Prolonging Anti-Deepfake Signatures Lifetime with Blockchain-Based Timestamps

Sohaib Saleem[1,2] and Pericle Perazzo[1]

[1]Department of Information Engineering, University of Pisa,
Via G. Caruso 16, 56122 Pisa, Italy
[2]Department of Information Engineering, University of Florence,
Via Santa Marta 3, 50139 Florence, Italy

Abstract. As AI-generated synthetic media, such as deepfake images, proliferate, verifying the authenticity of digital images has become a significant challenge. Traditional digital signature techniques become invalid if images are cropped; therefore, special croppable signatures have been proposed in the literature. However, both traditional and croppable signatures remain valid only as long as their associated public key certificate remains valid. This could be problematic for authenticated images, as they often circulate over the Internet for long periods of time, beyond the expiration of their public key certificates. Re-signing each image with a new key requires re-distributing all affected images, and this may be impractical for large-scale systems. To address this issue, we propose an image authentication system with croppability and post-expiration validity features, using BLS (Boneh–Lynn–Shacham) short signatures, the Ethereum blockchain as a decentralized trusted timestamping service, and IPFS (InterPlanetary File System) as a decentralized storage solution. Additionally, we employ two methods: a baseline method, in which the web server hosting the images does not pay any transaction fees, and an optimized method, which produces very little traffic on the web browser. Experimental evaluations are conducted in Pakistan and Italy under real Wi-Fi and simulated 4G cellular connections using Linux traffic control (tc) to demonstrate the system's performance. Results showed that, in the baseline method, the network traffic overhead and communication delay increase linearly with the image size. Meanwhile, the optimized method achieves constant-time performance for retrieval and verification.

Keywords: Image authentication, BLS signatures, blockchain, decentralized timestamping, IPFS.

1 Introduction

In today's world, with the rise of synthetic media and algorithmically modified content created by Generative Adversarial Networks (GANs), protecting the authenticity of digital images has become an essential concern. The growth of deepfakes, such as fake images and videos, is and will be used for evidence fabrication, impersonation, and spreading misinformation, which can potentially lead to social and legal consequences [1–3]. These kinds of threats are increasing in frequency and severity as generative models continue to advance, exposing the limitations of traditional verification methods, such as metadata inspection, digital watermarking, and manual forensic analysis, especially when visual media circulates without a known or trusted source [4].

David C. Wyld et al. (Eds): CNSA, CCSEIT, NLAI, SIGML, ICCIOT, AIAP – 2026
pp. 11-30, 2026. CS & IT - CSCP 2026 DOI: 10.5121/csit.2026.160502

To address these challenges, researchers have developed various machine learning and cryptographic techniques to detect deepfakes and prevent tampering. For example, deepfake detection algorithms based on machine learning rely on statistical pattern recognition, which is trained on existing manipulation patterns; they may not perform well against more advanced or novel forgeries. Alternatively, cryptography-based provenance verification approaches are more reliable in distinguishing authentic from synthetic images [5, 2, 6].

When applied to images, existing digital signatures such as RSA or ECDSA face the challenge of maintaining their validity even after minor transformations. These systems are inherently fragile because even a slight modification to an image at the bit level can invalidate signatures [7]. In a prior study [8], Perazzo et al. introduced *croppable signatures* that can authenticate JPEG images, thereby enabling verification after cropping, which is a typical manipulation on online platforms. However, this approach lacked a mechanism to ensure the long-term validity of signatures. In fact, when public key certificates have expired or been revoked, the authenticity of the images becomes de facto unverifiable. This issue is particularly exacerbated in use cases such as digital forensics, journalism, and legal documentation, where maintaining the authenticity of an image is crucial in the long run. Re-signing each image with a new key after certificate expiration requires re-distributing all affected images and their updated signatures, which is an impractical burden. To ensure long-term authenticity without re-signing, a reliable timestamping service and secure metadata storage are necessary; otherwise, it becomes impossible to verify when a particular signature was created at a given time or to prove the signing time of an image. According to the validity model proposed by [9], referred to as "modified shell model" in [10], a digital signature is trustworthy only if it was generated during the validity period of the related certificate. A timestamping scheme that securely associates the signing time with the image data is the primary gap that we address in the present paper.

Blockchain technologies offer a solid foundation for trusted and distributed timestamping. With features like immutability and transparency, especially public ledgers like Ethereum, they enable validity over the long-term in an auditable and tamper-resistant manner. This can be achieved by timestamping the signature metadata on the blockchain, which records when the signing occurred, ensuring verifiable proof of signing time without the need for a trusted third party. [11]. Embedding this metadata into the blockchain ensures that any verifier can confirm when the signatures of an image existed, even if the certificate has expired or been revoked. Simultaneously, the IPFS (InterPlanetary File System) serves as a distributed, peer-to-peer, and content-addressed storage layer where actual signature files are stored, ensuring signatures remain accessible over a long period of time without depending on a single host [12, 13]. Unfortunately, it is not trivial to times-

tamp a croppable signature on the Ethereum while maintaining good performance, because it grows linearly with the image.

This paper designs and evaluates a decentralized image authentication system that ensures the long-term validity of croppable signatures without re-signing, even if the corresponding public key certificate has expired or been revoked, and the signing key is compromised. We utilized an Ethereum-based timestamping scheme for the temporal anchoring of signature metadata, allowing the verification of the signature's generation time. In light of the evolving regulatory landscape, our approach aligns with recent efforts to integrate blockchain-based timestamping into legal frameworks such as eIDAS regulation. According to Sorge et al. [14], decentralized timestamps can meet the requirements of advanced electronic signatures under eIDAS, ensuring trust, persistence, and integrity. Two distinct methods are introduced in our work: a baseline method, in which the verifier (for example a web browser acting as an image consumer) must download data proportional to the image size from IPFS but the web server does not need to pay the transaction fee for storing metadata on Ethereum; and an optimized method, in which the web server must upload data to Ethereum, thus incurring in transaction fees, but the browser downloads only constant-size data. These methods facilitate verification with different latency and bandwidth trade-offs. Finally, we perform a series of experimental evaluations for both methods to assess the network traffic overhead and the communication delay on the browser side over a simulated 4G cellular network (emulating mobile verifier conditions) and a real Wi-Fi environment. Moreover, our timestamping approach is general and can also be applied to recent zk-SNARK–based image authentication schemes [15, 16].

The rest of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces background concepts for our system. Section 4 provides a detailed description of our proposed image authentication system and its technical workflow. Section 5 discusses the threat model. Section 6 presents our experimental performance evaluation and key findings. Finally, the paper concludes by presenting the future research direction in Section 7.

2 Related Work

We divide the related work section into three parts: the first discusses digital signatures for manipulable images (Section 2.1). The second looks at blockchain-based timestamping for the digital image integrity (Section 2.2). Finally, the third part explores zk-SNARK-based image authentication (Section 2.3).

2.1 Digital Signatures for Manipulable Images

The authenticity of an image is traditionally ensured using cryptographic signatures, such as RSA or ECDSA, which verify that the content has not changed by

matching a hash signed with a private key [14, 5]. However, these bit-level schemes are fragile, where even benign operations like resizing or recompression, common on social media and news platforms, can break the signature, flagging legitimate transformations as tampering [7]. Such schemes are highly vulnerable to format-specific breakages, including hash mismatches caused by non-malicious edits, making them unsuitable for lossy formats like JPEG.

Researchers examined robust and content-based signatures to overcome this issue. The early studies proposed homomorphic image signatures, which are able to handle specific image manipulations, such as cropping or recompression, by constructing the signatures based on features, rather than on exact pixel values. Johnson et al. [17] first demonstrated that limited JPEG compression can be used in a signature scheme, considering compression as an allowable transformation. More recently, Erfurth et al. [7] attempted to construct a more formal version of JPEG-compressible signatures, which defined unforgeability with controlled recompression and guaranteed the visual integrity of signed images. Their scheme trades off transformation flexibility (support only JPEG compression, not arbitrary edits) against formal security and efficiency.

An alternative approach is to use perceptual hashing, where the hash only changes if the content of an image undergoes substantial alteration. This means that the hash would remain the same if an image were recompressed or slightly resized. Nevertheless, it has been demonstrated that perceptual hash-based signatures could be unreliable, as all they produce either false positives (accepting a fake image as genuine) or false negatives (rejecting a valid image) at rates that are not insignificant [18]. Even a custom JPEG-aware perceptual hash, such as the one proposed by Lin and Chang [19], can be fooled by tiny, carefully inserted perturbations in an image, allowing manipulated content to be accepted as authentic in certain situations. As a result, content-based methods lack cryptographic strength.

2.2 Blockchain-Based Timestamping

Besides verifying the type of content created, it is also important to authenticate the timing and order of content creation and signing. Trusted timestamping is essential for maintaining the integrity of digital content, often relying on traditional methods that usually depend on centralized TimeStamping Authorities (TSAs). These systems, such as TSA's compliance with the ISO/IEC 18014 standard or Public Key Infrastructure (PKI) services like RFC 3161 [20], work by providing digitally signed confirmation of data existence at a specific time. Unfortunately, these services come with risks, such as a single point of failure and dependency on centralized trust [21]. Even the most widely used standards of timestamping require clients to rely on the availability and reputation of TSAs [20]. To address these challenges, decentralized solutions using blockchain technology have proven to be promising alternatives.

In 2017, Gipp et al. [22] utilized the idea of securely timestamping digital documents with the Bitcoin blockchain through their OriginStamp platform. This method verifies that a scholar's manuscript existed at a specific point in time. A submitted document is first hashed with the CryptSubmit system. Then, a single aggregated hash is created from multiple hashes using a Merkle tree. Finally, the tree root value is recorded on the blockchain by creating a Bitcoin transaction. Their earlier 2016 research [23] demonstrated that the preservation of video integrity is achieved by embedding tamper-proof timestamps through the Bitcoin blockchain. These systems offer public immutability and verifiability, surpassing the trust limitations of centralized TSAs.

However, this setup was only designed for non-manipulable documents and videos. It is not suitable for visual media that can be susceptible to manipulation, such as cropping. In contrast, our approach employs timestamping for the signatures and metadata of images, and it allows for cropping modifications.

Breitinger and Gipp [24] further expanded OriginStamp by addressing issues related to verifiability and long-term accessibility of intellectual property using a decentralized blockchain-based timestamping service. They proposed scenarios where the origin of files that prove the ownership of ideas must be independently verifiable, without relying on cloud or third-party service infrastructure. Again, these solutions are intended solely for archival purposes and do not address media manipulation, which is the focus of our image authentication system.

Chronos^{+} [25] is another blockchain-based timestamping scheme for outsourced data, specifically designed for high timestamp accuracy and cloud storage verification. Chronos^{+} achieves minute-level accuracy in timestamps, making it suitable for time-sensitive applications. Their scheme securely timestamps files hosted by cloud service providers by collecting metadata and recording the final digest along with related cloud storage operations as a transaction on the Ethereum blockchain. Our system does not require high-accuracy timestamping since our timestamps are compared against the validity period of digital certificates, which are typically expressed in days.

Meng et al. [26] introduced a long-term timestamping protocol called BLTTS, which uses blockchain technology to support the renewal of cryptographic algorithms and maintain the integrity of digital assets over time, even after the original cryptographic algorithms become outdated. They utilize Merkle trees, regularly update cryptographic evidence, and employ a chained hash function to address cryptographic algorithm renewal, as well as secure anchoring proofs on a public blockchain like Ethereum. Since our system timestamps BLS signatures, which can be viewed as generic data from the perspective of the BLTTS protocol, it can be seamlessly integrated with it to handle cryptographic algorithm renewals.

2.3 zk-SNARK-Based Image Authentication

Although our system uses BLS short signatures and blockchain-based timestamping to verify the authenticity of an image, existing cryptographic schemes investigate other paradigms. It is worth noting that *zk-SNARKs* (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge) provide a general cryptographic proof framework for image verification that supports complex transformations, including compression, resizing, and blurring. For instance, the PhotoProof [27], introduced by Naveh and Tromer, enables image authentication across many transformations allowed by the transformation commitment. Kang et al. [16] further extended this using the ZK-IMG library, which ensures privacy-preserving verification of image transformations through zk-SNARKs. More recently, Datta et al. [15] developed VerITAS, designed to support a zk-SNARK verification pipeline at a large scale for validating transformation history.

Existing zk-SNARK–based image authentication systems, such as VerITAS and ZK-IMG, mainly verify how an image was transformed, but they do not guarantee the time when the image or proof was generated. Moreover, these systems typically rely on a signing key that may eventually expire or be revoked. Once the signer's key expires or is revoked, an adversary with the old key may forge valid proofs. Our blockchain-based timestamping can complement these approaches too, by anchoring the proof to an immutable time reference on a decentralized ledger to detect post-expiration forgeries. In this way, zk-proofs can still handle transformation verification, while the blockchain ensures long-term validity. Notably, this approach does not require modifying existing zk-circuits and keeps the timestamping process lightweight and independent of the proving logic.

3 Background

In this section, we present the essential technical components that underpin our system. We first discuss croppable signatures, then blockchain-based timestamping and IPFS.

3.1 Croppable Signatures

The concept of croppable signatures [8] aims to verify the authenticity of digitally shared content that has been cropped. Croppable signatures divide an image into distinct graphical blocks that can be verified independently. Each block is signed individually, so it is possible to extract and validate a specific block without needing access to the full signature set. In a croppable signature framework, three different entities interact. First, the *signer* (for example, a camera device) divides the image into fixed-sized blocks and then uses a private key to create *block-wise signatures* for each block. These block-wise signatures are generated using the BLS signature

scheme due to its property of supporting individual signature creation and compact aggregation. All these signatures, along with their metadata, comprise the *full signature*. Next, the *cropper* (for example, a news website) extracts the blocks inside a particular cropping rectangle (identified by cropping coordinates) and collects their respective block-wise signatures. The cropper performs the aggregation operation on these signatures to produce a single, constant-size signature, which is called a *cropped signature*. Finally, the *verifier* (for example, a web browser) verifies the authenticity of every block in the cropping rectangle with the help of the cropped signature and the signer's public key. Any alteration to the cropped image or its cropped signatures will invalidate the verification process, thus guaranteeing the integrity of the cropped image. Please refer to [8] for the mathematical details of the entire process.

3.2 Blockchain Timestamping and IPFS

The distributed consensus algorithm of a blockchain network, such as Ethereum, ensures that once data is stored in the blockchain, it cannot be modified retroactively, thus providing an immutable audit trail of events. This immutability, coupled with its decentralized verification, makes blockchain an ideal foundation for a trusted timestamping service where long-term authenticity is an essential requirement. In practice, timestamping is achieved by embedding a cryptographic hash of data in a blockchain transaction, which is then included in a validated block that carries a consensus-agreed timestamp (in the block header). Once confirmed on the blockchain (e.g., Ethereum), this timestamp serves as verifiable proof that the record existed at or before the recorded block time. Of course, to timestamp and store data, an entity interacting with the blockchain (e.g., uploading data) must pay transaction fees to miners or validators. Ethereum timestamps, embedded in block headers, may drift slightly from the real time due to miner manipulation. However, the bounded range is typically up to 2 hours in the worst case due to network consensus variability [28].

The InterPlanetary File System (IPFS) is a decentralized peer-to-peer storage protocol that identifies files using cryptographic hashes of the file content called "content identifiers" (CIDs). In contrast to the conventional web architecture, which uses location-based addressing, IPFS retrieves content from any node with a matching CID, while ensuring redundancy and data integrity.

4 Proposed System

In this section, we describe the architecture and cryptographic features of our image authentication system, which aims to provide verifiable image signatures even in the case of cropping transformations and after the public key certificate corresponding

to the signing key expires or is revoked. Our proposed scheme presents a decentralized system consisting of croppable cryptographic signatures, content-addressable distributed storage, and blockchain-based timestamping. This integration guarantees that the authenticity of the cropped image can be verified in the long term, even when the public key certificate is no longer valid, without re-signing or relying on any centralized authority.

The system model consists of three main entities, as shown in Fig. 1: the signer, who is responsible for authenticating the original image content; the cropper, which extracts the particular region of the image based on the cropping coordinates required for further distribution; and the verifier, who checks the authenticity of the cropped image.

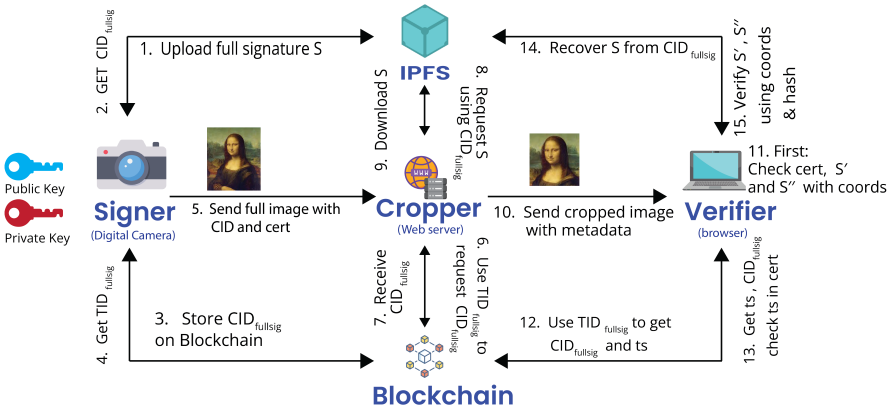

Fig. 1. System model (baseline method).

The fundamental building blocks of croppable signatures rely on bilinear pairing functions [29]. Let G_1 and G_2 be two additive cyclic groups, each with prime order r, and let G_T be the multiplicative target group under a bilinear pairing $e : G_1 \times G_2 \rightarrow G_T$, and a generator $P_2 \in G_2$. Every image is divided into a matrix of $w \times h$ blocks. Let x_{ij} represent the block pixel data at the position (i, j), and $H : \{0, 1\}^* \rightarrow G_1$ be a cryptographic hash function that maps each image block message to an element of G_1. Let k be the private key of the signer, and pk be the corresponding public key. For such keys, the signer can use any digital signature scheme, like RSA or ECDSA, as long as it provides two core algorithms: a signing function $Sign_k$, which takes as input the message and private key k and returns a signature, and a corresponding verification function $Verify_{pk}$, which checks the validity of the signature using the public key pk. The signer generates a unique *ephemeral key pair* (k_e, pk_e), where k_e is a randomly chosen *ephemeral private key* from $\mathbb{Z}_r^*$, and $pk_e = k_e \cdot P_2$ is the corresponding *ephemeral public key*. We used the PBC library [30] configured with

an MNT curve over a 512-bit prime field with embedding degree equal to 2, which corresponds to approximately 80 bits of security. Signatures were generated using the BLS short signature scheme, and both public and private keys were initialized using PBC's built-in "type A" parameter sets. Different ephemeral keys are used for different images to ensure that block-wise signatures are tightly bound to the specific image they were generated for, thereby preventing an adversary from taking a block from one image along with its block-wise signature and inserting it into another image.

The proposed system deploys two methods for guaranteeing the long-term authenticity of the cropped images: the *baseline method* and the *optimized method*. In the baseline method, the signer's responsibility is to publish metadata on the blockchain and pay the related transaction fees. On the other hand, the optimized method shifts this responsibility to the cropper, allowing for more efficient verification in case of expired or revoked certificates by enabling the verifier to download only constant-sized data ($\mathcal{O}(1)$). These two methods are designed to investigate a trade-off between traffic cost (i.e., lowering the verifier's overhead) and monetary cost (i.e, limiting the cropper's payment for transaction fees). The following Section 4.1 explains the baseline method. Then, a comparative description of the optimized method is presented in Section 4.2.

4.1 Baseline Method

Full Signature Construction The process begins with the signer, who produces or receives the full image and divides it into uniformly sized x_{ij} graphical blocks, where $i \in [1, h]$ and $j \in [1, w]$ are their block indices. Our system does not impose strict constraints on the dimensions of individual blocks. The choice of the block size impacts the granularity of potential cropping operations; finer blocks allow for more precise cropping, whereas coarser blocks reduce the size of the full signature. On the other hand, the size of the cropped signature is constant and independent from the block size. Each block is signed separately using the BLS signature scheme with the ephemeral private key to produce a matrix of $w \times h$ block-wise signatures:

$$S_{ij} = k_e \cdot H(i \,|\, j \,|\, x_{ij}) \in G_1, \tag{1}$$

where '|' indicates concatenation. This guarantees that any change to the content or position of the block will invalidate the associated block-wise signatures. After signing all the blocks, the signer runs the $Sign_k$ function to generate a full signature on an image using his private key k. Specifically, it calculates the *binding signature* as:

$$S' = Sign_k(w \,|\, h \,|\, pk_e), \tag{2}$$

which binds the ephemeral public key pk_e and the image dimensions to the signer's identity. The full signature S is composed as the binding signature plus all the

block-wise signatures:

$$S = \left(S',\ \{S_{ij}\}_{\forall\ 1\le i\le h,\ 1\le j\le w}\right). \tag{3}$$

Signature Storage and Timestamping When the signer has generated the full set of block-wise signatures S_{ij} and the binding signature S', he stores them in a single file S that it uploads to IPFS. The associated content identifier $CID_{fullsig}$ is returned by the IPFS network upon uploading S, and it ensures retrievability and integrity. The signer then issues a transaction on the blockchain that stores $CID_{fullsig}$. After the $CID_{fullsig}$ gets stored on the blockchain, the signer receives a unique transaction identifier $TID_{fullsig}$. Note that the 2-hour bounded error in blockchain-based timestamping [28] is acceptable for our application, since the verifier only uses these timestamps for certificate expiration checks, which typically requires only day-level precision. Then the signer embeds $TID_{fullsig}$ with the certificate *cert* of the signer's public key within the full image, for example, in a comment section in case of JPEG format [8]. Note that the public IPFS nodes do not guarantee the long-term availability of data. Therefore, the signer could adopt reliable IPFS pinning services or decentralized storage overlays like Filecoin or Arweave to ensure the persistent availability of the full signature even after certificate expiration.

Note that the signer has to pay some fees to issue the transaction on the blockchain. To reduce such a fee, our system can adopt Merkle tree batching. Specifically, the signer may combine multiple full-image signatures in a single Merkle tree, upload them to IPFS along with their Merkle proofs, and timestamp only the Merkle root on the blockchain. The verifier then fetches the Merkle root from the blockchain and uses the IPFS-hosted Merkle proof to validate the presence of a given signature in the Merkle tree. For example, if the system uses Ethereum, storing a 32-byte CID costs 41,000 gas [31], which at a fee rate of $\sim 2 \times 10^{-9}$ ETH[1] per gas and an ETH price of 2,200 USD[2] corresponds to ≈0.18 USD per image. Thus, anchoring 1,000 images individually would cost ≈180 USD, whereas batching via a Merkle root reduces this to a single on-chain transaction (≈0.18 USD), providing ∼1000× cost savings.

Cropped Signature Construction The cropper downloads the full signature S from IPFS using the $CID_{fullsig}$ (after retrieving $TID_{fullsig}$ from the full image's comment section and getting $CID_{fullsig}$ from Ethereum) and crops a rectangular (submatrix) region from the full image, defined by cropping coordinates (i_1, j_1) to (i_2, j_2). Here, the i_1 and j_1 indices correspond to the top-left block and i_2 and j_2

[1] Etherscan Gas Tracker, `https://etherscan.io/gastracker`, visited on 2026-02-05, considering 1 Gwei = 10^{-9} ETH.

[2] CoinMarketCap, `https://coinmarketcap.com/currencies/ethereum/`, visited on 2026-02-05.

refers to the bottom-right block of the cropping rectangle[3]. It retrieves the relevant subset of image blocks and their associated block-wise signatures S_{ij} from the full signature. Then, the cropper calculates the *aggregated signature* S'' for the cropped region using:

$$S'' = \sum_{i_1 \le i \le i_2,\, j_1 \le j \le j_2} S_{ij} \in G_1. \tag{4}$$

This aggregation highlights the advantages of the linearity of BLS signatures during pairing operations. Therefore, the cropped signature S_{crop} ultimately consists of the pair:

$$S_{crop} = (S', S''). \tag{5}$$

Where S' authenticates the ephemeral public key pk_e and S'' authenticates the cropped content, which is the aggregation of all the block-wise signatures associated with the blocks of the image that fall inside the cropping rectangle, respectively, in (i_1, j_1) and in (i_2, j_2). Note that the cropper does not need to access the private key of the signer because the proposed system design allows for signature aggregation as a public operation. The cropper puts the cropped signature, the cropping coordinates, the signer's public key certificate, and $TID_{fullsig}$ inside the cropped image, for example, in a comment section in case of JPEG format. Finally, it sends the cropped image to the verifier.

Cropped Signature Verification The verifier validates the long-term authenticity of a cropped image and its corresponding signature by fetching the relevant metadata (S_{crop}, (i_1, j_1, i_2, j_2), *cert*, $TID_{fullsig}$) embedded in the cropped image. For verifying the cropped image, the verifier first verifies the signer's public key certificate *cert* (this includes verifying the CA's trustworthiness, the certificate's signature, etc.), then it checks its temporal validity, meaning, it is not expired or revoked at the present time. Then, two alternative cases are foreseen.

If the certificate is not expired nor revoked, then the verifier uses the $Verify_{pk}$ function to check the authenticity of the binding signature with the public key by verifying:

$$Verify_{pk}\left(S',\, w \,|\, h \,|\, pk_e\right) \stackrel{?}{=} True. \tag{6}$$

Afterwards, the verifier checks the aggregated signature S'' by using pairing operations:

$$e(S'', P_2) = \prod_{i_1 \le i \le i_2,\, j_1 \le j \le j_2} e(H(i \,|\, j \,|\, x_{ij}),\, pk_e)\,. \tag{7}$$

Note that in this case, the verifier downloads only constant-size data to verify the image, and it does not interact with any blockchain or distributed content-addressed filesystem.

[3] Of course, in case the cropper wants to publish the image as a whole, he can set $(i_1, j_1) = (0, 0)$ and $(i_2, j_2) = (h, w)$.

Otherwise, if the certificate is expired or revoked, the verifier uses the embedded $TID_{fullsig}$ to query the blockchain network and download the relevant metadata fields, such as the identifier of the full signature $CID_{fullsig}$, and the associated timestamp ts, which represents the time at which the signer generated and stored the signature on the blockchain. The verifier performs this step to ensure that the retrieved timestamp falls within the validity period of the public key certificate, meaning the certificate was not expired or revoked at that time. The verifier then downloads the full signature identified by $CID_{fullsig}$ from the IPFS network, re-crops it based on the cropping rectangle (i_1, j_1, i_2, j_2) specified in the cropped image by using Equation 4, and recomputes the aggregated signature S''. Then it compares the resulting signature S'' with the embedded S'' in the image and validates the aggregated signature to verify its authenticity. Once the aggregated signature's integrity is verified, the verifier runs Equation 6 specifically to confirm that the ephemeral public key pk_e used to generate individual block signatures S_{ij} was indeed created and signed by an authorized signer, and compares the recomputed S' with the S' that was stored on IPFS.

Notably, the baseline method does not require the cropper to issue transactions on the blockchain; therefore, it incurs no transaction fees. This is a non-negligible advantage, since in the typical scenario, there is a single signer but many croppers, each of which may crop the same image with different cropping rectangles.

4.2 Optimized Method

The baseline method may incur non-negligible costs in terms of data transmission for the verifier, which in case the signer's certificate is expired or revoked and must download the full signature from IPFS, whose size is proportional to the number of blocks in the image. To minimize this traffic overhead, we present an optimized method in which the cropper uploads the cropped signature to the IPFS and commits the associated $CID_{cropsig}$ to the blockchain. This change in the system design enables the verifier to fetch only the cropped signature from IPFS in $\mathcal{O}(1)$ time, instead of downloading the full signature.

In this method, the signer's responsibilities remain unchanged, such as the generation of a BLS signature for each block x_{ij} and its upload to IPFS. Note that the format of the full image and the cropped image remains the same as in the baseline method, except that the TID included in the cropped image is now referring to the transaction issued by the cropper and not by the signer.

If the signer's certificate is expired or revoked, the verifier then fetches only S_{crop} from IPFS, along with relevant metadata such as $CID_{cropsig}$ and the timestamp ts from the blockchain network after retrieving the associated $TID_{cropsig}$ from the cropped image. He then verifies whether the S_{crop} downloaded from IPFS matches the one embedded in the cropped image. After that, he verifies S'' by using Equation 7, and S' by using Equation 6.

The optimized method significantly reduces the network load on the verifier side by transferring blockchain transaction costs to the cropper and decreasing the retrieval size. This method is particularly suitable for real-world applications involving environments where the bandwidth is limited, such as mobile clients or IoT. However, it requires the croppers to issue transactions on the blockchain, and thus, they must pay the associated transaction fees.

Note that the system is flexible enough to allow two croppers to use different cropping methods, either the baseline or the optimized one, for the same full image signed by the same signer. In this setup, both the signer and the optimized-method cropper upload their respective CIDs to the blockchain (respectively $CID_{fullsig}$ and $CID_{cropsig}$). After the signer's certificate expires or is revoked, the verifiers that download the cropped image from a baseline-method cropper will incur a linear overhead. In contrast, the ones that download it from an optimized-method cropper will incur a constant overhead.

5 Threat Model

Targeting the long-term authenticity of the block-wise signatures, we analyze two threats: pre-expiration forgery attack and post-expiration forgery attack.

A *pre-expiration forgery attack* refers to a situation in which an adversary tries to create a deepfake image with a valid signature when the certificate is still valid, without knowing the signer's private key. To achieve this, he can sign a maliciously generated ephemeral key into a deepfake image, link it to the legitimate signer, and finally upload all the manipulated data to IPFS and blockchain. Such manipulation cannot fool the verifier into accepting a deepfake image as authentic. This is because he verifies the link between the ephemeral public key, the image dimensions, and the signer's public key (see Equation 6), and the adversary cannot generate a valid binding signature without knowing the private key. The verification of a forged full or cropped image will fail because the valid S' cannot be recreated without the original private key of the signer. This prevents unauthorized entities from creating deepfake images during the validity period of the signer's certificate, and even if the IPFS and Ethereum contain manipulated or injected metadata, the forged image is rejected.

A *post-expiration forgery attack* refers to a situation in which the certificate of the signer has expired or been revoked, and an adversary has somehow compromised the signer's private key, and he tries to create a deepfake image with a valid signature. To achieve this, the attacker first generates a malicious ephemeral key pair and, using the compromised private key, he then creates malicious block-wise signatures and a malicious binding signature. Then the attacker can upload malicious S as a full signature to IPFS and store the relevant $CID_{fullsig}$ on the blockchain. The forged S' authenticates a malicious ephemeral public key, which leads to successful verification. This attack is avoided by the blockchain-based timestamping.

During verification, the verifier downloads the CID and ts from the blockchain network, retrieves the full signature S from IPFS, conducts security checks, and makes sure that S was generated before the certificate's expiration or revocation. Therefore, the attacker cannot backdate the timestamp of his malicious S' with an earlier timestamp. This process ensures that forged signatures produced when the certificate is expired or revoked are rejected.

Note that if the private key is compromised *before* the certificate's expiration, the attacker could indeed forge a validly signed deepfake, so it is important to promptly revoke the certificate as soon as the compromise is discovered, and the revocation timestamp must be precedent to the compromise event. This, of course, is generally true in all public-key infrastructures.

6 Experimental Evaluation

We conducted a series of experiments to assess the practical feasibility and effectiveness of our decentralized image long-term verification system, focusing on network traffic overhead, overall communication delay, and system performance under constrained connectivity requirements. All the tests were performed from the perspective of the verifier, typically in environments where the verifier uses mobile or desktop devices to surf the web. We used the Ethereum Sepolia testnet for conducting all the experiments. Full implementation details, including the source code and parameter configurations, are available upon request for reproducibility. Since the image verification occurs after the possible cropping and transmission, our evaluation focuses on the responsiveness and communication efficiency of the verifier.

6.1 Network Traffic Overhead

The initial part of the assessment involves measuring the network traffic generated by the verifier to obtain the cryptographic proofs required to verify the authenticity of images using the two methods explained in Section 4, under the assumption that the signer's certificate has expired. In the baseline method, the verifier retrieves the entire set of block-wise signatures from IPFS after querying the Ethereum blockchain for the corresponding $CID_{fullsig}$. This process includes four network interactions: requesting the $CID_{fullsig}$ from the blockchain network, receiving the response, sending the $CID_{fullsig}$ to IPFS, and downloading the full signature S. In the optimized method, on the other hand, the verifier retrieves only the cropped signature from IPFS, thus eliminating the need to download all the block-wise signatures.

To measure the traffic in both methods, we developed a prototype and employed `tcpdump` to record the packet-level data during verifier operations and analyze the generated *.pcap* (packet capture) files with *wireshark*. Each interaction was broken

down into outgoing request bytes and incoming response bytes to determine the used bandwidth. We did not include the traffic to download the image data, so we measured only the overhead due to the verification process. The same tests were repeated with different numbers of image blocks to examine the scalability of both methods. For example, an image with 4800 8×8 blocks roughly corresponds to an image with 640×480 resolution, which is approximately 250KB in the JPEG format. Other test cases varied similarly: a small image of 223×131 (≈82KB in JPEG) contains 476 8×8 blocks, while a larger image, such as 2400×3000 (≈4MB in JPEG), contains 112,500 8×8 blocks. Note that although we used an 8×8 block size in our experiments, which is the native block size of the JPEG format, our system architecture also supports different granularities, such as 16×16 or 32×32. Of course, using larger block sizes will proportionally decrease the total number of blocks, which leads to a smaller size of full signatures and also results in lower overall IPFS upload traffic during the signer phase.

Fig. 2 shows the incoming traffic relative to the number of blocks for both methods.

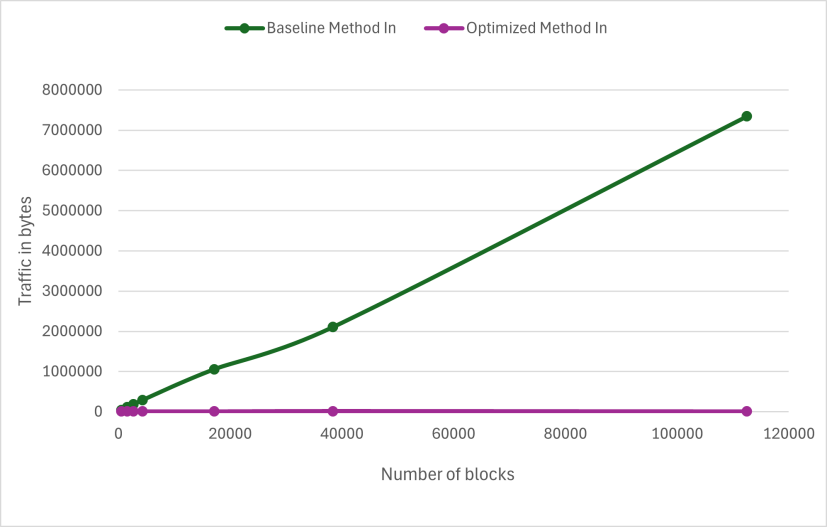

Fig. 2. Incoming traffic in bytes vs number of blocks

We omitted the outgoing traffic from the graph, as it was negligible in our experiments. The figure shows that the baseline method causes a rapid increase in incoming traffic volume as the number of image blocks grows, eventually reaching up to 7.4MB for 112,500 blocks. Conversely, the optimized method maintains a nearly constant bandwidth profile, indicating that the overhead remains minimal, regardless of the block count. These findings confirm the advantage of cryptographic

aggregation performed by the optimized method, which significantly reduces bandwidth requirements.

6.2 Communication Delay Analysis over 4G Networks

In the second part of our evaluation, we examined the total communication delay under a simulated 4G cellular network, which reflects real-world mobile environments, such as when content verification is performed via smartphones or tablets. We utilized the baseline method during all 4G experiments to evaluate the system's performance on the verifier side under different network conditions. This decision is made because, in the baseline method, the number of blocks affects the overall communication delay. Conversely, in the optimized method, the verifier only downloads the fixed-size cropped signature, which keeps the network overhead constant and is less meaningful for the analysis of delay scalability. Again, we assume that the signer's certificate has expired.

To emulate such constrained connectivity conditions, we configured Linux's `tc` (traffic control) utility to simulate a realistic 4G cellular connection. We specifically set a packet loss rate of 0.2%, a downlink and uplink delay of 50ms, along with an extra ±10ms jitter, and set a bandwidth limit of 20Mbps. These parameters were carefully selected based on the recommendations of [32] and reflect typical conditions in 4G networks in urban areas. Additionally, the `tc` tool utilized TBF (Token Bucket Filter) for bandwidth shaping and NetEm (Network Emulation) to introduce packet loss, delay, and jitter. We carried out our experiments in two distinct geographical regions, Pakistan and Italy, to ensure the generalizability of our findings.

The overall communication delay in seconds for different numbers of image blocks in 4G cellular environments is illustrated in Fig. 3.

As expected, the delay increases notably with the number of blocks, due to the larger volume of data retrieved from the IPFS network, with a higher delay observed in Pakistan. This difference can likely be attributed to the smaller number of IPFS nodes in Pakistan, which results in longer content retrieval times. Indeed, from our experience, the main bottleneck in communication latency is fetching data from IPFS, while the interaction time with the blockchain network is relatively negligible. For the highest-resolution image, namely 112,500 blocks, the verification delay exceeds 30 seconds in Pakistan, almost twice as long as in Italy, making real-time feedback in a browser-based or embedded system considerably more challenging. Even with fewer blocks, which allows for faster retrieval, noticeable delays of 7-9 seconds in Pakistan and 5-7 seconds in Italy were observed.

In the optimized method, the cropper uploads only the cropped signature to IPFS, which further decreases IPFS usage compared to full signature uploads in the baseline method. Anyways, for a high-TRL deployment, we suggest a deferred

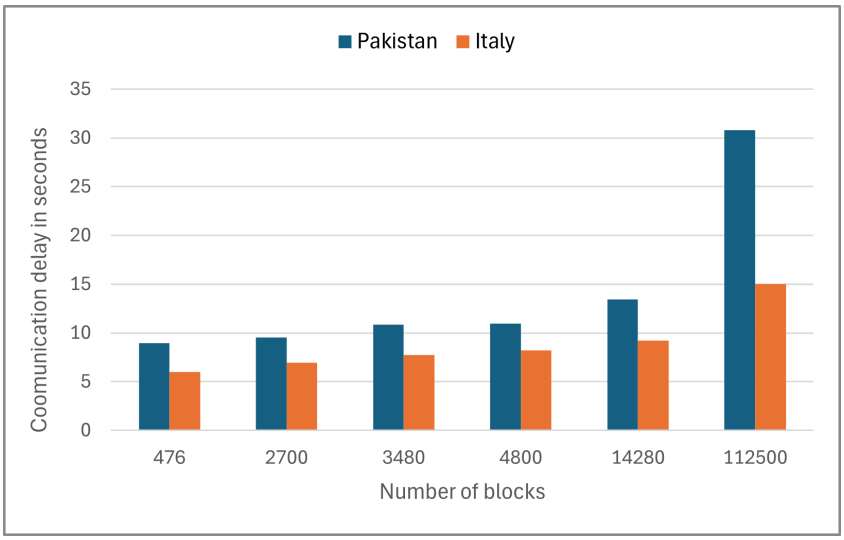

Fig. 3. Communication delay vs number of blocks over cellular 4G environment

verification method to minimize delays without compromising the user's experience, allowing images to be initially rendered without an authentication overlay. Then, verification can happen synchronously in the background. After verification is complete, the verifier can update the display (for example, by adding a green checkmark) to confirm the authenticity of the image.

6.3 Comparison under High-Bandwidth Conditions (Wi-Fi)

In high-throughput environments like Wi-Fi, we re-evaluated the same steps to determine the performance. These experiments simulate realistic usage conditions in which a verifier operates a desktop browser connected to a standard home Wi-Fi router, reflecting typical scenarios for image authentication. As shown in Fig. 4, the measurements of communication delay in Pakistan and Italy also revealed a similar type of geographic differences under Wi-Fi, consistent with our 4G experiments.

The total delays were significantly lower than in the 4G scenario across all experiments. However, for a large number of image blocks, the signatures still exhibited linear delay scaling, as retrieving from IPFS remained the primary factor. In contrast, for a small number of block signatures, a relatively flat profile is maintained, with total verification times below 4 seconds in Italy and under 7 seconds in Pakistan, even for high-resolution images. These results highlight that distribution and proximity in the IPFS network noticeably affect communication latency, even in high-speed environments.

As a result, we demonstrated how verification performance worsens as the number of blocks increases by employing the baseline method, which is a specifically

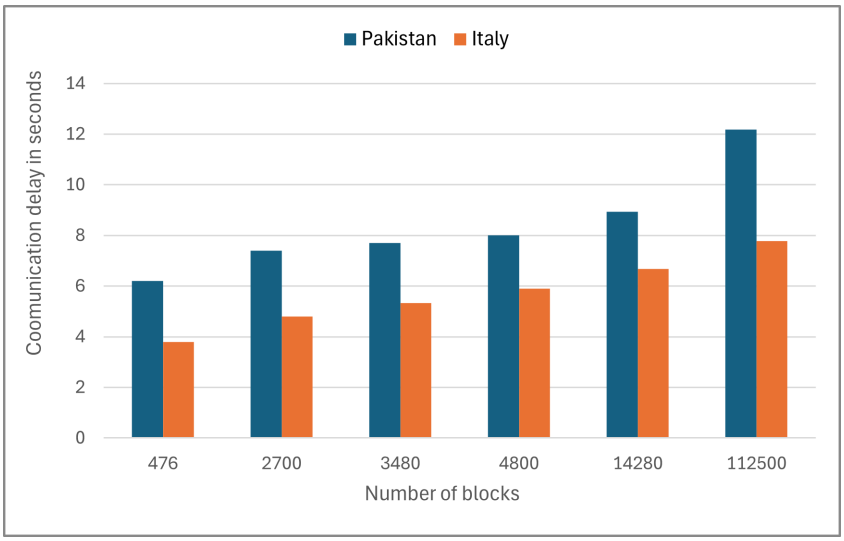

Fig. 4. Communication delay vs number of blocks over Wi-Fi connection

important factor in mobile environments where this latency and bandwidth can be critical and limited.

We stress that the verifier communicates (in both methods) with IPFS nodes only when the certificate becomes invalid, such as after expiration or revocation, not during the certificate's validity period. Therefore, all the previous performance measurements are relative to a post-expiration situation. On the other hand, when the certificate is still valid, the performance overhead of the verification is negligible, because the verifier directly uses only the cropped signature embedded in the cropped image without any IPFS or blockchain interaction.

7 Conclusion

In this paper, we presented a scalable and resilient image authentication system designed to verify a cropped image even after the expiration or revocation of the signer's public key certificate. It uses the BLS short signatures, IPFS as a decentralized peer-to-peer storage system, and a blockchain like Ethereum to anchor timestamps. The performance and practicality of our system under resource-constrained environments were confirmed through evaluations conducted over Wi-Fi and simulated 4G cellular links. After implementing and comparing both the baseline and optimized methods, we demonstrated that the optimized method significantly reduces the verification overhead and communication delay on the verifier side by achieving $\mathcal{O}(1)$ validation time. Future work will focus on exploring the integration of blockchain-based timestamping with SNARKs to support post-expiration validity for multiple manipulation types.

Acknowledgments

This work was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU – FF4ALL: Detection of Deep Fake Media and Life-Long Media Authentication, CUP D43C22003050001, and AQuSDIT: Advanced and Quantum-safe Solutions for Digital Identity and digital Tracing, CUP H73C22000880001. This work was also partially supported by Regione Piemonte in the framework of the EU programme ERDF (Italian FESR) 2021/2027 programme, project "IDV4NIS2" (CUP J89J24000520006).

References

1. I. Amerini, M. Barni, S. Battiato, P. Bestagini, G. Boato, T. S. Bonaventura, V. Bruni, F. De Natale, R. De Nicola, L. Guarnera *et al.*, "Deepfake media forensics: State of the art and challenges ahead," in *Advances in Social Networks Analysis and Mining*. Cham: Springer Nature Switzerland, 2025, pp. 33–48.
2. I. Amerini, M. Barni, S. Battiato, P. Bestagini, G. Boato, V. Bruni, R. Caldelli, F. De Natale, R. De Nicola, L. Guarnera *et al.*, "Deepfake media forensics: Status and future challenges," *Journal of Imaging*, vol. 11, no. 3, p. 73, 2025.
3. J. Mo, X. Kang, Z. Hu, H. Zhou, T. Li, and X. Gu, "Towards trustworthy digital media in the aigc era: An introduction to the upcoming isojpegtrust standard," *IEEE Communications Standards Magazine*, vol. 7, no. 4, pp. 2–5, 2023.
4. A. Vilesov, Y. Tian, N. Sehatbakhsh, and A. Kadambi, "Solutions to deepfakes: Can camera hardware, cryptography, and deep learning verify real images?" *arXiv preprint arXiv:2407.04169*, 2024.
5. Coalition for Content Provenance and Authenticity (C2PA), "C2pa technical specification," https://spec.c2pa.org/specifications/specifications/2.2/index.html, 2023, accessed: 2026-01-25.
6. ISO/IEC JPEG Committee, "Jpeg trust — core foundation," https://jpeg.org/jpegtrust/, 2025, accessed: 2026-01-25.
7. S. Erfurth, "Digital signatures for authenticating compressed jpeg images," in *Proceedings of the 1st Workshop on Security-Centric Strategies for Combating Information Disorder*, 2024, pp. 1–12.
8. P. Perazzo, M. Mattei, G. Anastasi, M. Avvenuti, G. Dini, G. Lettieri, and C. Vallati, "Jpegs just got snipped: Croppable signatures against deepfake images," in *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–6.
9. D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Rfc 5280: Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," https://datatracker.ietf.org/doc/html/rfc5280, 2008, accessed: 2026-01-25.
10. J. Buchmann and J. Buchamann, *Introduction to cryptography*. Springer, 2004, vol. 335.
11. G. Estevam, L. M. Palma, L. R. Silva, J. E. Martina, and M. Vigil, "Accurate and decentralized timestamping using smart contracts on the ethereum blockchain," *Information Processing & Management*, vol. 58, no. 3, p. 102471, 2021.
12. IPFS Docs, "What Is IPFS," https://docs.ipfs.tech/concepts/what-is-ipfs/, accessed: 2026-01-25.
13. J. Benet, "Ipfs - content addressed, versioned, p2p file system," https://arxiv.org/abs/1407.3561, 2014.

14. C. Sorge and M. Leicht, "Blockchain-based electronic time stamps and the eidas regulation: The best of both worlds," *SCRIPTed*, vol. 19, p. 61, 2022.
15. T. Datta, B. Chen, and D. Boneh, "Veritas: Verifying image transformations at scale," in *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2025, pp. 4606–4623.
16. D. Kang, T. Hashimoto, I. Stoica, and Y. Sun, "Zk-img: Attested images via zero-knowledge proofs to fight disinformation," *arXiv preprint arXiv:2211.04775*, 2022.
17. R. Johnson, L. Walsh, and M. Lamb, "Homomorphic signatures for digital photographs," in *International Conference on Financial Cryptography and Data Security*. Springer, 2011, pp. 141–157.
18. L. Du, A. T. Ho, and R. Cong, "Perceptual hashing for image authentication: A survey," *Signal Processing: Image Communication*, vol. 81, p. 115713, 2020.
19. C.-Y. Lin and S.-F. Chang, "A robust image authentication method distinguishing jpeg compression from malicious manipulation," *IEEE transactions on circuits and systems for video technology*, vol. 11, no. 2, pp. 153–168, 2001.
20. C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, "Rfc3161: Internet x.509 public key infrastructure time-stamp protocol (tsp)," 2001. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3161
21. T. Hepp, A. Schoenhals, C. Gondek, and B. Gipp, "Originstamp: A blockchain-backed system for decentralized trusted timestamping," *it-Information Technology*, vol. 60, no. 5-6, pp. 273–281, 2018.
22. B. Gipp, C. Breitinger, N. Meuschke, and J. Beel, "Cryptsubmit: introducing securely timestamped manuscript submission and peer review feedback using the blockchain," in *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, 2017, pp. 1–4.
23. B. Gipp, J. Kosti, and C. Breitinger, "Securing video integrity using decentralized trusted timestamping on the bitcoin blockchain," 2016.
24. C. Breitinger and B. Gipp, "Virtual patent-enabling the traceability of ideas shared online using decentralized trusted timestamping." in *ISI*, 2017, pp. 89–95.
25. Y. Zhang, C. Xu, N. Cheng, H. Li, H. Yang, and X. Shen, "Chronos^{+}: An accurate blockchain-based time-stamping scheme for cloud storage," *IEEE Transactions on Services Computing*, vol. 13, no. 2, pp. 216–229, 2019.
26. L. Meng and L. Chen, "A blockchain-based long-term time-stamping scheme," in *European Symposium on Research in Computer Security*. Springer, 2022, pp. 3–24.
27. A. Naveh and E. Tromer, "Photoproof: Cryptographic image authentication for any set of permissible transformations," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 255–271.
28. H. Noon. (2020) Why blockchain timestamps are inaccurate sometimes. https://hackernoon.com/why-blockchain-timestamps-are-inaccurate-sometimes. Accessed: 2026-02-03.
29. N. El Mrabet and M. Joye, *Guide to pairing-based cryptography*. CRC Press, 2017.
30. B. Lynn, "The pairing-based cryptography library," https://crypto.stanford.edu/pbc/, 2006, accessed: 2026-02-03.
31. G. Wood *et al.*, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum foundation, Tech. Rep., 2023.
32. J. Araújo, "Real-world latency and packet loss," https://blog.codavel.com/performance-report-defining-use-cases, December 2019, accessed: 2026-01-25.