Digital Signatures for Authenticating Compressed JPEG Images

Simon Erfurth
University of Southern Denmark
Odense, Denmark
simon@serfurth.dk

ABSTRACT

We construct a digital signature scheme for images that allows the image to be compressed without invalidating the signature. More specifically, given a JPEG image signed with our signature scheme, a third party can compress the image using JPEG compression, and, as long as the quantization tables only include powers of two, derive a valid signature for the compressed image, without access to the secret signing key, and without interaction with the signer. Our scheme is constructed using a standard digital signature scheme and a hash function as building blocks. This form of signatures that allow image compression could be useful in mitigating some of the threats posed by generative AI and fake news, without interfering with all uses of generative AI.

Taking inspiration from related signature schemes, we define a notion of unforgeability and prove our construction to be secure. Additionally, we show that our signatures have size 32.5 kb under standard parameter choices. Using image quality assessment metrics, we show that JPEG compression with parameters as specified by our scheme, does not result in perceivably reduced visual fidelity, compared to standard JPEG compression.

CCS CONCEPTS

• **Security and privacy** → **Digital signatures**; *Social network security and privacy*; • **Computing methodologies** → **Image compression**.

KEYWORDS

digital signatures, JPEG Compression, homomorphic signatures

1 INTRODUCTION

Digital signature schemes are a classical and widely used tool in modern cryptography (the canonical reference is [11], and [9] contains some current standards). For standard digital signature schemes, messages are required to be bit-wise identical to when they were signed, in order to be authenticated. For many applications—for example text—this makes perfect sense, but for others—for example images—it can become a limitation. When an image is shared online, it is very often compressed, typically to save both storage space and communication bandwidth. In general, compression is required to not fundamentally change the content of the image. After compression, it is (typically) not possible to restore the image to be bit-wise identical to the version that was uploaded, so compression invalidates any standard digital signature that was associated with the image.[1] In order to allow a signature to still be valid after compression, we construct a special digital signature scheme which allows signatures for a JPEG image to be updated during compression of the image, such that the updated signature is valid for the compressed image. Crucially, updating a signature does not require knowledge of the secret key used to generate the signature, yet the updated signature is still signed with the same secret key as the original signature. The only requirement for our signature scheme is that JPEG compression has to be performed using quantization tables containing only powers of two.

Creating multiple signatures for the image is not sufficient, since the signer does not know how the image might be compressed by others. While the signer can control how the image is compressed on the initial website it is posted on, the image might be shared on social media and other websites, where the signer does not control how the image is going to be compressed.

Digital signature schemes for images allowing compression, in one way or another, have been considered before. However, these all suffer from one major drawback or another, such as resulting in compressed images of much lower visual quality [18], only working for the JPEG2000 standard which was never widely adopted [44], giving only very weak and unclear notions of unforgeability [23], or being computationally expensive to compute [10]. In contrast, our construction avoids all of these drawbacks, at the cost of being less flexible than some of these schemes.

Technical overview. Our signature scheme is constructed to work with JPEG compression, since this is a widely used compression standard, and is used to compress images uploaded to social media. JPEG compression works by using that human vision is more sensitive to some features than it is to others. Concretely, JPEG compression makes use of two specific features: *a*) Humans are more sensitive to changes in luminance than to changes in color, so JPEG compression preserves more information about luminance than color, and *b*) the human vision system is less likely to notice high frequency changes in intensity than low frequency changes in intensity.[2] Thus, if an image contains both an area with high frequency changes in luminance and color shades (like grass or a treetop), and an area with only low frequency changes (like a blue sky), humans will be much more sensitive to small changes in the second area than in the first. Hence, JPEG compression will generally preserve more information about the latter area. This is done by splitting images into into 8×8 pixel blocks, which are then transformed with the discrete cosine transformation (DCT) into a representation using a basis of discrete cosine waves. Now, as the lossy step of JPEG compression, all coefficients are divided by a value from a *quantization table* and rounded. Generally, coefficients

[1] The exemption to this is loss-less compression schemes which allow images to be restored to be bit-wise identical to their uncompressed versions. For these compression schemes standard digital signature schemes will work. However, for many use cases loss-less compression schemes cannot achieve high enough compression ratios, and thus lossy compression schemes are used.

[2] Note that here *frequency* does not refer to the wavelength of light, but to the frequency of the intensity of a color/luminance an area of a picture. For example, an area with no change in color/intensity will have very low frequency and an area with where the pixels alternate between being black and white will have very high frequency.

corresponding to high frequency changes and/or luminosity are divided by larger values than coefficients corresponding to low frequency changes and/or color. This results in less information being lost due to rounding for features humans are sensitive to, compared to features humans are less sensitive to.

This lossy step is exactly what prevents standard digital signatures from being used for images that will be compressed using JPEG compression. For our signature scheme, this step is also crucial. A key observation is that when the value in the quantization table is a power of two, division and rounding down acts as truncation of the DCT coefficient. The idea is therefore that when an image is compressed with a quantization table containing only powers of two, our signature scheme should still be able to authenticate the bits that have not been truncated. Thus, we create a signature on what is essentially the root of a tree of hashes, where each hash takes as input a combination of hashes from a deeper level of the tree and bits from the DCT coefficients of the image. Now, if an image is compressed with a quantization table containing only powers of two, a signature for the image can be updated to one for the compressed image, by including a subset of the hashes that were made using now truncated bits from the image in the signature. Using that coefficients corresponding to the same DCT basis element in different blocks are always cropped by the same amount, we can create an efficient signature that requires $O(1)$ more work than a regular digital signature, and also result in a signature of constant size (depending only on the choices of parameters, and not on the size of the image). A bonus from the construction of the scheme is that it is fully backwards compatible, in the sense that any JPEG image viewer can display JPEG images compressed with our scheme (even if they might not be able to authenticate the signature for the image). Additionally, our scheme allows repeat compression and updating of signatures, as long as all used quantization tables only contain powers of two, and the signature is kept up to date in each compression.

Application. A use case of our scheme could be for mitigating to consequences of fake news, and in particular to mitigate the threat posed by generative AI for images. The consequences of fake news and misinformation are many, and they pose an increasingly greater threat to democracy and society. Apart from the immediate consequences in the form of uncertainty about whether the content one is looking at is true or not, fake news and misinformation also lead to an increasing level of distrust in the media [30]. One reason that the threat is increasing is a fundamental change in how news are consumed. Rather than being consumed directly from news media outlets (such as newspapers, TV, and first-party websites), news is increasingly consumed on social media platforms [29]. This is an issue, because people tend to be unable to recall which news brand a story originates from when they are exposed to it on social media [20]. Since the news media's image is used as an important heuristic when people evaluate the quality of news [37], this change allows misinformation to spread. Thus, using digital signatures seems like a natural suggestion; doing so could bind stories shared on social media to the news media that published them, allowing use of this important heuristic. For images specifically, the developments and rise of generative AI over the last few years proves additional cause for concern. With it, everyone can generate (mostly) realistic looking images of anything and everything they can imagine. Considering images' emotional pull and highly persuasive influence [3], they are an effective medium for spreading fake news, and hence the sheer volume of (potentially misinforming) images that generative AI can create is highly problematic.

A prevalent method for addressing misinformation on social media involves flagging potential misinformation, employing either manual or automated detection systems. However, studies have consistently demonstrated that flagging problematic content may backfire, exacerbating its adverse impacts [22, 32]. Attempting to detect and flag all AI generated images is also not an ideal solution. Not only would it hinder legitimate uses of AI generated images, but it could also lead to the generative AI models being trained to not trigger the detection system, as a variation of the widely used generative adversarial network method for training generative AI [15]. This would result in essentially an arms race between models for detecting and models generating images [17].

Digital signatures can be used to complement the prevalent approach of checking and flagging misinformation. Specifically, if news media sign the images they post, images can be accompanied by signatures signed by news media when they are shared on social media, whether by the news media or by other users. This would provide a guarantee of the provenance of the image (something that is currently missing), helping people evaluate the quality of any news story associated with the image. If only news media meeting a minimum standard of journalistic quality are allowed to sign their pictures, it would also help tell apart quality journalistic content from potential misinformation [7].Specifically, the absence of a digital signature for a picture would be the first red flag, indicating to users that they should be sceptical. For images produced by generative AI, this approach has some clear advantages over a detection-and-flagging approach. With this approach, it is not possible to "just" train the generative AI models to not be detected. Additionally, this approach still allows news media to use AI generated content for their stories. Images shared on social media are compressed when uploaded, so for this approach to work, the signature needs to allow compression of the image. Therefore, our signature scheme is perfectly suited for this use case.

The idea of mitigating the effects of fake news and misinformation, by using digital signatures to verify the source of images, has been considered by others. One example is the Coalition for Content Provenance and Authenticity (C2PA) [8], which involves many companies, including Adobe, the BBC, Google, Microsoft, and Twitter. C2PA focuses on providing a history of an image, i.e., which device was used to capture it, how it has been edited and by whom, etc. However, their approach relies on trusting software used to edit an image to act honestly. Adding support for compression, without needing to compute a new signature, could perhaps increase the versatility of their approach.

Structure. We provide an overview of related work in Section 2, and Section 3 covers JPEG compression in greater detail. In Section 4, we give a generic definition of digital signature schemes for images allowing compression, define an unforgeability notion for such schemes (Section 4.2), construct our signature scheme (Section 4.3), and show that our scheme is secure with respect to the notion of unforgeability (Section 4.4). Finally, we analyze the

complexity of our scheme (Section 4.5). A visual evaluation of the JPEG compression used by our scheme is done in Section 5, where we show that our scheme is almost as good as JPEG compression without any restrictions. To finish up, in Section 6, we consider potential optimizations to our scheme, and where further research could go.

Our contribution. We give a generic definition for how a digital signature scheme for images allowing JPEG compression should work, and a natural definition for what it means for such a scheme to be unforgeable. We then describe a specific construction, and prove that (when instantiated using cryptographic secure primitives) it satisfies the notion of unforgeability. Compared to other schemes that allows signed images to be compressed, our scheme trades supporting multiple types of modifications and/or arbitrary JPEG compression for either being more efficient, having a meaningful notion of security, or having higher visual fidelity.

2 RELATED WORK

The observation that JPEG compression can act as truncation of the DCT coefficients, and that this could be used to construct a homomorphic digital signature scheme for images, was first made in [18]. The focus of their work is on developing a signature scheme that allows cropping, but as an auxiliary result the authors observe that if every DCT coefficient is truncated by the same amount (in their work *cropped*), their signature scheme also allows some JPEG compression. However, truncating all DCT coefficients by the same amount results in their scheme not making use of the key idea behind JPEG compression; namely that the human eye is less likely to notice high frequency changes in intensity than low frequency changes. This results in the visual fidelity of images compressed according to their scheme being lower than the visual fidelity of images compressed under standard JPEG compression parameters at similar sizes. Additionally, for JPEG compression to act as truncation, the quantization table need to consists only of powers of two, and hence their approach only allows 8 different quantization tables, leading to their approach being very inflexible, on top of reducing the visual fidelity. We demonstrate these problems in Section 5.

Homomorphic Digital Signatures. Homomorphic digital signatures have been studied in a number of different contexts, both for different types of data (images, text, different data structures, etc.) and for different operations that the signature is homomorphic with respect to (cropping, redaction, various set operations, etc.). In the article discussed above [18], their constructed signature can (depending on specific choices) be homomorphic with respect to image cropping, scaling, or (very restricted) JPEG compression. Other homomorphic signature schemes for images have been considered, in particular for the JPEG2000 image standard. For example, [44] gives an overview of signatures for JPEG2000 images that are homomorphic with respect to extraction of various representations from single code streams. However, the JPEG2000 image standard was never widely adopted, and the only web browser supporting JPEG2000 is Safari [1].

Much more generally, [2] provides a generic definition and construction of homomorphic signatures, which allows deriving a signature on m' from a signature on m, as long as $P(m, m') = 1$ for a (univariate and closed) predicate P. Their definition encompasses many examples of individual well-studied homomorphic signatures, including arithmetic, quotable, redactable, and transitive signatures [21, 43, 7, 34, 19, 26, 6], but the generic scheme is generally less efficient than more specialized schemes.

Perceptual Hashing. Another approach that in principle gives rise to digital signatures allowing image compression, is the use of perceptual hashing [25] in place of the cryptographic hash functions typically used by digital signature schemes. Conceptually, a perceptual hash function is a hash function where the hash of an image only changes if the images is *perceivably* different enough. Thus, JPEG compression (to some extent) should not change the perceptual hash of an image. Combined with a standard digital signature scheme, a perceptual hash function should therefore create a digital signature scheme allowing JPEG compression. The issue with this approach is that all perceptual hashes known to the authors have an overlap between having false positives and having false negatives. That is, all schemes will either accept (randomly) manipulated images as being authentic with a non-negligible probability, or they will reject authentic images as being manipulated with a non-negligible probability [12]. Thus, perceptual hashing cannot reasonably be used in place of a cryptographic hash in a digital signature scheme.

Considering JPEG compression specifically, Lin and Chang [23] created a perceptual hash function with this in mind. They find relationships between the 8×8 pixel blocks in an image that are somewhat invariant under JPEG compression. However, even this approach allows a manipulated image to be accepted with non-negligible probability. Specifically, the authors perform a practical experiment where they change a random block, and, using various parameters, find that the probability of a manipulated image being accepted is between 0.04 and 0.00001 when the image is not compressed, between 0.09 and 0.0002 with a quality factor of 50, and between 0.2 and 0.02 with a quality factor of 20. That is despite these attacks assuming that the manipulator had no knowledge of which blocks were being compared, and made a non-targeted attack by editing one chosen at random. An attacker with knowledge of which blocks are being compared could potentially make a targeted attack with even higher success chance. In contrast, it follows from the correctness of our scheme, that we have no false negatives, and from unforgeability that there is only a negligible probability of false positives.

Cryptographic approaches to mitigating Misinformation through Images. Using digital signatures to prevent misinformation through images has also been considered in [4]. In this article, the authors suggest using standard digital signatures directly on images, and focus more on the technical considerations for how this could be implemented. One obstacle to using their approach is that in practice, images are almost always compressed when uploaded online (for example to social media) and all standard digital signature schemes require the image to be bit-for-bit identical to the signed image. In order to fix this shortcoming, their work could be changed to instead use our digital signature scheme, in which case it considers the technical details of implementing digital signatures allowing compression.

In [33], the authors discuss on a more general level how cryptographically proving provenance can be a proactive partial solution to mitigating misinformation. Based on literature from human-centered computing and usable security, journalism, and cryptography, they consider both advantages and challenges of such a system, and find properties a system should have.

A different suggestion for using cryptography to prevent misinformation through images is made in [10]. They suggest using succinct non-interactive zero-knowledge proofs to verify the metadata of an image, and that the image has only been modified in some claimed ways. More concretely, their suggested solution is to use a camera that adds metadata to an image when it is taken, and signs the image and metadata (such a camera was recently released by Leica in collaboration with the C2PA [24]). When an image is later edited, the editor also generates a zero knowledge proof of the following statement: *"The prover (i) knows an unedited photo that is properly signed by a C2PA camera, (ii) the metadata on the unedited signed photo is the same as the one attached to the public photo, and (iii) the public photo in the news article is the result of applying the claimed edits to the unedited photo."* Before an image is displayed, the zero-knowledge proof is verified, and the image is then displayed together with its accompanying metadata. While their construction allows arbitrary compression, and many other operations, their solution does not appear to be efficient enough to use for images shared on social media. Generating a proof takes minutes, even on a modern system, and considering how many images are uploaded to social media, it would not be feasible to generate proofs for all of them. Prior to [10], it was suggested in [28] to use zero-knowledge proofs to authenticate that only permissible transformations has been made to an image. However, the proving time of their implementation is even longer (around 5 minutes to generate proofs for 128×128 pixel images). In contrast, our construction requires at most 1025 hash function evaluations and one key generation, signing, or verification of a standard signature scheme.

3 JPEG COMPRESSION

In order to construct our signature scheme, we need a general understanding of how JPEG compression works. For each step, more information can be found in [38], and in the official JPEG standard [16].

As we mentioned in the introduction, the key observation behind JPEG compression is that humans are much better at noticing some types of details than others. In particular, humans are less likely to notice high frequency changes in intensity of both color shades and luminance, than low frequency changes. Similarly, humans are less likely to notice changes in color compared to changes in luminance. JPEG compression uses this to perform lossy compression without sacrificing too much perceived image quality, by preserving more information about the coefficients for low frequency changes and about luminance, and less information about high frequency changes and about color.

The first step of JPEG compression is to convert the image to the YCbCr color-space,[3] which, later in the process, allows preserving more information about luminance than about color. As an optional second step, the color channels can be down-sampled either just along one axis, or along both axes, meaning that the resolution of one or two dimensions is halved: a mean value of two (or four) pixels is found, and used for two (or four) pixels. Steps three and four are applied to each 8×8 block of the image separately (with appropriate padding when the image dimensions are not multiples of 8). As the third step, the discrete cosine transformation is applied to the 64 values to transform them into 64 DCT coefficients. Roughly, this can be thought of as changing each pixel from representing the intensity of that specific pixel into representing the intensity of a particular (discrete) cosine function over the entire 8×8 pixels block; see Figure 1c. This step allows preserving more information for the low frequency cosine waves that represents low frequency changes in the image. The fourth step is referred to as quantization and is the only lossy part of JPEG compression. In this step, each entry of the 8×8 pixel block is divided by an entry from the *quantization table* and rounded. The quantization table is an 8×8 table, consisting of values in the range 1 to 256. Generally, entries representing low frequency cosine waves in the block are divided by smaller values from the quantization table, and entries representing high frequency changes are divided by larger values. Hence, less information is lost for low frequency cosine waves due to rounding, i.e., when the process is reversed by multiplying with the entries from the quantization table, the coefficients for low frequency cosine waves generally end up closer to their original value than coefficients for high frequency cosine waves. By choosing different values for the quantization table, it is possible to control the trade-off between how much the file size is reduced and how much the quality of the image is reduced. Finally, the image is encoded using a lossless entropy encoding, usually Huffman encoding. A number of tricks are applied as preprocessing in this step, but for brevity (and since they are not directly relevant to this work), we will not discuss them, but refer instead to [16, Section 4.3], and the related standards. To display an image, each of these steps are reversed in the opposite order.

To summarize, JPEG compression consists of the following steps, which we have also illustrated in Figures 1a to 1d.

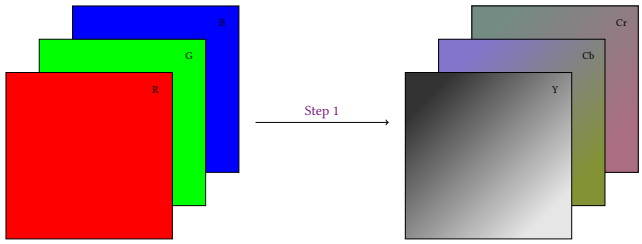
1. Convert the image from RGB to the YCbCr color-space.
2. Optionally down-sample the color channels (Cb and Cr).
3. For each 8×8 pixels block in each channel:
   a. Apply the discrete cosine transformation to the block.
   b. Quantizise the block, using the quantization table.
4. Encode the image using a lossless entropy encoder.

3.1 DCT Transformation

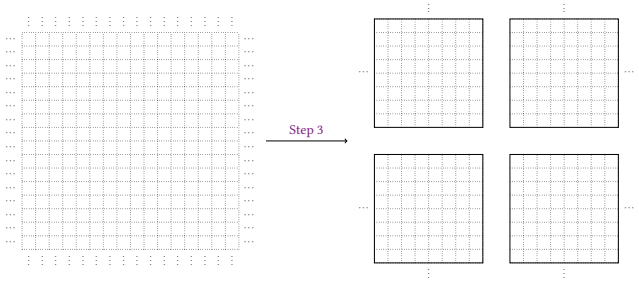
Step 3a, illustrated in Figure 1c, is a change of basis from using the standard basis for 8×8 matrices to the DCT basis. That is, instead of using the basis $\{e_{i,j}\}_{i,j\in\{0,\dots,7\}}$, where

$$(e_{i,j})_{p,q} = \begin{cases} 1 & \text{if } i = p \text{ and } j = q \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

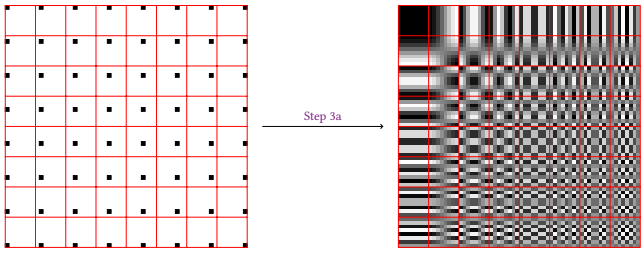
[3] Meaning that instead of representing the image using red, green, and blue channels (RGB), it is represented as one luminance channel (Y) and two color channels (Cb and Cr). Mathematically, this is a lossless transformation, but in practice there will of course be some losses due to rounding errors. For simplicity, we assume that before this step, all images are 8 bit RGB images. However, all constructions are easily changed to work on 10-bit images.



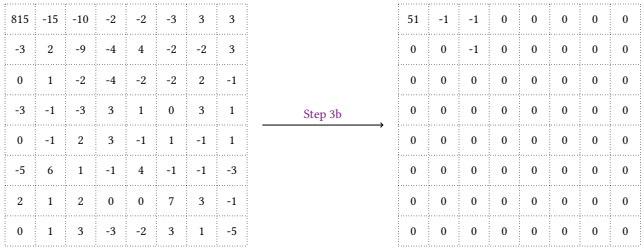
(a) Transformation from RGB color space to YCbCr color space.



(b) Splitting into 8×8 pixel blocks.



(c) Changing to DCT representation, e.g., instead of each pixel representing its own value, it now represents one of the DCT basis functions. As illustrated in the figure, this can also be thought of as a change of basis from the standard basis to the DCT basis.



(d) Quantizise each block using the quantization table. Observe how information is preserved for the values corresponding to low-frequency DCT basis elements in the upper left corner. Values are the luminance values of a random block from the Lenna test image, quantized using the quantization table generated with $Q = 50$, as discussed in Section 3.2.

Figure 1: Select steps of JPEG compression.

we now use the basis $\{B_{i,j}\}_{i,j\in\{0,\dots,7\}}$, where

$$(B_{i,j})_{p,q} = \cos\left(\frac{\pi(2p+1)i}{2\cdot 8}\right)\cdot\cos\left(\frac{\pi(2q+1)j}{2\cdot 8}\right) \tag{2}$$

for $p, q \in \{0,\dots,7\}$. The resulting discrete cosine functions are illustrated on the right side of Figure 1c. We stress that what is stored in the 8×8 matrices are the *coefficients*, $a_{i,j}$ and $a'_{i,j}$, of the basis elements, i.e., $a_{i,j}$'s and $a'_{i,j}$'s such that the 8×8 block can be expressed as

$$\sum_{i,j} a_{i,j}e_{i,j} = \sum_{i,j} a'_{i,j}B_{i,j}. \tag{3}$$

Since both the standard basis and the DCT basis are spanning, this is a lossless operation (up to rounding). As mentioned, the purpose of this step is to allow the next step to preserve more information for low frequency changes, which will generally be represented by the basis elements where i and j are small; see Equation (2) and the right side of Figure 1c.

3.2 Quantization

In Step 3b of the JPEG compression, each value in the 8×8 block of DCT coefficients is quantized by dividing it by a value from a quantization table and rounding the result. Despite the standard [16] specifying that the result should be rounded towards the nearest integer, this appears to not be the case in practise. As an example, the widely used implementation from The Independent JPEG Group (IJG) [35] rounds towards 0. When displaying a compressed image, the value is multiplied with the value from the quantization table again. Hence, information can be lost in the step. For example, different values may end up being mapped to the same, see the first row of Figure 1d.

As mentioned earlier, one of the tricks that JPEG compression uses is that the human eye is much more sensitive to low frequency changes. Hence, it will generally be the case that the values in the quantization table grow as one goes from the upper left corner to the lower right. Additionally, since the human eye is more sensitive to luminance than color, different tables are used for the luminance channel, Y, and for the color channels, Cb and Cr.

However, there are principally no requirements for quantization tables (apart from them being 8×8 tables with values between 1 and 256), and indeed, this is one of the main points where various programs implementing JPEG compression can differ.[4] However, one standard approach is to let the *quality factor* p be an integer with $1 \le p \le 100$, fixing the table for quality factor 50 to be Q_{50}, and then deriving the other tables from Q_{50}. For example, in the IJG implementation [35],

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} \tag{4}$$

[4] Section 3 of [36] gives an overview of quantization tables for different purposes.

and for $1 \leq p \leq 100$ we have

$$Q_p = \begin{cases} \frac{50}{p} Q_{50} & \text{if } p < 50 \\ \frac{200 - 2p}{100} Q_{50} & \text{otherwise.} \end{cases} \tag{5}$$

Finally, the entries in Q_p are rounded, and it is ensured that every entry is at least 1 and at most 255. This definition means that quantization with Q_{100} does not destroy information (Q_{100} is all 1's), and as the quality factor decreases, the values in Q_p increase, so that more detail is lost, but the compression also results in a smaller file.

4 SIGNATURE CONSTRUCTION

Sections 4.1 and 4.2 contains a generic definition of what a digital signature scheme for images allowing compression is, and what it means for it to be unforgeable. From Section 4.3 and onward, we construct such a signature scheme, show that it is unforgeable, and analyse its performance.

4.1 Generic Definition

In general, a digital signature scheme for images allowing compression consists of four efficient algorithms, KeyGen, Sign, Compress, and Verify. These are essentially the three standard digital signature algorithms for key generation, signing, and verification, with an added compression algorithm. The compression algorithm allows a signature for an image to be updated to a signature for a compressed version of the image, given both the original image, the signature for the original image, and the compressed image. For simplicity of further definitions, we make the abstraction that the compression algorithm also performs the compression. We use λ to denote the security parameter. Summarizing, we require that the four algorithms acts as follows.

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$ takes as input the security parameter 1^λ, and outputs a key pair.
- $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(i)$ takes as input a secret key sk and an image i. Outputs a signature for i that allows compression.
- $(i', \sigma') \leftarrow \mathsf{Compress}(i, \sigma, P)$ takes as input an image i, a signature σ for i, and additional parameters P specifying how compression should be done. In the case where the compression is JPEG compression, P is the quantization tables. Outputs the compressed image i' and a signature σ' for i', that is still signed with the private key that σ was signed with. Note that i and σ could have been obtained from an earlier compression.
- $\top/\bot \leftarrow \mathsf{Verify}_{\mathsf{pk}}(i, \sigma)$ takes as input a public key pk, an image i, and a signature σ for i. Outputs $\top$ if σ' is a valid signature for i with respect to pk, and $\bot$ otherwise.

A standard notion of correctness should be satisfied, meaning that a genuine signature should always be accepted.

4.2 Security Notion

Our notion of security takes inspiration from other digital signature scheme variants allowing some form of modification, e.g., redactable signatures [34, 19], quotable signatures [7], the image signatures from [18], and generic P-homomorphic signatures [2]. Essentially,

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$
$(i^*, s^*) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{pk})$
/ denote the queries that $\mathcal{A}$ make to the signing oracle by $i_1, i_2, \ldots, i_Q$,
/ and the answers by $\sigma_1, \sigma_2, \ldots, \sigma_Q$.
if $(\mathsf{Verify}_{\mathsf{pk}}(i^*, s^*) = \top) \wedge (\forall k \in \{1, 2, \ldots, Q\} : i^* \notin \mathsf{CSpan}(i_k, \sigma_k))$
  return 1

Figure 2: The Unforgeability experiment. We write $\mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}$ to indicate that $\mathcal{A}$ is given access to an orcale that simply signs images under sk **using** Sign**.**

these notions of security say that the scheme is unforgeable if no adversary can produce a signature for a message that is not either a message the signing oracle has provided a signature for, or the result of performing an "allowed" operation on a message the signing oracle has provided a signature for. In our case, this means that the image the adversary outputs cannot be obtained by performing allowed compression on any of the images queried to the signing oracle. To make the definition general, we define an "allowed compression" to be any compression that can be done by Compress under valid input. For our scheme specifically, the definition says that no adversary can output a signature for an image that is not either an image the adversary has queried the signing oracle for, or the result of compressing one of these images using quantization tables that consists of only powers of two. We formally define this in Definition 4.1.

Definition 4.1 (Unforgeability). For a signature scheme

$$\mathsf{CS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Compress}, \mathsf{Verify}) \tag{6}$$

allowing image compression, we define the compression span of Compress on an image I with valid signature σ to be

$$\mathsf{CSpan}(I, \sigma) := \left\{ I' \;\middle|\; (I', \sigma') \leftarrow \mathsf{Compress}(I, \sigma, P) \right\}, \tag{7}$$

where P is valid extra parameters to Compress. That is, $\mathsf{CSpan}(I)$ is the set of all images that I can be compressed to by Compress. The signature scheme CS is said to be *existentially unforgeable*, if for every probabilistic polynomial time adversary $\mathcal{A}$, the probability of the experiment in Figure 2 returning 1 is negligible.

4.3 Our construction

Conceptually, our idea builds on the observation that if all the entries in the quantization table are powers of two, then we can consider the quantization step as being *truncation* of the least important information. This allows us to construct a signature where one can provide some (small) piece of information that allows a signature for an image to be verified, even if the image has been compressed.

To illustrate the idea, we consider an example where we have just one 8 bit value, $b = b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0$, which we wish to sign in a way that allows us to truncate a number of (least significant) bits. This can be done by constructing a *chain of hashes* as follows. The first node in the chain is the hash of the least significant bit, $H(b_0)$. Any other node is the hash of the concatenation of the previous node and the next least significant bit, i.e. the second node will be $H(H(b_0) \,\|\, b_1)$. Finally, one signs the last node in the chain (the *end*

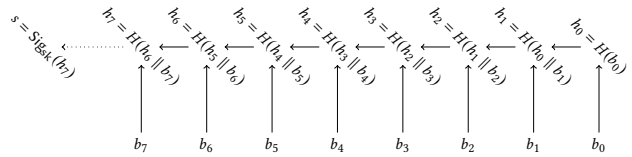


Figure 3: The chain of hashes and signature for one byte $b = b_7b_6b_5b_4b_3b_2b_1b_0$.

node) using a standard digital signature scheme. We illustrate this in Figure 3.

Now it is possible to truncate out some of the least significant bits of b, and, if one instead provides the node of the chain of hashes corresponding to the most significant of the truncated bits, the signature for b can still be authenticated, since h_7 can still be computed. This is done by calculating the chain of hashes, starting from the node of the least significant bit that was not truncated, which can be calculated using the provided value. In Section 4.4 we show that this still binds the non-truncated bits, in the sense that these bits cannot be changed without invalidating the signature. We also argue that for our use case, this signature provides a meaningful notion of security. Of course, for this example, it would have been much more space efficient to send the truncated bits instead of a hash value. However, as we see in Section 4.5, this is not the case when we consider (larger) images, rather than just single bytes.

Going back to full images, we recall that JPEG compression works on 8×8 pixel blocks, where every block is handled the same way. In particular, the coefficient for a specific DCT basis element will be truncated by the same number of bits in every block. Thus, we only need one chain of hashes for each basis element, regardless of the size of the image. For any basis element $B_{i,j}$, the first node in the chain of hashes is the hash of the concatenation of the least significant bits of all bytes at location i, j in each of the 8×8 blocks, in an arbitrary, but fixed, order. For all other nodes in the chain of hashes, the node is the hash of the concatenation of the previous node and the next least significant bits of all bytes at location i, j. In Figure 4, we illustrate how the chain of hashes is calculated for the entries corresponding to the first DCT basis element, $B_{0,0}$.

For each of the two quantization tables (one for luminance and one for color), we calculate the chain of hashes for each of the $8 \cdot 8 = 64$ DCT basis elements, obtaining 128 end nodes, one for each basis element in each channel. The end nodes are then hashed together, in order to get one final hash, h_{root}, which is signed using a standard digital signature scheme. This final process is illustrated on Figure 5.

Now any party can compresses the image while still allowing the signature to be verified, by performing the compression using a quantization table with only powers of two in it, and providing the relevant nodes from the 128 chains of hashes. To be specific, if entry (i, j) in the quantization table is 2^k, the compressing party adds node h_{k-1} from the chain of hashes corresponding to $B_{i,j}$ to the signature (if $k = 0$ no compression is done, and no node is added). We denote the added nodes as *the truncated hashes*.

To summarize, our digital signature scheme allowing some JPEG compression works as follows, where H is a cryptographic hash

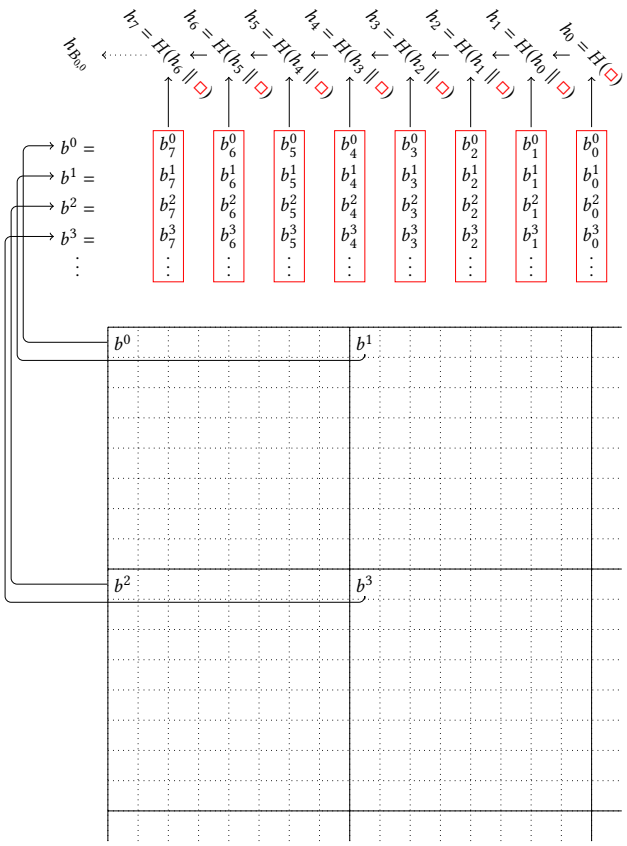


Figure 4: Example illustrating construction of one of the chains of hashes.

function and $\mathrm{DS} = (\mathsf{KeyGen}^{\mathrm{DS}}, \mathsf{Sign}^{\mathrm{DS}}, \mathsf{Verify}^{\mathrm{DS}})$ is a standard digital signature scheme.

- $\mathsf{KeyGen}(1^\lambda)$: Identical to $\mathsf{KeyGen}^{\mathrm{DS}}(1^\lambda)$.
- $\mathsf{Sign}_{\mathsf{sk}}(i)$: Compute the 128 chains of hashes, as described above, compute the hash of the end nodes to obtain h_{root}, and sign this using $\mathsf{Sign}^{\mathrm{DS}}$.
- $\mathsf{Compress}(i, \sigma, P)$: Given an image, a signature for the image, and quantization tables containing only powers of two, first perform standard JPEG compression, always rounding down. Then compute the chains of hashes from the original image, and extract the 128 truncated hashes. Together with the signature for the uncompressed image, the truncated hashes form the signature for the compressed image. Output the compressed image and the signature for the compressed image. Observe that a compressed image can be further compressed (using a quantization table with at least as large powers of two) and the signature updated with new truncated hashes to be valid for the further compressed image.
- $\mathsf{Verify}_{\mathsf{pk}}(i, \sigma)$: Use the image, and if it is compressed the truncated hashes, to find h_{root}, and verify with respect to the original signature using $\mathsf{Verify}^{\mathrm{DS}}$.

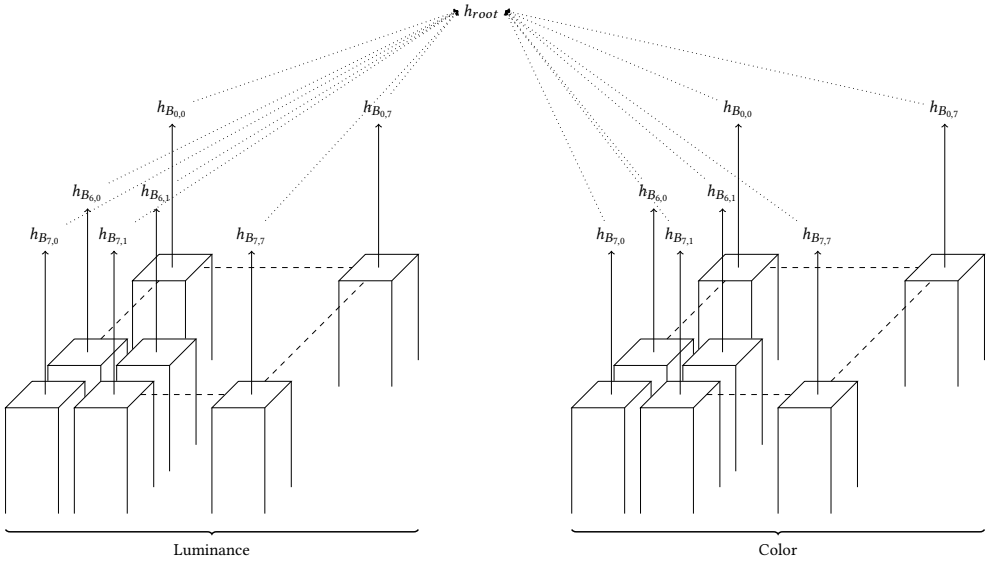


Figure 5: The 128 end nodes (64 from the to the luminance channel and 64 from the color channels) are used to calculate h_{root}. Then h_{root} is signed using the standard digital signature scheme, obtaining the digital signature for the uncompressed image.

Note that this matches the description of digital signature schemes allowing compression, outlined at the end of Section 4.1.

4.4 Security Analysis

Theorem 4.2 shows that if the primitives used in the scheme constructed in Section 4.3 are secure, the scheme is secure in the sense of Definition 4.1. We show this by arguing that an adversary for our scheme can be used to construct an adversary for at least one of the primitives.

Theorem 4.2. *Under the assumption that*

- *H comes from a family of cryptographic secure hash functions,*
- *$\mathsf{DS} = (\mathsf{KeyGen}^{\mathsf{DS}}, \mathsf{Sign}^{\mathsf{DS}}, \mathsf{Verify}^{\mathsf{DS}})$ is an existentially unforgeable standard signature scheme,*

$\mathsf{CS} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Compress}, \mathsf{Verify})$ constructed as described above, is an existentially unforgeable signature scheme allowing image compression.

Proof. Assume that $\mathcal{A}$ is a probabilistic polynomial time adversary against the unforgeability of CS, as defined in Definition 4.1. We show that the probability of $\mathcal{A}$ being successful is negligible. Let (i^*, s^*) be the output $\mathcal{A}$, with $s^* = (\mathsf{Sign}^{\mathsf{DS}}_{\mathsf{sk}}(h_{root}), \{h^{i,j,c}_k\})$, i.e., s^* consists of the standard signature for h_{root} of i^*, and the (possibly empty) set of relevant nodes from the chains of hashes, indexed by i, j, c, where i, j specifies a DCT basis element, and c specifies if it is from the luminance or the color quantization table.

Consider first the case where h_{root} of i^* (which can be found using i^* and $\{h^{i,j,c}_k\}$) is different from h_{root} of each of the Q images $i_1, \ldots, i_Q$ that were $\mathcal{A}$'s queries to the signing oracle. In this case, (h_{root}, s^*) is a forgery against DS, and since DS is assumed existentially unforgeable, this can happen with at most negligible probability ϵ_{DS}.

For the other case, let $\hat{i}$ denote the image queried to the signing oracle such that h_{root} of $\hat{i}$ is the same as h_{root} of i^*. In the following, we will use $\hat{h}$ to indicate that a hash is generated from $\hat{i}$, and just h for values generated from i^*. With this notation, we are considering the case where $\hat{h}_{root} = h_{root}$. We now compare each of the 128 hash values (end nodes) that were used to generate $\hat{h}_{root}$ and h_{root}. That is, for $i, j \in \{0, \ldots, 7\}$ and $c \in \{\mathrm{Y}, \mathrm{Cb/Cr}\}$, we compare $\hat{h}^{i,j,c}_{B_{i,j}}$ and $h^{i,j,c}_{B_{i,j}}$. If $\hat{h}^{i,j,c}_{B_{i,j}}$ and $h^{i,j,c}_{B_{i,j}}$ differ in at least one location i, j, c, we have found a collision for H.

If $\hat{h}^{i,j,c}_{B_{i,j}}$ and $h^{i,j,c}_{B_{i,j}}$ are the same at every location, we now go one step further down, and look at each chain of hashes. That is, for each location i, j, c, we consider now $\hat{h}_k$ and h_k for $k \in \{0, \ldots, 7\}$ (as illustrated on Figure 4). There must be at least one location i, j, c, k where the bits from the image used to calculate $\hat{h}_k$ are different from the bits from i^* used to generate h_k. To see this, observe that if not, one could obtain i^* from $\hat{i}$ by compressing $\hat{i}$ with suitable values in the quantization tables (the values being the ones that truncate the chains of hashes for $\hat{i}$ to the length of the chains of hashes for i^*, possibly 1 if no compression was performed). If $\hat{h}_k = h_k$, this is clearly a collision for H. On the other hand, if $\hat{h}_k \neq h_k$, consider instead $\hat{h}_{k+1}$ and h_{k+1}. If $\hat{h}_{k+1} = h_{k+1}$, we have instead found a collision here, since

$$H(\hat{h}_k \parallel \cdots) = \hat{h}_{k+1} = h_{k+1} = H(h_k \parallel \cdots), \tag{8}$$

where $\cdots$ indicates that the relevant bits from the images are inserted. Alternatively, we have $\hat{h}_{k+1} \neq h_{k+1}$, in which case and continue on to consider the next nodes in the chain. Since the end nodes of the chains are the same, i.e., $\hat{h}_{B_{i,j}} = h_{B_{i,j}}$, we are guaranteed that the nodes will eventually be the same, and hence we are guaranteed that we find a collision for H. In all cases where we did not find a signature forgery, we have instead found a collision for H. Since H is assumed to come from a family of cryptographic secure

hash functions, this happens with at most negligible probability ϵ_H.

It follows that the probability of $\mathcal{A}$ being successful is at most $\epsilon_{\mathsf{DS}} + \epsilon_H$, which is negligible. □

4.5 Performance Analysis

From the construction of our signature scheme, it is immediate that the signature size is bounded by a constant, depending only on the size of the output of the hash function used and the size of the underlying standard digital signature scheme, see Corollary 4.3. For a hash function with a 256 bit output (for example, the widely used SHA3-256), the scheme has a signature size of at most $128 \cdot 256 = 32,768$ bits or 4 kB on top of the standard digital signature. Section 6 includes suggestions for making the signature smaller.

Corollary 4.3. *If H is a hash function with output size $|H|$ and* DS *is a standard digital signature scheme with signature size $|S|$, the scheme* CS *constructed as described above, has signature size $|S|$ for uncompressed images and signature size*

$$128 \cdot |H| + |S| \tag{9}$$

for compressed images.

Proof. For an uncompressed image, the signature for the image consists of the standard digital signature for h_{root}. For a compressed image, the signature consists of the standard digital signature for h_{root} and (at most) one node from each of the 128 chains of hashes, see Figure 5. □

In terms of computation requirements, it follows from the construction of the signature scheme that key generation, signing, and verification require one key generation, signing, or verification from the standard digital signature scheme. Additionally, for signing, compression, and verification, it is necessary to compute h_{root}. Doing so requires computing each of the 128 chains of hashes, each of which requires computing (up to) 8 hashes, plus a final hash to obtain h_{root}. When the image is compressed, verification requires computing fewer than 8 hashes per chain. In total, computing h_{root} therefore requires up to

$$128 \cdot 8 + 1 = 1025 \tag{10}$$

hash function evaluations. Table 1 summarizes the performance of our scheme, in terms of the size of the signature before and after compression, and computation required by each of the four algorithms.

To provide some context, Figure 6 shows the ratio between the size of an image signed with our signature and the size of an image that is not signed, and also the ratio between the size of an image signed with our signature and directly signing it with a standard digital signature (and hence having to provide the entire image, all the time). For this comparison we consider an image size of 2 megabytes, a hash function with 256 bit output and a standard digital signature scheme with signature size 512 bits. These parameters correspond to using our scheme with SHA3-256 and EdDSA using the Ed25519 curve, both of which are currently thought to be secure, and widely used [13, 9]. Choosing these parameters result in a signature size of $128 \cdot 256 + 512 = 33,280$ bits, or 32.5 kb, for our scheme. Figure 6 illustrates that, for these parameters, using our signature scheme adds just under 4% overhead when images are compressed down to 5% of their original size. Replacing EdDSA with a post-quantum signature scheme (and thus making our scheme post-quantum secure), increases the overhead of our scheme, but does not fundamentally change the figure. For example, using the post-quantum signature scheme Dilithium in the form suggested by NIST [27], changes the signature size to be between 2420 and 4595 bytes, rather than 512 bits. Even in the 4595 bytes case, the overhead is only just over 8%, when images are compressed down to 5% of their original size. In this case, our scheme has signature size $128 \cdot 256 + 36,760 = 69,528$ bits, or just under 8.5 kB. Finally, Figure 6 also illustrates that, rather obviously, our scheme performs drastically better than just signing the hash of the image directly, since in this case the entire image has to be provided to verify the signature, and thus no compression can be done.

Table 1: Upper bounds on the performance of our construction of a signature scheme allowing image compression, assuming it is constructed with a hash function with output size $|H|$ and a standard digital signature scheme DS **with signature size $|S|$.**

	Computation Time	Signature Size
Key generation	Same as $\mathsf{KeyGen}^{\mathsf{DS}}$	—
Signing	1025 hashes and time of $\mathsf{Sign}^{\mathsf{DS}}$	$\|S\|$
Compression	1025 hashes	$128\|H\| + \|S\|$
Verification	1025 hashes and time of $\mathsf{Verify}^{\mathsf{DS}}$	—

5 VISUAL EVALUATION

An essential question to ask about a construction that modifies how JPEG compression is performed is how it impacts the image quality. In order to evaluate this, we used the IJG implementation [35] with both the standard quantization tables and quantization tables that contained only powers of two (as in our construction and [18]). For any fixed image, we chose the quantization tables with powers of two to be the tables giving the size closest to the size obtained with standard quantization tables. For our construction, we found the tables in the following way. First, a function r rounding a value v to either the closest smaller or the closest larger power of two is defined. Rather than just rounding v to the closest value, the function takes an additional parameter $q \in [0, 1]$, which defines where the cutoff between rounding down and rounding up is. The function is defined as:

$$r(v, q) = \begin{cases} 2^{\lceil \log(v) \rceil} & \text{if } v > (1+q) \cdot 2^{\lfloor \log v \rfloor} \\ 2^{\lfloor \log v \rfloor} & \text{otherwise.} \end{cases} \tag{11}$$

This allows us to tweak q in order to get the size of the image compressed with the generated quantization table as close to the size of the image compressed with the standard quantization tables as possible. For a given quality factor p (and hence a pair of standard quantization tables), we perform a binary search to find the value of q that gives the closest compressed image size. As an example, using the image in Figure 7a and quality factor $p = 50$, we obtain

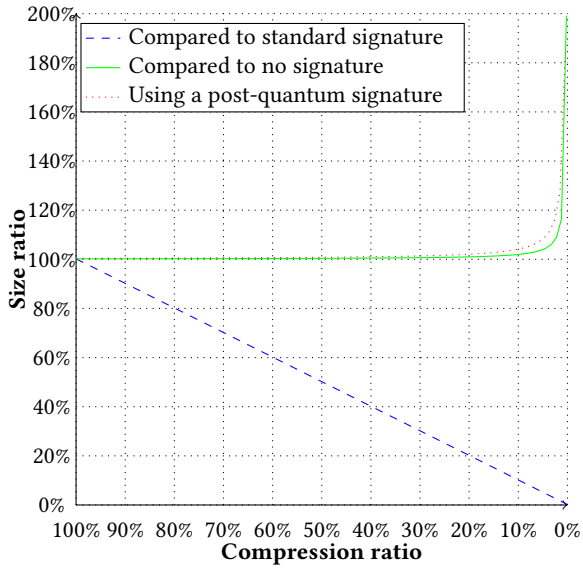


Figure 6: Relative size of an image and signature from our scheme (image size + provided nodes + standard signature) compared to not signing the image at all (image size) in dashed blue and compared to using a standard digital signature scheme (uncompressed image size + standard signature) in solid green. Additionally, relative size compared to not using a signature, when our scheme uses a post-quantum signature in dotted red. For this comparison, we consider an image with an uncompressed size of 2 megabytes, a hash function with output size 256 bits, a standard digital signature scheme with signature size 512 bits, and a post-quantum digital signature scheme with signature size 36760 bits.

the following luminance quantization table from the luminance quantization table in Equation (4):

$$Q = \begin{bmatrix} 16 & 8 & 8 & 16 & 16 & 32 & 64 & 64 \\ 8 & 8 & 16 & 16 & 32 & 64 & 64 & 64 \\ 16 & 16 & 16 & 16 & 32 & 64 & 64 & 64 \\ 16 & 16 & 16 & 32 & 64 & 64 & 64 & 64 \\ 16 & 16 & 32 & 64 & 64 & 128 & 128 & 64 \\ 16 & 32 & 64 & 64 & 64 & 128 & 128 & 64 \\ 64 & 64 & 64 & 64 & 128 & 128 & 128 & 128 \\ 64 & 64 & 64 & 128 & 128 & 128 & 128 & 128 \end{bmatrix}. \tag{12}$$

Having found the different quantization tables, we compress the image with both the standard quantization tables, with our modified quantization tables, and with quantization tables consisting of only one power of two, i.e., giving images that visually match the ones obtained by the approach suggested in [18]. For [18], we manually found the power of two resulting in a compressed image of size closest to the standard quantization tables.

Purely ocular evaluation by the author and colleagues could not reliably tell the compressed images apart, see Figure 7. In order to evaluate the similarity of the produced images, we instead use the following four similarity measures to compare compressions of all the images in [31] to the uncompressed reference images.

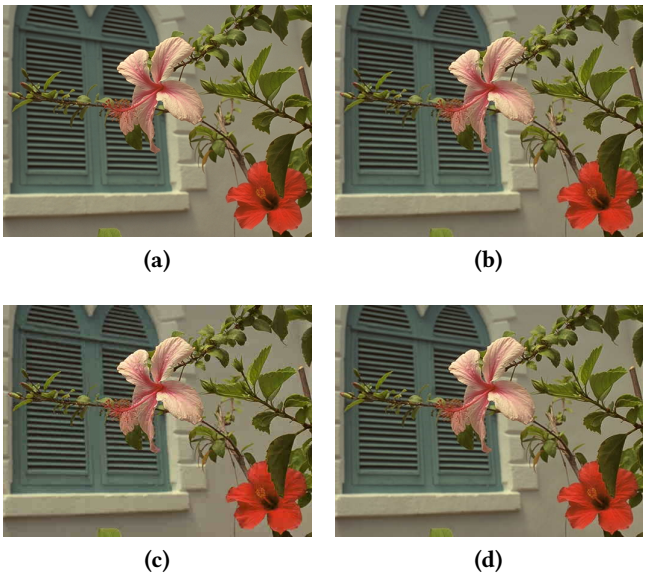


Figure 7: An image from the TID2008 database [31]. In 7a uncompressed, in 7b compressed using the default Q_{50} quantization tables, in 7c using the approach suggested in [18], and in 7d using our approach.

- **MultiScale Structural SIMilarity (MS-SSIM):** Image quality assessment measure intended to match the human perception by moving from a pixel comparison to a structure comparison [39].
- **Feature SIMilarity (FSIM):** Image quality assessment measure intended to match the human perception by using that humans understand pictures mainly by their low-level features. Newer and claimed (by its authors) to be closer to human perception than MS-SSIM [42]. FSIMc is the color variant of FSIM.
- **Mean Squared Error (MSE):** Classical distance measure not matching the human perception. Found by calculating the mean squared distance between pixels in the images being compared, i.e., when comparing images I_1 and I_2, both of size $M \times N$, we have

$$\mathrm{MSE}(I_1, I_2) = \frac{1}{M \cdot N} \sum_{\substack{1 \le m \le M \\ 1 \le n \le N}} (I_1(m,n) - I_2(m,n))^2 . \tag{13}$$

- **Peak Signal Noise Ratio (PSNR):** Classical distance measure not matching the human perception. Derived from the MSE, and taking into account the maximal dynamic range of the image. In the same setting as above, and with R being the maximal dynamic range of I_1 and I_2 (so 255 for 8-bit images), we have

$$\mathrm{PSNR}(I_1, I_2) = 10 \cdot \log_{10}\left(\frac{R^2}{\mathrm{MSE}(I_1, I_2)}\right). \tag{14}$$

The procedure described above was implemented in a Matlab script, with calls to the IJG compression implementation [35]. Functions for calculating MS-SSIM, MSE, and PSNR are provided by

Table 2: Average results for compressions of the test images from [31] compared to the uncompressed images. We started from 3 different quality factors for the unmodified compression (25, 50, and 80), and for each of these we derived the quantization table with only powers of two giving the closest size. For the approach suggested in [18], we include both the closest smaller and the closest larger variant (the value in the quantization tables is indicated in parentheses).

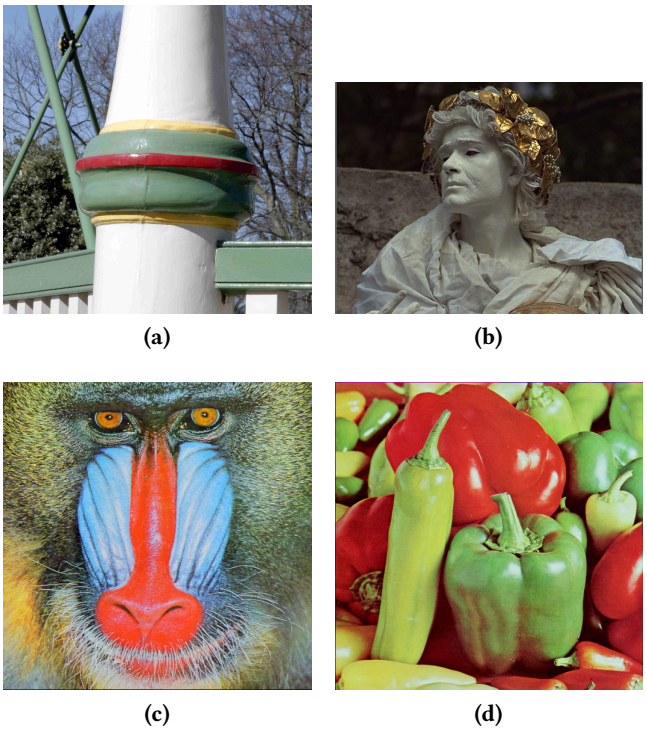
		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	16.0 kB	0.960	0.978	77.526	29.749
	Unmodified	15.1 kB	0.959	0.978	78.900	29.655
	[18] (64)	10.4 kB	0.914	0.936	109.016	28.021
	[18] (32)	20.5 kB	0.961	0.976	46.071	31.706
QF50	**Our tables**	25.4 kB	0.979	0.991	45.831	32.008
	Unmodified	24.4 kB	0.979	0.991	45.910	31.988
	[18] (32)	20.5 kB	0.961	0.976	46.071	31.706
	[18] (16)	36.5 kB	0.983	0.992	18.451	35.605
QF80	**Our tables**	43.9 kB	0.990	0.997	20.256	35.402
	Unmodified	43.4 kB	0.991	0.997	20.432	35.364
	[18] (16)	36.5 kB	0.983	0.992	18.451	35.605
	[18] (8)	60.4 kB	0.993	0.997	7.532	39.439

Matlab as `multissim`, `imse`, and `psnr`. A Matlab script for calculating FSIM was published as auxiliary material for the article [42, 41]. Our full code and additional material such as the reference images and compressed variants can be found on author's homepage[5].

In Table 2, we consider different quality factors ($p = 25$, $p = 50$, and $p = 80$, as described in Section 3.2), and compare the average of the images compressed with the standard quantization table with the compressed image obtain from our scheme, and with two images compressed with quantization tables with just one power of two in all entries, as suggest in [18]. We include two images compressed as suggested in [18], since this approach allows so little granularity in the quantization tables that it is often hard to get the compressed size even close to the images compressed with the standard quantization tables. This also demonstrates one of the major issues with the approach suggested in [18]; it only allows 8 different quantization tables, leading to very low granularity. Additionally, as can be seen in Table 2, even when we choose the quantization tables to be the ones resulting in a substantially larger file, the FSIMc score is comparable. Presumably, this is due to all entries in the quantization table having the same value. Thus, this approach does not use the human bias towards noticing low frequency changes.

Comparing the results of the unmodified compression and our compression, Table 2 shows that we can get very close to the the same size and image quality assessment scores (MS-SSIM and FSIMc). In fact, our compression obtain the same FSIMc score as the unmodified, and the MS-SSIM is 0.001 better in one case, and 0.001 worse in one case. This implies that the visual degradation of using our compression is comparable to visual degradation from using the default scheme. The MSE and PSNR values are also similar, but the differences are slightly larger.

[5] https://serfurth.dk/research/archive/



(a) **(b)** **(c)** **(d)**

Figure 8: Other standard test images we compared the unmodified JPEG compression, our compression, and the [18] compression on. Figure 8a is a test image from [5], Figure 8b is a test image from [31], and Figures 8c and 8d are test images from [40]. Test results can be found in Appendix A, but all show essentially the same results.

We also tested the Lenna test image and images from [5] and [40]. Results can be found in Appendix A, and examples of images in Figure 8. All tests showed essentially the same results.

6 FUTURE WORK

This paper extends the theory of digital signatures for images that still allow some form of compression. We present and analyze the first digital signature scheme that is first and foremost designed with the goal of allowing this. Additionally, we have suggested applications where this type of digital signatures could help in a small way towards solving some major societal issues. For these issues, additional work investigating if they would actually help users in a real setting would be a natural next step.

So far, our procedure for generating quantization tables for our compression, has been to fix an image compressed with standard quantization tables, and then finding the tables giving our compressed image size as close to first image as possible. This method is rather inefficient, because it requires many JPEG compressions, among other reasons. Instead of this inefficient method, we suggest defining a new set of standard quantization tables and a function for deriving quantization tables for different quality factors, similar to how it is currently done, but with the new standard and derived

tables consisting of only powers of two. These new standard quantization tables should be chosen to preserve the most image quality over a large set of different images and different quality factors, evaluated using both real-life tests with users and image quality assessment measures like the ones used in Section 5.

One optimization that could be made to our construction itself, is that chains of hashes that are always truncated by the same number of bits could be combined into one, so that only one hash needs to be provided in the signature for the compressed image. As an example, we refer back to the standard luminance quantization table in Equation (4), in which we see that the table has 14 in entries $(1, 2)$, $(2, 0)$, and $(3, 0)$, and hence the chains of hashes corresponding to these entries will always be truncated by the same amount in our construction. If new quantization tables containing only powers of two are standardized, this optimization would be particular effective, since these tables would necessarily have many identical values. Finally, an efficient prototype should be implemented, and the efficiency of the prototype evaluated and compared to, for example, state of the art of zero-knowledge based approaches [10].

A ADDITIONAL VISUAL TESTS

Tables 3 to 8 contains results comparing our compression, compression done with unmodified tables, and compression done with quantization tables giving the same result as [18]. We test the Lenna test image and images from [5], [31], and [40]. All test essentially show the same results as Table 2. For each image, we started from 3 different quality factors for the unmodified compression, and for each of these we derived the quantization table with only powers of two giving the closest size. For the approach suggested in [18], we include both the closest smaller and the closest larger variant (the value in the quantization tables is indicated in parentheses).

Table 3: Compressions of the Lenna test image compared to the uncompressed image. We started from quality factors 25, 50, and 80. While the Lenna test image is perhaps the most widely used test image, it is considered problematic by some [14], and hence we do not display it.

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	16.2 kB	0.945	0.981	56.287	30.627
	Unmodified	16.0 kB	0.946	0.981	54.794	30.743
	[18] (64)	10.0 kB	0.899	0.938	87.190	28.726
	[18] (32)	18.8 kB	0.947	0.975	46.244	31.480
QF50	**Our tables**	26.1 kB	0.967	0.992	38.510	32.275
	Unmodified	26.1 kB	0.967	0.992	37.919	32.342
	[18] (32)	18.7 kB	0.947	0.975	46.244	31.480
	[18] (16)	37.1 kB	0.971	0.991	26.696	33.866
QF80	**Our tables**	48.4 kB	0.980	0.997	25.708	34.030
	Unmodified	48.4 kB	0.980	0.997	25.467	34.071
	[18] (16)	37.1 kB	0.971	0.991	26.696	33.866
	[18] (8)	77.1 kB	0.985	0.997	16.528	35.949

Table 4: Compressions of the pillar test image [5] (Figure 8a) compared to the uncompressed image. We started from quality factors 10, 20, and 50.

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF10	**Our tables**	99.7 kB	0.938	0.987	30.192	33.332
	Unmodified	99.5 kB	0.929	0.987	37.320	32.411
	[18] (128)	68.3 kB	0.869	0.963	85.001	28.837
	[18] (64)	99.8 kB	0.938	0.991	30.168	33.335
QF20	**Our tables**	160.4 kB	0.973	0.997	12.069	37.314
	Unmodified	157.2 kB	0.966	0.997	14.471	36.526
	[18] (64)	99.8 kB	0.938	0.991	30.168	33.335
	[18] (32)	162.7 kB	0.973	0.998	11.798	37.413
QF50	**Our tables**	309.4 kB	0.991	0.999	4.179	41.920
	Unmodified	281.9 kB	0.989	0.999	4.628	41.477
	[18] (8)	435.1 kB	0.995	1.000	2.030	45.056
	[18] (16)	266.9 kB	0.988	0.999	4.779	41.338

Table 5: Compressions of just one test image from the TID2008 database [31] (Figure 7) compared to the uncompressed image. We started from quality factors 25, 50, and 80.

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	15.7 kB	0.976	0.983	46.963	31.413
	Unmodified	14.3 kB	0.974	0.983	50.496	31.098
	[18] (64)	9.3 kB	0.940	0.945	90.789	28.550
	[18] (32)	17.2 kB	0.975	0.979	35.967	32.572
QF50	**Our tables**	24.3 kB	0.989	0.993	27.010	33.816
	Unmodified	22.4 kB	0.987	0.993	28.489	33.584
	[18] (32)	17.2 kB	0.975	0.979	35.967	32.572
	[18] (16)	29.8 kB	0.990	0.993	14.219	36.602
QF80	**Our tables**	38.5 kB	0.994	0.998	12.963	37.004
	Unmodified	38.0 kB	0.994	0.998	13.124	36.950
	[18] (16)	29.7 kB	0.990	0.993	14.219	36.602
	[18] (8)	47.5 kB	0.996	0.998	5.916	40.410

Table 6: Compressions of just one test image from the TID2008 database [31] (Figure 8b) compared to the uncompressed image. We started from quality factors 25, 50, and 80.

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	13.8 kB	0.971	0.981	49.564	31.179
	Unmodified	12.6 kB	0.969	0.981	51.834	30.985
	[18] (64)	8.1 kB	0.924	0.935	82.781	28.951
	[18] (32)	15.2 kB	0.967	0.976	34.697	32.728
QF50	**Our tables**	20.2 kB	0.984	0.992	30.678	33.263
	Unmodified	20.1 kB	0.984	0.992	29.983	33.362
	[18] (32)	15.2 kB	0.967	0.976	34.697	32.728
	[18] (16)	27.3 kB	0.985	0.992	14.785	36.433
QF80	**Our tables**	36.2 kB	0.993	0.998	13.908	36.698
	Unmodified	35.8 kB	0.993	0.998	13.982	36.675
	[18] (16)	27.3 kB	0.985	0.992	14.785	36.433
	[18] (8)	47.7 kB	0.993	0.997	6.479	40.016

Table 7: Compressions of test image from [40] (Figure 8c) compared to the uncompressed image. We started from quality factors 25, 50, and 80.

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	32.6 kB	0.932	0.975	284.226	23.594
	Unmodified	32.5 kB	0.934	0.975	278.961	23.675
	[18] (64)	26.6 kB	0.897	0.938	302.342	23.326
	[18] (32)	57.9 kB	0.950	0.976	159.081	26.115
QF50	**Our tables**	54.3 kB	0.958	0.988	200.370	25.112
	Unmodified	54.2 kB	0.959	0.988	199.167	25.139
	[18] (64)	26.6 kB	0.897	0.938	302.342	23.326
	[18] (32)	57.9 kB	0.950	0.976	159.081	26.115
QF80	**Our tables**	99.0 kB	0.977	0.995	121.037	27.302
	Unmodified	98.8 kB	0.977	0.995	120.239	27.330
	[18] (32)	57.9 kB	0.950	0.976	159.081	26.115
	[18] (16)	110.6 kB	0.977	0.992	84.196	28.878

Table 8: Compressions of test image from [40] (Figure 8d) compared to the uncompressed image. We started from quality factors 25, 50, and 80.

		Size	MS-SSIM	FSIMc	MSE	PSNR
QF25	**Our tables**	17.8 kB	0.936	0.979	83.316	28.924
	Unmodified	17.7 kB	0.937	0.979	81.872	28.999
	[18] (64)	11.7 kB	0.893	0.938	108.072	27.794
	[18] (32)	22.2 kB	0.941	0.974	61.652	30.231
QF50	**Our tables**	29.1 kB	0.959	0.991	58.628	30.450
	Unmodified	29.0 kB	0.959	0.991	56.692	30.596
	[18] (32)	22.2 kB	0.941	0.974	61.652	30.231
	[18] (16)	48.5 kB	0.967	0.990	39.224	32.195
QF80	**Our tables**	54.6 kB	0.976	0.997	39.005	32.220
	Unmodified	54.2 kB	0.976	0.997	39.503	32.164
	[18] (16)	48.5 kB	0.967	0.990	39.224	32.195
	[18] (8)	98.1 kB	0.983	0.997	25.347	34.091

REFERENCES

[1] Adobe. 2023. JPEG2000 files. https://www.adobe.com/creativecloud/file-types/image/raster/jpeg-2000-file.html. (Dec. 6, 2023).

[2] Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, Abhi Shelat, and Brent Waters. 2015. Computing on authenticated data. *Journal of Cryptology*, 28, 2, 351–395. DOI: 10.1007/S00145-014-9182-0.

[3] Adam L. Alter and Daniel M. Oppenheimer. 2009. Uniting the tribes of fluency to form a metacognitive nation. *Personality and Social Psychology Review*, 13, 3, 219–235. DOI: 10.1177/1088868309341564.

[4] Edward L. Amoruso, Stephen P. Johnson, Raghu Nandan Avula, and Cliff C. Zou. 2022. A web infrastructure for certifying multimedia news content for fake news defense. In *IEEE Symposium on Computers and Communications - ISCC 2022*. IEEE, 1–7. DOI: 10.1109/ISCC55528.2022.9912787.

[5] Nicola Asuni and Andrea Giachetti. 2013. TESTIMAGES: A large data archive for display and algorithm testing. *Journal of Graphics Tools*, 17, 4, 113–125. DOI: 10.1080/2165347X.2015.1024298.

[6] Mihir Bellare and Gregory Neven. 2002. Transitive signatures based on factoring and RSA. In *Advances in Cryptology - ASIACRYPT 2002* (LNCS). Vol. 2501. Springer, 397–414. DOI: 10.1007/3-540-36178-2_25.

[7] Joan Boyar, Simon Erfurth, Kim S. Larsen, and Ruben Niederhagen. 2023. Quotable signatures for authenticating shared quotes. In *Conference on Cryptology and Information Security in Latin America - LATINCRYPT 2023* (LNCS). Vol. 14168. Springer, 273–292. DOI: 10.1007/978-3-031-44469-2_14.

[8] C2PA. 2023. Coalition for content provenance and authenticity (C2PA). https://c2pa.org/. (Dec. 1, 2023).

[9] Lily Chen, Dustin Moody, Andrew Regenscheid, and Angela Robinson. 2023. Digital Signature Standard (DSS). Federal Inf. Process. Stds. (NIST FIPS). National Institute of Standards and Technology, Gaithersburg, MD, USA. DOI: 10.6028/NIST.FIPS.186-5.

[10] Trisha Datta and Dan Boneh. 2023. Using ZK proofs to fight disinformation. https://medium.com/@boneh/using-zk-proofs-to-fight-disinformation-17e7d57fe52f. (Nov. 28, 2023).

[11] Whitfield Diffie and Martin E. Hellman. 1976. New directions in cryptography. *IEEE Transactions on Information Theory*, 22, 6, 644–654. DOI: 10.1109/TIT.1976.1055638.

[12] Ling Du, Anthony T. S. Ho, and Runmin Cong. 2020. Perceptual hashing for image authentication: A survey. *Signal Processing: Image Communication*, 81, 115713. DOI: 10.1016/J.IMAGE.2019.115713.

[13] Morris J. Dworkin. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Inf. Process. Stds. (NIST FIPS). National Institute of Standards and Technology, Gaithersburg, MD, USA. DOI: 10.6028/NIST.FIPS.202.

[14] Finch. 2024. Losing lena: removing one image to make millions of women feel welcome. https://www.prnewswire.com/news-releases/losing-lena-removing-one-image-to-make-millions-of-women-feel-welcome-300960513.html. (Mar. 25, 2024).

[15] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Communications of the ACM*, 63, 11, 139–144. DOI: 10.1145/3422622.

[16] International Telecommunication Union. 1992. T.81 – Digital Compression and Coding of Continuous-Tone Still Images – Requirements and Guidelines. https://www.w3.org/Graphics/JPEG/itu-t81.pdf. (1992).

[17] Magnus Stenaa Jensen. 2024. Detecting AI-manipulated content is a challenging arms race. https://www.dtu.dk/english/news/all-news/detecting-ai-manipulated-content-is-a-challenging-arms-race?id=f7be2a68-b6c4-4e39-bd21-6650009d287b. (Mar. 25, 2024).

[18] Rob Johnson, Leif Walsh, and Michael Lamb. 2011. Homomorphic signatures for digital photographs. In *Financial Cryptography and Data Security - FC 2011* (LNCS). Vol. 7035. Springer, 141–157. DOI: 10.1007/978-3-642-27576-0_12.

[19] Robert Johnson, David Molnar, Dawn Xiaodong Song, and David A. Wagner. 2002. Homomorphic signature schemes. In *Topics in Cryptology - CT-RSA 2002* (LNCS). Vol. 2271. Springer, 244–262. DOI: 10.1007/3-540-45760-7_17.

[20] Antonis Kalogeropoulos, Richard Fletcher, and Rasmus Kleis Nielsen. 2018. News brand attribution in distributed environments: do people know where they get their news? *New Media & Society*, 21, 3, 583–601. DOI: 10.1177/1461444818811880.

[21] Maxwell N. Krohn, Michael J. Freedman, and David Mazières. 2004. On-the-fly verification of rateless erasure codes for efficient content distribution. In *IEEE Symposium on Security and Privacy - S&P 2004*. IEEE Computer Society, 226–240. DOI: 10.1109/SECPRI.2004.1301326.

[22] Stephan Lewandowsky, Ullrich K. H. Ecker, Colleen M. Seifert, Norbert Schwarz, and John Cook. 2012. Misinformation and its correction: continued influence and successful debiasing. *Psychological Science in the Public Interest*, 13, 3, 106–131. DOI: 10.1177/1529100612451018.

[23] Ching-Yung Lin and Shih-Fu Chang. 2001. A robust image authentication method distinguishing JPEG compression from malicious manipulation. *IEEE*

Transactions on Circuits and Systems for Video Technology - (TCSVT), 11, 2, 153–168. DOI: 10.1109/76.905982.

[24] Santiago Lyon. 2023. Leica launches world's first camera with content credentials. https://contentauthenticity.org/blog/leica-launches-worlds-first-camera-with-content-credentials. (Nov. 28, 2023).

[25] David Marr and Ellen Hildreth. 1980. Theory of edge detection. *Proceedings of the Royal Society B*, 207, 1167, 187–217. DOI: 10.1098/rspb.1980.0020.

[26] Silvio Micali and Ronald L. Rivest. 2002. Transitive signature schemes. In *Topics in Cryptology - CT-RSA 2002* (LNCS). Vol. 2271. Springer, 236–243. DOI: 10.1007/3-540-45760-7_16.

[27] National Institute of Standards and Technology. 2023. Module-Lattice-Based Digital Signature Standard. Federal Inf. Process. Stds. (NIST FIPS). National Institute of Standards and Technology, Gaithersburg, MD, USA. DOI: 10.6028/NIST.FIPS.204.ipd.

[28] Assa Naveh and Eran Tromer. 2016. Photoproof: Cryptographic image authentication for any set of permissible transformations. In *IEEE Symposium on Security and Privacy - S&P 2016*. IEEE Computer Society, 255–271. DOI: 10.1109/SP.2016.23.

[29] Nic Newman, Richard Fletcher, Antonis Kalogeropoulos, and Rasmus Kleis Nielsen. 2019. Reuters Institute Digital News Report 2019. Tech. rep. Reuters Institute for the Study of Journalism. https://www.digitalnewsreport.org/survey/2019/.

[30] Katherine Ognyanova, David Lazer, Ronald E. Robertson, and Christo Wilson. 2020. Misinformation in action: fake news exposure is linked to lower trust in media, higher trust in government when your side is in power. *Misinformation Review*, 1, 4. DOI: 10.37016/mr-2020-024.

[31] Nikolay Ponomarenko, Vladimir Lukin, Alexander Zelensky, Karen Egiazarian, Marco Carli, and Federica Battisti. 2009. TID2008 — A database for evaluation of full-reference visual quality assessment metrics. *Advances of Modern Radioelectronics*, 10, 4, 30–45. https://www.ponomarenko.info/papers/mre2009tid.pdf.

[32] Leonie Schaewitz and Nicole C. Krämer. 2020. Combating disinformation: effects of timing and correction format on factual knowledge and personal beliefs. In *Multidisciplinary International Symposium on Disinformation in Open Online Media - MISDOOM 2020* (LNCS). Vol. 12259. Springer, 233–245. DOI: 10.1007/978-3-030-61841-4_16.

[33] Emily Sidnam-Mauch, Bernat Ivancsics, Ayana Monroe, Eve Washington, Errol Francis II, Kelly Caine, Joseph Bonneau, and Susan E. McGregor. 2022. Usable cryptographic provenance: A proactive complement to fact-checking for mitigating misinformation. In *AAAI Conference on Web and Social Media - ICWSM 2022*. DOI: 10.36190/2022.55.

[34] Ron Steinfeld, Laurence Bull, and Yuliang Zheng. 2001. Content extraction signatures. In *Information Security and Cryptology - ICISC 2001* (LNCS). Vol. 2288. Springer, 285–304. DOI: 10.1007/3-540-45861-1_22.

[35] The Independent JPEG Group (IJG). 2022. JPEG Standard Reference Implementation (version 9e). https://jpegclub.org/reference/reference-sources/. (2022).

[36] Milan Tuba and Nebojsa Bacanin. 2014. JPEG quantization tables selection by the firefly algorithm. In *Conference on Multimedia Computing and Systems - ICMCS 2014*. IEEE, 153–158. DOI: 10.1109/ICMCS.2014.6911315.

[37] Juliane Urban and Wolfgang Schweiger. 2014. News quality from the recipients' perspective. *Journalism Studies*, 15, 6, 821–840. DOI: 10.1080/1461670X.2013.856670.

[38] Gregory K. Wallace. 1991. The JPEG still picture compression standard. *Communications of the ACM*, 34, 4, 30–44. DOI: 10.1145/103085.103089.

[39] Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. 2003. Multiscale structural similarity for image quality assessment. In *IEEE Asilomar Conference on Signals, Systems and Computers*. Vol. 2. IEEE, 1398–1402. DOI: 10.1109/ACSSC.2003.1292216.

[40] Allan Weber. 2024. The USC-SIPI image database. https://sipi.usc.edu/database/database.php?. (Apr. 4, 2024).

[41] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. 2011. Feature SIMilarity index for IQA. https://web.comp.polyu.edu.hk/cslzhang/IQA/FSIM/FSIM.htm. (2011).

[42] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. 2011. FSIM: A feature similarity index for image quality assessment. *IEEE Trans. Image Process.*, 20, 8, 2378–2386. DOI: 10.1109/TIP.2011.2109730.

[43] Fang Zhao, Ton Kalker, Muriel Médard, and Keesook J. Han. 2007. Signatures for content distribution with network coding. In *IEEE Symposium on Information Theory - ISIT 2007*. IEEE, 556–560. DOI: 10.1109/ISIT.2007.4557283.

[44] Bin Benjamin Zhu, Mitchell D. Swanson, and Shipeng Li. 2004. Encryption and authentication for scalable multimedia: current state of the art and challenges. *Internet Multimedia Management Systems V*, 5601, 157–170. DOI: 10.1117/12.571869.