

Cryptographic Source Authentication of JPEG Images using IMEI and Digital Signature Embedded in COM Segment

David Arboledas Brihuega

(Asociación Española de imagen científica y forense, Spain)

<https://orcid.org/0009-0009-4002-1045>

Abstract: Digital image authentication poses growing technical and legal challenges, particularly in forensic contexts. The ease with which images can be altered using standard software, combined with the rapid rise of generative artificial intelligence techniques, undermines the credibility of much digital evidence. This paper presents a practical and technically feasible model to ensure the integrity and provenance of JPEG-format images. The proposed method involves generating a hash value of the image, retrieving the IMEI of the capturing device, and cryptographically signing this information using the manufacturer's private key. The signed data is then embedded into the JPEG file's COM marker segment (0xFFFF), enabling authenticity verification without relying on easily manipulated metadata. This approach aligns with core principles of forensic science and enhances the traceability and reliability of digital visual evidence.

Index Terms— JPEG, IMEI, digital signature, image authentication, forensic analysis, metadata, integrity.

DOI: pending assignment upon publication

1 INTRODUCTION

The authentication of digital images presents an increasingly complex technical and legal challenge, particularly in judicial and forensic settings. The ease with which images can be altered using conventional editing software—combined with the rapid emergence of generative artificial intelligence tools—has significantly undermined the reliability of visual evidence [22], [39].

Today, mobile phones have become the primary means of image capture, owing to their ubiquity and the high technical quality they offer. This makes every user a potential generator of visual evidence. However, traditional validation methods—such as EXIF metadata analysis or visual inspection—are proving insufficient. Metadata can be removed, falsified, or regenerated using widely available tools, rendering it a weak source of probative value [12].

Likewise, forensic techniques for source identification—such as Photo-Response Non-Uniformity (PRNU) analysis, Discrete Cosine Transform (DCT) coefficient inspection, or interpolation detection—while valuable in controlled environments, present critical limitations: high sensitivity to successive compressions, the need for reference images, low reproducibility in real-world conditions, and reliance on post-capture analysis rather than verification at the moment of capture. The PRNU fingerprint, though widely used, is notably fragile: it degrades under compression, scene content variation, sensor pipeline mismatches, and can even lose distinctiveness across devices of the same model [23], [17].

To address these shortcomings, this study proposes an alternative source-level authentication model, applicable to mobile devices. The proposal is based on three core components: (1) computing a hash value of the image's visual content, (2) extracting the IMEI identifier of the capturing device, and (3) digitally signing both pieces of data with the manufacturer's private key. The resulting signed data is embedded directly into the JPEG file using the standard COM marker segment (0xFFFF), without altering the file's structure or visual rendering.

This approach enables validation of both the content's integrity and the identity of the capturing device, without relying on manipulable metadata or statistical inference. Aligned with forensic science principles, the solution provides a verifiable, tamper-resistant, and computationally efficient mechanism that strengthens the traceability and reliability of digital images as legal evidence [27].

Under European legislation, the use of the IMEI number must be assessed in light of the General Data Protection Regulation (Article 4(1) GDPR), which acknowledges that technical identifiers such as mobile device numbers may constitute personal data when they allow the direct or indirect identification of a natural person [9]. In this

context, the IMEI—being persistently linked to a specific terminal—may be considered personal information, especially when combined with other data that could associate the captured image with a specific user.

This implies that its processing—including collection, storage, and access—must adhere to the core principles of the GDPR: lawfulness, data minimization, purpose limitation, and security. Additional requirements also apply in forensic or judicial contexts, such as traceability of access and control over the chain of custody. Consequently, any implementation of the proposed model must incorporate appropriate data protection safeguards, such as encryption, secure execution environments, and, where applicable, functional anonymization of the IMEI during subsequent analysis phases.

2 STATE OF THE ART AND EXISTING SOLUTIONS

In recent years, the cryptographic authentication of digital images has made significant advances, driven by the need to combat visual disinformation and ensure the integrity of multimedia content. Initiatives such as C2PA (Coalition for Content Provenance and Authenticity) and JUMBF (JPEG Universal Metadata Box Format) [8], [14], have been adopted by manufacturers like Leica [31] and are currently being integrated by Nikon [30] and Sony [32], [34]. These models enable the encapsulation of structured metadata—authorship, editing history, software used, and more—cryptographically signed within the file using complex manifest-based structures.

The model proposed in this paper can be seen as a simpler and more focused alternative, specifically designed for forensic, judicial, or institutional contexts, where the main objective is to verify the integrity of an image and its linkage to the capturing device at the moment of acquisition, rather than reconstructing its entire editing history.

2.1 Structural Comparison with C2PA and JUMBF

Feature	C2PA / JUMBF	Proposed Model (IMEI + Signature in COM)
Insertion Structure	JUMBF (APP11), structured JSON/CBOR	Standard JPEG COM segment (0xFFFE)
Digital Signature	COSE/CMS, based on complex manifests	RSA over concatenated string (IMEI + hash); future migration to EdDSA
Metadata Included	Full history, authorship, software, license	Hash + IMEI + digital signature
Robustness Against Editing	High (if JUMBF block is preserved)	Limited (COM segment may be removed by software)
Verification Tools	Adobe suite, CLI, C2PA validators	Offline via manufacturer's public key
Implementation Complexity	High (manifests, certificates, infrastructure)	Low (can be implemented directly in firmware)

The model proposed here avoids the use of complex structures and is fully compatible with the JPEG standard, without relying on vendor-specific tools. This makes it particularly well-suited for environments lacking constant connectivity, specialized validators, or hierarchical certificate infrastructures—such as in forensic procedures or decentralized digital custody scenarios.

2.2 Specific Applications in Forensic Context

While C2PA is primarily designed for editorial and publishing scenarios—where a complete record of content editing and transfer is required [3], [31], [32]—the model presented in this paper is better suited to the following contexts:

- Primary acquisition of digital evidence, establishing integrity at the point of capture without intermediary handling.
- Image capture in offline environments, where access to a centralized verification platform is not possible.
- Scenarios requiring verification of the time of capture, source device, and content integrity, without the need to trace subsequent transformations [5].

2.3 Implementation Cost Comparison

The following table provides a comparative cost estimate of current commercial solutions versus the proposed model:

Solution / Standard	Estimated Cost per Unit	Key Features	Requirements
Leica M11-P with Content Credentials	> €9,000	Professional camera with secure chip and C2PA signing	Dedicated hardware, Adobe ecosystem
Sony α7 IV / ZV-E1 with in-camera signing	€2,000–€3,500	Device-based key signing, closed validation	Proprietary Sony ecosystem
C2PA implementation (Adobe, Truepic) SDK (Adobe, Truepic)	Approx. €500 per device	Structured signing using JUMBF manifests	External server and certificate infrastructure
Proposed model (IMEI + signed COM)	< €1 per device	Embedded authentication via IMEI and digital signature in firmware	Mobile Trusted Platform Module (TPM Mobile)

The proposed model offers a low-cost alternative with minimal hardware requirements, making it especially suitable for large-scale deployments or integration into mobile devices through firmware-level implementations. Its affordability and simplicity do not compromise its alignment with forensic validation needs, offering a balance between security and feasibility.

3 TECHNICAL FOUNDATIONS

3.1 Lifecycle of a Digital Image

The generation of a digital image involves a series of optical, electronic, and computational processes [38]. Light enters through the device’s lens and passes through a Color Filter Array (CFA), allowing the sensor—typically CMOS or CCD—to convert it into an electrical signal [20], [7]. This signal is then interpolated via demosaicing and subjected to internal adjustments such as white balance, color correction, sharpness, contrast, and gamma. The result is a digital file in RAW or compressed format, most commonly JPEG [7] (Figure 1).

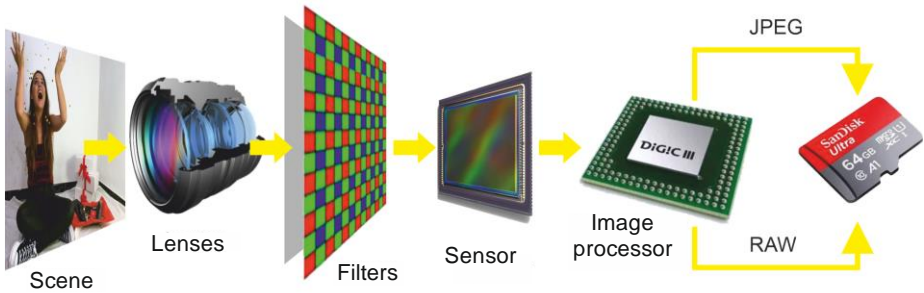


Figure 1: Lifecycle of a digital image from optical capture to storage in JPEG or RAW format. The diagram illustrates key stages of internal processing in image-capturing devices.

In mobile devices, JPEG is the dominant format due to its efficiency. However, once captured, an image can be edited, forwarded, or recompressed, making it difficult to trace its original content and source. Understanding these stages is critical for designing authentication mechanisms that intervene as early as possible in the process—specifically, at the moment of capture.

3.2 JPEG Format Structure

The JPEG standard (ISO/IEC 10918-1) organizes images as a sequence of segments delimited by binary markers [28], [13], as illustrated in Figure 2.

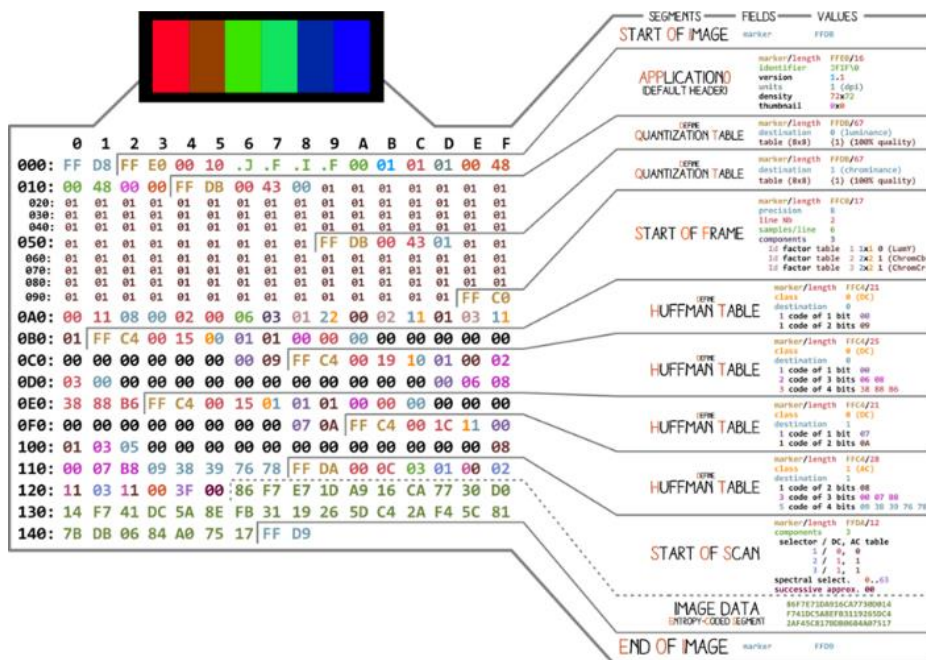


Figure 2: Schematic structure of a JPEG file, showing binary markers and typical segment organization according to ISO/IEC 10918-1. From [19].

Every JPEG file begins with the SOI (Start of Image, 0xFFD8) marker and ends with the EOI (End of Image, 0xFFD9) marker [37]. Between these, various functional markers are included, such as [37].

- APP0 / APP1: Sections reserved for metadata (e.g., EXIF or JFIF).
- COM (0xFFFE): Arbitrary comments.
- DQT: Quantization tables.
- SOF: Start of frame.
- DHT: Huffman coding tables.
- SOS: Start of scan.

It is worth noting that a JPEG file may contain more than one SOI/EOI marker pair. This occurs, for example, when a thumbnail image is embedded within EXIF metadata, resulting in a nested image structure within a single file [2].

3.2.1. Insertion of Custom Data

The JPEG standard does not natively provide a dedicated space for storing cryptographic authentication data. However, it is technically feasible to insert custom data—such as a digital seal or device identifier—without compromising the file’s structure or visual rendering.

The COM marker (0xFFFE) offers a standardized mechanism for embedding additional information. This segment can hold up to 65,533 bytes of usable data, excluding the two bytes reserved for length specification [28], which is more than sufficient to include an IMEI identifier, a hash value, and a digital signature using RSA or EdDSA. This structural extensibility enables the integration of verification mechanisms directly within the image file, while maintaining full compatibility with standard image viewers and processing libraries.

3.2.2 Towards an Evolution of the Standard

Currently, many image transmission platforms—such as social networks and messaging apps—routinely strip or overwrite metadata, thereby undermining the traceability of the original image [21], [35]. For this reason, it is desirable that future revisions of the JPEG standard formally incorporate dedicated structural mechanisms for authentication and digital signing, with protection against compression, editing, or transmission-related loss.

Native integration of such elements would enhance the traceability of digital images and support their use as technical evidence in forensic, documentary, or judicial contexts [16].

3.3 Limitations of Metadata

Metadata remains the most common mechanism for storing technical information associated with digital images. Through segments such as APP1 (EXIF), an image can include details such as:

- Date and time of capture,
- Camera model and manufacturer,
- Geolocation,
- Optical settings of the device (lens type, aperture, shutter speed, etc.),
- Information about the software used for processing or editing.

These metadata are useful for organizing, classifying, and—in some cases—inferring the authenticity or context of an image. However, they have a critical vulnerability: they can be easily modified, deleted, or forged using widely available tools such as ExifTool, without leaving visible traces [11].

This model does not replace judicial procedures for evidence validation but provides a complementary technical safeguard

3.3.1 Evidentiary Fragility

In legal contexts, this vulnerability renders metadata a low-reliability source of evidence. Their alteration does not require advanced skills or privileged access to the system. As a result, metadata cannot be relied upon as the sole basis for asserting the origin or integrity of an image.

Furthermore, there is currently no universally accepted standard for embedding integrity proofs directly into a JPEG file in a way that is independent of structured metadata. The absence of cryptographic protection leaves the file open to deliberate or accidental manipulation, compromising the evidentiary validity of the image [16].

3.3.2 The Need for Alternative Mechanisms

This situation highlights the urgent need for embedded cryptographic authentication systems that are independent of metadata and resilient to their removal. Such systems must be based on digital signature techniques, support public validation, and be linked to trusted device identifiers (such as the IMEI) in order to be admissible as robust technical evidence in forensic or judicial settings.

3.4 IMEI as a Device Identifier

The IMEI (International Mobile Equipment Identity) is a unique numerical identifier assigned to every GSM mobile device [10]. It consists of 15 digits and is standardized by the 3rd Generation Partnership Project. Its internal structure follows the TAC-SNR-SP format [14] where:

- TAC (Type Allocation Code): identifies the device model,
- SNR (Serial Number): denotes the individual device's serial number,
- SP (Spare): a reserved field for future use.

This code is hardcoded at the hardware level, typically within the read-only memory (ROM) of the device's modem, making it extremely difficult to forge without physically altering the device or compromising its base firmware [1].

3.4.1 Probative Value of the IMEI

The strength of the IMEI as an identifier lies in its persistence and relative immutability, even in the face of device reboots, software updates, or operating system modifications. Although spoofing techniques exist for rooted or modified devices, such methods are detectable through forensic analysis and require privileged access to the hardware [6], [1].

Therefore, the IMEI serves as a suitable component for integration into source-level image authentication mechanisms. Its inclusion in the digital signing process provides an additional layer of assurance regarding the link between the captured image and the physical device that generated it.

3.4.2 Integration with Authentication Systems

By incorporating the IMEI into the chain of digitally signed data, a verifiable relationship is established between the JPEG file and the capturing hardware. This enhances the traceability of the visual content and allows for the certification not only of the image's integrity but also of its physical origin with a high degree of reliability [16], [1].

This approach does not require external verification infrastructure or constant access to mobile networks. The IMEI can be retrieved locally, validated using the manufacturer's public key, and compared against the embedded data in the JPEG file, as will be detailed in later sections.

4 PROPOSED MODEL

The proposed model aims to ensure the authenticity of a digital image from the moment of capture, addressing two fundamental aspects:

1. The integrity of the visual content, ensuring it has not been altered after acquisition.
2. The identification of the capturing device, establishing a verifiable link to its physical origin.

Unlike retrospective forensic approaches, this model adopts a prospective strategy by generating cryptographic evidence at the moment of capture. Authentication data is embedded directly into the JPEG file using a standard segment (COM, 0xFFFE), without affecting image rendering or structural compatibility.

4.1 Technical Requirements

The experimental implementation of the model was carried out in a mixed Android/PC environment using the following:

Hardware:

- Forensic workstation running Windows 11
- Mobile device with Android operating system (Samsung Galaxy M31)

Software:

- Microsoft Windows 11, version 23H2
- Android SDK (Android Studio)
- Python 3.11.5
- Cryptographic library PyCryptodome 3.11.0
- Java (for IMEI access via TelephonyManager)
- Android Debug Bridge (ADB) for forensic IMEI extraction without user interaction

The system was designed to be easily integrated into a device's firmware or used externally in forensic analysis without requiring app installation on the original device.

4.2 IMEI Extraction

The IMEI can be obtained at the time of capture through the Android API. In Java, this is accomplished using the TelephonyManager [4], [33]:

```
TelephonyManager telephonyManager =  
(TelephonyManager) context.getSystemService(Context.TELEPHONY_SERVICE);  
String imei = telephonyManager.getDeviceId();
```

Since there is no direct and standardized way to access a mobile device's IMEI using Python, Java was used in conjunction with Android, following the approach outlined by [25].

For forensic environments, the IMEI can be retrieved without direct interaction with the device using ADB from a workstation:

```
adb shell service call iphonesubinfo 1
```

This method does not require root access or additional software installation and allows the identifier to be extracted from the telephony subsystem.

4.3 Image Hash Calculation

The system generates a cryptographic digest of the image's visual content, excluding metadata and non-visual segments. The hash is computed over the byte range between:

- The Start of Scan (SOS) marker (0xFFDA), which denotes the beginning of the encoded data, and
- The End of Image (EOI) marker (0xFFD9), which marks the end of the file.

For the proof of concept, the MD5 algorithm was used due to its efficiency; however, it is recommended to replace it with more secure hash functions (such as SHA-256 or SHA-3) in final implementations [26].

The hash functions as a unique fingerprint of the image: any alteration, no matter how minimal, will produce a different output. Ideally, the hash computation should be executed within a secure environment, such as a Mobile Trusted Platform Module (TPM Mobile) integrated into the device.

4.4 Data Signing

Once the IMEI and the cryptographic hash of the visual content have been generated, they are concatenated to form an authentication string. This string is digitally signed using the RSA algorithm, with the device manufacturer's private key [18]. Ideally, this process is executed within a secure environment such as TPM Mobile, which prevents direct access to the private key or the unsigned hash thus preserving the integrity of the cryptographic operation [36].

4.4.1 Components of the Authenticated Chain:

- Device IMEI: A unique identifier physically tied to the hardware (Figure 3a).
- Hash of the image's visual content: Computed from the Start of Scan (SOS) marker (0xFFDA) to the End of Image (EOI) marker (0xFFD9), excluding segments such as APPn or COM (Figure 3b).

The resulting chain is signed with the private key, and the output is Base64-encoded to facilitate its storage as text within the JPEG file (Figure 3c). Finally, the complete chain—including IMEI, hash, and signature—is embedded into the COM segment of the JPEG file (Figure 3d).

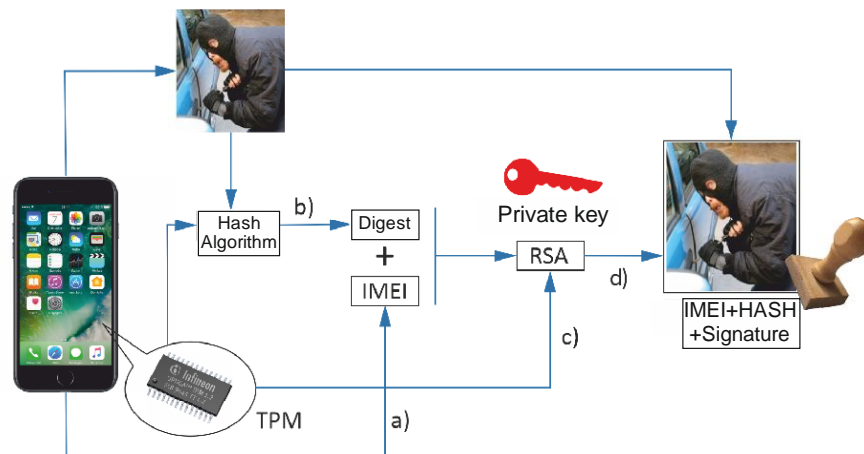


Figure 3: Diagram of the Proposed Authentication Process. The captured image is processed on the mobile device, where a hash of the visual content is computed, concatenated with the IMEI, and signed using the manufacturer's private key. The resulting data is embedded into the JPEG file as a COM segment.

This mechanism ensures that the authenticated data:

1. Originate from a legitimate source—a specific device.
2. Have not been altered since the moment of capture.

The manufacturer's public key is stored in an externally accessible file for use in verification processes. In contrast, the private key remains protected within the device's firmware, ideally embedded in a Trusted Execution Environment (TEE), ensuring it never leaves the secure environment of the cryptographic chip [36].

Similarly, the image hash value is never exposed outside the TPM Mobile prior to being signed. This design ensures that both the content and the signature remain fully protected throughout the process, minimizing the risk of interception or tampering.

4.5 Insertion of the Cryptographic Seal into the JPEG Image

The COM segment (0xFFFE) is part of the JPEG standard and is intended to store arbitrary comments. It has a simple structure: two length bytes followed by up to 65,533 bytes of payload [13], which is sufficient to include the IMEI, the hash, and the digital signature.

In this model, the cryptographic seal is inserted just before the End of Image (EOI) marker (0xFFD9), ensuring it does not interfere with the visual content decoding performed by standard viewers or libraries.

4.5.1 Advantages of the Approach

- **Standardization:** The COM segment is formally recognized in the JPEG specification.
 - **Full compatibility:** The image remains viewable in common applications such as Windows Photo Viewer, GIMP, or Photoshop.
 - **Non-intrusive:** It does not alter the visual content or primary metadata.
 - **Embedded authentication:** The cryptographic seal is inseparably bound to the captured image.
- This design allows for the image's authenticity to be verified at any later point without requiring access to the original device, provided the corresponding public key is available.

Furthermore, embedding the seal does not require any modification to JPEG compression algorithms or alterations to visual encoding. The image remains visually identical to the original capture, except for the inclusion of the COM segment, which is ignored by conventional decoders.

4.6 Authentication System Workflow

The process is structured into five main phases:

1. **IMEI extraction from the device:** Unambiguous identification of the capturing mobile terminal.
2. **Hash computation of the JPEG image:** Generation of a digital fingerprint sensitive to any modification.
3. **Construction of the cryptographic metadata:** Concatenation of the IMEI and the hash into an authentication string.
4. **Digital signature of the chain:** Asymmetric encryption of the string using the manufacturer's private key.
5. **Insertion of the seal into the JPEG file:** Embedding of the digital signature in a standard COM segment, prior to the EOI marker.

This architecture ensures that authentication is self-contained, embedded, verifiable offline, and resistant to post-capture manipulation—provided the seal remains intact.

4.7 Analysis and Validation Process

During forensic analysis, the authenticated JPEG file is processed in order to verify its integrity and origin. The embedded data located in the COM segment (0xFFFE) is extracted, containing:

1. The IMEI of the device that generated the image.
2. The hash value of the visual content, calculated at the moment of capture.
3. The RSA digital signature generated with the manufacturer's private key.

This process enables validation of both the file's authenticity and its integrity without requiring access to the original device.

4.7.1 Reading the Device's IMEI

In a forensic setting, the mobile device can be connected to a workstation via a standard USB interface. The IMEI can be extracted without installing any additional software, as long as the Android SDK environment is available. Using Android Debug Bridge (ADB), it is possible to access internal system services and retrieve the hardware identifier of the device (Figure 4).


```
Símbolo del sistema x + v
Microsoft Windows [Versión 10.0.22631.3593]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\David Arboledas>adb shell "service call iphonesubinfo 4 | cut -c 52-66 |
tr -d '[:space:]'"

512667520717890

C:\Users\David Arboledas>
```

Figure 4: Example of IMEI extraction from an Android device using the adb shell command on a forensic workstation with the Android SDK. The command accesses internal system services without requiring direct interaction with the device.

This approach is non-invasive and adheres to digital evidence preservation principles, making it suitable for judicial and forensic procedures.

4.7.2 Recalculation of the Visual Content Hash and Verification of the Digital Signature

The next step in the forensic process involves extracting the original hash and digital signature from the COM segment (Figure 5).

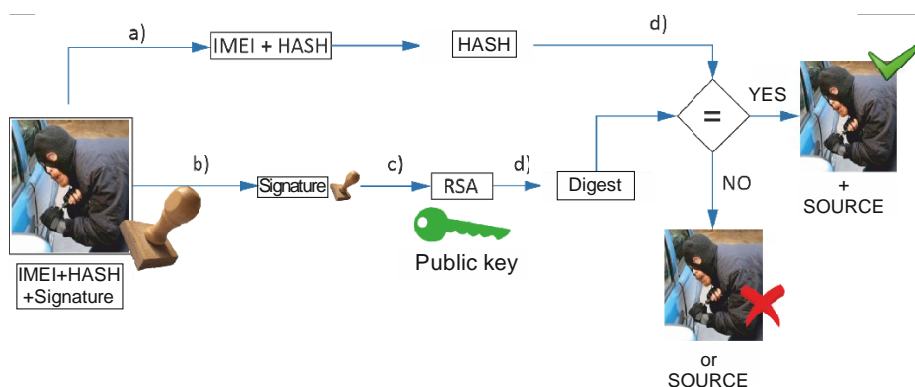


Figure 5: Forensic validation process. The embedded data (IMEI + HASH + SIGNATURE) is extracted, the signature is verified using the manufacturer's public key, and the hash is recalculated to confirm the image's integrity. If the data match, the image's legitimate origin is confirmed; otherwise, tampering is detected.

The hash of the visual content is then recalculated by processing only the data between the Start of Scan (SOS) marker (0xFFDA) and the End of Image (EOI) marker (0xFFD9), explicitly excluding the COM segment itself. The newly calculated hash is compared with the original hash stored in the file. If both values match, this confirms that the image has not been modified.

Finally, the digital signature is verified using the manufacturer's public key. If the verification succeeds, it confirms that the data was generated by the legitimate device.

This procedure ensures two critical aspects:

- That the image has not been modified since the moment of capture.
- That it was generated by a legitimate and verifiable device whose public key is known and trusted.

If there is a mismatch between the hash values or if the signature verification fails, it is concluded that the image has been tampered with, thereby invalidating its evidentiary value.

5 EXPERIMENTAL RESULTS

To evaluate the technical feasibility of the proposed model, a test environment was implemented consisting of:

- A forensic workstation running Windows 11
- A set of Python and Java scripts integrated with the Android SDK

The system was successfully validated in both the emulation of the image capture phase and the subsequent forensic analysis. The insertion of the cryptographic seal into the COM marker did not affect the rendering or structure of the JPEG file, confirming compatibility with standard image viewers.

To support reproducibility and facilitate practical application in forensic contexts, the scripts developed for generating, digitally signing, and embedding the cryptographic seal—as well as for verifying authenticated JPEG images—have been published in an open repository

<https://github.com/darboledas/jpegAuthentication/>

The repository includes working Python examples, usage instructions, test images, and all necessary dependencies. The code is released under the MIT license and is intended for researchers, forensic analysts, and developers interested in integrating or evaluating the proposed model.

5.1 Sealed Image Generation

Once the image is captured in JPEG format, the authentication workflow is executed automatically, including:

1. Extraction of the IMEI via the Android API
2. Calculation of the MD5 hash of the visual content, limited to the byte range between the SOS and EOI markers
3. Digital signature of the IMEI + MD5 chain using the RSA algorithm and the manufacturer's private key
4. Insertion of the resulting seal (IMEI + MD5 + signature) into a COM segment (0xFFFE) within the JPEG file

The output is a structurally valid JPEG file, fully compatible with standard viewers and equipped with a cryptographically verifiable authentication seal (Figure 6).

```
000000D8 00 00 01 02 03 FF FE 00 EC 49 4D 45 49 3D 33 35 36 37 38 39 31 30 34 35 .....IMEI=3567891045
000000F0 38 32 33 36 31 7C 4D 44 35 3D 30 34 65 61 31 34 64 62 30 32 65 66 35 31 82361MD5=04ea14db02ef51
00000108 33 35 63 37 38 62 65 61 33 34 66 65 35 38 31 30 38 63 7C 53 49 47 3D 70 35c78bea34fe58108cSIG=p
00000120 52 59 46 6D 49 38 57 72 70 2B 67 42 6A 4E 38 52 76 72 58 66 4F 2F 4E 66 RYFmI8Wrp+gBjN8RvrXF0/Nf
00000138 30 37 4C 55 4C 74 67 78 55 46 71 69 64 41 50 37 4E 55 39 31 61 69 49 2B 07LULtgxUFqIdAP7NU91aII+
00000150 42 67 6F 43 46 67 43 6B 61 75 66 56 73 6E 6A 79 6E 67 68 63 53 7A 74 63 BgoCFgCkaufVsnjynghcSztc
00000168 58 43 6D 37 36 6D 31 2B 65 70 53 73 4A 6F 61 6A 47 77 64 6C 32 2B 43 47 Xcm76m1+epSsJoajGwdl2+CG
00000180 58 77 62 62 6B 74 34 42 44 49 6A 73 78 67 4A 6E 50 53 30 72 65 74 64 76 Xwbbkt48DIjsxgJnPS0retdy
00000198 2B 6B 57 37 6F 4C 4A 31 70 67 69 67 62 72 49 47 42 47 4B 52 32 38 72 75 +kW7oLJlpgigbrIGBGKR28ru
000001B0 62 73 79 37 6B 52 4D 47 51 37 49 41 70 77 65 72 4C 6E 4D 4B 39 36 4C 38 bsy7kRMGQ7IApwerLnMK96L8
000001C8 7A 73 3D FF DA 00 0C 03 01 00 02 10 03 10 00 00 02 F4 8C 55 94 00 00 28 zs= .....fU0..(
```

Figure 6: Hexadecimal fragment of the JPEG file showing the COM segment (0xFFFE) with the embedded authentication seal. The segment includes the IMEI identifier, the hash value, and the RSA-generated digital signature.

5.2 Image Validation

Multiple authenticated images were processed following the procedure described in the previous section. The validation consisted of:

- Extracting the IMEI + HASH + RSA digital signature chain from the COM segment
- Recomputing the MD5 hash of the visual content of the image, limited to the data between the Start of Scan (SOS) marker (0xFFDA) and the End of Image (EOI) marker (0xFFD9)
- Verifying the digital signature using the manufacturer's public key

In all cases where the images had not been tampered with, the software successfully validated both the integrity of the content and the authenticity of the source (Figure 7).

```
C:\WINDOWS\system32 x + v
(base) C:\Users\David Arboledas\Desktop\Paper\com>python verify_jpeg_signature.py prueba.jpg public.pem
Extracted COM block:
IMEI=356789102345678|MD5=c6a8d6f915595f1a9f4d98bd16a5b7e3|SIG=j1e1q0RS+0qeBkSmLGM0ltiZ6N7fr8tuj8y8lb5S
uS+P2mvYyfr91s1WeJ2NosTmdIeLiNVH/TyY42YZDe0rPSl0HDrGz3wLou4LPIIcb88sEEyL0gJieAFSrG22FNz2C/Ki1yoldoRn
6bEG8XlyZisDVqA8B2okVEyFKY=
Reported IMEI: 356789102345678
MD5 stored in COM: c6a8d6f915595f1a9f4d98bd16a5b7e3
Computed MD5: c6a8d6f915595f1a9f4d98bd16a5b7e3
✔ VALID SIGNATURE: The image is authentic and corresponds to the specified IMEI.
(base) C:\Users\David Arboledas\Desktop\Paper\com>
```

Figure 7: Result of the validation process in command-line interface. The hash is confirmed to match the value stored in the COM segment, and the RSA signature is successfully verified using the manufacturer's public key.

These results support the following conclusions:

- The image was not modified after capture.
- The file was generated by a legitimate device, whose IMEI matches the declared identifier.

5.3 Tamper Detection

To evaluate the system's ability to detect tampering, intentional modifications were applied to previously authenticated images. These included visual content alterations (e.g., pixel-level changes) and manual edits to the IMEI field within the COM segment.

In all cases, the validation system successfully detected the manipulations through:

- Inconsistencies between the recalculated and stored hash values
- Invalid digital signature when verified using the manufacturer's public key

These tests confirm the effectiveness of the model in detecting altered images, thereby preserving the evidentiary value of those that remain unmodified (Figure 8).

```
C:\WINDOWS\system32 x + v
(base) C:\Users\David Arboledas\Desktop\Paper\com>python verificar_firma_jpeg.py prueba.jpg public.pem
Extracted COM block:
IMEI=357435934870459|MD5=04ea14db02ef5135c78bea34fe58108c|SIG=dFXrhDqgBhLmN76ApSBRLhXqyHh/fr03Q3EMZNJMb
Mk8VQC8vNYBR8cQUxKprygQtJycwtVKHvG3QsbpwwLLo7gsfS73L3jXt3LViksQxF2mCFeh6cixDw3Lu49bCoZSEfHjAVvc/3vM9T
uAmEEEXnrHLziUt+M4e50+z7xZxQ=
Reported IMEI: 357435934870459
MD5 stored in COM: 04ea14db02ef5135c78bea34fe58108c
Computed MD5: 76272dda0220c03536612779189511cc
✖ ERROR: MD5 hash mismatch with actual image content.
```

Figure 8: Result of the validation process for a tampered image. A mismatch between the recalculated and stored hash, combined with a failed signature verification process, indicates that the image was altered after the capture process.

5.4 Performance and Compatibility

Experimental testing showed that the complete cryptographic sealing process—including MD5 hash computation and RSA signature generation—incurrs very low computational overhead on a standard mobile device.

No issues were observed when viewing the resulting images. Files containing the cryptographic seal were successfully opened using standard applications such as Windows Photo Viewer, GIMP, and Adobe Photoshop. No errors or warnings were reported. The embedded COM segment was ignored by standard image viewers but remained detectable and extractable using specialized tools or custom verification scripts.

6 DISCUSSION

Verifying the origin and integrity of digital images is an increasingly pressing issue in contexts where visual evidence holds probative or documentary value [24], [29]. The proposed model addresses this need from a technical, practical, and strategic perspective: by generating a cryptographic seal at the moment of capture and embedding it directly into the JPEG image—without relying on volatile metadata or external infrastructures. Unlike retrospective systems (which rely on compression artifacts, sensor noise, or statistical analysis), this prospective approach provides authentication at the point of origin. Furthermore, as shown in comparison to existing standards, it achieves this with significantly lower complexity, minimal cost, and valuable technological independence—making it well suited for resource-constrained or decentralized environments.

6.1 Strengths of the Model

The proposed system offers several key advantages that position it as an effective solution in contexts where the reliability and authenticity of digital images are critical:

- **Robust traceability:** The use of the IMEI as a unique device identifier allows each image to be unequivocally linked to its physical origin.
- **Independent verifiability:** Any analyst with access to the manufacturer's public key can verify the signature without requiring proprietary infrastructure.
- **Universal compatibility:** By using the standard COM segment, the image remains viewable with common tools without breaking its structure or altering its visual content.
- **Automated integration:** The process can be implemented at the firmware level, ensuring continuous and user-transparent authentication.
- **Low operational cost:** The model can be adopted with minimal overhead compared to commercial solutions such as C2PA-compliant cameras.

These features make the proposal a viable alternative for organizations with critical needs for image authentication but lacking access to proprietary ecosystems or specialized hardware.

6.2 Limitations and Challenges

Despite its potential, the model faces certain technical and strategic limitations that must be considered for real-world deployment:

- **Manufacturer dependency:** Effective adoption requires manufacturers to integrate the signing process into the device firmware. This presents a significant entry barrier, although it could be addressed through regulatory frameworks or official certification programs.
- **Public key management:** To ensure universal verifiability, a trusted and accessible repository of manufacturer public keys would be required, along with a revocation mechanism and integrity control system.
- **Seal persistence:** The COM segment may be stripped by editing platforms or messaging apps. However, such removal is detectable. Using a marker recognized by forensic tools reinforces the seal's evidentiary validity and its admissibility in legal contexts.
- **Hash vulnerability:** MD5 was used in proof-of-concept tests for simplicity and performance. However, a secure hash function such as SHA-256 or SHA-3 [26] must be employed for production-grade implementations.

6.3 Attack Vectors and Mitigations

Below are plausible attack vectors, along with their risk level and the model's built-in mitigations:

- **Seal extraction and reuse:** An attacker might attempt to copy the COM segment from an authentic image and insert it into another. This is detectable since the recalculated hash would not match the altered content.
- **IMEI spoofing:** On compromised (rooted) devices, an attacker might attempt to forge the IMEI. However, the attack would also require access to the manufacturer's private key—implying a severe hardware-level security breach.
- **Fake signature injection:** This attack is mitigated through the use of RSA with a protected private key. If hosted in a secure environment such as a TPM Mobile, the integrity of the signature is ensured.
- **COM segment removal:** If the block is stripped by software or social media platforms, the image becomes forensically questionable, as its integrity can no longer be verified.
- **Hash Collision Attacks:** Although the proposed model recommends replacing MD5 with a secure hash function such as SHA-256 or SHA-3, the theoretical possibility of hash collision attacks must also be considered. In a

collision scenario, two distinct image contents could produce the same digest value, potentially allowing a manipulated image to pass integrity verification. While such attacks remain computationally infeasible for modern hash functions under current technology, ongoing advances in quantum computing and distributed brute-force techniques warrant continuous monitoring. Therefore, periodic algorithm updates and migration to post-quantum-resistant hash standards are advisable to ensure long-term cryptographic robustness.

In summary, the model demonstrates adequate resilience against common attack vectors, provided that the cryptographic ecosystem (keys, repositories, devices) is managed in accordance with robust security standards.

6.4 Forensic Classification of Images with Embedded Digital Signatures (COM Block in JPEG)

Based on the proposed model, the following table defines a forensic classification scheme for JPEG images that incorporate a digital signature embedded in the COM marker (0xFFFE). This signature links the image to the capturing device via its IMEI and a cryptographic hash.

This typology provides a systematic framework for categorizing the reliability of analyzed images in judicial or forensic contexts, and serves to guide subsequent actions related to verification, chain of custody, or exclusion (Table 1).






Category	Technical Condition	Forensic Interpretation	Recommended Actions
 Authentic	- COM segment present - Valid signature (verifies with public key) - IMEI and hash match actual data	Valid Original image, reliably linked to the capturing device	Accept as strong evidence. Record fingerprint and metadata.
 Questionable	- COM segment missing, corrupted, or altered –or– invalid signature or mismatched hash	Potentially altered or untrustworthy image	Treat as non-verifiable. Request original source if possible.
 Tampered	- COM segment present - Correct IMEI - Invalid signature (verification fails) or hash mismatch	Evidence of post-capture modification	Consider as manipulated. Document inconsistencies in detail.
 Unreadable	- JPEG format corrupted or unreadable - COM segment and/or hash cannot be extracted	Authenticity cannot be determined	Reject as direct evidence; attempt technical recovery if justified.
 Partially Valid	- Valid signature, but IMEI does not match the expected device (e.g., possible clone)	Image itself is valid, but possibly misattributed	Requires further investigation; may indicate device spoofing or identity fraud.

Table 1: Forensic classification criteria for JPEG images with embedded COM-based cryptographic seals.

This classification framework operationalizes the cryptographic validation model and provides an objective guideline for forensic decision-making in the handling of digital images.

7 TOWARD A FORENSIC SPECIFICATION FOR JPEG

Since the JPEG standard already recognizes the COM marker (0xFFFE) as a valid space for arbitrary data, we propose formalizing a specification for embedding cryptographic authentication information, including:

- Structured encoding of the IMEI (or other hardware identifier)
- A robust hash function (preferably SHA-256 or SHA-3)
- A digital signature using RSA-2048, Ed25519, or ECDSA P-256
- Optional: a timestamp and operator or application ID
- Automatic rejection rules in the event of discrepancies or tampering

This would enable the creation of a forensic JPEG profile, aimed at decentralized primary authentication with low computational complexity, but strong legal and evidentiary validity. Moreover, it would facilitate integration into digital chain-of-custody platforms and forensic and judicial workflows.

8 CONCLUSIONS

This paper presents a practical and functional model for authenticating JPEG images at the moment of capture, based on the use of the device's IMEI, cryptographic hash functions, and digital signatures. The proposal addresses an urgent need in forensic analysis and digital evidence validation: ensuring that an image is authentic, has not been tampered with, and originates from a specific device.

The method stands out for its low computational cost, compatibility with existing standards, and potential for direct integration into mobile devices. Experimental results confirm its effectiveness in both validating legitimate images and detecting post-capture manipulation—overcoming many limitations of traditional approaches based on metadata heuristics or compression artifacts.

However, widespread adoption would require active collaboration from device manufacturers and the promotion of standardization initiatives to define secure and persistent spaces for storing cryptographic metadata within the JPEG format.

The presented approach offers a cost-effective pathway toward standardized, device-embedded image authentication for digital forensics

9 FUTURE WORK

Despite significant progress in image authentication, several open challenges remain. Future research should address the robustness of authentication techniques against increasingly sophisticated generative models, such as deepfake synthesis and vision–language models capable of producing photorealistic content.

In particular, the following directions should be pursued:

1. **Improvement of the Cryptographic Scheme:** Replace the MD5 algorithm with a collision-resistant hash function such as SHA-256 or SHA-3, implemented directly in the device firmware.
2. **Integration with Secure Hardware:** Implement key generation and storage in TPM Mobile with Trusted Execution Environment to enhance system security.
3. **Formalization of a JPEG-Forensics Profile:** Submit proposals to standardization bodies such as ISO/IEC to define dedicated markers or structures for storing authentication metadata and digital signatures—addressing the growing demand for image traceability, especially in legal, journalistic, and documentary domains. This initiative could build upon the structural framework of ISO/IEC 19566-5 (JPEG Universal Metadata Box Format, JUMBF), extending it with a dedicated forensic profile that formally defines fields for cryptographic identifiers and integrity proofs.
4. **Automation of Forensic Analysis:** Develop tools and application programming interfaces (APIs) to automatically verify large volumes of images through graphical interfaces or by integrating them into digital forensic analysis platforms.
5. **Applications in Digital Chain of Custody:** Extend the model to environments requiring a robust digital chain of custody, such as law enforcement, investigative journalism, or forensic expert reporting.

In summary, the model presented here is not intended to replace existing forensic techniques but to complement them with a source-level authentication approach—one that strengthens trust in digital images as objective and verifiable evidence. In an era where millions of images are generated daily—and where generative AI enables the creation of photorealistic content disconnected from reality—embedding cryptographic mechanisms directly into images is becoming a necessity, not a luxury.

In today's media landscape—where millions of images are produced every day, and synthetic generation techniques can create photorealistic content with no basis in reality—source-level authentication is not merely a technical feature; it is an epistemological requirement for trusting digital imagery [38].

The model proposed here does not aim to replace traditional forensic techniques but to complement them from the moment of acquisition, generating digital evidence that is objectively verifiable, universally interpretable, and resistant to tampering.

Adopting cryptographic authentication mechanisms that are integrated into the image file itself, standardized, and openly accessible may be key to preserving the legal and documentary validity of digital images in a future dominated by visual ambiguity.

Appendix A. JPEG Authentication System Pseudocode

This appendix presents the core pseudocode routines used in the proposed model for cryptographic authentication of JPEG images at capture time.

A.1 Secure RSA Key Pair Generation

```
FUNCTION GenerateRSAKeys(bitLength):  
    keyPair ← RSA.Generate(bitLength)  
    publicKey ← keyPair.GetPublicKey()  
    WriteToFile("public.pem", publicKey, format = "PEM")  
    IF TPMIsAvailable() THEN  
        TPM.StorePrivateKey(keyPair.PrivateKey)
```

A.2 Secure Chain Signing via MOBILE TPM

```
FUNCTION SecureSignChain(chain):  
    hash ← ComputeHash(chain) // Use SHA-256 in production  
    IF TPMIsAvailable() THEN  
        signature ← TPM.Sign(IMEI, hash) // Signing done inside TPM  
    encodedSignature ← Base64Encode(signature)  
    RETURN encodedSignature
```

A.3 Cryptographic Seal Generation

```
FUNCTION GenerateSeal(jpegImage):  
    imei ← GetDeviceIMEI()  
    content ← ExtractVisualContent(jpegImage) // Between SOS and EOI  
    hash ← ComputeHash(content) // SHA-256  
    data ← Concatenate(imei, hash)  
    signature ← SecureRSASignChain(data)  
    seal ← Package(imei, hash, signature)  
    RETURN seal
```

A.4 Inserting the Seal into the JPEG Image

```
FUNCTION InsertSealIntoJPEG(jpegImage, seal):  
    comSegment ← CreateCOMSegment(seal)  
    newImage ← InsertCOMSegment(jpegImage, comSegment)  
    RETURN newImage
```

A.5 Verification of an Authenticated Image

```
FUNCTION VerifyImage(jpegImage, manufacturerPublicKey):  
    seal ← ExtractSeal(jpegImage)  
    content ← ExtractVisualContent(jpegImage)  
    currentHash ← ComputeHash(content)  
    verifiedData ← Concatenate(seal.imei, seal.hash)  
    signatureValid ← VerifyRSA(seal.signature, verifiedData, manufacturerPublicKey)  
  
    IF currentHash = seal.hash AND signatureValid = TRUE THEN  
        RETURN "Authentic"  
    ELSE  
        RETURN "Invalid"
```

A.6 Extraction of the Seal from the COM Marker

```
FUNCTION ExtractSeal(jpegImage):  
    comSegment ← FindMarker(jpegImage, COM)  
    imei ← ReadIMEI(comSegment)  
    hash ← ReadHash(comSegment)  
    signature ← ReadSignature(comSegment)  
    seal ← CreateSealObject(imei, hash, signature)  
    RETURN seal
```

A.7 Automated Forensic Classification

```
FUNCTION ClassifyImage(jpegImage, publicKey):  
    IF NOT HasCOMSegment(jpegImage) THEN  
        RETURN "Questionable"  
  
    seal ← ExtractSeal(jpegImage)  
    result ← VerifyImage(jpegImage, publicKey)  
  
    IF result = "Authentic" THEN  
        RETURN "Authentic"  
    ELSE IF seal.hash ≠ ComputeHash(ExtractVisualContent(jpegImage)) THEN  
        RETURN "Tampered"  
    ELSE IF seal.imei ≠ ExpectedIMEI THEN  
        RETURN "Partially Valid"  
    ELSE  
        RETURN "Questionable"
```

Declaration of competing interest

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

References (IEEE style for IJERT)

- [1] 3rd Generation Partnership Project, Technical Specification Group Core Network and Terminals; Numbering, Addressing and Identification (Release 18), 3GPP TS 23.003 v18.0.0, 2024. [Online]. Available: https://www.3gpp.org/ftp/Specs/archive/23_series/23.003/23003-i00.zip
- [2] N. A. Abdullah, R. Ibrahim, K. M. Mohamad, and N. A. Hamid, "Carving linearly JPEG images using unique hex patterns (UHP)," in Proc. 1st Int. Conf. on Advanced Data and Information Engineering (DaEng-2013), Lecture Notes in Electrical Engineering, vol. 285, pp. 365–374, Springer, 2014. DOI: 10.1007/978-981-4585-18-7_33
- [3] Adobe Inc., Adobe Content Authenticity (Beta), 2025. [Online]. Available: <https://helpx.adobe.com/creative-cloud/apps/adobe-content-authenticity.html>
- [4] Android Developers, TelephonyManager.getId(), 2025. [Online]. Available: <https://developer.android.com/reference/android/telephony/TelephonyManager#getIdMei/> (accessed Aug. 7, 2025).
- [5] K. Balan, R. Learney, and T. Wood, "A framework for cryptographic verifiability of end-to-end AI pipelines," arXiv preprint arXiv:2503.22573v1, 2025. [Online]. Available: <https://arxiv.org/abs/2503.22573>
- [6] E. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet, 3rd ed., Academic Press, 2011.
- [7] C. Chen, X. Zhao, and M. C. Stamm, "Detecting anti-forensic attacks on demosaicing-based camera model identification," in Proc. IEEE Int. Conf. on Image Processing (ICIP 2017), pp. 1512–1516, IEEE, 2017.
- [8] Coalition for Content Provenance and Authenticity, C2PA Technical Specifications, Version 2.2, 2024. [Online]. Available: <https://spec.c2pa.org/specifications/specifications/2.2/index.html>
- [9] European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (GDPR), Official Journal of the European Union, L 119, 2016. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:02016R0679-20160504>
- [10] S. Faraz, "Analysis of international numbers of a cell phone," J. Inf. Syst. Commun., vol. 3, no. 1, pp. 243–246, 2012.
- [11] H. Farid, "Image forgery detection: A survey," IEEE Signal Process. Mag., vol. 26, no. 2, pp. 16–25, 2009. DOI: 10.1109/MSP.2008.931079
- [12] D. P. Ganwar and A. Pathania, "Authentication of digital image using Exif metadata and decoding properties," Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol., vol. 3, no. 8, pp. 335–341, 2018. DOI: 10.32628/CSEIT183815
- [13] E. Hamilton, JPEG File Interchange Format (Version 1.02), C-Cube Microsystems, 1992. [Online]. Available: <https://www.w3.org/Graphics/JPEG/jfif3.pdf>
- [14] International Organization for Standardization, Information Technologies — JPEG Systems — Part 5: JPEG Universal Metadata Box Format (JUMBF), ISO/IEC 19566-5:2023, 2023. [Online]. Available: <https://www.iso.org/standard/84635.html>
- [15] International Telecommunication Union (ITU), International Mobile Equipment Identity (IMEI) Database Specifications, 2023. [Online]. Available: <https://www.itu.int/en/ITU-T/publications>
- [16] ISO/IEC JTC 1/SC 29/WG 1, Use Cases and Requirements for JPEG Trust (Version 1.1), WG1-N100719-102, Technical Report, 102nd Meeting of the Joint Photographic Experts Group, San Francisco, 2024. [Online]. Available: https://ds.jpeg.org/documents/jpegtrust/wg1n100719-102-REQ-Use_Cases_and_Requirements_for_JPEG_Trust_v1_1.pdf
- [17] M. Iuliani, M. Fontani, and A. Piva, "A leak in PRNU-based source identification—Questioning fingerprint uniqueness," IEEE Access, vol. 9, pp. 52455–52463, 2021. DOI: 10.1109/ACCESS.2021.3070478

- [18] J. Jonsson and B. Kaliski, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1 (RFC 3447), Internet Engineering Task Force, 2003. DOI: 10.17487/RFC3447
- [19] Y. Khalid, "Understanding and decoding a JPEG image using Python," Yasoob Khalid, 2020. [Online]. Available: <https://yasoob.me/posts/understanding-and-writing-jpeg-decoder-in-python/>
- [20] M. Kirchner, "Efficient estimation of CFA pattern configuration in digital camera images," in Media Forensics and Security II, SPIE Proc., vol. 7541, Article 75410W, 2010. [Online]. Available: https://ws.binghamton.edu/kirchner/papers/2010_SPIE_CFA.pdf
- [21] O. Laurent, "Study exposes social media sites that delete photographs' metadata," 1854 Photography, 2013. [Online]. Available: <https://www.1854.photography/2013/03/study-exposes-social-media-sites-that-delete-photographs-metadata>
- [22] X. Lin, J.-H. Li, S.-L. Wang, A.-W.-C. Liew, F. Cheng, and X.-S. Huang, "Recent advances in passive digital image security forensics: A brief review," Engineering, vol. 4, no. 1, pp. 29–39, 2018. DOI: 10.1016/j.eng.2018.02.008
- [23] Manisha, C.-T. Li, X. Lin, and K. A. Kotegar, "Beyond PRNU: Learning robust device-specific fingerprint for source camera identification," Sensors, vol. 22, no. 20, 7871, 2022. DOI: 10.3390/s22207871
- [24] N. Muhammad, M. Hussain, G. Muhammad, and G. Bebis, "Copy-move forgery detection using dyadic wavelet transform," in Proc. 8th Int. Conf. on Computer Graphics, Imaging and Visualization (CGIV 2011), pp. 103–108, IEEE, 2011. DOI: 10.1109/CGIV.2011.29
- [25] S. M. Myat and M. T. Kyaw, "Analysis of Android applications by using reverse engineering techniques," Int. J. Innov. Sci. Res. Technol., vol. 4, no. 3, pp. 551–558, 2019. [Online]. Available: <https://www.ijisrt.com/analysis-of-android-applications-by-using-reverse-engineering-techniques>
- [26] National Institute of Standards and Technology (NIST), Secure Hash Standard (SHS), FIPS PUB 180-4, 2015. DOI: 10.6028/NIST.FIPS.180-4
- [27] A. Naveh and E. Tromer, "PhotoProof: Cryptographic image authentication for any set of permissible transformations," in Proc. IEEE Symp. on Security and Privacy (SP 2016), pp. 255–271, IEEE, 2016. DOI: 10.1109/SP.2016.23
- [28] W. B. Pennebaker and J. L. Mitchell, JPEG Still Image Data Compression Standard, Kluwer Academic Publishers, 1992.
- [29] B. Sarma and G. Nandi, "A study on digital image forgery detection," Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 4, no. 5, pp. 878–882, 2014.
- [30] J. Schneider, "Leica and Nikon adding content authenticity tech into their cameras," PetaPixel, 2022. [Online]. Available: <https://petapixel.com/2022/10/18/leica-and-nikon-adding-content-authenticity-tech-into-their-cameras>
- [31] M. S. Smith, "This Leica camera stops deepfakes at the shutter," IEEE Spectrum, 2023. [Online]. Available: <https://spectrum.ieee.org/leica-camera-content-credentials>
- [32] Sony Corporation, Camera Authenticity Solution, 2025. [Online]. Available: <https://authenticity.sony.net/camera/es-es/index.html>
- [33] X. Sun, X. Chen, L. Li, H. Cai, J. Grundy, J. Samhi, T. F. Bissyandé, and J. Klein, "Demystifying hidden sensitive operations in Android apps," arXiv preprint arXiv:2210.10997, 2022. DOI: 10.48550/arXiv.2210.10997
- [34] F. Temmermans and L. Rosenthal, "Adopting the JPEG universal metadata box format for media authenticity annotations," in Applications of Digital Image Processing XLIV, vol. 11842, Article 118420M, SPIE, 2021. DOI: 10.1117/12.2597651
- [35] Times of India Tech Desk, "How to remove location and other EXIF data from photos on Android, Windows, macOS, and iPhone," The Times of India, 2024. [Online]. Available: <https://timesofindia.indiatimes.com/technology/tech-tips/how-to-remove-location-and-other-exif-data-from-photos-on-android-windows-macos-and-iphone/articleshow/114317609.cms>
- [36] Trusted Computing Group, Trusted Platform Module Library, Family 2.0 (TPM 2.0) Specification: Parts 1–4, 2013. [Online]. Available: <https://trustedcomputinggroup.org/resource/tpm-library-specification/>
- [37] V. van der Meer, J. van den Bos, H. Jonker, and L. Dassen, "Problem solved: A reliable, deterministic method for JPEG fragmentation point detection," Forensic Sci. Int.: Digital Investigation, vol. 48 (Suppl.), 301687, 2024. DOI: 10.1016/j.fsidi.2023.301687
- [38] J. Xiao, Z. Sun, H. An, H. Zhao, M. Qiu, and X. Li, "Optical image processing and applications empowered by vision-language models," iOptics, vol. 1, no. 1, Article 100003, 2025. DOI: 10.1016/j.iopt.2025.100003
- [39] P. Zhou, X. Han, V. I. Morariu, and L. S. Davis, "Learning rich features for image manipulation detection," in Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), pp. 1053–1061, IEEE, 2018. DOI: 10.1109/CVPR.2018.00116