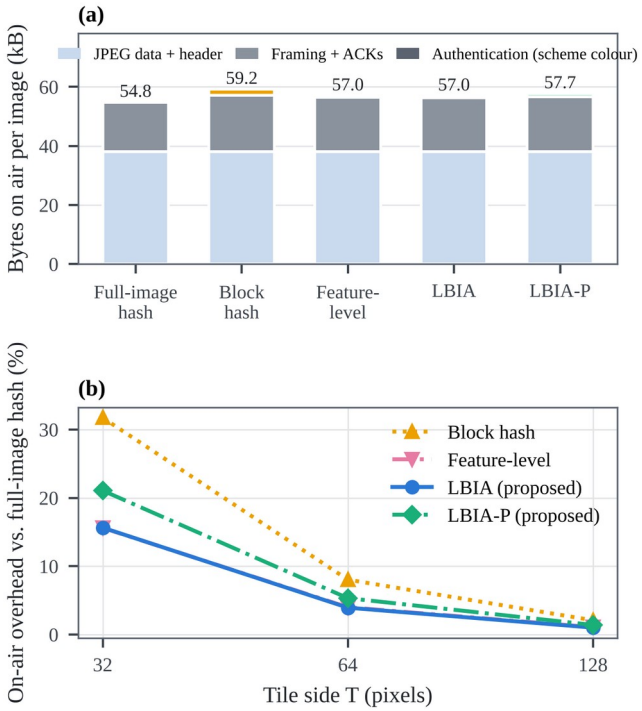


Fig. 3. (a) Bytes on air per image with link-layer ARQ and no loss at Q = 75, T = 64, split into JPEG data, framing plus ACKs, and authentication data (coloured cap). (b) On-air overhead relative to full-image hashing versus tile side; the feature-level curve coincides with LBIA.

Fig. 3(a) decomposes the loss-free cost. Framing and acknowledgements are the largest overhead for every scheme: they account for 30.5% of the 54.8 kB that full-image hashing puts on air, which is a property of 127-byte frames rather than of authentication. Against this background the authentication data are small: 64 B for FH, 576 B for LBIA, 1088 B for LBIA-P and 2112 B for block hashing. In total, LBIA needed 3.94% more bytes than FH, LBIA-P 5.33%, the feature-level scheme 4.01% and block hashing 8.04%. The raw-frame reference needed 377.0 kB, 587.78% more than FH, which quantifies the motivating problem: uncompressed transmission is not an option at these rates.

Fig. 3(b) shows a cost of tiling that the coding results in Fig. 2(c) do not reveal. Because tiles never share a frame, every tile ends in a partly filled frame. With 32-pixel tiles this framing loss raised LBIA's overhead to 15.7% and block hashing's to 31.7%, while 128-pixel tiles reduced LBIA's to 1.02%. The tile size is therefore a three-way trade-off between loss isolation, framing efficiency and, as shown below, memory. $T = 64$ is used as the nominal design point.

C. One-Shot Delivery under Loss

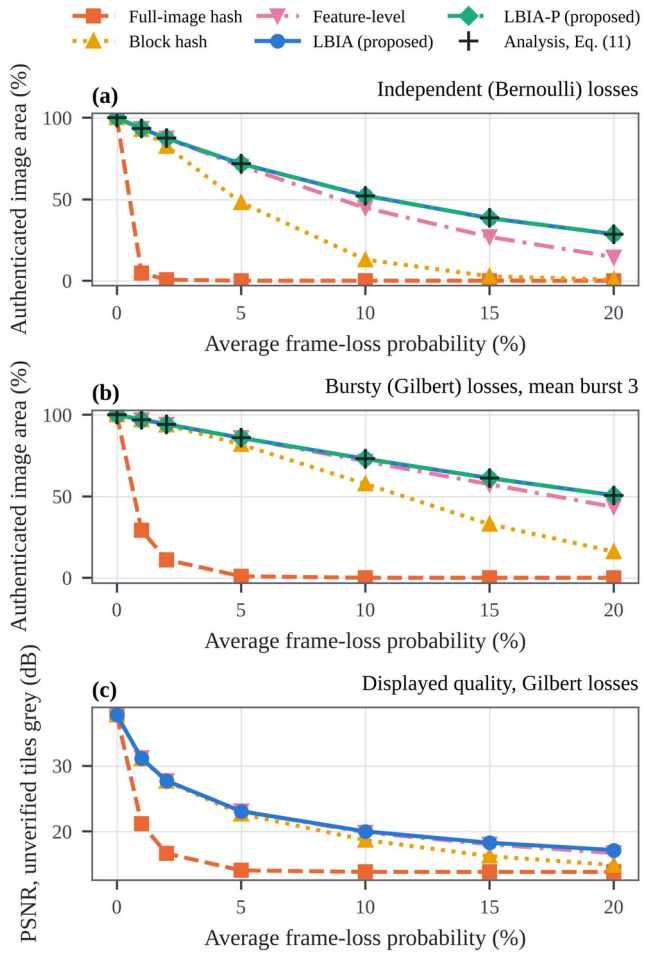


Fig. 4. One-shot delivery (no feedback), Q = 75, T = 64, mean of 12 images x 500 runs. Authenticated image area under (a) independent and (b) bursty losses; crosses are Proposition 2. (c) PSNR of the displayed image when unauthenticated tiles are shown as mid-grey (Gilbert channel).

Fig. 4 is the central result. Full-image hashing collapses at the first percent of loss: at p = 1% it authenticated 4.8% of images under independent loss and 29.2% under bursty loss, and essentially none from 5% onwards. Bursts help FH because losses cluster into fewer images. LBIA degraded gracefully, authenticating 52.3% (independent) and 73.0% (bursty) of the area at 10% loss and 50.7% at 20% bursty loss. The Monte Carlo results matched Proposition 2 to within 0.09 percentage points at every point, which also validates the channel implementation.

The tile-level baselines show why the position of the authentication data matters as much as its granularity. Block hashing and the feature-level scheme verify tiles individually, but only after their manifest arrives, and BH's 2.1-kB manifest spans 23 frames. At 10% independent loss BH authenticated 13.0% of the area even though 39.8% of the area had arrived intact; that content was received but could not be verified. The shorter FL manifest lost 7.8% of the area in the same way, and LBIA lost none, because its tags travel with the tiles. LBIA-P matched LBIA within 0.2 points. Displayed quality follows the same order, Fig. 4(c): at 5% bursty loss LBIA delivered 23.1 dB against 14.1 dB for FH, whose output is an all-grey frame.

D. Retransmission with a Deadline

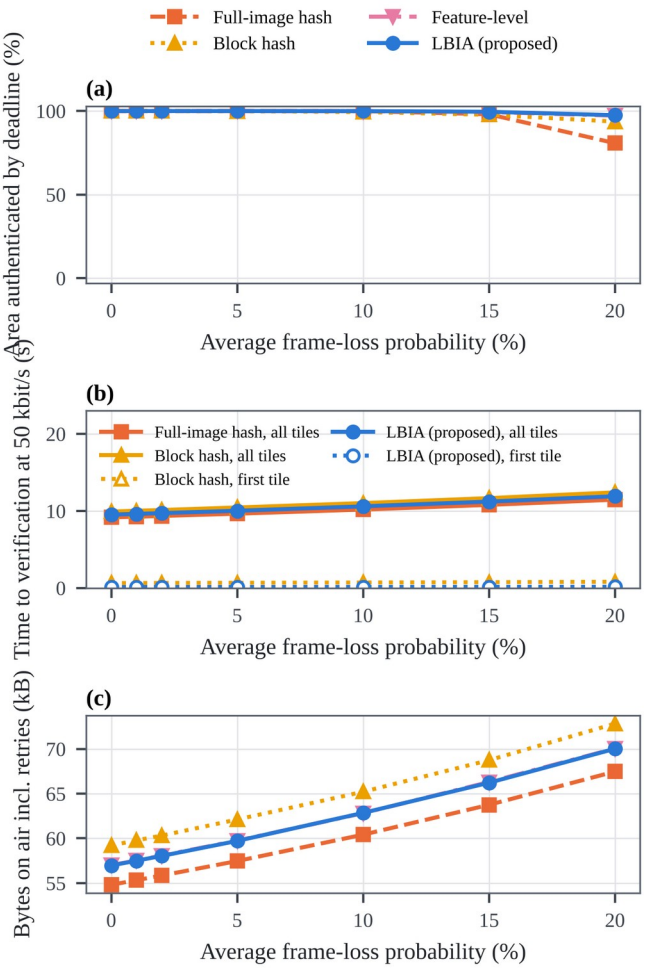


Fig. 5. Link-layer ARQ with unlimited retries, Gilbert channel, 50 kbit/s, Q = 75, T = 64. (a) Area authenticated before a deadline of 1.3 x the loss-free FH delivery time. (b) Time to the first verified tile (dotted) and to complete verification (solid). (c) Bytes on air including retransmissions.

With retransmission every scheme eventually delivers everything, so the comparison shifts to time. In Fig. 5(b), full-image hashing verified nothing until the whole image had arrived, 9.17 s after the first frame at 50 kbit/s without loss and 11.47 s at 20% loss. LBIA verified its first tile after 0.12 s and its last after 9.53 s; the feature-level and block-hash schemes needed 0.27 s and 0.64 s for the first tile because their manifests must arrive first. Without loss, complete verification took 3.9% longer for LBIA than for FH, consistent with its extra bytes. At 20 kbit/s the same pattern held, with 0.30 s against 22.3 s to first verified content.

Under the deadline, Fig. 5(a), LBIA authenticated 97.5% of the area at 20% bursty loss against 80.8% for FH. The complete-image picture is different, and less favourable to LBIA: the probability that the whole image was verified before the deadline was 80.8% for FH but only 49.6% for LBIA, because the deadline was set from FH's own delivery time and LBIA carries 3.94% more bytes. When the application needs whole frames or nothing, full-image hashing is the better choice at high loss; when partial evidence has value, LBIA is. Retransmissions raised bytes on air by 23% for FH and 23% for LBIA at 20% loss, Fig. 5(c), preserving their ranking.

E. Authentication Accuracy

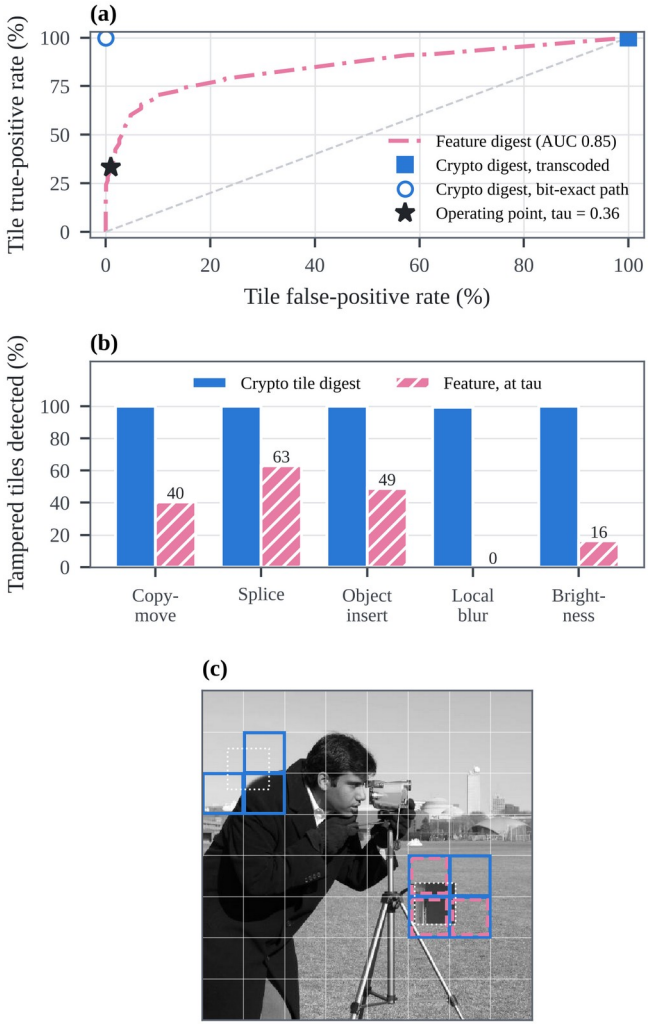


Fig. 6. Authentication accuracy at Q = 75, T = 64. (a) Tile-level ROC of the feature digest over 1454 tampered and 1536 benignly transcoded tiles, with the cryptographic digest as points. (b) Fraction of tampered tiles detected by the cryptographic digest and by the feature digest at the 1%-FPR threshold. (c) Example: a splice (lower right) and a local blur (upper left, white dotted boxes) on camera; solid blue tiles are flagged by the cryptographic digest, dashed pink tiles by the feature digest.

Cryptographic tile digests detected 100.0%, 100.0%, 100.0% and 100.0% of tampered tiles for copy–move, splice, object insertion and brightness attacks, and 99.4% for local blur; the few misses were blurred flat regions where requantisation reproduced the original coefficients, i.e. the stored content had not changed. No untouched tile was flagged, so localisation precision was 100%. The same digests flagged 100.0% of tiles after benign transcoding to $Q' = 50$ and 100.0% at $Q' = 30$, which is correct behaviour for an integrity check and the reason a feature layer is needed for archives.

The feature digest separated attacks from transcoding with an overall AUC of 0.850, Fig. 6(a), but at a threshold $\theta = 0.360$ that keeps the tile-level FPR at 0.98% it detected only 33.3% of tampered tiles. Per attack, Fig. 6(b), detection ranged from 63.1% for splicing to 16.2% for local brightness changes and was 0.0% for local blur, whose AUC of 0.479 is no better than chance: blurring preserves both the low-frequency signs and the tile mean that the digest encodes. The example in Fig. 6(c) shows this directly. At image level the threshold flagged 12.5% of benignly transcoded images, because each image offers 64 chances for a false alarm. These are negative results for the feature-level approach under the conditions tested: it can show that an archived image is broadly the one that was captured, but it cannot be relied on to expose a small or smoothing edit.

F. Computation and Verification Delay

TABLE III
SENDER AUTHENTICATION COST (MODELLED) AND LEDGER GROWTH

Scheme	SHA work (kB)	MCU time (ms) at 20 / 50 / 100 cyc/B	Ledger B/img
Full-image hash	38.1	11.9 / 29.8 / 59.5	282
Block hash	42.1	13.2 / 32.9 / 65.8	2298
Feature-level	0 (DCT only)	20.5 / 20.5 / 20.5	762
LBIA	64.5	20.2 / 50.4 / 100.8	4.41 (K = 64)
LBIA-P	64.5	40.6 / 70.9 / 121.3	4.41 (K = 64)

SHA-256 bytes processed per image incl. padding blocks; 64 MHz MCU; signature cost excluded (one per image for all schemes).

Table III gives the modelled sender cost. LBIA hashes every coded byte twice and adds $N_T - 1$ inner nodes, so at the nominal 50 cycles per byte it needs 50.4 ms per image against 29.8 ms for FH, a 69% increase; LBIA-P adds 655,360 multiply–accumulates for the feature digests and reaches 70.9 ms. These are modelled values, with the per-byte cost taken as an assumption and swept over 20–100 cycles per byte, and they omit the signature, which every scheme computes once. Relative to the seconds spent on the radio (Fig. 5), the extra computation is small, but it is not zero and it falls on the most constrained device.

At the gateway the measured host costs were 0.27 ms to hash a whole image, 8.15 ms to check all 64 tags and 5.69 ms to rebuild a 64-leaf Merkle root, with 132 µs for one Ed25519 verification. The tag and Merkle figures are dominated by Octave's per-call interpreter overhead rather than by hashing, and a compiled gateway would be faster; either way, verification delay is set by the link, not by computation.

G. Ledger Growth and Anchoring Delay

Fig. 7(a) and (b) quantify Proposition 3. A per-image record costs 282 B for FH, 762 B for the feature-level scheme and 2298 B for block hashing. With $K = 16$, LBIA's ledger share falls to 17.63 B per image at a mean anchoring delay of 25.5 s; $K = 64$ reaches 4.41 B and 97.5 s. For the assumed cluster this is 0.046 GB of ledger growth per year against 2.97 GB for FH and 24.17 GB for block hashing; writing one transaction per tile would need 21.3 transactions per second and saturate the assumed 50 tx/s ledger at 46 cameras. All inclusion proofs verified and a proof for a modified root was rejected. Because verification at the gateway does not wait for the ledger, a delay of minutes affects only when public auditability begins, so K can be chosen from the ledger budget with (14).

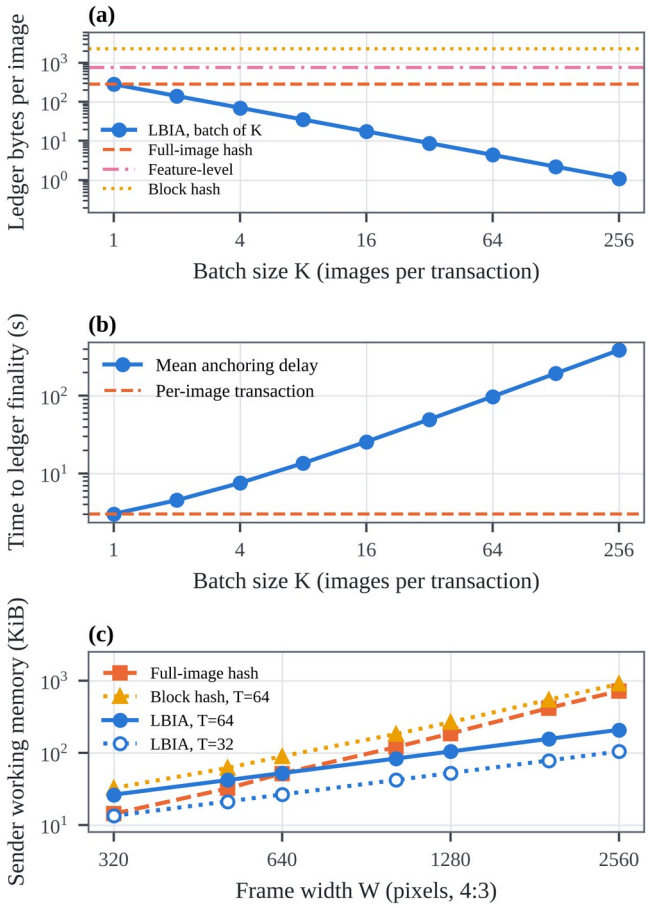


Fig. 7. (a) Ledger bytes per image and (b) mean time to ledger finality versus batch size K for 20 cameras at one frame per minute (assumed ledger parameters, Table II). (c) Sender working memory versus frame width for 4:3 frames at 1.16 bits per pixel.

H. Memory and Storage

Fig. 7(c) and Proposition 4 give a result that does not favour LBIA at small frame sizes. For 64-pixel tiles LBIA's 64-line strip buffer exceeds the compressed image itself below a width of about 696 pixels (Proposition 4 predicts 685), so at 512 x 384 it needs 41.7 KiB against 32.0 KiB for full-image hashing. Above the crossover the picture reverses: at 1920 x 1472 LBIA needs 155.3 KiB against 415.4 KiB, because the full-image scheme must retain the whole bitstream until the image is verified. With 32-pixel tiles the crossover falls to 320 pixels and LBIA needs 21.1 KiB at 512 pixels, at the framing cost shown in Fig. 3(b). Block hashing needs the most memory at every size because its manifest precedes the tiles.

Store-and-forward capacity is governed by compression, not by authentication. A 4-MiB flash buffer holds 15 raw frames but 110 JPEG frames at $Q = 75$ with FH and 108 with LBIA (104 with BH); at $Q = 30$ the figures rise to 224 and 217. Authentication metadata therefore cost about two frames in a hundred, while lowering Q from 75 to 30 doubles capacity at a loss of 4.3 dB PSNR.

VIII. Comparative Analysis

TABLE IV
Comparative Analysis of the Simulated Schemes

Metric (Q = 75, T = 64)	FH	BH	FL	LBIA	LBIA-P
On air, no loss (kB)	54.8	59.2	57.0	57.0	57.7
Overhead vs. FH (%)	0	8.04	4.01	3.94	5.33
Area, 1-shot, 10% burst (%)	0.0	57.6	71.5	73.0	73.1
Area, 1-shot, 10% indep. (%)	0.0	13.0	44.7	52.3	52.1
First verified tile (s)	9.17	0.64	0.27	0.12	0.12
Whole image by deadline, 20% (%)	80.8	17.0	46.5	49.6	48.9
Tamper localisation	None	Exact	Weak	Exact	Exact
Survives transcoding	No	No	Yes	No	Partly
Ledger bytes / image	282	2298	762	4.41	4.41
Memory, 512 px (KiB)	32.0	61.5	≈ BH	41.7	42.1
Memory, 1920 px (KiB)	415.4	542.0	≈ BH	155.3	160.7
Sender auth. time (ms)	29.8	32.9	20.5	50.4	70.9

FH: full-image hash; BH: block hash; FL: feature-level. Gilbert channel and 50 kbit/s unless stated; ledger for LBIA at K = 64; FL memory within 3% of BH; sender time at 50 cycles/B (modelled). LBIA-P survives transcoding through its feature layer only.

Table IV collects the measured trade-offs. No scheme dominates. Full-image hashing is the cheapest on air, in sender computation and in memory for small frames, and it completes whole images most reliably under a tight deadline; it fails completely once frames are lost without feedback and gives no localisation. Block hashing restores localisation at the highest cost in bytes, memory and ledger space, and its manifest makes it fragile under one-shot delivery. The feature-level scheme is compact and survives transcoding but detects at most two thirds of tampered tiles at a 1% false-positive rate. LBIA combines exact tile localisation, the highest authenticated area under loss and the lowest ledger growth, at 3.94% more bytes, about 69% more sender hashing, and more memory than FH below about 700-pixel frames.

Against the literature in Table I, LBIA differs in kind rather than degree. Per-image or per-frame ledger records [8], [11] are its $K = 1$ special case, and per-block signatures [9] correspond to the block-hash baseline. Perceptual-hash ledgers [12], [11] correspond to the feature-level baseline, whose measured weakness against blur and brightness edits is consistent with the robustness gaps reported for traditional hashes [20]. A direct numerical comparison with those works is not meaningful because none reports bytes on air, loss behaviour or ledger growth under a common setting; the baselines here were implemented to make that comparison possible.

IX. Key Outcomes and Limitations

A. Key Outcomes

Novelty. Authentication is attached to the smallest unit that a lossy link delivers intact, the JPEG tile, through inline 64-bit

tags, while public verifiability is kept by one signature per image and one ledger transaction per K images.

Engineering improvement. Under one-shot delivery LBIA authenticated 73.0% of the image area at 10% bursty loss where full-image hashing authenticated none, and under ARQ it produced the first verified content about 73 times sooner at 5% bursty loss, for 3.94% more bytes on air and 4.41 ledger bytes per image at $K = 64$.

Practical applications. Remote environmental stations and wildlife cameras gain partial but provable images over unconfirmed long-range links; industrial sensors gain tile-level localisation of tampering in archived inspection images; disaster-response equipment gains usable, verifiable imagery while links are degraded, with ledger load that grows slowly with the number of cameras.

Industrial and social significance. Evidence from field cameras supports enforcement, insurance and relief decisions; making partial images verifiable means that degraded links reduce the amount of evidence rather than eliminate it.

B. Limitations

All results are simulations; no radio, MCU or ledger hardware was used, and the MCU timings rest on assumed cycle costs. The ledger parameters are assumptions, and real permissioned ledgers carry larger transaction envelopes. The images are generic grey-scale test images rather than field imagery, and colour and chroma subsampling were not modelled. The feature digest is unkeyed, so an adversary who knows it could search for colliding edits; keyed digests [18] were not evaluated. Key distribution, revocation and compromise of the camera were out of scope. The loss channel is a two-state model with fixed burst length; measured traces would test robustness to other loss structures. Finally, the deadline result shows that LBIA is not the best choice when only complete images are useful.

X. Conclusion

This paper examined image authentication for cameras on constrained, lossy links and proposed LBIA, which tags each independently decodable JPEG tile, signs one Merkle root per image and anchors image roots in batches. Simulation over 12 images showed that tile-aligned coding is lossless in quality and costs 0.39% in rate at 64-pixel tiles; that inline tags avoid the received-but-unverifiable content that manifest-first designs suffer; that LBIA keeps most of an image verifiable at loss rates at which full-image hashing verifies nothing; and that batching reduces ledger growth by a factor equal to the batch size at the price of a predictable anchoring delay. The same experiments identified where LBIA is not preferable: when only complete images matter under tight deadlines, and for small frames, where its strip buffer exceeds a full-image buffer. The feature-level digest was found useful for archival provenance but unreliable against small or smoothing edits. Future work will validate the scheme on LoRa and IEEE 802.15.4 hardware with field imagery, measure MCU cost directly, evaluate keyed feature digests and colour coding, and integrate key management and revocation.

Data Availability

The MATLAB-language source, result logs, figure scripts and a README describing parameters, run order and seed are provided as a supplementary reproducibility bundle.

Acknowledgment

The authors thank the Research and Development Division of Tekstein Solutions Private Limited for computing support. [Funding and further acknowledgments to be confirmed by the authors.]

References

[1] M. S. Norouzzadeh *et al.*, “Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning,” *Proc. Natl. Acad. Sci. USA*, vol. 115, no. 25, pp. E5716–E5725, 2018.

[2] P. Lech, B. Marciniak, and K. Okarma, “A low-cost energy-efficient IoT camera trap network for remote forest surveillance,” *Electronics*, vol. 14, no. 21, Art. no. 4266, 2025, doi: 10.3390/electronics14214266.

[3] A. H. Jebril, A. Sali, A. Ismail, and M. F. A. Rasid, “Overcoming limitations of LoRa physical layer in image transmission,” *Sensors*, vol. 18, no. 10, Art. no. 3257, 2018, doi: 10.3390/s18103257.

[4] Q. Guo, F. Yang, and J. Wei, “Experimental evaluation of the packet reception performance of LoRa,” *Sensors*, vol. 21, no. 4, Art. no. 1071, 2021, doi: 10.3390/s21041071.

[5] *IEEE Standard for Low-Rate Wireless Networks*, IEEE Std 802.15.4-2020, 2020.

[6] T. Xie, J. Kay, T. Haucke, and S. Beery, “Saliency-guided deployment-adaptive compression for wildlife camera traps,” presented at the NeurIPS 2025 Workshop on Tackling Climate Change with Machine Learning, 2025.

[7] K. Christidis and M. Devetsikiotis, “Blockchains and smart contracts for the Internet of Things,” *IEEE Access*, vol. 4, pp. 2292–2303, 2016.

[8] D. Danko, S. Mercan, M. Cebe, and K. Akkaya, “Assuring the integrity of videos from wireless-based IoT devices using blockchain,” in *Proc. IEEE 16th Int. Conf. Mobile Ad Hoc Smart Syst. Workshops (MASSW)*, Monterey, CA, USA, 2019, pp. 48–52, doi: 10.1109/MASSW.2019.00016.

[9] Y. Li, Y. Tu, J. Lu, and Y. Wang, “A security transmission and storage solution about sensing image for blockchain in the Internet of Things,” *Sensors*, vol. 20, no. 3, Art. no. 916, 2020, doi: 10.3390/s20030916.

[10] Y.-C. Chen, Y.-P. Chou, and Y.-C. Chou, “An image authentication scheme using Merkle tree mechanisms,” *Future Internet*, vol. 11, no. 7, Art. no. 149, 2019, doi: 10.3390/fi11070149.

[11] D. Xu, N. Ren, and C. Zhu, “Integrity authentication based on blockchain and perceptual hash for remote-sensing imagery,” *Remote Sens.*, vol. 15, no. 19, Art. no. 4860, 2023, doi: 10.3390/rs15194860.

[12] C. C. White and M. Paul, “A practical blockchain framework using image hashing for image authentication,” arXiv:2004.06860, 2020.

[13] D. Stefanescu, L. Montalvillo, P. Galán-García, J. Unzilla, and A. Urbieta, “A systematic literature review of lightweight blockchain for IoT,” *IEEE Access*, vol. 10, pp. 123138–123159, 2022.

[14] M. Khan, F. den Hartog, and J. Hu, “A survey and ontology of blockchain consensus algorithms for resource-constrained IoT systems,” *Sensors*, vol. 22, no. 21, Art. no. 8188, 2022, doi: 10.3390/s22218188.

[15] P. W. Khan and Y. Byun, “A blockchain-based secure image encryption scheme for the industrial Internet of Things,” *Entropy*, vol. 22, no. 2, Art. no. 175, 2020, doi: 10.3390/e22020175.

[16] K. Saito, “Proof of authenticity of general IoT information with tamper-evident sensors and blockchain,” arXiv:2512.18560, Dec. 2025.

[17] C.-Y. Lin and S.-F. Chang, “A robust image authentication method distinguishing JPEG compression from malicious manipulation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 153–168, Feb. 2001.

[18] A. Swaminathan, Y. Mao, and M. Wu, “Robust and secure image hashing,” *IEEE Trans. Inf. Forensics Security*, vol. 1, no. 2, pp. 215–230, Jun. 2006.

[19] C.-M. Pun, C.-P. Yan, and X.-C. Yuan, “Robust image hashing using progressive feature selection for tampering detection,” *Multimedia Tools Appl.*, vol. 77, no. 10, pp. 11609–11633, 2018, doi: 10.1007/s11042-017-4809-4.

[20] R. Biswas and P. Blanco-Medina, “State of the art: Image hashing,” arXiv:2108.11794, 2021.

[21] P. Aberna and L. Agilandeeswari, “Optimal semi-fragile watermarking based on maximum entropy random walk and Swin Transformer for tamper localization,” *IEEE Access*, vol. 12, pp. 37757–37781, 2024, doi: 10.1109/ACCESS.2024.3370411.

[22] T. Pal and S. Das Bit, “An improved low-overhead secure image compression over wireless multimedia sensor network,” *Wireless Pers. Commun.*, vol. 132, no. 2, pp. 1049–1081, 2023, doi: 10.1007/s11277-023-10648-1.

[23] M. Castro and B. Liskov, “Practical Byzantine fault tolerance,” in *Proc. 3rd USENIX Symp. Operating Syst. Design Implement. (OSDI)*, New Orleans, LA, USA, 1999, pp. 173–186.

[24] E. Androulaki *et al.*, “Hyperledger Fabric: A distributed operating system for permissioned blockchains,” in *Proc. 13th EuroSys Conf.*, Porto, Portugal, 2018, Art. no. 30.

[25] A. Dorri, S. S. Kanhere, R. Jurdak, and P. Gauravaram, “LSB: A lightweight scalable blockchain for IoT security and anonymity,” *J. Parallel Distrib. Comput.*, vol. 134, pp. 180–197, 2019.

[26] B. Laurie, A. Langley, and E. Kasper, “Certificate transparency,” IETF RFC 6962, Jun. 2013.

[27] *Information Technology—Digital Compression and Coding of Continuous-Tone Still Images—Requirements and Guidelines*, ITU-T Rec. T.81, International Telecommunication Union, Geneva, Switzerland, 1992.

[28] H. Krawczyk, M. Bellare, and R. Canetti, “HMAC: Keyed-hashing for message authentication,” IETF RFC 2104, Feb. 1997.

[29] National Institute of Standards and Technology, “Secure hash standard (SHS),” FIPS PUB 180-4, Aug. 2015.

[30] R. C. Merkle, “A digital signature based on a conventional encryption function,” in *Advances in Cryptology—CRYPTO ’87*, LNCS vol. 293, C. Pomerance, Ed. Berlin, Germany: Springer, 1988, pp. 369–378.

[31] S. Josefsson and I. Liusvaara, “Edwards-curve digital signature algorithm (EdDSA),” IETF RFC 8032, Jan. 2017.

[32] E. N. Gilbert, “Capacity of a burst-noise channel,” *Bell Syst. Tech. J.*, vol. 39, no. 5, pp. 1253–1265, Sep. 1960.

[33] E. O. Elliott, “Estimates of error rates for codes on burst-noise channels,” *Bell Syst. Tech. J.*, vol. 42, no. 5, pp. 1977–1997, Sep. 1963.

[34] Q. Dang, “Recommendation for applications using approved hash algorithms,” NIST SP 800-107 Rev. 1, Aug. 2012.

[35] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: From error visibility to structural similarity,” *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[36] J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring, *GNU Octave Version 8 Manual: A High-Level Interactive Language for Numerical Computations*, 2023. [Online]. Available: https://octave.org