Tamper-Evident Industrial Visual Inspection Using Blockchain and Feature-Level Hashing

A calibrated block-feature fingerprint anchored in a hash-chained ledger

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

***Abstract*—Images captured by automated optical inspection stations are the evidence behind accept and reject decisions in manufacturing, yet they are stored as ordinary files that can be replaced or retouched after the fact. A cryptographic digest exposes any change, but it cannot distinguish a forged defect region from the routine brightness adjustment or recompression that an image undergoes when it is archived, viewed or exported, so in practice it raises alarms on nearly every legitimate copy. This paper presents a tamper-evident inspection record that keeps an exact digest for strict audits and adds a block-level feature fingerprint for tolerant verification. Each image is divided into 64 blocks, and every block is summarised by normalised intensity, gradient energy, edge density and a contrast-thresholded local binary pattern histogram. The quantised fingerprint and the digest form a record whose hash is committed to an append-only hash chain through Merkle batching. At audit time a trimmed regression removes globally uniform feature changes, each feature group is scaled by its own benign variability, and blocks whose fused score exceeds a calibrated threshold are reported as tampered. In a simulation with 200 synthetic steel-surface images, 2400 benign images and 2342 forgeries, the scheme detected 98.2 percent of defect insertions, removals and substitutions at a 0.67 percent benign false-alarm rate, against 34.8 percent for a global perceptual hash. Tampered blocks were located with 94.1 percent precision. The method is not robust to small translations or blurring, which caused false alarms on most images and define the main direction for future work.**

***Index Terms*—Image authentication, perceptual hashing, tamper localisation, blockchain, Merkle tree, local binary patterns, surface defect inspection, quality-control audit trail.**

I. INTRODUCTION

Automated optical inspection has become the default quality gate for rolled metal, castings, glass, printed circuits and textiles, and its output is increasingly consumed by people who were not present at the line: customers who audit supplier quality, regulators in safety-critical sectors and engineers investigating field failures months later [1]. In each case the stored inspection image is treated as evidence. Regulations for electronic records in regulated manufacturing require such evidence to be trustworthy and protected against undetected alteration [2]. Image editing, however, has become routine, and a removed scratch or a pasted pit is difficult to detect visually once the image has been re-encoded [3].

The obvious safeguard is to store a cryptographic digest such as SHA-256 [4] with each image and to anchor the digests in a time-stamped, append-only structure [5], [6]; blockchain-based traceability has been proposed widely for Industry 4.0 for exactly this purpose [7]. A digest, however, certifies bytes rather than content. An inspection image that is legitimately brightened for display, recompressed by an archive server or exported at a different quality factor no longer matches its digest, and the verifier cannot tell such a copy from a forgery. Perceptual hashes [8], [9] tolerate those changes, but a single global hash summarises the whole image in a few dozen bits, so a defect occupying one or two percent of the frame barely moves it. Global perceptual hashes are also known to be manipulable by an adversary [10], [11].

The gap addressed here is therefore a verification primitive that is (i) tolerant to the photometric and compression changes that inspection images routinely undergo, (ii) sensitive to small, local edits of the defect regions on which the quality decision rests, (iii) able to say where the image was changed and (iv) cheap enough to anchor in a ledger for every image a line produces. The contributions of this paper are as follows.

- A two-tier inspection record that combines an exact pixel digest with a 832-byte block-feature fingerprint (BFF), committed to a hash-chained ledger through Merkle batches.
- A verification rule that removes spatially uniform feature changes by trimmed regression and fuses feature groups after scaling each by its own calibrated benign variability, with a threshold set for a target false-alarm rate.
- A controlled simulation with ground-truth edit masks that compares the scheme with SHA-256 and a global DCT perceptual hash, and an ablation that identifies which design choices matter, including a failed first design.
- An explicit account of the operating envelope, including the geometric and blur conditions under which the method fails.

II. RELATED WORK

Cryptographic integrity and ledgers. Linked time-stamping [5] and Merkle trees [17] allow many digests to be committed with a single root, and public or permissioned blockchains [6], [16] distribute that commitment so that no single party can rewrite it. These mechanisms prove that a record existed and

was not altered, but they inherit the brittleness of the digest stored in the record.

Perceptual hashing. Robust image hashes map visually similar images to nearby codes [8], [9]. White and Paul [15] and Bhowmik and Feng [14] combined such hashes or watermarks with a blockchain for media authentication. Global hashes trade sensitivity for robustness, and recent work shows that they can be driven to collisions or evasions deliberately [10], [11].

Passive forensics. Copy-move and splicing detectors [12], [13] look for internal inconsistencies without any reference, which avoids enrolment but provides no guarantee and is computationally heavy. Surveys of media forensics note that such detectors degrade under recompression [3].

Inspection features. Local binary patterns (LBP) [18] and their completed variants [19] are standard texture descriptors for steel-surface defect classification, and edge maps [20] are a common pre-processing step [1]. They have been used to recognise defects, not to certify that a stored image is the one that was inspected. Table 1 summarises the positioning.

TABLE 1 POSITIONING OF THE PROPOSED RECORD

Approach	Benign tolerant	Localises	Tamper-evident record
Digest + ledger [4]–[6]	No	No	Yes
Global perceptual hash [8], [9]	Yes	No	Optional [15]
Passive forensics [12], [13]	Partly	Yes	No
Defect texture features [18], [19]	—	—	No
Proposed BFF + ledger	Photometric only	Block level	Yes

III. PROBLEM FORMULATION

An inspection station produces an 8-bit image I of size $H \times W$ together with a decision. At enrolment the station computes a record and commits it to a ledger that the verifier trusts to be append-only. At audit time the verifier receives an image I' claimed to be I. I' is *benign* if it differs from I only by global photometric changes (offset, gain, gamma), JPEG recompression or mild sensor noise. I' is *tampered* if a region was edited to insert or hide a defect, or if I' is a different image. The verifier must decide which holds, keep the false-alarm rate on benign copies at a target α (1 percent here) and, when tampering is declared, report the affected region. The adversary is assumed to edit pixels with ordinary tools and may recompress afterwards, but does not know the fingerprint design in detail; feature-aware adaptive attacks are outside this study. Geometric changes (crop, shift, rotation) are not considered benign in the design and are tested only as a stress condition.

IV. PROPOSED METHOD

Fig. 1 shows the enrolment and verification paths. Both apply the same feature extraction, so the description below applies to I and I' alike.

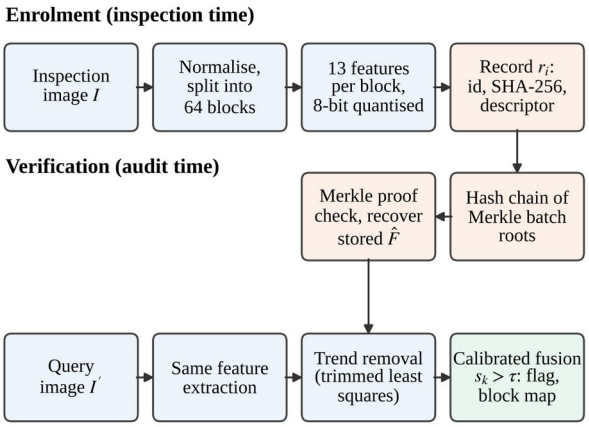


Fig. 1. Enrolment and verification architecture. Shaded boxes on the right are ledger-side components.

A. Block Features

The image is first normalised to zero mean and unit variance,

$$z(x, y) = [I(x, y) - \mu_I] / \sigma_I, \quad (1)$$

which cancels any global offset and gain exactly. The normalised image is divided into $K = 64$ non-overlapping blocks of 32×32 pixels. For block k the fingerprint row is

$$\mathbf{f}_k = [\, m_k, g_k / g_0, e_k, \mathbf{h}_k \,] \in \mathbb{R}^{13}, \quad (2)$$

where m_k is the block mean of z, g_k the mean Sobel gradient magnitude of z smoothed with a Gaussian of $\sigma = 1$ pixel, g_0 the image-wide median gradient, e_k the fraction of block pixels marked by a Canny detector [20] with hysteresis thresholds at the 80th and 95th gradient percentiles, and $\mathbf{h}_k$ a 10-bin histogram of rotation-invariant uniform LBP codes [18]. Percentile thresholds and the ratio g_k/g_0 make the gradient and edge terms independent of global contrast. The LBP code at pixel $\mathbf{p}$ uses eight neighbours $\mathbf{p}_n$ at radius 2 and a contrast margin t,

$$s_n = 1[\, z_s(\mathbf{p}_n) - z_s(\mathbf{p}) > t \,], \quad L(\mathbf{p}) = \Sigma_n s_n \text{ if } U \le 2, \text{ else } 9, \quad (3)$$

where z_s is the smoothed image and U the number of 0/1 transitions around the circle. The margin $t = 0.1$ suppresses code flips caused by quantisation noise in flat regions, a known weakness of the zero-threshold operator on low-contrast metal surfaces. The four groups are intensity (m), gradient (g), edge (e) and texture ($\mathbf{h}$).

B. Ledger Record

Each feature is quantised to 8 bits over a fixed range, giving an 832-byte descriptor $Q(\mathbf{F})$. Image i yields the record and leaf

$$r_i = \mathrm{id}_i \,\|\, \mathrm{SHA256}(I_i) \,\|\, Q(\mathbf{F}_i), \quad \lambda_i = \mathrm{SHA256}(r_i). \quad (4)$$

Leaves are grouped into batches; block j of the chain stores the Merkle root M_j of its batch [17] and the header hash

$$H_j = \mathrm{SHA256}(H_{j-1} \,\|\, M_j \,\|\, j). \quad (5)$$

Only the 32-byte leaf needs to be on-chain; the record itself may be held off-chain because any change to it changes λ_i. The exact digest inside r_i supports a strict mode in which only bit-identical images are accepted.

C. Verification Rule

The verifier first checks the Merkle proof of r_i against the chain, then computes $\mathbf{F}'$ for I' and the change $\Delta\mathbf{F} = \mathbf{F}' - \hat{\mathbf{F}}$, where $\hat{\mathbf{F}}$ is the de-quantised stored descriptor. Benign operations change

all blocks in a similar way, whereas edits change a few. For each feature c the verifier therefore fits a global affine trend by least squares over a trimmed block set T,

$$(a_c, b_c) = \arg\min \Sigma_{k\in T} (\Delta F_{k,c} - a - b\,\hat{F}_{k,c})^2, \quad (6)$$

with T re-estimated twice as the 80 percent of blocks with the smallest residuals, and keeps the residual $\rho_{k,c} = \Delta F_{k,c} - a_c - b_c\hat{F}_{k,c}$. The group distance and block score are

$$d_{k,g} = \Sigma_{c\in g} |\rho_{k,c}| / \sigma_c, \quad s_k = \max_g d_{k,g} / q_g, \quad (7)$$

where σ_c is the spread of feature c across blocks of the calibration images and q_g is the 99th percentile of $d_{k,g}$ over all blocks of benign calibration variants. The image is declared tampered and the block set Ω is reported when

$$\max_k s_k > \tau, \quad \Omega = \{ k : s_k > \tau \}, \quad (8)$$

with τ the 99th percentile of $\max_k s_k$ over the same benign calibration set, which targets $\alpha = 1$ percent.

Remark. If a benign operation changes feature c of every block by a common affine map plus noise, Eq. 6 recovers the map and ρ reduces to the noise, independently of the map parameters. An edit confined to fewer than 20 percent of the blocks is excluded from T whenever its residuals exceed those of the untouched blocks. This is a heuristic property rather than a guarantee: gamma changes are not affine in the gradient and LBP features, and a strongly non-stationary texture weakens the common-trend assumption. Section VI quantifies how much the step contributes.

Algorithm and complexity. Enrolment: (1) normalise; (2) extract 64×13 features; (3) quantise and form r_i; (4) append λ_i to the current batch and commit M_j and H_j when the batch is full. Verification: (1) check the Merkle path and header chain; (2) extract $\mathbf{F}'$; (3) apply Eq. 6 to Eq. 8. Feature extraction is O(HW), the regression and fusion are O(KD) with $D = 13$, and a Merkle proof costs O(log n_b) hashes for batch size n_b. The descriptor occupies KD bytes per image.

V. Simulation Setup

Real inspection archives with authenticated originals and forensic ground truth are not publicly available, so the study uses synthetic images whose edit masks are known exactly. Surfaces are generated as anisotropically filtered Gaussian noise imitating rolling texture, with a random illumination ramp and fine sensor noise. Up to three defects per image are drawn from four types: pits, bright or dark scratches, random-walk cracks and low-contrast stains. Table 2 lists the protocol. Edits are of two kinds: *insertion* stamps a new defect of a given size, and *removal* hides an existing defect with a feathered clone patch whose mean is matched to the surroundings, as a careful forger would do. Removal is applied only to images that contain a defect. Ground-truth blocks are those with at least 5 percent of pixels changed by more than two grey levels.

Two baselines are evaluated on identical inputs: SHA-256 over raw pixels, and a 63-bit global DCT perceptual hash (8×8 low-frequency coefficients of a 32×32 thumbnail, median threshold) whose Hamming threshold is calibrated to the same 99th-percentile rule. All code was written in Python 3 with NumPy, SciPy and scikit-image and run on a two-vCPU 2.8 GHz Intel Xeon virtual machine with a fixed global seed (2026). The ledger is a single-node simulation of the data structures; it includes no consensus or network delay.

TABLE 2 SIMULATION PARAMETERS

Item	Value
Images (calibration / test)	100 / 200, 256×256, 8 bit
Blocks, features	64 blocks of 32×32, 13 features
Benign variants (11 per image)	Offset ±15, gain 0.85/1.15, gamma 0.8/1.25, JPEG Q90/75/50, noise $\sigma = 2$, offset 10 + JPEG Q75
Stress variants	2-pixel horizontal shift, Gaussian blur $\sigma = 0.8$
Edits per image	Insertion and removal at 12, 16, 24, 32, 48 px; each at 24 px then re-saved; substitution
Calibration targets	q_g, τ at 99th percentile ($\alpha = 1$ %)
Ledger	SHA-256, Merkle batches of 50 (test) and 256 (scaling)

VI. Results and Discussion

A. Image-Level Detection

Table 3 reports the fraction of images flagged as tampered under each condition. SHA-256 flags every modified image, which is correct for edits and useless for benign copies: over the 2400 benign and unmodified images its false-alarm rate is 91.7 percent, and the 8.3 percent that pass are simply the unmodified originals. The global perceptual hash behaves in the opposite way. Its benign false-alarm rate of 0.83 percent meets the target, but it detects only 34.8 percent of edits, and only 14.3 percent of defect removals, because a clone patch with matched mean leaves the low-frequency DCT coefficients almost unchanged. The proposed fingerprint detects 98.2 ± 0.5 percent of edits at a benign false-alarm rate of 0.67 ± 0.33 percent (95 percent intervals), a balanced accuracy of 98.8 percent against 67.0 percent for the perceptual hash. Re-saving a forged image with a brightness change and JPEG Q75 did not hide the edit (100.0 and 97.5 percent detection).

TABLE 3 FRACTION OF IMAGES FLAGGED AS TAMPERED (%)

Condition	n	SHA-256	pHash	BFF
Unmodified	200	0.0	0.0	0.0
Benign photometric (6 types)	1200	100.0	0.9	0.0
Benign JPEG Q90/75/50	600	100.0	1.2	2.5
Benign noise / combined	400	100.0	0.5	0.3
Stress: 2-pixel shift	200	100.0	11.5	79.5
Stress: Gaussian blur 0.8	200	100.0	0.5	83.0
Defect insertion (all sizes)	1000	100.0	38.7	98.3
Defect removal (all sizes)	785	100.0	14.3	97.3
Insertion then re-save	200	100.0	50.5	100.0
Removal then re-save	157	100.0	8.9	97.5
Image substitution	200	100.0	100.0	100.0

Benign false alarms of the proposed method are concentrated in JPEG Q50, which was flagged in 7.0 percent of images; all other benign variants stayed at or below 0.5 percent. Strong compression removes fine texture unevenly across blocks, which the global-trend model cannot absorb. The stress rows expose a clear limitation: a 2-pixel shift and a Gaussian blur of $\sigma = 0.8$ were flagged in 79.5 and 83.0 percent of images, whereas the perceptual hash, which sees only coarse structure, stayed at 11.5 and 0.5 percent. The block grid and the gradient features are inherently sensitive to these changes. A deployment must therefore either forbid resampling in the archive chain or add registration before comparison.

B. Effect of Edit Size and Defect Type

Fig. 2 shows detection against edit size. The perceptual hash improves with size but stays below 60.0 percent for insertions and 34.4 percent for removals even at 48 pixels. The proposed method detects 93.0 percent of 12-pixel insertions, about a quarter of one block, and reaches 100.0 percent at 24 pixels. Removals are slightly harder at small sizes (94.3 percent at 16 pixels) because the hidden defect is itself small and the clone patch reproduces the surrounding texture. Across defect types, pits and stains were detected in every insertion trial, while thin scratches and cracks were missed in 3.5 and 3.4 percent of insertions; removals of scratches, cracks and stains were missed in 2.9, 3.8 and 3.8 percent of cases, and pit removals never. Thin linear defects change few pixels per block, so the block mean barely moves and detection relies on the gradient group.

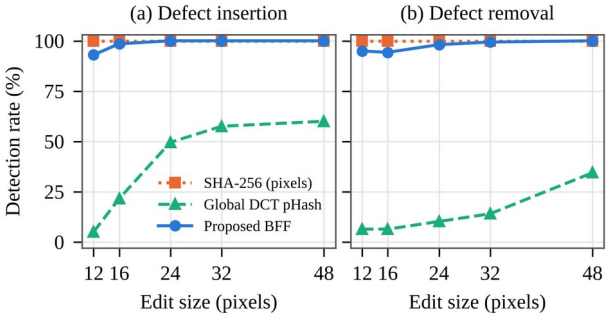


Fig. 2. Detection rate versus edit size for (a) defect insertion and (b) defect removal. SHA-256 is at 100 percent for all sizes.

C. Localisation

Over 2028 edited images with ground-truth masks (on average 2.6 affected blocks), the flagged block set had a precision of 94.1 percent and a recall of 79.0 percent (F1 85.9 percent). Every image declared tampered had at least one flagged block inside the true edit region, so a reported location is always a correct pointer to the edit, although the edges of an edit that only grazes a block are often missed. Fig. 3 illustrates the case of a removed pit: the JPEG Q50 copy is accepted while the forged copy is flagged in the edited blocks only.

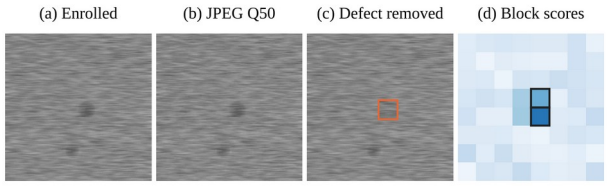


Fig. 3. Example: (a) enrolled image with a pit, (b) benign JPEG Q50 copy (accepted), (c) pit hidden by a clone patch (box), (d) block score map with flagged blocks outlined.

D. Ablation

Table 4 isolates each design choice on a subset of 100 test images, with every variant recalibrated to the same 1 percent target. The first implementation summed the raw group distances without per-group scaling; at the calibrated threshold it detected 0.0 percent of edits. The LBP histogram computed with a zero threshold changed by tens of standard units under JPEG recompression, which inflated the threshold until no edit could cross it. Scaling each group by its benign 99th percentile alone raised detection to 94.7 percent with the same features, so calibration, not the choice of features, was the decisive change. Conversely, the redesigned features with trend removal but without calibration reached only 56.1 percent. The contrast-thresholded LBP, smoothed gradients and the trend removal of Eq. 6 together added a further 4.1 points. Max and mean fusion performed alike. Within the final design, intensity carries most of the signal: removing it lowered detection to 83.2 percent, whereas removing the edge or LBP group changed it by less than one point and texture alone detected only 29.9 percent. On these synthetic surfaces the texture descriptor therefore adds little; whether it helps on real surfaces with subtler defects remains to be shown.

TABLE 4 ABLATION ON 100 TEST IMAGES (1100 BENIGN, 531 EDITED)

Variant	FPR (%)	TPR (%)
v0 features, raw sum (first design)	1.5	0.0
v1 features + trend removal, raw sum	1.1	56.1
v0 features, calibrated mean	0.9	94.7
v1, no trend removal, calibrated mean	1.0	98.5
v1, no trend removal, calibrated max	1.1	98.9
v1 + trend removal, calibrated mean	1.3	99.1
Proposed (v1 + trend removal, max)	1.2	98.9
without intensity	1.0	83.2
without gradient	0.7	95.1
without edge density	0.8	99.1
without LBP texture	1.2	98.9
Intensity only	1.6	95.1
Texture (LBP) only	1.2	29.9

E. Latency, Storage and Ledger Cost

Table 5 summarises the cost per image. Verification with the proposed method took 25.3 ms on average, dominated by feature extraction (21.6 ms); the Merkle and chain check took 47 µs. These are unoptimised Python timings and exclude any network or consensus delay, which in a real permissioned ledger [16] would dominate commit time but not verification. The descriptor is 832 bytes, 1.3 percent of the raw 64 KiB image, and the on-chain footprint is one 32-byte leaf plus a share of a 104-byte header. Fig. 4(b) shows that proof verification grew from 14 to 19 µs as the ledger grew from 10^2 to 10^5 records, consistent with logarithmic proof length plus memory effects; in every case all genuine records verified and a record with one flipped bit was rejected. Fig. 4(a) shows the separation between benign and edited score distributions that underlies Table 3.

TABLE 5 COST PER IMAGE

Method	Verify (ms)	Stored bytes
SHA-256 (pixels)	0.23	32
Global DCT pHash	0.48	8
BFF features + fusion	25.3	832
Ledger proof check	0.047	32 on-chain

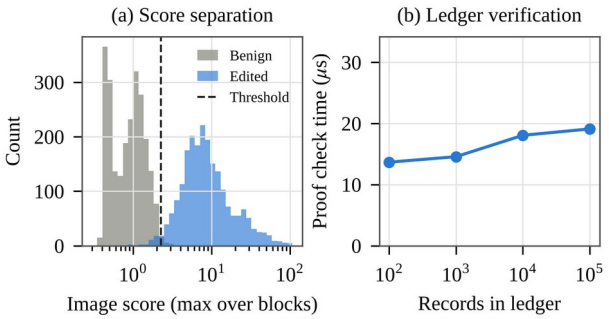


Fig. 4. (a) Distribution of image scores for benign and edited images with the calibrated threshold (log scale). (b) Merkle proof verification time against ledger size.

VII. Limitations

The findings rest on synthetic textures and synthetic defects; real surfaces have stronger non-stationarity, specular highlights and defect classes not modelled here, so the absolute rates should be treated as an upper bound until the scheme is tested on real inspection archives, for example public anomaly datasets with re-enacted forgeries [21]. Benign and test variants come from the same family of transformations used for calibration, which favours the method; unseen operations such as blur and resampling raised false alarms sharply. No adaptive adversary was modelled: a forger who knows the 13 features could try to construct an edit that preserves block statistics, as has been demonstrated for perceptual hashes [10], [11]. The ledger results cover data-structure cost only. Finally, the fixed 32-pixel grid limits localisation resolution and couples the method to image alignment.

VIII. Conclusion

This paper presented a tamper-evident record for industrial inspection images that pairs an exact digest with an 832-byte block-feature fingerprint and anchors both in a hash-chained ledger. In simulation it accepted routine brightness, gamma and JPEG changes with a 0.67 percent false-alarm rate while detecting 98.2 percent of defect insertions, removals and substitutions and locating the edit in every detected case, where a pixel digest rejected 91.7 percent of benign copies and a global perceptual hash missed most edits. The ablation showed that calibrating each feature group against its own benign variability was the decisive design step, and that intensity and gradient features carry most of the signal on the surfaces studied. Shifts and blur remain unsolved. Future work will add registration before comparison, evaluate on real inspection images with forensic ground truth, test feature-aware adaptive forgeries and measure end-to-end cost on a permissioned ledger.

Acknowledgment

[To be completed by the authors: sponsor or funding acknowledgment, if any.]

References

[1] Y. Chen, Y. Ding, F. Zhao, E. Zhang, Z. Wu, and L. Shao, “Surface defect detection methods for industrial products: A review,” *Appl. Sci.*, vol. 11, no. 16, Art. no. 7657, 2021.

[2] U.S. Food and Drug Administration, “Electronic records; electronic signatures,” *Code of Federal Regulations*, Title 21, Part 11.

[3] L. Verdoliva, “Media forensics and deepfakes: An overview,” *IEEE J. Sel. Topics Signal Process.*, vol. 14, no. 5, pp. 910–932, Aug. 2020.

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

[5] S. Haber and W. S. Stornetta, “How to time-stamp a digital document,” *J. Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.

[6] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[7] T. Alladi, V. Chamola, R. M. Parizi, and K.-K. R. Choo, “Blockchain applications for Industry 4.0 and industrial IoT: A review,” *IEEE Access*, vol. 7, pp. 176935–176951, 2019.

[8] R. Venkatesan, S.-M. Koon, M. H. Jakubowski, and P. Moulin, “Robust image hashing,” in *Proc. IEEE Int. Conf. Image Process.*, vol. 3, 2000, pp. 664–666.

[9] V. Monga and B. L. Evans, “Perceptual image hashing via feature points: Performance evaluation and tradeoffs,” *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3452–3465, Nov. 2006.

[10] Q. Hao, L. Luo, S. T. K. Jan, and G. Wang, “It’s not what it looks like: Manipulating perceptual hashing based applications,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2021.

[11] L. Struppek, D. Hintersdorf, D. Neider, and K. Kersting, “Learning to break deep perceptual hashing: The use case NeuralHash,” in *Proc. ACM Conf. Fairness, Accountability, Transparency (FAccT)*, 2022.

[12] J. Fridrich, D. Soukal, and J. Lukáš, “Detection of copy-move forgery in digital images,” in *Proc. Digital Forensic Research Workshop*, 2003.

[13] D. Cozzolino, G. Poggi, and L. Verdoliva, “Efficient dense-field copy–move forgery detection,” *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 11, pp. 2284–2297, Nov. 2015.

[14] D. Bhowmik and T. Feng, “The multimedia blockchain: A distributed and tamper-proof media transaction framework,” in *Proc. 22nd Int. Conf. Digital Signal Process. (DSP)*, London, U.K., 2017.

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

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

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

[18] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[19] Q. Luo, Y. Sun, P. Li, O. Simpson, L. Tian, and Y. He, “Generalized completed local binary patterns for time-efficient steel surface defect classification,” *IEEE Trans. Instrum. Meas.*, vol. 68, no. 3, pp. 667–679, Mar. 2019.

[20] J. Canny, “A computational approach to edge detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[21] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD—A comprehensive real-world dataset for unsupervised anomaly detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2019, pp. 9592–9600.